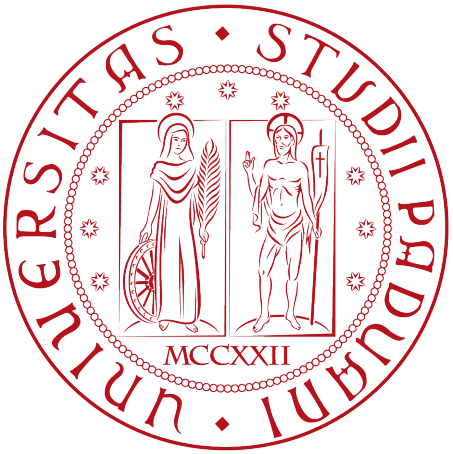
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Realizzazione a microservizi di una applicazione di process mining per il filtraggio degli eventi

Tesi di laurea triennale

Relatore

Prof.Tullio Vardanega

Laureando

Fontolan Carlo

Anno Accademico 2018-2019

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, dalla durata di circa trecentoquaranta ore, del laureando Carlo Fontolan presso l'azienda *Siav S.p.A.* Lo studente vuole descrivere con riflessioni critiche e oggettive quanto appreso e maturato dal punto di vista professionale e delle competenze in tutto l'arco di durata dello stage. In primo luogo era richiesto lo sviluppo di una libreria di filtraggio in abito process mining, che verrà poi inserità all'interno di una architettura a microservizi. In secondo luogo, è stata richiesta le riscrittura di alcune sezioni di un loro vecchio prodotto, in cui la libreria sopradescrittà farà da protagonista, per quanto riguarda la sezione di filtraggio. Le caratteristiche richieste dall'azienda proponente erano che il prodotto fosse idoneo a girare su cloud e che fosse costituito da un'architettura a microservizi e sviluppato con framework all'avanguardia. Siccome in corso d'opera sono sorte alcune criticità in merito allo sviluppo della libreria, in accordo con il tutor aziendale, abbiamo deciso di rinegoziare alcuni requisiti andando ad eliminare la sezione riguardante il *back-end*, facendo spazio ad uno stub in modo da poter osservare il comportamento dell'interfaccia, soddisfando così gli obiettivi sia a livello aziendale che personale.

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Tullio Vardanega, relatore della mia tesi, per l'aiuto, la pazienza ed il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare i miei genitori per il loro sostegno e per essermi stati vicini in ogni momento durante gli anni di studio non facendomi mai mancare il loro supporto.

Ringrazio i miei parenti più stretti che mi hanno sempre stimolato a dare il meglio di me

Desidero ringraziare infine i miei amici, ed i colleghi per tutti questi anni passati insieme e per tutte le esperienze vissute.

Padova, Dicembre 2019 Fontolan Carlo

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Il contesto aziendale

1.1 L'azienda

Siav è una delle più importanti realtà italiane di sviluppo software e di servizi informatici specializzata nella dematerializzazione e nella gestione documentale e nei processi digitali. Presenta diverse filiali nel suolo italiano che cooperano tra loro per un fine comune. Siav inoltre punta molto sulla collaborazione tra aziende, in campo nazionale sia a livello pubblico che privato, ma anche a livello internazionale. Una fra tutti Microsoft. I loro servizi puntano in primo luogo, ad una miglior gestione, controllo ed automazione di tutti i principali processi aziendali; in secondo luogo, ad offrire diverse soluzioni concrete anche a singoli privati che necessitano di sistemi di gestione ed integrazione per lo loro piccola realtà. Siav si colloca all'interno del mercato come una

Figura 1.1: I prodotti offerti da Siav (https://tinyurl.com/sm45xtu)

tra la più importanti realtà italiane a livello di *Enterprise Content Management* offrendo servizi per poter migliorare processi e gestioni aziendali spaziando da un gestore di caselle PEC, ad un applicativo di process mining denominato *Bipod*, fino ad arrivare al loro prodotto di punta: Archiflow/Silloge: un software in continuo sviluppo, mantenuto e aggiornato da piccoli team che lavorano in sinergia per raggiungere un obiettivo comune. Archiflow/Silloge offre una soluzione alla gestione di una cospicua mole di documenti, categorizzandoli in varie sezioni permettendone un facile reperimento.

1.2 Clientela rivolta a Siav

La maggior parte dei clienti che si affidano a *Siav* sono aziende che presentano il bisogno di automatizzarsi e migliorare i propri processi interni andando cosi ad incrementare la propria efficenza sotto l'aspetto lavorativo. Tali aziende sono di vario genere con diverse necessità, dal semplice ristorante ad una nota catena di supermercati fino ad aziende metalmeccaniche. Per ogni settore, l'azienda è in continua ricerca di nuove opportunità e soluzioni che possano potare a miglioramenti aziendali, cercando di spaziare verso più ambienti lavorativi, allargando sempre più il proprio campo applicativo per portare verso di sè una clientela più varia. Questo ha portato l'azienda a dover standardizzare i propri prodotti in modo da poter coinvolgere una maggior porzione di clientela, per ogni settore. Siav è catalogata come *Software house* e presenta un ampio catalogo di prodotti atti a soddisfare le principali necessità organizzative e gestionali di un'azienda, sta al potenziale cliente poi, valutarne l'acquisto in base alle proprie necessità.

1.3 Processi aziendali

In questa sezione sono descritti i pricipali processi e strumenti che ho visto utilizzare all'interno dell'azieda durante il mio periodo di permanenza, dividendoli in metodologie e strumenti.

1.3.1 Metodologie di sviluppo

L'azienda normalmente si ritrova a dover far fronte ad alcune problematiche derivate da bug, criticità o variazioni importanti di requisiti che possono essere riscontrate durante i vari processi aziendali ed incidere in modo significativo sull'andamento dello sviluppo. Per cercare di venire incontro a ciò l'azienda ha adottato una metodologia di tipo *Agile*, cercando di mantenere un atteggiamento flessibile rispetto ai processi e gli obiettivi. Risulta quindi fondamentale l'organizzazione di incontri durante vari momenti della giornata o settimana in modo da confrontarsi sull'andamento dei processi e lo stato di avanzamento del prodotto. È consuetudine dei team di lavoro un confronto giornaliero tramite *daily meeting*, per discutere su quanto fatto durante la giornata precedente indicando le eventuali criticità riscontrate, pianificando cosi gli obiettivi per la giornata odierna. Ad inizio settimana invece viene organizzato un *weekly meeting* per constatare gli stati di avanzamento rispetto alla settimana precedente, per poi fissare gli obiettivi per la settimana successiva. Gli obiettivi, le attività preposte e le scadenza prefissate a livello complessivo di team vengono solitamente appesi in una lavagna posta in luogo ben visibile all'interno del reparto di sviluppo, per ordine di priorità; in questo modo è quindi possibile tener sott'occhio le principali attività. Il rapporto con il cliente è di vitale importanza per *Siav*. L'azienda infatti presente un servizio di assistenza clienti tramite il quale vengono effettuate segnalazioni di *bug* o assistenza sui proprio prodotti

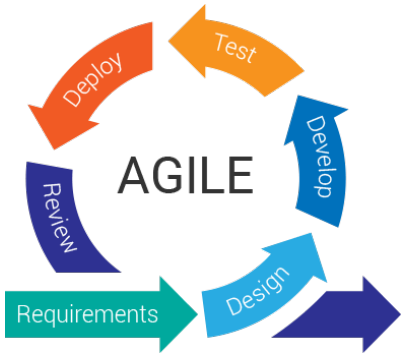


Figura 1.2: Ciclo di vita della metodologia Agile (`https://tinyurl.com/tkqyaww`)

1.3.2 Strumenti di supporto

L'azienda mette a disposizione diversi strumenti di supporto, allo scopo di gestire e tenere traccia nel migliore dei modi tutte le attività di progetto. Facendo fede alla metodologia *Agile*, la maggior parte degli strumenti servono a gestire tutti gli aspetti che riguardano il codice. Viene utilizzato TFS, uno strumento di *Microsoft* per la gestione delle principali attivtà di progetto che offre un set di strumenti per la collaborazione. Tra le principali funzionalità offerte sono presenti: gestione del repository tramite

tecnologie Git o TFVC, gestione dei requisiti e strumenti di *build* automatizzati. Tale strumento aderisce in maniera completa alle tecniche *agile*, andando quindi ad integrarsi in modo solido all'interno della realtà lavorativa. Per quanto riguarda il tracciamente delle attività viene utilizzato *Evernote*: un software che permette la scrittura e la condivisione di note all'interno di un gruppo di utenti. Cosi facendo ogni membro del team di sviluppo ha sotto controllo ogni attività svolta dagli altri membri. Un ulteriore strumento utilizzato per il tracciamento dello attività è *Google Docs*. Tramite quest'ultimo è possibile assegnare attività a tutti i membri che possiedono i privilegi per accedere al documento in questione. Solitamente i task assegnati presentano una struttura semplice: la data di creazione, la possibilità o meno di menzionare direttamente l'interessato a cui risulta assegnato il task ed un casella di risposta per poter descrivere la sua effettiva terminazione, oppure un messaggio di testo di altro genere, che solitamente indica le problematiche per cui il task non è stato possibile risolverlo. *Google Docs* non è l'unico strumento di *ticketing* che utilizza l'azienda ma è stato quello che ho potuto visionare durante il priodo di stage.
Per poter garantire una cooperazione efficente tra tutto il personale viene spesso utilizzato *TeamViewer*: un software per il controllo e la gestire altre macchine da remoto, in questo modo è possibile raggiungere un buon grado di cooperazione tra il personale dell'azienda, pur trovandosi in filiali diverse. Tale metodologia viene spesso utilizzata anche all'interno dell'area di *testing* per fornire assistenza immediata ai clienti.

