Università degli Studi di Modena e Reggio Emilia

Dipartimento di Ingegneria “Enzo Ferrari”

Corso di Laurea in Ingegneria Informatica

Sito Web per l’ importazione dei contratti da un CRM ad un ERP nel caso di Siti b&t

Relatore:

Sonia Bergamaschi

Laureando:

Khadija Najem

Anno Accademico 2019-2020

1.INTRODUZIONE

Implementare un sistema di importazione dei contratti di clienti da CRM (Lyra) a ERP (Baan) mettendo a disposizione dell'operatore un'interfaccia di gestione che permetta manutenzione e modifica dei dati.

Il sistema sarà utilizzato dai *Controlling Commesse* che si occupano del Controllo di Gestione. Ogni volta che un Controlling Commesse doveva trasferire un contratto da CRM a ERP lo richiedeva all'IT Manager che interveniva direttamente sul database. Il fine ultimo del progetto è quello di creare un sistema automatico che si occupi di questa operazione.Il risultato sarà un sito web attraverso il quale si potranno trasferire i contratti da Lyra a Baan.

Le richieste del Controlling Commesse sono :

1. Avere a disposizione un'interfaccia iniziale contenente la lista dei contratti disponibili per essere trasferiti.

2. Per ogni contratto visualizzare una pagina di dettaglio in cui sono presenti le righe dell'ordine. L' utente può modificare o aggiungere nuove righe al contratto.
In questa schermata deve essere presente il tasto trasferimento.

Cosa vuol dire trasferire i contratti ? In sostanza consiste nel creare quattro file con nomi univoci per ogni contratto, questi file verranno riconosciuti e letti dal sistema Baan. Quando parliamo di contratto intendiamo il legame che l'azienda ha con il cliente in particolare ognuno riporterà varie informazioni sui clienti e i rispettivi ordini. Ogni file creato avrà un contenuto speciale che creeremo attraverso una stored procedure.

Oltre al trasferimento del contratto, sarà possibile la modifica delle righe del contratto che presenteremo sotto forma di tabella creata attraverso la libreria jQgrid. Ogni cella della tabella sarà modificabile in modo che quella particolare informazione venga modificata anche nei corrispondenti file del trasferimento.

Il seguente elaborato ha lo scopo di mostrare il lavoro svolto e di comprendere le varie tecnologie utilizzate per la realizzazione del sito Web.

Il secondo capitolo si occupa della descrizione degli ERP e CRM in particolare della differenza tra i due in modo da comprendere il perché molte aziende utilizzano questi strumenti per facilitare il compito degli utenti.

Il terzo capitolo tratta dei supporti utilizzati per poter realizzare il progetto.Partiamo con una breve introduzione a PhpStorm , procedendo ad una descrizione esaustiva del Web, ossia di tutte le relazioni necessarie per poter creare un sito Web, quindi parleremo di URL, HTML,HTTP,

Javascript, PHP e soprattutto jQuery ossia la libreria che ci permetterà di creare tabelle modificabili. Concluderemo con una breve descrizione del DBMS.

Il quarto capito consiste nella vera e propria descrizione del progetto. Inizieremo con una descrizione del database messo a disposizione , procederemo con una piccola spiegazione dell'ERP e del CRM utilizzati dall'azienda ospitante.
Dopodiché procederemo con i vari punti del progetto descrivendo passo dopo passo i vari passaggi svolti.

Il quinto capitolo presenta il problema dell'SQL injection.

Il capitolo Conclusioni riporta le note conclusive relative al lavoro svolto.

2.-CRM e ERP

2.1 -Che cos' è un CRM

Customer Relationship Management : significa gestire in modo efficace i rapporti con i propri clienti.
Il CRM stabilisce un nuovo approccio al mercato che pone il cliente al centro del business e non più il prodotto.
Non è solo un sistema informatico ma riguarda una visione dell'azienda nella sua totalità.
In questo elaborato porremo l'attenzione alla parte informatica , parliamo del CRM operativo, senza però dimenticare che riguarda solo una parte del CRM.
Il CRM operativo deve tener conto di tutte le relazioni che si hanno con il cliente in particolare ai suoi contatti e le relazioni azienda-cliente.
Il fine ultimo del CRM è quello di mantenere relazioni stabili con i clienti acquisiti e di ampliare il campo a clienti nuovi.

2.2 -Che cos' è un ERP

Enterprise Resource Planning : significa pianificazione delle risorse d' impresa.
Un ERP è un software di gestione che integra tutti i processi di business rilevanti di un'azienda(vendite, acquisti, gestione del magazzino, contabilità). Questo sistema mantiene gli ordini fatti da un fornitore, aggiorna il valore del magazzino quando entra una nuova merce e si prepara a ricevere la fattura dal fornitore, in questo modo il controllore può verificare la qualità della merce in modo semplice, indica dove posizionarla per la produzione, in caso la merce non vada bene la fattura verrà bloccata e farà partire le attività per la risoluzione del problema.
Attraverso l' ERP tutti i dati sono online e sempre disponibili in modo chiaro evitando errori dovuti alla comunicazione del dato tra utenti.
Il fine ultimo è quello di mantenere i dati aggiornati e reperibili da un unico sistema .

2.3 -Differenza tra ERP e CRM

Il CRM serve per gestire le relazioni con i clienti parliamo quindi del customer care, mentre il software ERP serve per controllare i processi per l' approvvigionamento delle materie prime, la produzione , la vendita e i contratti con i fornitori.

Quindi la differenza principale è che il primo genera e organizza i dati dei clienti , il secondo coordina i flussi organizzativi dell'azienda.

3 -Supporto tecnologico alla realizzazione del sito Web

E' necessario definire gli strumenti per creare un generico sito Web dinamico, ovvero un sito che permetta all'utente di interagire con lo stesso.

3.1 -PhpStorm IDE

E' un prodotto JetBrains pensato per lo sviluppo PHP , fornisce un editor per PHP, HTML e JavaScript.Spesso l'IDE aiuta lo sviluppatore segnalando errori di sintassi del codice direttamente in fase di scrittura, oltre a tutta una serie di strumenti e funzionalità di supporto alla fase di sviluppo e debugging.
Normalmente è uno strumento software che consiste di più componenti :

- un editor di codice sorgente
- un compilatore
- un tool di building automatico
- un debugger

3.2 -Web

La comunicazione tra processi su diversi in esecuzione su host terminali diversi serve per realizzare servizi di rete, uno di questi è il World Wide Web (WWW) che utilizza il protocollo applicativo HTTP per realizzare un modello client/server.
I tre elementi principali del Web sono :

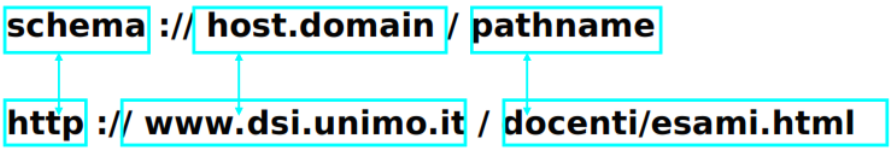
- URL : sistema di indirizzamento delle risorse
- HTML: linguaggio di markup ipertestuale
- HTTP : protocollo per le richieste delle risorse

Per risorse si intendono tutti quelli elementi presenti nel Web quali:

- pagine (testo, immagini, suoni, video,..)
- risultati di esecuzione
- programmi eseguibili

3.2.1 Uniform Resource Locator (URL)

I campi dell'URL sono :



- **schema**: indica il modo con cui accedere alla risorsa, cioè quale protocollo bisogna usare per interagire con il server che controlla la risorsa. Il metodo di accesso più comune è HTTP (protocollo nativo del WWW per il recupero di risorse Web)
- **host.domain**: è l'hostname del nodo nel quale risiede la risorsa Web.
- **pathname**: identifica la risorsa presso il server Web.

Linguaggio di markup
Per permettere la corretta visualizzazione dell'informazione su qualsiasi piattaforma hardware connessa in rete è stato necessario definire un nuovo linguaggio di markup per la formattazione delle pagine:

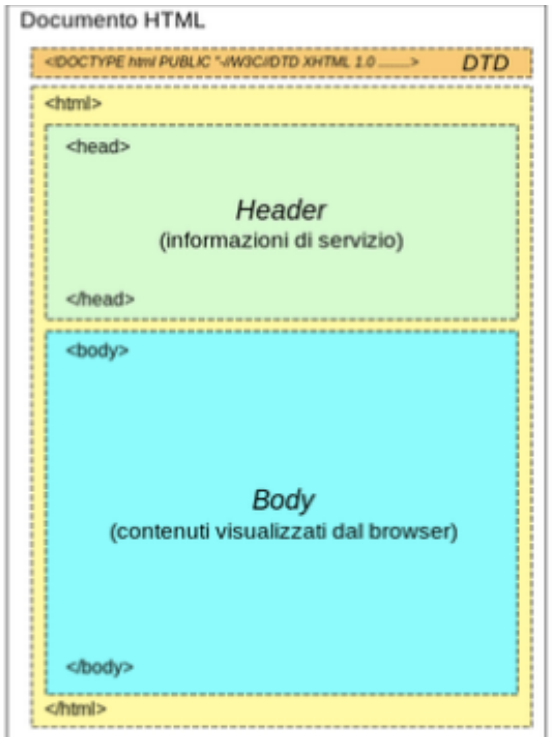
- fornisce delle linee guida generali per la rappresentazione del contenuto.
- non specifica esattamente il formato e la posizione del testo, lasciando ai browser la definizione dei dettagli.

Due browser potrebbero visualizzare lo stesso documento in modo differente

3.2.2 Linguaggio HTML

- Sebbene siano stati proposte modifiche ed altri standard, a tutt'oggi HyperText Markup Language (HTML) nelle sue varie incarnazioni rimane il "linguaggio del Web".
- è un linguaggio di formattazione che descrive le modalità di impaginazione o visualizzazione grafica (layout) del contenuto, testuale e non, di una pagina web attraverso tag di formattazione. Sebbene l'HTML supporti l'inserimento di script e oggetti esterni quali immagini o filmati, non è un linguaggio di programmazione: non prevedendo alcuna definizione di variabili, strutture dati, funzioni o strutture di controllo che possano realizzare programmi, il suo codice è in grado soltanto di strutturare e decorare dati testuali.
- la potenza di Internet consiste nell'essere un insieme esteso di contenuti connessi tra loro. Il linguaggio HTML e i link sono alla base di questo meccanismo che consente di muoversi

rapidamente tra testi, immagini, video, applicazioni e quant'altro creando percorsi i propri percorsi di navigazione in totale autonomia.



• la pagina HTML è un file di solo testo ASCII.
• il contenuto del testo e specifiche di formato possono essere inseriti nello stesso file.
• la descrizione dei contenuti dell'ipertesto viene effettuata inserendo all'interno del testo stesso alcune istruzioni dette marcatori o markup o tag che producono le visualizzazioni e le azioni specificate.
• gli statement HTML sono racchiuse tra parentesi angolari, nella forma <tag>, e vengono terminate da un tag di chiusura nella forma </tag>.
(Il testo è una miscela di testo classico con una serie di elementi di suggerimento marcati da parentesi angolati (tag) che ne indicano il significato)
• ciascuna immagine è contenuta in un file differente dal testo, vengono usati espliciti tag per le immagini:
<IMG SRC "Pathname_del_file" >

Schema di un documento HTML

L'istruzione più innovativa dell'HTML è l'àncora delimitata dai tag <A>…</A>, in quanto tale elemento permette di trasformare un normale testo in ipertesto multimediale ossia un link.
Un àncora può far riferimento ad una sezione della stessa pagina oppure ad una qualsiasi risorsa (testuale, multimediale, eseguibile) presente sul Web, denotata mediante un URL che va inserito all'interno del tag àncora:
<A HREF="http://www.unimo.it/studenti/erasmus.html"> Programma Erasmus </A>
usando l'HREF indico che il contenuto è un collegamento che richiede il documento specificato nel tag, il testo *Programma Erasmus* viene visualizzato in modo differente e risulta un link simbolico selezionabile via mouse. Il tag àncora consente l'ipermedialità su scala geografica conseguenza dirompente dal punto di vista storico e sociale: l'informazione non è più organizzata in rigidi schemi (es., biblioteca), ma è accessibile (anche) mediante link decisi da singole persone, aziende, organizzazioni, …
Ogni link inserito tra una pagina Web e un'altra determina una nuova modalità di accesso all'informazione
I documenti HTML vengono immagazzinati sui dischi rigidi di macchine elaboratrici (computer-server) costantemente collegate e connesse alla rete Internet. Su queste macchine è installato un software specifico (web server) che si occupa di produrre e inviare i documenti ai browser degli utenti che ne fanno richiesta usando il protocollo HTTP per il trasferimento dati. La formattazione di documenti HTML può avvenire attraverso un linguaggio chiamato CSS.

