Sviluppo della gestione delle note di credito in un programma di fatturazione

Davide Fontana

20 luglio 2016

Indice

Capitolo 1

Introduzione

Questa relazione riguarda il tirocinio da me effettuato presso l' azienda Consoft Informatica S.r.l tra Febbraio e Aprile 2016 nella sede di Padova. Lo stage è cominciato con una prima parte di formazione sulle tecnologie usate all'interno dell'azienda tra cui Java J2EE, SQL, Hibernate e Spring MVC. In seguito, dopo la formazione, è seguita una parte pratica con la progettazione e la successiva implementazione della gestione delle note di credito di un programma di fatturazione online che, attualmente, è ancora in fase di sviluppo all' interno dell' azienda.

Capitolo 2

Azienda Ospitante

L' azienda in cui è stato svolto il tirocinio è la Consoft Informatica S.r.l operante nel settore Information & Communication Tecnology. Come riportato sul sito ufficiale [1], l'azienda è una società giovane, dinamica e innovativa la quale mette a disposizione competenze, servizi e strutture professionali tecnologicamente all'avanguardia a medie e grandi imprese al fine di potenziarne al massimo la competitività sul mercato. I professionisti che permettono a Consoft Informatica di perseguire il proprio obiettivo sono altamente qualificati e grazie ad un costante investimento nella formazione attraverso un addestramento legato alla metodologia del corpo docenti e all'utilizzo di strumenti software avanzati, sono in grado di rispondere alle esigenze del cliente garantendo affidabilità, convenienza ed efficienza. Consoft Informatica opera con i suoi oltre 150 dipendenti su tutto il territorio nazionale e ha 3 sedi ubicate a Padova, Firenze e infine Roma.

Capitolo 3

Tecnologie utilizzate

Durante le prime due settimane sono stato inserito all' interno di un corso tenuto dalla mia tutor aziendale Giulia Paoli in cui ho potuto apprendere la programmazione Java per web insieme a framework e librerie indispensabili per poter operare su progetti web di grandi dimensioni.

3.1 Java per il Web

Il primo passo è stato quello di imparare a creare un applicativo web in Java. Questo applicativo doveva chiedere in un form il nome della persona che accedeva alla pagina web. Appena l' utente scriveva il nome e premeva il bottone `submit` si veniva ridirezionati in un' altra pagina in cui veniva salutato il nuovo utente chiamandolo con il nome precedentemente inserito. Un applicativo web in Java si compone di essenzialmente di due parti:

- front-end: Corrisponde a ciò che vede l' utente essenzialmente pagine web. In questo caso non abbiamo pagine `.html` bensì pagine `.jsp` (Java Server Pages) ovvero pagine html che contengono anche codice scritto in Java. Per inserire codice scritto in Java in una pagina `jsp` si usano dei tag specifici ovvero `<% %>`. Per poter stampare invece una variabile all' interno del codice `html` si usa il tag `<%= %>`. Infine il tag `<%@ %>` viene usato per dare delle direttive al server che deve eseguire la pagina `jsp` oppure ad esempio per importare delle librerie all' interno della pagina. Un esempio lo possiamo vedere nel pezzo di codice sottostante.

  Esempio di tag JSP

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
    <body>
        <% String s = "Titolo"; %>
        <h1><%= s %></h1>
    </body>
</html>
```

- back-end: Corrisponde alla parte logica ed è composta da file Java come in un qualsiasi progetto Java standard.

La parte front-end di richiesta del nome è piuttosto semplice infatti basta scrivere un form html.

Form di inserimento dati.

```
<html>
    <body>
    <form method="get" action="PrimaServlet">
        <label>Inserisci il Nome:
        <input type="text" name="testo"/></label>
        <input type="submit"/>
    </form>
    </body>
</html>
```

dove method indica il tipo di richiesta Http (nel nostro caso la get) mentre la action indica dove inviare i dati appena inseriti. Per poter ricevere i dati si utilizza una classe Java particolare chiamata `Servlet`. La `Servlet` è una classe Java che estende la classe HttpServlet e implementa dei metodi specifici che servono per intercettare le tipologie di richieste http. Il codice sottostante è l' implementazione dell' esercizio.

Codice della Servlet.

