Università degli Studi di Padova

Dipartimento di Matematica 'Tullio Levi-Civita'

Corso di Laurea in Informatica

Un *frontend* per l'applicazione *TripHippie* per l'organizzazione di viaggi

Tesi di laurea triennale

Relatore

Prof. Tullio Vardanega

Laureando

Davide Romano

Anno Accademico 2022-2023

Look up at the stars and not down at your feet. Try to make sense of what you see, and wonder about what makes the universe exist. Be curious. .

— Stephen Hawking

Sommario

Il presente documento descrive dettagliatamente il tirocinio da me svolto presso l'azienda Sync Lab S.R.L. nella sede di Padova durante il periodo che va dal 04/09/2023 al 05/11/2023.

L'esperienza di stage presso l'azienda Sync Lab ha avuto una durata complessiva di 320 ore ed è stata coordinata e supervisionata dal mio tutor aziendale, l'ingegnere Fabio Pallaro, e dal mio relatore il professor Tullio Vardanega.

Lo scopo dello stage era realizzare un *frontend*, in particolare alcune maschere, per il progetto *TripHippie*, un progetto per l'organizzazione di viaggi, con le maschere che sarebbero servite per la comunicazione tra utenti, utilizzando i linguaggi *TypeScript* e Java e i *framework* Angular e Spring.

Il percorso di stage ha richiesto il ripasso di alcune metodologie di lavoro e, in merito ai linguaggi ed i *framework* da utilizzare, ha richiesto in parte un periodo di ripasso e in parte un periodo di studio.

Il seguente documento è stato strutturato in 4 capitoli:

1. **Capitolo 1:** presentazione del contesto organizzativo e produttivo nel quale sono stato inserito, tecnologie utilizzate, i processi interni utilizzati e propensione dell'azienda per l'innovazione;
2. **Capitolo 2:** presentazione della proposta di stage, per obiettivi e vincoli, la loro relazione con la strategia di sviluppo dell'organizzazione ospitante e motivazioni della scelta;
3. **Capitolo 3:** descrizione dettagliata del progetto di stage, con particolare approfondimento delle fasi del progetto, metodo di lavoro utilizzato, tecnologie trattate e soluzioni progettuali attuate al fine di raggiungere gli obiettivi attesi dall'azienda;
4. **Capitolo 4:** resoconto conclusivo, con valutazione retrospettiva, del percorso di stage, con approfondimento in particolare su difficoltà riscontrate, obiettivi raggiunti, codice e documenti prodotti e competenze professionali acquisite.

In merito alla redazione del testo, in relazione al documento ho adottato specifiche convenzioni tipografiche al fine di garantire chiarezza e comprensione. Le pratiche adottate sono le seguenti:

- Gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati sono chiaramente definiti nel glossario, collocato alla conclusione del presente testo;
- Per la prima occorrenza dei termini inclusi nel glossario, viene adottata la seguente nomenclatura: parola[g];
- Tutte le immagini presenti nel documento verranno corredate con la fonte d'origine nel caso fossero acquisite da fonti esterne o siti terzi.

"The greatest glory in living lies not in never falling, but in rising every time we fall"

— Nelson Mandela

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Tullio Vardanega, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto i miei genitori, Francesco e Luciana, per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Ho desiderio di ringraziare poi i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.

Padova, Dicembre 2023 Davide Romano

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Il contesto aziendale

1.1 L'azienda

1.1.1 Descrizione

SyncLab è un'azienda italiana nata nel 2002 a Napoli, che ha rapidamente conquistato una solida posizione nel mercato dell'Information and Communications Technology (ICT). Inizialmente una software house, l'azienda si è evoluta in un system integrator, sviluppando notevolmente le competenze tecnologiche e metodologiche. Questa crescita ha consentito a SyncLab di diversificare le sue attività, ampliando il suo campo d'azione per includere la *cybersecurity*, la videosorveglianza e il *mobile*, alimentando nel frattempo un'intensa attività di ricerca e sviluppo.

Figura 1.1: Logo e motto aziendale

Fonte: synclab.it

Attualmente, SyncLab conta oltre 300 dipendenti distribuiti in 6 sedi in tutta Italia: Roma, Napoli, Milano, Verona, Padova e Como. Queste sedi collaborano sinergicamente, impiegando numerosi specialisti per supportare una vasta gamma di clienti e

partner nell'implementazione di soluzioni aziendali.

Figura 1.2: Elenco delle sedi di SyncLab in Italia

Fonte: synclab.it

Tra i clienti di rilievo figurano Enel, *Sky*, *Vodafone*, Tim, *Fastweb*, Trenitalia, Posteitaliane, UniCredit e Intesa SanPaolo. L'obiettivo principale di SyncLab è assistere i clienti nella progettazione, implementazione e gestione delle soluzioni IT, non solo dal punto di vista tecnologico ma anche nell'ambito del cambiamento organizzativo.

1.1.2 Prodotti

SyncLab opera in diverse aree del campo dell'informatica, tra cui consulenza aziendale, consulenza informatica e finanziamento di progetti. Nel corso degli anni, l'azienda ha messo a punto numerosi prodotti, con un costante focus sull'assicurare elevati standard di qualità. A tale scopo, SyncLab ha conseguito diverse certificazioni di riconoscimento, tra cui ISO 9001, ISO 14001, ISO 27001 e ISO 45001.
Nel contesto della ricerca e dello sviluppo, l'azienda vanta una serie di prodotti che si distinguono per la loro innovazione e la qualità intrinseca.
Di seguito, sono elencati alcuni di questi prodotti di spicco:

- ***SynClinic:*** soluzione software integrata dedicata alla gestione delle strutture sanitarie, dotata di una vasta gamma di funzionalità tra cui la cartella clinica, la gestione delle fatturazioni e del magazzino. Questa applicazione è essenziale per il personale medico, in quanto consente di pianificare, coordinare e monitorare in modo completo tutte le fasi del percorso di cura dei pazienti, contribuendo a mitigare i rischi clinici. La realizzazione di SynClinic è frutto dell'implementazione di un'architettura basata su microservizi utilizzando il linguaggio di programmazione Java, con un *frontend* sviluppato in *TypeScript* e basato sul framework[g] Angular;

- **DPS 4.0**: un'applicazione *web* progettata per la gestione del Regolamento Generale sulla Protezione dei Dati (GDPR) in diverse aziende. Questa soluzione offre un approccio guidato per la revisione e la modifica dei documenti relativi alla *privacy*, garantendo la conformità agli standard di riferimento. La parte *backend* di DPS 4.0 è stata interamente sviluppata utilizzando il linguaggio di programmazione Java con il *framework* Spring, mentre per la gestione degli eventi è stato impiegato RabbitMQ come broker. Il frontend dell'applicazione è stato realizzato utilizzando il *framework* Angular;

- ***StreamLog:*** un'applicazione *software* progettata per la gestione della conformità al provvedimento emanato dal Garante per la protezione dei dati personali, con particolare riferimento agli Amministratori di Sistema. Questo strumento agevola l'adempimento dei requisiti stabiliti dal Garante, consentendo un controllo semplice ed efficiente degli accessi degli utenti. Dal punto di vista dello sviluppo, *StreamLog* si basa su un'architettura di tipo monolitico, in cui convergono sia la componente backend che quella frontend;

- ***StreamCrusher:*** soluzione tecnologica progettata per fornire assistenza nell'analisi delle informazioni legate alle decisioni aziendali, identificando rapidamente le aree critiche e consentendo la riorganizzazione dei processi in base alle nuove esigenze. Il motore centrale del prodotto è stato sviluppato esclusivamente utilizzando il linguaggio di programmazione Java SE;

- ***FidCity:*** soluzione *software* dedicata al *proximity marketing*, con un'integrazione di componenti social su entrambe le piattaforme *web* app e app *mobile*. L'applicazione *FidCity* è stata sviluppata come un'app nativa sia per la piattaforma Android che per iOS.
  WAVE, che sta per *Wide Area Videosurveillance Environment*, è un'applicazione impiegata nei sistemi di videosorveglianza per la completa integrazione con i Sistemi Informativi Territoriali (GIS). Questa integrazione consente un controllo completo dell'area soggetta a sorveglianza. *WAVE* opera come un *plugin*[g] installabile sulla piattaforma *Milestone System A/S*.

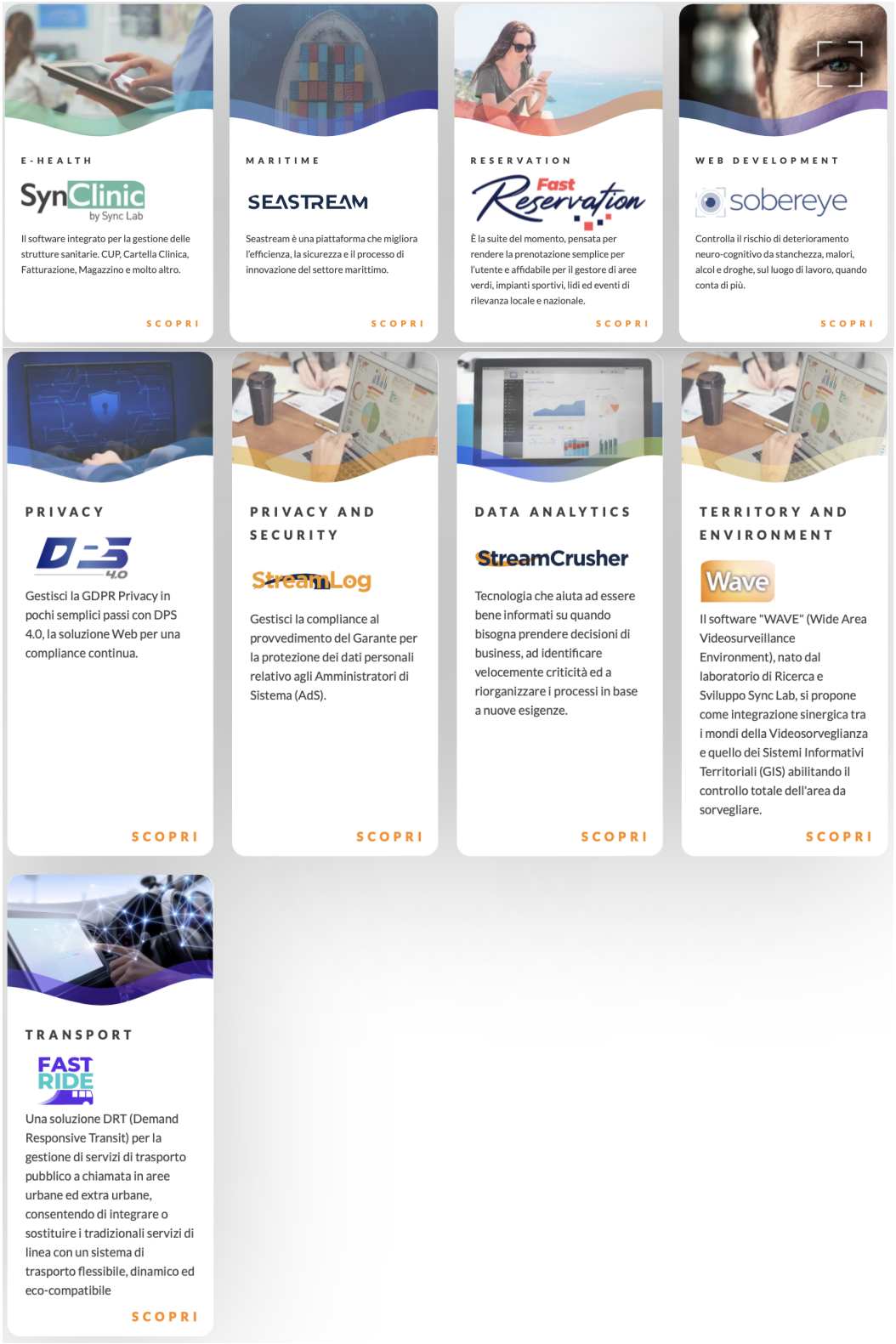

Figura 1.3: Prodotti principali di SyncLab

Fonte: synclab.it

1.2 I processi interni

1.2.1 Sviluppo

Visual Studio Code

Visual Studio Code, spesso abbreviato in *VS Code*, è un popolare ambiente di sviluppo integrato (Integrated Development Environment (IDE))[g] *open source* creato da *Microsoft*. Si tratta di un'applicazione leggera, altamente personalizzabile e altamente estendibile che è ampiamente utilizzata dagli sviluppatori software per scrivere, modificare e *debuggare* il codice.
Principali caratteristiche:

- **Editor di testo avanzato:** *VS Code* offre un potente editor di testo che supporta il rilevamento automatico del linguaggio, evidenziazione della sintassi, completamento automatico e molte altre funzionalità utili;
- **Estensioni:** È possibile personalizzare *VS Code* installando estensioni per supportare una vasta gamma di linguaggi di programmazione, *framework* e strumenti. Ciò consente di adattare l'ambiente di sviluppo alle proprie esigenze specifiche;
- ***Debugging* integrato:** *VS Code* integra strumenti di *debug* per diversi linguaggi di programmazione, semplificando notevolmente il processo di individuazione e correzione degli errori nel codice;
- **Controllo di versione:** L'IDE supporta il controllo di versione tramite strumenti come *Git*, consentendo agli sviluppatori di gestire il codice sorgente direttamente dall'interfaccia utente di VS Code;
- **Terminale integrato:** *Visual Studio Code* include un terminale integrato, che consente di eseguire comandi direttamente dall'IDE senza dover passare a un'altra applicazione;
- **Integrazione con servizi *cloud*:** È possibile integrare Visual Studio Code con servizi cloud come Azure per semplificare il deployment e il monitoraggio delle applicazioni cloud;
- **Multi-piattaforma:** *VS Code* è disponibile per Windows, macOS e Linux, rendendolo una scelta flessibile per sviluppatori su diverse piattaforme.

Grazie alla sua versatilità, alle numerose estensioni disponibili e al supporto di una vasta gamma di linguaggi di programmazione, Visual Studio Code è diventato uno degli ambienti di sviluppo preferiti da molti sviluppatori in tutto il mondo.

Intellij IDEA

IntelliJ IDEA è un ambiente di sviluppo integrato (IDE) altamente popolare e potente sviluppato da *JetBrains*. È progettato principalmente per sviluppatori di *software* e offre un'ampia gamma di funzionalità per semplificare il processo di sviluppo di applicazioni in diversi linguaggi di programmazione, in particolare per Java.
Principali caratteristiche:

- **Potente editor di codice:** *IntelliJ IDEA* offre un editor di codice altamente avanzato con funzionalità di evidenziazione della sintassi, completamento automatico intelligente, *refactoring*, analisi del codice e molte altre funzionalità utili per aumentare la produttività degli sviluppatori;
- **Integrazione del controllo di versione:** L'IDE supporta una vasta gamma di sistemi di controllo di versione, tra cui *Git*, SVN e altri, consentendo agli sviluppatori di gestire il codice sorgente in modo efficiente;
- ***Refactoring* automatizzato:** *IntelliJ IDEA* offre una serie di strumenti di *refactoring* automatizzati che semplificano la riorganizzazione del codice, migliorando la leggibilità e la manutenibilità del software;
- **Analisi del codice:** L'IDE esegue un'analisi statica avanzata del codice per individuare errori e suggerire miglioramenti. Questo aiuta gli sviluppatori a scrivere codice di alta qualità;
- **Supporto per una vasta gamma di linguaggi:** Benché inizialmente orientato verso Java, *IntelliJ IDEA* supporta anche molti altri linguaggi, tra cui Kotlin, Groovy, Scala, JavaScript, TypeScript, HTML, CSS e altri;
- **Strumenti per sviluppo web:** *IntelliJ IDEA* offre strumenti specifici per lo sviluppo *web*, tra cui il supporto per *framework* come Spring, Java EE e strutture di *frontend* come Angular, React e Vue.js;
- **Estensioni e *plugin*:** È possibile personalizzare *IntelliJ IDEA* installando estensioni e plugin da un *repository online*. Questo consente di adattare l'IDE alle proprie esigenze specifiche;
- **Multi-piattaforma:** *IntelliJ IDEA* è disponibile per Windows, macOS e Linux.

Grazie alle sue caratteristiche avanzate e alla sua vasta comunità di sviluppatori, *IntelliJ IDEA* è diventato uno degli IDE più apprezzati per la programmazione Java e molti altri linguaggi, ed è ampiamente utilizzato in tutto il settore dello sviluppo *software*.

Git

Git è un sistema di controllo delle versioni distribuito ampiamente utilizzato nella gestione del codice sorgente durante lo sviluppo software. È stato creato da Linus Torvalds nel 2005 ed è diventato uno degli strumenti più popolari per il controllo delle versioni e la collaborazione tra sviluppatori.
Principali caratteristiche:

- **Distribuito:** *Git* è un sistema di controllo delle versioni distribuito, il che significa che ogni sviluppatore ha una copia completa del *repository*[g]sul proprio sistema locale. Questo permette di lavorare offline e di collaborare in modo efficace;
- **Velocità:** *Git* è noto per la sua velocità nell'eseguire operazioni come il commit, il branching e il merging. Questa velocità è dovuta al fatto che la maggior parte delle operazioni avviene localmente, senza la necessità di una connessione a un server remoto;

- ***Branching* e *merging:*** *Git* semplifica la gestione dei branch, consentendo agli sviluppatori di creare facilmente nuovi branch per sviluppare funzionalità o correzioni di bug isolate. Successivamente, è possibile unire i branch in modo efficiente.
- **Storia completa del progetto:** *Git* tiene traccia di tutte le modifiche apportate al codice nel tempo, consentendo di esplorare la storia completa del progetto. È possibile visualizzare chi ha apportato le modifiche, quando sono state apportate e cosa è stato cambiato.
- **Collaborazione:** *Git* facilita la collaborazione tra sviluppatori. Più persone possono lavorare sullo stesso progetto contemporaneamente, gestendo le proprie copie del *repository* e combinando le modifiche in modo ordinato;
- **Sistema di rami (*branching*) flessibile:** *Git* offre un sistema di branch molto flessibile che consente di creare, combinare e rinominare branch in modo efficiente. Questo è utile per sviluppare nuove funzionalità senza influire sul codice principale;
- ***Repository* remoto:** *Git* supporta la creazione di *repository* remoti, che possono essere ospitati su server dedicati o servizi di hosting come *GitHub*, *GitLab* o *Bitbucket*. Questi *repository* consentono di condividere il codice e collaborare con altri sviluppatori.
- ***Open source* e ampiamente supportato:** *Git* è un *software open source* ed è supportato da una vasta comunità di sviluppatori. Ciò significa che esistono molte risorse, strumenti e servizi basati su *Git* disponibili per gli sviluppatori;

SyncLab come piattaforma di *hosting* per il controllo delle versioni utilizza *GitHub*, che consente di creare organizzazioni all'interno delle quali inserire i *repository* remoti. La figura 1.4 mostra la modalità di collaborazione tra i vari strumenti

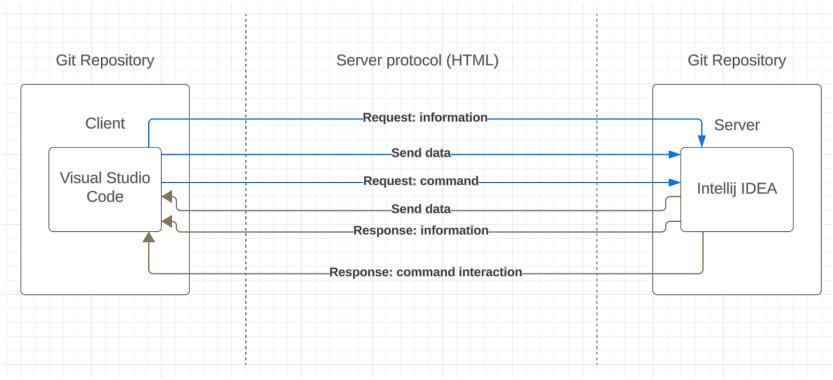

Figura 1.4: Collaborazione degli strumenti utilizzati

adottati per i processi di sviluppo, in quanto questi non operano in maniera indipendente gli uni dagli altri ma seguono processi di collaborazione ben specifici.

