Università degli studi di Camerino

Scuola di Scienze e Tecnologie

Corso di Laurea in Informatica (L-31)

Analisi di bilancio: progettazione e sviluppo di un applicazione desktop per l'ordine dei commercialisti di Ancona

Laureando

Danilo Paoletti

Matricola 085353

Relatrice

Dr.ssa Barbara Re

Anno accademico 2017-2018

Dediche

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 nel contesto economico l'analisi di bilancio, poi sarà discussa la fase di analisi e progettazione che ha preceduto la realizzazione del software, descrivendo attraverso diagramma Unified Modeling Language (UML) [**uml**] i vari requisiti, vincoli e funzionalità del prodotto.
Successivamente 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 sarà resa scaricabile dal sito dell'Ordine e deve essere operativa per 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. Nella fase di inserimento dei valori dello Stato Patrimoniale e del Conto Economico, il sistema calcola tramite formule pre-impostate, le varie riclassificazioni dello Stato Patrimoniale (Funzionale, Finanziario o entrambi) 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'autenticazione dell'utente tramite Cohesion SSO in tutte le modalità (debole, forte, dominio e smart card)	Must have
2	Il programma deve permettere la ricerca e la visualizzazione delle fatture visibili al proprio ruolo	Must have
3	Il programma deve gestire gli stati di avanzamento del pagamento delle fatture	Must have
4	Il programma deve permettere di accettare/rifiutare le fatture	Must have
5	Il programma dovrebbe permettere di inserire manualmente i decreti a tutti i dipendenti di strutture che non hanno ancora adottato il software OpenAct	Should have
6	Il programma dovrebbe permettere di inserire manualmente i mandati provvisori	Should have
7	Il programma dovrebbe tenere traccia degli stati della fattura in ordine cronologico	Should have
8	Il programma potrebbe fornire dei report che presentino dati aggregati sulle fatture di interesse statistico e strategico	Could have
9	Il programma potrebbe permettere all'utente di sospendere il pagamento della fattura (interrompendo il conteggio dei giorni) fino al termine della sospensione	Could have
10	Il programma potrebbe permettere il download in un formato open-data (CSV) delle fatture trovate	Could have
11	Il programma non deve permettere di creare mandati e decreti che non siano già presenti negli altri gestionali specifici	Won't 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 interoperabile con il nodo di interscambio regionale (IntermediaMarche) per quanto riguarda la ricezione di fatture, le notifiche di scarto e le notifiche di accettazione/rifiuto	Must have
2	Il programma deve essere interoperabile con il software del protocollo (Paleo) per quanto riguarda la ricezione di messaggi di assegnazione RP alle fatture	Must have
3	Il programma deve essere interoperabile con il software dei decreti (OpenAct) per la ricezione dei decreti di pagamento fatture	Must have
4	Il programma deve essere interoperabile con il software della contabilità (Siagi) per la ricezione dei mandati di pagamento e l'invio delle informazioni sulle fatture per la comunicazione alla PCC	Must have
5	Deve essere garantita la consistenza tra le informazioni presenti nel programma e quelle presenti negli altri programmi gestionali collegati	Must have
6	Il sito dovrebbe essere visibile da ogni browser (cross-browser)	Should have
7	La navigazione del sito, in particolar modo il caricamento delle pagine di ricerca, dovrebbe essere veloce	Should have
8	Il sito può essere adattabile alle differenti risoluzioni dello schermo (responsive)	Could have

Tabella 2.2: Requisiti non funzionali

2.2.3 Requisiti di dominio

Si tratta di requisiti derivati dal dominio applicativo del sistema software piuttosto che da necessità dettate dagli utenti. Sono pertanto requisiti del tutto ovvi a persone che lavorano nel dominio (vedi esistenza di leggi giuridiche, regola matematica, legge fisica, ecc.) e sono spesso difficili da capire per chi non ha conoscenze nel dominio stesso.

ID	Descrizione	MoSCoW
1	Il database del programma deve essere in un server MSSQL	Must have
2	Il sito deve essere ospitato da una macchina Windows Server 2008 o superiore con IIS e .NET installati	Must have
3	Il programma deve rispettare gli obblighi previsti dalla normativa italiana in materia di fatturazione elettronica	Must have
4	Il programma deve essere in grado di interpretare i dati della fattura xml inviata da Intermedia e definiti nelle regole tecniche dell'Agenzia delle Entrate	Must have
5	Il sito deve soddisfare i criteri di accessibilità e usabilità descritti nella legge Stanca della normativa italiana sui siti web della PA	Must have

Tabella 2.3: Requisiti di dominio

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.

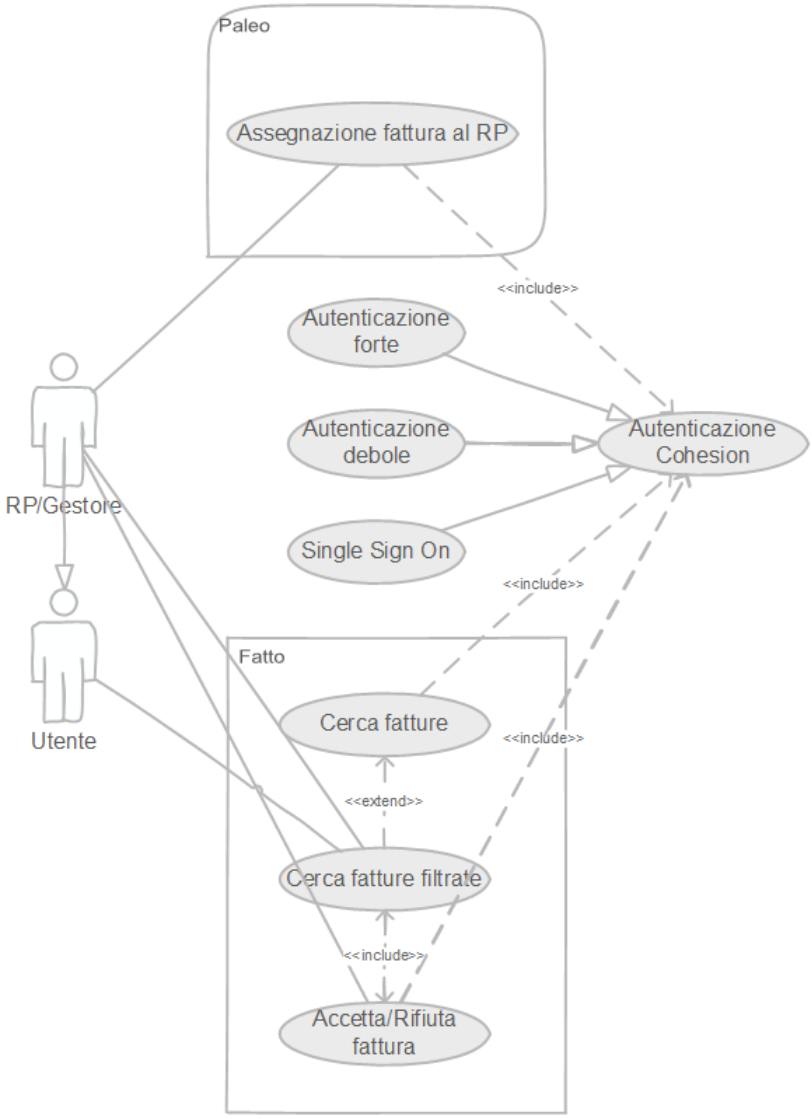

Figura 2.1: UML Use Case Diagram - accettazione

Nel diagramma in figura 2.1 abbiamo due attori, RP/Gestore e Utente, il secondo è la generalizzazione del primo, questo significa che tutti i casi d'uso

