MASTER IN TECNOLOGIE INTERNET
ANNO 2011
PROGETTAZIONE DI SITI WEB:
GESTIONE ORDINI DA PIU' SEDI

RELATORE:

ING. STEFANO GHELARDINI

COSA FA IL PROGRAMMA

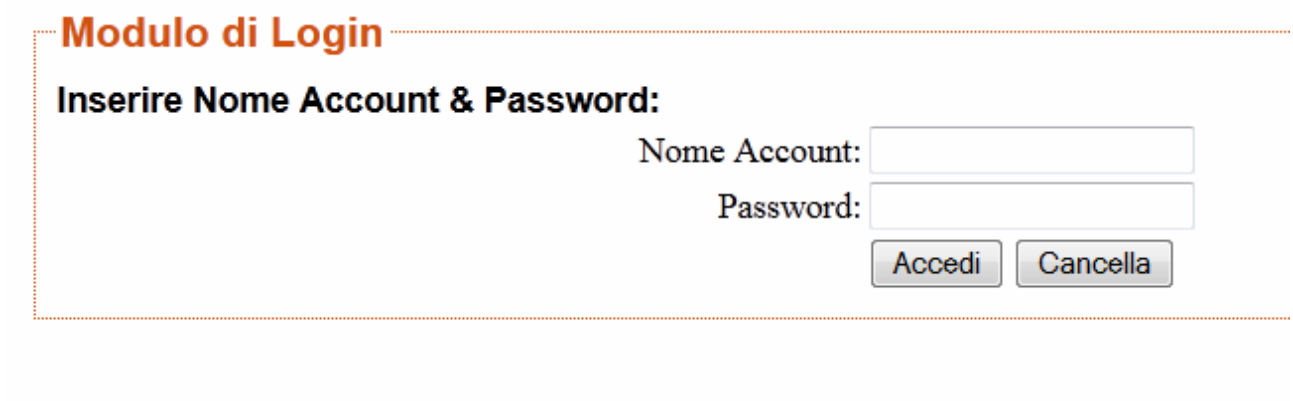
Questo programma, effettua la gestione ordini e clienti di una agenzia di una multinazionale che si occupa di formazione, software, vendita di libri.
All'agenzia fanno capo più zone : La Spezia, Massa Carrara, Lucca, Viareggio.
In ogni zona è prevista una sede dell'agenzia; la sede centrale dell'agenzia si trova a Sarzana .
Ogni sede vende i prodotti della multinazionale più prodotti di altre aziende.
Ogni azienda di cui le varie sedi dell'agenzia sono concessionarie, ha un proprio sistema di gestione ordini diverso dalle altre.

Lo scopo di questo software è quello di avere una gestione complessiva di: prodotti, clienti, ordini, per tutte le sedi dell'agenzia e per tutte le aziende di cui le varie sedi hanno i prodotti.
Per poter ottenere questo risultato, nel software, ad ogni prodotto è associato il nome dell'azienda che lo produce.

Questo programma, è stato realizzato da una singola persona quindi non prevede tutte le funzionalità di un programma di gestione ordini, ma solo una parte di queste; per questa ragione, il programma è stato creato come un nucleo (core) su cui è possibile aggiungere ulteriori parti per poter rispondere a requisiti successivi.

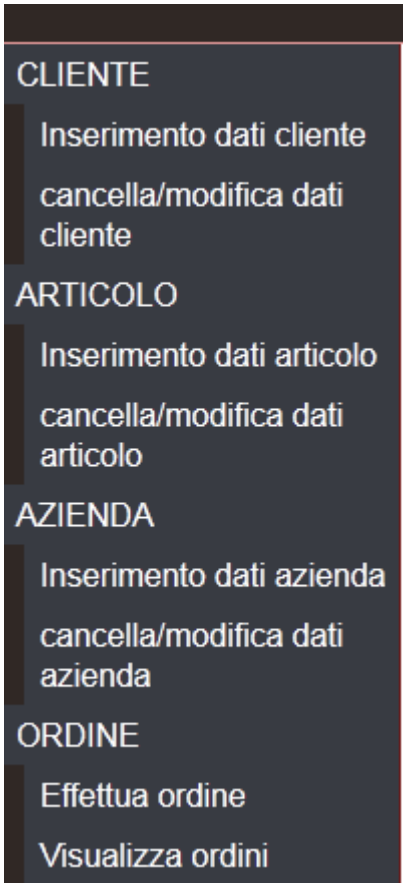
COME FUNZIONA IL PROGRAMMA

quando viene scritto il giusto indirizzo IP, compare il modulo di login che permette l'accesso a tutte le funzionalità del programma



effettuato il login , compare la pagina di entrata che permette, tramite il menu sulla sinistra, di accedere a tutte le funzionalità del programma.

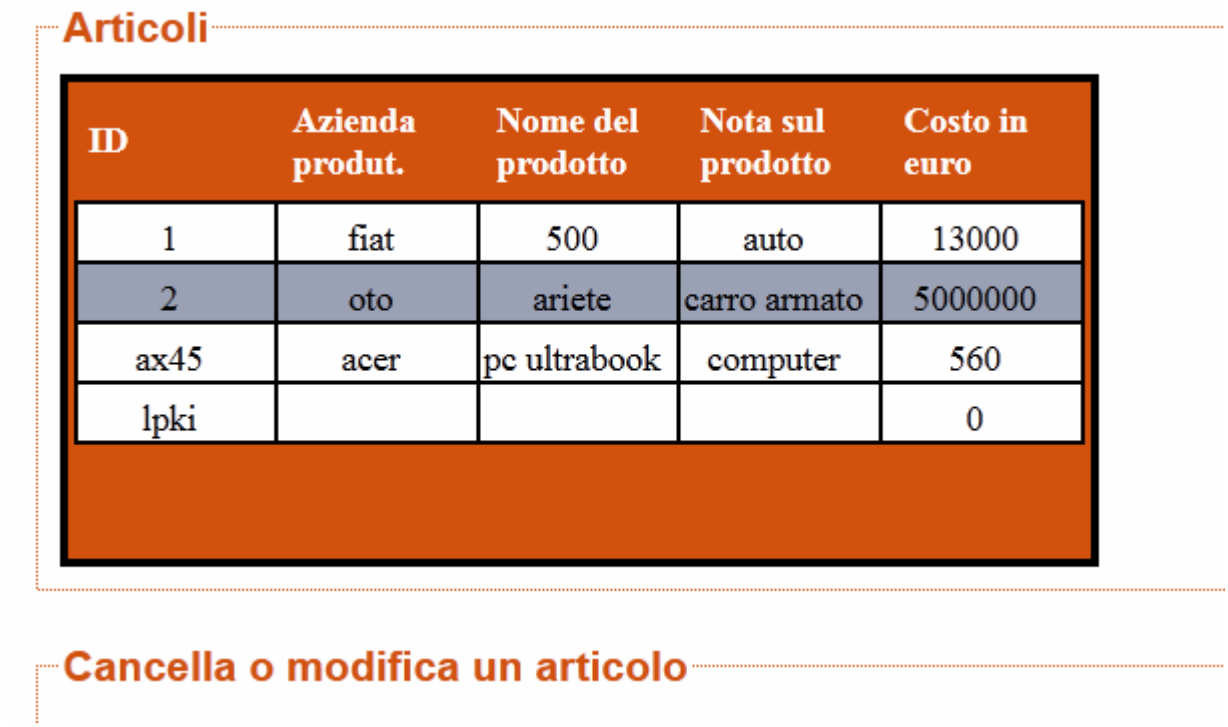
il menu è formato dalle seguenti parti:



Grazie al menù, possiamo accedere alle varie jsp che permettono di inserire i dati di un nuovo cliente, articolo, azienda e di effettuare un ordine.
Cliente, articolo, azienda, possono essere modificati o cancellati se andiamo a un opportuna pagina, però se questi elementi entrano a far parte di un ordine, non possono essere più né modificati né cancellati.

