Progetto:

FitGYM44

Titolo del documento:

Documento di architettura

Document Info

Doc. Name	*D5-FitGYM44_FinalReport*	Doc. Number	5
Description	Report Finale del progetto: organizzazione del lavoro, ruoli, tempo complessivo e di ciascun membro dedicato al progetto, criticità, autovalutazione		

Indice:

0. Scopo del documento

Il presente documento rappresenta il report finale che illustra come il nostro team ha contribuito allo sviluppo del progetto. Inizialmente, verranno esposti i metodi utilizzati nell'Ingegneria del Software. Seguirà una descrizione dell'organizzazione e della distribuzione dei compiti tra i membri del team, oltre a un riepilogo delle ore di lavoro totali e individuali dedicate ai vari deliverables. Successivamente, verranno evidenziate le criticità e le problematiche affrontate durante il progetto, concludendo con una breve autovalutazione del nostro operato.

1. Approcci all'Ingegneria del Software

1.1 - Seminario BlueSensor: Un Approccio Integrato allo Sviluppo di Programmi AI

Il seminario BlueSensor ha presentato un approccio innovativo allo sviluppo di programmi AI, combinando il metodo a cascata con quello Agile. Le fasi di sviluppo seguono una sequenza ben definita:

1. **Studio di fattibilità preliminare**: In questa fase, si esamina l'idea del progetto e si valutano i requisiti funzionali e non funzionali.
2. **Progettazione software e hardware**: Si crea un progetto architetturale del programma e si definiscono due set di dati completi, uno per il test e uno per l'addestramento.
3. **Sviluppo e test**: Le fasi di sviluppo, addestramento e test seguono un modello agile. Il programma viene suddiviso in moduli che implementano funzionalità di base. Ogni modulo viene progressivamente completato, integrato con gli altri e rilasciato per ottenere feedback dal cliente.
4. **Rilascio finale**: Alla fine di questo sviluppo iterativo, avviene il rilascio definitivo del software, che può richiedere aggiornamenti o integrazioni future.

Questo approccio combina le migliori caratteristiche dei metodi a cascata e agile, garantendo una gestione efficace del progetto.

Importanza della Coesione del Team

Durante il seminario, abbiamo anche esplorato l'importanza di creare un team coeso. La comunicazione e la collaborazione all'interno del team sono fondamentali per migliorare la produttività. Tuttavia, abbiamo notato che le attività di "Team Building" possono creare sottogruppi coesi ma isolati. Inoltre, è cruciale avere nel team qualcuno con conoscenze specifiche dell'ambito in cui il prodotto software verrà utilizzato, poiché spesso bisogna sviluppare prodotti per contesti sconosciuti.

In sintesi, il seminario ha evidenziato l'importanza di un approccio metodologico integrato e di un team coeso per lo sviluppo di prodotti software di successo.

1.2 – Seminario sul Metodo Kanban e WIP Limit

Il Kanban è un approccio pratico per gestire il flusso di lavoro e prevenire la sovrapproduzione. Durante il seminario, abbiamo messo in pratica il Kanban attraverso una simulazione, riproducendo eventi reali tramite astrazioni. I bigliettini utilizzati nel Kanban sono uno strumento prezioso. Non solo mostrano il flusso di lavoro, ma consentono anche di monitorare i tempi di sviluppo di ciascuna attività. È importante mantenere la lavagna aggiornata regolarmente per evitare problemi. Tuttavia, la mancanza di una base temporale specifica per le fasi del Kanban può portare a sfide nello sviluppo.

Durante la simulazione, abbiamo notato come avere un WIP limit possa diminuire notevolmente il lead time, permettendo un approccio incrementale con continui feedback. Questi due aspetti portano a una maggiore attenzione alle esigenze del cliente, riducendo il rischio di errori nei requisiti, spesso costosi. Inoltre, il veloce raggiungimento del MVP e il continuo miglioramento del prodotto aumentano la soddisfazione del cliente e la competitività sul mercato. Se, ad esempio, notiamo un'opportunità di mercato relativa al nostro prodotto, possiamo immediatamente sfruttarla nel prossimo incremento.

