Università degli Studi di Padova

Dipartimento di Matematica
"Tullio Levi-Civita"

Corso di Laurea in Informatica

Meccanismi di programmazione back-end e analisi funzionale in ambito bancario

Tesi di laurea triennale

Laureando
Abdelilah Lahmer

Relatore
Prof. Tullio Vardanega

Dicembre 2017

Dedicato alla mia famiglia e agli amici più stretti.

Sommario

Il presente documento riassume il lavoro svolto durante il periodo di stage, della durata di circa 300 ore, presso l'azienda Sopra Steria Group S.p.A con sede a Padova. La mission di Sopra Steria Group consiste nell'accompagnare e aiutare i suoi clienti a conseguire il successo attraverso il processo di trasformazione dei loro processi di business e dei loro sistemi informativi. Io sono stato inserito nella Divisione Servizi Finanziari, sezione che si occupa di sviluppo e manutenzione di sistemi bancari. Il tirocinio formativo e orientativo che mi è stato proposto ha avuto lo scopo di avviarmi verso la conoscenza della realtà lavorativa, approfondendo e verificando l'apprendimento ricevuto nel percorso degli studi con un'esperienza soggettiva legata direttamente alla realtà economica e produttiva del territorio, svolta nell'ambito di una realtà multinazionale. Nello specifico sono stato inserito in un gruppo di lavoro che opera su vari progetti di analisi e sviluppo, nonché manutenzione, di soluzioni bancarie per primari istituti di credito sul territorio italiano, quali Banca Popolare di Verona e Banco Popolare di Milano. L'obiettivo minimo del tirocinio è stato l'acquisire padronanza dell'ambiente di sviluppo MAINFRAME, del linguaggio di programmazione COBOL e l'essere in grado di comprendere correttamente le analisi tecniche. Obiettivo desiderabile è stato raggiungere anche una discreta autonomia nell'analisi di funzionalità e il concepimento, anche se parziale, di modalità di traduzione di queste in analisi tecnica.

Indice

Elenco delle figure

Elenco delle tabelle

Convenzioni tipografiche

Per la stesura del documento ho adottato le seguenti norme tipografiche:

- L'utilizzo del corsivo per le parole di ambito tecnico o in lingua inglese che non presentano un corrispettivo termine in italiano;
- L'indicazione con una G a pedice della prima occorrenza del paragrafo di tutti i termini che necessitano di una spiegazione esplicita, definita nel glossario presente a fine documento.

Capitolo 1

L'azienda

1.1 Profilo aziendale

L'azienda presso la quale ho svolto il mio stage è Sopra Steria Group S.p.A, leader europeo in ambito di trasformazione digitale. Essa offre servizi di consulenza, *system integration* e sviluppo software spaziando diversi mercati come Fashion, Banking, Energy, Aeronautica, Settore pubblico, Difesa e Trasporti.

Figura 1.1: Logo di Sopra Steria Group S.p.A. - Fonte: sito internet dell'azienda

Sopra Steria è partner di riferimento delle principali aziende ed organizzazioni pubbliche e private, proponendo progetti di trasformazione digitale che puntano ad un'alta qualità dei servizi erogati, valore aggiunto e innovazione. Conta 38.000 collaboratori in più di 20 paesi con un fatturato nel 2015 di 3,6 miliardi di euro ed opera sul territorio italiano con circa 700 risorse distribuite nelle sue sedi di Assago (MI), Roma, Collecchio (PR) e Padova.

Il gruppo è il risultato di una fusione, avvenuta nel 2014, ad opera di due aziende francesi Sopra Group SA and Groupe Steria SCA, comunemente chiamate Sopra e Steria, fondate rispettivamente nel 1968 e 1969. Ad oggi l'azienda si presenta internamente ben strutturata in Business Unit relative agli ambiti di sviluppo e adotta una politica di recruiting che mira alla competenza dei dipendenti da cui deriva la qualità dei prodotti, punto di forza dell'azienda.

Io sono stato inserito nella divisione "Servizi Finanziari e Assicurazioni" della sede di Padova, nata negli ultimi anni a partire da pochi dipendenti e che a breve vedrà il suo ampliamento in una nuova sede, anch'essa a Padova, per via della continua espansione della società. In particolare il mio ruolo è stato quello del programmatore web e sono stato affiancato dal mio tutor aziendale Andrea Faccio, uno dei principali sviluppatori web di questa sede.

1.2 Prodotti e servizi offerti

1.2.1 Prodotti

Nei mercati francesi, dove l'azienda è radicata, è attiva la vendita di prodotti bancari, ovvero software già pronto e configurabile in poco tempo presso il cliente. In Italia la situazione è differente e per i principali clienti, nell'ambito finanziario, raramente si vendono pacchetti di prodotti finiti ma si adotta una politica di personalizzazione secondo le esigenze del cliente. I prodotti principali offerti dalla business unit in cui sono stato formato sono quindi riassumibili in:

- Applicazioni web per la gestione di finanziamenti bancari e la relativa evoluzione e manutenzione;
- Programmi *host* di gestione dati e della loro consistenza e persistenza.

1.2.2 Servizi

Il sistema di gestione per la qualità dei servizi offerti ai clienti di Sopra Steria Group è certificato ISO 9001:2008 ed è annualmente sottoposto a verifiche da parte di un ente accreditato di terza parte. I principali servizi erogati dalla divisione per l'ambito bancario sono:

- Consulenza in ambito informatico per l'ampliamento ed il soddisfacimento della clientela da parte dei commerciali;
- Analisi delle necessità del cliente e dei conseguenti requisiti software;
- Progettazione e realizzazione di nuove applicazioni o nuove funzionalità di applicativi già in uso;
- Verifica e collaudo del software prodotto con finale rilascio nei sistemi del cliente;
- Manutenzione dei contenuti proposti al cliente in un'ottica a lungo termine.

1.3 Processi aziendali

1.3.1 Organizzazione interna

Le grandi dimensioni dell'azienda implicano una forte strutturazione interna e un'attenta gestione delle attività di coordinamento. Durante il mio stage ho avuto modo di osservare alcuni aspetti di questo complesso sistema.

Una delle prime cose che si imparano di quest'azienda è la sua propensione alla cura dei rapporti con il cliente, poiché vengono offerte numerose sessioni di consulenza. La filosofia è quella di collaborare per aiutarli a trasformare i loro sistemi informativi e, grazie all'esperienza del settore, offrire valore aggiunto mediante le soluzioni. Per raggiungere tale scopo il gruppo ha stretto delle partnership strategiche con Microsoft, IBM, Oracle e HP. La missione principale del gruppo è di industrializzare e ottimizzare le proprie operazioni per migliorare la competitività e le performance in un'ottica a lungo termine.

Un altro aspetto a cui l'azienda tiene in particolar modo è la gestione delle risorse umane. Anticipare e sostenere lo sviluppo dell'evoluzione di queste ultime è considerata una priorità per il successo aziendale e per mantenere un alto livello di soddisfazione e di motivazione dei dipendenti. Per questo Sopra Steria si impegna a conoscere i profili e le competenze di ciascun collaboratore, al fine di poter offrire agli stessi prospettive di crescita e percorsi di carriera in grado di soddisfare sia le loro aspettative che il mercato.

Per quanto riguarda il governo e la gestione del gruppo, i diversi livelli di poteri decisionali, sia a livello funzionale che produttivo, sono distribuiti nella gerarchia operativa oltre che nella direzione. Alla base di ciò, un'organizzazione complessa si ramifica nelle varie nazioni in cui l'azienda si estende, delegando l'amministrazione di questi filoni e di altri reparti di supporto a manager selezionati. A livello più basso si collocano le Business Unit, ovvero le varie divisioni aziendali adibite all'erogazione di determinate tipologie di prodotti e servizi identificate anche in base al mercato di riferimento. Anche queste ultime risultano distribuite nel territorio, in ogni filiale infatti possono coesistere più reparti.

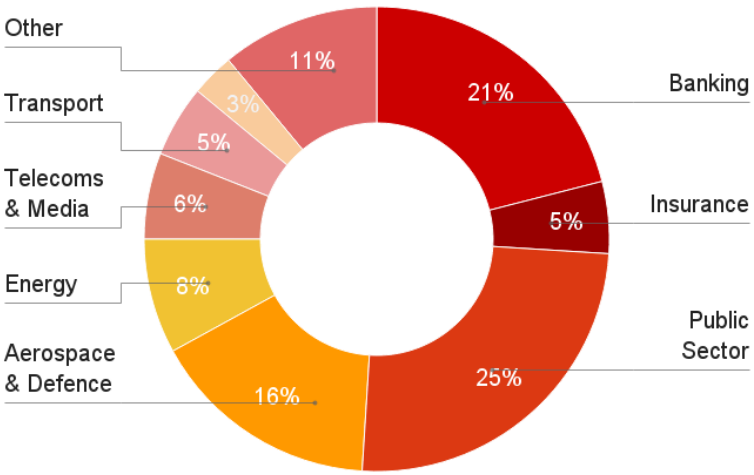

Figura 1.2: Suddivisione dell'azienda nei vari mercati

Nella divisione in cui sono stato collocato vi sono diverse figure che si occupano dei vari processi di produzione. In ordine gerarchico è presente un direttore di Business Unit per l'amministrazione delle risorse della divisione, i project manager per la gestione dei progetti e dei loro costi, i commerciali che si occupano delle relazioni con i clienti in ambito di redazione dei contratti, i team di analisti e consulenti che si occupano dei requisiti del cliente e i team di sviluppo software, suddivisi in programmatori web e sviluppatori *host*.

L'azienda adotta un sistema di qualità anche per i suoi funzionamenti interni, in particolare è applicato per:

- Gestire i processi di prevendita e sviluppo di progetti e servizi;
- Gestire le risorse umane nel campo di reclutamento, carriera, gestione delle competenze, formazione interna e comunicazioni interne;
- Gestire e monitorare i processi aziendali, garantendo una manutenzione interna.

Sopra Steria si fa carico inoltre delle responsabilità d'impresa negli ambiti di uno sviluppo sostenibile, cura dell'ambiente, diritti umani e del lavoro e nel favorire le uguali opportunità. La pubblicazione del rapporto di tali attività fa parte di un processo di trasparenza, correttezza e dialogo con gli *stakeholder*: collaboratori, clienti, azionisti, fornitori, partner e attori della società civile.

1.3.2 Modello Incrementale

Il ciclo di sviluppo software adottato da Sopra Steria nella divisione dei Servizi Finanziari è un'implementazione del modello incrementale, questa scelta è dovuta al fatto che l'azienda tratta per la maggior parte progetti già avviati, che richiedono aggiunte sulla base delle funzionalità essenziali già sviluppate.

I punti di forza del procedimento incrementale sono i seguenti:

- L'integrazione delle parti del sistema è distribuita nel tempo e non collassata nelle fasi finali;
- Ogni incremento porta valore aggiunto, con lo sviluppo di nuove funzionalità e il soddisfacimento di alcuni requisiti;
- Ad ogni incremento si guadagnano esperienza e affidabilità, riducendo i rischi di fallimento;
- Le funzionalità essenziali sono sviluppate nei primi incrementi e attraversano più fasi di verifica, diventano quindi più stabili con ciascuna iterazione;

Questo modello si presta bene alle necessità dell'azienda perché i clienti richiedono che vengano effettuati lavori di manutenzione e amplificazione definibili in attività distinte, assimilabili facilmente tramite un ciclo di sviluppo ad incrementi. In figura vengono rappresentate le fasi del modello.

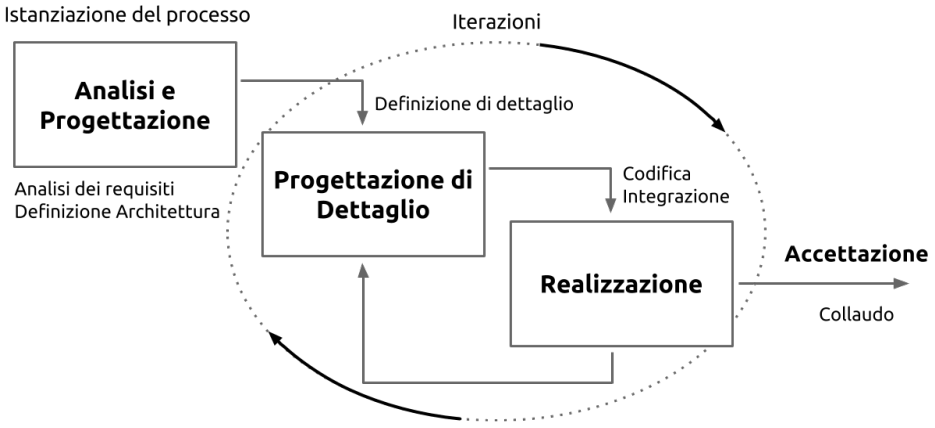

Figura 1.3: Diagramma del modello di sviluppo incrementale

Il modello incrementale è un ciclo di sviluppo definito dallo standard ISO 12207 che combina la logica del modello a cascata, dove ogni fase è rigidamente sequenziale, e la filosofia iterativa della prototipazione.

È prevista una prima fase di analisi dei requisiti fondamentali e di progettazione architetturale intesa a stabilire le fondamenta del software. Tale fase è essenziale per definire i successivi incrementi e non si ripete.

Le fasi successive di realizzazione incrementale vera e propria, possono ripetersi più volte e mirano ad attività di progettazione di dettaglio, codifica e prove, in cui vengono trattati prima i requisiti essenziali e poi quelli desiderabili. Le implementazioni subiscono i trattamenti di integrazione e collaudo, successivamente avviene un eventuale rilascio.

È prevista la prototipazione delle nuove funzionalità che si vanno ad implementare per la validazione complessiva del sistema, reiterando alle fasi di progettazione e realizzazione in caso di errori o problematiche. In questo modo è possibile di volta in volta acquisire maggiore competenza riguardo al problema, riducendo i rischi successivi e le tempistiche globali di produzione software.

1.3.3 Il modello incrementale in Sopra Steria

Ogni ciclo di incremento inizia con la raccolta e l'analisi dei requisiti presso il cliente, che espone le sue necessità tramite riunioni oppure mediante opportuna documentazione.

Gli analisti a questo punto si occupano di raggruppare i requisiti in macro attività, calcolare le tempistiche necessarie per il loro completamento e stilare i documenti di analisi funzionale e specifica tecnica per ognuna di esse, dov'è compresa la progettazione di dettaglio. Tale documentazione risulta necessaria ai vari team di sviluppo per la comprensione e l'applicazione delle implementazioni richieste.

Gli analisti rimangono a disposizione degli sviluppatori anche nelle fasi successive per eventuali chiarimenti e specificazioni, in modo da non rallentare o interrompere le fasi successive. I documenti vengono inviati ai team competenti a cui sono state attribuite le macro attività e da quel momento inizia la realizzazione. Tali gruppi di lavoro possono risultare distribuiti nelle varie sedi del territorio italiano, perciò sono previste molte comunicazioni telefoniche o tramite posta elettronica e occasionali trasferte; al fine di allineare le procedure di sviluppo o rendere noto quando è possibile procedere con determinate modifiche.