1.4 Tecnologie utilizzate

Le tecnologie utilizzate dell'azienda per la realizzazione dei propri prodotti sono fondate principalmente sulla possibilità di un utilizzo tramite *Cloud*, dedicandosi principalmente sullo sviluppo di *Webapp*; dovendo fornire servizi di tipo gestionali e organizzativi l'azienda si trova spesso a dover far fronte alla gestione dei cosiddetti *BigData*: fattore crucuiale per *Siav* che interessa la maggior parte dei propri prodotti. Per far fronte a tali problemi l'azienda utilizzia tecnologie innovative e all'avanguardia. Quest'ultime possono essere raggruppate in due macrocategorie: *Front-end* e *Back-end*.

1.4.1 *Front-end*

Da quel che ho potuto constatare durante l'attività di stage, per quanto riguarda lo sviluppo web lato *client* viene utilizzata prevalentemente la piattaforma *Angular*. Quest'ultima è ideale per lo sviluppo di *webapp* multipiattaforma, offrendo diverse funzionalità allo sviluppatore, potendosi adattare a svariate tipologie di architetture. Tramite il consistente numero di componenti prefabbricati presenti in rete e una vasta gamma di funzionalità e servizi disponibili, è possibile sviluppare un'interfaccia *front-end* solida e di qualità, soddisfando nel miglior modo possibile le aspettative del cliente. Un'altro aspetto fondmentale di tale *Framework* è la possibilità di essere utilizzato su più dispositivi, non restringendo quindi il campo applicativo di *Siav* al solo *Desktop o Laptop* ma spaziando verso tutti principali dispositivi presenti sul mercato.

1.4.2 *Back-end*

Per quanto riguarda lato back-end vengono utilizzate molteplici tecnologie in base alle necessità che prevede il software di sviluppo; *RabbitMQ*: questa tecnologia è stata pensata principalmente per un'architettura a microservizi e viene utilizzata in

più prodotti all'interno dell'azienda. *RabbitMQ* viene definito come un *Broker* di messaggisticia: ossia un sistema che monitora la trasmissioni di messaggi tra servizi attraverso una coda in cui vengono memorizzate i messaggi prima di essere inviati. Un'altro sistema utilizzato, restando in ambito di microservizi è *Kubernetes*: un sistema per la gestione di *container* che viene utilizzato dall'azienda per effettuare la duplicazione di contenitori in modo da far fronte ad una cospicua mole di operazioni e richieste da parte di utenti. Per poter sfruttare al massimo l'ambiente *Cloud*, su cui *Siav* investe molte risorse, viene utilizzato *Apache Kafka*: tale sistema è in grando di gestire un gran numero di operazioni in tempo reali provienienti da diversi *Client*, sia per quanto rigaurda la fase di lettura che scrittura. L'azienda quindi cura molto le tecnologie che utilizza all'interno del propri prodotti cercando di trarne il massimo vantaggio da ognuna di esse.

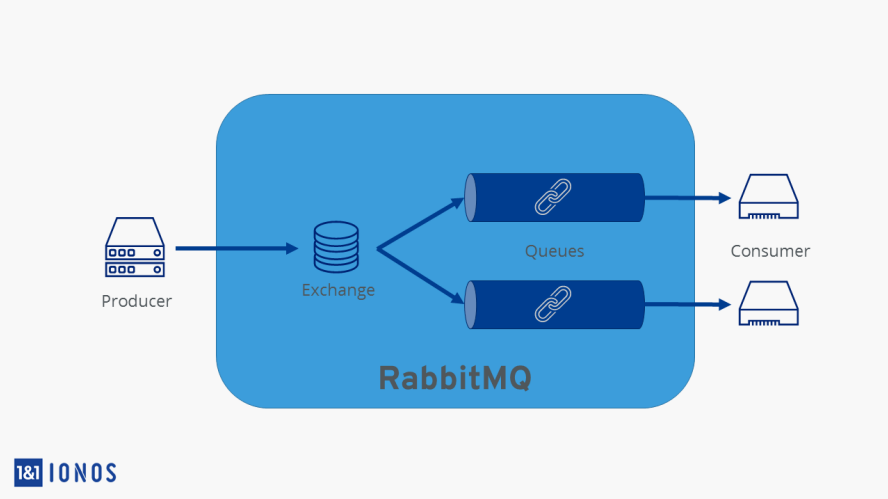


Figura 1.3: Diagramm illustrativo sul funzionamento di RabbitMQ (https://www.ionos.it/digitalguide/siti-web/programmazione-del-sito-web/rabbitmq/)

1.5 Propensione all'innovazione

Siav è un'azienda che fa dell'innovazione un suo punto cardine. I prodotti che offre sono costantemente aggiornati, cercando di adattarsi alle necessità del cliente. Per poter far ciò l'azienda ha messo a disposizione un serivizio di *helpdesk* in modo da fornire assistenza in merito ai propri prodotti per i propri clienti. L'azienda inoltre è alla continua ricerca di nuove tecnologie per poterle integrare all'interno dei propri applicativi; è presente all'interno dell'organico un team di ricerca e sviluppo che analizza, le varie piattaforme e tecnologie presenti sul mercato. Spesso e volentieri tali attività vengono contestualizzate all'interno di proposte di stage, in modo da poter osservare nel concreto se gli studi e le ricerche fatte in precedenza possano portare a benefici per i prodotti dell'azienda, facendo anche un'attenta analisi dei costi.

Capitolo 2

L'attività di stage all'interno della strategia aziendale

2.1 Vantaggi aziendali

Siav da molti anni è propensa a svolgere diverse attività di stage in collaborazione con l'Università di Padova. È in continua ricerca di nuove figure da poter inserire all'interno del loro organigramma, soprattutto per quanto riguarda studenti o neo laureati. Essendo un'azienda in forte crescita ed espansione, necessita costantemente di nuovo personale, ma soprattutto, di nuove idee e tecnologie da poter poi attuare all'interno dei propri prodotti. La maggior parte degli stage vengono avviati dal settore di Ricerca e Sviluppo proprio per soddisfare questa necessità; solitamente le attività di stage proposte comprendono un periodo di formazione iniziale per poter prendere dimestichezza con le nuove tecnologie che si andranno ad attuare, per poi metterle in pratica con il supporto costante del team. Questa sezione presente all'interno di *Siav* garantisce una costante dedizione alla ricerca, non soffermandosi a ciò che è già presente all'interno dell'azienda, ma cercando di spaziare tra più tematiche in contemporanea, procedendo all'attivazione di più stage in ambiti diversi, spaziando quindi la ricerca su più fronti. L'azienda, inoltre, avendo stipulato un cospicuo numero di *partnership* si ritrova spesso e volentieri far partecipare i propri dipendenti a diversi *meeting* o seminari per quanto riguarda lo studio e approfondimento di nuove tecnologie emergenti o di tendenza. Tali aspetti vengono poi valutati dal team tramite un studio di fattibilità per poter verificare se le tematiche affrontate possono portare qualcosa di redditizio all'interno del contesto aziendale. Quest'ultime portano *Siav* ad essere una delle realtà italiane più attive e propense alla ricerca e allo sviluppo.

