Report finale

Scopo del documento

Il presente documento costituisce un rapporto dettagliato sull'attività svolta, in cui saranno delineati i seminari frequentati insieme alle loro caratteristiche distintive e l'applicazione pratica delle conoscenze acquisite all'interno del progetto.

Successivamente, si fornirà un'analisi dell'organizzazione del lavoro, la distribuzione dei compiti all'interno del team, il carico di lavoro e le sfide affrontate. Infine, sarà presentata un'autovalutazione del progetto.

Approcci all’Ingegneria del Software

Verranno ora descritti brevemente i vari seminari che sono stati offerti durante il corso, mostrando caratteristiche, specificità, punti di forza e criticità degli approcci all’ingegneria del software presentati, discutendo infine l’approccio da noi scelto.

1 - BlueTensor

BlueTensor, un'impresa situata in Trentino specializzata in tecnologie di intelligenza artificiale, ha tenuto un seminario durante il quale ha esposto il processo di sviluppo di **Eyerus**, uno dei suoi principali prodotti. Nel corso di questa presentazione, è stata sottolineata l'importanza di un'accurata analisi di fattibilità all'inizio di ogni progetto, al fine di prevenire inutili sprechi di tempo e risorse.

In seguito, viene preparato un documento comprensivo degli obiettivi specifici da conseguire prima di avanzare verso la realizzazione effettiva della soluzione richiesta.

Nella fase di sviluppo, l'azienda effettua molteplici rilasci intermedi, permettendo così al cliente di fornire feedback sulle nuove funzionalità introdotte. Tale strategia facilita l'immediata identificazione e correzione di qualsivoglia discordanza tra le esigenze del cliente e le soluzioni proposte, assicurando il raggiungimento degli obiettivi prefissati.

2 - Metodo Kanban

Durante il seminario è stato introdotto un metodo largamente adottato nell'ambito dell'ingegneria del software denominato Kanban, attraverso una simulazione. Questa metodologia si basa sulla gestione di una serie di micro-task che possono essere assegnate, sospese, o concluse dai membri del team. Ogni task è rappresentata su una bacheca suddivisa in colonne, le quali indicano le attività non ancora assegnate, quelle in corso di elaborazione e quelle completate. Ogni componente del team è responsabile di un determinato numero di task, con una quantità variabile in base a una specifica formula progettata per ottimizzare il flusso di lavoro e incrementare l'efficienza produttiva.

L’introduzione di un limite al numero di task (WIP) ha evidenziato l’efficacia del metodo, ovvero consentendo di migliorare la prevedibilita’ riguardo alla data di conclusione del progetto.

Il nostro team ha deciso di implementare questa soluzione per tutta la durata del corso.

3 - IBM

Nel corso del seminario sono stati presentati i servizi Cloud di IBM, un'azienda multinazionale specializzata in servizi software e hardware. L'utilizzo degli strumenti cloud forniti da IBM offre notevoli vantaggi, tra cui una maggiore sicurezza da attacchi esterni e la possibilità di distribuire contenuti in modo rapido e accessibile a livello globale.

IBM ha adottato una pratica diffusa nell'utilizzo di documenti, artefatti e diagrammi come parte integrante delle sue operazioni interne. Questi elementi, spesso sviluppati in-house, costituiscono una componente cruciale nella documentazione e nella rappresentazione visiva dei processi, dei progetti e delle soluzioni aziendali.

Tuttavia, è importante notare che l'adozione di questo approccio, sebbene abbia dimostrato la sua utilità in termini di documentazione dettagliata e rappresentazione visiva, può comportare alcune sfide intrinseche. In particolare, esso può rendere le modifiche o le correzioni a tali documenti e artefatti più complesse e richiedere una pianificazione accurata.

4 - META

Durante la sua conferenza, il relatore rappresentante di Meta, ex-studente dell’Università di Trento, ha posto l'accento sulla discrezionalità concessa ai vari team aziendali nella gestione delle proprie attività lavorative, sia per quanto riguarda la scelta del metodo di gestione che la preferenza del linguaggio di programmazione. Questa autonomia dimostra che l'interesse primario dell'organizzazione è focalizzato sul raggiungimento degli obiettivi prefissati piuttosto che sulle specifiche modalità operative adottate per conseguirli. Nonostante l’assenza di una struttura operativa rigida, l’importanza di mantenere una documentazione condivisa e canali di comunicazione specializzati e’ stata enfatizzata.