L'evoluzione degli incrementi software attraversa ambienti distinti in base alle mansioni svolte su di essi. Esistono in particolare i seguenti ambienti:

- **Sviluppo**: ambiente di programmazione locale, qui avviene l'implementazione delle modifiche software;
- **Integrazione**: in questo ambiente vengono raccolte le implementazioni delle attività e si verifica che non generino conflitti, garantendo la stabilità del sistema;
- **Collaudo**: ambiente di validazione delle funzionalità complessive del software, utilizzato anche per dimostrare al cliente la loro consistenza;
- **Produzione**: questo ambiente varia per ogni cliente o applicazione sviluppata e rappresenta lo stato finale del prodotto in cui viene effettivamente utilizzato dal cliente.

È responsabilità del programmatore che prende in carico lo sviluppo delle funzionalità dichiarare il loro completamento, almeno a livello di prototipo, per rilasciarlo in integrazione. Determinati team si occupano poi di testare l'applicazione nelle sue nuove funzioni, accertando il soddisfacimento dei requisiti ed eventualmente contattando gli analisti per eventuali modifiche progettuali. In caso di problematiche le modifiche vengono respinte in ambito di sviluppo altrimenti vengono approvate per il collaudo. In collaudo è possibile utilizzare le funzioni sviluppate da altri team e validare il lavoro svolto per presentarlo poi al cliente, rilasciando in produzione la nuova versione del software.

1.3.4 Strumenti a supporto di processi e servizi

Linguaggi

Sviluppando principalmente applicativi web, nella Business Unit in cui ho svolto il tirocinio è stata adottata la piattaforma Java per il web (Java EE) e ovviamente le tecnologie standard relative alla presentazione e al comportamento delle pagine.

Ho utilizzato quindi anche HTML, CSS e JavaScript che nella quasi totalità dei casi sono generati dinamicamente lato server tramite pagine JSP.

Java Platform Enterprise Edition, o Java EE, è un'estensione di Java SE (Standard Edition) e rappresenta una piattaforma di sviluppo software molto usata per applicazioni d'impresa. Java EE è mantenuto mediante il Java Community Process ovvero un processo che permette ad una comunità di esperti industriali, organizzazioni (commerciali e *open source*) e ad un

incredibile numero di individui di dare il loro contributo seguendo determinati standard. Esso fornisce gli strumenti utili per la programmazione Web, tra cui un ambiente *runtime* e librerie utili allo sviluppo di applicazioni distribuite, scalabili, affidabili e sicure che prevedono Java come linguaggio di programmazione primario.

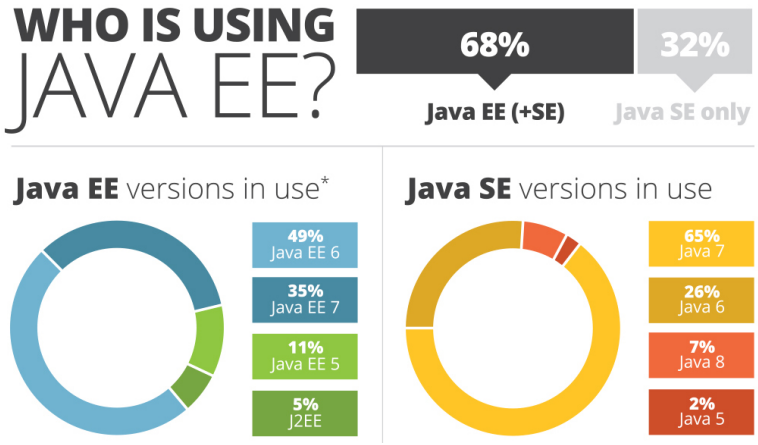

Figura 1.4: Utilizzi della piattaforma Java nelle sue versioni - Fonte: RebelLabs

Framework

Per definire la spina dorsale delle applicazioni web, invece, l'azienda utilizza diversi framework tra cui Struts, Maven e Hibernate. Apache Struts è un framework *open source*, usato nel progetto di stage, per lo sviluppo di applicazioni web su piattaforma Java EE, gestisce le richieste client e smista il flusso applicativo in base alla logica configurata mediante file XML, dove vengono definite le associazioni tra i vari elementi che compongono il sistema, sotto forma di *action*.

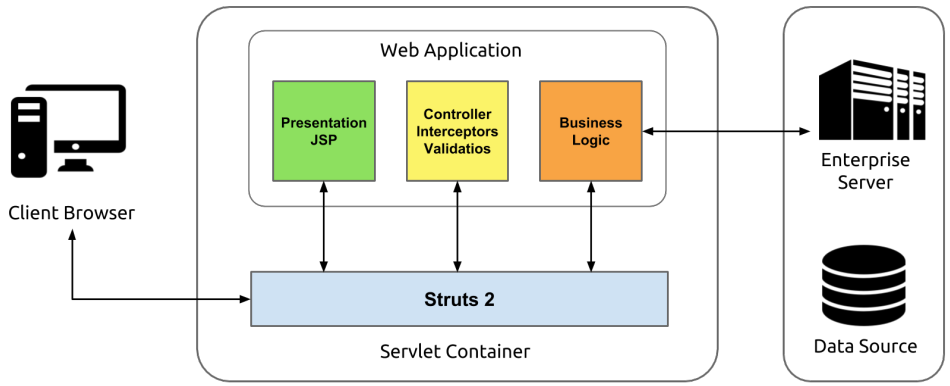

Figura 1.5: Architettura di un'applicazione web che adotta il framework Struts

L'utilizzo di Struts permette lo sviluppo di applicazioni web di notevoli dimensioni, inoltre agevola la suddivisione dello sviluppo del progetto fra i vari dipendenti. I programmatori web e i vari gruppi di sviluppatori possono quindi gestire in parallelo e autonomamente la loro parte del progetto.

Database e applicazioni host

Il salvataggio dei dati per le applicazioni in ambito bancario e assicurativo avviene solitamente tramite DBMS$_G$ relazionali come DB2 dell'IBM, Microsoft SQL Server e MySQL di Oracle. DB2 è nato nel 1983 ma tutt'oggi è uno tra i DBMS più usati, specie in questo settore. In origine era nato come DBMS per i mainframe CICS$_G$ poi si è diffuso su qualsiasi tipo di server. Per questo banche e assicurazioni, enti che esistono da molto prima, inizialmente hanno adottato questa tecnologia mediante sistemi EIS$_G$ implementati in linguaggio COBOL$_G$ che tutt'oggi gli forniscono le funzionalità necessarie senza il bisogno di adottare tecnologie più recenti e sviluppate secondo le esigenze dei moderni paradigmi di programmazione.

Microsoft SQL Server e MySQL di Oracle sono utilizzati come soluzioni secondarie per la gestione di dati in sola lettura di natura statica.

Gestione di progetto

A supporto della gestione delle attività progettuali e dell'organizzazione delle comunità aziendali, Sopra Steria mette a disposizione dei suoi dipendenti un portale comune dove poter ottenere informazioni sulla vita aziendale e la gestione dei gruppi di lavoro.

Il portale aziendale, Face2Face, gestisce molteplici attività e problematiche. Tramite esso i dipendenti sono tenuti a riportare settimanalmente le proprie attività di lavoro e gli ambiti di progetto al fine di inviare i dati alla direzione, permettendole di coordinare le risorse a disposizione. Il sito consente inoltre di consultare le news aziendali e gli eventi organizzati, come ad esempio i corsi di formazione interna a cui gli sviluppatori sono invitati a partecipare.

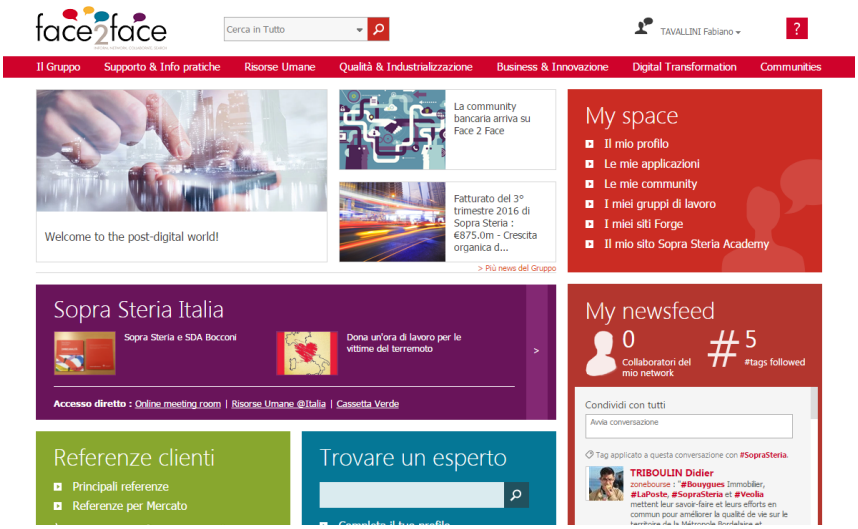

Figura 1.6: La pagina iniziale di Face2Face - Fonte: portale interno dell'azienda

Face2Face è accedibile anche da rete remota come anche il sistema di posta elettronica, facilitando il lavoro in trasferta, e consente anche di inviare ai tecnici informatici segnalazioni riguardo i propri strumenti di lavoro per ricevere assistenza.

Oltre a questo, i vari team di progetto comunicano molto spesso via email utilizzando Microsoft Outlook e telefonicamente per coordinarsi nei lavori. I capi progetto inoltre mantengono l'organizzazione delle attività, confrontandosi con gli analisti, utilizzando Microsoft Excel come strumento di supporto e stesura dei piani di lavoro.

Documentazione

Per ogni attività risultante dall'analisi dei requisiti vengono redatti due documenti: l'Analisi Funzionale e la Specifica Tecnica. Il primo affronta i requisiti ad alto livello, enunciando le principali funzionalità ed i cambiamenti rispetto alla versione attualmente in produzione dell'applicativo.

Il secondo documento affronta nel dettaglio gli aspetti tecnici che vanno modificati o aggiunti trattando principalmente i programmi COBOL_G lato *host*, dai quali poi anche i programmatori web possono individuare i parametri da utilizzare nelle richieste via rete per recuperare i dati e quindi allinearsi. Si può quindi dire che per la programmazione delle interfacce risulta di primaria importanza il primo documento.

Al termine dello sviluppo dei requisiti, ad avvenuta validazione, viene inoltre redatto un documento di collaudo da consegnare al cliente per accertare i lavori eseguiti.

Il software utilizzato per la produzione dei documenti è Microsoft Word, i cui formati sono standard sia per l'azienda che per i clienti.

Sistemi di versionamento

Nei diversi prodotti software che la Business Unit ha sotto la propria responsabilità vengono utilizzati principalmente due sistemi di versionamento del software:

- **SVN (Subversion, Apache)**: è un software di versionamento del codice distribuito gratuitamente sotto licenza Apache. E' ampiamente usato e sviluppato da una comuinità globale di collaboratori. Offre alcune funzionalità chiave per chi lo sceglie come *commit* visti come operazioni atomiche, operazioni di *branch*, *changelists* e design client-server;

- **RTC (Rational Team Concert, IBM)**: è costruito su IBM Jazz, una piattaforma estensibile che aiuta i team a integrare i task attraverso il ciclo di vita del software. Ha un'architettura client-server e permette ai team di sviluppo di tenere traccia del loro lavoro tramite *work items*, *source control*, *reporting* e *build management* in un singolo ambiente.

Ambienti di sviluppo

Il principale ambiente di sviluppo adottato per le applicazioni web è Eclipse. Esso racchiude la globalità delle caratteristiche necessarie ad uno sviluppatore in questo ambito.

Rappresenta un'ottima soluzione e agevolazione per il processo di sviluppo, in quanto offre funzionalità di collegamento ai sistemi di versionamento, *debugging* del codice *runtime* e avvio del software mediante gli *application server* che si desidera installare, oltre alle molteplici caratteristiche offerte dai comuni editor di testo orientati allo sviluppo dei sorgenti software.

Figura 1.7: I vantaggi dell'uso di Eclipse in collaborazione con RTC

Altri programmi di supporto sono invece i diversi browser in cui bisogna testare il funzionamento delle pagine web tra cui Internet Explorer, Firefox e Chrome e gli editor di testo utili in situazioni dov'è richiesta più praticità come Notepad++.

Sistemi operativi

Per le postazioni di sviluppo è previsto un sistema centralizzato di utenze a cui è attribuito un proprio ambiente di lavoro ed uno spazio assegnato a cui accedere da qualsiasi computer aziendale.

Nelle macchine aziendali è consueto l'utilizzo di Windows 7 come sistema operativo primario per ovviare a discrepanze nelle postazioni dei diversi dipendenti e per omogenizzare ulteriormente il processo di sviluppo. Questo comporta una buona soluzione per concentrarsi unicamente sul proprio lavoro e avere al contempo una garanzia nell'utilizzo quotidiano.

1.4 Clientela e trasformazione digitale

1.4.1 Target

I target di Sopra Steria, per quanto riguarda la divisione dei Servizi Finanziari e Assicurazioni, sono principalmente gruppi bancari e assicurativi. Essi necessitano una qualche forma di innovazione o evoluzione che le garantisca continuità di produzione ma anche i corretti adeguamenti previsti dai cambiamenti legislativi.

Per permettere questo, gli analisti si incaricano di entrare in contatto con i responsabili ICT_G della società cliente, dai quali poi si ricavano diverse richieste implementative; dalla variazione di qualche caratteristica alla creazione di funzionalità completamente nuove interne o meno all'applicativo che essi adottano.

1.4.2 Innovazione

L'innovazione in ambito bancario e assicurativo rappresenta uno scoglio non indifferente, in quanto questi enti sono da sempre legati a tecnologie primordiali come il linguaggio COBOL$_G$ e la relativa implementazione in mainframe CICS$_G$: si è preferito infatti non migrare da essi o evolvere in altra tecnologia software.

Per l'azienda, che fa della trasformazione digitale la sua bandiera, questo rappresenta una sfida e desidera mettersi in gioco offrendo le soluzioni adeguate, tenendo conto delle priorità del cliente e delle sue possibilità. Queste caratteristiche sono molto ricercate dalle aziende che vogliono rinnovarsi, trasformando i loro processi e servizi nel mondo digitale, adeguandosi ai moderni canoni di utilizzo e facendosi avanti nei mercati, con la possibilità di offrire prodotti di maggiore qualità e raggiungere molti più clienti.

Fortunatamente per quanto riguarda il fornt-end$_G$ degli applicativi, l'innovazione si fa strada mediante l'utilizzo di tecnologie adatte alla presentazione dei contenuti nel web e di conseguenza maggiormente inclini a spinte evolutive dovute alla modernizzazione degli standard. Lo sviluppo di nuovi requisiti imposti dai clienti, inoltre, è sempre gestito in un'ottica che mira all'aggiornamento, anche per quanto riguarda le tecnologie e le loro versioni di utilizzo, per garantire, dove possibile, una futura compatibilità dove possibile.