3.2.3 -LINGUAGGIO CSS

L'acronimo CSS significa Cascading Style Sheets (fogli di stile a cascata) ed è un linguaggio che definisce lo stile dei documenti web. I CSS comunicano al browser o un altro programma utente su come il documento debba essere presentato all'utente, per esempio controllare i bordi, colori, margini, allineamenti, font e tante altre proprietà per ottenere l'effetto visivo desiderato.
La pratica migliore è posizionare i fogli di stile in un file esterno all'HTML che sarà richiamato dall'HTML stesso all'avvio del file, o tramite JavaScript al verificarsi di un evento come il click del mouse. Inserendo i fogli di stile in file separati sarà possibile riutilizzare lo stesso CSS in più pagine in particolare è possibile definire un selettore in HTML per un elemento che verrà richiamato nella pagina .css e definirne lo stile in modo separato dagli altri. Il foglio di stile può essere inserito anche all'interno della pagina HTML, ma è necessario copiare ed incollare il codice da una pagina all'altra per riutilizzarlo. Diventa quindi difficile la sincronizzazione e l'aggiornamento.Il terzo modo per inserire i fogli di stile all'interno dell'HTML è detto inline, vuol dire inserire lo stile direttamente all'interno del tag di apertura di un elemento HTML

3.2.4 -PROTOCOLLO HTTP

HyperText Trasmission Protocol (HTTP) è il protocollo che permette il reperimento delle risorse Web.
È un protocollo applicativo di tipo request/reply basato sulla suite di protocolli TCP/IP.
Tutti i client e server Web devono supportare il protocollo HTTP per poter scambiare richieste e risposte. Per questa ragione i client e i server Web sono chiamati anche client HTTP e server HTTP.
HTTP usa TCP come protocollo di trasporto

- il client inizia la connessione TCP verso il server sulla porta 80
- il server accetta la connessione TCP dal client
- messaggi HTTP di tipo testuale scambiati tra browser e Web server
- chiusura della connessione TCP
- TCP offre un servizio di trasferimento affidabile: i messaggi di richiesta/risposta sono consegnati integri al destinatario

HTTP è un protocollo stateless (senza stato)

- il server non conserva nessuna informazione riguardante le richieste dei client passati
- I protocolli che conservano lo stato sono complessi!
  - la storia passata (lo stato) deve essere memorizzata
  - se il server/client subiscono un crash, la vista dello stato può essere inconsistente e deve essere ristabilita

I messaggi HTTP sono di due tipi:

- messaggi di richiesta HTTP

- messaggi di risposta HTTP

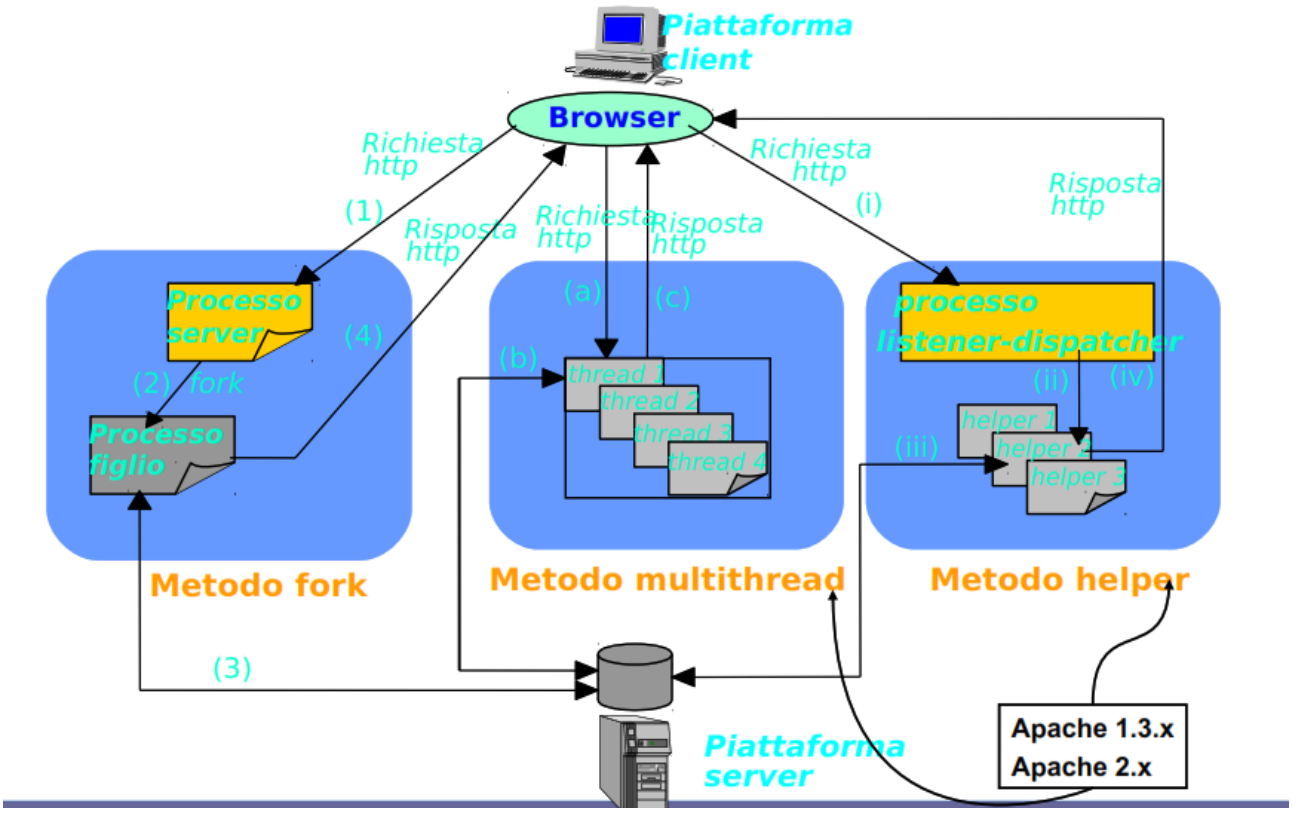
Una richiesta HTTP comprende
  – metodo
  – URL
  – identificativo della versione del protocollo HTTP
  – insieme di extension header

• Il metodo specifica il tipo di operazione che il client richiede al server. Il metodo più comune è GET che serve per acquisire pagine Web.

- Gli header contengono informazioni addizionali, quali la data e l'ora della comunicazione, il tipo di software utilizzato dal client, i tipi di dato che il browser è in grado di visualizzare, per un totale di circa 50 tipi di header differenti.

3.2.5 -WEB SERVER

Il Web server è un applicazione software che si trova in un server. Per usufruire dei servizi che offre ci sono varie operazioni che vengono eseguite.
Innanzitutto il sistema operativo deve dichiarare di voler instaurare una connessione che viene aperta , sia dalla parte del client sia dalla parte del server.



Il server assume di definire la connessione prima del client e rimane in attesa che il client si connetta alla porta specificata, infatti ogni servizio avrà una porta dedicata ossia un numero univoco tramite cui avviene la connessione client-server.
Il client assume che il server sia già attivo e prova a connettersi specificando l' indirizzo IP e la porta del server.
Tutta la comunicazione dei dati nel Web avviene attraverso il protocollo HTTP: avremo richieste e risposte HTTP.
Alla fine dello scambio dati vi è la chiusura della connessione.
Il Web server ha diversi modi di gestire le richieste HTTP con il Browser :

- metodo fork()
- metodo multithread
- metodo helper

Gli ultimi due sono utilizzati da Apache, ossia il Web server che utilizzeremo per la realizzazione del nostro sito Web.

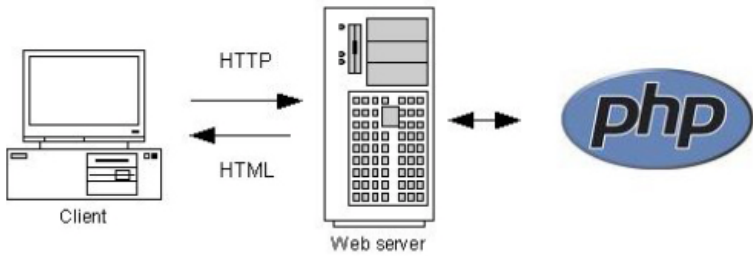
3.2.6 -APACHE

È la piattaforma server Web modulare più diffusa, in grado di operare su una grande varietà di sistemi operativi, tra cui UNIX/Linux, Microsoft Windows.
Apache, oltre alle funzionalità generali di un server Web ,supporta anche il cosiddetto server-side scripting: ciò permette di generare dinamicamente i documenti da servire. Tra i linguaggi di scripting supportati il più popolare è PHP .Il Web Server presenta un'architettura modulare, quindi ad ogni richiesta del client vengono svolte funzioni specifiche da ogni modulo di cui è composto, come unità indipendenti. Ciascun modulo si occupa di una funzionalità, ed il controllo è gestito dal core.Il core passa poi la richiesta ai vari moduli in modo sequenziale, usando i parametri di uscita di un modulo come parametri di accesso per il successivo, creando così l'illusione di una comunicazione orizzontale fra i moduli (Pipeline).
Le fasi principali di un il ciclo sono:

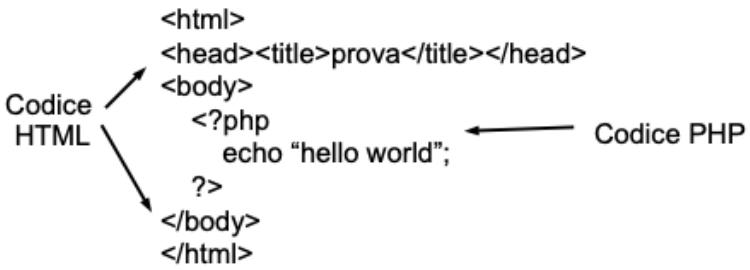
- **Translation**: traduce la richiesta del client
- **Access Control**: controlla le richieste in base ai criteri di autorizzazione
- **MIME Type**: identifica il tipo di contenuto
- **Response**: invia la risposta al client e attiva eventuali procedure
- **Logging**: tiene traccia di tutto ciò che è stato fatto

3.2.7 -PHP

PHP è un linguaggio per lo scripting server-side, vuol dire che la generazione dei documenti dinamici avviene direttamente sul server, ovvero un linguaggio che risiede in un server in remoto e che in fase di esecuzione interpreta le informazioni ricevute da un client grazie al Web server, le elabora e restituisce un risultato al client che ha formulato la richiesta.
Il server remoto attraverso un linguaggio di scripting interpreta la richiesta del client ed invia una risposta (ad esempio una pagina HTML, un oggetto JSON o un XML) al client. A questo punto il client è in grado di interpretare la risposta ricevuta e fornirla all'utente; nel caso del browser riceverà una pagina HTML che mostrerà all'utente.



La pagina PHP si presenta come un insieme di statement HTML standard e di istruzioni PHP:



Il codice PHP è embedded nelle pagine HTML e viene racchiuso tra i tag <?php e ?>.
La sua caratteristica più importante è la ricchezza delle funzioni usate per interfacciarsi con un Database, punto di forza importante per lo sviluppo di siti Web.
Un altra caratteristica importante è che non esiste il tipo delle variabili ossia è un linguaggio non tirato, vi è un casting automatico in caso di Overflow nelle operazioni, inoltre non c'è il bisogno che le variabili siano dichiarate, le notazioni usate per la loro rappresentazione:

- $variabile
-${variabile}
-{$variabilie}

Una features importante del linguaggio PHP riguarda il fatto che gli array sono una struttura dati associativa, queste strutture dati prendono il nome di Dictionary o Hashtables, l' array contiene dei valori ciascuno identificato da una chiave.
Le Hashtable

-sono array di puntatori a liste concatenate

-gestisce una coppia di tipo chiave valore(K,V).

Data la chiave K si applica una funzione Hash per identificare quale lista di Bucket usare.

3.2.8 -JAVASCRIPT

JavaScript è un linguaggio di programmazione orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione di effetti dinamici interattivi nei siti e nelle applicazioni web, tramite funzioni di script invocate da eventi innescati a loro volta in vari modi dall'utente sulla pagina web.Tali funzioni di script possono essere inserite in file HTML o PHP. L'integrazione avviene direttamente nel file HTML racchiusa dal tag <script> o tramite appositi file separati con estensione *.js*, richiamati inserendo nel tag <script> l'attributo *src* che abbia come valore l'URL del file (indirizzo univoco della risorsa online).
Una delle librerie più importanti di Javascript è jQuery di cui ci occuperemo nel seguito.

AJAX

E' l' acronimo di Asynchronous JavaScript and XML ed è oggi la tecnica più utilizzata per sviluppare applicazioni web interattive, altrimenti dette RIA (Rich Internet Application).Si definisce asincrona una chiamata ad una risorsa esterna che non interferisce con l'esecuzione della risorsa chiamante: i risultati della risorsa esterna saranno utilizzabili solo quando disponibili senza "tempi morti" per l'utilizzatore (il caricamento della risorsa esterna avviene in background). Attraverso Ajax è possibile cambiare dinamicamente il contenuto di una pagina o una sua porzione senza effettuare il ricaricamento dell' URL, recuperando informazioni aggiornate da una risorsa Web che potrà essere:

- statica (un file HTML o un XML)
- dinamica (script PHP , JSP o altro)

che viene contatta attraverso una chiamata HTTP lanciata tramite Javascript.Sarà Javascript ad occuparsi di gestire eventuali errori e di manipolare il risultato ricevuto in risposta interagendo con il DOM del documento. Il concetto che sta alla base di una chiamata AJAX è quello di poter scambiare dati tra client e sever senza dover ricaricare la pagina.Lo scambio avviene in background tramite chiamata asicnrona dei dati di solito utilizzando l' oggetto XMLHttpRequest, facile da implementare quando bisogna usare jQuery e sarà il nostro caso.

3.2.9 FRAMEWORK JQUERY

Nasce con l'obiettivo di semplificare la selezione, la manipolazione, la gestione degli eventi e l'animazione di elementi DOM (Document Object Model) in pagine HTML, nonché semplificare l'uso di funzionalità AJAX, la gestione degli eventi e la manipolazione dei CSS.Le sue caratteristiche permettono di astrarre le interazioni a basso livello con i contenuti delle pagine HTML. L'approccio di tipo modulare di jQuery consente la creazione semplificata di applicazioni web e contenuti dinamici versatili.
jQuery è un framework, ossia una libreria composta da funzioni potenti che facilitano l'interazione con il linguaggio javascript, ad esempio permette di cambiare con facilità il contenuto della pagina, la formattazione degli elementi e soprattutto di animarli, ossia diventano in grado di rispondere ad eventi precisi. In particolare tramite la tecnologia AJAX collegarsi in background ad un server richiedere delle informazioni e con queste aggiornare il contenuto, in questo modo all' utente sembrerà di non abbandonare mai la pagina e di avere un interazione che si avvicina alle applicazioni offline.
Un altra particolare funzione è la compatibilità cross-browser che risolve il problema della visualizzazione della pagina web in modo diverso nei vari browser, ad esempio colonne non allineate ecc., quindi l' aspetto rimane coerente passando da un browser all' altro.

3.3 -DBMS

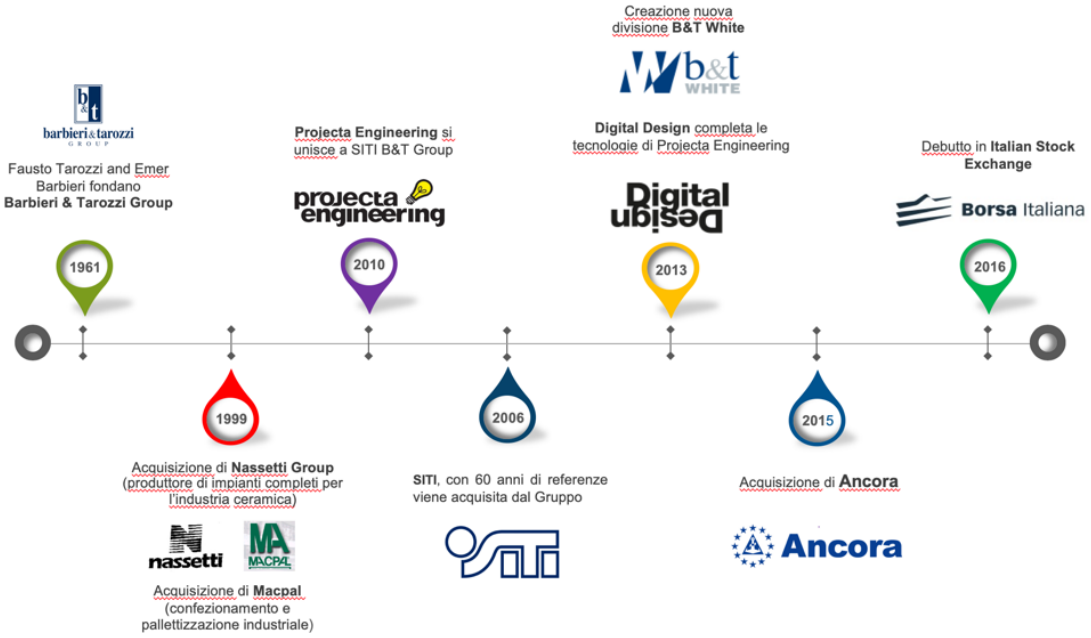
Microsoft SQL Server è un DBMS relazionale (Relational Database Management System RDBMS), prodotto da Microsoft,ossia un sistema software progettato per consentire la creazione, la manipolazione e l'interrogazione efficiente di database.
E' relazionale perché tutti i dati sono rappresentati come relazioni e manipolati con gli operatori dell'algebra relazionale o del calcolo relazionale.
Nel nostro caso utilizzeremo operazioni JSON per le operazioni da effettuare sul database.