Trello

Trello è una popolare piattaforma di gestione progetti e attività che utilizza un formato di bacheca virtuale basato su schede. È stato sviluppato da *Fog Creek Software* nel 2011 e successivamente è stato acquisito da Atlassian.
Principali caratteristiche:

- **Bacheche e schede:** In Trello, i progetti sono organizzati in bacheche, ognuna delle quali rappresenta un progetto o un flusso di lavoro. All'interno di queste bacheche, le attività o le unità di lavoro sono visualizzate come schede;
- **Sistemi *Kanban*:** Trello si basa sul sistema *Kanban*, che consente agli utenti di spostare le schede attraverso colonne denominate come "Da fare", "In corso" e "Fatto". Questo fornisce una panoramica chiara dello stato di avanzamento del lavoro;
- **Personalizzazione:** Gli utenti possono personalizzare le bacheche, le colonne e le schede per adattarle alle proprie esigenze. È possibile aggiungere etichette, scadenze, allegati e commenti alle schede;
- **Collaborazione in tempo reale:** Trello supporta la collaborazione in tempo reale, consentendo agli utenti di lavorare insieme su progetti, assegnare attività, discutere dettagli e monitorare i progressi.
- **Integrazioni:** Trello offre numerose integrazioni con altre applicazioni e servizi, consentendo agli utenti di collegare facilmente Trello ad altre parti del loro stack di strumenti.
- **Versione *mobile* e *desktop*:** Trello è disponibile su diverse piattaforme, inclusi dispositivi mobili e *desktop*, consentendo agli utenti di accedere alle loro bacheche e schede ovunque si trovino.

Trello è ampiamente utilizzato in vari settori e contesti, tra cui la gestione di progetti, la pianificazione di attività personali, la collaborazione aziendale, lo sviluppo *software* e molto altro. È noto per la sua semplicità e flessibilità, il che lo rende una scelta popolare per le persone e le organizzazioni che cercano un modo visuale ed intuitivo per gestire il proprio lavoro e le proprie attività.

1.2.2 Manutenzione

Dopo aver lanciato un prodotto, SyncLab gestisce le attività di manutenzione per l'intera durata del *software*, adattando costantemente nuove funzionalità alle esigenze dei clienti e apportando correzioni quando necessario. In particolare, SyncLab fornisce tre categorie di servizi di manutenzione:

- **Manutenzione correttiva:** insieme di attività volte a risolvere e correggere i difetti o i problemi esistenti in un *software*. Questa tipologia di manutenzione è fondamentale per garantire che il *software* sia sempre stabile e affidabile per gli utenti, affrontando prontamente le problematiche che possono sorgere nel corso del tempo;
- **Manutenzione adattiva:** insieme di attività volte ad adattare un *software* alle nuove esigenze o cambiamenti dell'ambiente in cui opera. Questa tipologia di manutenzione può includere l'aggiunta di nuove funzionalità, la modifica di parti

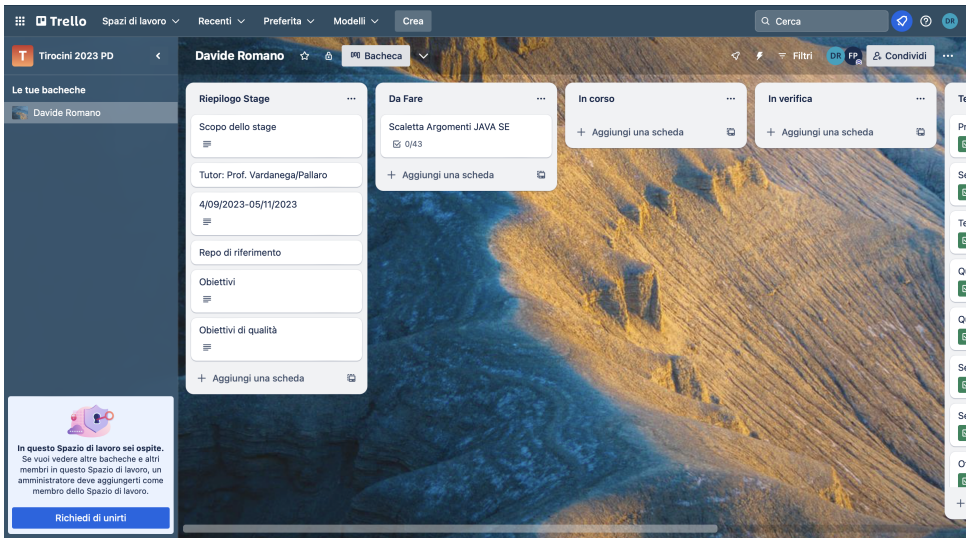

Figura 1.5: Esempio di bacheca Trello

Fonte: trello.com

del *software* per conformarsi a nuovi standard o l'ottimizzazione delle prestazioni per gestire aumenti di carico o nuove tecnologie;

- **Manutenzione evolutiva:** insieme di attività volte a sviluppare e migliorare un *software* esistente introducendo nuove funzionalità o aggiornamenti in risposta alle esigenze dell'utente o alle nuove opportunità di mercato. Questa tipologia di manutenzione è essenziale per mantenere il *software* competitivo e rilevante nel tempo, consentendo alle organizzazioni di sfruttare nuove opportunità e di adattarsi a un mercato in continua evoluzione.

1.2.3 Organizzazione del lavoro

In merito all'organizzazione del lavoro, l'approccio di SyncLab si basa sulla conoscenza dei processi che si fondano sul modello di sviluppo agile[g].
La metodologia agile è un approccio di sviluppo dei progetti che si concentra sulla collaborazione, l'adattamento continuo e la consegna rapida di risultati di alta qualità. A differenza delle metodologie tradizionali di sviluppo dei progetti, come il modello a cascata, che si basano su piani e documentazione estesa, le metodologie agile sono flessibili e si adattano man mano che il progetto avanza.
Ecco alcuni concetti chiave della metodologia agile:

- **Iterazione**: I progetti agile vengono suddivisi in cicli di sviluppo chiamati "iterazioni" o "sprint", di solito con una durata fissa, come 2 o 4 settimane. Alla fine di ogni iterazione, viene consegnato un incremento del prodotto che aggiunge valore;

- **Collaborazione:** Si promuove la comunicazione costante e la collaborazione tra i membri del team di sviluppo e gli interessati (*stakeholders*[g]). Team multidisciplinari lavorano insieme per prendere decisioni e risolvere problemi;

- **Prioritizzazione:** Il cliente o il proprietario del prodotto (product owner) definisce le priorità e lo scope del progetto. Le caratteristiche più importanti vengono sviluppate per prime, consentendo di ottenere un prodotto minimo funzionante (MVP) in tempi brevi;
- **Consegna continua:** L'obiettivo è consegnare parti del prodotto funzionante regolarmente, anziché attendere che l'intero progetto sia completo. Questo permette di ricevere feedback precoce e apportare modifiche secondo necessità;
- **Flessibilità:** Le metodologie agile sono adattabili e possono cambiare in base alle necessità del progetto e al feedback ricevuto dai clienti o dagli utenti.

Le metodologie agile includono diverse pratiche specifiche come scrum[g], *Kanban*, e *Extreme Programming (XP)*, che offrono linee guida dettagliate per l'implementazione di questi principi. L'obiettivo principale delle metodologie agile è migliorare la velocità, la flessibilità e la qualità dello sviluppo del software e di altri progetti.

Il *framework Scrum*

SyncLab ha scelto di adottare il *framework scrum* per la gestione di tutti i suoi progetti. *scrum* è un *framework* di sviluppo agile ampiamente utilizzato per la gestione dei progetti, specialmente nel campo dello sviluppo *software*. È progettato per essere leggero, flessibile e incentrato sulla consegna di valore in modo iterativo, seguendo delle regole di collaborazione, come illustrato nella figura 1.6.
Ruoli *scrum*:

- ***Scrum Master:*** Il ruolo del *Scrum Master* è quello di facilitare il processo *scrum*, rimuovere gli ostacoli che impediscono al team di lavoro di progredire e garantire che le regole di *scrum* vengano rispettate;
- ***Product Owner:*** Il *Product Owner* è responsabile di definire le priorità del lavoro e di assicurarsi che il *Team* di Sviluppo stia lavorando alle funzionalità più importanti per il cliente;
- ***Team* di Sviluppo:** Il *Team* di Sviluppo è composto dai professionisti che effettivamente realizzano il lavoro. Sono responsabili di pianificare, progettare, sviluppare, testare e consegnare il lavoro durante le sprint.

Eventi *scrum*:

- ***Sprint:*** È un periodo di tempo fisso (solitamente da 2 a 4 settimane) durante il quale il Team di Sviluppo lavora per consegnare un incremento di prodotto funzionante;
- ***Sprint Planning:*** Questo è un incontro in cui il team pianifica quali elementi del Product Backlog verranno inclusi nella prossima sprint;
- ***Daily Scrum:*** Un breve incontro quotidiano in cui il Team di Sviluppo condivide l'avanzamento, identifica eventuali impedimenti e aggiorna il piano per il giorno;
- ***Sprint Review:*** Alla fine di ogni sprint, si tiene una revisione in cui il team presenta il lavoro completato e riceve il feedback dai clienti o dagli *stakeholders*;
- ***Sprint Retrospective:*** Dopo la Sprint Review, il team riflette sul proprio lavoro durante la sprint e identifica miglioramenti da apportare al processo.

Artefatti *Scrum*:

- ***Product Backlog:*** È una lista prioritizzata di tutte le funzionalità, le modifiche e i requisiti che devono essere sviluppati nel progetto;
- ***Sprint Backlog:*** Una selezione delle attività dal *Product Backlog* che il *Team* di Sviluppo si impegna a completare durante la sprint;
- **Incremento**: È il risultato del lavoro completato dal *Team* di Sviluppo alla fine di una *sprint*. Dovrebbe essere una versione potenzialmente rilasciabile del prodotto.

scrum promuove la trasparenza, l'ispezione e l'adattamento costante, consentendo ai team di rispondere in modo rapido e flessibile ai cambiamenti nei requisiti o nelle circostanze. È ampiamente utilizzato in molte industrie oltre allo sviluppo *software*, inclusi progetti di ricerca, sviluppo di prodotti, gestione dei servizi IT e altro ancora.

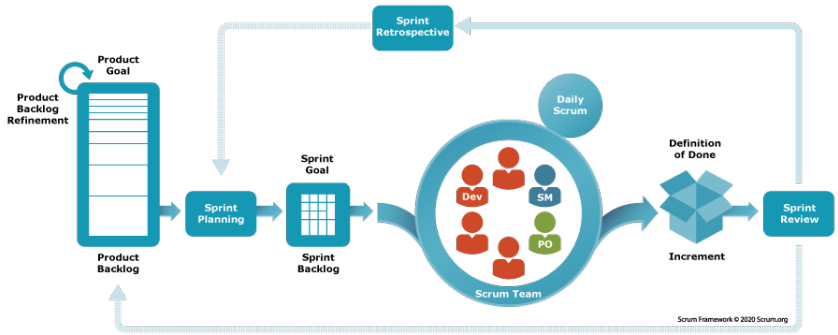

Figura 1.6: Metodo *Scrum*

Fonte: scrum.org

Modalità di applicazione del *framework Scrum*

Riflettendo sulla mia esperienza di stage, ho notato che le attività del progetto, suddivise secondo la classificazione riportata in figura 1.7, denominate *sprint*, sono caratterizzate da un approccio flessibile e adattabile alle necessità del momento. Ad ogni sprint, ci impegniamo a introdurre nuove funzionalità e garantire la soddisfazione del cliente attraverso un'attenta verifica. La nostra metodologia di lavoro si sviluppa nei seguenti passaggi:

- Iniziamo definendo un elenco delle attività da completare, noto come *product backlog*, che manteniamo su una tavola virtuale con tutte le attività e i requisiti del progetto;
- Successivamente, pianifichiamo preventivamente gli *sprint* che abbiamo intenzione di portare a termine, attraverso un processo di *sprint planning*;
- Dopo questa fase, identifichiamo uno *sprint backlog*, che rappresenta un insieme di obiettivi da raggiungere durante uno specifico *sprint*, basato sul nostro *product backlog*;

- Procediamo quindi con l'esecuzione dello *sprint*, che ha una durata definita, solitamente tra 1 e 2 settimane nel mio caso, ma comunque non superiore ai 30 giorni in generale.
- Alla conclusione di uno *sprint*, valutiamo il raggiungimento del suo obiettivo e determiniamo l'incremento effettivo ottenuto.

Durante il processo di sviluppo, l'azienda organizza regolarmente incontri con gli *stakeholders* e i membri del progetto per monitorare costantemente l'avanzamento degli *sprint*. In particolare, ho avuto l'opportunità di partecipare a due tipi di riunioni in linea con il metodo *scrum*:

- **Riunione quotidiana di squadra:** Questo incontro breve si svolge giornalmente con i membri del team ed è finalizzato a esaminare l'andamento attuale dello sprint, rispondendo a tre domande fondamentali:
  - Cosa è stato completato fino a oggi?
  - Cosa verrà affrontato domani?
  - Quali ostacoli stanno influenzando il mio progresso?
- **Revisione dello *sprint*:** Questo incontro si tiene al termine di ciascuno *sprint* ed è dedicato all'ispezione dei risultati ottenuti durante l'incremento, coinvolgendo gli *stakeholders* nella discussione di eventuali modifiche da apportare al *product backlog*.

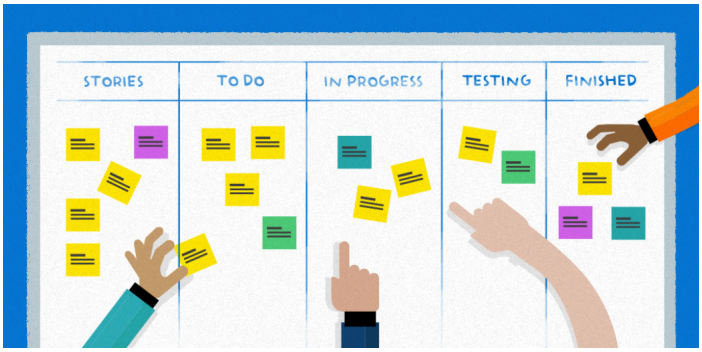

Figura 1.7: Organizzazione delle attività secondo il metodo *Scrum*

Fonte: govwebworks.com

SyncLab mi ha concesso il privilegio di partecipare alle sessioni di revisione degli *sprint* prima ancora dell'inizio del mio percorso di stage, concentrandoci specificamente sul progetto *TripHippie*, al quale ho dedicato il mio impegno durante il tirocinio. Questa opportunità mi ha permesso di acquisire una comprensione profonda e pratica di questo modello operativo, e in particolare delle revisioni in campo.

Queste riunioni, oltre a fornire una verifica dello stato di avanzamento dei progetti, si sono rivelate strumenti preziosi per rilevare eventuali disfunzioni o carenze nelle funzionalità implementate. Questo approccio ci ha consentito di riorientare i successivi *sprint*, al fine di evitare il ripetersi degli stessi errori.

Per il monitoraggio e la gestione delle attività, abbiamo utilizzato la *Scrum board* su Trello, una piattaforma *online*. Questo strumento ci ha consentito di tenere traccia dello stato di avanzamento di ciascuno *sprint* in modo efficace e organizzato.

La comunicazione con gli altri membri del *team* di sviluppo è stata altamente collaborativa, sia in presenza, poiché condividevamo gli stessi giorni in sede durante la settimana, sia attraverso l'uso di servizi di messaggistica istantanea, tra cui Discord, che ha agevolato la comunicazione e la condivisione delle informazioni in modo tempestivo.

1.3 Tecnologie utilizzate

SyncLab offre una vasta gamma di servizi nel panorama dell'*Information Technology* (IT), abbracciando diverse aree e sfruttando un ampio spettro di tecnologie per l'implementazione di soluzioni *software* robuste e all'avanguardia.
Alla base del successo e dell'efficienza di SyncLab nell'ambito dell'ingegneria del *software*, troviamo una serie di decisioni prese in campo tecnologico per la gestione efficace dei progetti, il controllo di versione, l'automazione dei flussi di lavoro e l'ottimizzazione generale dei processi operativi, utilizzando congiuntamente alcune tecnologie (figura 1.8). Queste tecnologie, inoltre, rappresentano le fondamenta dei processi interni menzionati nella sezione precedente.
Alcuni linguaggi scelti da SyncLab in base alle soluzioni richieste sono i seguenti:

- ***Typescript:*** *Typescript* è un linguaggio di programmazione *open source* che si basa sullo standard *ECMAScript* 6 e offre una serie di caratteristiche aggiuntive che risultano particolarmente utili nello sviluppo di applicazioni *web* e, in particolare, nei *framework* come *Angular*. Questo linguaggio estende la sintassi di JavaScript, il che significa che qualsiasi programma scritto in JavaScript può essere utilizzato senza problemi anche in *Typescript*.
  Tra le principali funzionalità aggiuntive di *Typescript*, spicca la possibilità di dichiarare e gestire i tipi delle variabili, il che porta a una maggiore sicurezza e chiarezza nel codice, contribuendo così a ridurre gli errori di *runtime*. Inoltre, *Typescript* permette di definire interfacce e classi, agevolando la creazione di una struttura ben organizzata per il codice;

- **Java:** Java è un linguaggio di programmazione orientato agli oggetti di notevole rinomanza ed è una componente di primaria importanza nelle applicazioni sviluppate da SyncLab. Questo linguaggio è particolarmente preminente all'interno dell'ambito dell'ingegneria del *software*, soprattutto quando associato all'integrazione del *framework* Spring. Tale integrazione riveste un'importanza significativa, in quanto Java, con il supporto di Spring, costituisce il pilastro fondamentale per la creazione di servizi REST nell'ambito dell'ingegneria del software orientato al *web*.
  Le ragioni che giustificano questa scelta sono numerose e cruciali. Java è ampiamente riconosciuto per la sua affidabilità, sicurezza e portabilità, rendendolo il linguaggio ideale per lo sviluppo di applicazioni *web* robuste;

- **JavaScript:** JavaScript è un linguaggio ampiamente diffuso nel contesto delle applicazioni *web* lato client e rappresenta un pilastro cruciale per l'implementazione di elementi interattivi all'interno dei siti *web*. La sua capacità di gestire eventi consente di conferire dinamicità e coinvolgimento all'esperienza utente,

rendendolo uno strumento imprescindibile per l'arricchimento delle funzionalità e dell'interfaccia dei siti *web*;

- **HTLM5 e CSS:** HTML5 e CSS sono due linguaggi di *markup* e di stile ampiamente utilizzati nel vasto mondo dello sviluppo *web*.
  HTML5, il linguaggio di *markup*, rappresenta lo scheletro di un sito *web*, definendo la struttura e l'organizzazione del contenuto. Questa versione avanzata di HTML offre elementi semantici che migliorano la comprensione del contenuto da parte dei motori di ricerca e dei *browser*, oltre a semplificare la creazione di siti *web* ben strutturati.
  Il CSS, il linguaggio di stile, è essenziale per definire la presentazione visiva di un sito web. Consentendo di applicare regole di stile ai componenti HTML, CSS garantisce un design coerente e attraente. La separazione tra HTML e CSS consente di apportare modifiche di stile senza influire sulla struttura del contenuto, facilitando così la manutenzione e l'evoluzione dei siti *web* nel tempo.