```
package it.consoft;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class PrimaServlet
 */
@WebServlet("/PrimaServlet")
public class PrimaServlet extends HttpServlet {

        private static final long serialVersionUID = -11183296668082578L;

        /**
         * @see HttpServlet#HttpServlet()
         */
        public PrimaServlet() {
                super();
        }

        /**
         * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
         *      response)
         */
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
                request.getRequestDispatcher("pages/arrivo.jsp").forward(request, response);
        }

        /**
```

```
         * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
         *      response)
         */
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
                doGet(request, response);
        }
}
```

Come possiamo vedere sopra la dichiarazione della classe abbiamo una notazione la quale indica a quale azione del form essa risponde. Nel metodo `doGet` viene fatta attraverso l'oggetto `response` una ridirezione alla pagina. Infatti prima attraverso il metodo `getRequestDispatcher` si richiede la risorsa e poi con il metodo `redirect` si procede al redirect vero e proprio sulla risorsa selezionata. Notare che in questo caso vengono passati come parametri sia l'oggetto `request` che `response` della chiamata precedente. Infine quindi abbiamo la pagina `arrivo.jsp` in cui arriva la ridirezione. Qui sotto il codice della pagina.

Pagina di saluto del nuovo utente

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<% String s = request.getParameter("testo");%>
        <h1>Ciao <%= s  %></h1>
</body>
</html>
```

Possiamo notare come nella pagina jsp è presente un oggetto chiamato `request` di tipo `HttpServletRequest` senza che esso sia stato dichiarato. Questo infatti è possibile in quanto per ogni pagina .jsp sono definite alcune variabili implicite tra cui le variabili `request` e `response` esattamente dello stesso tipo della `Servlet`. Per visualizzare il nome della persona inserita utilizziamo il metodo `getParameter` in cui all' interno deve essere indicato il nome del parametro di cui vogliamo prendere il contenuto.

3.2 Hibernate

Successivamente abbiamo imparato ad accedere ad un database tramite Hibernate. Hibernate è un framework ORM (Object Relation Mapping) che mappa le tabelle di un database relazionale in oggetti Java. Questo framework rispetto ad accedere al database attraverso le normali API di Java offre diversi vantaggi.

- Non bisogna scrivere codice SQL in quanto Hibernate ha delle funzioni specifiche per inserire, modificare e restituire dati.
- Le colonne restituite da una funzione sono una lista di oggetti contenitore e quindi possono essere trattati con maggiore semplicità.

- Le funzioni di Hibernate sono trasparenti rispetto al tipo di DBMS usato. Infatti Hibernate compone automaticamente le query.

Ovviamente però tutte queste cose non possono essere fatte automaticamente. È necessario far sapere ad Hibernate a quale database si deve connettere e a quali sono le tabelle su cui deve mappare il contenuto tutto questo seguendo regole ben precise.

3.3 Spring

Capitolo 4

Progetto delle Note di Credito

Successivamente mi è stato chiesto di progettare e implementare la gestione delle note di credito di un programma di fatturazione che attualmente è in sviluppo presso Consoft Informatica.

4.1 Analisi dei Requisiti

La nota di credito viene applicata ad una fattura e ha due finalità:

1. Correttiva Parziale: Serve per correggere l'importo di una fattura nel caso in cui sia presente un errore per eccesso dell' imponibile. In questo caso l'importo della detrazione non è mai uguale o superiore al totale attuale della fattura.

2. Correttiva Totale: L'importo della detrazione è uguale a quello totale della fattura. Questo utilizzo è meno frequente e serve per annullare una fattura nel caso in cui non si riesca a risalire all' importo da correggere.

