Università degli studi di Camerino

Scuola di Scienze e Tecnologie

Corso di Laurea in Informatica (L-31)

Progettazione e sviluppo di un applicazione desktop per l'analisi di bilancio di un'azienda

Laureando

Danilo Paoletti

Matricola 085353

Relatrice

Dr.ssa Barbara Re

Anno accademico 2017-2018

Indice

Introduzione

L'analisi di bilancio è una delle procedure attraverso cui vegono determinati i fattori che determinano i guadagni, le risorse attraverso cui l'azienda ripaga i debiti e il grado di indebitamento. In questa tesi verrà mostrato il software da me realizzato per il processo di analisi di bilancio attraverso la riclassificazione dello Stato Patrimoniale, del Conto Economico e del forecast per indici di previsioni.
Inizialmente verrà descritto il contesto economico dell'analisi di bilancio, successivamente verranno trattate la fase di analisi e progettazione che ha preceduto la realizzazione del software, descrivendo attraverso diagramma Unified Modeling Language (UML) [1] i vari requisiti, vincoli e funzionalità del prodotto.
Successivamente saranno descritte le tecnologie utilizzate e si entrerà nel merito della realizzazione di tutte le funzionalità richieste.
In conclusione si farà riferimento ad eventuali sviluppi futuri che potrebbero essere implementati nel software realizzato.

Capitolo 1

Bilancio di un'azienda

L'analisi di bilancio, è uno strumento tecnico, che permette di studiare i risultati d'esercizio derivanti dalle attività aziendali, focalizzando l'attenzione su:

- situazione economica: condizioni che determinano la produttività dell'azienda e costituisce la capacità di determinare utili d'esercizio;
- situazione finanziaria: condizioni che determinano la capacità di far fronte ai debiti a breve scadenza e all'accesso a prestiti di finanziamento;
- situazione patrimoniale: condizioni che determinano il rapporto tra le immobilizzazioni e i debiti verso terzi.

In altre parole, con lo strumento dell'analisi di bilancio, si cerca di individuare nelle diverse aree (economica, finanziaria e patrimoniale) i primi spunti di riflessione per l'analisi della gestione.

Il bilancio di un'azienda, nella sua forma più semplice, è formato da due documenti: lo **Stato Patrimoniale**, che rappresenta in modo sintetico la composizione qualitativa e quantitativa del patrimonio della società al giorno della chiusura dell'esercizio, e il **Conto Economico**, che espone il risultato economico dell'esercizio attraverso la rappresentazione dei costi e degli oneri sostenuti, nonché dei ricavi e degli altri proventi conseguiti nell'esercizio;.

1.1 Stato Patrimoniale

Lo Stato Patrimoniale, definito dall'art. 2424 del Codice Civile, rappresenta la situazione patrimoniale ad una certa data di un'impresa ed è suddiviso in due sezioni contrapposte chiamate Attività e Passività.

In particolare, le attività sono distinte secondo criteri di realizzabilità ed esigibilità, in :

- crediti verso soci per versamenti ancora dovuti, con separata indicazione della parte già richiamata;
- immobilizzazioni: rappresentano gli investimenti destinati per un periodo superiore a 12 mesi, a trasformarsi in denaro. Sono iscritte al netto dei fondi di ammortamento;
- attivo circolante: rappresentano gli investimenti destinati per un periodo inferiore a 12 mesi, a trasformarsi in denaro. Iscritto al netto di eventuali fondi di svalutazione;
- ratei e riscontri;

Le passività, nell'analisi di bilancio, vengono distinte secondo le scadenze di pagamento in:

- patrimonio netto;
- fondi per rischi e oneri;
- trattamento di fine rapporto di lavoro subordinato;
- debiti;
- ratei e riscontri;

Si possono individuare due criteri di riclassificazione dello stato patrimoniale per acquisire migliori informazioni sulle dinamiche aziendali: il criterio funzionale e quello finanziario.

1.1.1 Riclassificazione Funzionale (Operativo)

Secondo il criterio funzionale invece le attività (impieghi) e le passività (fonti) sono riclassificate in base all'area gestionale di appartenenza: area caratteristica/operativa (nella quale ricomprendere se marginale anche quella accessoria), comprendente tutti i valori attinenti il core business; area finanziaria, comprendente tutti i valori relativi alla negoziazione di liquidità.
Gli impieghi sono pertanto suddivisi in:

- attività operative: assets materiali e immateriali, crediti operativi, rimanenze, ratei e risconti;
- attività finanziarie: investimenti finanziari (a breve e a medio-lungo), crediti finanziari e disponibilità liquide.

Le fonti sono invece suddivise in:

- patrimonio netto: grandezza non riconducibile né all'area operativa né a quella finanziaria;
- passività operative: fondi rischi ed oneri, debiti operativi e ratei e risconti;
- passività finanziarie: ovvero i debiti finanziari a prescindere dalla scadenza.

Lo stato patrimoniale classificato secondo la logica funzionale mira a verificare l'equilibrio fra investimenti e fonti di finanziamento, e quindi di ausilio a sviluppare l'analisi della solidità

1.1.2 Riclassificazione Finanziario

Con il criterio finanziario le attività (impieghi) sono classificate e raggruppate secondo il loro grado di liquidabilità, ovvero in funzione della loro capacità di trasformarsi in liquidità in tempi più o meno rapidi, mentre le passività (fonti) in base alla loro durata temporale, ovvero in base alla loro velocità di estinzione.

L'arco temporale preso a riferimento con termine congruo per circoscrivere il breve dal medio-lungo termine corrisponde a 12 mesi.

Gli impieghi sono pertanto suddivisi, in funzione alla loro effettiva possibilità di trasformarsi in liquidità, in:

- attività correnti, atte ad essere liquidate in un arco temporale inferiore a 12 mesi, ovvero assets destinati alla vendita entro 12 mesi, attività finanziarie detenute a scopo di negoziazione, crediti in scadenza entro 12 mesi, rimanenze (per la parte che presenta un tasso di rotazione inferiore a 12 mesi), liquidità, ratei e risconti;

- attività non correnti, destinate a rimanere vincolate nel medio-lungo periodo, ovvero assets materiali, immateriali e finanziarie (eccetto quelle destinate alla vendita nel breve termine), crediti con scadenza oltre il 12 mesi, rimanenze (per la parte che presenta un tasso di rotazione inferiore a 12 mesi).

