UNIVERSITA' DEGLI STUDI DI MODENA E REGGIO EMILIA
FACOLTA' DI INGEGNERIA DI MODENA

CORSO DI LAUREA SPECIALISTICA IN INGEGNERIA INFORMATICA

Progetto e Realizzazione di uno strumento di installazione automatica di applicazioni Web su piattaforma WebSphere

RELATORE:
CHIAR.MO PROF. SONIA BERGAMASCHI

CANDIDATO:
FILIPPO BATTILANI

ANNO ACCADEMICO 2006-2007

Ringraziamenti

Un particolare ringraziamento va alla Professoressa Sonia Bergamaschi, relatore di questa tesi, per i preziosi suggerimenti forniti e la collaborazione per la stesura del lavoro.

Desidero inoltre ringraziare i colleghi di lavoro della Società Cooperativa Banca Popolare dell'Emilia Romagna, per la disponibilità dimostrata durante la realizzazione del progetto.

Infine, desidero ringraziare tutte le persone care per la fiducia, il supporto e l'affetto profusi durante il corso degli studi.

INDICE

INTRODUZIONE

Oggetto della presente tesi è il lavoro realizzato presso la Società Cooperativa Banca Popolare dell'Emilia Romagna di Modena, Capogruppo dell'omonimo Gruppo Bancario, e consiste nella progettazione e realizzazione di uno strumento di installazione automatica di applicazioni Web su piattaforma WebSphere.

La crescente importanza che le applicazioni Web stanno rivestendo nel sistema informativo aziendale, infatti, ha fatto emergere la necessità di adottare uno strumento per il versioning e l'installazione automatizzata delle stesse, come già avviene per i programmi dell'ambiente legacy.

L'obiettivo che si intende pertanto conseguire consiste nella realizzazione di uno strumento centralizzato in grado di standardizzare ed automatizzare la gestione del Ciclo di Vita delle applicazioni Web. L'azienda dispone di numerosi Application Server, però il progetto che si realizzerà sarà principalmente dedicato alla piattaforma WebSphere per z/OS versione 5.1, pur avendo validità generale nelle linee guida, in quanto su di essa risiedono alcune delle applicazioni più critiche per l'azienda, e perché permette di toccare numerosi aspetti che rendono il progetto completo.

Per essere vincente uno strumento di questo tipo deve inserirsi armoniosamente all'interno del sistema informativo aziendale, ovvero deve integrarsi con gli strumenti di lavoro già esistenti, e deve essere semplice da utilizzare per essere fruibile anche da parte di un utente che non conosce nel dettaglio l'ambiente su cui è installata l'applicazione.

La tesi è composta da quattro capitoli. Il primo presenta un'analisi dettagliata delle attuali metodologie di lavoro aziendali indicando i compiti degli uffici coinvolti, e descrive la configurazione dell'Application Server WebSphere in azienda.

Il secondo capitolo, invece, introduce la fase di analisi e progettazione illustrando le caratteristiche del software per il versioning Serena ChangeMan DS, utilizzato come base del progetto, e gli strumenti adottati per realizzare gli automatismi. In questo capitolo viene anche descritta la struttura dati creata per realizzare l'interfaccia Web semplificata per gli utenti.

Il terzo descrive dettagliatamente l'automatismo realizzato per l'installazione delle applicazioni Web su piattaforma WebSphere.

L'ultimo capitolo, il quarto, è dedicato alla descrizione della fase di sviluppo

dell'interfaccia Web semplificata.

Seguono le conclusioni, la bibliografia, e due appendici, la prima descrive come il Portale aziendale chiama le applicazioni che lancia, l'altra mostra la Shell Unix realizzata per generare in modo dinamico gli script di installazione delle applicazioni Web su piattaforma WebSphere.

LA SEDE DI REALIZZAZIONE DEL PROGETTO

La storia del Gruppo Banca popolare dell'Emilia Romagna

La Banca popolare dell'Emilia Romagna trae le sue origini dall'Istituto fondato nel 1867 sotto la denominazione di Banca Popolare di Modena, società cooperativa a responsabilità limitata, ha accresciuto nel tempo la propria dimensione attraverso l'incorporazione di altre banche popolari della regione. A seguito della fusione con la Banca Cooperativa di Bologna (1983), la Banca ha cambiato la denominazione in "Banca Popolare dell'Emilia", ulteriormente modificata nell'attuale ragione sociale a seguito dell'incorporazione, avvenuta nel 1992, della Banca Popolare di Cesena.

Dal 1993 in poi il processo di sviluppo della Banca si è attuato, principalmente, attraverso operazioni di aggregazione e/o acquisizione di altre realtà creditizie, in ogni parte del Paese. Alcune di esse presentavano, al momento dell'acquisizione, problematiche gestionali e reddituali. Tutte sono state completamente risanate ed hanno intrapreso un percorso virtuoso di crescita, assumendo dimensioni di rilievo nelle singole realtà territoriali servite.

L'esperienza maturata da Banca popolare dell'Emilia Romagna e gli eccellenti risultati delle operazioni di aggregazione del passato sono il frutto dello spirito che anima il Gruppo, come insieme di partners **impegnati in un comune progetto di crescita**, che crei valore per ciascuno degli associati (istituzione e suoi soci privati) e per la Capogruppo, operanti in sinergia e stretta **collaborazione** tra di loro.

Al 30 Settembre 2005 il Gruppo risulta costituito, oltre che dalla Capogruppo, da 13 banche.

Banca C.R.V. - Cassa di Risparmio di Vignola S.p.A.
Banca del Monte di Foggia S.p.A.
Banca di Sassari S.p.A.
Banca popolare del Materano S.p.A.
Banca popolare dell'Emilia Romagna (Europe) International s.a.
Banca popolare di Aprilia S.p.A.
Banca popolare di Crotone S.p.A.
Banca popolare di Lanciano e Sulmona S.p.A.

Banca popolare di Ravenna S.p.A.

Banca della Campania S.p.A.

Banco di Sardegna S.p.A.

Cassa di Risparmio della Provincia dell'Aquila S.p.A.

Eurobanca del Trentino S.p.A.

ATTIVITA' SVOLTE

Prima di descrivere le attività svolte, si evidenzia attraverso il seguente diagramma il percorso seguito per la realizzazione del progetto.

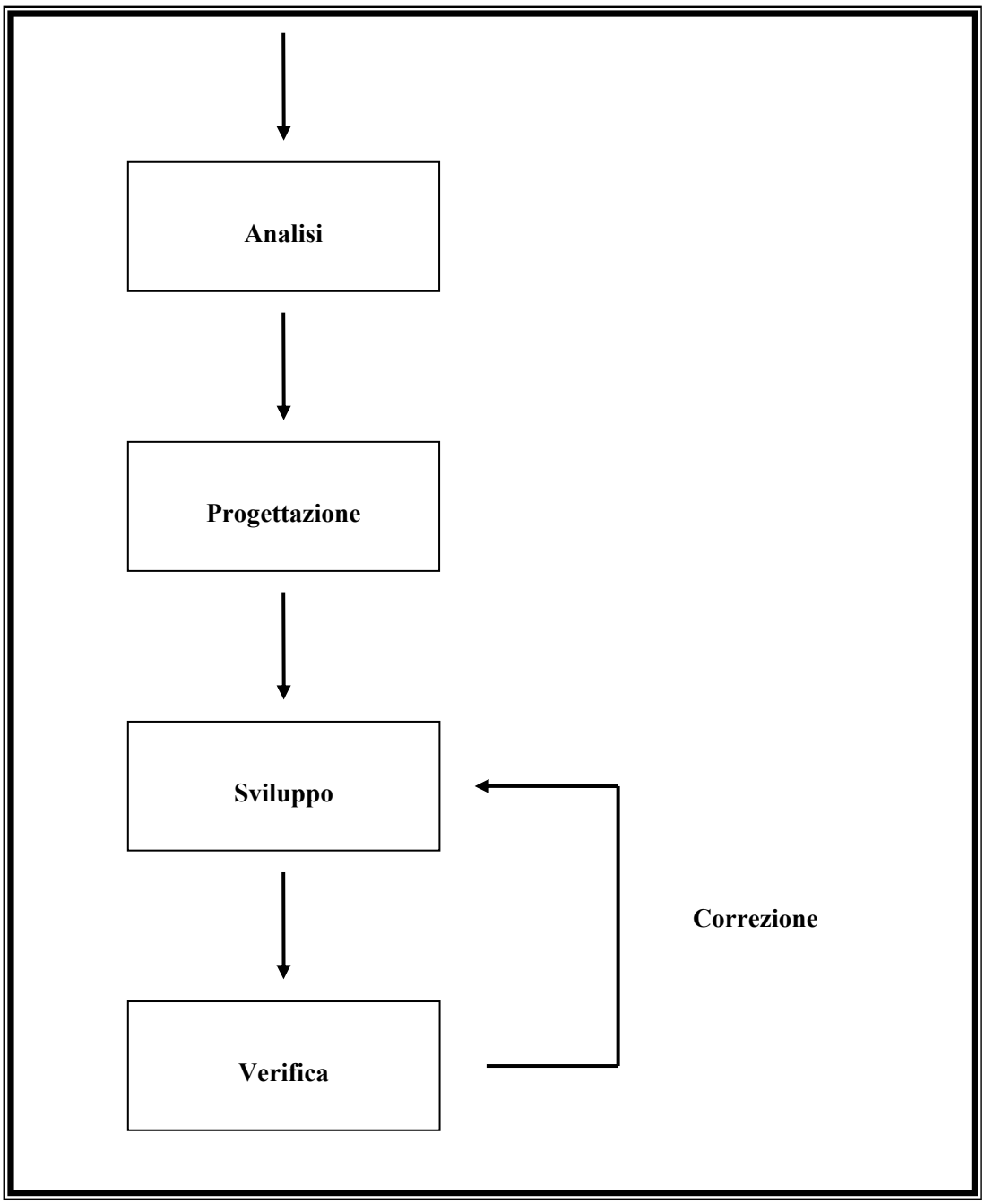


Figura 1

Le attività svolte nel corso della realizzazione del progetto sono le seguenti:

1) **Analisi:**

- analisi delle metodologie di lavoro aziendali per lo sviluppo e l'installazione delle applicazioni Web;

- approfondimento delle problematiche che il progetto dovrà risolvere;

- scelta del progetto "pilota" per la modellazione dei Cicli di Vita delle applicazioni Web.

2) **Progettazione:**

- studio dell'Application Server IBM WebSphere per z/OS versione 5.1 e della console WSADMIN;

- analisi delle caratteristiche del Software per il versioning Serena ChangeMan DS di cui l'azienda dispone;

- modellazione del processo automatizzato per la gestione del Ciclo di Vita delle applicazioni Web residenti sull'Application Server IBM WebSphere;

- suddivisione del progetto in più parti distinte;

- progettazione di un'interfaccia Web utente semplificata;

- analisi delle caratteristiche che presentano le architetture di sistema e gli ambienti di sviluppo esistenti oggi sul mercato;

- scelta delle entità, dei relativi attributi e delle associazioni da inserire nel database, necessari per memorizzare e gestire correttamente le informazioni che saranno utilizzate dall'interfaccia Web;

3) **Sviluppo:**

- configurazione e modellazione del Software ChangeMan DS per la realizzazione del

progetto aziendale;

- creazione del processo necessario per automatizzare la generazione ed il BIND dei DBRM sul DBMS IBM DB2, nel caso fosse previsto un accesso ai dati tramite SQL statico;

- realizzazione di un automatismo per la generazione di script di installazione automatizzata delle applicazioni sull'Application Server WebSphere;

- integrazione all'interno di ChangeMan DS dei processi realizzati;

- realizzazione dei piani di installazione ed eventuale BIND delle applicazioni, in base a quanto generato dagli automatismi sopra citati;

- creazione dello schema Entity Relationship (E/R) del database necessario per la realizzazione dell'interfaccia Web semplificata;

- definizione delle tabelle e dei campi del database;

- definizione della struttura dell'interfaccia Web ed integrazione della stessa nel Portale aziendale;

- realizzazione dell'interfaccia Web semplificata;

- verifica del corretto funzionamento del progetto;

- correzione delle anomalie ed inserimento delle particolarità mancanti.

CAPITOLO 1 SITUAZIONE ATTUALE

1.1 INSTALLAZIONE APPLICAZIONI WEB: METODOLOGIE DI LAVORO

L'organizzazione del Centro di Elaborazione Dati presente oggi in azienda evidenzia alcuni aspetti che si intendono migliorare attraverso il nuovo progetto che si realizzerà. Per rendere più chiari i bisogni e le prospettive verranno quindi descritte in questo capitolo le comuni metodologie di lavoro utilizzate per l'installazione delle applicazione Web, in particolare sull'application server di IBM WebSphere per z/OS versione 5.1, che fornisce compatibilità completa con lo standard J2EE, e che in seguito chiameremo WAS5 per semplicità.

L'azienda dispone di numerosi Application Server, però il progetto che si realizzerà sarà principalmente dedicato a WAS5, pur avendo validità generale nelle linee guida, in quanto su di esso risiedono alcune delle applicazioni più critiche per l'azienda, e perché permette di toccare numerosi aspetti che rendono il progetto completo.

In seguito si descriverà la configurazione di WAS5 in azienda, per poi indicare i compiti attribuiti ai due uffici che saranno coinvolti in fase di realizzazione del progetto.

1.1.1 IBM WebSphere Application Server per z/OS versione 5.1: configurazione aziendale

Prima di iniziare a descrivere i compiti degli uffici interessati si ritiene importante effettuare un richiamo relativamente alla configurazione in azienda del Mainframe, e di conseguenza di WAS5.

Per quanto riguarda l'ambiente di test è presente una sola partizione del Mainframe, mentre per l'ambiente di produzione le partizioni sono tre, sulle quali sono distribuiti gli stessi servizi per diverse banche del gruppo.
Per ogni partizione è installata una istanza di WAS5, e in ogni istanza devono essere presenti le medesime applicazioni allineate come versione.

WAS5 consente anche una configurazione multi-server su ogni istanza, ovvero permette di creare più server configurabili distintamente in base alle applicazioni che vi sono deployate. Tale soluzione ha il vantaggio di poter isolare le applicazioni più critiche su server dedicati, infatti è stata adottata in azienda.

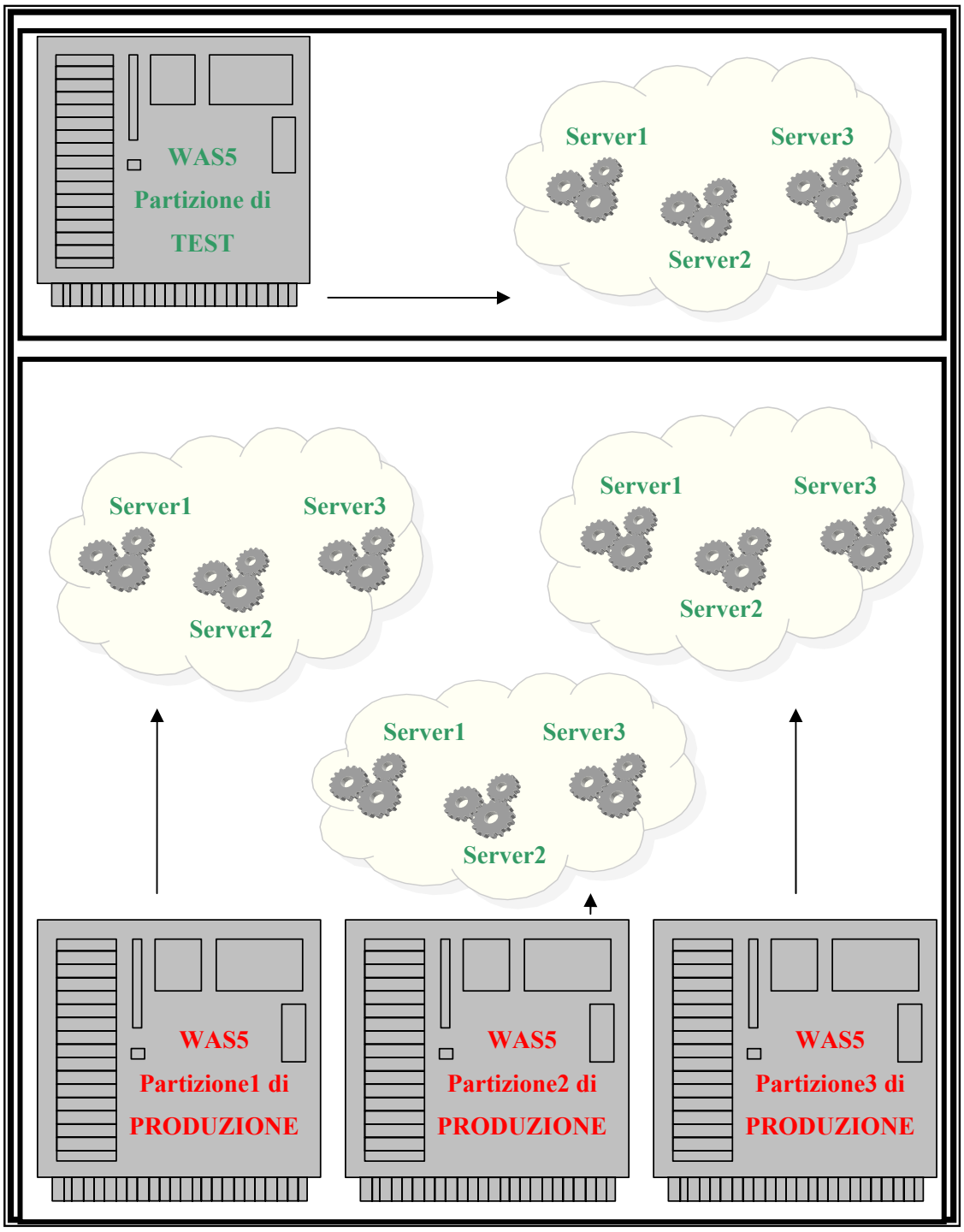


Figura 2

1.1.2 Ufficio Analisi e Pianificazione

L'ufficio Analisi e Pianificazione ha i seguenti **compiti** per ogni applicazione da installare su WAS5:

- sviluppo dell'applicazione se ciò avviene internamente, oppure gestione delle relazioni con il fornitori se sviluppata esternamente;

- installazione dell'applicazione tramite la console di amministrazione e, nel caso fosse previsto SQL statico, preparazione e BIND dei DBRM, sulla partizione dell'ambiente di TEST;

- test dell'applicazione;

- gestione delle versioni del software.

Le attività di installazione, preparazione e BIND dei DBRM, e gestione delle versioni del software sono svolte manualmente, senza l'ausilio di uno strumento che le possa automatizzare.

1.1.3 Ufficio Sistemisti

L'ufficio Sistemisti ha i seguenti **compiti**:

- configurazione del sistema per la nuova applicazione;

- installazione dell'applicazione e, nel caso fosse previsto SQL statico, preparazione e BIND dei DBRM, su tutte le partizioni dell'ambiente di produzione.

Anche nell'ufficio sistemisti le attività di installazione e preparazione e BIND dei DBRM sono svolte manualmente, senza l'ausilio di uno strumento che le possa automatizzare.

1.2 OBIETTIVO

L'obiettivo che l'azienda si prefigge di realizzare consiste nella creazione di uno strumento centralizzato capace di automatizzare le attività descritte in precedenza, ovvero di standardizzare la gestione del Ciclo di Vita delle applicazioni installate su WAS5.
In particolare lo strumento per essere vincente dovrebbe avere le seguenti funzionalità:

- gestione del versioning del software;

- installazione delle applicazioni e, nel caso fosse previsto SQL statico, preparazione e BIND dei DBRM, in modo automatizzato, sia in ambiente di test sia in ambiente di produzione;

- verifica dei file modificati tra rilasci successivi delle applicazioni J2EE da installare su WAS5, che sono impacchettate in file Enterprise Archive (EAR) o Web Archive (WAR);

con le seguenti caratteristiche:

- semplice da utilizzare, in modo tale da essere fruibile anche da parte di un utente che non conosce nel dettaglio l'ambiente su cui è installata l'applicazione;

- in grado di far rispettare le regole definite in fase di progettazione;

- totalmente web per l'utente finale;

- integrato nel sistema informativo esistente.

CAPITOLO 2 ANALISI E PROGETTAZIONE

2.1 SUDDIVISIONE DEL PROGETTO IN SOTTOPROGETTI

In seguito allo studio degli obiettivi da raggiungere per la realizzazione del progetto abbiamo ritenuto importante suddividere il progetto in sottoprogetti, ovvero in più parti che si concludono con il raggiungimento di un traguardo interno, in modo da poterne verificare l'andamento con risultati concreti e funzionanti in tempi relativamente brevi. In base alla struttura del progetto che stiamo realizzando e ad alcune considerazioni ho optato per considerare il progetto composto dalle seguenti parti o sottoprogetti, in cui ho collocato gli obiettivi che intendiamo raggiungere al loro interno:

Sottoprogetto	***Obiettivi***
I	- possibilità di automatizzare, tramite il software per il versioning di cui dispone l'azienda, l'installazione delle applicazioni Web su WebSphere per z/OS; - possibilità di automatizzare, tramite il software per il versioning di cui dispone l'azienda, la generazione ed il BIND su DB2 dei DBRM delle applicazioni Web da installare su WebSphere per z/OS, nel caso fosse previsto un accesso ai dati tramite SQL statico
II	- possibilità di effettuare le attività previste dal Sottoprogetto I tramite un'interfaccia Web semplificata integrata nel Portale aziendale

III	- possibilità di verificare le differenze tra due rilasci successivi della medesima applicazione Web, in termini di file modificati, aggiunti od eliminati

Tabella 1

Evidentemente la scelta di trattare inizialmente le parti più complicate risulta fondamentale, perché la struttura portante deve essere il punto di partenza: una piccola modifica alla struttura generale può causare la perdita di molte parti legate ad essa, se sono già state sviluppate.

