Applicazioni web in Java

Tecnologie e progettazione di sistemi informatici e di telecomunicazioni

Versione n. 11 del 18/04/2025

prof. Roberto FULIGNI

Istituto Tecnico Tecnologico
"Giacomo Fauser"
Novara

Applicazioni web in Java

1. Servlet

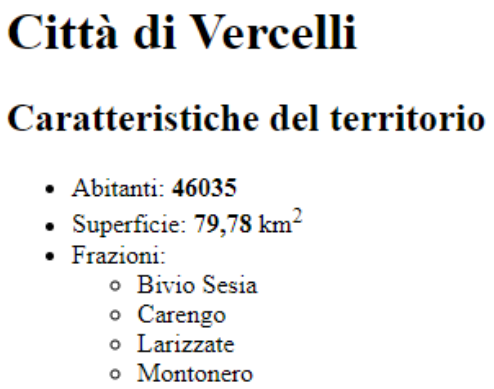
Esempio n. 1.1 (metodo GET)

Realizzare un’applicazione web che eroghi un servizio di consultazione di informazioni riguardanti alcuni capoluoghi di provincia del Piemonte mediante servlet.

Il client invia una richiesta HTTP con il metodo GET al seguente URL:

http://localhost:8080/info-piemonte/citta?sigla=vc

Inserendo nella query string il parametro *sigla* contenente la targa automobilistica della città desiderata.



Il server, dopo aver validato la richiesta, risponde inviando una pagina web in cui sono presenti alcune informazioni sulla città. Nel caso in cui non sia possibile soddisfare la richiesta, il server segnala l’errore con un apposito *status code*.

L’applicazione offre inoltre una pagina principale contenente un form con cui l’utente può selezionare, tramite un apposito elenco, il capoluogo da visualizzare.

Servlet (codifica Java)

```
public class CittaServlet extends javax.servlet.http.HttpServlet {
    protected void doPost(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
IOException {

    }

    protected void doGet(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
IOException {
        String nome;
        int abitanti;
```

```
        double superficie;
        String[] frazioni;
        String paramSigla = request.getParameter("sigla");

        if (paramSigla == null) {
            response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
            return;
        }

        if (paramSigla.equals("no") == true) {
            nome = "Novara";
            abitanti = 104268;
            superficie = 103.05;
            frazioni =  new String[] {"Agognate", "Lumellogno", "Veveri", "Vignale"};
        } else if (paramSigla.equals("vc") == true) {
            nome = "Vercelli";
            abitanti = 46035;
            superficie = 79.78;
            frazioni = new String[]{"Bivio Sesia", "Carengo", "Larizzate",
"Montonero"};
        }
        else {
            response.setStatus(HttpServletResponse.SC_NOT_FOUND);
            return;
        }

        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.format("<h1>Città di %s</h1>%n", nome);
        out.println("<h2>Caratteristiche del territorio</h2>");
        out.println("<ul>");
        out.format("<li>Abitanti: <strong>%d</strong></li>%n", abitanti);
        out.format("<li>Superficie: <strong>%.2f</strong> km<sup>2</sup></li>%n",
superficie);
        out.format("<li>Frazioni:<ul>%n");
        for(String f : frazioni)
            out.format("<li>%s</li>%n", f);

        out.format("</ul></li>%n");
        out.println("</ul>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

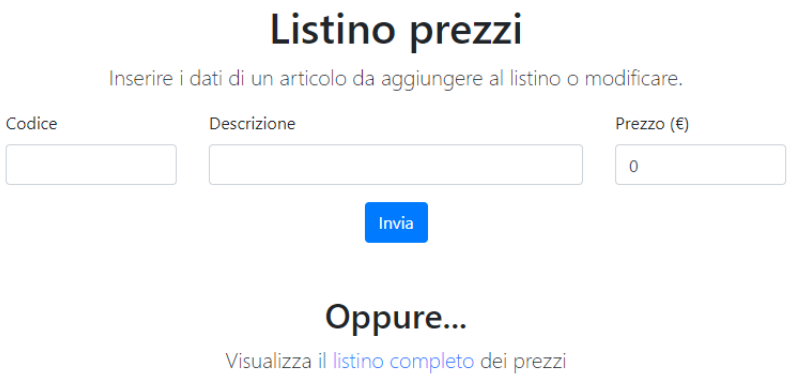
Pagina principale (html)

```
<h1>Capoluoghi di provincia del Piemonte</h1>
<form action="/info-piemonte/citta" method="get">
  <p>
    <label for="sigla">Capoluogo di provincia da visualizzare: </label>
    <select id="sigla" name="sigla">
```

```
      <option value="no">Novara</option>
      <option value="vc">Vercelli</option>
      <option value="to">Torino</option>
    </select>
  </p>
  <p>
    <input type="submit" value="Invia" />
  </p>
</form>
```

Esempio n. 1.2 (metodo POST)

Realizzare un’applicazione web che eroghi un servizio di visualizzazione e aggiornamento di un listino prezzi. L’applicazione offre un’interfaccia grafica attraverso cui un utente può visualizzare i prezzi di un certo numero di articoli (ogni articolo è caratterizzato da un codice intero, una descrizione e un prezzo in Euro). L’applicazione permette anche di aggiornare i dati degli articoli presenti oppure aggiungere nuovi articoli al listino.



L’interfaccia contenuta nella pagina principale dell’applicazione (URL http://localhost:8080/listino-prezzi) è suddivisa in due parti:

1) *Inserimento/aggiornamento articoli*: contiene un form HTML in cui sono inseriti tre dati: codice (numero intero), descrizione (stringa), prezzo (numero reale). I dati sono inviati all’applicazione con il metodo POST e sono processati secondo la seguente regola:
   - Se il codice inviato non è presente, un nuovo articolo è aggiunto in coda al listino.
   - Se il codice è già presente nel listino, i dati dell’articolo corrispondente (descrizione e prezzo) sono aggiornati con le informazioni inviate dall’utente.

   Terminato l’aggiornamento, l’applicazione restituisce all’utente l’esito dell’operazione e il nuovo listino prezzi.

2) *Visualizzazione listino*: richiama il listino dei prezzi aggiornato all’ultima modifica. La richiesta è inviata con il metodo GET.

Entrambe le operazioni sono gestite da una servlet associata al seguente URL: http://localhost:8080/listino-prezzi/listino .

A partire dalla seguente operazione di inserimento:

L’applicazione restituisce:

Pagina principale (html + css)

```
<!doctype html>
<html lang="it">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">
    <title>Listino prezzi</title>
    <style type="text/css">
        body {
            padding-top: 2rem;
        }

        main {
            padding: 1rem 1.5rem;
            text-align: center;
        }
    </style>
</head>
<body>
```

```
<main role="main" class="container">
    <h1>Listino prezzi</h1>
    <p class="lead">Inserire i dati di un articolo da aggiungere al listino o
                    modificare.</p>
    <div class="row">
        <div class="col-sm-10 offset-sm-1 col-md-8 offset-md-2 text-center">
            <form action="/listino-prezzi/listino" method="post">
                <div class="row text-left">
                    <div class="form-group col-md-3">
                        <label for="codice">Codice</label>
                        <input type="number" id="codice" name="codice"
                               class="form-control"/>
                    </div>
                    <div class="form-group col-md-6">
                        <label for="descr">Descrizione</label>
                        <input type="text" id="descr" name="descr"
                               class="form-control"/>
                    </div>
                    <div class="form-group col-md-3">
                        <label for="prezzo">Prezzo (&euro;)</label>
                        <input type="number" value="0" min="0" step="0.01"
                              data-number-to-fixed="2"
                              data-number-stepfactor="100"
                              id="prezzo" name="prezzo" class="currency form-control"/>
                    </div>
                </div>
                <button type="submit" class="btn btn-primary">Invia</button>
            </form>
        </div>
    </div>
    <h2 style="margin-top: 3rem">Oppure...</h2>
    <p class="lead">Visualizza il <a href="listino">listino completo</a> dei prezzi</p>
</main>
</body>
</html>
```

Servlet (codifica Java)

```
public class ListinoServlet extends javax.servlet.http.HttpServlet {

    private Listino listino;

    protected void doPost(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
IOException {
        String strCodice = request.getParameter("codice");
        String strDescr = request.getParameter("descr");
        String strPrezzo = request.getParameter("prezzo");

        try {
            boolean flagNuovo = listino.aggiorna(strCodice, strDescr, strPrezzo);
            PrintWriter out = response.getWriter();
            inizioRisposta(out);
            out.println("<div class='alert alert-success' role='alert'>");
            out.println(String.format("Articolo cod. %s %s con successo</div>",
                    strCodice, flagNuovo ? "Inserito" : "modificato"));
            stampaListino(out);
            fineRisposta(out);
```

```
        }
        catch (Exception e) {
            response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
            return;
        }
    }

    protected void doGet(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
IOException {
        PrintWriter out = response.getWriter();
        inizioRisposta(out);
        stampaListino(out);
        fineRisposta(out);
    }

    private void stampaListino(PrintWriter pw) {
        pw.println("<h1>Listino prezzi</h1>");
        ArrayList<Articolo> lista = listino.getArticoli();
        if (lista.isEmpty()) {
            pw.println("<p style='margin: 2em 0'>Listino vuoto</p>");
            return;
        }
        pw.println("<div class='row'>");
        pw.println("<div class='col-sm-10 offset-sm-1 col-md-8 offset-md-2'>");
        pw.println("<table class='table table-striped text-left'>");
        pw.println("<thead>");
        pw.println("<tr>");
        pw.println("<th scope='col'>Codice</th>");
        pw.println("<th scope='col'>Articolo</th>");
        pw.println("<th scope='col'>Prezzo</th>");
        pw.println("</tr>");
        pw.println("</thead>");
        pw.println("<tbody>");
        for(Articolo a : lista) {
            pw.println("<tr>");
            pw.println(String.format("<th scope='row'>%d</th>", a.getCodice()));
            pw.println(String.format("<td>%s</td>", escapeHTML(a.getDescrizione())));
            pw.println(String.format("<td>&euro; %.2f</td>", a.getPrezzo()));
            pw.println("</tr>");
        }
        pw.println("</tbody>");
        pw.println("</table>");
        pw.println("</div>");
        pw.println("</div>");
    }

    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        ServletContext ctx = getServletContext();
        String nome = ctx.getRealPath("WEB-INF/listino.txt");
        listino = new Listino(nome);
        listino.leggi();
    }

    public void destroy() {
        listino.scrivi();
    }

    private void inizioRisposta(PrintWriter pw) {
        pw.println("<!doctype html>");
        pw.println("<html lang='it'>");
```

```
        pw.println("<head>");
        pw.println("<meta charset='utf-8'>");
        pw.println("<meta name='viewport' content='width=device-width, initial-scale=1, shrink-
to-fit=no'>");
        pw.println("<link
href='https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css'
rel='stylesheet'");

pw.println("integrity='sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh'
crossorigin='anonymous'>");
        pw.println("<title>Listino prezzi</title>");
        pw.println("<style type='text/css'>");
        pw.println("body {padding-top: 0rem;}");
        pw.println("main {padding: 1rem 1.5rem; text-align: center;}");
        pw.println("</style>");
        pw.println("</head>");
        pw.println("<body>");
        pw.println("<main role='main' class='container'>");

    }

    private void fineRisposta(PrintWriter pw) {
        pw.println("<div class='row'>");
        pw.println("<div class='col-sm-8 offset-sm-2 col-md-6 offset-md-3 text-center'>");
        pw.println("<a class='btn btn-primary' href='/listino-prezzi'>Torna alla pagina
iniziale</a>");
        pw.println("</div>");
        pw.println("</div>");
        pw.println("</main>");
        pw.println("</body>");
        pw.println("</html>");
    }

    public static String escapeHTML(String str) {
        StringBuilder sb = new StringBuilder(str.length());
        for (int i = 0; i < str.length(); i++) {
            char ch = str.charAt(i);
            if (ch == '&' || ch == '"' || ch == '<' || ch == '>' || ch > 127) {
                sb.append(String.format("&#%d;", (int) ch));
            } else {
                sb.append(ch);
            }
        }
        return sb.toString();
    }
}
```

Classe Listino (codifica Java)

```
public class Listino {
    private String nomeFile;

    ArrayList<Articolo> articoli = new ArrayList<>();

    public Listino(String nf) {
        nomeFile = nf;
    }

    public ArrayList<Articolo> getArticoli() {
        return articoli;
    }
```

```
    public void leggi() {
        articoli.clear();
        try (BufferedReader br = new BufferedReader(new FileReader(nomeFile))) {
            String linea;
            while ((linea = br.readLine()) != null) {
                Articolo a = Articolo.parseArticolo(linea);
                articoli.add(a);
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public void scrivi() {
        try (PrintWriter pw = new PrintWriter(new FileWriter(nomeFile))) {
            for(Articolo a : articoli)
                pw.println(a.toString());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public boolean aggiorna(String c, String d, String p) {
        Articolo nuovo = Articolo.generaArticolo(c, d, p);
        for(Articolo a : articoli) {
            if (a.getCodice() == nuovo.getCodice()) {
                a.setDescrizione(nuovo.getDescrizione());
                a.setPrezzo(nuovo.getPrezzo());
                return false;
            }
        }
        articoli.add(nuovo);
        return true;
    }
}
```

Classe Articolo (codifica Java)

