Progetto:

BUDGETpal

Titolo del documento:

ReportFinaleDelProgetto

Document Info:

Doc. Name	D5-BUDGETpal_ReportFinaleDelProgetto	**Doc. Number**	**I45**
Description	Organizzazione e suddivisione del lavoro e ruoli, tempo complessivo e di ciascun membro dedicato al progetto, descrizione delle criticità e autovalutazione.		

Indice

1 Scopo del documento

Il presente documento rappresenta il report conclusivo relativo allo sviluppo del progetto BUDGETpal da parte del nostro team, noto come G45.

Inizialmente, verrà delineata l'organizzazione e la suddivisione del lavoro tra i membri del team, seguita dalle ore di lavoro complessive di ciascun membro per ogni deliverable, con una spiegazione delle differenze riscontrate.

Successivamente, saranno elencate le criticità e le problematiche incontrate durante il progetto e le eventuali strategie di risoluzione.

Infine, sarà inclusa un'autovalutazione del progetto basata sui punti precedentemente discussi.

2 Organizzazione del lavoro

Inizialmente, ci siamo impegnati a distribuire il lavoro in modo equo tra i membri del nostro team, tuttavia, nel corso del progetto, abbiamo incontrato alcune sfide nell'ottenere una distribuzione uniforme delle attività. Descriveremo brevemente le modalità di lavoro adottate e le relative sfide incontrate.

Fin dall'inizio, abbiamo cercato di favorire la suddivisione e la cooperazione tra i tre membri del gruppo. Tuttavia, abbiamo trovato difficoltà nel bilanciare equamente la mole di lavoro, soprattutto considerando il progresso dell'anno universitario e le esigenze individuali di ciascun membro.

I compiti sono stati affrontati principalmente individualmente, previa organizzazione preliminare tra i membri del gruppo, solitamente tramite chiamate virtuali. Le riunioni, sebbene inizialmente programmabili anche in presenza, si sono svolte principalmente online, durante le quali abbiamo discusso e presentato lavoro svolto, espresso le opinioni in confronto e pianificato i prossimi deliverable. Tuttavia, la frequenza di tali incontri è stata altalenante in base alla quantità di lavoro assegnata.

Nonostante queste sfide, all'inizio del progetto, grazie alla regolare partecipazione alle lezioni, siamo riusciti a definire chiaramente gli obiettivi principali da raggiungere.

Per la stesura dei deliverable, abbiamo utilizzato Microsoft Word, mentre per lo scambio di materiale abbiamo impiegato diverse piattaforme cloud che consentono la collaborazione e la revisione delle versioni del documento. GitHub è stato utilizzato solo per il caricamento finale dei deliverable da consegnare, poiché lo sviluppo dell'applicazione non ha coinvolto una collaborazione basata su Git. I diagrammi sono stati principalmente creati con LucidChart e VisualParadigm, mentre i MockUp delle pagine frontend nel deliverable D1-G45 sono stati realizzati con semplice codice HTML e CSS.

3 Ruoli e attività

Nella seguente tabella andremo a riassumere il ruolo che ha avuto ogni componente del team nel progetto.

In certi casi c'è stata un suddivisione diversamente equa della mole di lavoro, in altri casi no invece.

Componenti del team	Ruolo	Principali attività
Gabriele Menestrina	Analista dei requisiti funzionali nel D1 e non funzionali nel D2, redattore principale del Deliverable 1 (D1 scopo dell'applicazione web, requisiti funzionali e requisiti non funzionali, descrizione frontend e backend), sviluppatore di MockUp frontend, analista aiutante per i diagrammi dei contesti e revisionatore dei diagrammi dei requisiti funzionali, redattore principale del Deliverable 2 (D2), e aiuto nella pianificazione del diagramma delle classi D3. Redattore del Deliverable 4 e relativi aspetti, sviluppatore esclusivo della WebApp BUDGETpal, sia frontend che backend, sviluppatore di tutte le API, tester del codice, sviluppatore dell'integrazione di Swagger, redattore principale del D5 (include anche la descrizione parziale dei riassunti).	Ha redatto il Deliverable D1 e ha svolto un ruolo nella redazione del D2, ad eccezione della descrizione del diagramma delle componenti. Ha delineato lo scopo e obbiettivo dell'applicazione nel D1, redatto i requisiti funzionali in dettaglio nel D1 e contribuito alla descrizione dei requisiti non funzionali nel medesimo deliverable, la descrizione del frontend e relativi MockUp e contribuito nella stesura della descrizione del backend sempre nel D1. Nel D2, ha contribuito alla descrizione dei requisiti funzionali e alla stesura dei relativi diagrammi, ha fornito una descrizione completa dei requisiti non funzionali e ha curato la sistemazione dei diagrammi di contesto insieme alla loro descrizione. Nel D3 ha contribuiti nella pianificazione (aiuto) del diagramma delle classi. Ha sviluppato l'intero software, dal backend al frontend, e ha redatto l'intero D4, includendo lo sviluppo di tutti i diagrammi delle API e delle risorse con l'ausilio di Visual Paradigm. Di conseguenza, si è occupato di tutto il D4, incluso la descrizione del codice. Ha sviluppato tutte le API, integrato Swagger, testato il codice e implementato il deployment dell'applicazione. Inoltre, ha svolto il ruolo di redattore del D5 e descritto parzialmente riassunti D5.
Danyyil Aheyev	Analista requisiti non funzionali, analista e revisionatore dei requisiti funzionali nel D1. Analista diagrammi dei requisiti funzionali Use Case con relative	Ha svolto all'interno del D1 l'individuazione e la descrizione dello scopo e dell'obbiettivo dell'applicazione Web, la descrizione dettagliata dei

