UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II

PROGETTO DELL'ESAME DI APPLICAZIONI TELEMATICHE

Studente: Salvatore MIROLLA
Matricola: M63000031

1.INTRODUZIONE

Lo studio delle Web application riveste un ruolo sempre più importante nell'ambito della programmazione per il Web. Infatti, la maggior parte della tecnologia moderna sia in ambito militare sia nelle aziende private che nelle pubbliche amministrazioni si basa su sistemi "Office Automation" condivisibili in rete, sia per la riduzione dei costi sia per la condivisione delle informazioni che per la realizzazione di sistemi distribuiti.

2. OBIETTIVO DEL PROGETTO

Scopo del progetto è quello di realizzare un progetto Web con la tecnologia Java allo scopo di gestire le informazioni riguardanti le tabelle organiche per la gestione delle risorse umane in ambito militare. In pratica, quando ad un soldato o a più soldati viene assegnato un incarico (il medesimo incarico può essere assegnato a più soldati per la sua realizzazione) in un determinato reparto, ovunque si trova geograficamente, è associato un binding tra l'incarico e il soldato o i soldati che memorizzeremo in un database. Tale binding dovrà solo essere memorizzato e non dovrà essere visualizzato sul Web poiché in ambito militare tale informazione è ritenuta riservata.

3. REQUISITI HARDWARE

Per il nostro progetto adotteremo i seguenti requisiti hardware:

- sistema operativo Windows 7 o versione superiore con architettura x64 e processore AMD o Intel pentium Dual Core o versione superiore
- sono richiesti almeno 4 GB di Ram
- sono richiesti almeno 256 GB di hard disk e file system NTFS

4. REQUISITI SOFTWARE

Come tecnologia di sviluppo si è scelto Java poiché risulta gratuita, soddisfa le prescrizioni europee di software libero e riutilizzabile a livello di pubblica amministrazione; é standardizzato sicuro e completo, poiché dotato di innumerevoli librerie scaricabili facilmente dal Web. Per la realizzazione del nostro progetto sono necessari i seguenti software:

- eclipse jee Neon(eclipse-inst-win64.exe)
- jdk-8u91-windows-x64.exe
- jre-8u91-windows-x64.exe
- Apache Tomcat ver. 8.0 x64

- Visual Studio 2008 oppure 2010
- DB MySQL: mysql-installer-community 5.7.14.0.msi
- DB MySQL: mysql-workbench-community 6.3.7.winx64.msi
- Driver MySQL: mysql-connector-java-5.1.39

5. DESIGN PATTERN

Come sistema di progettazione, largamente usato nello sviluppo di applicazioni semplici e complesse, così come descritto da Xerox in diverse pubblicazioni alla fine degli anni 80, useremo il modello MVC (Model View Controller). Quest'ultimo individua una logica di programmazione in tre componenti fondamentali distinte:

- **Model:** legato alla logica applicativa di persistenza e manipolazione dei dati, rappresenta i dati veri come quelli aziendali o di una pubblica amministrazione e non si interessa della loro presentazione o del loro uso. Tipicamente è realizzato tramite una serie di Javabeans che incapsulano delle risorse (Database, files, risorse di rete, informazioni di sessione, informazioni delle richieste http, etc.)

- **View:** legato alla presentazione con cui l'applicazione viene visualizzata all'utente e quindi specifica la modalità con cui interfacciare i dati con l'utente finale. Questa parte è tipicamente implementata tramite JSP che accedono ai Javabeans del Model.

- **Controller:** legato all'elaborazione delle richieste e lega il Model e il View in funzione delle richieste provenienti dal View o dei dati presenti nel Model. Questa parte è tipicamente realizzata tramite Servlet o JSP.

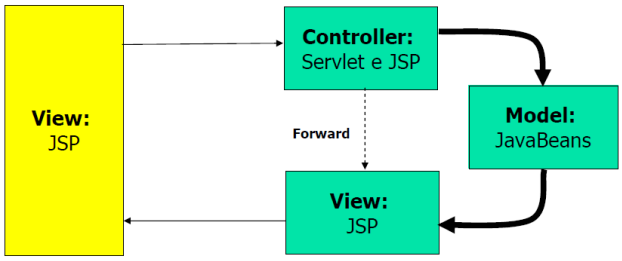

Fig.-5.1: Design Pattern:Il modello MVC

Tale livello di separazione è importante sia per motivi di stabilità dell'applicazione come aspetti grafici che possono cambiare in funzione delle esigenze del cliente sia per motivi di sicurezza in quanto la separazione dei componenti permette la

distribuzione degli stessi su differenti server con differenti modalità di accesso e sistemi di sicurezza.

6. NOZIONE DI BASE PER LA REALIZZAZIONE DI UNA WEB APPLICATION

Una Web Application è formata in genere da Servlet, pagine html o xhtml JSP, classi e altre componenti, che possono essere installate ed eseguite all'interno di un Container (Application Server) che presenta una JVM. Ad ogni Web Application è associato un ServletContext legato alla Servlet. Infine, un'applicazione è sempre mappata attraverso una gerarchia di directory memorizzabili su file system e solitamente esportati come file .war e .jar . Quindi ipotizzeremo che il progetto che andremo a sviluppare abbia come root la cartella progettoAT, questa verrà deployata sotto la directory webapps di Tomcat. Sotto la root potranno essere presenti altre directory come WEB-INF, accessibile, per problemi di sicurezza, solo dall'application Server. Quindi il front-end è a livello di root e il back-end è nascosto nella cartella WEB-INF. Tra le principali componenti di quest'ultima cartella si trovano web.xml, la cartella classes che contiene i files oggetto delle classi e Servlet e la cartella lib, che contiene tutte le librerie usate dal progetto in questione. Inoltre, il Web Application Server che useremo come Container è Tomcat 8.0, server capace di gestire nelle ultime versioni le Servlet e le pagine JSP che individuano la tecnologia per la creazione di pagine html o xhtml dinamiche lato server, così rappresentando un'estensione delle Servlet e fornendo un codice nel contempo statico(html) e dinamico(java). Quest'ultimo è contenuto nei tag <%%> del progetto e precompilato prima di inviare la risposta al client. Le Servlet sono classi Java che vengono eseguite nel Container, come sviluppato nel progetto con Tomcat 8.0, e vengono esposte come risorse Web standard all'esterno. Esse elaborano i dati provenienti dai form html agendo in modalità request/response, gestendo le richieste generate da uno o più client e le informazioni con stato e memorizzando i dati sul DB. Come già detto in precedenza una Servlet dispone di molteplici funzionalità e corrisponde al controller del pattern MVC. Avvalendosi delle java API implementano funzionalità importanti come l'accesso al DB. Nel nostro progetto cercheremo di realizzare un client che invia una richiesta (Request) ad una Servlet presente in un Web Application Server; il server istanzia e carica la Servlet avviando un thread per gestire la comunicazione; se la Servlet é stata già caricata in precedenza, supponendo differenti chiamate al server, allora, viene creato un ulteriore thread associato al nuovo client, senza ricaricare la Servlet. Il Server invia la richiesta del client alla Servlet e quest'ultima costruisce la risposta (Response) e la inoltra al server, che in ultimo, la re-invia al client. Si sottolinea che la Servlet non presenta quasi mai codice algoritmico

all'interno, ma questo viene demandato alle classi con cui la stessa interfaccia. Request e Response sono gestiti dalle relative interfacce e sono le seguenti:

- **javax.Servlet.http.HttpServletRequest** che consente ad una Servlet di ricavare informazioni sull'ambiente relativo al client.
- **javax.Servlet.ServletContext** che permette di trovare un riferimento al contesto di una applicazione, e quindi una serie di informazioni condivise a livello globale tra i componenti dell'applicazione stessa.

6.1. CICLO DI VITA DI UNA SERVLET

Le Servlet presentano un ciclo di vita ben definito e caratterizzato dai seguenti elementi e metodi:

- costruzione e dall'inizializzazione della Servlet attraverso il metodo init() invocato una tantum alla prima chiamata del Servlet Engine al termine del caricamento della Servlet;

- risposta alle richieste di tipo POST metodo doPost(HttpServletRequest req, HttpServletResponse res);

- risposta alle richieste di tipo GET metodo doGet(HttpServletRequest req, HttpServletResponse res);

- Chiusura e distruzione della servlet, metodo destroy(), invocato dal Servlet Engine per scaricare una Servlet dalla memoria.

Nel nostro progetto troveremo all'interno della Web Application l'uso delle "annotations" che rappresentano un modo per aggiungere metadati nel codice sorgente disponibili durante l'esecuzione come alternativa alla tecnologia XML contenuta nel web.xml, utili per le classi java ed interpretate correttamente dai container del Web. Nel seguente progetto verrà creata una servlet di controllo che individuerà differenti azioni in seguito alla valutazione di parametri provenienti dal form di inserimento dati. Le azioni saranno conseguenze di chiamate al metodo doGet tramite link diretti e al metodo doPost tramite submit dei form.

7. INSTALLAZIONE DEL DB MySQL

MySQL, appartenente da qualche anno ad Oracle è un database relazionale (RDBMS) professionale, versatile e potente che lavora sia in ambiente Unix che in

Windows, molto utile per la creazione e la gestione di DB in ambienti di sviluppo e di produzione. Fondamentalmente è un software libero e quindi viene molto usato sia come DB per siti e applicativi che per applicativi professionali utilizzati nelle pubbliche amministrazioni. Di seguito indicheremo i passi principali per l'installazione e la creazione della Web Application da sviluppare. Per installare MySQL è sufficiente andare sul sito di Oracle http://www.mysql.it e scaricare due elementi importanti che sono i seguenti:

- DB MySQL: mysql-installer-community 5.7.14.0.msi dal sito http://dev.mysql.com/downloads/mysql/ che é il DB.
- MySQL Workbench: DB MySQL: mysql-workbench-community 6.3.7.winx64.msi dal sito http://dev.mysql.com/downloads/workbench/ che l'applicazione visuale che verrà usata per la creazione e la gestione del DB e dei relativi scheme-data.

