Money Expense

Report finale del progetto

Scopo del documento

In questo documento viene presentato il report conclusivo del progetto portato avanti dal gruppo **G09** durante il corso di Ingegneria del Software 2023/2024.
Il documento inizia con una serie di brevi descrizioni e riflessioni relative ai vari seminari seguiti durante il corso.
Segue la parte in cui viene articolato come il lavoro durante il corso è stato diviso ed organizzato tra i membri del team.
A seguire, presentiamo il tempo che ogni singolo componente del team ha dedicato allo sviluppo del progetto.
In quanto le ore di lavoro sono state suddivise in base alle attività, sarà quindi possibile evincere chi ha lavorato maggiormente a quale documento e le eventuali motivazioni.
Verrà introdotto successivamente un elenco delle criticità riscontrate ed eventualmente i mezzi utilizzati per risolverle.
In conclusione, presenteremo un'autovalutazione indicativa relativamente al lavoro svolto sia come gruppo che come singolo membro.

Approcci all'ingegneria del Software

Bluetensor
Durante il primo seminario, l'azienda ha illustrato il processo di sviluppo delle loro soluzioni nel campo dell'intelligenza artificiale. Hanno delineato una tipica giornata lavorativa e la struttura del loro approccio.

In particolare, l'azienda adotta un approccio fortemente orientato all'Agile poiché ogni progetto è unico e richiede un feedback continuo con il cliente.

L'attenzione è focalizzata inizialmente su uno studio di fattibilità del progetto, fondamentale per evitare perdite di tempo significative e progetti infruttuosi. Successivamente, redigono un documento dettagliato contenente gli obiettivi da raggiungere, prima di procedere con l'effettiva implementazione della soluzione richiesta.

Durante la fase di sviluppo, vengono effettuati numerosi rilasci per consentire al cliente di valutare se ciò che viene sviluppato corrisponde alle aspettative. Questo approccio è stato apprezzato poiché consente di individuare rapidamente eventuali incongruenze tra la richiesta del cliente e l'offerta, correggerle e raggiungere il risultato desiderato.

Complessivamente, l'approccio è stato considerato un eccellente connubio tra praticità e formalità.

Laboratorio Kanban
Durante questo seminario, è stato presentato un approccio ampiamente diffuso nell'ingegneria del software: il metodo Kanban. Questo metodo prevede una pool di micro-task che possono essere assegnate, messe in pausa e completate dai vari membri del team.

Ciascuna di queste task è visualizzata su una lavagna partizionata in sezioni in cui si mostrano le tasks non ancora assegnate, quelle attualmente in elaborazione e quelle da completare. Ogni lavoratore ha un insieme di task a cui può dedicarsi. Il numero di task per ogni lavoratore risulta variabile secondo una formula ben definita che facilita non solo il ricircolo delle task ma dimostra anche una maggiore produttività effettiva.

Abbiamo ritenuto questa metodologia particolarmente efficace, nonostante il suo approccio "artigianale" (si fa uso di biglietti fisici). Segnando le task come completate, si può aumentare la soddisfazione individuale dei lavoratori, contribuendo potenzialmente al successo complessivo del progetto.

È stato dimostrato che questo metodo incrementa la produttività di coloro che lo adottano. Tuttavia, abbiamo riscontrato un problema, ossia la sua elevata compartimentazione, che potrebbe limitare la flessibilità. È possibile che questa percezione sia solo esterna e che, una volta che la metodologia viene attivamente utilizzata all'interno del progetto, questa limitazione possa attenuarsi.

IBM
Il protagonista di questo seminario è stata IBM, un'azienda notevolmente diversa in termini di longevità e dimensioni rispetto a quella precedentemente analizzata. In particolare, l'attenzione del seminario si è concentrata sul dipartimento Cloud e sulle soluzioni che questo settore è in grado di offrire.

Durante la presentazione, è stato illustrato un esempio di progetto seguito da una demo. È emerso un primo contrasto con BlueTensor poiché i progetti presentati da IBM sembravano seguire una struttura simile a quella a cascata (waterfall). Questa osservazione ha portato alla consapevolezza che non esiste una metodologia universalmente perfetta per ogni progetto, bensì una soluzione ottimale e adattabile a ogni caso specifico.