Vediamo ora come funziona il sottomenu relativo a articolo:
1) scriviamo i dati da memorizzare negli opportuni input text.

2) premiamo il pulsante invia



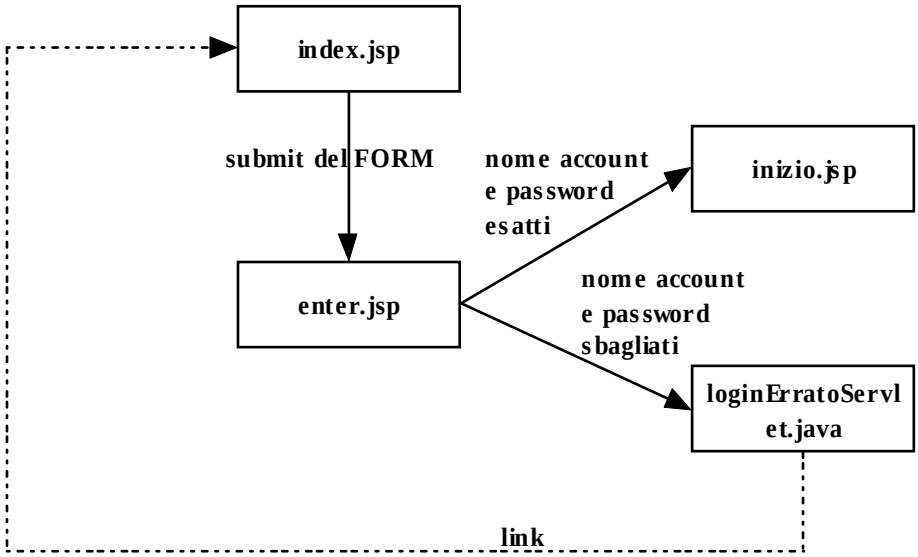
Fatto questo , compare la tabella articoli che contiene l'insieme degli articoli inseriti compreso l'ultimo che abbiamo appena inserito.

Se abbiamo sbagliato a inserire l'articolo , possiamo effettuare la correzione tramite i pulsanti cancella o modifica, che compaiono dopo che selezioniamo una riga della tabella.
La riga della tabella la selezioniamo posizionando il mouse sopra la riga interessata (la riga cambia di colore e diventa azzurrina) e facendo clic con il pulsante sinistro del mouse.

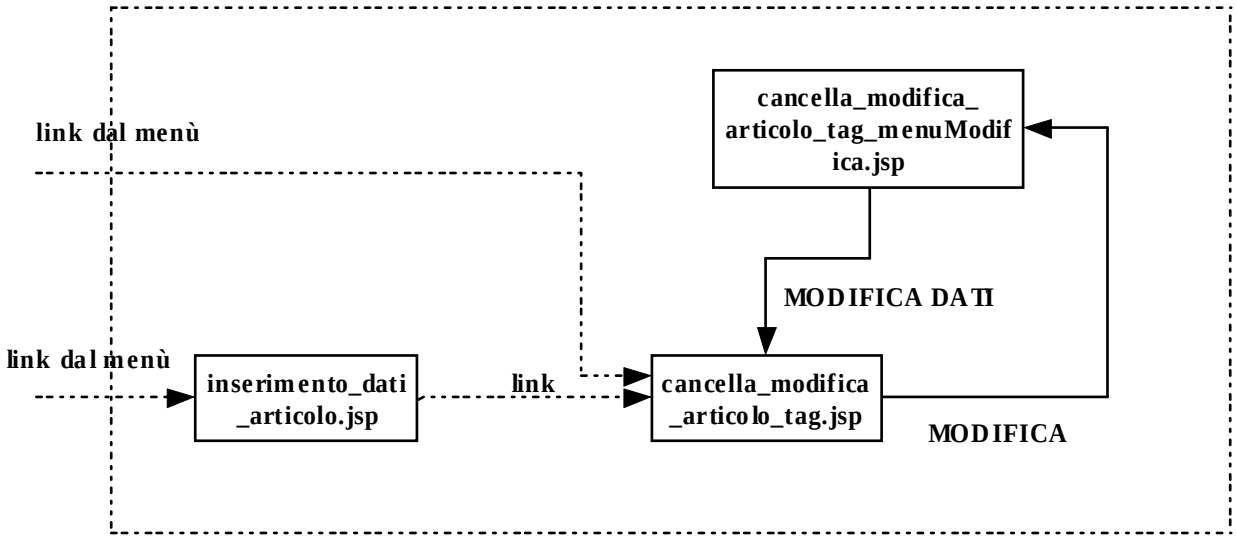
un discorso analogo viene ripetuto per i sottomenu relativi a cliente e azienda.

In un capitolo successivo vedremo come funziona la jsp effettua ordine e come opera il suo codice (vedi indice).