Di seguito si descrivono passo per passo le indicazioni per l'installazione del DB su sistema operativo Windows 10 come riportato di seguito:

1. Eseguire come amministratore e lanciare l'applicativo mysql-installer-community 5.7.14.0.msi accettando la licenza e premendo next;

2. Successivamente nella finestra che compare, riportata qui di seguito, si lasci come tipologia di installazione l'opzione del Wizard Developer Default;

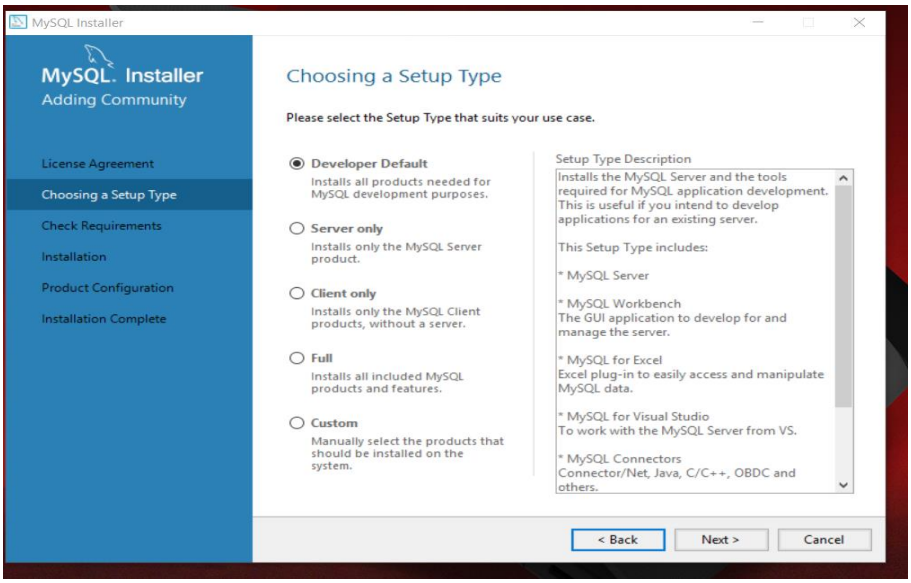

Fig.-7.1: Tipologia di Installazione Server MySQL

3. Successivamente partirà l'installazione dei server con le varie componenti come riportato nella figura qui di seguito;

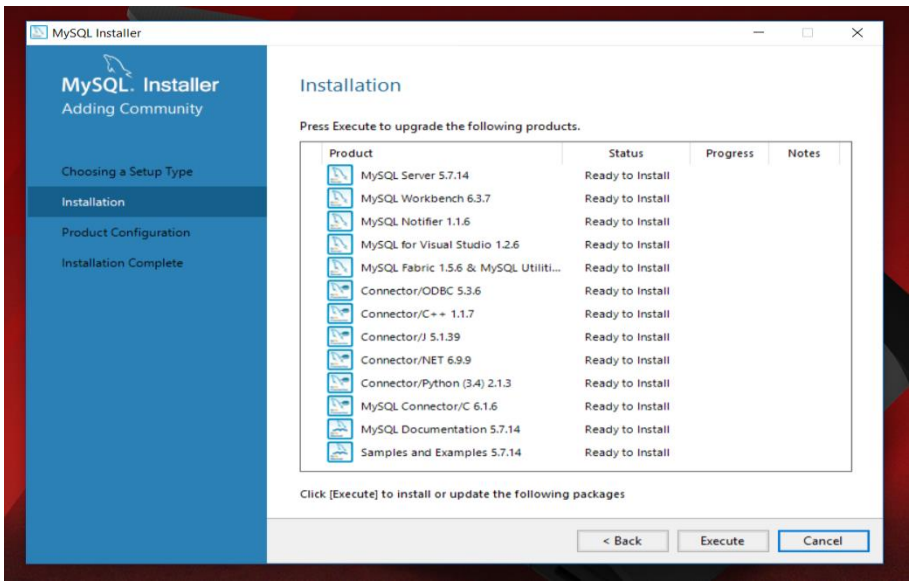

Fig.-7.2: Installazione dei componenti del Server MySQ

4. Al termine dell'installazione la figura 7.2 assumerà l'aspetto riportato qui di seguito. Le spunte di colore verde stanno a significare che tutti i componenti del DB sono stati installati correttamente;

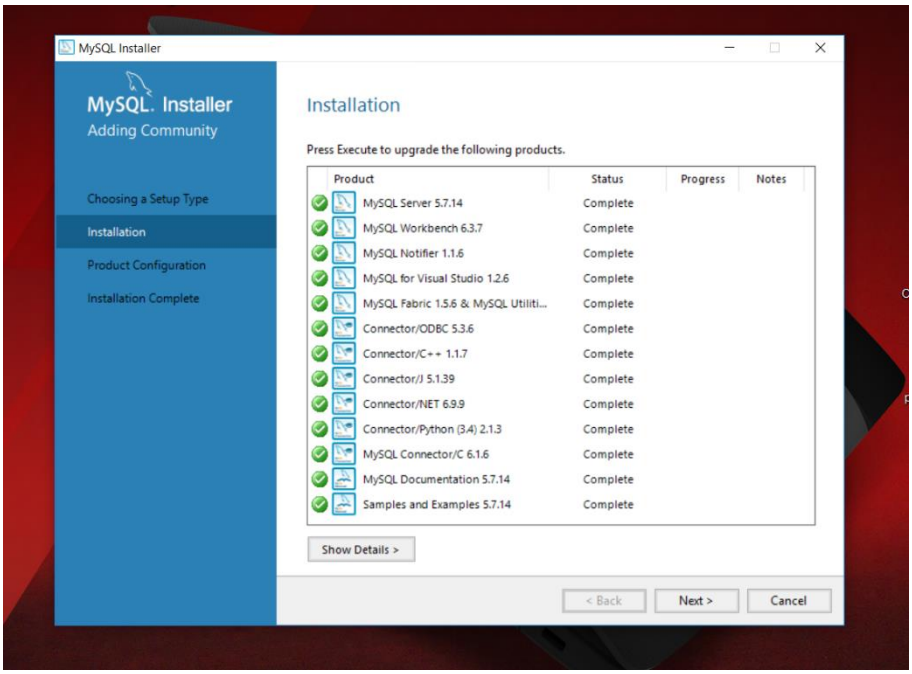

Fig.-7.3: Componenti del Server MySQL installati correttamente

5. Successivamente viene proposta l'opzione del Wizard per la configurazione della porta standard 3306 come riportato nella figura qui di seguito;

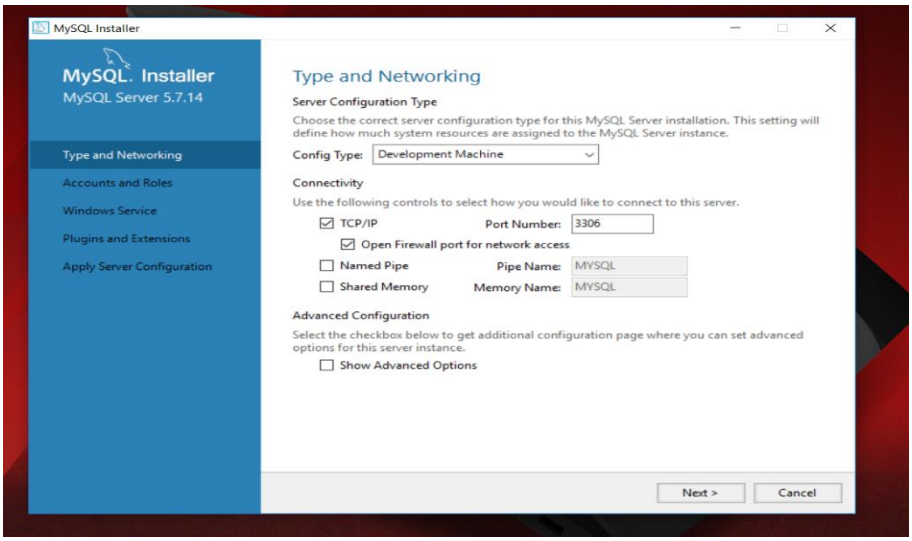

Fig.-7.4: Configurazione della porta relativa all'installazione del Server MySQL

6. Successivamente viene chiesta di impostare nell'opzione del Wizard la password di amministratore. Nel caso specifico impostiamo come password:

Admin e come nome utente: root così come riportato nella figura qui di seguito;