4. Realizzazione del progetto del caso analizzato

4.1 L' azienda Siti b&t : introduzione alla realtà aziendale

Il Barbieri & Tarozzi Group è una realtà storica del comparto maccano-ceramico del distretto di Sassuolo (MO), presente sul territorio sin dal 1961, che vede il suo core business nella progettazione, ingegnerizzazione, produzione e installazione presso il cliente di macchinari e impianti completi per l'industria ceramica.
Il nuovo Gruppo è in grado di offrire i propri servizi come fornitore completo di pianti ceramici : un offerta globale che va dalla progettazione alla realizzazione di tutti i processi produttivi in ogni

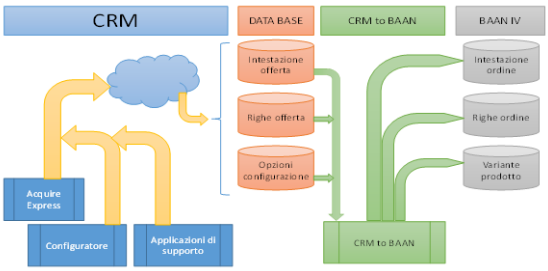
settore dell'industria ceramica.Il Gruppo b&t, così come si presenta oggi conta, tra capo-gruppo, consociate e filiali 600 dipendenti, 203 milioni di euro di ricavi da vendite e oltre 2500 clienti in tutto il mondo. SITI-b&t si occupa di progettare e costruire impianti completi per piastrelle ceramiche e sanitari.



4.2 Strumenti utilizzati: Baan , Lyra e Portale 2013

4.2.1 Funzionalità di BAAN

BAAN è il sistema ERP utilizzato in Siti b&t , ed è uno strumento in grado di interagire con tutti i processi rilevanti dell' azienda (contabilità, produzione,, warehousing…).



Si presenta all' utente come un elenco di funzionalità , permette la customizzazione del profilo utente in modo da mettere in evidenza e rendere più facilmente accessibili gli strumenti necessari alle specifiche attività di analisi e gestione. Il nostro interesse non è quello di descrivere le varie funzionalità di questo ERP ma di conoscere i dati dei contratti che utilizzeremo, e che verranno descritti mano a mano che gli incontreremo nella stesura del progetto.

4.2.2 Lyra

E' l'applicativo sviluppato da Softeam SpA, le principali caratteristiche di Lyra sono:

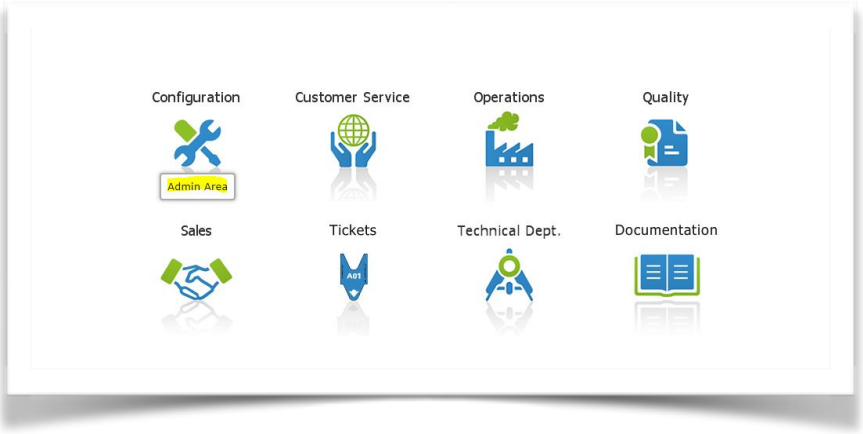
- configurazione commerciale di prodotto
- analisi e legame con CRM
- post vendita centralizzato

racchiude in un'unica interfaccia totalmente personalizzabile la gestione puntuale e centralizzata di tutti i dati e le informazioni connesse alle relazioni con prospect e clienti.

4.2.3 Portale 2013

Il portale presenta l'ambiente in cui svilupperemo il nostro sito in particolare ci occuperemo dell'area Sales. Sfrutteremo in particolare l'Admin Area in cui è possibile inserire un nuovo collegamento a un sito, parte che verrà approfondita in seguito.

Gli utenti che usufruiranno del sito in creazione accederanno a partire da questo sito dopo aver effettuato il login alla pagina.



4.3 VARI PUNTI DEL PROGETTO

I dati che verranno utilizzati sono contenuti in vari database che saranno messi a disposizione per la realizzazione del progetto. In particolare saranno di interesse alcune viste create attraverso la lettura di dati importati da Lyra e Baan, in modo da poter mettere a disposizione dell'utente un interfaccia capace di modificare le righe dei contratti e di inserire queste modifiche nei file che saranno letti da Baan in modo automatico. La creazione dei file sarà chiamata trasferimento ed è il fine ultimo del sito Web tale sezione verrà approfondita nel seguito.

Il primo sito web conterrà la lista di tutti i contratti che potranno essere modificati, ognuno è identificato dal *Numero Ordine*. Per ogni contratto creeremo una sezione *Dettaglio* che riporterà tutte le righe specifiche di quel contratto che potranno essere modificate. Infine per ogni contratto avremo a disposizione tre operazioni possibili:

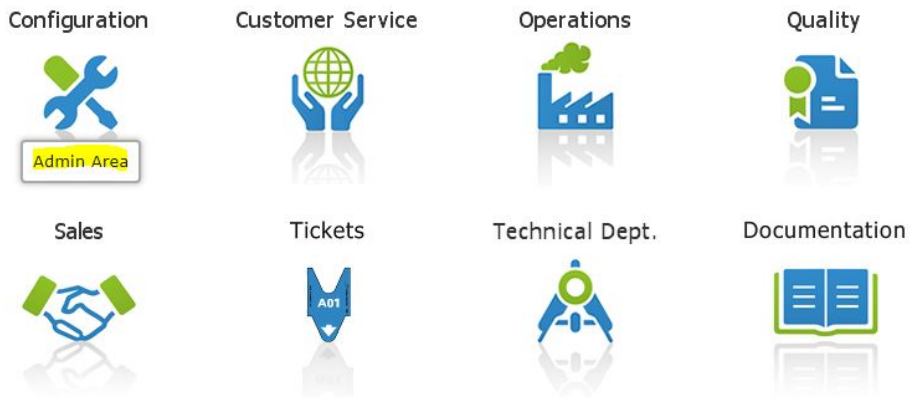
1. Reset, che resetterà tutte le righe del contratto leggendole direttamente da Lyra

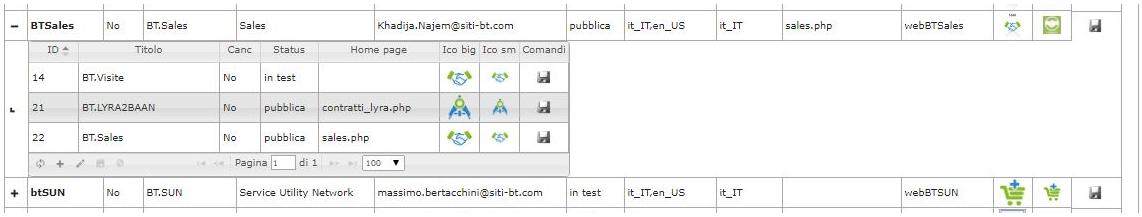
2.Trasferimento, consiste nella creazione di quattro file che verranno automaticamente letti da Baan.

3.Totale, calcolerà il totale del contratto suddividendo per reparti.

4.3.1 Impostare una nuova app sul Portale intranet sotto il gruppo di applicazioni *Sales*

Iniziamo creando il nostro sito: questa funzionalità è già stata implementata dalla sezione *Configuration* del Portale di Siti b&t, quindi non la descriveremo dal lato software ma solo dal lato client.





I siti dovranno essere creati nell'area *Sales* per cui il nome del sito di partenza sarà aggiunto sotto quel dominio.

Adesso aprendo la finestra *Sales* avremo due finestre quella da noi creata è *BT.LYRAA2BAAN*

Server su cui lavoreremo
Sono tre:

1. LIBRA
   contiene il database webBTSales che sarà quello di nostro interesse in cui saranno create le stored procedure che vorremmo eseguire e le tabelle.In particolare è il server sotto il quale sono presenti tutti i database del Portale Siti b&t, in cui abbiamo creato anche il nostro sito.
2. SRVLYRA
   contiene le tabelle e le viste da cui ricaveremo i dati relativi a Lyra che vorremmo visualizzare.

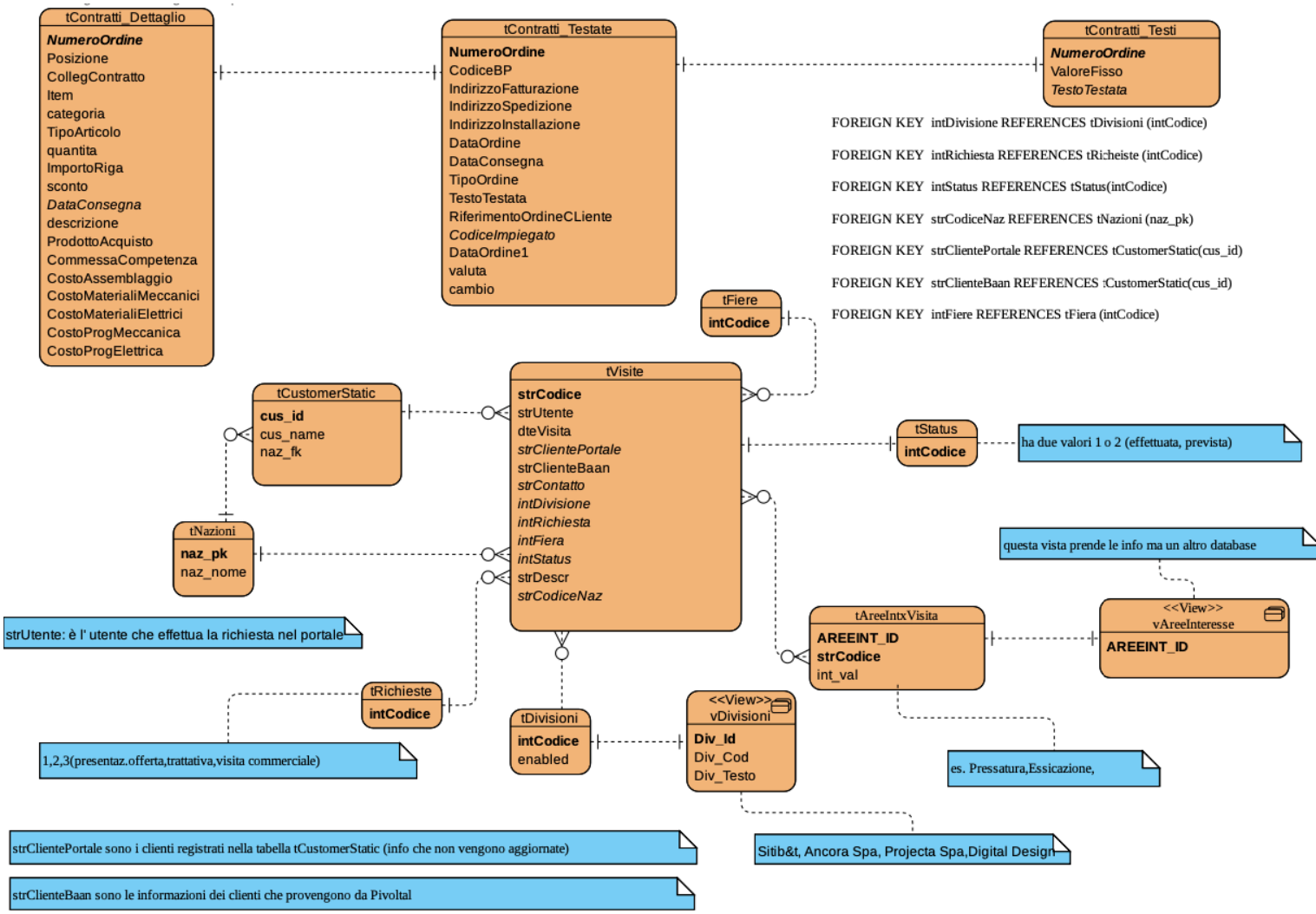
3. PROD
   contiene le tabelle dove sono inseriti i dati dei contratti in Baan

La struttura del database

Descriviamo la struttura del database utilizzato nel progetto:
Il database principale su cui lavoreremo è *BTSales* e riguarda tutta la parte delle visite dei clienti e dei contratti, quindi in generale i rapporti che si hanno con i clienti.
Ogni visita è legata :

-Customer (**tCustomerStatic**): ogni cliente è rappresentato da un id (*cus_id*) , da un nome (*cus_nome*) e dalla nazione di provenienza, l'attributo *naz_fk* è una foreign key verso la tabella tNazioni in cui ci sono le relazioni tra codice nazione e nome nazione.

- Area (**tAreeIntxVisita**): ogni visita avrà un area di riferimento in cui si svolgerà la visita all'interno di una divisione.