Questo modello organizzativo è stato descritto come agile, basato sulla premessa che un impegno condiviso e serio da parte di tutti i membri del team possa tradursi in risultati di elevata qualità. Tuttavia, è stato anche rilevato che in circostanze dove l'impegno di alcuni componenti del gruppo possa risultare carente, un approccio più strutturato potrebbe rivelarsi benefico.

5 - U-Hopper

U-Hopper è un’azienda che opera nel campo dell’intelligenza artificiale e del data analysis. Dopo aver introdotto l'azienda, le tecnologie impiegate e i linguaggi di programmazione utilizzati, il relatore ha fornito un dettagliato resoconto delle pratiche ingegneristiche di U-Hopper. L'organizzazione adotta una metodologia

Agile, procedendo attraverso sprint bi-settimanali, durante i quali si perseguono micro-obiettivi strategici per il conseguimento dell'obiettivo finale.

Ogni sprint si articola in tre fasi principali: lo sviluppo del codice, il suo testing e la revisione del codice. L'impiego di git e l'organizzazione del codice in branch specifici è stato evidenziato come cruciale per mantenere una struttura progettuale efficace. In particolare, sono stati menzionati due branch locali dedicati rispettivamente allo sviluppo delle API e allo sviluppo dell'interfaccia grafica.

In conclusione, è stata sottolineata l'importanza della documentazione prodotta, delle revisioni periodiche e del rilascio del progetto al suo completamento, dimostrando l'attenzione dell'azienda verso un rigoroso processo di sviluppo software che garantisca qualità e efficienza.

6 - RedHat

Il seminario ha posto un'enfasi particolare sull'Open Source, evidenziandone l'importanza non soltanto per gli sviluppatori individuali ma anche per le aziende. È stato sottolineato come l'Open Source rappresenti un'opportunità per le imprese di rafforzare la propria immagine di marca.
L'adozione di pratiche Open Source non beneficia unicamente le organizzazioni, ma anche le attività svolte dagli sviluppatori possono servire come efficace vetrina per le aziende in cerca di programmatori. Il relatore ha citato la propria esperienza personale, evidenziando come sia stato assunto da RedHat senza la necessità di sostenere un colloquio, grazie alla rilevanza del suo contributo nel mondo Open Source quale dimostrazione delle sue competenze nello sviluppo software.

Durante l'esposizione, sono stati illustrati i principi fondamentali dell'Open Source, insieme alle diverse modalità attraverso le quali è possibile generare entrate. Si è discusso delle dinamiche che regolano lo sviluppo di un progetto Open Source, sottolineando come il successo di tali iniziative dipenda fortemente dalla qualità della documentazione, che deve essere estremamente dettagliata, accessibile e comprensibile, al fine di facilitarne il contributo. È stato evidenziato che, quando una problematica viene risolta, questa soluzione deve essere comunicata a tutta la comunità, permettendo ai futuri contributori di impegnarsi nel progetto o di proseguire il lavoro senza dover risolvere nuovamente problemi già affrontati.

7 - Microsoft

Nel corso del seminario, è stata approfondita l'importanza dell'implementazione dei test all'interno delle procedure di Microsoft, sottolineando come l'esecuzione di un'estesa serie di test sia cruciale per assicurare la robustezza delle soluzioni

software, dato il notevole volume di traffico che tali sistemi sono chiamati a gestire. I test rappresentano uno strumento essenziale non solo per la prevenzione dei rischi ma anche per consolidare la fiducia degli stakeholder nei confronti dei prodotti sviluppati.

La filosofia operativa di Microsoft in ambito di ingegneria del software mostra una significativa similitudine con quella adottata da Meta: entrambe le aziende conferiscono grande autonomia ai propri team di sviluppo nella scelta delle metodologie di gestione del lavoro, pur sottolineando l'importanza della documentazione come strumento indispensabile per coordinare efficacemente le attività.