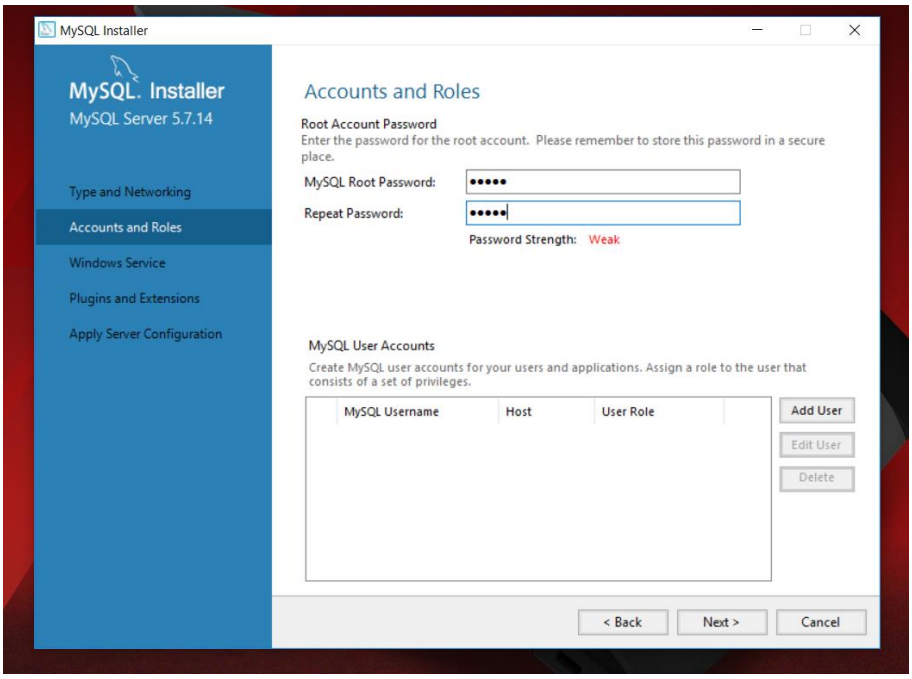

Fig.-7.5: Configurazione della password di Amministratore relativa all'installazione del Server MySQL

7. Successivamente viene fornita la possibilità di inserire il server come servizio di Windows come riportato nella figura qui di seguito:

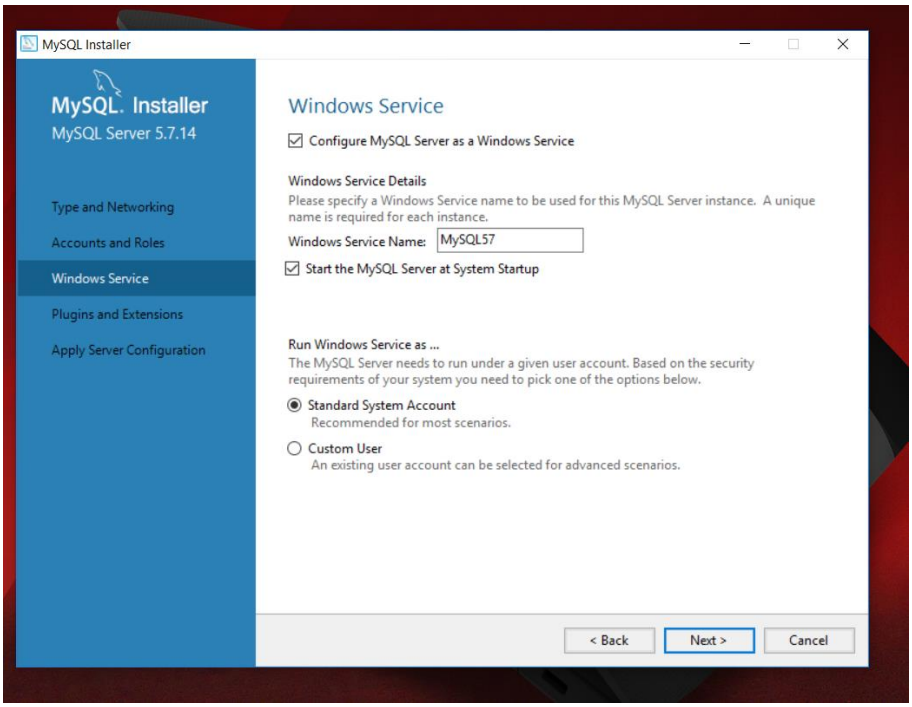

Fig.-7.6: Configurazione dell'installazione del Server MySQL come servizio di Windows

8. Successivamente vengono applicate le impostazioni relative ai tipi di plugin ed estensioni come riportato nella figura qui di seguito;

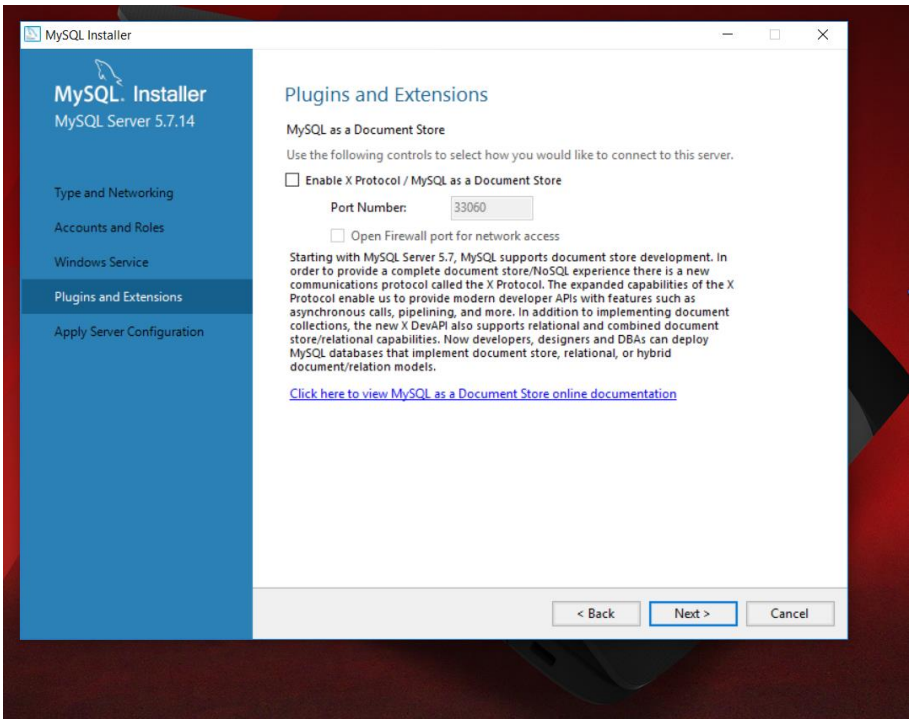

Fig.-7.7: Configurazione delle estensioni e dei plugin del Server MySQL

9. Successivamente vengono applicate le impostazioni relative ai servizi su Windows 10 come riportato nella figura qui di seguito;

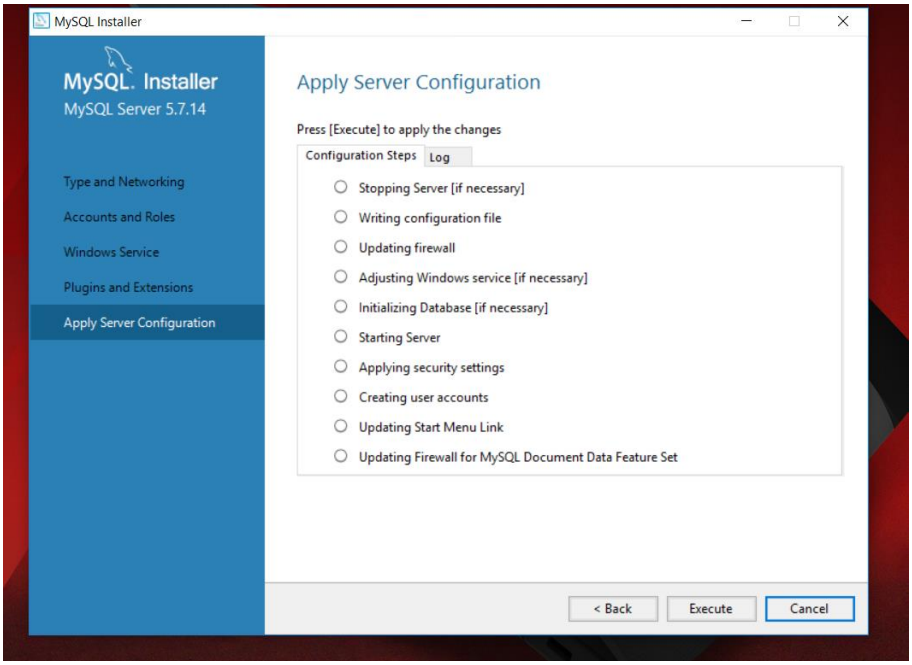

Fig.-7.8: Configurazione di alcuni servizi del Server MySQL

10. Successivamente le spunte di colore verde confermano la corretta configurazione dei servizi su Windows 10 nella figura qui di seguito;

