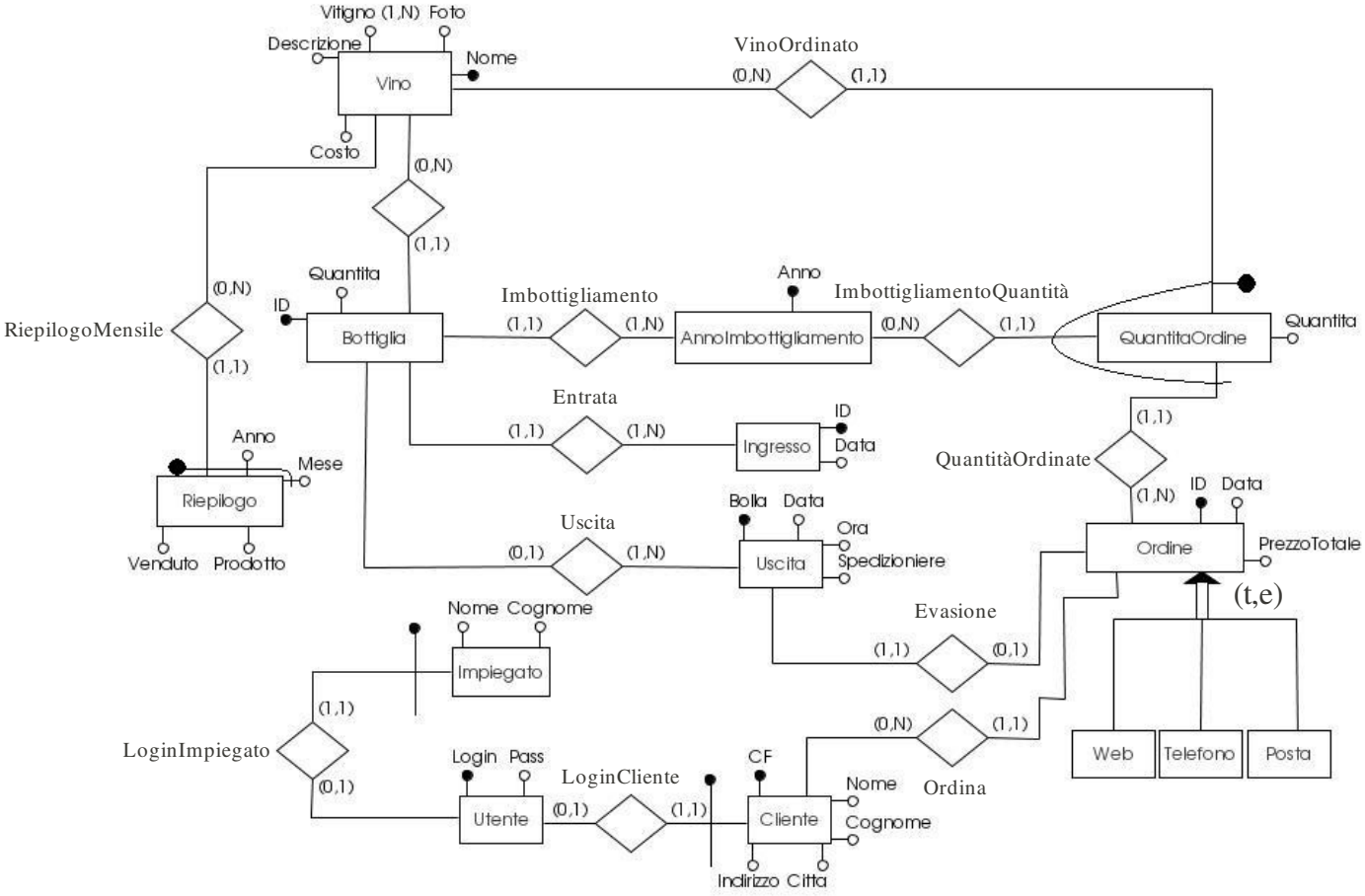
DAVIDE FERRARETTO – VR092904

PROGETTAZIONE DELLA BASE DI DATI

PROGETTAZIONE CONCETTUALE



Considerazioni, ipotesi e strategie adottate:

