Università degli Studi di Trento
Facoltà di Informatica
Corso di Ingegneria del Software

FixMi
Report Finale

Gruppo G43:
Giovanni Santini
Riginel Ungureanu
Valerio Asaro

Anno Accademico 2023/2024
Trento

CONTENTS

0.1 Scopo del documento

Il seguente documento contiene una valutazione finale del processo e l'organizzazione per lo sviluppo dell'applicazione "FixMi".

0.2 Informazioni del Documento

Campo	Valore
Titolo del Documento	Report Finale
Titolo del Progetto	FixMi
Autori del Documento	Giovanni Santini Riginel Ungureanu Valerio Asaro
Amministratore Progetto	Riginel Ungureanu
Versione del documento	1.1

CHAPTER

ONE

REPORT

1.1 Approcci all'ingegneria del software

1.1.1 Blue Tensor

Riassunto

L'ingegnere Jonni Malacarne inizia la presentazione raccontando la nascita di "Blue Tensor", un'azienda nata a Firenze nel 2018 che si pone come obiettivo lo sviluppo di prodotti basati sull'intelligenza artificiale in ambito industriale. Il principale punto di forza di Blue Tensor, sottolinea l'ingegnere, è la forza di coesione del team di sviluppo in quanto affrontare progetti di grandi dimensioni può risultare talvolta complicato. Dopo aver mostrato alcuni prodotti offerti da Blue Tensor, con particolare attenzione per la "Computer Vision", il CEO continua il suo discorso spiegando l'approccio adottato dal suo team per lo sviluppo del software. Una dettagliata composizione del gruppo lavorativo, dotato di ruoli ed occupazioni ben specifiche, viene poi seguito da una "RoadMap" dell'intero progetto, rendendo l'intera produzione più fluida e semplice da gestire. Blue Tensor, in particolare, adotta un approccio a cascata, formato da 6 "Work Packages":

- WP0: PMO and Quality Assurance
- WP1: Analisi e Fattibilità
- WP2: Setup e Design
- WP3: Data Preparation

- WP4: Sviluppo, Training, Test, Deploy
- WP5: Rilascio Finale

Il CEO conclude la presentazione con una lista di strumenti ingegneristici per l'analisi e la comunicazione del cliente che il team di Blue Tensor utilizza:

- UML
  - Use Case Diagrams
  - Activity Diagrams
  - State Machine Diagrams
  - Architecture Diagrams
- E-R per i DataBase
- Mockups - per il design UI
  - Low-Fidelity Mockups
  - High-Fidelity Mockups
- Prototipi
- Mappe Mentali

Cosa abbiamo imparato

Abbiamo imparato che il primo passo per il raggiungimento di un obiettivo è un'ottima squadra di sviluppo, che sappia individuare i propri punti di forza ma anche di debolezza e che sappia agire di conseguenza. Come secondo passo, invece, c'è una rigorosa pianificazione dal punto di vista organizzativo del progetto, cercando di non lasciare niente al caso.

Il nostro approccio

Il nostro team ha agito in linee molto simili a quelle presentate dal CEO di Blue Tensor: anche noi abbiamo individuato dei ruoli nella nostra squadra, ci siamo organizzati in base a questi ultimi ed abbiamo suddiviso in maniera equa ed intelligente il lavoro rispettando i limiti di ogni singolo individuo all'interno del gruppo.

1.1.2 Kanban

Riassunto

Il seminario ha mostrato, tramite un laboratorio interattivo, le potenzialità del metodo Kanban. Nonostante il termine Kanban venga utilizzato del mondo automobilistico, nell'informatica Kabnan è un metodo per gestire il lavoro utilizzato da molte aziende come Microsoft durante lo svilupo di X-box. Il metodo si basa sulla divisione del lavoro in diverse sezioni, come "Design", "Implementation" e "Done". Una certa task passerà tramite queste fasi di sviluppo, in modo da fornire una visione chiara dello stato di una task. kanban si basa sul fatto che ci può essere un numero limitato di task su cui si sta lavorando contemporaneamente. Questo permette di concentrare l'intero team su un unico obiettivo e completare tasks più in fretta. Lo scopo del laboratorio è stato mostrare come questo sistema funziona.

Cosa abbiamo imparato

Un sistema di gestione del lavoro è fondamentale per un processo di sviluppo efficiente e produttivo. Kanban fornisce un metodo di organizzare il lavoro per questi scopi.