2.2 DEFINIZIONE DEGLI STANDARD DA SEGUIRE

2.2.1 Cicli di Vita

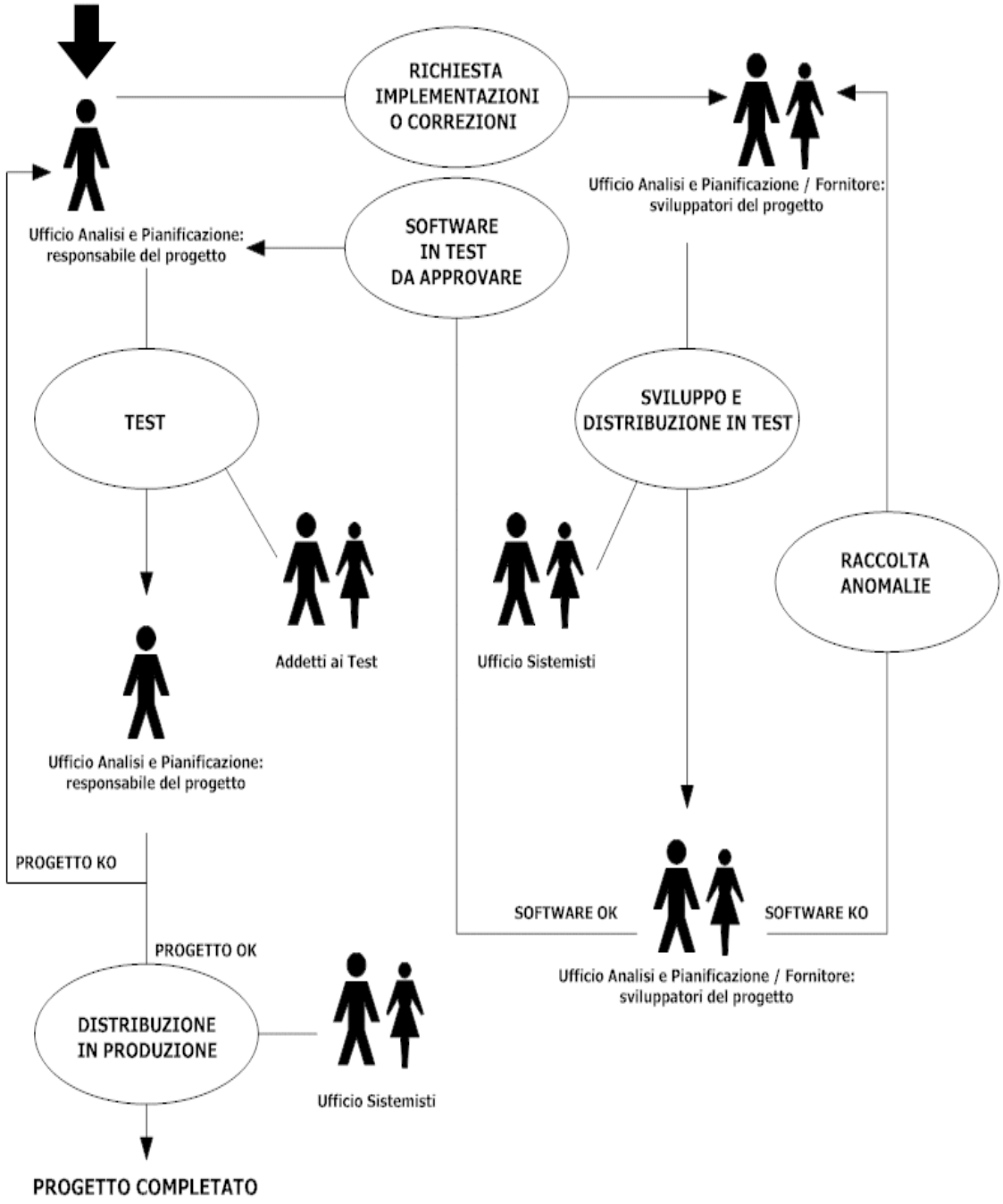


Figura 3

Descrizione del diagramma

Il responsabile del progetto, appartenente all'Ufficio Analisi e Pianificazione, richiede agli sviluppatori l'implementazione del software relativo ad un nuovo progetto. Gli sviluppatori, interni od esterni all'azienda, procedono con lo sviluppo e l'installazione del software realizzato in ambiente di test, con la collaborazione dell'Ufficio Sistemisti per la preparazione dell'ambiente. Se il software non funziona o non risponde alle aspettative degli sviluppatori vengono raccolte le anomalie per la risoluzione, altrimenti viene richiesta l'approvazione da parte del responsabile del progetto, il quale con la collaborazione degli addetti ai test stabilisce se il progetto può essere portato in produzione o meno. In caso affermativo si procede con la distribuzione del progetto in ambiente produzione accordandosi con l'Ufficio Sistemisti per le eventuali configurazioni di ambiente, altrimenti vengono richieste ulteriori implementazioni o correzioni agli sviluppatori.

2.3 SOFTWARE PER IL VERSIONING: SERENA CHANGEMAN DS

2.3.1 Descrizione

Il software di base sul quale è stato sviluppato l'intero progetto si chiama ChangeMan DS, un prodotto concepito per il versioning e la distribuzione delle applicazioni dipartimentali.
La scelta di ChangeMan DS rispetto ad altri prodotti è stata motivata anche dal fatto che consente di colloquiare con le partizioni Unix residenti su sistema operativo z/OS, ambiente in cui è installato l'Application Server WebSphere di cui abbiamo precedentemente parlato.
Le componenti principali di ChangeMan DS sono le seguenti:

Server: macchina di riferimento che pilota tutte le attività. Essa accede ad un database proprietario;

Database: fonte dati in cui sono storicizzate tutte le informazioni relative ai progetti censiti, in particolare le versioni e le corrispondenze tra nome logico e locazione fisica. L'utilizzo di ChangeMan DS prevede infatti che l'amministratore del sistema si interessi della locazione fisica di un progetto solo in fase di censimento, in seguito può utilizzare il nome logico;

Agent: demoni da installare sulle macchine di sviluppo, consegna o installazione di un progetto. Il Server può colloquiare soltanto con le macchine in cui è installato un Agent;

Client: deve essere installato sui PC degli utenti che utilizzano il Client nativo di ChangeMan DS per gestire i progetti.

2.3.2 Introduzione in azienda

In questa sede verranno descritte le scelte progettuali che saranno seguite nell'implementazione di ChangeMan DS per la gestione del Ciclo di Vita delle applicazioni installate su WebSphere per z/OS, per tarare il processo di Change Management in ambiente distribuito. Le considerazioni espresse avranno valore globale, esprimendo cioè regole da adottare nella gestione dell'intero patrimonio applicativo del mondo distribuito, fatte salve le eccezioni determinate dalle specifiche caratteristiche delle singole applicazioni.
Le considerazioni espresse riguardano l'approccio logico e organizzativo, mentre per i dettagli

delle singole implementazioni si rimanda alla documentazione specifica.

2.3.3 Definizione degli ambienti

Il Ciclo di Vita delle applicazioni verterà su quattro ambienti distinti, con il primo ambiente che avrà caratterizzazione differente a seconda che l'applicazione sia sviluppata internamente o meno. Nel corso di questo capitolo verranno dettagliate le caratteristiche dei quattro ambienti.

Sviluppo / Consegna

L'ambiente di sviluppo ha una differente caratterizzazione a seconda che le applicazioni siano o meno sviluppate internamente dal cliente. In caso di sviluppo interno, in questo ambiente vengono effettuate le modifiche (sulla stazione di lavoro degli utenti o su un server centralizzato) e anche una prima fase di test. In questo ambiente sono presenti sia i sorgenti sia gli eseguibili dell'applicazione.
Nel caso di sviluppo esterno, invece, questo ambiente viene più propriamente chiamato "Consegna" e rappresenta il punto di ingresso del software nel Ciclo di Vita. In questo secondo caso la collocazione dell'ambiente è sul Server di ChangeMan DS.

Test

Ambiente nel quale vengono effettuati i test funzionali e i test di accettazione. In questo ambiente transitano unicamente le componenti che servono al corretto funzionamento dell'applicazione (eseguibili, file di configurazione o file non soggetti a processi di compilazione).

Produzione

Rappresenta l'ambiente finale del Ciclo di Vita, quello in cui viene installato il software per l'effettivo utilizzo. In questo ambiente arriva il medesimo software che transita per l'ambiente di test.

Baseline

E' un ambiente aggiuntivo introdotto per le necessità di versioning del software. Risiede sul Server di ChangeMan DS e deve essere posto sotto policy di backup.

2.3.4 Definizione del ciclo di vita

Nella definizione del Ciclo di Vita si è cercato di avere un Ciclo di Vita uniforme sia per le applicazioni sviluppate internamente, sia per i pacchetti acquisiti esternamente. Tenuto conto di diversi fattori, tra cui l'integrazione con i tools di sviluppo e le caratteristiche delle aree di ChangeMan DS, si è deciso di adottare un Ciclo di Vita come quello rappresentato in figura.

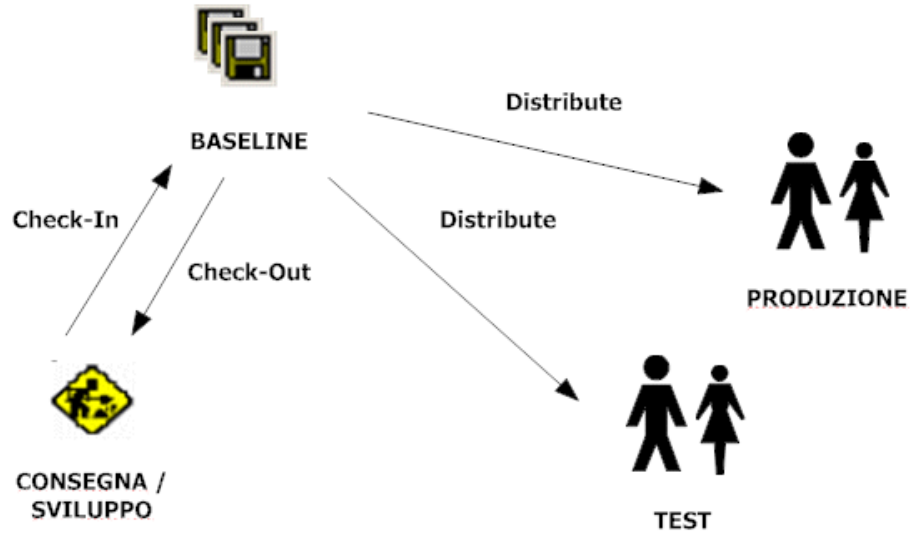


Figura 4

Gli ambienti di Sviluppo / Consegna sono legati alle Baseline tramite operazioni di Check-Out Check-In. Questo implica che la Baseline diventa il repository degli oggetti consolidati dallo sviluppo e non più il repository degli oggetti pronti per essere distribuiti in Esercizio. Il poter effettuare Check-In direttamente dall'ambiente di Sviluppo è un requisito per poter utilizzare l'integrazione con gli strumenti di sviluppo e risolve il problema dei salvataggi (normalmente i PC degli utenti non sono sottoposti a backup). Dalla Baseline vengono alimentati gli ambienti di Test e Produzione. Prima di passare il software in test, dovrà essere fatto uno step di Freeze del progetto distribuito.

2.3.5 Implementazione del ciclo di vita in ChangeMan DS

Per l'implementazione del ciclo di vita delle applicazioni del mondo distribuito saranno modellati in ChangeMan DS gli ambienti precedentemente esaminati introducendo delle aree opportune, e saranno definiti dei progetti per la gestione degli oggetti che dovranno essere posti sotto controllo dello strumento.

Aree

A ciascun ambiente sarà fatta corrispondere un'area di ChangeMan DS: un'area rappresenta logicamente una determinata partizione di disco nel quale è collocato il software controllato da ChangeMan DS. A seconda delle caratteristiche di ciascun ambiente sarà ad esso associato un'area di una particolare tipologia (Development, Quality Assurance, End User).

Un'area di fondamentale importanza è quella di Baseline, nella quale sono presenti le versioni correnti dei file, e le versioni precedenti. Ciascuna area sarà associata ad un determinato server, ed è pertanto necessario individuare esattamente quali saranno quelli che interagiranno con ChangeMan DS:

- l'area di Baseline è, come per tutte le applicazioni, associata al server di ChangeMan DS e l'accesso al software potrà avvenire soltanto tramite ChangeMan DS stesso;

- le aree di Sviluppo risiederanno nelle macchine utilizzate dagli sviluppatori e corrisponderanno alle directory di lavoro nelle quali avverrà la creazione o la modifica dei file; la collocazione più opportuna di tali directory verrà individuata e riproposta in tutte le postazioni di lavoro;

Esclusa l'area di Baseline che è utilizzata per il versioning, si hanno a disposizione tre tipologie di aree per la mappatura degli ambienti sopra descritti. In virtù delle caratteristiche degli ambienti, la mappatura scelta è la seguente:

Sviluppo: development area;
Consegna: development area;
Test: end-user area;
Produzione: end-user area.

Ad ogni area è possibile associare uno script, che viene invocato tutte le volte che un file od un progetto di ChangeMan DS transita attraverso l'area stessa.

Progetti

All'interno di ChangeMan DS il software verrà organizzato per progetti, cioè secondo aggregazioni logiche che consentono di mantenere legati componenti logicamente connessi. In particolare i componenti sviluppati con strumenti che prevedono al loro interno la definizione di progetti (Visual Basic per esempio) verranno posti sotto il controllo di ChangeMan DS ereditandone la medesima organizzazione per progetti (un progetto Visual Basic diventa un progetto ChangeMan DS). Per le altre componenti software sarà invece individuata in accordo con il gruppo di sviluppo una suddivisione logica che verrà tradotta come gerarchia di progetti in ChangeMan DS.
A livello di progetti verranno definite le procedure di compilazione qualora le componenti applicative prevedano tale necessità.
A livello di progetto verrà attuata l'operazione di Freeze (congelamento); la Freeze di un progetto non è altro che un legame logico tra le varie versioni degli oggetti associati al progetto stesso.

2.3.6 Architettura aziendale

L'immagine sottostante raffigura la configurazione adottata in azienda e su cui si svilupperà il progetto.

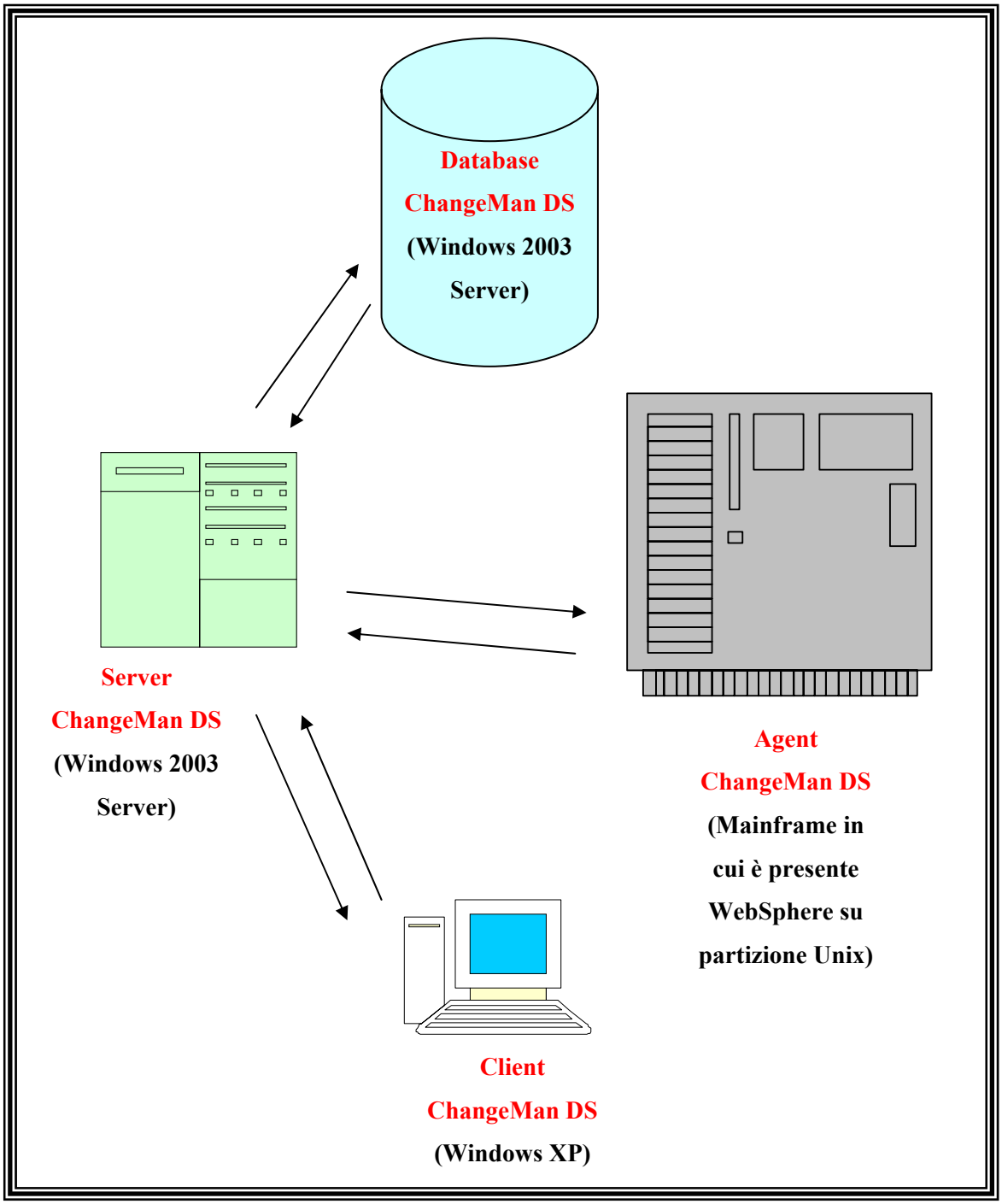


Figura 5

2.3.7 Metodi esposti

Serena ChangeMan DS prevede un'interfaccia a riga di comando ben documentata, che consente di effettuare tramite script le operazioni di gestione dei progetti.
Gli script in cui di possono inserire i comandi sopra citati sono batch DOS in ambiente Windows, e Shell in ambiente Unix. Nel nostro caso specifico sono stati utilizzati entrambi, in quanto sono stati coinvolti sia l'ambiente Windows in cui è installato il server di ChangeMan DS, sia l'ambiente Unix in cui è installato l'Application Server WebSphere.

2.4 LA CONSOLE WSADMIN

WSADMIN è un'interfaccia a riga di comando dell'Application Server WebSphere, disponibile a partire dalla versione 5, che consente l'automazione di numerose attività di amministrazione.
L'utilizzo di tale interfaccia è stata di fondamentale importanza per la realizzazione del progetto, in quanto ha consentito di automatizzare l'installazione delle applicazioni nei server di test e produzione.
Per il raggiungimento dei nostri obiettivi ci siamo soffermati sugli script di installazione, disinstallazione e su quelli per reperire le informazioni necessarie dagli EAR/WAR e dall'application server.

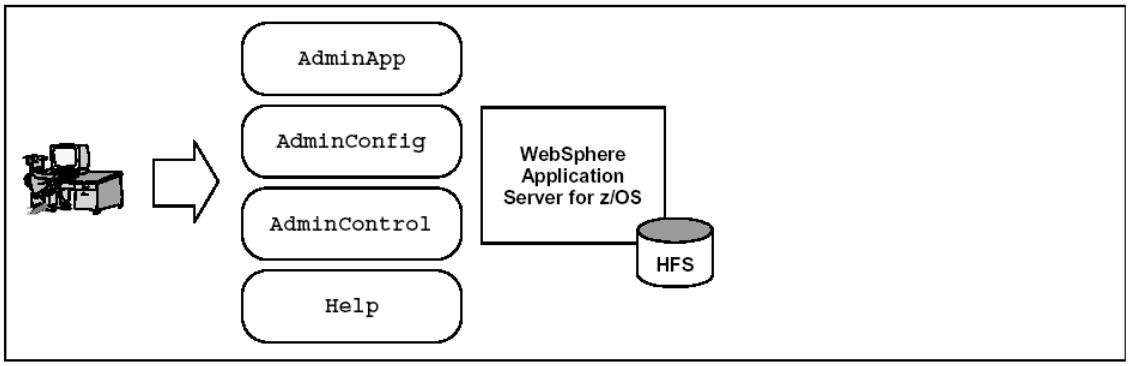


Figura 6

L'installazione di un EAR/WAR nell'application server richiede l'impostazione di numerose opzioni da parte dell'utente, di conseguenza anche l'installazione batch deve prevedere questi parametri, come indicato in seguito.