- Per rendere il sito più sicuro, si è deciso di avere “user name” diversi per ogni utente, in modo tale che non si verifichino le seguenti condizioni:
  - due impiegati con lo stesso user
  - due clienti con lo stesso user
  - un cliente e un impiegato con lo stesso user
- Dato che cliente può essere identificato univocamente sia dal “codice fiscale” che dal proprio “user name” si è deciso di adottare queste ultime come identificatori.
- Nelle specifiche progettuali consegnatomi, QuantitaOrdine e Bottiglia fanno riferimento allo stesso attributo “anno di imbottigliamento”. Onde evitare troppe inconsistenze nella base di dati, si è deciso di promuovere l'attributo “anno di imbottigliamento“ in entità. In questo modo un'istanza di QuantitaOrdine dovra fare riferimento ad un vino nella base di dati ed ad un anno di imbottigliamento nella base di dati. Stessa cosa vale per Bottiglia. Ovviamente c'è il rischio che una quantita d'ordine faccia riferimento ad una coppia (anno, vino) della quale non esiste in memoria una bottiglia con quel vino e quell'anno.
- Le foto delle bottiglie verranno memorizzate in locale nel disco del server, mentre i vari path nel DB.

PROGETTAZIONE LOGICA (vedere script: create_tables.sql)

POPOLAMENTO DELLA BASE DI DATI (vedere script: popola.sql)