8 - Molinari

Durante il seminario è stata evidenziata la complessità rappresentata dai sistemi legacy nell'ambito dell'ingegneria del software. La problematica principale con tali sistemi risiede nella loro fondazione su tecnologie datate, il che rende il loro aggiornamento verso soluzioni tecnologiche più recenti un processo estensivo e laborioso. La gestione e l'aggiornamento di questi sistemi richiedono l'intervento di personale altamente specializzato, dotato delle competenze specifiche necessarie per affrontare le peculiarità delle vecchie tecnologie.

L'importanza dell'ingegneria del software si manifesta con particolare evidenza nel contesto di progetti di ampia scala, dove l'analisi diretta del codice può risultare eccessivamente onerosa. In questi scenari, diventa cruciale la disponibilità di una documentazione dettagliata e aggiornata, che consente di comprendere e gestire l'infrastruttura software esistente. Questo approccio enfatizza la necessità di investire risorse significative non solo nello sviluppo di nuove soluzioni ma anche nella cura e nel mantenimento di quelle preesistenti, assicurando così la loro funzionalità e sicurezza nel tempo.

9 - Marsiglia

Nel seminario tenuto da Gerardo Marsiglia, è stata enfatizzata l'importanza dell'adozione di metodologie DevOps nell'ambito dell'enterprise architecture per affrontare le sfide poste dalle evoluzioni tecnologiche e dal passaggio a infrastrutture cloud. Il relatore ha illustrato come il DevOps faciliti la collaborazione tra i team di sviluppo e operativi, promuovendo pratiche di Continuous Delivery per l'automazione dei test e del deployment, essenziali per la modernizzazione delle applicazioni e la scalabilità dei progetti su diverse piattaforme cloud.

Il seminario ha anche esplorato l'evoluzione delle metodologie di ingegneria del software, dall'approccio a cascata all'Agile, fino al DevOps e alla Continuous Delivery, evidenziando la necessità di adattare queste metodologie in base a fattori come team e tecnologie. L'incidenza delle tecnologie cloud è stata

sottolineata come elemento rivoluzionario, che rende il DevOps sempre più centrale per progetti che richiedono rapidità, integrazione e testing continui.

10 - APSS

Il seminario conclusivo ha offerto uno sguardo approfondito sul reparto IT dell'APSS, evidenziando l'integrazione di tecnologie avanzate come il cloud e l'intelligenza artificiale nel sistema sanitario trentino. Si è discusso di come queste tecnologie migliorino i servizi sanitari, con particolare attenzione all'uso di AI per diagnosi più rapide e precise. È stato sottolineato l'approccio di co-design nello sviluppo software, che enfatizza l'integrazione tra aspetti tecnologici e funzionali, la sicurezza dei dati e l'importanza della corretta categorizzazione delle informazioni. Il seminario ha confermato che le metodologie e le tecnologie di ingegneria del software sono cruciali per il successo e l'efficienza di organizzazioni di ogni dimensione, dal settore pubblico alle startup, applicabili oltre il settore sanitario a vari ambiti aziendali.

Organizzazione del lavoro

La divisione delle attività e' avvenuta in maniera abbastanza equilibrata, adattandosi ai vari impegni dei membri del team.

Nella prima fase del progetto, ovvero i primi 2 documenti, l'attività è stata svolta principalmente in gruppo, soprattutto durante la stesura dei requisiti della nostra applicazione. In questa prima parte il focus si è incentrato nell'individuare le funzionalità che la nostra applicazione avrebbe dovuto avere.

Finita la stesura dei requisiti funzionali abbiamo impostato una tabella kanban che abbiamo rispettato per tutta la durata del corso. La tabella era formata da piu' colonne:

- **Backlog**: utilizzata per le task da iniziare;
- **In progress**: utilizzata per le task in svolgimento da uno o piu' membri del team;
- **Review**: utilizzata per segnalare quali task siano state completate e quindi sono pronte alla review di un componente del team che non ha partecipato allo svolgimento di tale task;
- **Tutor Feedback**: eventualmente, nel caso di dubbi, alcune task venivano spostate in questa colonna per segnalare un problema da discutere con un tutor/professore;
- **Done**: utilizzata per le task concluse e verificate.

Ogni volta che veniva aggiunto un nuovo argomento da svolgere nei vari documenti, veniva segnato nella tabella Kanban come nuova task e veniva spostata nella colonna di *backlog*. L'assegnamento delle task avveniva attraverso una discussione sul gruppo Telegram oppure in classe.