```
public class Articolo {
    int codice;
    String descrizione;
    double prezzo;

    public Articolo(int cod, String descr, double pr) {
        codice = cod;
        descrizione = descr;
        prezzo = pr;
    }
    @Override
    public String toString() {
        return String.format("%d|%s|%.2f", codice, descrizione, prezzo);
    }

    static Articolo parseArticolo(String s) {
        String[] parti = s.split("\\|");
        return generaArticolo(parti[0], parti[1], parti[2]);
    }

    public int getCodice() {
        return codice;
    }
```

```
    public String getDescrizione() {
        return descrizione;
    }

    public void setDescrizione(String descrizione) {
        this.descrizione = descrizione;
    }

    public double getPrezzo() {
        return prezzo;
    }

    public void setPrezzo(double prezzo) {
        this.prezzo = prezzo;
    }

    public static Articolo generaArticolo(String strCodice, String strDescr, String strPrezzo) {
        Articolo a = null;
        int codice = Integer.parseInt(strCodice);
        double prezzo = Double.parseDouble(strPrezzo.replace(",", "."));
        String descr = strDescr.replace("|", "/");
        a = new Articolo(codice, descr, prezzo);
        return a;
    }
}
```

2. Gestione delle sessioni

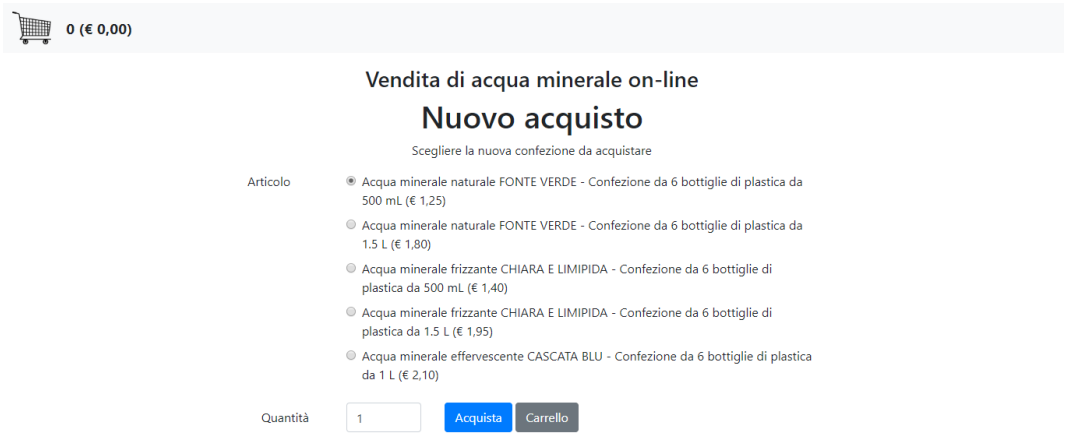
Esempio n. 2.1 (Carrello della spesa)

Realizzare un’applicazione web basata su servlet che simuli la vendita on-line di acque minerali.

Struttura del server

L’applicazione web (*Application Context*: **/vendita-acqua**), attraverso la servlet *VenditaServlet*, è in grado di svolgere le seguenti operazioni:

- **acquisto**: fornisce, all’interno di un form HTML, un elenco di confezioni di acque minerali acquistabili;

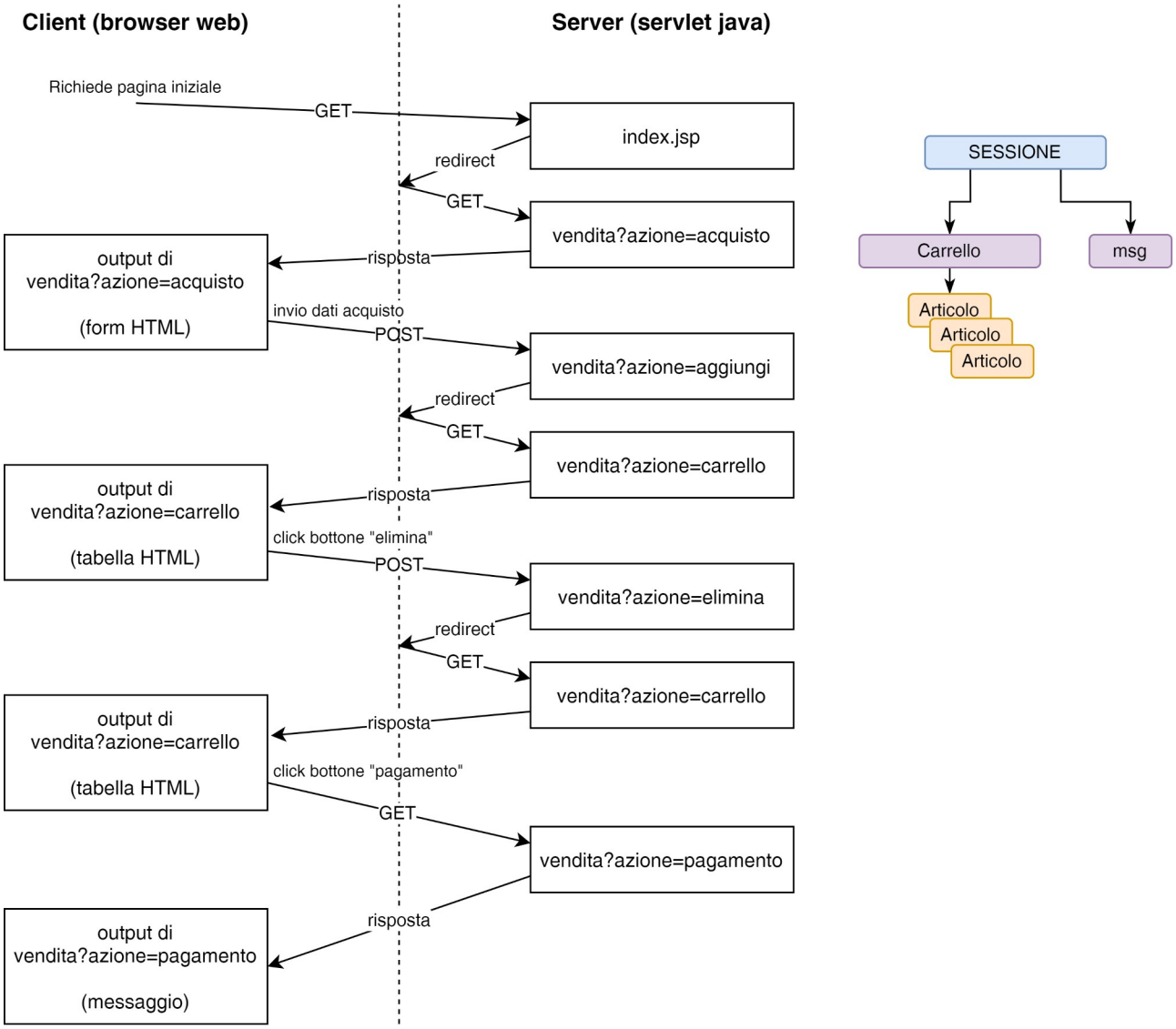


- **aggiungi**: inserisce la confezione acquistata nel carrello della spesa;
- **carrello**: mostra il contenuto del carrello;

- **elimina**: rimuove una data confezione dal carrello;
- **pagamento**: conclude l’acquisto simulando il pagamento della spesa.

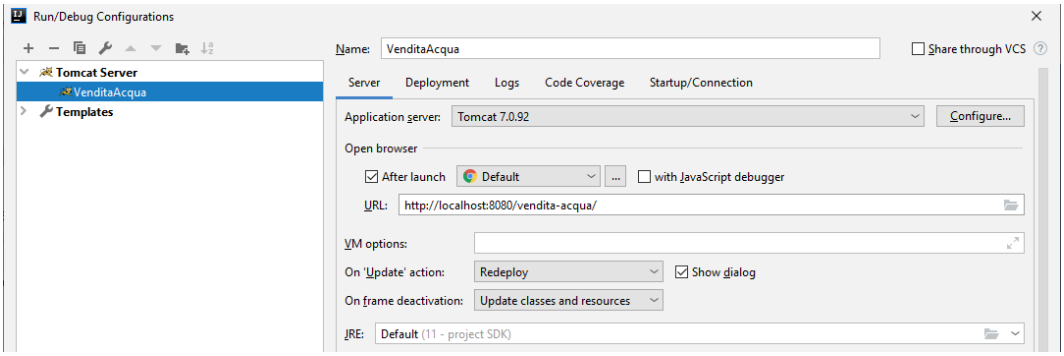
Le operazioni da eseguire sono indicate dal client specificando nelle richieste HTTP il parametro *operazione*. Le operazioni *acquisto*, *carrello* e *pagamento* sono inviate con il metodo GET, le altre con POST.

A ogni utente è associato un carrello della spesa in cui sono memorizzati i prodotti acquistati. Il carrello è gestito come attributo della sessione Java. Nella sessione è inoltre presente un atttributo **msg** utilizzato per visualizzare di messaggi durante le varie operazioni**.**



Per evitare, durante un eventuale refresh della pagina web, che un articolo selezionato nel form HTML e inviato al server con il metodo POST sia processato ripetutamente, causando la comparsa di articoli duplicati nel carrello, si ricorre al pattern POST/REDIRECT/GET[1]: il server, ricevuta una richiesta POST, anziché elaborare la pagina web di risposta, invia un codice di risposta della famiglia HTTP 3xx (Redirect) che porta il client a formulare una nuova richiesta GET per ottenere il contenuto aggiornato del carrello.

Configurazione



File web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">
    <servlet>
        <servlet-name>VenditaServlet</servlet-name>
        <servlet-class>VenditaServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>VenditaServlet</servlet-name>
        <url-pattern>/vendita</url-pattern>
    </servlet-mapping>
</web-app>
```

Pagina iniziale (index.jsp)

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%  response.sendRedirect("vendita?azione=acquisto"); %>
```

Servlet (codifica Java)

```
public class VenditaServlet extends javax.servlet.http.HttpServlet {
    ArrayList<Acqua> disponibili;
    protected void doPost(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
IOException {
        /*
         * Le richieste effettuate con il metodo POST possono specificare le seguenti operazioni:
         *   "aggiungi": aggiunge un articolo al carrello della spesa
```

1 https://en.wikipedia.org/wiki/Post/Redirect/Get
https://www.youtube.com/watch?v=DCC7ufuFD2w

```
         *   "elimina": elimina un articolo dal carrello
         */
        response.setContentType("text/html; charset=UTF-8");
        String operazione = request.getParameter("operazione");
        if (operazione == null) operazione = "";
        switch(operazione) {
            case "aggiungi":
                aggiungi(request, response);
                break;
            case "elimina":
                elimina(request, response);
                break;
            default:
                response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
                return;
        }
    }

    private void aggiungi(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        int codice, qta;
        try {
            codice = Integer.parseInt(request.getParameter("codice"));
            qta = Integer.parseInt(request.getParameter("qta"));
        }
        catch (NumberFormatException ex) {
            response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
            return;
        }
        HttpSession sessione = request.getSession();
        Carrello c = getCarrello(sessione);
        for(Acqua a : disponibili) {
            if (codice == a.getCodice()) {
                c.aggiungi(new Articolo(codice, a.getDescrizione(), a.getPrezzo(), qta));
                sessione.setAttribute("msg", "Acquisto completato. Un nuovo articolo è
                      stato aggiunto al carrello.");
                response.sendRedirect("vendita?operazione=carrello");
                return;
            }
        }
        response.setStatus(HttpServletResponse.SC_NOT_ACCEPTABLE);
    }

    private void elimina(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        int riga;
        try {
            riga = Integer.parseInt(request.getParameter("riga"));
        }
        catch (NumberFormatException ex) {
            response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
            return;
        }
        HttpSession sessione = request.getSession();
        Carrello c = getCarrello(sessione);
        c.elimina(riga);
        sessione.setAttribute("msg", "Articolo rimosso dal carrello con successo.");
        response.sendRedirect("vendita?operazione=carrello");
    }
```

```
    protected void doGet(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
IOException {
        /*
         * Le richieste effettuate con il metodo GET possono specificare le seguenti operazioni:
         *   "acquisto": richiede l'elenco dei prodotti acquistabili (operazione di default)
         *   "carrello": richiede il contenuto del carrello della spesa
         *   "pagamento": conclude l'acquisto pagando gli articoli presenti nel carrello
         */
        response.setContentType("text/html; charset=UTF-8");
        String operazione = request.getParameter("operazione");
        if (operazione == null) operazione = "acquisto";
        switch (operazione) {
            case "acquisto":
                acquisto(request, response);
                break;
            case "carrello":
                carrello(request, response);
                break;
            case "pagamento":
                pagamento(request, response);
                break;
            default:
                response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
                return;
        }

    }

    private void acquisto(HttpServletRequest request, HttpServletResponse response)
       throws IOException {
        HttpSession sessione = request.getSession();
        Carrello c = getCarrello(sessione);
        PrintWriter out = response.getWriter();
        inizioRisposta(out, c);
        out.println("<h1>Nuovo acquisto</h1>");
        out.println("<p>Scegliere la nuova confezione da acquistare</p>");
        out.println("<div class='row'>");
        out.println("<div class='col-sm-12 col-md-8 offset-md-2 text-center'>");
        out.println("<form action='vendita' method='post'>");
        // Campo nascosto per indicare l'operazione da eseguire
        out.println("<input type='hidden' name='operazione' value='aggiungi' />");
        out.println("<fieldset class='form-group'>");
        out.println("<div class='row text-left'>");
        out.println("<legend class='col-form-label col-sm-2 pt-0'>Articolo</legend>");
        out.println("<div class='col-sm-10'>");

        for(int i = 0; i < disponibili.size(); i++) {
            Acqua a = disponibili.get(i);
            out.println("<div class='form-check mb-2'>");
            out.print("<input class='form-check-input' type='radio' name='codice' ");
            out.format("id='articolo%d' value='%d' %s />", a.getCodice(),
                      a.getCodice(), i == 0 ? "checked" : "");
            out.format("<label class='form-check-label' for='articolo%d'>%s (&euro;
                      %.2f)</label>",
                    a.getCodice(),
                    escapeHTML(a.getDescrizione()),
                    a.getPrezzo()
            );
            out.println("</div>");
        }
```

```
        out.println("</div>");
        out.println("</div>");
        out.println("</fieldset>");
        out.println("<div class='form-group row'>");
        out.println("<label for='qta' class='col-sm-2 col-form-label'>Quantit&agrave;</label>");
        out.println("<div class='col-sm-2'>");
        out.println("<input type='number' class='form-control' id='qta' name='qta'
              value='1' min='1' max='20' />");
        out.println("</div>");
        out.println("<div class='col-sm-8 text-left'>");
        out.println("<button type='submit' class='btn btn-primary'>Acquista</button>");
        out.println("<a class='btn btn-secondary' href='vendita?operazione=carrello'
              role='button'>Carrello</a>");
        out.println("</div>");
        out.println("</div>");
        fineRisposta(out);
    }

    private void carrello(HttpServletRequest request, HttpServletResponse response)
       throws IOException {
        HttpSession sessione = request.getSession();
        Carrello c = getCarrello(sessione);
        PrintWriter out = response.getWriter();
        inizioRisposta(out, c);
        String msg = (String) sessione.getAttribute("msg");
        if (msg != null) {
            out.println("<div id='messaggio' class='alert alert-success' role='alert'>");
            out.println(msg);
            out.println("</div>");
            out.println("<script>");
            out.println("window.setTimeout(function() {");
            out.println(" $('#messaggio').fadeTo(500, 0).slideUp(500, function() {
                      $(this).remove();});");
            out.println("}, 2000);");
            out.println("</script>");
            sessione.removeAttribute("msg");
        }

        out.println("<h1>Carrello della spesa</h1>");
        out.println("<table class='table'>");
        out.println("<thead class='thead-light'>");
        out.println("<tr>");
        out.println("<th scope='col'>N.</th>");
        out.println("<th scope='col'>Codice</th>");
        out.println("<th scope='col'>Descrizione</th>");
        out.println("<th scope='col'>Costo unitario</th>");
        out.println("<th scope='col'>Quantit&agrave;</th>");
        out.println("<th scope='col'>Costo totale</th>");
        out.println("<th> </th>");
        out.println("</tr>");
        out.println("</thead>");
        out.println("<tbody>");
        int riga = 1;
        for(Articolo a : c.getArticoli()) {
            out.println("<tr>");
            out.println(String.format("<td>%d</td>", riga));
            out.println(String.format("<td>%d</td>", a.getCodice()));
            out.println(String.format("<td style='text-align: left'>%s</td>",
                      escapeHTML(a.getDescrizione())));
            out.println(String.format("<td>&euro; %.2f</td>", a.getCostoUnitario()));
            out.println(String.format("<td>%d</td>", a.getQuantita()));
            out.println(String.format("<td>&euro; %.2f</td>", a.getCostoTotale()));
```