Infine, la simulazione ci ha permesso di capire come un sistema che lavora al 100% della sua capacità massima di lavoro si comporti in modo imprevedibile. Questo avviene perché è incapace di gestire gli imprevisti, avendo già tutte le risorse allocate. Questo ragionamento non vale solo per la realizzazione di un software, ma anche per l'organizzazione del lavoro di un gruppo o di un singolo individuo.

1.3 – IBM

Durante il seminario su IBM Cloud, abbiamo esplorato la piattaforma distribuita IBM Cloud, che offre servizi di potenza computazionale e tecnologia software a consumo. Questa piattaforma consente di creare rapidamente server, macchine virtuali, database e altro ancora, con costi accessibili e senza doversi preoccupare dei dettagli tecnici. Un aspetto particolarmente utile è la possibilità di rendere facilmente, velocemente e in modo sicuro un'applicazione disponibile su Internet per tutti.

Inoltre, è stato presentato IBM Cloud Satellite, un servizio importante che gestisce automaticamente tutti i servizi IBM su macchine locali definite "satellite". Queste macchine sono collegate al Cloud IBM tramite un collegamento satellitare. Durante il seminario, è stata sottolineata l'importanza di avere un'architettura ben progettata, con un focus sulla sicurezza e illustrando esempi pratici. È stato evidenziato anche quanto sia cruciale l'adattabilità dei servizi offerti, che devono rimanere all'avanguardia rispetto alle nuove tecnologie.

D'altra parte, abbiamo esaminato i vantaggi di un'infrastruttura Cloud. Abbiamo notato come un servizio cloud sia estremamente flessibile alle richieste del cliente per diverse risorse, sia hardware (non sempre ho bisogno della stessa quantità) che software (potrei aver bisogno di diversi software, ed è comodo averli tutti in un unico

punto) nel tempo. Tuttavia, abbiamo anche riconosciuto il grande costo di questa soluzione, che potrebbe risultare insostenibile per le aziende con attività a basso valore aggiunto. Inoltre, siamo rimasti sorpresi dalla quantità di personalizzazione possibile in ogni aspetto.

In conclusione, abbiamo riflettuto sull'importanza di considerare quanto evolverà l'ambito quando si sceglie tra "buy", "make" o "outsource". Questo ci ha portato a riconoscere che un aspetto fondamentale è valutare quanto l'evoluzione sarà prevista nel contesto specifico.

1.4 – Meta

Durante il seminario META, abbiamo esplorato gli strumenti e le figure essenziali per la realizzazione di progetti software. L'accento è stato posto sulla comunicazione efficace tra team diversi e tra i membri dello stesso gruppo di lavoro, riconoscendo quanto questa comunicazione possa portare a risultati significativi in tempi brevi. Per mantenere alta la produttività nei vari gruppi, le persone hanno la libertà di esprimersi liberamente e, se necessario, di cambiare team.

META non adotta metodologie specifiche di ingegneria del software; i team leader di ogni gruppo seguono principalmente un approccio agile. L'obiettivo finale viene suddiviso in sottoprocessi assegnati alle persone in base alle loro competenze. I progressi vengono monitorati attraverso diversi strumenti, tra cui il più importante all'interno di META è Asana. Inoltre, sono presenti strategie basate su fogli di calcolo ("Spread Sheets"), dove i vari compiti vengono organizzati in formato tabellare, rappresentando un importante strumento di monitoraggio per il progresso dei progetti.

Questo seminario ci ha fatto riflettere su come la semplicità nella comunicazione tra colleghi e la capacità di adattamento dei singoli siano più importanti della conoscenza di molti linguaggi di programmazione o di avere un background specifico.