Esempio Shell di Installazione applicazione:

```
#Path WSADMIN
PATH=$PATH:/WebSphere/V5R1M0/AppServer/bin/

#Intestazione comando di installazione ed indicazione del nome fisico e logico dell'applicazione
wsadmin.sh -conntpe SOAP -host localhost -c '$AdminApp install /serena/produzione/ear/gestioneContante.war {-appname gestioneContante_war \

#Mappatura Host Virtuale
-MapWebModToVH  { \
{ "gestioneContante" \
"gestioneContante.war,WEB-INF/web.xml" \
"default_host" \
} \
} \

#Mappatura DataSource
-MapResRefToEJB  { \
{ "gestioneContante" \
"" \
"gestioneContante.war,WEB-INF/web.xml" \
"jdbc/gecods" \
"javax.sql.DataSource" \
"jdbc/BPERds" \
} \
} \

#Chiusura comando di installazione ed indicazione del server, della cella e del contesto in cui installare l'applicazione
-server serve11 -node OSN1 -cell OSN1Network -contextroot "/gestioneContante"};$AdminConfig save'
```

Esempio Shell di Disinstallazione applicazione:

```
#Path WSADMIN
PATH=$PATH:/WebSphere/V5R1M0/AppServer/bin/

#Verifica se l'applicazione è installata, in caso affermativo
disinstalla
APPL=gestioneContante_war
GIA_INSTALL=`wsadmin.sh -c '$AdminApp list' ! grep -x
gestioneContante_war`
if test -z ${GIA_INSTALL}
then
else
if [  ${APPL}  -eq  ${GIA_INSTALL} ]
then
#Comando di disinstallazione dell'applicazione
wsadmin.sh -conntpe SOAP -host localhost -c '$AdminApp uninstall
gestioneContante_war;$AdminConfig save'
fi
fi
```

Esempio Shell di Start applicazione:

```
#Path WSADMIN
PATH=$PATH:/WebSphere/V5R1M0/AppServer/bin/

#Comando di start dell'applicazione
wsadmin.sh -conntpe SOAP -host localhost -c 'set appManager
[$AdminControl queryNames
cell=OSN1Network,node=OSN1,type=ApplicationManager,process=serve
11,*];$AdminControl invoke $appManager startApplication
gestioneContante_war'
```

Esempio Shell di Stop applicazione:

```
#Path WSADMIN
PATH=$PATH:/WebSphere/V5R1M0/AppServer/bin/

#Verifica se l'applicazione è installata, in caso affermativo
esegue lo stop
APPL=gestioneContante_war
GIA_INSTALL=`wsadmin.sh -c '$AdminApp list' ! grep -x
gestioneContante_war`
if test -z ${GIA_INSTALL}
then
else
if [  ${APPL}  -eq  ${GIA_INSTALL} ]
then
#Comando di stop dell'applicazione
wsadmin.sh -conntpe SOAP -host localhost -c 'set appManager
[$AdminControl queryNames
cell=OSN1Network,node=OSN1,type=ApplicationManager,process=serve
11,*];$AdminControl invoke $appManager stopApplication
gestioneContante_war'
fi
fi
```

2.5 INTERFACCIA WEB

In questa fase sono effettuate le scelte progettuali relative all'interfaccia Web da rendere disponibile all'utente finale. L'interfaccia, che in seguito chiameremo CMDS Web, deve essere sviluppata con l'obiettivo di integrarsi nel sistema di sicurezza e profilatura aziendale, e soprattutto per semplificare la gestione del Ciclo di Vita degli applicativi Web censiti su ChangeMan DS, nascondendo i dettagli tecnici dell'infrastruttura realizzata a tal fine.
In particolare CMDS Web dovrà svincolare l'utente finale da ogni strumento client, e consentire l'inserimento dei nuovi pacchetti applicativi nell'apposita Area di ChangeMan DS tramite semplici UpLoad Ftp.
CMDS Web dovrà scatenare eventi attraverso chiamate a ChangeMan DS, e reperirà le informazioni necessarie attraverso un database che realizzeremo. Tengo a precisare che il database conterrà informazioni aggiuntive rispetto a quello di ChangeMan DS, pertanto non provocherà alcuna ridondanza di dati. Il database dovrà contenere le informazioni relative ai progetti e loro Ciclo di Vita, in base all'ambiente ed alla tipologia, ed agli utenti ai quali l'amministratore deciderà di concedere i privilegi di accesso.

2.5.1 Raggruppamento dei dati in categorie logiche

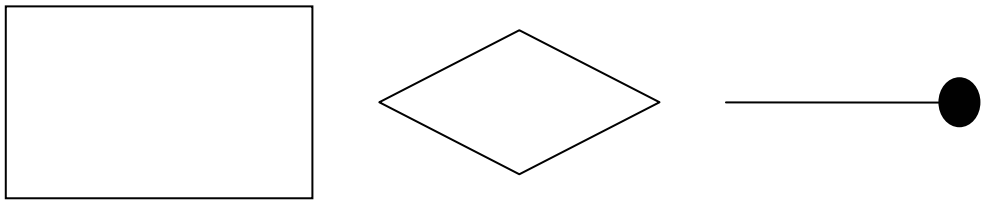
Giunti a questo punto, risulta necessario analizzare i dati che si hanno a disposizione e raggrupparli logicamente in tabelle. Per affrontare questa importante fase della progettazione sono stati studiati con cura i documenti precedentemente illustrati cercando di organizzare nel modo ottimale le informazioni che contengono ed eliminando le ridondanze di dati ma, soprattutto, si è cercato di tramutare in una grande raccolta di dati le numerose necessità elencate nel primo capitolo, tenendo come punto di riferimento i concetti fondamentali ed il modello di sviluppo appena descritti.

2.5.1.1 Progettazione concettuale

La progettazione concettuale permette tramite due componenti fondamentali di rappresentare la realtà di un problema. Questi due componenti sono il modello e lo schema. Il modello è l'insieme di regole e strutture che permettono la rappresentazione della realtà d'interesse, mentre lo schema è la rappresentazione di una specifica realtà secondo un determinato modello.

Il modello utilizzato nella programmazione concettuale del progetto che si sta realizzando è quello “Entity Relationship” sviluppato da Chen nel 1976 e successivamente esteso con altre primitive di rappresentazione.

Gli elementi di base di questo modello sono tre: l’entità, l’associazione, l’attributo. L’entità rappresenta un insieme di oggetti della realtà di cui si individuano proprietà comuni, l’associazione rappresenta un legame logico tra due o più relazioni, l’attributo rappresenta proprietà elementari di entità o associazioni.



Entità **Associazione** **Attributo**

Figura 7

2.5.1.2 Schema Entity Relationship

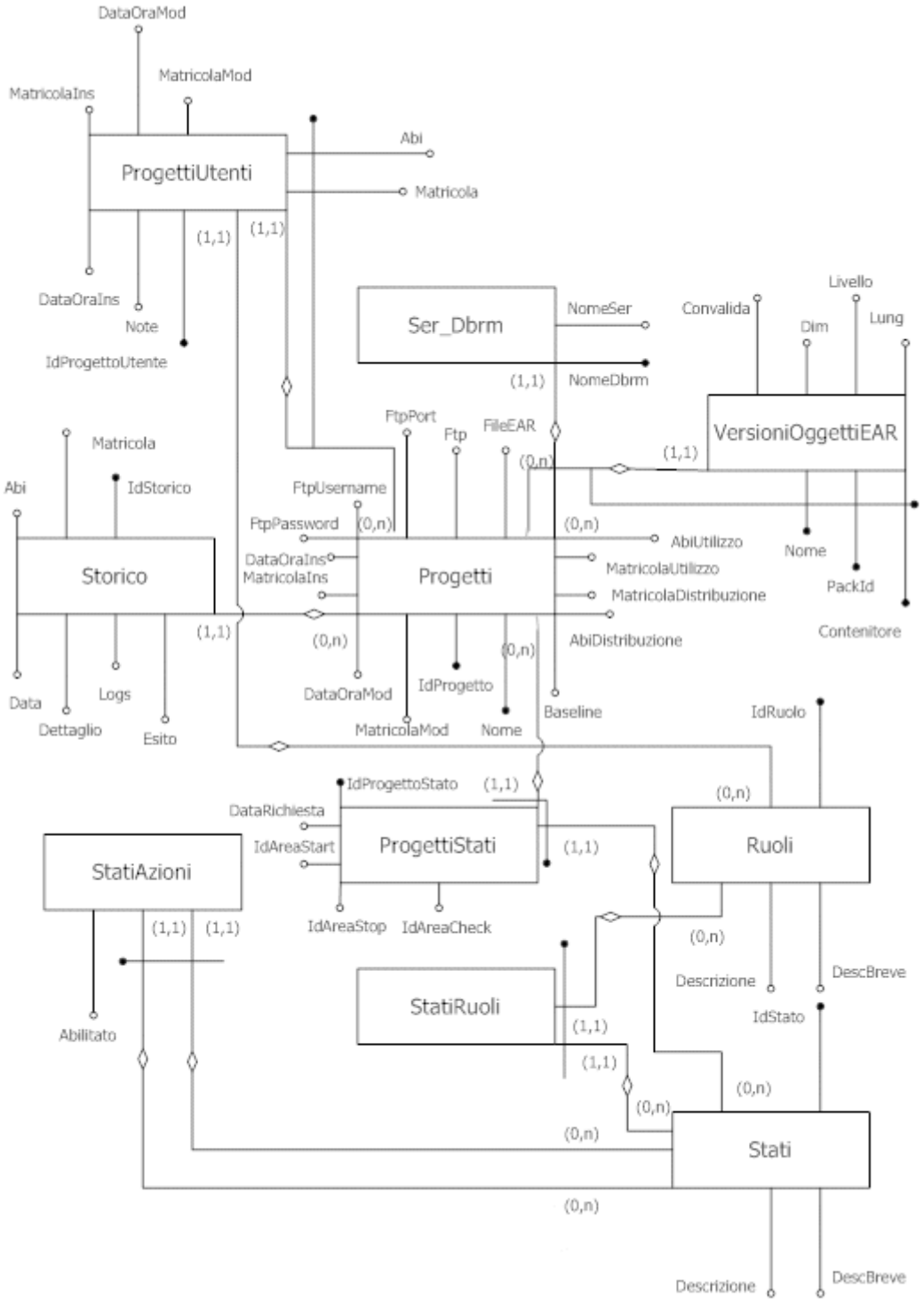


Figura 8

2.5.1.3 Definizione delle tabelle e dimensionamento sei campi

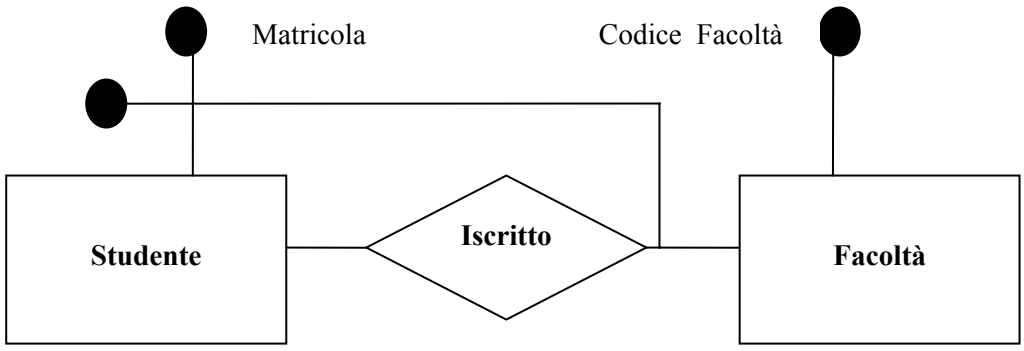
Il riferimento utilizzato per effettuare questo passo è la teoria relazionale che rappresenta lo standard di progettazione e creazione di database relazionali.

La teoria relazionale dei dati è stata introdotta da Codd nel 1970 ed è basata sul concetto matematico di relazione. Tale teoria prevede che ogni archivio sia composto da tabelle suddivise, a loro volta, in campi.

Ogni tabella deve possedere un identificatore principale (chiave primaria) che ne identifica in modo univoco tutti gli elementi.

Gli elementi contenuti nelle tabelle non solo possono essere identificati univocamente dalla chiave primaria, ma non devono dipendere da campi secondari (terza forma normale). I campi di tabelle diverse possono essere collegati, rispettando però i vincoli di integrità referenziale.

Ecco un esempio di traduzione da schema E/R a schema relazionale:



- **Studente** (matricola, codice_facoltà)

FK: codice_facoltà REFERENCES facoltà

- **Facoltà** (codice_facoltà)

N.B. I campi sottolineati fanno parte della chiave primaria della tabella.

Prima di costruire le tabelle occorre anche impostare il dimensionamento dei campi. Le notazioni utilizzate per descrivere il tipo di dati dei vari campi sono le seguenti:

Intero (Int)	Tipo numerico intero a 32 bit
Intero (BigInt)	Tipo numerico intero a 64 bit
Alfanumerico (Varchar)	Tipo per lettere e cifre
Data (DateTime)	Tipo per rappresentare date
Memo (Text)	Tipo alfanumerico con lunghezza non definita

Il dimensionamento dei campi alfanumerici è stato effettuato controllando le dimensioni massime dei valori trattati, con un aumento del 20% per avere maggior sicurezza.

In seguito sono riportate le tabelle che compongono la base dati. Si ricorda che un carattere *Alfanumerico* è codificato a 16 bit (2 byte), che il tipo Data occupa 32 bit e non è possibile quantificare lo spazio richiesto dal tipo *Memo*.
L'ultima colonna, invece, indica la categoria del campo. La dicitura *Ammetti Null* indica che il valore non deve essere inserito obbligatoriamente nel campo all'atto della creazione di un nuovo record, questo perché si tratta di un dato indispensabile per la struttura del database o, semplicemente, per le necessità di che dovrà utilizzare il software. Ovviamente i campi non obbligatori sono descrittivi o comunque di importanza secondaria.

Tabella Ruoli

Nome colonna	Tipo di dati	Lunghezza	Ammetti Null
IdRuolo	int	4	
Descrizione	varchar	50	
DescBreve	varchar	10	

Ruoli (IdRuolo, Descrizione, DescBreve)

Questa tabella contiene i ruoli previsti per gli utenti dell'applicativo, per ogni progetto l'amministratore può associare all'utente un determinato ruolo. Oltre all'identificatore univoco *IdRuolo* la tabella contiene i campi *Descrizione* e *DescBreve*, si riportano i valori previsti per il campo *Descrizione*:

Sviluppatore Esterno: profilo associato all'utente che sviluppa internamente alla struttura come consulente esterno

Sviluppatore: profilo associato all'utente che sviluppa od effettua i test, e fa parte dell'organico della struttura

Sistemista: profilo associato all'utente che effettua l'installazione negli ambienti di test e produzione

Power User: profilo associato all'utente che può effettuare qualsiasi attività sul progetto

Valori del Campo **Descrizione**

Sviluppatore Esterno
Sviluppatore
Sistemista
Power User

Tabella Stati

Nome colonna	Tipo di dati	Lunghezza	Ammetti Null
IdStato	int	4	
Descrizione	varchar	70	
DescBreve	varchar	10	

Stati (IdStato, Descrizione, DescBreve)

Questa tabella contiene gli stati/attività che l'amministratore può associare ai progetti. Per stato si intende la posizione del progetto nel suo Ciclo di Vita in base all'attività svolta dall'utente che lo gestisce, secondo quanto previsto in fase di analisi della struttura. Oltre all'identificatore univoco *IdStato* la tabella contiene i campi *Descrizione* e *DescBreve*, si riportano i valori previsti per il campo *Descrizione*:

Check-In: attività per portare il progetto in utilizzo allo stato di Check-In

Check-Out: attività per prendere in lavorazione il progetto ed aggiornarlo

Promote 1: questa attività è prevista per spostare il progetto in lavorazione dall'area corrente ad un'altra area

Promote 2: come *Promote 1*

Freeze: questa attività è necessaria per "congelare" la nuova versione del progetto

Annulla: consente di interrompere la fase di aggiornamento e riportare il progetto allo stato *Check-In*.

Distribute Test: consente di distribuire un pacchetto in ambiente di test

Distribute Prod: consente di distribuire il nuovo pacchetto in ambiente di produzione

Valori del Campo **Descrizione**

Check-In
Check-Out
Promote 1
Promote 2
Freeze
Annulla
Distribuite Test
Distribuite Prod

Tabella Progetti

Nome colonna	Tipo di dati	Lunghezza	Ammetti Null
idProgetto	int	4	
Nome	varchar	50	
Baseline	varchar	200	✓
AbiDistribuzione	int	4	✓
MatricolaDistribuzione	int	4	✓
AbiUtilizzo	int	4	✓
MatricolaUtilizzo	int	4	✓
FileEar	varchar	50	✓
ftp	varchar	200	✓
ftpport	varchar	5	✓
ftpusername	varchar	20	✓
ftppassword	varchar	20	✓
DataOraIns	datetime	8	
MatricolaIns	varchar	10	
DataOraMod	datetime	8	✓
MatricolaMod	varchar	10	✓

Progetti (IdProgetto, Nome, Baseline, AbiDistribuzione, MatricolaDistribuzione, AbiUtilizzo, MatricolaUtilizzo, FileEar, ftp, ftpport, ftpusername, ftppassword, DataOraIns, MatricolaIns, DataOraMod, MatricolaMod)

Questa tabella contiene le informazioni relative ai progetti che l'utente amministratore deve inserire in fase di censimento. Il Nome e la Baseline vengono reperiti attraverso chiamate a ChangeMan DS, mentre le altre informazioni sono specifiche dell'applicazione web.
In seguito una breve descrizione dei campi:

IdProgetto: identificativo univoco del progetto
Nome: nome del progetto reperito da ChangeMan DS
Baseline: nome della Baseline del progetto reperita da ChangeMan DS
AbiDistribuzione: codice Abi dell'utente che ha effettuato l'ultima distribuzione del pacchetto, questo dato è ridondante, ma è stato inserito per motivi di performance. Il codice Abi, identificativo unico indicante la Banca, è necessario perché l'applicativo gestisce utenti di più banche appartenenti al Gruppo Bancario.
MatricolaDistribuzione: matricola dell'utente che ha effettuato l'ultima distribuzione del pacchetto, anche questo dato è ridondante, ma è stato inserito
AbiUtilizzo e MatricolaUtilizzo: analoghi ad *AbiDistribuzione* e *MatricolaDistribuzione*, necessari per identificare se e chi sta lavorando il progetto. E' importante che l'applicativo sappia se il progetto è il fase di lavorazione perché gli utenti non possono aggiornare contemporaneamente lo stesso progetto
FileEar: nome fisico del file EAR/WAR associato al progetto
Ftp, ftpPort, ftpUserName, ftpPassword: dati del server FTP che punta all'Area di Consegna in cui l'utente può caricare il nuovo EAR/WAR da installare.
Gli altri campi riguardano data ed ora di inserimento ed ultima modifica del progetto.

Tabella StatiRuoli

Nome colonna	Tipo di dati	Lunghezza	Ammetti Null
IdStato	int	4	
IdRuolo	int	4	

ProgettiRuoli (IdStato, IdRuolo)

FK: IdStato REFERENCES Stati

FK: IdRuolo REFERENCES Ruoli

Ogni record contenuto in questa tabella indica che l'utente con ruolo *IdRuolo* può effettuare attività che portano un progetto allo stato *IdStato*. Ad esempio, l'utente con profilo *Sviluppatore Esterno* non può effettuare l'attività *Distribute Prod*, essendo un'attività non consentita ai consulenti esterni per direttive aziendali

Tabella ProgettiStati

Nome colonna	Tipo di dati	Lunghezza	Ammetti Null
IdProgettoStato	int	4	
NomeProgetto	varchar	50	
IdStato	int	4	
IdAreaStart	varchar	50	
IdAreaStop	varchar	50	
IdAreaCheck	varchar	50	✓
DataRichiesta	datetime	8	✓

ProgettiStati (IdProgettoStato, NomeProgetto, IdStato, IdAreaStart, IdAreaStop, IdAreaCheck, DataRichiesta)

FK: NomeProgetto REFERENCES Progetti

FK: IdStato REFERENCES Stati

Contiene gli stati che l'amministratore ha associato ad ogni progetto. Tali attività, che corrispondono ad operazione "scatenate" su ChangeMan DS, possono essere parametrizzate per ogni progetto, ovvero si possono inserire le aree di ChangeMan DS coinvolte. IdAreaStart corrisponde all'area di partenza, IdAreaStop quella di prima destinazione e IdAreaCheck quella che identifica il termine dell'operazione.

Tabella ProgettiUtenti

Nome colonna	Tipo di dati	Lunghezza	Ammetti Null
IdProgettoUtente	int	4	
Abi	int	4	
Matricola	int	4	
NomeProgetto	varchar	50	
IdRuolo	int	4	
Note	varchar	500	✓
DataOraIns	datetime	8	✓
MatricolaIns	varchar	10	✓
DataOraMod	datetime	8	✓
MatricolaMod	varchar	10	✓

ProgettiUtenti (IdProgettoUtente, Abi, Matricola, NomeProgetto, IdRuolo, Note, DataOraIns, MatricolaIns, DataOraMod, MatricolaMod)

FK: NomeProgetto REFERENCES Progetti

FK: IdRuolo REFERENCES Ruoli

Questa tabella contiene le associazioni tra progetti ed utenti, ovvero chi può lavorare il progetto e con quale ruolo. Ad esempio, l'utente della banca avente *Abi* = 05387 con *Matricola* = 9999 può lavorare sul progetto avente *NomeProgetto* = XX con il *Ruolo* = Sistemista. Come per altre tabelle descritte in precedenza, sono previsti campi in cui storicizzare data ed ora di inserimento e modifica dei record.

Tabella StatiAzioni

Nome colonna	Tipo di dati	Lunghezza	Ammetti Null
IdStatoAttuale	int	4	
IdAzione	int	4	
Abilitato	int	4	

StatiAzioni (IdStatoAttuale, IdAzione, Abilitato)

FK: IdStatoAttuale REFERENCES Stati

Questa tabella è stata creata perché alcune attività che l'utente può compiere sui progetti hanno un ordine preciso da rispettare, e l'applicativo deve essere in grado di garantirlo per evitare eventuali errori. La tabella infatti contiene, per ogni stato previsto, quelli ai quali l'utente può accedere. *IdStato* è lo stato attuale in cui si trova il progetto, *IdAzione* è lo stato in cui portare il progetto, ed *Abilitato* è un flag che indica se tale operazione è consentita. La funzionalità di questa tabella sarà evidenziata in modo più chiaro in fase di descrizione delle pagine dell'applicativo.

Tabella Storico

Nome colonna	Tipo di dati	Lunghezza	Ammetti Null
IdStorico	bigint	8	
IdProgetto	int	4	
Abi	int	4	
Matricola	int	4	
NomeProgetto	varchar	50	
Data	datetime	8	
Dettaglio	varchar	3000	
Logs	text	16	✓
Esito	int	4	✓

Storico (IdStorico, IdProgetto, Abi, Matricola, NomeProgetto, Data, Dettaglio, Logs, Esito)

FK: IdProgetto REFERENCES Progetti

FK: NomeProgetto REFERENCES Progetti

Questa tabella contiene lo storico delle attività svolte dagli utenti sui progetti. Il campo *Logs* è stato previsto per inserirvi eventuali risultati di operazioni particolari, ed il campo *Esito* per memorizzare se l'operazione è terminata con esito positivo o negativo. Nel campo *Dettaglio* possono essere inseriti la descrizione dell'operazione ed eventuali commenti dell'utente.

Tabella Ser_Dbrm

Nome colonna	Tipo di dati	Lunghezza	Ammetti Null
Servizio	char	2	
nomeser	varchar	50	
nomedbrm	varchar	50	

Ser_Dbrm (Servizio, nomeser, nomedbrm)
FK: Servizio REFERENCES Progetti

Questa tabella non è stata realizzata per l'applicativo web, ma per supportare la generazione automatizzata dei DBRM nel caso in cui il progetto preveda l'utilizzo di SQL statico. Si descrivono brevemente i campi introdotti, nel capitolo "Generazione DBRM per SQL statico" è illustrato nel dettaglio il processo realizzato.

Servizio: nome del progetto

NomeSer: nome del file .Ser contenuto nel pacchetto EAR/WAR, contenente il codice SQL per l'accesso ai dati

NomeDbrm: nome del DBRM che viene creato e "bindato" sul Database.