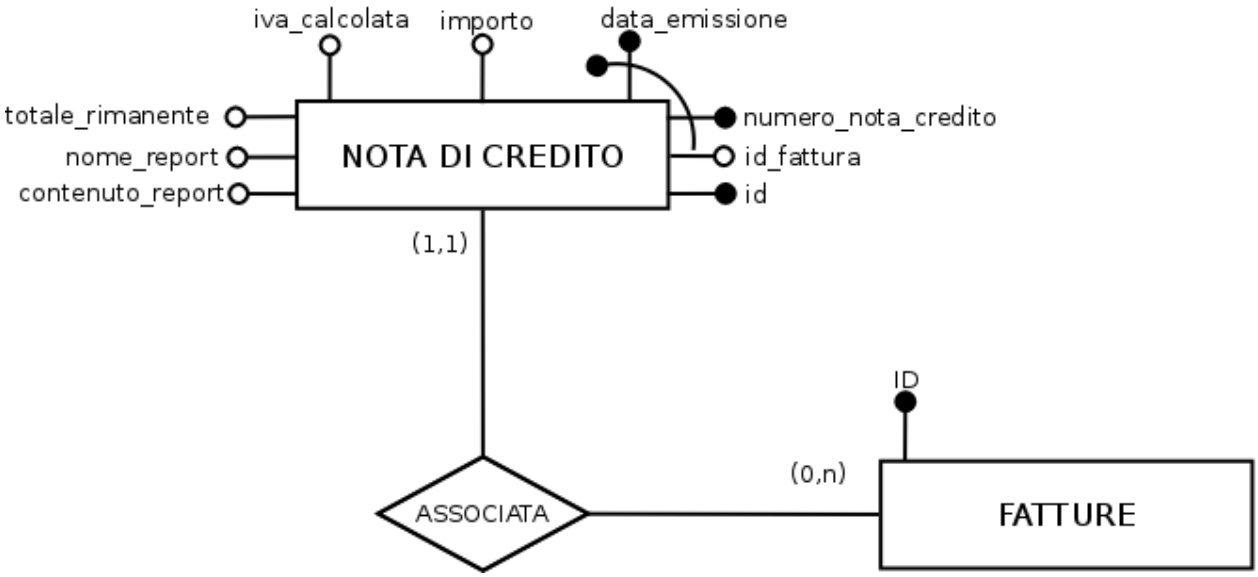
Ogni nota di credito è relativa ad un' unica fattura ma una fattura può avere più note di credito. E' composta dalle informazioni della azienda e il numero della fattura a cui questa nota di credito fa riferimento. Il numero della fattura è incrementale solo rispetto all' anno in corso e non alla fattura in cui la nota di credito è presente. La grafica deve essere in linea con quella attuale e deve consentire per ogni fattura (scaduta e non) di aggiungere note di credito e poter vedere, modificare, cancellare e stampare i report di quelle attualmente associate. La visualizzazione delle note di credito deve essere per data partendo dalla più recente facendo in modo che solo l'ultima nota credito sia modificabile/cancellabile mentre le altre devono avere solo la possibilità di essere visualizzate/stampate. Nel caso in cui l' ultima nota di credito venisse cancellata bisogna rendere disponibile la modifica di quella che c'era precedentemente. Infine l'imponibile nella lista delle fatture bisogna che sia compreso delle detrazioni rendendo poi impossibile la modifica della fattura stessa essendo che quindi la fattura è già stata emessa.

4.2 Database

Data quindi l' analisi dei requisiti si è cominciato a progettare la base di dati per salvare le informazioni. Qui sotto sono descritte tutte le fasi di progettazione.

4.2.1 Schema Concettuale

Essendo che le note di credito erano relative alle fatture si è deciso di creare una sola entità come mostrato nello schema sottostante.



Sono state attuate le seguenti scelte sugli attributi:

1. L' attributo `totale_rimanente` anche se attributo derivato viene comunque messo per velocizzare le operazioni di visualizzazione delle fatture in modo che non si debba fare ogni volta i calcoli dell' imponibile con le note di credito.

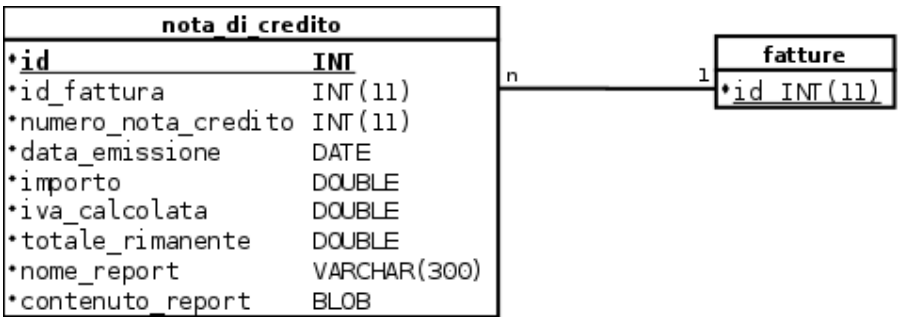
2. Sono presenti due chiavi, la prima, chiave primaria, è un `id` numerico auto incrementante mentre la seconda, chiave canditata, è data dalla coppia `numero_nota_credito` e `data_emissione`. L'attributo `id_fattura` è una chiave esterna ed è associata con l'attributo `id` che è chiave dell' entità fattura.

3. Abbiamo due attributi, `nome_report` e `contenuto_report`, che contegono il nome e il contenuto del file `.pdf` della nota di credito appena generata.

4. Sono stati inseriti inoltre solo l'importo e l' IVA calcolata in quanto la percentuale dell' IVA utilizzata e l'importo totale della nota di credito possono essere ricavate dagli attributi importo e IVA calcolata.

4.2.2 Schema Logico

Si è quindi passato poi allo schema logico ricavato da quello concettuale.