che valgono per il secondo valgono anche per il primo, ma non viceversa. I casi d'uso talvolta includono più passaggi al loro interno, in quei casi è indacata una freccia "include", come nel caso della "Ricerca fatture" che include "Autenticazione Cohesion". Alcuni casi d'uso sono estensione di altri, in altre parole sono una "specializzazione", un esempio è la "Ricerca fatture filtrata" che è una specializzazione della "Ricerca fatture". I riquadri indicano i diversi sistemi software e distinguono a quale di essi appartengono i vari casi d'uso.

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.

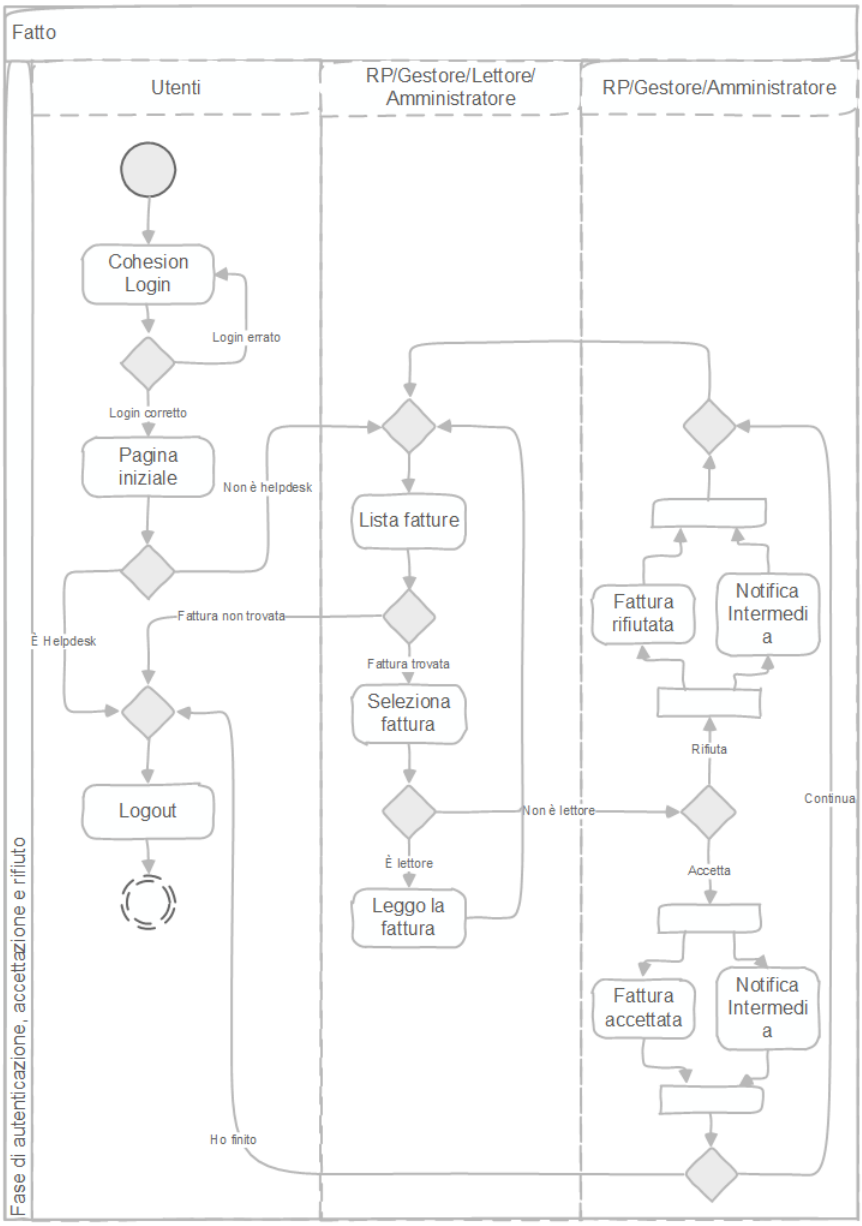

Figura 2.2: UML Activity Diagram - autenticazione e accettazione

Nella figura 2.2 vediamo descritta la stessa fase dello Use Case precedente,

cioè quella di autenticazione e accettazione/rifiuto, ma dal punto di vista delle attività da svolgere in sequenza per raggiungere un certo risultato (l'accettazione in questo caso). Il diagramma si legge a partire dal pallino in alto, detto nodo iniziale, seguono le azioni indicate dai rettangoli verdi. I rombi indicano le decisioni, cioè le diverse direzioni che può prendere il flusso a seguito di certe condizioni o scelte, ad esempio dopo l'azione "Cohesion login", se il login è errato si ritorna alla pagina di login, altrimenti si raggiunge la pagina iniziale. Altri elementi dell'activity diagram sono il "fork" e la "join", dove il primo sta ad indicare che diverse azioni possono essere eseguite in parallelo, mentre la seconda indica che per passare all'azione successiva devono essere completate tutte le azioni in ingresso. Nell'esempio il blocco che inizia con la fork e finisce con la join indica che ci sono due azioni: il salvataggio del nuovo stato accettata/rifiutata e l'invio della notifica a Intermedia; entrambe le azioni devono essere completate per proseguire, ma non importa in che ordine vengano eseguite.

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.