Capitolo 2

Il Progetto

2.1 Interesse aziendale nello stage

Sopra Steria Group S.p.A. è un'azienda ben strutturata e affermata in molti settori, nonostante conti diversi anni alle sue spalle è in costante espansione e rinnovazione.

Il suo ruolo svolto in Italia non fa eccezione, per questo l'area personale è sempre in cerca di nuove risorse da inserire nelle diverse Business Unit. Non mi ha stupito quindi il fatto che abbia attivato una convenzione con l'Università di Padova e partecipi agli eventi STAGE-IT per accogliere iniziative di stage sia curricolare che retribuito.

La politica aziendale prevede una decorrenza di sei mesi per completare il ciclo di stage. Successivamente, nella maggioranza dei casi, vi è la propensione all'assunzione, in quanto la nuova risorsa è considerata pronta per essere effettivamente inserita nei team di sviluppo.

Il tirocinio è quindi visto dall'azienda come uno strumento utile a contribuire alla selezione di nuovi talenti e verificare che da entrambe le parti vi sia un interesse a proseguire il rapporto lavorativo.

2.2 Il progetto all'interno dell'azienda

Il corso di studi rende obbligatorio lo svolgimento di uno stage che deve ricoprire un ammontare di 300-320 ore totali. È stato dunque stabilito che il lavoro di stage doveva svolgersi in 320 ore nell'arco di 8 settimane: i tempi di consegna delle funzionalità relative al progetto richiesti dall'azienda superavano la data prevista di fine stage, ciò ha permesso che lo svolgimento del progetto non subisse ulteriori vincoli temporali.

Il progetto di stage consiste nello studio delle tecnologie necessarie alla programmazione web in ambito Java EE e l'implementazione di funzionalità aggiuntive per un applicativo web per la gestione di finanziamenti in ambiente bancario. Esso è funzionante da anni presso il cliente e continuamente oggetto di evoluzioni. Tale prodotto segue l'architettura Java EE ed il suo sviluppo è vincolato alla compatibilità richiesta dal cliente per i suoi operatori di sportello, che effettivamente lo utilizzano in ambiente di produzione.

Dopo una prima fase di formazione sulle tecnologie ed una di esercitazione secondo i metodi aziendali di sviluppo, quindi, il lavoro di stage si è concentrato su questo software, chiamato ELISE. Ho portato avanti il suo ampliamento aggiungendo le nuove funzionalità il cui sviluppo mi è stato assegnato.

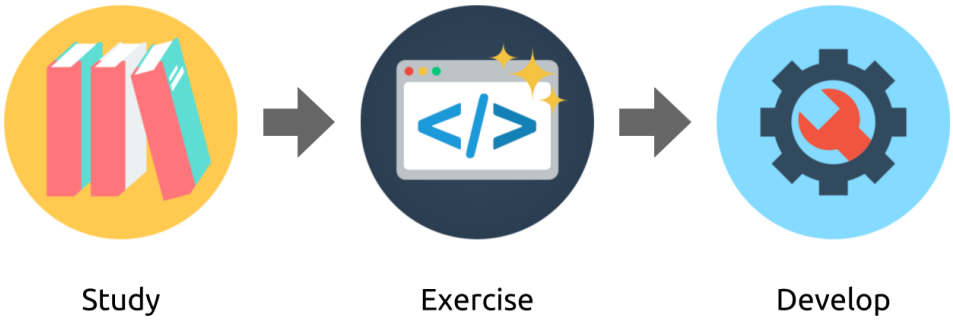

Figura 2.1: Le fasi principali del progetto di stage

2.2.1 ELISE

ELISE (Extended Loans Integrated System) è un sistema per la gestione integrata di tutte le problematiche di business relative all'area dei finanziamenti. È in sviluppo per il Banco Popolare, il più grande gruppo bancario in Italia a matrice cooperativa, nato ufficialmente il 1° luglio 2007 dalla fusione fra il Banco Popolare di Verona e Novara e la Banca Popolare Italiana.

ELISE si basa su un accesso ad utenza a cui sono predisposte delle abilitazioni. Ogni utente appartiene ad una filiale operativa e risulta responsabile di determinate attività da svolgere mediante l'applicazione: richiesta di attivazione finanziamenti, verifica della documentazione, gestione pratiche di accollo o surroga, ecc. Normalmente gli utenti hanno solo determinate funzioni abilitate per sicurezza, la sede centrale invece possiede tutti i privilegi di operatività.

I sistemi informativi di produzione in cui risiede l'applicazione web, sono in gestione presso una società ICT$_G$ di terze parti. Tramite i loro server, il software è configurato alla comunicazione con un altro ambiente, quello di gestione dei dati, implementato su mainframe CICS$_G$ basato appunto sulle transazioni dati.

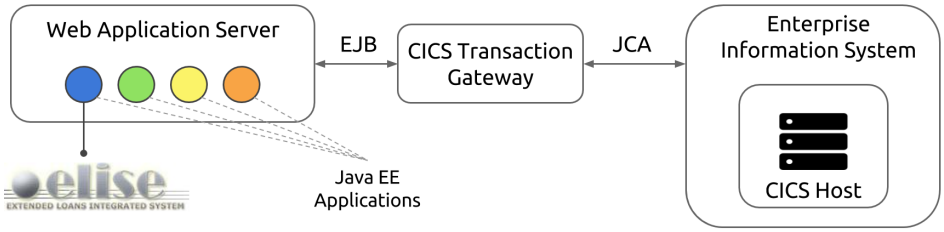

Figura 2.2: Architettura dei sistemi di comunicazione tra ELISE e l'ambiente CICS, implementata mediante Enterprise JavaBeans$_G$ e le tecnologie CTG$_G$ e JCA$_G$

L'applicazione web offre innumerevoli funzionalità in ambito finanziario, tra le più importanti:

- Stipula di preventivi finanziari sulla base dei tassi in vigore e dei dati dichiarati;
- Richiesta di attivazione di un nuovo finanziamento a partire da un preventivo;
- Gestione di un finanziamento nei suoi passi per effettuare l'erogazione e attivazione delle pratiche attribuite alle diverse filiali. Consentendo la stampa e la raccolta della documentazione richiesta e la verifica dei dati del finanziamento;
- Gestione dei prodotti finanziari e dei loro parametri di periodicità, rateizzazione e spese di gestione da proporre ai clienti del gruppo bancario;

- Gestione dei tassi da applicare ai finanziamenti e della loro struttura fissa o variabile;
- Gestione delle convenzioni stipulate per enti i cui parametri finanziari differiscono dal comune;
- Accesso a numerose utilità di amministrazione, calcolo di rate, tassi o piani di ammortamento e vendita a privati, imprese e agevolati.

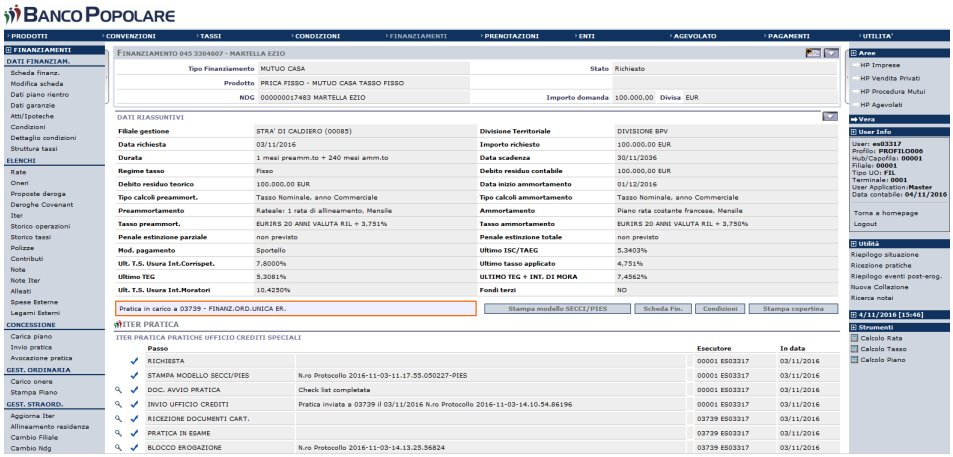

Figura 2.3: Interfaccia dell'applicazione ELISE - Fonte: Ambiente di sviluppo dell'applicazione

Il portale viene utilizzato ogni giorno dai dipendenti di ogni filiale del gruppo bancario. Rappresenta quindi un mezzo di nota importanza per il cliente e l'erogazione dei suoi servizi.

2.2.2 Finalità del progetto

L'azienda ha l'obiettivo di espandersi e aumentare il proprio organico anche attraverso l'attivazione di stage con l'ottica di una futura assunzione; per questo i tirocinanti vengono inseriti nei team di sviluppo dei progetti già presi in carico per conto dei clienti. Il mio caso non ha fatto eccezione.

Nel momento del mio inserimento in azienda, il progetto prevedeva nuove funzionalità da sviluppare. Il tutor aziendale ha quindi ritenuto adeguata una selezione di tali attività per caratterizzare il lavoro di stage, in previsione del fatto egli sarebbe stato presente per guidarmi soprattutto nelle prime fasi e successivamente educarmi all'autonomia.

Il tutor si impegna in tal senso sia argomentando e motivando le scelte implementative che lo stagista è tenuto a prendere, sia lasciando la libertà di studiare soluzioni e alternative ai problemi proposti.

Le attività che mi sono state assegnate nell'ultima fase di stage, dopo la formazione teorica e pratica, sono:

- **Aggiornamento stampe preventivo**: prima dell'erogazione di un finanziamento, il cliente è tenuto a valutare le offerte dei prodotti calcolandone i preventivi. L'applicazione già si occupa di generare i PDF di tale documentazione, ma vi era il bisogno di adeguare tali stampe a nuovi parametri e valori assicurativi.
- **Visualizzazione scadenziario covenant**$_G$: i finanziamenti bancari possono comprendere dei valori di convenzione attribuiti ad un certo periodo di validità. Era esigenza della banca poter visualizzare in modo intuitivo, con un calendario, le scadenze previste nei vari giorni e poterle analizzare nel dettaglio.
- **Composizione titolari di un finanziamento**: era richiesto di creare una nuova pagina di visualizzazione dei titolari dei finanziamenti con la possibilità di modificarne, oltre a determinate categorie fiscali, la percentuale di titolarità. Inoltre dovevano essere implementate le logiche secondo cui, dopo la modifica ed il salvataggio dati, sarebbe dovuta apparire la possibilità di stampare la documentazione di sollevamento delle responsabilità da far firmare ai titolari.
- **Revisione funzionalità di ricerca tassi**: quest'attività prevedeva di implementare un filtro sul periodo di validità del valore dei tassi, per ottimizzare le performance della pagina di ricerca e presentazione dei risultati.
- **Ristrutturazione layout di una procedura**: nella procedura di attivazione di un finanziamento, un particolare passo di avanzamento riguarda la chiamata di un web service esterno, incluso nella pagina dell'applicativo mediante un iframe$_G$. Era richiesto di strutturare la presentazione di tale finestra e in particolare di sviluppare delle funzioni JavaScript per renderla ridimensionabile e trascinabile all'interno della pagina.

In questo percorso di espansione del software, l'azienda dispone di una risorsa in più per attuare il proprio lavoro e il soddisfacimento delle richieste del cliente riguardo il prodotto. Il tirocinante ne sarebbe quindi risultato maturato e munito di nuove conoscenze e abilità a lui utili ad un futuro lavorativo.

L'applicativo è stato provvisto quindi di numerose aggiunte e modifiche utili all'adeguamento a nuovi termini legali in vigore, al miglioramento delle performance e della stabilità operativa e ovviamente alla predisposizione di nuove funzioni che amplificano il raggio d'azione e l'utilità di utilizzo per conto degli utenti.

2.2.3 Vincoli di progetto

Imposti dal cliente

Per quanto riguarda ELISE, il cliente ha esposto diversi vincoli per lo sviluppo.

Alcuni sono impliciti nell'utilizzo di tecnologie, in particolare per interfacciarsi all'ambiente di gestione dati. L'applicazione adotta alcune librerie basate sulla Reflection$_G$ per la codifica e la decodifica dei flussi dati, per la generazione di JavaBeans$_G$ appositi. Un ulteriore vincolo tecnologico è rappresentato dalla necessità di mantenere compatibile il software per i browser utilizzati nelle filiali bancarie.

Altri sono specificati ad ogni nuova richiesta di espansione dell'applicazione. Possono riguardare aspetti grafici, come la posizione di determinati elementi, oppure aspetti tecnici, come la necessità di garantire determinate performance per alcune funzionalità.

Per il progetto ELISE e in particolare nelle attività a me attribuite, i vincoli riguardavano implementazioni specifiche:

- La necessità di ridurre il carico della pagina di visualizzazione dei valori tassi, implementando un filtro sul periodo di validità;
- Per la gestione dei titolari di un finanziamento il vincolo era di generare i documenti di stampa seguendo gli standard dell'applicazione;
- Per il layout dell'iframe$_G$, utile alla visualizzazione del servizio esterno, era richiesta la compatibilità con Microsoft Edge per garantire le prestazioni;
- Per lo scadenziario dei valori covenant$_G$, i vincoli riguardavano solo l'aspetto che doveva avere la pagina del calendario, per questo ho avuto una buona libertà di sviluppo.

Imposti dall'azienda

Dal punto di vista dell'azienda, l'obiettivo è sempre quello di innovare e apportare trasformazioni mirate all'adeguamento delle tecnologie utilizzate.

Per questo motivo, a livello di sviluppo delle nuove funzionalità, vengono applicate sempre le ultime versioni delle tecnologie possibili, cercando di contrastare l'arretratezza dei meccanismi di programmazione web in ambito bancario.

Sopra Steria però deve adottare anche le giuste precauzioni per far fronte alle spese dei progetti, per questo alle volte non è stato possibile impiegare troppo tempo per l'implementazione dei requisiti e di conseguenza pensare all'impiego di tecnologie più moderne.

Anche tra i vincoli imposti dall'azienda, per il mio progetto di stage, vi sono stati diverse implicazioni tecnologiche.

L'intera applicazione infatti si basa sull'architettura Java EE e il sistema di versionamento usato per coordinare i team di sviluppo e integrare il loro lavoro, è RTC. Di conseguenza ho dovuto studiare, imparare e mettere in pratica le tecnologie previste da tale architettura, come i JavaBeans$_G$ le classi Servlet$_G$, le pagine JSP, i tag JSTL oltre ai linguaggi standard per la programmazione web HTML, CSS e JavaScript.

2.3 Il mio stage in Sopra Steria

Grazie alle offerte presenti a STAGE-IT 2016 e a molte altre che ho ricevuto, ho potuto selezionare le realtà lavorative che erano più orientate verso i miei criteri e ho deciso di partecipare al colloquio collettivo con Sopra Steria.

