D5: Report Finale

G06

June 16, 2024

Contents

1 Scopo del Documento

Questo documento è il report finale del progetto **EasyLib** del gruppo **G06** del corso di **Ingegneria del Software** a.a. 2023-24. Nel Presente verranno descritti i seminari visti durante il corso, e come sono stati applicati all'interno del nostro progetto; e poi il gruppo valuterà il lavoro svolto e le difficoltà riscontrate.

2 Seminari

2.1 Seminario Bluetensor

Riassunto:
La tematica trattata durante il seminario riguarda l'utilizzo e la creazione di progetti di intelligenza artificiale nel mondo dell'ingegneria del software. L'azienda Bluetensor, che è un azienda del territorio fondata da un ex studente dell'università di Trento, si occupa proprio di questo; in particolare crea dei prodotti su misura del cliente basati sull'AI che aiutano le aziende tramite:

- Supporto decisionale.
- Esperienza utente.
- Ottimizzazione della produzione aziendale.
- Gestione del Know How aziendale.

Oltre a questo è anche impegnata nella creazione di tool per il lavoro con l'AI, come ad esempio la creazione di assistenti virtuali per i siti aziendali; e nell'ambito dell'analisi predittiva che sfrutta l'intelligenza artificiale e i dati passati di un azienda per cercare di predire l'andamento di mercato futuro ricavano numeri importanti allo sviluppo aziendale. Il seminario ci dà indicazioni su alcune best practice riguardanti lo sviluppo software AI nell'ingegneria del software. I primi aspetti necessari da considerare sono:

1. Disponibilità e qualità dei dati.
2. Staff multidisciplinare, e competenze nell'ambito del progetto.
3. Aspetti etici e di privacy.
4. Integrazione di flussi già esistenti.

Dopo aver considerato questi aspetti si può passare al descrive il flusso di un progetto AI, che solitamente viene descritto come diviso in 5 parti principali (più o meno le stesse dei nostri Deliverables) e sempre incentrate attorno ai 3 vincoli fondamentali per la gestione del progetto: **Tempo**, **Risorse** sia Hardware che software e lo **scopo** ultimo del progetto.

Le parti sono divise in:

1. Prima analisi del progetto riguardo alla sua fattibilità, raggruppamento delle informazioni necessarie e dei vari requisiti funzionali e non.
2. Una prima stesura di come vengono implementati i requisiti e incontri con il cliente per definire le funzionalità (Story mapping).
3. Si passa poi a una fase di creazione e selezione dei dati da utilizzare per gli algoritmi, le loro caratteristiche (attributi). Questo lavoro ci permette di ottenere in output un data set di casistiche che servirà per creare tutte le ipotesi possibili da passare alla macchina per fargli capire come agire.
4. Successivamente viene la parte di sviluppo del codice con annesso testing e deployment del prodotto finale.
5. L'ultima fase è quella di rilascio finale del prodotto con raccolta dei feedback e miglioramenti rispetto alle versioni precedenti, in questa fase il cliente accetta il prodotto finale.

Tutte le varie fasi sono in continua modifica ed è per questo che il metodo agile è quello più consono per questa tipologia di progetti perché permette continue durante tutto il corso del lavoro. Molti strumenti infine, sono utili all'intero corso del progetto quali UML diagrams per la schematizzazione dei requisiti, oltre che a tutte le varie tecnologie per gestire tutte gli step necessari al completamento del progetto: dal versioning con Git, passando per lo sviluppo con i vari linguaggi (python, java, SQL) fino ad arrivare al deployment.