2.2 Introduzione al *Process mining*

Il process mining è una tecnica di gestione dei processi attraverso il quale è possibile portare ad un miglioramento dei processi aziendali. Essi vengono analizzati attraverso particolari strutture chiamate log degli eventi. All'interno di tale struttura è presente una sequenza di eventi, che forma delle tracce. Una traccia può essere vista come un processo aziendale; un esempio utilizzato più volte è stato la produzione di un ingranaggio, individuando tutte le varie attività che si susseguono, partendo dal materiale grezzo fino ad arrivare al prodotto completo. Tramite l'analisi di tali processi è possibile studiare le varie procedure che intercorrono all'interno della realtà aziendale, cercando di osservare le dinamiche collegate ad ogni processo, per poi migliorarle ed ottimizzarle tramite un'analisi ben fatta e tramite l'utilizzo di vari *tool* presenti sul mercato. A questo merito *Siav* ha deciso di fornire ai propri clienti un serivizio che sfruttasse questi principi.

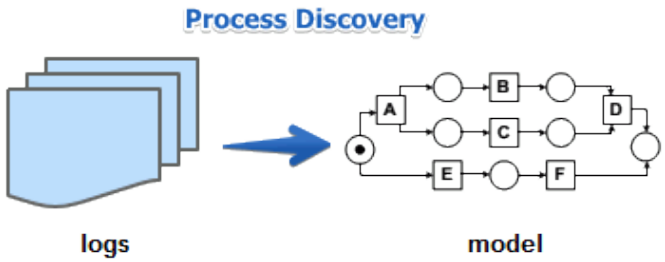


Figura 2.1: Come viene rappresentato graficamente un log degli eventi http://mlwiki.org/index.php/Process_Mining

2.3 Introduzione al progetto

Siav attraverso i numerosi progetti di stage mira ad esplorare nuove tematiche e nuove opportunità tecnologiche che possano portare ad un miglioramento dei propri prodotti. L'attività che mi è stata proposta mirava proprio al miglioramento di un loro applicativo. Visto in scala più ampia, il progetto finale risulterà una rivisitazione di *Bipod*, un applicativo in ambito process mining, utilizzato come gestore di processi aziendali. Tale software, risulta funzionante in ogni sua parte, anche se la sua struttura risulta poco estendibile. Per questo motivo mi è stato proposto di sviluppare una parte di esso che si andrà poi ad integrare con il nuovo applicativo, andando a rimpiazzare l'attuale esistente. Nello specifico le principali richieste a cui ho fatto fronte sono state le seguenti:

* Sviluppo di una libreria di process mining per filtraggio su log degli eventi.
* Sviluppo di un'interfaccia *fronted* tenendo come punto di riferimento il vecchio applicativo.
* Scrittura di Stub per verificare il correntto comportamento dell'interfaccia di filtraggio.

La funzionalità che mi sono state proposte sono tra le più cruciali per quanto riguarda il prodotto finale, essendo le gestione dei filtri un'aspetto fondamentale nell'analisi dei processi. Per questo motivo è stato necessario investire una caspicua parte di tempo a disposizione per l'attività di formazione. Tramite questo stage è stato possibile per *Siav* migliorare uno dei propri prodotti che aveva perso la sua mantenibilià, essendo stato sviluppato da un piccolo team che ora non era più presente in azienda. Oltretutto la documentazione è risultata scarsa e di difficile comprensione; è stato quindi deciso di riscrivere l'applicativo da zero, cercando di mantenere le principali funzionalità, analizzando qualche aspetto critico che presenteva la vecchia architettura e cercando un'alternativa solida basandosi su nuove tecniche e tecnologie che all'epoca non erano state prese in considerazione. Durante la prima parte dello stage l'obiettivo è quello di analizzare le funzionalità e interfaccia legate al filtraggio dei log presente nell'applicativo in modo da poter individuare tutte le possibili funzionalità disponibili all'utente. Dopo una revisione completa ed approfondita dell'applicativo ne è emerso che dal punto di vista delle funzionalità il software non aveva nulla da invidiare rispetto ai più comuni *tool* di *process mining*, presentando tutte le principali funzionalità per la modellazione di log. Dal punto di vista della sua efficenza però, le cose andavano diversamente: tutto il sistema era gestito tramite chiamate sicrone; ciò comportava la presenza di lunghi tempi di attesa per ogni operazione che l'utente intendeva effettuare. Solamente per effettuare un'operazione di filtraggio in un log della dimensione di qualche *Megabyte* era necessario un tempo di attesa che andava da 1 a 5 secondi. Il che può essere accettabile, se non fosse che in ambito *process mining*, per poter gestire di un cospicuo numero di processi, vengono analizzati log che possono essere di gran lunga più corposi e impegnativi da analizzare per il sistema. Da ciò ne è scaturatia l'idea di una restaurazione dell'applicativo.

2.3.1 Progettazione libreria in relazione all'architettura finale

In seguito all'attività di analisi del vecchio applicativo, un obiettivo dello stage risiede nella progettazione di un libreria di filtraggio in grado di poter effettuare la principali operazioni di pulizia su log degli eventi. Tale libreria dovrà soddisfare alcuni requisiti fonadmentali:

* dovrà essere mantenibile nel tempo, facendo uso quindi di oppurtuni principi cardine della Programmazione ad Oggetti e dovrà essere documentata in modo preciso, facendo sì che chiunque la analizzi in futuro per poter effettuare alcune modifiche possa apprendere in modo chiaro e trasparente ogni sua parte e la sua struttura.
* dovrà essere affidabile e testata in ogni sua parte tramite *Unit Test* in modo da garantire una base solida di partenza per poi contestualizzarla all'interno di un'architettura a microservizi.

2.3.2 Progettazione interfaccia *front-end*

Successivamente all'implementazione della libreria sarà necessario progettare l'interfaccia *fronted*; per quest'ultima sarà necessario avvalersi del loro vecchio applicativo *Bipod*, per poi confrontarlo con la libreria sviluppata discutendo con il tutor eventuali modifiche in caso di discrepanze. Tale interfaccia sarà rivista in modo più preciso ed approfondito una volta terminata la libreria, avendo quindi una visuale più ampia delle funzionalità offerte. Inoltre dovrà essere sviluppata con un framework ed un linguaggio più attuale rispetto al vecchio software; inoltre è stata richiesta un'interfaccia più intuitiva e semplificata che sia in grado di adattarsi a più dispositivi.

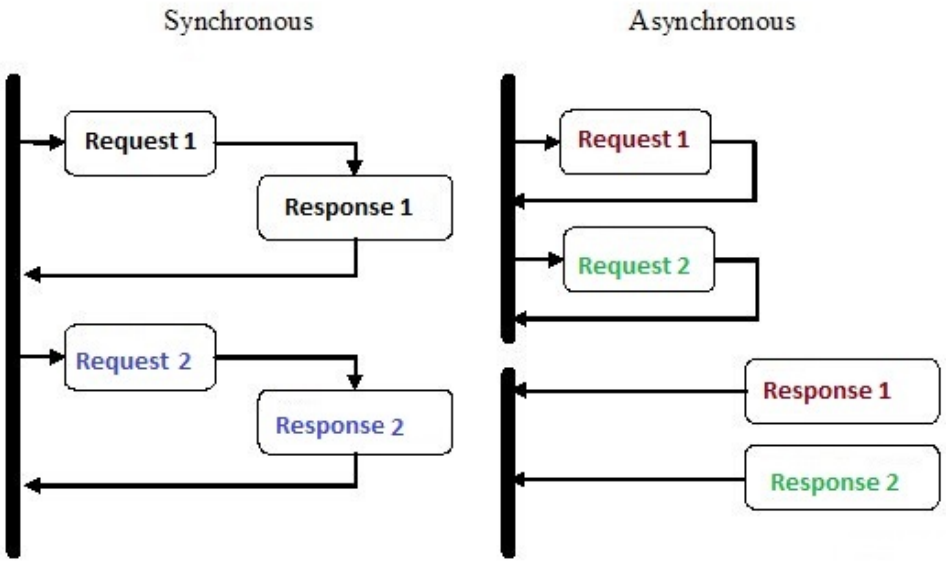


Figura 2.2: Illustazione della differenza tra chiamate sincrone ed asincrone

2.4 Pianificazione del progetto

Durante la stesura del piano di lavoro ho discusso con il tutor tutte le principali attività che avrei dovuto svolgere nell'arco dei due mesi preposti. Tali attività, anche se in forma generica, sono state inserite nel diagramma di Gantt presente nella sezione più in basso.(Figura 2.4). Per poter entrare in maniera concreta all'interno dei processi aziendali, tramite il supporto del tutor e del team nel quale sono stato inserito, ho cercato di apprendere nella miglior maniera possibile i principi cardine dell'*Agile programming*. Non avendo mai avuto l'occasione prima d'ora di approcciarmi a tale metodologia è stato impegnativo, ma appagante, entrare nei suoi meccanismi pratici e nelle sue dinamiche. Tuttavia sono riuscito, almeno in parte, a comprendere alcune importanti metodologie che sono state applicate durante tutto il periodo di stage. Le attività sono state tracciate tramite semplici note, condivise all'interno del team, in cui è descritta una breve cronologia di tutte le attività svolte giorno per giorno, indicando eventuali problematiche riscontrate e come sono state affrontate. La stessa motodologia è stata utilizzata per descrivere le tematiche trattate durante gli incontri di formazione e per descrivere le scelte fatte per far fronte alle diverse problematiche emerse. Per poter verificare la corretta pianificazione era prassi organizzare *Daily meeting* tra tutti i membri del team e gli stagisti presenti, in modo da potersi confrontare sullo stato di avanzamento del prodotto, discutendo di eventuali criticità riscontrate e pianificando le attività per la giornata. In questo modo era possibile avere una panoramica generale sul prodotto in maniera completa ed un'opinione sul proprio operato da parte degli altri membri, capendo così se la strada intrapresa fosse quella più corretta.