	descrizioni, revisionatore dei parametri misurabili dei RNF, analista sviluppo diagrammi di contesto nel D2. Redattore principale del Deliverable 3. Ideatore dell'architettura delle classi, analista del diagramma delle classi, analista OCL con relativa descrizione nel D3. Aiuto nella redazione di D4. Riassunti della parte dei seminari nel D5.	requisiti non funzionali ed ha lavorato insieme con Gabriele sull'individuazione e descrizione di quelli funzionali. Ha sviluppato e descritto i diagrammi Use Case per i requisiti funzionali e si è occupato della creazione dei diagrammi del contesto nel D2 e infine ha contribuito in parte alla redazione del documento D2. Per quanto riguarda D3, ha individuato l'architettura e le classi necessarie con relativi attributi e metodi. Basandosi su ciò ha sviluppato una versione del diagramma delle classi che sta alla base del diagramma finale. Ha individuato i vincoli e li ha descritti in linguaggio OCL per integrarli successivamente nel diagramma delle classi completo. Ha fornito aiuto nella redazione del documento D4. Infine ha riassunto la propria parte dei seminari nel D5.
Riccardo Mascotto	Diagramma del backend D1. Sviluppo diagramma delle componenti e descrizione nel D2. e analista diagramma delle classi D3 e la relativa descrizione delle classi, attributi e metodi coinvolti. Analista User flow del D4. Riassunti seminari D5.	Ha collaborato alla realizzazione del diagramma del backend all'interno del deliverable D1 e ha contribuito alla revisione dell'obiettivo delineato nel D1. Nel D2 ha progettato, disegnato e descritto il diagramma dei componenti, e contribuito alla revisione del D2. Ho progettato, disegnato e descritto completamente il diagramma delle classi nel D3. Ha progettato e disegnato l'User Flow. Riassunto parzialmente i seminari tenutisi nel corso dell'anno all'interno del D5, e contribuito alla revisione dei riassunti nel D5.

4 Carico e distribuzione del lavoro

Nella tabella qua sotto presente vengono mostrate le diverse ore dedicate da ogni componente del gruppo per il progetto suddiviso nei 5 Deliverable.

	D1	D2	D3	D4	D5	Totale
Gabriele Menestrina	22,5	30,5	10	160	12	240
Danyyil Aheyev	20	26,5	45	10	11	112,5
Riccardo Mascotto	25	25	40	15	10	115

Per una maggiore chiarezza ora provvederemo a descrivere per ogni lavoro come è stato diviso il lavoro.

Deliverable 1:

- **Gabriele** si è occupato di redare il documento (redattore principale) descrivendo lo scopo del progetto, i requisiti funzionali ed ha contribuito nella stesura dei requisiti non funzionali. Ha descritto il frontend e disegnato i diversi MockUp della WebApp, inoltre ha contribuito alla stesura dei requisiti del backend.
- **Danyyil** si è occupato dell'individuazione e della descrizione dello scopo del progetto, della stesura dettagliata dei requisiti non funzionali ed ha aiutato nella descrizione e revisione di quelli funzionali. Ha contributo anche nella descrizione dei requisiti del backend.
- **Riccardo** si è occupato di disegnare il diagramma del backend. Ha contribuito nel delineare lo scopo.

Deliverable 2:

- **Gabriele** si è occupato di redare il documento (redattore principale) descrivendo interamente i requisiti non funzionali, ha contribuito alla descrizione dei diagrammi Use Case. Ha curato la sistemazione dei diagrammi di contesto e si è occupato della descrizione piu' dettagliata di quest'ultimi.
- **Danyyil** si è occupato di una parte della redazione del documento descrivendo la parte dei requisiti funzionali e spiegando il diagramma di contesto, disegnato i diagrammi dei requisiti funzionali (Use Case) e descritto quest'ultimi, aiutato a revisionare e perfezionare la parte delle misure relative ai requisiti non funzionali, creato i diagrammi del contesto.
- **Riccardo** si è occupato della creazione del diagramma dei componenti e della sua descrizione

Deliverable 3:

- **Danyyil** redattore principale del documento, si è occupato dell'individuazione preliminare dell'architettura, definizione delle classi nel loro complesso (ideazione dei ruoli degli attributi e delle funzionalità svolte dai metodi) con la descrizione in linguaggio naturale del significato degli attributi e delle funzionalità svolte dai metodi di una parte delle classi; dello sviluppo di una versione iniziale del diagramma delle classi che sta alla base del diagramma finale. Inoltre, il lavoro svolto comprende l'individuazione dei vincoli, la loro definizione formale con il linguaggio OCL e loro descrizione in linguaggio naturale. Infine ha integrato OCL nel diagramma delle classi complessivo finale.
- **Riccardo** co-redattore del documento, nello specifico ha ampliato l'architettura, definendone una nuova parte e ha organizzato il diagramma delle classi, perfezionando le interazioni tra le classi. Si è occupato della descrizione più dettagliata in linguaggio naturale del significato degli attributi e delle funzionalità svolte dai metodi di una parte delle classi.