È stato evidenziato che IBM fa un ampio uso di documenti, artefatti e diagrammi, molti dei quali sviluppati attraverso strumenti interni. Nonostante questo approccio si sia rivelato funzionale, è stato notato che potrebbe lasciare meno spazio per eventuali correzioni e presentare sfide maggiori rispetto al metodo proposto nel seminario precedente.

Meta
La presentazione ha avuto inizio sottolineando che a Meta vengono utilizzati diversi linguaggi di programmazione e che la focalizzazione principale non è sulla tecnologia stessa, bensì sul software risultante e sulla sua qualità. Il relatore ha specificato che a Meta non viene adottata una metodologia specifica, bensì un approccio definito "di buon senso", che si avvicina all'Agile ma è più artigianale e si adatta più facilmente alle esigenze specifiche dei singoli team. Questo approccio mira a garantire massima libertà di espressione a tutti i contributors al progetto, evitando al contempo una burocrazia superflua.

Pur non seguendo una struttura rigidamente definita, sono comunque presenti documenti condivisi e canali di comunicazione dedicati, ritenuti essenziali per evitare la dispersione delle energie in progetti multipli quando sarebbe opportuno concentrarsi su uno. Questo approccio è stato descritto come snello, con l'idea che, se tutti i membri del team sono impegnati seriamente, può portare a risultati eccellenti.
Tuttavia, è stato anche sottolineato che un approccio più strutturato potrebbe essere vantaggioso nel caso in cui alcuni membri del team non si impegnino completamente.

U-Hopper
Durante questo seminario, è stata presentata un'azienda di dimensioni più locali, ma nonostante ciò, la sessione è stata estremamente interessante. Dopo una breve introduzione sulla compagnia, le tecnologie utilizzate e i linguaggi di programmazione, sono stati esaminati diversi aspetti delle metodologie di ingegneria del software.

In particolare, è stato evidenziato come l'azienda segua un approccio Agile, lavorando per *sprint* di due settimane, durante i quali vengono affrontate micro-task che contribuiscono alla realizzazione di un progetto più ampio. L'uso di *git* e dei *branch* è stato sottolineato come elemento fondamentale per garantire una struttura ottimale al

progetto complessivo. Considerando la natura di consulenza dell'azienda e la gestione di progetti, è stata posta un'attenzione particolare sul feedback loop, sia interno che esterno, coinvolgendo così il cliente in tutto il processo.

Sono stati illustrati dettagliatamente i documenti e gli artefatti prodotti, inclusi test, la documentazione iniziale e in corso d'opera, l'importanza delle revisioni e il rilascio del progetto completato. Sebbene il processo sembrasse molto articolato, è emersa una certa perplessità riguardo all'ampia quantità di artefatti e burocrazia, che è sembrata eccessiva per l'azienda. Tuttavia, il relatore ha spiegato che, dato il contesto di clienti esterni spesso non esperti di tecnologia, tale approccio è necessario.

Alla fine, sembrava essere la soluzione migliore per il team e per la tipologia di progetti affrontati. È stato inoltre enfatizzato l'importante ruolo degli strumenti condivisi da tutto il team, al fine di garantire workflow comuni e fornire un supporto reciproco, elementi fondamentali in un ambiente di lavoro collaborativo.

RedHat
Questo seminario ha posto particolare enfasi sul mondo open source e ha fornito anche diverse considerazioni sulla sfera dell'ingegneria del software.
Oltre a delineare i concetti fondamentali dell'open source e le possibilità di guadagno, sono state illustrate anche le dinamiche dello sviluppo di un progetto open source.

Data la presenza di numerosi contributori, sparsi geograficamente e spesso non remunerati, il successo di un progetto open source richiede una documentazione estremamente ben strutturata, condivisa e facilmente comprensibile.
Inoltre, ogni singola attività deve essere accuratamente documentata, indicando chiaramente il suo stato attuale. È essenziale che se un partecipante incontra e risolve un problema, lo renda noto, permettendo ai futuri contributori di beneficiare della soluzione senza dover dedicare eccessivo tempo alla ricerca di un approccio diverso.
Le metodologie presentate potrebbero sembrare fortemente burocratiche e non facilmente applicabili a un team come quello coinvolto nel progetto da noi sviluppato, ma risultano adatte per progetti open source.

Va sottolineato che alcuni di questi progetti sono gestiti da vere e proprie aziende, comportando un livello di apertura variabile e decisioni più centralizzate. Pertanto, ogni caso nel mondo open source va valutato individualmente.