Tabella VersioniOggettiEAR

Nome colonna	Tipo di dati	Lunghezza	Ammetti Null
Servizio	char	2	✓
Pack_Id	decimal	9	✓
Nome	varchar	250	
Lung	numeric	9	
Dim	numeric	9	
Contenitore	varchar	50	
Livello	int	4	
Convalida	int	4	✓

VersioniOggettiEAR (Servizio, Pack_Id, Nome, Contenitore , Lung, Dim, Livello, Convalida)

FK: Servizio REFERENCES Progetti

Anche questa tabella non è stata realizzata per l'applicativo web, ma per supportare la generazione automatizzata delle differenze tra rilasci successivi di EAR/WAR. Si descrivono brevemente i campi introdotti, nel capitolo "Differenze oggetti tra rilasci successivi di EAR/WAR" è illustrato nel dettaglio il processo realizzato.

Servizio: nome del progetto
Pack_Id: versione del pacchetto a cui è associato il file, che corrisponde a quella della freeze
Nome: nome del file, compreso il path
Contenitore: contenitore del file, che può essere ad un esempio un file JAR
Lung: attributo lunghezza del file all'interno del pacchetto
Dim: attributo dimensione del file all'interno del pacchetto
Livello: livello di annidamento del file nel pacchetto. Ad esempio, un file contenuto in un JAR che a sua volta è contenuto in un WAR ha *Livello* = 2
Convalida: questo flag indica se l'oggetto appartiene ad una versione del progetto che è stata approvata o meno dall'utente, ha valore 0 finché la fase di inserimento della nuova release del progetto non è completata.

Stored Procedure per estrarre le differenze tra le due ultime versioni di un EAR/WAR

```
CREATE PROCEDURE [ConfrontoOggettiEAR]

@SERVIZIO            char(2)
as
declare    @VECCHIO   int
declare    @NUOVO     int
set @VECCHIO =  (select max (Pack_Id)
                from VERSIONIOGGETTIEAR
                where Servizio = @SERVIZIO
                and Pack_Id <    (select max (Pack_Id)
                                  from VERSIONIOGGETTIEAR
                                  where Servizio = @SERVIZIO))
SET @NUOVO =    (select max (Pack_Id)
                from VERSIONIOGGETTIEAR
                where Servizio = @SERVIZIO)

select     'ELIMINATO' as Tipo, old.Nome, old.Contenitore,
           0 as Lung, 0 as Dim, old.Lung as Lung_Old,
           old.Dim AS Dim_Old
from       VERSIONIOGGETTIEAR old
           left join VERSIONIOGGETTIEAR new
           on    old.Nome = new.Nome
                 and old.Servizio = new.Servizio
                 and old.Contenitore = new.Contenitore
                 and old.Pack_Id = @VECCHIO
                 and  new.Pack_Id = @NUOVO
where      old.Pack_Id = @VECCHIO
           and old.Servizio = @SERVIZIO
           and new.Nome is NULL
           and old.Lung > 0
```

```
UNION ALL
select     'NUOVO' as Tipo, new.Nome, new.Contenitore, new.Lung,
           new.Dim, 0 as Lung_Old, 0 as Dim_Old
from       VERSIONIOGGETTIEAR old
           right join VERSIONIOGGETTIEAR new
           on    old.Nome = new.Nome
                 and old.Servizio = new.Servizio
                 and old.Contenitore = new.Contenitore
                 and old.Pack_Id = @VECCHIO
                 and new.Pack_Id = @NUOVO
where      new.Pack_Id = @NUOVO
           and old.Nome is NULL
           and new.Servizio = @SERVIZIO
           and new.Lung > 0

UNION ALL
select     'VARIATO'   as   Tipo,   new.Nome,   new.Contenitore,
           new.Lung, new.Dim, old.Lung as Lung_Old,
           old.Dim as Dim_Old
from       VERSIONIOGGETTIEAR old
           inner join VERSIONIOGGETTIEAR new
           ON    old.Nome = new.Nome
                 and old.Servizio = new.Servizio
                 and old.Contenitore = new.Contenitore
                 and old.Pack_Id = @VECCHIO
                 and new.Pack_Id = @NUOVO
where      (old.LUNG <> NEW.LUNG or old.Dim <> new.Dim)
           and old.Servizio=@SERVIZIO

GO
```

2.6 J2EE: PACKAGING E DEPLOYMENT

In questo paragrafo si descrivono gli standard J2EE per il *packaging* ed il *deployment* delle applicazioni, con l'obiettivo di rendere più chiare al lettore le attività descritte nei capitoli relativi alla realizzazione del progetto.
La J2EE platform consente agli sviluppatori di creare differenti parti delle loro applicazioni sotto forma di componenti riusabili. Questi componenti possono essere poi raggruppati in insiemi, detti *moduli*, a seconda di caratteristiche o funzionalità comuni. Il processo di assemblaggio dei componenti in moduli e dei moduli in enterprise applications è detto *packaging*.
Il processo di installazione e personalizzazione di un'applicazione in un certo ambiente operativo è detto *deployment*. Affinché sia possibile personalizzare un'applicazione i suoi componenti devono poter essere configurabili mediante un meccanismo standard.
La J2EE fornisce strumenti standard per semplificare questi processi di packaging e deployment. Essa usa file JAR come package standard per i moduli e le applicazioni, e dei files XML-based per configurare questi moduli detti *deployment descriptors*.

2.6.1 Ruoli e compiti

I processi di packaging e deployment coinvolgono tre differenti ruoli identificabili in tre figure professionali non necessariamente svolte da persone diverse:

Application component providers: hanno il compito di sviluppare gli Enterprise Java Beans, le pagine HTML e JSP, e tutte le ulteriori classi associate. Essi forniscono tutte le informazioni strutturali del deployment descriptors per ciascun componente, che includono, come già visto, l'indicazione della home e della remote interface e della classe degli EJB, il meccanismo di persistenza usato, e il tipo di risorse che questi componenti utilizzano. Senza l'ausilio dei deployment descriptors tutte queste informazioni sarebbero da includere nel codice dei componenti riducendo drasticamente la loro flessibilità e riusabilità.

Application assemblers: forniscono informazioni relative all'applicazione nel suo complesso. Tipicamente ad essi spetta il compito di mappare le varie servlet sui diversi URL, definire le pagine di errore da usare, i vincoli di sicurezza dei vari componenti, ecc.

Deployers: sono responsabili del deployment dei componenti in un particolare ambiente operativo. Ad essi spetta il compito di installare i diversi componenti sul J2EE server (o sui J2EE servers) fornendo le informazioni aggiuntive specifiche per l'ambiente operativo. Ad esempio essi si occupano di mappare gli utenti e gli accounts esistenti con i security roles definiti dall'Application assembler.

2.6.2 Moduli J2EE

Un'applicazione J2EE è impacchettata in un file Enterprise Archive (EAR), un file standard Java JAR con estensione .ear. L'obiettivo di questo meccanismo di packaging è di fornire una unità di deployment che sia portabile.
Un file EAR contiene uno o più moduli J2EE e un *J2EE application deployment descriptor*, che contiene indicazioni sui moduli stessi.

2.6.2.1 Moduli EJB

Un modulo EJB è l'unità più piccola di EJB della quale si può fare il deployment. Tale modulo è impacchettato in un file EJB JAR, cioè un file standard Java JAR con estensione .jar che contiene:

le classi Java degli EJB e le loro remote e home interface. Se si tratta di un entity bean deve essere compresa anche la classe della primary key;

tutte le altre classi Java dalle quali dipendono gli EJB che non siano già comprese nella J2EE platform;

un EJB deployment descriptors, tipicamente chiamato ejb-jar.xml, del quale si è già mostrato il contenuto minimo nell'esempio della Calcolatrice, che deve essere contenuto in una particolare directory di nome META-INF.

E' da notare che un EJB JAR file differisce da un normale JAR file perché nel primo è contenuto anche il deployment descriptors.

2.6.2.2 Moduli web

Un modulo Web è l'unità più piccola di risorse Web della quale si può fare il deployment. Tale modulo è impacchettato in un file Web Archive (WAR), un file standard Java JAR con estensione .war che contiene:

le classi Java delle servlet e le classi dalle quali queste dipendono, eventualmente impacchettate a loro volta in un normale file JAR;

le pagine JSP e le classi Java dalle quali dipendono;

documenti statici, come ad esempio le pagine HTML, le immagini, i file sonori, ecc.;

le applets;

un Web deployment descriptor, tipicamente chiamato web.xml e contenuto in una speciale directory chiamata WEB-INF.

2.6.2.3 Moduli Application client

Un modulo application client è impacchettato in un file JAR con estensione .jar e contiene:
le classi Java che implementano il client;
un Application client deployment descriptor che descrive gli EJB e le risorse esterne referenziate dall'applicazione.
La J2EE non specifica tools per effettuare il deployment di un application client, o meccanismi per installarlo. Alcune piattaforme J2EE sofisticate possono consentire il deployment dell'application client direttamente nel J2EE server e metterlo così automaticamente a disposizione di alcuni clients intranet. Altre piattaforme J2EE possono invece richiedere che il deployment dell'application client sia manualmente effettuato su ciascuna macchina client.

CAPITOLO 3 REALIZZAZIONE AUTOMATISMI PER INSTALLAZIONE EAR/WAR

In questo capitolo sono descritti dettagliatamente gli automatismi realizzati per predisporre un'infrastruttura che, tramite il software ChangeMan DS, permette di seguire l'intero ciclo di vita delle applicazioni JAVA dipartimentali installate su WebSphere per z/OS. In particolare queste applicazioni sono costituite da file EAR/WAR, creati secondo le specifiche illustrate nel precedente capitolo.

Per l'ambiente di TEST e PRODUZIONE tali attività sono già automatizzate dal software ChangeMan DS, il quale permette di mantenere anche le versioni degli applicativi, e l'utente deve soltanto compiere poche semplici operazioni descritte in dettaglio successivamente.

Secondo l'infrastruttura realizzata le applicazioni sono associate a Progetti di ChangeManDS creati dagli amministratori, in particolare per le applicazioni trattate essi sono costituiti dal file EAR/WAR e da un eventuale file ZIP contenente i DBRM necessari, nel caso venisse utilizzato SQL statico per accedere al database. L'utilizzo di SQL statico da parte dell'applicazione comporta infatti la generazione sulla partizione Unix di TEST dei DBRM, necessari per poter effettuare correttamente la fase di BIND sul database, ed essa viene effettuata in modo completamente automatizzato e parametrizzato dall'infrastruttura.

Nel seguito sono elencate nel dettaglio tutte le fasi, per chiarezza si indica anche una breve legenda e la descrizione delle macchine interessate.

3.1 ARCHITETTURA PROCESSO DI PREPARAZIONE DEL PACCHETTO DI INSTALLAZIONE

Legenda

Nome	*Descrizione*
PROGETTO	Progetto di ChangeManDS, ovvero un insieme di file correlati
FREEZE	la freeze di un progetto non è altro che un legame logico tra le varie versioni dei file associati al progetto stesso
AREA	directory residente su un PC/HOST in cui è stato installato un AGENT, ovvero il demone che permette a ChangeMan DS di comunicare e trasferire file tra PC/HOST. ChangeManDS infatti "vede" soltanto le AREE, i cui privilegi di accesso sono gestiti dall'amministratore del Software attraverso un'apposita utilità
BASELINE_AREA	AREA in cui risiedono le versioni aggiornate dei file relativi ai PROGETTI
SVILUPPO_AREA	AREA di passaggio necessario per lo sviluppo/preparazione di un PROGETTO
CONSEGNA_AREA	particolare AREA_SVILUPPO necessaria per inserire

	in ChangeMan DS i pacchetti applicativi realizzati esternamente
ENDUSER_AREA	AREA in cui viene distribuita un'applicazione, tipicamente può consistere nell'ambiente di TEST o di PRODUZIONE
CHECKOUT	trasferimento di un file o di un progetto da una BASELINE_AREA ad una SVILUPPO_AREA o CONSEGNA_AREA
CHECKIN	trasferimento di un file o di un progetto da una SVILUPPO_AREA o CONSEGNA_AREA ad una BASELINE_AREA
PROMOTE_TO_SVILUPPO	trasferimento di un file o di un progetto tra due SVILUPPO_AREA
DISTRIBUTE_TO_ENDUSER	distribuzione di un file o di un progetto in una ENDUSER_AREA

Tabella 2

La legenda, pur descrivendo anche concetti citati nel capitolo "2.3 SOFTWARE PER IL VERSIONING: SERENA CHANGEMAN DS", è stata riportata per rendere più chiare al lettore le fasi esposte nel corso del capitolo.

Macchine interessate

Server di ChangeMan DS (chiamato in seguito SRVCMDS):

SISTEMA OPERATIVO: Windows 2003 Server
In questa macchina è installata la componente Server di ChangeMan DS, che può colloquiare con le macchine in cui sono installati gli agent di ChangeMan DS.

Partizione UNIX dell'ambiente di Test su Mainframe (chiamato in seguito LP02):

SISTEMA OPERATIVO: UNIX
In questa partizione risiede WebSphere, pertanto su di essa è stato installato l'agent di ChangeMan DS che dove colloquiare con esso. Inoltre attraverso tale agent è possibile interagire con il DB2 per creare i DBRM, nel caso fosse previsto un accesso ai dati tramite SQL statico.

Diagramma delle Fasi

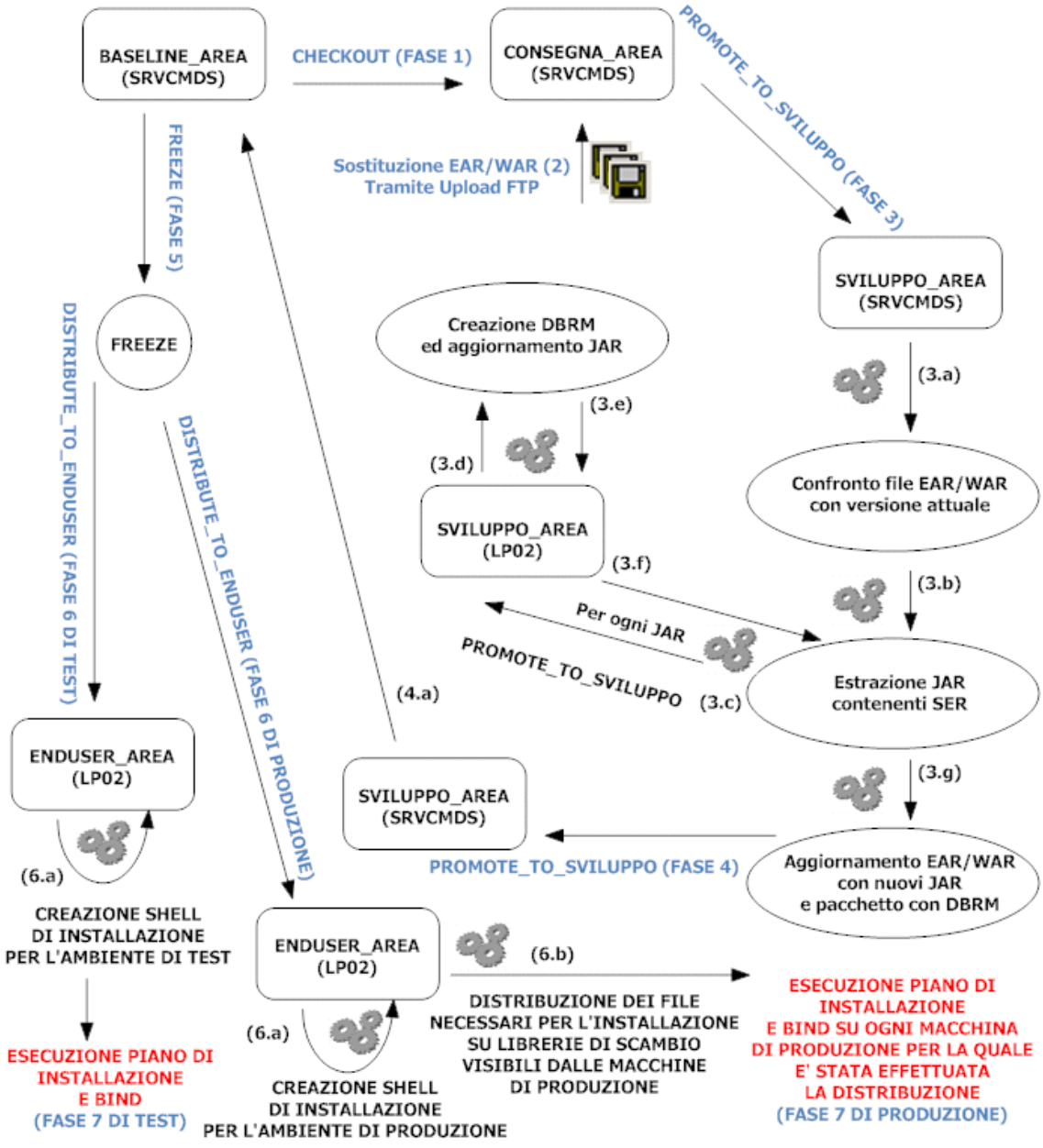


Figura 9

Descrizione delle Fasi

FASE 1)
CHECKOUT del PROGETTO dalla BASELINE_AREA ad un'apposita CONSEGNA_AREA, entrambe residenti su SRVCMDS;

FASE 2)
sostituzione del vecchio EAR/WAR residente nella CONSEGNA_AREA con quello nuovo.

FASE 3)
PROMOTE_TO_SVILUPPO del PROGETTO, indicando un'apposita SVILUPPO_AREA, che innescherà automaticamente le seguenti attività tramite gli script ad essa associati:

confronto di tutti i file contenuti nella versione attuale dell'EAR/WAR con quella appena inserita, per evidenziare all'utente gestore del progetto l'entità delle modifiche, soprattutto in caso di sviluppo esterno;

scompattazione del file EAR ed estrazione di tutti i file JAR contenuti;

scompattazione dei file JAR e creazione di un file di parametri, denominato ListaSER%ACRONIMOPROGETTO%.txt, contenente tante righe quanti sono i file SER contenuti nei file JAR del PROGETTO;

trasferimento di ogni file JAR verso un'apposita SVILUPPO_AREA residente su LP02, in cui viene creata una libreria PDS temporanea e vengono eseguite, per ogni file JAR, le seguenti attività:
scompattazione del file JAR;
esecuzione del comando DB2PROFC per ogni file SER contenuto nel JAR, in modo da generare i nuovi SER ed i DBRM, i quali vengono collocati nella libreria PDS creata;
sostituzione dei vecchi SER con i nuovi;
ricompattazione del file JAR;

trasferimento del file DBRM%ACRONIMOPROGETTO%.zip, contenente i DBRM relativi all'EAR scaricato dalla BASELINE_AREA, verso la stessa SVILUPPO_AREA del punto 3 residente su LP02, in cui vengono eseguite le seguenti attività:

copia dei DBRM creati al punto 3 in una directory in partizione UNIX;
generazione del nuovo file DBRM%ACRONIMOPROGETTO%.zip, contenente tutti i nuovi DBRM ed il file parametri necessario per la fase di distribuzione ListaSER%ACRONIMOPROGETTO%.txt;
trasferimento dei file JAR e del file DBRM%ACRONIMOPROGETTO%.zip dall'area residente su LP02 a quella residente su SRVCMDS;

rigenerazione del file EAR inserendovi i JAR aggiornati;

FASE 4)
PROMOTE_TO_SVILUPPO del PROGETTO, indicando un'apposita SVILUPPO_AREA, che innescherà automaticamente le seguenti attività:

pulizia delle aree temporanee;

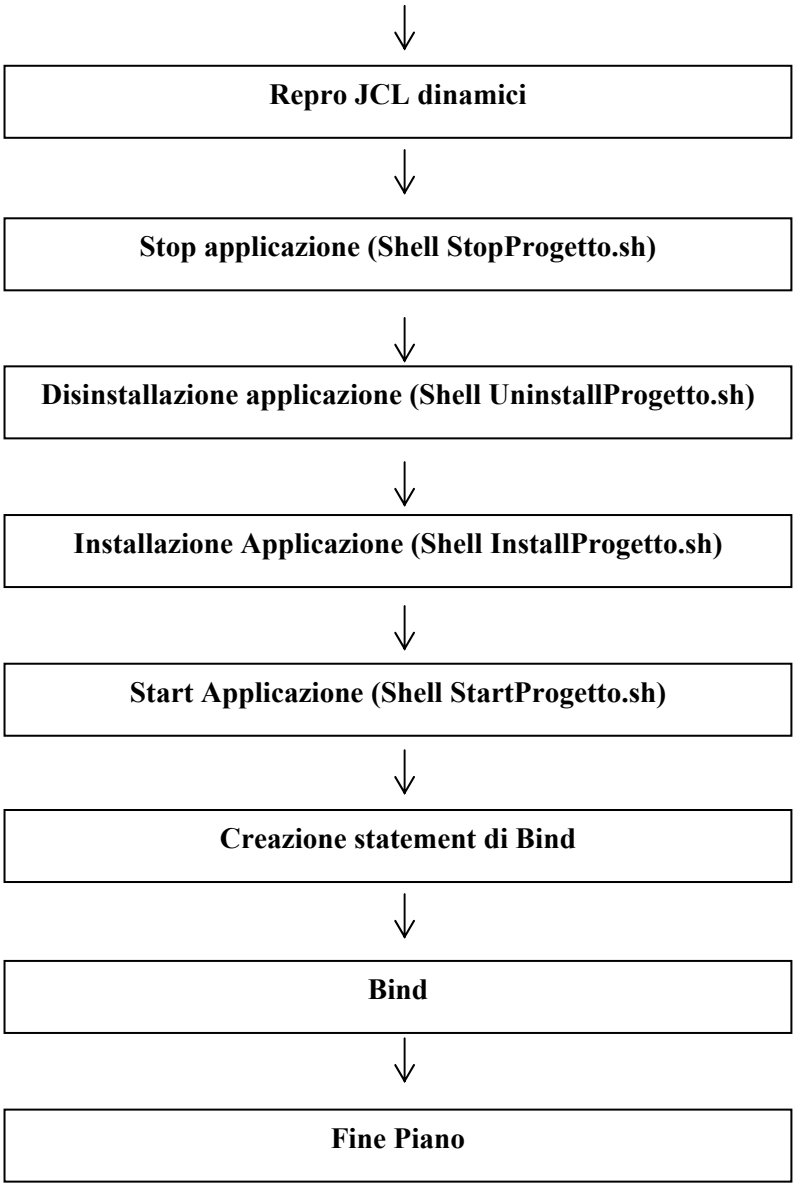
CHECKIN del PROGETTO;

FASE 5)
generazione di una nuova FREEZE, il cui nome deve corrispondere alla versione dei file del PROGETTO;

FASE 6 di Test)
DISTRIBUTE_TO_ENDUSER della FREEZE verso l'area di TEST, in cui vengono trasferiti i file del PROGETTO e create le SHELL UNIX necessarie per disinstallare la vecchia versione dell'applicazione, installare la nuova ed effettuare gli eventuali BIND.