PROGETTAZIONE DEL SITO WEB
PROGETTAZIONE LOGICA

```
page-schema HomePage unique (
        login_cliente: form (
                login: text;
                pass: password;
                invia: submit ( );
        );
        login_impiegato: form(
                login: text;
                pass: password;
                invia: submit ( );
        );
        vini_page: link( qui, *ViniPage );
        foto_vini: list_of(
                foto: string
        );
)

DB to page-schema HomePage
parameter( )
(
        foto: SELECT foto FROM Vino;
)

DB from page-schema HomePage(
        login_cliente: // questa query ha il duplice scopo di verificare se login e password sono corretti
                       // e di prelevare le informazioni necessarie sul cliente per la pagina ClientePage
                       if ( SELECT cf, nome, cognome, indirizzo, citta, C.login AS login
                            FROM Utente U INNER JOIN Cliente C ON U.login = C.login
                            WHERE U.login = ?login? AND U.pass = ?pass? )
                       then *HomePage else *ClientePage end;
        login_impiegato: // verifico se login e password sono corretti
                         if ( SELECT login
                              FROM Utente INNER JOIN Impiegato ON login = matricola
                              WHERE login = ?login? AND pass = ?pass? )
                         then *HomePage else *EvasionePage end;
)

page-schema ClientePage (
        cf: string;
        nome: string;
        cognome: string;
        indirizzo: string;
        citta: string;
        login: string;
        elenco_ordini: list_of (
                codice: link( cod_ord: integer, *OrdinePage );
                data: date;
        ) ;
        // disponibilita dei vini raggruppati per anno
        vini_disponibili: list_of (
                vino: link( nome_vino: string, *VinoPage );
                anno: string;
```

```
            disponibilità: integer;
        );
        ordina: form (
            //quantita ordinata, nome vino, anno imbottigliamento
            o_1: text; v_1: hidden; a_1: hidden;
            .
            .
            .
            o_n: text; v_n: hidden; a_n: hidden;
            // numero delle quantita disponibili(vino,anno)
            size: hidden;
            invia: submit( );
        );
        home: link( HomePage, *HomePage );
)

DB to page-schema ClientePage
parameter( login )
(
        cf, nome, cognome, indirizzo, citta, login:
            //questa query viene usata solo dopo il login iniziale, prima si usa sempre la query usata in
            //login_cliente
            SELECT cf, nome, cognome, indirizzo, citta, login
            FROM Cliente C
            WHERE login = ?login?;
        cod_ord, data:
            SELECT id, data
            FROM Ordine INNER JOIN Cliente ON cliente = cf
            WHERE login = ?login?
            ORDER BY data DESC;
        nome_vino, anno, disponibilità:
            SELECT BM.vino AS vino, BM.anno AS anno, bot_magazzino - COALESCE(bot_non_disp,0) AS
            quantita
            FROM (
                // conta il numero di bottiglie in magazzino per ogni anno e vino
                SELECT vino, anno, count(*) AS bot_magazzino
                FROM Bottiglia WHERE uscita IS NULL
                GROUP BY vino, anno
            ) BM LEFT JOIN (
                // conta il numero di bottiglie ordinate e non ancora spedite per ogni anno e vino
                SELECT vino, anno, SUM(quantita) AS bot_non_disp FROM QuantitaOrdine Q
                WHERE NOT EXISTS (
                        SELECT * FROM Uscita U WHERE U.ordine = Q.ordine )
                GROUP BY vino, anno
            ) BND ON (BM.vino = BND.vino AND BM.anno = BND.anno)
            WHERE bot_magazzino - COALESCE(bot_non_disp,0) > 0
            ORDER BY vino, anno;
)

DB from page-schema ClientePage(
        ordina:
            // numero di tipologie di vini (vino, anno) presenti nella form
            n = SIZE;
            // costo dell'ordine totale
            costo = 0;
            // per ogni quntità considerando solo quelle con almeno una bottiglia ordinata
            for(i = 0; i < n; i++){
                // verifico se la quantità ordinata è ancora disponibile
```

```
                    if ( SELECT BM.vino
                        FROM (
                            // bottiglie in magazzino
                            SELECT vino, anno, count(*) AS bot_magazzino
                            FROM Bottiglia
                            WHERE vino = ?v_i? AND anno = ?a_i? AND uscita IS NULL
                            GROUP BY vino, anno
                        ) BM LEFT JOIN (
                            // bottiglie ordinate e non ancora spedite
                            SELECT vino, anno, SUM(quantita) AS bot_non_disp
                            FROM QuantitaOrdine Q
                            WHERE NOT EXISTS (
                                    SELECT *
                                    FROM Uscita U
                                    WHERE U.ordine = Q.ordine )
                            GROUP BY vino,anno
                        ) BND ON BM.vino = BND.vino AND BM.anno = BND.anno
                        WHERE bot_magazzino - COALESCE(bot_non_disp,0) – ?o_i? >= 0)
                    then
                        // calcolo il costo della singola quantita ordinata e l'accumolo
                        costo += o_i * ( SELECT costo FROM Vino WHERE nome = ?v_i?);
                    else
                        break;
                    end
                }
                // se tutte le quantita sono disponibili
                if i == n then
                        // recupero codice fiscale del cliente dall'user
                        cf = SELECT cf FROM Cliente WHERE login = ?login?;
                        // calcolo della prima chiave disponibile per l'ordine
                        id = (  SELECT MIN(id + 1) AS id
                                FROM Ordine
                                WHERE id + 1 NOT IN (SELECT id FROM Ordine)
                            );
                        // inserimento ordine
                        INSERT INTO Ordine(id, data, prezzo_totale, tipo, cliente)
                        VALUES(?id?,current_date,?costo?,'w',?cf?);
                        // inserimento delle varie quantita' in cui e' stata ordinata almeno una bottiglia
                        for(i = 0; i < n; i++){
                                INSERT INTO QuantitaOrdine(anno, vino, ordine, quantita)
                                VALUES(?a_i?,?v_i?,?id?,?o_i?);
                        }
                        // vai alla pagina dell'ordine id
                        *OrdinePage;
                else
                        // bottiglie non più sufficienti
                        // vai alla pagina del cliente
                        *ClientePage;
                end
)

page-schema OrdinePage (
        codice: integer;
        data: date;
        prezzo_totale: real;
        tipo: string;
        data_evasione: date;
        ora_evasione: time;
```

```
        spedizioniere: string;
        bolla: integer;
        vini_ordinati: list_of (
                vino: string;
                anno: string;
                quantita: integer;
        );
        home: link( HomePage, *HomePage );
        cliente_page: link( ClientePage, *ClientePage );
        vini_page: link( ViniPage, *ViniPage );
)

DB to page-schema OrdinePage
// id dell'ordine
parameter( id )
(
        codice, data, prezzo_totale, tipo, bolla, data_evasione, ora_evasione, spedizioniere:
                SELECT id, O.data AS data, prezzo_totale, tipo, bolla, U.data AS data_evasione, ora, spedizioniere
                FROM Ordine O LEFT JOIN Uscita U ON id = ordine
                WHERE id = ?id?;
        vino, anno, quantita:
                SELECT vino, anno, quantita
                FROM QuantitaOrdine
                WHERE ordine = ?id?
                ORDER BY vino, anno;