-Divisione(**tDivisioni**) : rappresentano le diverse divisioni dell'azienda Siti B&T:

1.Ancora Spa

2.Projecta

3.Digital Design

-richiesta(**tRichieste**): i codici di una richiesta possono essere i valori numerici 1 , 2 e 3 che rappresentano tre diverse richieste

1: presentazione offerta

2: trattativa

3: visita commerciale

-stato(**tStatus**): lo stato di una vista può essere:

1: effettuata

2:prevista

in modo da tenere traccia delle visite ancora da effettuare e quelle che sono state effettuate e potranno diventare veri e propri contratti

-fiera(**tFiere**): se la visita non si svolge in una fiera questo campo sarà NULL

La tabella **tVisite** presenta i seguenti attributi:

- *strCodice* : che è la PRIMARY KEY è rappresenta il codice univo di una visita con cui viene identificata
- *strUtente*: è il nome dell' utente che si occuperà di gestire la richiesta di visita
-*dteVisita*: è la data in cui si terrà la visita
-*strClientePortale*: è il codice univoco del Customer presente in tCustomer
-*strClienteBaan*: è un codice con cui identifico il cliente all' interno del CRM Baan
-*intDivisione*: può avere valore 1,2,3 in base alla divisione (tDivisioni)
-*intRichiesta*: può assumere i valori: 1,2,3 in base alla richiesta(tRichiesta)
-*intFiera*: codice della fiera (tFiera)
-*intStatus*: stato della visita (tStatus)
-*strDescr*: è una descrizione breve di quello che verrà svolto durante la visita
-*strCodiceNazione*: codice della Nazione (tNazioni)

Per quanto riguarda i contratti avremo tre tabelle (tContratti_Testate, tContratti_Testi, tContratti_Dettaglio) di cui parleremo nel seguito in quanto rappresentano una parte del progetto vero e proprio.

4.3.2 Creazione della pagina contratti_lyra.php

Questa pagina presenterà la lista di tutti i contratti visualizzabili che sono presi da Lyra. La pagina web sarà una griglia filtrabile e ordinabile. Dovrà inoltre contenere un'icona nel campo Comandi collegata alla pagina web di dettaglio del contratto.
Per la realizzazione della griglia utilizziamo un plugin di JavaScript : jqGrid i campi della griglia saranno :

week	CodiceContratto	CodiceCliente	RagioneSociale	Descrizione contratto	Importato in Baan, (si/no)	RigheContratto	C M D

Innanzitutto creiamo una vista : *vContrattiLyra* che recupererà i dati da altre tre viste messe a disposizione da diversi database:

a. **LyraBaan.dbo.VIEW_ORDER_TESTATA:** vista contenente i dati dei Contratti Lyra
b. **BAAN.tccom100900:** rappresenta i dati dei Clienti e Fornitori
c. **BAAN.ttdsls400900:** contiene dati dei Contratti Baan

Per ogni riga avremo i dati selezionati dalla vista vContrattiLyra.
La query per creare la vista vContrattiLyra è riportata nel seguito:

```sql
CREATE VIEW [dbo].[vContrattiLyra] AS
SELECT DISTINCT
    SUBSTRING(c.NumeroOrdine, 2, 6) AS CodiceContratto, b.T$BPID AS CodiceCliente, b.T$NAMA AS RagioneSociale,
    SUBSTRING(c.TestoTestata, 2, LEN(c.TestoTestata) - 2) AS DescrizioneContratto,
    CASE WHEN a.T$CONT IS NULL THEN 'NO' ELSE 'SI' END AS ImportatoBaan,
    CASE WHEN substring(c.NumeroOrdine, 2, 6) IN (SELECT substring(NumeroOrdine, 2, 6)
                                                  FROM OPENQUERY(SRVLYRA, 'SELECT * FROM LyraBaan.dbo.VIEW_ORDER_ROWS2'))
    THEN (SELECT COUNT(*)
          FROM   OPENQUERY(SRVLYRA, 'SELECT * FROM LyraBaan.dbo.VIEW_ORDER_ROWS2') AS a
          WHERE        substring(c.NumeroOrdine, 2, 6) = substring(a.NumeroOrdine, 2, 6))
    ELSE '0' END AS RigheDisponibili,
    DATEPART(week, d.DataConsegna) AS week


FROM (SELECT TestoTestata, NumeroOrdine, SUBSTRING(CodiceBP, 2, 6) AS CodiceBP FROM OPENQUERY(SRVLYRA, 'SELECT * FROM LyraBaan.dbo.VIEW_ORDER_TESTATA')) AS c
     LEFT OUTER JOIN
      OPENQUERY(PROD, 'select * from tcdsls400900') AS a ON c.CodiceBP = a.T$BPID
     INNER JOIN
      (SELECT       T$BPID, T$NAMA FROM OPENQUERY(PROD, 'select * from ttccom100900')) AS b ON c.CodiceBP = b.T$BPID
     LEFT OUTER JOIN
      (SELECT DataConsegna, NumeroOrdine FROM OPENQUERY(SRVLYRA, 'SELECT * FROM LyraBaan.dbo.VIEW_ORDER_ROWS2')) AS d ON d.NumeroOrdine = c.NumeroOrdine
GO
```

- **SUBSTRING (expression, start, length)** : i dati presenti nelle viste sono tra virgolette per questo utilizzeremo la funzione substring in modo da togliere per una corretta visualizzazione dei dati.
- **DATEPART (datepart, date)** :la prima colonna week contiene il numero della settimana dell'anno, per fare questo usiamo la funzione datepart che ha come variabili la DataConsegna e il tipo in cui si vuole venga trasformata nel nostro caso usiamo week.
- **OPENQUERY (linked_server,'query')** : per ricavare i dati dalle viste abbiamo utilizzato la funzione openquery in quanto le viste non sono presenti nello stesso server della vista vContrattiLyra quindi non è possibile eseguire la query direttamente ma bisogna connettersi al server indicato dalla prima variabile.

  La tecnica usata per visualizzare la griglia di dati nel nostro siti web è jqGrid.

  Dobbiamo però dividere in due momenti la stampa della griglia. Per fare questo abbiamo definito una variabile *a* che per default ha valore 0 dopodichè nell'url della jqGrid viene cambiato il suo valore in 1 in modo che venga eseguito il codice PHP.

  1. innanzitutto dobbiamo definire lo spazio della griglia: ciò viene fatto in HTML (*a*=0). Il codice javascript è eseguito nella sezione <head> di HTML quindi sempre in *a*=0. Nel momento in cui viene eseguito il codice di jqGrid il valore di *a* cambia e vi è una chiamata al server
  2. La chiamata al server avviene tramite PHP usando la funzione jqGridRead verrà tutto approfondito nel seguito.

A questo punto dobbiamo capire come caricare la tabella in modo che sia visualizzata nella pagina Web.

Soluzione adottata: jqGrid

//estesa

```
jQuery("#tableId").jqGrid({/*options*/});
```

//abbreviata

```
$("#tableId").jqGrid({/*options*/});
```

jqGrid è un componente che presenta dati tabulari per una facile manipolazione in una griglia del browser web.

- jQuery è un framework
- vengono utilizzate chiamate Ajax per recuperare informazioni e costruire la rappresentazione utilizzando il MODELLO COLONNA jqGrid (colModel, si approfondirà nel seguito).
- per comprendere come funziona jqGrid dobbiamo lavorare su due aspetti:
    1. La rappresentazione tabulare lato client
    2. La rappresentazione tabulare lato server
- rende semplice anche la modifica dei dati nel database sul lato server utilizzando PHP o qualsiasi altro linguaggio di programmazione comune
- supporta il paging, l'ordinamento e il filtro sul lato client sul lato server
- la tabella viene divisa in quattro parti:
    1. Livello didascalia: informazioni dei dati sottostanti
    2. Livello intestazione: contiene i nomi delle colonne
    3. Strato del corpo: sono i dati richiesti dal server e visualizzati in base al colModel
    4. Strato di navigazione: contiene informazioni aggiuntive e azioni che possono essere impostate come true o false in modo che vengano rispettivamente abilitate e inabilitate.
- per utilizzare jqGrid sono necessari tre elementi:
    1. Database (nel nostro caso sarà webBTSales)
    2. Una pagina HTML per mostrare i dati
    3. Una componente lato server che funge da interfaccia tra la tua pagina web e il database
- per utilizzarlo si deve creare in HTML il tag <table> in modo da riservare lo spazio della griglia che verrà creata

CARICAMENTO GRIGLIA IN JAVASCRIPT

```
$(document).ready(function()
{
    $("#mcTable").jqGrid({
        url:'?a=1',
        datatype: "json",
```

```
        colModel:[
            {name:'week', label: 'week', index:'week',
width:100,align:"center"},
            {name:'CodiceContratto', label: 'CodiceContratto',
index:'CodiceContratto', width:100,align:"center"},
            {name:'CodiceCliente', label: 'CodiceCliente',
index:'CodiceCliente', width:100,align:"center"},
            {name:'RagioneSociale', label:'RagioneSociale',
index:'RagioneSociale', width:300},
            {name:'DescrizioneContratto', label: 'Descrizione',
index:'DescrizioneContratto', width:400},
            {name:'ImportatoBaan', label: 'ImportatoDaBaan',
index:'ImportatoBaan', width:105,align:"center"},
            {name:'RigheDisponibili', label: 'RigheDisponibili',
index:'RigheDisponibili', width:105,align:"center"},

            {name:'cmd', label: 'Comandi', index:'cmd', width:70,align:"center"}
        ],
        rowNum: 20,
        rowList: [20,50,100,1000],
        pager: '#mcTools',
        sortname: 'CodiceContratto',
        viewrecords: true, // visualizzare numero di record TOTALI
        sortorder: "desc",
        gridComplete: function() {

          var ids = $("#mcTable").jqGrid('getDataIDs');

            for(var i=0;i < ids.length;i++){
                var row_id = ids[i];
                var htmlCmd = '';
                //pulsante dettaglio
                htmlCmd += '<a href="contratti_lyra_dettaglio.php?
strCodiceContratto=' + row_id + '"><img class="ico_cmd" src="/images/interface/
icons/ico_details.gif" ' +
                    ' title="Dettaglio" alt="<?=$txt[150]?>"/></a>';
            $("#mcTable").jqGrid('setRowData',row_id,{cmd: htmlCmd});
            }
        },
        caption: "Elenco contratti"
    });



 $("#mcTable").jqGrid('navGrid',"#mcTools", {search: false, edit:false,
add:false, del:false});

 $("#mcTable").jqGrid('inlineNav',"#mcTools", {del:false, edit:false,
add:false});
});
```

URL:

- Ci dice da dove ottenere i dati
- Di solito una funzione lato server con una connessione database che restituisce le informazioni da compilare nel livello BODY

colModel:

- Un array che descrive il modello delle colonne
- Esempio:

```
{name:'CodiceContratto', label: 'CodiceContratto', index:'CodiceContratto',
width:100, align:"center"}
```

-Il nome sarà quello presente nella colonna (non deve essere per forza il nome della tabella del database)

-L' index rappresenta il vero e proprio nome che indica la colonna del database , è l'odinamento del lato server.

-width: è la larghezza della colonna stessa

-align: l'allineamento della colonna

ROWNUM

- Numero di righe che si vogliono visualizzare al caricamento della tabella

ROWLIST

- Un array per costruire un elemento box che permette di selezionare righe aggiuntive

DATATYPE:

-definisce il formato in cui ci aspettiamo che si presentino i dati nel nostro caso saranno JSON.

PAGER:

- Definisce che vogliamo una barra che ci permetta di navigare attraverso le righe
- Deve essere un elemento HTML
- Nel nostro esempio abbiamo la div mcTools

VIEWRECORDS

- Se true, jqGrid visualizza il numero di record iniziale e finale della griglia rispetto al numero totale di record nella query. Queste informazioni sono visualizzare nella barra PAGER

SORTNAME

- Presenta il nome della colonna secondo cui saranno ordinate le righe caricate dal server

SORTORDER

- legato a sortname e indica se l'ordinamento è decrescente o crescente.

gridComplete

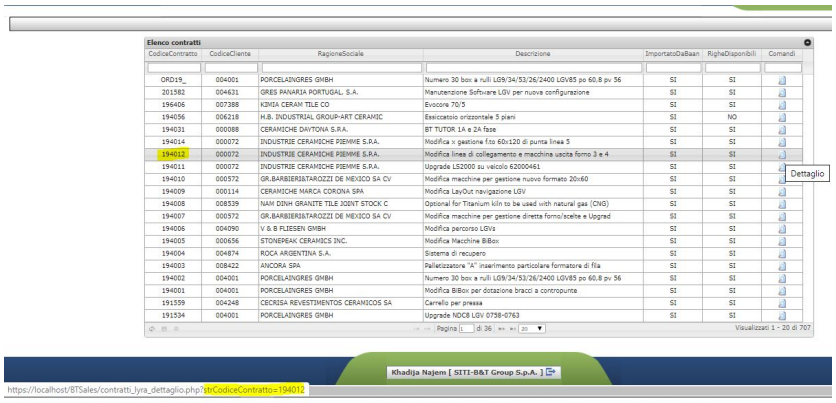
- Si attiva quando tutti i dati sono caricati nella griglia e tutti gli altri processi sono completi
- Qui avviene la creazione per ogni riga del comando Dettagli
- La variabile ids contiene il valore univoco che identifica ogni riga

```
var ids = $("#mcTable").jqGrid('getDataIDs');
```

Il metodo '*getDataIDs*' : questo metodo restituisce un array di ID della griglia corrente. Restituisce un array vuoto se non sono disponibili dati.
Impostiamo due variabili

1. *row_id* come elemento i dell'array ids
2. *htmlCmd* è definito tramite il tag <a> ossia il collegamento ipertestuale a cui aggiungiamo un altro tag <img> che ci serve per definire l' immagine di dettaglio

- href definisce la destinazione del collegamento, nel nostro caso sarà il link della seconda pagina che presenterà i dettagli.
Come si può notare appoggiando il mouse sull'immagine dettaglio abbiamo il collegamento specifico per quel contratto:



- da notare che ogni pagina contratti_lyra_dettaglio.php è legata ad un univoco CodiceContratto per fare questo il codice è proprio la variabile rowid.
- infine aggiorniamo ogni riga attraverso la funzione setRowData : aggiorna i valori nella riga con rowid nel nostro caso aggiorniamo solo la colonna cmd attraverso la variabile creata htmlCmd

```
- $("#mcTable").jqGrid('navGrid',"#mcTools", {search: false, edit:false,
    add:false, del:false});
```

jqGrid quindi presenta anche una funzionalità dell'interfaccia utente che consente una facile accessibilità per funzioni come Search, Find, Modify nel nostro caso sono tutte a false, ma le impostiamo perché potrebbero servire per un aggiornamento futuro.

```php
switch ($a)
{

    case 1:
        $resultData = jqGridRead(
            $OConnDB,
             sSQL: "SELECT * from vContrattiLyra " ,
            array('CodiceContratto', 'CodiceCliente', 'RagioneSociale', 'DescrizioneContratto', 'ImportatoBaan'),
            array('CodiceContratto', 'CodiceCliente', 'RagioneSociale', 'DescrizioneContratto', 'ImportatoBaan'),  AOrderFields: null,  namedField: true
        );

        if (!empty($resultData->rows))
        {
            foreach($resultData->rows as &$row)
            {
                $row['cell']['DescrizioneContratto']= iconv( in_charset: "CP1252",  out_charset: "UTF-8", $row['cell']['DescrizioneContratto']);
                $row['cell']['RagioneSociale']= iconv( in_charset: "CP1252",  out_charset: "UTF-8", $row['cell']['RagioneSociale']);
                //strFromIdx($row['cell']['bitDeleted'], $ACheckStr);
            }
        }



        die (json_encode($resultData));
        break;

}
```

- Quando si utilizza jqGrid viene eseguito il file identificato dall'URL che richiederà i dati dal server.
  Il valore di default della variabile *a* è 0: `$a = GETn("a", 0);`
  Nel nostro caso usiamo lo stesso URL precedente, quindi rimaniamo nella stessa pagina cambiando il valore della variabile *a* che assume valore 1 in modo che quando si entra nello switch iniziale venga scelto *case* 1 .
- Nel codice eseguito con *case* 1 il server restituirà i dati a jqGrid in un formato comprensibile.