Il nostro approccio

Anche se non è stato usato il metodo Kanban, durante lo sviluppo il team ha diviso il lavoro in più task e si è concentrato su un obiettivo alla volta.

1.1.3 IBM

Riassunto

Il seminario viene presentato dal Cloud Technical Specialist Ferdinando Gorga di IBM. Comincia presentandoci i servizi cloud di IBM, i diversi livelli di astrazione dal bare metal al serverless, e i vantaggi della piattaforma. Procede facendo l'esempio di un'architettura molto vulnerabile, e i vari metodi per migliorarla utilizzando tecnologie cloud. Successivamente ci mostra il catalogo di IBM cloud, dove è possibile acquistare varie componenti che funzionano tra di loro senza problemi, per poi mostrarci il code engine, che è alla base del serverless. Una particolarità è la load distribution, in quanto le istanze dell'applicazione si moltiplicano e dividono in base al carico, arrivando perfino a spegnersi in caso

di mancanza di traffico. Inoltre ci mostra un semplice linguaggio di scripting grafico nominato NodeRED.

Infine ci mostra il concetto di Cloud Satellite: Un modo per integrare i propri computer e server all'interno dell'ecosistema cloud di IBM.

Cosa abbiamo imparato

Abbiamo imparato quanto sia importante il cloud al giorno d'oggi, in quanto i giusti servizio e tecnologie permettono di dedicarsi al codice e non all'infrastruttura sottostante.

Il nostro approccio

Il deploy non lo abbiamo fatto su nessuna piattaforma di cloud, tuttavia abbiamo seguito l'approccio dei container per i nostri microservizi. Grazie a ciò non soffriamo di problemi di compatibilità tra le macchine, il software è portabile e siamo certi che con un po' di lavoro si potrebbe rilasciare sul cloud.

1.1.4 META

Riassunto

L'ingegnere del software Heorhi Raik presenta la sua esperienza all'interno di "META", una delle più grandi aziende informatiche del mondo. Dopo aver raccontato brevemente i suoi studi, la sua carriera e il suo ruolo all'interno dell'azienda, pone grande attenzione sui ruoli lavorativi che META propone: Solitamente, dopo una serie di colloqui in cui l'attenzione viene posta più sulle capacità di pensiero critico e logico dell'individuo piuttosto che sulla conoscenza di specifici linguaggi di programmazione, si inizia il proprio percorso all'interno di "META" con il titolo di "Junior": il tipico programmatore addetto alla scrittura e produzione di codice. Dopo qualche anno, tuttavia, il ruolo di "Junior" si evolve in "Senior". L'importanza dell'individuo all'interno dell'azienda aumenta, in quanto adesso può essere a capo di gruppi di sviluppo, di progetti o addirittura di interi dipartimenti. È chiaro, quindi, che le capacità richieste al tipico "Senior" sono organizzative e di pianificazione, caratteristiche di un vero e proprio "Ingegnere del Software". La presentazione si conclude con una breve parentesi sugli strumenti di sviluppo utilizzati all'interno di META.

Cosa abbiamo imparato

Entrare a far parte di un'azienda di grande importanza come "META" non richiede necessariamente abilità in specifici linguaggi di programmazione o particolari modi di scrivere codice. Anzi, si potrebbe far parte di un colosso dell'informatica scrivendo solo una o due righe di codice al giorno: il ruolo di leader del gruppo può essere talvolta più complicato e impegnativo. Proprio per questo, non bisogna sottovalutare il ruolo di "Software Engineer", in quanto ha le abilità necessarie per potersi porre a capo di un eventuale team di sviluppo.

Il nostro approccio

All'interno del nostro gruppo abbiamo posto grande attenzione nella scelta del leader, considerando la quantità di conoscenze possedute ma soprattutto le capacità amministrative e organizzative. Questo ha sicuramente reso il percorso di sviluppo più semplice e meno tortuoso.