In particolare erano presenti circa 20 ragazzi che come me erano in procinto di laurearsi e volevano assicurarsi un posto in questa azienda. I recruiter hanno valutato il mio interesse e il potenziale che rappresentavo per loro nell'ambito sviluppativo in generale e il feedback è risultato positivo. Contando che per la sede di Padova erano a disposizione solamente due posti per attivare un tirocinio, mi sono sentito premiato e soddisfatto della mia scelta.

Il mio stage in Sopra Steria consisteva nello sviluppo di nuove funzioni per un applicativo web già operativo presso il cliente. Ho sviluppato tali funzionalità modificando e aggiungendo codice sorgente utilizzando Java EE e le tecnologie che mette a disposizione. Mi sono servito anche di Eclipse per agevolarmi nella realizzazione del progetto.

2.3.1 Motivo della scelta

Da quando mi sono avvicinato al mondo dell'informatica ho avuto il desiderio di toccare con mano situazioni di implementazione reale delle tecnologie che ho studiato, in particolare nell'ultimo anno dove ho avuto modo di trattare linguaggi e metodologie di programmazione moderni.

Mi interessava quindi fare esperienza in ambito lavorativo e volevo distaccarmi da piccole realtà di sviluppo, per questo quando ho partecipato a diversi colloqui con aziende che cercavano stagisti per i loro progetti, ho avuto un occhio di riguardo all'ambiente in cui avrei lavorato.

Ho tenuto conto inoltre del fatto che probabilmente in un secondo momento avrei potuto iniziare la mia carriera in tale contesto, come reso noto dalla maggioranza delle aziende che al giorno d'oggi cercano di ampliare il loro organico nel settore informatico. Questo è stato un ulteriore motivo che mi ha fatto cercare un'azienda di rilievo.

Dei vari progetti che mi sono stati presentati, molti trattavano tecnologie certamente più innovative, ma Sopra Steria è stata comunque più convincente sotto il punto di vista a lungo termine, offrendomi la possibilità in futuro di cambiare settore e ambito lavorativo in base alle mie esigenze e preferenze.

2.3.2 Obiettivi pianificati

All'interno del piano di lavoro del tirocinio io e il tutor aziendale abbiamo stabilito degli obiettivi da soddisfare per valutare positivamente l'attività di stage. La maggior parte riguardano lo svolgimento dello stage in azienda, altri invece sono relativi allo sviluppo dei progetti da me svolti.

I requisiti sono stati suddivisi in **obbligatori**: vincolanti in quanto primari e di diretto impatto sulla valutazione dello stage; **desiderabili**: non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiuntivo; **facoltativi**: rappresentanti valore aggiuntivo non strettamente necessario.

Obbligatori

- Studio e acquisizione di padronanza dell'ambiente di sviluppo Eclipse;
- Studio e comprensione della piattaforma Java EE;
- Installazione e utilizzo di diversi *application server* e diverse JVM;
- Acquisizione familiarità con la programmazione web in ambito Java (tecnologie JSP, JSTL, Servlet$_G$);

- Acquisizione tecniche di programmazione con framework Struts;
- Studio e utilizzo del sistema di versionamento RTC;
- Studio e utilizzo della tecnologia AJAX$_G$;
- Implementazione di applicazioni di esempio per le funzionalità basilari;
- Analisi di una complessa applicazione reale in architettura Java EE;
- Integrazione nel team di sviluppo e acquisizione competenze nelle dinamiche di gruppo;
- Comprensione e acquisizione familiarità con la documentazione di analisi e specifica delle attività;
- Implementazione di modifiche basilari dell'applicazione ambito di progetto;
- Implementazione di modifiche complesse dell'applicazione ambito di progetto.

Desiderabili

- Raggiungimento di un buon livello di autonomia nell'utilizzo di Java EE;
- Raggiungimento di un buon livello di autonomia nell'utilizzo di tecnologie web standard (HTML, CSS, JavaScript);
- Raggiungimento di un buon livello di autonomia nell'utilizzo di tecnologie web in ambito Java (JSP, JSTL, Servlet$_G$);
- Acquisizione tecniche di programmazione con framework alternativi (Maven e Hibernate);
- Studio e utilizzo di un sistema di versionamento alternativo (SVN);
- Studio e utilizzo della libreria per le stampe PDF, iText;
- Studio e utilizzo della libreria per il *logging*, Log4J;
- Capacità di portare a termine le attività lavorative secondo le tempistiche stabilite, anche in situazioni critiche;
- Conoscenza delle norme di sicurezza relative all'ambiente di lavoro.

Facoltativi

- Studio delle meccaniche di comunicazione con l'area di business per la gestione dei dati;
- Partecipazione alle attività di test di integrazione dell'applicazione ambito di progetto;
- Partecipazione alle attività di collaudo dell'applicazione ambito di progetto;
- Rilascio delle nuove funzionalità sviluppate nelle attività assegnate.

Il grafico in figura 2.4 mostra la suddivisione degli obiettivi nelle diverse tipologie, facendo notare che quelli obbligatori compongono il 50% dei totali concordati e l'altra metà si suddivide in desiderabili e facoltativi.

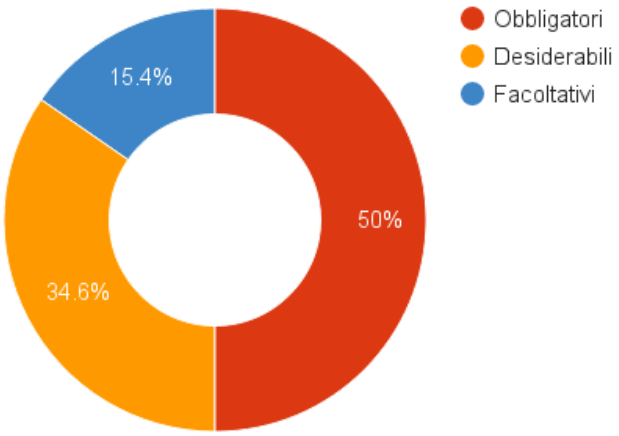

Figura 2.4: Suddivisione degli obiettivi dello stage

2.3.3 Obiettivi personali

Iniziando lo stage, le mie aspettative erano piuttosto semplici: desideravo apprendere nuove tecnologie per la programmazione web, in modo da ampliare le mie conoscenze a riguardo e specializzarmi abbastanza in questo ambito per giocare un ruolo chiave nelle attività dell'azienda.

Mi aspettavo di lavorare inizialmente in un ambiente distaccato dai team di sviluppo per un periodo iniziale, in modo da formarmi adeguatamente e poi entrare in contatto con il mondo lavorativo vero e proprio per dare il mio contributo.

Volevo ampliare profondamente le mie conoscenze riguardo gli ambienti di sviluppo che sarei andato ad utilizzare e le tecniche previste nell'ambito in cui sarei stato inserito: creazione di progetti, inserimento e sviluppo di nuove parti nei prodotti e utilizzo di debugging.

Speravo di andare a conoscere i più diffusi framework e le più diffuse librerie presenti, imparando a utilizzarli al meglio. Il linguaggio Java, impiegato in modo massiccio per questo stage, era già di mia conoscenza grazie al percorso di studi offerto dall'Università, ma la sua applicazione in ambito web mi era ancora sconosciuta e volevo comprendere al meglio i meccanismi e le tecnologie che permettono tale tipo di sviluppi.

Gli altri linguaggi per la realizzazione di pagine web erano già di mia conoscenza, desideravo applicarli in un progetto realmente utilizzato e guadagnare autonomia nella loro applicazione per completare il percorso di apprendimento.

Desideravo formarmi professionalmente, accrescere il mio potenziale in ambito lavorativo e al contempo diventare autonomo nelle mie attività; senza sottovalutare però l'interesse verso i casi implementativi, perché volevo trasformare quello che speravo diventasse il mio futuro lavoro in una passione.

Capitolo 3

Stage

3.1 Pianificazione del lavoro

Per raggiungere gli obiettivi pianificati nel piano di stage e rispettare i requisiti minimi imposti dall'Università, io e il tutor aziendale abbiamo previsto 320 ore di lavoro, distribuite in 8 settimane da 40 ore ciascuna. Ho iniziato lo stage il 26/09/2016 e ho terminato, a causa di un giorno di festività, nel lunedì 21/11/2016, rimanendo in linea con quanto preventivato inizialmente, senza incorrere in imprevisti.

3.1.1 Definizione del piano di lavoro

Un mese prima dell'inizio dello stage ho redatto un Piano di Lavoro, definendo gli obiettivi e la pianificazione delle attività a granularità settimanale, elencando nel dettaglio i compiti da svolgere per ogni fase. Ho specificato inoltre le modalità di interazione col tutor, revisioni di avanzamento e una previsione delle competenze guadagnate dallo stagista alla fine delle attività. Le fasi identificate in tale documento sono:

- **Formazione teorica**: durante la prima fase del percorso formativo il tirocinante viene introdotto alle modalità di approccio alla programmazione web mediante:
  - utilizzo dell'ambiente di sviluppo Eclipse per implementazioni e *debug*;
  - studio della piattaforma Java EE, integrazione di *application server* e utilizzo di diverse JVM;
  - implementazione di interfacce grafiche tramite JSP;
  - studio di framework di uso comune come Struts, Hibernate e Maven;

  - sviluppo delle funzionalità mediante l'utilizzo di linguaggi Java e JavaScript.

- **Formazione pratica**: in questa fase lo stagista inizia a lavorare a stretto contatto con il resto del gruppo di lavoro, imparando come affrontare correttamente le attività da un punto di vista sia tecnico sia analitico. Vengono utilizzati i sistemi di versionamento per lo sviluppo di programmi di esempio, dai più semplici (solo Java) ad una completa applicazione web in tecnologia Java EE, secondo la metodologia di lavoro corretta. Questo periodo termina con l'analisi della struttura di una complessa applicazione reale.

- **Sviluppo su applicazione reale**: analisi e implementazione di modifiche relative ad attività reali dell'applicazione ELISE, progetto di stage, in affiancamento al gruppo di lavoro. Il tirocinante si impegna ad analizzare e successivamente stimare le attività che gli verranno assegnate portandole a compimento al pari di una qualsiasi delle figure del team nelle giuste tempistiche.

Nome	Data d'inizio	Data di fine
Inizio stage	26/09/16	26/09/16
Studio delle tecnologie	26/09/16	07/10/16
Utilizzo dell'ambiente di sviluppo Eclipse	26/09/16	26/09/16
Studio della piattaforma Java EE	27/09/16	02/10/16
Integrazione di application server con Eclipse e utilizzo di diverse JVM	03/10/16	03/10/16
Studio di HTML e JavaScript secondo la tecnologia AJAX	04/10/16	04/10/16
Studio di framework di uso comune: Struts, Hibernate, Maven	05/10/16	06/10/16
Utilizzo di strumenti di versioning: SVN e RTC	07/10/16	07/10/16
Esercitazione pratica	10/10/16	21/10/16
Sviluppo di programmi di esempio semplici utilizzando Java	10/10/16	13/10/16
Sviluppo di una completa applicazione web in tecnologia Java EE	14/10/16	20/10/16
Analisi della struttura di una complessa applicazione reale	21/10/16	21/10/16
Sviluppo applicazione ELISE	24/10/16	21/11/16
Aggiornamento stampe preventivo	24/10/16	28/10/16
Implementazione scadenziario covenant	31/10/16	04/11/16
Ristrutturazione layout di una procedura	07/11/16	08/11/16
Revisione funzionalità di ricerca tassi	09/11/16	10/11/16
Composizione titolari di un finanziamento	11/11/16	21/11/16
Fine stage	21/11/16	21/11/16

Figura 3.1: Pianificazione delle attività di stage

Questa pianificazione dettagliata mi ha permesso di distribuire il carico di lavoro e di verificare l'allineamento tra il lavoro effettivamente svolto e il lavoro pianificato, al termine di ogni settimana.

3.1.2 Livello di autonomia

Durante lo svolgimento dello stage il tutor aziendale è stato disponibile per ogni mia necessità, fornendomi consigli riguardo le tecnologie che andavo ad affrontare, specie nelle prime fasi. Ho lavorato in un ambiente che mi consentiva di essere a stretto contatto con lui in modo da favorire l'interazione e garantire il raggiungimento degli obiettivi prefissati.

Sono state effettuate verifiche di avanzamento sia settimanali che giornaliere, quando ritenuto necessario, con relativa revisione dei prodotti per assicurarsi del corretto punto di completamento rispetto alla pianificazione concordata in partenza.

Una volta presa confidenza con gli ambienti e la strumentazione aziendale, ho avuto modo di agire liberamente nelle mie attività, ricevendo dal tutor solo indicazioni sulla strada da percorrere e sui vincoli da rispettare. Successivamente sono stato capace di lavorare in maniera autonoma, com'era desiderabile aspettarsi, ricevendo solo qualche saltuaria delucidazione.

3.2 Studio degli strumenti di sviluppo

Con il mio arrivo in azienda è iniziata la prima fase del lavoro di stage, ovvero lo studio delle tecnologie adottate dalla divisione Servizi Finanziari per l'adempimento dei suoi scopi. L'obiettivo era quello di conferirmi una formazione teorica da consolidare in seguito con delle prove pratiche a scopo esercitativo.

Nelle prime settimane ho quindi studiato i linguaggi, le tecniche di progettazione e gli ambienti di sviluppo che assieme al tutor aziendale avevo pianificato di trattare nel piano di lavoro. Nelle settimane successive avrei poi messo in pratica e ottenuto padronanza di tali tecnologie secondo le metodologie aziendali, approfondendo le conoscenze su tali argomenti, in preparazione alla fase di sviluppo sull'applicazione ELISE.

3.2.1 Tecnologie utilizzate

Piattaforma web

Java Platform Enterprise Edition, o Java EE, è un'estensione di Java SE (Standard Edition) e rappresenta una piattaforma di sviluppo software molto usata per applicazioni d'impresa.

Java EE è sviluppato mediante il Java Community Process, ovvero un processo che permette ad una comunità di esperti industriali, organizzazioni (commerciali e *open source*) e un'infinità di individui di dare il loro contributo seguendo determinati standard. Esso fornisce gli strumenti utili per la programmazione web, tra cui un ambiente *runtime* e librerie utili allo sviluppo di applicazioni distribuite, scalabili, affidabili e sicure che prevedono Java come linguaggio di programmazione primario.