)

page-schema ViniPage unique (
        vini: list_of (
                vino: link( nome: string, *VinoPage );
                disponiblità: integer;
        );
        home: link( HomePage, *HomePage );
        // mostra il seguente link solo se loggato
        cliente_page: link( ClientePage, * ClientePage );
)

DB to page-schema ViniPage
parameter( )
(
        nome, disponiblità:
        //ANCHE SE A PRIMA VISTA PUO' NON SEMBRARE,
        //E' ASSOLUTAMENTE ERRATO ESEGUIRE IL CONTEGGIO E SOMMA RAGGRUPPANDO SOLO
        //PER VINO E NON ANCHE PER ANNO
                SELECT nome AS vino, quantita
                FROM Vino LEFT JOIN (
                   // conta il numero di bottiglie disponibili per ogni vino
                   SELECT BM.vino AS vino, SUM(bot_magazzino - COALESCE(bot_non_disp,0)) AS quantita
                   FROM (
                        // conta il numero di bottiglie in magazzino per ogni vino e anno
                        SELECT vino, anno, count(*) AS bot_magazzino
                        FROM Bottiglia
                        WHERE uscita IS NULL
                        GROUP BY vino, anno
                   ) BM LEFT JOIN (
                        // conta il numero di bottiglie ordinate e non ancora spedite per ogni vino e anno
                        SELECT vino, anno, SUM(quantita) AS bot_non_disp
```

```
                        FROM QuantitaOrdine Q
                        WHERE NOT EXISTS (
                                SELECT *
                                FROM Uscita U
                                WHERE U.ordine = Q.ordine)
                                GROUP BY vino, anno
                    ) BND ON (BM.vino = BND.vino AND BM.anno = BND.anno)
                    // condizione necessaria per far si che non appaiano valori negativi
                    WHERE bot_magazzino - COALESCE(bot_non_disp,0) > 0
                    GROUP BY BM.vino
                ) AS ViniDisp ON nome = vino
                ORDER BY vino;
)

page-schema VinoPage (
        nome: string;
        descrizione: string;
        foto: string;
        costo: real;
        vitigni: list_of (
                vitigno: string;
        );
        home: link( HomePage, *HomePage );
        // mostra il seguente link solo se loggato
        cliente_page: link( ClientePage, *ClientePage );
        vini_page: link( ViniPage, *ViniPage );
)

DB to page-schema VinoPage
parameter( nome )
(
        nome, descrizione, foto, costo:
                SELECT nome, descrizione, foto, costo
                FROM Vino
                WHERE nome = ?nome?;
        vitigno:
                SELECT vitigno
                FROM Vitigno
                WHERE vino = ?nome?;
)

page-schema EvasionePage unique (
        ordini: list_of (
                codice: link( id: integer, *UscitaPage );
                prezzo_totale: real;
        );
        home: link( HomePage, *HomePage );
        riepilogo_page: link( Riepilogo Vendite, *RiepilogoPage );
)

DB to page-schema EvasionePage
parameter( )
(
        id, prezzo_totale:
                SELECT id, prezzo_totale
                FROM Ordine
                WHERE NOT EXISTS(
```

```
                    SELECT *
                    FROM Uscita
                    WHERE ordine = id
                ) ORDER BY data;
)

page-schema UscitaPage (
        codice: integer;
        data: date;
        prezzo_totale: real;
        tipo: string;
        vini_ordinati: list_of (
                vino: string;
                anno: string;
                quantita: integer;
        );

        cf: string;
        nome: string;
        cognome: string;
        indirizzo: string;
        citta: string;
        login: string;

        bottiglie_disponibili: list_of (
                tipo_vino: string;
                anno_imbottigliamento: string;
                codici_bottiglie: list_of(
                        id_bottiglia: integer;
                );
        );

        uscita: form(
                bottiglie: checkbox( b_1, …, b_n );
                // codine ordine
                ordine: hidden;
                // numero di bottiglie visualizzate
                n_bot: hidden;
                ora: text;
                spedizioniere: text;
                invia: submt( );
        );
        home: link( HomePage, *HomePage );
        evasione_page: link( EvasionePage, *EvasionePage );
)

DB to page-schema UscitaPage
parameter( codice_ordine )
(
        codice, data, prezzo_totale, tipo, cf, nome, cognome, indirizzo, citta, login:
                SELECT id, data, prezzo_totale, tipo, cliente AS cf, nome, cognome, indirizzo, citta, login
                FROM ordine INNER JOIN Cliente ON cliente = cf
                WHERE id = ?codice_ordine?;
        vino, anno, quantita:
                SELECT vino, anno, quantita
                FROM QuantitaOrdine
```

```
            WHERE ordine = ?codice_ordine?
            ORDER BY vino, anno;
        id_bottiglia, tipo_vino, anno_imbottigliamento:
            SELECT id, B.vino AS vino, B.anno AS anno
            FROM Bottiglia B INNER JOIN QuantitaOrdine Q ON (B.anno = Q.anno AND B.vino = Q.vino)
            WHERE ordine = ?codice_ordine? AND uscita IS NULL
            ORDER BY B.vino, B.anno;
)

DB from page-schema UscitaPage(
        uscita:
            // verifico se l'ordine è già stato evaso
            if ( SELECT bolla FROM Uscita WHERE ordine = ?ordine? )
            // l'ordine e' già stato evaso, quindi non fare nulla
            then NO_OPERATION;
            else (
                // verifico che tutte le bottiglie siano ancora disponibili
                n = bottiglie selezionate;
                // scorro tutte le bottiglie selezionate
                for(i = 0; i < n; i++){
                    if( SELECT id
                        FROM Bottiglia B
                        WHERE id = ?id_bottiglia? AND uscita IS NULL
                      )
                    then BREAK;
                    end
                }
                // se tutte le bottiglie sono disponibili
                if i==n then(
                    // trovo il primo numero di bolla disponibile
                    bolla = (SELECT MIN(bolla + 1) AS bolla
                            FROM Uscita
                            WHERE bolla + 1 NOT IN (SELECT bolla FROM Uscita));
                    // inserisco l'uscita
                    INSERT INTO Uscita(bolla, ora, data, spedizioniere, ordine)
                    VALUES(?bolla?,?ora?,current_date,?spedizioniere?,?ordine?);
                    // per ogni bottiglia selezionata, setto l'uscita
                    for(i = 0; i < n; i++){
                        (UPDATE Bottiglia
                        SET uscita = ?bolla?
                        WHERE id = ?id_bottiglia?);
                    }
                )
                end
            )
            end
            // vado a evasionePage
            *EvasionePage
)

page-schema RiepilogoPage (
        riepilogo: list_of(
            vino: string;
            bottiglie_vendute: integer;
            bottiglie_prodotte: integer;
        );
        lista_riepiloghi: list_of(
```

```
                mese: integer;
                anno: string;
        );
        select_riepilogo: form(
                mese_anno: select( ma_1, …, ma_n );
                invia: submit( *RiepilogoPage );
        );
        home: link( HomePage, *HomePage );
        evasione_page: link( EvasionePage, *EvasionePage );
)

DB to page-schema RiepilogoPage
parameter( mese_riepilogo, anno_riepilogo )
(
        vino, bottiglie_prodotte, bottiglie_vendute:
                SELECT vino, prodotto, venduto
                FROM Riepilogo
                WHERE anno = ?anno_riepilogo? AND mese = ?mese_riepilogo?
                ORDER BY vino;
        anno, mese:
                SELECT DISTINCT anno, mese
                FROM Riepilogo
                ORDER BY anno DESC, mese DESC;
)

page-schema ErrorePage (
        home: link( HomePage, *HomePage );
)
```