Le fonti sono invece suddivise in:

- patrimonio netto, grandezza vincolata e quindi fonte di lungo periodo;

- passività correnti, destinate al rimborso entro i 12 mesi, ossia: debiti a breve (comprese le rate a breve di finanziamenti a medio-lungo termine), ratei e risconti passivi, fondi rischi ed oneri (per la parte che avrà manifestazione finanziaria nel breve periodo);

- passività non correnti, con scadenza superiore a 12 mesi, ossia: debiti a medio-lungo, risconti passivi pluriennali, fondi rischi ed oneri (per la parte che avrà manifestazione finanziaria oltre 12 mesi).

Lo stato patrimoniale classificato secondo la logica finanziaria permette di verificare la capacità dell'azienda di far fronte ai propri impegni di breve periodo con impieghi di egual durata (capitale circolante), ed è pertanto propedeutico all'analisi della liquidità;

1.2 Conto Economico

Il Conto Economico, definito dall'art. 2425 del Codice Civile, è l'elenco, ordinato per categorie, dei costi e dei ricavi di competenza dell'esercizio, ossia di competenza di quel lasso di tempo intercorrente tra la data di riferimento del bilancio attuale e quella del bilancio precedente.

Si intendono ricavi le vendite dei propri beni, gli interessi attivi o i fitti attivi. Sono, invece, esempi di costi gli acquisti, le utenze, le spese del personale, i fitti passivi le imposte e le tasse.

In tale contesto è bene far presente che l'Iva, ancorché un'imposta, non rappresenta un costo per la società.
Si tratta, infatti, di un debito o un credito che l'azienda ha nei confronti dell'Erario nel momento in cui compie una vendita o un acquisto e, per questo motivo, trova allocazione nel passivo o nell'attivo dello Stato Patrimoniale.

Esso ha una struttura a forma scalare e una classificazione dei costi per natura (invece che per destinazione). È formato da quattro sezioni (individuate con le prime lettere dell'alfabeto), più alcune voci che illustrano il risultato d'esercizio, ante e dopo le imposte.

Sezioni che compongono il conto economico:

- valore della produzione;
- costi della produzione;
- proventi ed oneri finanziari;
- rettifiche di valore di attività e passività finanziarie.

Il Conto Economico può assumere diverse forme e configurazioni. Le riclassificazioni più utilizzate, per l'analisi di bilancio sono:

- il conto economico a margine di contribuzione, che si basa sulla suddivisione dei costi operativi tra costi fissi e costi variabili;
- il conto economico a costo del venduto, che si basa sulla suddivisione dei costi operativi tra costi diretti e costi indiretti;

- il conto economico a valore aggiunto, che si basa sulla suddivisione dei costi operativi tra costi relativi alle risorse esterne e costi relativi alle risorse interne.

Nel software è stata presa in considerazione solo la riclassificazione a valore aggiunto.

1.2.1 Riclassificazione al Valore Aggiunto

Il valore aggiunto, quale differenza tra ricavi operativi e costi operativi sostenuti per l'acquisto di risorse esterne, esprime la capacità dell'azienda di creare ricchezza per remunerare i fattori produttivi e i diversi portatori di interesse.

In particolare tale margine deve essere in grado di remunerare:

- il personale - costo del personale;
- gli investimenti - ammortamenti e svalutazioni;
- i finanziatori esterni - componenti finanziarie;
- gli eventi straordinati - componenti straordinarie;
- l'Amministrazione finanziaria - imposte.

Deve infine garantire un'adeguata remunerazione, tramite la distribuzione del risultato d'esercizio, ai soci e permettere con l'utile residuo non distribuito un adeguato autofinanziamento.

1.3 Forecast

Il Forecast è un prospetto simile a un normale bilancio. A differenza di quest'ultimo, che registra valori a consuntivo, il Forecast è una tabella che contiene valori di natura previsionale. Nello specifico, dà una previsione che guarda avanti di qualche mese.

L'impostazione e lo scopo del Forecast sono simili a quelli del budget. Cambia però il periodo di analisi. Infatti, mentre il budget è una previsione fatta sull'anno successivo, il Forecast è una previsione sull'anno in corso.

Capitolo 2

Analisi e progettazione

2.1 Descrizione del problema

Si vuole realizzare un'applicazione desktop per l'analisi di bilancio di un'azienda per l'ordine dei commercialisti della provincia di Ancona. L'applicazione deve funzionare in tutti i vari tipi di sistema operativo, perciò il software deve essere cross-platform. Per poter realizzare un prodotto che sia cross-platform, ossia implementato per più piattaforme di elaborazione, la scelta delle tecnologie da utilizzare è stata fondamentale. Quando si vuole sviluppare un'applicazione desktop la prima cosa a cui si pensa è per quale sistema operativo svilupparla, quale linguaggio usare ed opzionalmente a quale base dati si ha bisogno di connettersi. Il mondo dell'informatica intanto si sta evolvendo verso l'utilizzo di applicazioni web per la loro semplicità di distribuzione e aggiornamento, la compatibilità con ogni sistema operativo e la superfluità di un'installazione, allora perchè non sfruttare questa tecnologia? Per questo la scelta è ricaduta su un compromesso valido, Electron. Definita dallo slogan "Build cross platform desktop apps with Javascript, HTML, and CSS", Electron è una libreria open source sviluppata dal team di GitHub che permette di sviluppare un'applicazione desktop utilizzando le stesse tecnologie di un sito web, unendo la potenza del browser Chromium, la flessibilità di Node.js e il più grande ecosistema di librerie open source al mondo: npm. In aggiunta a tutto questo la possibilità di pacchettizzare l'applicazione per Mac, Windows, e Linux con lo stesso risultato in termini di grafica e funzionalità.

L'applicazione realizzata, nello specifico, dovrà permettere all'utente che la

utilizza l'inserimento in input dei dati relativi all'Anagrafica, all'Analisi Qualitativa, dello Stato Patrimoniale e del Conto Economico di un'azienda. Nella fase di inserimento dei valori dello Stato Patrimoniale e del Conto Economico, il sistema calcola le varie riclassificazioni dello Stato Patrimoniale (Funzionale, Finanziario o entrambi, a seconda della scelta da parte dell'utente) e del Conto Economico (al Valore Aggiunto). L'utente deve avere la possibilità di salvare in corso d'opera il progetto e poterlo riaprire in un secondo momento. Inoltre, va implementata una funzione che permette la stampa in un file pdf del progetto.

2.2 Analisi dei requisiti

Al fine di esporre i requisiti in maniera rigorosa, verrà adottata la notazione MoSCoW, la quale fa uso delle seguenti etichette:

- Must Have: sottolinea un requisito di fondamentale importanza per il funzionamento del sistema;
- Should Have: indica un requisito importante ma non essenziale per il funzionamento del sistema;
- Could Have (o Nice to Have): indica un requisito non importante come i precedenti e volto principalmente a migliorare la soddisfazione del cliente.
- Won't Have: requisiti che non devono essere implementati (negazione di una funzione).