FASE 7 di Test)
Lancio di un piano OPC su LP02 il quale, utilizzando le SHELL UNIX generate nella precedente FASE, installa la nuova versione dell'applicativo ed effettua gli eventuali BIND. Questo piano può essere schedulato o lanciato dall'utente, e deve ricevere in ingresso un parametro corrispondente all'acronimo dell'applicazione che deve essere installata.

Schema del piano OPC per l'ambiente di TEST



L'infrastruttura è stata preparata anche per la distribuzione delle applicazioni in ambiente di PRODUZIONE però, a differenza dell'ambiente di TEST, nelle relative MACCHINE per motivi di sicurezza non sono presenti AGENT che permettano di comunicare direttamente con ChangeManDS.
Pertanto sono stata implementate due ulteriori FASE 6 e 7 di Produzione, simili a quelle realizzate per l'ambiente di TEST, descritte di seguito:

FASE 6 di Produzione)
DISTRIBUTE_TO_ENDUSER della FREEZE verso l'area di PRODUZIONE (virtuale), anch'essa residente su LP02, in cui vengono trasferiti i file del PROGETTO e create le SHELL UNIX in directory riservate alle diverse MACCHINE di PRODUZIONE. Per esempio, se l'ambiente di PRODUZIONE è costituito da due MACCHINE, sarà presente una directory contenente i file del PROGETTO pronti per l'installazione in PRODUZIONE, ed altre due directory contenenti rispettivamente le SHELL relative alla prima ed alla seconda MACCHINA.
La generazione delle SHELL UNIX e delle direttive di BIND ha diversi gradi di libertà, grazie al fatto che sono stati creati file di configurazione che contengono, per ogni MACCHINA e PROGETTO, numerosi parametri.
Successivamente i file necessari all'installazione vengono zippati in tanti pacchetti d'installazione quante sono le MACCHINE, ed inseriti in apposite librerie OMVS di scambio visibili da tutte le MACCHINE, create in base alla seguente convenzione:

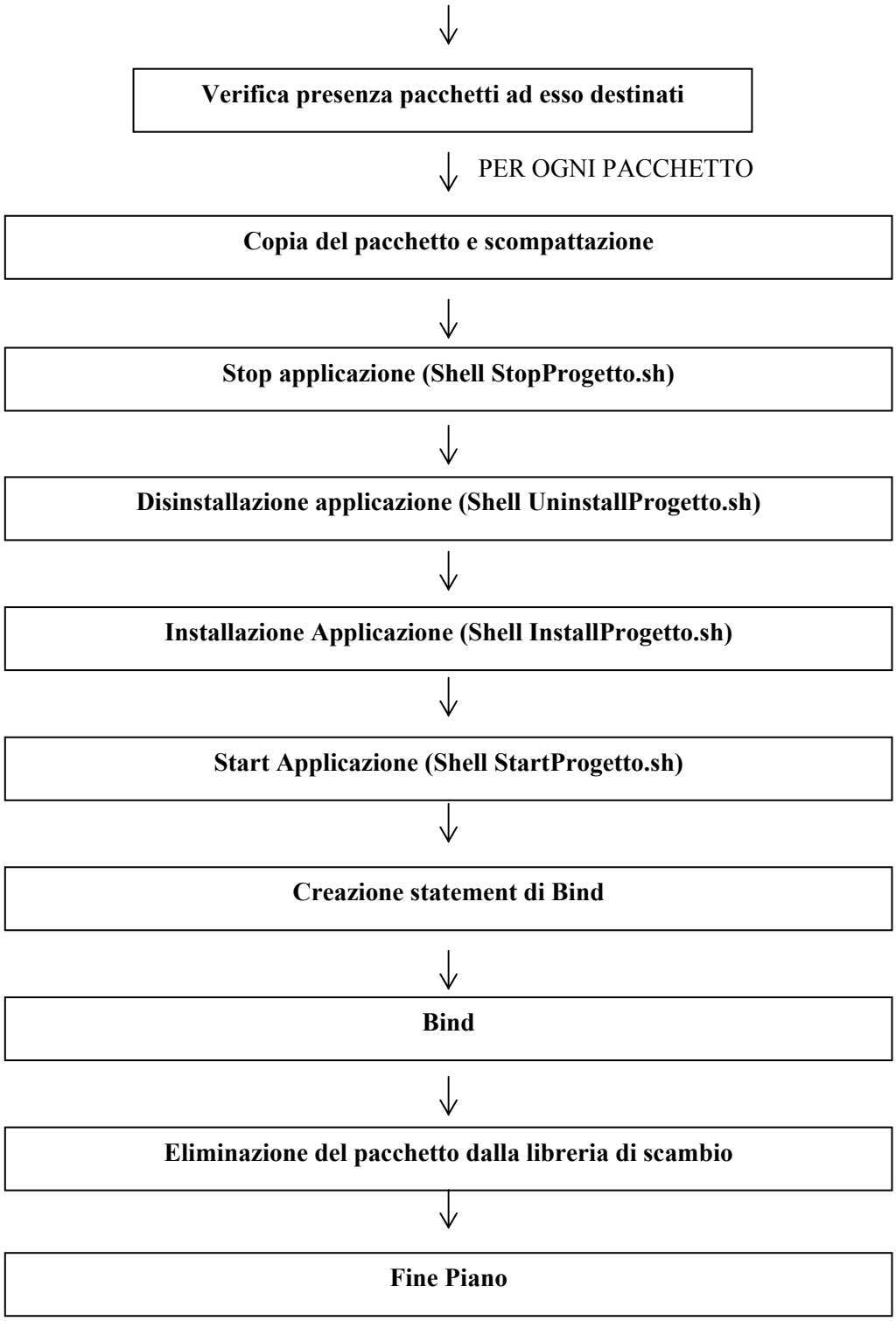
BASE.SYSIN.NomeMacchinaWEB.AcronimoProgettoProgressivo

Esempio: BASE.SYSIN.OS01WEB.KW1

FASE 7 di Produzione)
In questa fase è necessario trasferire fisicamente i file necessari all'installazione nelle directory delle MACCHINE di PRODUZIONE, quindi lanciare per ognuna di esse un piano OPC analogo a quello descritto in precedenza per la FASE 7 di Test.
Per effettuare il trasferimento è stato aggiunto all'inizio dei piani OPC relativi alle MACCHINE di PRODUZIONE uno nuovo STEP, che esegue il "pescaggio" e la scompattazione dei pacchetti dalle librerie OMVS di scambio generate nella FASE 7 di Produzione.
In ogni MACCHINA di PRODUZIONE il piano opportunamente schedulato verifica se esistono pacchetti a lui destinati controllando il NomeMacchina, ed in caso affermativo effettua per ognuno di essi le operazioni necessarie ed infine li elimina dalle librerie di scambio.

Schema del piano OPC per l'ambiente di PRODUZIONE

Il piano è stato schedulato tutte le notti in tutte le MACCHINE di PRODUZIONE, ed effettua le seguenti attività:



Il sistema consente di avviare simultaneamente le attività descritte per PROGETTI diversi, anche se vengono utilizzate le medesime aree di lavoro, perché l'elaborazione è stata totalmente parametrizzata.
In seguito si descrivono in modo dettagliato alcune fasi importanti del processo appena rappresentato.

3.1.1 Generazione e BIND dei DBRM per SQL statico

Generazione

La generazione dei DBRM, che avviene nella FASE 3) del processo, viene effettuata per ogni file SER contenuto nei file JAR del PROGETTO che sono stati trasferiti nell'apposita SVILUPPO_AREA residente su LP02. Tale operazione può essere effettuata soltanto in questa area perché è l'unica che può colloquiare con la base dati DB2 sulla quale i DBRM dovranno essere "bindati".
Come già anticipato, nella SVILUPPO_AREA viene creata una libreria PDS temporanea e vengono eseguite, per ogni file JAR, le seguenti attività:

1) scompattazione del file JAR;

2) esecuzione del comando DB2PROFC per ogni file SER contenuto nel JAR, in modo da generare i nuovi SER ed i DBRM (4 ogni SER), i quali vengono collocati nella libreria PDS creata.
   Il file di parametri generato, denominato ListaSER%ACRONIMOPROGETTO%.txt, contiene alcune informazioni necessarie per il comando DB2PROFC, ed in particolare:

   a. Path del file SER all'interno del JAR
   b. Nome del file SER
   c. Nome del DBRM associato, che può avere una lunghezza massima di 7 caratteri e dev'essere univoco. Questo nome viene generato basandosi sulla tabella Ser_Dbrm
   d. Nome del file JAR contenitore
   e. Nome del file EAR/WAR
   f. Nome del progetto censito su ChangeMan DS

Esempio:

```
it/bper/sqlj Cassa_SJProfile0.ser Cassa01 Giornale.jar
AGENZIAWEB CW
```

e le altre sono contenute in un ulteriore file di parametri generato dall'amministratore di ChangeMan DS contenente queste informazioni:

g. Nome del servizio
h. Suffisso Application Server
i. Datasource
j. Connettore CTG
k. Nome del db2
l. Qualifier del db2
m. Owner del db2
n. Collection
o. Nome macchina (node) da inserire nell'ultima riga della shell per l'installazione batch
p. Nome istanza application server (server) da inserire nell'ultima riga della shell per l'installazione batch
q. Context Root (solo per WAR) da inserire nell'ultima riga della shell per l'installazione batch
r. Path Application Server
s. Datasource da mappare (SOLO 1 E SOLO PER I WAR, ALTRIMENTI "NO". Funziona solo se il nome del modulo web corrisponde al nome del war)
t. Se non ci sono Datasource da mappare "NO", altrimenti "SI"
u. MapWebModToVH ("SI" o "NO")

Esempio:

```
CW:SPOR:portaleds:DSCICSCW:BPERDB2C:MO:FAIBIND:KWCOJD01:
OSN2:serve1:NO:/WebSphere/V5R1M0/AppServer/bin/:NO:SI:SI
```

Il comando DB2PROFC, generato automaticamente da una Shell Unix, diventa come il seguente:

```
db2profc -online=BPERDB2C -schema=FAIBIND -pgmname=Cassa01
Cassa_SJProfile0.ser
```

L'ultimo file di parametri interessato, che viene generato in automatico, viene letto online dal comando DB2PROFC e specifica le seguenti informazioni:

```
DB2SQLJSSID=NomeDB2
DB2SQLJPLANNAME=Plan
DB2CURSORHOLD=YES
DB2SQLJDBRMLIB=UtenteAgenteChangeManDS.NomeProgetto
```

(la libreria PDS in cui copiare i DBRM creati)

3) sostituzione dei vecchi SER con i nuovi;

4) ricompattazione del file JAR;

Successivamente, come già indicato, i JAR aggiornati sono inseriti nel file EAR/WAR ed i DBRM introdotti nel file DBRM%ACRONIMOPROGETTO%.zip.

BIND

Le direttive di BIND vengono preparate nella FASE 6) e l'esecuzione avviene nella FASE 7).
In particolare la fase di preparazione consiste nel lancio di una Shell Unix che, basandosi sui file di configurazione descritti precedentemente, genera tante direttive come la seguente per ogni file SER:

Indicazione del DB2 su cui effettuare i BIND, inserita soltanto all'inizio delle direttive

```
DSN SYSTEM(DB2C)
```

Primo DBRM

```
BIND PACKAGE(KWCOJD01) QUALIFIER(MO) ACTION(REPLACE) -
MEMBER (CASSA011) VALIDATE(BIND) ISOLATION(UR) -
CURRENTDATA (NO) OWNER(FAIBIND) -
RELEASE(COMMIT) ENABLE(*) EXPLAIN(YES)
```

Secondo DBRM

```
BIND PACKAGE(KWCOJD01) QUALIFIER(MO) ACTION(REPLACE) -
MEMBER (CASSA012) VALIDATE(BIND) ISOLATION(CS) -
CURRENTDATA (NO) OWNER(FAIBIND) -
RELEASE(COMMIT) ENABLE(*) EXPLAIN(YES)
```

Terzo DBRM

```
BIND PACKAGE(KWCOJD01) QUALIFIER(MO) ACTION(REPLACE) -
MEMBER (CASSA013) VALIDATE(BIND) ISOLATION(RS) -
CURRENTDATA (NO) OWNER(FAIBIND) -
RELEASE(COMMIT) ENABLE(*) EXPLAIN(YES)
```

Quarto DBRM

```
BIND PACKAGE(KWCOJD01) QUALIFIER(MO) ACTION(REPLACE) -
MEMBER (CASSA014) VALIDATE(BIND) ISOLATION(RR) -
CURRENTDATA (NO) OWNER(FAIBIND) -
RELEASE(COMMIT) ENABLE(*) EXPLAIN(YES)
```

Il valore del campo MEMBER corrisponde al quello del DBRM creato con il comando DB2PROFC, ovvero campo NomeDbrm della tabella Ser_Dbrm concatenato con i valori 1, 2, 3 o 4.

L'effettivo BIND avviene durante l'esecuzione del Piano OPC, che prima di lanciare le direttive scompatta il file DBRM%ACRONIMOPROGETTO%.zip e copia i DBRM in un'apposita libreria.

3.1.2 Differenze oggetti tra rilasci successivi di EAR/WAR

Questo modulo è stato realizzato per verificare, soprattutto nel caso di sviluppo del progetto esternamente all'azienda, quali oggetti contenuti all'interno dell'EAR/WAR sono stati aggiunti, modificati od eliminati nel passaggio da una versione alla successiva.

L'automatismo memorizza nella tabella VersioniOggettiEAR le informazioni relative ad ogni oggetto contenuto, come indicato nella descrizione della tabella del Capitolo2, mostrando anche la versione di ChangeMan DS associata a quel pacchetto.
La complessità del modulo deriva dalla struttura dei pacchetti J2EE, che possono essere paragonabili a file zippati annidati, come ad esempio un EAR contenente un WAR, che a sua volta contiene file JAR. Pertanto è stato necessario realizzare un programma che effettua una scompattazione ricorsiva dei file contenitori per estrarre tutti gli oggetti, e tale attività è stata effettuata con l'ausilio della Command Line di WinZip 9.0.

Per chiarezza si riportano i campi relativi alla tabella:

Servizio: nome del progetto
Pack_Id: versione del pacchetto a cui è associato il file, che corrisponde a quella della freeze
Nome: nome del file, compreso il path
Contenitore: contenitore del file, che può essere ad un esempio un file JAR
Lung: attributo lunghezza del file all'interno del pacchetto
Dim: attributo dimensione del file all'interno del pacchetto
Livello: livello di annidamento del file nel pacchetto. Ad esempio, un file contenuto in un JAR che a sua volta è contenuto in un WAR ha *Livello* = 2
Convalida: questo flag indica se l'oggetto appartiene ad una versione del progetto che è stata approvata o meno dall'utente, ha valore 0 finché la fase di inserimento della nuova release del progetto non è completata.

E' poi stata realizzata una Stored Procedure denominata "ConfrontoOggettiEAR", mostrata nel Capitolo2, che estrae dal Database le differenze tra le ultime due versioni dei pacchetti associati ad un progetto, per renderle visibili all'utente, basandosi sui valori dei campi *Lung* e *Dim.*

3.1.3 Generazione Shell Unix di installazione

Questa fase molto importante del processo è stata realizzata basandosi sulla Console WSADMIN di WebSphere, che consente anche di reperire informazioni dagli EAR/WAR e dalla configurazione dell'application server, come ad esempio la presenza o meno di un'applicazione, od il nome dei DataSource di un'applicazione da mappare sul Server.

Come indicato nel Capitolo2, sono state automatizzate le generazioni delle Shell necessarie per arrestare, disinstallare, installare ed avviare un'applicazione, basandosi su parametri inseriti dall'amministratore di ChangeMan DS in un apposito file, e sulle informazioni reperite tramite la Console WSADMIN, non ci soffermiamo sul dettaglio implementativo che viene riportato nell'Appendice2.

CAPITOLO 4 REALIZZAZIONE INTERFACCIA WEB

4.1 INTEGRAZIONE CON SISTEMI ESISTENTI

L'interfaccia Web è stata realizzata rispettando i requisiti necessari per integrarla nel sistema informativo aziendale, in particolare nel Portale aziendale.
Il Portale si presenta come un ambiente accentratore delle applicazioni fruibili in filiale e negli uffici centrali, ed è integrato con il sistema di sicurezza RACF (Resource Access Control Facility).
L'utente accede all'interfaccia Web tramite il menu del Portale, e riceve da RACF il profilo con il quale può abilitarsi. L'applicazione, che rispetta le specifiche, gestisce tale profilo e concede più o meno funzionalità/visibilità all'utente, ma soprattutto consente l'accesso soltanto transitando dal Portale.

4.2 Ambiente di sviluppo

Elemento portante del sistema è un'applicazione Web "dinamica", le cui pagine sono generate utilizzando le informazioni memorizzate all'interno del database relazionale descritto in precedenza.

Fogli di stile
Sono stati utilizzati fogli di stile, oggi fondamentali nello sviluppo di qualsiasi progetto di questo tipo, perché consentono di uniformare l'aspetto del testo utilizzato. E' sufficiente modificare uno stile, perché tutti gli elementi dello stesso tipo vengano automaticamente aggiornati secondo le nuove impostazioni.

All'interno di una pagina Web l'utilizzo di fogli di stile presenta inoltre un grande vantaggio, quello di **semplificare la scrittura del codice HTML** che la compone, evitando allo sviluppatore di inserire in ogni paragrafo di ogni pagina comandi specifici per la formattazione del testo (font, grandezza, colore, allineamento, …).

CMDS Web
CMDS Web, il cui funzionamento sarà approfondito nel prossimo capitolo, è un'Applicazione

Web-Database che svolge il ruolo di “cerniera” tra ChangeMan DS ed il Portale aziendale. CMDS Web è’ stato sviluppato con l’obiettivo di integrarsi nel sistema di sicurezza e profilatura aziendale, e soprattutto per semplificare la gestione del Ciclo di Vita degli applicativi Web, nascondendo i dettagli tecnici all’utente finale.

4.2.1 Architettura di sistema

L'applicazione CMDS Web consiste in un'applicazione Web "dinamica", le cui pagine vengono generate utilizzando le informazioni memorizzate all'interno di un database relazionale e manipolandole nel modo opportuno. Lo standard seguito per lo sviluppo è rappresentato dalle applicazioni Java/JSP eseguite dal server web (**applicazioni web server-side**).

Il funzionamento di un sistema che prevede l'utilizzo di applicazioni Java/JSP è semplice e risulta molto simile a quello di un normale motore di ricerca. I parametri di ricerca, introdotti dall'utente all'interno del browser Internet (Figura 1), vengono inviati al web server (Figura 2) che, riconosciuta la richiesta, la inoltra direttamente all'applicazione Java/JSP (Figura 3).

Figura 1

Figura 2

Figura 3

L'applicazione Java/JSP genera dinamicamente una o più interrogazioni, chiamate tecnicamente "query" (Figura 4), che vengono inviate al database server a cui si appoggia l'applicazione. Il database server elabora i dati (Figura 5), restituendo all'applicazione Java/JSP le informazioni richieste (Figura 6).

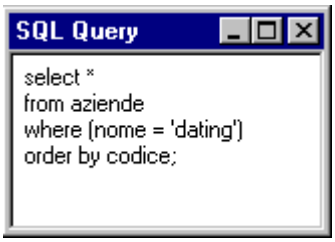


Figura 4

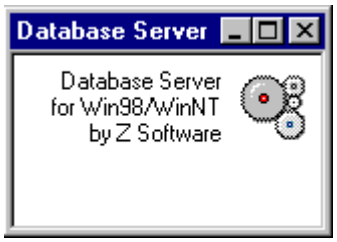


Figura 5

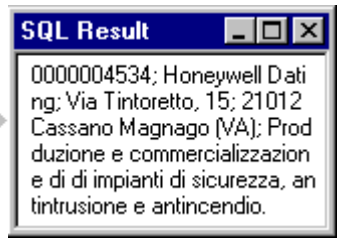


Figura 6

L'applicazione Java/JSP (Figura 7) è in grado di creare dinamicamente una pagina Internet contenente i risultati ottenuti dal database e di inviarla al web server (Figura 8) che a sua volta la fornisce all'utente che ne ha fatto richiesta. Il risultato finale viene quindi mostrato all'utente sotto forma di pagina Internet all'interno del proprio browser (Figura 9).

Figura 7

Figura 8

Figura 9

4.2.1 Layout di sistema

Il layout utilizzato prevede che tutto il software necessario per il funzionamento di CMDS Web venga installato sul **Web Server**, mentre il database su un **database server** dedicato. Questa particolare configurazione offre notevoli spunti di interesse riguardanti le problematiche legate alla sicurezza dei dati e l'architettura del sistema, che, però, non è possibile approfondire in questa sede.