```
            out.println("<td>");
            out.println("<form action='vendita' method='post'>");
            out.println(String.format("<input type='hidden' name='operazione'
                    value='elimina'/>", riga));
            out.println(String.format("<input type='hidden' name='riga' value='%d'/>", riga));
            out.println("<button type='submit' class='btn btn-danger
                    btn-sm'>Elimina</button>");
            out.println("</form>");
            out.println("</td>");
            out.println("</tr>");
            riga++;
        }
        out.println("</tbody>");
        out.println("<tfoot>");
        out.println("<tr>");
        out.println("<td colspan='5' class='text-right'>Totale:</td>");
        out.println(String.format("<td><strong>&euro; %.2f</strong></td>", c.totale()));
        out.println("<td> </td>");
        out.println("</tr>");
        out.println("</tfoot>");
        out.println("</table>");
        out.println("<div>");
        out.println("<a class='btn btn-primary' href='vendita?operazione=acquisto'
              role='button'>Nuovo acquisto</a>");
        out.println("<a class='btn btn-success' href='vendita?operazione=pagamento'
              role='button'>Paga e concludi</a>");
        out.println("</div>");

        fineRisposta(out);
    }

    private void pagamento(HttpServletRequest request, HttpServletResponse response)
              throws IOException {
        HttpSession sessione = request.getSession();
        Carrello c = getCarrello(sessione);
        double pagato = c.totale();
        sessione.invalidate();

        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        inizioRisposta(out, null);
        out.println("<h1>Pagamento eseguito</h1>");
        out.println("<div class='row pt-4 pb-4'>");
        out.println("<div class='col-sm-12 offset-md-2 col-md-8 text-center'>");
        out.println("<p>L'acquisto degli articoli presenti nel carrello è stato
              effettuato correttamente e il carrello è stato svuotato.");
        out.println(String.format("L'importo pagato è pari a <strong>&euro;
              %.2f</strong>.</p>", pagato));
        out.println("</div>");
        out.println("</div>");
        out.println("<a class='btn btn-primary' href='vendita?operazione=acquisto'
              role='button'>Nuovo acquisto</a>");
        fineRisposta(out);
    }

    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        disponibili = new ArrayList<Acqua>();
        disponibili.add(new Acqua(1, "Acqua minerale naturale FONTE VERDE – Confezione da
              6 bottiglie di plastica da 500 mL", 1.25));
        disponibili.add(new Acqua(2, "Acqua minerale naturale FONTE VERDE - Confezione
```

```
            da 6 bottiglie di plastica da 1.5 L", 1.80));
        disponibili.add(new Acqua(3, "Acqua minerale frizzante CHIARA E LIMIPIDA -
            Confezione da 6 bottiglie di plastica da 500 mL", 1.40));
        disponibili.add(new Acqua(4, "Acqua minerale frizzante CHIARA E LIMIPIDA -
            Confezione da 6 bottiglie di plastica da 1.5 L", 1.95));
        disponibili.add(new Acqua(5, "Acqua minerale effervescente CASCATA BLU -
            Confezione da 6 bottiglie di plastica da 1 L", 2.10));
    }

    private void inizioRisposta(PrintWriter pw, Carrello c) {
        pw.println("<!doctype html>");
        pw.println("<html lang='it'>");
        pw.println("<head>");
        pw.println("<meta charset='utf-8'>");
        pw.println("<meta name='viewport' content='width=device-width, initial-scale=1,
            shrink-to-fit=no'>");
        pw.println("<link href='https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/
            bootstrap.min.css' rel='stylesheet'");

        pw.println("integrity='sha384-Vkoo8x4CGsO3+Hhxv8T/
            Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh' crossorigin='anonymous'>");
        // Per visualizzare i messaggi a scomparsa è richiesta la libreria JQuery
        pw.println("<script src='https://code.jquery.com/jquery-3.4.1.js' ");
        pw.println("integrity='sha256-WpOohJOqMqqyKL9FccASB9O0KwACQJpFTUBLTYOVvVU='
       crossorigin='anonymous'></script>");
        pw.println("<title>Vendita acqua minerale</title>");
        pw.println("<style type='text/css'>");
        pw.println("body {padding-top: 0rem;}");
        pw.println("main {padding: 1rem 1.5rem; text-align: center;}");
        pw.println("</style>");
        pw.println("</head>");
        pw.println("<body>");
        pw.println("<nav style='height: 64px;' class='navbar navbar-expand-sm bg-light'>");
        if (c != null) {
            pw.println("<img width='50' src='img/carrello.png' />");
            pw.println(String.format("<strong style='padding-left: 1em; font-size:
                      larger'>%d (&euro; %.2f)</strong>",
                    c.numeroArticoli(), c.totale()));
        }
        pw.println("</nav>");
        pw.println("<main role='main' class='container'>");
        pw.println("<h3>Vendita di acqua minerale on-line</h3>");

    }

    private void fineRisposta(PrintWriter pw) {
        pw.println("</main>");
        pw.println("</body>");
        pw.println("</html>");
    }

    public static String escapeHTML(String str) {
        StringBuilder sb = new StringBuilder(str.length());
        for (int i = 0; i < str.length(); i++) {
            char ch = str.charAt(i);
            if (ch == '&' || ch == '"' || ch == '<' || ch == '>' || ch > 127) {
                sb.append(String.format("&#%d;", (int) ch));
            } else {
                sb.append(ch);
            }
        }
```

```
        return sb.toString();
    }

    private Carrello getCarrello(HttpSession sessione) {
        Carrello c = (Carrello) sessione.getAttribute("carrello");
        if (c == null) {
            c = new Carrello();
            sessione.setAttribute("carrello", c);
        }
        return c;
    }
}
```

Classe Carrello

```
public class Carrello {
    ArrayList<Articolo> articoli;

    public Carrello() {
        articoli = new ArrayList<>();
    }

    public ArrayList<Articolo> getArticoli() {
        return articoli;
    }

    public int numeroArticoli() {
        return articoli.size();
    }

    public void aggiungi(Articolo a) {
        articoli.add(a);
    }

    public void elimina(int riga) {
        if (riga > 0 && riga <= articoli.size())
            articoli.remove(riga - 1);
    }

    public double totale() {
        double ris = 0;
        for(Articolo a : articoli)
            ris += a.getCostoTotale();
        return ris;
    }
}
```

Classe Articolo

```
public class Articolo {
    int codice;
    String descrizione;
    double costoUnitario;
    int quantita;

    public Articolo(int codice, String descrizione, double costoUnitario, int quantita) {
```

```
        this.codice = codice;
        this.descrizione = descrizione;
        this.costoUnitario = costoUnitario;
        this.quantita = quantita;
    }

    public int getCodice() {
        return codice;
    }

    public String getDescrizione() {
        return descrizione;
    }

    public double getCostoUnitario() {
        return costoUnitario;
    }

    public double getCostoTotale() {
        return costoUnitario * quantita;
    }

    public int getQuantita() {
        return quantita;
    }
}
```

Classe Acqua (utilizzata per creare un piccolo archivio in memoria)

```
public class Acqua {
    int codice;
    String descrizione;
    double prezzo;

    public Acqua(int codice, String descrizione, double prezzo) {
        this.codice = codice;
        this.descrizione = descrizione;
        this.prezzo = prezzo;
    }

    public int getCodice() {
        return codice;
    }

    public String getDescrizione() {
        return descrizione;
    }

    public double getPrezzo() {
        return prezzo;
    }
}
```

3. Accesso al database con JDBC

Esempio n. 3.1 (Interrogazioni SQL e inserimento dati)

Realizzare un’applicazione web per la gestione dell’anagrafica dei clienti di una piccola azienda. Ogni cliente è caratterizzato da: id numerico, cognome, nome e data di nascita. L’applicazione deve consentire di ottenere l’elenco completo dei clienti registrati, ricercare un dato cliente tramite id e aggiungere un nuovo cliente.

File SQL (MariaDB)

```
CREATE DATABASE IF NOT EXISTS dbanagrafica
    CHARACTER SET = 'utf8'
    COLLATE = 'utf8_unicode_ci';

USE dbanagrafica;
DROP TABLE IF EXISTS anagr_clienti;
CREATE TABLE anagr_clienti
(
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    cognome NVARCHAR(50) NOT NULL,
    nome NVARCHAR(50) NOT NULL,
    data_nascita DATE NOT NULL
);

INSERT INTO anagr_clienti (cognome, nome, data_nascita) VALUES
    ('Rossi', 'Mario', '1995-06-30'),
    ('Bianchi', 'Carla', '1998-02-21'),
    ('Verdi', 'Luca', '1992-10-02'),
    ('Neri', 'Paola', '1994-12-05');
```

Pagina index.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
    <title>Anagrafica SQL</title>
</head>
<body>
    <h1>Gestione anagrafica clienti</h1>
    <p>
        <a href="anagrafica">Visualizza dati anagrafici dei clienti</a>
    </p>
    <h2>Nuovo cliente</h2>
    <form action="anagrafica" method="post">
        <p>
            <label for="cognome" style="display: inline-block; width:
7em">Cognome:</label>
            <input type="text" name="cognome" id="cognome" size="30" maxlength="50"/>
        </p>
```

```
        <p>
            <label for="nome" style="display: inline-block; width: 7em">Nome:</label>
            <input type="text" name="nome" id="nome" size="30" maxlength="50"/>
        </p>
        <p>
            <label for="nascita"style="display: inline-block; width: 7em">Data di
                nascita:</label>
            <input type="text" name="nascita" id="nascita" size="20" maxlength="20"/>
        </p>
        <p>
            <input type="submit" value="Invia" />
            <input type="reset" value="Reimposta" />
        </p>
    </form>
    <h2>Ricerca cliente</h2>
    <form action="anagrafica" method="get">
        <p>
            <label for="idcliente" >ID del cliente da cercare:</label>
            <input type="number" name="idcliente" id="idcliente" value="1" />
        </p>
        <p>
            <input type="submit" value="Cerca" />
            <input type="reset" value="Reimposta" />
        </p>
    </form>
</body>
</html>
```

Classe AnagraficaServlet

```
package edu.fauser.netlab.anagraficasql;

/* Scaricare dal sito del corso (classi quinte) la libreria DbUtility,
 * creare la cartella lib e copiare al suo interno il file .jar.
 * In IntelliJ, fare click con il tasto destro sulla cartella lib,
 * quindi selezionare "Add as Library...".
 */
import edu.fauser.DbUtility;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;

@WebServlet(name = "AnagraficaServlet", value = "/anagrafica")
public class AnagraficaServlet extends HttpServlet {

    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        /* Impostazione dei parametri di configurazione per l'accesso al database.
         * Questi parametri possono essere recuperati dalla servlet durante l'elaborazione delle richieste.
         * La classe edu.fauser.DbUtility è utilizzabile anche per il collegamento ai database di Labs3.
         */
        DbUtility dbu = DbUtility.getInstance(getServletContext());
        // Credenziali per l'accesso al database di produzione di Labs3 (Mariadb) con pool di 2 connessioni.
        dbu.setProdCredentials("jdbc:mariadb://localhost:3306/db123?maxPoolSize=2&pool", "db123", "******");
```

```
        // Credenziali per l'accesso a un database di supporto in fase di sviluppo
        //dbu.setDevCredentials("jdbc:mariadb://localhost:3306/dbanagrafica?maxPoolSize=2&pool", "root", "");
        dbu.setDevCredentials("jdbc:h2:"+getServletContext().getRealPath("/WEB-INF/dbanagrafica"), "sa", "");

        // In alternativa allo script SQL di inzializzazione da eseguire sul server di database, è possibile
        // creare e popolare le tabelle tramite codice Java.
        try(Connection cn = DriverManager.getConnection(dbu.getUrl(), dbu.getUser(), dbu.getPassword());
            Statement st = cn.createStatement()) {
            String sql = "CREATE TABLE IF NOT EXISTS anagr_clienti (" +
                    "id INT NOT NULL AUTO_INCREMENT PRIMARY KEY, "+
                    "cognome NVARCHAR(50) NOT NULL, " +
                    "nome NVARCHAR(50) NOT NULL, " +
                    "data_nascita DATE NOT NULL);";

            st.executeUpdate(sql);

            // Verifica l'assenza di dati della tabella
            if (st.executeQuery("SELECT 1 FROM anagr_clienti LIMIT 1").next() == false) {
                // Poiché la tabella è vuota, si aggiungono alcuni record di test
                sql = "INSERT INTO anagr_clienti (cognome, nome, data_nascita) VALUES " +
                        "('Rossi', 'Mario', '1995-06-30')," +
                        "('Bianchi', 'Carla', '1998-02-21')," +
                        "('Verdi', 'Luca', '1992-10-02')," +
                        "('Neri', 'Paola', '1994-12-05');";

                st.executeUpdate(sql);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
```

```
        if (request.getParameter("idcliente") == null)
            // Esegue una query SQL standard
            doElencoCompleto(request, response);
        else
            // Esegue una query SQL parametrica
            doRicercaCliente(request, response);
    }

    private void doElencoCompleto(HttpServletRequest request, HttpServletResponse response) throws IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");

        // Acquisizione delle credenziali di accesso al database mediante la classe DbUtility
        DbUtility dbu = DbUtility.getInstance(getServletContext());
        String urlDb = dbu.getUrl();
        String userDb = dbu.getUser();
        String passwordDb = dbu.getPassword();

        String sql = "SELECT id, cognome, nome, data_nascita FROM anagr_clienti ORDER BY id";
        try(Connection cn = DriverManager.getConnection(urlDb, userDb, passwordDb);
            Statement ps = cn.createStatement();
            ResultSet rs = ps.executeQuery(sql))
        {
            out.println("<h1>Anagrafica clienti</h1>");
            if (!rs.next()) {
                out.println("<p>Nessun cliente presente in archivio</p>");
            }
            else {
                out.println("<table border='1' cellpadding='5'>");
                out.println("<tr><th>ID</th><th>Cognome</th><th>Nome</th><th>Data di nascita</th></tr>");

                DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd/MM/yyyy");

                do {
```

```
                    int id = rs.getInt("id");
                    String cognome = rs.getString("cognome");
                    String nome = rs.getString("nome");

                    // Conversione da SQL Date a stringa in formato g/m/a
                    // mediante le classi LocalDate e DateTimeFormatter introdotte in Java 8
                    // Trasformazione java.sql.Date => LocalDate => "15/02/2020"
                    String dataNascita = dtf.format(rs.getDate("data_nascita").toLocalDate());

                    out.println(String.format("<tr><td>%d</td><td>%s</td><td>%s</td><td>%s</td>",
                            id, cognome, nome, dataNascita));
                }
                while (rs.next());

                out.println("</table>");
            }
        } catch (SQLException e) {
            e.printStackTrace(out);
        }
        out.println("<p><a href='index.jsp'>Torna alla pagina principale</a></p>");
        out.println("</body></html>");
    }

    private void doRicercaCliente(HttpServletRequest request, HttpServletResponse response) throws IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");

        // Acquisizione delle credenziali di accesso al database mediante la classe DbUtility
        DbUtility dbu = DbUtility.getInstance(getServletContext());
        String urlDb = dbu.getUrl();
        String userDb = dbu.getUser();
        String passwordDb = dbu.getPassword();

        String sqlParam = "SELECT id, cognome, nome, data_nascita FROM anagr_clienti WHERE id = ?";
```

```
        try(Connection cn = DriverManager.getConnection(urlDb, userDb, passwordDb);
            PreparedStatement ps = cn.prepareStatement(sqlParam);
        )
        {
            int idCliente = Integer.parseInt(request.getParameter("idcliente"));
            ps.setInt(1, idCliente);
            try (ResultSet rs = ps.executeQuery()) {
                out.println("<h1>Risultato della ricerca</h1>");
                if (!rs.next()) {
                    out.println("<p>Id non presente in archivio</p>");
                } else {
                    int id = rs.getInt("id");
                    String cognome = rs.getString("cognome");
                    String nome = rs.getString("nome");

                    // Conversione java.sql.Date => LocalDate => "15/02/2020"
                    DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                    String dataNascita = dtf.format(rs.getDate("data_nascita").toLocalDate());

                    out.println("<table border='0' cellpadding='5'>");
                    out.println(String.format("<tr><th style='text-align: right'>ID:</th><td>%d</td></tr>", id));
                    out.println(String.format("<tr><th style='text-align: right'>Cognome:</th><td>%s</td></tr>", cognome));
                    out.println(String.format("<tr><th style='text-align: right'>Nome:</th><td>%s</td></tr>", nome));
                    out.println(String.format("<tr><th style='text-align: right'>Data nascita:</th><td>%s</td></tr>", dataNascita));
                    out.println("</table>");
                }
            }
        } catch (SQLException | NumberFormatException e) {
            e.printStackTrace(out);
        }
        out.println("<p><a href='index.jsp'>Torna alla pagina principale</a></p>");
        out.println("</body></html>");
    }

    @Override
```

```
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String cognome = request.getParameter("cognome");
        String nome = request.getParameter("nome");
        String nascita = request.getParameter("nascita");

        if (cognome == null || nome == null || nascita == null) {
            response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
            return;
        }

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>Inserimento nuovo cliente</h1>");

        // Acquisizione delle credenziali di accesso al database mediante la classe DbUtility
        DbUtility dbu = DbUtility.getInstance(getServletContext());
        String urlDb = dbu.getUrl();
        String userDb = dbu.getUser();
        String passwordDb = dbu.getPassword();

        String sql = "INSERT INTO anagr_clienti(cognome, nome, data_nascita) VALUES (?, ?, ?)";
        try(Connection cn = DriverManager.getConnection(urlDb, userDb, passwordDb);
            PreparedStatement ps = cn.prepareStatement(sql))
        {
            ps.setString(1, cognome);
            ps.setString(2, nome);

            // Conversione "15/02/2020" => LocalDate => java.sql.Date
            DateTimeFormatter dtf = DateTimeFormatter.ofPattern("d/M/yyyy");
            Date dataNascita = Date.valueOf(LocalDate.parse(nascita, dtf));
            ps.setDate(3, dataNascita);

            if (ps.executeUpdate() == 0) {
                throw new SQLException("Errore di inserimento dati");
```

```
            }
            out.println("<p>Dati del nuovo cliente inseriti correttamente.</p>");

        } catch (SQLException | DateTimeParseException e) {
            e.printStackTrace(out);
        }

        out.println("<p><a href='index.jsp'>Torna alla pagina principale</a></p>");
        out.println("</body></html>");
    }
}
```