Deliverable 4:

- **Gabriele** si è occupato della stesura intera di questo documento e della creazione totale della WebApp, dal frontend al backend. Ha quindi sviluppato l'intero software, dal backend al frontend e ha redatto l'intero D4, includendo lo sviluppo di tutti i diagrammi presenti nel documento, tra cui il diagramma riassuntivo delle API e i diversi diagrammi delle risorse. Oltre allo sviluppo di tutto il documento è il software, ha provveduto nello loro totale interezza delle API, ha integrato il framework di Swagger settando opportunamente tutto ciò che ne concerne e si è occupato completamente della fase di testing e deployment.
- **Danyyil** si è occupato di una parte relativa alla revisione del D4.
- **Riccardo si** è occupato della creazione e della descrizione dell'intero diagramma User Flow presente in questo documento.

Deliverable 5:

- **Gabriele** si è occupato di redare il documento (redattore principale), descrivendo i componenti che, dopo una valutazione congiunta, riguardano esperienze comuni a tutti i membri del gruppo. Inoltre, elaborato la propria parte dei riassunti che sono stati opportunamente suddivisi tra i tre membri del gruppo e comprendono seminari RedHat, Molinari e APSS & Trentino.ai.
- **Danyyil** dopo la discussione collettiva, si è occupato dell'ampliamento della descrizione di alcune parti soggettive, l'inserimento della parte personale riguardante la suddivisione del lavoro e attività svolte, stesura dei riassunti dei seminari Bluetensor, META e della presentazione di Gerardo Marsiglia. Ha collaborato nella supervisione e stesura ultima del documento.
- **Riccardo** si è occupato dell'inserimento della parte personale che riguarda le attività svolte e della stesura dei suoi riassunti che comprendono seminari IBM, U-Hopper e Microsoft. Ha collaborato nella supervisione e stesura ultima del documento.
- Al completamente di questo documento è stata fatta una revisione totale e definitiva di quest'ultimo tutti assieme.

Criticità

Durante il progetto abbiamo affrontato diverse difficoltà, talvolta non facili da superare. Abbiamo incontrato alcuni problemi minori legati alla comprensione degli obiettivi, ma fortunatamente li abbiamo risolti senza ulteriori complicazioni. Inoltre, abbiamo dovuto affrontare sfide nella collaborazione, che hanno causato una distribuzione del lavoro non sempre molto equa. Come risultato della suddivisione del carico non ottimizzata nelle fasi iniziali (per la maggior parte D1 e D2), il compito poteva essere svolto da più membri indipendentemente, il che ha poi comportato i problemi nell'integrazione reciproca dei risultati del lavoro di ciascuno per formare un'unica versione del documento. Gli strumenti di collaborazione come Google Suite, ma soprattutto un'organizzazione più consapevole all'interno del gruppo, hanno reso possibile ottimizzare questo processo.

Nonostante queste difficoltà, abbiamo sempre mantenuto un clima di disponibilità, consentendo un dialogo costruttivo. Ognuno di noi è riuscito a completare i propri compiti seguendo le linee guida stabilite, come concordato all'inizio.

È importante evidenziare che c'è stata sempre una volontà di collaborare non solo sul proprio lavoro, ma anche nell'offrire aiuto agli altri membri in caso di necessità.

In sintesi, nonostante gli imprevisti lungo il percorso, siamo riusciti a portare a termine il progetto nel migliore dei modi e rispettando le scadenze.

Autovalutazione

Per quanto riguarda la valutazione del progetto in sé, sicuramente è una valutazione positiva.

Come però previsto da questa sezione l'autovalutazione non riguarda solamente il risultato ottenuto ma tiene conto di un insieme di aspetti per valutarsi opportunamente. Quindi qua sotto oltre alle valutazione ogni membro del gruppo ha avuto la possibilità di scrivere anche una motivazione della propria autovalutazione.

Gabriele Menestrina:

- Considerando quando è stato detto precedentemente, la grande mole di lavoro impiegata personalmente e l'impegno che ne è seguito mi sento di darmi come autovalutazione 30.

Danyyil Aheyev:

- Ritengo che abbia dedicato sufficiente impegno per fare al meglio la mia parte, e spero di essere riuscito ad applicare abbastanza le conoscenze acquisite alle lezioni. Purtroppo a causa di alcuni problemi personali che ho incontrato lungo il percorso, non legati né ai membri del gruppo, né al progetto, ma che hanno generato difficoltà nella gestione del tempo, non sono riuscito a dare il contributo significativo allo sviluppo del codice dell'applicazione, anche se ho seguito l'andamento di questa fase. Con dispiacere riconosco che ho messo in difficoltà altri membri del team, ma ho avuto il privilegio di lavorare in un gruppo di persone serie e motivate, grazie a cui questa mia piccola mancanza temporanea non ha influenzato le tempistiche delle fasi successive.

Riccardo Mascotto:

- Ho dedicato abbastanza tempo al progetto, ma ammetto di non aver sempre rispettato le scadenze. Sto lavorando per migliorare la mia gestione del tempo e garantire una maggiore puntualità. Questo progetto mi ha sicuramente insegnato come lavorare in gruppo e mi ha fatto notare le mie lacune nella gestione del tempo. Considerando i miei sforzi e le aree in cui posso migliorare, mi darei come voto 26.

La nostra autovalutazione:

Membro del gruppo	**Voto**
Gabriele Menestrina	**30**
Danyyil Aheyev	**25**
Riccardo Mascotto	**26**

5 Riassunti Seminari

Riassunto Seminario Red Hat:

Red Hat è un'azienda americana fondata negli anni '90, che si occupa principalmente dello sviluppo e della fornitura di prodotti software open source per scopi commerciali. A partire da ora, opera come una filiale di IBM. La sessione si è aperta con un'introduzione all'Open Source in cui Mario Fusco ha raccontato la sua presentazione su "Il mio viaggio nel mondo Open Source".