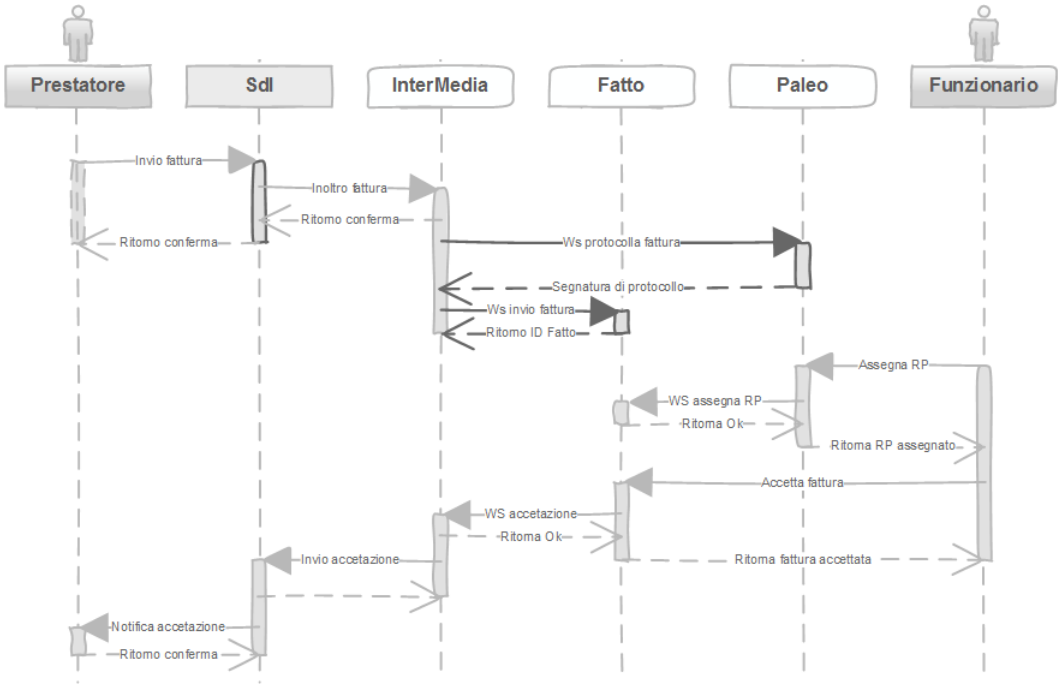

Figura 2.3: UML Sequence Diagram - ricezione e accettazione

Nella figura 2.3 vediamo descritto il flusso iniziale della fatturazione passiva che inizia con l'invio della fattura da parte del prestatore e termina con la ricezione di accettazione (in questo caso) della stessa da parte del prestatore. I componenti del diagramma sono il prestatore, il funzionario/committente e i vari sistemi software in comunicazione tra loro tramite Web Service (WS). Possiamo vedere lo schema come l'unione di due operazioni:

1. La ricezione
2. L'accettazione

Nella fase di ricezione il prestatore invia la fattura allo SdI, questo effettua i controlli formali e la inoltra senza modifiche al nodo di interscambio Intermedia, il quale effettua un controllo della firma e spacchetta il file in caso di lotto

poi invia la singola fattura, prima al protocollo, che restituisce la segnatura (codice univoco del protocollo), poi a Fatto (il sistema di fatturazione) insieme alla segnatura. Quando Intermedia riceve la fattura ritorna allo SdI un messaggio di avvenuta ricezione il quale viene a sua volta inoltrato al prestatore. Nella seconda fase il committente assegna un Responsabile del Procedimento (RP) alla fattura tramite Paleo, questo comporta l'assegnazione del RP anche in Fatto, poi il RP in questione effettua l'accettazione da Fatto che a sua volta richiama Intermedia. Se tutto va bene Intermedia risponde a Fatto che solo allora aggiorna lo stato della fattura ad "Accettata" e risponde al committente. Infine Intermedia richiama lo SdI che a sua volta avverte il prestatore dell'avvenuta accettazione.

2.6 Modello Entità-Relazione (E-R)

Il modello Entità-Relazione è un modello concettuale di dati e, come tale, fornisce una serie di costrutti atti a descrivere la realtà di interesse in una maniera facile da comprendere e che prescinde dai criteri di organizzazione dei dati nei calcolatori [**basidati**]. Tali costrutti vengono utilizzati per definire schemi che descrivono l'organizzazione e la struttura delle istanze. I costrutti principali del modello concettuale sono:

- Entità, rappresentano classi di oggetti che hanno proprietà comuni ed esistenza "autonoma" ai fini dell'applicazione di interesse. Una occorrenza di un'entità è un oggetto della classe che l'entità stessa rappresenta. In uno schema, ogni entità ha un nome che la identifica univocamente, e viene rap-presentata graficamente tramite un rettangolo con il nome dell'entità al suo interno.

- Relazioni (o associazioni), rappresentano legami logici, significativi per l'applicazione di interesse, tra due o più entità. Una occorrenza di relazione è un'ennupla (coppia nel caso più frequente di relazione binaria) costituita da occorrenze di entità, una per ciascuna delle entità coinvolte. Di norma viene rappresentata graficamente da un rombo contenente il nome dell'associazione. Il nome può essere un verbo in modo da fornire una direzione di lettura, oppure può essere un sostantivo in modo da non dare una direzione di lettura.

- Attributi, descrivono le proprietà elementari di entità o relazioni che sono di interesse ai fini dell'applicazione. Un attributo associa a ciascuna occorrenza di entità (o di relazione) un valore appartenente a un insieme detto dominio, che contiene i valori ammissibili per l'attributo. Può risultare comodo, a volte, raggruppare attributi di una medesima entità o relazione che presentano affinità nel loro significato o uso: l'insieme di attributi che si ottiene in questa maniera viene detto attributo composto.