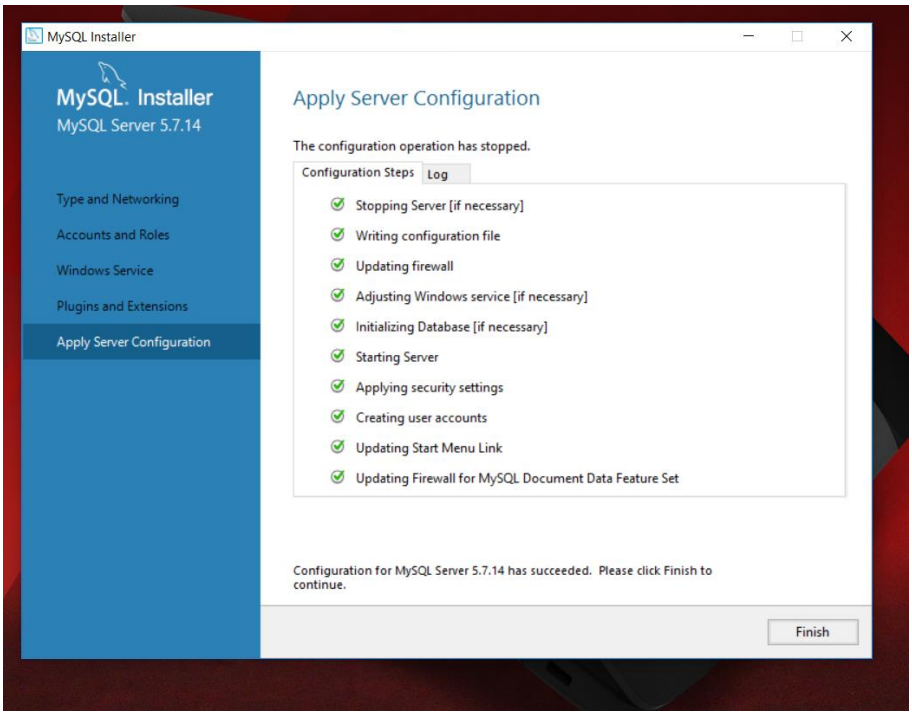

Fig.-7.9: Configurazione dei servizi del Server MySQL installati correttamente

11. Successivamente viene verificata la connessione con il server come riportato nella figura qui di seguito;

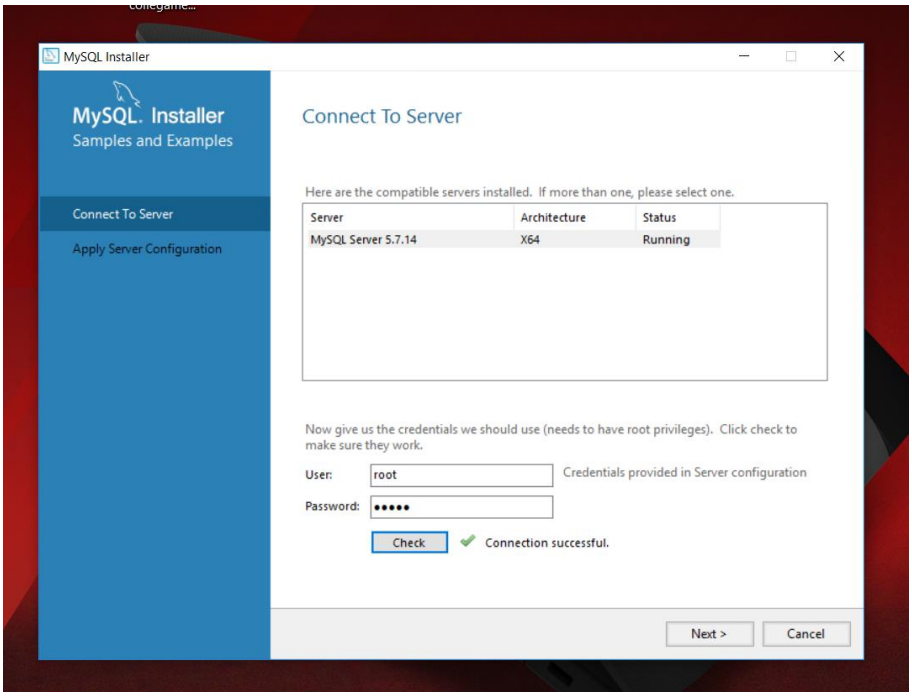

Fig.-7.10: Verifica della connessione del Server MySQL

12. Infine vengono mostrati gli ultimi passi del processo di installazione del DB come riportato nella figura qui di seguito;

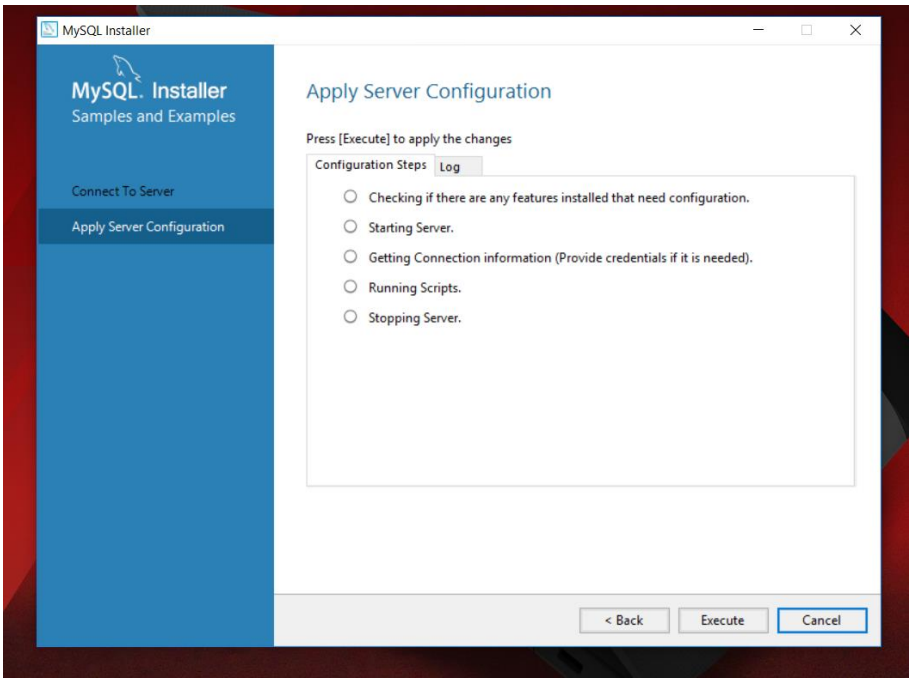

Fig.-7.11:Ultimi passi dell' Installazione del Server MySQL

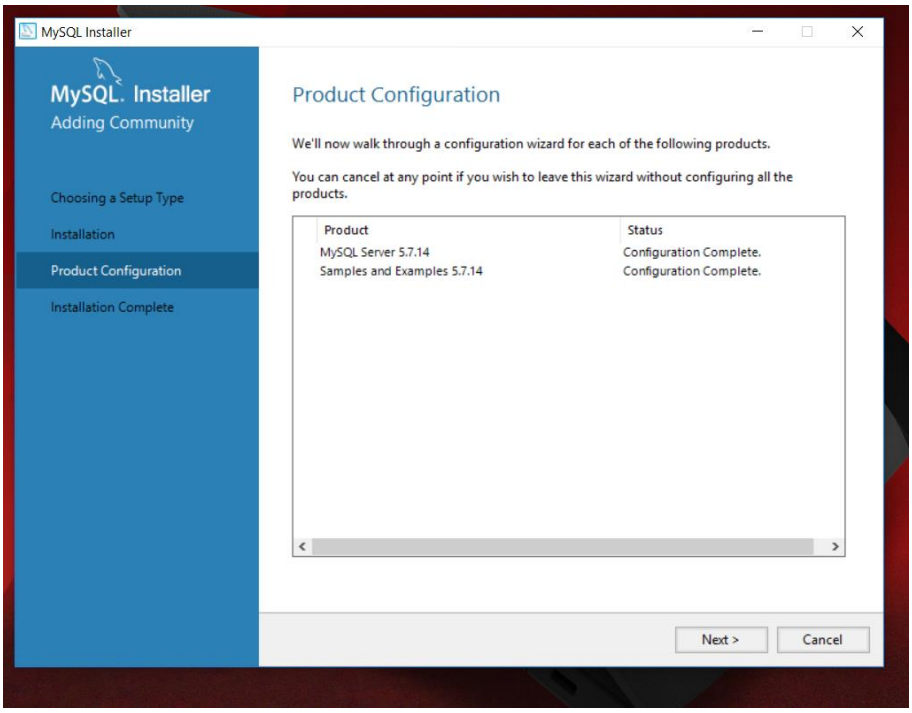

Fig.-7.12: Ultimi passi dell' Installazione del Server MySQL

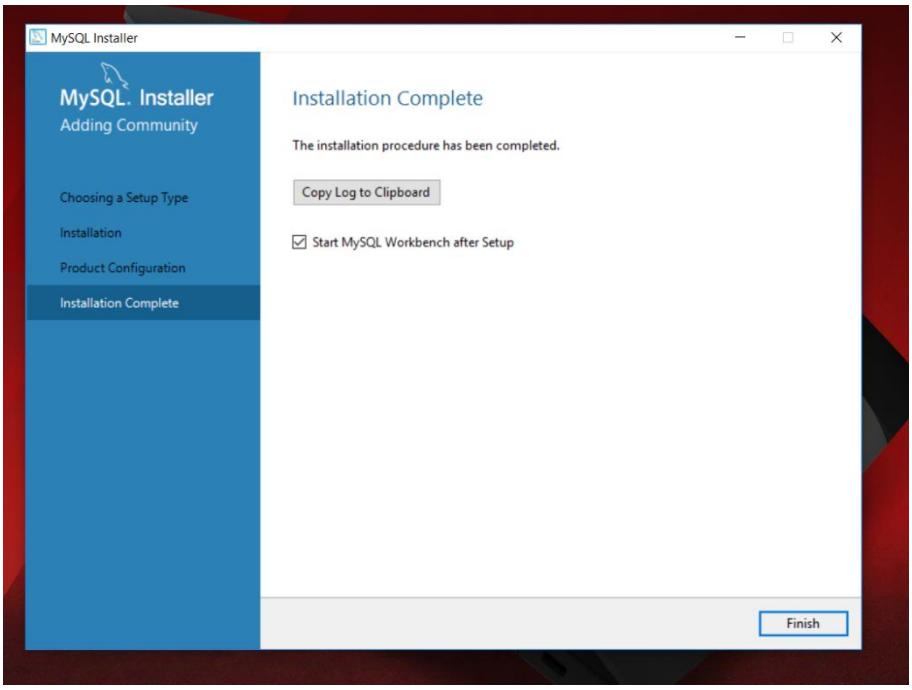

Fig.-7.13: Ultimi passi dell' Installazione del Server MySQL

13. Per quanto riguarda l'installazione del mysql-workbench-community 6.3.7.winx64.msi, lanciare il pacchetto come amministratore, selezionare il tipo di installazione completa e attenderne la corretta configurazione.