La storia di Fusco è iniziata con il suo primo lavoro come sviluppatore presso Aeonware con sede a Stoccarda. Successivamente, ha terminato gli studi in ingegneria del software e si è trasferito in Svizzera, dove ha iniziato a lavorare su iniziative open source e alla fine è entrato in Red Hat come sviluppatore principale di Drools.

La manipolazione delle collezioni Java era una caratteristica cruciale dei suoi progetti. Uno di loro alla fine si trasformò in una biblioteca che abbracciava diversi progetti. Il suo interesse per l'open source affonda le sue radici nell'assenza di espressioni lambda in Java, cosa che lo ha spinto a rilasciare la libreria come progetto open source. Mentre viveva a Berlino, lavorando per NumberFour, contribuì ulteriormente all'open source creando un motore di regole con Scala, attirando l'attenzione di Red Hat, e finì per essere impiegato da loro.

Nella seconda parte del seminario si è parlato del funzionamento dell'Open Source e delle sue applicazioni in diversi ambiti.

Ci sono molti vantaggi nel partecipare allo sviluppo di software open source, uno di questi sicuramente è la condivisione della conoscenza, che consente di apprendere nuovi linguaggi di programmazione. La peer review è un'altra funzionalità che consente a numerose persone di lavorare e contribuire ai progetti, migliorandone i contenuti. Infine, anche la meritocrazia gioca un ruolo cruciale in questo campo poiché contano davvero solo le idee e le realizzazioni. Inoltre, la visibilità è un grande vantaggio quando si parla di open source perché le tue competenze possono essere valutate da altri senza un colloquio e perché costruisci una comunità in cui le persone si aiutano a vicenda, creando una rete di supporto. Ogni software open source ha licenze specifiche che ne tutelano i diritti di distribuzione e utilizzo; queste licenze possono essere divise principalmente in due categorie: copyleft, che è più restrittiva di quella non copyleft.

Infine si è parlato della sostenibilità del business dietro al mondo del open source. I software open source si basano su strategie come supporto, sviluppo personalizzato e offerta di versioni premium con funzionalità avanzate per consentire alle aziende di ottenere guadagni pur mantenendo il codice sorgente aperto e disponibile alla comunità.

Seminario: Molinari

Durante il seminario sui sistemi legacy, si è parlato dei sistemi informatici obsoleti che vengono ancora utilizzati perché le organizzazioni non intendono o non possono sostituirli. "Legacy" significa "retrodatato" rispetto alle tecnologie attuali e può essere tradotto in italiano come "obsoleto" o "vecchio". Questi sistemi possono includere hardware, software, reti, dati, processi e personale ormai superati. Il punto centrale del seminario non è tanto l'obsolescenza dei sistemi legacy, quanto il loro utilizzo attuale nonostante questa.

I sistemi legacy hanno caratteristiche distintive come hardware di classe mainframe, hardware datato, effetto lock-in con i fornitori, personale specializzato, interfacce utente basate su caratteri e l'occupazione di grandi spazi. Tra i principali fornitori di hardware ci sono IBM, Amdahl, Cray, HP, Fujitsu-Siemens, Unisys,

Hitachi e Atos, con quest'ultima come rara eccezione europea. Per il software, i principali attori sono z/OS, VMs, Unix e Linux, spesso sviluppati per specifiche esigenze hardware.

I problemi principali dei sistemi legacy includono l'obsolescenza, soprattutto del software, un'esperienza utente inadeguata, persistenza dei dati non al passo coi tempi e alti costi di gestione e manutenzione. Tuttavia, molte organizzazioni mantengono questi sistemi per preservare gli investimenti fatti, poiché aggiornare i sistemi richiede tempo, denaro e gestione parallela. Anche se robusti e affidabili, i sistemi legacy presentano una conoscenza limitata, rendendo il cambiamento difficile e spaventoso.

Un esempio significativo è quello di IBM, che vent'anni fa aveva annunciato la fine dei mainframe, per poi ritrattare rapidamente, dando vita a un movimento per preservare i sistemi legacy con nuovi plugin e ambienti di sviluppo.

Si è parlato anche di come integrare i sistemi legacy con i moderni sistemi informativi. Le opzioni includono la dismissione, con smantellamento del vecchio sistema e migrazione dei dati; la migrazione, con trasferimento di dati e software nel nuovo ambiente; l'interazione, dove il sistema legacy continua a esistere ma interagisce con il nuovo; e l'inclusione, dove il sistema legacy viene incapsulato nel nuovo sistema tramite tecniche di emulazione.

Seminario: APSS & Trentino.ai

Durante il seminario si è parlato di temi incentrati all'innovazione tecnologica, iniziative riguardo la digitalizzazione dell'APSS-Azienda Provinciale per i Servizi Sanitari. L'APSS è l'ente di coordinamento responsabile della programmazione e della gestione dei servizi sanitari e sociali su tutto il territorio della provincia di Trento.

Il dipartimento tecnologico è strutturato con diversi servizi. L'unità che si occupa della supervisione degli aspetti di tutto il ciclo di vita dei vari dispostivi medici è l'Ingegneria Clinica.

L'unità che si occupa di supervisionare l'ambiente IT (reti, data center, call center e dispositivi mobili) è Gestione IT e Supervisione Infrastruttura. L'utilizzo del cloud computing è un punto fondamentale per migliorare sia la sicurezza, sia la scalabilità dei servizi IT, che il miglioramento dell'efficienza. Inoltre vanno tenute presente le iniziative per l'aggiornamento dell'infrastruttura di rete aziendale.