RELAZIONE TRA I VARI ELEMENTI DEL PROGRAMMA

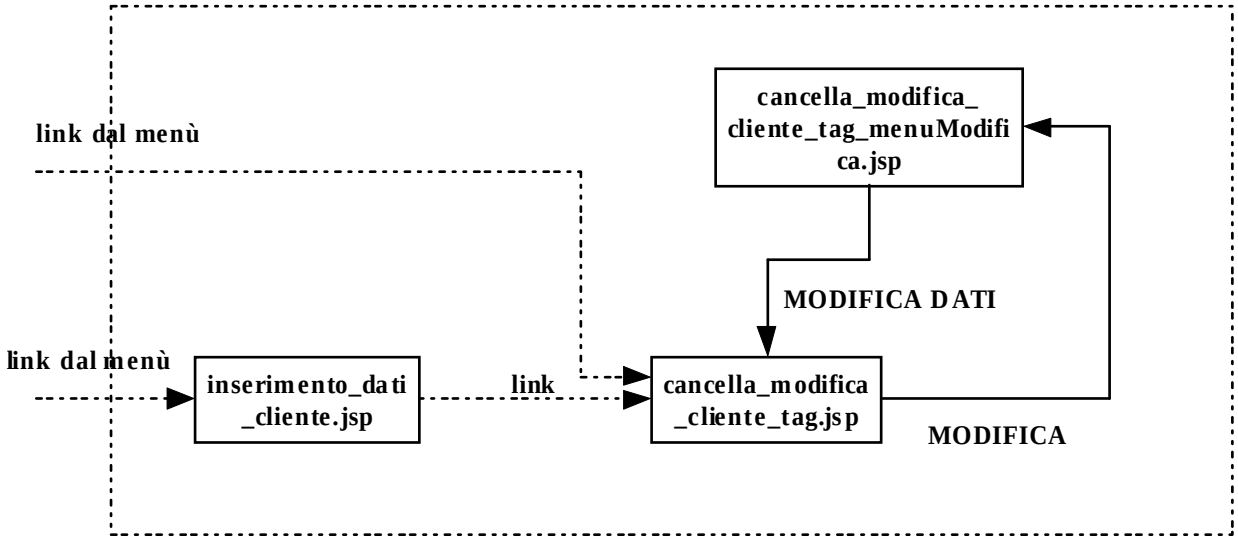


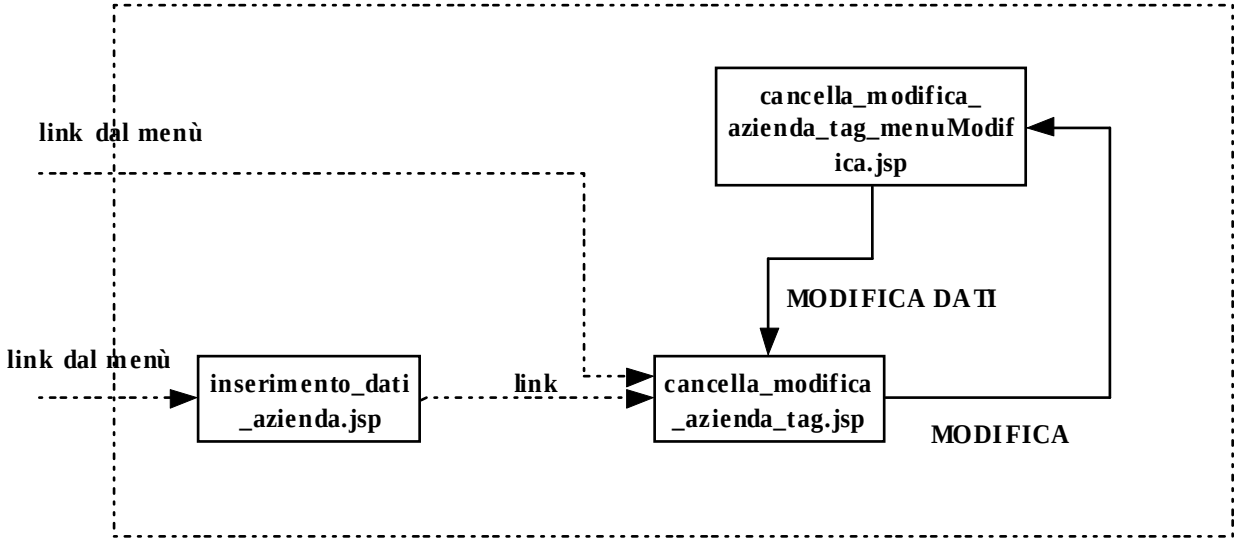
La jsp index appare quando ci connettiamo al programma .
Noi scriviamo la password e il nome utente e premiamo il pulsante invio; la password e il nome utente sono inviati alla jsp enter.
Se la password e il nome utente sono corretti, il programma esegue la jsp inizio e abilita l'utilizzo del menu di navigazione e di tutte le jsp da cui è formato il programma, altrimenti il programma va alla servlet loginErratoServlet e tutte le jsp che formano il programma sono bloccate .



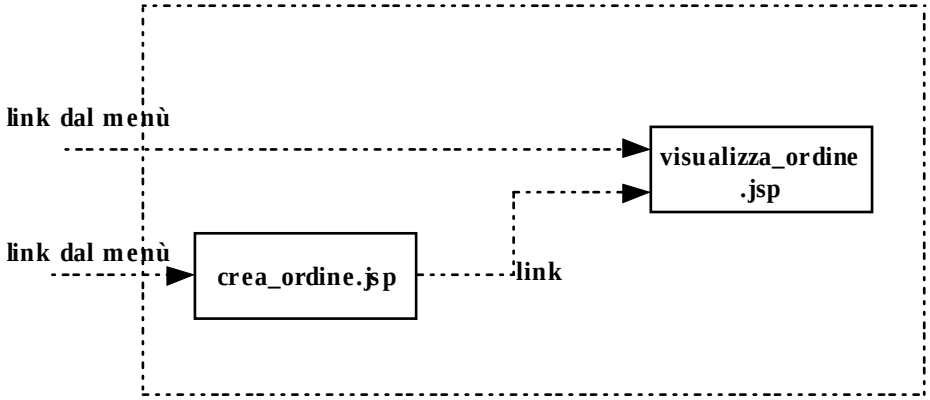
Dal menu di navigazione noi possiamo arrivare alla jsp inserimento_dati_ articolo oppure direttamente alla jsp cancella_modifica_articolo.
Quando siamo nella jsp cancella_modifica_articolo, la modifica di un articolo avviene premendo il pulsante MODIFICA.
Il pulsante MODIFICA, compare dopo che abbiamo selezionato una riga della tabella articoli (tabella che contiene gli articoli memorizzati fino a quel momento).
Nel momento in cui viene premuto il pulsante modifica, il controllo passa alla jsp cancella_modifica_articolo_tag_menu_modifica.
Dopo aver modificato i dati, la pressione del pulsante MODIFICA DATI presente nella jsp cancella_modifica_articolo_tag_menu_modifica riporta il controllo del programma alla jsp cancella_modifica_articolo.

Quanto scritto per gli elementi Articolo è valido anche per gli elementi Clienti e Aziende (vedi gli schemi seguenti)





Di seguito vediamo le jsp che gestiscono gli elementi di tipo Ordine

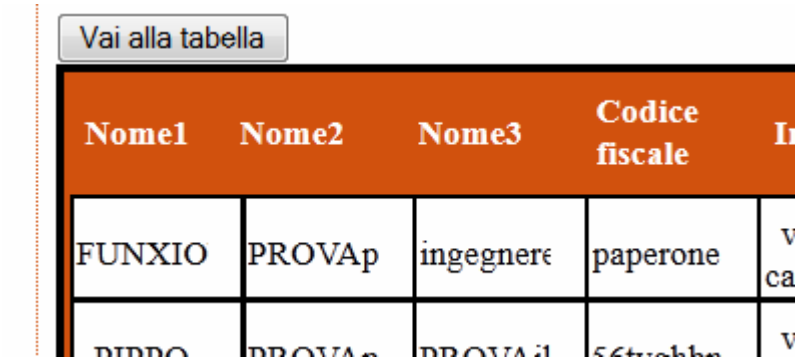


UN ESEMPIO DI COME FUNZIONA E DI COME OPERA IL CODICE DI UNA JSP:LA JSP CREA ORDINE

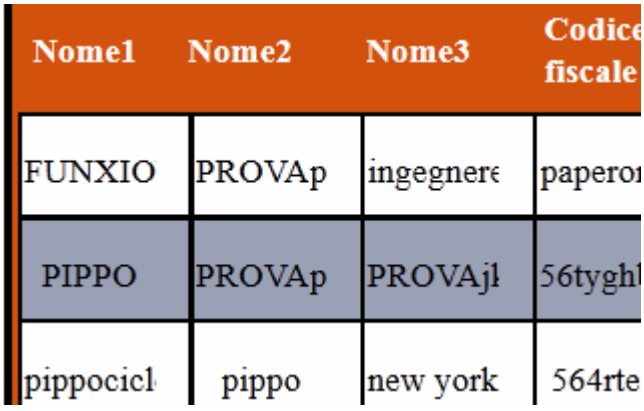
Iterazione con l'utente

1)inizialmente si seleziona o un'azienda o una persona.