Microsoft
Il nucleo di questo seminario ha riguardato il testing del software e la sua rilevanza, sottolineando la diversità di approcci.

Nell'ambito di Microsoft, in quanto azienda di grandi dimensioni, è necessario garantire che i servizi siano sottoposti a test sufficientemente robusti, considerando anche il traffico che attraggono e la loro importanza.
Il testing diventa quindi un elemento integrante e cruciale nell'ambito dell'ingegneria del software e assicura il successo dei progetti.

È stato evidenziato che anche in Microsoft esistono diverse metodologie, adatte ai vari team e, di conseguenza, ai diversi tipi di prodotti che gestiscono, ma tutte generalmente ispirate all'approccio Agile, organizzato in *sprint*.
Tale scelta è sembrata pragmatica, considerando le dimensioni dell'azienda e il coinvolgimento di numerosi *stakeholder* in progetti di varia complessità.
L'unico piccolo svantaggio menzionato è la burocrazia considerevole, che potrebbe richiedere più tempo per la redazione di documenti rispetto allo sviluppo effettivo del codice. Tuttavia, questa burocrazia è ritenuta necessaria per coordinare un'ampia forza lavoro impegnata in progetti di tale portata.

Molinari
In questo seminario, abbiamo ottenuto una panoramica su come sia lavorare con sistemi legacy e quanto essi siano ancora rilevanti nell'attuale panorama tecnologico. Tuttavia, gestire questi sistemi è tutt'altro che agevole, poiché molte delle tecnologie utilizzate non sono più all'avanguardia e molti sviluppatori specializzati in tali linguaggi non sono più attivi. Inoltre, il lavoro su sistemi legacy è spesso caratterizzato da un costante rinnovamento e raramente da innovazione.

Dal punto di vista dell'ingegneria del software, il relatore ci ha spiegato che lavorare con sistemi legacy è diverso rispetto al progetto che stiamo sviluppando, in quanto l'approccio Agile difficilmente può essere applicato. Le figure professionali coinvolte sono spesso nettamente distinte e la sincronizzazione tra di esse è di scarsa utilità, poiché le modifiche avvengono su componenti che non influiscono significativamente sull'intero sistema.
Inoltre, i sistemi legacy tendono ad essere accompagnati da documentazioni estese, il che rende cruciale l'adattamento delle metodologie al contesto specifico del progetto, senza ottenere così un approccio universale.
In generale, il tempo richiesto per lavorare su questi sistemi è spesso superiore rispetto ai sistemi non legacy.

Una costante tra lo sviluppo più tradizionale e quello legacy è l'ampio utilizzo dei diagrammi UML come forma di documentazione.
Sebbene non abbiamo potuto sperimentare direttamente queste metodologie durante il corso, riteniamo che in futuro, anche se solo in minima parte, potremmo trovarci ad affrontarle. Pertanto, abbiamo trovato il seminario molto utile in questo contesto.

Marsiglia
Questo seminario ha concentrato la sua attenzione sulle metodologie e sulla storia dell'ingegneria del software, offrendo un'analisi molto interessante.

È iniziato con un'esplorazione dei metodi più tradizionali, come il modello a cascata (waterfall), per poi passare all'Agile, al DevOps e infine alla continuous delivery. Durante la presentazione, sono stati illustrati i cambiamenti che queste diverse metodologie subiscono e come si adattano a seconda dell'ambiente in cui vengono applicate. È stato

evidenziato che la metodologia deve essere adattata in base a vari fattori, tra cui la geografia, il team, le tecnologie e i progetti.

L'excursus è proseguito con un'analisi delle tecnologie *cloud*, spiegando perché sono rivoluzionarie e come stiano rendendo sempre più predominante il DevOps come metodologia nello sviluppo del software. Questo approccio si adatta bene a progetti che richiedono una rapida messa in produzione, fasi di testing e integrazione continua. Complessivamente, il seminario si è rivelato interessante sia per la sua trattazione delle metodologie allo sviluppo del software che per l'approfondimento sulle tecnologie cloud.