MANIPOLAZIONE DATI jqGrid

Con la prima versione di jqGrid l'unico modo possibile per ottenere i dati era tramite XML , successivamente si potevano ottenere i dati tramite JSON quindi un array e infine con i nomi effettivi.

I dati da caricare nella tabella sono ricavati da SQLSERVER tramite la funzione jqGridRead, funzione locale definita come segue:

```php
function jqGridRead($OConn, $sSQL, $AFields, $ASearchFieldsLIKE = null, $AOrderFields = null, $namedField = false, $ASearchFieldsEXACT = null)
```

- $OConn, è la variabile che permette la connessione al database definita nel file include.php come segue:

```php
$_CONFIG['db_Host'] = "libra";
$_CONFIG['db_Portal_ConnInfo'] = array(
    "UID"=>"Portal",
    "PWD"=>[redacted]
    "Database => WebPortal");

$OConn = sqlsrv_connect($_CONFIG['db_Host'], $_CONFIG['db_Portal_ConnInfo']) or sqlsrv_str_error( sFile: __FILE__,  sLine: __LINE__,  sErr: 'Cannot connec
```

  Il server utilizzato è Libra in cui è contenuto il nostro database webBTSales.
  Inizialmente ci colleghiamo al database webPortal che contiene tutti i servizi che servono per il login e soprattutto per l'interfaccia iniziale che contiene tutte le icone iniziali. L' icona di nostro interesse è quella Sales (collegata al database webBTSales)
- $sSQL , è la variabile che contiene la stringa della query , ossia ciò che si vuole venga eseguito nel database.

- $AFields : array dei campi da leggere

Nel nostro caso utilizziamo la funzione per fare la *select* di tutti i parametri della vista vContrattiLyra il nostro array $AFields sarà ordinato secondo i parametri della vista.
La funzione *iconv* serve per utilizzare un unico formato UTF-8 , è necessaria perché in caso contrario non vengono stampate le righe in quanto json_encode non riconosce la variabile $resultData.

4.3.3 Creare nel database del portale LIBRA.webBTSales tre nuove tabelle

Creare nel database del portale LIBRA.webBTSales tre nuove tabelle: tContratti_Testate, tContratti_Dettaglio e tContratti_Testi legate dalla chiave "numero contratto". Queste tabelle dovranno contenere i dati contratto importati da Lyra e successivamente modificati.
Utilizzare le viste:

a. LyraBaan.dbo.VIEW_ORDER_TESTATA2 (Contratti Lyra)
b. LyraBaan.dbo.VIEW_ORDER_ROWS2
c. LyraBaan.dbo.VIEW_ORDER_HEADER_TESTO

Creiamo tre tabelle che avranno come colonne le stesse delle viste elencate in precedenza:

1. tContratti_Dettaglio : LyraBaan.dbo.VIEW_ORDER_ROWS2 (aggiungiamo il campo ImportoTotale)
2. tContratti_Testata : LyraBaan.dbo.VIEW_ORDER_TESTATA2
3. tContratti_Testo : LyraBaan.dbo.VIEW_ORDER_TESTO

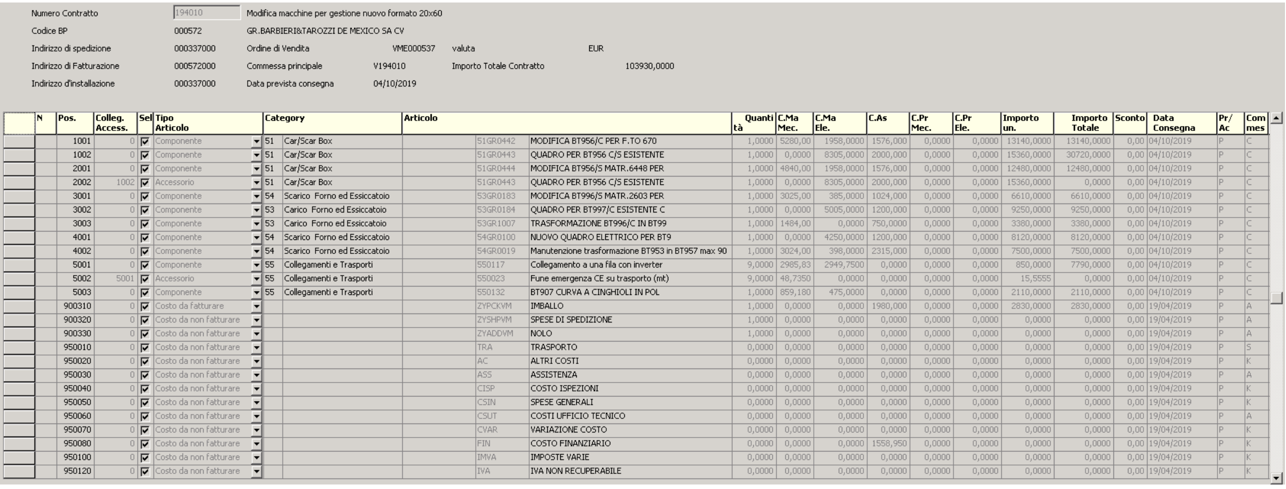
```
select * into tContratti_Testate
from srvlyra.LyraBaan.dbo.VIEW_ORDER_TESTATA2
where 1=2

--drop table tContratti_Testate
```

Ricordiamo che il tasto dettaglio è collegato a una pagina definita univoca per ogni CodiceContratto, come spiegato precedentemente. Una volta che si clicca nell'immagine il comando href ricarica la pagina all' indirizzo specificato.

4.3.4 Creazione della pagina web contratti_lyra_dettaglio.php

Creare una nuova pagina web a cui si accede cliccando sull'icona dettaglio nell'elenco contratti.
Al caricamento della pagina verificare se il contratto non è ancora presente nelle tabelle definite precedentemente e in tal caso importare i dati da Lyra.
Visualizzare poi i campi delle tabelle che dovranno corrispondere a quelli disponibili nella sessione contratti di Baan:



Numero Contratto	194010	Modifica macchine per gestione nuovo formato 20x60	
Codice BP	000572	GR.BARBIERI&TAROZZI DE MEXICO SA CV	
Indirizzo di spedizione	000337000	Ordine di Vendita VME000537	valuta EUR
Indirizzo di Fatturazione	000572000	Commessa principale V194010	Importo Totale Contratto 103930,0000
Indirizzo d'installazione	000337000	Data prevista consegna 04/10/2019	

N	Pos.	Colleg. Access.	Sel	Tipo Articolo	Category	Articolo	Quantità	C.Ma Mec.	C.Ma Ele.	C.As	C.Pr Mec.	C.Pr Ele.	Importo un.	Importo Totale	Sconto	Data Consegna	Pr/Ac	Commes
	1001	0	☑	Componente	51 Car/Scar Box	51GR0442 MODIFICA BT956/C PER F.TO 670	1,0000	5280,00	1958,0000	1576,000	0,0000	0,0000	13140,0000	13140,0000	0,00	04/10/2019	P	C
	1002	0	☑	Componente	51 Car/Scar Box	51GR0443 QUADRO PER BT956 C/S ESISTENTE	1,0000	0,0000	8305,0000	2000,000	0,0000	0,0000	15360,0000	30720,0000	0,00	04/10/2019	P	C
	2001	0	☑	Componente	51 Car/Scar Box	51GR0444 MODIFICA BT956/S MATR.6448 PER	1,0000	4840,00	1958,0000	1576,000	0,0000	0,0000	12480,0000	12480,0000	0,00	04/10/2019	P	C
	2002	1002	☑	Accessorio	51 Car/Scar Box	51GR0443 QUADRO PER BT956 C/S ESISTENTE	1,0000	0,0000	8305,0000	2000,000	0,0000	0,0000	15360,0000	0,0000	0,00	04/10/2019	P	C
	3001	0	☑	Componente	54 Scarico Forno ed Essiccatoio	53GR0183 MODIFICA BT996/S MATR.2603 PER	1,0000	3025,00	385,0000	1024,000	0,0000	0,0000	6610,0000	6610,0000	0,00	04/10/2019	P	C
	3002	0	☑	Componente	53 Carico Forno ed Essiccatoio	53GR0184 QUADRO PER BT997/C ESISTENTE C	1,0000	0,0000	5005,0000	1200,000	0,0000	0,0000	9250,0000	9250,0000	0,00	04/10/2019	P	C
	3003	0	☑	Componente	53 Carico Forno ed Essiccatoio	53GR1007 TRASFORMAZIONE BT996/C IN BT99	1,0000	1484,00	0,0000	750,0000	0,0000	0,0000	3380,0000	3380,0000	0,00	04/10/2019	P	C
	4001	0	☑	Componente	54 Scarico Forno ed Essiccatoio	54GR0100 NUOVO QUADRO ELETTRICO PER BT9	1,0000	0,0000	4250,0000	1200,000	0,0000	0,0000	8120,0000	8120,0000	0,00	04/10/2019	P	C
	4002	0	☑	Componente	54 Scarico Forno ed Essiccatoio	54GR0019 Manutenzione trasformazione BT953 in BT957 max 90	1,0000	3024,00	398,0000	2315,000	0,0000	0,0000	7500,0000	7500,0000	0,00	04/10/2019	P	C
	5001	0	☑	Componente	55 Collegamenti e Trasporti	550117 Collegamento a una fila con inverter	9,0000	2985,83	2949,7500	0,0000	0,0000	0,0000	850,0000	7790,0000	0,00	04/10/2019	P	C
	5002	5001	☑	Accessorio	55 Collegamenti e Trasporti	550023 Fune emergenza CE su trasporto (mt)	9,0000	48,7350	0,0000	0,0000	0,0000	0,0000	15,5555	0,0000	0,00	04/10/2019	P	C
	5003	0	☑	Componente	55 Collegamenti e Trasporti	550132 BT907 CURVA A CINGHIOLI IN POL	1,0000	859,180	475,0000	0,0000	0,0000	0,0000	2110,0000	2110,0000	0,00	04/10/2019	P	C
	900310	0	☑	Costo da fatturare		ZYPCKVM IMBALLO	1,0000	0,0000	0,0000	1980,000	0,0000	0,0000	2830,0000	2830,0000	0,00	19/04/2019	P	A
	900320	0	☑	Costo da non fatturare		ZYSHPVM SPESE DI SPEDIZIONE	1,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,00	19/04/2019	P	A
	900330	0	☑	Costo da non fatturare		ZYADDVM NOLO	1,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,00	19/04/2019	P	A
	950010	0	☑	Costo da non fatturare		TRA TRASPORTO	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,00	19/04/2019	P	S
	950020	0	☑	Costo da non fatturare		AC ALTRI COSTI	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,00	19/04/2019	P	K
	950030	0	☑	Costo da non fatturare		ASS ASSISTENZA	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,00	19/04/2019	P	A
	950040	0	☑	Costo da non fatturare		CISP COSTO ISPEZIONI	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,00	19/04/2019	P	K
	950050	0	☑	Costo da non fatturare		CSIN SPESE GENERALI	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,00	19/04/2019	P	K
	950060	0	☑	Costo da non fatturare		CSUT COSTI UFFICIO TECNICO	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,00	19/04/2019	P	A
	950070	0	☑	Costo da non fatturare		CVAR VARIAZIONE COSTO	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,00	19/04/2019	P	K
	950080	0	☑	Costo da non fatturare		FIN COSTO FINANZIARIO	0,0000	0,0000	0,0000	1558,950	0,0000	0,0000	0,0000	0,0000	0,00	19/04/2019	P	K
	950100	0	☑	Costo da non fatturare		IMVA IMPOSTE VARIE	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,00	19/04/2019	P	K
	950120	0	☑	Costo da non fatturare		IVA IVA NON RECUPERABILE	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,00	19/04/2019	P	K

Creiamo la nuova pagina php sotto il dominio BT.Sales con il nome contratti_dettaglio.php

La prima cosa da fare è definire le due variabili :

1. La prima variabile è *a* che ci permette di definire diversi eventi di creazione della pagina.
   ```
   $a = GETn("a", 0);
   ```
   quando *a*=0 viene caricata la pagina HTML attraverso una richiesta http la risposta sarà in codice javascript che crea cosi la griglia vuota senza dati. Per valorizzare le colonne si dovrà eseguire il codice PHP , ossia quando l' url presente in jqGrid ne cambierà il valore.
2. La seconda variabile è il codice del contratto che viene passato dall'URL
   ```
   $CC = GETn('strCodiceContratto');
   ```

Qui le colonne di nostro interesse sono le seguenti e saranno caricate nel colModel della jqGrid, da notare il campo editable che potrà essere true o false.

```
colModel:[
    {name:'pk', label: 'pk', index:'pk', hidden:true},
    {name:'NumeroOrdine', label: 'Ordine', index:'NumeroOrdine', width:80,align:"right",editable:false },
    {name:'Posizione', label: 'Pos', index:'Posizione', width:80,align:"right",editable:true },
    {name:'CollegContratto', label:'Colleg', index:'CollegContratto', width:80,align:"right",editable:true},
    {name:'TipoArticolo', label: 'Tipo', index:'TipoArticolo', width:80,align:"center",editable: true},
    {name:'categoria', label: 'categ', index:'categoria', width:80,align:"center",editable:true},
    {name:'Item', label: 'Item', index:'Item', width:80,align:"left",editable:true},
    {name:'descrizione', label: 'descr', index:'descrizione', width:240,align:"left",editable:true},
    {name:'quantita', label: 'q.ta', index:'quantita', width:80,align:"right",editable:true},
    {name:'CostoMaterialiMeccanici', label: 'CostoMecc', index:'CostoMaterialiMeccanici', width:80,align:"right",editable:true},
    {name:'CostoMaterialiElettrici', label: 'CostoElett', index:'CostoMaterialiElettrici', width:80,align:"right",editable:true},
    {name:'CostoAssemblaggio', label: 'CostoAss', index:'CostoAssemblaggio', width:80,align:"right",editable:true},
    {name:'CostoProgMeccanica', label: 'CostoPrMecc', index:'CostoProgMeccanica', width:80,align:"right",editable:true},
    {name:'CostoProgElettrica', label: 'CostoPrElett', index:'CostoProgElettrica', width:80,align:"right",editable:true},
    {name:'ImportoRiga', label: 'Importo', index:'ImportoRiga', width:80,align:"right",editable:true},
    {name:'ImportoTotale', label: 'ImportoTotale', index:'ImportoTotale', width:80,align:"right",editable:false},
    {name:'sconto', label: 'sconto', index:'sconto', width:80,align:"center",editable:true},
  // {name:'DataConsegna', label: 'DataConsegna', index:'DataConsegna', width:150,align:"center",editable:true},
    {name:'DataConsegna', index:'DataConsegna', label:'', width:80, editable:true, hidden: true},
    {name:'DataConsegna_UI', index:'DataConsegna', label:'DataCons', width:80, editable:true, edittype:'custom', editoptions:{
            custom_element: $.jgrid.getMethod('dateEdit_Elem'),
            custom_value: $.jgrid.getMethod('dateEdit_Value'),
            dateFormat: '<?=$txt[91]?>'}},
    {name:'ProdottoAcquisto', label: 'PA', index:'ProdottoAcquisto', width:80,align:"center",editable:true},
    {name:'CommessaCompetenza', label: 'CommessaComp', index:'CommessaCompetenza', width:80,align:"center",editable:true},
    {name:'cmd', label: 'Comandi', index:'cmd', width:70,align:"center"}
```