2.2.1 Requisiti funzionali

I requisiti funzionali descrivono le funzionalità del sistema software, in termini di servizi che il sistema software deve fornire, di come il sistema software reagisce a specifici tipi di input e di come si comporta in situazioni particolari.

ID	Descrizione	MoSCoW
1	Il programma deve permettere l'inserimento e la visualizzazione dei dati dell'azienda di cui si vuole analizzare il bilancio	Must have
2	Il programma deve permettere all'utente il salvataggio del progetto	Must have
3	Il programma deve permettere all'utente l'apertura di un progetto salvato in precedenza	Must have
4	Il programma deve permettere all'utente di scegliere quale tipo di riclassificazione per lo stato patrimoniale utilizzare	Must have
5	Il programma deve calcolare le riclassificazioni sia per lo stato patrimoniale che per il conto economico	Must have
6	Il programma deve permettere all'utente di poter inserire l'annualità di riferimento dell'analisi	Should have
7	Il programma deve permettere all'utente di poter stampare in un formato pdf del progetto	Should have
8	Il programma deve calcolare in automatico il forecast per le annualità previste dall'utente	Could have

Tabella 2.1: Requisiti funzionali

2.2.2 Requisiti non funzionali

Descrivono le proprietà del sistema software in relazione a determinati servizi o funzioni e possono anche essere relativi al processo:

- Caratteristiche di efficienza, affidabilità, safety, ecc.
- Caratteristiche del processo di sviluppo (standard di processo, linguaggi di programmazione, metodi di sviluppo, ecc.
- Caratteristiche esterne (interoperabilità con sistemi di altre organizzazioni, vincoli legislativi, ecc.)

ID	Descrizione	MoSCoW
1	Il programma deve essere sviluppato per tutte le piattaforme di elaborazione	Must have
2	Il programma dovrebbe controllare i formati di immissione dei dati nella maschera di input	Should have
3	L'avvio e in particolar modo la ricezione dell'input da parte dell'utente, dovrebbe essere veloce	Should have
4	Il programma deve calcolare ina maniera veloce le riclassificazioni	Should have
5	Il programma può essere adattabile alle differenti risoluzioni dello schermo (responsive)	Could have

Tabella 2.2: Requisiti non funzionali

2.3 Use case diagram

Gli Use Case Diagrams descrivono il comportamento funzionale del sistema, come visto dall'utente. Sono diagrammi di facile interpretazione dove abbiamo uno o più attori che possono attivare diverse funzioni/servizi all'interno di uno o più sistemi. Servono a riepilogare i dettagli degli utenti del sistema (noti anche come attori) e le loro interazioni con il sistema.

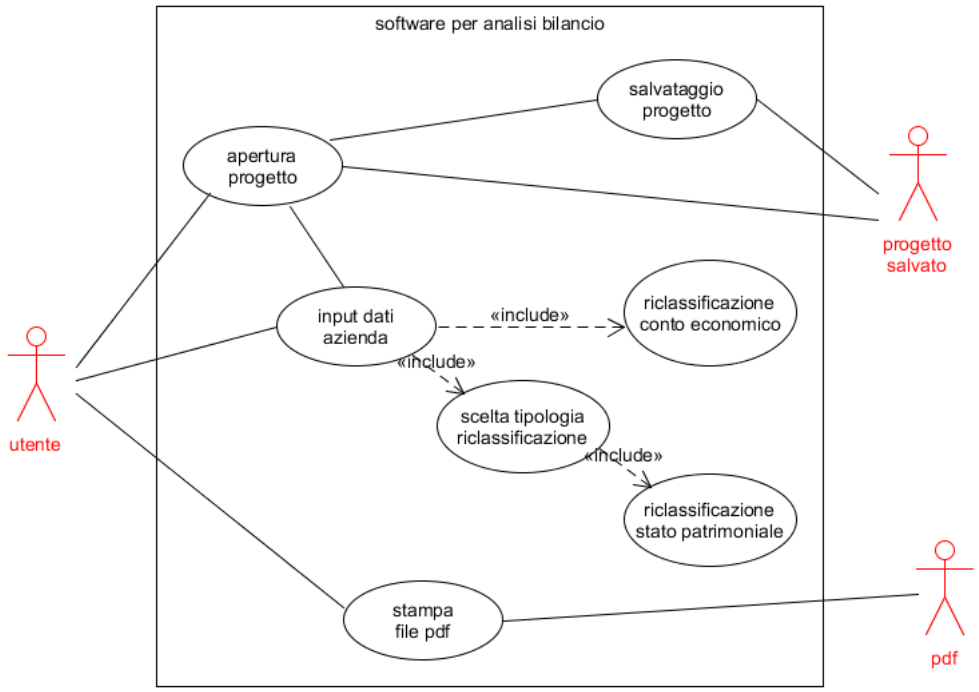

Figura 2.1: UML Use Case Diagram

Nel diagramma in figura 2.1 abbiamo tre attori, l'utente che inserisce i dati dell'azienda, il file contenente il progetto salvato e il file pdf generato dal software. I casi d'uso talvolta includono più passaggi al loro interno, in quei casi è indicata una freccia "include", come nel caso del "input dati azienda" che include "scelta tipologia riclassificazione" che a sua volta include "riclassificazione stato patrimoniale".

2.4 Activity diagram

L'Activity Diagram è un diagramma che definisce le attività da svolgere per realizzare una data funzionalità ed è spesso utilizzato come modello complementare dello Use Case Diagram.