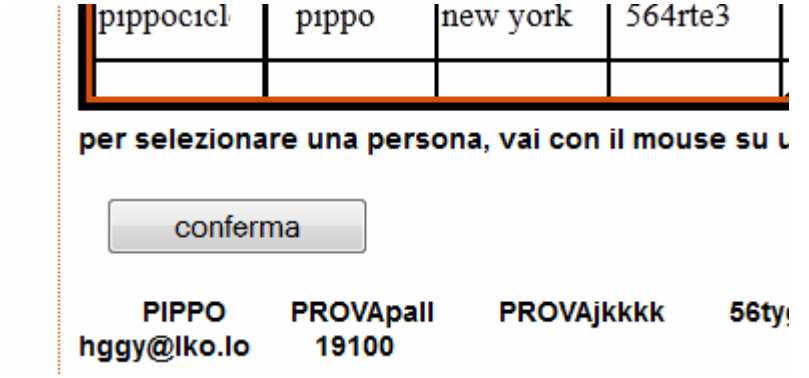
2)premendo il pulsante "Vai alla tabella" , viene visualizzata la tabella che contiene tutti i dati delle persone o delle aziende presenti nel database.



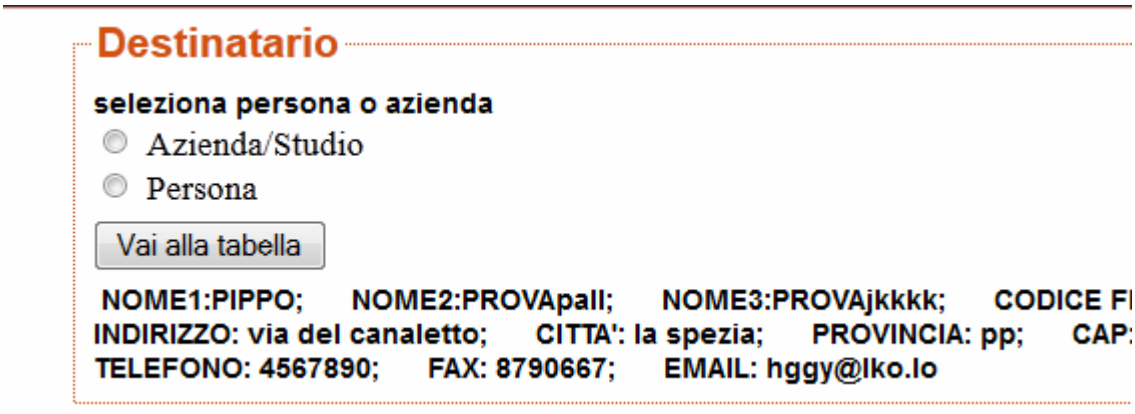
3)Passando con il mouse sopra una riga della tabella, tutta la riga cambia di colore.



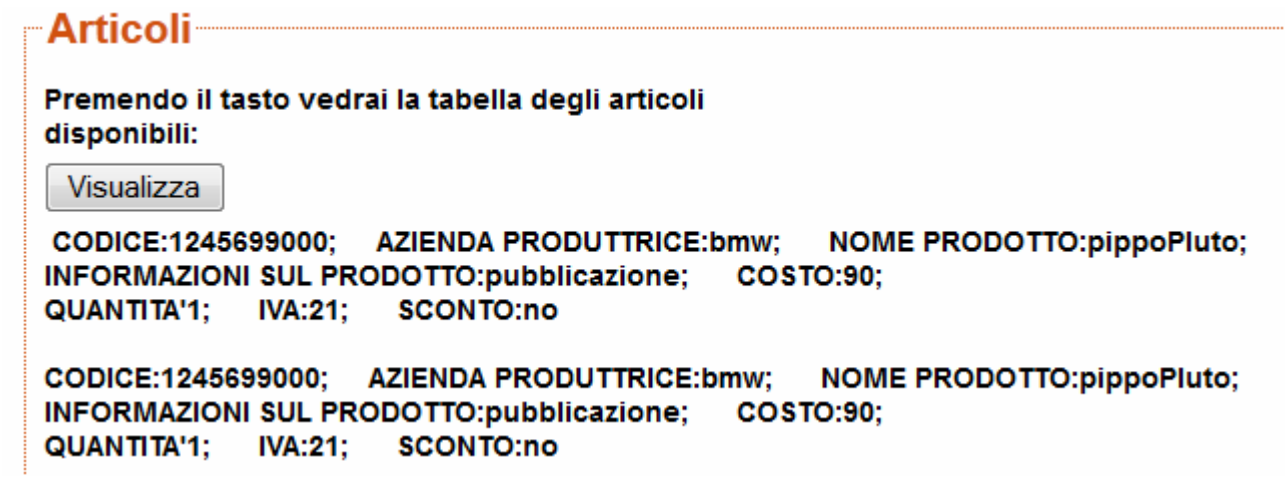
4)Se faccio click con il pulsante sinistro del mouse sulla riga che ha cambiato di colore, tutta la riga viene visualizzata come una stringa inoltre, sotto la tabella, compare un pulsante "conferma" che se premuto, permette di memorizzare la riga della tabella (riga che rappresenta un'azienda o una persona) sull'ordine



Il risultato che ottengo dopo aver premuto il pulsante conferma è il seguente

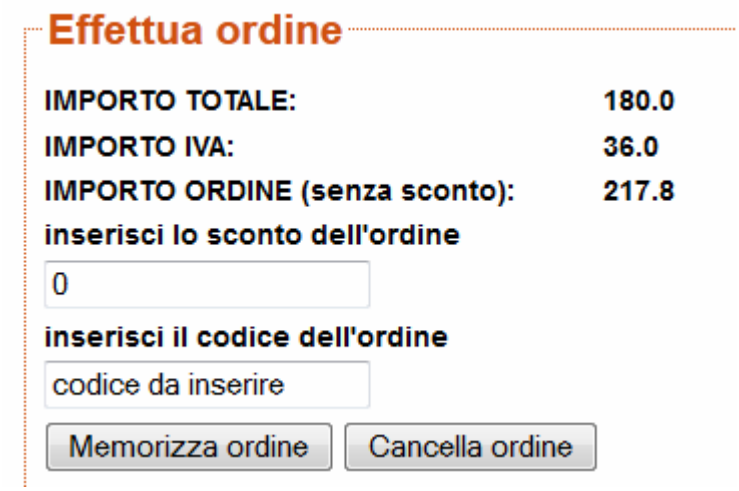


5) Per selezionare un articolo , ripeto quanto visto sopra per la selezione di una persona o di una azienda.
In questo caso, a differenza del destinatario dell'ordine che deve essere unico, possono selezionare più articoli.
Di ogni articolo bisogna inserire la quantità ordinata, l'Iva sull'articolo, e se è presente, uno sconto sul singolo articolo.