I dati che ci servono sono contenuti in tContratti_Dettaglio inizialmente vuota.
Viene aggiornata usando una stored procedure che chiamiamo *spUpdateContratti*, che non fa altro che caricare i dati nelle tre tabelle create (tContratti_Dettaglio, tContratti_Testate, tContratti_Testo)

La stored procedure consiste nel riempito delle tabelle nel caso non contenessero il CodiceContratto che viene passato.

```
--
create PROCEDURE [dbo].[spUpdateContatti]
(  @CodiceContratto varchar (6) )
AS
IF @CodiceContratto in (select a.NumeroOrdine as  CodiceContratto  from [dbo].[tContratti_Dettaglio] as a )
    IF @CodiceContratto in (select a.NumeroOrdine  from [dbo].[tContratti_Testi] as a )
            IF @CodiceContratto in (select a.NumeroOrdine as  CodiceContratto  from [dbo].[tContratti_Testate] as a )
                return 3
            else --IF @CodiceContratto not in (select substring(a.NumeroOrdine,2,6) as  CodiceContratto from [dbo].[tContratti_Testate] as a )
                insert into tContratti_Testate(NumeroOrdine, CodiceBP, IndirizzoFatturazione, IndirizzoSpedizione, IndirizzoInstallazione,DataOrdine,DataConsegna,
                TipoOrdine,TestoTestata,RiferimentoOrdineCLiente,NonUsato1,CodiceImpiegato,NonUsato2,NonUsato3,DataOrdine1,valuta,cambio)
                select substring(a.NumeroOrdine,2,6) as NumeroOrdine, substring(a.CodiceBP,2,6) as CodiceBP, IndirizzoFatturazione, IndirizzoSpedizione, IndirizzoInstallazione,DataOrdine,DataConsegna,
                    case when a.CodiceLinea='MOD'
                    THEN (select tipomodifica from openquery(PROD,'SELECT * FROM VWEB_TIPIORDINIBP' ) as q
                    where q.T$BPID= substring(a.CodiceBP,2,6))
                    else  (select tipovendita from openquery(PROD,'SELECT * FROM VWEB_TIPIORDINIBP' ) as q
                    where q.T$BPID= substring(a.CodiceBP,2,6))
                    end TipoOrdine,
                substring(TestoTestata,2,len(TestoTestata) - 2) as TestoTestata,RiferimentoOrdineCliente,NonUsato1,substring(a.CodiceImpiegato,2,6) as CodiceImpiegato,
                NonUsato2,NonUsato3,DataOrdine1,valuta,cambio
                from openquery(SRVLYRA,'SELECT * FROM Lyra8aan.dbo.VIEW_ORDER_TESTATA2' ) as a
                where @CodiceContratto=substring(a.NumeroOrdine,2,6)

    else --IF @CodiceContratto not in (select substring(a.NumeroOrdine,2,6) as  CodiceContratto from [dbo].[tContratti_Testate] as a )
    insert into tContratti_Testi(NumeroOrdine,ValoreFisso,TestoTestata)
    select substring(a.NumeroOrdine,2,6) as NumeroOrdine,ValoreFisso,substring(TestoTestata,2,len(TestoTestata) - 2) as TestoTestata
    from openquery(SRVLYRA,'SELECT * FROM Lyra8aan.dbo.VIEW_ORDER_HEADER_TESTO' ) as a
    where @CodiceContratto=substring(a.NumeroOrdine,2,6)
```

Inserimento nella tabella tContratti_Dettaglio: da notare che oltre ai dati contenuti nella vista VIEW_ORDER_ROWS2 bisogna inserire altre righe provenienti da un'altra vista che contiene righe di default necessarie per il corretto funzionamento dei file che verranno creati (spiegazione nel dettaglio in seguito)

```
else --IF @CodiceContratto not in (select substring(a.NumeroOrdine,2,6) as  CodiceContratto from [dbo].[tContratti_Testate] as a )
    insert into tContratti_Dettaglio(NumeroOrdine, Posizione, CollegContratto, Item, categoria,TipoArticolo,quantita,ImportoRiga,sconto,DataConsegna,descrizione,ProdottoAcquisto,
    CommessaCompetenza,CostoAssemblaggio,CostoMaterialiMeccanici,CostoMaterialiElettrici,CostoProgMeccanica,CostoProgElettrica,ImportoTotale)
    select substring(a.NumeroOrdine,2,6) as NumeroOrdine, Posizione, CollegContratto,
    case when
    substring(Item,2,len(Item) -2) IN (SELECT Item from tCodici_Corrispondenze)
    then (select z.Corrispondenza from tCodici_Corrispondenze as z
            where z.Item = substring(a.Item,2,len(a.Item) -2))
    else substring(Item,2,len(Item) -2)
    end  Item,
    substring(categoria,2,len(categoria) -2) as categoria,substring(TipoArticolo,2,len(TipoArticolo) -2) as TipoArticolo,quantita,ImportoRiga,sconto,
    convert (datetime,DataConsegna,105) AS DataConsegna,substring(descrizione,2,len(descrizione) - 2) as descrizione,substring(ProdottoAcquisto,2,len(ProdottoAcquisto) -2) AS ProdottoAcquisto,
    substring(CommessaCompetenza,2,len(CommessaCompetenza) -2) as CommessaCompetenza,   cast(CostoAssemblaggio as decimal(38,2)) as CostoAssemblaggio ,CostoMaterialiMeccanici,CostoMaterialiElettrici,
    CostoProgMeccanica,CostoProgElettrica, (quantita * ImportoRiga) as ImportoTotale
    from openquery(SRVLYRA,'SELECT * FROM Lyra8aan.dbo.VIEW_ORDER_ROWS2' ) as a
    where @CodiceContratto=substring(a.NumeroOrdine,2,6)

    insert into tContratti_Dettaglio( NumeroOrdine, Posizione, CollegContratto, Item, categoria,TipoArticolo,quantita,ImportoRiga,sconto,DataConsegna,descrizione,ProdottoAcquisto,
    CommessaCompetenza,CostoAssemblaggio,CostoMaterialiMeccanici,CostoMaterialiElettrici,CostoProgMeccanica,CostoProgElettrica,ImportoTotale)
    select @CodiceContratto  as NumeroOrdine, Posizione, CollegContratto, substring(Item,2,len(Item) -2) as Item,substring(categoria,2,len(categoria) -2) as categoria,
    substring(TipoArticolo,2,len(TipoArticolo) -2) as TipoArticolo,quantita,ImportoRiga,sconto,
    (select top 1 DataConsegna from openquery(SRVLYRA,'SELECT * FROM Lyra8aan.dbo.VIEW_ORDER_ROWS2' ) as a
    where @CodiceContratto=substring(a.NumeroOrdine,2,6) ) as DataConsegna,substring(descrizione,2,len(descrizione) - 2) as descrizione,
    substring(ProdottoAcquisto,2,len(ProdottoAcquisto) -2) AS ProdottoAcquisto,substring(CommessaCompetenza,2,len(CommessaCompetenza) -2) as CommessaCompetenza,
    cast(CostoAssemblaggio as decimal(38,2)) as CostoAssemblaggio ,CostoMaterialiMeccanici,CostoMaterialiElettrici,CostoProgMeccanica,CostoProgElettrica,(quantita * ImportoRiga) as ImportoTotale
    from openquery(SRVLYRA,'SELECT * FROM SRVLYRA.BT.dbo. tLyra2Baan_ServiziDefault' ) as a
    where Posizione not in (select Posizione from tContratti_Dettaglio where NumeroOrdine = @CodiceContratto)
```

Per richiamare la stored procedure usiamo il seguente comando:

```
switch ($a)
{
    case 1:
        //  Prima i fare la SELECT dalla tabella tContratti_Dettaglio eseguiamo una stored procedure per verificare se il contratto è presente altrimenti lo aggiungiamo
        sqlsrv_query($OConnDB,  tsql: "EXEC spUpdateContatti '" . $CC . "';") or sqlsrv_str_error( sFile: __FILE__,  sLine: __LINE__);
```

sqlsrv_query() ,funzione di PHP che serve per eseguire su database la cui connessione è indicata dalla prima variabile, in caso di errore viene eseguita *sqlsrv_str_error* che mostra una finestra con l' errore generato.
Quando viene ricaricato l'url con *a*=1 allora viene fatto l' EXEC della stored procedure in questione, *a* questo punto possiamo fare la *select* sapendo già che il contratto sarà contenuto nelle tabelle.

```
sqlsrv_query($OConnDB, tsql: "EXEC spUpdateContatti '" . $CC . "';") or sqlsrv_str_error( sFile: __FILE__, sLine: __LINE__ );
$resultData = jqGridRead(
    $OConnDB,
    sSQL: "SELECT distinct NumeroOrdine + Posizione as pk,
           NumeroOrdine,Posizione,CollegContratto,TipoArticolo,categoria,Item,descrizione,
           quantita,cast(CostoMaterialiMeccanici as decimal(38,2)) as CostoMaterialiMeccanici,
           cast(CostoMaterialiElettrici as decimal(38,2)) as CostoMaterialiElettrici,
           cast(CostoAssemblaggio as decimal(38,2)) as CostoAssemblaggio,
           CostoProgMeccanica,
           CostoProgElettrica,
           cast(ImportoRiga as decimal(38,2)) as ImportoRiga,sconto,
           CONVERT(VARCHAR, DataConsegna ,  " .  $txt[120] . ") AS DataConsegna,
           ProdottoAcquisto,CommessaCompetenza,cast((quantita * ImportoRiga)as decimal(10,2)) as ImportoTotale
           from tContratti_Dettaglio   where NumeroOrdine=$CC  " ,
    array('pk','NumeroOrdine', 'Posizione', 'CollegContratto', 'TipoArticolo', 'categoria','Item',
           'descrizione','quantita','CostoMaterialiMeccanici','CostoMaterialiElettrici','CostoAssemblaggio',
           'CostoProgMeccanica','CostoProgElettrica','ImportoRiga','sconto','DataConsegna','ProdottoAcquisto',
           'CommessaCompetenza','ImportoTotale'),
    array('NumeroOrdine','Posizione', 'CollegContratto', 'TipoArticolo', 'categoria','Item','descrizione',
           'quantita','CostoMaterialiMeccanici','CostoMaterialiElettrici','CostoAssemblaggio','CostoProgMeccanica',
           'CostoProgElettrica','ImportoRiga','sconto','DataConsegna','ProdottoAcquisto','CommessaCompetenza',
           'ImportoTotale'), AOrderFields: null, namedField: true
);
//CostoProgMeccanica,CostoProgElettrica SONO DUE NUMERI INTERI ANCHE SE COSTI
```

A questo punto avremo la griglia con tutti i valori necessari.

Il passo successivo è permettere all' utente di modificare e aggiungere le righe secondo le specifiche:

- La colonna Posizione non dovrà mai essere modificata ma potrà essere aggiunta come campo di una nuova riga utilizziamo la funzionalità di jqGrid in questo modo: usando le funzioni ondblClickRow e afterInsertRow

```
gridComplete: function() {
    var ids = $("#mcTable").jqGrid('getDataIDs');
    if(ids.length==0)
        alert("Contratto vuoto");

    var totContratto=parseFloat(0).toFixed(2);

    for(var i=0;i < ids.length;i++) {
        var row_id = ids[i];
        var rowData = $(this).jqGrid ('getRowData', row_id);
        totContratto=parseFloat(totContratto)+ parseFloat(rowData['ImportoTotale']);
    }
   // $("#NumeroOrdine").text("Numero Ordine: "+ NumeroOrdine ) ;
    $("#TotContratto").text("Importo totale contratto: "+ totContratto.toFixed(2) + "€") ;
    //$("#CodiceBP").text("Codice BP : "+ Cod ) ;
},
ondblClickRow: function() {
    $("#mcTable").jqGrid('setColProp', 'Posizione', { editable: false });
},
afterInsertRow: function(){
    $("#mcTable").jqGrid('setColProp', 'Posizione', { editable: true });
},

caption: "Dettagli Contratto",

});
```

Definisco i campi in modalità editable e add in questo modo sempre utilizzando le funzionalità di jqGrid:

```
$("#mcTable").jqGrid('navGrid',"#mcTools", {search: false, edit:false, add:false, del:false});
$("#mcTable").jqGrid('inlineNav',"#mcTools", {del:false, edit:true, add:true});   //se lo metto
```

A questo punto posso aggiungere le righe alla mia tabella, ma dobbiamo fare in modo che vengano salvate anche nel database.Anche in questo caso usiamo una funzionalità di jqGrid in particolare il campo editurl in modo da cambiare la variabile a=2 per distinguere il momento del caricamento della tabella e il momento dell' edit o dell' add.

```
editurl: '?a=2&strCodiceContratto=<?=$CC?>',
```