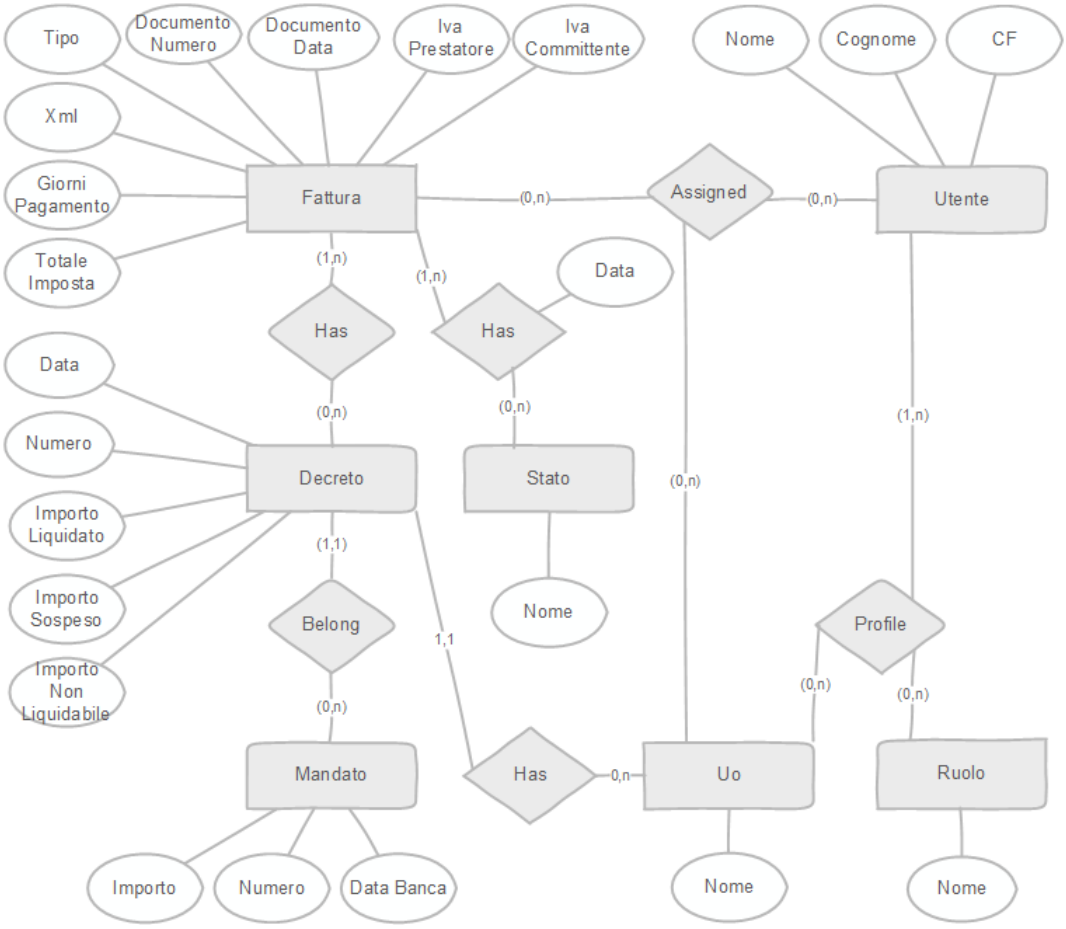

Figura 2.4: E/R Diagram

Nella Figura 2.4 possiamo distinguere le entità individuate da rettangoli, le relazioni rappresentate come rombi e gli attributi indicati come cerchi collegati alle entità o talvolta alle relazioni. L'entità "Fattura" ha un attributo chiamato "Xml" che contiene il tracciato serializzato della fattura EXtensible Markup Language (XML). Una Fattura può avere zero o più Decreti, mentre un Decreto può avere una o più Fatture (non zero), un discorso simile vale per i Mandati. Una Fattura può avere uno o più Stati, uno dei quali è sicuramente "Ricevuta"; la relazione in questo caso possiede un attributo "Data" perché gli stati sono incrementali: per vedere qual'è lo stato attuale di una Fattura si guarda semplicemente quello con la Data maggiore. Una relazione chiamata "Assigned" collega la Fattura con l'Utente e l'UO, questa assegnazione viene fatta da Paleo tramite il WS di assegnazione RP. Infine una relazione chiamata

"Profile" indica l'esistenza di un profilo Ruolo-Uo per un certo Utente.

Capitolo 3

Sviluppo

Lo sviluppo di un software web come questo richiede l'utilizzo di diverse tecnologie, metodologie e linguaggi di programmazione, anche se non esiste una concreta linea di demarcazione, possiamo distinguere tre specifiche aree di sviluppo:

- Database, comprende la traduzione iniziale del modello E/R in uno schema Structured Query Language (SQL) operando le opportune normalizzazioni, la creazione di query di ricerca, la creazione di viste e la gestione delle migrazioni dei dati ogniqualvolta una modifica riguarda i dati in produzione.
- Back-End, comprende tutto ciò che riguarda la mappatura Object-Relational Mapping (ORM) delle tabelle in classi, lo sviluppo della logica del software, ovvero cosa fare con i dati, come trattare quelli ricevuti, cosa restituire all'utente.
- 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.
- Web Service, l'insieme dei software che permettono la comunicazione e lo scambio dati tra la nostra applicazione web e gli altri programmi.

3.1 Sviluppo del Database

Prima di scrivere la prima riga di codice è necessario procedere con la definizione dello schema del database, partendo dal modello Entità Relazioni definito in fase di analisi. A questo punto è necessario scegliere un Database Management System (DBMS) che implementi il modello, nel nostro caso abbiamo scelto Microsoft SQL Server un DBMS relazionale che usa T-SQL come linguaggio di query. Una volta definito il linguaggio possiamo procedere alla codifica in SQL delle tabelle.

```
CREATE TABLE [dbo].[Fattura]
(
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Id_Stato] [tinyint] NOT NULL,
    [Id_Paleo] [int] NULL,
    [Id_TipoDocumento] [tinyint] NULL,
    [Id_CausaleRifiuto] [tinyint] NULL,
    [Sdi] [int] NULL,
    [SegnaturaProtocollo] [varchar](256) NULL,
    [RegistroNumero] [int] NULL,
    [RegistroData] [date] NULL,
    [CodiceUfficio] [varchar](6) NULL,
    [TrasmittenteIdFiscaleIvaPaese] [varchar](2) NULL,
    [TrasmittenteIdFiscaleIvaCodice] [varchar](28) NULL,
    [PrestatoreIdFiscaleIvaPaese] [varchar](2) NULL,
    [PrestatoreIdFiscaleIvaCodice] [varchar](28) NULL,
    [PrestatoreNome] [varchar](256) NULL,
    [CommittentePaese] [varchar](2) NULL,
    [CommittenteCodiceFiscale] [varchar](16) NULL,
    [CommittenteNome] [varchar](80) NULL,
    [Cig] [varchar](16) NULL,
    [Cup] [varchar](16) NULL,
    [DatiGeneraliDocumentoData] [date] NULL,
    [DatiGeneraliDocumentoNumero] [varchar](20) NULL,
    [Causale] [varchar](256) NULL,
    [ImportoTotale] [decimal](14, 2) NULL,
    [ImportoTotaleImponibile] [decimal](14, 2) NULL,
    [ImportoTotaleImposta] [decimal](14, 2) NULL,
    [IbanFattura] [varchar](34) NULL,
    [NomeFile] [varchar](128) NULL,
    [NoteAccettazioneRifiuto] [varchar](255) NULL,
    [NoteSospensione] [varchar](255) NULL,
    [MaxGiorniPagamento] [tinyint] NULL,
    [DataConsegnaSdi] [datetime] NULL,
    [DataPagamentoMax] [datetime] NULL,
    [Xml] [varchar](max) NULL,
    CONSTRAINT [PK_Fattura] PRIMARY KEY CLUSTERED
    (
        [Id] ASC
    )
    WITH (ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
)
ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

Figura 3.1: SQL per la creazione della tabella Fattura

```sql
CREATE TABLE [dbo].[Ruolo] (
    [Id] [tinyint] NOT NULL,
    [CodiceDescr] [varchar](16) NOT NULL,
    [Nome] [varchar](64) NULL,
    CONSTRAINT [PK_Ruolo_1] PRIMARY KEY CLUSTERED (
        [Id] ASC
    ) WITH (ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

CREATE TABLE [dbo].[Uo] (
    [Id] [smallint] NOT NULL,
    [Id_Padre] [smallint] NULL,
    [Codice] [varchar](32) NOT NULL,
    [CodiceDescr] [varchar](16) NULL,
    [Descrizione] [varchar](max) NULL,
    CONSTRAINT [PK_Uo] PRIMARY KEY CLUSTERED (
        [Id] ASC
    ) WITH (ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]