1.1.5 U-Hopper

Riassunto

Daniele Miorandi, Chief AI Officer di U-Hopper, ha dedicato il seminario per familiarizzare gli studenti con tecnologie utilizzate in azienda, portando l'esempio di U-Hopper. Le tecnologie mostrate sono le seguenti:

- **Ingestion**: Per reperire grandi quantità di dati, vengono utilizzati dei sistemi a coda come dei buffers e dei lettori e scrittori. Esempi di queste tecnologie sono Kafka, MQTT
- **Batch Processing**: Computazione in blocchi su cluster. Esempio di Spark.
- **Stream Processing**: Elaborazione dei dati in streams, spesso usati nei sistemi di pagamento. alcune tecnologie sono Spark Streaming e Flink
- **In-memory DB**: Tenere tutti i dati in memoria per una maggiore velocità di lettura. Esempio: Redis
- **Storage**: PostgeSQL, MongoDB, MySQL, Cassandra (Per grandi dati molto sparsi, inserisce per colonne), CouchDB, Elastic (NpSQL per letture e scritture veloci), InfluxDB (lavoro per serie temporali come dati

di sensori e logs), MinIO (Object storage), TileDB(gestione di dati cartografici)

- **Task Management / Orchestration**: Tecnologie per controllare l'esecuzione e i flow dei dati, rappresentati tramite DAG. Ad esempio Celery e AirFlow
- **Logs**: Sentry, ELK, TIG
- **Linguaggi di programmazione**: Python, Scala, go
- **AI**: MLFlow, Tenserflow, Pythorch
- **DEployment**: Docker, IASS
- **DEsign Patterns**: Microservizi, Event Driven Architecture, Asynchronous
- **Processi e strumenti per SE**: Agile, CI / CD, documentazione

Cosa abbiamo imparato

Esistono numerose tecnologie per la realizzazione di software utilizzate in azienda. Queste tecnologie permettono di risolvere problemi o migliorare il processo di sviluppo.

Il nostro approccio

Nella creazione dello stack per la nostra applicazione, abbiamo tenuto conto delle tecnologie più utilizzate nell'industria compatibili con i nostri requisiti ed esperienza. Abbiamo utilizzato un design pattern basato sui microservizi ed una serie di tecnologie utili allo sviluppo come linters, tests, agile developement.

1.1.6 RedHat

Riassunto

Il seminario, presentato da Mario Fusco di RedHat, introduce il mondo dell'open source e delle sue dinamiche. Vengono elencati i vantaggi nell'open source quali la condivisione di conoscenza, il controllo del codice da molti e la community. Vengono discusse le licenze, divise nelle due categorie Copyleft e non-Copyleft. La prima forza la distribuzione del codice delle applicazioni modificate dall'originale, la seconda non obbliga nessuna restrizione sull'utilizzo del codice. Il seminario si conclude mostrando come poter contribuire ad un progetto open source, principalmente guardando la sezione "Issues" su github.

Cosa abbiamo imparato

L'open source è un ottimo sistema per lo svuiluppo di software sotto tanti aspetti.

1.1.7 Microsoft

Riassunto

Il seminario viene presentato dal Senior Software Engineer Diego Colombo. Comincia presentandoci il concetto di testing, i suoi utilizzi e i vari tipi, e usa come esempio particolare l'acceptance testing e il behaviour driven development(bdd). Successivamente ci spiega perchè è importante testare e perchè sono fondamentali degli health check all'interno di un'architettura a microservizi. Dopodichè dimostra i vari tipi di tool per il testing - dal Unit Test framework ai test runner continui, per poi dirci che i test vengono idealmente svolti in diversi contesti e diverse fasi: Nella macchina di sviluppo, nell'environment di build, in production e perfino nel computer dell'utente finale. Ci mostra degli esempi di strumenti di testing scrivendo il codice in tempo reale:

- Mocha, framework di test e Chai, framework di asserzioni
- Wallaby, un test runner continuo e con supporto di live comment
- Playwright, che simula un browser per testare le funzionalità di ui e permette di fare mocking

Infine ci spiega la filosofia del Test Driven Development, dove i test vanno scritti ancora prima del codice.

Cosa abbiamo imparato

Questo seminario ci ha fatto capire l'importanza dei test e il modo in cui la loro presenza possa facilitare lo sviluppo. Inoltre il test driven development è un approccio interressante che sicuro adotteremo in futuri progetti.

Il nostro approccio

Per il testing delle nostre API abbiamo usato Jest, molto simile a Mocha, e Supertest, ottima per testare le richieste http. Non abbiamo seguito un approccio TDD, però abbiamo messo grande importanza nel testing di ogni API.