Nello switch aggiungiamo il caso in cui a=2 e dividiamo i casi di edit e add

```
case 2:
    if ($_POST["oper"] == 'add')
    {

        $sSQL = "INSERT INTO tContratti_Dettaglio (NumeroOrdine,Posizione,CollegContratto,Item,categoria,TipoArticolo,
                quantita,ImportoRiga,sconto,DataConsegna,descrizione,ProdottoAcquisto,CommessaCompetenza,CostoAssemblaggio,
                    CostoMaterialiMeccanici,CostoMaterialiElettrici,CostoProgMeccanica,CostoProgElettrica,ImportoTotale)
                 VALUES ('" .  $CC ."','" . POSTsDB( sParam: "Posizione") ."','" . POSTsDB( sParam: "CollegContratto") ."','" .
            POSTsDB( sParam: "Item") ."','" . POSTsDB( sParam: "categoria") ."','" . POSTsDB( sParam: "TipoArticolo") ."','" .
            POSTsDB( sParam: "quantita") ."','" . POSTsDB( sParam: "ImportoRiga") ."','" . POSTsDB( sParam: "sconto") ."','" .
            POSTsDB( sParam: "DataConsegna") ."','" . POSTsDB( sParam: "descrizione") ."','" . POSTsDB( sParam: "ProdottoAcquisto") .
            "','" . POSTsDB( sParam: "CommessaCompetenza") ."','" . POSTsDB( sParam: "CostoAssemblaggio") ."','" .
            POSTsDB( sParam: "CostoMaterialiMeccanici") ."','" . POSTsDB( sParam: "CostoMaterialiElettrici") ."','" .
            POSTsDB( sParam: "CostoProgMeccanica") ."','" . POSTsDB( sParam: "CostoProgElettrica") ."'," .
            POSTsDB( sParam: "ImportoRiga"). "*" . POSTsDB( sParam: "quantita") .")";
        $result= sqlsrv_query($OConnDB, $sSQL) ; //or sqlsrv_str_error(__FILE__, __LINE__);
        if($result == false)
        {
            $errore=sqlsrv_errors();
            if($errore[0]['code'] == 2627)
            {
                echo($txt[792]);
            }
            else sqlsrv_str_error( sFile: __FILE__, sLine: __LINE__);
        }
```

```php
else if ($_POST["oper"] == 'edit')
{
    $nID = POSTn( sParam: "id");
    if ($nID>0)
    {
        $sSQL = "UPDATE tContratti_Dettaglio SET " .
            " CollegContratto = '" . POSTsDB( sParam: "CollegContratto") . "'," .
            " TipoArticolo = '" . POSTsDB( sParam: "TipoArticolo") . "'," .
            " quantita = '" . POSTsDB( sParam: "quantita") . "'," .
            " Item = '" . POSTsDB( sParam: "Item") . "'," .
            " categoria = '" . POSTsDB( sParam: "categoria") . "'," .
            " ImportoRiga = '" . POSTsDB( sParam: "ImportoRiga") . "'," .
            " sconto = '" . POSTsDB( sParam: "sconto") . "'," .
            " DataConsegna = '" . POSTsDB( sParam: "DataConsegna") . "'," .
            " descrizione = '" . POSTsDB( sParam: "descrizione") . "'," .
            " ProdottoAcquisto = '" . POSTsDB( sParam: "ProdottoAcquisto") . "'," .
            " CommessaCompetenza = '" . POSTsDB( sParam: "CommessaCompetenza") . "'," .
            " CostoAssemblaggio = '" . POSTsDB( sParam: "CostoAssemblaggio") . "'," .
            " CostoMaterialiMeccanici = '" . POSTsDB( sParam: "CostoMaterialiMeccanici") . "'," .
            " CostoMaterialiElettrici = '" . POSTsDB( sParam: "CostoMaterialiElettrici") . "'," .
            " CostoProgMeccanica = '" . POSTsDB( sParam: "CostoProgMeccanica") . "'," .
            " CostoProgElettrica = '" . POSTsDB( sParam: "CostoProgElettrica") . "'," .
            "ImportoTotale = " . POSTsDB( sParam: "ImportoRiga"). "*" .  POSTsDB( sParam: "quantita")  .
            " WHERE  (NumeroOrdine + Posizione)= " . $nID; /*pk è la chiave della tabella tContratti_De
        sqlsrv_query($OConnDB, $sSQL) or (sqlsrv_str_error( sFile: __FILE__,  sLine: __LINE__));
```

A questo punto si può realmente modificare il database utilizzando le due operazioni edit e add.

AGGIUNGERE CAMPI DELLA TESTATA BAAN

Innanzitutto definiamo due array che conterranno i dati che saranno selezionati dalla query :

```php
$ATestataBaan=sqlsrv_get_row($OConnDB, sSQL: "select * from openquery(PROD,'select  T\$Desc,T\$NAMA,t\$orno,a.T\$CPRJ from tcdsls400900 a,' ||
                                                                            'ttccom100900 b  where a.T\$BPID=b.T\$BPID AND T\$CONT=''".$CC."''')") ;
$ATestata = sqlsrv_get_row($OConnDB,  sSQL: "select * from tContratti_Testate where NumeroOrdine = $CC ");
```

La funzione *sqlsrv_get_row* permette di convertire i dati selezionati da una query in un parametro array in modo che possano essere utilizzati con facilità

```php
// Restituisce la prima riga di valori come array
function sqlsrv_get_row($OConn, $sSQL)
{
    $result = null;

    $rs = sqlsrv_query($OConn, $sSQL) or sqlsrv_str_error( sFile: __FILE__,  sLine: __LINE__);
    $row_rs = sqlsrv_fetch_array($rs,  fetch_type: SQLSRV_FETCH_ASSOC);

    return $row_rs;
}
```

L' utilizzo dell'array per costruire la testata è davvero semplice:

```
<div id="testataContratto" class="clear">
    <div class="left" id="NumeroOrdine" >Numero Ordine: <span class="bold"><?= $ATestata['NumeroOrdine'] ?></span></div>
    <div class="left" ><?= $ATestataBaan['T$DESC'] ?></div>
    <div class="left" >CodiceBP: <?= $ATestata['CodiceBP'] ?></div>
    <div class="left">
        <form action="contratti_lyra_dettaglio.php?strCodiceContratto=<?= $CC ?>" method="post" class="formMan" name="trasferisci">
                <input class="formMan_submit" type="submit" value="trasferisci"/>
                <input type="hidden" name="trasferisci" value="1"/></form>
    </div>
    <div class="left" ><?= $ATestataBaan['T$NAMA'] ?></div>
    <div class="left" id="IndirizzoSpedizione" >Indirizzo di Spedizione <?= $ATestata['IndirizzoSpedizione'] ?></div>
    <div class="left" >Ordine Vendita: <?= $ATestataBaan['T$ORNO'] ?></div>
    <div class="left">
        <form action="contratti_lyra_dettaglio.php?strCodiceContratto=<?= $CC ?>" method="post" class="formMan" name="reset">
            <input class="formMan_submit" type="submit" value="reset" onclick="  $.dlgConfirm('<?=$txt[795]?>', 'document.reset.submit()'); return false;"/>
            <input type="hidden" name="reset" value="1"/> </form>
    </div>
    <div class="left">valuta: <? echo $ATestata['valuta']; ?></div>
    <div class="left">Indirizzo di Fatturazione <?= $ATestata['IndirizzoFatturazione'] ?></div>
    <div class="left">Commessa Principale: <?= $ATestataBaan['T$CPRJ'] ?></div>
    <div class="left">
          <form action="contratti_lyra_totali.php?strCodiceContratto=<?= $CC ?>" method="post" class="formMan" name="totali" >
           <input class="formMan_submit" type="button" value="totali" onclick="window.open('contratti_lyra_totali.php?strCodiceContratto=<?= $CC ?>',
                'totali' ,'width=800,height=640,location=no,menubar=no,resizable=yes,scrollbars=yes,titlebar=yes')"/></form>
    </div>
    <div class="left" id="TipoOrdine">Tipo Ordine: <?= $ATestata['TipoOrdine'] ?></div>
    <div class="left" id="TotContratto">Importo totale contratto:</div>
```

Ricordiamo che abbiamo definito le grandezze(altezza e lunghezza) per ogni div all' interno di questo id=testataContratto

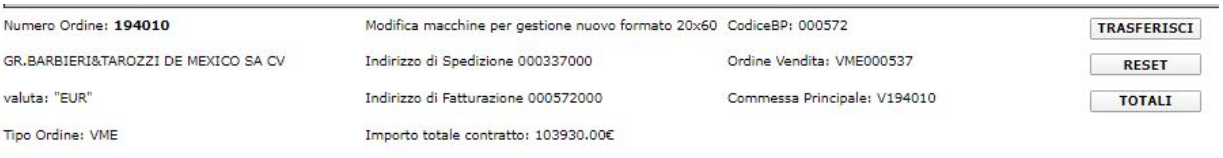
```
#testataContratto{
    width: 1600px;
    height: 160px;
}

#testataContratto div{
    width:400px;

    height:40px;

}
```

Cosi facendo per ogni riga avremo quattro div nel nostro caso saranno tre campi e il quarto campo sarà sempre un bottone(approfondiremo nel seguito) esempio testata



Nella stessa pagina dettaglio inserire i seguenti comandi:

- reset del contratto
- trasferimento del contratto a Baan
- calcolo dei totali

Abbiamo già definito i tre bottoni in HTML

1.TRASFERIMENTO

```
<div class="left">
    <form action="contratti_lyra_dettaglio.php?strCodiceContratto=<?= $CC ?>" method="post" class="formMan" name="trasferisci">
            <input class="formMan_submit" type="submit" value="trasferisci"/>
            <input type="hidden" name="trasferisci" value="1"/></form>
</div>
```

```
$trasferisci=POSTn( sParam: "trasferisci");
        if($trasferisci==1){
            $result1 =  sqlsrv_query($OConnDB,  tsql: " EXEC [spTrasferisciContratto] '" . $CC . "';")
                        or sqlsrv_str_error( sFile: __FILE__,  sLine: __LINE__);
        if( sqlsrv_next_result($result1) == null)
            $MSG=$txt[793];
        else
            $MSG=$txt[794];
        }
```

In HTML abbiamo definito il value=1 quindi nel momento in cui verrà fatto il click l'input del bottone assumerà quel valore e verrà eseguito il codice corrispondente nel momento in cui si arriverà al corrispondente codice in php.
Eseguiamo la stored procedure spTrasefrisciContratto : Questa stored procedure non fa altro che trasformare i campi contenuti nelle tabelle nei quattro file che serviranno al trasferimento in Baan. I quattro file sono :

1. Righe.txt
2. Testata.txt
3. Testo.txt
4. CODICE.lst (dove codice corrisponde al NumeroOrdine presente nelle tabelle)

2. RESET

```
<div class="left">
    <form action="contratti_lyra_dettaglio.php?strCodiceContratto=<?= $CC ?>" method="post" class="formMan" name="reset">
        <input class="formMan_submit" type="submit" value="reset" onclick="  $.dlgConfirm('<?=$txt[795]?>',
                'document.reset.submit()'); return false;"/>
        <input type="hidden" name="reset" value="1"/> </form>
</div>
```

Nel caso si faccia click nel bottone reset si esegue la store procedure spUpdateBaan

```
$reset=POSTn("reset");
```

attraverso il comando **Postn** possiamo ricavare il valore che assume la variabile reset in HTML se è stato fatto click allora assumerà valore 1 e verrà eseguita la store procedure **spUpdateBaan**

```
if($reset==1){

        $result1 =  sqlsrv_query($OConnDB,  tsql: " EXEC [spUpdateBaan] '" . $CC . "';")
        or sqlsrv_str_error( sFile: __FILE__,  sLine: __LINE__);
    if( sqlsrv_next_result($result1) == null) echo "NON UPDATE";
```

Il bottone totali non fa altro che aprire una finestra popup nel momento del click, il cui indirizzo sarà collegato al CoidceContratto corrispondente.

```
<div class="left">
      <form action="contratti_lyra_totali.php?strCodiceContratto=<?= $CC ?>" method="post" class="formMan" name="totali" >
       <input class="formMan_submit" type="button" value="totali"
             onclick="window.open('contratti_lyra_totali.php?strCodiceContratto=<?= $CC ?>',
              'totali' ,'width=800,height=640,location=no,menubar=no,resizable=yes,scrollbars=yes,titlebar=yes')"/></form>
</div>
```

4.3.5 Creare una nuova pagina web di tipo popup

Creare una nuova pagina web di tipo popup che si apra cliccando sul comando “totali”. In questa pagina mostrare una griglia “Somma per reparto”, ottenuta sommando gli importi delle righe raggruppate per int(numero di riga / 1000). Es:

	A	B	C	E
1	**Reparto**	**Somma Importi unitari**	**Somma importi Totali**	**Somma importi abbinamenti**
2	0	€ 264.998,00	€ 264.998,00	€ 264.998,00
3	1	€ 1.306.008,00	€ 1.306.008,00	€ 1.311.108,00
4	2	€ 152.859,00	€ 152.859,00	€ 152.859,00
5	5	€ 598.232,00	€ 598.232,00	€ 598.232,00
6	6	€ 144.526,00	€ 146.310,00	€ 146.310,00
7	900	€ 1.940.000,00	€ 1.940.000,00	€ 1.940.000,00
8	950	€ 0,00	€ 0,00	€ 0,00

Creiamo una nuova pagina php che chiamiamo contratti_lyra_totali.php che non fa altro che raggruppare le righe in reparti: un reparto è rappresentato dalla prima cifra del campo Posizione per ricavarlo dividiamo per 1000, e calcoliamo i totali in base a questo valore. Riutilizziamo sempre lo stesso metodo jqGrid per creare la tabella.