CREATE TABLE [dbo].[Utente] (
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Username] [varchar](64) NULL,
    [Password] [varchar](64) NULL,
    [DataPassword] [datetime] NULL,
    [Email] [varchar](256) NULL,
    [CodiceFiscale] [varchar](16) NULL,
    [Nome] [varchar](64) NULL,
    [Cognome] [varchar](64) NULL,
    CONSTRAINT [PK_Utente] PRIMARY KEY CLUSTERED (
        [Id] ASC
    ) WITH (ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

CREATE TABLE [dbo].[Utente_Uo_Ruolo] (
    [Id] [int] IDENTITY(352,1) NOT NULL,
    [Id_Utente] [int] NOT NULL,
    [Id_Uo] [smallint] NOT NULL,
    [Id_Ruolo] [tinyint] NOT NULL,
    CONSTRAINT [PK_UtenteUoRuolo] PRIMARY KEY CLUSTERED (
        [Id] ASC
    ) WITH (ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Figura 3.2: SQL per la creazione delle tabelle Utente, Ruolo e UO

3.2 Sviluppo del Back-End

Lo sviluppo di Back-End può essere realizzato sia con linguaggi di programmazione lato server come ASP, Java, PHP, sia facendo uso di frameworks come ASP.NET, Spring MVC, Laravel, Django, etc.

Un framework è una serie di librerie di codice utilizzabili in fase di linking con uno o più linguaggi di programmazione, spesso corredate da una serie di strumenti di supporto allo sviluppo del software, come ad esempio un Integrated Development Environment (IDE), un debugger o altri strumenti ideati per aumentare la velocità di sviluppo del prodotto finito.

Per realizzare il back-end, è stato scelto di utilizzare il .NET Framework, dato che la maggior parte dei software dell'Ente sono web application in ASP.NET Web Forms, ma si è scelta l'ultima versione del framework allora disponibile e la più recente tecnologia di sviluppo web, ASP.NET MVC.

3.2.1 .NET Framework

Il .NET Framework (si pronuncia dot net) è un framework sviluppato da Microsoft eseguibile principalemente su Microsoft Windows. Include una grande libreria di classi, interfacce e tipi di dato chiamata Framework Class Library (FCL) e garantisce interoperabilità tra i linguaggi di programmazione (ogni linguaggio può usare codice scritto in un altro linguaggio). I programmi scritti in .NET sono eseguiti in un ambiente software (invece che in un ambiente hardware), conosciuto come Common Language Runtime (CLR), una virtual machine che fornisce servizi di sicurezza, gestione della memoria e gestione delle eccezioni. FCL e CLR insieme compongono il .NET Framework.

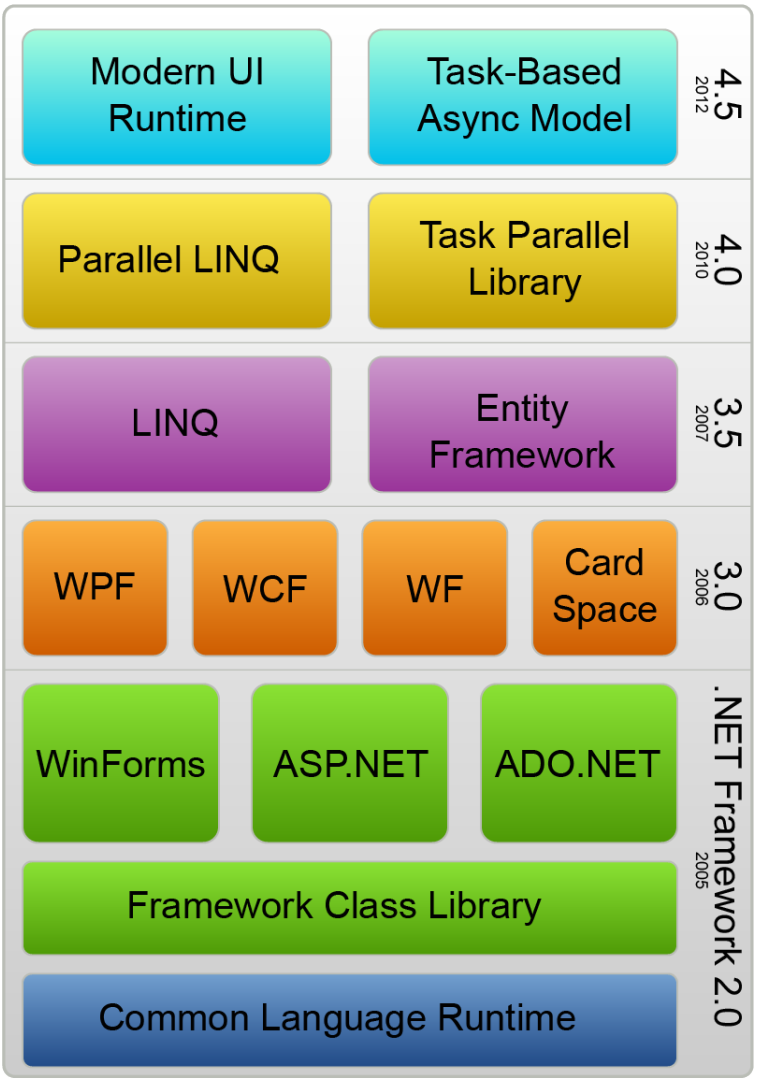

Figura 3.3: .NET Framework - Pila dei componenti

Oltre al CLR e FCL .NET possiede un insieme di tecnologie per lo sviluppo su diverse piattaforme, alcune di queste tecnologie sono state utilizzate nel nostro progetto, tra le quali:

- ASP.NET
- Entity Framework
- Web API

3.2.2 ASP.NET

ASP.NET è un insieme di tecnologie software per lo sviluppo di web application e web services, si appoggia sul .NET Framework e offre tre frameworks per la creazione di web application:

- ASP.NET Web Pages, pagine dinamiche dove il markup HTML e il codice sono insieme nello stesso file
- Web Forms, che offre una libreria di controlli che incapsulano markup HTML, script javascript e logiche lato server per garantire una gestione ad eventi simile a quella delle applicazioni desktop
- ASP.NET MVC, alta separazione delle logiche di controllo, da quelle di presentazione e dai dati grazie al paradigm MVC, tipico anche di altri framework moderni

ASP.NET MVC [**mvc**] fa uso dell'architettura multi-tier chiamata Model View Controller (MVC) che prevede la separazione del codice in 3 livelli:

- Model (Modello), gestisce l'accesso ai dati dell'applicazione
- View (Vista), gestisce la visualizzazione dei dati all'utente
- Controller (Controllore), gestisce le interazioni dell'utente