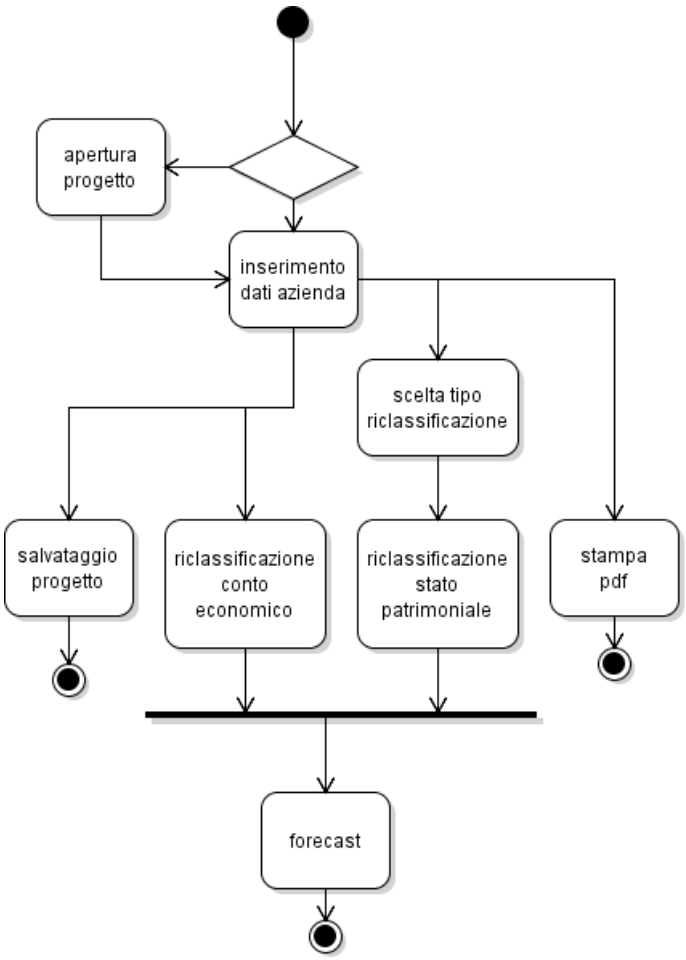

Figura 2.2: UML Activity Diagram

Nella figura 2.2 vediamo descritto il software come nello Use Case precedente, ma dal punto di vista delle attività da svolgere in sequenza per raggiungere un certo risultato. Il diagramma si legge a partire dal pallino in alto, detto nodo iniziale, seguono le azioni indicate dai rettangoli. I rombi indicano le decisioni, cioè le diverse direzioni che può prendere il flusso a seguito di certe condizioni o scelte, ad esempio all'avvio del software se si vuole aprire un progetto salvato precedentemente o inserire i dati per un nuovo progetto. Un altro

elemento dell'activity diagram è la "join", che indica che per passare all'azione successiva devono essere completate tutte le azioni in ingresso, come ad esempio per il calcolo del forecast, che avviene una volta inseriti i dati e quindi calcolate le riclassificazioni dello stato patrimoniale e del conto economico.

2.5 Sequence diagram

Il sequence diagram è uno strumento di modellazione visuale che mette in evidenza come i vari componenti di un sistema interagiscono tra di loro in relazione al tempo.

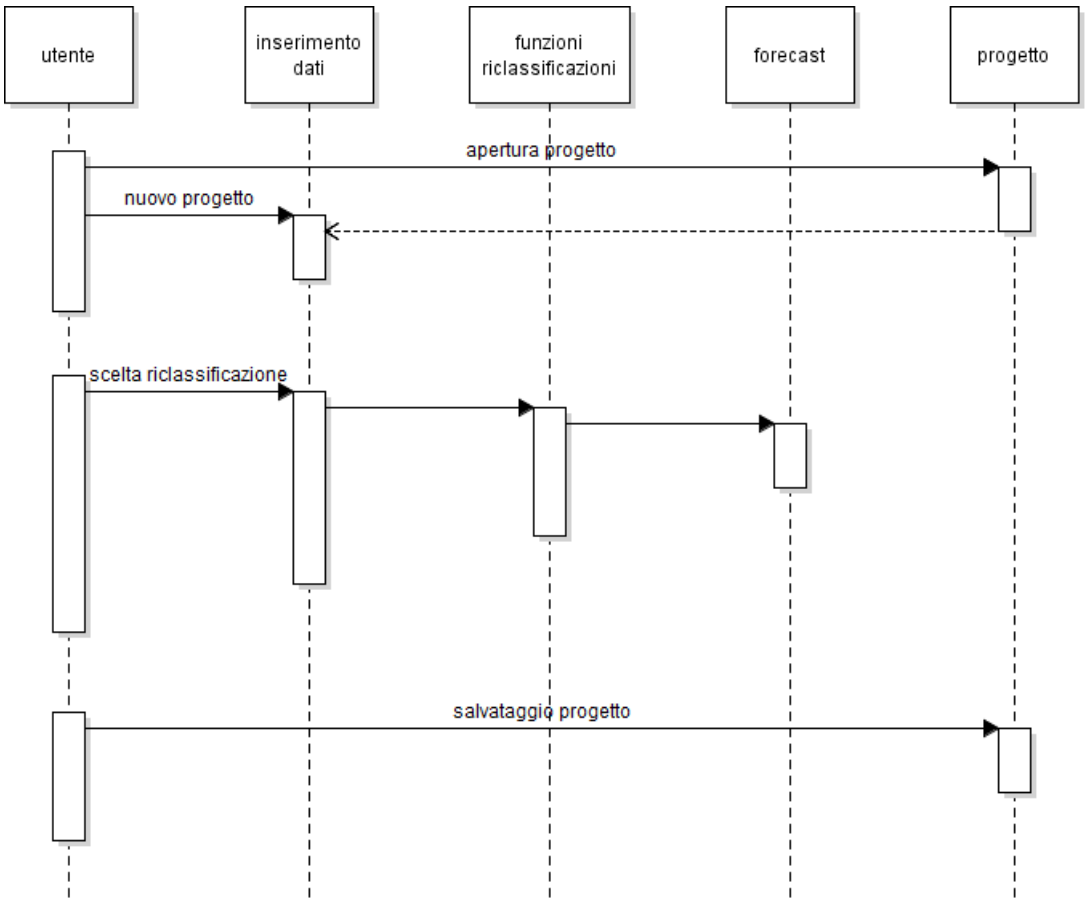

Figura 2.3: UML Sequence Diagram

Nella figura 2.3 vediamo descritto il flusso iniziale del software che inizia con la scelta da parte dell'utente di creazione di un nuovo progetto o di aprire un progetto salvato precedentemente e termina con il salvataggio dello stesso dopo aver calcolato tutte le riclassificazioni. I componenti del diagramma sono l'utente, la form che permette l'input dei dati, le funzioni per le riclassificazioni, la funzione del forecast e il progetto salvato esternamente. Nella fase di inserimento dei dati da parte dell'utente, il sistema calcola le varie riclassificazioni, tendendo conto della scelta dell'utente della tipologia scelta per lo Stato

Patrimoniale, e mette tutto a disposizione per il salvataggio dei dati e per la stampa del pdf.