Operazioni IT e Infrastruttura è l'area responsabile delle infrastrutture informatiche: reti, data center, call center e dispositivi mobili. Tra i differenti progetti che saranno realizzati entro la fine del 2023 ci sono sia la migrazione dei servizi applicativi su nuove piattaforme cloud, sia la trasformazione dei servizi on-site nell'ambiente digitale del Trentino, sia l'estensione delle infrastrutture telefoniche e di rete, che l'aumento della sicurezza e della continuità dei servizi IT.

È stato illustrato un esempio concreto dell'impiego dell'intelligenza artificiale per lo screening della retinopatia diabetica. In Trentino, circa 24.700 pazienti affetti da diabete devono sottoporsi a uno screening oculistico due volte all'anno per individuare e gestire questa patologia. Il progetto si avvarrà di dati e immagini della retina analizzati da algoritmi di intelligenza artificiale per giudicare le immagini della retina e prevedere lo sviluppo della retinopatia.

L'APSS implementerà un gateway che servirà da punto di ingresso principale per i ricercatori esterni (per la verifica degli accessi e il controllo delle resinose e i dati dell'APSS). Ciò facilita l'integrazione e l'accesso ai dati non immediatamente disponibili nella piattaforma dati.

BlueTensor:

Il seminario intitolato "Metodi di Ingegneria del SW applicati allo sviluppo di un progetto di AI" è presentato da Ing. Jonni Malacarne, fondatore e CEO dell'azienda giovane trentina Bluetensor Srl che si occupa di sviluppare progetti e prodotti industriali basati sull'intelligenza artificiale.

In particolare, uno degli aspetti importanti del loro contributo per l'industria è la linea di prodotti e soluzioni verticali che forniscono supporto alle decisioni, ottimizzazione della produzione e la gestione di know-how aziendale. Dal punto di vista tecnologico lavorano con tre tecnologie come Natural Language Processing, utile per la gestione del linguaggio scritto e parlato, Computer vision, utile per l'analisi delle immagini e Analisi predittiva che aiuta a lavorare con dati multidimensionali per poterne estrarre modelli previsionali.

Concetto chiave dello sviluppo delle soluzioni informatiche - capire cosa si aspetta il cliente per poter rendere utile il prodotto finale. Quindi sono applicabili le metodologie standard per giungere a tale scopo. Infatti sviluppare un progetto AI dal punto di vista metodologico è simile allo sviluppo di un SW "tradizionale" ma con qualche aspetto in più da tenere in considerazione. Tra questi è la disponibilità e la qualità dei dati per l'addestramento preciso, così come la multidisciplinarità dello staff, ovvero l'introduzione e comunicazione efficace con la risorsa che ha competenze nell'ambito di applicazione del progetto. Inoltre con lo sviluppo degli algoritmi AI ci sono da considerare aspetti privacy ed etici per garantire la sicurezza ai dati dei clienti trattati. Infine l'applicativo sviluppato deve essere integrato con I flussi di lavoro esistenti.

Il flusso dello sviluppo di un progetto di AI parte da un'analisi di fattibilità preliminare (1-5gg) per capire se l'idea è realizzabile. In seguito avviene l'approfondimento dello studio di fattibilità vero e proprio per ridurre al minimo investimento dell'azienda. Esso può essere declinato in due risultati: Proof of Concept (Versione base del concetto di sistema per mostrare il risultato finale) oppure Minimum Viable Product (Prodotto con alcune funzionalità già utilizzabili ma non ancora rifinito).

Dopo questa parte segue il periodo di Engineering, ovvero la scrittura e ottimizzazione del codice. Una volta rilasciato il codice, il sistema in esercizio continua a raccogliere nuovi dati con I quali grazie a determinati meccanismi lo si può riaddestrare per avere gli output sempre migliori.

Il team per un progetto solitamente consiste di seguenti figure: Project Manager, Senior SW Architect, AI Engineers, Front-End Developer, Back-end Developer, Dev Ops, Team di ricerca.

Piano di lavoro tipico consiste negli step operativi che sono raggrupati nelle unità di lavoro con output specifici detti Work Package e sono 5: WP1 Analisi e Fattibilità, WP2 Setup e Design, WP3 Data Preparation, WP4 Sviluppo/Training/Test/Deploy, WP5 Final Release e un Work Package Trasversale di gestione del

progetto. La parte iniziale prosegue in cascata perché in questo modo è possibile stimare con più precisione I possibili costi. Questo è importante per le aziende clienti perché possono pianificare più precisamente il budget dedicato a questo progetto. Con l'approccio Agile non sarebbe possibile prevedere tutti I costi dello sviluppo.

In dettaglio il Work Package 0 riguarda le modalità della gestione del progetto dalla parte di una figura che ha la consapevolezza di cosa si aspetta il cliente e riesce a gestire efficientemente le Risorse e il Tempo per raggiungere tale scopo.

Per quanto riguarda gli step operativi:

Step Operativo 1: WP1 - Insieme al cliente si studia le funzionalità richieste e criticità in evidenza così come il contesto di riferimento per definire obiettivi specifici e limiti oppure confutare la fattibilità del progetto. In output viene prodotto il documento di analisi preliminare con il report delle prove di fattibilità. In questa fase è importante documentare le funzionalità base insieme alle specifiche non funzionali. Per l'intelligenza artificale un ruolo importante è dedicato anche ai dati disponibili, in base ai quali si stima la possibilità di realizzazione, performance ed affidabilità del prodotto finale.