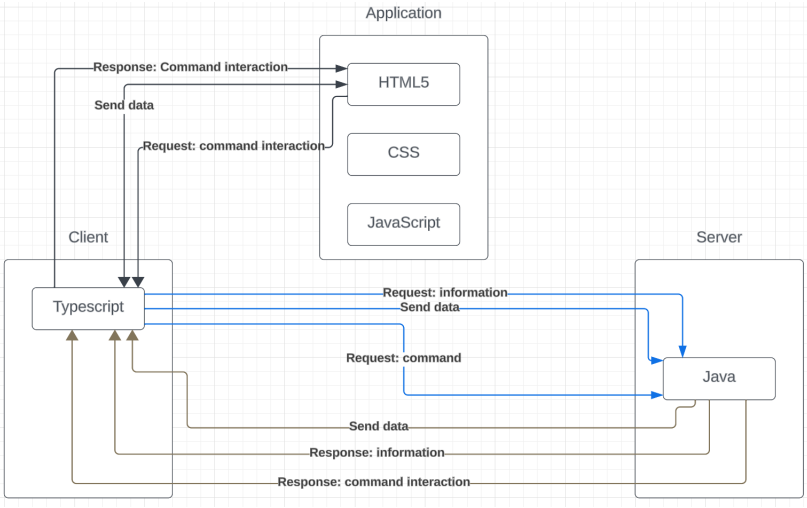

Figura 1.8: Interazione delle tecnologie nelle applicazioni

Framework utilizzati

L'utilizzo di un *framework* semplifica il processo di sviluppo, migliora la qualità del codice, accelera il *time-to-market* e riduce i rischi. Per questi motivi SyncLab ha scelto di adottare alcuni *framework* integrati ai linguaggi sopra menzionati:

- **Angular e AngularJS:** Angular e AngularJS sono due framework rilevanti nel dominio dello sviluppo *web*, particolarmente orientati alla creazione di *Single Page Application* (SPA) altamente performanti e *reactive*, che interagiscono in sinergia con un *backend* attraverso servizi REST. Angular fa affidamento sul motore JavaScript, consentendo di fornire un'esperienza di navigazione sul sito *web* che si avvicina notevolmente a quella di un'applicazione *desktop*.
  Nelle iterazioni più recenti, Angular ha introdotto nuove funzionalità incentrate sulla gestione delle dipendenze e sui *test*, il che semplifica ulteriormente l'apprendimento e la manutenzione. L'adozione del pattern Model-View-ViewModel (MVVM)[g]in Angular, oltre al precedente Model-View-Controller (MVC)[g]utilizzato

in AngularJS, illustrato in figura 1.9, costituisce un aspetto distintivo.
Questi pattern consentono di estendere agevolmente le funzionalità dell'applicazione a livello di codice, promuovendo le migliori pratiche nell'uso dei design-pattern[g] e contribuendo a garantire la scalabilità e la manutenibilità del progetto.
Per facilitare lo sviluppo della grafica delle pagine *web*, è stata creata la libreria Angular *Material*, basata sullo stile grafico di *Google*. Questa libreria mette a disposizione componenti grafiche già fatte e testate anche dal punto di vista dell'accessibilità;

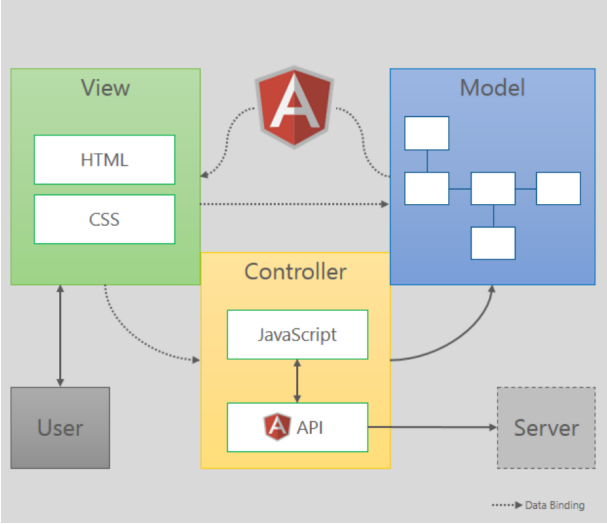

Figura 1.9: Design pattern architetturale Angular

Fonte: software-architects.com

- **Spring:** Spring costituisce un framework solidamente radicato in Java, ampiamente sfruttato per sviluppare applicazioni *server-side*, tra cui servizi REST, applicazioni *desktop* e *web*. Le caratteristiche di Spring sono state progettate per favorire l'adozione di pattern di progettazione, con un'enfasi particolare sulla creazione di un'architettura durevole nel tempo e pronta a essere estesa in prospettiva futura.
  La modularità offerta da Spring consente di separare e organizzare, come mostrato in figura 1.10, l'applicazione in componenti ben definite, rendendo più gestibile lo sviluppo e la manutenzione del software.

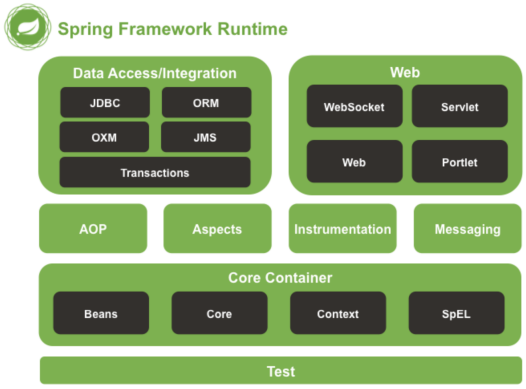

Figura 1.10: Moduli del framework Spring

Fonte: docs.spring.io

Comunicazione

Durante il periodo della pandemia, SyncLab ha implementato una serie di soluzioni organizzative volte a potenziare l'interazione e la comunicazione tra i gruppi di lavoro. L'adozione dello smart working ha spinto l'azienda a esplorare nuove strategie adattabili alle esigenze dei dipendenti e dei team di progetto, con l'obiettivo di ottimizzare i processi aziendali.
Pertanto, anche durante il mio stage, l'azienda ha sfruttato i seguenti strumenti:

- ***Google Meet:*** *Google Meet* è un'applicativo *software* che agevola la comunicazione con vari membri del *team* aziendale e facilita l'organizzazione di riunioni virtuali con una considerevole partecipazione. Inoltre, permette di sincronizzare in modo agevole il proprio programma di appuntamenti con le altre applicazioni incluse nella *suite* di *Google*, come ad esempio *Google Calendar*;
- ***Discord:*** *Discord* è una piattaforma di comunicazione gratuita che offre *chat* in tempo reale, con la possibilità di gestire numerosi canali vocali e testuali suddivisi in base agli argomenti di interesse. Questo strumento è accessibile su diverse piattaforme, inclusi sistemi operativi desktop e mobili, ed è da notare che, contrariamente ad altre soluzioni come *Slack*, le sue funzionalità principali sono completamente gratuite.
  Tra le varie funzioni fornite da *Discord*, c'è la possibilità di organizzare i membri in gruppi specifici, consentendo una migliore identificazione in relazione ai rispettivi progetti e attribuendo i relativi permessi. Tra le motivazioni che hanno portato SyncLab a scegliere *Google Meet* per le riunioni virtuali si aggiunge che la parte dedicata alle videoconferenze in *Discord* è meno avanzata e presenta alcune limitazioni, tra cui l'impossibilità di pianificare le riunioni in anticipo.
- **Trello:** si rimanda alla sezione 1.2.1

1.4 Propensione all'innovazione

All'interno dell'azienda SyncLab è presente un dipartimento chiamato "il laboratorio delle **possibilità**".

Questo laboratorio, unito alla ferma volontà dell'azienda di dare vita ad un nucleo solido di professionisti di alto profilo che si occupassero prevalentemente di tematiche di ricerca avanzata, ha consentito a SyncLab di avviare negli anni numerose iniziative e collaborazioni sia sul piano industriale che tecnologico. L'azienda vanta collaborazioni con le maggiori università italiane tra le quali:

Figura 1.11: Elenco delle università che collaborano con SyncLab

Fonte: synclab.it

SyncLab inoltre è impegnata in vari progetti di ricerca e sviluppo nel campo dell'*Information and Communications Technology* (ICT). Questi progetti spaziano in diverse aree e mirano a innovare, migliorare e sviluppare soluzioni tecnologiche avanzate.
I principali progetti sono:

Figura 1.12: I principali progetti di ricerca e sviluppo

Fonte: synclab.it

SyncLab ha recentemente ampliato il suo ambito di interesse, abbracciando nuovi settori come la *Cybersecurity*, la *Blockchain* e il *Big Data*. Questo ha portato all'avvio di progetti di ricerca dedicati in varie sedi, mirati all'esplorazione e all'acquisizione di conoscenze sulle nuove tecnologie.

L'azienda promuove attivamente la condivisione di idee e innovazioni tra i colleghi, sfruttando piattaforme come *Discord* per facilitare la comunicazione. In questo contesto, i membri del *team* condividono regolarmente messaggi nei canali testuali per discutere soluzioni migliori e innovative, creando un ambiente di discussione stimolante. Personalmente, ho avuto l'opportunità di partecipare a queste discussioni, che spesso mi hanno arricchito dal punto di vista delle conoscenze.

Inoltre, ho avuto modo di partecipare a discussioni e collaborazioni in loco con altri esperti del settore, il che ha contribuito in modo significativo al mio apprendimento e al mio miglioramento durante lo sviluppo del mio progetto di stage. Questo ambiente di lavoro favorisce una costante spinta all'innovazione, che si riflette direttamente nei progetti, compreso il mio lavoro su *TripHippie*. Infatti, la ricerca e l'adozione di nuove tecnologie sono state un aspetto fondamentale nello sviluppo dell'applicazione, in linea con la filosofia aziendale di cercare costantemente nuovi approcci e soluzioni al di fuori delle convenzioni tradizionali.

Capitolo 2

Introduzione allo stage

2.1 Il punto di vista dell'azienda

Lo stage rappresenta un momento cruciale nel percorso formativo di uno studente universitario, poiché offre l'opportunità di arricchire significativamente il proprio bagaglio professionale. Le aziende che propongono tirocini spesso stabiliscono collaborazioni con le università, e ciò comporta diversi vantaggi:

- **Favorire il contatto tra studenti e aziende**: Da un lato, le università facilitano l'incontro degli studenti con aziende che offrono nuove prospettive occupazionali, preparandoli all'ingresso nel mondo del lavoro;
- **Collaborazione con studenti universitari su progetti**: Dall'altro lato, le aziende collaborano con gli studenti su progetti, inclusi quelli di ricerca e sviluppo, offrendo opportunità di lavoro attraverso selezioni mirate;

Adottando il punto di vista di SyncLab, l'azienda mira a promuovere attività di ricerca e sviluppo, esplorando nuove tecnologie e sperimentando soluzioni innovative integrabili nei requisiti attuali del mercato IT. I motivi principali che hanno spinto l'azienda ad offrire programmi di stage includono:

- **Esplorare nuove tematiche**: L'azienda si propone di affrontare tematiche di interesse per gli studenti, incoraggiando la loro approfondita esplorazione. Soggetti correlati agli sviluppi contemporanei risultano particolarmente interessanti,contribuendo all'innovazione del mercato;
- **Integrare nuove tecnologie**: SyncLab presenta una serie di tecnologie utilizzabili nei progetti, ma incoraggia gli stagisti a condividere idee e opinioni, apportando una visione fresca e innovativa. Questa collaborazione permette sia all'azienda che agli studenti di apprendere e confrontarsi con nuove tecnologie;
- **Possibilità di assunzione**: L'azienda riconosce l'importanza di offrire opportunità di lavoro agli studenti stagisti, permettendo la selezione e l'inserimento di giovani talenti nel mondo del lavoro. Gli stage fungono da anteprima del modo di lavorare dell'azienda;
- **Basso impatto economico**: Gli stage sono spesso finanziati dall'università, riducendo l'impatto economico sulle aziende. L'investimento limitato consente a SyncLab di collaborare su diversi progetti con un ampio numero di tirocinanti,

in *partnership* con l'Università di Padova, anche in periodi difficili che possono richiedere compromessi.

In sintesi, gli stage offrono numerosi vantaggi e SyncLab ha deciso di dedicare tempo ed energie per coinvolgere stagisti in diversi progetti, stabilendo collaborazioni significative con l'Università di Padova, anche in periodi di sfide che richiedono flessibilità.

2.2 Il progetto *TripHippie*

Il progetto *TripHippie* ha origine dalla necessità di consentire agli utenti di organizzare e partecipare a viaggi in compagnia, compresi aspetti relativi a pasti durante il viaggio e l'eventuale consumo di alcol.

Fin dalle prime interazioni con il tutor aziendale, l'obiettivo del mio percorso di stage era l'implementazione di una funzionalità di chat all'interno del progetto *TripHippie*. Questa funzionalità permette a tutti gli utenti registrati sul sito di inviare e ricevere messaggi in tempo reale.

Il progetto non ha richiesto di essere progettato e sviluppato da zero, ma disponeva già di alcune funzionalità:

- ***Sign-up***: funzionalità per utenti che utilizzavano per la prima volta l'applicazione;
- ***Login***: possibilità per utenti già registrati di accedere all'applicazione con il proprio *account*;
- ***Logout***: uscita dall'applicazione (in alternativa è possibile direttamente chiudere il *browser*);
- L'interfaccia principale (*home*) fornirà un accesso intuitivo alla propria area personale, consentendo agli utenti di esplorare facilmente diverse sezioni. Tra le opzioni disponibili, si avrà l'opportunità di:
  - **Area Personale**: Questa sezione rappresenta uno spazio dedicato a ciascun utente, dove potranno gestire le proprie informazioni personali, impostazioni dell'*account* e ottenere un riepilogo personalizzato delle attività;
  - **Esplora Viaggi**: Questa funzionalità offre agli utenti la possibilità di cercare viaggi in base ai propri interessi. Sarà possibile filtrare le opzioni in modo da trovare esperienze che rispecchino le preferenze individuali, garantendo una ricerca personalizzata e mirata;
  - **Elenco Viaggi**: In questa sezione, gli utenti avranno una panoramica completa di tutti i viaggi a cui hanno preso parte o che hanno organizzato. Questo elenco include sia le esperienze organizzate personalmente che quelle in cui hanno partecipato, fornendo una cronologia dettagliata delle avventure passate.

  2.1 mira a ottimizzare l'esperienza dell'utente, fornendo un accesso rapido alle informazioni essenziali e alle funzionalità principali dell'applicazione.

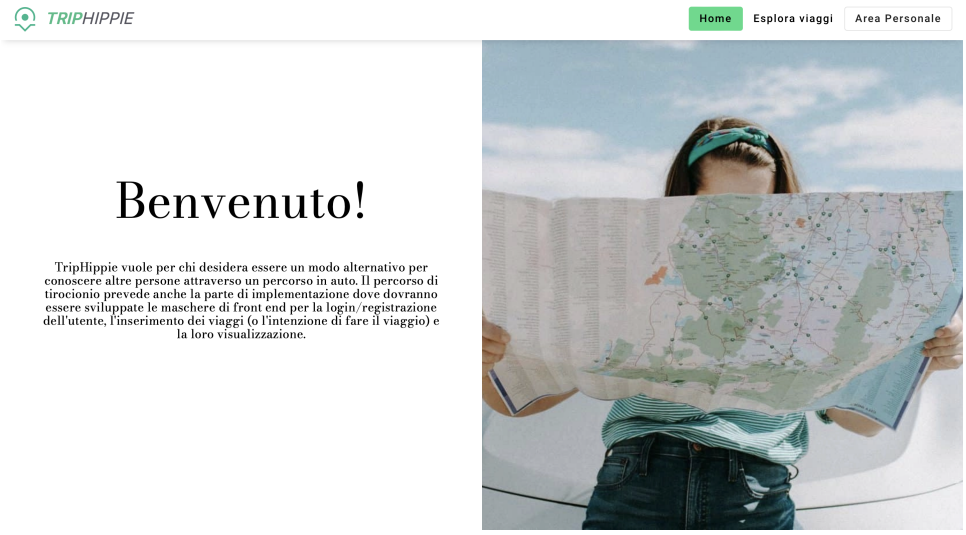

Figura 2.1: Interfaccia principale del progetto *TripHippie*

- L'interfaccia secondaria (Area Personale, 2.2) fornirà un accesso intuitivo alla propria area personale, consentendo agli utenti di esplorare facilmente diverse sezioni. Tra le opzioni disponibili, si avrà l'opportunità di:
  - **I miei viaggi**: Questa sezione rappresenta uno spazio dedicato alla visualizzazione di tutti i propri viaggi, suddivisi in due schermate, una contenente quelli organizzati e l'altra contenente quelli partecipati.
  - **Inizia Viaggio**: Questa funzionalità offre agli utenti la possibilità di organizzare viaggi in base ai propri interessi, inserendo tutti i dati necessari;
  - **Modifica Profilo**: In questa sezione, gli utenti hanno la possibilità di modificare i propri dati con cui si sono registrati all'applicazione.

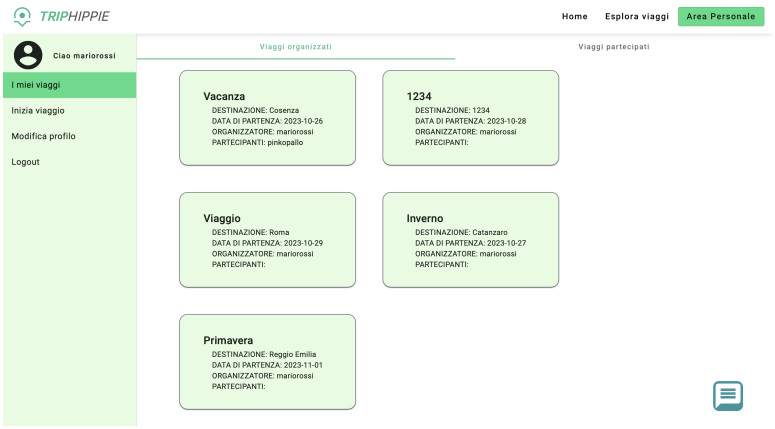

Figura 2.2: Area personale all'interno del progetto *TripHippie*

- Visualizzazione di tutti i dettagli relativi ad un singolo viaggio;

2.3 Scopo dello stage

Una volta identificati i principali componenti richiesti dal progetto, SyncLab ha creato varie aree di formazione coinvolgendo numerosi stagisti per approfondire lo studio, la progettazione e l'implementazione delle diverse parti dell'applicativo. Ciascun componente è stato assegnato a diversi tirocinanti, e per ciascuno è stata elaborata una strategia mirata a consentire lo sviluppo simultaneo delle varie parti, mirando a ottenere un prodotto funzionante nel minor tempo possibile.