Relativamente allo sviluppo del codice, per velocizzare le operazioni, si è deciso di spartirsi il carico in modo tale che ogni membro operi nell'ambito in cui sia più esperto.

Per concludere questa sezione elenchiamo i software utilizzati nella fase di stesura dei deliverables, progettazione e sviluppo del software:

- **Google Docs**: abbiamo adottato questo strumento perché permette la sincronizzazione e aggiornamento del lavoro insieme alla possibilità di inserire commenti.
- **Git**/**GitHub**: strumento fondamentale, utilizzato non solo come drive condiviso, ma anche come sistema per organizzare il lavoro attraverso la tabella Kanban.
- **Figma**: questo software non solo è stato utilizzato per la realizzazione di un primo prototipo, ma si è reso utile anche durante la creazione di alcuni diagrammi.

- **LucidChart**: ci siamo serviti di questo software per la realizzazione di diversi diagrammi richiesti nei deliverables.
- **Telegram** e **Discord**: strumenti utilizzati per la comunicazione nel team, permettendoci di mantenerci aggiornati.

Suddivisione delle task

A questo punto riportiamo i ruoli che ognuno di noi ha assunto durante lo sviluppo del progetto E-20.

Componente del team	**Ruolo**	**Principale attivita'**
Marian Ologu	Project Leader, Frontend	1. Si è occupato del design dell'app e della successiva realizzazione del frontend 2. Ha realizzato la stesura di parte del D1, D2 e D4 3. Ha contribuito attivamente alla review delle attività svolte
Giulio Pimenoff Verdolin	Backend, Testing, Deployment	1. Si è occupato della realizzazione del backend 2. Ha realizzato la maggior parte dei diagrammi presenti in questi documenti 3. Ha realizzato la stesura di buona parte del D4, mentre ha contribuito a D1,D2 e D3
Michele Pezzo	Documentazione	1. Si è occupato della realizzazione del mock up su Figma 2. Ha realizzato la stesura di buona parte del D1, D2, D3 e D5, mentre ha contribuito al D4

Criticità

Nella fase iniziale, come menzionato in precedenza, abbiamo trascorso molto tempo collaborando direttamente, il che ha avuto un impatto positivo, soprattutto per la direzione che volevamo dare al progetto in quanto ci ha permesso di chiarire qualsiasi dubbio sul nascere, per poi essere sulla stessa pagina nella fasi successive del progetto, ma ha anche significativamente prolungato i tempi di lavoro, causando frequenti ritardi nelle consegne programmate. Di conseguenza, lo sviluppo del software è stato posticipato fino all'ultimo momento. Tuttavia, una volta iniziata la fase di codifica, siamo stati in grado di riorganizzare il nostro approccio, distribuendo in modo più efficace i

compiti, ciò ci ha permesso di completare il lavoro entro le scadenze stabilite. La fase di sviluppo del codice si è svolta con rapidità ed efficienza: le questioni relative all'integrazione tra frontend e backend sono state affrontate tramite discussioni sul nostro gruppo Telegram e soprattutto facendo riferimento alla documentazione condivisa, facilitando una rapida risoluzione dei problemi.

Carico e distribuzione del lavoro

Ora presenteremo in forma tabellare un resoconto più preciso della distribuzione del lavoro per il progetto, indicando le ore impiegate nella redazione dei vari documenti e nella scrittura del codice.

	D1	**D2**	**D3**	**D4**	**D5**	**Totale**
Marian Ologu	18	21	9	65	5	**119**
Giulio Pimenoff	16	27	13	47	7	**110**
Michele Pezzo	26	23	17	20	13	**99**
Totale	**60**	**71**	**39**	**132**	**25**	**327**

Autovalutazione finale

Nel corso di questi mesi, nonostante i numerosi impegni, abbiamo dedicato tempo al progetto, supportandoci reciprocamente e scambiandoci conoscenze.

Non tutti i membri del team avevano sviluppato un progetto di tale ampiezza, quindi riteniamo che non solo sia stata un esperienza formativa dal punto di vista pratico, ma anche dal punto di vista relazionale. In base a queste riflessioni, procediamo ora con la nostra autovalutazione:

Componente del team	**Valutazione**
Marian Ologu	30
Giulio Pimenoff	30
Michele Pezzo	30