Step Operativo 2: WP2 - Progettazione del sistema dal punto di vista HW e SW. Infrastruttura HW selezionata deve essere idonea a produrre informazione rilevante per elaborazione dalla parte SW, le funzionalità della quale viene definita analizzando e formalizzando la descrizione sotto forma di User Stories in linguaggio naturale definite insieme al cliente. In questa fase quindi viene prodotto il documento di analisi dei requisiti funzionali e non insieme ad architettura HW e SW, in base ai quali vengono definiti vari blocchi funzionali.

Step Operativo 3: WP3 - Lavoro sui dati include la selezione e processing dei dati più idonei all'algoritmo da sviluppare al fine di ottenere un data set etichettati per addestrare e successivamente testare il funzionamento della macchina.

Step Operativo 4: WP4 - Avviene in modalità Agile con microcicli di analisi, design, sviluppo, training, test e deployment per rilasciare continuamente piccole porzioni di codice testato.

Step Operativo 5: WP5 - Il SW viene rilasciato ufficialmente con la raccolta del feedback. Si apre la fase di Continuous Learning per avere la possibilità di migliorare le performance del prodotto.

Strumenti ingegneristici per l'analisi e la comunicazione con il cliente fanno uso del linguaggio UML per la costruzione di diagrammi Use Case, Activity, State Machine e Architecture. Per modellazione di DB sono utilizzati I diagrammi Entita-Relazione, mentre per le fasi di UI design vengono prodotti Mockups di vari livelli di fedeltà.

META:

Un'altro seminario dedicato al tema Ingegneria del Software e' stato presentato da host Heorhi Raik, Ingegnere del Software e rappresentante dell'azienda multinazionale META che ha esposto gli aspetti piu' significativi collegati al processo dello sviluppo del software nel contesto di una azienda di notevole portata. All'interno di azienda l'ospite si occupa principalmente della piattaforma dati degli per fini di pubblicita' ed ecommerce.

Nella parte iniziale centrata su come fanno software, sono state menzionate le principali responsabilita' di SW Engineer e le relative aspettative. In META il ruolo di SW Engineer è abbastanza ampio. SW Engineers all'interno dell'azienda sono pensati di dover guidare I progetti. In particolare non riguarda soltanto le responsabilità strettamente tecniche, come coding, che è essenziale all'inizio del percorso di un Ingegnere del SW, ma anche di fare design dell'architettura, ovvero pensare di come legare parti diversi dell'infrastruttura in sviluppo all'interno di un singolo prodotto finale. Con esperienza nel campo, nelle responsabilità di Senior SW Engineers inizia a rientrare anche la parte di project management. Ruoli come Engineering Manager, Product Managers, Data Scientists, UX Designers sono necessari per il progetto e Ingegneri del SW hanno il compito di introdurre le persone che ricopreranno questi ruoli e legare insieme il lavoro di tutti questi componenti del team. Altrimenti, Ingegnere del SW stesso parzialmente ricopre il ruolo mancante. Ulteriori responsabilità non tecniche per gli Ingegneri del SW di più alto livello gerarchico riguardano interviewing degli altri Ingegneri del SW che vogliono far parte dell'azienda, oppure mentoring/coaching di Ingegneri junior per aiutargli a crescere professionalmente.

La professionalità dell'Ingegnere è per la maggior parte misurata dall'impatto e non tanto ad esempio dalla quantità di codice scritto.

Per quanto riguarda le metodologie di sviluppo, non c'è uno standard imposto dall'azienda per tutti e quindi ogni team è libero di scegliere quello che funziona per loro. Quindi lo skill piu' apprezzato è la flessibilità e rapido adattamento. A volte un semplice uso di buon senso nell'organizzazione del lavoro, che probabilmente anche corrisponde a qualche passaggio di metodi standardizzati, porta comunque al risultato.

Tecnologie utilizzati devono essere scalabili sufficientemente per un'azienda di calibro come META. Quindi molti strumenti utilizzati sono frutto dell'azienda stessa che sono stati o sviluppati o personalizzati per soddisfare meglio I propri bisogni. Ad esempio viene utilizzata una versione sviluppata appositamente per META del sistema di version control basato su Mercurial, perché essa stessa oppure Git non hanno funzionalità sufficienti per la quantità di dati gestita.

Con un commento a riguardo viene anche spiegato che solitamente se le persone lavorano su qualche parte del codice, lo fanno su parti diverse del sistema. Questo è garantito dalla divisione efficiente di ruoli. Nonostante ciò potrebbero esserci dei conflitti che sono risolti dal sistema di versioning oppure, se questa risoluzione non è possibile, viene proposta la modifica manuale del codice.

La discussione sulle idee di features avviene principalmente nell'ambiente GSuite di Google. Non è stato possibile capire come viene garantita la riservatezza dei dati sensibili all'interno dell'ambiente di un'altra corporazione concorrente. Tutte le features, prima di essere implementate devono passare la verifica sulla la loro conformità con la legge.

I linguaggi più dominanti includono Hack (dialetto di PHP, sviluppato per la stessa ragione come version control), React JS (sviluppato sempre da Meta, adesso una delle più usate librerie di JS), Python, SQL. Altri linguaggi come Haskell e Ocaml sono riservati ai task molto specifici.