6)A questo punto viene calcolato l'importo complessivo dell'ordine e possiamo memorizzare l'ordine nel database facendo click sul pulsante " memorizza ordine" oppure cancellare l'ordine premendo il pulsante "cancella ordine".

Per memorizzare l'ordine è necessario scrivere (nell'input text dove è scritto codice da inserire) il codice dell'ordine; non possono esistere ordini aventi codici uguale.



Come funziona il codice

1) seleziono persona o azienda nel radio button (crea_ordini.jsp riga 452):

```
<span>
    <label for="destinatario" class="etich"> seleziona persona o azienda</label><br>
    <input type="radio"  name="destinatario" value="azienda" /> Azienda/Studio<br />
    <input type="radio"  name="destinatario" value="persona" /> Persona
</span>
<div>
    <input type="submit" name="vai_alla_tabella" value="Vai alla tabella" />
</div>
```

Creo il radio-button destinatario

Creo il bottone di submit "vai alla tabella"

2) premo il pulsante di submit "vai alla tabella".
Il risultato della selezione del radio button visto al punto 1), permette di attivare una delle due condizioni when (libreria jstl) scritte sotto (crea_ordini.jsp riga 461):

```
<c:choose>
    <c:when test="${param.destinatario=='persona'}">
                                :
                                :
    </c:when>
    <c:when test="${param.destinatario=='azienda'}">
                                :
                                :
    </c:when>
```

Questo when è attivo se nel radio-button "destinatario" ho selezionato persona

l'attivazione di una delle due condizioni when permette di creare la tabella HTML che visualizza i dati presenti nel database; supponiamo di attivare il when relativo a azienda, viene eseguito il seguente codice(crea_ordini.jsp riga 524):

```
<%--crea tabella dati per la visualizzazione  (commento)--%>
<% visDati.creaTabellaAzienda();%>
<% visDati.visualizzaTabellaScrollabileProvaJS();%>

<%--visualizza la tabella (commento)--%>
<jsp:getProperty name="visDati" property="dati"  />
```

visDati è il nome del bean che gestisce la visualizzazione dei dati, cioè che crea dinamicamente le tabelle HTML relative alle varie entità.

Il bean di nome visDati, viene generato nella head della jsp mediante la seguente istruzione
<jsp:useBean id="visDati" scope="session" class="database.VisualizzazioneDati" />

L'istruzione <jsp:getProperty name……. permette di visualizzare il contenuto della variabile (proprietà) dati del bean visDati. Questa variabile contiene il codice HTML che permette la visualizzazione della tabella sul browser del client

Il metodo **creaTabellaAzienda()** possiede (tra le altre) le seguenti istruzioni **(VisualizzazioneDati.java)**

```
1)Azienda c = new Azienda();
2)c.leggiDatiEsportazione();
3)tabella=new String[maxRighe][maxColonne];
4)tabella=c.getTabella();
```

1) crea un oggetto di tipo azienda

2) viene chiamato il metodo leggiDatiEsportazione() (contenuto in Azienda.java) che crea un array in cui, ogni elemento, contiene un elemento della entità azienda del database, più una riga iniziale che contiene il nome delle varie colonne della tabella

3) nella classe visualizzazioneDati, viene creato un array avente le stesse dimensioni dell'array ottenuto dal metodo leggiDatiEsportazione()

4) all' array presente nella classe VisualizzazioneDati, viene assegnato il riferimento (indirizzo) dell'array generato dalla classe azienda;
in questo modo nella classe VisualizzazioneDati abbiamo una variabile istanza (tabella[][]) che contiene tutti i dati della entità azienda presenti nel database

il metodo **visualizzaTabellaScrollabileProvaJS()** , opera nel seguente modo: prende l'array di valori ottenuto con il metodo creaTabellaAzienda() e li trasforma in una tabella HTML visualizzabile dal browser del client.
Per rendere sul client l'effetto della scrollabilità della tabella, si usa il seguente artifizio: invece di una singola tabella ne vengono create due, una dentro l'altra, quella più esterna fissa che contiene il nome delle colonne, mentre quella più interna scrollabile che contiene i dati.

3) il cambiamento di colore della riga della tabella viene ottenuto grazie ai CSS
(CSS/layout.css riga 68):

```
tr.riga:hover{
   background-color#9aa1b5;
}
```

4) creata la tabella si lavora sul **lato client**;
in particolare esiste una linea del codice del metodo **visualizzaTabellaScrollabileProvaJS()** che associa ad ogni riga della tabella HTML il metodo Javascript mostra(this) :
(database/VisualizzazioneDati.java riga 193)

```
dati+="<tr onclick=\"mostra(this)\" class=\"riga\" id=\"" + i + "\" >";
```

this restituisce l'oggetto riga della tabella su cui il mouse è puntato

Questo metodo viene attivato quando faccio click con il pulsante sinistro del mouse sulla riga della tabella che ha cambiato colore.
Vediamo più in dettaglio come funziona il metodo mostra() contenuto nella parte javascript di crea_ordini.jsp (riga 127) :

```
function mostra(x){
//se la tabella, ha righe di 11 nodi, hai una tabella cliente
        if (x.childNodes.length==11){
                :
                :
//se la tabella, ha righe di 12 nodi, hai una tabella azienda
        if (x.childNodes.length==12){
                :
                :
//se la tabella, ha righe di 5 nodi, hai una tabella articolo
        if (x.childNodes.length==5){
```

Alla funzione javascript **mostra(x)** viene passato un oggetto x che rappresenta la riga di una tabella. L'istruzione x.childNodes.length permette di calcolare la quantità di oggetti figli di x (DOM) quindi nel nostro caso, questa istruzione calcola il numero di elementi (colonne) da cui è formata la riga x:
se ho 11 elementi la riga appartiene alla tabella cliente che è formata da 11 colonne;
se ho cinque elementi, la riga appartiene alla tabella articolo.

Supponiamo ora di aver passato alla funzione mostra(x) la riga di una tabella azienda.
Nella jsp crea_ordine (riga 535) sono presenti una serie di pulsanti hidden (nascosti) ossia non visibili dal utente.

```
<input type="hidden" name="nome1" id="idNome1" >
```

```
<input type="hidden" name="nome2" id="idNome2" >
<input type="hidden" name="nome3" id="idNome3" >
<input type="hidden" name="partita_iva" id="idPartita_iva" >
```

a ognuno di questi pulsanti , viene assegnato il valore di un elemento della riga che abbiamo selezionato con il mouse.

```
for(var i=0;i<12;i++){
        if(i==0){
           nome1=x.childNodes[i].textContent;
           document.getElementById('idNome1').value=nome1;

        }
        if(i==1){
           nome2=x.childNodes[i].textContent;
           document.getElementById('idNome2').value=nome2;
        }
        if(i==2){
           nome3=x.childNodes[i].textContent;
           document.getElementById('idNome3').value=nome3;
        }
        if(i==3){
           partita_iva=x.childNodes[i].textContent;
           document.getElementById('idPartita_iva').value=partita_iva;
        }
                :
                :
        messaggio += "        " + x.childNodes[i].textContent;
        msg=messaggio;


        document.getElementById('etichettaCli').innerHTML=msg;
```

Se i =0, il contenuto del nodo zero (che è il primo elemento del riga selezionata con il mouse) viene assegnata alla variabile nome1 dopodiché il valore della variabile nome1 viene assegnato al pulsante avente id= idNome1

Grazie al ciclo for , viene creato un messaggio che contiene tutti gli elementi della riga della tabella azienda

Quando con il mouse faccio clic sulla riga della tabella che ha cambiato colore, msg viene visualizzato nella seguente etichetta presente nel codice jsp di creaordine.jsp
(riga 560)
 <jsp:getProperty name="visDati" property="stringa" />
è da notare che in questa situazione la proprietà "stringa" del bean visDati vale "" cioè stringa vuota.