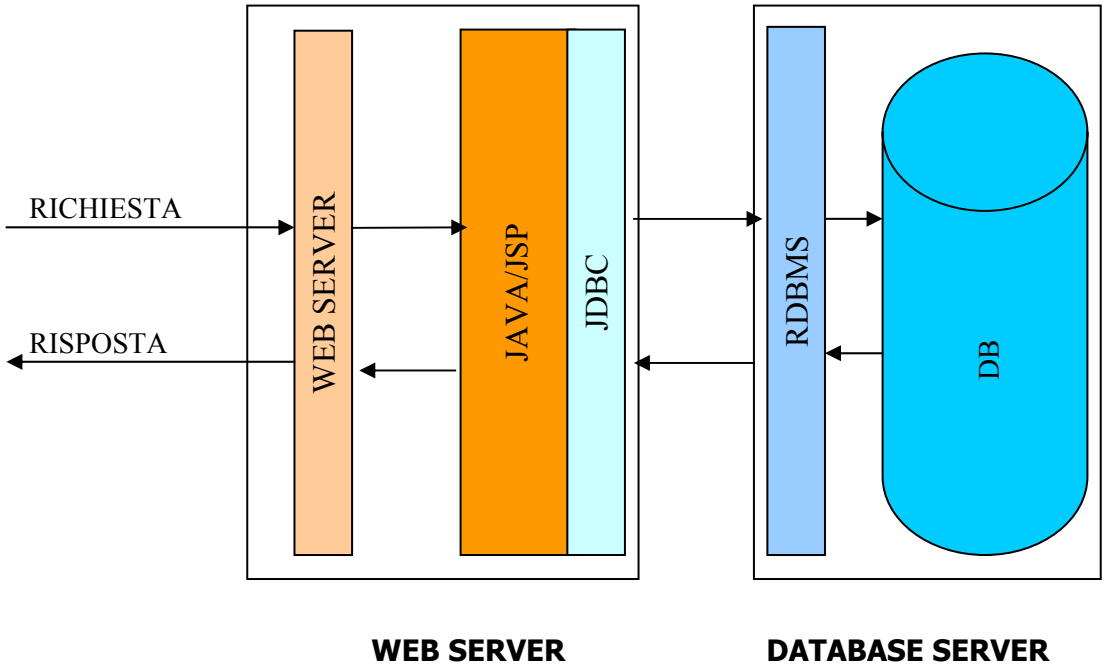


Figura 10

4.2.3 Vantaggi della piattaforma J2EE

L'obiettivo della piattaforma Java 2 Enterprise Edition (J2EE), distribuita della Sun Microsystem, consiste nel definire funzionalità standard che supportino gli sviluppatori nella realizzazione di applicazioni distribuite, ovvero degli involucri attraverso i quali vengono trasmesse le informazioni verso il mondo di internet. Durante la trattazione si cercheranno di mettere in evidenza le numerose potenzialità che l'architettura a **tre livelli** prevista nelle specifiche J2EE presenta rispetto ad una più semplice architettura a **due livelli**.
All'interno del capitolo sono stati inseriti anche alcuni semplici esempi con la finalità di rendere più facile al lettore la comprensione dei concetti chiave.
La piattaforma J2EE offre una serie di benefici agli sviluppatori di applicazioni distribuite.

4.2.3.1 Architettura e sviluppo semplificati

La J2EE supporta un modello di sviluppo *component-based* semplificato. Essendo basata sul linguaggio di programmazione Java e sulla Java 2 Standard Edition (J2SE platform), questo modello offre grandi vantaggi di portabilità (sistema *Write Once*, *Run Anyway*) e garantisce il supporto su tutti i server conformi a questo standard.
Il modello di sviluppo component-based J2EE arricchisce la produttività delle applicazioni in diversi modi:

assicura flessibilità alle funzionalità desiderate, consentendo diverse possibilità di configurazione dell'architettura della applicazione, a seconda di vari fattori, come ad esempio il tipo di client, la sicurezza richiesta per gli accessi alle sorgenti di dati, e altri che verranno approfonditi in seguito. Il modello component-based inoltre semplifica la manutenzione dell'applicazione, in quanto i singoli componenti possono essere aggiornati o sostituiti indipendentemente.

assicura ai componenti la disponibilità di una serie di servizi nell'ambiente di run-time, e la possibilità di essere dinamicamente connessi ad altri componenti, semplicemente fornendo alcune interfacce. Ad esempio attraverso i *deployment descriptors* (un file di testo in formato XML che specifica il comportamento di un componente attraverso tag XML) è possibile comunicare specifici parametri all'ambiente di run-time personalizzando l'applicazione senza dover modificare il codice dei componenti.

supporta la suddivisione dei compiti, in quanto ciascun set di componenti può essere associato ad un certo ambito di sviluppo, consentendo a ciascuna figura professionale di concentrarsi specificatamente sulle proprie competenze e abilità. Questa suddivisione dei compiti, oltre a favorire la specializzazione e quindi migliorare la qualità dei singoli componenti, consente un criterio di sviluppo dell'applicazione in parallelo, aumentandone la velocità di produzione e manutenzione.

4.2.3.2 Scalabilità

La J2EE fornisce supporti per adattare in modo semplice e funzionale una applicazione ad eventuali aumenti o riduzioni delle dimensioni del progetto senza comprometterne l'efficienza delle prestazioni. Ad esempio il supporto per il *connection pooling* (il pool di connessioni è un meccanismo molto utile per diminuire il gravoso tempo di accesso ai database) assicura rapido accesso ai dati anche ad un numero crescente di clients.
Inoltre l'architettura distribuita che la J2EE supporta consente di ottenere un vantaggioso sistema di bilanciamento del carico di lavoro, permettendo la configurazione dei vari livelli dell'architettura per la gestione du server diversi.

4.2.3.3 Integrazione su sorgenti di informazione già esistenti

La J2EE platform, assieme alla J2SE platform, include una serie di *API* (*Application Program Interface*) standard per accedere alle varie sorgenti di informazione esistenti nell'impresa.
Si riporta l'elenco di queste API.

JDBC (*Java Database Connectivity*) è l'API per la connessione ai database relazionali. Essa fornisce un'interfaccia a livello di applicazione usata nel codice dei componenti per accedere al database in modo indipendente dal particolare DBMS utilizzato; sarà poi necessario utilizzare un driver specifico che si ponga come interfaccia tra l'API JDBC e lo specifico DBMS utilizzato. Grazie a questo meccanismo a due livelli è possibile mantenere il codice dei componenti dell'applicazione indipendente dal DBMS che può essere sostituito con un altro semplicemente cambiando il driver, e senza necessità di riscrittura del codice.

JTA (*Java Transaction API*) è l'API per la gestione e il coordinamento delle transazioni

attraverso sorgenti di informazioni transazionali eterogenee.

JNDI (*Java Naming and Directory Interface*) è l'API per accedere ad informazioni attraverso un sistema di nomi e percorsi, utilizzando così lo stesso meccanismo utilizzato per riferire i files nel File System.

JMS (*Java Message Service*) è l'API per inviare e ricevere messaggi attraverso sistemi di enterprise-messaging come l'IBM MQ Series e TIBCO Rendez-vous che richiedono l'attivazione del servizio attraverso un JMS provider.

JavaMail è l'API utilizzata dai componenti per inviare e ricevere email.

Java IDL è l'API per richiamare oggetti CORBA remoti, attraverso il protocollo IIOP. Questi oggetti sono tipicamente esterni alla J2EE e possono essere scritti in qualsiasi linguaggio.

Oltre a questi servizi che già sono presenti nell'attuale versione della J2EE platform, ve ne sono altri in via di sviluppo che saranno aggiunti nelle successive versioni attraverso l'architettura dei *Connectors*.

4.2.3.4 Scelta di servizi, tools e componenti

Il marchio J2EE è punto centrale per creare un mercato di servizi, tools e componenti tutti conformi allo stesso standard.
Le organizzazioni che sviluppano applicazioni J2EE possono contare su una varietà sempre crescente di fornitori di piattaforme J2EE con diverse possibilità di scelta sia a livello hardware, sia a livello di sistemi operativi e configurazioni di server, a seconda degli obiettivi e strategie adottate.
I vari componenti J2EE, con particolare riferimento agli EJB e alle JSP di cui si tratterà dettagliatamente in seguito, sono stati ideati per essere facilmente manipolati da tool grafici che consentono di automatizzare tanti dei tradizionali compiti richiesti, come ad esempio il debugging. Gli sviluppatori J2EE hanno così la possibilità di scegliere il tool conforme allo standard J2EE che meglio si addice alle loro esigenze.
Grazie al modello component-based si ha poi la possibilità di sviluppare componenti conformi allo standard che possono essere così riutilizzati da una qualsiasi applicazione J2EE.

4.2.3.5 Modello di sicurezza semplificato e unificato

Gli sviluppatori possono specificare i requisiti di sicurezza di un componente. Attraverso un sistema dichiarativo, sia gli EJB-component sia i Web-component possono specificare attraverso i deployment descriptor i criteri per associare i vari livelli di accesso a diversi ruoli e quindi alle diverse categorie di utenti. Per gli EJB esiste addirittura la possibilità di definire ruoli diversi per ciascun metodo.
Le specifiche J2EE incoraggiano l'utilizzo di questo sistema dichiarativo per consentire al codice dei componenti di rimanere totalmente indipendente dai vincoli legati alla sicurezza. Ciò non sempre è possibile, poiché in taluni casi è necessario creare livelli e vincoli di sicurezza che non possono essere espressi con un semplice mapping tra ruoli e utenti. Perciò viene comunque mantenuta la possibilità di inserire security-checks anche all'interno del codice attraverso opportune API.

4.3 Realizzazione

In questa sede si descrive in dettaglio la struttura dell'applicazione, l'interfaccia utente e la logica delle pagine che costituiscono l'applicazione. Durante la realizzazione si è cercato di sfruttare al meglio le opportunità fornite dal linguaggio Java/JSP e, in modo particolare, si è sempre seguita la strada della semplicità.

Per quanto riguarda il database progettato, durante le realizzazione delle pagine si è constatato che le scelte effettuate sono sempre risultate vincenti ed hanno permesso di gestire semplicemente le informazioni memorizzate.

Per gestire i progetti censiti su ChangeMan DS, l'applicazione effettua "lanci" di batch DOS opportunamente preparati, che contengono gli script corrispondenti alle azioni del processo da invocare.

Infine, all'interno delle pagine realizzate sono stati inseriti tutti i vincoli e le regole che la prima parte di progettazione non comprendeva ma che sono considerati fondamentali per il corretto funzionamento dell'intero progetto.

4.3.1 Struttura dell'applicazione

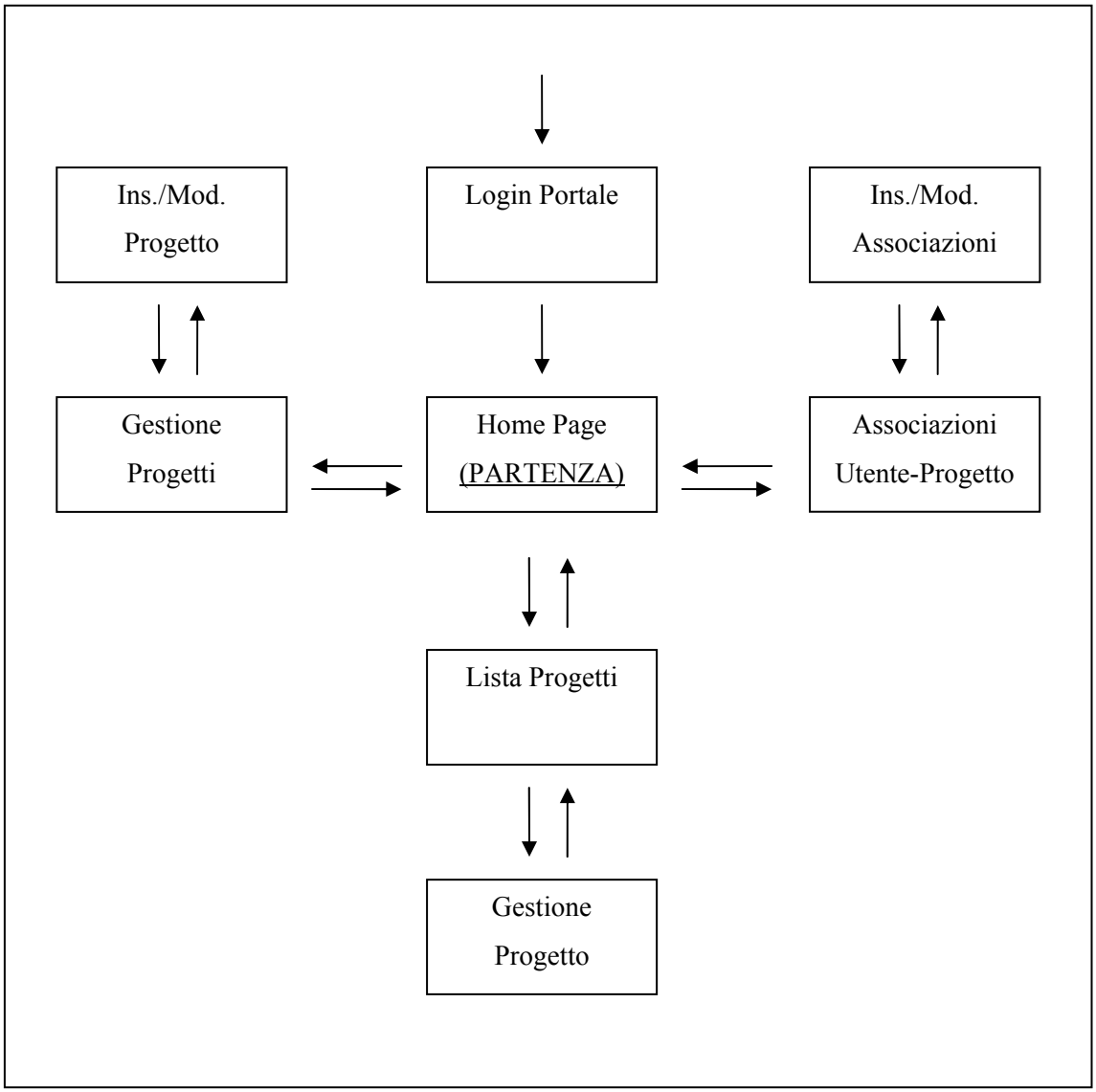


Figura 11

Questa è la struttura generale dell'interfaccia Web, ma si ricorda che solo pochissimi utenti possono accedere ad essa nella sua totalità.

4.3.2 L’interfaccia Utente

In questo paragrafo vengono presentate le interacce grafiche realizzate, cioè la parte visualizzata dall’utente attraverso il proprio browser.
Data l’elevata criticità delle operazioni attivate dall’applicativo, sono stati realizzati controlli software che, in fase di elaborazione delle stesse, non consentano all’utente di cliccare più volte sullo stesso bottone.
La scelta dei colori è stata effettuata con l’intento di renderli armoniosi con il Portale aziendale che “ospita” l’applicativo.

4.3.2.1 La Home Page

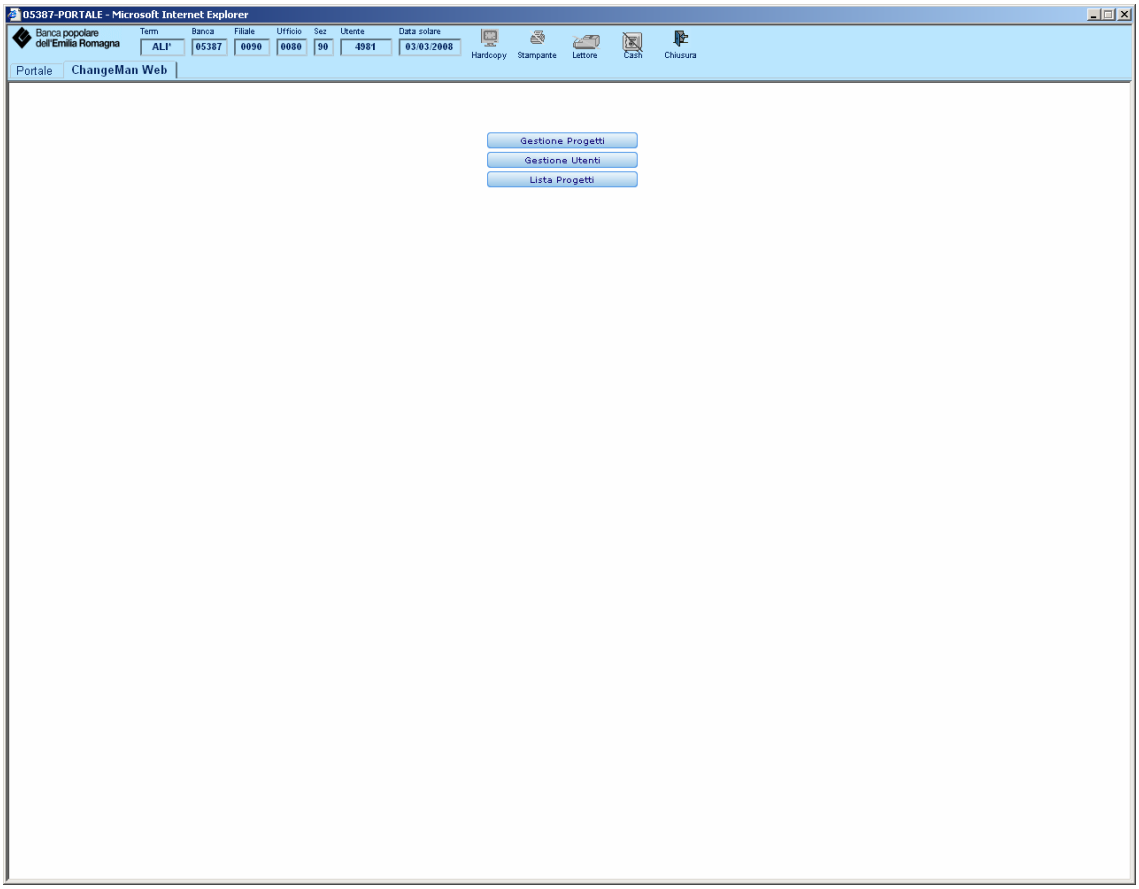


Figura 12

Questa è la maschera iniziale dell'utente con profilo di amministratore, dalla quale si può accedere alle sezioni "Gestione Progetti", "Gestione Utenti" e "Lista Progetti"; gli utenti non amministratori sono indirizzati direttamente alla sezione"Lista Progetti".
In seguito si descrivono dettagliatamente le tre sezioni.

4.3.2.2 La Pagina Gestione Progetti

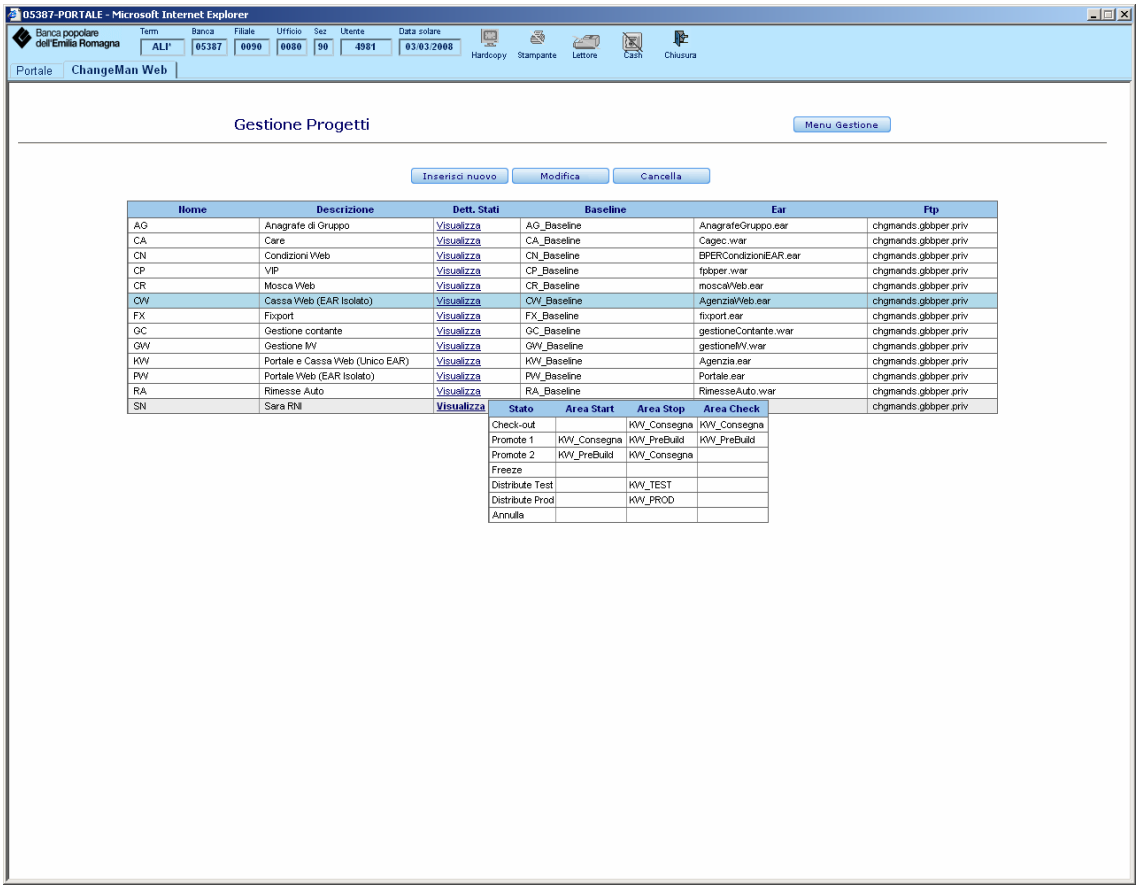


Figura 13

Questa sezione, alla quale può accedere soltanto l'utente amministratore, consente di inserire, modificare e cancellare i progetti. In particolare per ogni progetto vengono visualizzate le seguenti informazioni:

Nome: codice univoco lungo due caratteri che identifica il progetto, corrispondente a quello censito su ChangeManDS

Descrizione: breve descrizione reperita da ChangeManDS

Dati Attività: passando con il mouse sulla parola "Visualizza" del campo vengono mostrate le attività previste per il progetto, precisando per ognuna di esse le eventuali aree di ChangeManDS coinvolte

Baseline: nome repository di ChangeManDS in cui sono memorizzati l'EAR/WAR e l'eventuale pacchetto contenente i DBRM, per convenzione si chiama XX_Baseline, dove con XX si intende il nome del progetto

Ear: nome fisico del file EAR/WAR

Ftp: indirizzo del server Ftp che punta all'Area di Consegna in cui l'utente può caricare il nuovo EAR/WAR da installare.