APSS e Trentino.ai
Questo seminario ha posto un forte accento sull'importanza del dipartimento tecnologie all'interno di un'azienda e su come tale influenza possa plasmare l'organizzazione aziendale.
La concezione di uno sviluppo software interno non si limita più all'analisi funzionale, ma abbraccia un approccio di co-design che integra completamente gli aspetti tecnologici e funzionali. Questo sottolinea la rilevanza dell'organizzazione nel corso dello sviluppo software. Si mira ad ottimizzare l'efficienza, favorire la collaborazione tra i diversi reparti aziendali, massimizzare l'utilizzo delle risorse disponibili e garantire l'implementazione corretta delle best practices. In tal modo, si contribuisce al raggiungimento degli obiettivi aziendali globali e alla creazione di soluzioni software che rispondono appieno alle specifiche esigenze di ogni settore interno.

Un altro tema cruciale emerso è stato la sicurezza dei dati e la loro categorizzazione, particolarmente significativa quando si sviluppa un software che gestisce informazioni conformemente alle politiche in vigore.
Nell'ultima parte del seminario è stato affrontato l'argomento dell'intelligenza artificiale, evidenziando il suo notevole contributo nel settore sanitario. Tuttavia, è stato sottolineato che queste opportunità non sono limitate a pochi settori, ma possono trovare applicazione in qualsiasi contesto aziendale.

In definitiva, il seminario, pur trattando argomenti specifici del settore sanitario, ha fornito conoscenze che possono essere applicate in qualsiasi altro settore aziendale.

Il nostro approccio, in relazione a ciò che abbiamo imparato
Durante il corso del progetto e dei diversi seminari, abbiamo acquisito la consapevolezza che documenti, artefatti e documentazione sono fondamentali per la riuscita di ogni progetto. La comunicazione e la sincronizzazione sono elementi chiave per evitare disastri e inefficienze. Tuttavia, è importante sottolineare che non esiste una quantità ideale di questi elementi, poiché variano notevolmente in base al team, al progetto e al cliente finale. Ogni situazione richiede una miscela ottimale, e il continuo sforzo è volto a migliorarla, come abbiamo cercato di fare anche noi.

Abbiamo posto una particolare enfasi sulla sincronizzazione durante il lavoro, adottando feedback frequenti e revisioni continue del lavoro dei colleghi. Abbiamo evitato la produzione di documenti superflui, concentrandoci solo su quelli necessari per il

progetto, al fine di evitare un eccessivo appesantimento burocratico e mantenere un approccio agile. Nel complesso, la nostra metodologia è stata una via di mezzo tra quella presentata in BlueTensor e in Meta. Probabilmente il fatto che fossero tra i primi a trattare tali approcci ha influenzato la nostra scelta.

In definitiva, la nostra metodologia si è evoluta nel tempo con vari adattamenti, riconoscendo la necessità di flessibilità a lungo termine e di adattamento continuo. Riteniamo quindi che questa flessibilità sia fondamentale per ottenere un successo duraturo.

Organizzazione del lavoro

Per quanto riguarda la pianificazione del lavoro abbiamo deciso di essere abbastanza elastici. In particolare, non solo abbiamo cercato di venire incontro alle esigenze extra universitarie di ognuno di noi, ma si è anche data la possibilità di lavorare maggiormente nelle sezioni del progetto in cui il singolo risultava più competente e maggiormente interessato.

Si è seguito l'approccio per cui all'aggiunta di nuovi argomenti del corso, venivano aggiustate adeguatamente anche le sezioni dei deliverables.

Per quanto riguarda la spartizione del lavoro, si è deciso di fare uso di Asana, un software utilizzato a livello lavorativo per la fase di task management. Il lavoro veniva quindi suddiviso in dei piccoli task visibili su una dashboard da tutti i membri del gruppo. Ogni volta che un membro iniziava una nuova sezione, lo comunicava via chat agli altri membri del gruppo.
Per essere più concreti, portiamo un esempio: se il task consisteva nello sviluppare il diagramma delle classi, veniva aggiunto alla dashboard di Asana da un membro del gruppo e si avvertiva il resto del team dell'inizio dello svolgimento della task in modo da evitare lavoro duplicato; la task veniva completata e in questo modo il progetto subiva continue evoluzioni.

Relativamente allo sviluppo del codice, si è deciso di spartire il lavoro tra frontend e backend. La motivazione principale consisteva nel fatto che due membri del gruppo presentavano maggiore manualità nello sviluppo della UI e integrazione con il backend rispetto al terzo, che aveva invece più esperienza nello sviluppo di REST API.