2.5 Aspettative aziendali

Al termine delle 340 ore l'azienda si aspetta di avere un libreria di filtraggio solida ed estendibile, in grado di eseguire le principali funzionalità di filtraggio al'interno di un log degli eventi, un'interfaccia *front-end* semplice ed intuitiva in grado di utilizzare tutte le funzionalità incluse all'interno della libreria e uno *Stub* per osservare il comportamento di un'interfaccia, in relazione dell'architettura finale. Il tutto deve essere pensato per un'erogazione in modalità *cloud*, pertanto le scelte architetturali dovranno tener conto di questo prerequisito. Tali attività mi hanno portato a definire i principali requisiti

Figura 2.3: Illustazione semplificata di un'architettura server cloud (https://www.volico.com/services/cloud-hosting/)

con il tutor aziendale, che sono poi stati inseriti all'interno del piano di lavoro. In seguito a ciò il tutor ha individuato gli obiettivi per il progetto suddividendoli in due categorie: obbligatori e desiderabili.

Tabella 2.1: Obiettivi obbligatori dello stage

Obiettivi obbligatori
Inquadramento del problema: stesura di un documento di specifica del problema affrontato;
Analisi dei requisiti e specifiche tecniche di progettazione: stesura dei relativi documenti;
Implementazione del front-end web e dei servizi di back-end necessari a gestire le funzionalità di filtraggio degli eventi
Collaudo del sistema: il progetto deve prevedere una fase di test del software implementato, con documentazione dei risultati ottenuti

Tabella 2.2: Obiettivi desiderabili dello stage

Obiettivi desiderabili
point-and-click sul processo per inserimento di filtri sulle successioni di eventi
implementazione test front-end

2.5.1 Vincoli

I vincoli che mi sono stati posti durante la stesura del piano di lavoro possono essere raggruppati in 3 categorie:

Vincoli temporali

Rappresentano i vincoli dal punto di vista del tempo. Lo svolgimento dello stage ha avuto una durata di 340 ore. Tali ore sono state dstribuite in modo uniforme durante l'arco di 9 settimane. Inizialmente sono state concordate 8 settimane, ma in seguito ad alcune criticità riscontrate in fase di sviluppo, con la conseguente negoziazione dei requisiti, sono state portate a 9; includendo quindi una settimana aggiuntiva per terminare tutte le attività concordate. Nel periodo antecedente all'inizio dello stage l'azienda ha redatto una scansione temporale delle attività su base settimanale suddivisa nel seguente modo:

* **Settimana 1**: Studio introduttivo del Process Mining con le relative tecnologie. Studio dell'architettura di storage messa a disposizione e del framework ProM
* **Settimana 2**: Studio di fattibilità per la realizzazione dei moduli necessari a erogare le funzionalità di filtraggio
* **Settimana 3-4-5**: Progettazione dei moduli di discovery del processo, visualizzazione log e filtraggio
* **Settimana 6-7**: Implementazione del software e *deploy* su docker
* **Settimana 8**: Test e sperimentazione del software

Durante tutto il periodo sopraindicato è stata richiesta anche una documentazione sul lavoro, includendo una breve cronologia delle attività svolte, indicando le eventuali criticità riscontrate e specificando in che modo sono state affrontate e risolte.
Tale pianificazione è stata rinegoziata a partire dalla sesta settimana; in quanto l'architettura finale non era stata ancora ideata in modo specifico. Questo ha portato, in accordo con il tutor ad una rinegoziazione dei requisiti e quindi della attività programmate; sostituendo l'attività di *deploy* su *docker* con l'implmentazione di alcuni stub in modo da poter verificare il comportamento dell'interfaccia front-end.

#	Attività	Settimane							
		1	2	3	4	5	6	7	8
1	Studio introduttivo PM								
2	Studio librerie								
3	Studio di fattibilità								
4	Analisi dei requisiti								
5	Progettazione web service e GUI								
7	Implementazione del *software*								
8	Test e sperimentazione del *software*								
9	Documentazione								

Figura 2.4: Diagramma di gantt relativo alle attività di stage

Vincoli metodologici

Lo stage è stato svolto presso la sede di Rubano(PD) in comune accordo con il tutor. Ciò è stato deciso con lo scopo di poter confrantarsi ed interagire in maniera sistematica ogni qualvolta ce ne fosse stato bisogno; non soltanto tramite un rapporto stagista e tutor, ma confrontandosi con altri figure presenti all'interno dell'azienda in modo da poter consolidare nuovi rapporti di tipo professionale, in modo da avere supporto aggiuntivo in caso ce ne fosse stata necessità. Oltre a ciò è stato richiesto un continuo confronto sulle attività effettuate e quelle in programma tramite alcune note di testo condivise tra il tutor ed il team di Ricerca e Sviluppo. Tali note sono poi oggetto di discussione tramite *Daily meeting*: in cui vi è un momento di confronto tra i membri del team per discutere in merito alle attività svolte e quelle in programma per la giornata. Mi è stata messa a disposizione una postazione di lavoro con personal computer, connessione ad Internet ed alla rete locale con accesso al server di sviluppo, in modo da poter effettuare *deploy* di eventuali macchine virtuali necessarie allo sviluppo. Sono stati forniti una serie dati di esempio per testare quanto sviluppato, oltre a tutte le librerie necessarie allo sviluppo stesso dell'applicativo.

Vincoli tecnologici

Come sopra descritto l'obiettivo fondamentamentale per l'azienda è che l'applicativo finale fosse in grado di girare su server *Cloud*. Questo ha influito in modo significativo sui vincoli tecnologici che sono stati imposti. Innanzitutto tramite lo sviluppo di una libreria che fosse scritta in modo semplice e leggibile, per far sì che chi dovrà inserirla nell'ambiente a microservizi abbia un'idea precisa delle sue caratterische ed il suo funzionamento; in secondo luogo tramite la gestione dell'asincronia all'interno dell'interfaccia *front-end*, andando a gestire i messaggi in arrivo dal server, categorizzandoli