4.3.2.3 La Pagina Dati Progetto

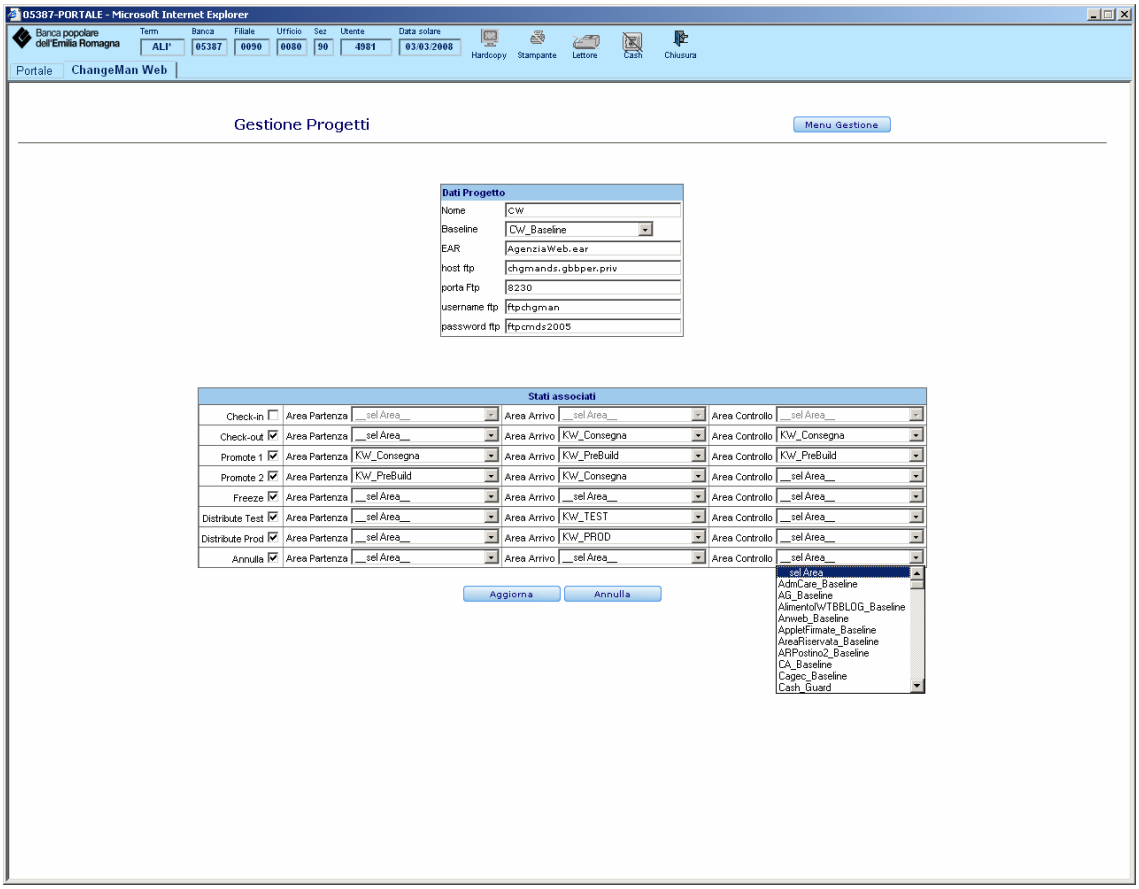


Figura 14

Questa pagina, alla quale si può accedere per inserire o modificare un progetto, consente di associare ad esso tutte le informazioni necessarie per la gestione.

Sono presenti due sezioni, una con i dati del progetto precedentemente illustrati, e l'altra in cui si possono selezionare le attività, tra quelle previste, che l'utente può effettuare, permettendo di personalizzare le aree di ChangeManDS coinvolte.

Per ogni attività, se necessario, occorre inserire l'area di partenza dei file di progetto, l'area di prima destinazione e quella di controllo finale per verificare se il processo è stato completato.

Si descrivono in seguito le attività previste:

Check-In: attività per portare il progetto in utilizzo allo stato di Check-In, tale passaggio è stato previsto per implementazioni future, ma nel processo che stiamo descrivendo è compresa nell'attività Promote 2

Check-Out: attività per prendere in lavorazione il progetto ed aggiornarlo. Quando un progetto è in lavorazione da parte di un utente viene bloccato, ovvero nessun altro può aggiornarlo contemporaneamente finché non è stato riportato allo stato di Check-In. L'area di partenza corrisponde alla Baseline del progetto, pertanto non occorre specificarla, mentre quella di arrivo è la KW_Consegna, in cui l'utente potrà inserire il nuovo EAR/WAR

Promote 1: questa attività, che porta il progetto dall'area KW_Consegna alla KW_Prebuild, scatena i processi che consentono si verificare le differenze rispetto al progetto attualmente nelle Baseline, e la preparazione degli eventuali DBRM ed EAR/WAR se è previsto SQL statico.

Promote 2: questa attività, che porta il progetto dall'area KW_Prebuild alla KW_Consegna, conclude la fase di aggiornamento sovrascrivendo la Baseline attuale con i file nuovi, e riportando il progetto allo stato Check-In.

Freeze: questa attività è necessaria per "congelare" la nuova versione del progetto e consentire la distribuzione dello stesso negli ambienti di test o di produzione

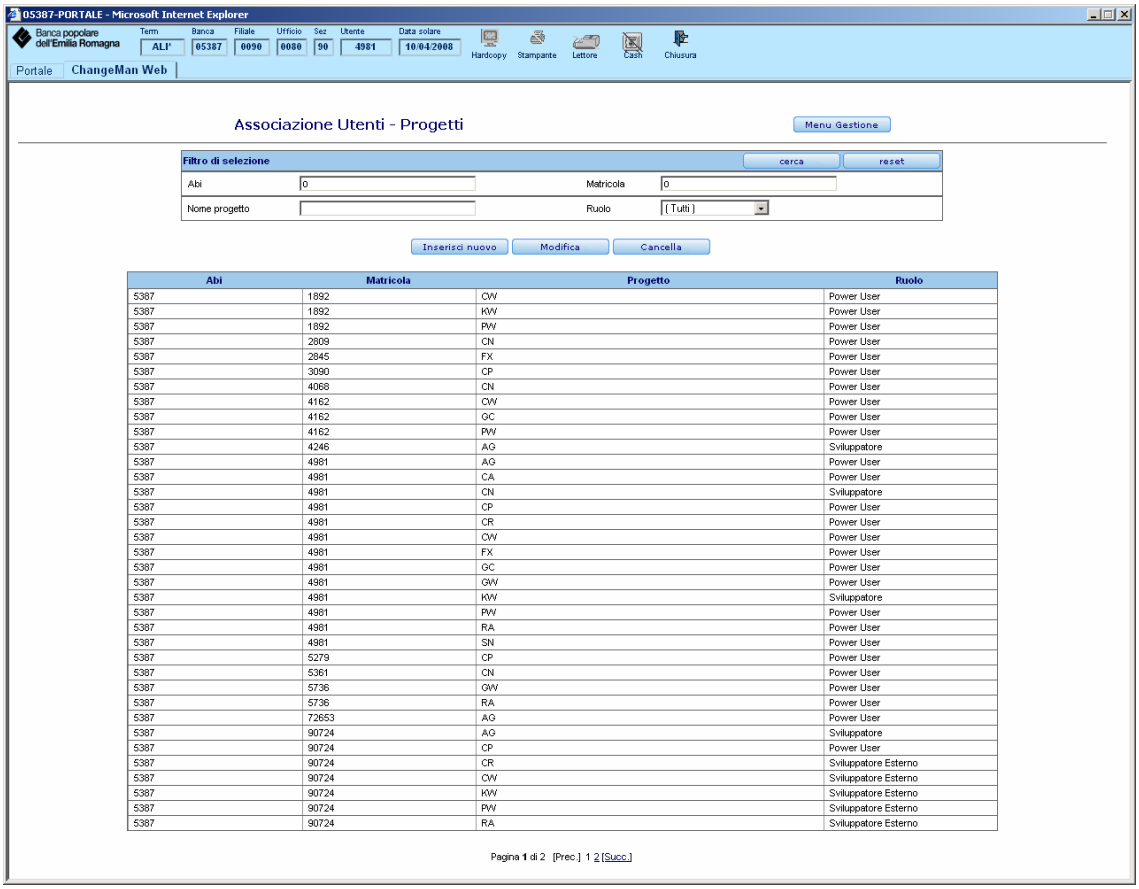
Annulla: consente di interrompere la fase di aggiornamento e riportare il progetto allo stato Check-In.

Distribute Test: consente di distribuire il nuovo pacchetto in ambiente di test (il deploy ed il bind saranno effettuati da un apposito piano automaticamente)

Distribute Prod: consente di distribuire il nuovo pacchetto in ambiente di produzione (il deploy ed il bind saranno effettuati da un apposito piano automaticamente)

In seguito è mostrata la pagina a disposizione dell'utente per effettuare le attività appena descritte.

4.3.2.4 La Pagina Associazione Utenti-Progetti



Abi	Matricola	Progetto	Ruolo
5387	1892	CW	Power User
5387	1892	KW	Power User
5387	1892	PW	Power User
5387	2809	CN	Power User
5387	2845	FX	Power User
5387	3090	CP	Power User
5387	4068	CN	Power User
5387	4162	CW	Power User
5387	4162	GC	Power User
5387	4162	PW	Power User
5387	4246	AG	Sviluppatore
5387	4981	AG	Power User
5387	4981	CA	Power User
5387	4981	CN	Sviluppatore
5387	4981	CP	Power User
5387	4981	CR	Power User
5387	4981	CW	Power User
5387	4981	FX	Power User
5387	4981	GC	Power User
5387	4981	GW	Power User
5387	4981	KW	Sviluppatore
5387	4981	PW	Power User
5387	4981	RA	Power User
5387	4981	SN	Power User
5387	5279	CP	Power User
5387	5361	CN	Power User
5387	5736	GW	Power User
5387	5736	RA	Power User
5387	72653	AG	Power User
5387	90724	AG	Sviluppatore
5387	90724	CP	Power User
5387	90724	CR	Sviluppatore Esterno
5387	90724	CW	Sviluppatore Esterno
5387	90724	KW	Sviluppatore Esterno
5387	90724	PW	Sviluppatore Esterno
5387	90724	RA	Sviluppatore Esterno

Figura 15

Questa sezione, anch'essa a disposizione solo dell'amministratore, consente di associare gli utenti ai progetti, ovvero di stabilire chi può aggiornarli e/o installarli.
Ad un utente deve essere associato un ruolo per ogni progetto su cui ha visibilità, tra i seguenti previsti:

Power User: può effettuare tutte le attività previste
Sviluppatore: può effettuare tutte le attività previste, tranne la distribuzione in ambiente di produzione
Sistemista: può distribuire le freeze dei progetti in negli ambienti di test e di produzione

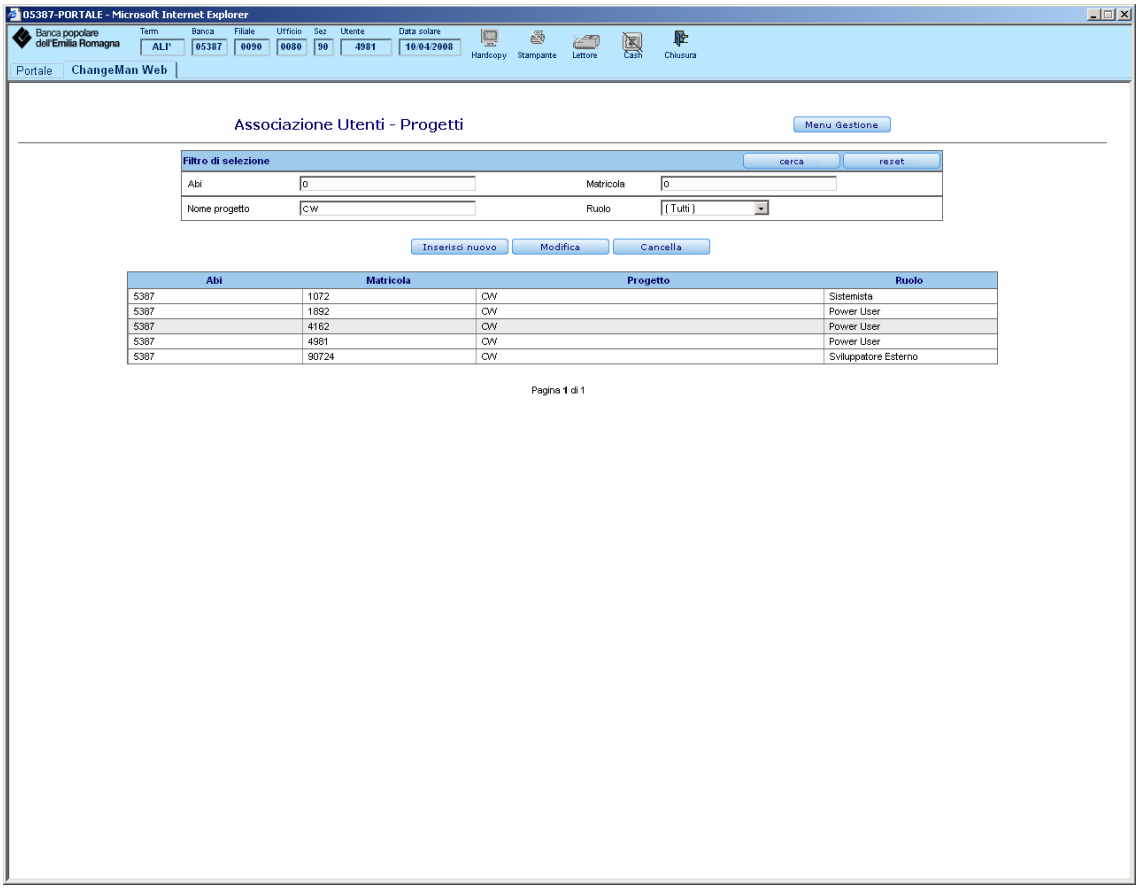


Figura 16

In questa videata viene evidenziata la possibilità di effettuare una ricerca applicando filtri sull'Abi, sul Nome del progetto, sulla Matricola aziendale e sul Ruolo dell'utente.

4.3.2.5 La Pagina Inserimento Associazione Utente-Progetto

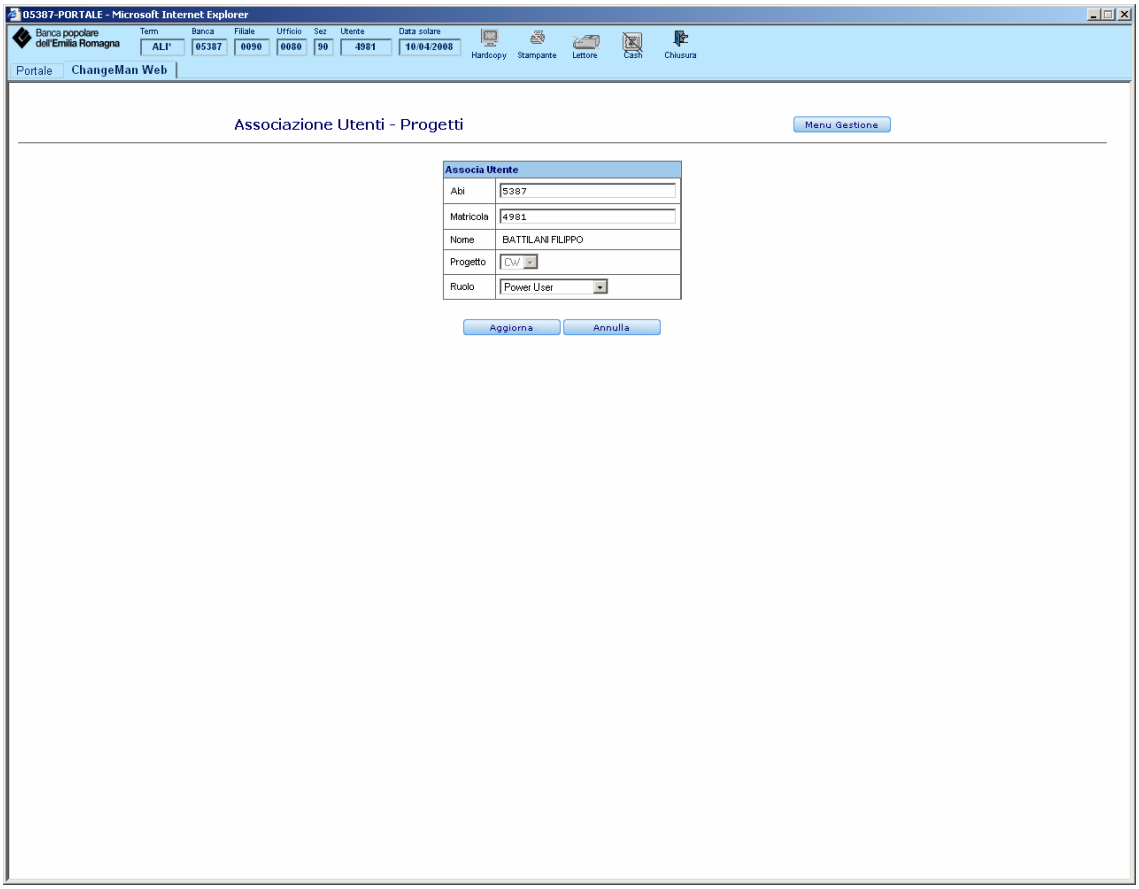
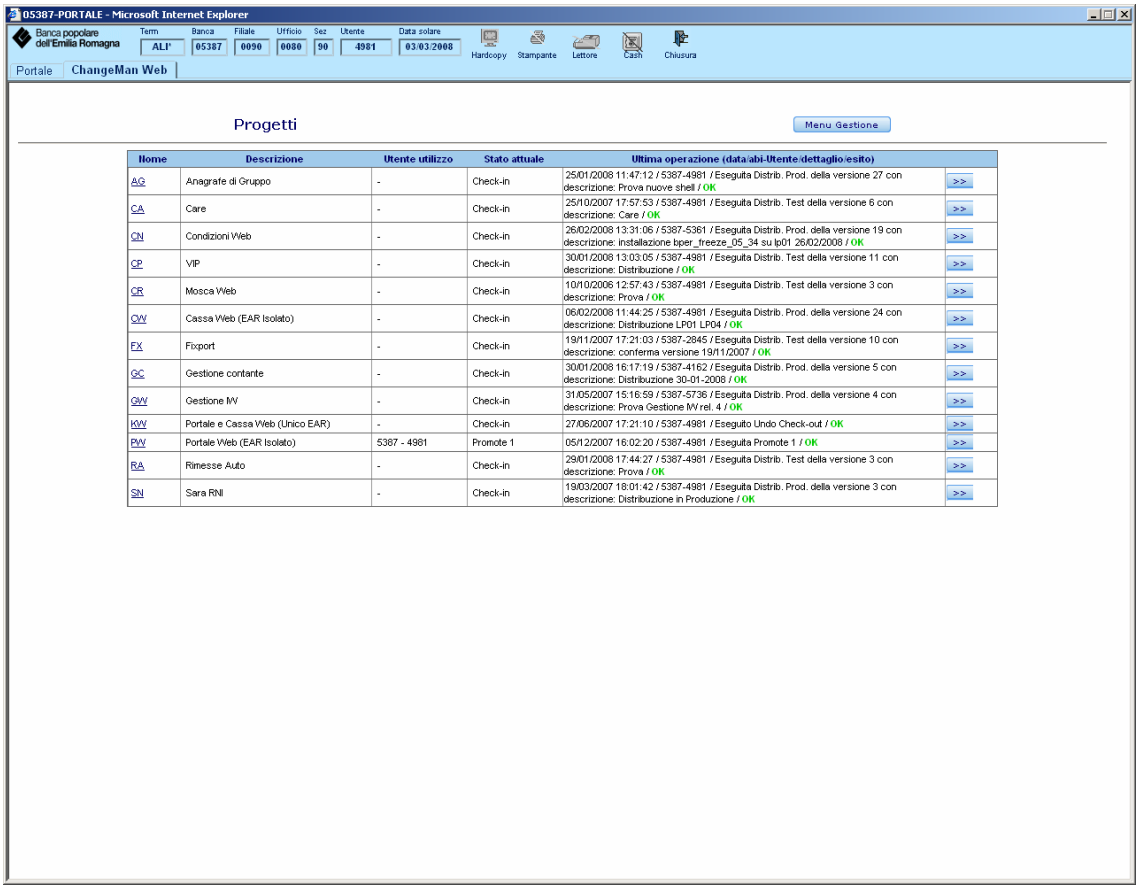


Figura 17

Questa è la pagina in cui l'amministratore può inserire/modificare un'associazione utente-progetto. Dopo l'inserimento dall'Abi e della Matricola, vengono restituiti in automatico nome e cognome dell'utente tramite una chiamata ad un servizio web aziendale che accede all'anagrafe generale, in seguito occorre inserire il nome del progetto ed il ruolo stabilito.