8. PROGETTAZIONE ED IMPLEMENTAZIONE DEL DB MYSQL PER LA WEB APPLICATION

Si ipotizzi di progettare un db per poter salvare gestire un certo numero di incarichi assegnati ad uno o più militari. Partiamo dalla progettazione secondo il modello entità relazioni. L'analisi dei requisiti ha portato a definire due entità fondamentali: l'incarico del Reparto o Ufficio ed il militare, legate dal fatto che ad ogni incarico corrispondono uno o più militari e che un militare possa avere soltanto un incarico. Un incarico di reparto, o semplicemente reparto, avrà i seguenti campi:

- id_company: stringa alfanumerica per identificare nella tabella organica di reparto o d'ufficio il particolare incarico.
- nome_reparto: nome del reparto a cui corrisponde quel determinato incarico.
- ufficio: nome dell'ufficio appartenente ad un determinato reparto a cui corrisponde quel determinato incarico.
- sezione: nome della sezione facente parte di un determinato ufficio il quale a sua volta e' parte di un determinato reparto.
- incarico: tipo di incarico che dovrà essere assegnato ad uno oppure più militari

- cell_svz: utenza telefonica che viene fornita ad un militare che ha un determinato incarico
- ruolo_protocollo_inf: stringa che specifica il ruolo in un applicativo documentale chiamato Adhoc per la gestione delle pratiche e che è assegnato ad uno oppure più militari.
- date_ins: data di inserimento dei dati del database

Il militare sarà caratterizzato dai seguenti campi:

- cf: codice fiscale del militare
- grado: grado del militare
- categoria: tale stringa determina la specialità del militare ovvero definisce che tipo di incarico può svolgere;
- nome: il nome del militare
- cognome: il cognome del militare
- data_nascita: data di nascita del militare
- luogo_nascita: luogo di nascita del militare
- cmd:badge assegnato al militare avente una stringa univoca
- fk_company: legame tra l'entità militare e l'entità incarico di reparto.

Quindi creiamo il database con le seguenti tabelle:

- reparto
- persomil

Table reparto:

NOME CAMPO	LUNGHEZZA	TIPO	VINCOLI
id_company	**45**	**VARCHAR**	**Primary Key**
nome_reparto	45	VARCHAR	Not Null
ufficio	**45**	**VARCHAR**	**Not Null**
sezione	45	VARCHAR	Not Null
incarico	**45**	**VARCHAR**	**Not Null**
email_incarico	45	VARCHAR	Not Null
cell_svz	**45**	**VARCHAR**	**Not Null**
ruolo_protocollo_inf	45	VARCHAR	Not Null
date_ins	**-**	**DATE**	**Not Null**

Table persomil:

NOME CAMPO	LUNGHEZZA	TIPO	VINCOLI
cf	45	VARCHAR	Primary Key
grado	45	VARCHAR	Not Null
categoria	45	VARCHAR	Not Null
nome	45	VARCHAR	Not Null
cognome	45	VARCHAR	Not Null
data_nascita	45	VARCHAR	Not Null
luogo_nascita	45	VARCHAR	Not Null
cmd	45	VARCHAR	Not Null
fk_company	45	VARCHAR	Not Null (Vincolo di integrità referenziale)
date	-	DATE	Not Null

Un'ultima considerazione è da farsi sulla **progettazione logica/concettuale**. Con il Workbench è possibile sia partire, come appena fatto dalla definizione delle tabelle a valle **della progettazione concettuale e logica**, per poi effettuare l'implementazione fisica della DB, oppure partire da zero dallo schema entità-relazioni per poi ottenere le tabelle. In ultimo, è possibile estrarre lo schema grafico della progettazione logica dagli schemi e dalle tabelle esistenti tramite una modalità **di Reverse Engineer** e cioé un grafico che può essere inserito in documentazione. È sufficiente andare in Home e dal tab Database selezionare Reverse Engineer. Successivamente è sufficiente seguire le istruzioni del Wizard cliccando su next e infine selezionando il DB attraverso l'inserimento del nome utente e della password e successivamente selezionando lo schema in questione. È possibile quindi ottenere lo schema logico di riferimento come mostrato in figura qui di seguito:

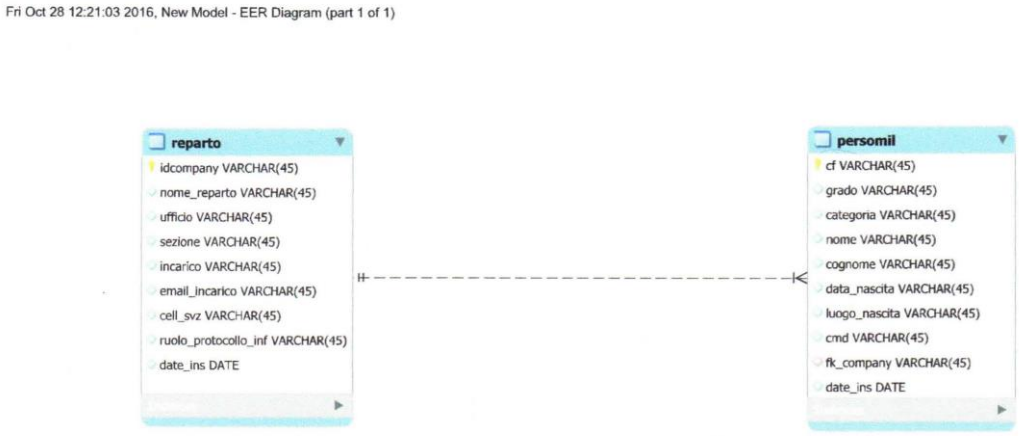

Fig.-7.13: Schema entità relazioni generate del Server MySQL

9. INSTALLAZIONE DELL'IDE ECLIPSE

- Scarichiamo l'IDE eclipse jee Neon (eclipse-inst-win64.exe) dal sito www.eclipse.org e installiamolo nella directory C:\users\sasaman\Desktop\eclipse
- All'atto del primo lancio dell'IDE ci verrà chiesto di selezionare o meglio di configurare la directory per il salvataggio dei progetti. Impostiamo quindi il seguente percorso C:\users\sasaman\Desktop\eclipse. D'ora in poi ogni progetto che creeremo lo troveremo nella cartella eclipse. Infine spuntiamo l'opzione: "Use this as default and do not ask again" per non impostare ogni volta la directory di salvataggio dei progetti..
- Per poter lavorare con Java in ambiente eclipse occorre installare e configurare in ambiente eclipse la JDK. Scarichiamo dal sito di Oracle il pacchetto jdk-8u91-windows-x64.exe e installiamolo. Successivamente configuriamolo nel-l'IDE andando in Windows → Preferences → JAVA → Installed JREs e cerchiamo il percorso dove è installata la JDK.

10. INSTALLAZIONE DI APACHE TOMCAT 8.0 E CONFIGURAZIONE IN ECLIPSE

- Scarichiamo dal sito http://tomcat.apache.org/ il pacchetto Apache Tomcat ver. 8.0 x64 e scompattiamolo in C:\. Il file scompattato genera una cartella Apache Software Foundation che include la cartella Tomcat-8.0 che sarà la cartella di produzione e nel caso in esame, anche di sviluppo dell'applicazione. In una situazione aziendale che prevede diversi ambienti (sviluppo, test, Collaudo, produzione) è fondamentale mantenere la coerenza tra le versioni del server nei diversi ambienti per evitare errori.

- È necessario, in sviluppo locale, configurare il server nell'IDE di riferimento per avere sotto controllo i parametri del server stesso. Quindi andiamo in eclipse selezioniamo il tab server, clicchiamo sul collegamento che chiede la configurazione di un nuovo server e scegliamo Apache Tomcat 8.0.

- Dopo la configurazione del server e' possibile andare sul tab server dove comparirà il server appena configurato. È sufficiente con il tasto destro aprire il menu contestuale ed avviare il server. Se si va sul browser è possibile vedere il server che è attivo, digitando la seguente url sulla barra degli indirizzi: http://localhost:8080/ apparirà la pagina di benvenuto del server.