1.1.8 Molinari

Riassunto

Il Professore Andrea Molinari presenta il settore "Legacy", quel particolare settore informatico caratterizzato dalla presenza di elementi Hardware, Software, Reti, Dati e Processi "obsoleti", cioè aventi prestazioni e caratteristiche di gran lunga superate e migliorate, ma che sono ancora in uso. Il mondo legacy coinvolge maggiormente il mondo delle grandi società informatiche, società che quindi hanno effettuato un grande investimento in "Mainframes" e "Super sistemi" difficili da rimpiazzare o migliorare con facilità.Stiamo parlando di sistemi aventi in media:

- Server da 240 CPU
- 40 TeraByte di RAM
- Dischi di persistenza dalle dimensioni di PetaBytes

Il professore sottolinea il problema del mondo "Legacy" da un punto di vista della produzione del software, più che del rimpiazzamento hardware: I linguaggi utilizzati per la realizzazione dei programmi diventano velocemente obsoleti, esempio noto "Cobol": un linguaggio ancora oggi ampiamente utilizzato e che sta alla base di quasi tutte le transazioni economiche globali. Non solo, ma cambiare radicalmente tecnologia dall'oggi al domani è altamente rischioso per una qualsiasi azienda, questo per 4 semplici motivi:

- Continuità: "Se funziona, perchè dovrei rimpiazzarla?"
- Ambiguità: "Visto che stiamo rinnovando il sistema, miglioriamo questa funzionalità, aggiungiamo quest'altra"
- Complessità del codice e della documentazione: E' quasi impossibile riscrivere in pochi mesi un software che ha richiesto anni per essere realizzato.
- Recovery: Molte aziende ancora oggi tendono ad avere un cloud ibrido, cioè che hanno un mainframe privato per il backup in locale e anche nel cloud

Qualche soluzione, continua il professore, è possibile: Ultimamente si sta sperimentando molto con delle Intelligenze Artificiali che possano riscrivere il codice obsoleto, velocizzando drasticamente il processo. Un altra soluzione potrebbe

essere l'incapsulamento del vecchio sistema legacy all'interno di un nuovo sistema, sotto forma di componente. L'ultima parte della presentazione è stata dedicata alla gestione organizzativa per un progetto legacy: il metodo AGILE qui fallisce miseramente in quanto:

- Il progetto Legacy è molto grande, solitamente è il risultato di anni ed anni di modifiche e sviluppi
- Il Team legacy è un team complesso, composto da elementi di tutte le generazioni e di tutti i tipi. Molti ruoli diversi, impossibili da gestire con semplicità.
- La comunicazione con il cliente è complicata, se non del tutto inesistente: i progetti Legacy vedono solitamente protagonisti multisocietà formate da aziende di medio-grosso calibro: tante linee di pensiero impossibili da riunire in uno solo e con cui potere interloquire.

Cosa abbiamo imparato

Abbiamo imparato che la produzione di codice e di documentazione non deve essere solo fine a se stessa e alla conclusione del progetto, ma deve anche dare un importante sguardo verso un futuro prossimo. Un futuro in cui potrebbe essere necessario tornare indietro per modificare o riscrivere parte dell'applicazione per portarla nuovamente agli standard del periodo.

Il nostro approccio

La suddivisione della nostra applicazione in più microservizi rende l'intero progetto altamente scalabile e facile da modificare. Le tecnologie utilizzate per la realizzazione del progetto sono inoltre all'avanguardia per il nostro periodo, dunque non dovrebbero esserci problemi in un prossimo futuro.

1.1.9 Marsiglia

Riassunto

Il seminario viene presentato dal Enterprise Architect Gerardo Marsiglia Comincia presentandoci i vari ruoli all'interno di un progetto, quali Specialista, Architetto, Consulente, Ingegnere e Manager. Ci parla del ciclo di vita del software che, soprattutto nei grandi e vari team di sviluppo, è difficile da gestire. E' fondamentale definire standard, organizzare lo sviluppo e utilizzare

tool di automazione. Successivamente ci parla dell'evoluzione delle metodologie di sviluppo, partendo da waterfall, passando a quelle iterative, all'Agile e infine al DevOps. Dopodichè spiega il concetto di Continous Delivery, processo fondamentale del DevOps che si integra con il Continous Integration. Ciò che è più evidente è che il processo di Continous Delivery è spesso automatizzabile, il che velocizza tutto. Successivamente parla di Modernizzazione Applicativa, un processo fondamentale per modernizzare le applicazioni tradizionali e ottenere i vantaggi delle nuove tecnologie. In questo segmento parla anche dell'evoluzione dell'architettura software, partendo dal monolith per arrivare poi all'architettura a microservizi, e i vari livelli di astrazione dal Bare metal ai container. Infine ci mostra l'esempio della domotizzazione di casa sua, dove ha utilizzato una soluzione open source per unire i vari dispositivi di diversi marchi in un' unica applicazione dalla quale controllarli.