1.5 – U-Hopper

Il seminario di U-Hopper ha evidenziato l'importanza di una gestione efficace dei big data. Nei grandi servizi cloud, uno degli stadi finali è l'ingestione, che utilizza un sistema a code seguito dall'elaborazione dei dati tramite batch processing o stream

processing. Questo consente di elaborare i dati in blocchi o in tempo reale. I dati vengono archiviati in database in memoria centrale per garantire tempi di accesso rapidi. È fondamentale che gli strumenti utilizzati registrino i log per valutare le prestazioni e facilitare il debugging.

Durante lo sviluppo, U-Hopper adotta la metodologia Agile con sprint di circa due settimane. Le milestone vengono suddivise in issue, e per ciascuna issue si crea un branch su Git per lo sviluppo. La code review viene avviata al momento del merge. Inoltre, il codice scritto viene testato per garantire una copertura del 70%. Questo seminario ha evidenziato la necessità di un'architettura robusta con bassi livelli di accoppiamento per una gestione efficace dei big data, rendendo il sistema altamente scalabile. L'utilizzo di Git è fondamentale per garantire uno sviluppo isolato e parallelo, mantenendo elevati standard di qualità.

Ascoltando il CEO di U-Hopper, abbiamo avuto l'esempio di come una piccola azienda possa combinare innovazione e metodologie agili, al fine di rimanere allo stato dell'arte in un ambito in continua evoluzione come l'IA. La scelta di una metodologia agile è particolarmente azzeccata nel loro ambito, considerando che il loro mercato è in costante mutamento. Inoltre, è probabile che il cliente medio non abbia un'idea chiara di ciò che vuole, soprattutto perché solo negli ultimi anni si sono avute le prime applicazioni business dell'IA. Questa difficoltà si traduce nella sfida di definire chiaramente le esigenze del cliente e, di conseguenza, i requisiti del prodotto desiderato.

1.6 – Seminario RedHat: Vantaggi dell'Open Source

Il Seminario RedHat è stato molto interessante, anche se non si è concentrato esclusivamente su metodi di ingegneria del software o gestione. Invece, l'attenzione è stata rivolta alla distribuzione del codice e, in particolare, all'Open Source.

Dopo una breve presentazione personale, il relatore, Mario Fusco, ha illustrato i vantaggi dell'Open Source:

1. **Condivisione di informazioni**: Scrivere e visualizzare il codice sono i migliori modi per apprendere. L'Open Source favorisce la diffusione delle conoscenze tra gli sviluppatori.

2. **Supervisione da parte di altri sviluppatori**: Poiché il codice è visibile a tutti, chiunque può contribuire e individuare eventuali criticità. Questo processo di revisione continua migliora la qualità del software.
3. **Meritocrazia e visibilità**: Le buone idee e il codice parlano per il loro creatore. L'Open Source offre un terreno di gioco equo in cui il valore viene riconosciuto in base alle competenze e ai contributi.
4. **Senso di comunità**: Partecipare a progetti Open Source crea un forte legame con altri sviluppatori. La collaborazione e il supporto reciproco sono fondamentali.

Durante il seminario, sono state spiegate le principali categorie di licenze Open Source e come partecipare a un progetto di questo tipo. Inoltre, sono state evidenziate alcune pratiche per mantenere un progetto Open Source sano.

Riutilizzo del Codice Open Source

Quando un ingegnere del software deve decidere come produrre il software di un componente comune, spesso sceglie il riutilizzo del codice già esistente, spesso open source. Questo riduce notevolmente i costi, poiché si utilizzano componenti già pronti anziché crearli da zero, e migliora la qualità del codice, dato che molte persone hanno analizzato il codice alla ricerca di problemi.

Tuttavia, è importante fare attenzione alle licenze. L'ingegnere deve evitare di utilizzare una licenza non-copyleft, altrimenti sarà obbligato a rendere anche il suo codice open source. Inoltre, c'è il rischio di un lock-in per quanto riguarda la porzione di codice utilizzata, il che potrebbe creare una posizione di debolezza nei confronti dei creatori del codice in caso di problemi.