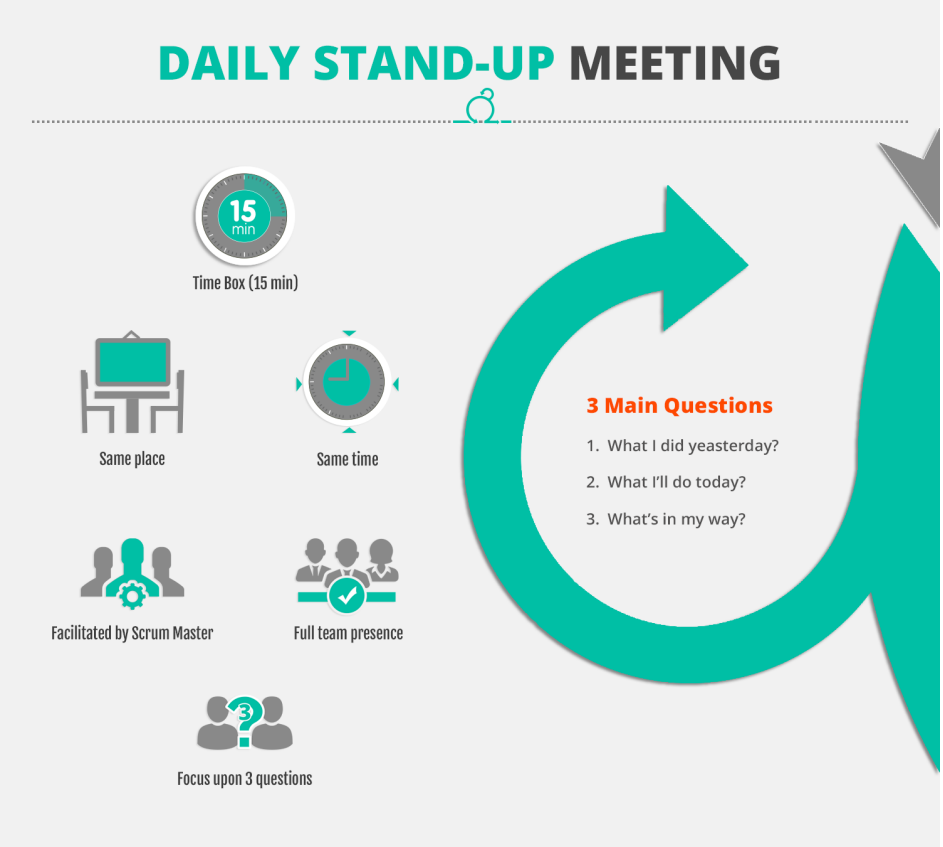


Figura 2.5: Illustrazione semplificata di *Daily Meeting* (https://tinyurl.com/ub822u3)

e adattando il comportamente dell'interfaccia a seconda della caratterizzazione del messaggio in arrivo.

2.6 Aspettative personali

Tutte le mie passate esperienze sono riconducibili esclusivamento all'ambito accademico; l'esperienza che più mi ha avvicinato all'ambito lavorativo è stato il progetto di Ingegneria del Software; tramite il quale ho potuto constatare alcune dinamiche presenti all'interno di un contesto professionale. Per poter approcciarmi al mondo lavorativo ho partecipato all'evento organizzato dall'Università di Padova *Stage-IT*; tramite il quale ho potuto affrontare diversi incontri con realtà aziendali ben formate e strutturate. Durante la fiera ho potuto valutare diverse attività di stage tenendo in considerazione le tematiche trattate e quanto ciò potessero incidere nel mio grado di formazione. La maggior parte di esse però, non sono risultate, a mio parere, stimolanti per la mia maturità a livello professionale. Le aspettative che mi sono preposto prima dell'evento di *Stage-IT* sono state le seguenti:

* Introduzione a nuove metodologie di lavoro.
* Visione e apprendimento di nuove tecnologie.

* Lavoro in modo autonomo sotto la supervisione che tutor che mi presti supporto.
* Visione del contesto lavorativo in maniera più ampia, non legata strettamente all'attività di stage.

Successivamente a *Stage-IT* sono stato contattato da altre aziende operanti nel settore dell'informatica, ognuna delle quali mi ha proposto varie attività affrontando diversi argomenti e tecnologie. Una su tutte che mi ha colpito è stata proprio *Siav* che, attraverso le loro proposte e la loro costante dedizione in merito alla ricerca di nuove tematiche e tecnologie mi ha convinto a svolgere l'attività di stage presso la loro azienda, collocandomi proprio all'interno del team di Ricerca e Sviluppo. Ciò ha comportato un aumento delle mie aspettative di fronte a questa attivita:

* Valutare quanto possano essermi state utili le nozioni apprese durante il percorso universitario.
* Apprendimento dei principi cardine del process mining.
* Apprendimento sul funzionamento di un'architettura a microservizi.
* Instaurazione di discussioni con il personale dell'azienda in modo da potersi confrontare ed avere diversi punti di vista rispetto ad alcune tematiche.

Capitolo 3

Resoconto dello stage

3.1 Studio preliminare

In questa sezione sono descritte le principali attività svolte durante il periodo di formazione preliminare. Tale periodo è stato di fondamentale importanza per permettermi un buon grado di autonomia durante lo svolgimento di tutta l'attività di stage.

3.1.1 *Process mining*

Prima essermi inoltrato all'interno del problema, è stato necessario un periodo di formazione che ha ricoperto la durata di due settimane, durante il quale ho appreso i principi cardine del *process minig*, cercando di capire le sua dinamiche e la sua utilità all'interno di un contesto aziendale ben formato, per poter poi metterle in pratica durante la realizzazione del progetto. Tale periodo è stato caratterizzato dalla visione di videolezioni online, per spiegare l'argomento in generale, e da incontri programmati con il tutor in cui verificava il mio effettivo apprendimento. Inizialmente ho trovato qualche difficoltà durante lo studio di tali nozioni, essendo stato per me un argomento completamente nuovo; ma dopo aver compreso le sue meccaniche sono riuscito ad applicare in maniera completa i suoi principi durante tutte l'arco dello stage.

3.1.2 Angular e *typescript*

In concomitanza con il periodo di formazine sopracitato sono stati svolti alcuni studi in merito al noto *framework*, in modo da avere una buona base di partenza per poter procedere in autonomia allo sviluppo dell'interfaccia. Inizialmente, sotto consiglio del tutor, ho guardato la documentazione presente all'interno del sito di Angular, in modo da poter avere una panoramica generale. Successivamente sono sono stato affiancato per un breve periodo da diverse figure all'interno dell'azienda, sviluppando alcune interfacce basilari che poi mi sarebbero state utili al fine del prodotto finale, in modo da rendere più metodica ed immediata l'assimilazione delle principali strutture presenti in Angular;

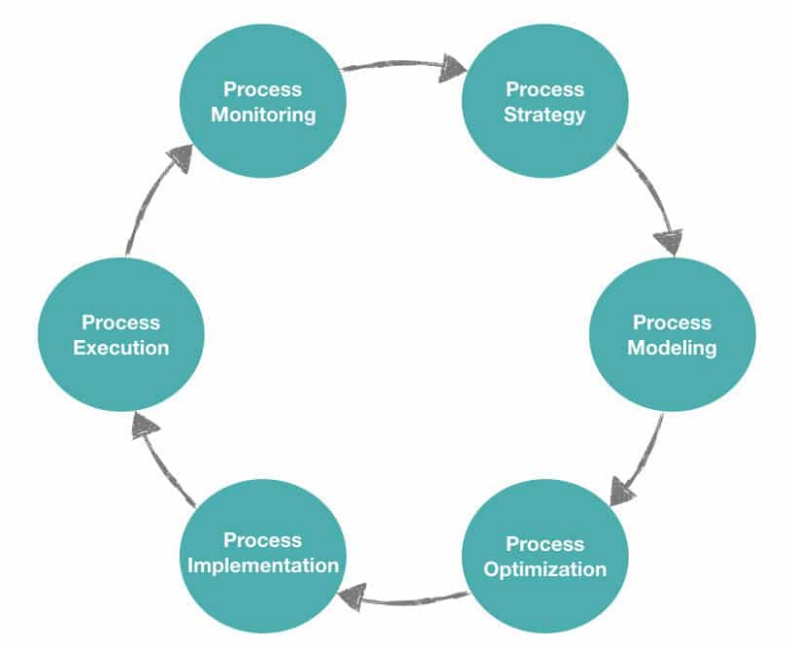


Figura 3.1: Ciclo di vita della gestione dei processi https://lanalabs.com/en/what-is-process-mining-bpm/

3.2 Analisi dei requisiti

Per poter tracciare i requisiti specifici necessari al compimento degli obiettivi, tramite consiglio del tutor, mi sono servito dei principali *tool di process mining* presenti sul mercato, assieme al software interno all'azienda: *Bipod*. In questo modo è stato possibile delineare una categorizzazione dei moduli di filtraggio richiesti, cercando di capire in che modo operassero nel concreto all'interno del log. Allo stesso modo sono stati tracciati ed analizzati i requisiti relativi all'interfaccia *front-end*, confrontando le interfacce del loro applicativo con altri tool presenti in rete come *ProM* e *Disco*. Per quanto riguarda lo *stub* implementato è stato necessario uno studio preliminare dell'architettura finale, cercando di simularne il comportamento a seguito di richieste di filtraggi.