Nel mio caso specifico, in collaborazione con il tutor aziendale, l'ingegnere Fabio Pallaro, abbiamo definito una serie di argomenti di studio rilevanti per questo progetto, focalizzandoci sulla ricerca, lo sviluppo e l'innovazione nel contesto del *web*. Tra i vari aspetti di studio richiesti, l'azienda ha proposto un'analisi delle tecnologie nel campo dello sviluppo *web*, soprattutto per quanto riguarda il lato frontend. Inoltre, è stata richiesta un'approfondita analisi dei protocolli per la comunicazione bidirezionale in tempo reale, con l'obiettivo di implementare specifiche funzionalità nel progetto *TripHippie*. Queste funzionalità prevedono la creazione di interfacce utente che consentiranno agli utenti di interagire attraverso chat con altri partecipanti al progetto.

Un esempio di tali protocolli è il *WebSocket* che, come mostrato nella figura 2.3, permette di aprire bidirezionale e *full-duplex* tra un *browser* e un *server web*. *WebSocket* è un protocollo ampiamente utilizzato anche da giganti del *web* come *Facebook* e piattaforme di comunicazione popolari come *Slack*. In particolare, *Facebook* sfrutta i *WebSocket* per abilitare una comunicazione efficiente in tempo reale, consentendo anche l'invio immediato di notifiche *push*, come ad esempio quelle relative a messaggi, richieste di amicizia o attività degli amici.
Questo aspetto sottolinea l'importanza strategica che SyncLab attribuisce all'adozione di tecnologie avanzate, supportando così l'innovazione nel suo ambito di competenza.

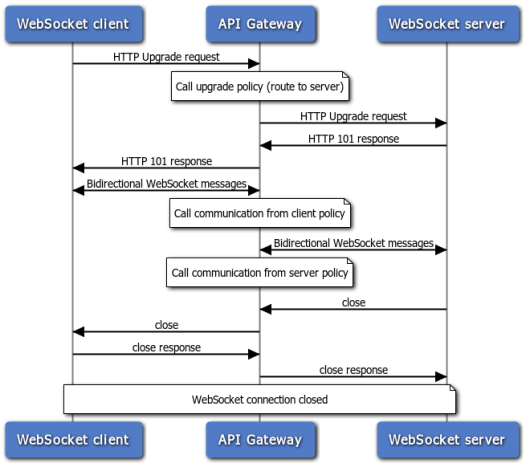

Figura 2.3: Funzionamento del protocollo *WebSocket*

Fonte: docs.oracle.com

A partire da questo approfondimento, ho concordato sia con l'azienda che con gli altri stagisti, responsabili dell'implementazione di questa funzionalità, di sviluppare un *proof of concept*. L'obiettivo era chiarire il funzionamento dei protocolli di comunicazione a un livello strettamente tecnico, valutando attentamente il grado di efficacia e la reale sicurezza per ciascuno di essi. Il medesimo *proof of concept*, elaborato durante la fase di progettazione, è stato successivamente utilizzato per integrare una soluzione nell'applicativo già esistente.
Tale soluzione doveva essere in grado di gestire completamente le chat degli utenti e le richieste verso il *backend*, facendo uso del protocollo Hypertext Transfer Protocol (HTTP)[g].

Durante questo processo, sono stati presi in considerazione diversi aspetti chiave, tra cui:

- L'implementazione di una connessione sicura con gli altri utenti;
- La gestione della lista degli utenti, distinguendo chi è *online* da chi è *offline*;
- La differenziazione delle connessioni con vari utenti per rafforzare la sicurezza complessiva;
- La gestione delle richieste asincrone mediante il protocollo HTTP, rivolte a risorse protette nel backend attraverso un servizio in *Application Program Interface (API)*[g]REST;
- La gestione graduale dell'aumento del numero di utenti e delle attività nella chat;
- La gestione fluida del *logout* dall'applicativo.

Complessivamente, in collaborazione con l'azienda, abbiamo definito una serie di argomenti mirati a arricchire il mio bagaglio di conoscenze, concentrandoci soprattutto sull'acquisizione di nuovi linguaggi e tecnologie. Questo approccio mirato ha contribuito in modo significativo alla mia crescita professionale, specialmente nell'ambito dello sviluppo di applicazioni *web*.

2.4 Vincoli organizzativi

Per avviare il progetto di stage in modo efficace, ho concordato sin dall'inizio con il mio tutor aziendale sugli argomenti, gli obiettivi, i vincoli e la pianificazione temporale. L'obiettivo era affrontare in modo completo tutti gli aspetti relativi alla mia parte di prodotto, stabilendo in modo chiaro e preciso le modalità di interazione remota attraverso lo *smart working*. Il piano di lavoro è stato poi esaminato e confermato sia dal mio relatore, il professor Tullio Vardanega, che dal mio tutor aziendale, l'ingegnere Fabio Pallaro.

L'attività di stage è stata organizzata in modalità mista, lavorando in presenza due volte a settimana e in modalità *smart working* per le restanti tre giornate. Questa pianificazione ibrida ha evitato intoppi organizzativi, consentendomi di sviluppare il prodotto richiesto nei tempi concordati. In collaborazione con il tutor aziendale e il relatore, abbiamo progettato un piano di lavoro che soddisfacesse la necessità di interazione con entrambe le parti, garantendo un costante aggiornamento su ogni progresso compiuto.

Per quanto riguarda il tutor aziendale, è stata creata una bacheca all'interno della piattaforma Trello. Questa bacheca ha raccolto tutte le attività settimanali, e ogni settimana è stato effettuato un breve colloquio per verificare lo stato di avanzamento.

Per quanto riguarda il relatore, ho mantenuto un costante flusso di comunicazione informandolo regolarmente dei miei progressi tramite email ogni cinque giorni lavorativi. Questo approccio garantiva una trasparenza continua sul mio avanzamento rispetto al piano di lavoro.

2.5 Obiettivi attesi

2.5.1 Notazione

In accordo con il mio tutor aziendale, il riferimento ai requisiti segue le seguenti notazioni:

- 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.

Le sigle precedentemente indicate sono seguite da una coppia sequenziale di numeri, identificativo del requisito.

2.5.2 Obiettivi di qualità

Si prevede di raggiungere i seguenti obiettivi:

- **Obbligatori**:
  - Verificabilità del codice;
  - Manutenibilità del codice;
- **Opzionali**:
  - Portabilità del codice;
  - Riusabilità del codice;

2.5.3 *Testing*

Nel contesto dello sviluppo del progetto, è essenziale dedicare una sezione alle attività di testing che saranno svolte. Questo momento cruciale del ciclo di sviluppo garantisce la validazione, la qualità e la robustezza dell'applicazione in fase di realizzazione.
Attraverso una serie di test mirati, ci proponiamo di assicurare che ogni componente e funzionalità risponda in modo affidabile e coerente alle specifiche del progetto. Le diverse fasi di testing pianificate consentono inoltre di delineare le strategie da adottare per garantire un prodotto finale affidabile e conforme alle aspettative.

Le attività di testing per verificare il raggiungimento degli questi obiettivi elencati nella sottosezione precedente sono:

- La verifica della corretta inizializzazione e nomenclatura delle variabili utilizzate;
- Tracciamento in ingresso, uscita, eccezioni delle procedura/metodi/funzioni;
- Numero massimo di linee di codice per procedura/metodi/funzioni;
- Verifica dei limiti delle strutture dati utilizzati (es. array);
- Test di unità;
- Utilizzo di *tool* di analisi statica.

2.5.4 Prodotti attesi

In relazione all'obiettivo dell'esperienza di stage, l'azienda ha formulato la necessità di sviluppare e consegnare una serie di prodotti, i quali sono stati chiaramente delineati in punti distinti, come segue:

- **Primo punto**: Sviluppo del codice di *frontend* ed eventuali parti di *backend*;
- **Secondo punto**: Documento tecnico che descriva le scelte fatte durante gli sviluppi.

2.6 Motivazione della scelta

Annualmente, l'Università di Padova promuove StageIt, un incontro dedicato alle principali aziende del territorio e agli studenti, facilitando l'ingresso nel mondo professionale attraverso proposte di stage che si integrano perfettamente con il percorso di studi della Laurea triennale in Informatica. La realizzazione di questo evento è possibile grazie alla collaborazione del professor Tullio Vardanega e dell'associazione degli imprenditori AssIndustria VenetoCentro.

La partecipazione a StageIt è stata un passo fondamentale nella mia esperienza, permettendomi di entrare in contatto con diverse aziende e valutare varie opportunità di tirocinio. Questo contesto mi ha offerto l'opportunità di esplorare tematiche di mio interesse attraverso colloqui con diverse realtà aziendali.
L'evento si è svolto in presenza, con ogni azienda che aveva uno stand dedicato, fornendo una piattaforma per presentare progetti agli studenti. Questo approccio ha consentito di seguire più aziende e condurre colloqui multipli, massimizzando l'efficienza del tempo a disposizione.

Inizialmente, non avevo un'idea precisa del percorso che avrei voluto intraprendere. La mia strategia è stata quella di svolgere il maggior numero possibile di colloqui per poi valutare, tra le varie proposte, quella più stimolante. La mia intenzione era di approfondire tematiche legate allo sviluppo web, concentrandomi sull'integrazione e sull'utilizzo di nuove tecnologie per rispondere alle esigenze del mercato IT. Grazie al progetto di Ingegneria del *Software*, avevo già iniziato a esplorare nuove tecnologie *web*, come Angular (un *framework web* in *Typescript* per lo sviluppo di siti *web*). Tuttavia,

volevo approfondire ulteriormente un argomento affine, coinvolgendo un'attività di progettazione di servizi essenziali alla *business logic* di un sito web, andando oltre la creazione di semplici maschere.

La comunicazione in tempo reale su Internet era un altro ambito che desideravo esplorare ulteriormente. Lo studio e l'approfondimento dei protocolli di comunicazione rappresentavano un aspetto interessante, considerando le molteplici opportunità lavorative. SyncLab mi ha presentato un percorso di stage che avrebbe abbracciato tutti questi aspetti, proponendo un progetto con un prodotto innovativo sul mercato. Questo progetto costituiva una sfida stimolante, poiché avrei dovuto imparare nuove tecnologie, un aspetto cruciale in un contesto di evoluzione continua delle richieste del settore. L'azienda ha dimostrato grande professionalità, disponibilità immediata e sicurezza, sottolineate dalla consistenza del personale, dalle risorse a loro disposizione e dalla vasta rete di clienti nel settore ICT. La scelta di collaborare con SyncLab è stata allineata con le aree di ricerca che desideravo esplorare per ampliare le mie competenze.

Capitolo 3

Il progetto di stage

3.1 Pianificazione

La durata complessiva dello stage è stata di 8 settimane, equivalenti a un totale di 320 ore di lavoro effettive. Come già accennato, ho svolto il tirocinio sia in modalità smart working che in presenza presso la sede aziendale.
Di seguito, viene presentata la ripartizione settimanale delle attività:

PIANIFICAZIONE SETTIMANALE		
SETTIMANA	**ORE TOTALI**	**ATTIVITÀ**
1	40	Incontro con le persone coinvolte nel progetto per discutere i requisiti e le richieste relativamente al sistema da sviluppare
		Presentazione strumenti di lavoro per la condivisione del materiale di studio e per la gestione dell'avanzamento;
		Condivisione scaletta di argomenti
		Ripasso concetti metodologia agile/*scrum*
		Ripasso del linguaggio Java SE
		Ripasso concetti *Web* (*Servlet, servizi Rest, JSON*[g] *ecc.)*
		Studio principi generali di *Spring Core* (*IOC, Dependency Injection*)
2	40	Studio *SpringBoot*
		Studio *Spring Data*/*DataRest*
		Realizzazione prototipo di servizio REST con metodi *get*[g]/*post*[g]/*put*[g]/*delete*[g]
3	40	Ripasso linguaggio *JavaScript*
		Studio del linguaggio *TypeScript*
4	40	Studio piattaforma *NodeJS* e AngularCLI
		Studio *framework* Angular
		Realizzazione di due maschere demo di *login* con successiva *HomePage* prototipale
5	40	Analisi e studio del progetto *TripHippie*
		Progettazione ed implementazione della nuova maschera di *chat*
6	40	Progettazione ed implementazione nuova maschera "*chat*"
		Scrittura dei *service* (su *frontend*) di chiamata al *backend*
7	40	Eventuale implementazione su *backend* per la funzionalità di *chat*
8	40	Termine integrazioni e collaudo finale
TOTALE ORE		**320**

Tabella 3.1: Tabella riassuntiva delle ore settimanali di stage

In merito alla suddivisione oraria delle attività principali svolte durante il tirocinio, ogni singola attività è stata pianificata in modo da coprire integralmente tutte le fasi di sviluppo del prodotto atteso.
Di seguito, viene presentata una panoramica delle principali attività e della distribuzione oraria:

RIPARTIZIONE ORARIA			
DESCRIZIONE ATTIVITÀ	**RIPARTIZIONE**	**ORE**	**ORE TOTALI**
Formazione sulle tecnologie			**38**
Definizione architettura di riferimento e relativa documentazione	Analisi del problema e del dominio applicativo	12	**38**
	Progettazione della piattaforma e relativi test	22	
	Stesura documentazione relativa ad analisi e progettazione	4	
Collaudo finale	Collaudo	30	**38**
	Stesura documentazione finale	5	
	Incontro di presentazione della piattaforma con gli *stakeholders*	1	
	Live demo di tutto il lavoro di stage	2	

Tabella 3.2: Ripartizione oraria delle principali attività di sviluppo

3.2 *Way of working*

3.2.1 Metodo di lavoro

Come gia menzionato nella sezione 1.2, dedicata ai processi interni, SyncLab adotta il metodo agile, con particolare attenzione al *framework* Scrum, come modello di sviluppo. Questo stesso approccio è stato implementato durante lo stage, offrendo l'opportunità di acquisire familiarità con le pratiche operative dell'azienda e di esplorare in concreto le dinamiche e l'efficacia di un'implementazione Scrum nell'ambito dello *smart working*.

Riguardo alle dinamiche di comunicazione e le videoconferenze, in conformità con le indicazioni del tutor aziendale, ho fatto uso di alcune delle tecnologie precedentemente trattate nella sezione 1.3, di seguito riportate:

- ***Discord***: Attraverso questo strumento, ho instaurato comunicazioni dirette con il tutor aziendale e i colleghi, sfruttando diversi canali vocali e testuali, ciascuno dedicato a specifici argomenti di discussione;
- ***Google Meet***: Ho impiegato questo strumento per partecipare alle riunioni programmate. Mi ha permesso inoltre di condurre una presentazione conclusiva al termine del mio periodo di stage, illustrando quanto realizzato;

- **Trello**: Utilizzando Trello, ho pianificato, come mostrato in figura 1.5 in anticipo la lista delle attività e degli sprint in collaborazione con il tutor aziendale. Ogni settimana ho monitorato e tracciato le attività svolte, aggiornando lo stato corrispondente sulla bacheca (*Scrum Board*) per valutare il progresso in ottica agile.

Ogni settimana, ho partecipato attivamente a riunioni cruciali che si sono concentrate su aspetti fondamentali per lo svolgimento del mio tirocinio, affrontando tematiche sia in modo diretto che trasversale. Questi incontri, svolti con un gruppo di lavoro più limitato, poiché coinvolgevano solo coloro che si occupavano di una specifica funzionalità, avevano l'obiettivo di definire in modo dettagliato l'ambito su cui concentrarsi, esaminando accuratamente i passaggi di ogni progetto.

In termini di configurazione, l'azienda proponente ha fornito una struttura organizzativa che racchiudeva tutti i *repository* di progetto su *GitHub*, comprendente l'insieme completo del prodotto.

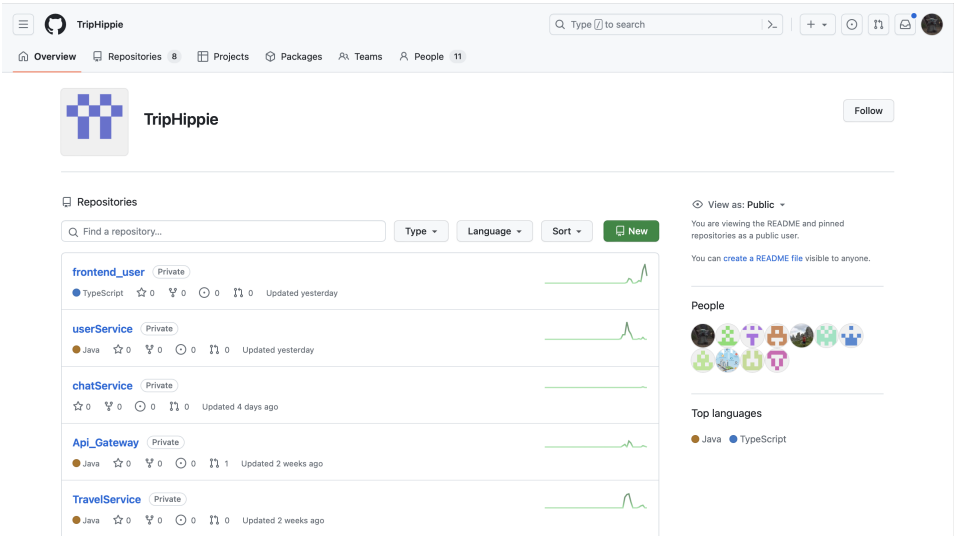

Figura 3.1: Organizzazione *GitHub* del progetto *TripHippie*

Fonte: github.com

In base ai prodotti attesi, ho concordato con l'azienda proponente i principali obiettivi di qualità che devono essere perseguiti a livello di processo e di prodotto.

3.2.2 Conseguimento degli obiettivi di qualità

Riguardo agli obiettivi qualitativi del prodotto, ho deciso di adottare un approccio di verifica e validazione, sia per il *software* che per la documentazione, considerandoli entrambi come un'unica entità di prodotto. In particolare, ho definito quanto segue:

- **Verifica**: un processo di controllo attraverso il quale garantisco la qualità dei processi di fornitura del prodotto;
- **Validazione**: un processo di controllo del prodotto, volto a confermare le aspettative, i requisiti e le funzionalità concordate;

Attraverso questi processi, cerco di garantire che il prodotto rispecchi principalmente i seguenti canoni:

- **Adeguatezza funzionale**: il prodotto deve essere completo, funzionalmente corretto e appropriato al contesto in cui verrà utilizzato;
- **Conformità**: il prodotto deve seguire gli standard e le convenzioni rilevanti, nell'ambito del proprio contesto operativo;
- **Usabilità**: il prodotto deve essere fruibile e comprensibile da parte degli utenti principali, garantendo operabilità e protezione dagli errori;
- **Sicurezza**: il prodotto deve garantire la sicurezza nella manipolazione dei dati, assicurando integrità e controlli nelle funzionalità più vulnerabili;
- **Manutenibilità**: il prodotto deve essere modulare e riutilizzabile in base alle necessità dell'azienda;
- **Affidabilità**: quanto realizzato deve essere utilizzabile per un lungo periodo di tempo e facilmente riparabile in caso di guasti e malfunzionamenti.