Questa operazione viene fatta perché la tabella viene visualizzata lato client, mentre a noi serve di inviare i dati selezionati sul server.
Per fare questo, è necessario utilizzare un form presente lato client, che invii i dati contenuti nei bottoni nascosti al server, quando viene premuto un pulsante di submit.

Nel nostro caso il pulsante di submit è confermaAziendaCli che viene creato con il seguente codice javascript

```
if(document.getElementById('idConferma')==null){
   var nodo = document.getElementById("bottoni");
   //crea il bottone conferma
   var conferma = document.createElement("input");
   conferma.setAttribute("type", "submit");
   conferma.setAttribute("name", "confermaAziendaCli");
   conferma.setAttribute("value", "conferma");
   conferma.setAttribute("class", "bottone");
   conferma.setAttribute("id", "idAziendaConferma");
   conferma.setAttribute("onClick", "scrivi()");
   nodo.appendChild(conferma);
}
```

La funzione scrivi() serve a cancellare la tabella e la scritta sotto la tabella sul client

Vediamo ora cosa accade quando premiamo il pulsante "conferma" (confermaAziendaCli):
viene inviato il contenuto della riga della tabella lato client al server (crea_ordini.jsp riga 551), viene creata (lato server) la stringa di dati che poi viene visualizzata dal bean visDati al posto della tabella e che serve a visualizzare sull'ordine, l'azienda destinataria dell'ordine stesso.

```
<c:when test="${param.confermaAziendaCli!=null}">
    <jsp:setProperty name="azienda" property="*" />
    <%azienda.setDati();%>
    <%visDati.setStringa(azienda.getDati());%>
</c:when>
```

Questo istruzione invia tutti i valori contenuti nei pulsanti nascosti al bean azienda

Il metodo **setDati()** inizializza la proprietà dati del bean azienda con i dati arrivati dal client, aggiungendo dei parametri come NOME1, NOME2, NOME3,PARTITA IVA ecc

Questo metodo visualizza la stringa dati del bean azienda ottenuta con le istruzioni precedenti

5) a questo punto selezioniamo uno o più articoli.
Il modo con cui avviene la selezione di un articolo è simile a quanto visto per l'azienda con la differenza che, nel caso di un articolo si inserisce il numero di oggetti, l'Iva e lo sconto su ogni oggetto e che è possibile selezionare articoli diversi nello stesso ordine (riga 582).

```
<c:choose>
  <c:when test="${param.visualizza_articoli!=null}">

    <%--crea tabella dati per la visualizzazione --%>
    <% visDati.creaTabellaArticolo();%>
    <% visDati.visualizzaTabellaScrollabileProvaJS();%>

    <%--visualizza tabella articoli --%>
    <jsp:getProperty name="visDati" property="dati"  />

    <%--visualizza altri elementi html --%>
    <div id="scritta">per selezionare un articolo, vai con il mouse su  una riga della
        tabella e fai click con il pulsante destro  </div>
    <div id="bottoni"></div>

    <%--creo i bottoni nascosti (hidden)per inviare i dati del cliente selezionato dal
                                                client al server--%>
    <input type="hidden" name="id" id="idID" >
    <input type="hidden" name="azienda_produttrice"  id="idAzienda_produttrice" >
    <input type="hidden" name="nome_prodotto" id="idNome_prodotto" >
    <input type="hidden" name="informazione_prodotto" id="idInformazione_prodotto" >
    <input type="hidden" name="costo" id="idCosto" >
    <input type="hidden" name="id" id="idID" >
    <input type="hidden" name="quantita" id="idQuantita" >
    <input type="hidden" name="iva" id="idIva" >
    <input type="hidden" name="sconto" id="idSconto" >
  </c:when>

  <c:when test="${param.confermaArt!=null}">
    <jsp:setProperty name="articolo" property="*" />
    <%articolo.aggiungiArticoloAllaLista();%>
    <%articolo.restituisciListaArticoli();%>
  </c:when>
</c:choose>
```

6) passiamo quindi alla registrazione dell'ordine (riga 644)

```
<fieldset>
  <legend>Effettua ordine</legend>
    <%ordine.calcolaImportoTotale(articolo.getVettoreArticoli());%>
    <%ordine.calcolaImportoIva(articolo.getVettoreArticoli());%>
    <%ordine.calcolaImportoOrdineSenzaSconto(articolo.getVettoreArticoli());%>
```

I tre metodi precedenti servono a calcolare il valore dell'importo totale dell'ordine (cioè il costo dei singoli articoli senza Iva), dell'Iva complessiva e dell'importo dell'ordine senza lo sconto. Per effettuare i conti è necessario fornire ai tre metodi l'array list degli articoli che compongono l'ordine;
array list è ottenuta con il articolo.getVettoreArticoli()

```
 <div>
    <span class="etich" >IMPORTO  TOTALE:</span>
    <span class ="etich" id="idImportoTotale"><jsp:getProperty name="ordine"
                property="importo_totale"  /></span>
 </div>
 <div><span class="etich" >IMPORTO IVA:</span>
      <span class ="etich" id="idImportoIva"><jsp:getProperty name="ordine"
                          property="importo_iva"  /></span>
</div>
<div><span class="etich" >IMPORTO ORDINE        (senza sconto):</span>
      <span class ="etich" id="idImportoOrdine"><jsp:getProperty name="ordine"
                                property="importo_ordine"  /></span>
</div>
```

Questa serie di istruzioni creano le etichette in cui verranno scritte l'Iva, l'importo dell'ordine e l'importo dell'ordine senza sconto

```
<div>
  <label for="sconto_ordine" class="etich"> inserisci lo sconto dell'ordine</label><br>
  <input type="text" name="sconto_ordine" id="idScontoOrdine" value="0"><br>
</div>
<div>
  <label for="idOrdine" class="etich"> inserisci il codice dell'ordine</label><br>
  <input type="text" name="idOrdine" id="IDidOrdine" value="codice da inserire" >
  <%--<jsp:setProperty name="ordine" property="idOrdine" param="codiceOrdine"/>--%>
</div>
```

Questa serie di istruzioni creano gli imput text in cui l'utente dovrà inserire il codice dell'ordine (obbligatorio per avere la demonizzazione dell'ordine) e lo sconto del ordine (opzionale)

```
 <div>
  <input type="submit" name="memorizzaOrdine" onClick="cancellaScritte()"
                                              value="Memorizza ordine" />
  <input type="submit" name="cancellaOrdine"  value="Cancella ordine" />
</div>

<c:choose>
        <%--hai selezionato un cliente--%>
  <c:when test="${param.memorizzaOrdine!=null && stato=='persona'}">
     <c:if test="${param.idOrdine!='codice da inserire' }">
       <jsp:setProperty name="ordine" property="idOrdine" param="idOrdine" />
```