Come numero di cifre intere è stato scelto undici per rimanere in linea con il numero di cifre dell' id della tabella `fattura`

4.2.3 Implementazione Fisica

Il codice della tabella in SQL è il seguente

Codice SQL della tabella

```
te table nota_di_credito(
    id bigint auto_increment check(id > 0),
    id_fattura int(11) not null, -- id della tabella fatture
    numero_nota_credito int(11) not null check(numero_nota_credito > 0),
    data_emissione date not null check(anno >= 0), -- anno di emissione della fattura
    importo double not null check(importo >= 0),
    iva_calcolata double not null check(iva_calcolata >= 0),
    totale_rimanente double not null check(totale_rimanente>= 0),
    nome_report varchar(300) not null,
    contenuto_report blob not null,
    unique (numero_nota_credito, data_emissione),
    primary key(id),
    foreign key (id_fattura) references fatture(id)
);
```

Per tutti i campi è stato deciso di renderli tutti non nulli perché alla generazione della nota di credito tutti i campi devono essere compilati.

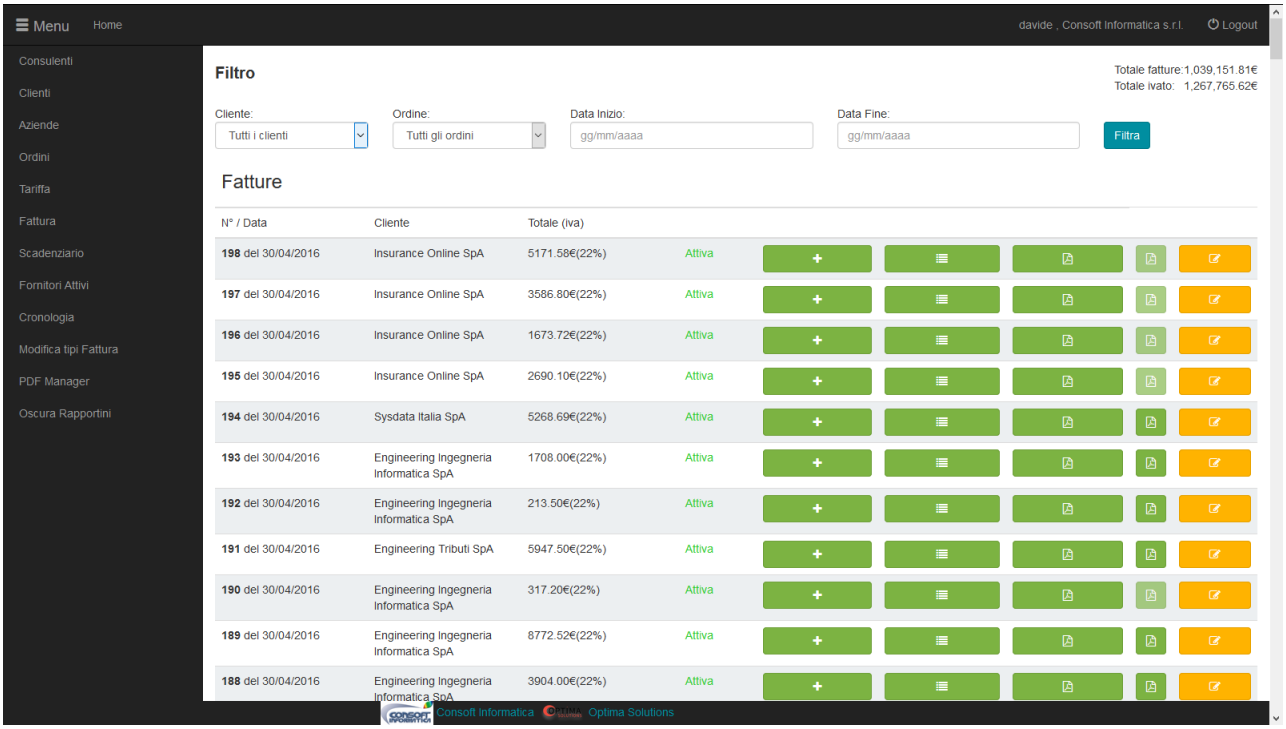
Capitolo 5

Implementazione

Si è passati poi alla scrittura del codice per integrare la nuova funzionalità nell' applicativo. Per questa parte sono stato aiutato da due sviluppatori interni all' azienda di nome Mauro Veronese della sede di Padova e Rocco Spenza della sede di Roma.

5.1 front-end

Per la parte front-end abbiamo deciso come prima cosa di integrare l' inserimento delle note di credito all' interno dell' interfaccia di inserimento fatture.



I primi due bottoni verdi sono infatti per l' inserimento e la visualizzazione del

5.2 back-end

Bibliografia

[1] Sito ufficiale Consoft Informatica
`http://www.consoftinformatica.it`

[2] Descrizione delle pagine jsp
`https://it.wikipedia.org/wiki/JavaServer_Pages`