STRUTTURA DELL'APPLICAZIONE WEB

MODULI

Tutto l'applicativo è distribuito in tre parti secondo la politica *mvc-2*:

- MODEL: comprende la classe *DBMS.java* e tutte le classi che descrivono i vari DataBean. *DBMS.java* si occupa di interrogare il DB e di manipolare i dati al suo interno. La comunicazione con *Main.java* avviene tramite DataBean in ambo le direzioni.
- CONTROL: comprende la sola classe *Main.java*, che non è altro che la servlet centrale che si occupa di ricevere tutte le richieste GET e POST, di richiedere a *DBMS.java* tutte le informazioni di cui si ha bisogno, e di inoltrare quest'ultime insieme alla richiesta HTTP alla pagina JSP appropriata.
- VIEW: comprende tutte le pagine JSP, le *librerie javascript* usate, le immagini della cantina, e le foto delle bottiglie di vino immagazzinate all'interno del server WEB.

I primi due moduli sono all'interno di due package differenti.

TRANSIZIONI IN JAVA

Per java ogni interrogazione o manipolazione di dati SQL sono singole transizioni. Se si desidera effettuare una transizione composta da più query è necessario disabilitare l'autocommit ed impostare il livello di isolamento della transizione. Inoltre bisogna gestire esplicitamente il commit e il rollback. Se la transizione accede ad una risorsa non disponibile, allora java lanciera un eccezione ed è compito del programmatore far ricominciare dall'inizio l'intera transizione. Il livello di isolamento più rigido è *SERIALIZABLE*.