11. STRUTTURA ED IMPLEMENTAZIONE DELLA WEB APPLICATION

Creiamo la pagina jsp

formPersomil.jsp

```
1 <%@ page import="progettoAT.bean.RepartoBean"%>
2 <%@ page import="progettoAT.bean.PersomilBean"%>
3 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
4 pageEncoding="ISO-8859-1"%>
5 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
6
7
8
<%
9 String cf="";
10 String grado="";
11 String categoria="";
12 String nome="";
13 String cognome="";
14 String data_nascita="";
15 String luogo_nascita="";
16 String cmd="";
17 String fk_company="";
18
19 PersomilBean employee=new PersomilBean();
20 boolean employeeFilled=false;
21
22 if(request.getSession()!=null && request.getSession().getAttribute("EMPLOYEE")
!=null){
23 employee=(PersomilBean)request.getSession().getAttribute("EMPLOYEE");
24 cf=employee.getCf()+"";
25 grado=employee.getGrado()+"";
26 categoria=employee.getCategoria()+"";
27 nome=employee.getNome()+"";
28 cognome=employee.getCognome()+"";
29 cmd=employee.getCmd();
30 fk_company=employee.getFk_company()+"";
31
32 employeeFilled=true;
33 }
34
35 RepartoBean company=new RepartoBean();
36 if(request.getSession()!=null && request.getSession().getAttribute("COMPANY")
!=null){
37 company=(RepartoBean)request.getSession().getAttribute("COMPANY");
38 fk_company=company.getIdcompany()+"";
39 }
40
41
42
43 %>
44
45
46
47
48
49 <html>
50 <head>
```

```
51 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
52 <title>PERSOMIL</title>
53 </head>
54
55
56
57
58
59 <body>
60
61
62
63
64
65
66 <table width="100%">
67
68 <tr>
69 <td width="10%" align="left"> <img alt=""
src="C:\Users\sasaman\Desktop\eclipse\progettoAT\img\RIS.JPG"></td>
70
71 <td width="80%" align="center"> <b><font size="10">PERSONALE
MILITARE</font></b></td>
72
73 <td width="10%" align="right"> <img alt=""
src="C:\Users\sasaman\Desktop\eclipse\progettoAT\img\GAB.JPG"></td>
74 </tr>
75
76 </table>
77
78 <img width="100%" alt=""
src="C:\Users\sasaman\Desktop\eclipse\progettoAT\img\BARRA2.JPG">
79
80  
81
82 <table>
83
84
85 <tr>
86 <td>
87 <a href="/progettoAT/RepartoMgmServlet?whatsend=homepage" target="_top">HOME
PAGE</a>
88  
89 <a href="/progettoAT/RepartoMgmServlet?whatsend=company" target="_top">REPARTO</a>
90  
91
92
93 <%if(employeeFilled){ %>
94 <a href="/progettoAT/RepartoMgmServlet?whatsend=employee" target="_top"> INSERISCI
UN
ALTRO MILITARE</a>
95  
96 <a href="/progettoAT/RepartoMgmServlet?whatsend=saveCompany" target="_top"> SALVA
DATI</a>
97
98
99 <%} %>
100
101 </td>
102 </tr>
```

```
103
104
105 <tr>
106 <td>
107 <form name="persomilForm" action="/progettoAT/RepartoMgmServlet" method="post">
108
109 <table border="0" cellspacing="10" cellpadding="10">
110
111 <tr>
112 <td> CODICE FISCALE: </td>
113 <td>
114 <%if(employeeFilled){ %>
115 <%=cf %>
116 <%}else{ %>
117 <input name="cf" value=" <%=cf %>" type="text" maxlength="45" size="45" >
118 <%} %>
119 </td>
120 </tr>
121
122
123
124 <tr>
125 <td>GRADO:</td>
126 <td>
127
128 <input name="grado" value=" <%=grado %>" type="text" maxlength="45"
size="45">
129
130 </td>
131 </tr>
132
133 <tr>
134 <td>CATEGORIA:</td>
135 <td>
136
137 <input name="categoria" value=" <%=categoria %>" type="text"
maxlength="45" size="45">
138
139 </td>
140 </tr>
141
142
143 <tr>
144 <td>NOME:</td>
145 <td>
146
147 <input name="nome" value=" <%=nome %>" type="text" maxlength="45"
size="45">
148
149 </td>
150 </tr>
151
152
153 <tr>
154 <td>COGNOME:</td>
155 <td>
156
157 <input name="cognome" value=" <%=cognome %>" type="text" maxlength="45"
size="45">
158
```

```
159 </td>
160 </tr>
161
162
163
164 <tr>
165 <td>DATA DI NASCITA:</td>
166 <td>
167
168 <input name="data_nascita" value=" <%=data_nascita %>" type="text"
maxlength="45" size="45">
169
170 </td>
171 </tr>
172
173
174 <tr>
175 <td>LUOGO DI NASCITA:</td>
176 <td>
Page 3
formPersomil.jsp
177
178 <input name="luogo_nascita" value=" <%=luogo_nascita %>" type="text"
maxlength="45" size="45">
179
180 </td>
181 </tr>
182
183
184
185 <tr>
186 <td>NUMERO CMD:</td>
187 <td>
188
189 <input name="cmd" value=" <%=cmd %>" type="text" maxlength="45" size="45">
190
191 </td>
192 </tr>
193
194
195
196
197
198
199
200 </table>
201
202
203 <input name="FK_company" value=" <%=fk_company %>" type="hidden">
204 <input name="whatsend" value="employee" type="hidden">
205 <input type="submit" value="INSERISCI MILITARE">
206
207 </form>
208
209</table>
210</body>
211</html>
```

Creiamo la pagina jsp

formReparto.jsp

```
1 <%@ page import="progettoAT.bean.RepartoBean" %>
2 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
3 pageEncoding="ISO-8859-1"%>
4 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
56
<%
7 String idcompany="";
8 String nome_reparto="";
9 String ufficio="";
10 String sezione="";
11 String incarico="";
12 String email_incarico="";
13 String cell_svz="";
14 String ruolo_protocollo_inf="";
15
16 //Campi provenienti dalla sessione se attualmente presenti
17 RepartoBean company=new RepartoBean();
18 boolean companyFilled=false;
19
20 if(request.getSession()!=null && request.getSession().getAttribute("COMPANY")
!=null){
21 company=(RepartoBean)request.getSession().getAttribute("COMPANY");
22
23
24 idcompany=company.getIdcompany()+"";
25 nome_reparto=company.getNome_reparto()+"";
26
27
28 companyFilled=true;
29 }
30 %>
31
32
33
34
35
36
37
38
39
40
41 <html>
42 <head>
43 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
44 <title>REPARTO</title>
45 </head>
46
47
48
49
50
51 <body>
52
53 <table width="100%">
54 <tr>
55 <td width="10%" align="left"> <img alt=""
```

```
src="C:\Users\sasaman\Desktop\eclipse\progettoAT\img\RIS.JPG"></td>
56
57 <td width="80%" align="center"> <b><font size="10">ENTI, REPARTI E
DISTACCAMENTI</font></b></td>
58
59 <td width="10%" align="right"> <img alt=""
src="C:\Users\sasaman\Desktop\eclipse\progettoAT\img\GAB.JPG"></td>
60 </tr>
61
62
63
64
65 </table>
66
67 <img width="100%" alt=""
src="C:\Users\sasaman\Desktop\eclipse\progettoAT\img\BARRA2.JPG">
68
69  
70
71 <table>
72
73 <tr>
74 <td> <a href="/progettoAT/RepartoMgmServlet?whatsend=homepage" target=_top>HOME
PAGE</a>
75
76  
77
78 <a href="/progettoAT/RepartoMgmServlet?whatsend=deletePersomil"
target=_top>ELIMINA MILITARE</a>
79
80 <%if(companyFilled){ %>
81 <a href="/progettoAT/RepartoMgmServlet?whatsend=employeeInsert" target="_top">
INSERISCI MILITARE</a>
82 <%} %>
83 </td>
84 </tr>
85
86
87 <tr>
88 <td>
89
90 <form name="repartoForm" action="/progettoAT/RepartoMgmServlet" method="post">
91
92 <table border="0" cellspacing="10" cellpadding="10">
93 <tr>
94 <td>ID REPARTO:</td>
95 <td>
96 <%if(companyFilled){ %>
97 <%=idcompany %>
98 <%}else{ %>
99 <input name="idcompany" value=" <%=idcompany %>" type="text"
maxlength="45" size="35">
100 <%} %>
101 </td>
102 </tr>
103
104
105 <tr>
106 <td>NOME REPARTO:</td>
107 <td>
```

```
108
109 <input name="nome_reparto" value=" <%=nome_reparto %>" type="text"
maxlength="45" size="35">
110
111 </td>
112 </tr>
113
114
115 <tr>
116 <td>UFFICIO:</td>
117 <td>
118
119 <input name="ufficio" value=" <%=ufficio %>" type="text" maxlength="45"
size="35">
120
121 </td>
122 </tr>
123
124
125
126 <tr>
127 <td>SEZIONE:</td>
128 <td>
129
130 <input name="sezione" value=" <%=sezione %>" type="text" maxlength="45"
size="35">
131
132 </td>
133 </tr>
134
135
136 <tr>
137 <td>INCARICO:</td>
138 <td>
139
140 <input name="incarico" value=" <%=incarico %>" type="text" maxlength="45"
size="35">
141
142 </td>
143 </tr>
144
145
146 <tr>
147 <td>EMAIL INCARICO:</td>
148 <td>
149
150 <input name="email_incarico" value=" <%=email_incarico %>" type="text"
maxlength="45" size="35">
151
152 </td>
153 </tr>
154
155
156 <tr>
157 <td>CELL. DI SERVIZIO:</td>
158 <td>
159
160 <input name="cell_svz" value=" <%=cell_svz %>" type="text" maxlength="45"
size="35">
161
```

```
162 </td>
163 </tr>
164
165
166
167
168 <tr>
169 <td>RUOLO DI PROT. INF.:</td>
170 <td>
171
172 <input name="ruolo_protocollo_inf" value=" <%=ruolo_protocollo_inf %>"
type="text" maxlength="45" size="35">
173
174 </td>
175 </tr>
176
177 </table>
178 <input name="whatsend" value="company" type="hidden">
179 <input type="submit" value="INSERISCI REPARTO">
180
181 </form>
182 </td>
183 </tr>
184
185
186
187
188
189 </table>
190
191
192
193 </body>
194 </html>
```