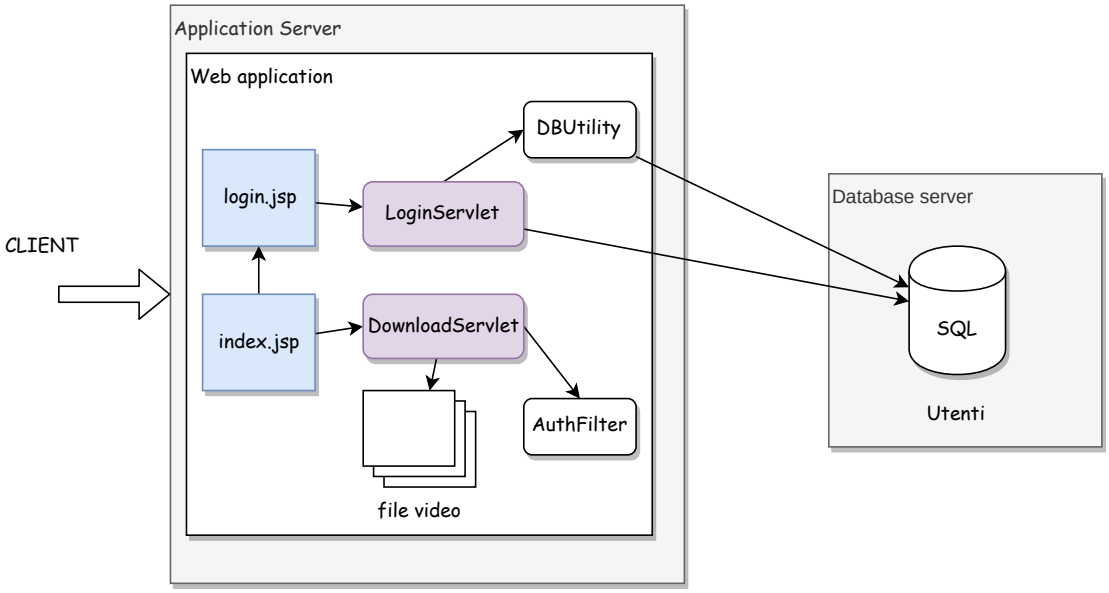
Esempio n. 3.2 (Autenticazione)

Realizzare un’applicazione web che consenta la riproduzione di contenuti video. Il servizio è limitato ai soli utenti registrati ai quali è richiesto l’inserimento delle proprie credenziali prima di accedere ai contenuti. Si supponga che la registrazione degli utenti sia effettuata con un software esterno e che le credenziali prodotte siano memorizzate in un database.

Struttura del server

L’applicazione web (*Application Context*: **/mostra-video**) utilizza JDBC per accedere al database SQL **dbvideo** contenente la tabella **video_utenti** in cui sono memorizzati i dati degli utenti registrati. Le stringhe di connessione al database e le relative credenziali di accesso per gli ambienti di sviluppo e di produzione sulla piattaforma Labs3 sono gestite mediante la classe *DbUtility*.

Sono inoltre disponibili alcuni video, in formato *.webm*, memorizzati in una cartella privata.



L’applicazione utilizza di due servlet prive di interfaccia :

- *LoginServlet* (*mapping*: **login**): esegue la procedura di accesso dall’area riservata verificando la correttezza delle credenziali. Al termine dell’autenticazione, crea una nuova sessione HTTP in cui inserisce l’attributo **nomecompleto** contenente il nome e il cognome dell’utente.
- *DownloadServlet* (*mapping*: **download**): invia il video in un formato immediatamente riproducibile da parte del client. La richiesta, di tipo GET, include il paramento **video** contenente il nome del file da riprodurre. L’accesso a questa servlet è limitato agli utenti registrati.
  Un esempio di richiesta di download è il seguente:

  ```
  http://localhost:3306/mostra-video/download?video=file1.webm
  ```

L’interfaccia grafica dell’applicazione è realizzata mediante due file JSP:

- **index.jsp**:costituisce il punto d’ingresso all’applicazione. Mostra agli utenti generici un invito ad accedere all’area riservata oppure, in presenza dell’attributo di sessione *nomecompleto*, visualizza

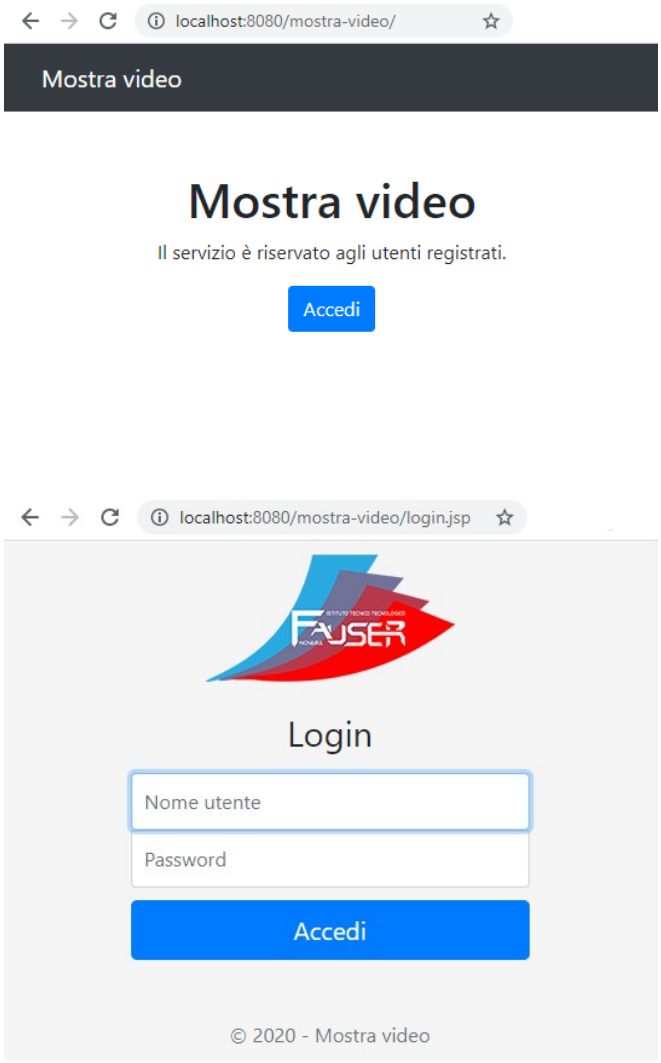
l’elenco dei video selezionabili.

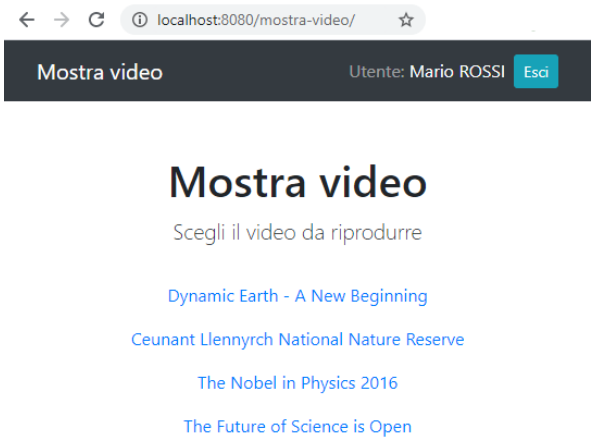
- **login.jsp**: presenta un form HTML per l’inserimento delle credenziali e invia i dati raccolti alla servlet *LoginServlet.*

Per evitare la riproduzione non autorizzata dei video, è necessario controllare l’accesso al servizio erogato da *DownloadServlet: n*el caso di richiesta proveniente da un cliente non registrato, l’applicazione deve reindirizzare l’utente alla pagina di login prima di inviare il contenuto.

L’attività di controllo e reindirizzamento è svolta da un componente software denominato *filter*. L’applicazione definisce e utilizza la classe *AuthFilter*.

Anteprime





File SQL (MariaDB)

```sql
CREATE DATABASE IF NOT EXISTS dbvideo
    CHARACTER SET = 'utf8'
    COLLATE = 'utf8_unicode_ci';

USE dbvideo;

DROP TABLE IF EXISTS video_utenti;
CREATE TABLE video_utenti
(
    username      VARCHAR(20)   NOT NULL PRIMARY KEY,
    salt          VARCHAR(32)   NOT NULL,
    password_hash VARCHAR(64)   NOT NULL,
    cognome       NVARCHAR(50)  NOT NULL,
    nome          NVARCHAR(50)  NOT NULL
);

DROP PROCEDURE IF EXISTS video_nuovoutente;
DELIMITER //
CREATE PROCEDURE video_nuovoutente(
    param_username VARCHAR(20),
    param_password VARCHAR(20),
    param_cognome NVARCHAR(50),
    param_nome NVARCHAR(50)
)
DETERMINISTIC
BEGIN
    SET @s = MD5(RAND());
    SET @h = SHA2(CONCAT(@s, param_password), 256);
    INSERT INTO video_utenti(username, salt, password_hash, cognome, nome)
    VALUES (param_username, @s, @h, param_cognome, param_nome);
END //
DELIMITER ;

CALL video_nuovoutente('rossi', 'pwrossi', 'Rossi', 'Mario');
CALL video_nuovoutente('verdi', 'pwverdi', 'Verdi', 'Carla');
CALL video_nuovoutente('neri', 'pwneri', 'Neri', 'Giacomo');
```

Pagina index.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!doctype html>
<html lang='it'>
  <head>
    <meta charset='utf-8'>
    <meta name='viewport' content='width=device-width, initial-scale=1, shrink-to-fit=no'>
    <link href='https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css' rel='stylesheet'
    integrity='sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh' crossorigin='anonymous'>
    <title>Mostra video</title>
    <style type='text/css'>
      body {padding-top: 0rem;}
      ul#listavideo {margin-top: 2em}
      ul#listavideo li {margin-bottom: 1em;}
    </style>
  </head>
  <body>
    <%
      HttpSession sessione = request.getSession(false);
      String nomeCompleto = (String) sessione.getAttribute("nomecompleto");
    %>

    <nav class="navbar navbar-dark bg-dark">
      <div class="container-fluid">
        <div class="navbar-header">
          <a class="navbar-brand" href="#">Mostra video</a>
        </div>
        <%
          if (nomeCompleto != null) {
        %>

          <form class="form-inline" action="login" method="post">
            <input type="hidden" name="esci" />
            <span class="navbar-text ">Utente: <span style="color: white"><%= nomeCompleto %></span></span>
            <input class="ml-2 btn btn-sm btn-info" type="submit" value="Esci"/>
          </form>
        <% } %>
```

```
      </div>
    </nav>

    <main class="container">
      <div class="row">
        <div class="col-lg-12 text-center">
          <h1 class="mt-5">Mostra video</h1>