Gerardo Marsiglia:

Il seminario centrato sul tema "Application Lifecycle management and modernization" è stato presentato da Gerardo Marsiglia, Ingegnere del SW con 30 anni di esperienza nelle società multinazionali. La sua ampia

esperienza comprende il lavoro in veste di Architetto SW, Tecnico di prevendita e di Enterprise architect. Si occupa principalmente del ciclo di vita del SW, in particolare oggi orientato su una modalita' di approccio Agile e si occupa di problematiche infrastrutturali attorno all'applicazione, facendo quindi particolare attenzione ai requisiti non funzionali posti, introducendo I meccanismi interni ed esterni all'applicazione che permettono alle applicazioni di girare in un certo modo per garantire un certo livello di servizio.

Durante la presentazione siamo entrati in merito di tematiche relative al ciclo di vita del SW e infine di un tema molto sentito in questo periodo di trasformazione digitale legato alla modernizzazione di applicazioni. Si intende di capire di come beneficiare dall'inserimento delle nuove tecnologie e dei nuovi paradigmi, ad esempio legati all'utilizzo di intelligenza artificiale, machine learning ecc. Inoltre è stato affrontato il tema dell'evoluzione dell'approccio allo sviluppo del software in termini organizzativi ed operativi.

Introduzione ai ruoli e professioni disponibili nell'ambito dell'Ingegneria del SW:

- Specialista - qualcuno con una conoscenza abbastanza specifica ed approfondita su un certo ambito o tecnologia

- Architetto - qualcuno che riesce ad avere una visione a 360 Gradi degli aspetti architetturali dell'applicazione, ovvero la struttura in termini di componenti insieme ai meccanismi destinati a soddisfare I requisiti non funzionali

- Consulente - qualcuno con una conoscenza più approfondita del dominio di business dell'azienda oltre ad avere una visione tecnica generale. Aiuta a comprendere ai tecnici di comprendere il business dell'azienda.

- Ingegnere - assimilabile ad architetto

- Manager - figura che riesce a guidare il team tecnico

SW e' il meccanismo che supporta il business delle aziende. Il SW e' evoluto anche dal punto di vista dell'infrastruttura che lo ospita, perche' per garantire certi livelli di servizio. Domini architetturali di un'organizzazione, di cui bisogna tenere traccia per lo sviluppo corretto del SW sono molteplici. In questo caso è la responsabilità di Enterprise architect, una possibile evoluzione del ruolo dell'Ingegnere del SW.

Principalmente si occupa del livello Application Architect, ma è altrettanto importante anche il livello soprastante dei Dati, perché le applicazioni funzionano grazie all'utilizzo dei dati. Di certo non è trascurabile anche il livello Tecnologico, che sta alla base della piramide, perché senza il supporto della tecnologia sottostante l'applicazione non puó girare, e quindi certi aspetti tecnologici vengono considerati ancora prima di sviluppare l'applicazione, per garantirne corretto funzionamento. Ad esempio in caso dell'applicazione distribuita, è fondamentale garantire la comunicazione tra I componenti che risiedono nei posti diversi, incluse le problematiche di trasferimento o sicurezza dei dati. E infine viene preso in considerazione il livello di Businesss, per individuare come supportare e rappresentare I processi necessari all'interno dell'applicazione.

Entrando nel merito del ciclo di vita del SW, ogni disciplina dello sviluppo del SW avrà il suo peso e modo di organizzazione a seconda della tipologia del progetto. Quelle discipline relative al ciclo di vita sono Pianificazione, Raccolta Requisiti, Analisi dei Requisiti/Design dell'architettura, Codice, Test, Deployment,

Manutenzione. Ultimamente le tendenze impongono che lo sviluppo del SW avvenga in stetto contatto con gli aspetti infrastrutturali per evitare I problemi nella fase di deployment. Perché con 50% di probabilità accade che una volta l'applicazione sviluppata, non è possibile farla funzionare una volta installata all'interno dell'infrastruttura. A questo punto c'è bisogno di ulteriori passaggi di ritorno al processo dello sviluppo. Altre problematiche che ci possono essere nella gestione di progetti di una certa complessità includono Team, distribuzione geografica, tracciamento di attività, standard architetturali (importante, perché garantisce beneficio sulla manutenzione), tecnologie, collaborazione efficace e strutturata.

Nella loro evoluzione i processi e le pratiche dello sviluppo del SW quindi sono passate dall'approccio tradizionale, come ad esempio Waterfall, nel quale dei problemi ci si poteva accorgere soltanto al passaggio di testing, alle pratiche iterative ed incrementali, spesso use-case driven. In seguito l'attenzione è passata alle tecniche Agile, che si concentrano su iterazioni progettuali molto piccoli, e si parla di Continuos Integration del codice attraverso tool appositi e test continui incluso il collaudo finale che coinvolgono l'utente finale. Il passo successivo, chiamato DevOps, aggiunge una serie di pratiche ai principi di Agile, che consentono di rendere il più fluido possibile tutto il processo di delivery del SW. Tra I principali citati sono Automazione e Collaborazione.

Esistono una elevata quantità di tool che supportano tutte le attivita' del processo di delivery. La scelta è influenzata da vari aspetti come tipologia del progetto, tipo di tecnologia sui cui si lavora ecc.

Il bisogno di modernizzare le applicazioni tradizionali è dovuto alla possibilità di trarre benefici dalle nuove architetture SW, nuove tecnologie, nuove piattaforme e le potenzialità del cloud.

Da una parte bisogna rinnovare il landscape applicativo, ma dall'altra anche l'infrastruttura che è a supporto di quell'applicazione.