In conclusione, l'utilizzo del codice open source è spesso una buona scelta, a patto di stare attenti a eventuali copyleft indesiderati o lock-in eccessivi.

1.7 - Microsoft

Durante il seminario, abbiamo esplorato approfonditamente il concetto di testing. Siamo stati introdotti a vari tipi di test, dai più basilari come quelli unitari ai più complessi come quelli funzionali, di integrazione e di accettazione. Un aspetto cruciale è stata l'enfasi sul test driven design, una pratica essenziale per scrivere buon codice. L'ingegnere di Microsoft ci ha mostrato quanto sia cruciale, ma anche

complesso, testare correttamente una grande code base. La scrittura del codice per il testing risulta più semplice rispetto alla scelta della modalità di testing più adatta a seconda del contesto. Bisogna tenere conto delle esigenze dello sviluppo senza perdere di vista le conferme cercate dagli stakeholders.

Crediamo che quando si sviluppa, sia fondamentale avere un team verticalizzato nel testing. Questo team può guidare gli sviluppatori nella realizzazione del prodotto attraverso test case definiti a priori, coinvolgendo anche gli stakeholder più tecnici.

Inoltre, durante il seminario di Diego Colombo, sono stati presentati vari strumenti per effettuare test di diverso tipo, tra cui framework per i test unitari, strumenti di mocking, code coverage, DSL per il BDD (Behavior-Driven Development) e automazione dei test UI. È stato sottolineato quanto sia critico rispettare la regola di mantenere sempre "act" e "assert" allo stesso livello. L'approccio "TBD" (Test Before Development), che integra il testing fin dall'inizio, è stato raccomandato per ridurre al minimo il debito tecnico.

In sintesi, il seminario è stato un'ampia e approfondita esplorazione del testing, fornendoci una visione lungimirante sugli strumenti utilizzati.

1.8 - Molinari

Il seminario di Molinari sui sistemi legacy è stato molto interessante. I sistemi legacy, essendo obsoleti nelle loro componenti o parti, spesso presentano caratteristiche come mainframe e hardware datati, personale non aggiornato e una UI scadente. Tuttavia, continuano a esistere per vari motivi, tra cui problemi di investimento, robustezza e paura delle tecnologie moderne.

Il "tech debt" è un problema noto nel mondo del software. Si verifica quando un software datato viene aggiornato invece di essere sostituito. Questo può portare a inibizioni nei nuovi sviluppi, insoddisfazione del cliente, costi aggiuntivi e rischi per la sicurezza.

Per portare i sistemi legacy nel mondo moderno, esistono quattro metodi:

1. Dismissione: Il sistema viene smantellato, le procedure sostituite da nuovo software e i dati migrati.
2. Migrazione: I dati e il software vengono modernizzati e il sistema legacy viene gradualmente dismesso.

3. Interazione: Il sistema legacy continua a esistere e interagisce con il nuovo sistema.
4. Inclusione: Il sistema legacy diventa parte attiva del nuovo sistema tramite emulazione.

Gestire un progetto legacy può essere complesso, specialmente quando coinvolge grandi dimensioni e ruoli difficili da definire. In questi casi, metodologie di gestione più predittive, meno focalizzate sull'approccio al cliente finale e più incentrate sui processi, software funzionante, contratti e piani precisi, possono essere più efficaci rispetto all'approccio Agile.

D'altra parte, nella realtà, gran parte dei sistemi ancora in uso, anche in ambiti fortemente critici come la finanza, fanno parte del mondo del "legacy". Questi sistemi sono obsoleti e continuano ad essere utilizzati poiché chi li usa o non intende o non può rimpiazzarli. Per quanto riguarda l'ingegneria del software, bisogna quindi tenere conto che tali sistemi sono estremamente difficili da mantenere o sostituire, in quanto richiedono personale specializzato molto raro e costoso. Inoltre, data la criticità del contesto, è fondamentale che restino sempre operativi. Di conseguenza, una continua disponibilità di manutenzione preventiva e reattiva è essenziale.