Scrivi il codice dell'ordine (scritto nella input text precedente) nella variabile idOrdine del bean ordine

<%-- **calcola importo ordine**--%>

```
<jsp:setProperty name="ordine" property="sconto_ordine" param="sconto_ordine" />
<%ordine.calcolaImportoOrdine(articolo.getVettoreArticoli());%>
```

Viene calcolato l'importo complessivo dell'ordine; per fare questo viene applicata la seguente formula
FORMULA:
importo_ordine= (costo di un oggetto x quantità) + (costo x IVA) - (sconto x quantità) " l'operazione è ripetuta per ogni articolo presente nell'ordine"- *sconto ordine*

<%-- **inserimento dati ordine**--%>

```
<%ordine.inserisciDati();%>
```

I dati dell'ordine sono inseriti nelle entità ordine del database

```
<%ord_cli.inserisciDati(ordine.getIdOrdine(),cliente.getCodice_fiscale());%>
```

Gli id dell'ordine ed il codice fiscale del cliente sono inseriti nelle entità ord_cli del database

```
<%ord_art.inserisciListaDati(articolo.getVettoreArticoli(),ordine.getIdOrdine());%>
```

L' id dell'ordine e gli id dei vari articoli che formano l'ordine sono inseriti (in righe separate) nella entità ord_art del database

<%-- **cancella dati ordine**--%>

```
<%azienda.resetVariabili();%>
```

Vengono azzerate le variabili istanza (le più portanti) della classe azienda

```
<%articolo.resetLista();%>
<%articolo.cancellaVettoreArticoli();%>
```

Viene azzerato la lista e le variabili istanze della classe articolo

```
<%ordine.resetVariabili();%>
```

Vengono azzerate le variabili istanza della classe ordine

```
<script>
   document.getElementById("etichettaArt").style.display="none";
   document.getElementById("etichettaCli").style.display="none";
   document.getElementById("idImportoTotale").style.display="none";
   document.getElementById("idImportoIva").style.display="none";
   document.getElementById("idImportoOrdine").style.display="none";
</script>
```

Viene usato il linguaggio javascript e il DOM per rendere non visibili i nodi che contengano i vari dati dell'ordine

```
   <div class="etich">ordine effettuato</div>
```

```
<%--hai selezionato una azienda --%>
  <c:when test="${param.memorizzaOrdine!=null && stato=='azienda'
     <c:if test="${param.idOrdine!='codice da inserire' }">
        <jsp:setProperty name="ordine" property="idOrdine" param="idOrdine" />

        <%-- calcola importo ordine--%>
        <jsp:setProperty name="ordine" property="sconto_ordine" param="sconto_ordine" />
        <%ordine.calcolaImportoOrdine(articolo.getVettoreArticoli());%>

        <%-- inserimento dati ordine--%>
        <%ordine.inserisciDati();%>
        <%ord_az.inserisciDati(ordine.getIdOrdine(),azienda.getPartita_iva());%>
        <%ord_art.inserisciListaDati(articolo.getVettoreArticoli(),ordine.getIdOrdine());%>

        <%-- cancella dati ordine--%>
        <%azienda.resetVariabili();%>
        <%articolo.resetLista();%>
        <%articolo.cancellaVettoreArticoli();%>
```

```
<%ordine.resetVariabili();%>
<script>
   document.getElementById("etichettaArt").style.display="none";
   document.getElementById("etichettaCli").style.display="none";
   document.getElementById("idImportoTotale").style.display="none";
   document.getElementById("idImportoIva").style.display="none";
   document.getElementById("idImportoOrdine").style.display="none";

</script>
<div class="etich">ordine effettuato</div>
```

Come funziona il semaforo

```
<c:if test="${semaforo1.passa}">
        <c:set var="vis" value="0" />
        <%azienda.resetVariabili();%>
        <%cliente.resetVariabili();%>
        <%visDati.resetStringa();%>
        <%articolo.resetLista();%>
        <%articolo.cancellaVettoreArticoli();%>
        <%ordine.resetVariabili();%>
        <script>
                document.getElementById("etichettaArt").style.display="none";
                document.getElementById("etichettaCli").style.display="none";
                document.getElementById("idImportoTotale").style.display="none";
                document.getElementById("idImportoIva").style.display="none";
                document.getElementById("idImportoOrdine").style.display="none";

        </script>
 </c:if>
```

Semaforo serve per poter cancellare eventuali elementi presenti nell'ordine la prima volta che l'ordine viene eseguito.
Per fare questo, è stato creato un bean semaforo contenente una sola variabile booleana passa che è definita come true (cioè il semaforo è verde), quindi all'inizio il codice contenuto nell'if viene eseguito.
Il semaforo blocca (cioè il semaforo diventa rosso) quando all'interno dell'ordine viene selezionata o una persona, o una azienda, o un articolo, infatti in queste condizioni le scritte che appaiono sullo schermo non devono essere cancellate.

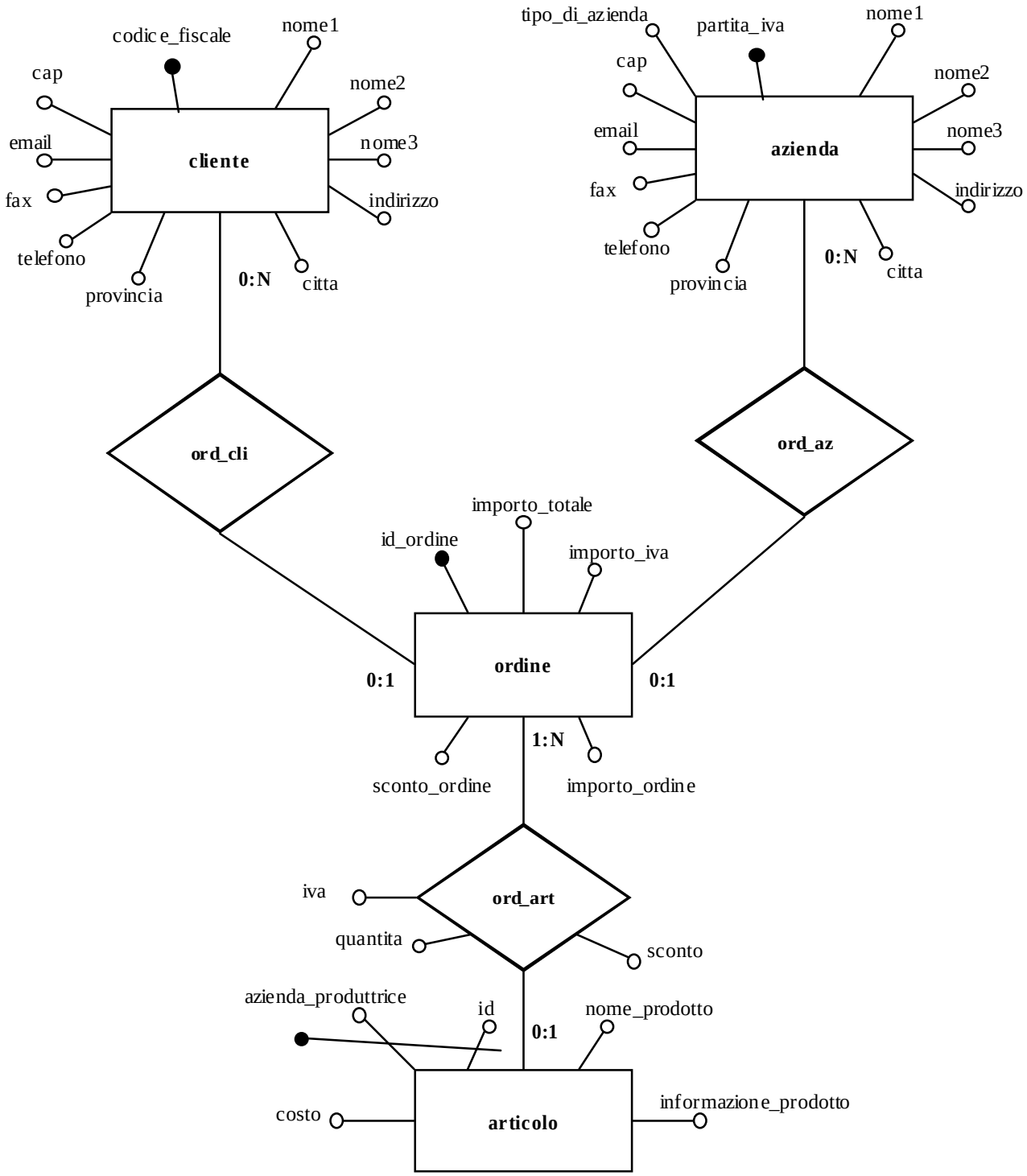
```
<%-- ***********************************************
                    PERSONA NEL DATABASE
 ***********************************************--%>

      <%semaforo1.bloccaSemaforo();%>
```