Evoluzione dell'architettura SW:

Le applicazioni sviluppate in maniera monolitica sono difficili da mantenere in caso di problemi. Questo è dovuto al fatto che tanti oggetti sono tra di loro molto correlati ma sono divisi in pochi layer di grandi dimensioni che sono solitamente tre: GUI Layer, Business Layer e Data layer.

Una struttura applicativa migliore riguarda l'approccio a componenti, dove l'applicazione è un insieme di componenti, dove questi rispettano una certa serie di caratteristiche e parlano tra di loro attraverso un'interfaccia ben definita. Poi vengono strutturati in modo da costituire le funzionalità necessarie.

Con l'arrivo dei cloud e bisogno delle applicazioni cloud-native si passa all'architettura a Microservizi, dove un componente viene visto come un servizio molto specifico, ma che si porta dietro anche I dati necessari. In questo modo ogni componente è autoconsistente.

Seminario IBM:

IBM, è un'azienda statunitense fondata nel 1911 che si occupa del settore informatico è una delle aziende più importanti al mondo di questo settore. Produce e commercializza hardware, software per computer,

middleware e servizi informatici, offrendo infrastrutture, servizi di hosting, cloud computing, intelligenza artificiale, computazione quantistica e consulenza nel settore informatico e strategico.

In questo seminario il Sig. Ferdinando Gorga (Public Cloud Technical Specialist presso IBM Technology Sales Italy) ci presenta uno dei servizi che l'azienda propone il Cloud IBM.

Il Cloud IBM è una piattaforma distribuita che offre sia potenza computazionale che tecnologia software che permette a chiunque la possibilità di modernizzare i suoi sistemi software o per la creazione di nuovi sistemi.

IBM offre potenza computazionale in diversi modi per i vari utilizzi in cui si può applicare:

- Bare Metal
- Virtual Machines
- Containers
- Serverless Code Engine

Il Cloud IBM offre un ambiente sicuro per eseguire software, proteggendo dalle minacce informatiche. Utilizza crittografia avanzata e integra tecnologie all'avanguardia come Intelligenza Artificiale e Blockchain. Favorisce la crescita delle imprese basate sul software e promuove pratiche informatiche sostenibili. Gli studenti possono hanno la possibilità di sperimentare i servizi gratuitamente.

Seminario u-hopper:

In questo seminario il CEO dell'azienda u-hopper Daniele Miorandi inizia spiegando il suo percorso formativo e continua spiegando lo scopo dell'azienda u-hopper. U-hopper è una deep-tech company che ha lo scopo di sviluppare strumenti per l'analisi di dati con l'AI. L'azienda nel 2022 insieme alle aziende Delta Informatica, Thread Solutions e Dimensions fonda il consorzio Trentino.ai con lo scopo di condividere competenze per lo sviluppo di soluzioni nel campo Big Data & AI. Nella seconda parte del seminario il sig. Miorandi procede ad esporre le tecnologie che da loro vengono utilizzate, come i sistemi che loro utilizzano per la creazione di sistemi per la gestione di dati. Successivamente a spiegato la forte utilità di avere un sistema di logging per rendere il proprio lavoro più longevo. Oltre a queste due cose ci ha parlato dei linguaggi di programmazione che loro usano (90% python, 5% scala e 5% go).

Successivamente ci ha esposto il metodo di sviluppo chiamato "Sprint" e ci ha spiegato come funziona e i vantaggi, questo sistema ha uno sviluppo circolare diviso in 3 parti, 1 parte di codice seconda parte di Testing del codice (la fase di testing occupa circa il 70% del progetto) e la parte di revisione.

Nella parte finale della presentazione ci ha parlato dei tool che si usano per lo sviluppo, facendo confronti per accentuare vantaggi e svantaggi di ognuno, si è soffermato più volte sull'importanza dell'utilizzo dell'AI per tutti i capi di sviluppo.

Seminario Microsoft:

Microsoft è una multinazionale statunitense fondata nel 1975 da Bill Gates e Paul Allen. In questo seminario il sig. Diego Colombo ci parla della fase di testing del codice e della sua importanza. Il seminario inizia con

una definizione di testing: il testing è una verifica dei requisiti funzionali e non funzionali del software che può essere automatica o non automatica, in questo seminario il sig. Colombo ci mostrerà come eseguire questo testing per la risoluzione di Bug, a cui stesso viene data una definizione cioè “un comportamento inaspettato del codice”. Successivamente ci viene esposta una lista di test e vengono esposte brevemente le definizioni degli stessi:

- unit testing: questo testing ha lo scopo di collaudare ogni singola unità del software
- funcional testing: questo testing ha lo scopo di verificare se il software corrisponde alla sua progettazione.
- integraton testing: questo testing ha lo scopo di testare l'intero modulo software oppure, se

costituito da più moduli software, questi vengono combinati e quindi testati come gruppo.

- acceptance testing: questo testing chiamato anche collaudo ha lo scopo della verifica finale del progetto prima della consegna
- regession testing: questo testing ha lo scopo di garantire che il software sviluppato e testato in precedenza funzioni ancora come previsto dopo una modifica
- check-in testing
- smoke testing: è un test preliminare per verificare l’integrità per rivelare semplici errori
- stress testing: è un test in cui si spinge al limite il software per verificare l’integrita del codice anche in condizioni non normali.
- chaos testing: è il testing per verificare la capacita del software di tollerare i guasti e per verificare la resilienza del codice.

successivamente ci ha mostrato degli esempi in typescript per farci capire come funzionano questi tipi di testing anche utilizzando dei tool specifici, tra questi ci ha presentato Wallaby che permette di ricevere dei feedback per Javascript.