Nella pagina JSP formPersomil è possibile notare come il militare presenti un campo hidden con nome FK_company. Tale campo é valorizzato dal campo id_company inserita la corrispondente posizione tabellare organica di reparto. Il tutto verrà implementato con un sistema di sessioni. Per distinguere i due inserimenti da parte di due form differenti, è possibile lato server in maniera semplificata inserire un campo hidden che permetta di distinguere se il submit proviene dal form Reparto oppure dal form Persomil e quindi agire di conseguenza salvando i relativi campi. Il campo whatsend serve per stabilire l'azione nella servlet. Di seguito si riporta il codice della della Servlet:

RepartoMgmServlet.java

```
1 package progettoAT;
23
import java.io.IOException;
21
22 /**
23 * Servlet implementation class RepartoMgmServlet
24 */
25 @WebServlet("/RepartoMgmServlet")
26 public class RepartoMgmServlet extends HttpServlet {
27 private static final long serialVersionUID = 1L;
```

```
28
29 /**
30 * @see HttpServlet#HttpServlet()
31 */
32 public RepartoMgmServlet() {
33 super();
34 // TODO Auto-generated constructor stub
35 }
36
37 /**
38 * @see Servlet#init(ServletConfig)
39 */
40 public void init(ServletConfig config) throws ServletException {
41 // TODO Auto-generated method stub
42 }
43
44 /**
45 * @see Servlet#destroy()
46 */
47 public void destroy() {
48 // TODO Auto-generated method stub
49 }
50
51 /**
52 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
53 */
54 protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws
ServletException, IOException {
55 String whatsend=request.getParameter("whatsend");
56 System.out.println("whatsend GET :: "+whatsend);
57
58 if(whatsend.equals("employeeInsert")){
59
60 ServletContext sc=request.getSession().getServletContext();
61 RequestDispatcher rd=sc.getRequestDispatcher("/formPersomil.jsp");
62 rd.forward(request, response);
63
64
65 }
66 else if(whatsend.equals("employee")){
67 request.getSession().removeAttribute("EMPLOYEE");
68 ServletContext sc=request.getSession().getServletContext();
69 RequestDispatcher rd=sc.getRequestDispatcher("/formPersomil.jsp");
70 rd.forward(request, response);
71
72
73
74 }
75
76 else if(whatsend.equals("company")){
77
78 ServletContext sc=request.getSession().getServletContext();
79 RequestDispatcher rd=sc.getRequestDispatcher("/formReparto.jsp");
80 rd.forward(request, response);
81
82
83
84 }
85
```

```
86 else if(whatsend.equals("deletePersomil")){
87 ServletContext sc=request.getSession().getServletContext();
88 RequestDispatcher rd=sc.getRequestDispatcher("/formDeletePersomil.jsp");
89 rd.forward(request, response);
90
91
92
93
94 }
95 else if(whatsend.equals("homepage")){
96 //response.sendRedirect("/company_management/hello.jsp");
97 request.getSession().removeAttribute("COMPANY");
98 request.getSession().removeAttribute("EMPLOYEE");
99 response.sendRedirect("/progettoAT/formReparto.jsp");
100 }
101
102
103 else if(whatsend.equals("saveCompany")){
104 RepartoBean company=new RepartoBean();
105 company=(RepartoBean)request.getSession().getAttribute("COMPANY");
106 SaveMySQL saveCompany=new SaveMySQL();
107
108
109 try{
110 saveCompany.insertCompany(company);
111 }catch(SQLException e){
112 System.out.println("ERROR:"+e.getErrorCode()+":"+e.getMessage());
113 e.printStackTrace();
114 }
115
116
117
118 ArrayList<RepartoBean> companyInDB=new ArrayList<RepartoBean>();
119
120
121 try{
122 companyInDB=saveCompany.searchCompanies();
123
124
125 }catch(SQLException e){
126 System.out.println("ERROR:"+e.getErrorCode()+":"+e.getMessage());
127 e.printStackTrace();
128 }
129 ServletContext sc=request.getSession().getServletContext();
130 request.getSession().removeAttribute("COMPANIES");
131 request.getSession().setAttribute("COMPANIES", companyInDB);
132
133 RequestDispatcher rd=sc.getRequestDispatcher("/ListReparto.jsp");
134 rd.forward(request, response);
135 }
136
137
138
139
140
141
142 }
143
144 /**
145 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
```

```
146 */
147 protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws
ServletException, IOException {
148
149 String whatsend=request.getParameter("whatsend");
150 System.out.println("whatsend POST :: "+whatsend);
151 RepartoBean company=new RepartoBean();
152
153
154
155
156 if(whatsend.equals("company")){
157
158 String idcompany=request.getParameter("idcompany");
159 System.out.println("idcompany :: "+idcompany);
160
161 String nome_reparto=request.getParameter("nome_reparto");
162 System.out.println("nome_reparto:: "+nome_reparto);
163
164 String ufficio=request.getParameter("ufficio");
165 System.out.println("ufficio :: "+ ufficio);
166
167
168 String sezione=request.getParameter("sezione");
169 System.out.println("sezione :: "+ sezione);
170
171 String incarico=request.getParameter("incarico");
172 System.out.println("incarico :: "+ incarico );
173
174
175 String email_incarico=request.getParameter("email_incarico");
176 System.out.println("email_incarico :: "+email_incarico);
177
178 String cell_svz =request.getParameter("cell_svz");
179 System.out.println(" cell_svz :: "+ cell_svz );
180
181 String ruolo_protocollo_inf =request.getParameter("ruolo_protocollo_inf");
182 System.out.println(" ruolo_protocollo_inf :: "+ ruolo_protocollo_inf );
183
184
185
186 ArrayList<PersomilBean> companyEmployees=new ArrayList<PersomilBean>() ;
187
188 company.setIdcompany(idcompany);
189 company.setNome_reparto(nome_reparto);
190 company.setUfficio(ufficio);
191 company.setSezione(sezione);
192 company.setIncarico(incarico);
193 company.setEmail_incarico(email_incarico);
194 company.setCell_svz(cell_svz);
195 company.setRuolo_protocollo_inf(ruolo_protocollo_inf);
196 company.setRepartoEmployees(companyEmployees);
197
198 ServletContext sc=request.getSession().getServletContext();
199 request.getSession().removeAttribute("COMPANY");
200 request.getSession().setAttribute("COMPANY", company);
201
202 RequestDispatcher rd=sc.getRequestDispatcher("/formReparto.jsp");
203 rd.forward(request, response);
```

```
204
205
206
207 PrintWriter out=response.getWriter();
208 out.println("<html><body>Inserimento azienda avvenuto con
successo</body></html>");
209
210 }
211
212
213
214 else if(whatsend.equals("deletePersomil")){
215
216
217 String cf=request.getParameter("cf");
218 System.out.println("cf :: "+cf);
219
220
221 PersomilBean personale=new PersomilBean();
222 personale.setCf(cf);
223 int j;
224
225 try {
226
227 DeleteMySQL d=new DeleteMySQL();
228 d.readPersomil();
229 j=d.controllo(cf);
230
231 if(j>=0){ d.delete(cf);}
232 else{
233 System.out.println("Impossibile eliminare l'utente con cf:"+cf+ "
poichè non presente nell' archivio dati");
234 }
235
236
237
238
239
240 } catch (ClassNotFoundException | SQLException e) {
241 // TODO Auto-generated catch block
242 e.printStackTrace();
243 }
244
245 ServletContext sc=request.getSession().getServletContext();
246 request.getSession().removeAttribute("EMPLOYEE");
247 request.getSession().setAttribute("EMPLOYEE", personale);
248
249 RequestDispatcher rd=sc.getRequestDispatcher("/formDeletePersomil.jsp");
250 rd.forward(request, response);
251
252
253
254
255
256
257
258
259
260
261
```

```
262
263
264 }
265
266
267
268 else if(whatsend.equals("employee")){
269
270 String cf=request.getParameter("cf");
271 System.out.println("cf :: "+cf);
272
273 String grado =request.getParameter("grado");
274 System.out.println(" grado:: "+grado);
275
276 String categoria=request.getParameter("categoria");
277 System.out.println(" categora:: "+categoria);
278
279 String nome=request.getParameter("nome");
280 System.out.println(" nome:: "+nome);
281
282 String cognome=request.getParameter("cognome");
283 System.out.println(" cognome:: "+cognome);
284
285 String data_nascita=request.getParameter("data_nascita");
286 System.out.println("data di nascita :: "+data_nascita);
287
288 String luogo_nascita=request.getParameter("luogo_nascita");
289 System.out.println("luogo di nascita :: "+luogo_nascita);
290
291 String cmd=request.getParameter("cmd");
292 System.out.println("cmd :: "+cmd);
293
294 String fk_company=request.getParameter("FK_company");
295 System.out.println("FK_company :: "+fk_company);
296
297
298 PersomilBean employee=new PersomilBean();
299 employee.setCf(cf);
300 employee.setGrado(grado);
301 employee.setCategoria(categoria);
302 employee.setNome(nome);
303 employee.setCognome(cognome);
304 employee.setData_nascita(data_nascita);
305 employee.setLuogo_nascita(luogo_nascita);
306 employee.setCmd(cmd);
307 employee.setFk_company(fk_company);
308
309 ServletContext sc=request.getSession().getServletContext();
310 request.getSession().removeAttribute("EMPLOYEE");
311 request.getSession().setAttribute("EMPLOYEE", employee);
312
313 RequestDispatcher rd=sc.getRequestDispatcher("/formPersomil.jsp");
314 rd.forward(request, response);
315
316
317 //Aggiunta impiegato della compagnia
318
319 if(request.getSession()!=null &&
request.getSession().getAttribute("COMPANY") !=null){
320 company=(RepartoBean)request.getSession().getAttribute("COMPANY");
```

```
321 ArrayList<PersomilBean> companyEmployees=company.getRepartoEmployees();
322 companyEmployees.add(employee);
323 company.setRepartoEmployees(companyEmployees);
324 request.getSession().removeAttribute("COMPANY");
325 request.getSession().setAttribute("COMPANY", company);
326 }
327 company=(RepartoBean) request.getSession().getAttribute("COMPANY");
328 ArrayList<PersomilBean>CompanyEmployeeList=company.getRepartoEmployees();
329
330 for(PersomilBean i:CompanyEmployeeList){
331 System.out.println("Cognome :: "+i.getCognome()+"Nome ::
"+i.getNome());
332 }
333
334
335 PrintWriter out=response.getWriter();
336 out.println("<html><body>Inserimento impiegato di Post employee avvenuto
con successo</body></html>");
337
338 }
339
340 }
341 }
342
343
344
```