3.2.3 Conseguimento della qualità di processo

In relazione alla qualità di processo, mi sono impegnato a perseguire i principi di efficacia ed efficienza nel corso dello sviluppo, cercando di costituire sempre qualcosa di valido e funzionante ad ogni incremento.
In particolare:

- **Efficacia**: il prodotto deve soddisfare tutte le richieste dell'azienda e deve essere convalidato rispetto a quanto pianificato;
- **Efficienza**: i processi che portano alla realizzazione del prodotto devono convergere con costi ridotti in termini di risorse, garantendo al contempo la stessa qualità del prodotto, rientrando quindi nel tempo pianificato a mia disposizione.

3.2.4 Interazioni con il tutor aziendale

La comunicazione costante e collaborativa ha contraddistinto le interazioni con il tutor aziendale, aspetti fondamentali per il buon andamento del progetto di stage. Abbiamo adottato diverse modalità per mantenere un dialogo efficace e assicurare una comprensione chiara degli obiettivi e delle attività pianificate.

Abbiamo stabilito riunioni periodiche, sia orizzontali che verticali, per affrontare tematiche generali del progetto e concentrarci su specifiche funzionalità o componenti. Le riunioni orizzontali coinvolgevano sia l'intero *team* del progetto che persone esterne al progetto stesso, consentendo di definire i singoli componenti da frammentare e sviluppare per l'applicativo *TripHippie*, assegnando priorità e risorse necessarie. Le riunioni verticali, invece, coinvolgevano un gruppo più ristretto per analizzare dettagliatamente gli aspetti specifici di ciascun progetto.

Abbiamo utilizzato piattaforme di comunicazione come *Discord* per interagire in modo diretto attraverso canali vocali e testuali, facilitando la discussione su temi specifici. Abbiamo inoltre impiegato *Google Meet* per le riunioni programmate, consentendo

anche la presentazione di risultati al termine del percorso di stage.

L'approccio collaborativo e multidirezionale ha quindi caratterizzato le interazioni con il tutor aziendale, garantendo un flusso costante di informazioni e una chiara comprensione degli obiettivi del progetto.

3.2.5 Revisioni di progresso

Per la gestione delle attività e la tracciatura degli avanzamenti, abbiamo adottato Trello, organizzando una lista di attività e sprint preventivamente concordati. Ogni settimana, ho aggiornato lo stato delle attività sulla bacheca (*Scrum Board*), consentendo un monitoraggio costante del progresso in ottica agile.

Durante lo stage, le riunioni settimanali con il mio tutor aziendale hanno garantito un'efficace gestione e un costante monitoraggio delle attività svolte. Abbiamo adottato diverse modalità per assicurare una comunicazione chiara e una valutazione accurata degli avanzamenti.

Durante le riunioni individuali settimanali, abbiamo analizzato lo stato di avanzamento rispetto al piano di lavoro precedentemente definito. Utilizzando la bacheca di Trello, abbiamo valutato le attività completate, verificato eventuali ritardi e discusso le prossime tappe. Questo approccio ha consentito di mantenere una visione chiara delle attività svolte e di pianificare le azioni future in modo sinergico.

In aggiunta, le riunioni collettive settimanali coinvolgevano tutti i membri del *team* di sviluppo dedicati alla stessa funzionalità. In questo contesto, abbiamo esaminato l'avanzamento generale rispetto alle funzionalità da implementare, assicurandoci che fossero robuste e prive di errori o malfunzionamenti. Questa verifica ci ha permesso di prendere decisioni informate sui passi successivi e di affrontare eventuali sfide in modo collaborativo.

L'utilizzo di strumenti come Trello ha agevolato la tracciatura delle attività e ha favorito una gestione efficiente del progetto. Le riunioni settimanali hanno rappresentato momenti chiave per la valutazione critica del lavoro svolto e per la pianificazione delle attività future, garantendo un processo di stage ben strutturato e orientato agli obiettivi.

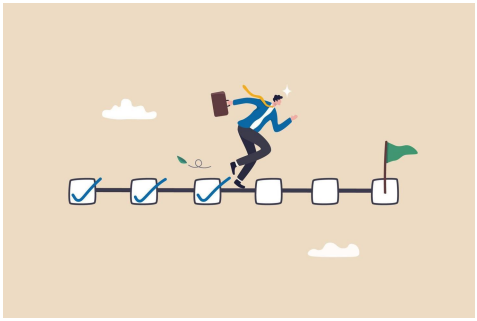

Figura 3.2: Solo attraverso buone revisioni di progresso si arriva a destinazione

Fonte: it.vecteezy.com

3.2.6 Utilizzo di diagrammi

L'utilizzo dei diagrammi "*Way of Working*[g]" durante lo stage ha svolto un ruolo essenziale nel rendere trasparenti il processo di sviluppo e le modalità operative in modo chiaro e comprensibile. Questi diagrammi, spesso rappresentati attraverso diagrammi di flusso o mappe concettuali, hanno offerto una visione visiva e schematicamente organizzata del modo in cui il lavoro veniva svolto nell'ambito del progetto.

La creazione e l'utilizzo dei diagrammi *Way of Working (WOW)* hanno avuto diversi scopi, tra cui:

- **Comunicazione Efficace**: I diagrammi hanno fornito un mezzo visivo per comunicare in modo chiaro il modo in cui le attività erano strutturate, connesse e integrate nel contesto più ampio del progetto;
- **Orientamento del Team**: I diagrammi *WOW* hanno permesso di orientare il team rispetto alle procedure, ai processi e alle pratiche da seguire durante lo sviluppo. Ciò ha facilitato la comprensione delle responsabilità di ciascun membro del team e ha contribuito a mantenere coerenza e coesione;
- **Formazione e *Onboarding***: I diagrammi *WOW* sono stati utilizzati per facilitare l'*onboarding* di nuovi membri del team o di stagisti, fornendo loro una risorsa didattica, offrendo una panoramica visiva delle metodologie di lavoro adottate;
- **Identificazione di Miglioramenti**: Esaminando i diagrammi, abbiamo identificato inefficienze e aree di miglioramento nei processi, contribuendo all'ottimizzazione del flusso di lavoro durante lo stage;
- **Riferimento Costante**: I diagrammi *WOW* sono diventati un riferimento costante per tutti i membri del *team*, fornendo una guida visuale pronta all'uso per le attività quotidiane.

L'uso di diagrammi *Way of Working* ha quindi favorito una comprensione più profonda e strutturata del processo di sviluppo, promuovendo una collaborazione più efficiente e un allineamento comune all'interno del *team*.

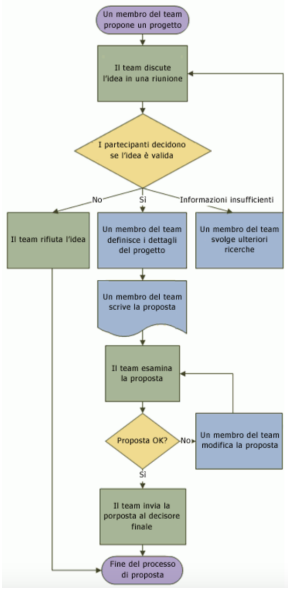

Figura 3.3: Diagramma di flusso mostrante le fasi successive ad una proposta di progetto

Fonte: microsoft.com

3.2.7 Tecniche di analisi

Nel contesto di un progetto, le tecniche di analisi giocano un ruolo cruciale nel comprendere, valutare e interpretare dati, requisiti e performance. Queste tecniche vengono utilizzate per identificare *trend*, problemi, e per prendere decisioni informate durante tutte le fasi del ciclo di vita del progetto.
Durante questo progetto, abbiamo adottato una serie di metodologie e approcci operativi per affrontare le sfide e conseguire gli obiettivi. Alcune delle principali strategie utilizzate includono:

- **Analisi dei Requisiti**: L'analisi dei requisiti è un'attività fondamentale iniziale. Coinvolge la raccolta, l'identificazione, l'organizzazione e la documentazione dei requisiti del progetto. Questa analisi può coinvolgere interviste, *workshop*, analisi dei documenti esistenti e conversazioni con gli *stakeholders* per comprendere appieno cosa il progetto deve raggiungere;
- **Analisi SWOT**: L'analisi SWOT (*Strengths*, *Weaknesses*, *Opportunities*, *Threats*) viene utilizzata per valutare gli aspetti interni ed esterni di un progetto. Questa tecnica aiuta a identificare i punti di forza e di debolezza del progetto, insieme alle opportunità e minacce che potrebbero influenzarlo;
- **Analisi dei Processi**: Questa tecnica si concentra sull'esame e miglioramento dei processi coinvolti nel progetto. L'obiettivo è identificare inefficienze, ridondanze o punti critici e proporre soluzioni per ottimizzare il flusso di lavoro;
- **Analisi delle Prestazioni**: L'analisi delle prestazioni coinvolge la valutazione delle performance del progetto rispetto agli obiettivi prefissati. Questa tecnica fornisce un quadro chiaro su come il progetto sta funzionando e se sta raggiungendo gli obiettivi fissati.

3.2.8 Strumenti di verifica

Gli strumenti di verifica adottati per garantire la qualità e la correttezza del lavoro svolto hanno rappresentato un elemento cruciale nel corso di questo progetto.
Abbiamo impiegato una serie di strumenti mirati per controllare, valutare e migliorare costantemente il nostro lavoro:

- **Analisi Statica del Codice**: Ho utilizzato strumenti di analisi statica del codice come *SonarQube* per individuare e correggere potenziali errori, incoerenze o discrepanze nel codice sorgente. Questi strumenti ci hanno aiutato a mantenere uno standard uniforme e a individuare anomalie nel codice prima della sua esecuzione;
- **Test Automatici**: Abbiamo implementato test automatici utilizzando *framework* come *Jasmine*. Questi test hanno permesso di eseguire controlli sistematici delle varie funzionalità del software, garantendo che ogni componente funzioni come previsto e che eventuali modifiche non compromettano il funzionamento del sistema;
- **Strumenti di Controllo della Versione**: Come già accennato nella sezione 1.2, ho adottato Git come sistema di controllo della versione per tenere traccia delle modifiche apportate al codice. Questi strumenti hanno permesso di gestire le varie versioni del software, monitorare le modifiche e, se necessario, ripristinare versioni precedenti;
- **Analisi dinamica del codice** Abbiamo eseguito analisi dinamiche del codice per valutare il comportamento del *software* in esecuzione. Queste analisi includono il *profiling* del codice, l'analisi delle prestazioni e l'individuazione di eventuali falle di sicurezza;
- **Ispezioni e Revisioni del Codice**: Insieme al *team* di progetto ed ai colleghi dell'azienda abbiamo condotto regolari sessioni di ispezione e revisione del codice, in cui le persone esaminavano il lavoro degli altri per individuare errori, proporre migliorie e garantire l'allineamento agli *standard* aziendali;

L'utilizzo combinato di questi strumenti ha consentito di mantenere elevati standard di qualità, garantendo un'implementazione solida e affidabile del progetto.

3.3 Analisi dei requisiti

3.3.1 Casi d'uso

L'analisi dei requisiti rappresenta l'attività preliminare e fondamentale di qualsiasi progetto. L'obiettivo principale è definire con chiarezza e precisione cosa il prodotto dovrà soddisfare, sia dal punto di vista funzionale che non funzionale.

Attori principali

Gli attori individuati sono i seguenti:

- **Utente non autenticato**: un utente che non ha completato il processo di autenticazione o registrazione all'interno dell'applicazione *web*, utilizzando la piattaforma senza aver effettuato l'accesso con credenziali valide o senza aver completato il processo di registrazione;

- **Utente autenticato**: si riferisce a un utente che ha completato il processo di autenticazione all'interno dell'applicazione *web*. Questo utente ha l'autorizzazione per accedere a tutte le informazioni e le funzionalità dell'applicazione, informazioni che sarebbero altrimenti inaccessibili agli utenti non autenticati.

Insieme all'utente autenticato, ho individuato due ulteriori attori all'interno dei casi d'uso: l'utente mittente e il destinatario. Entrambi, essendo utenti autenticati, costituiscono una generalizzazione insieme all'utente autenticato:

- **Utente mittente**: Attore che si pone come l'origine di un messaggio, inviandolo a uno o più destinatari, che possono essere altri utenti o un gruppo di utenti all'interno del sistema;
- **Utente destinatario**: Attore che riceve un messaggio proveniente dall'utente mittente, trovandosi dall'altra parte dell'interazione di messaggistica o di comunicazione all'interno del sistema.

Elenco dei casi d'uso

Per analizzare l'utilizzo del prodotto, insieme agli altri componenti del *team* abbiamo adottato la creazione di diagrammi specifici. I diagrammi dei casi d'uso, noti anche come *Use Case Diagram* nell'ambito *Unified Modeling Language* (*Unified Modeling Language (UML)*[g]), si focalizzano sulla rappresentazione delle funzioni e dei servizi forniti da un sistema. Questa visualizzazione si concentra sulle interazioni degli attori con il sistema, mostrando come gli utenti percepiscono e utilizzano le varie funzionalità offerte.

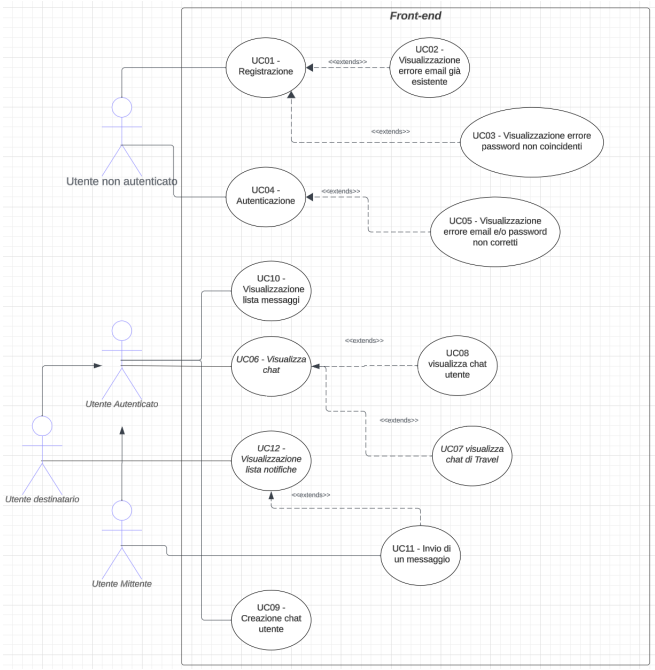

Figura 3.4: Scenario principale

Nella figura 3.4 abbiamo evidenziato numerosi casi d'uso per le funzionalità di *chat* da implementare, definendo ogni requisito con la struttura seguente, indicando:

- Attori principali;
- Precondizioni affinché gli attori principali possano eseguire il caso d'uso;
- Descrizione delle azioni che vogliono svolgere gli attori principali;
- Postcondizioni, successive all'esecuzione del caso d'uso mostrando gli effetti risultanti dall'azione compiuta nel caso d'uso stesso;
- Scenario principale, indicando le azioni che vengono svolte dagli attori principali;
- Eventuali estensioni legate al caso d'uso.

3.3.2 Tracciamento dei requisiti

Il tracciamento dei requisiti è cruciale per mantenere la coerenza e la trasparenza nel processo di sviluppo del prodotto.
Attraverso un'approfondita valutazione dei requisiti e degli scenari d'uso del progetto, è stata compilata una tabella di corrispondenza tra i requisiti e gli scenari d'uso.
Sono stati categorizzati vari tipi di requisiti, ciascuno contrassegnato da un codice identificativo univoco per una distinzione chiara.

Casi d'uso

Il codice identificativo dei casi d'uso ha la seguente struttura:

$$UC[X].[Y] - [T]$$

Dove:

- X: numero del caso d'uso;
- Y: numero del sottocaso del caso d'uso principale;
- T: titolo del caso d'uso.

Requisiti

Ogni requisito è strutturato come segue:

- codice identificativo;
- classificazione;
- descrizione;
- fonti.

Codice identificativo Il codice identificativo dei requisiti ha la seguente struttura:

$$R[X]\_[Y]$$

Dove:

- **X**: Tipologia: può assumere i seguenti valori:
  - **F**: requisito funzionale;
  - **Q**: requisito di qualità;
  - **V**: requisito di vincolo;
- **Y**: numero del requisito per classificazione.

Classificazione La classificazione del requisito può assumere i seguenti valori:

- obbligatorio;
- desiderabile;
- opzionale.

Classificazione Le fonti dei requisiti possono essere le seguenti:

- UC[numero]: indica un caso d'uso;
- Committente: indica il capo progetto;
- Interno: indica il gruppo di stagisti.

Requisiti funzionali

Dopo un'approfondita valutazione dei requisiti e degli scenari d'uso del progetto, abbiamo identificato 33 requisiti funzionali, suddivisi secondo la classificazione precedentemente illustrata:

- 25 requisiti obbligatori;
- 5 requisiti desiderabili;
- 3 requisiti opzionali;

Requisiti di qualità

Dopo un'approfondita valutazione dei requisiti e degli scenari d'uso del progetto, abbiamo identificato 4 requisiti di qualità, suddivisi secondo la classificazione precedentemente illustrata:

- 4 requisiti obbligatori;
- 0 requisiti desiderabili;
- 0 requisiti opzionali;

Requisiti di vincolo

Dopo un'approfondita valutazione dei requisiti e degli scenari d'uso del progetto, abbiamo identificato 6 requisiti di vincolo, suddivisi secondo la classificazione precedentemente illustrata:

- 2 requisiti obbligatori;
- 0 requisiti desiderabili;
- 4 requisiti opzionali;

3.4 *Proof Of Concept*

Per avviare l'implementazione delle funzionalità di *chat*, ho collaborato con l'ingegnere Fabio Pallaro, responsabile del progetto aziendale, per determinare una serie di elementi da presentare tramite un'applicazione di esempio che ho sviluppato insieme ad altri stagisti, noto come *proof of concept*(PoC). Le ragioni che ci hanno spinto a dedicare tempo a questa iniziativa sono dettagliate qui di seguito:

- **Esplorazione pratica delle tecnologie**: Il *PoC* ha consentito di esplorare in maniera pratica le tecnologie identificate durante il processo di analisi. Ho trovato prezioso questo approccio pratico poiché mi ha permesso di comprendere meglio il costo e le risorse necessarie per l'integrazione di ciascuna tecnologia durante l'attività di progettazione;
- ***Baseline* di riferimento**: Il *PoC* ha agito come punto di partenza, offrendo una base di riferimento che ha guidato l'attività di progettazione dettagliata e lo sviluppo del prodotto. In alcuni casi, abbiamo potuto riutilizzare parti delle tecnologie e degli elementi implementati durante questa attività iniziale;
- **Approccio agile**: La realizzazione del *PoC* mi ha consentito di presentare agli *stakeholders* un esempio basilare del prodotto finale, agevolando la comprensione degli obiettivi finali. Questa pratica ha supportato la definizione più accurata delle scelte progettuali sin dalle prime fasi.

Attraverso il *PoC*, abbiamo potuto mostrare in maniera pratica e tangibile ciò che l'azienda si aspettava come prodotto finale. Questo processo mi ha fornito una comprensione più profonda dell'ambito applicativo del progetto, aiutandomi a identificare nel dettaglio le componenti e i servizi richiesti per la realizzazione delle maschere di *chat*.