Cosa abbiamo imparato

Abbiamo imparato che la gestione del ciclo di vita è veramente importante e non va mai trascurato, ci ha offerto uno scorcio nel mondo del DevOps e in particolare del CI/CD.

Il nostro approccio

Per il progetto non abbiamo seguito una metodologia precisa, tuttavia durante alcune fasi dello sviluppo del codice abbiamo seguito un approccio simile al Continous Integration. Inoltre abbiamo sviluppato un'architettura a microservizi.

1.1.10 APSS & Trentino.ai

Riassunto

Il seminario discute le tencologie e il data flow utilizzati nell'Azienda Provinciale per i Servizi Sanitari (APSS). L'APSS segue un gran numero di dipendenti, professionisti e pazienti su tutto il territorio occupandosi della manutenzione delle attrezzature sanitarie e della gestione dei dati. Infatti vengono utilizzate diverse tecnologie per l'elaborazione e la visualizzazione dei dati come KNIME e Tableau. Inoltre, vi è un interesse nell'adozione di soluzioni basate su AI nel mondo della sanità.

1.2 Organizzazione del lavoro

Il team di sviluppo ha deciso di adottare una struttura organizzativa orizzontale: una struttura organizzativa in cui tutti i membri della squadra di sviluppo possiedono eque responsabilità. Avendo suddiviso il progetto in microservizi è stato possibile gestire la mole di lavoro in maniera asincrona, garantendo una fluidità lavorativa semplice ed efficace: la stesura della documentazione e la scrittura del codice sono state divise in modo equo tra i membri del team di sviluppo in modo da poter lavorare in contemporanea. A tale scopo il lavoro è stato diviso in macro obiettivi (rappresentati dai cinque deliverables) a loro volta suddivisi in obiettivi comuni (individuati dai capitoli di ciascun deliverable) e task individuali (rappresentate dalle contribuzioni di ciascun individuo al fine di completare l'obiettivo). La realizzazione del lavoro è stata suddivisa, specialmente nell'ultimo periodo, in più sprints, dove il team di sviluppo si è impegnato a svolgere le proprie task individuali entro una scadenza ben precisa, concordata tra i membri stessi . Durante questi periodi il team si è riunito con frequenza settimanale per discutere riguardo lo stato del progetto, degli obiettivi in corso e quelli successivi. Per l'organizzazione dello sviluppo il team ha adottato un Workflow basato sul Branching e sulle Pull Requests di GitHub: ogni membro ha lavorato alla propria versione del progetto, inviando le proprie modifiche agli altri membri del gruppo tramite la creazione di "Pull Requests": dopodichè viene effettuato, da parte degli altri membri, un controllo sulla qualità e la correttezza delle modifiche. Il team di sviluppo ha fatto uso dei seguenti strumenti per lo sviluppo dell'applicazione

- Trello
  - Software collaborativo per la gestione e l'organizzazione del lavoro di un gruppo.