3.2.1 Libreria

Il *tool* che mi ha permesso una maggior comprensione delle meccaniche di filtraggio, dandomi un buono spunto per l'implementazione della libreria è stato senz'altro *ProM*; data la sua struttura modulare, a cui è possibile aggiungere esensioni di vario tipo, mi è stato possibile analizzarlo a fondo tramite lo studio del proprio repository. Mentre tramite gli altri *tool* sono stato in grado di capire le dinamiche di filtraggio, tramite il repository di *ProM* ho trovato alcuni esempi di implementazioni, anche se mal documentati e poco comprensibili; quest'ultimi sono però stati fondamentali per comprendere gli standard a cui far affidamento. Nel caso della libreria, lo standard adottato è stato *OpenXES*. Quest'ultimo rappresenta la struttura del log in formato .xes, una struttura molto simile ai file di tipo *xml*. *OpenXES* descrive il log degli eventi tramite una struttura dati chiamata XLog. A partire da questa struttura è possibile

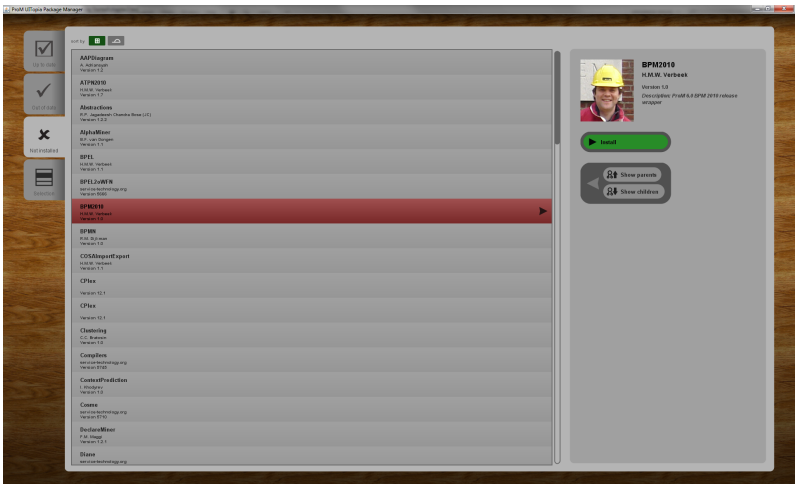


Figura 3.2: Package manager di ProM 6.x http://www.promtools.org/doku.php?id=gettingstarted:installation

modellare i dati in arrivo a proprio piacimento gestendoli come vere e proprie liste di eventi. Tali eventi, rappresentati dalla classe XEvent, contengono un serie di attributi che vanno a caratterizzare l'evento e ne permettono il loro reperimento. I principali requisiti che sono emersi da questa analisi riguardano la categorizzazione delle tipologie di filtraggio, la loro implementazione ed il loro scopo. Sono state quindi delineate le seguenti tipologie:

* Filtraggio basato su valori di *timestamp*
* Filtraggio basato sulle performance degli eventi.
* Filtraggio basato su eventi consecutivi
* Filtraggio basato sugli attributi
* Filtraggio basato sulle varianti
* Filtraggio basato su *endpoint*

Oltre alle funzionalità di filtraggio la libreria dovrebbe includere una serie di metodi legati al rilevamento di alcune caratteristiche del log in forma generica, in modo da poter determinare a priori alcune caratteristiche generali per poi agire di conseguenza. Tali funzionalità di categorizzazione sono state pensate per favorire l'adattamento dell'interfaccia *front-end* a fronte alle diverse tipologie di log che potevano venir esaminate dall'utente.

3.2.2 *Front-end*

Per quanto rigurda i requisiti lato *front-end* mi sono basato innanzitutto sullo studio dell'attuale *tool* presente in azienda, cercando di inquadrare in modo chiaro tutte le interfacce atte allo svolgimento di operazioni di filtraggio. Dopo aver effettuato questa analisi, ho discusso a fondo con il tutor in merito ad una specifica interfaccia: quella riguardante il filtraggio sulle performance. Il vecchio applicativo presentava

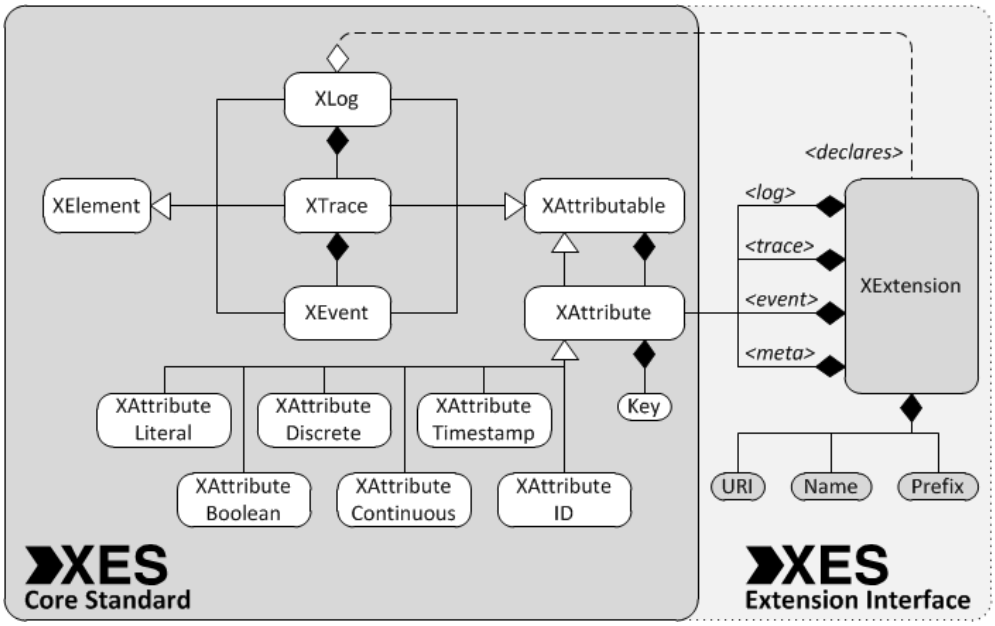


Figura 3.3: Diagramma dei package dello standard OpenXes (http://www.xes-standard.org/openxes/start)

un'interfaccia poco intuitiva in cui non era ben chiaro nè la sua utilità, nè le sue funzionalità. È stato quindi deciso di riscrivere l'interfaccia cercando di semplificare la visualizzazione attuale, redendo più piacevole ed intuitiva l'esperienza utente. Inoltre è stato fortemente richiesto dal tutor che l'interfaccia risultasse scalabile ed adattabile in modo da poter estendere il suo utilizzo a qualsiasi tipo di *device*. Il risultato finale sarebbe quindi stata un'interfaccia solida ed intuitiva introducendo il supporto per le comunicazioni tramite *Websocket* e la gestione dei messaggi in arrivo dal server. Sono state quindi fissate le interfacce per ogni tipologia di filtraggio presente all'interno della libreria.

3.2.3 *Stub*

Dopo la fase di negoziazione dei requisiti con il tutor aziendale, è stato deciso in comune accordo che la parte riguardante l'architettura del *back-end* venisse accantonata, dato il tempo limitato a mia dispoizione e la sua rilevante complessità. Questo per far spazio ad uno *Stub* che andasse a simulare il comportamento dell'architettura finale a fronte di una richiesta di filtraggio per valori temporali. Tale *Stub* avrebbe dovuto comunicare con il *Client* tramite un canale *Websocket*, andando a gestire tutte le tipologie di messaggi ritornati dal server.

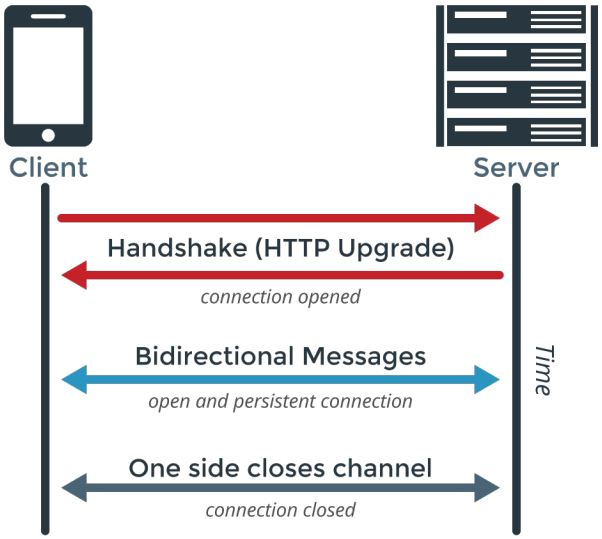


Figura 3.4: Ciclo di vita di una connessione tramite websocket https://radu-matei.com/blog/aspnet-core-websockets-middleware/

3.3 Progettazione

3.3.1 Libreria

Dopo aver delineato i principali requisiti da dover soddisfare sono passato alla fase di progettazione, cercando di strutturare la libreria in modo semplice, ma allo stesso tempo efficace, andando a documentare ogni singola funzionalità tramite *Javadoc*, in modo da rendere mantenibile l'intera libreria e soprattutto renderla comprensione ai futuri utilizzatori. È stato deciso di dividerla in più classi, ognuna di esse riguardante una tipologia di filtraggio, in modo da rendere la sua lettura più semplice ed immediata. Tutti i metodi presenti all'interno della libreria operano su tipi di dato in formato standard derivati da *OpenXES*, questo è stato un requisito fortemente richiesto dal tutor interno. Oltre alle operazioni di filtraggio sono stati inclusi altri metodi per poter effettuare degli studi preliminare sul log preso in esame, andando a catalogarlo sotto determinati aspetti che, al fine di tutto l'applicativo, risultano di fondamentale importanza. Le principali classi che comporrano la libreria di filtraggio saranno dunque sei, ossia quelle che sono state delineate tramite analisi dei requisiti. In aggiunta saranno presenti altre due classi che andranno a delineare i metodi di categorizzazione generale del log sulla base del *lifecycle* e delle varianti. All'interno di ogni classe dovranno essere presenti dei metodi pubblici a cui l'utente avrà accesso diretto ed una parte di metodi privati che saranno invocati da quelli appena descritti ed andranno ad effettuare operazioni elementari sul log in modo da poter aderire al principio *open-closed*.