Il semaforo viene disattivato (ritorna verde) quando viene effettuato l'ordine, in modo che, quando i dati dell'ordine sono inviate al database, la schermata del programma viene svuotata di tutti i dati inseriti.

DATABASE

Diagramma ER del database



Dal diagramma ER allo schema relazionale

cliente (codice fiscale, nome1, nome2, nome3, indirizzo, città, provincia, telefono, fax, e-mail,cap)

azienda (partita Iva , nome1, nome2, nome3, indirizzo, città, provincia, telefono, fax, e-mail,cap, tipo di azienda)

ordine (id ordine, importo totale, importo Iva, importo ordine, sconto ordine)

articolo (id, azienda produttrice, informazioni prodotto, costo)

ord_cli(ordine rif, cliente rifer)

Foreign Key: ordine rif fa riferimento a id ordine della entità ordine
Foreign Key: cliente rif fa riferimento a codice fiscale della entità cliente

ord_az(ordine rif, azienda rif)

Foreign Key: ordine rif fa riferimento a id ordine della entità ordine
Foreign Key: azienda rif fa riferimento a partita Iva della entità azienda

ord_art(ordine rif,articolo rif, articolo azienda rif, iva, quantità, sconto)

Foreign Key: ordine rif fa riferimento a id ordine della entità ordine
Foreign Key: articolo rif fa riferimento a id della entità articolo
Foreign Key: articolo azienda rif fa riferimento a azienda produttrice della entità articolo

CODICE JAVA

gestione del database

ad ogni entità del database viene associato un bean (classe) in Java.
Questo permette di lavorare sul database tramite i bean.

Visualizzazione dei dati delle entità presenti nel database

viene usato la classe (bean) "visualizzazioneDati" che permette di costruire la tabella HTML delle entità: azienda, cliente, ordine e articolo.
L'operazione di visualizzazione dati è ottenuta con due passi:

-il primo passo consiste nell'inizializzare la variabile istanza "String[][] tabella" che è un array di stringhe, con i valori contenuti nelle entità: azienda o cliente o ordine oppure articolo, in base alle richieste del programma.
Per fare questo viene utilizzato uno dei seguenti metodi: creaTabellaAzienda() oppure creaTabellaCliente() oppure creaTabellaOrdine() oppure creaTabellaArticolo (), in base all'entità di cui vogliamo visualizzare i dati.

Di seguito vediamo il codice del metodo creaTabellaArticolo ()

```
public void creaTabellaArticolo(){
        // crea un oggetto cliente
        Articolo c = new Articolo();
        //leggi la tabella azienda del database
        c.leggiDatiEsportazione();
        //dimensioni della tabella
        maxRighe=c.getMaxRighe();
        maxColonne=c.getMaxColonne();
        //creo la tabella
        tabella=new String[maxRighe][maxColonne];
        tabella=c.getTabella();
}
```

- Il secondo passo, consiste nella costruzione della tabella in formato HTML.
Poiché la tabella deve permettere lo scroll dei dati durante la visualizzazione, non realizziamo una sola tabella, ma due, una dentro l'altra, in modo che la seconda tabella possa scrollare liberamente all'interno della prima.
La prima tabella contiene i nomi delle colonne che sono fissi.
Il codice è il seguente

```
public void visualizzaTabellaScrollabileProvaJS()throws Exception{
        //creo la tabella in HTML
        dati="<table class=\"tabellauno\" id=\"idTabellauno\"><thead>";

        //scansione delle righe
        int i=0;
        dati+="<tr>";
        for(int j=0;j<maxColonne;j++){
            //scansionedelle colonne
            int idcol=j+1;
            dati+="<th id=\"header" +  idcol
                  + "\"><div class=\"divInternoCella\">" + tabella[i][j]
                  + "</div></th>";
        }
        dati+="</tr></thead>";
```

Viene creata la tabella uno e la sua intestazione tramite il tag HTML `<thead>`

Nell'intestazione della tabella uno vengono scritti i titoli delle varie colonne.
Fatto questo viene terminata l'intestazione

viene creato il body della tabella uno mediante il tag `<tbody>`, quindi si crea una riga (tag `<tr>`) che sarà l'unica riga da cui è formata la tabella uno.
A sua volta la riga sarà formata da un'unica colonna (cella) `colspan=\""+ maxColonne + "` che conterrà al suo interno un tag che a sua volta conterrà la tabella due

```
        dati+= "<tbody><tr><td colspan=\""+ maxColonne + "\">"
            + "<div class=\"divinterno\"><table class=\"tabelladue\">";
```

NOTA:l'attributo `colspan` permette di raggruppare le celle all'interno delle colonne in modo da avere, ad esempio, una riga da 5 colonne (tabella articolo) oppure da 12 colonne (tabella azienda) .

Vengono create le righe della tabella due;
a ogni riga è associato il metodo JavaScript `mostra(this)` che permette di visualizzare la riga selezionata sotto la tabella

```
for(i=1;i<maxRighe;i++){
            dati+="<tr onclick=\"mostra(this)\" class=\"riga\" id=\"" + i
                  + "\" >";
            for(int j=0;j<maxColonne;j++){
                int idcol=j+1;
                dati+="<td id=\"header" +  idcol
                        + "\"><div class=\"divInternoCella\">" + tabella[i][j]
                        + "</div></td>";
            }
            dati+="</tr>";
        }

        dati+="</tr></tbody></table></div></td></tr></tbody></table>";
}
```

Vengono create le celle della tabella due.
Le celle contengono i dati da visualizzare e sono contenuti nell' array `tabella[i][j]`