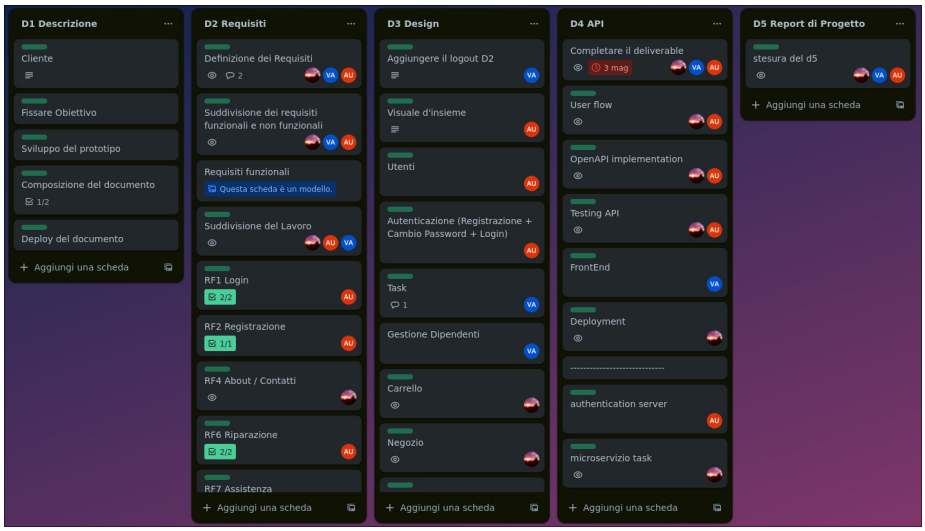

Figure 1.1: Organizzazione del lavoro su Trello

- GitHub
  - Piattaforma per lo sviluppo di software che permette di creare, immagazzinare, gestire e condividere il codice.