Partendo dall'idea, il *proof of concept* ha preso le seguenti direzioni:

- Creazione di una connessione *websocket* utilizzando un itbackend per facilitare la comunicazione tra utenti;
- Sviluppo di un servizio basilare per l'autenticazione all'interno di una *web* app, offrendo agli utenti la possibilità di accedere inserendo le loro credenziali o di registrarsi;
- Implementazione di un servizio di comunicazione all'interno del progetto, utilizzando il protocollo HTTP per trasmettere e ricevere messaggi;
- Creazione di un servizio per la gestione delle notifiche, consentendo la gestione della loro posizione e il limite massimo di notifiche gestibili;
- Realizzazione di diverse interfacce per una *web* app di esempio, sfruttando il *framework* Angular e incorporando i relativi servizi e componenti utilizzando *TypeScript*.

Lo sviluppo completo di questi aspetti ha permesso di dimostrare in modo esaustivo le competenze acquisite, offrendo una visione più approfondita del funzionamento dei singoli componenti. In particolare, ho eseguito alcune delle procedure chiave all'interno del *proof of concept* per testare l'efficacia dei protocolli.

Nel contesto del protocollo *WebSocket*, ho condotto test su una pagina privata che

consentiva la creazione di una lista di *chat*. Ho realizzato questo attraverso molteplici registrazioni di utenti, di cui ho mostrato il processo nella figura 3.5, dando vita a una serie di *chat* dopo aver effettuato una richiesta al *backend*. Quest'ultimo, dopo aver stabilito una connessione con gli utenti coinvolti, restituiva al *frontend* l'elenco delle chat.

Come già menzionato, il *Proof of Concept* si basa su un *backend*, il quale però non solo gestisce solo le connessioni *websocket* con gli altri utenti, ma utilizza anche *Postgres* per archiviare i profili utente e i messaggi scambiati nelle *chat* singole.

Spring fornisce supporto per l'integrazione di database *PostgreSQL* attraverso moduli come Spring *Data JPA* e Spring *JDBC*. Questi moduli semplificano l'accesso e la gestione dei dati nel database *PostgreSQL* all'interno di un'applicazione Spring.

Per connettersi ad un database *PostgreSQL* utilizzando Spring, è necessario configurare correttamente le dipendenze nel file *Project Object Model (POM)*[g] [6](questo si tratta di un progetto Maven[8]). Successivamente, è possibile configurare le proprietà di connessione nel file di configurazione di Spring (*application.properties* o *application.yml*), specificando i dettagli di connessione al database *PostgreSQL* come URL, nome utente, *password*, ecc.

Una volta configurata la connessione, è possibile utilizzare le annotazioni e le funzionalità offerte da Spring *Data JPA* per definire entità, *repository* e interrogazioni al database *PostgreSQL* in modo più semplice e dichiarativo. Completate tutte queste fasi, è possibile avviare sia il *backend* che il *frontend* dell'applicazione.

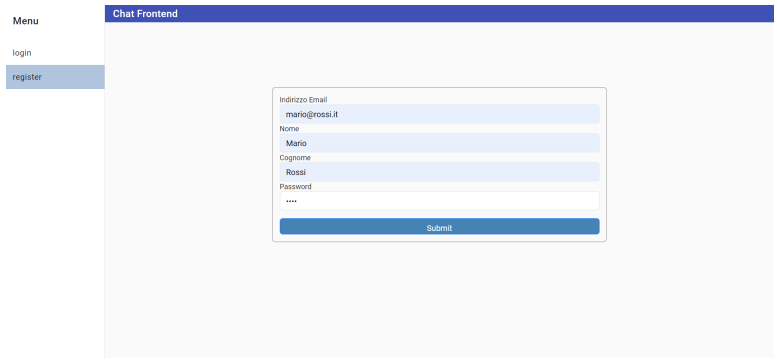

Figura 3.5: Registrazione di un utente

Durante l'accesso, il sistema *backend* utilizza la connessione con il *database PostgreSQL* per eseguire una verifica dei dati inseriti al fine di confermare la loro accuratezza e autenticità.
Questo processo avviene attraverso una richiesta al *database*, consentendo al sistema di verificare la validità dei dati forniti durante la procedura di accesso.
Tale approccio sfrutta la connessione stabilita con *PostgreSQL* per garantire l'integrità e la correttezza delle informazioni fornite dall'utente al momento del *login*.

Poiché si tratta solo di una attività prototipale, non abbiamo sviluppato un sistema per gestire gli errori. Di conseguenza, se vengono inseriti dati non validi durante la registra-

zione o dati errati durante il *login*, tali azioni non vengono completate e il processo di accesso o registrazione non ha luogo.

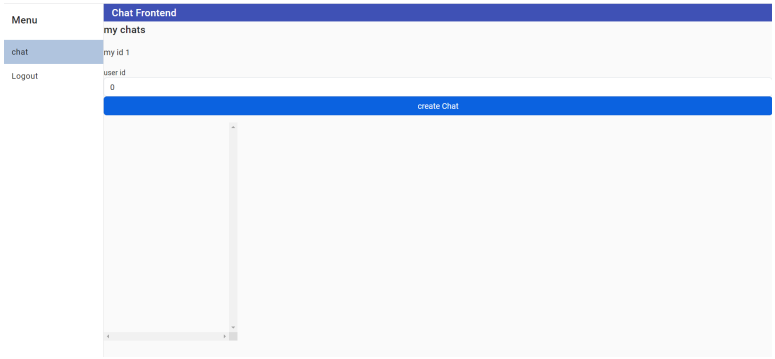

Figura 3.6: Pagina principale

Una volta completata l'autenticazione, il sistema permette all'utente di visualizzare la pagina principale. Come mostrato nella figura 3.6, l'interfaccia utente presentata è di natura elementare, poiché offre principalmente la possibilità di eseguire il *logout* e di avviare nuove conversazioni con altri utenti attraverso la creazione di *chat* e l'invio di messaggi.

Il processo per creare una nuova *chat* segue un approccio simile alla funzionalità di ricerca, poiché l'utente è tenuto a inserire l'identificativo dell'utente desiderato per avviare una conversazione.

Se l'identificativo inserito corrisponde ad un utente nel sistema, viene generata una nuova *chat* associata a quel profilo e viene aggiunta all'elenco delle conversazioni disponibili. Se, al contrario, l'identificativo non è associato a nessun utente registrato nel sistema, non verrà creata alcuna *chat* poiché non è possibile istanziare una conversazione con un utente inesistente.

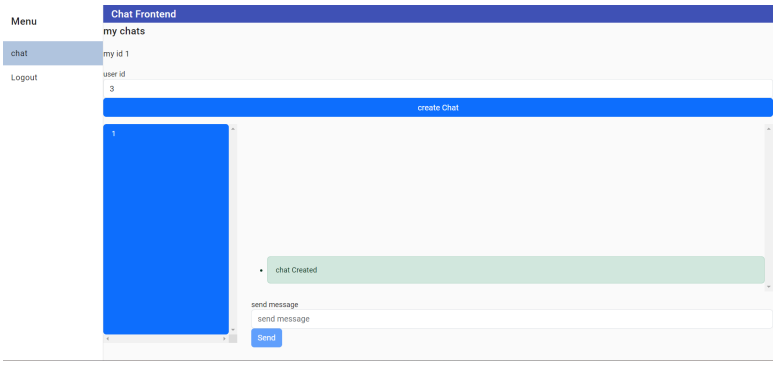

Figura 3.7: Creazione di una *chat* con un altro utente

Una volta stabilita la connessione con un altro utente e creata la *chat* (nella figura 3.7 viene mostrato un esempio), è possibile verificare la corretta configurazione e funzionalità inviando un messaggio. Se il messaggio viene inviato con successo e viene ricevuto dall'altro utente nella *chat* appena creata, conferma che la connessione e la trasmissione dei

messaggi funzionano correttamente. Questo test di invio e ricezione è un buon modo per assicurarsi che la comunicazione tra gli utenti avvenga come previsto dopo la creazione della *chat*.

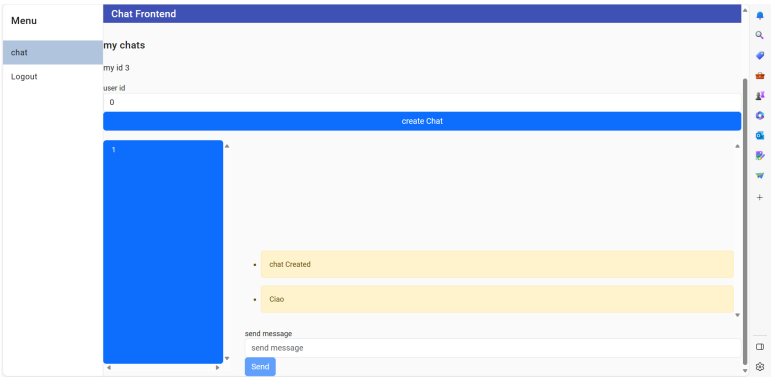

Figura 3.8: Ricezione di un messaggio arrivato da un altro utente

La figura 3.8, visualizzata dall'utente 3, mostra una connessione *websocket* stabilita in precedenza. Nell'elenco a sinistra dello schermo, sono presenti le informazioni sull'identificativo del mittente, in questo caso, l'utente con ID 1. Nel resto della figura, è evidente una notifica relativa alla creazione della *chat*, accompagnata dal messaggio inviato durante questa interazione specifica.

L'implementazione del *PoC* ha incluso l'utilizzo della libreria *ngx-toastr* all'interno del sistema di notifica per soddisfare il requisito di notificare agli utenti i messaggi appena ricevuti. Questo servizio utilizza *ngx-toastr*, un'efficiente libreria di notifiche *pop up* per le applicazioni Angular, per fornire avvisi visivi all'utente durante l'utilizzo dell'applicazione.

Il servizio di notifica si occupa di mostrare un *popup* contenente il nome del mittente e il messaggio appena ricevuto. Questo avviso consente di informare immediatamente l'utente dell'arrivo di un nuovo messaggio, anche se al momento potrebbe essere coinvolto in altre conversazioni all'interno dell'applicazione.
Nella figura 3.9 viene mostrato un esempio di notifica, in cui il *popup* generato mostra il nome del mittente e il messaggio che ha inviato.

Il test eseguito per verificare il funzionamento di questo sistema di notifica consisteva nell'inviare un messaggio e controllare se il servizio di notifica mostrava correttamente il *pop up* con le informazioni del mittente e del messaggio ricevuto. L'integrazione di questa funzionalità nel *PoC* permette di chiudere questo processo di sviluppo e concentrarsi sulla successiva integrazione delle conoscenze acquisite nel progetto.

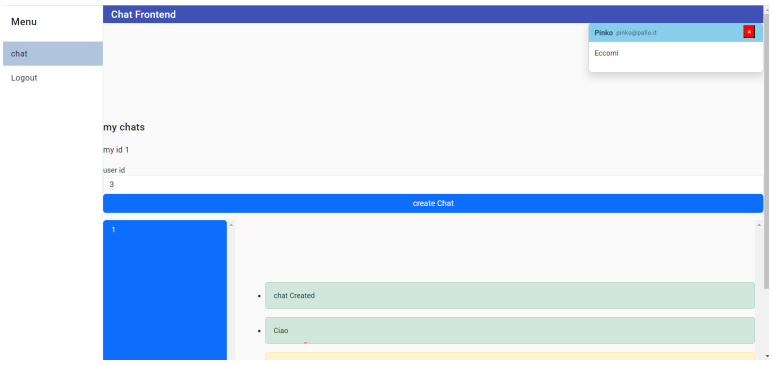

Figura 3.9: Notifica di un messaggio appena arrivato

Per differenziare i messaggi inviati da quelli ricevuti, abbiamo implementato un sistema che utilizza una colorazione distintiva per ciascun tipo di messaggio (figura 3.9).

Abbiamo adottato uno schema di colori dove i messaggi inviati sono visualizzati con uno sfondo di colore diverso rispetto a quelli ricevuti. Questo approccio consente agli utenti di distinguere facilmente tra i messaggi che hanno inviato e quelli che hanno ricevuto.

Questa differenziazione visiva può migliorare significativamente l'usabilità dell'applicazione, consentendo agli utenti di comprendere rapidamente quali messaggi sono hanno inviato loro stessi e quali invece hanno ricevuto da altri partecipanti alla conversazione.

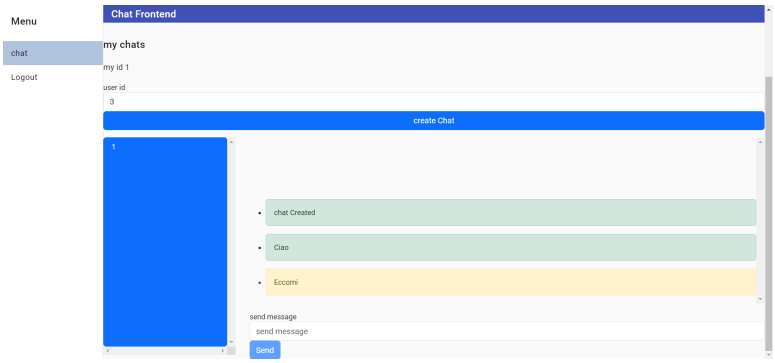

Figura 3.10: Sistema di colorazione dei messaggi

3.5 Progettazione

3.5.1 Tecnologie utilizzate

TypeScript *TypeScript* è un linguaggio di programmazione *open source* che si basa sulle caratteristiche di *ECMAScript* 6. Estende la sintassi di *JavaScript*, consentendo a qualsiasi programma scritto in *JavaScript* di funzionare anche con *TypeScript*. Tra le

sue funzionalità aggiuntive, *TypeScript* offre la possibilità di tipizzare le variabili e di definire interfacce e classi.

Angular Angular[1] è un *framework open source* basato su *JavaScript*, progettato per sviluppare applicazioni *web* dinamiche attraverso un insieme di strumenti e funzionalità integrate. La sua architettura modulare consente una strutturazione ottimale delle applicazioni, favorisce un alto livello di riutilizzo del codice e facilita la manutenibilità grazie alla suddivisione dei componenti, ciascuno adibito a una singola funzione.

Angular Material *Angular Material*[2] è una libreria grafica appositamente creata per Angular, ispirata allo stile grafico di *Google*. Mette a disposizione componenti grafici predefiniti e testati anche per quanto riguarda l'accessibilità, facilitando lo sviluppo dell'aspetto visivo delle pagine *web*.

Node.js Node.js è un *framework* utilizzato da Angular per la gestione delle dipendenze. Consente di dichiarare due tipi di dipendenze:

- Dipendenze necessarie agli sviluppatori;
- Dipendenze richieste per il funzionamento dell'applicativo.

Questa separazione consente di differenziare le librerie necessarie nel processo di *deploy*[g], ad esempio, quelle utilizzate esclusivamente per i test. Inoltre, *Node.js* offre semplici comandi per scaricare le librerie necessarie all'interno del progetto, facilitando operazioni come la *Continuous Integration/Continuous Delivery (CI/CD)*[g].

Cypress *Cypress* è un *framework* di *testing end-to-end* (E2E) per applicazioni *web*. È progettato per automatizzare i test che simulano l'interazione degli utenti con l'applicazione, consentendo di verificare il comportamento dell'applicazione in un ambiente reale del *browser*. *Cypress* offre un'esperienza di sviluppo semplice, integrando strumenti di test, *debug* e asserzioni all'interno di un unico ambiente, agevolando la scrittura e l'esecuzione dei test.

Docker *Docker* [4] è una piattaforma che consente la creazione, la distribuzione e la gestione di applicazioni all'interno di *container*[g] [9]. I *container Docker* rappresentano un metodo di virtualizzazione a livello di sistema operativo che permette di isolare un'applicazione e le sue dipendenze in un ambiente autonomo, chiamato contenitore.

3.5.2 Progettazione della comunicazione in tempo reale

Una volta scelte le tecnologie da utilizzare per lo sviluppo del frontend, il passo successivo era la progettazione della *chat* in tempo reale.
A tale scopo Angular permette di integrare all'interno delle applicazioni due tecnologie:

- *WebSocket*;
- *WebRTC*.

WebSocket

WebSocket è un protocollo di comunicazione bidirezionale, *full-duplex* e basato su TCP, utilizzato per consentire una comunicazione interattiva e in tempo reale tra un *server* e un *client* attraverso una singola connessione persistente. A differenza di HTTP, che è un protocollo di comunicazione unidirezionale, *WebSocket* consente la trasmissione simultanea di dati sia dal *server* al *client* che dal *client* al *server*, senza la necessità di stabilire una nuova connessione ogni volta che si desidera scambiare informazioni.

WebRTC

WebRTC, acronimo di *Web Real-Time Communication*, è invece una tecnologia *open source* che consente la comunicazione in tempo reale, inclusa la voce e la videochiamata, nonché lo scambio di dati tra *browser web* e altre applicazioni. *WebRTC* è un insieme di *API* e protocolli che permettono di stabilire comunicazioni dirette tra i dispositivi degli utenti senza la necessità di installare *plugin*[g] o *software* aggiuntivo.
Le caratteristiche principali di *WebRTC* includono la crittografia *end-to-end* per garantire la sicurezza delle comunicazioni, la compatibilità *cross-browser*, la gestione automatica delle reti *peer-to-peer* e la riduzione della latenza.

Nonostante siano due tecnologie molto simili, la scelta è stata principalmente influenzata dalle esigenze specifiche del progetto. In questo contesto, poiché la struttura dell'applicazione coinvolge sia un *client* che un *server* e poiché sia io che il mio collega responsabile del *backend* avevamo già esperienza con *WebSocket*, abbiamo optato per l'utilizzo di *WebSocket* per implementare una funzionalità di *chat* in tempo reale. Questa scelta ci ha permesso di supportare aggiornamenti in tempo reale e altre interazioni interattive, come la possibilità di inviare messaggi istantanei. Inoltre, abbiamo mantenuto una connessione persistente tra il *server* e il *client* per ridurre al minimo la latenza. Infine, abbiamo scelto *WebSocket* anche per garantire la compatibilità con tutti i principali *browser web*, assicurandoci che l'applicazione fosse accessibile da un'ampia base di utenti.

3.5.3 Soluzioni progettuali

Abbiamo adottato un approccio ponderato e ben strutturato nel definire le soluzioni progettuali per le funzionalità di *chat*, guidato dall'interpretazione dei requisiti del progetto e dall'applicazione delle *best practice* del *framework* di sviluppo, in questo caso Angular, come punto di riferimento. Questo ha garantito un'implementazione allineata alle linee guida e agli standard consigliati dal *framework*.

Lo studio iniziale e approfondito di Angular, ha permesso di discutere e valutare con il tutor aziendale diverse soluzioni di progettazione. Questo approccio collaborativo ha contribuito a identificare le scelte di progettazione più adeguate e adottare le migliori pratiche per lo sviluppo del prodotto, garantendo un risultato finale all'altezza delle aspettative e delle esigenze del progetto.