2.2 Seminario IBM

Riassunto:
Il seminario si basa sull'uso della tecnologia del cloud computing e in particolare dell'uso che ne è stato fatto all'interno dell'azienda IBM tramite il loro modello Cloud IBM, che permette l'utilizzo di tantissime risorse messe a disposizione per gli sviluppatori tramite i software cloud. Cloud IBM è pensato come un catalogo di strumenti per sviluppatori che spaziano su tutto l'ambito dell'ingegneria del software: dalla messa a disposizione di Database personalizzabili fino all'utilizzo di strumenti come per il Deployment, il testing e la creazione di applicazioni. Il catalogo degli strumenti viene organizzato secondo due caratteristiche principali:
la **potenza computazionale**, incentrata su modelli di business e manipolabile a seconda del grado di controllo nell'implementazione che si vuole avere.
La **Tecnologia software** necessaria agli scopi, e creata in base alle nostre esigenze tramite dei modificatori appositi. Il tutto garantisce di avere un prodotto sicuro, attinente al modello di business prescelto e che sarà sempre funzionante e operativo quando ne avremo bisogno. La certezza di avere la miglior tecnologia di sistemi software a livello industriale protetta e sicura è forse il motivo più importante dell'esistenza del Cloud IBM e il motivo per cui i clienti sono spinti

ad acquistare da quel catalogo. Gli esempi di utilizzo sono molteplici e comprendono interamente il settore dell'ingegneria del software ogni aspetto. Alcuni esempi possono essere la creazione di applicazioni direttamente all'interno del Cloud che possono essere sfruttate solamente in caso di bisogno, e quindi il loro costo è basato sul tempo di impiego di esse. Un altro esempio è legato alla sicurezza del prodotto che noi vogliamo sviluppare, infatti usando infrastrutture apposite del catalogo riusciremo a creare un prodotto che sarà molto più sicuro e protetto da attacchi alle informazioni in essi contenuti e quindi renderà il nostro progetto più affidabile e interessante agli occhi di un possibile utilizzatore.

2.3 Seminario Meta

Riassunto:
Il seminario è incentrato sul come viene sviluppato software di qualità all'interno di una multinazionale leader del settore come Meta. Innanzitutto è importante la divisione dei ruoli all'interno di un team; che vengono creati e gestiti dai responsabili di progetto. Ogni membro ha le sue **responsabilità** che riguardano: il coding, spesso delegato agli ingegneri junior; il design dell'architettura, che viene gestito dagli ingegneri più esperti, ed infine la gestione del progetto che è curata dai membri più esperti del gruppo di ricerca e che quindi, possono gestire un volume maggiore di responsabilità nel progetto. Il rapporto tra i membri del team per ricevere Feedback è molto importante, infatti vengono organizzate sessioni di team building per capire cosa si sta facendo in maniera corretta e cosa no; inoltre per gli ingegneri più giovani sono presenti programmi di coaching, fatte da impiegati esperti, per dare una mano hai neo assunti raccontando le esperienze di dipendenti più anziani e cercare di favorire lo sviluppo. Al di fuori delle responsabilità dei dipendenti, vengono anche considerarti gli **aspetti tecnici** all'interno di una grande multinazionale: come ad esempio gli strumenti messi a disposizione di ogni team, che sono spesso creati su misura dal team di Meta per favorire l'adattabilità al tipo di lavoro svolto all'interno dell'azienda. Spesso infatti i software sono versioni particolari di modelli open source accessibili da chiunque come ad esempio version control che serve a gestire meglio il codice in un team ed è una versione di Mercurial sviluppata ad hoc da Meta. Anche i linguaggi utilizzati spesso hanno delle estensioni apposite come ad esempio React per JS, introdotta da Meta e ora largamente utilizzata su scala globale. Tutte queste personalizzazioni rendono più agevole il lavoro all'interno di Meta perché studiate appositamente ma, di contro, renderebbero difficile il passaggio ad un altra azienda per un dipendete perché si troverebbe a lavorare in un ambiente esterno a quello creato all'interno di Meta.

2.4 Seminario U-Hopper

Riassunto:
Il seminario ci parla di come un'azienda di medie dimensioni, del territorio lavora nel mercato dell'ingegneria del software. L'azienda, U-Hopper, si occupa di servizi presso altre aziende per aiutarle a estrarre informazioni relative al

modello di business necessario e ricavare dati importanti per lo sviluppo da esse. La loro azienda conta un gruppo ristretto di dipendenti, ma nonostante questo lavora con aziende affermate sia in Italia che all'estero. Il loro lavoro è svolto con tecnologie open-source e alcuni dei loro risultati possono essere trovati su cloud pubblici. L'azienda lavora per le compagnie utilizzando grandi quantità di dati, strutturati e non che vanno elaborati in 2 modi possibili:

- Batch Processing: In cui tutti i dati vengono elaborati allo stesso tempo.
- Stream Processing: dove i dati vengono elaborati in tempo reale uno ad uno.