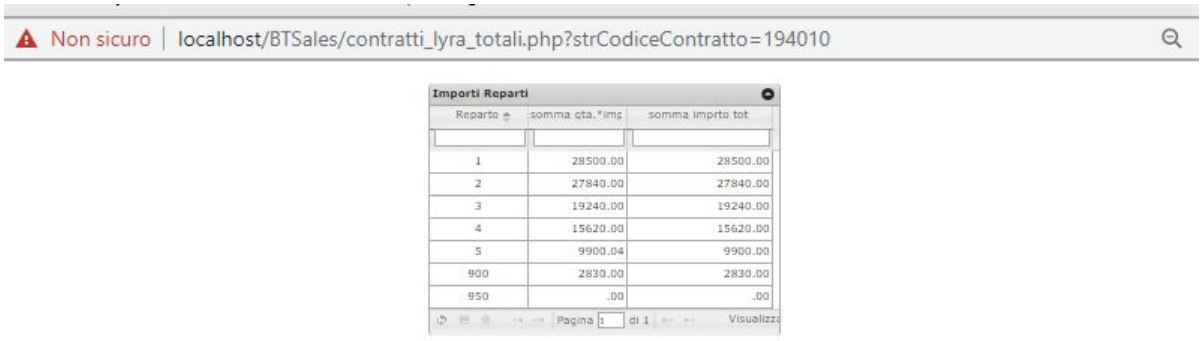
```php
$resultData = jqGridRead(
    $OConnDB,
     sSQL: "select (a.Posizione / 1000 ) as Reparto,
                sum (a.quantita* cast(a.ImportoRiga as decimal(38,2))) as qtaImportoUnit,
                sum (cast(a.ImportoTotale as decimal(38,2))) as SommaTot
           from tContratti_Dettaglio as a
           where NumeroOrdine = '" . $CC . "' group by (a.Posizione / 1000 )"
    ,
    array('Reparto', 'qtaImportoUnit', 'SommaTot'),
    array('Reparto', 'qtaImportoUnit', 'SommaTot'),  AOrderFields: null,  namedField: true

);
```

La tabella avrà tre campi

```javascript
$(document).ready(function()
{
    $("#mcTable").jqGrid({
        url:'?a=1&strCodiceContratto=<?=$CC?>',
        datatype: "json",
        colModel:[
            {name:'Reparto', label: 'Reparto', index:'Reparto', width:100,align:"center"},
            {name:'qtaImportoUnit', label: 'somma qta.*importo unit', index:'somma qta.*importo unit', width:100,align:"right"},
            {name:'SommaTot', label:'somma imprto tot', index:'SommaTot', width:150,align:"right"}
            // {name:'cmd', label: 'Comandi', index:'cmd', width:70,align:"center"}
        ],
        rowNum: 1000,
        pager: '#mcTools',
        sortname: 'Reparto',
        viewrecords: true,
        sortorder: "asc",
```

Il risultato finale si presenta in questo modo:



A questo punto dobbiamo fare delle modifiche alla logica di edit e add:

- Spieghiamo cosa significa la colonna Collegamento: se viene inserita una riga con collegamento valorizzato, rappresenta il numero Posizione di un'altra riga, ossia vuol dire che quella riga rappresenta un accessorio di quell'oggetto a cui fa riferimento.
- Quindi quando viene fatta una edit di una riga modificando la cella Collegamento bisognerà eseguire alcune operazioni:
  1. Modificare TipoArticolo in Accessiorio='A'
  2. Aggiornare la colonna ImportoTotale della riga a cui fa riferimento in quanto bisogna conteggiare anche l' importo dell'accessorio.
  3. Modificare la cella ImportoTotale dell'accessorio in zero ossia viene lasciato solo il valore di importo singolo senza conteggiare ImportoTotale.

Per affrontare queste modifiche bisognerà innanzitutto costruire delle query adeguate e in seguito decidere in quale momento eseguirle.

ANALISI DEL CASO ADD

Nel caso in cui venga aggiunta una nuova riga dopo l' esecuzione della query di insert decido di modificare il campo :

```
$nID = $CC + POSTsDB("Posizione");
```

che rappresenta l' id della riga corrente in questo modo in caso in cui il Collegamento del contratto non sia vuoto posso modificare il TipoArticolo in 'A' anche nel caso in cui l' utente non l' abbia fatto. E di conseguenza cambio anche la colonna ImportoTotale uguale a zero

```
$nID = $CC + POSTsDB( sParam: "Posizione");

$sSQL = "UPDATE tContratti_Dettaglio SET " .
    " TipoArticolo = (  select case when a.CollegContratto <> 0 then  'A'
                        else (select TipoArticolo from [tContratti_Dettaglio]
                              WHERE  (NumeroOrdine + Posizione)= " . $nID." )
                        end TipoArticolo
                        FROM [dbo].[tContratti_Dettaglio] as a
                        WHERE  (NumeroOrdine + Posizione)= " . $nID ."
                    ),
                    ImportoTotale=0" .
    " WHERE  (NumeroOrdine + Posizione)= " . $nID; /*pk è la chiave della tabella tCo
sqlsrv_query($OConnDB, $sSQL) or (sqlsrv_str_error( sFile: __FILE__, sLine: __LINE__));
```

Una volta modificata la riga appena aggiunta dobbiamo modificare anche la riga collegata:
Dobbiamo fare il conteggio dell'ImportoTotale della riga del contratto appena aggiunta e quella già esistente per farlo salviamo ID della riga da modificare come CollegContratto, in caso sia zero non verrà modificata nessuna riga.
Faccio l' update dei campi in base a questo ID.

```
// se il CollegContratto è zero l' update seguente non modificherà nessuna riga
$nID=$CC + POSTsDB( sParam: "CollegContratto");

$sSQL = "UPDATE tContratti_Dettaglio SET " .
    " ImportoTotale = (  select a.ImportoTotale + ". POSTsDB( sParam: "ImportoRiga") ."*". POSTsDB( sParam: "quantita")."
                         from [tContratti_Dettaglio] a
                         WHERE  (NumeroOrdine + Posizione)= " . $nID." ) " .
    " WHERE  (NumeroOrdine + Posizione)= " . $nID; /*pk è la chiave della tabella tContratti_Dettaglio per distinguere le
sqlsrv_query($OConnDB, $sSQL) or (sqlsrv_str_error( sFile: __FILE__,  sLine: __LINE__));
```

ANALISI DEL CASO EDIT

La prima cosa da fare nella edit dopo l' update è quella di salvare il campo ImportoTotale:

```
$Importo=POSTsDB("ImportoRiga") *  POSTsDB("quantita");
```

```
$sSQL = "UPDATE tContratti_Dettaglio SET " .
    " TipoArticolo = (  select case when a.CollegContratto <> 0 then  'A'
                        else (select TipoArticolo from [tContratti_Dettaglio]
                        WHERE  (NumeroOrdine + Posizione)= " . $nID."
                        AND NumeroOrdine=".$CC.")
                        end TipoArticolo
                        FROM [dbo].[tContratti_Dettaglio] as a
                        WHERE  (NumeroOrdine + Posizione)= " . $nID ."
                        AND NumeroOrdine=".$CC."
                ),ImportoTotale=( select case when a.CollegContratto <> 0 then 0
                        else (select ImportoTotale from tContratti_Dettaglio
                        WHERE  (NumeroOrdine + Posizione)= " . $nID . "
                        AND NumeroOrdine=".$CC.")
                        END ImportoTotale
                        FROM [dbo].[tContratti_Dettaglio] as a
                        WHERE  (NumeroOrdine + Posizione)= " . $nID . "
                        AND NumeroOrdine=".$CC." )
                " .
    " WHERE  (NumeroOrdine + Posizione)= " . $nID."
      AND NumeroOrdine=".$CC; /*pk è la chiave della tabella tContratti_Dettaglio per

sqlsrv_query($OConnDB, $sSQL) or (sqlsrv_str_error( sFile: __FILE__,  sLine: __LINE__));
```

Ripeto lo stesso ragionamento effettuato per l' add .
Salvo di nuovo ID della riga che voglio modificare e aggiorno la colonna ImportoTotale.

```
$nID=$CC + POSTsDB( sParam: "CollegContratto");
// se il CollegContratto è zerol' update seguente non modificherà nessuna riga
$sSQL = "UPDATE tContratti_Dettaglio SET " .
    " ImportoTotale = (  select a.ImportoTotale + ". $Importo ."
                        from [tContratti_Dettaglio] a
                        WHERE  (NumeroOrdine + Posizione)= " . $nID." AND NumeroOrdine=".$CC." ) " .
    " WHERE  (NumeroOrdine + Posizione)= " . $nID ."
AND NumeroOrdine=".$CC; /*pk è la chiave della tabella tContratti_Dettaglio per distinguere le diversi
sqlsrv_query($OConnDB, $sSQL) or (sqlsrv_str_error( sFile: __FILE__,  sLine: __LINE__));
```

5 Problema dell'SQL Injection

Le scelte impostate dal progetto sono state indotte dall'azienda, affinché sia inserito nel Portale preesistente. Alcune scelte potrebbero causare SQL Injection, infatti la scrittura delle query tra virgole è molto insicura, può causare l'inserimento da parte di terzi di codice malevolo.
L' azienda ha inserito una parte di autenticazione iniziale più protetta per garantire l'accesso alle sole persone autorizzate affinché entrino nel Portale, per cui vi è una protezione di ingresso, ma non sempre questa operazione può bastare.
Un attacco SQL Injection consiste nell'esecuzione di una query SQL sfruttando un input non controllato dell'applicazione. Un attacco di questo tipo permette di leggere, modificare, distruggere ed eseguire molte altre operazioni all'interno di un DBMS e in certe occasioni anche di arrivare direttamente al Sistema Operativo.
Nel caso la Web app consenta l'inserimento di input non controllati, ad esempio un carattere escape come l'apostrofo, la query SQL sarà diversa da quella inserita realmente, probabilmente il Web Server restituirà un errore di sintassi, da qui l' attaccante saprà di poter manipolare la query SQL specificando altri parametri.
A rendere l'SQL Injection particolarmente pericoloso è che non richiede l'uso di particolari strumenti, molte tipologie di architettura software adoperate per la realizzazione di siti Web interfacciati con un database sono potenzialmente vulnerabili ad un attacco SQL Injection. Tale rischio non è legato ad un particolare linguaggio, quindi nel nostro caso all'utilizzo di PHP , né a quello di un particolare server DB.
Per questo è importante adottare almeno misure minime di protezione, particolare attenzione deve essere data alla messa in sicurezza del codice che si occupa di moduli di autenticazione e pagine di ricerca che in genere richiedono di interfacciarmi con un database e rappresentano le potenziali porte di accesso.
Un Hacker esperto potrebbe inviare particolari istruzioni attraverso le pagine del sito che prevedono il dialogo con il DB, con l' obiettivo di ottenere un accesso non autorizzato all'applicazione stessa, per recuperare le informazioni o modificarle.

Per prevenire l'iniezione di query su applicazioni Web che interagiscono con un DB è importante la fase di implementazione, ossia una programmazione che preveda un controllo di tutte le potenziali porte di accesso all' archivio di gestione dei dati. Bisognerebbe:

- Prestare attenzione all'utilizzo degli elementi di codice SQL potenzialmente a rischio (virgolette singole e parentesi) che potrebbero essere integrati con opportuni caratteri di controllo sfruttati per usi non autorizzati.
- Usare un estensione di SQL protetta.
- Disattivare la visibilità delle pagine degli errori. Spesso tali informazioni si rilevano preziose per l'attaccante, che potrebbe risalire all'identità e alla struttura del server DB.In PHP si utilizza la funzione **bin2hex()**. Secondo questa prassi l'istruzione in esadecimale, prima di essere eseguita, viene riconvertita in una stringa tramite la funzione **unhex()** di Mysql. Così facendo la sequenza di caratteri riconvertita, non essendo intrepretata come comando, non rappresenta più un potenziale pericolo.
- Un'altra tecnica di difesa da usare per attenuare le conseguenze di un SQL injection sono le query parametrizzate. Prevede tre passaggi principali:

    1. La *preparazione*: viene creato un template di istruzioni SQL (si usano dei parametri segnaposto **?, placeholder** per indicare i valori dei dati da specificare). Il template viene inviato al database tramite funzione prepare().
    2. L'*analisi*: il template dell'istruzione sql ricevuto dal db viene analizzato, compilato e memorizzato senza eseguirlo. Ai segnaposti vengono associate le variabili ed il tipo atteso rispettando l'ordine di posizione.
    3. L'*esecuzione*: l' applicazione esegue le istruzioni dopo aver assegnato i valori ai parametri.

6 Conclusioni

A conclusione di questo elaborato, in cui è stata realizzata una web application per l'importazione dei contratti da Lyra a Baan, è possibile evidenziare i punti di forza della soluzione descritta e gli eventuali sviluppi futuri.

Innanzitutto il punto di forza principale è il eseguire molte operazioni sui contratti senza intervenire dal database quindi ad ogni utente a cui viene data l' autorizzazione per quest'area del Portale è possibile modificare, aggiornare o aggiungere righe ai contratti e trasferirli in modo semplice e veloce senza avere una conoscenza del database.
Molte volte in azienda si perde tempo nel richiedere ad altri colleghi più esperti di eseguire questo tipo di operazioni quindi la realizzazione di questo tipo di applicazioni web semplifica il lavoro e i costi in termini di tempistiche sono minori.

Un possibile sviluppo futuro è la velocità di caricamento del sito, possibile ad esempio attraverso la riscrittura delle viste o delle stored procedure utilizzate in modo che sia più immediata la loro esecuzione.
La difficoltà nel capire quale dato presente nel database era di importanza per la realizzazione del sito e quale no, è stato un ostacolo per lo sviluppo di questa parte del progetto, il tempo passato in azienda è stato più o meno di tre mesi per cui sviluppare anche il testing purtroppo non è stato possibile.
Quindi si potrebbe migliorare questo aspetto del progetto.
Un altro aspetto importante è la sicurezza come discusso nel paragrafo precedente.

Ringraziamenti

Desidero innanzitutto ringraziare la Professoressa Sonia Bergamaschi per i preziosi insegnamenti e le per le ore dedicate alla mia tesi. Inoltre, ringrazio sentitamente Massimo Bertacchini che mi ha seguito durante il mio percorso di Tirocinio.
Ringrazio le mie compagne di corso , in particolare Lin e Nore per i numerosi consigli e il supporto durante i periodi più duri. Infine, ho desiderio di ringraziare con affetto i miei genitori per il sostegno ed il grande aiuto che mi hanno dato. Ringrazio Tarik , Sana e Mehdi che mi sono stati emotivamente vicini ogni momento.
Ringrazio il mio compagno di vita Rabia Farhat, insieme abbiamo affrontato questo cammino, passo dopo passo, giorno dopo giorno, superando tutte le difficoltà, festeggiando insieme ogni vittoria e rialzandoci più forti di prima dopo ogni sconfitta, senza di lui non avrei raggiunto questo traguardo. Nei momenti più duri eri lì a supportarmi sempre. Grazie per tutto quello che hai fatto e che fai per farmi sentire forte ogni giorno .
Infine ringrazio me stessa, per la grinta , per non aver mollato mai.
STAY FOCUSED.

A cura di Khadija Najem

7 Bibliografia e sitografia

[1] https://www.eos-solutions.it/it/news-details/sistemi-erp-vantaggi-aziende.html
[2] http://www.magicsales.it/blog/differenze-crm-erp/
[3] https://www.html.it/pag/31966/apache-web-server-un-introduzione/
[4] http://www.aggaricerche.it/AggaRicerche-DescrzioneProgetti/04ab-Crm-to-baan.html(immagine baan)
[5] https://www.cybercoords.it/index.php/php-e-mysql
[6] https://free-jqgrid.github.io/getting-started/index.html
[7] http://www.trirand.com/jqgridwiki/doku.php?id=wiki:jqgriddocs
[8] https://www.mrwebmaster.it/jquery/lavorare-ajax_10463.html
[9] http://www.trirand.com/jqgridwiki/doku.php?id=wiki:retrieving_data
[10]J.F. Kurose, K.W. Ross, “Reti di Calcolatori e Internet”, Pearson, 2017, 7a edizione
[11] Slide del corso Reti di calcolatori UNIMORE
[12] https://www.cybersecurity360.it/nuove-minacce/sql-injection-come-funziona-e-come-difendersi-dalla-tecnica-di-hacking-delle-applicazioni-web/