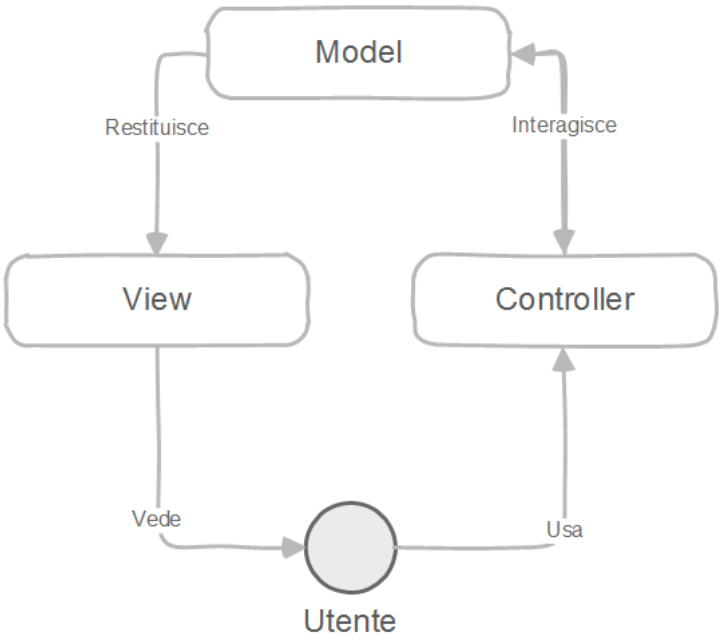

Figura 3.4: Architettura MVC

Il modo in cui questi 3 livelli interagiscono può cambiare da linguaggio a linguaggio, ma nel caso specifico di ASP.NET MVC funziona così:

1. l'utente invia una richiesta al browser (può essere una GET o una POST);
2. la richiesta viene intercettata dal web server (IIS nel nostro caso);
3. il web server fa gestire la richiesta al controller appropriato;
4. il controller la analizza e decide a quale specifico metodo farla gestire (nel nostro caso i metodi del controller si chiamano Action);
5. il controller può scegliere se lavorare o meno con il Model, in ogni caso alla fine del metodo ritornerà all'utente una View, cioè una pagina web di risposta.

Nel caso in cui il controller dovesse interfacciarsi con il database chiamerebbe il Model, il quale ha accesso diretto al database e colloquia con esso attraverso query SQL. Di seguito in Figura 3.5 vediamo un esempio di Action per la visualizzazione dei Dettagli: tra i parametri del metodo c'è l'ID, passato al

metodo tramite URL, e come valore di ritorno c'è la View specifica per Dettagli, a cui viene passato l'oggetto vmFattura.

```
1   using Fatto.Filters;
2   using Fatto.Helpers;
3   using Fatto.Models;
4   using Fatto.ViewModels;
5   using log4net;
6   using PagedList;
7   using System;
8   using System.Collections.Generic;
9   using System.Linq;
10  using System.Transactions;
11  using System.Web.Mvc;
12
13  namespace Fatto.Controllers
14  {
15      [MyAuthorize(Roles = "Rup, Uo, Lettore, Amministratore")]
16      [MyProfileFilter]
17      public class FatturaController : Controller
18      {
19          private FatturaElettronicaEntities db = new FatturaElettronicaEntities();
20
21          // GET: /Fattura/Dettagli/5
22          public ActionResult Dettagli(int id = 0)
23          {
24              // Cerco la Fattura
25              var fattura = db.ViewDettagliFattura.Find(id);
26
27              if (fattura == null)
28              {
29                  // Non trovata -> Ritorno 404
30                  return HttpNotFound();
31              }
32              else
33              {
34                  // Trovata -> Continuo
35
36                  // Carico il model nel viewmodel
37                  VmFattura vmFattura = new VmFattura(fattura);
38
39                  return View(vmFattura);
40              }
41          }
42
```

Figura 3.5: Fattura Controller - Action

A volte può essere necessario aggiungere alcuni controlli ricorrenti all'inizio di alcune Actions o sul Controller stesso (in modo che valgano per tutte le Action del Controller), come ad esempio un metodo che controlli se l'utente è autenticato, in quel caso è possibile creare dei Filters o utilizzare quelli di base forniti dal framework (come Authorize). Nel nostro caso è stato necessario creare un nuovo Filter che ereditasse Authorize, ma che gestisse l'autenticazione Cohesion, richiamando quindi il WS in caso di mancata autenticazione, come in Figura 3.6.

```
using Fatto.Helpers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace Fatto.Filters
{
    public class MyAuthorize : AuthorizeAttribute
    {
        protected override void HandleUnauthorizedRequest(AuthorizationContext filterContext)
        {
            var session = System.Web.HttpContext.Current.Session;

            if (session["ReturnUrl"] == null)
            {
                session["ReturnUrl"] = filterContext.HttpContext.Request.Url.AbsoluteUri;
            }

            // Ridireziono alla pagina di login con Cohesion
            filterContext.Result =
                new RedirectToRouteResult(
                    new RouteValueDictionary
                    {
                        { "action", "Login" },
                        { "controller", "Account" }
                    });
        }
    }
}
```

Figura 3.6: MyAuthorize Filter

3.2.3 Entity Framework

La generazione delle classi del Model e i metodi per accedere al database possono essere gestiti da Entity Framework (EF). Entity Framework è un framework open-source ORM per ADO.NET, una parte del .NET Framework: il compito svolto dagli ORM è quello di nasconderci il funzionamento di un database relazionale, permettendoci di pensare al nostro modello dei dati come ad un insieme di oggetti. Mediante l'uso di EF è possibile interrogare il database facendo uso di Language INtegrated Query (LINQ), componente del .NET Framework costituito interamente da "extension methods", ovvero metodi che consentono di estendere le funzionalità di determinati tipi senza dover appositamente definire specifici tipi derivati; esso aggiunge la possibilità di effettuare query su collezioni di dati, con una sintassi simile alla sintassi di SQL oppure simile a quella della programmazione ad oggetti (Lamda Expression). Entity Framework mette a disposizione tre differenti modalità per interagire con i dati:

- Database first, è previsto nel caso in cui il database sia già disponibile, verrà generato in automatico il Data Model che consiste nell'insieme di classi e proprietà che corrispondono alle tabelle ed alle colonne del database. La corrispondenza tra il Data Model così generato e lo schema del database è descritto in formato XML in un file di estensione .edmx. Questo approccio permette modifiche dirette sul database, seguite poi da un'aggiornamento del file edmx.

- Model first, consiste nell'uso di uno strumento grafico per la creazione di Data Models chiamato Entity Framework Designer. Analogamente al caso precedente, la corrispondenza tra il database ed il Data Model è descritta mediante un file edmx. Questo approccio non permette modifiche né lato codice né lato database, ma solo tramite lo strumento grafico.

- Code first, è un approccio applicabile sia nel caso in cui si disponga già del database, sia in caso contrario. Attraverso la creazione di classi si definiscono le tabelle e loro relazioni, non necessita di un file .edmx e permette di eseguire delle migrazioni di dati da una versione all'altra del database.

Nel nostro progetto abbiamo utilizzato l'approccio Database-first.

3.3 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

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.

Nel web, l'usabilità 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 pagina web o dell'applicazione web;
- 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.