Queste elaborazioni di grandi quantità di dati vengono svolte tramite dei motori, che possono gestire entrambe le tipologie di elaborazioni, un esempio usato all'interno di U-Hopper è Apache Spark. I dati successivamente devono essere salvati in alcuni Database, che possono essere;

- In memory Database: I dati vengono salvati in memoria
- Storage: archiviazione su Database relazionale (MySQL) e non (MongoDB)

La programmazione avviene al 90% tramite Python, usato per la programmazione ad oggetti, la restante parte avviene con Scala, un linguaggio funzionale, e Go, usato per la programmazione in parallelo. Il linguaggio viene mediato dai motori di processing visti in precedenza.

il **Deployment** avviene tramite delle infrastrutture come ad esempio Docker, che utilizza dei contenitori per poter testare molteplici Software sulla propria macchina. Viene fatto prima di tutto un deployment su un lavoro intermedio (staging) cosi da poter svolgere test su di esso, e successivamente passati i test si fa il deployment della versione che andrà in produzione.

Tutto il processo di sviluppo viene fatto utilizzando la metodologia Agile con brevi sprint di codice e testing. Tutto il processo viene fatto utilizzando **Git** e tutte le sue funzionalità che rendono il processo più sicuro ed efficiente per i membri del team. Proprio per aiutare i membri del team a capire meglio cosa viene fatto, la parte di documentazione API è fondamentale per sfruttare al meglio il lavoro di gruppo, ogni API sviluppata andrà documentata con le sue specifiche. Infine vengono presentati alcuni dei progetti svolti recentemente dall'azienda, che spaziano molto tra i vari settori delle aziende, alcun di questi sono:

- Ottimizzazione dello stock in magazzino
- Analitiche dei supermercati e product placement
- Settore Manufatturiero, con ottimizzazione dei processi e servizio di manutenzione.
- Consumi Energetici delle aziende
- In ambito bancario, i servizi di valutazione del credit scoring.

2.5 Seminario Red Hat

Riassunto:
In questo seminario si è evidenziata l'importanza del mondo Open Source e delle opportunità che può portare, soprattutto nell'ingegneria del software.

All'inizio il signor Fusco ha parlato della sua carriera lavorativa e di come Red Hat lo contattò per lavorare a Drools, un progetto di cui poi ne diviene responsabile, dopo la visione di due progetti open source, a cui. Di seguito ha rimarcato quindi l'importanza di partecipare a progetti open source per diversi fattori:

1. **Visibilità:** le aziende possono vedere il tuo codice e come lavori il che aiuta molto in ambito di offerte di lavoro.
2. **Conoscenza dalla condivisione:** uno dei migliori metodi di apprendimento per programmare è leggere codice di progetti open source.
3. **Peer reviewing:** virtualmente infinite persone possono vedere e contribuire allo sviluppo del codice.
4. **Community:**, il fatto di lavorare ad un progetto open source va creare una comunità di lavoro e non solo.

I progetti open source hanno però anche delle difficoltà in cui districarsi, una di queste è la licenza del progetto, può essere di due tipi:

- copy-left: è la meno permissiva, se qualcuno riutilizza il codice open source per esempio per creare un prodotto, deve riutilizzare lo stesso tipo di licenza e il codice usato, anche se modificato, deve di nuovo essere open source
- non-copy-left: più permissiva, non obbliga la ripubblicazione del codice, comprese le parti dell'originale, si può usare un tipo di licenza diversa

Un'altra problematica dell'open source è la difficoltà nel monetizzare i progetti, ci sono diversi modi con cui un'azienda ricava soldi da questi progetti, tra cui:

- Tramite donazioni degli utilizzatori.
- Creando feature aggiuntive, cosiddette premium, a pagamento.
- Tramite coaching su l'utilizzo del prodotto.
- Tramite brevetti, anche se porre brevetti su del software è spesso impossibile.
- Aggiornando il codice su richiesta di un certo cliente, tenendo comunque le modifiche pubbliche.