Capitolo 3

Sviluppo

Quando si vuole sviluppare un'applicazione desktop la prima cosa a cui si pensa è per quale sistema operativo svilupparla, quale linguaggio usare ed opzionalmente a quale base dati si ha bisogno di connettersi. In questo caso la scelta è caduta su Electron, un framework Open Source sviluppato dal team di GitHub che permette lo sviluppo di app cross platform utilizzando il medesimo approccio di Apache Cordova, ovvero attraverso JavaScript, HTML, e CSS. Si tratta di un modulo Npm, quindi eredita tutte le API e le funzionalità di Node.js.
Il vero potenziale di Electron è dato dal fatto che si può costruire una desktop app utilizzando qualsiasi libreria scritta in Node.js, rendendolo di fatto uno strumento molto potente anche per interagire con il sistema operativo.
Nello specifico dello sviluppo del sofware, si possono distinguere tre specifiche aree:

- Front-End, comprende tutto ciò che riguarda l'interfaccia utente, quindi come presentare i dati passati dal Back-End, la grafica, il template, il typewriting, l'accessibilità e l'usabilità.

- Back-End, comprende tutto ciò che riguarda lo sviluppo della logica del software, ovvero cosa fare con i dati, come trattare quelli ricevuti, cosa restituire all'utente.

- Packaging dell'App, la creazione del prodotto finale, ovvero il software in versione eseguibile per tutte le tipologie di piattaforme di elaborazione.

3.1 Sviluppo del Front-End

Per lo sviluppo di un interfaccia utente efficace si deve tener conto di due aspetti oltre a quello tecnico, che sono:

- Accessibilità
- Usabilità

In generale i requisiti di accessibilità sono i seguenti:

- Alternative testuali, obbligatorie per tutte i contenuti non testuali (come immagini, filmati e audio);
- Adattabilità, cioè prevedere che i contenuti si adattino a diversi layout in base alla grandezza e al formato dello schermo utente (responsive web design);
- Distinguibile, rendere più semplice agli utenti la visione e l'ascolto dei contenuti, separando i contenuti in primo piano dallo sfondo;
- Accessibile da tastiera, cioè garantire una buona navigabilità prevedendo un percorso di TAB;
- Colori, il contrasto tra le scritte e lo sfondo deve essere chiaro, ma non si devono usare colori discordanti tra loro o lampeggianti che possano causare crisi epilettiche;
- Navigabile, fornire una struttura del sito chiara, dove l'utente possa orientarsi e raggiungire qualsiasi sezione attraverso links;
- Assistenza per l'inserimento, fornire gli aiuti e le spiegazioni necessarie affinché l'utente sia in grado di compilare correttamente le form del sito;
- Compatibilità, il sito deve essere accessibile da tutte le piattaforme e deve utilizzare tecnologie standard.

L'**usabilità** è un approccio alla progettazione volto a rendere l'interazione tra il prodotto e l'utente migliore sotto i seguenti aspetti:

- Efficacia, cioè permettere agli utenti di raggiungere i loro obiettivi in maniera precisa e completa;
- Efficienza, cioè l'ottimizzazione delle risorse impiegate;
- Soddisfazione, come libertà dal disagio e attitudine positiva con cui gli utenti raggiungono specifici obiettivi attraverso l'uso del prodotto.

L'usabilità invece si pone i seguenti obiettivi:

- Presentare l'informazione all'utente in modo chiaro e conciso, evitando termini tecnici o specialistici;
- Semplificare la struttura del compito;
- Offrire all'utente le scelte corrette, in una maniera che risulti ovvia;
- Organizzare ogni pagina in modo che l'utente riconosca la posizione e le azioni da compiere;
- Eliminare ogni ambiguità relativa alle conseguenze di un'azione (es. fare clic su cancella/rimuovi/compra);
- Mettere la cosa più importante nella posizione giusta della struttra;
- Fare in modo che l'utente abbia un rapido feedback ad ogni azione compiuta, ad esempio la comparsa di un messaggio di successo o errore all'invio di dati con una form;
- Rendere la grafica accattivante ed interessante dal punto di vista visivo attraverso l'uso di diagrammi, tabelle, sezioni informative e coordinando bene i colori;
- Ridurre gli sforzi cognitivi dell'utente.

In generale cercare di rendere il sistema il più semplice possibile da usare, in modo da ridurre al minimo gli sforzi sull'utilizzo del mezzo.

3.1.1 Tecnologie usate nello sviluppo del front-end

Mentre per lo sviluppo del back-end abbiamo a disposizione una gran quantità di tecnologie concorrenti per svolgere lo stesso compito, per il front-end le tecnologie di base sono sempre le stesse: Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript e Jquery.

HTML è un linguaggio di markup che serve per descrivere le modalità di impaginazione, formattazione o visualizzazione grafica (layout) del contenuto, testuale e non, di una pagina web. Per specificare tali disposizioni e proprietà si utilizzano specifici marcatori chiamati "tag", i quali delimitano gli elementi stessi (ad esempio <p>per i paragrafi, <h1>per un titolo di primo livello, <address>per informazioni relative ad un indirizzo, etc.).

La versione più recente è chiamata HTML 5 e rispetto alle versioni precedenti ha l'aggiunta di alcuni tag per gli elementi mutlimediali, una rivisitazione delle regole di strutturazione del testo riguardante sezioni, paragrafi e capitoli, il miglioramento i controlli di input per le form, il controllo della geolocalizzazione e l'intruduzione dello Web Storage come alternativa ai cookies.

CSS descrive come devono essere visualizzati gli elementi in una pagina web. A differenza dell'HTML che è usato per strutturare un documento (definendo cose come titoli e paragrafi e permettendo di incorporare immagini, video e altri media), il CSS passa attraverso e specifica i layout di stile della pagina del documento, i colori e i caratteri. Il codice CSS può essere esterno, interno o in linea, nel software realizzato la scelta è stata quella di separare in due file distinti il codice HTML e il codice CSS.

```css
html, body {
    width: 100%;
    height: 100%;
    margin-top: 30px;
    background-color: white;
    overflow: scroll;
}

.form-group input {
    width: 300px;
}

.form-group .text-area {
    resize: vertical;
    width: 40%;
}

.form-group .totale {
    background-color: rgb(204, 255, 51) !important
}
```

Figura 3.1: esempio di utilizzo di css nel software

JavaScript è rimasto confinato per quasi 2 decenni nel browser, prima di diventare un linguaggio di programmazione anche per lo sviluppo del backend con la nascita di Node.js. Lato front-end comunque, Javascript viene utilizzato principalmente per aggiungere effetti dinamici interattivi basati su eventi, come il click di un bottone, la selezione di una voce da una tendina, la pressione di un tasto o il caricamento di una pagina, ed è comunque consigliato per aumentare l'esperienza utente.

```
2546  <script>
2547    // Clicco l'elemento con "defaultOpen"
2548    document.getElementById("defaultOpen").click();
2549
2550    function apriTab(evt, nomeTab) {
2551      var i, tabcontent, tablinks;
2552
2553      tabcontent = document.getElementsByClassName("tabcontent");
2554      for (i = 0; i < tabcontent.length; i++) {
2555        tabcontent[i].style.display = "none";
2556      }
2557
2558      tablinks = document.getElementsByClassName("tablinks");
2559      for (i = 0; i < tablinks.length; i++) {
2560        tablinks[i].className = tablinks[i].className.replace(" active", "");
2561      }
2562
2563      document.getElementById(nomeTab).style.display = "block";
2564      evt.currentTarget.className += " active";
2565    }
2566  </script>
```