          <% if (nomeCompleto != null) { %>

            <p class="lead">Scegli il video da riprodurre</p>
            <ul class="list-unstyled" id="listavideo">
                <li><a href="download?video=earth.webm">Dynamic Earth - A New Beginning</a></li>
                <li><a href="download?video=nature-reserve.webm">Ceunant Llennyrch National Nature Reserve</a></li>
                <li><a href="download?video=nobel.webm">The Nobel in Physics 2016</a></li>
                <li><a href="download?video=science.webm">The Future of Science is Open</a></li>
            </ul>

          <% } else { %>

            <p>Il servizio è riservato agli utenti registrati.</p>
            <div><a class="btn btn-primary" role="button" href="login.jsp">Accedi</a></div>

          <% } %>

        </div>
      </div>
    </main>
  </body>
</html>
```

Pagina login.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!doctype html>
<html lang='it'>
```

```
    <head>
        <meta charset='utf-8'>
        <meta name='viewport' content='width=device-width, initial-scale=1, shrink-to-fit=no'>
        <link href='https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css' rel='stylesheet'
              integrity='sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh' crossorigin='anonymous'>
        <title>Mostra video</title>
        <link href="login.css" rel="stylesheet">
    </head>

    <body class="text-center">
        <form class="form-signin" action="login" method="post">
            <input type="hidden" name="redirect" value="<%= request.getParameter("redirect") != null ? request.getParameter("redirect") : "" %>" />
            <img class="mb-4" src="img/logo.png" alt="" width="200" >

            <% if (request.getParameter("errore") != null) { %>

                <div class="alert alert-danger" role="alert">
                    Nome utente e/o password non validi.
                </div>

            <% } %>

            <h1 class="h3 mb-3 font-weight-normal">Login</h1>
            <label for="username" class="sr-only">Nome utente</label>
            <input type="text" id="username" name="username" class="form-control" placeholder="Nome utente" required autofocus />
            <label for="password" class="sr-only">Password</label>
            <input type="password" id="password" name="password" class="form-control" placeholder="Password" />
            <button class="btn btn-lg btn-primary btn-block" type="submit">Accedi</button>
            <p class="mt-5 mb-3 text-muted">&copy; 2022 - Mostra video</p>
        </form>
    </body>
</html>
```

Classe LoginServlet

```
import edu.fauser.DbUtility;

import javax.servlet.*;
```

```java
import javax.servlet.http.*;
import javax.servlet.annotation.*;
import java.io.IOException;
import java.net.URLEncoder;
import java.sql.*;

@WebServlet(name = "LoginServlet", value = "/login")
public class LoginServlet extends HttpServlet {
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        // Impostazione dei parametri di configurazione per l'accesso
        // al database Mariadb "dbvideo" con pool di 2 connessioni
        // La classe edu.fauser.DbUtility è utilizzabile anche per il
        // collegamento ai database di Labs3.

        DbUtility dbu = DbUtility.getInstance(getServletContext());
        dbu.setDevCredentials("jdbc:mariadb://localhost:3306/dbvideo?maxPoolSize=2&pool", "root", "");
        dbu.setProdCredentials("jdbc:mariadb://localhost:3306/db123?maxPoolSize=2&pool", "db123", "***");
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        boolean esci = request.getParameter("esci") != null;
        if (esci)
            eseguiLogout(request, response);
        else
            eseguiLogin(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
    }

    private void eseguiLogin(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        HttpSession sessione = request.getSession(false);
        if (sessione != null && sessione.getAttribute("nomecompleto") != null) {
            response.sendRedirect(request.getContextPath());
            return;
        }
```

```
String user = request.getParameter("username");
String pwd = request.getParameter("password");
String redirect = request.getParameter("redirect");
if (redirect == null || redirect.isEmpty())
    redirect = request.getContextPath();

DbUtility dbu = DbUtility.getInstance(getServletContext());
String urlDb = dbu.getUrl();
String userDb = dbu.getUser();
String passwordDb = dbu.getPassword();

try (Connection cn = DriverManager.getConnection(urlDb, userDb, passwordDb)) {
    String strSql = "SELECT cognome, nome FROM video_utenti " +
            "WHERE username = ? AND SHA2(CONCAT(salt, ?), 256) = password_hash";
    try (PreparedStatement ps = cn.prepareStatement(strSql)) {
        ps.setString(1, user);
        ps.setString(2, pwd);
        ResultSet rs = ps.executeQuery();
        if (rs.next() == true) {
            String cognome = rs.getString("cognome").toUpperCase();
            String nome = rs.getString("nome");
            String nomeCompleto = String.format("%s %s", nome, cognome);

            sessione = request.getSession(true);
            sessione.setAttribute("nomecompleto", nomeCompleto);
            sessione.setMaxInactiveInterval(30*60);      // 30 minuti
            response.sendRedirect(redirect);
        } else {
            String url = String.format("%s/login.jsp?errore&redirect=%s",
                    request.getContextPath(),
                    URLEncoder.encode(redirect, "utf-8")
            );
            response.sendRedirect(url);
        }
    }
} catch (SQLException e) {
    response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
    e.printStackTrace();
}
```

```
    }

    private void eseguiLogout(HttpServletRequest request, HttpServletResponse response) throws IOException {
        HttpSession session = request.getSession(false);
        String nc = (String) session.getAttribute("nomecompleto");
        if (nc != null) {
            session.invalidate();
        }
        response.sendRedirect(request.getContextPath());
    }
}
```

Classe DownloadServlet

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.nio.file.Paths;

@WebServlet(name = "DownloadServlet", value = "/download")
public class DownloadServlet extends HttpServlet {
    protected void doPost(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) throws
javax.servlet.ServletException, IOException {

    }

    protected void doGet(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) throws
javax.servlet.ServletException, IOException {
        String video = request.getParameter("video");
        if (video != null) {
            video = Paths.get(video).getFileName().toString();      // Rimuove eventuali directory
            inviaVideo(request, response, video);
```

```
        }
    }

    private void inviaVideo(HttpServletRequest request, HttpServletResponse response, String nomeVideo) {
        ServletContext context = request.getServletContext();
        String videoDir = context.getRealPath("/WEB-INF/video");
        File file = new File(String.format("%s/%s", videoDir, nomeVideo));

        try {
            FileInputStream fis = new FileInputStream(file);
            ServletContext ctx = getServletContext();
            String mimeType = ctx.getMimeType(file.getAbsolutePath());

            response.setContentType(mimeType != null ? mimeType:"application/octet-stream");
            response.setContentLength((int) file.length());

            ServletOutputStream os = null;
            os = response.getOutputStream();
            byte[] bufferData = new byte[4096];
            int read;
            while((read = fis.read(bufferData)) != -1){
                os.write(bufferData, 0, read);
            }
            os.flush();
            os.close();
            fis.close();
        } catch (FileNotFoundException e) {
            response.setStatus(HttpServletResponse.SC_CONFLICT);
        } catch (IOException e) {
            response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
        }
    }
}
```

Classe AuthFilter

```
import javax.servlet.*;
import javax.servlet.annotation.*;
```

```java
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.net.URLEncoder;

@WebFilter(filterName = "AuthFilter", value = "/download")
public class AuthFilter implements Filter {
    public void destroy() {
    }

    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws ServletException, IOException {
        HttpServletRequest request = (HttpServletRequest) req;
        HttpServletResponse response = (HttpServletResponse) resp;
        HttpSession sessione = request.getSession(false);
        boolean autenticato = sessione != null && sessione.getAttribute("nomecompleto") != null;

        if (autenticato == false) {
            String dest = request.getRequestURI();
            if (request.getQueryString() != null)
                dest += "?" + request.getQueryString();

            String r = String.format("%s/login.jsp?redirect=%s",
                    request.getContextPath(),
                    URLEncoder.encode(dest, "utf-8")
            );
            response.sendRedirect(r);
        } else {
            chain.doFilter(req, resp);
        }
    }

    public void init(FilterConfig config) throws ServletException {
    }
}
```

Classe DbUtility 1.5.0 (classe esterna, disponibile sul sito del docente)

```
public class DbUtility {
    public static final String SERVER_KEY = "edu.fauser.nomeserver";
    public static final String SERVER_NAME = "labs3";
    private String devUrl = "jdbc:mariadb://localhost:3306/test";
    private String devUser = "root";
    private String devPassword = "";
    private String prodUrl = "";
    private String prodUser = "";
    private String prodPassword = "";
    private static Boolean production = null;

    public static synchronized DbUtility getInstance(ServletContext context){
        final String attrib = "dbUtility";
        DbUtility instance = (DbUtility) context.getAttribute(attrib);
        if (instance == null) {
            instance = new DbUtility();
            context.setAttribute(attrib, instance);
        }
        return instance;
    }

    public static synchronized boolean isProduction() {
        if (production == null) {
            String server = System.getProperty(SERVER_KEY, "");
            production = server.equals(SERVER_NAME);
        }
        return production;
    }

    public synchronized void setDevCredentials(String url, String user, String password) {
        devUrl = url;
        devUser = user;
        devPassword = password;
    }
    public synchronized void setProdCredentials(String url, String user, String password) {
        prodUrl = url;
```

```
        prodUser = user;
        prodPassword = password;
    }

    public synchronized void loadDevCredentials(InputStream propertiesStream) throws IOException {
        Properties p = new Properties();
        p.load(propertiesStream);
        devUrl = p.getProperty("url");
        devUser = p.getProperty("user");
        devPassword = p.getProperty("password");
    }

    public synchronized void loadProdCredentials(InputStream propertiesStream) throws IOException {
        Properties p = new Properties();
        p.load(propertiesStream);
        prodUrl = p.getProperty("url");
        prodUser = p.getProperty("user");
        prodPassword = p.getProperty("password");
    }