Partecipando a progetti open source capita che lavorando su alcune features o issues, le nostre modifiche vengano contestate o messe in dubbio, portando quindi ad una fase di stallo il progetto se si trattasse di una parte core dello stesso, quindi per evitare stalli molto lunghi, sta ai chi lavora su queste issues mantenere una buona comunicazione con gli altri contributori, scrivere codice comprensibile, usando commenti magari, non essere gelosi del proprio lavoro stando pronti a passare il testimone. La cosa più importante sta nel mantenere buone relazioni sociali, anche tramite media, altre community o tramite conferenze. Per concludere, va notato che lavorare nel mondo open source può portare benefici di:

- Autonomia sul proprio lavoro.
- Acquisizione di skills.
- Network professionale.

2.6 Seminario Microsoft

Riassunto:
L'argomento principalmente trattato all'interno del seminario è il **Testing** con tutte le sue metodologie e come viene svolto all'interno di una big tech come Microsoft. Il testing viene definito come la creazione d i pezzi di codice che servono a verificare il corretto svolgimento di altri pezzi di codice. La verifica delle operazione che il nostro codice deve svolgere, e che andiamo a controllare tramite test, viene svolta sia in modalità manuale, che automatica, basandosi sui requisiti funzionali e non. Il testing è anche incentrato alla correzione dei bug, ovvero comportamenti imprevisti da parte del software, che a volte sfuggono alle revisioni iniziali. Tutti i test lavorano per soddisfare dei determinati requisiti, che spesso vengono concordati anche con il cliente per sapere cosa deve essere sviluppato, questi test vengono chiamati **Acceptance test**. Tramite i test viene poi anche implementata una metodologia di sviluppo, il TDD (Test driven design), che essenzialmente consiste nel pianificare e svolgere la modellazione di un progetto, basandosi sugli esiti dei test che vengono svolti passo passo durante tutta la realizzazione; i suoi vantaggi sono che aiuta a gestire meglio il lavoro da svolgere, separare i requisiti fondamentali da tutto ciò che è extra e infine permette di avere sempre una linea guida in fase di sviluppo con la certezza di restare sempre attinenti a ciò che è stato richiesto. La qualità di un test viene valutata in base a quanto codice del progetto è stato verificato e testato (Code coverage), e nel tempo si sono sviluppati sempre un maggior numero di tool per gestire i test, tra gli esempi proposti figurano:

- Jest.
- Unit Test Framework.
- Test Continuativi.

Infine viene introdotta la possibilità di svolgere test sulla propria macchina, come stage 1, che precede lo stage 2 caratterizzato da test automatizzati, spesso svolti prima di effettuare una pull request, per concludere con lo stage 3 dove i test vengono svolti e creati all'interno di macchine di sviluppo o ambienti creati ad hoc, il cui scopo è solo quello di valutare i test e la loro copertura. E' importante sottolineare che il testing, svolto nel modo ideale avviene in maniera progressiva, partendo prima dalle classi valutando la loro correttezza, per poi passare a librerie, applicazioni e servizi; tutti studiati e testati continuamente ad ogni integrazione di nuovi elementi.

2.7 Seminario Molinari

Riassunto:
La tematica principale di questo seminario sono i sistemi Legacy, ovvero tutti quei sistemi informativi antiquati che si ritrovano ad essere obsoleti interamente o in qualcuna delle sue parti; ma che sono ancora in uso tutt'oggi. Le caratteristiche principali di questi sistemi sono:

- Hardware di tipo **mainframe class**, datato.
- Grande consumo energetico.
- Occupazione di grandi spazi.
- Legami a lungo termine con i fornitori.