1.9 – Marsiglia

Durante il seminario sulla vita dell'applicazione, il suo mantenimento e la modernizzazione, sono stati esplorati vari aspetti dello sviluppo delle applicazioni. Un focus importante è stato sulla gestione del team di sviluppo e sull'importanza di seguire standard e regole ben definite per evitare il caos.

L'evoluzione degli standard per lo sviluppo, dai metodi tradizionali all'approccio Agile, è stata discussa. Inoltre, è stato introdotto il metodo DevOps e spiegate le pratiche di Continuous Delivery, che consentono di sviluppare e testare continuamente parti del progetto prima di consegnarle.

Sono stati presentati anche strumenti utili per lo sviluppo. Infine, si è parlato della modernizzazione delle applicazioni, un processo impegnativo che richiede la comprensione completa del funzionamento dell'applicazione da modernizzare per integrare nuove tecnologie.

Un punto interessante è che lo sviluppo delle applicazioni sta passando dai sistemi monolitici a componenti collegati tra loro (microservizi), consentendo risultati continui e costanti che possono essere riutilizzati.

Standard di Qualità e Sicurezza

Parlando con un "Enterprise architect", una figura esperta nello sviluppo del software, abbiamo notato come gli standard di qualità dei prodotti siano notevolmente migliorati nel tempo. Ad esempio, nel campo della disponibilità, si parla di "x nines", che rappresenta la percentuale di "availability" basata sulla prima cifra.

Per ottenere questi risultati, è fondamentale migliorare radicalmente il processo di sviluppo del software. L'approccio "DevOps", che promuove la stretta collaborazione tra team di sviluppo e infrastrutturale per costruire un prodotto robusto e resistente agli errori, è stato enfatizzato. Tuttavia, dal nostro punto di vista, la sicurezza dovrebbe ricevere maggiore attenzione. Riteniamo fondamentale collaborare con un team specializzato in sicurezza e condurre "penetration testing" sul codice prodotto. Altrimenti, gli "x nines" rischiano di basarsi troppo sulla benevolenza o l'indifferenza dei malintenzionati.

1.10 - APSS

Il seminario sul dipartimento di tecnologie dell'Azienda Provinciale per i Servizi Sanitari ha affrontato alcuni principi fondamentali dell'azienda, tra cui la continuità delle cure, l'equità dei servizi e l'adozione delle nuove tecnologie. Il dipartimento di tecnologie ha il compito di definire, sviluppare e mantenere le tecnologie dell'informazione, con l'obiettivo di massimizzare i benefici per la comunità.

Un aspetto cruciale è la necessità di competenze trasversali e un'organizzazione del lavoro efficiente per garantire lo sviluppo efficace del software. Il dipartimento gestisce una vasta gamma di servizi essenziali per l'APSS. Durante il seminario, è stato sottolineato l'importanza dei dati, considerati una risorsa trasversale essenziale per il funzionamento dell'ente.

Inoltre, l'applicazione dell'intelligenza artificiale è emersa come un'area promettente di sviluppo, con applicazioni sia gestionali che cliniche. Queste tecnologie mirano a ottimizzare le operazioni e migliorare le diagnosi attraverso l'analisi automatica dei

dati. Nonostante gli ostacoli finanziari e burocratici, i metodi dell'APSS rappresentano un importante passo avanti nel miglioramento dei servizi sanitari offerti ai cittadini.

Innovazioni Tecnologiche nel Settore Sanitario

Durante l'incontro, abbiamo potuto vedere come il settore pubblico della sanità sta mutando con le innovazioni tecnologiche del mondo del software. Essendo questo un settore fortemente critico e regolamentato, sono molte le considerazioni che un ingegnere del software deve fare.