LOGIN E SESSIONI

Una volta che un cliente e/o un impiegato fanno il login viene associata una sessione al client-host relativo, nella quale viene mantenuto l'attributo ***user_cliente*** per un cliente e ***user_impiegato*** per un impiegato. Entrambi gli attributi sono stringhe contenente il relativo nome utente. In maniera trasparente viene memorizzato anche un attributo che identifica l'id della sessione, che sarà lo stesso che verrà allocato nella macchina client dell'utente tramite *cookie.* In questo modo dopo aver fatto un login, è possibile accedere a tutte le pagine private senza dover rinviare user e pass ad ogni richiesta HTTP. Ogni sessione dura 600 secondi(10 minuti), dopo di che essa verrà "eliminata" e se avviene un altra richiesta da parte di un host a cui è scaduta la sessione, allora esso verrà reindirizzato alla HomePage con il relativo messaggio di sessione scaduta. Da HomePage è anche possibile effettuare il logout, in questo caso verrà eliminato l'attributo user_cliente e user_impiegato dalla relativa sessione, rispettivamente che venga effettuato il logout per un cliente o di un impiegato. **L'utilizzo di SESSIONI implica che il browser del client debba avere l'accettazione di COOKIE attiva per poter navigare sul sito**. L'utilizzo di due attributi di sessione è dipeso dal fatto che in questo modo è possibile garantire che un cliente e un impiegato possano essere loggati sulla stessa macchina, e non esclusivamente solo uno dei due.

PATH E CONTEXT

Il **context** tomcat del sito WEB è "***progetto***", mentre il path relativo da context per raggiungere la servlet Main è ./**main.** La porta per del server è 8080. Quindi l'URL per una richiesta HTTP al server sarà del tipo:

http://server_web:8080/progetto/main

GESTIONE RICHIESTE

La servlet centrale identifica e gestisce le richieste HTTP sulla base che esse siano GET o POST e dipendentemente dal valore del parametro *ps* passato nella richiesta.

Per le **richieste GET**, ***ps*** può assumere i seguenti valori:

- **cliente**: vengono recuperati i dati necessari e viene inoltrata la richiesta a **Cliente.jsp**
  Non ci c'è bisogno di ulteriori paramentri, perchè il nome utente lo si ricava dagli attributi di sessione.
- **vini**: vengono recuperati i dati necessari e viene inoltrata la richiesta a **Vini.jsp**
- **logout_cliente**: se esiste, viene rimosso l'attributo di sessione ***user_cliente*** e viene caricata **Home.jsp**
- **logout_impiegato**: se esiste, viene rimosso l'attributo di sessione ***user_impiegato*** e viene caricata **Home.jsp**
- **vino:** vengono recuperati i dati necessari e viene inoltrata la richiesta a **Vino.jsp**
  Paramentri passati nella richiesta:
  - vino: nome del vino su cui ottenere informazioni
- **ordine:** vengono recuperati i dati necessari e viene inoltrata la richiesta a **Ordine.jsp**
  Paramentri passati nella richiesta:
  - ordine: codice dell'ordine su cui si vogliono ottenere informazioni
- **evasione:** vengono recuperati i dati necessari e viene inoltrata la richiesta a **Evasione.jsp**
- **uscita**: vengono recuperati i dati necessari e viene inoltrata la richiesta a **Uscita.jsp**
  Paramentri passati nella richiesta:
  - ordine: codice dell'ordine che si vuole evadere
- **riepilogo**: vengono recuperati i dati necessari e viene inoltrata la richiesta a **Riepilogo.jsp**.
  Paramentri passati nella richiesta:
  - anno_mese: questa stringa indica quale riepilogo deve essere mostrato nella pagina. E'

una stringa del tipo aaaa_mm, dove aaaa indica l'anno e mm il mese. Se questo parametro non viene passato, allora *Main.java*, recupererà le informazioni dal riepilogo più recentemente memorizzato.

- **stringa vuota o paramentro assente**: vengono caricati i dati necessari e viene caricata **Home.jsp**

Tutti gli invii di dati tramite **form** vengono gestiti tramite **richieste POST** e ***ps*** può assumere i seguenti valori:

- **login_cliente**: viene inviato user e password di un cliente dalla **form cliente** di **Home.jsp**. Se questi sono validi, allora si crea una sessione valida e si carica **Cliente.jsp.** Altrimenti viene caricata la **Home.jsp** con relativo messaggio di user e pass errati.
  Parametri passati nella rchiesta:
  - login: user name del cliente
  - pass: password del cliente
- **login_impiegato**: viene inviato user e password di un cliente dalla **form impiegato** di **Home.jsp**. Se questi sono validi, allora si crea una sessione valida e si carica **Evasione.jsp.** Altrimenti viene caricata la **Home.jsp** con relativo messaggio di user e pass errati.
  Parametri passati nella rchiesta:
  - login: user name dell'impiegato
  - pass: password del'impiegato
- **registra_ordine**: dalla **form** di **Cliente.jsp** vengono inviate le quantità di bottiglie di vino ordinate dal cliente. Il comportamento è descritto in **db from page-schema ClientePage**.
  L'intera *transione* che controlla le quantità, ed eventualmente inserisce l'ordine nel DB, è *SERIALIZABLE*.
  Parametri passati nella richiesta:
  - size: numero di vini presenti nella form
  - oXX: quantita ordinata alla riga XX
  - aXX: anno relativo alla riga XX
  - vXX: vino relativo alla riga XX