L'**accessibilità** è un aspetto di fontamentale importanza, specialmente per un gestionale web di una PA; il suo scopo è garantire l'accesso alle risorse (prodotti e servizi) da parte di chiunque, siano essi soggetti con disabilità, con scarse competenze informatiche o con dispositivi diversi. I criteri dell'accessibilità sono quelli delle linee guida del World Wide Web Consortium (W3C) che attraverso una sua sezione denominata Web Accessibility Initiative (WAI) [**w3c-wai**] ha definito i linguaggi e le procedure standard per rendere il Web uno strumento realmente democratico e universale. Queste direttive sono state percepite in Italia con la Legge "Stanca" (Legge 4/2004) che sancisce obblighi e sanzioni per la PA e le aziende appaltate. In generale i requisiti di accessibilità contenuti nella Legge Stanca e successive modificazioni, contenuti anche in un documento online sul sito del governo [**agid-access**], 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.

3.3.1 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) e JavaScript.

HTML [**w3c-html**] è un linguaggio di markup per le pagine web i cui standard sono definiti dal W3C e serve per descrivere il contenuto di una pagina web, sia testuale che multimediale, attraverso dei tag. Sebbene il linguaggio sia in grado di definire anche regole di formattazione come la centratura del testo o la dimensione dei caratteri, il suo scopo primario è quello di descrivere i contenuti e il loro significato semantico, attraverso l'uso dei tag appropriati (ad esempio <p>per i paragrafi, <h1>per un titolo di primo livello, <address>per informazioni relative ad un indirizzo, etc.). Attualmente la versione più recente delle specifiche è chiamata HTML 5 e comporta rispetto alle precedenti versioni l'aggiunta di alcuni tag per gli elementi multimediali, il miglioramento i controlli di input per le form, il controllo della geolocalizzazione e l'intruduzione dello Web Storage come alternativa ai cookies.

CSS [**w3c-css**] è il linguaggio usato per definire il layout (impaginazione), la formattazione del testo e l'aspetto grafico in generale; può essere definito sia all'interno della pagina HTML che all'esterno come file a sé stante, che è generalamente la scelta migliore. Nel nostro caso sono state usate alcune delle proprietà più recenti del linguaggio come la box-shadow o la border-radius appartenenti alle specifiche CSS 3. Per mantenere la compatibilità tra i differenti browser (e nei limiti del possibile anche con le versioni più vecchie degli stessi) sono state applicate ad ogni proprietà CSS 3 delle tecniche di

cross-browsing, cioè la riscrittura della stessa proprietà in modi differenti, vedi figure 3.7 e 3.8.

```
div.shadow
{
    /* CSS 3 */
    box-shadow: 3px 3px 4px #44444;

    /* Mozilla Firefox */
    -moz-box-shadow: 3px 3px 4px #44444;

    /* Apple Safari, Google Chrome */
    -webkit-box-shadow: 3px 3px 4px #44444;

    /* Microsoft Internet Explorer */
    -ms-filter:
        "progid:DXImageTransform.Microsoft.Shadow(Strength=4, Direction=135, Color='#444444')";
    filter:
        progid:DXImageTransform.Microsoft.Shadow(Strength=4, Direction=135, Color='#444444');
}
```

Figura 3.7: CSS 3 text-shadow cross-browser

```
.round
{
    /* Safari 3-4, iOS 1-3.2, Android 1.6- */
    -webkit-border-radius: 11px;

    /* Firefox 1-3.6 */
    -moz-border-radius: 11px;

    /* Opera 10.5, IE 9, Safari 5, Chrome, Firefox 4, iOS 4, Android 2.1+ */
    border-radius: 11px;
}
```

Figura 3.8: CSS 3 border-radius cross-browser

JavaScript [**javascript**] è un linguaggio di scripting adatto per creare 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. Anche se con il javascript è possibile creare applicazioni vere e proprie si deve sempre tener conto che è un linguaggio accessorio, ovvero c'è la possibilità per l'utente di disabilitarlo, specialmente nel caso si abbia delle disabilità o dei problemi di visualizzazione. Pertanto nessuna funzionalità deve contare unicamente su javascript per funzionare, ma il suo uso è comunque consigliato per aumentare l'esperienza utente (quegli obiettivi dell'usabilità esposti sopra) ad esempio prevedendo dei messaggi di errore nel caso si inseriscano dei valori errati nei campi, o un calendario all'immissione di un campo data, o anche per sostituire delle proprietà del CSS 3 nel caso il browser dell'utente non la supporti (i cosiddetti polyfill).

jQuery [**jquery**] è una libreria Javascript molto diffusa il cui scopo è 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", anch'essa è stata inclusa nella nostra applicazione web e gli script personalizzati fanno tutti riferimento alla sua sintassi.

Asynchronous JavaScript and XML (AJAX) è un'altra tecnologia molto importante per migliorare l'esperienza utente, perché permette di aggiornare il contenuto di una pagina dinamicamente senza effettuare il "refresh" e quindi caricando solo le informazioni necessarie, anziché tutta la pagina. Si tratta, più che di un nuovo linguaggio, di un insieme di tecniche che fanno uso delle tecnologie sopra descritte, quindi JavaScript per inviare le richieste asincrone e XML o JSON per ricevere le risposte dal server sottoforma di oggetti. Questa tecnica è stata utilizzata nella nostra applicazione web nella fase di autenticazione post-Cohesion, dove viene richiesto all'utente di scegliere un ruolo da una tendina e automaticamente sulla base di questa scelta viene popolata la tendina delle Unità Organizzativa (UO) riferite a quel ruolo. Un altro utilizzo è nella pagina di inserimento di un nuovo utente, dove a seguito della digitazione di almeno 3 caratteri in una casella di testo, compare un menù di autocomplete che suggerisce i possibili nomi degli utenti cercati.