Prima di tutto, deve tenere conto che questi sistemi hanno un lungo tempo di vita e passeranno attraverso diverse normative, di conseguenza la manutenibilità è fondamentale. In secondo luogo, deve considerare la sensibilità dei dati utilizzati. È quindi fondamentale garantire la sicurezza sia a livello software che fisico del sistema. La creazione di un polo nazionale in cloud è un'ottima idea per affrontare queste sfide.

In terzo luogo, bisogna tenere conto della necessità di avere un servizio robusto ai fallimenti e distribuito che raggiunga l'intero territorio in ogni momento. Tutti questi aspetti rendono fondamentale una progettazione architetturale impeccabile.

Infine, abbiamo parlato di come le innovazioni tecnologiche più recenti potranno cambiare il mondo della sanità. Dal nostro punto di vista, un ingegnere del software deve tenerne conto e predisporre l'architettura in modo tale da integrarle senza troppi problemi.

2. Organizzazione del lavoro

Il lavoro è stato organizzato centrato sulle competenze dei membri del gruppo. Io quindi, avendo letto più di qualche libro sul Project Management e sull'Agile, ho assunto la guida delle risorse umane. Di conseguenza, lo sviluppo è stato delegato quasi completamente agli altri due membri del gruppo mentre, io, mi sono occupato della scrittura della maggior parte dei contenuti presenti nei deliverables e alla loro revisione. Ho ritenuto opportuno dividere il carico di diagrammi UML equamente tra tutti i membri del gruppo.
Abbiamo quindi deciso di utilizzare una metodologia Agile con meeting con cadenza

settimanale per monitorare continuamente lo stato di avanzamento del processo produttivo, sia per quanto riguarda i deliverables, sia per quanto riguarda lo sviluppo del codice, dando precedenza alla realizzazione di un prodotto che fosse utilizzabile -anche se con poche funzionalità- fin da subito.
Inoltre, prediligendo un approccio di questo tipo, in caso di riscontro di problematiche si è sempre potuto tornare indietro e andare a correggere ciò che di sbagliato era stato fatto.
I tool di utilizzo sono stati molteplici:

- Trello: Per la progettazione dell'app, il brainstorming, le to-do list e i backlog
- Google Docs: Per la scrittura dei nostri documenti
- Draw.IO: Per la creazione di diagrammi UML
- Visual Studio Code: Come Editor Di Testo/IDE per la programmazione
- Figma: Per la realizzazione dei MockUp presenti nel D1.
- GitHub: Per creare la nostra repository e mantenere aggiornato lo stato del processo.
- LiveShare: Un'estensione di Visual Studio Code che consente di sviluppare codice in contemporanea tra più persone.

3. Ruoli e attività

Componente del team	Ruolo	Principali attività
Luca Buonocore	Capo progetto, analista requisiti funzionali e non con relative specifiche, coordinatore risorse umane e supervisore progetto	Il ruolo principale è stata la gestione del progetto, la compilazione dei campi e la risoluzione dei problemi al loro insorgere. Ho svolto la maggior parte del D1, del D2 e del D3. Ho supervisionato personalmente tutti i deliverable.
Pietro Gasparini	Supporto ideazione e descrizione dei componenti del progetto, responsabile sviluppo applicazione	Ha collaborato nella ideazione del progetto, nella scrittura delle descrizioni nei deliverables D1, D2 e D3. Attivo nella parte di sviluppo e realizzazione dell'applicazione nella parte del back-end

Jasnoor Singh	Supporto analisi requisiti funzionali e non, sviluppatore database e web app, addetto al testing	Ha contribuito parzialmente ai deliverable D1, D2, D3 e attivamente al D4 nello sviluppo del back-end e parte del front-end