Figura 3.2: javascript per selezione tab

Nella figura 3.2 viene mostrato un esempio di come Javascript venga utilizzato lato front-end, nel caso specifico per la selezione e la visualizzazione delle tab del software.

jQuery è una libreria JavaScript che contiene tutto il necessario per creare interazioni avanzate per le pagine web, con lo scopo di semplificare la selezione, la gestione e la manipolazione degli eventi, mantenendo però anche la possibilità di utilizzare il linguaggio nella vecchia maniera.
L'uso di jQuery non aggiunge quindi nessuna funzionalità, rende solamente più semplice l'utilizzo del linguaggio e quindi più elevata la produttività del programmatore, non a caso il suo motto è "Write less, do more".

```
<script>
  $(document).ready(function () {
    function switchRiclassificazioneSP() {
      var selectedOption = $(".dropdownRiclassificazioneSP  option:selected").val();
      if (selectedOption == 'funzionale') {
        $(".grpRiclassificazioneSPFunzionale").show();
        $(".grpRiclassificazioneSPFinanziario").hide();
      }
      else if (selectedOption == 'finanziario') {
        $(".grpRiclassificazioneSPFunzionale").hide();
        $(".grpRiclassificazioneSPFinanziario").show();
      }
      else {
        $(".grpRiclassificazioneSPFunzionale").show();
        $(".grpRiclassificazioneSPFinanziario").show();
      }
    }
    $(".dropdownRiclassificazioneSP").change(function () {
      switchRiclassificazioneSP();
    });
    switchRiclassificazioneSP();
  });
</script>
```

Figura 3.3: jquery per scelta tipologia riclassificazione

```
2519    <script>
2520      $(document).ready(function () {
2521        // utilizzo la funzione maskMoney per impostare il formato della valuta
2522        $('.currency').maskMoney();
2523        $('.totale').maskMoney();
2524        // utilizzo la funzione maskMoney per impostare il formato della percentuale
2525        $('.percentuale').maskMoney({suffix:" %"})
2526        // impedisco l'inserimento dati nei campi con la proprietà readonly
2527        $('input[readonly]').keydown(function(){
2528          return false;
2529        })
2530    </script>
```

Figura 3.4: jquery per maskmoney

Jquery nell'applicazione realizzata, come si può vedere nelle figure 3.3 e 3.4, viene utilizzato nello sviluppo lato back-end per la selezione della tipologia di riclassificazione, e attraverso l'utilizzo dei selettori per richiamare la funzione MaskMoney, importata da una libreria esterna, la quale permette la formattazione dei valori nell'interfaccia, a seconda della tipologia di dato (valuta o percentuale).

3.2 Sviluppo del Back-End

Nella struttura di un app basata su Electron, bisogna discutere di due tipi di processo esistenti:

- il processo chiamato **main** che esegue lo script indicato nel file package.json è il processo principale. Lo script che viene eseguito nel processo principale può visualizzare una Graphical User Interface (GUI) tramite la creazione di pagine web. Il processo principale crea pagine web mediante la creazione di istanze di BrowserWindow, dove vengono eseguite nel proprio processo di rendering. Quando viene eliminata un'istanza di BrowserWindow, il processo di rendering corrispondente viene anch'esso terminato. Il processo principale quindi, gestisce tutte le pagine web e il corrispondente processo di rendering.

- il processo di **rendering**, invece è il processo che gestisce ogni pagina web, visualizzata tramite Chromium con la sua architettura multiprocesso, tale processo è isolato e può occuparsi solo delle pagine web in esecuzione in esso. Pertanto nelle pagine web chiamare le API dell'interfaccia grafica nativa non è consentito perché la gestione delle risorse di sistema nelle pagine web è molto pericolosa ed è facile perdere risorse. Se si desidera eseguire operazioni di GUI in una pagina web, il processo di rendering della pagina web deve comunicare con il processo principale per richiedere che il processo principale esegua tali operazioni.

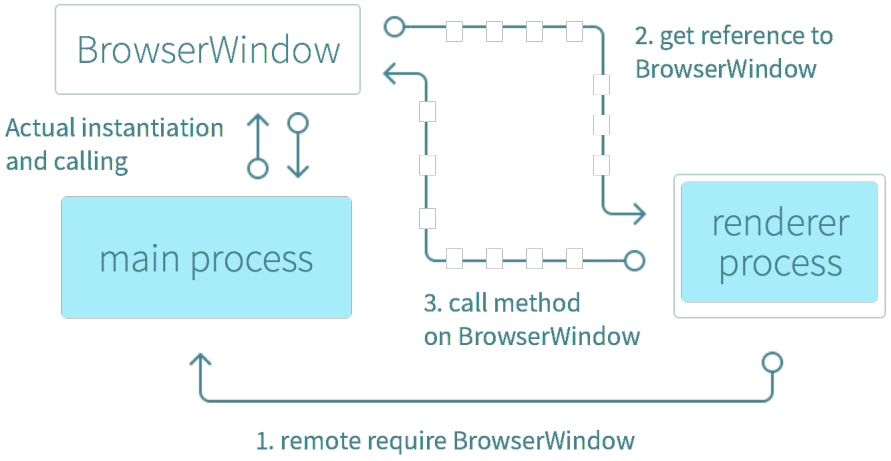

Figura 3.5: Electron - Architettura di un app

3.2.1 Utilizzo delle API Electron

Electron offre una serie di API che supportano lo sviluppo di un'applicazione desktop sia nel processo principale che nel processo di rendering. In entrambi i processi, devi accedere alle API di Electron richiedendo il relativo modulo incluso:

```
1
2   const electron = require('electron')
3
```

Figura 3.6: importazione API Electron

A tutte le API Electron viene assegnato un tipo di processo. Molti di essi possono essere utilizzati solo dal main, alcuni solo da un processo di rendering, alcuni da entrambi. Ad esempio, una finestra in Electron viene creata utilizzando la classe BrowserWindow, disponibile solo nel processo principale.

```
3
4   const { app, BrowserWindow, Menu } = require('electron')
5
```