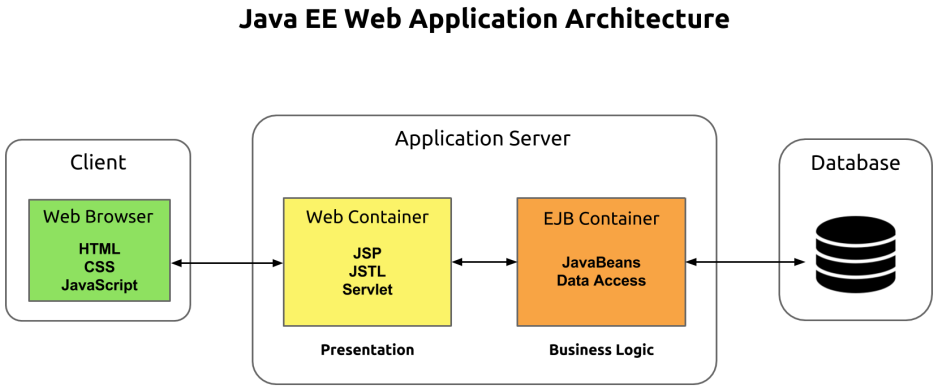

Figura 3.2: Architettura di un'applicazione sviluppata in Java EE

Utilizzando Java EE risulta fondamentale l'utilizzo di un *application server* (o *servlet*$_G$ *container*) per l'esecuzione e la distribuzione dell'applicativo nei diversi nodi della rete. Si tratta di un ambiente che estende le funzionalità offerte da un normale Web Server, ovvero il paradigma client-server e la comunicazione dei contenuti mediante protocolli web. Esso è strutturato nei diversi livelli architetturali (architettura multi-tier) di cui un'applicazione web ha bisogno per eseguire:

- **Presentation layer**: rappresenta la logica di presentazione delle pagine web dell'applicazione;
- **Business layer**: strato della logica funzionale dell'applicazione, utile per la generazione di contenuti dinamici;
- **Persistent layer**: ovvero la gestione dei dati e della loro persistenza.

Sono disponibili diversi *application server* tra cui JBoss (RedHat) e WebSphere (IBM) e servlet container come Tomcat (Apache).

La piattaforma Java EE adotta una convenzione per la configurazione mediante XML delle componenti dell'applicazione, in particolare il file web.xml (contenuto in *WebContent/WEB-INF/*) contiene le impostazioni principali riguardanti la gestione delle richieste inviate al server.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3    <display-name>ProvaWeb</display-name>
4    <welcome-file-list>
5      <welcome-file>index.jsp</welcome-file>
6    </welcome-file-list>
7    <servlet>
8      <description></description>
9      <display-name>Servlet</display-name>
10     <servlet-name>Servlet</servlet-name>
11     <servlet-class>it.soprasteria.provaweb.servlet.Servlet</servlet-class>
12     <init-param>
13       <description></description>
14       <param-name>param</param-name>
15       <param-value>10</param-value>
16     </init-param>
17   </servlet>
18   <context-param>
19     <param-name>propertyFile</param-name>
20     <param-value>config.properties</param-value>
21   </context-param>
22   <servlet-mapping>
23     <servlet-name>Servlet</servlet-name>
24     <url-pattern>/Servlet</url-pattern>
25   </servlet-mapping>
26 </web-app>
27
```

Figura 3.3: Un esempio di configurazione imposta tramite il file web.xml - Fonte: ambiente di sviluppo in Eclipse

Quando si compila un applicazione Java EE viene generato un JAR$_G$ (Java Archive) di tipo EAR (Enterprise Archive) o WAR (Web Archive), questi file vengono eseguiti rispettivamente dall'*application server* e dal *servlet*$_G$ *container*, due framework che si occupano quindi di avviare l'applicazione.

Presentazione

Per la creazione delle interfacce grafiche delle applicazioni durante lo stage, ho utilizzato gran parte delle tecnologie standard, ovvero HTML, CSS per l'aspetto e la struttura e JavaScript per il comportamento. Tali tecnologie erano già di mia conoscenza e non richiedevano troppo impegno da parte mia.

I contenuti generati con queste tecnologie, però, sono stati inglobati in un ambito di cui io non ero ancora a conoscenza: le pagine JSP.

JSP (Java Server Pages) è una tecnologia di programmazione web in Java EE per lo sviluppo della logica di presentazione delle applicazioni, eseguito tipicamente fornendo contenuti dinamici in formato HTML e inglobando le tecnologie citate prima. Al momento della compilazione del software, tali pagine vengono lette dal compilatore JSP e trasformate nelle apposite classi Java Servlet$_G$, ovvero una specifica estensione di Java pensata per l'utilizzo web.

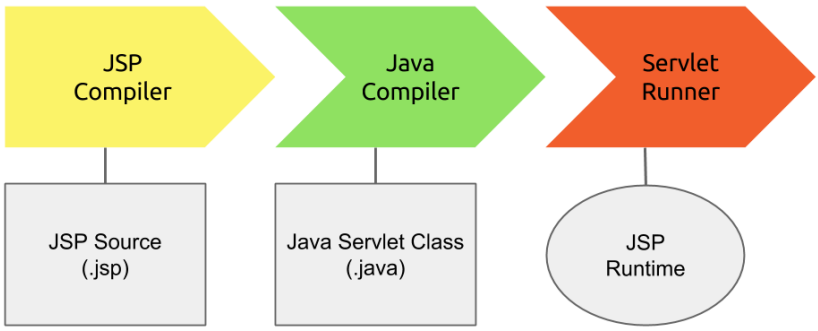

Figura 3.4: Flusso di compilazione di una JSP

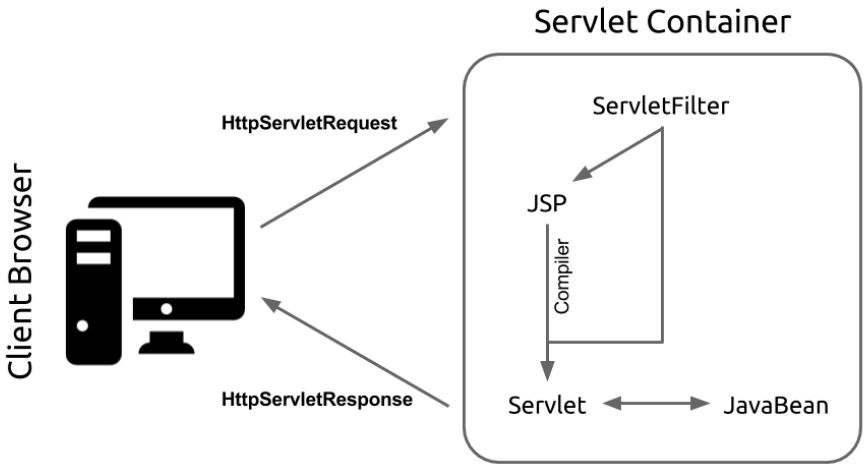

Figura 3.5: Rappresentazione della cominicazione client-server in ambito JSP

All'interno di una JSP viene definito l'HTML in cui viene immerso il codice Java sottoforma di *scriptlet*. Il tutto viene poi compilato e come risultato viene prodotta una Java Servlet$_G$. In particolare questo tipo di classi fornisce al programmatore la possibilità di manipolare la comunicazione client-server mediante appositi oggetti. A questo punto, il codice HMTL viene stampato in pagina mediante l'oggetto *HttpServletResponse*.

Queste pagine si basano anche su un insieme di librerie di tag JSTL (Java Standard Tag Library), con cui possono essere invocate funzioni predefinite sotto forma di classi Java. In aggiunta, permette di creare librerie di nuovi tag che estendono l'insieme dei tag standard (JSP Custom Tag Library).

Framework

Durante il lavoro di stage ho sempre utilizzato il framework Struts, in particolare per lo sviluppo delle nuove funzionalità per l'applicazione ELISE dell'ultima fase.

Si tratta di uno strumento utile alla creazione di applicazioni sviluppate secondo Java EE. Nella fase di studio delle tecnologie ho avuto modo di comprendere la sua forza, anche grazie alle mie conoscenze, acquisite durante il corso di studi. Struts infatti estende le Java Servlet$_G$, implementando il design pattern MVC$_G$ (Model-View-Controller), definendo una solida struttura per il software che lo adotta.

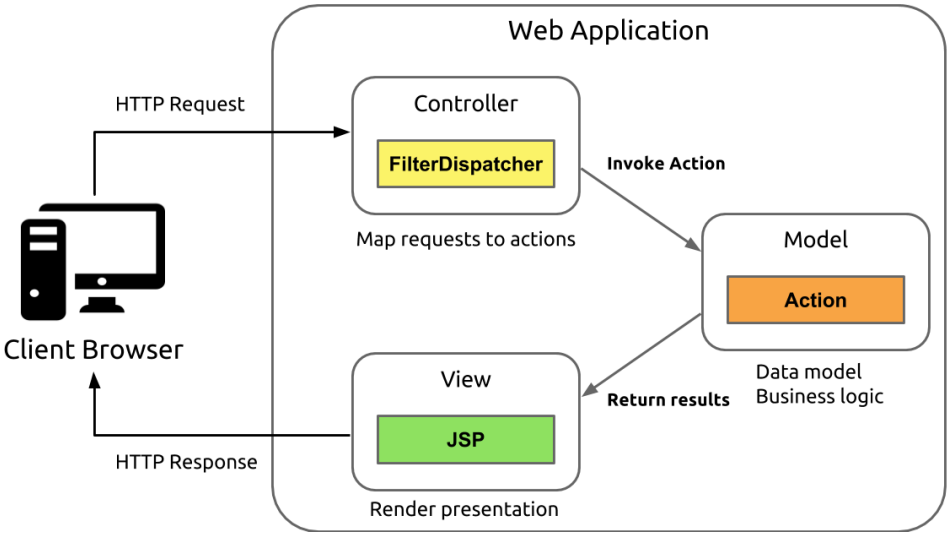

Figura 3.6: Struttura di un'applicazione sviluppata mediante il design pattern imposto da Struts

L'utilizzo di questo framework permette lo sviluppo di applicazioni web di notevoli dimensioni, inoltre agevola la suddivisione dello sviluppo del progetto fra i vari dipendenti. I programmatori web e i vari gruppi di sviluppatori possono quindi gestire in parallelo e autonomamente la loro parte del progetto.

Questo risulta maggiormente utile se si associano le proprie attività di sviluppo ad un sistema di versionamento, dove poter integrare le porzioni di software, ben strutturato secondo il framework.

Properties

Data la grandezza del progetto ELISE ed il suo elevato numero di pagine, è stato adottato un meccanismo per la gestione delle *label*, ovvero le scritte statiche da presentare in pagina.

In particolare grazie alla tecnologia JNDI$_G$ è possibile creare un vocabolario (memorizzato su file .properties) di stringhe Java, codificate mediante un nome identificativo, da richiamare nelle pagine JSP in modo da facilitare il mantenimento e la modifica di tale aspetto.

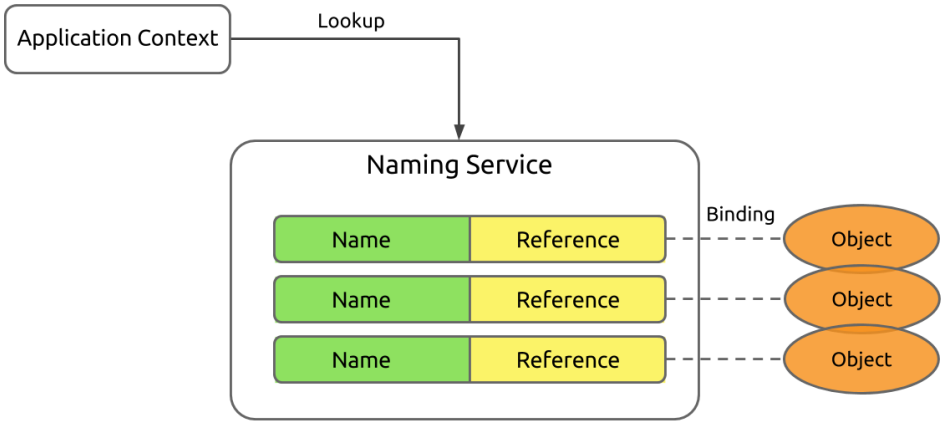

Figura 3.7: Sistema di vocabolario generato dall'uso di JNDI

Ovviamente in questo modo più pagine possono riferirsi alla stessa stringa utilizzando lo stesso identificativo, nel momento in cui un label deve essere modificato basterà cambiarlo nel vocabolario e tutte le pagine che si riferiscono ad esso verranno aggiornate.

Logging

Una tecnica molto utile che ho imparato durante dello stage è l'uso dei log, che nel percorso di studi avevo solo visto a grandi linee nel corso di Programmazione Concorrente e Distribuita.

Tale tecnica prevede di utilizzare di un servizio, spesso una libreria esterna, per tracciare il flusso applicativo e stampare su file le informazioni rilevanti dei vari stati che il software ha incontrato durante la sua esecuzione. Nel mio caso è stato adottato Log4J, una libreria molto nota a questo scopo, utilizzata anche nel progetto ELISE.

Essa permette di emettere delle notifiche nei vari punti del software, associandone il livello di importanza e livello generale dell'applicativo impostare il livello di debug da utilizzare durante l'esecuzione. Una volta lanciato, il programma produrrà il log contenente solo le notifiche che rispettano il livello selezionato, nascondendo quelle di minore importanza.

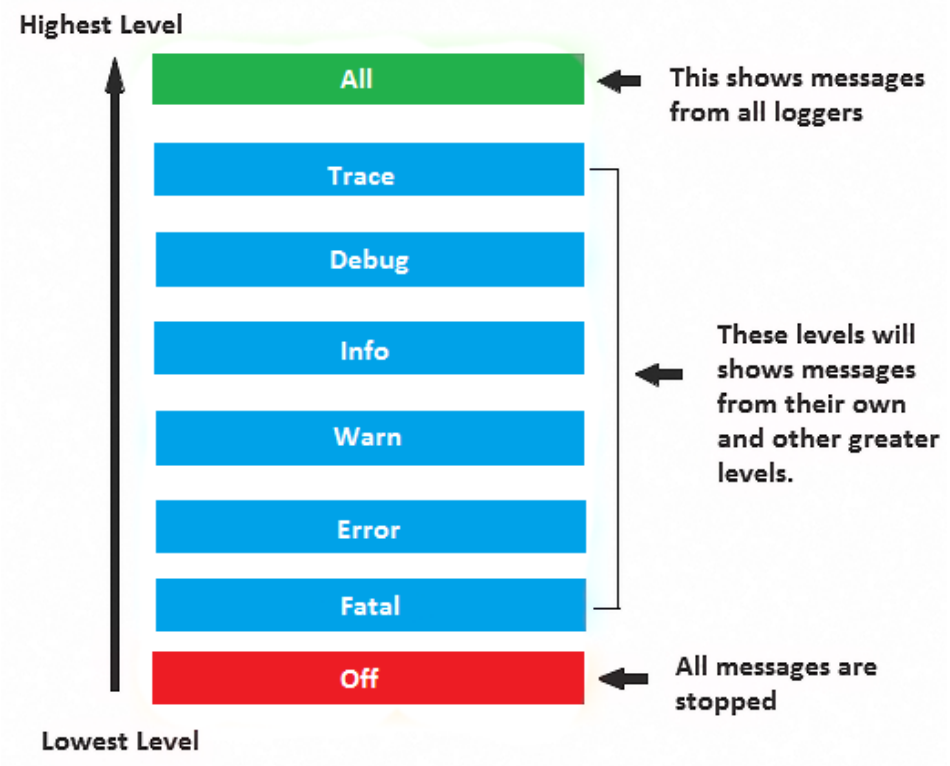

Figura 3.8: I diversi livelli di log selezionabili con la libreria Log4J - Fonte: tutorial online sull'uso della libreria

In molte occasioni è stato necessario consultare i file di log per risalire ad eventuali errori oppure per comprendere meglio il comportamento dell'applicazione. Talvolta è stato scelto di modificare il livello di debug per analizzare più a fondo i messaggi prodotti dal software.

Stampe PDF

Un'altra tecnica adotata dal progetto ELISE è la generazione dinamica di documenti PDF, da produrre nella filiale di competenza che li deve stampare per conto degli utenti allo sportello.

Per ottenere tutto questo sono previste delle classi Java specifiche dell'applicativo che si servono della libreria iText per l'inizializzazione di un documento ed il relativo riempimento con i giusti contenuti, sotto forma di tabelle, paragrafi, frasi, ecc. Anche questa libreria è tra le più usate per questo scopo, il che rende facile reperire guide e documentazione ufficiale su internet.

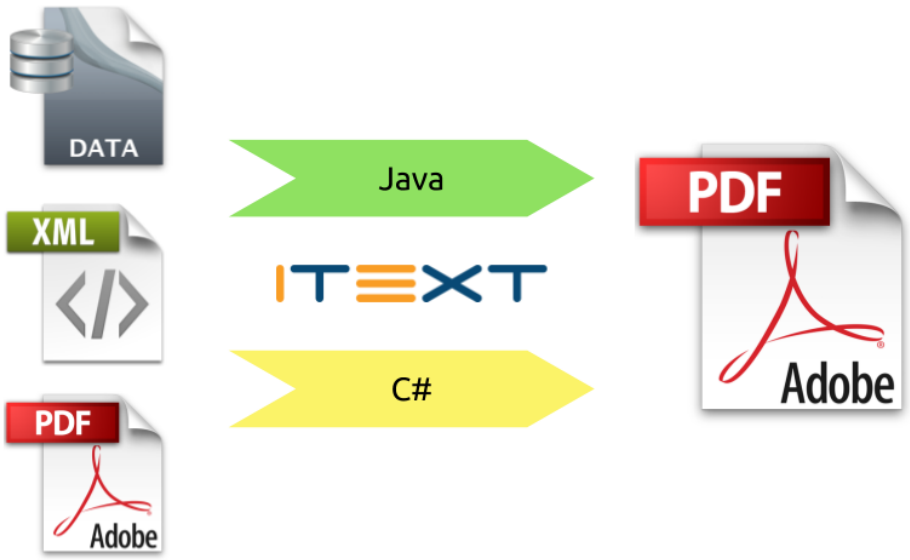

Figura 3.9: Illustrazione del processo di creazione dei documenti PDF mediante la libreria iText

3.2.2 Alternative analizzate

Hibernate

Hibernate è un framework *open source* per lo sviluppo di applicazioni Java. Fornisce un servizio di *object-relational mapping* (ORM) ovvero gestisce la

persistenza dei dati sul database attraverso la rappresentazione e il mantenimento su database relazionale di un sistema di oggetti Java. Come tale dunque, nell'ambito dello sviluppo di applicazioni web, si frappone tra il livello logico di business e quello di persistenza dei dati sul database.

La funzione primaria di Hibernate quindi, è di mappare classi Java in tabelle del database e tipi di dati Java in tipi di dati SQL. Questo framework non è adottato dal team per questo progetto in quanto la persistenza dei dati mediante un sistema CICS$_G$ non è supportata. Infatti il processo di codifica del flusso dati per comunicare con l'ambiente *host* è già ampiamente gestito mediante apposite classi Java che, con l'uso della Reflection e di mappe XML, si occupano di generare gli appositi JavaBean$_G$.

Spring

Spring è un altro framework *open source* per lo sviluppo di applicazioni su piattaforma Java. L'aspetto centrale nell'utilizzo di Spring è avere a disposizione il suo *inversion of control container*, ovvero un ambiete che invita lo sviluppatore ad applicare il design pattern architetturale *dependency injection*, permettendo di gestire in maniera consistente oggetti Java usando la Reflection.

Il contenitore si occupa del ciclo di vita degli oggetti gestiti, detti beans, mediante una configurazione fornita su file XML. Oltre a questa funzionalità, il framework offre supporto ad attività di *transaction management*, accesso ai dati, messaggistica e verifica. Spring non è adottato nello sviluppo di ELISE in quanto il progetto è già ampiamente strutturato secondo il framework Struts.

Maven

Apache Maven è un framework per la gestione di un progetto software. Esso può gestire la compilazione del progetto, l'invio di segnalazioni e la documentazione basandosi sul concetto centrale di *project object model* (POM), un file XML che descrive le dipendenze fra il progetto e le varie versioni di librerie necessarie.

Maven effettua automaticamente il download di librerie Java e plug-in Maven dai vari repository definiti, scaricandoli in locale o in un repository centralizzato lato sviluppo. Questo permette di recuperare in modo uniforme i vari file JAR$_G$ e di poter spostare il progetto indipendentemente da un ambiente all'altro avendo la sicurezza di utilizzare sempre le stesse versioni delle librerie.

Questo framework è stato da me studiato solo a scopo formativo, in quanto l'ambito di progetto non richiedeva di usufruire delle funzionalità che esso offre. In particolare la portabilità del software trattato è garantita dall'utilizzo degli ambienti di collaudo del cliente, senza bisogno di renderlo compatibile con altre macchine.

SVN

SVN (Subversion, Apache) è un sistema di versionamento, già descritto in sezione 1.3.4, alternativo ad RTC. È stato analizzato da parte mia a scopo formativo ed è utilizzato da parte dell'azienda per lo sviluppo altri progetti.

3.3 Processo di sviluppo

Al termine delle prime due fasi, come pianificato, il lavoro di stage aveva occupato un mese, metà del tempo a disposizione. Come ultima attività ho svolto un'analisi conoscitiva dell'applicazione ELISE, a cui avrei poi partecipato nello sviluppo. Mi preparavo quindi ad entrare nell'ultima fase, quella di implementazione delle espansioni su di un applicativo reale, per occupare il restante 50% del lavoro.

3.3.1 Analisi dei requisiti

Come prima attività, per reperire il lavoro da svolgere, l'azienda esegue delle attività di consulenza e stabilisce i contratti con i clienti. Successivamente è tutto pronto, a livello burocratico, per concentrarsi sulle esigenze del cliente e raccogliere i suoi requisiti.

Seguendo la struttura di governo aziendale, i team di analisti si confrontano con i responsabili tecnici e direttivi dell'ente che richiede il lavoro. In questa fase l'azienda si concentra molto per stabilire al meglio i requisiti del cliente, analizzando con cura i suoi bisogni e cercando di capire quali soluzioni possono soddisfarli. In questo modo garantisce che le successive attività di progettazione e codifica avvengano su una base solida, senza il rischio di dover ripetere attività successive a causa di errori a monte del progetto.

Questo processo avviene solitamente mediante trasferte da parte dei consulenti o, in caso di piccole attività, anche per via telefonica. Viene redatto, come output, un documento di programmazione delle macro attività da svolgere, che resta in mano ai project manager. Oltre a questo documento, gli analisti si occupano di redigere anche un'Analisi Funzionale per ognuna delle attività.

Tale documento affronta i requisiti ad alto livello, enunciando le principali funzionalità ed i cambiamenti rispetto alla versione attualmente in produzione dell'applicativo.

Le sezioni principali di questo documento sono:

- **Matrice dei requisiti**: enunciato discorsivo per introdurre il problema proposto;
- **Descrizione funzionale**: per spiegare il comportamento dell'applicativo lato web, presentando anche un'anteprima delle pagine che verranno aggiunte;
- **Casi oggetti di collaudo**: qui vengono indicate le componenti che saranno analizzate in fase di testing per verificarne il corretto comportamento;
- **Dettaglio tecnico**: qui vengono segnalate solo alcune particolarità e vengono enunciate le componenti software che saranno modificate o aggiunte.

Questo documento di analisi risulta di primaria importanza per la programmazione delle interfacce. Sotto il punto di vista di un programmatore web infatti, è necessario comprendere come le pagine andranno strutturate e quale comportamento dovrà adottare l'applicazione in risposta alle interazioni con l'utente o l'ambiente *host*.

Nel mio lavoro di stage ho dovuto svolgere diverse attività e per ciascuna di esse avevo a disposizione tale documento, da poter consultare in ogni momento oltre ad un approfondito studio iniziale, prima di sviluppare alcuna componente.

Per le attività a cui sono stato assegnato, mi sono occupato di analizzare il problema in questione e di catalogare i requisiti che mi venivano forniti dagli analisti, oltre a raccoglierne di nuovi. Ho attribuito ai requisiti la seguente forma:

```
Importanza Tipologia - Identificativo       Titolo
```

Dove:

- **Importanza**: denota il peso del requisito di riferimento e ne stabilisce una priorità. Assume i valori:
  - **O**: Obbligatorio, il requisito deve essere soddisfatto per considerare il progetto completo;

  - **D**: Desiderabile, se il requisito viene implementato porta valore aggiunto al progetto ma non è strettamente necessario.

- **Tipologia**: indica la natura del requisito e assume i seguenti valori:
  - **F**: Funzionale, rappresenta una funzionalità che il prodotto finale dovrà fornire, che sia essa espressa in forma generale a livello di sistema o espressa nel dettaglio delle componenti;
  - **V**: Di vincolo, specifica un vincolo che il software deve rispettare;
  - **P**: Prestazionale, si tratta di una caratteristica di performance che il prodotto deve soddisfare;
  - **T**: Tecnico, ovvero un requisito implementativo che l'applicazione dovrà possedere per garantire determinate funzioni.

- **Identificativo**: un numero, generato per incremento, che identifica il requisito in modo univoco se considerato assieme alla codifica della sua importanza e tipologia;

- **Titolo**: una breve descrizione del requisito e dei suoi riferimenti.

In totale, per i miei ambiti ho raccolto 66 requisiti. Ho compreso le funzionalità richieste dal cliente e quelle che l'applicativo doveva implementare, analizzando le prime nel dettaglio per ricavare le seconde. Per fare ciò mi sono servito sia dei documenti di Analisi Funzionale, sia dell'esperienza del tutor e degli altri colleghi del team. Al termine delle fasi di analisi avevo i requisiti necessari per pilotare le successive attività di progettazione e sviluppo.

Nel seguente grafico viene mostrata la suddivisione dei requisiti secondo la loro importanza.

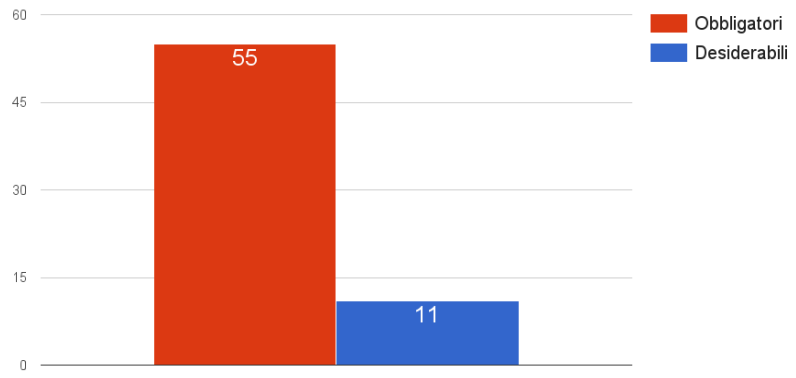

Figura 3.10: Suddivisione dei requisiti in base alla tipologia

Nel seguente grafico mostro invece la suddivisione dei requisiti secondo la loro tipologia.

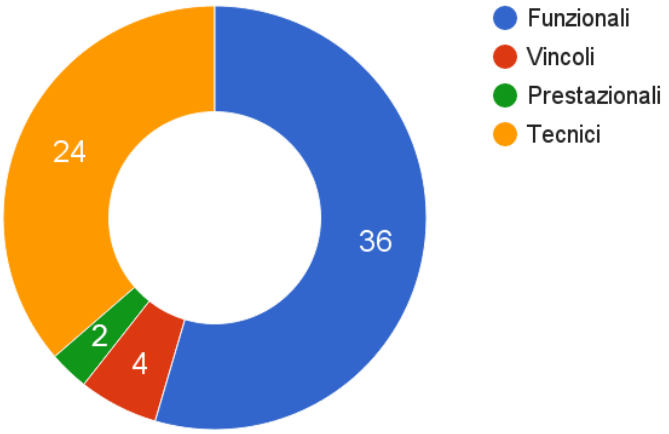

Figura 3.11: Suddivisione dei requisiti in base alla tipologia

3.3.2 Progettazione

Le scelte architetturali sono già state prese in passato alla nascita dell'applicativo e sono radicate in tutto il software. A differenza di queste, le scelte progettuali per le singole funzionalità, che si vanno ad aggiungere col passare del tempo durante la manutenzione e l'evoluzione del software, vengono stabilite ad ogni incremento del ciclo di sviluppo.

Terminata la fase di analisi dei requisiti, i team di competenza dell'ambiente di gestione dei dati dell'applicazione si occupano di progettare i programmi lato *host*. Questa attività influenza in modo decisivo anche il modo in cui verranno poi sviluppate le interfacce web per tali programmi.

Come output di questa fase viene prodotto quindi il documento di Specifica Tecnica. Questo secondo documento affronta nel dettaglio gli aspetti tecnici trattando principalmente i programmi COBOL_G, dai quali poi anche i programmatori web possono individuare i parametri da utilizzare nelle richieste via rete per recuperare i dati. In questo modo lo sviluppo delle pagine web del software viene allineato con le implementazioni fatte su *host*, permettendo una progettazione corretta anche dal punto di vista dei programmatori *front-end*.

Progettazione architetturale

Lo studio della struttura di ELISE mi ha permesso di conoscere i suoi funzionamenti, comprendendo quali scelte fossero adeguate o meno in tale sistema. L'applicazione, come menzionato, si basa sul framework Struts per la definizione delle proprie fondamenta. Il vantaggio più significativo portato da tale scelta è rappresentato dalla struttura che ne deriva.

In particolare, Struts gestisce il flusso delle richieste in entrata all'applicazione e le identifica in *action* da svolgere. Queste azioni vengono definite in un apposito file XML del framework, in cui si associa una classe Java ad un URL, in modo da collegare le interazioni con l'utente nelle pagine web con la logica di business dell'applicazione. Ogni azione ha poi una pagina JSP che rappresenta il risultato delle operazioni a basso livello, Struts si occupa di eseguirne il *render* una volta ottenuti i dati di cui essa ha bisogno. Spesso tali dati sono incapsulati in JavaBeans$_G$.

Tutto ciò non sarebbe possibile se la struttura imposta dal framework non seguisse una logica ben precisa. Il software infatti implementa in questo modo un *design pattern* architetturale molto usato a livello strutturale: MVC$_G$. Per l'applicazione ELISE questo pattern è implementato secondo il modello web, ovvero a partire da una pagina JSP si richiama la logica di business (sottoforma di Enterprise JavaBeans) attraverso il server (rappresentato da classi Servlet).

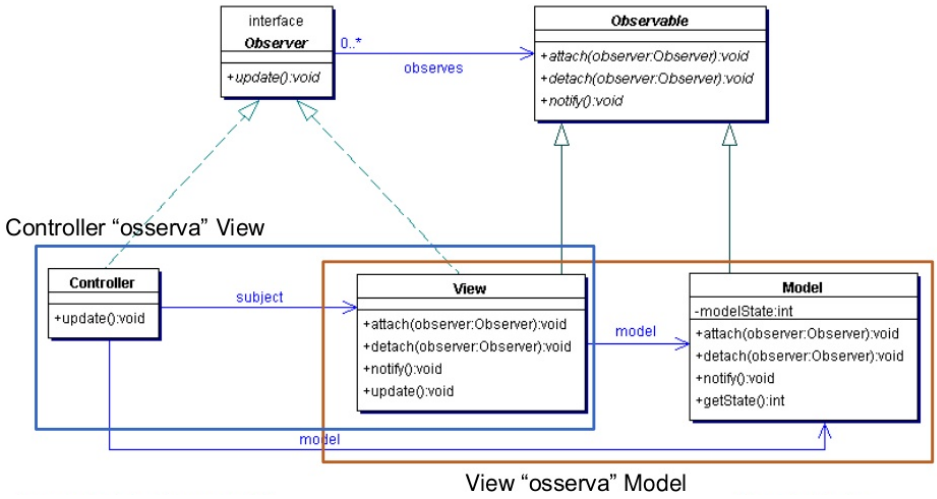

Figura 3.12: Diagramma delle classi per l'implementazione del *design pattern* architetturale MVC - Fonte: slides del corso di Ingegneria del Software mod. B

In questo modo possono essere realizzate più viste grafiche che rimandano alla stessa componente Java, definendo le apposite parti che si occupano del controllo delle richieste e stabilendo i collegamenti nel file *struts-config.xml*.