    public String getUrl() {
        return isProduction() ? prodUrl : devUrl;
    }
    public String getUser() {
        return isProduction() ? prodUser : devUser;
    }
    public String getPassword() {
        return isProduction() ? prodPassword : devPassword;
    }
}
```

4. Applicazioni web con interfaccia grafica avanzata

Esempio n. 4.1 (Domande & Risposte)

Realizzare un’applicazione web che consenta di gestire un archivio di domande poste a una comunità di utenti e le relative risposte.

Struttura del server

L’applicazione web (*Application Context*: **/domande-risposte**) dispone di un’interfaccia grafica costruita a partire da un template gratuito.
Il codice del template, disponibile all’indirizzo: **https://colorlib.com/wp/template/unapp/**, è riorganizzato all’interno di pagine JSP.

L’applicazione fornisce le seguenti pagine JSP:

- *index.jsp*: presenta un elenco di domande estratte da una tabella SQL.
- *nuovadomanda.jsp*: invia il testo di una domanda alla servlet specializzata nella creazione di nuove domande.
- *nuovarisp.jsp*: consente di specificare la risposta a una domanda e alcuni dati sull’utente (nickname, immagine del profilo). La form di inserimento supporta l’attributo *enctype = "multipart/form-data"* per l’upload dell’immagine.

L’applicazione esegue le varie attività attraverso le seguenti servlet:

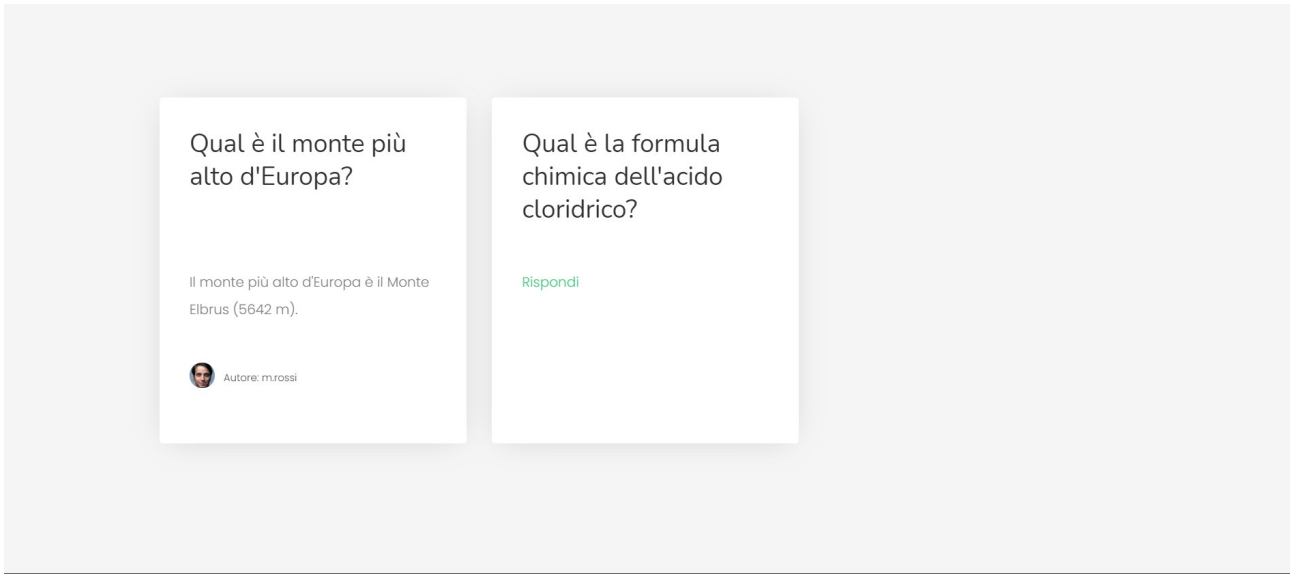
- *DomandaServlet* (*mapping*: **domanda**): inserisce una nuova domanda (priva di risposta) nella tabella SQL a partire dal testo ricevuto con il metodo POST.
- *RispostaServlet* (*mapping*: **risposta**): riceve i dati della risposta con il metodo POST e aggiorna la relativa domanda nella tabella SQL; il contenuto dell’immagine è memorizzato in un campo di tipo “Blob” (Binary Large Object).
- *ImmagineServlet* (*mapping*: **immagine**): invia al client l’immagine richiesta prelevando il contenuto dalla tabella SQL.

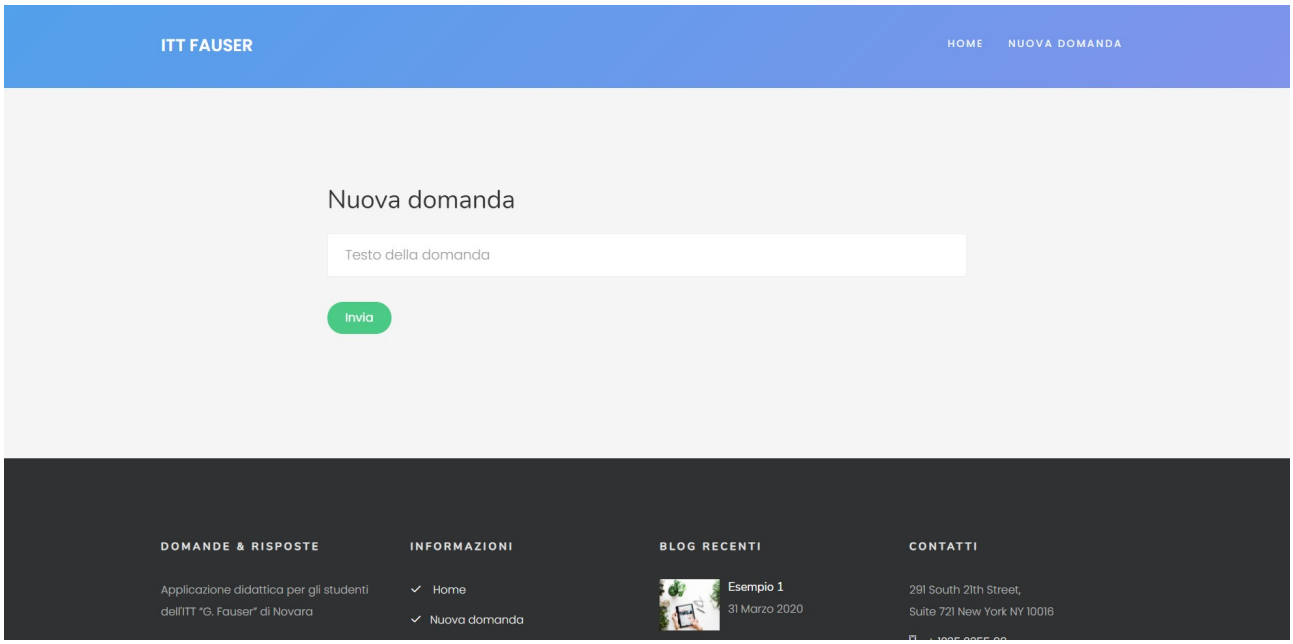
Credenziali di accesso al database

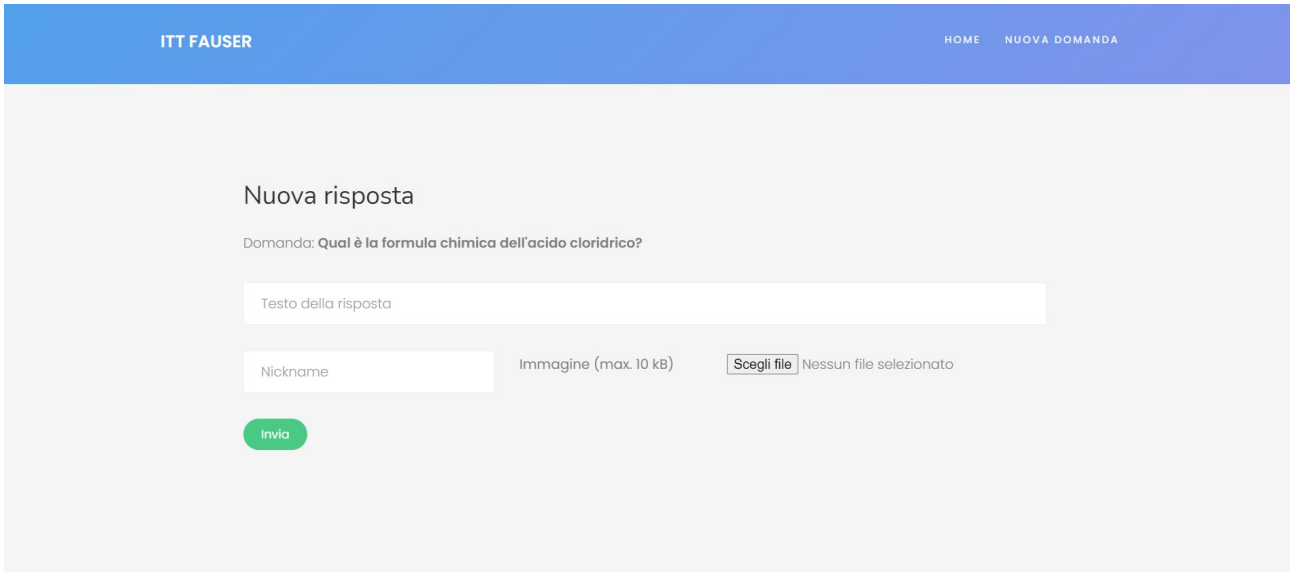
Tutte le servlet accedono al database con credenziali gestite da un’istanza comune della classe *DbUtility*. L’inserimento delle credenziali è effettuato durante l’inizializzazione dell’applicazione. Per questo scopo si ricorre a un *Web Listener* e si inserisce nel suo metodo *contextInitialized* la configurazione dei parametri di accesso ai database di sviluppo e di produzione su Labs3.

Anteprime


localhost:8080/domande-risposte/
ITT FAUSER
HOME
NUOVA DOMANDA
DOMANDE & RISPOSTE
Applicazione didattica per gli studenti dell'ITT “G. Fauser” di Novara
Elenco domande


Qual è il monte più alto d'Europa?
Il monte più alto d'Europa è il Monte Elbrus (5642 m).
Autore: m.rossi
Qual è la formula chimica dell'acido cloridrico?
Rispondi


ITT FAUSER
HOME
NUOVA DOMANDA
Nuova domanda
Testo della domanda
Invia
DOMANDE & RISPOSTE
Applicazione didattica per gli studenti dell'ITT "G. Fauser" di Novara
INFORMAZIONI
Home
Nuova domanda
BLOG RECENTI
Esempio 1
31 Marzo 2020
CONTATTI
291 South 21th Street,
Suite 721 New York NY 10016


ITT FAUSER
HOME
NUOVA DOMANDA
Nuova risposta
Domanda: Qual è la formula chimica dell'acido cloridrico?
Testo della risposta
Nickname
Immagine (max. 10 kB)
Scegli file
Nessun file selezionato
Invia

File SQL (MariaDB)

```
CREATE DATABASE IF NOT EXISTS dbdomande
    CHARACTER SET = 'utf8'
    COLLATE = 'utf8_unicode_ci';

USE dbdomande;

DROP TABLE IF EXISTS dr_domande;
CREATE TABLE dr_domande
(
    codice    INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    domanda   NVARCHAR(100)  NOT NULL,
    risposta  NVARCHAR(100),
    nickname  NVARCHAR(25),
    img       LONGBLOB,
    tipo      NVARCHAR(30)
);

INSERT INTO dr_domande(domanda) VALUES ('Qual è il monte più alto d''Europa?');
INSERT INTO dr_domande(domanda) VALUES ('Qual è la formula chimica dell''acido cloridrico?');
```

Pagina index.jsp

```
<%@ page import="java.sql.*" %>
<%@ page import="edu.fauser.DbUtility" %>
<%@ page import="edu.fauser.netlab.AppUtility" %>
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%

    DbUtility dbu = (DbUtility) application.getAttribute("dbutility");

    try(Connection cn = DriverManager.getConnection(dbu.getUrl(), dbu.getUser(), dbu.getPassword());
        Statement s = cn.createStatement();
```

```
        ResultSet rs = s.executeQuery(
            "SELECT codice, domanda, risposta, nickname FROM dr_domande");) {
%>
<!DOCTYPE HTML>
<html>
<%@include file="inc/head.jsp"%>
<body>
<%@include file="inc/header-index.jsp"%>
<div class="colorlib-blog" id="elenco">
    <div class="container">
        <div class="row">
            <% int conta = 0;
                while (rs.next()) {
                    conta++;
                    if (conta > 1 && (conta % 3) == 1) {
            %>
                        </div><div class="row">
            <%      } %>
                    <div class="col-md-4 animate-box" >
                        <article>
                            <h2 style="height: 5em;"><%= AppUtility.escapeHTML(rs.getString("domanda")) %></h2>
                            <% if(rs.getString("risposta") == null) { %>
                                <p style="height: 5em;">
                                    <a href="nuovarisp.jsp?codice=<%= rs.getInt("codice") %>">Rispondi</a>
                                </p>
                                <p style="height: 2.5em;" class="author-wrap"> </p>
                            <% } else { %>
                                <p style="height: 5em;"><%= AppUtility.escapeHTML(rs.getString("risposta")) %></p>
                                <p style="height: 2.5em;" class="author-wrap">
                                    <span class="author-img" style="background-image: url(immagine?codice=<%= rs.getInt("codice")%>);"></span>
                                    <span class="author">Autore: <%= AppUtility.escapeHTML(rs.getString("nickname")) %></span></p>
                            <% } %>
                        </article>
                    </div>
            <%      } // fine while %>
        </div>
    </div>
</div>
```

```
<%@include file="inc/footer.jsp"%>
</body>
</html>
<%
    } catch (Exception e) {
        e.printStackTrace(response.getWriter());
    }
%>
```

Pagina nuovadomanda.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<!DOCTYPE HTML>
<html>
<%@include file="inc/head.jsp"%>
<body>
<%@include file="inc/header.jsp"%>
<div id="colorlib-contact">
    <div class="container">
        <div class="row">
            <div class="col-md-8 col-md-offset-2 animate-box">
                <h2>Nuova domanda</h2>
                <form action="domanda" method="post">
                    <div class="row form-group">
                        <div class="col-md-12">
                            <input type="text" maxlength="100" id="domanda" name="domanda" class="form-control"
                                   placeholder="Testo della domanda" required>
                        </div>
                    </div>

                    <div class="form-group">
                        <input type="submit" value="Invia" class="btn btn-primary">
                    </div>
                </form>
            </div>
        </div>
```

```
    </div>
</div>

<%@include file="inc/footer.jsp"%>
</body>
```

Pagina nuovarisp.jsp

```
<%@ page import="java.sql.Connection" %>
<%@ page import="java.sql.PreparedStatement" %>
<%@ page import="java.sql.ResultSet" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<%
    String testo = null;
    int codice = -1;
    try {
        codice = Integer.parseInt(request.getParameter("codice"));
    }
    catch (NumberFormatException e) {
        e.printStackTrace(response.getWriter());
    }

    DbUtility dbu = (DbUtility) application.getAttribute("dbutility");

    try(Connection cn = DriverManager.getConnection(dbu.getUrl(), dbu.getUser(), dbu.getPassword())) {
        String strSql = "SELECT domanda FROM dr_domande WHERE codice = ?";
        try (PreparedStatement ps = cn.prepareStatement(strSql)) {
            ps.setInt(1, codice);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next() == false) {
                    response.setStatus(HttpServletResponse.SC_NOT_FOUND);
                    return;
                }
                testo = rs.getString("domanda");
            }
```

```
        }
%>
<%@ page import="edu.fauser.DbUtility" %>
<%@ page import="java.sql.DriverManager" %>
<%@ page import="edu.fauser.netlab.AppUtility" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE HTML>
<html>
<%@include file="inc/head.jsp"%>
<body>
<%@include file="inc/header.jsp"%>
<div id="colorlib-contact">
    <div class="container">
        <div class="row">
            <div class="col-md-10 col-md-offset-1 animate-box">
                <h2>Nuova risposta</h2>
                <form action="risposta" method="post" enctype = "multipart/form-data">
                    <input type="hidden" name="codice" value="<%= codice %>" />
                    <div class="row form-group">
                        <div class="col-md-12">
                            <div>Domanda: <strong><%= AppUtility.escapeHTML(testo) %></strong></div>
                        </div>
                    </div>
                    <div class="row form-group">
                        <div class="col-md-12">
                            <input type="text" maxlength="100" id="risposta" name="risposta" class="form-control"
                                   placeholder="Testo della risposta" required>
                        </div>
                    </div>
                    <div class="row form-group">
                        <div class="col-md-4">
                            <input type="text" maxlength="100" id="nickname" name="nickname" class="form-control"
                                   placeholder="Nickname" required>
                        </div>
                        <label for="img" class="col-md-3 col-form-label">Immagine (max. 10 kB)</label>
                        <div class="col-md-5">
                            <input type = "file" name = "img" id="img" size = "50" required />
                        </div>
                    </div>
```

```
                    <div class="form-group">
                        <input type="submit" value="Invia" class="btn btn-primary">
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
<%@include file="inc/footer.jsp"%>
<%
    } catch (Exception e) {
        e.printStackTrace(response.getWriter());
    }
%>
```

Classe DomandaServlet

```
@WebServlet(name = "DomandaServlet", value = "/domanda")
public class DomandaServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        String domanda = request.getParameter("domanda");
        System.out.println(domanda);
        if (domanda == null) {
            response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
            return;
        }
        try {
            ServletContext ctx = request.getServletContext();
            DbUtility dbu = (DbUtility) ctx.getAttribute("dbutility");
            nuovaDomanda(domanda, dbu);
            response.sendRedirect("index.jsp");
        } catch (SQLException e) {
            response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
        }
    }
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // Metodo GET non gestito da questa servlet
        response.setStatus(HttpServletResponse.SC_METHOD_NOT_ALLOWED);
    }

    private void nuovaDomanda(String domanda, DbUtility dbu) throws SQLException {
        try (Connection cn = DriverManager.getConnection(dbu.getUrl(), dbu.getUser(), dbu.getPassword())) {
            String strSql = "INSERT INTO dr_domande(domanda) VALUES (?)";
            try (PreparedStatement ps = cn.prepareStatement(strSql)) {
                ps.setString(1, domanda);
                if (ps.executeUpdate() == 0) {
                    throw new SQLException("Errore di inserimento dati");
                }
            }
        }
    }
}
```

Classe RispostaServlet

```
import edu.fauser.DbUtility;
import edu.fauser.netlab.AppUtility;

// Upload limitato a immagini di dimensioni non superiori a 10 kB.

@WebServlet(name = "RispostaServlet", value = "/risposta")
@MultipartConfig(fileSizeThreshold = 10240, maxFileSize = 10240, maxRequestSize = 12240)
public class RispostaServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // Metodo GET non gestito da questa servlet
        response.sendError(HttpServletResponse.SC_BAD_REQUEST);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        String strCod = request.getParameter("codice");
```

```
String risposta = request.getParameter("risposta");
String nickname = request.getParameter("nickname");
Part img = null;
try {
    // Acquisisce la parte della richiesta contenente l'immagine
    img = request.getPart("img");
} catch (IllegalStateException e) {
    // La dimensione dell'immagine supera il limite imposto dal server
    response.sendError(HttpServletResponse.SC_REQUEST_ENTITY_TOO_LARGE);
    return;
}

if (strCod == null || risposta == null || nickname == null || img == null) {
    response.sendError(HttpServletResponse.SC_BAD_REQUEST);
    return;
}
ServletContext ctx = request.getServletContext();
Integer codice = Integer.parseInt(strCod);
String nomeFile = estraiFileName(img);
String tipo =  ctx.getMimeType(nomeFile);       // esempi:  image/jpeg, image/png, ecc.
DbUtility dbu = (DbUtility) ctx.getAttribute("dbutility");

try (Connection cn = DriverManager.getConnection(dbu.getUrl(), dbu.getUser(), dbu.getPassword()))
{
    String strSql = "UPDATE dr_domande SET risposta = ?, nickname = ?, img = ?, tipo = ? WHERE codice = ?";
    try (PreparedStatement ps = cn.prepareStatement(strSql)) {
        ps.setString(1, risposta);
        ps.setString(2, nickname);
        // Copia lo stream dell'immagine nel campo BLOB "img"
        ps.setBinaryStream(3, img.getInputStream());
        ps.setString(4, tipo);
        ps.setInt(5, codice);
        if (ps.executeUpdate() == 0) {
            // Aggiornamento fallito, si mostra l'errore SQL nella console del server
            AppUtility.mostraErroreSql(ps.getWarnings());
            response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
            return;
        }
        response.sendRedirect("index.jsp");
```

```
            }
        } catch (Exception e) {
            e.printStackTrace(response.getWriter());
        }
    }
    private static String estraiFileName(Part part) {
        // Ricava il nome del file contenuto nella richiesta HTTP
        for (String cd : part.getHeader("content-disposition").split(";")) {
            if (cd.trim().startsWith("filename")) {
                String fileName = cd.substring(cd.indexOf('=') + 1).trim().replace("\"", "");
                return fileName.substring(fileName.lastIndexOf('/') + 1).substring(fileName.lastIndexOf('\\') + 1); // MSIE fix.
            }
        }
        return null;
    }
}
```