Figura 3.7: importazione classe BrowserWindow

```
55
56  // Creo la finestra iniziale
57  window = new BrowserWindow({
58    // Imposto il colore di sfondo
59    backgroundColor: "#D6D8DC",
60    show: false
61  })
62  // visualizzo finestra in fullscreen
63  window.maximize()
64
```

Figura 3.8: Creazione BrowserWindow

3.2.2 Apertura e salvataggio del progetto

Durante l'utilizzo del software l'utente deve avere la possibilità attraverso una funzione apposita, di poter salvare un progetto in stato d'opera e/o di leggere

un progetto precedentemente salvato. Per la realizzazione di questa funzionalità, si è scelto di non utilizzare un database ma di un semplice salvataggio dei dati attraverso un file JavaScript Object Notation (JSON) nel file system della piattaforma di elaborazione in uso. JSON è un modo per archiviare le informazioni in modo organizzato e di facile accesso per l'utilizzo delle stesse all'interno di una applicazione (web, mobile o di qualsiasi altro tipo).

```
7   function json() {
8
9       //creo array per json
10      let array = []
11      let testoProgetto = {}
12      // seleziono tutti gli id presenti nell'html
13      let listaId = document.querySelectorAll("[id]")
14
15      for (var i = 0; i < listaId.length; i++) {
16          var newName = listaId[i].id
17          var newValue = listaId[i].value
18          testoProgetto[newName] = newValue
19      }
20      // inserisco i valori nell'array per il json
21      array.push(testoProgetto)
22
23      var json = JSON.stringify(array);
24      return json;
25  }
```

Figura 3.9: Creazione Json

Nella figura 3.9 viene mostrata l'implementazione della funzione "json" per la creazione del file JSON contenente i valori inseriti nel progetto. Inizialmente, utilizzando la funzione querySelectorAll vengono selezionati tutti gli Id contenuti nell'html del main, successivamente, attraverso un ciclo for per l'iterazione degli stessi, vengono inseriti nell'array contenuto nel JSON.

```
26
27  // salvataggio progetto
28  ipc.on('salva', function (ev, data) {
29
30      //mostro finestra per salvataggio file
31      dialog.showSaveDialog((filename) => {
32          if (filename == undefined) {
33              window.alert("non hai salvato il progetto...")
34          }
35          //salvo file
36          fs.writeFile(filename, json(), function (err) {
37              if (err) {
38                  window.alert("Errore: " + err.message)
39                  alert.name()
40              }
41
42              window.alert("Progetto Salvato!")
43          })
44      })
45  })
46
```

Figura 3.10: funzione salvataggio

Electron per comunicare tra processi renderer e main, in entrambe le direzioni, utilizza la comunicazione interprocesso Inter-Process Communication (IPC).
Come mostrato nella figura 3.10 la funzione relativa al salvataggio del progetto viene richiamata tramite IPC al messaggio 'salva', e grazie all'utilizzo delle API di Node.js, viene creato all'interno del file system, il file JSON passato dalla funzione "json()" precedentemente illustrata.

3.2.3 Calcolo riclassificazioni e forecast

Come già visto precedentemente, per l'analisi di bilancio di un'azienda spesso si utilizzano diverse tipologie di riclassificazione, sia per lo stato patrimoniale che per il conto economico. Nel software realizzato si è deciso di mettere a disposizione all'utente, per quanto riguarda lo stato patrimoniale la scelta tra la riclassificazione finanziario, funzionale o entrambe, mentre per il conto economico viene calcolata solo la riclassificazione al valore aggiunto. Le riclassificazioni calcolate vengono poi usate per calcolare gli indici per il forecast per l'anno di riferimento del progetto. Per il calcolo delle varie riclassificazioni e forecast sono state realizzate delle specifiche classi Javascript, in cui attraverso l'utilizzo di Jquery per l'acceso al Document Object Model (DOM) e attraverso la funzione importata con la libreria del MaskMoney illustrata precedentemente, vengono applicate tutte le formule del caso. Dove c'è la necessità di una sommatoria di più valori, ove possibile, è stato utilizzato l'attributo name del DOM e attraverso un ciclo for calcolato il totale. Come in figura 3.11, dove viene mostrato l'esempio per il Totale patrimonio netto, calcolato prendendo tutti gli elementi che hannno l'attributo name all'interno dei vari tag impostato a SPFOPatrimonioNetto

```
function RFOsommaPatrimonioNetto() {
    var arr = $(document.getElementsByName('SPFOPatrimonioNetto')).maskMoney('unmasked');
    var tot = 0;
    for (var i = 0; i < arr.length; i++) {
            tot += arr[i];
    }
    $('#SPFOtotalePatrimonioNetto').maskMoney('mask',tot)[0]
}
```

Figura 3.11: for per somma riclassificazione

Altrimenti, come nel caso specifico del forecast, in cui non poteva essere usato l'attributo name, si fa riferimento all'attributo id. Un esempio in figura 3.12

```
64      /* Tab. 2 - Capitale circolante */
65      // crediti verso clienti
66      var FCcreditiVersoClienti =
67      $('#SPFcreditiCommerciali').maskMoney('unmasked')[0] /
68      ($('#CEricaviVendite').maskMoney('unmasked')[0] / 365)
69
70      $('#FCcreditiVersoClienti').maskMoney('mask',FCcreditiVersoClienti)[0]
```

Figura 3.12: esempio di calcolo per il forecast

3.2.4 Creazione pdf

Per la creazione di un report dei dati inseriti e calcolati, è stata utilizzata la comunicazione interprocesso IPC.

```
81  const printPDFbutton = document.getElementById('print-pdf')
82
83  printPDFbutton.addEventListener('click', event => {
84      ipc.send('print-to-pdf')
85  })
```

Figura 3.13: input da pulsante

Nel processo di rendering viene impostato un handler atrraverso la funzione addEventListener() per poter inviare al processo principale il messaggio 'print-to-pdf', non appena l'utente che utilizza il sistema clicca sul bottone per la stampa del pdf, come da figura 3.13

```
// creazione pdf
ipc.on('print-to-pdf', event => {
  const pdfPath = path.join(os.tmpdir(), 'Analisi Bilancio.pdf')

  const win = BrowserWindow.fromWebContents(event.sender)

  win.webContents.printToPDF({marginsType: 1, pageSize:'Tabloid'}, (error, data) => {
    if (error) return console.log(error.message)

    fs.writeFile(pdfPath, data, err => {
      if (err) return console.log(err.message)
      shell.openExternal('file://' + pdfPath)
    })
  })
})
```