Comunicazione in tempo reale Per la comunicazione in tempo reale, come menzionato in precedenza, abbiamo scelto il protocollo *WebSocket*. Abbiamo utilizzato questo protocollo insieme a quello HTTP, la scelta di integrare sia il protocollo *WebSocket* che l'HTTP per gestire le comunicazioni in tempo reale e l'invio/ricezione dei messaggi dimostra un approccio combinato e versatile. Mentre *WebSocket* offre una

connessione bidirezionale in tempo reale, HTTP viene utilizzato per il trasferimento dei messaggi. Questa strategia ibrida sfrutta il punto di forza di entrambi i protocolli: la reattività e la persistenza di *WebSocket* insieme alla solidità e alla familiarità di HTTP per la trasmissione dei dati. Incorporare l'esperienza precedente dal PoC evidenzia una pratica che sfrutta le conoscenze acquisite, sfruttando entrambi i protocolli per migliorare l'efficienza e l'affidabilità delle comunicazioni in tempo reale.

Integrazione Insieme agli altri stagisti, abbiamo condotto l'attività iniziale di sviluppo delle funzionalità di *chat* separatamente dal progetto principale. L'obiettivo era creare un'applicazione indipendente che, al momento dell'integrazione nel progetto principale, avremmo potuto integrare senza causare conflitti. Questo approccio ha permesso di concentrarsi sulla creazione di un modulo o un'applicazione chat che sarebbe stata facilmente incorporabile nel contesto più ampio del progetto senza generare complicazioni o problemi di compatibilità.

Observables over Promises L'utilizzo degli *Observables* anziché le *Promise* in Angular indica una scelta avanzata per gestire chiamate asincrone. Mentre le *Promise* gestiscono singoli eventi, gli *Observables* gestiscono zero o più eventi nel tempo, consentendo manipolazioni complesse dei flussi di dati. Offrono maggior flessibilità nella gestione degli eventi, operazioni sui dati e sottoscrizioni, adatti soprattutto a situazioni con flussi di dati multipli e manipolazioni sofisticate.

3.5.4 Design Pattern utilizzati

Angular incorpora diversi *design-pattern* rilevanti, che riflettono soluzioni provenienti dai principi chiave delineati dalla "*Gang of Four*" nel libro "*Design Patterns*"[5]. Le principali integrazioni includono:

- ***Singleton***: Utilizzato ampiamente nei *service* di Angular, gestisce un'unica istanza di classe, permettendo il riutilizzo della stessa istanza per diverse operazioni. Nei servizi, questo *pattern* è stato sfruttato per la gestione delle sessioni e delle chiamate HTTP;
- ***Decorator***: Angular offre diverse funzionalità *decorator*, come *@Injectable()*, che consentono di aggiungere nuove funzionalità agli oggetti esistenti senza dipendenze eccessivamente vincolanti. Questo *design-pattern* è stato utile per iniettare dipendenze in alcune classi del progetto;
- ***Dependency Injection***: Angular mette in evidenza l'uso della *dependency injection*, permettendo di aggiungere dipendenze direttamente durante la costruzione degli oggetti. È stato enfatizzato soprattutto nella costruzione di servizi e componenti, semplificando l'individuazione e l'utilizzo dei servizi nel codice;
- ***MVVM***: Utilizzato da Angular per la comunicazione tra modello dati (*model*) e vista (*view*), favorisce una maggiore separazione tra le classi e i componenti. Rispetto al design MVC, MVVM accresce il ruolo della vista (*View*) e implementa il *ViewModel* come un'estensione del *Model*, integrando funzionalità per la manipolazione dei dati e l'interazione con la vista.
  La figura 3.11 mostra la modalità di comunicazione, che avviene in modo bidirezionale, con la *View* che invia input al *ViewModel*, il *ViewModel* che accede o modifica il Model e notifica la *View* di eventuali aggiornamenti.

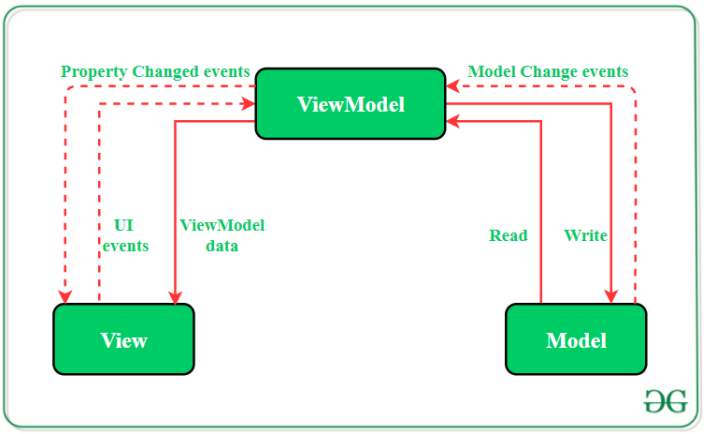

Figura 3.11: Pattern MVVM

Fonte: geeksforgeeks.org

Queste integrazioni dimostrano l'uso avanzato di *design-pattern* all'interno del *framework* Angular, consentendo una gestione efficiente e una struttura organizzativa solida per i servizi, i componenti e le interazioni tra modello e vista nell'applicazione.

3.5.5 Progettazione delle maschere

Dopo la presentazione del *proof of Concept* in una riunione di Stato Avanzamento Lavori (SAL), abbiamo pianificato un incontro con il committente per determinare l'integrazione delle funzionalità mostrate nell'applicativo *TripHippie*.
Durante la progettazione dell'interfaccia, abbiamo deciso di adottare un approccio simile a quello dei siti *web* che collocano la *chat* nell'area inferiore della pagina per l'assistenza. La *chat* viene visualizzata come un *pop-up* che appare e scompare al clic dei rispettivi bottoni dedicati, garantendo così un accesso immediato senza la necessità di dedicare una pagina separata alla funzione di *chat*.
Le regole seguite includono:

- Elenco delle *chat* sul lato sinistro del *pop-up* per offrire un'ampia visibilità delle *chat* disponibili;
- Uso del colore rosso per i bottoni di eliminazione per avvisare gli utenti dell'effetto distruttivo dell'azione associata a quel comando. Aiuta a evidenziare la potenziale conseguenza negativa, come la rimozione di dati o contenuti importanti, cercando di prevenire azioni accidentali;
- Uso di un colore differente per i bottoni disabilitati aiuta gli utenti a distinguere chiaramente lo stato di tali elementi, utilizzando una tonalità più chiara rispetto allo stato attivo per evidenziare visivamente che il bottone non è attualmente disponibile per l'interazione. Questo contrasto visivo aiuta gli utenti a comprendere chiaramente lo stato del bottone e a distinguere tra funzionalità attive e disattivate.

3.6 Codifica

3.6.1 Maschere

Chat

Questa pagina è dedicata agli utenti registrati e autenticati, offrendo, tra le varie funzionalità, la possibilità di avviare conversazioni tramite la *chat* all'interno dell'applicazione *web*.

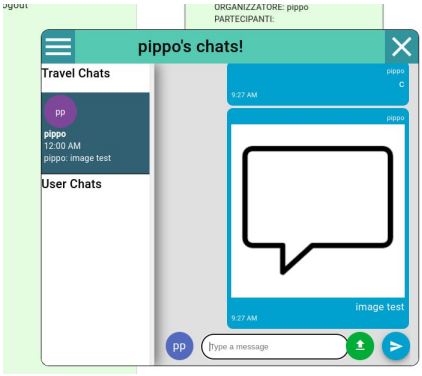

Figura 3.12: *Pop-up* dedicato alla *chat*

3.12 mostra un esempio di *chat* dove è possibile notare che nell'elenco delle chat venga mostrato l'ultimo messaggio inviato nella *chat* e il nome del mittente.

Nella figura 2.2 è rappresentata l'area personale dell'utente, dove è possibile notare che il *pop-up* della *chat* appare chiuso e come venga mostrato con un'icona riconducibile ad un servizio di messaggistica al fine di facilitare la navigazione dell'utente.

Questa maschera fa uso di due componenti:

- *ChatComponent*;
- *MessageComponent*.

Abbiamo inoltre adottato un'approccio di progettazione basato su servizi (*service*) per separare le responsabilità.

Abbiamo quindi identificato due servizi:

- *ws-chatService*: con il compito di gestire le connessioni tramite *WebSocket*;
- *chatService*: con il compito di gestire la comunicazione in tempo reale.

Notifica

Questa pagina è dedicata agli utenti registrati e autenticati, offrendo, tra le varie funzionalità, la possibilità di visualizzare le notifiche relative ai messaggi ricevuti da parte dell'utente.

Figura 3.13: Elenco delle notifiche derivanti dai messaggi ricevuti

Nella figura 3.13, è possibile notare un esempio del sistema di notifica, con ciascuna notifica contenente le informazioni precedentemente inserite. Il numero di notifiche gestibili varia in base alle dimensioni dello schermo poiché ho deciso di riservare solo una colonna per la visualizzazione delle notifiche. Se dovessero essere presenti più notifiche di quante ne possano essere visualizzate, le notifiche eccedenti vengono nascoste fino a quando quelle precedenti non saranno gestite.
Questa maschera fa uso di una sola componente:

- *Toastr-ntf*.

Coerentemente con l'approccio di progettazione adottato basato su servizi (*service*) per separare le responsabilità, ho identificato il seguente servizio:

- *notificationService*.

3.6.2 Componenti

ChatComponent

Questa componente è responsabile della gestione dell'elenco delle *chat* associate a un utente, consentendogli di visualizzare tutti i messaggi all'interno di ciascuna *chat* ogni volta che questa viene selezionata.

Come menzionato prima, l'utente deve essere autenticato per accedere alle funzionalità di *chat*, altrimenti le uniche opzioni disponibili saranno:

- Accedi;
- Registrati;

Se invece l'utente ha eseguito l'autenticazione, accede alla pagina mostrata in figura 2.2 che, oltre ad offrire la funzionalità di organizzare e partecipare a viaggi, offre la possibilità di avviare conversazioni tramite la chat con altri utenti.

Servizi utilizzati

- *ws-chatService*;
- *chatService*.

Metodi Le funzionalità gestite in questa componente includono la visualizzazione dell'elenco di tutte le *chat* associate all'utente e, per ciascuna di esse, la visualizzazione dei messaggi all'interno di ogni chat.
I metodi che vengono utilizzati a tale scopo sono:

- ***ngOnInit()***: All'inizializzazione della componente, viene stabilita una connessione *WebSocket* attraverso il servizio *ws-chatService*, e la componente rimane in ascolto per ricevere i messaggi inviati tramite questa connessione. Una volta stabilita la connessione e messa in ascolto, la componente fa uso del servizio *chatService* per gestire le funzionalità associate.
- ***selectChat()***: Nel contesto della componente, viene gestita la visualizzazione esaustiva dei messaggi presenti all'interno di ciascuna *chat*, permettendo un'esplorazione dettagliata e completa delle conversazioni e delle interazioni avvenute. Per chiamare questo metodo è sufficiente selezionare su una delle *chat* presenti nell'elenco alla sinistra del *pop-up* nella figura 3.12.

MessageComponent

Questa componente avrà il compito di gestire la chat individuale, permettendo all'utente di inviare e ricevere messaggi.

Servizi utilizzati

- *chatService*.

In questa componente, non è necessario utilizzare il servizio relativo ai *WebSocket* poiché il suo unico compito è gestire l'unica funzionalità di chat non gestita nella chatComponent: l'invio dei messaggi.

Metodi

- ***sendMsg()***: Ogni volta che un utente invia un messaggio, il sistema utilizza il servizio *chatService* per inviare il messaggio ai destinatari coinvolti, sfruttando una chiamata di tipo *post*. Questo meccanismo consente di recapitare il messaggio agli utenti destinatari specifici, garantendo la corretta gestione e trasmissione delle informazioni scambiate all'interno della *chat*.

Toastr-ntf

Questa componente è responsabile della gestione, insieme a *chatComponent*, del sistema di notifica, notificando l'utente ogni qualvolta riceva un messaggio da parte di una sua *chat*.
La gestione delle notifiche avviene infatti proprio in *chatComponent*, mentre l'obiettivo di questa componente è gestire una singola notifica alla volta.

Servizi utilizzati

- *notificationService*.

Metodi Le funzionalità gestite in questa componente includono la visualizzazione dell'elenco di tutte le *chat* associate all'utente e, per ciascuna di esse, la visualizzazione dei messaggi all'interno di ogni chat.
I metodi che vengono utilizzati a tale scopo sono:

- ***selectNotification()***: Questo metodo viene attivato quando l'utente clicca su una notifica ricevuta. La sua funzione è quella di aprire la *chat* associata alla notifica, consentendo all'utente di visualizzare direttamente il messaggio ricevuto all'interno di quella specifica *chat*.
  Poiché, come menzionato precedentemente, la gestione del sistema di notifica avviene in un'altra componente, per comunicargli quale notifica è stata selezionata è possibile utilizzare la notazione *@Output*, la quale sfrutta la libreria *@angular/core* per scambiare dati con un'altra componente.

- ***removeNotification()***: Questo metodo, attivato quando l'utente clicca sul bottone rosso di una notifica ricevuta, ha l'unico scopo di far scomparire la notifica stessa senza eseguire altre azioni o cambiamenti nel sistema.

3.6.3 Servizi

Coerentemente con l'approccio adottato basato sulla separazione delle responsabilità, abbiamo creato alcuni servizi per consentire alle componenti di eseguire le operazioni per cui sono state progettate, ad esempio ricevere e inviare messaggi, mentre le componenti stesse si concentravano principalmente sulla presentazione dei dati e sulla creazione dell'interfaccia utente per visualizzare l'elenco delle chat.

ws-chatService

Metodi

- ***connectToSocket()***: Questo metodo viene eseguito all'avvio dell'applicazione per consentire all'utente di creare una connessione *WebSocket* che gli permetta di messaggiare con altri utenti;

- ***connectToChat()***: Questo metodo viene impiegato dal metodo *connectToSocket()* per recuperare tutti i messaggi contenuti nelle *chat* associate all'utente;

- ***sendMsg()***: Questo metodo viene attivato ogni volta che un utente invia un messaggio;

- ***getMessages()***: Questo metodo viene eseguito all'inizio dell'applicazione e consente all'utente di ricevere i messaggi che gli sono stati inviati;

- ***disconnect()***: Questo metodo viene eseguito alla chiusura dell'applicazione da parte dell'utente (che può avvenire tramite il *logout* o la chiusura della pagina *web*, ad esempio) e disattiva la connessione *WebSocket* creata con il metodo *connectToSocket()*.

chatService

Questo servizio fa uso di un oggetto HTTP, fondamentale per poter effettuare le chiamate previste da questo protocollo, che viene "iniettato" attraverso la *Dependency Injection*.

Metodi

- ***createChat***: Questo metodo viene eseguito ogni volta che un utente crea un viaggio o decide di partecipare ad uno già esistente, in quanto consente la creazione della *chat* associata al viaggio;
- ***sendMessage***: Questo metodo viene eseguito ogni volta che un utente invia un messaggio;
- ***getTravelChats***: Questo metodo viene utilizzato esclusivamente all'avvio dell'applicazione per consentire all'utente di visualizzare tutte le *chat* di gruppo associate a lui;
- ***getMessages***: Questo metodo segue la stessa logica di quello precedente, poiché viene attivata quando viene selezionata una *chat* e permette di visualizzare i messaggi al suo interno.

notificationService

Questo servizio fa uso della libreria *ngx-toastr* la quale fornisce notifiche "*toast*" predefinite.
Tuttavia, al fine di personalizzare ulteriormente le notifiche *toast* in modo che si adattino meglio all'aspetto estetico dell'applicazione, ho creato, come menzionato precedentemente, una componente aggiuntiva: *Toastr-ntf*.

Metodi

- **Metodi *"get()"***: Utilizzati esclusivamente per recuperare informazioni specifiche;
- ***showInfo()***: Questo metodo ha il compito di visualizzare le notifiche a schermo, includendo l'username del mittente e il messaggio inviato.

3.7 Verifica e validazione

3.7.1 Analisi statica

Ho effettuato l'analisi statica mediante l'utilizzo della piattaforma *open source SonarQube*, una piattaforma *open-source* di analisi statica del codice che fornisce agli sviluppatori e alle organizzazioni uno strumento per migliorare la qualità del codice, identificare e risolvere i problemi, monitorare le metriche di qualità e applicare le *best practice* di *sviluppo software*. Questa piattaforma analizza il codice sorgente alla ricerca di problemi di codifica, vulnerabilità, *bug*, duplicazioni, cattive abitudini e altre aree di miglioramento.

Il processo inizia con l'ottenimento della piattaforma per l'analisi statica, usando il modulo *"sonarqube-scanner"* che non richiede una completa installazione. Si può

includere il modulo come dipendenza o installarlo globalmente tramite il comando *"npm install -g sonarqube-scanner"*.

Successivamente, viene creato un file di configurazione *"sonar-project.properties"*, specificando una chiave univoca. Avviare il *server SonarQube* con *Docker*[3] usando il comando *"docker run -d –name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest"*.

Una volta configurato il progetto e avviato il server, si esegue l'analisi statica. Si accede a *SonarQube*, si crea manualmente un progetto con chiave identica a quella nel file di configurazione.

Si genera un *token* da *SonarQube*, necessario per effettuare il *login*. Utilizzando questa opzione di analisi *"Locally"*, si inserisce il *token*, si specifica il linguaggio del progetto e il sistema operativo, ottenendo il comando per avviare l'analisi statica.

Infine, eseguendo il comando generato nel terminale, si completa l'analisi e si accede a *SonarQube* per visualizzare i risultati navigando nella sezione "Progetti" e selezionando il progetto di interesse.

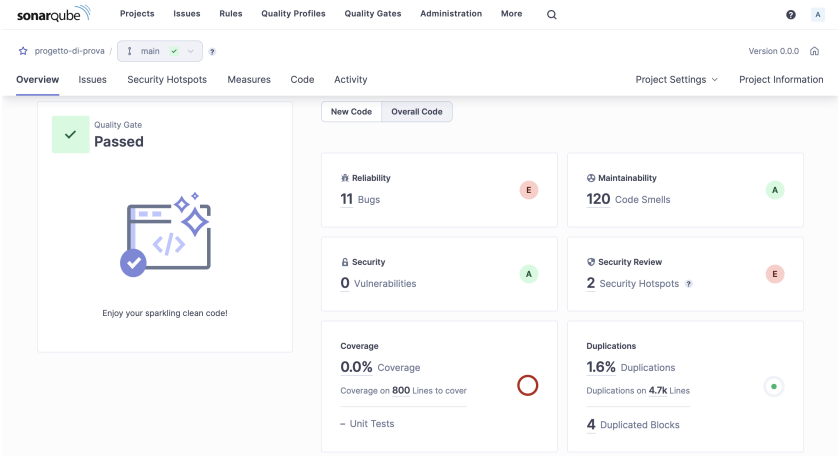

Figura 3.14: Risultati dell'analisi statica effettuata con *SonarQube*

Risultati ottenuti

Il *"Quality Gate"* include misurazioni e metriche specifiche che valutano aspetti come la copertura del codice, la complessità del codice, la presenza di *bug*, la conformità agli *standard* di codifica e altre caratteristiche del *software*. Il superamento di questi criteri, come mostrato nella figura 3.14, indica che il codice soddisfa gli standard di qualità richiesti.

Tuttavia, sebbene i risultati ottenuti siano stati altamente positivi per quanto riguarda la sicurezza del codice, la sua manutenibilità e la minimizzazione del codice duplicato, non possiamo dire lo stesso per quanto riguarda l'affidabilità e la scoperta

di *"security hotspot"*.