Classe ImmagineServlet

```
import edu.fauser.DbUtility;

@WebServlet(name = "ImmagineServlet", value = "/immagine")
public class ImmagineServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Integer codice = Integer.parseInt(request.getParameter("codice"));
        if (codice == null) {
            response.sendError(HttpServletResponse.SC_BAD_REQUEST);
            return;
        }

        ServletContext ctx = request.getServletContext();
        DbUtility dbu = (DbUtility) ctx.getAttribute("dbutility");
        try (Connection cn = DriverManager.getConnection(dbu.getUrl(), dbu.getUser(), dbu.getPassword()))
        {
            String strSql = "SELECT img, tipo FROM dr_domande WHERE codice = ?";
            try (PreparedStatement ps = cn.prepareStatement(strSql)) {
                ps.setInt(1, codice);
```

```
                ResultSet rs = ps.executeQuery();
                if (rs.next() == false) {
                    response.sendError(HttpServletResponse.SC_NOT_FOUND);
                    return;
                }
                // Stream collegato alla risposta HTTP
                ServletOutputStream out = response.getOutputStream();

                // Apre uno stream collegato al campo BLOB "img"
                InputStream img = rs.getBinaryStream("img");
                String tipo = rs.getString("tipo");
                // Imposta il Content-Type in base al tipo di immagine
                response.setContentType(tipo);

                // Copia il contenuto dell'immagine nello stream di output
                byte[] buffer = new byte[4096];
                while (img.read(buffer) > 0) {
                    out.write(buffer);
                }
                out.flush();
                rs.close();
            }
        } catch (SQLException e) {
            e.printStackTrace(response.getWriter());
            response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
        }
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // Metodo POST non gestito da questa servlet
        response.sendError(HttpServletResponse.SC_METHOD_NOT_ALLOWED);
    }
}
```

Classe AppListener

```
import edu.fauser.DbUtility;
```

```java
@WebListener
public class AppListener implements ServletContextListener, HttpSessionListener, HttpSessionAttributeListener {

    public AppListener() {
    }

    @Override
    public void contextInitialized(ServletContextEvent sce) {
        // Crea una nuova istanza della classe DbUtility e la condivide con
        // le servlet e le pagine JSP sotto forma di attributi del contesto

        DbUtility dbu = new DbUtility();
        dbu.setDevCredentials("jdbc:mariadb://localhost:3306/dbdomande?maxPoolSize=2&pool", "root", "");
        dbu.setProdCredentials("jdbc:mariadb://localhost:3306/db123?maxPoolSize=2&pool", "db123", "******");

        ServletContext ctx = sce.getServletContext();
        ctx.setAttribute("dbutility", dbu);
    }
}
```

Classe AppUtility

```java
package edu.fauser.netlab;

import java.sql.SQLWarning;

public class AppUtility {
    public static String escapeHTML(String str) {
        StringBuilder sb = new StringBuilder(str.length());
        for (int i = 0; i < str.length(); i++) {
            char ch = str.charAt(i);
            if (ch == '&' || ch == '"' || ch == '<' || ch == '>' || ch > 127) {
                sb.append(String.format("&#%d;", (int) ch));
            } else {
                sb.append(ch);
            }
```

```
        }
        return sb.toString();
    }

    public static void mostraErroreSql(SQLWarning warning) {
        // Stampa l'avvertimento nella console del Container
        if (warning != null) {
            System.out.println("n---Warning---n");
            do {
                System.out.println("Message: " + warning.getMessage());
                System.out.println("SQLState: " + warning.getSQLState());
                System.out.print("Vendor error code: ");
                System.out.println(warning.getErrorCode());
                System.out.println("");
                warning = warning.getNextWarning();
            }
            while (warning != null);
        }

    }
}
```

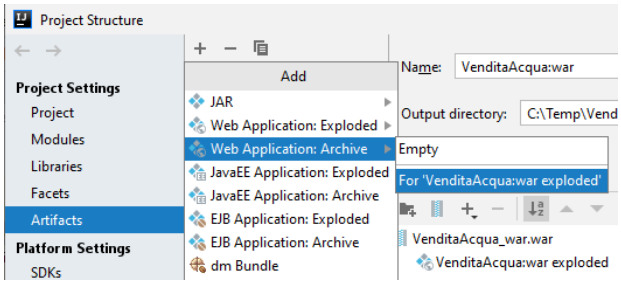
Appendice

A1 – Memorizzazione di un’applicazione web in formato WAR

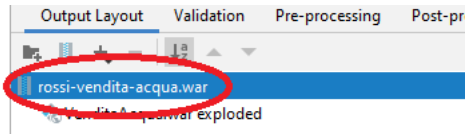
Prima di eseguire il deployment di un’applicazione web in un server remoto, occorre memorizzare il contenuto dell’applicazione (classi, file .jsp, risorse, ecc.) in formato WAR (Web Archive).

Per generare un file .war in Intellij Idea, eseguire le seguenti operazioni:

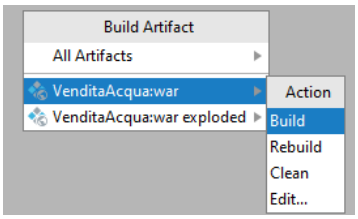
1. Selezionare il comando **File → Project Structure…**
2. All’interno della sezione *Project Settings* selezionare la voce **Artifacts,** quindi premere il bottone **+** e selezionare la voce **Web Application: Archive → For <nomeapp>:war exploded**

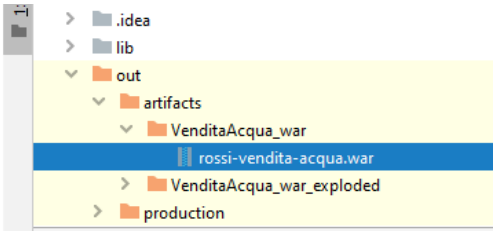


3. Allo scopo di facilitare la pubblicazione delle proprie applicazioni sulla piattaforma Netlab, si consiglia di rinominare il file .war specificando il proprio cognome e l’*application context*.



4. Per generare l’applicazione in formato WAR, eseguire il comando **Build → Build artifacts…**, quindi selezionare il nome dell’*artifact* e l’azione **Build**. IntelliJ Idea crea il file .war all’interno della sottocartella *out/artifacts*.





A2 – Deployment di applicazioni web su server Ubuntu Linux 20.04 LTS

A2.1 – Installazione del software JDK 11

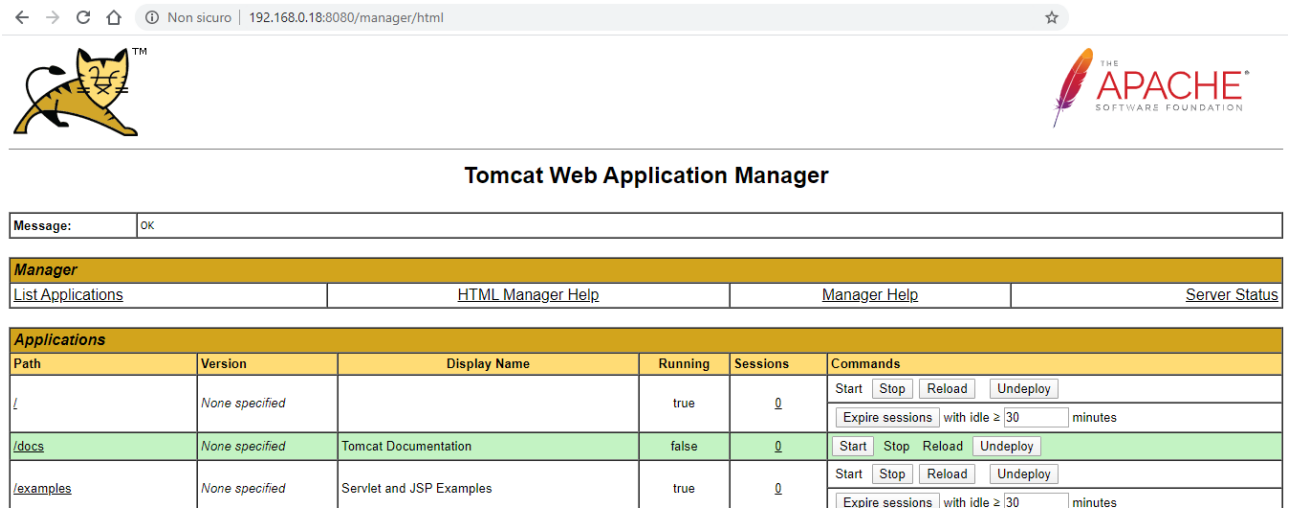
```
sudo apt update
sudp apt upgrade
sudo apt install default-jdk
```

A2.2 – Installazione del software Tomcat 9

```
sudo apt install tomcat9 tomcat9-admin
sudo nano /etc/tomcat9/tomcat-users.xml
```

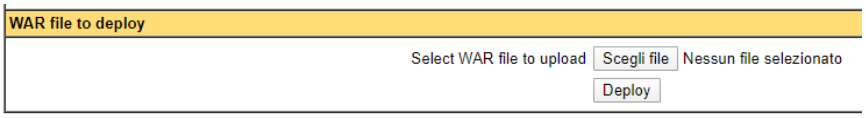
Aggiungere nella sezione *<tomcat-users>* le seguenti righe:

<role rolename="manager-gui"/>
<role rolename="admin-gui"/>
<user username="tomcat" password="123456" roles="manager-gui,admin-gui"/>
Accedere all’applicazione di gestione di Tomcat: **http://<indirizzo-ip>:8080/manager**

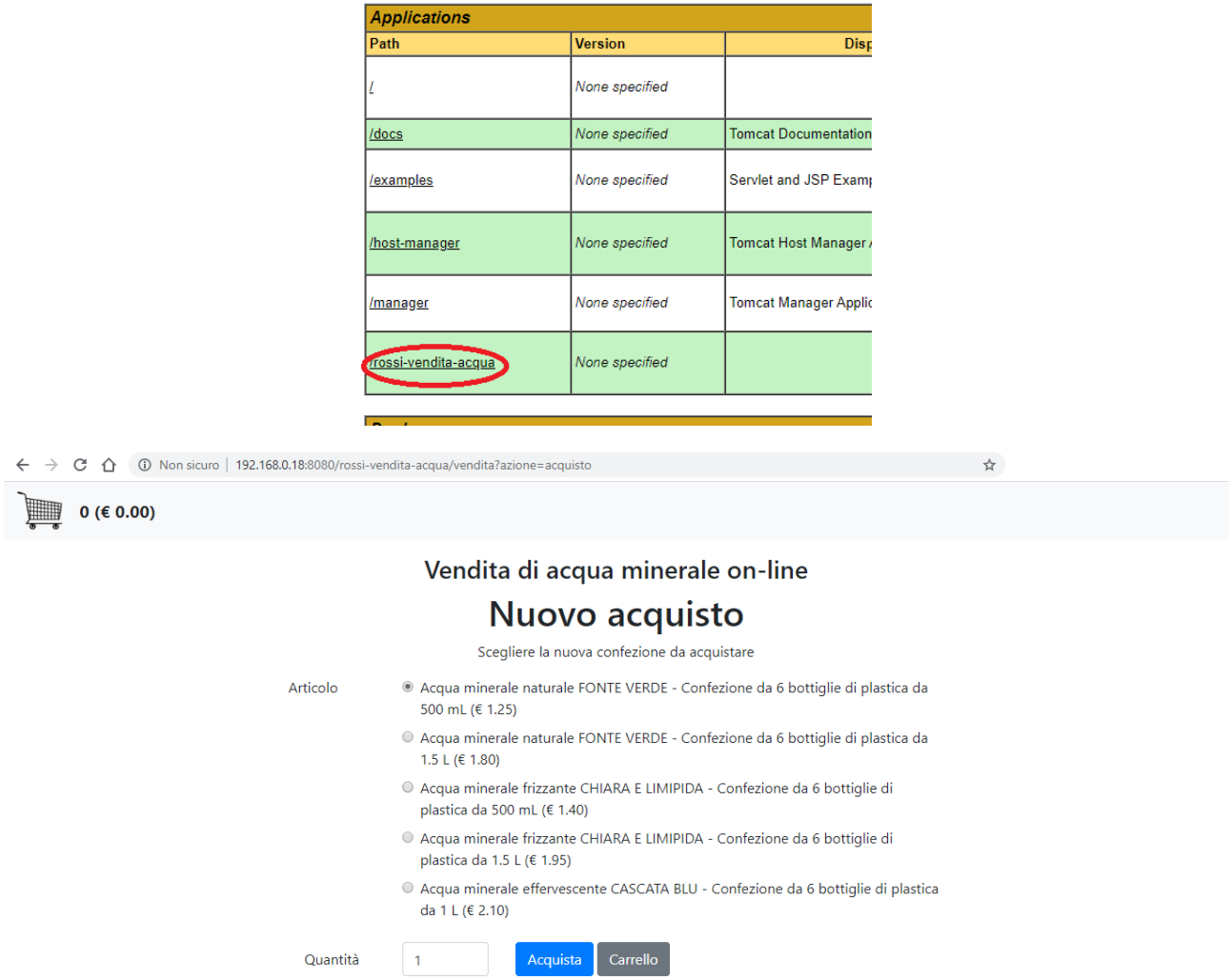


A2.3 – Deployment dell’applicazione

All’interno della sezione **WAR file to deploy** Application Manager di Tomcat, fare click sul pulsante **Scegli file,** scegliere il file .war da pubblicare e infine premere il pulsante **Deploy**.



Per avviare l’applicazione web, fare click sul relativo collegamento presente nell’elenco delle applicazioni gestite da Tomcat.