3.3.2 *Front-end*

Per poter eseguire una fase di Progettazione trasparente rispetto ai requisiti analizzati mi sono avvalso di alcuni *mockup* tramite i quali, sotto la supervisione del tutor, mi hanno permesso di tracciare tutte le interfacce di filtraggio, andando a rispettare tutti i requisiti preposti. Tali modelli sono stati ideati allo stesso modo con cui è stata

progettata la libreria: utilizzando un *layout* a schede, dove, per ognuna di esse, è presente una tipologia di filtraggio. Un requisito che è stato fortmente richiesto da parte del tutor è stata la scalabilità dei componenti, andando ad inserire misure in percentuale ad ogni aspetto dell'interfaccia; Come per la fase di analisi dei requisiti anche in questo caso il filtraggio sulle performance ha richiesto una particolare attenzione: nel vecchio applicativo tale interfaccia era mal strutturata e non era ben chiaro all'utente che tipo di filtraggio stesse utilizzando. Ciò era dovuto ad un aspetto che prima d'ora non era stato preso in considerazione: il *lifecycle* delle attività. Questo aspetto ha portato ad un lunga discussione con il tutor ed i membri del team in merito alle scelte da adottare. Ne è emerso che sarebbe stato necessario una differente interfaccia utente in base alla tipologie di *lifecycle* presente all'interno del log. Tali metodi sono stati opportunamente implementati all'interno della libreria.
Per quanto rigurarda la futura interazione con i servizi *back-end* è stato necessario avvalersi di *RxJS*; una libreria sviluppata per *javascript* che presenta tutte le principali funzionalità per la gestione dei messaggi tramite canali *websocket*. Questa libreria è in grando di reperire i messaggi in arrivo tramite un gestore di *callback* che differenzia i vari messaggi e fa reagire l'interfaccia in modo diverso a seconda della tipologia in arrivo. I messaggi vengono trasmessi all'interno del canale sottoforma di file *JSON*.

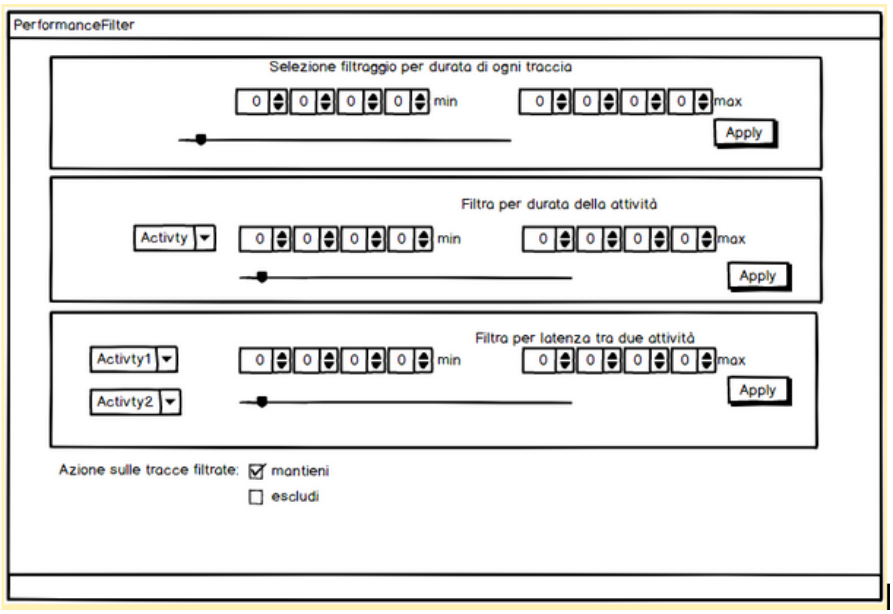


Figura 3.5: Mockup ottenuto per l'interfaccia riguardante il filtraggio per performance

3.3.3 *Stub*

Prima di poter passare alle realizzazione dello Stub è stato necessario dare una particolare attenzione all'architettura finale del *back-end*. Purtroppo, date le strette tempistiche da rispettare e le varie criticità riscontrate durante le fasi precedenti è stato possibile comprenderla solamente a livello teorico. Tale architettura si basava quindi su un sistema a microservizi all'interno del quale erano presenti degli *worker* che reagivano a determinate richieste lato *client*. Nel momento in cui veniva presa in carico l'operazione, il *worker* ritornava messaggi di progresso, indicando lo stato

di avanzamento dell'operazione, che andavano quindi ad ag aggiornare l'interfaccia. Al momento della terminazione, il servizio inviava il relativo messaggio, che andava a notificare l'utente sull'effettivo completamento dell'operazione.

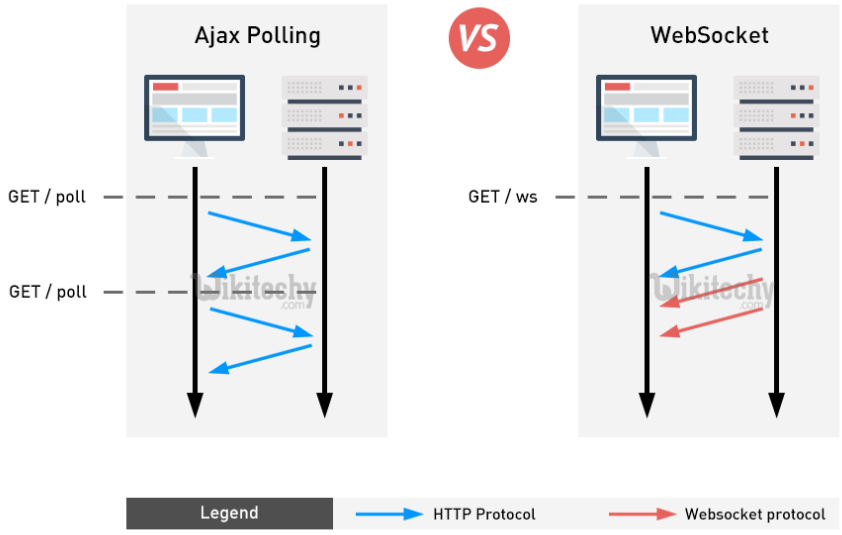


Figura 3.6: Differenza tra una chiamata websocket ed una http https://www.wikitechy.com/tutorials/socket/differences-between-websockets-and-ajax

3.4 Codifica

Avendo svolto una buona analisi dei requisiti seguiti da una solida progettazione, la fase di codifica non ha portato a grosse criticità. Tale processo ha influito però su alcuni aspetti che non erano stati presi in considerazione prima d'ora.

3.4.1 Libreria

Per la codifica della libreria ho seguito in modo meticoloso i punti cardine fissati durante le fasi precedenti. Le principali criticità riscontrate riguardavano l'efficenza dei metodi. Un log degli eventi può rappresentare una struttura molto grande e complessa, a tal proposito non era ben chiaro se il log filtrato, ritornato dai vari metodi, dovesse essere una copia, creando di conseguenza una cospicua occupazione di memoria. Per far fronte a ciò è stato deciso di delegare tale scelta all'architettura del *back-end*; andando a scegliere in base alla dimensione del log preso in esame quale fosse la scelta più performante. Il risultato ottenuto dopo tale processo è stata un libreria contente sette classi, categorizzate in base allo loro utilità. Sono stati implementati nel complessivo 22 metodi pubblici e 9 privati, ognuno di essi documentato secondo lo standard *Javadoc*.