La tipologia dell'hardware inoltre è sempre in scale molto grandi (40 TB RAM, 240 CPU server), e, al contrario di come si è soliti fare al giorno d'oggi, il sistema operativo è creato tutto attorno all'hardware. Il linguaggio più importante dei sistemi Legacy è **Cobol**, creato più di 60 anni fa, e ancora molto in uso oggi, soprattutto in ambito bancario dove è ancora il linguaggio numero 1. Questo perché è ottimo per il registro delle transazioni ed è ancora solido, basti pensare che l'80 percento delle transazioni quotidiane vengono ancora registrate con Cobol. Perché allora i sistemi Legacy vengono esclusi? I motivi principali sono:

- User experience non adatta.
- Persistenza dei dato inadeguata.
- Mancanza di carriere Professionali, che quindi non attraggono nuove figure.
- Costi di gestione e manutenzione elevati.
- Carenza di competenze dei nuovi impieghi.

Inoltre i sistemi Legacy sono anche responsabili dell'aumento del **Tech Debt**, ovvero il debito che le aziende accumulano con l'invecchiare delle tecnologie per cui hanno speso dei soldi; proprio perché si cerca sempre di correre dietro alla modernità e le nuove tecnologie.

Allora perché resistono? Innanzitutto perché i sistemi legacy sono ancora utilizzabili e affidabili e andando a sostituirli si perderebbe l'investimento fatto eliminando sistemi ancora funzionanti, inoltre c'è, come per ogni rinnovamento, paura di migrare verso nuove tecnologie diverse. Ancora ad oggi infatti non esiste un alternativa hardware-software al pari dei mainframe per la gestione delle transazioni, inoltre la potenza di calcolo sarebbe eguagliabile solo facendo dei grandi investimenti. Anche per Cobol si stanno cercando punti di incontro per continuare a utilizzarlo, i modi per fare risorgere un linguaggio sarebbero:

1. Dismissione: smantellare e migrare il sistema, ma avrebbe un periodo di down in cui le transazioni non sarebbero gestite.
2. Migrazione: spostamento in un nuovo ambiente, ma con molto codice da spostare.
3. Interazione: Un sistema nuovo che interagisce con Legacy.
4. Inclusione: Il vecchio viene incapsulato nel nuovo.

Riguardo l'ingegneria del software i progetti Legacy sono spesso progetti di grandi dimensioni con persone di varie competenze che arrivano da tutto il mondo, sono porgetti che hanno costi molto elevati e la probabilità di fallimento del progetto non è considerevole. riguardo le metodologie Agile non è cosi usato proprio per la durata dei progetti che differiscono da uno sprint, inoltre vengono molto usati i linguaggi di modellazione, come UML per creare diagrammi per l'intero team che lavora al progetto rendendo più comprensibile il comunicare e l'organizzazione del lavoro.

In conclusione, Legacy non è morto, è un ambiente stimolante che sta attraversando una fase di modernizzazione e non sta scomparendo (hybrid cloud basato su Legacy). Quello che non c'è più è l'approccio alla Legacy che si è rinnovato on l'avvento delle nuove tecnologie.

2.8 Seminario Marsiglia

Riassunto:
L'obiettivo di questo seminario è fare una panoramica sul mondo del lavoro nell'ingegneria del software. Come primo tema vengono descritte le posizioni lavorative nel mondo dello sviluppo software, con le loro peculiarità:

- Specialista: Ha conoscenze specifiche in un certo ambito.
- Architetto: ha una visione d'insieme su tutti gli aspetti di struttura dei componenti e loro componenti.
- Consulente: ha una conoscenza del modello di business del azienda, spesso ha anche una visione tecnica.
- Ingegnere: molto simile all'architetto ma con meno esperienza

- Manager: Ha un ruolo di gestione del team e delle risorse, spesso ha anche un background tecnico.

I ruoli poi vengono inseriti tra i vari domini architetturali che sono Business, dati, applicazioni e tecnologia, che in questo momento è la prima cosa da guardare per lo sviluppo. Messi insieme, i domini architetturali, compongono il modello di un organizzazione. L'infrastruttura supporta il lavoro sulle applicazioni, che a loro supportano il lavoro sui dati che sono incentrati sul business dell'organizzazione.