  XX va da 0 a size.
- **registra_uscita**: dalla **form** di **Uscita.jsp** vengono inviati i codici delle bottiglie selezionate per il relativo ordine, e i dati sullo spedizioniere e sull'ora di spedizione. Il comportamento è descritto in **db from page-schema UscitaPage**.
  L'intera *transione* che controlla se l'ordine è stato spedito e se le bottiglie sono ancora disponibili, ed eventualmente inserisce l'uscita nel DB, è *SERIALIZABLE*.
  Paramentri passati nella richiesta:
  - ordine: codice dell'ordine da evadere
  - bXX : XX e' il numero della bottiglia nell'ordine in cui si presenta nella form. Esso contiene l'id della bottiglia
  - n_bot: numero bottiglie presenti nella form
  - spedizioniere: nome dello spedizioniere
  - ora: ora di spedizione

  xx va da 0 a n_bot.
- **stringa vuota o paramentro assente**: vengono caricati i dati necessari e viene caricata **Home.jsp**

GESTIONE DI ERRORI SULLE PAGINE JSP

Se si verifica un errore su una qualunque pagina JSP, allora viene caricata, automaticamente dal sistema, la pagina **Errore.jsp**. Essa non è altro che una pagina, con un link a **Home.jsp**.

PAGINE JSP E PARAMETRI PASSATI

- **Errore.jsp**: pagina che viene caricata automaticamente da tomcat, quando avviene un errore in un altra pagina JSP. Al suo interno vi è solo un semplice link a **Home.jsp.**
- **Home.jsp**: pagina nella quale viene presentata la cantina. Vengono recuperate sei foto casualmente(se disponibili) di bottiglie di vino dal DB e mostrate. Al suo interno ci sono anche due form per il login di impiegato e cliente. Una volta che è stato effettuato il login, allora la relativa form non verrà più mostrata, ma al suo posto sarà disponibile un link alla rispettiva pagina privata e un link per il logout.
  Parametri passati alla pagina:
  - foto: Vector, di VinoBean, che contiene i path delle foto da visualizzare
  - logC è un intero che vale:
    0. se il cliente non e' loggato, quindi verra' mostrata la form di login senza ulteriori messaggi
    1. se il cliente e' loggato, quindi verra' mostrato un link diretto alla sezione cliente, senza mostrare la form
    2. viene mostrata la form di login, e viene visualizzato il messaggio che user e pass sono errati
    3. viene mostrata la form di login, e viene visualizzato il messaggio che la sessione utente e' scaduta
  - logI è come logC, ma per l'impiegato
- **Evasione.jsp**: pagina nella quale vengono mostrati tutti gli ordini da evadere.
  Parametri passati alla pagina:
  - ordini: Vector di OrdineBean nei quali sono presenti codici e prezzi totali dei vari ordini da evadere
  - reg_uscita: se presente e' un oggetto boolen. Se vale true indica che l'uscita inviata dalla pagina UscitaPage e' stata registrata correttamente
  - reg_fail: se presente e' un oggetto boolen. Se vale true indica che l'uscita inviata dalla pagina UscitaPage non e' stata registrata
- **Cliente.jsp**: pagina nella quale vengono mostrati i dati sul cliente, compresi gli ordini effettuati. Inoltre vi è anche una form nella quale vengono mostrati tutti i vini disponibili, divisi per anno di imbottigliamento, con la relativa quantità disponibile, dove è possibile inserire il numero di bottiglie da ordinare. La form, prima di inviare i dati al server, esegue un controllo tramite una funzione in JQUERY. Questo script controlla:
  1. che il valori inseriti siano tutti un intero positivo
  2. che tutte le quantità inserite siano minori uguali alla quantita disponibile
  3. che sia stata ordinata almeno una bottiglia

  Se c'è un errore su una determinata riga della form questa viene mostrata in rosso.
  Nella stessa form, alla pressione del tasto reset, viene chiamata un altra funzione che riporta il colore di tutte le righe della form allo stato originale.
  Parametri passati alla pagina:
  - cliente: e' un oggetto ClienteBean che contiente i dati del cliente
  - ordini: e' un Vector di OrdineBean che contiene i vari ordini effettuati dal cliente
  - vini: e' un Vector di QuantitaBean che contiene il numero di bottiglie per ogni vino e anno, che possono essere ordinate