Le attività lavorative si sono svolte principalmente in modalità individuale.
Molte delle attività teoriche svolte non richiedevano una collaborazione attiva, ma solamente di rimanere coordinati nella creazione di ogni singola componente e sulla revisione finale.
Gli aggiornamenti all'interno del gruppo erano molto frequenti e avvenivano sia in modalità offline che online, via Asana e gruppo Telegram. Questo ha fatto sì che il lavoro fosse sempre sincronizzato e aggiornato.

Allo scadere della consegna di ogni documento provvedevamo a revisionare l'intero lavoro svolto in modo da assicurarci la coerenza non solo della struttura interna dei singoli deliverable ma soprattutto dell'intero progetto.

Ci siamo sempre organizzati in modo da avere come minimo uno studente che fosse presente a lezione. Le lezioni erano frequentate sia dal vivo che in remoto e molto spesso da tutti i membri del gruppo. Sfruttavamo a quel punto l'occasione per aggiornarci e per verificare la disponibilità della singola persona per future lezioni.

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

- Google docs, abbiamo adottato questo strumento perché permette una veloce e intuitiva sincronizzazione e aggiornamento del lavoro insieme alla possibilità di inserire commenti ai membri del gruppo.
- Asana, questo tool è stato utilizzato principalmente come strumento di task management e per segnare le ore lavorative del singolo membro.
- Git/GitHub, ci hanno aiutato ad instaurare un ottimo workflow grazie alla funzionalità dei commit e al processo di CI/CD. Grazie a questi due tool si gestiva la parte relativa al codice e la memorizzazione dei deliverable.
- Figma, si è rivelato molto utile come strumento di progettazione del frontend dell'applicazione.
- LucidChart e FlowMap, ci siamo serviti di questi software per la realizzazione dei diversi diagrammi richiesti nei deliverable.
- Telegram, da non dimenticare questo strumento chat che ci ha permesso di comunicare efficacemente tra noi e di mantenerci aggiornati.

Ruoli ed attività

Riportiamo a questo punto i ruoli che ognuno di noi ha assunto durante lo sviluppo del progetto **Money Expense**.
Vorremmo far notare come la divisione dei compiti per lo sviluppo risulti particolarmente marcata, mentre per i vari documenti è stata assai più labile e flessibile.
Quindi di base ognuno ha avuto la possibilità di fornire il proprio contributo ad ogni documento, grande o piccolo che sia.

Componente del team	**Ruolo**	**Principale attività**
Ilya Emeliyanov	Project leader, analista requisiti funzionali e non, sviluppatore frontend	1. Si è occupato di coordinare il lavoro del gruppo, marcando lacune, inadeguatezze e incongruenze nei singoli componenti. Ha lavorato principalmente sul D2, D3 e D4.

		2. Ha provveduto a realizzare buona parte del frontend. 3. Ha contribuito a stendere e sistemare le sezioni testuali dei singoli documenti. 4. Si è occupato della progettazione dei design mockup con Figma.
Ivano Lu	Analista requisiti funzionali e non, sviluppatore frontend	1. Si è occupato della progettazione dei design mockup con Figma. 2. Ha realizzato parte del frontend. 3. Ha aiutato a stendere principalmente le sezioni dei documenti sul frontend.
Matteo Possamai	Analista requisiti funzionali e non, sviluppatore backend	1. Ha contribuito principalmente al D3, D4 e al D5. 2. Ha fornito la maggior parte dei diagrammi presenti in questi documenti, e in parte di quelli del D2. 3. Ha sviluppato il backend della piattaforma e la sua documentazione. 4. Creato la documentazione e il testing.

Carico e distribuzione del lavoro

Riportiamo ora in tabella la distribuzione del lavoro per il progetto in maniera più deterministica, mediante le ore che abbiamo dedicato alla produzione dei vari documenti e del codice.

	D1	**D2**	**D3**	**D4**	**D5**	**TOTALE**
Ilya Emeliyanov	17	29	17	69	2	134
Ivano Lu	21	19	13	38	3	94
Matteo Possamai	15	25	19	65	4	128
Totale	53	73	49	172	9	356

Criticità

Nel gruppo non sono state riscontrate notevoli criticità che abbiano rischiato di compromettere la realizzazione del progetto.
È giusto però annotare che ci sono solo state frizioni minori che mano a mano abbiamo appianato in modo da poter raggiungere il risultato migliore possibile.
Il problema iniziale che abbiamo riscontrato consisteva nel fatto che tutti i membri del gruppo lavoravano su tutto e non vi era un'efficace spartizione del lavoro. Questo, come si può intuire, ha portato ad un iniziale calo di produttività.
Perciò, in modo da ovviare al problema abbiamo deciso di dividerci i vari task a priori, imponendo un forte focus su una sincronizzazione "asincrona" (via messaggi, asana, commenti google docs) estremamente costante.