Figura 3.14: creazione e apertura del pdf

Il processo principale, ricevuto il messaggio 'print-to-pdf' dal processo renderer, genera il file pdf in un percorso temporaneo del sistema operativo in uso e lo apre non appena creato.

```
@media print {
  /* Per stampare tutto l'html */
  body {
      overflow: visible !important;
      -webkit-print-color-adjust: exact !important;
  }
  /* nascondo la scrollbar */
  ::-webkit-scrollbar {
      width: 0px;
      background: transparent;
  }
  /* nascondo il bottone per stampare il Pdf */
  .bottoneStampaPdf {
      display: none
  }
  /* nascondo i pulsanti delle tab */
  .tab {
      display: none
  }
  /* rendo visibili le intestazioni delle tab, solo per stampa */
  .tabcontent h2 {
      display: block !important;
      -webkit-print-color-adjust: exact !important;
      color: rgb(100, 17, 17) !important;
  }
  /* mostro il contenuto di tutte le tab nella stampa, e genero un'altra pagina alla fine di ognuna */
  .tabcontent {
      display: block !important;
      page-break-after: always;
  }
```

Figura 3.15: specifica Css per stampa documenti

Il pdf viene generato grazie all'aiuto della specifica CSS per la stampa dei documenti, mostrato in, dove tra le varie istruzioni, vengono nascosti alcune parti del front end, come la scrollbar o i pulsanti delle tab, per mostrare invece tutta la parte del frontespizio del pdf generato.

3.3 Packaging dell'applicazione

L'imballaggio e la distribuzione di app sono parte integrante del processo di sviluppo di un'applicazione desktop, poiché Electron è un framework di sviluppo di applicazioni desktop multipiattaforma, anche l'imballaggio e la distribuzione di app per tutte le piattaforme dovrebbero essere un'esperienza senza soluzione di continuità. Per questo scopo, è stato utilizzato uno strumento da riga di comando e una libreria Node.js che ci consente di confezionare e distribuire la nostra app Electron con pacchetti specifici del sistema operativo (.app, .exe, ecc.), tramite Command Line Interface (CLI), chiamato Electron Packager. Electron Packager funziona per le seguenti piattaforme:

- Windows (32/64 bit)
- OS X (noto anche come macOS)
- Linux (x86/ x86-64)

Per utilizzare il pacchetto è necessario prima installarlo, usando all'interno della cartella del proprio progetto la riga di comando: "npm install electron-packager –save-dev". Per creare il risultato finale del software il comando da eseguire è "electron-packager <dir><nomeApp>–platform= <platform>–arch= <arch>" dove:

- <dir>è il percorso dove viene creata l'applicazione
- <nomeApp>è il nome dell'applicazione
- <platform>è la piattaforma di destinazione
- <arch>è la specifica dell'architettura del sistema operativo

Capitolo 4

Possibili sviluppi futuri

Ogni regione è stata chiamata per legge a sviluppare il proprio sistema di fatturazione e, eventualmente, a porsi come nodo locale di interscambio per le fatture degli enti locali. In Regione Lazio il sistema **HUB** svolge appunto questo ruolo, esso prevede un sito per l'adesione al sistema da parte delle e dei privati a tre tipi di servizio: fattura attiva, fattura passiva, conservazione. La Regione Emilia-Romagna predispone anch'essa un sistema di interscambio regionale chiamato **NoTI-ER**, il quale una volta ricevuta la fattura dalloprima la converte nel formato europeo PEPPOL, poi la invia al software **SICIPA-ER** (l'equivalente del nostro Fatto) e al sistema di conservazione Par-ER. Troviamo un sistema di interscambio anche per la Regione Toscana, chiamato **fERT**, che oltre alla comunicazione tr e Ente, si occupa anche della comunicazione con la In Regione Marche il ruolo di intermediario svolto da è quello di mettere in comunicazione lo con Fatto e Fatto con il sistema di protocollo, quindi presiede a tutto il processo di fatturazione fino al momento dell'accettazione/rifiuto. Successivamente le comunicazioni con la passano per il software della contabilità Siagi, il quale prima di inviare i dati relativi ai pagamenti, preleva le informazioni necessarie (associazioni tra fatture e decreti, tempi di accettazione e contabilizzazione, etc) da Fatto.

Conclusione

La mia collaborazione alla realizzazione di questo progetto mi ha permesso di valutare in un contesto concreto l'efficacia sia delle metodologie di analisi e progettazione, sia delle tecnologie di sviluppo utilizzate. Essendo stato partecipe inoltre ad alcuni processi decisionali nel ruolo di referente tecnico, ho potuto confermare l'importanza di una buona scrittura dei requisiti funzionali e non, accompagnata da una gestione delle attività e delle scadenze.

Dal punto di vista del personale amministrativo non ci sono state difficoltà nell'apprendere l'utilizzo del software, anche grazie agli accorgimenti presi per il front-end e alla continua disponibilità, mia e dei miei referenti. In ogni caso alcune problematiche di dominio sono venute fuori solo successivamente al rilascio della prima versione del software, ciò ha comportato un continuo lavoro di rianalisi e riadattamento che ha interessato tutte le parti responsabili dei software coinvolti nel processo di fatturazione.

Parlando di vantaggi al cittadino, da un lato si sono ottenuti dei tempi certi per quanto riguarda l'accettazione e il rifiuto (garantiti dalla spietatezza dei contatori informatici) e si è indirettamente dato uno standard ai vari software proprietari e non per la gestione delle fatture. Da un altro lato però sono aumentati gli oneri per i fornitori, perché anche ammettendo che si riesca a realizzare autonomamente l'xml della fattura da inviare o che il vecchio software di fatturazione in proprio possesso sia stato aggiornato senza ulteriori spese, si deve far fronte alle spese per la conservazione digitale, ovvero alle spese per un software di conservazione aderente agli standard e alle marche temporali digitali.

Tutti questi problemi spero si risolvano nel futuro, scegliendo magari di gestire la conservazione direttamente lato, con un maggiore sforzo implementativo da parte del Ministero.

Complessivamente mi ritengo soddisfatto di quanto fatto finora, ma sono consapevole che la realizzazione del progetto non è che un passo verso una complessiva digitalizzazione e sostanziale modifica del funzionamento della.

I futuri sviluppi che interesseranno il sistema di fatturazione saranno legati al prossimo rilascio da parte della dei web service per lo scambio delle informazioni relative ai pagamenti effettuati. Non è esclusa tuttavia anche una possibile adozione degli standard PEPPOL, relativi al processo di fatturazione o anche una parziale ridefinizione di alcuni processi amministrativi digitali.

Dediche

Bibliografia

Libri

[1] S. Gaburri (Traduttore) Martin Fowler (Autore) L. Baresi (a cura di). *UML distilled - Guida al linguaggio di modellazione standard, Quarta Edizione*. Pearson, 1981. ISBN: 978-8871925981.