Successivamente, dopo aver descritto i ruoli e i domini, bisogna capire come viene organizzato il ciclo di vita dello sviluppo software; ne possiamo distinguere alcune fasi necessarie:

1. Pianifica
2. Requisiti
3. Design
4. Codice
5. Test
6. Deployment
7. Mantenimento

A seconda dell'applicazione che si vuole sviluppare, le varie parti del ciclo avranno più o meno importanza e seguiranno approcci diversi, ma le fasi devono esserci tutte. Infine bisogna analizzare le pratiche di sviluppo del software che si possono diversificare a seconda del team, delle competenze e dei requisiti necessari da sviluppare, alcuni esempi sono:

- Tradizionale: a cascata, con grandi team
- Iterativo: incrementale, guidato dagli use case.
- Agile: piccole iterazioni, test driven, coinvolgimento dell'utente
- DevOps, formato da Continuous integration (CI) e Continuous delivery (CD).

Infine viene fatta una menzione speciale al mondo del cloud che già da qualche anno è entrato di forza nel mondo dell'ingegneria del software, proponendo dei mezzi molto potenti, come il **Cloud computing** che mette a disposizione servizi software molto potenti per l'utilizzo a distanza. Grazie a questi mezzi e alle grandi aziende come IBM, Amazon e Google che hanno iniziato ad offrire i servizi cloud, nel ruolo di **Cloud service providers**, l'architettura software si sta evolvendo, passando da un sistema composto di layer per le sezioni coinvolte (utente, dati e business) in un architettura monolitica, ad un unica parte dotata dei mezzi del cloud computing che svolge un unico servizio, ma sempre comunicando con le interfacce.

2.9 Seminario APSS e Trentino.ai

Riassunto:
In questo seminario si è parlato di come lavora un ingegnere del software per l'Azienda sanitaria trentina. Innanzitutto bisogna avere una gran capacità gestionale, dato che ci sono spese esorbitanti giornaliere, tra vari fattori, che sono tutte gestite da software. Software che gestisce anche la produzione di farmaci, ricette, prescrizioni, accesso APSS, ed altro ancora. Si è discusso e rimarcato come la produzione deve essere vista come processo, tutto ciò che si produce nella sanità è governato da software. Tornando alla capacità gestionale nel lavorare nel dipartimento IT dell' APSS,l 'azienda sanitaria deve fornire continuità, equità, potenziamento dei servizi sul territorio tramite tecnologie e informatizzazione dei sistemi gestionali. Dipartimento tecnologie dell'APSS ha tre tipi di clienti:

- operatori(medici, infermieri,...)
- cittadini(noi quando non siamo ancora presi in carico da strutture sanitarie)
- pazienti(noi quando siamo presi i carico, anche per una semplice prescrizione)

Il dipartimento IT dell'APSS è unico dato che dispone di autonomia gestionale, è un azienda nell'azienda, nel settore pubblico. Si è parlato di come l'azienda sanitaria deve essere una certezza su tutto il territorio e quindi di come i sistemi già informatizzati non possano fallire nel produrre. A questo proposito si è discusso di pro e contro tra il sistema di ingegneria software adottato ora dall'APSS, il waterfall, rispetto al sistema agile, proprio perché l'azienda sanitaria deve essere una certezza sia su territorio ma anche su tempo e costi, si preferisce waterfall, anche data l'enormità di dati, apparecchiature che deve gestire. Si è parlato di come nel futuro si voglia passare da datacenter a cloud per memorizzare dati dei pazienti e dei rischi che comporterebbe sia a livelli di costi che di privacy. Si è poi parlato di come si possa introdurre l'AI nella sanità, sia per l'aumento della capacità organizzativa, che della gestione clienti, consulenze a distanza e sviluppo di nuovi farmaci. A questo punto si è discusso in breve di nuovo i rischi che comporterebbe l'AI a livello di privacy, ma soprattutto della irreversibile perdita del lavoro che potrebbe portare. Si è parlato poi di come stanno organizzando il nuovo progetto per consultare il proprio fascicolo sanitario e delle problematiche che si stanno portando dietro da anni data la precedente poca inclinazione alla digitalizzazione di questi servizi, ad esempio tutti i referti nel fascicolo sanitario digitale sono solo delle scannerizzazioni di quelli fisici, quindi file pdf, il progetto vuol far si che si renda possibile atomicizzare i dati del paziente in modo da poter far si che da questi si possano poi generare i referti, ricette etc, cosicché poi si rendano anche più accessibili i dati come malattie pregresse, infortuni passati a medici anche al di fuori della propria regione o stato. Il punto chiave evidenziato maggiormente del seminario è stato di vedere la Produzione come processo, ovvero tutto ciò che viene prodotto