```
1  <?xml version="1.0" encoding="ISO-8859-1" ?>
2  <!DOCTYPE struts-config PUBLIC
3  "-//Apache Software Foundation//DTD Struts Configuration 1.0//EN"
4  "http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd">
5
6  <struts-config>
7
8     <form-beans>
9        <form-bean name="login" type="test.struts.LoginForm" />
10    </form-beans>
11
12    <action-mappings>
13       <action
14          path="/login"
15          type="test.struts.LoginAction" >
16
17          <forward name="valid" path="/jsp/MainMenu.jsp" />
18          <forward name="invalid" path="/jsp/LoginView.jsp" />
19       </action>
20    </action-mappings>
21
22 </struts-config>
```

Figura 3.13: Esempio di configurazione delle componenti di Struts mediante file XML

Progettazione di dettaglio

Per le singole attività di manutenzione ed espansione di ELISE, la progettazione dell'applicativo avviene in base ai requisiti e agli scopi di tali modifiche. Le fasi di progettazione di dettaglio si susseguono infatti ad ogni incremento applicato sul software, allo scopo di definire in modo corretto le componenti utili al soddisfacimento dei requisiti.

Io sono stato l'unico responsabile nel definire la struttura delle parti che componevano le mie attività. Il tutor mi ha insegnato precedentemente come agire e successivamente mi ha lasciato produrre in autonomia, revisionando il mio lavoro solo per eventuali accertamenti, ma è comunque rimasto sempre a mia disposizione.

Per le mie attività ho quindi cercato di analizzare il problema e ideare delle soluzioni adatte all'applicazione di progetto. In particolare, come descritto, avevo già a disposizione una solida struttura e molte librerie utili. Si trattava di muovere i primi passi in quel sistema e progettare delle soluzioni che

riutilizzassero quante più funzionalità già presenti. In tale ottica anche le mie soluzioni prodotte, se risultavano completamente nuove, andavano pensate per essere riutilizzate in futuro.

Ho cercato di soddisfare i requisiti del cliente al meglio, senza fretta nel produrre subito una codifica delle soluzioni, ma pensando bene alla loro organizzazione. In questo senso mi sono impegnato a organizzare il codice in classi che rispettassero una solida gerarchia e fossero estendibili, per garantirne un'utilità futura. Ad esempio, per la gestione delle stampe dei preventivi e la documentazione della composizione dei titolari di un finanziamento, ho realizzato tale struttura per definire le varie tipologie di stampa. In tal modo ho raggruppato le parti in comune nelle classi poste in alto nella gerarchia.

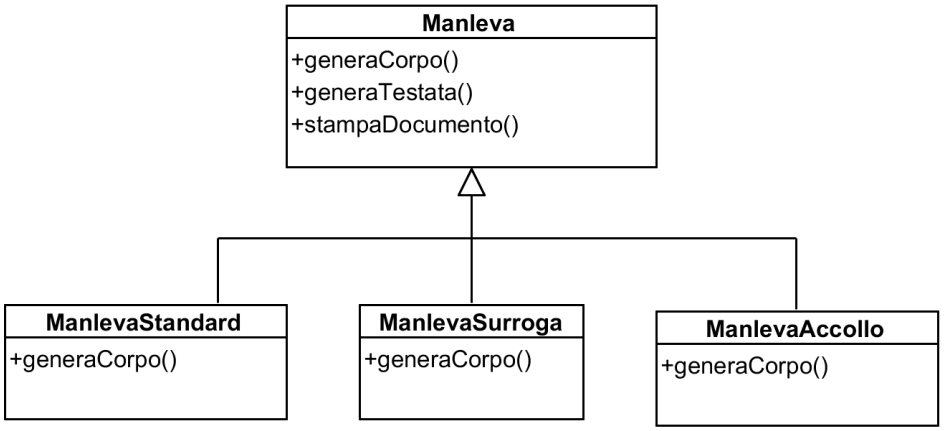

Figura 3.14: Diagramma delle classi semplificato della gerarchia per le stampe

Un'altra tecnica che ho avuto modo di studiare e applicare in maniera approfondita è la Reflection$_G$. L'ho usata principalmente per rendere noto al sistema software quali parti delle mie componenti dover recuperare per svolgere determinate funzioni. Nel dettaglio ho modificato e creato dei nuovi file di configurazione XML, in cui ho definito i parametri e i nomi dei campi dei JavaBeans$_G$ da utilizzare nella comunicazione con l'ambiente *host* e nelle pagine dell'applicazione.

Per quanto riguarda la tecnologia JSP, che racchiude diversi linguaggi in sé, è stato complicato definire uno stile generale per la creazione delle interfacce.

A tale scopo ho cercato di separare i concetti e le utilità comprese in ogni pagina, progettando piuttosto delle classi Java esterne o diverse componenti JSP da includere poi in un file principale. Ho creato quindi un'alta coesione delle componenti, identificando il più possibile il loro scopo e suddividendole in modo da non attribuire troppe responsabilità ad una singola parte. In questo modo andavo ad aumentare il livello di accoppiamento tra di esse, ma non risultava mai eccessivo da rappresentare uno svantaggio, in quanto il prodotto di grandi dimensioni diveniva così più comprensibile e strutturato, dando modo di riutilizzare le sue componenti.

Come già reso noto, durante il corso dello stage ho avuto modo di applicare alcuni *design pattern* studiati durante i miei studi all'Università. Un secondo pattern che desidero citare è il *decorator pattern*, molto usato nella progettazione delle componenti dell'applicativo. Grazie a questo principio ho implementato delle soluzioni che aggiungevano delle funzionalità o dei contenuti a delle porzioni generiche del software.

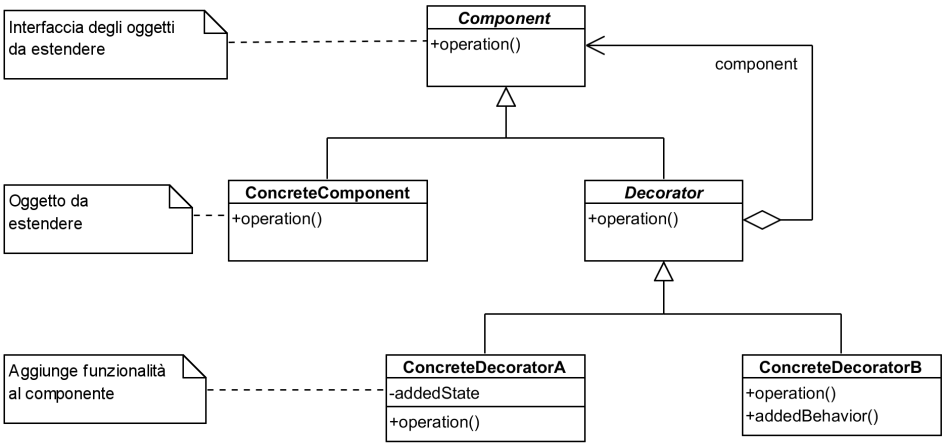

Figura 3.15: Diagramma delle classi del *decorator pattern*

3.3.3 Codifica

Una volta pianificate le azioni e progettate le componenti, per ogni attività, mi sono occupato di realizzare le soluzioni. Dal punto di vista implementativo, avere alla base del proprio lavoro un framework che si occupa di strutturare le parti del sistema e categorizzarle secondo i loro scopi, è certamente un vantaggio.

Le fasi di sviluppo sono state quelle che più mi hanno conferito padronanza degli strumenti e delle metodologie che dovevo imparare.

Con l'aiuto del tutor facevo chiarezza sulle implementazioni da realizzare e sulle tecniche da impiegare, guadagnando poco alla volta autonomia e responsabilità delle mie azioni.

Ho creato diverse pagine JSP nel corso dello stage e ho notato che tale tecnologia può causare numerose ambiguità per uno sviluppatore esterno. Organizzando il codice in modo comprensibile ho definito una struttura e raggruppato le porzioni di codice di ogni tipologia di linguaggio. Inizialmente definivo in una *scriptlet* le porzioni di codice Java utile a recuperare le componenti di business, solitamente i JavaBeans$_G$.

```
12
13 <%
14     UserForm form = request.getParameter("userForm");
15     String username = form.getUsername();
16     if ( username != null && username.length() > 0 ) {
17         // TO DO
18     }
19 %>
20
```

Figura 3.16: Esempio di codice Java incluso in una pagina JSP mediante *scriptlet*

In seguito solitamente ho dato inizio alla struttura della pagina mediante i *tag* personalizzati dell'applicazione, utilizzando quindi il suo *template* per mantenere coerenza grafica e strutturale.

Mi sono servito quindi dei tag standard di JSTL e ho sviluppato soluzioni con alcuni che invece sono realizzati da altri sviluppatori, come il *display tag*, per la generazione di tabelle. In particolare questo tag implementa il *decorator pattern*, ovvero sfrutta una classe Java appositamente realizzata per riempire la tabella con i relativi contenuti.

```
15
16 <display:table name="test" decorator="org.displaytag.sample.Wrapper" >
17
18   <display:column property="id" title="ID" />
19   <display:column property="email" />
20   <display:column property="status" />
21   <display:column property="date" />
22   <display:column property="money" />
23
24 </display:table>
25
```

Figura 3.17: Esempio di codice JSP per la creazione di una tabella implementando il *decorator pattern*

In seguito ho applicato le mie conoscenze di programmazione web, con i linguaggi HTML e CSS per completare la presentazione delle pagine.

In tutte le interfacce che ho sviluppato, vi era il bisogno di definire anche un aspetto comportamentale dell'applicazione a livello client. In coda ai sorgenti delle pagine, infatti, ho inserito porzioni di codice JavaScript, utile a realizzare effetti grafici e richiamare determinate funzioni o procedure già presenti nel software.

In tale contesto ho avuto modo di applicare la tecnica AJAX$_G$ per recuperare determinate informazioni che risiedono nel server, senza imporre un ricaricamento della pagina.