- reg_fail: e' un "oggetto boolean". Se esiste e il suo valore e' true, allora viene mostrato il messaggio che alcune bottiglie ordinate in precedenza non sono più disponibili

- **Ordine.jsp**: mostra i dettagli di un ordine, compresa la lista di vini acquistati, e i dettagli dell'evasione se spedito, altrimenti mostra un messaggio di “ordine non evaso”.
  Parametri passati alla pagina:
  - ordine: oggetto di tipo OrdineBean contenente i dettagli su un ordine
  - quantita: Vector di QuantitaBean, dove ognuno di questi contine nome, vino, anno e numero di bottiglie ordinate del relativo ordine
- **Uscita.jsp**: è una pagina in cui vengono mostrati le quantità ordinate di un relativo ordine e il cliente che ha effettuato l'ordine. C'è anche una form nella quale si può decidere quali specifiche bottiglie spedire, ed inserire il nome dello spedizioniere e l'ora di ritiro. La form prima di inviare tutti i dati al server, richiama una funzione JQUERY che controllerà:
  1. che il numero di bottiglie selezionate per ogni ordine sia corretto.
  2. che sia stato inserito lo spedizioniere
  3. che sia stata inserita l'ora di ritiro in modo corretto

  Se tutte le condizioni sono vere, allora invia i dati, altrimenti mostra a video un messaggio d'errore.
  Parametri passati alla pagina:
  - oc: oggetto di tipo OrdineClienteBean contenente i dettagli su un ordine e sul cliente che lo ha effettuato
  - quantita: Vector di QuantitaBean dove ognuno di questi contiene nome vino, anno e numero di bottiglie ordinate del relativo ordine
  - bottiglie: Vector di BottigliaBean dove ognuno di questi contiene l'id della bottiglia, l'anno di imbottigliamento e il nome del vino
- **Vini.jsp**: vengono visualizzate tutti i vini in vendita, con il relativo numero di bottiglie.
  Parametri passati alla pagina:
  - vini: e' un Vector di QuantitaBean che contiene il nome del relativo vino e il relativo numero di bottiglie disponibili
  - logCliente: e' un oggetto boolean. Se vale true indica che un cliente e' loggato e viene mostrato un link a ClientePage
- **Vino.jsp**: vengono visualizzate le informazioni su un vino specifico, compresa una foto della bottiglia.
  Parametri passati alla pagina:
  - vino: e' un oggetto di tipo VinoBean che contiene tutte le informazioni relative ad un vino
  - vitigli: e' un vector di VitignoBean, che contiene tutti i vitigli utilizzati per produrre il relativo tipo di vino
  - logCliente: e' un oggetto boolean. Se vale true indica che un cliente e' loggato e viene mostrato un link a ClientePage.
- **Riepilogo.jsp**: mostra un grafico nel quale vengono mostrate il numero di bottiglie vendute e prodotte per ogni vino di uno specifico mese dell'anno. Il grafico è stato realizzato con highcharts, che non è altro che una libreria JQUERY. Al suo interno c'è anche una form la quale permette di selezionare un altro mese e anno per visualizzare un altro rielilogo(passando da Main.java, e quindi ricaricando l'intera pagina).
  Parametri passati alla pagina:
  - riepilogo: e' un vector di RiepilogoBean, dove ogni elemento contiene nome vino, bottiglie prodotte, e le bottiglie vendute
  - list_anno_mese: e' un vector di AnnoMeseBean, dove ogni elemento contiene il mese e

anno di un riepilogo memorizzato nel DB

- anno_mese: e' un oggetto di tipo AnnoMeseBean che contiene il mese e anno del riepilogo che si sta visualizzando

LINK DA PAGINE PUBBLICHE A PAGINE PRIVATE

In **Vino.jsp** e **Vini.jsp** c'è un link a **Cliente.jsp** che compare solo se la sessione di un cliente è ancora attiva. Se non si è mai effettuato il login o la sessione è scaduta, allora non verranno mostrati i link appena citati.

CSS UTILIZZATI

Ogni pagina, tranne *Errore.jsp*, carica lo stile descritto nel file ***css/general_style.css***. La **Home.jsp** carica anche un altro file di stile che è ***css/home_style.css***.

PER ULTERIORI DETTAGLI SI CONSIGLIA DI VISIONARE IL CODICE.