sarà gestito da un processo, quindi da software che da noi ingegneri del software andrà sviluppato.

3 Applicazione dei Seminari al progetto

In questa sottoparte del D5 viene descritto come sono stati utilizzati, e le informazioni ricavate da essi, i seminari visti durante il corso, all'interno del nostro progetto. Partendo proprio dalla stesura e divisione dei documenti assegnati che aiutano nell'organizzazione e nell'avanzare del flusso del progetto, un po come visto nel seminario **Bluetensor**, e nel seminario **Marsiglia** con l'organizzazione del lavoro nelle aziende all'interno del mondo dell'ingegneria del software, molto simile a quella a cui ci siamo attenuti. Un altro metodo che ci è risultato molto utile è stata l'applicazione del metodo agile, descritto in molti seminari, **Bluetensor**, **U-Hopper** come la miglior pratica in ambito di sviluppo; infatti tutte le parti sono sempre state in continua modifica come previsto dal flusso agile. Un'altra tematica trattata nei seminari che è stata molto importante all'interno del nostro progetto è stata il testing, esso viene discusso nel seminario di **Microsoft**, in cui vengono descritti alcuni tool all'interno del mondo del testing, tra cui Jest, che abbiamo utilizzato per creare ed eseguire i nostri casi di test, e ottenere una copertura del codice il più alta possibile. Un'altra applicazione dei seminari è stata la parte di modellazione nei documenti iniziali con UML che ha permesso di descrivere visivamente i requisiti e le funzioni della nostra applicazione con vari diagrammi. Esso è descritto come fondamentale anche all'interno dei progetti Legacy visti nel seminario **Molinari**. Gli argomenti dei seminari sono stati molto vari e non tutti sono stati attinenti al lavoro svolto nel nostro progetto, ma ci hanno fornito una panoramica generale sia del mondo del lavoro nell'ambito dell'ingegneria del software, sia delle tante tipologie di aziende presenti nel mercato, aiutandoci a capire quale settore potrebbe essere più addetto alle nostre capacità e preferenze.

4 Organizzazione del Lavoro

Il lavoro è stato organizzato dividendo ogni documento in delle sottoparti che andavano assegnate ad ogni membro del gruppo; ogni volta finito un documento è stato fatto un incontro di verifica generale dei vari incarichi prima di concludere i documenti. Per esempio se Stefano faceva i requisiti funzionali, durante l'incontro lasciava da verificare il suo lavoro ad un altro membro del team, mentre lui a sua volta verificava un altro argomento svolto da uno degli altri. Nel complesso il lavoro è stato suddiviso equamente soprattutto nei primi documenti, quando il carico di lavoro è aumentato abbiamo prediletto lavorare in gruppo ma concentrati su più topic per volta per ottimizzare le tempistiche. Durante il progetto abbiamo cercato di adottare una metodologia simile all'Agile, soprattutto nei primi documenti, tornando spesso a modificarli, l'obiettivo era quello di delineare correttamente i primi 3 documenti per poi passare alla parte da implementare senza avere aspetti definiti precedentemente da modificare in un documento, e di conseguenza an che in quelli successivi.