Applications

Path	Version	Disp
/	None specified	
/docs	None specified	Tomcat Documentation
/examples	None specified	Servlet and JSP Exam
/host-manager	None specified	Tomcat Host Manager
/manager	None specified	Tomcat Manager Applic
/rossi-vendita-acqua	None specified	

Non sicuro | 192.168.0.18:8080/rossi-vendita-acqua/vendita?azione=acquisto

0 (€ 0.00)

Vendita di acqua minerale on-line

Nuovo acquisto

Scegliere la nuova confezione da acquistare

Articolo

- (•) Acqua minerale naturale FONTE VERDE - Confezione da 6 bottiglie di plastica da 500 mL (€ 1.25)
- ( ) Acqua minerale naturale FONTE VERDE - Confezione da 6 bottiglie di plastica da 1.5 L (€ 1.80)
- ( ) Acqua minerale frizzante CHIARA E LIMPIDA - Confezione da 6 bottiglie di plastica da 500 mL (€ 1.40)
- ( ) Acqua minerale frizzante CHIARA E LIMPIDA - Confezione da 6 bottiglie di plastica da 1.5 L (€ 1.95)
- ( ) Acqua minerale effervescente CASCATA BLU - Confezione da 6 bottiglie di plastica da 1 L (€ 2.10)

Quantità 1 Acquista Carrello

A3 – Argon2: un moderno sistema per la protezione delle password

Argon2 rappresenta uno dei più avanzati algoritmi di hashing di password disponibili oggi, progettato specificamente per resistere alle moderne tecniche di attacco. Vincitore della *Password Hashing Competition* nel 2015, Argon2 è stato sviluppato come risposta alle crescenti minacce alla sicurezza delle password memorizzate nei sistemi informatici.

Caratteristiche e funzionamento di Argon2

Argon2 si distingue per la sua capacità di essere memory-hard, ovvero richiede una quantità significativa di memoria per essere eseguito. Questo aspetto è fondamentale per contrastare gli attacchi basati su hardware specializzato come ASIC o GPU, rendendo economicamente proibitivo tentare attacchi a forza bruta su larga scala.

Il funzionamento di Argon2 si basa su tre parametri principali che possono essere configurati per adattarsi alle esigenze specifiche di sicurezza:

- Il parametro di memoria (m_cost) determina quanta RAM viene utilizzata durante il processo di hashing. Maggiore è questo valore, più difficile diventa eseguire attacchi paralleli su hardware specializzato.
- Il parametro di tempo (t_cost) definisce il numero di iterazioni dell'algoritmo. Aumentando questo valore si incrementa il tempo necessario per generare un hash, rendendo più costosi gli attacchi a forza bruta.
- Il parametro di parallelismo (parallelism) specifica quanti thread possono essere utilizzati contemporaneamente per calcolare l'hash. Questo permette di sfruttare architetture multi-core senza compromettere la sicurezza.

Varianti di Argon2

Argon2 è disponibile in tre varianti principali, ciascuna ottimizzata per scenari di sicurezza specifici:

- **Argon2d** è progettato per offrire la massima resistenza contro attacchi condotti con hardware specializzato (come GPU e ASIC), grazie all’uso di accessi alla memoria dipendenti dai dati. Tuttavia, questa caratteristica lo rende più vulnerabile ad attacchi *side-channel*, come quelli di tipo *timing* o *cache-based*. Per questo motivo, *Argon2d* è più adatto in contesti come le criptovalute, piuttosto che per il password hashing.
- **Argon2i** utilizza accessi alla memoria indipendenti dai dati, il che lo rende resistente agli attacchi *side-channel*. È quindi particolarmente indicato per l’hashing di password e altri scenari in cui la protezione contro attacchi di tipo *timing* è fondamentale.
- **Argon2id**, la variante implementata nella libreria in uso, rappresenta un compromesso tra *Argon2d* e *Argon2i*. Inizia con accessi indipendenti dai dati (come *Argon2i*) e prosegue con accessi dipendenti dai dati (come *Argon2d*), combinando così la resistenza agli attacchi *side-channel* con una buona protezione contro attacchi basati su hardware. Per questo motivo, *Argon2id* è considerata la scelta più equilibrata e raccomandata per la protezione delle password nella maggior parte dei casi.

Protezione delle password nei database SQL con Argon2

L'utilizzo di *Argon2* per proteggere le password nei database SQL rappresenta una pratica di sicurezza fondamentale. Quando un utente crea o modifica una password, questa viene elaborata attraverso l'algoritmo *Argon2* per generare un hash che viene poi memorizzato nel database. Durante la fase di autenticazione, la password fornita dall'utente viene nuovamente elaborata con *Argon2* e il risultato viene confrontato con l'hash memorizzato.

Questo approccio garantisce che, anche in caso di violazione del database, un attaccante non possa risalire direttamente alle password originali. Inoltre, grazie ai parametri configurabili di *Argon2*, è possibile calibrare il livello di sicurezza in base alle caratteristiche dell'hardware e alle esigenze specifiche dell'applicazione.

Il ruolo della chiave segreta (pepper) e la gestione mediante HSM

Nella sicurezza delle password, oltre al *salt* (un valore casuale pubblico aggiunto a ogni password), è possibile implementare un ulteriore livello di protezione attraverso una chiave segreta, comunemente chiamata "*pepper*". A differenza del *salt*, che viene memorizzato insieme all'hash, il *pepper* è una chiave segreta condivisa che non viene memorizzata nel database.

Il *pepper* aggiunge un ulteriore strato di sicurezza: anche se un attaccante dovesse ottenere accesso al database contenente gli hash delle password e i relativi *salt*, senza conoscere il *pepper* non sarebbe in grado di condurre attacchi efficaci. Nella libreria ***argon2_udf***, questo concetto è implementato attraverso il parametro "secret".

Per gestire in modo sicuro questa chiave segreta, un approccio consigliato è l'utilizzo di un *Hardware Security Module* (HSM). Un HSM è un dispositivo fisico specializzato che fornisce funzionalità crittografiche e protegge le chiavi crittografiche da accessi non autorizzati. I vantaggi di questo approccio sono molteplici: l'HSM mantiene la chiave segreta isolata dal sistema principale, impedendo che possa essere compromessa anche in caso di violazione del server; le operazioni crittografiche vengono eseguite all'interno dell'HSM stesso, evitando che la chiave venga esposta nella memoria del sistema.

Gli HSM offrono meccanismi di controllo degli accessi, logging e *tamper-resistance* (resistenza alla manomissione) che aumentano ulteriormente il livello di sicurezza. In un'architettura che utilizza HSM, l'applicazione invierebbe la password e il *salt* all'HSM, che aggiungerebbe il pepper e restituirebbe l'hash calcolato. Questo processo garantisce che il pepper non venga mai esposto al di fuori dell'HSM, mantenendo un elevato livello di sicurezza anche in caso di compromissione dell'applicazione o del database.

Installazione della libreria *argon2_udf* su MariaDB

L'installazione della libreria ***argon2_udf*** su server MariaDB differisce leggermente tra sistemi Windows e Linux, ma in entrambi i casi non richiede la ricompilazione del codice sorgente grazie alla disponibilità dei file precompilati **`argon2_udf.dll`** e argon2**`_udf.so`**.

Installazione su Windows

Per installare la libreria su un server MariaDB in ambiente Windows, seguire questi passaggi:

1. Copiare il file **`argon2_udf.dll`** nella directory dei plugin di MariaDB, tipicamente `C:\Program Files\MariaDB\MariaDB X.Y\lib\plugin\`.

2. Avviare il client MariaDB come amministratore di database e registrare le funzioni UDF con i seguenti comandi SQL:

```
CREATE FUNCTION argon2_password RETURNS STRING SONAME 'argon2_udf.dll';
CREATE FUNCTION argon2_pverify RETURNS INTEGER SONAME 'argon2_udf.dll';
```

3. Verificare che le funzioni siano state registrate correttamente:

```
SELECT * FROM mysql.func WHERE name LIKE 'argon2%';
```

Installazione su Linux

Per installare la libreria su un server MariaDB in ambiente Linux, procedere come segue:

1. Copiare il file **`argon2_udf.so`** nella directory dei plugin di MariaDB e impostare i permessi appropriati:

```
sudo cp argon2_udf.so $(mysql_config --plugindir)
sudo chmod 644 $(mysql_config --plugindir)/argon2_udf.so
```

2. Avviare il client MariaDB e registrare le funzioni UDF:

```
CREATE FUNCTION argon2_password RETURNS STRING SONAME 'argon2_udf.so';
CREATE FUNCTION argon2_pverify RETURNS INTEGER SONAME 'argon2_udf.so';
```

3. Verificare l'installazione:

```
SELECT * FROM mysql.func WHERE name LIKE 'argon2%';
```

Esempi d'uso in SQL delle funzioni argon2_password e argon2_pverify

Una volta installata correttamente la libreria, le funzioni **`argon2_password`** e **`argon2_pverify`** possono essere utilizzate all'interno delle query SQL per generare e verificare password in modo sicuro.

Generazione di hash di password

La funzione **`argon2_password`** offre diversi livelli di personalizzazione, permettendo di adattare i parametri di sicurezza alle esigenze specifiche dell'applicazione. Per generare un hash con parametri di default:

```
SELECT argon2_password('miapassword');
```

Questo comando genera un hash utilizzando un *salt* casuale e i parametri di default (memoria = 64MB, iterazioni = 3, parallelismo = 1, lunghezza hash = 32 byte).

Per aumentare la sicurezza utilizzando una chiave segreta (pepper):

```
SELECT argon2_password('miapassword', 'chiave_segreta_molto_complessa');
```

Per specificare un salt personalizzato oltre alla chiave segreta:

```
SELECT argon2_password('miapassword', 'chiave_segreta', 'salt_personalizzato');
```

Per personalizzare completamente i parametri di sicurezza:

```
-- Utilizzo di 128MB di memoria, 5 iterazioni e 2 thread
SELECT argon2_password('miapassword', '', '', 131072, 5, 2);
```

Per specificare anche la lunghezza dell'hash di output (16, 32 o 64 byte):

```
-- Generazione di un hash di 512 bit (64 byte)
SELECT argon2_password('miapassword', '', '', 131072, 5, 2, 64);
```

Verifica della password

La funzione **`argon2_pverify`** permette di verificare se una password corrisponde a un hash precedentemente generato.

Per verificare una password senza chiave segreta:

```
SELECT argon2_pverify('miapassword', 'hash_precedentemente_generato');
```

Per verificare una password con chiave segreta:

```
SELECT argon2_pverify('miapassword', 'chiave_segreta', 'hash_precedente');
```

La funzione restituisce 1 se la verifica ha successo e 0 in caso contrario.

Esempi di integrazione in applicazioni database

Di seguito è riportato un esempio di gestione degli utenti di un'applicazione mediante le funzioni della libreria **`argon2_udf.`**

Creazione di una tabella utenti con campo password crittografato:

```
CREATE TABLE utenti (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Inserimento di un nuovo utente con password crittografata:

```
INSERT INTO utenti (username, password_hash, email)
VALUES ('rossi', argon2_password('pwrossi'), 'rossi@example.org');
```

Procedura di login:

```
SELECT id, username FROM utenti
WHERE username = 'rossi'
AND argon2_pverify('pwrossi', password_hash) = 1;
```

Aggiornamento della password di un utente:

```
UPDATE utenti
SET password_hash = argon2_password('nuova_pwrossi')
WHERE id = 123;
```

Applicazioni proposte

I. [Applicazione **Quadri**] – Scrivere un'applicazione web basata su servlet che offra un servizio di descrizione di quadri. L'applicazione presenta un elenco di quadri disponibili (ogni quadro è caratterizzato da un codice univoco) da cui l'utente sceglie quello preferito. Successivamente è inviata una richiesta HTTP per ottenere la pagina web contenente i seguenti dettagli del quadro: titolo, artista, data di realizzazione, tecnica e materiali, dimensioni, ubicazione, descrizione oggettiva, descrizione soggettiva. La pagina del quadro può contenere collegamenti a immagini presenti su siti web esterni.

II. [Applicazione **Museo**] – All'interno di un museo sono presenti due aree: un'area "rossa" in cui sono presenti alcuni quadri di pittori famosi; un'area "verde" in cui gli artisti locali espongono i propri lavori. Il museo ammette solamente visite di gruppo e, a causa del numero elevato di opere d'arte da ammirare, ogni visita a una qualunque area dura esattamente un giorno.
I costi per visitare il museo sono i seguenti: 12 Euro per ogni membro del gruppo nel caso di l'accesso all'area rossa; 5 Euro nel caso di accesso all'area verde.
Scrivere un'applicazione web basata su servlet che permetta di prenotare una visita di gruppo a una delle due aree del museo. L'applicazione, attraverso un'opportuna inrfaccia grafica, deve consentire le seguenti operazioni:

   (a) *Inserimento di una nuova prenotazione*: l'utente prenota una nuova visita specificando il proprio nome, la data della visita, il tipo di area (rossa/verde) e il numero di partecipanti. L'applicazione valuta la richiesta: se nella data indicata è già prevista un'altra visita in quella stessa area, rifiuta la prenotazione informando l'utente con un opportuno messaggio; in caso contrario registra la prenotazione e restituisce un messaggio di conferma contenente l'importo totale da pagare.

   (b) *Visualizzazione delle prenotazioni*: l'utente può richiedere, specificando l'area del museo, l'elenco delle prenotazioni registrate nel server per quell'area. Per ogni prenotazione sono riportati: data, numero di partecipanti e nome della persona che ha effettuato la prenotazione.

III. [Applicazione **Musica**] – Scrivere un'applicazione web per la vendita di brani musicali in formato MP3. I clienti accedono a un'area riservata inserendo opportune credenziali, quindi acquistano i brani da un elenco proposto. Ogni brano dell'elenco è caratterizzato da: codice, titolo del brano, nome del cantante, durata, prezzo, nome del file MP3. Prima di procedere all'acquisto, il cliente può selezionare un apposito pulsante per riprodurre una breve anteprima del brano (max. 15 secondi). Al termine di ogni acquisto, l'applicazione riassume i brani acquistati fino a quel momento e il costo totale, permettendo al cliente di completare l'acquisto e procedere al pagamento.

   Si supponga che le credenziali dei clienti e le informazioni sui brani acquistabili siano memorizzate all'interno di tabelle SQL differenti di cui è richiesta la definizione della struttura. Le tabelle devono inoltre essere popolate con **almeno** tre utenti e cinque brani.

   Facoltativo 1: l'applicazione consente la modifica della password del cliente.
   Facoltativo 2 (difficile, richiede un approfondimento[2]): l'applicazione, concluso il pagamento, comprime i brani acquistati (in versione integrale) in un file .ZIP e invia il file compresso al cliente.

2 Per approfondire la tecnica di compressione di file nel formato .ZIP vedere: https://www.codejava.net/java-se/file-io/how-to-compress-files-in-zip-format-in-java