Una volta che si sono prelevati i valori dal form e' necessario gestirli per poterli collegare tra loro ed in ultimo salvarli sul database. Per effettuare tale operazione nell'ottica di separazione dei livelli secondo il **pattern MVC** occorre creare un oggetto che possa essere passato ad una classe di salvataggio (indipendentemente dalla modalità pratica di salvataggio sul database) e che possa incapsulare più oggetti in un singolo oggetto al fine di trasferirlo e serializzarlo su un differente server. Per tale esigenza abbiamo creato ed istanziato un oggetto (JavaBean) che conserva lo stato delle sue variabili per il tempo di utilizzo. Praticamente la classe avrà i metodi get e set delle variabili (**realizzazione del principio dell'incapsulamento**). Di fondamentale importanza, a livello di classe, e' legare una posizione tabellare ad uno o più militari. A tale scopo nel bean del Reparto si è individuato un ulteriore campo che consenta di identificare i militari. Tale campo è ottenuto definendo un ArrayList di oggetti militari o di tipo Persomil come mostrato di seguito:

```
ArrayList<PersomilBean>repartoEmployees;
```

Il codice riportato tra le parentesi angolari(<>) rappresenta i cosiddetti "generics" che permettono di definire un tipo parametrizzato e che viene esplicitato successivamente in fase di compilazione.

Si riportano qui di seguito il codice delle classi Bean in questione:

RepartoBean.java

```
1 /**
4 package progettoAT.bean;
56
```

```java
import java.io.Serializable;
89
/**
10 * @author sasaman
11 *
12 */
13 public class RepartoBean implements Serializable {
14
15
16 private static final long serialVersionUID = 4535572149583253018L;
17
18
19 private String idcompany;
20 private String nome_reparto;
21 private String ufficio;
22 private String sezione;
23 private String incarico;
24 private String email_incarico;
25 private String cell_svz;
26 private String ruolo_protocollo_inf;
27
28 private ArrayList<PersomilBean>repartoEmployees;
29
30
31 public String getIdcompany() {
32 return idcompany;
33 }
34 public void setIdcompany(String idcompany) {
35 this.idcompany = idcompany;
36 }
37
38
39
40 public String getNome_reparto() {
41 return nome_reparto;
42 }
43 public void setNome_reparto(String nome_reparto) {
44 this.nome_reparto = nome_reparto;
45 }
46
47
48
49 public String getUfficio() {
50 return ufficio;
51 }
52 public void setUfficio(String ufficio) {
53 this.ufficio = ufficio;
54 }
55
56
57
58 public String getSezione() {
59 return sezione;
60 }
61 public void setSezione(String sezione) {
62 this.sezione = sezione;
63 }
64
65
66
```

```
67 public String getIncarico() {
68 return incarico;
69 }
70 public void setIncarico(String incarico) {
71 this.incarico = incarico;
72 }
73
74
75
76 public String getEmail_incarico() {
77 return email_incarico;
78 }
79 public void setEmail_incarico(String email_incarico) {
80 this.email_incarico = email_incarico;
81 }
82
83
84
85 public String getCell_svz() {
86 return cell_svz;
87 }
88 public void setCell_svz(String cell_svz) {
89 this.cell_svz = cell_svz;
90 }
91
92
93
94
95 public String getRuolo_protocollo_inf() {
96 return ruolo_protocollo_inf;
97 }
98 public void setRuolo_protocollo_inf(String ruolo_protocollo_inf) {
99 this.ruolo_protocollo_inf = ruolo_protocollo_inf;
100 }
101
102 public ArrayList<PersomilBean> getRepartoEmployees() {
103 return repartoEmployees;
104 }
105 public void setRepartoEmployees(ArrayList<PersomilBean> repartoEmployees) {
106 this.repartoEmployees = repartoEmployees;
107 }
108}
```

Per restituire un oggetto dalla Servlet alla pagina JSP è necessario utilizzare un altro oggetto. In tale ipotesi abbiamo usato l'oggetto dispatch in quanto si presuppone che il server indica al Browser quale particolare JSP deve essere utilizzata e gli eventuali parametri da passare quali quelli in sessione o in request. L'utente finale, in tale ipotesi, non sa che la risposta venga renderizzata da una pagina JSP e il Browser gestisce la richiesta in maniera standard come mostra la seguente sintassi:

```
ServletContext sc=request.getSession().getServletContext();
RequestDispatcher rd=sc.getRequestDispatcher("/formReparto.jsp");
rd.forward(request, response);
```

Quindi si preleva il context della Servlet per poi effettuare il dispatching alla pagina JSP desiderata. Dopo aver collegato i due oggetti è necessario mantenere lo stato dei

due Bean. Questo è possibile sfruttando uno dei due strumenti request o session. Tra request o session cambia semplicemente lo scope degli attributi. La request viene annullata, ovvero si perdono i dati passati e momentaneamente salvati all'interno di un Bean appena la response è stata committata al client mentre la session termina con la chiusura della sessione(chiusura del Browsers). La request comporta la possibilità di perdere i dati nel caso in cui si lavori con sovrastrutture complesse per passare l'informazione attraverso differenti strati di software e quindi di classi. Di fondamentale importanza è sapere che quando si crea una nuova sessione il server predispone un'aria di memoria che conterrà l'informazione e marca il Browser dell'utente in modo da poter successivamente associare ogni richiesta di quest'ultimo con quella determinata sezione. In sostanza, viene registrato un cookie sul Browser con un numero di identificazione della sessione. L'informazione risiede sul server. Nelle pagine JSP troviamo "i session objects" mentre nella servlet sono richiamati tramite il metodo getSession(). Quindi è possibile richiamare la sessione lato server cioè nella Servlet con il seguente codice:

HttpSession session=request. getSession();

dove il request serve per ottenere il riferimento ad un oggetto di tipo HttpSession. Attraverso il metodo getSession() è restituito l'oggetto di tipo HttpSession relativo alla sessione. La Servlet , quindi, identifica il client da cui viene chiamata e se esiste un oggetto HttpSession per quel client session, precedentemente istanziato,e quindi farà riferimento a quello, altrimenti un nuovo oggetto session viene creato. Questo meccanismo permette il mantenimento dello stato attraverso la richiesta di più pagine. Situazione che non si genererebbe nel caso di semplice request. Così si è proceduto a riempire i Bean del progetto e settarli in sessione, per poter salvare un unico oggetto prima di inserirlo nel database. Quindi si è gestito l'oggetto reparto, si è settato il corrispondente Bean, lo si è messo in sessione per rimandarlo alla pagina JSP. In accordo **con il principio di progettazione del pattern MVC** abbiamo creato un dialogo tra controller e view mentre il salvataggio sul database rappresenta la comunicazione con il model. In tale situazione quando viene re-inviato alla pagina JSP, il Bean salvato in sessione viene prelevato dalla sessione valorizzando i campi. La pagina JSP all'inizio presenta un Bean vuoto e quindi il value dei parametri viene posto a vuoto. Nella Servlet viene valorizzato il Bean del Reparto e messo in sessione con l'attributo "Company". Si parte dal form formReparto.jsp vuoto, vengono valorizzati i valori e successivamente si ritorna alla pagina formReparto.jsp con i campi valorizzati con gli input immessi. La Servlet gestisce i vari campi, li inserisce in un Bean e li rimanda in sessione alla pagina JSP tramite l'oggetto dispatcher. Questa situazione diviene di fondamentale importanza nel caso di modifica dei dati inseriti. Quando si ritorna alla pagina JSP, si può notare che il campo id_company non è più modificabile ,in quanto **chiave** per il reparto e **campo chiave esterno** per il

militare(principio del vincolo di integrità referenziale). Così si attiva il collegamento per accedere alla pagina di inserimento del militare .Tale pagina e' gestita, anch'essa, con le sessioni. Dopo aver inserito il militare l'identificativo id_employee non è più modificabile e compare il collegamento per l'inserimento di ulteriori militari. Infine si è creato il collegamento per eliminare e per inserire il personale militare. Si precisa che per annullare gli oggetti in sessione, basta cliccare su Home Page, ritornando nella JSP corrispondente all'inserimento di una nuova posizione tabellare organica come riportato nel codice della Servlet. Il collegamento per il salvataggio dei dati è stato fatto mediante la creazione della DAO (DATA ACCESS OBJECT) SaveMySQL.java, gestendo il CRUD con il database e garantendo i requisiti di persistenza e ROLL-BACK().

Sommario

Bibliografia

Paolo Atzeni	Basi di Dati. Modelli e linguaggi di interrogazione	McGraw-Hill
B.Fadini, C. Savy	Fondamenti di Informatica	Liguori, 1991
Roger S. Pressman	Principi di Ingegneria del Software	McGraw-Hill, 2000
Simon Pietro Romano	Dispense del Corso di Applicazioni Telematiche	Anno 2011/2012