In merito all'affidabilità, *SonarQube* ha individuato 11 difetti che potrebbero comportare comportamenti anomali o il blocco dell'applicazione. Per quanto riguarda i *"security hotspot"*, è essenziale porre attenzione alle autorizzazioni, poiché alcune immagini *Docker* vengono avviate con privilegi di amministratore, aumentando il rischio di potenziali danni. Inoltre, l'analisi statica ha evidenziato una debolezza nella crittografia dell'applicazione, che costituisce una potenziale vulnerabilità.

3.7.2 Interazione con l'interfaccia utente da un punto di vista esterno

Cypress è uno strumento di testing *end-to-end* per le applicazioni *web*, che consente di scrivere e eseguire test automatici simulando le azioni degli utenti. Una volta installato tramite *npm*, è necessario configurare il *file tsconfig.json* per indicare al compilatore *TypeScript* di considerare solo le definizioni di tipo da *Cypress* e di compilare l'applicazione in *JavaScript ECMAScript* 5.

È importante escludere determinati *file* e cartelle come *cypress.config.ts*, *cypress* e *node_modules* per evitare conflitti di tipi con *Jest*, specificando tali esclusioni nel file *tsconfig.json*.

Avviare *Cypress* è semplice: basta eseguire il comando *"npx cypress open"* per aprire l'interfaccia di *Cypress* e accedere alle opzioni di gestione dei test. Da qui è possibile esplorare i *file .spec.ts* presenti nel progetto e avviare i test desiderati. La configurazione può essere adattata per esaminare automaticamente tutti i file *.spec.ts*, evitando l'inserimento manuale dei singoli file.

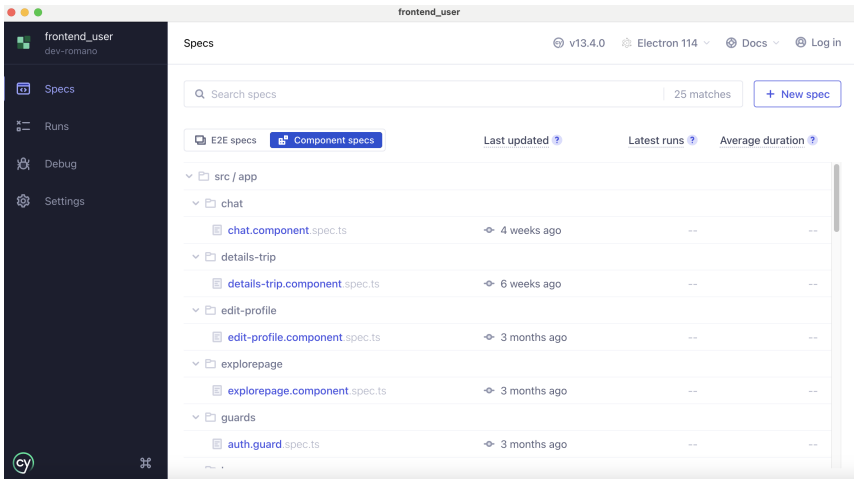

Figura 3.15: Elenco dei *file* di test presenti all'interno dell'applicazione

La visualizzazione che si aprirà sarà simile a quanto mostrato nella figura 3.15, consentendo di esplorare tutti i *file .spec.ts* presenti all'interno del progetto. Basta poi fare clic su uno di essi per esaminare i risultati delle analisi condotte sul *file* e sulla componente corrispondente.

3.7.3 Revisione collettiva

Regolarmente, sia in presenza del *team* di lavoro che di persone al di fuori del progetto, abbiamo tenuto incontri periodici al fine di esaminare lo stato d'avanzamento dei lavori. Durante tali riunioni, veniva presentato un modello funzionante dell'applicazione, accompagnato da una spiegazione delle scelte progettuali adottate, delle funzionalità già implementate e di quelle ancora da sviluppare.
In un secondo momento, ci si dedicava ad un'attività di verifica approfondita, conducendo test di stress per valutare il comportamento dell'applicazione in situazioni specifiche al fine di mettere alla prova la solidità del codice sorgente.

3.8 Consuntivo finale

Durante lo stage, ho svolto un ruolo cruciale nello sviluppo e nell'integrazione delle funzionalità di *chat* nell'applicazione *web TripHippie*. Attraverso un'approfondita analisi dei requisiti, ho compreso appieno le sfide e le opportunità offerte dal progetto.

Mi sono concentrato principalmente sulla progettazione e implementazione delle funzionalità di *chat*, considerando aspetti chiave come l'interfaccia utente, la gestione delle connessioni *WebSocket*, la notifica di messaggi e la corretta comunicazione in tempo reale tra gli utenti.

L'utilizzo di strumenti come *Cypress* per il *testing* automatizzato e *SonarQube* per l'analisi statica del codice ha giocato un ruolo fondamentale nell'assicurare l'efficienza e la correttezza delle funzionalità implementate.

L'integrazione di un sistema di notifica basato su *ngx-toastr* ha migliorato l'esperienza utente, fornendo notifiche chiare e immediate in caso di nuovi messaggi nella *chat*.

Complessivamente, ho incentrato il mio lavoro sull'efficienza, la qualità e l'integrazione delle funzionalità di *chat* nell'ambito dell'applicativo esistente, garantendo un miglioramento significativo dell'esperienza complessiva degli utenti.

Capitolo 4

Conclusioni

4.1 Raggiungimento degli obiettivi

Al termine dello stage, ho proceduto insieme al tutor aziendale a rivedere gli obiettivi iniziali di stage (Obiettivi attesi) per determinare il loro grado di soddisfazione.
Durante questa valutazione, ho analizzato sia i prodotti previsti, sia gli obiettivi qualitativi e quantitativi del mio lavoro, mettendo in luce le aree in cui ho soddisfatto le aspettative e quelle in cui avrei potuto migliorare durante il percorso.

Obiettivi di prodotto Le richieste di funzionalità comprendevano la creazione di *chat* individuali tra utenti, *chat* di gruppo (multiutente) e un sistema di notifica per informare gli utenti sui nuovi messaggi. Tuttavia, durante lo sviluppo, abbiamo realizzato solo due di queste funzionalità: le *chat* multiutente e il sistema di notifica. Abbiamo scelto di concentrarci sul sistema di notifica ponderando la scelta, considerando la sua priorità e l'importanza di integrare correttamente le *chat* multiutente all'interno del progetto *TripHippie*.

Le somiglianze nelle implementazioni delle due tipologie di *chat* hanno influito sulla decisione. Implementando le *chat* multiutente, avremmo ottenuto anche la base per la realizzazione delle *chat* individuali con pochi aggiustamenti, semplificando notevolmente l'implementazione futura delle *chat* singole.

A corredo di quanto realizzato, ho redatto, come richiesto dall'azienda, un documento tecnico in cui ho spiegato, in maniera dettagliata, tutte le scelte fatte durante la progettazione e l'implementazione delle funzionalità.
Ho iniziato con un'analisi approfondita dei requisiti e ho integrato le nuove funzionalità richieste senza generare conflitti con le funzionalità esistenti.

Obiettivi di qualità Il secondo aspetto su cui, durante l'analisi conclusiva, sia io che il tutor aziendale ci siamo focalizzati è l'adempimento degli obiettivi di qualità, riportati nella sezione 2.5.
Come riportato nella sezione 3.7, ho condotto l'attività di verifica e l'attività di validazione ottenendo buoni risultati, che mi hanno permesso di confermare la qualità del prodotto da me realizzato.

4.2 Conoscenze acquisite

Durante lo stage ho approfondito un gran numero di argomenti, tra i quali:

- **Protocolli di comunicazione in tempo reale**: Durante la prima fase dello stage, ho dedicato tempo all'analisi di due protocolli fondamentali per la comunicazione in tempo reale sul *web*, adottati anche da aziende di rilievo. In particolare, ho esaminato *WebSocket*, che garantisce interazioni bidirezionali e in tempo reale tra un *browser* e un server, mantenendo una connessione persistente e affidabile. Ho anche approfondito WebRTC, impiegato per chiamate vocali, videochiamate e lo scambio sicuro ed efficiente di dati attraverso reti *peer-to-peer*;
- **Nuovi linguaggi di programmazione**: Durante la prima fase dello stage, ho dedicato tempo allo studio approfondito del *framework* Angular e dei relativi strumenti di supporto per lo sviluppo e il testing, come *SonarQube* e *Cypress*. L'approfondimento di linguaggi come *TypeScript* e *JavaScript* ha arricchito la mia comprensione nello sviluppo di applicazioni *web*, consentendomi di scoprire nuove funzionalità, come le espressioni lambda e gli oggetti *Observables*, fondamentali per gestire gli eventi all'interno delle applicazioni;
- **Metodo lavorativo aziendale**: Durante il tirocinio, ho acquisito una comprensione pratica del metodo *Scrum* adottato dall'azienda. Questo mi ha permesso di organizzare in modo efficace le attività e gli obiettivi, garantendo il rispetto dei tempi stabiliti. Ho partecipato attivamente a tutti gli incontri programmati, sfruttando appieno le risorse a mia disposizione. Le interazioni con i colleghi sono state preziose: ho imparato da loro e ho avuto modo di confrontarmi durante le riunioni orizzontali. Le presentazioni di gruppo mi hanno inoltre fornito l'opportunità di esporre concetti in modo chiaro e sintetico di fronte agli altri membri del *team*.

Queste competenze hanno arricchito il mio percorso professionale, offrendomi una visione più ampia e pratica nel campo della comunicazione in tempo reale e nella programmazione, oltre a fornire una solida comprensione delle metodologie di lavoro aziendali.

4.3 Valutazione personale

L'esperienza di stage ha arricchito enormemente il mio bagaglio culturale e pratico. Ha messo alla prova sia le mie competenze professionali che personali, permettendomi di identificare punti di forza e debolezza. Ho potuto apprezzare l'importanza dell'autonomia nello studio di argomenti inediti, un'abilità cruciale nel mondo del lavoro. Guardando indietro, ritengo l'esperienza soddisfacente e vitale per la mia crescita professionale e personale.
Durante lo stage ho sperimentato una crescita significativa sia a livello personale che professionale. È stato il mio primo vero approccio al mondo lavorativo, mi ha consentito di comprendere meglio le mie capacità e i limiti personali, migliorando le mie competenze comunicative e di gestione del tempo. Grazie a questa opportunità ho potuto applicare conoscenze universitarie, migliorando il mio lavoro di squadra e acquisendo una migliore comprensione del contesto lavorativo.

4.4 Università e mondo lavorativo a confronto

Mi sono confrontato con una realtà diversa, scoprendo come il mondo universitario e quello lavorativo siano complementari. Le competenze acquisite all'università sono preziose, ma alcune informazioni potrebbero risultare datate rispetto alle necessità del mercato. Il mondo del lavoro richiede una maggiore metodicità e capacità organizzativa. Tuttavia, mentre l'università tende a mantenersi in una sfera di pensiero consolidato, il lavoro è più dinamico e in costante evoluzione. Integrare le due sfere potrebbe essere costruttivo, consentendo all'università di apprendere dalle aziende e viceversa. Personalmente, l'esperienza è stata positiva e mi ha permesso di apprezzare il mondo lavorativo. Ho trovato grande interesse nel progetto, in quanto ho esplorato un contesto che in passato avevo trovato molto interessante ma avevo poi trascurato, apportando innovazione e utilità. L'interazione con i colleghi è stata formativa, in un ambiente collaborativo. Sia il tutor aziendale che il tutor interno hanno svolto un ruolo fondamentale per delimitare il percorso e mantenere un ritmo costante nel progetto. In università, ho acquisito competenze utili per affrontare il mondo del lavoro, ma ho riscontrato carenze nell'incentivare l'uso di strumenti come Git, fondamentali per il lavoro di gruppo. Inoltre, alcuni corsi, nonostante il loro contributo fosse comunque presente e ben strutturato, potrebbero essere aggiornati per stimolare maggiormente l'interesse. Questa esperienza mi ha aiutato a delineare meglio le mie future scelte universitarie e lavorative, preparandomi ad affrontare le richieste del mercato in modo più informato.

Acronimi e abbreviazioni

API Application Program Interface. 23, 45, 60

CI Configuration Item. 60, 62

CI/CD Continuous Integration/Continuous Delivery. 44, 60

HTTP Hypertext Transfer Protocol. 23, 39, 45, 46, 52, 61

IDE Integrated Development Environment. 5, 6, 59

JSON Javascript Object Notation. 28, 59

MVC Model-View-Controller. 14, 46, 61

MVVM Model-View-ViewModel. 14, 46, 61

POM Project Object Model. 40, 59

UML Unified Modeling Language. 36, 62

WOW Way of Working. 33, 62

Glossario

agile Insieme di metodi di sviluppo del software emersi a partire dai primi anni 2000, fondati su un insieme di principi comuni, direttamente o indirettamente derivati dai princìpi del *Manifesto per lo sviluppo agile del software[7]*. 9, 10, 28–30, 32, 39, 60

API in informatica con il termine *Application Programming Interface API* (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. 59

baseline Nel ciclo di vita di un progetto, punto d'arrivo tecnico dal quale non si retrocede, sul quale calcolare l'avanzamento del lavoro in un progetto. 60, 62

Configuration Item Singolo elemento sottoposto ad attività di gestione di configurazione. Ogni CI è identificato univocamente da:

- Codice Identificativo (ID);
- Nome;
- Data;
- Autore;
- Registro delle modifiche;
- Stato corrente;

. 59

container Processo isolato dal resto del sistema che rappresenta l'istanza in esecuzione di una data immagine, secondo opportune configurazioni aggiuntive. 44, 60

Continuous Integration/Continuous Delivery Approccio che automatizza il processo di integrazione del codice e la distribuzione del software. CI si occupa dell'integrazione continua del codice, eseguendo test automatici, mentre CD si concentra sulla distribuzione automatica delle modifiche nel software. Insieme, migliorano l'efficienza, la rapidità e la qualità dello sviluppo del software. 59

DELETE Nel protocollo HTTP, viene utilizzato per rimuovere una risorsa specificata dal server. 28, 60

deploy L'insieme delle attività che permettono ad una componente software di essere disponibile ad un determinato uso. 44, 61

design pattern In ingegneria del software rappresentano soluzioni progettuali ripetibili a problemi comuni nello sviluppo del software. Offrono approcci strutturati e testati per risolvere ricorrenti sfide di progettazione, permettendo la creazione di codice più efficiente, manutenibile e riutilizzabile. Essi fungono da modelli o template per risolvere specifici problemi architetturali o di design nel software. 15, 46, 47, 61

framework Insieme di strumenti, librerie e convenzioni progettuali che forniscono una struttura predefinita per lo sviluppo di *software*. Sono progettati per semplificare e accelerare il processo di sviluppo fornendo funzionalità comuni e una struttura predefinita per costruire applicazioni. iii, 2, 5, 6, 10, 13–15, 25, 28, 35, 44, 45, 47, 57, 61

GET Nel protocollo HTTP, viene utilizzato per richiedere dati da una risorsa specificata. Viene utilizzato per ottenere informazioni, e non ha effetti collaterali sul server. 28, 61

HTTP Nell'ambito dell'invio e ricezione di dati, *HTTP, Hypertext Transfer Protocol* è il protocollo fondamentale utilizzato per la trasmissione di dati su Internet. Si occupa della comunicazione tra *client* e *server*, permettendo al *client* di richiedere e ricevere risorse come pagine *web* o *file*. Le richieste sono inviate in varie forme (*GET*, *POST*, *PUT*, *DELETE*) e ogni richiesta contiene informazioni sull'operazione desiderata. HTTP è senza stato, il che significa che ogni richiesta è trattata singolarmente senza conservare informazioni tra le richieste. Questo protocollo è alla base della navigazione e della trasmissione di dati *online*. 59

MVC In ingegneria del software *MVC, Model-View-Controller* è un design pattern che suddivide un'applicazione in tre componenti: il Model gestisce i dati e la logica di business, la View mostra l'interfaccia utente, e il Controller gestisce l'input dell'utente e il flusso dei dati tra Model e View. Questa separazione delle responsabilità facilita la manutenzione e la gestione dell'applicazione. 59

MVVM In ingegneria del software *MVVM, Model-View-ViewModel* è un pattern architetturale che separa chiaramente la logica di presentazione (UI) dalla logica di business in un'applicazione. Il Model rappresenta i dati e la logica di business, la View è l'interfaccia utente e il ViewModel funge da intermediario tra Model e View, preparando i dati per la presentazione e gestendo le interazioni utente. Questo design pattern favorisce la modularità, il riutilizzo del codice e semplifica la manutenzione dell'applicazione. 59

plugin Componente software che fornisce funzionalità aggiuntive ad estensione di un determinato prodotto software preesistente ma separatamente da esso. 3, 45, 61

POST Nel protocollo HTTP, viene utilizzato per inviare dati per essere processati a una risorsa specificata. Viene utilizzato per inviare dati come file, form dati, oggetti JSON, ecc. al server. 28, 50, 61

PUT Nel protocollo HTTP, viene utilizzato per aggiornare o sostituire una risorsa esistente o, se non esistente, per creare una nuova risorsa con il contenuto fornito. 28, 62

repository Base di dati centralizzata nella quale risiedono, individualmente, tutti i Configuration Item (CI) di ogni baseline nella loro storia completa. 6, 7, 30, 62

Scrum Framework Agile utilizzato principalmente nello sviluppo software per gestire progetti complessi. Si basa su cicli di lavoro iterativi e incrementali, chiamati sprint, in cui un team collabora per sviluppare, testare e consegnare prodotti o funzionalità specifiche entro un periodo di tempo definito. Scrum favorisce la flessibilità, la trasparenza e il coinvolgimento continuo del cliente nel processo di sviluppo.. 10–12, 28, 62

stakeholders Letteralmente, "portatore di interesse". Insieme delle persone che a vario titolo sono coinvolte nel ciclo di vita del software avendo influenza sul prodotto o sul processo; fanno parte di questa categoria fornitori, committenti e clienti. 9, 10, 12, 29, 34, 39, 62

UML In ingegneria del software *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. 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. 59

Way of Working Insieme di pratiche, procedure e metodologie adottate da un team o un'organizzazione per gestire il lavoro, le interazioni e le attività quotidiane in modo efficiente e coerente. Essenzialmente, rappresenta il modo in cui le persone collaborano, comunicano e affrontano le sfide all'interno di un contesto lavorativo specifico. 33, 59

Bibliografia

Riferimenti bibliografici

[5] Erich Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994 (cit. a p. 46).

Siti web consultati

[1] *Angular Docs*. URL: https://angular.io/docs (cit. a p. 44).

[2] *Angular Material Components*. URL: https://material.angular.io/components/categories (cit. a p. 44).

[3] *Docker Commands*. URL: https://docs.docker.com/engine/reference/commandline/docker/ (cit. a p. 53).

[4] *Docker Getting Started Guide*. URL: https://www.docker.com/get-started/ (cit. a p. 44).

[6] *Introduction To The POM*. URL: https://maven.apache.org/guides/introduction/introduction-to-the-pom.html (cit. a p. 40).

[7] *Manifesto Agile*. URL: http://agilemanifesto.org/iso/it/ (cit. a p. 60).

[8] *Maven Getting Started Guide*. URL: https://maven.apache.org/guides/getting-started/index.html (cit. a p. 40).

[9] *What is a Container?* URL: https://www.docker.com/what-container (cit. a p. 44).