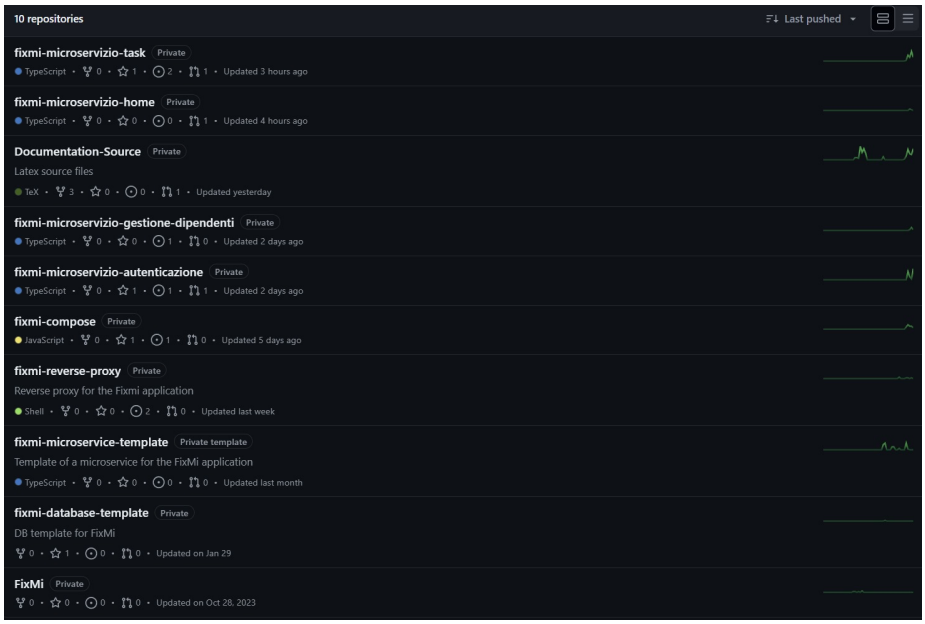

Figure 1.2: Organizzazione del progetto su GitHub

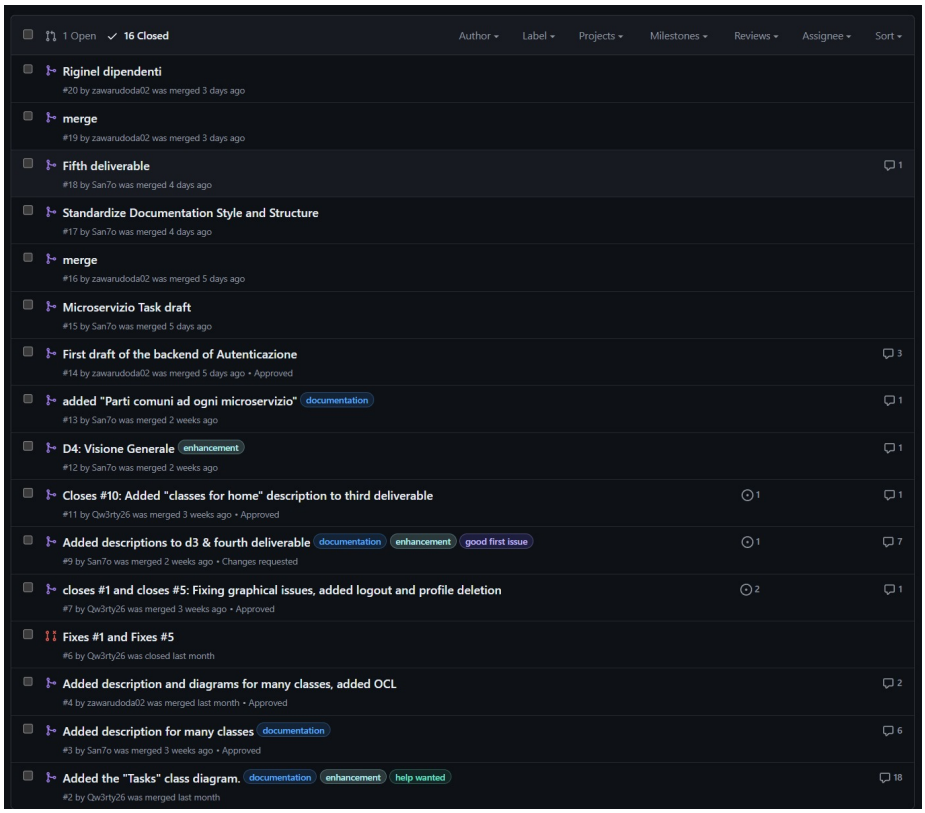

Figure 1.3: Pull Requests della repository "Documentation Source" su GitHub

- Google Drive
  - Software per la memorizzazione e la sincronizzazione dei files
- Draw.io
  - Software per la creazione e lo sviluppo di diagrammi

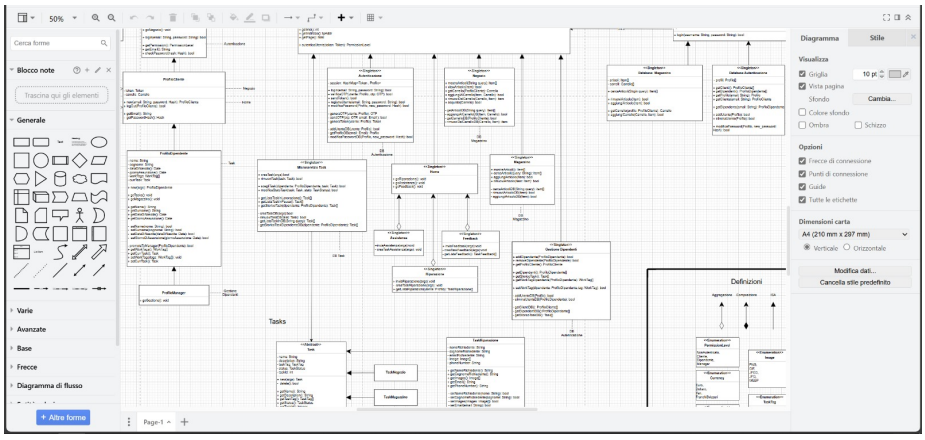

Figure 1.4: Realizzazione del "Diagramma delle Classi" del third-deliverable

- LaTeX
  - Un linguaggio MarkUp specializzato nella realizzazione e scrittura di documenti di alta qualità.