3.4.2 *Front-end*

Per poter rendere l'interfaccia limpida e trasparente all'utente è stato necessario ricercare all'interno di varie librerie presenti sul web i componenti *Angular* che si adattassero in maniera migliore rispetto alle aspettative. Ciò ha comportato ad un ingente investimento in termini di risorse, dovendo far fronte ad eventuali problematiche riscontrate dal funzionamento specifico di tali componenti. Questo ha portato ad una continua ricerca di componenti performanti, sia dal punto di vista grafico, in termini di scalabilità, sia dal punto di vista della gestione delle risorse. Per quanto riguarda l'integrazione con le connessioni tramite *websocket* è stato scelto di utilizzare una caratterizzazione univoca della richieste da inviare tramite questo canale: la struttura del messaggio ideata è stata implementata nella forma seguente:

```
{
  "operation": "stringa rappresentante il tipo di operazione",
  "id_log": "stringa che contiene l'id univoco del log",
  "id_job": "long che viene assegnato dal broker",
  "user": "stringa del nome utente che si è loggato",
  "options": [
          "lista di qualsiasi tipo di oggetto (tipicamente
     stringhe e numeri) rappresentano i valori di filtraggio"
  ]
}
```

Figura 3.7: Struttra della richiesta di filtraggio inviata al server tramite canale websocket

3.4.3 Stub

Lo stub non ha richiesto particolari criticità durante la sua fase di codifica, esso infatti è risultato un semplice file *python* contenente i metodi di risposta da dover ritornare al *front-end* a fronte di richieste di filtraggio. Tali messaggi dovevano poi essere intepretati dal *client* per poi reagire di conseguenza. Per la struttura dei messaggi in uscita è stato scelto un formato univoco per qualsiasi tipologia in modo da uniformarne la ricezione anche lato *front-end*.

3.5 Verifica e Validazione

Sia la fase di verifica che quella di validazione è stata svolta in collaborazione con il tutor. Un fase di verifica veniva svolta, tipicamente, alla fine della settimana lavorativa, in cui venivano confrontate le attività preposte ad inizio settimana e confrontate con quelle realmente fatte, osservandone attentamente i punti fondamentamentale per verificare la presenza di errori o malfunzionamenti del sistema.
La fase di validazione è stata svolta durante l'ultima settimana di stage; in questa fase è stata confrontata la lista di tutte le varie attività svolte con i relativi obiettivi e requisiti concordati all'interno del piano di lavoro e rinegoziati durante il periodo successivo. Ciò ha portato alla conferma dell'effettiva terminazione delle attività in modo soddisfacente.

3.6 Risultati ragguinti

La libreria è risultata ben formata in ogni sua parte, è stato fatto un buon lavoro di copertura del codice attraverso *unit test* mediante il quale sono stati testati 27 metodi dei 31 complessivi tra pubblici e privati; questo ha permesso di raggiungere una coprtura del 87%. Ciò ha portato ad un buon grado di soddisfacimento sia da parte del tutor, sia a livello personale.
Lato *front-end* sono state effettuate alcune prove strutturali dovute al ridimensionamento dell'interfaccia. In questo modo è stato possibile constatare che l'interfaccia fosse solida e ben formata. Purtroppo, a causa di alcune limitazioni dovute all'architettura del *back-end* illustati nel capitoli precenti, non è stato possibile testare il suo effettivo funzionamento in ogni sua parte; pertanto mi sono concentrato esclusivamente in un'unica interfaccia di filtraggio, gestendo in maniera soddisfacente ogni suo comportamento a fronte di eventi che venivano ritornati dallo stub.
Nel complesso tutti i processi svolti hanno avuto uno scopo ben prciso per la realizzazione del prodotto finale. La libreria ha raggiunto un buon grado di affidabilità e mantenibilità in cui sarà possibile includere eventuali funzionalità aggiuntive in maniera diretta. Il *front-end* è risultato molto solido a livello grafico, mantenendo la stessa struttura anche a fronte di ridimensionamti consistenti. L'integrazione delle chiamate *websocket* e del relativo stub hanno mantenuto comunque l'interfaccia solida senza comportare alcun tipo di malfunzionamento. Ciò ha portato ad un significativo passo in avanti guardando nell'ottica dell'applicativo finale che dovrà rilasciare *Siav*.

Capitolo 4

Valutazione retrospettiva

4.1 Soddifacimento obiettivi

Gli obiettivi prefissati con il tutor in seguito alle negoziazione dei requisiti sono stati portati a termine in maniera completa, ciò ho permesso di arrivare ad un buon grado di soddisfacimento sia personale, sia dalla parte del tutor.
Purtroppo alcune aspettative che mi ero prefissato all'inizio dello stage non state rispecchiate a pieno. Questo è stato causato da alcune problematiche che sono sorte in corso d'opera con il consenguente rallentamento della attività pianificate e la conseguente rinegoziazione dei requisiti. Ciò nonostante mi ha permesso di arrivare alla fine delle attività con un notevole aumento della mia consapevolezza di saper produrre prodotti di qualità anche di fronte a problematiche che spesso accadono all'interno di questi contesti lavorativi. Dal punto di vista degli obiettivi aziendali, le mie attività hanno portato ad una maturazione di quello che poi sarà il prodotto finale, includendo tutti i principali aspetti che riguardano la pulizia del log degli eventi, andando quindi a soddisfare tutti gli obiettivi obbligato ri che sono stati indicati all'interno del piano di lavoro. Per quanto riguarda gli obiettivi desiderabili non sono riuscito a soddisfarli in maniera completa a causa di altre esigenze che sono state messe in primo piano.

4.2 Conoscenze acquisite e difficoltà riscontrate

L'attività in sè mi ha permesso di poter comprendere le principali dinamiche che può presentare una tecnica di analisi di processi, capendone le sue dinamiche e la sua utilità. Inizialmente, dopo la fase iniziale di formazione, il concetto era molto generico e non di facile comprensione, ma tramite esempi pratici applicati al contesto aziendale ho avuto un notevole incremento delle mie conoscenze sia dal punto di visto teorico che applicativo. Per quanto riguarda il lato Angular, ho appreso le principali tecniche di analisi, progettazione e codifica di *webapp* integrata con i relativi servizi *websocket*. Quest'ultimi sono stati compresi in modo lineare, andando a

4.3 L'esperienza in relazione all'ambiente universitario

In generale tale esperienza è stata positiva per me, permettomi di entrare direttamente a contatto con il settore dell'informatica.

Glossario

Angular Angular 2+ è una piattaforma open source per lo sviluppo di applicazioni web con licenza MIT, evoluzione di AngularJS. Sviluppato principalmente da Google, la sua prima release è avvenuta il 14 settembre 2016. 3, 27

BigData raccolta estesa di dati in termini di volume, velocità e varietà da richiedere tecnologie e metodi analitici specifici per l'estrazione di valore o conoscenza. 3, 27

Broker Sistema che distribuisce vari aspetti del software su nodi differenti tramite l'utilizzo di oggetti remoti. 4, 27

Cloud paradigma di erogazione di servizi offerti on demand da un fornitore ad un cliente finale attraverso la rete Internet. 3, 27

Deploy consegna o rilascio al cliente, con relativa installazione e messa in funzione o esercizio, di una applicazione o di un sistema software tipicamente all'interno di un sistema informatico aziendale. 11, 27

Disco Tool di process mining sviluppato da Fluxicon. 18

Framework Architettura logica di supporto su cui un software può essere progettato e realizzato. 3, 27

Helpdesk Servizio clienti telematico. 5, 27

Javadoc Javadoc è un applicativo incluso all'interno del Java Development Kit della Sun Microsystems, utilizzato per la generazione automatica della documentazione del codice sorgente scritto in linguaggio Java. 21, 27

JSON JavaScript Object Notation. Formato adatto all'interscambio di applicazione client/server. 21

Mockup realizzazione a scopo illustrativo di un oggetto o sistema senza le complete funzioni originali. 21

Stage-IT Evento organizzate presso Padova Fiere in collaborazione con varie aziende locali che cercano di far approcciare gli studenti al mondo lavorativo tramite attività di stage. 13, 27

Stub è una porzione di codice utilizzata in sostituzione di altre funzionalità software in quanto può simulare il comportamento di codice esistente o l'interfaccia COM, e temporaneo sostituto di codice ancora da sviluppare. 8, 27

TFS Team foundation server. 3, 27

TFVC Team foundation version control. 3, 27

Unit test l'attività di testing di singole unità software.. 24

Websocket tecnologia web che fornisce canali di comunicazione full-duplex attraverso una singola connessione TCP. 20, 27

Bibliografia