5 Ruoli e Attività

Componenti del Team:

- **Zakaria Aoukaili:** Zakaria ha svolto il ruolo di Team leader del progetto essendo quello che aveva più esperienza tra noi nel campo dell'ingegneria del software. Ha partecipato a praticamente tutti i compiti nel progetto. Principalmente ha lavorato sul D4 come si potrà evincere dalla distribuzione delle ore, dando una mano soprattutto sul FrontEnd e sullo sviluppo delle API del Backend, ma ha dato una direzione generale su come organizzare il lavoro di tutto il progetto. Ha inoltre svolto una parte dei diagrammi nei primi documenti e lo User Flow presente anch'esso nel D4.

  Ruoli: Team Leader, Sviluppatore Back-End, Sviluppatore Front-End, Grafico, Progettista

- **Brando Corti:** Brando è stato il curatore della produzione di tutti i deliverables, ha contribuito alla creazione dei diagrammi UML nei primi documenti oltre che il diagramma delle classi e dei componenti. E' stato responsabile della parte di documentazione API nel D4 con Swagger e di prima stesura dello User FLow. Nel complesso anche lui ha lavorato su tutti i compiti.

  Ruoli: Analista, Progettista, Documentatore Back-End, Modellista.

- **Stefano Girardi:** Stefano come gli altri ha contributo a tutto il lavoro, in particolare è stato il gestore del Testing delle API e ha fatto parte dello

sviluppo API nel codice Backend durante il D4 insieme alla API extraction. Inoltre ha realizzato il diagramma del contesto del D2 e ha svolto la creazione di tutto il codice OCL all'interno del diagramma delle classi nel D3.

Ruoli: Progettista, Sviluppatore Back-end, Tester, Modellista.

6 Distribuzione del lavoro

	D1	D2	D3	D4	D5	Total
Zakaria Aoukaili	8	10	7	100	5	130
Brando Corti	8	21	13	69	10	121
Stefano Girardi	7	13	10	72	6	109
Total	23	44	30	241	21	360

Da come si può evincere dal conto delle ore, il lavoro è stato distribuito in maniera equa, e il lavoro più grande è stato svolto durante il D4.
La differenza di ore più significativa è proprio nel D4, dove Zakaria ha dedicato del tempo in più rispetto agli altri membri allo sviluppo del Front-End dell'applicazione, che ha svolto da solo. Oltre a questo le ore relative ai compiti svolti evidenziano come ogni membro del team abbia partecipato più o meno con lo stesso apporto degli altri a tutte le attività svolte.

7 Criticità

All'inizio del corso abbiamo trovato delle difficoltà nell'incontrarci in quanto avevamo orari diversi relativi alle lezioni e ai corsi seguiti, infatti ci sono stati periodi in cui solo uno di noi stava lavorando al progetto. inoltre nel primo documento il lavoro è stato svolto tutto in gruppo, senza dividercelo, perché non sapevamo ancora come gestire le assegnazioni. In seguito siamo riusciti a organizzare il lavoro in incarichi individuali che poi andavano ispezionati dagli altri membri del gruppo come metodo di verifica. Arrivati allo sviluppo del codice ci siamo trovati spiazzati da quanto lavoro c'era da fare per restare coerenti con ciò che era stato scritto nei documenti precedenti, e questo ha rallentato la nostra produzione oltre che impedirci di consegnare entro la nostra aspettativa della prima sessione disponibile. Solamente nel periodo antecedente alla sessione estiva, siamo riusciti con costanza a vederci e lavorare al progetto insieme.
All'inizio del progetto abbiamo inoltre avuto qualche difficoltà riguardo le idee riguardanti dell'applicazione, in quanto tra di noi avevamo delle idee diverse su cosa andava fatto, col progredirsi del corso le nostre idee si sono allineate su un idea comune e questo ci ha aiutati nella fase di progettazione.

8 Autovalutazione

In conclusione, ci sentiamo di esserci impegnati molto nell'intero progetto, e siamo felici del risultato ottenuto, anche considerando il tempo impiegato e la mole di lavoro fatta. Il team è riuscito a lavorare in gruppo in maniera ottimale, e ogni componente ha fatto la sua parte e detto la sua opinione all'interno del progetto. la nostra autovalutazione viene riportata nella tabella sottostante:

Studente	**Voto**
Zakaria Aoukaili	30
Brando Corti	30
Stefano Girardi	30