Questo ha permesso anche a chi avesse altri impegni personali di avere maggiore flessibilità e meno fiato sul collo.
Ognuno così si sentiva più libero e poteva quindi decidere se lavorare più intensamente ad un deliverable ed eventualmente rallentare nella stesura di un altro.
Questo approccio ha permesso ad ognuno di noi di mettere mano in almeno una parte di ogni deliverable, anche se in maniera diversa.

Ilya ha partecipato maggiormente alla realizzazione dei documenti D2,D3 e D4, si è occupato di stendere diverse sezioni del D1 e D5, ma in misura più ristretta. Ha svolto il revisionamento dei diversi deliverable e ha partecipato alla realizzazione del frontend dell'applicazione.
Ivano ha contribuito alla stesura del D1, D2 e D4. Si è occupato della realizzazione di buona parte dei mockup presenti nel progetto e parte del frontend.
Matteo si è concentrato prevalentemente nei D3, D4 e D5, e ha fornito anche buona parte dei diagrammi del D2. Ha scritto diversi paragrafi anche in D1 e D2, anche se in relazione alla sua contribuzione agli altri deliverable, qui risulta minore. Si è infine occupato della realizzazione del backend.

Per quanto riguarda lo sviluppo, la divisione marcata del lavoro ha permesso ad ognuno di noi di dare il massimo nell'ambito la cui expertise era maggiore.
In poche parole si è attuata una politica di "trust" per cui, chi si trovava a gestire il backend doveva solo preoccuparsi dello sviluppo del backend, dando per scontato il corretto funzionamento del frontend, e viceversa.
Ovviamente, prima dello sviluppo abbiamo concordato delle regole intra progettuali da seguire e le API comuni. Una criticità individuata è stata la necessità di modificare poi queste specifiche, ma in puro stile Agile. Siamo stati molto veloci ad aggiustare il tiro, e a permettere l'uscita del risultato migliore possibile.
Un'altra piccola criticità affrontata consisteva nella carenza di conoscenze pregresse per quanto riguarda alcune delle tecnologie proposte, come ad esempio MongoDB.
Nonostante ciò, la compartimentazione del lavoro, il feedback loop molto frequente e il costante lavoro, ci hanno permesso di superare questo limite.

Durante tutto il progetto, è stata mantenuta massima serietà e coordinazione. Abbiamo seguito le tempistiche e le scadenze per i singoli deliverable. Perciò siamo stati in grado di consegnare l'intero progetto per il primo appello senza grandi problemi. Abbiamo tutti investito molte ore in queste attività e ne abbiamo imparato molto.
Siamo più che sicuri che queste conoscenze ci torneranno utili nel mondo del lavoro e in caso di successivi progetti di ingegneria del software saremo in grado di applicare le conoscenze acquisite in questo corso.

Autovalutazione

Data la mole di lavoro che ognuno ha sostenuto per mettere in piedi questo progetto, abbiamo fatto le seguenti considerazioni.

Abbiamo tutti lavorato un numero considerevole di ore, e abbiamo avuto la possibilità di mettere mano a vari documenti. Ognuno ha avuto una sua contribuzione alla riuscita di tutto il progetto. Senza uno dei membri di questo team, questo progetto non avrebbe raggiunto il risultato attuale. Inoltre, avendo provato a mettere le mani su molte cose, possiamo dire di avere approfondito tutti le nostre conoscenze relative all'ingegneria del software, sia dal punto di vista puramente teorico, ma anche più pratico, a livello di come funzionano i diagrammi e le documentazioni.
Inoltre, abbiamo investito un considerevole ammontare di tempo per la parte di sviluppo dell'applicazione web che portiamo come prodotto di questo corso.
In conclusione, come gruppo pensiamo di aver fatto tutti un ottimo lavoro. L'unica discriminante che si può trovare è quella del numero di ore investite.

Di conseguenza, la nostra autovalutazione è la seguente:

PERSONA	VOTO
Ilya Emeliyanov	30
Ivano Lu	26
Matteo Possamai	30