4. Carico e distribuzione del lavoro

Nome	D1	D2	D3	D4	D5	Totale
Luca Buonocore	~50	~115	~105	~20	~3	~300
Pietro Gasparini	~20	~30	~70	~150	~1	~270
Jasnoor Singh	~20	~25	~50	~140	~1	~235
Totale	~90	~170	~225	~310	~5	~805

D1:

- Luca Buonocore:
  - Partecipazione attiva al brainstorming per i requisiti funzionali e non funzionali
  - Scrittura e definizione nel dettaglio dei requisiti e degli obiettivi
  - Realizzazione dei mockup tramite Figma
  - Revisione finale e verifica della coerenza del documento
- Pietro Gasparini:
  - Supporto ideativo durante la progettazione
  - Scrittura della descrizione dei requisiti funzionali
- Jasnoor Singh:
  - Supporto nella fase di identificazione dei requisiti
  - Scrittura di parte delle descrizioni dei requisiti funzionali

D2:

- Luca Buonocore:

    - Scrittura delle descrizioni degli Use Case relativi ai RF
    - Realizzazione dei diagrammi UML degli Use Case e degli Activity Diagram
    - Realizzazione delle tabelle relative ai RNF
    - Realizzazione del diagramma dei componenti
    - Revisione finale e verifica della coerenza del documento
- Pietro Gasparini:
    - Scrittura delle descrizioni degli Use Case
    - Creazione dei diagrammi degli Use Case e Activity Diagram
    - Scrittura della descrizione e scopo dei componenti
    - Creazione diagramma dei componenti
- Jasnoor Singh
    - Scrittura descrizioni di parte degli Use Case
    - Creazione dei diagrammi di alcuni Use Case e Activity Diagram
    - Scrittura della descrizione e scopo di parte dei componenti
    - Supporto nella creazione diagramma dei componenti

D3:

- Luca Buonocore:
    - Scrittura delle descrizioni delle classi relative al Class Diagram
    - Realizzazione del Class Diagram
    - Scrittura degli OCL
    - Revisione finale e verifica della coerenza del documento
- Pietro Gasparini
    - Scrittura delle descrizioni delle classi
    - Creazione dei diagrammi e modelli delle classi nel Class Diagram
    - Scrittura degli OCL
- Jasnoor Singh
    - Scrittura delle descrizioni delle classi assegnate
    - Creazione dei diagrammi delle classi assegnate

D4:

- Luca Buonocore:
    - Scrittura delle descrizioni degli UserFlow
    - Realizzazione dei diagrammi UML degli UserFlow

    - Realizzazione delle descrizioni relative al diagramma delle risorse
    - Revisione finale e verifica della coerenza del documento
- Pietro Gasparini
    - Sviluppo BackEnd
    - Sviluppo API
    - Realizzazione di API documentation
    -
- Jasnoor Singh
    - sviluppo e descrizione UserFlow
    - Sviluppo BackEnd
    - Sviluppo API
    - Testing

D5:

- Luca Buonocore:
    - Riassunzione dei seminari
    - Creazione delle tabelle
    - Descrizione della modalità di lavoro
- Pietro Gasparini:
    - Revisione del documento
- Jasnoor Singh:
    - Revisione del documento

5. Criticità

Il principale ostacolo al completamento del progetto è stato la carenza di esempi esaustivi e chiari per i vari deliverable. Questa mancanza di chiarezza ha portato il team a fraintendere i requisiti. Infine, l'evento più arduo da affrontare è stato la difficoltà nell'apprendimento di nuovi linguaggi richiesti per affrontare lo sviluppo.

6. Autovalutazione

Io e i miei compagni riteniamo di aver lavorato molto a questo progetto, gli sforzi richiesti sono stati notevoli soprattutto per quanto riguarda la fase di sviluppo.

Nonostante ciò riteniamo di aver creato un bel progetto.
La nostra autovalutazione è la seguente:

Componente	Voto
Luca Buonocore	30L
Pietro Gasparini	30L
Jasnoor Singh	30L