Razor [**razor**] infine, è un linguaggio di programmazione lato server, specifico del framework ASP.NET MVC, ma è stato incluso in questa sezione perché serve a descrivere, insieme all'HTML, il contenuto di una View. Una View che contiene codice Razor sarà un file di tipo .cshtml (nel caso il linguaggio lato server utilizzato sia il C#). Questo linguaggio può utilizzare molte delle classi del framework, semplicemente includendo il namespace nella pagina, ma il suo scopo principale è quello di posizionare gli elementi del Model nella View, passati dal Controller, eventualmente anche iterando una Collection affinché venga stampata a schermo una serie di elementi per ogni elemento della Collection, come in Figura 3.9.

```
64
65  <tbody>
66      @foreach (var item in Model)
67      {
68      <tr>
69          <td data-title="Prestatore">
70              <div>@Html.DisplayFor(modelItem => item.PrestatoreIva)</div>
71              <div>@Html.DisplayFor(modelItem => item.PrestatoreNome)</div>
72          </td>
73          <td data-title="Dati Generali - Data">@Html.DisplayFor(modelItem => item.DatiGeneraliDocumentoData.ToString("dd/MM/yyyy"))</td>
74          <td data-title="Dati Generali - Numero">@Html.DisplayFor(modelItem => item.DatiGeneraliDocumentoNumero)</td>
75          <td data-title="Causale" class="description">@Html.DisplayFor(modelItem => item.Causale)</td>
76          <td data-title="Stato Attuale">@Html.DisplayFor(modelItem => item.StatoAttuale)</td>
77          <td data-title="RP" class="description">@Html.DisplayFor(modelItem => item.Rup)</td>
78          <td data-title="Scadenze" class="scadenze">
79              @if (!(item.Rifiutata == true || item.Pagata == true || item.Sospesa == true))
80              {
81                  if (item.Accettata == true)
82                  {
83                      TimeSpan t = TimeSpan.FromSeconds((int)item.ScadenzaPagamento);
84                      <div>
85                          <span>@t.Days</span> giorni per il Pagamento
86                      </div>
87                  }
88                  else
89                  {
90                      TimeSpan t = TimeSpan.FromSeconds((int)item.ScadenzaRifiuto);
91                      <div>
92                          <span>@t.Days</span> giorni per il Rifiuto
93                      </div>
94                  }
95              }
96          </td>
97          <td>
98              @Html.ActionLink("Documento", "Documento", new { id = item.Id }) |
99              @Html.ActionLink("Dettagli", "Dettagli", new { id = item.Id })
100         </td>
101     </tr>
102     }
103 </tbody>
```

Figura 3.9: Esempio di codice Razor per la stampa degli elementi di una collection come righe di una tabella HTML

3.4 Sviluppo dei Web Service

Lo scambio dati tra diversi programmi può avvenire sostanzialmente in diversi modi:

- Salvando dati in un database comune
- Salvando dati sottoforma di file in una locazione di memoria comune
- Scambiandosi i dati tra applicazione attraverso web service

La soluzione più utilizzata, e nel nostro caso l'unica percorribile, è quella di esporre web service o utilizzare quelli messi a disposizione dalle altre applicazioni web.

3.4.1 REST e SOAP

Un web service è un metodo di comunicazione tra due applicazioni o dispositivi elettronici in rete. I web service possono essere di due tipi:

- Simple Object Access Protocol (SOAP)
- Representational State Transfer (REST)

SOAP [**soap**] definisce un protocollo (insieme di regole) tra le due applicazioni per la comunicazione mediante lo scambio di messaggi XML. Tale protocollo è descritto da un file chiamato Web Services Description Language (WSDL) che descrive i tipi di dato e i metodi che un client può richiamare per utilizzare il web service. SOAP utilizza HyperText Transfer Protocol (HTTP) come protocollo per lo scambio di messaggi, ma non è limitato né vincolato ad esso, dal momento che può benissimo usare altri protocolli, come Simple Mail Transfer Protocol (SMTP).

REST descrive un insieme di principi architetturali con i quali i dati possono essere trasmessi attraverso un'interfaccia standard (come HTTP). Alla base di un servizio RESTful c'è il concetto di risorsa, cioè un oggetto o una collection, a cui il client può accedere tramite un Uniform Resource Identifier (URI). I metodi con i quali il client può accedere sono quelli del protocollo HTTP (GET, POST, PUT, DELETE e HEAD) e il content-type è definito da chi espone il web service, tipicamente JavaScript Object Notation (JSON) o XML.

3.4.2 Web API 2

ASP.NET Web API [**web-api**] è un framework che rende semplice la costruzione di web service HTTP in grado di raggiungere un gran numero di client, tra i quali browser e dispositivi mobili. È la piattaforma ideale per creare applicazioni REST in .NET, in quanto utilizza lo stesso modello architetturale di ASP.NET MVC. Per creare un nuovo Web Service bisogna creare un controller riferito a una specifica risorsa, ad esempio la fattura, e creare la Action corrispondente al metodo con il quale si vuole consentire l'accesso alla risorsa, ad esempio GET. Come parametri del metodo Action si possono indicare parametri singoli, come ad esempio l'ID che costituirà l'URI della richiesta, oppure

oggetti che andranno a comporre il payload (tipicamente di una richiesta POST o PUT). Attraverso i filter è possibile gestire delle politiche di sicurezza come la Basic Authentication. La versione utilizzata per i web service del nostro software è chiamata Web API 2. Di seguito in Figura 3.10 vediamo parte del WS chiamato da Intermedia per aggiornare lo stato fattura quando arriva dallo Sistema di Interscambio (SdI) la notifica di accettazione per decorrenza termini oppure la notifica di scarto.

```
11  namespace Fatto.Controllers.Api
12  {
13      public class StatoFatturaController : ApiController
14      {
15          private FatturaElettronicaEntities db = new FatturaElettronicaEntities();
16
17          // WS 9     PUT api/StatoFattura?protocollo=0000697|23/12/2014|XYZ&numfattura=123
18          [HttpPut]
19          public HttpResponseMessage PutStatoFatturaExtra(
20              [FromUri]string protocollo,
21              string numfattura,
22              [FromBody]InseritoreStato sf)
23          {
24              // Cerco gli stati fattura validi
25              var sf_adt = db.Stato.Where(x => x.Codice.Equals("ADT")).SingleOrDefault();
26              var sf_isc = db.Stato.Where(x => x.Codice.Equals("ADT")).SingleOrDefault();
27
28              if ((sf_adt == null) || (sf_isc == null))
29              {
30                  // 500 Stati non trovati
31                  return Request.CreateErrorResponse(
32                      HttpStatusCode.InternalServerError,
33                      "Stati di aggiornamento non trovati.");
34              }
35              else if (sf.IdStato != sf_adt.Id && sf.IdStato != sf_isc.Id)
36              {
37                  // 400 Stato non valido
38                  return Request.CreateResponse(
39                      HttpStatusCode.BadRequest,
40                      "Lo Stato scelto non è valido.");
41              }
42              else
43              {
44                  // Cerco le fatture
45                  var fatture =
46                      db.Fattura
47                      .Where(f =>
48                          f.SegnaturaProtocollo == protocollo &&
49                          f.DatiGeneraliDocumentoNumero == numfattura)
50                      .ToList();
51
```

Figura 3.10: StatoFattura API Controller - PUT Stato

Capitolo 4

Lavori simili

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** [**hub**] svolge appunto questo ruolo, esso prevede un sito per l'adesione al sistema da parte delle Pubbliche amministrazioni (PPAA) 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** [**notier**], il quale una volta ricevuta la fattura dallo SdI prima la converte nel formato europeo PEPPOL [**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 tra SdI e Ente, si occupa anche della comunicazione con la PCC. In Regione Marche il ruolo di intermediario svolto da Intermedia [**intermedia**] è quello di mettere in comunicazione lo SdI, 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 PCC 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 SdI, 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 PA.

I futuri sviluppi che interesseranno il sistema di fatturazione saranno legati al prossimo rilascio da parte della PCC 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.

Bibliografia