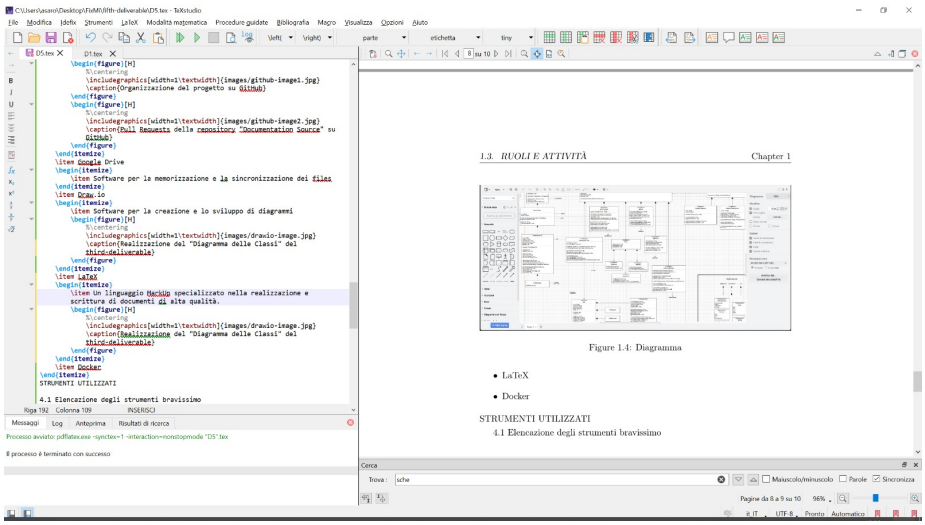

Figure 1.5: Scrittura del documento con LaTeX

- Docker
  - Software per l'esecuzione di processi e programmi in ambiente isolato tramite contenitori minimali e facilmente distribuibili.

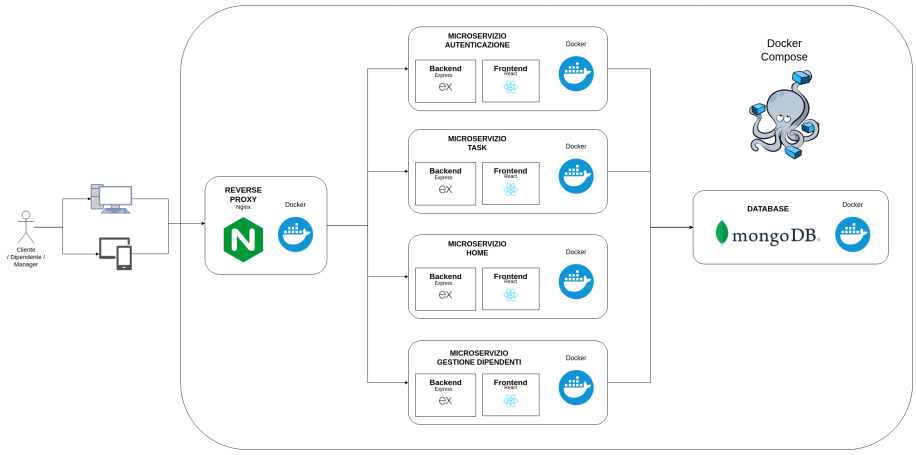

Figure 1.6: Struttura del progetto tramite Docker

1.3 Ruoli e Attività

Riginel Ungureanu

- **Ruolo**: Leader del gruppo
- **Principali Attività**: Il membro ha coordinato il team durante il progetto, ha realizzato in particolar modo il sistema di autenticazione, ha contribuito attivamente alla documentazione e alla progettazione dell'applicazione.

Giovanni Santini

- **Ruolo**: Responsabile Back-End
- **Principali Attività**: Il membro del team ha partecipato laboriosamente all'intero design e sviluppo dell'applicazione. In particolare, ha guidato il team durante la stesura del D3 e D4, si è occupato dell'infrastruttura e implementazione dei microservizi utilizzando tecnologie come Docker e Nginx; ha contribuito attivamente alla documentazione.

Valerio Asaro

- **Ruolo**: Responsabile Front-End e Designer

- **Principali Attività**: Il compito principale è stato la realizzazione della parte Front-End dell'applicazione ma ha anche contribuito alla stesura della documentazione e alla progettazione dell'applicazione.

1.4 Carico e distribuzione del lavoro

Di seguito una tabella rappresentativa della distribuzione del carico lavorativo.

Membri	D1	D2	D3	D4	D5	TOT
Giovanni Santini	39	22	30	108	5	204
Riginel Ungureanu	26	37	21	84	8	176
Valerio Asaro	21	16	17	54	5	113
TOT	86	75	68	246	18	493

1.5 Criticità

Il team di sviluppo ha trovato difficoltà nell'organizzare il proprio tempo, in particolar modo all'inizio del progetto. Il gruppo, non avendo esperienza pregressa nel design e lo sviluppo di software di questa scala, ha riscontrato complicazioni nella scrittura della documentazione e nell'apprendimento delle tecnologie utilizzate per la realizzazione dell'applicazione.

1.6 Autovalutazione

In definitiva il gruppo è molto soddisfatto del lavoro compiuto. Nonostante le criticità precedentemente descritte siamo riusciti a raggiungere i nostri obiettivi. Segue la tabella dell'autovalutazione di ciascun membro del gruppo.

Membri	Voto
Giovanni Santini	30L
Riginel Ungureanu	30
Valerio Asaro	30