```
33
34 function loadDoc() {
35     var xhttp = new XMLHttpRequest();
36     xhttp.onreadystatechange = function() {
37         if (this.readyState == 4 && this.status == 200) {
38             document.getElementById("demo").innerHTML = this.responseText;
39         }
40     };
41     xhttp.open("GET", "ajax_info.txt", true);
42     xhttp.send();
43 }
44
```

Figura 3.18: Esempio di codice JavaScript per implementare la tecnica AJAX

3.4 Verifica e validazione

Durante tutta la fase di implementazione delle funzionalità ho svolto anche attività di verifica e validazione. Durante la codifica ho compiuto tali attività mediante delle prove in locale, prima dei rilasci invece, attraverso un piccolo collaudo. Questo per verificare che tutti i requisiti fossero soddisfatti e che l'andamento dello sviluppo procedesse nella giusta direzione, al fine di portare in ambiente di integrazione delle soluzioni più corrette possibile ed evitare eventuali iterazioni nel ciclo di sviluppo.

Io e il tutor abbiamo dato priorità a verifiche effettuate su casi di utilizzo reali dell'applicazione analizzando il funzionamento nei vari casi d'uso previsti per ogni attività, testandola quindi manualmente piuttosto che in automatico.

Per il progetto ELISE infatti non era prevista la stesura a priori di test automatici da effettuare una volta prodotto il codice necessario, in quanto ogni attività di evoluzione, programmata per il software, si presenta ad un alto livello di granularità, garantendo una ristretta copertura di casi d'uso, facilitando quindi le attività di test.

3.4.1 Analisi statica

Lo strumento più usato per la verifica risulta essere l'ambiente di sviluppo Eclipse. Esso mette a disposizione dello sviluppatore un sistema di analisi statica che permette la visualizzazione di *error* e *warning* causati da problemi nel codice o utilizzi poco adatti del linguaggio.

Per ogni codice sorgente che andavo ad estendere o creare, avevo la responsabilità di rimuovere tutti gli errori e limitare i *warning* alle sole segnalazioni dovute all'utilizzo di versioni di JVM differenti. In tal modo si garantiva che il software non producesse malfunzionamenti dovuti ad uno scorretto uso della piattaforma Java EE.

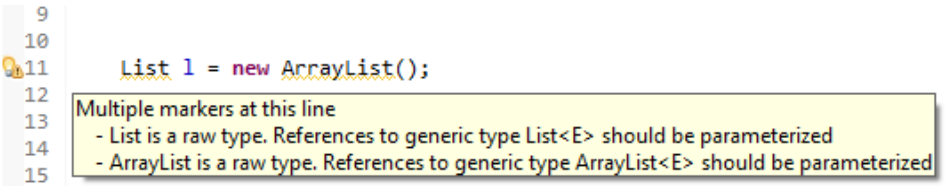

Figura 3.19: Interfaccia di Eclipse per la segnalazione di *warning*

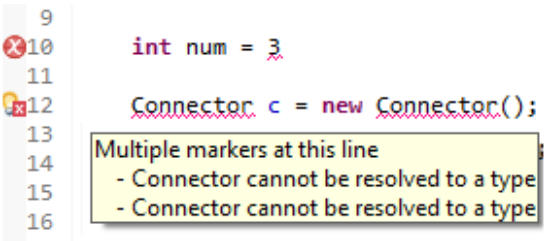

Figura 3.20: Interfaccia di Eclipse per la segnalazione degli *error*

Nelle fasi di sviluppo, utilizzando Eclipse, ho inoltre avuto modo di usufruire delle sue funzionalità di *debug* per il controllo di errori a tempo di esecuzione dell'applicazione. Questo mi ha permesso di scoprire errori comportamentali nel codice ma anche di comprendere i flussi intrapresi dal software durante il suo utilizzo.

Oltre a questo strumento ho utilizzato molto i file di log generati dalla stessa applicazione, per gli stessi scopi.

3.4.2 Analisi dinamica

Nell'ultima fase di stage poi, ho svolto col supporto del tutor le attività di validazione, composte dai test di sistema e dal collaudo.

Test di sistema

Per verificare le funzionalità che doveva includere il nuovo prodotto ho effettuato i test di sistema, ovvero una serie di prove al fine di verificare la copertura dei requisiti definiti con le attività di analisi. Questi sono stati svolti in ambiente di collaudo, dove l'integrazione delle componenti dell'applicazione risulta già testata e bisogna testare l'aspetto funzionale, verificando che il prodotto presenti le funzionalità stabilite e che queste non producano errori in fase di esecuzione.

Dopo aver compilato il codice e installato il prodotto, quindi, abbiamo testato le funzionalità da me trattate, una a una. Per prima cosa abbiamo stabilito degli scenari di prova per mappare i requisiti e abbiamo stabilito pre-condizioni e post-condizioni. Per ogni attività quindi sono state verificate le post-condizioni, accertando la copertura e il soddisfacimento dei requisiti assegnati ai vari scenari di prova.

Collaudo

Una volta effettuati i test, alcuni consulenti sono entrati in contatto con il cliente e si sono occupati della validazione dei requisiti funzionali.

Per tale attività vengono replicati i test di sistema, nello stesso ambiente di collaudo, per dimostrare ai clienti l'effettivo funzionamento delle *feature* richieste, mostrando loro che i requisiti sono stati soddisfatti.

3.5 Rilascio delle funzionalità

Nell'ultima settimana di stage, il tutor ha ritenuto corretto che trattassimo anche il rilascio delle nuove *feature* da me sviluppate. Col suo supporto quindi ho utilizzato il sistema di versionamento RTC, mediante Eclipse, per concludere le attività ed associarle alle apposite release unit. Così facendo ho reso disponibili le nuove funzionalità a tutto il team di sviluppo in ambiente di integrazione e, dopo una compilazione, il nuovo software risultava aggiornato.

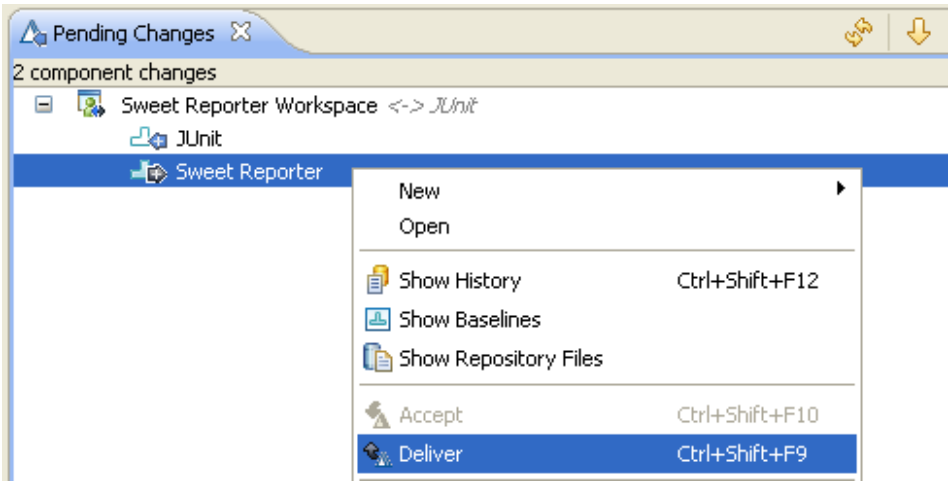

Figura 3.21: Interfaccia per eseguire il rilascio delle attività mediante Eclipse e il sistema di versionamento RTC - Fonte: il sito internet di IBM

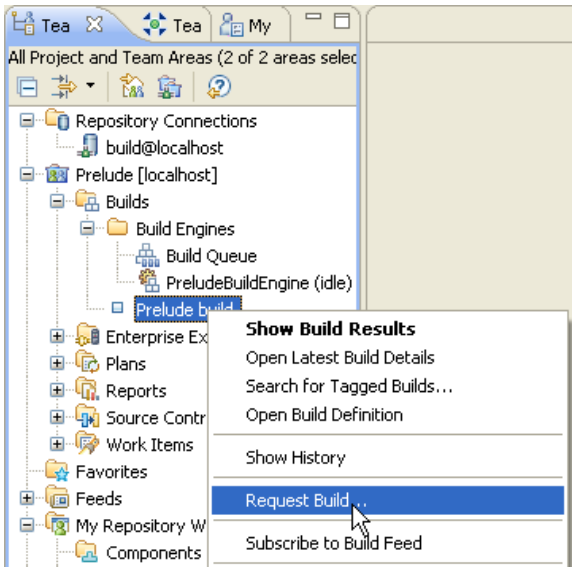

Figura 3.22: Interfaccia per eseguire la compilazione del software in un determinato ambiente mediante Eclipse e il sistema di versionamento RTC - Fonte: il sito di RTC jazz.net

Dopo le attività di test ho potuto effettuare i rilasci anche in ambiente di collaudo, dove ho svolto la validazione. Successivamente il cliente ha controllato il lavoro svolto, installando poi la nuova versione di ELISE nei server di produzione, per l'utilizzo vero e proprio nelle diverse filiali del gruppo del Banco Popolare.

Capitolo 4

Valutazione retrospettiva

4.1 Conoscenze preliminari

Per lo svolgimento del lavoro di stage non era richiesta una conoscenza approfondita dei linguaggi che si andavano ad utilizzare, proprio perché era prevista una prima fase di studio delle tecnologie e lo scopo dello stage era formativo, in modo da arricchire le mie competenze curricolari.

Nonostante ciò, la mia preparazione riguardo le tecnologie web, maturata nel corso di studi, è stata molto utile nel comprendere i nuovi aspetti che andavo a trattare in azienda, consolidando tali conoscenze. L'aspetto che ha richiesto maggiore impegno da parte mia è stata l'analisi del progetto ELISE e le prime fasi di approccio ad esso, nel tentativo di individuare percorsi a me conosciuti e comprendere al meglio i suoi funzionamenti, essendo un applicativo molto grande ed intricato nei suoi processi.

Il Corso di Laurea triennale in Informatica mi ha fornito una buona formazione nello sviluppo software, permettendomi la comprensione del ciclo di sviluppo applicato al progetto richiesto nel mio periodo di stage, senza troppe difficoltà.

Questi aspetti preliminari hanno permesso che il progetto di stage non subisse imprevisti o interruzioni dovute a mie lacune, concentrandomi per più tempo sugli studi e le applicazioni previste dal piano di lavoro.

4.2 Obiettivi raggiunti

Durante il lavoro di stage ho svolto le attività necessarie al raggiungimento degli obiettivi prefissati in fase di pianificazione. Le seguenti tabelle raccolgono i risultati ottenuti per ciascun obbiettivo, suddividendoli per la loro categoria.

Obbligatori

Obiettivo	Esito
Studio e acquisizione di padronanza dell'ambiente di sviluppo Eclipse	Completato
Studio e comprensione della piattaforma Java EE	Completato
Installazione e utilizzo di diversi *application server* e diverse JVM	Completato
Acquisizione familiarità con la programmazione web in ambito Java (tecnologie JSP, JSTL, Servlet$_G$)	Completato
Acquisizione tecniche di programmazione con framework Struts	Completato
Studio e utilizzo del sistema di versionamento RTC	Completato
Studio e utilizzo della tecnologia AJAX$_G$	Completato
Implementazione di applicazioni di esempio per le funzionalità basilari	Completato
Analisi di una complessa applicazione reale in architettura Java EE	Completato
Integrazione nel team di sviluppo e acquisizione competenze nelle dinamiche di gruppo	Completato
Comprensione e acquisizione familiarità con la documentazione di analisi e specifica delle attività	Completato
Implementazione di modifiche basilari dell'applicazione ambito di progetto	Completato
Implementazione di modifiche complesse dell'applicazione ambito di progetto	Completato

Tabella 4.1: Tabella degli obiettivi obbligatori raggiunti durante il lavoro di stage

Desiderabili

Obiettivo	Esito
Raggiungimento di un buon livello di autonomia nell'utilizzo di Java EE	Completato
Raggiungimento di un buon livello di autonomia nell'utilizzo di tecnologie web standard (HTML, CSS, JavaScript)	Completato
Raggiungimento di un buon livello di autonomia nell'utilizzo di tecnologie web in ambito Java (JSP, JSTL, Servlet$_G$)	Completato
Acquisizione tecniche di programmazione con framework alternativi (Maven e Hibernate)	Completato
Studio e utilizzo di un sistema di versionamento alternativo (SVN)	Completato
Studio e utilizzo della libreria per le stampe PDF, iText	Completato
Studio e utilizzo della libreria per il *logging*, Log4J	Completato
Capacità di portare a termine le attività lavorative secondo le tempistiche stabilite, anche in situazioni critiche	Completato
Conoscenza delle norme di sicurezza relative all'ambiente di lavoro	Completato

Tabella 4.2: Tabella degli obiettivi desiderabili raggiunti durante il lavoro di stage

Facoltativi

Obiettivo	Esito
Studio delle meccaniche di comunicazione con l'area di business per la gestione dei dati	Completato
Partecipazione alle attività di test di integrazione dell'applicazione ambito di progetto	Non completato
Partecipazione alle attività di collaudo dell'applicazione ambito di progetto	Completato
Rilascio delle nuove funzionalità sviluppate nelle attività assegnate	Completato

Tabella 4.3: Tabella degli obiettivi facoltativi raggiunti durante il lavoro di stage

Si denota quindi un completamento globale degli obiettivi pianificati, ad unica eccezione delle attività di integrazione software, in quanto queste sono state assegnate ad altri team di sviluppo.

4.3 Bilancio formativo

A partire dalle mie conoscenze preliminari nell'ambito ricoperto dal progetto ho avuto modo, tramite il lavoro di stage, di ampliare le mie conoscenze e spingermi verso concetti a me nuovi riguardo la progettazione architetturale nel web. Grazie alle spiegazioni del tutor aziendale, a cui devo la facilità con cui si è svolto il progetto, ho potuto ricevere molte nozioni, anche non approfondite ma che di volta in volta mi abilitavano a nuove possibilità di implementazione, inglobandomi in un ambiente stimolante e dinamico.

Tutto ciò mi ha permesso di imparare molto, arricchendo le mie conoscenze nello sviluppo web, utilizzando software di cui non ero pratico e consolidando le mie conoscenze a livello teorico, applicandole nella realtà.

Ho guadagnato padronanza degli sviluppi web mediante Java EE e le tecnologie incluse in tale piattaforma, come JSP e JSTL che non avevo mai utilizzato. Ho interagito con sistemi diversi di versionamento, ho imparato nuove tecniche di programmazione, come AJAX$_G$, la Reflection$_G$ e le *properties*, di cui non ero a conoscenza o avevo ricevuto solo alcuni accenni all'Università. Ho anche ottenuto una buona padronanza con un nuovo ambiente di sviluppo e diverse librerie molto utili, ad esempio per il *logging*.

Grazie allo stage in Sopra Steria ho avuto modo di crescere professionalmente e vivere personalmente un'esperienza lavorativa che ha avuto luogo in un ambiente molto dinamico e stimolante. Ho interagito con molti colleghi anche in altre sedi e ho avuto una visione generale dei processi di un'azienda rilevante nel settore ICT$_G$.

Presumibilmente il Corso di Laurea triennale in Informatica è ancora in fase di rodaggio, dopo il passaggio a semestri, si è sentita la pesantezza del carico distribuito male nei vari insegnamenti, specie riguardo le propedeuticità che spesso in questo corso rappresentano uno scoglio per gli studenti.

Ho apprezzato che il corso di Ingegneria del Software abbia dato luogo a diversi seminari tecnoloici e abbia incentrato i progetti presentati nell'uso di moderni linguaggi e paradigmi di programmazione, al contrario degli altri corsi che rimangono piuttosto canonici e trattano poche innovazioni negli ambiti di studio.

Oltre al successo nel soddisfare gli obiettivi prefissati per la valutazione positiva dello stage, a livello personale ho avuto molta soddisfazione nel raggiungere anche gli obiettivi personali che mi ero posto nella ricerca dello stage.

Ho lavorato in un ambiente professionale e collaborativo, applicando le mie competenze per scopi pratici e utili al team di sviluppo. Ho avuto modo di arricchire il mio profilo tecnico e lavorativo diventando un potenziale interesse ancora maggiore verso le aziende ICT_G, come la stessa Sopra Steria che mi ha ospitato.

Sono rimasto soddisfatto quindi dell'esperienza fatta, ma anche della formazione guadagnata presso l'Università nel Corso di Laurea triennale in informatica, stabilendo una solida base di studi su cui costruire la mia carriera.

Mi è stata data la possibilità di proseguire il mio percorso di stage in vista di una futura assunzione, dopo il conseguimento di un totale di sei mesi di tirocinio, compresi i due appena svolti. Pertanto ho intenzione di procedere in tale direzione, vista anche la soddisfazione da ambo le parti dopo questa esperienza.

Ringraziamenti

Ringrazio Dio poi i miei genitori, Najeh e Latifa, per avermi accompagnato e concesso di arrivare fin qui. Grazie inoltre alla mia intera famiglia per il sostegno e per essermi sempre stati vicini.

Ringrazio i miei amici e compagni di studi per tutti i bellissimi anni passati insieme, in particolare Abdourahmane, Hamza e Sara per tutto il loro affetto e sostegno ricevuto.

Ringrazio Sopra Steria Group S.p.A. e tutti i dipendenti della sede di Padova per avermi accolto e seguito durante il tirocinio.

Ringrazio sentitamente, infine, il prof. Tullio Vardanega, relatore della mia Tesi, per l'aiuto ed i consigli che mi ha dato per lo svolgimento del lavoro.

Padova, Dicembre 2017 Abdelilah Lahmer