4.3.2.6 La Pagina Lista Progetti



Nome	Descrizione	Utente utilizzo	Stato attuale	Ultima operazione (data/abi-Utente/dettaglio/esito)	
AG	Anagrafe di Gruppo	-	Check-in	25/01/2008 11:47:12 / 5387-4981 / Eseguita Distrib. Prod. della versione 27 con descrizione: Prova nuove shell / OK	>>
CA	Care	-	Check-in	25/10/2007 17:57:53 / 5387-4981 / Eseguita Distrib. Test della versione 6 con descrizione: Care / OK	>>
CN	Condizioni Web	-	Check-in	26/02/2008 13:31:06 / 5387-5361 / Eseguita Distrib. Prod. della versione 19 con descrizione: installazione biper_freeze_05_34 su lp01 26/02/2008 / OK	>>
CP	VIP	-	Check-in	30/01/2008 13:03:05 / 5387-4981 / Eseguita Distrib. Test della versione 11 con descrizione: Distribuzione / OK	>>
CR	Mosca Web	-	Check-in	10/10/2006 12:57:43 / 5387-4981 / Eseguita Distrib. Test della versione 3 con descrizione: Prova / OK	>>
CW	Cassa Web (EAR Isolato)	-	Check-in	06/02/2008 11:44:25 / 5387-4981 / Eseguita Distrib. Prod. della versione 24 con descrizione: Distribuzione LP01 LP04 / OK	>>
FX	Fixport	-	Check-in	19/11/2007 17:21:03 / 5387-2845 / Eseguita Distrib. Test della versione 10 con descrizione: conferma versione 19/11/2007 / OK	>>
GC	Gestione contante	-	Check-in	30/01/2008 16:17:19 / 5387-4162 / Eseguita Distrib. Prod. della versione 5 con descrizione: Distribuzione 30-01-2008 / OK	>>
GW	Gestione IW	-	Check-in	31/05/2007 15:16:59 / 5387-5736 / Eseguita Distrib. Prod. della versione 4 con descrizione: Prova Gestione IW rel. 4 / OK	>>
KW	Portale e Cassa Web (Unico EAR)	-	Check-in	27/06/2007 17:21:10 / 5387-4981 / Eseguito Undo Check-out / OK	>>
PW	Portale Web (EAR Isolato)	5387 - 4981	Promote 1	05/12/2007 16:02:20 / 5387-4981 / Eseguita Promote 1 / OK	>>
RA	Rimesse Auto	-	Check-in	29/01/2008 17:44:27 / 5387-4981 / Eseguita Distrib. Test della versione 3 con descrizione: Prova / OK	>>
SN	Sara RNI	-	Check-in	19/03/2007 18:01:42 / 5387-4981 / Eseguita Distrib. Prod. della versione 3 con descrizione: Distribuzione in Produzione / OK	>>

Figura 18

Questa sezione può essere consultata da tutti gli utenti, i quali possono visualizzare tutti i progetti su cui l'amministratore ha dato loro visibilità.
Le informazioni disponibili per ogni progetto sono le seguenti:

Nome progetto
Descrizione progetto
Utente Utilizzo: eventuale utente (Abi e Matricola) che sta aggiornando il progetto
Stato Attuale: possibili Check-In, Check-Out e Promote 1
Ultima Operazione Eseguita: evidenziati data, abi e matricola dell'utente, dettaglio dell'esito

Cliccando sul bottone azzurro di ogni progetto si può entrare nella pagina di lavorazione, di cui in seguito si descrivono dettagliatamente le fasi.

4.3.2.7 La Pagina Progetto

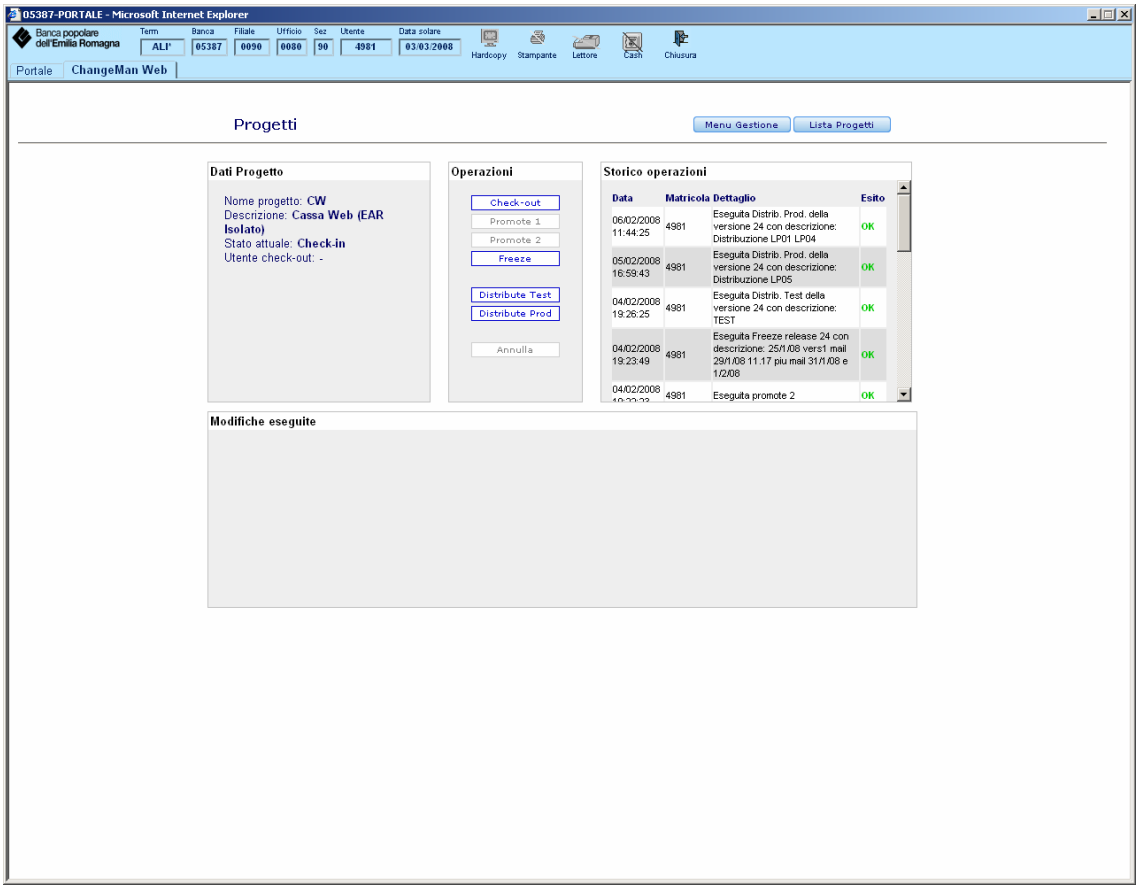


Figura 19

La pagina mostra tre riquadri, a sinistra i dati del progetto, al centro i bottoni in sequenza per effettuare le attività previste, ed a destra lo storico delle ultime attività eseguite.

In seguito si mostra la sequenza delle operazioni che l'utente effettua per aggiornare ed installare un progetto.

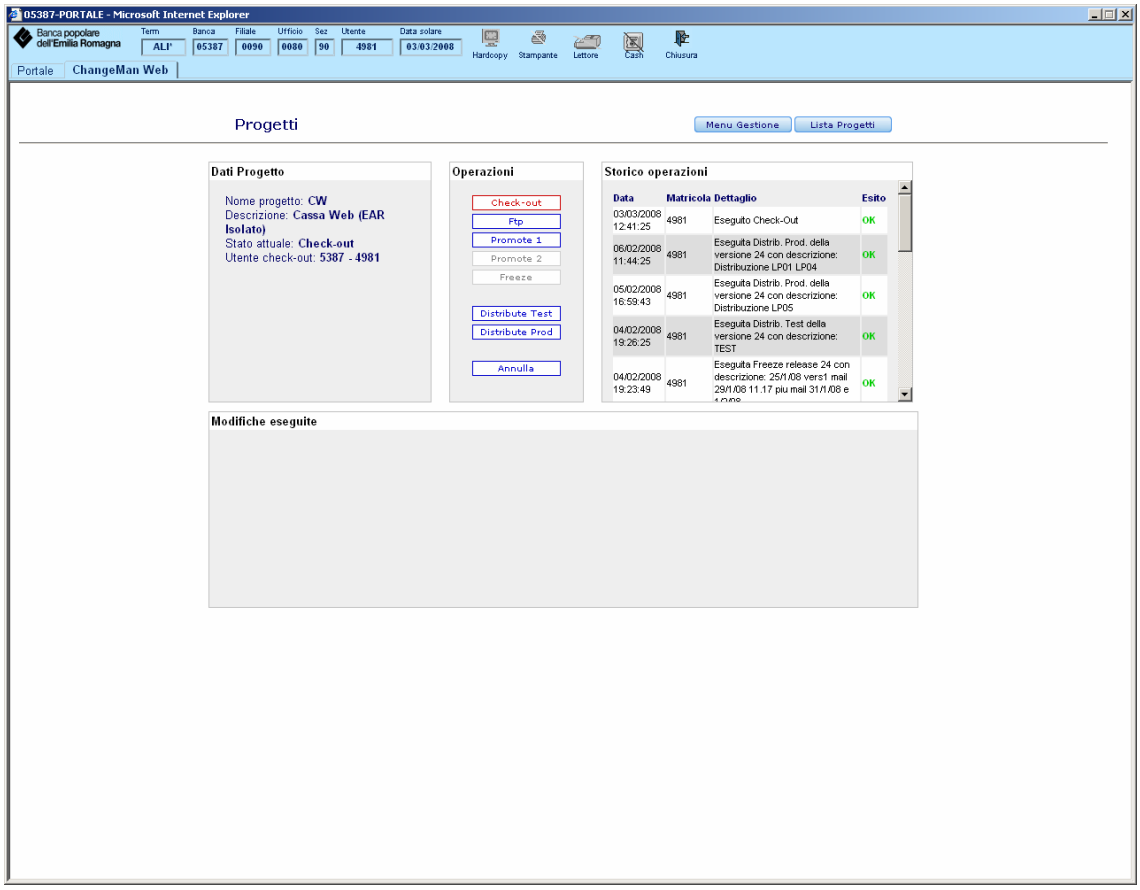


Figura 20

Dopo avere effettuato la fase Check-Out del progetto viene evidenziato in rosso il bottone associato, per indicare che tale operazione è stata completata. In automatico compare il bottone **Ftp**.

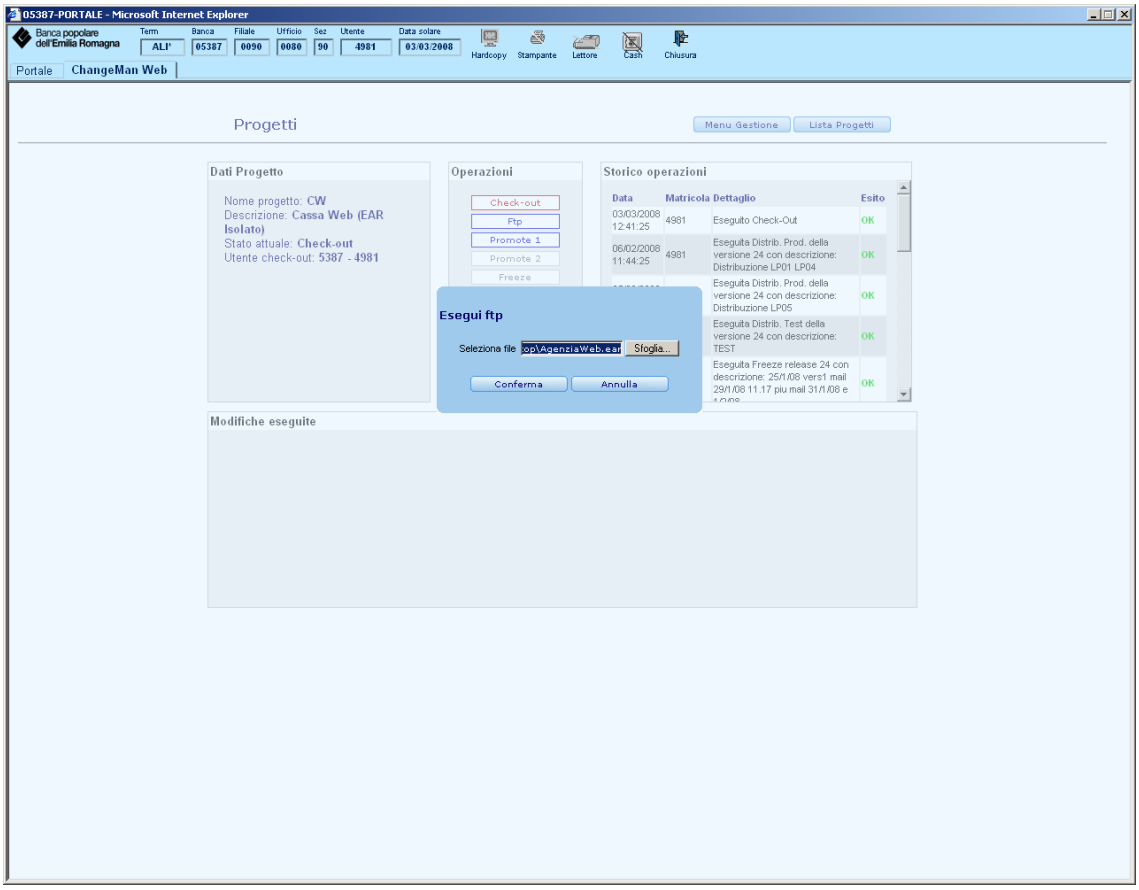


Figura 21

Cliccando sul bottone **Ftp** viene aperta una maschera per cercare sul proprio file system il nuovo EAR/WAR, dopo aver confermato l'applicazione controlla che il nome del file corrisponda con quello associato al progetto e, in caso affermativo, effettua un Ftp sul server indicato, che punta all'area KW_Consegna. Questa operazione consente di inserire il nuovo EAR/WAR nel processo di aggiornamento.

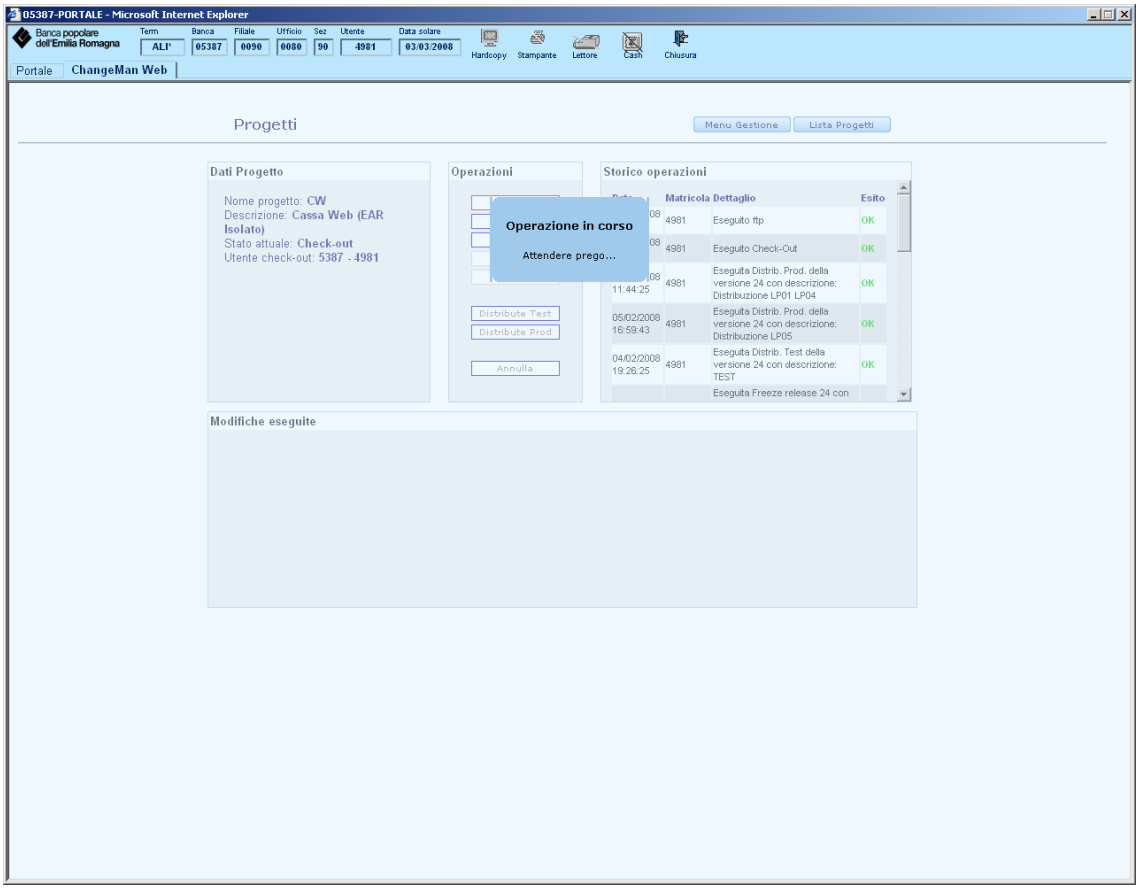


Figura 22

A questo punto l'utente può procedere con la Promote 1.

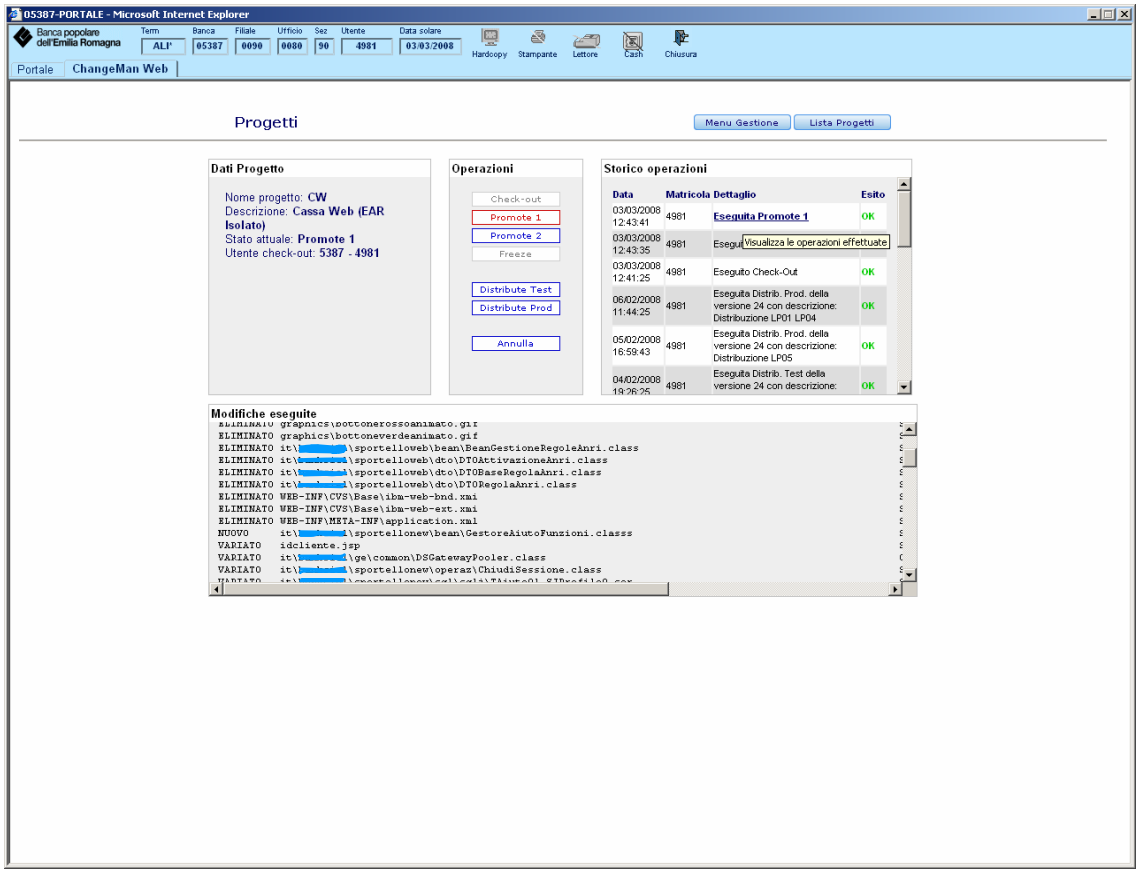


Figura 23

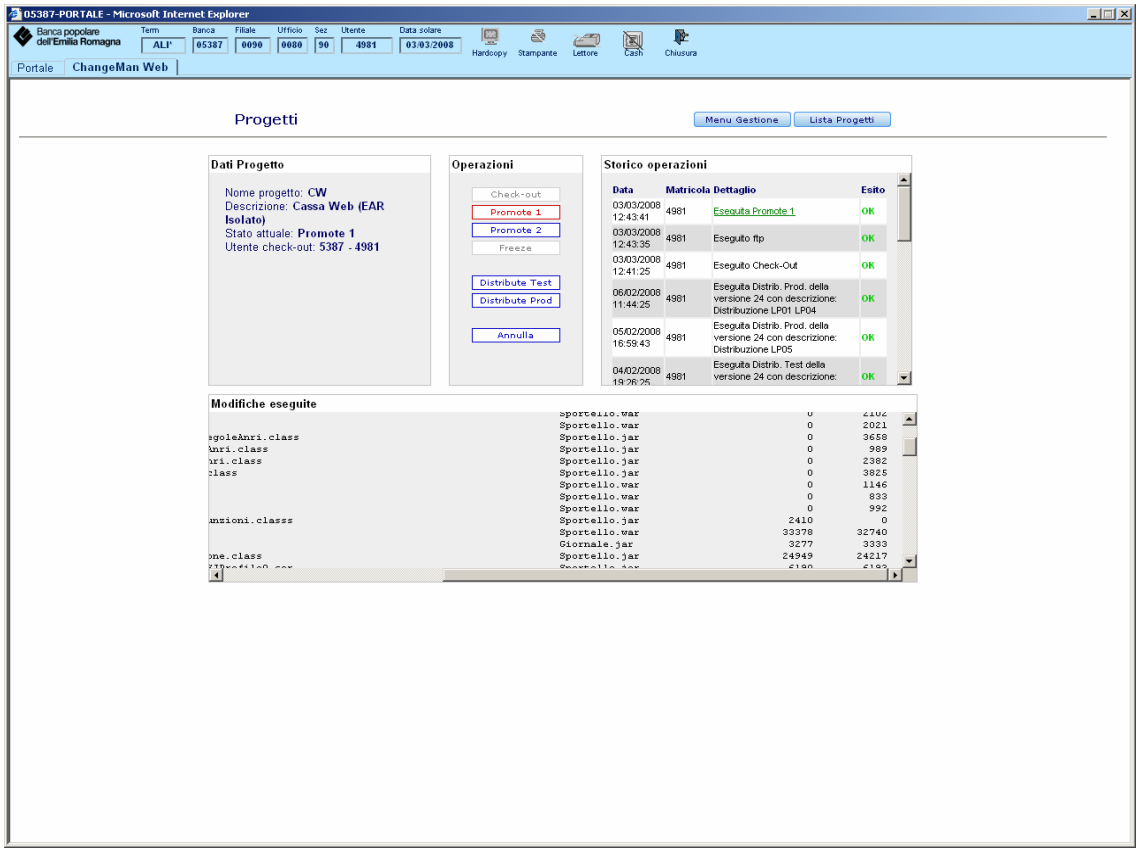


Figura 24

Terminata la fase di Promote 1 è disponibile nello storico il link **EseguitaPromote1**, che permette di visualizzare nel riquadro più in basso l'elenco dei file aggiunti, variati ed eliminati. Per ognuno di questi file viene visualizzato il path, il contenitore (che può essere un modulo War, Jar ecc…), e la dimensione nella vecchia e nella nuova versione.

Tali informazioni possono risultare utili per verificare l'entità della modifica, soprattutto in caso di sviluppo del progetto da parte di una società esterna.

Se l'utente ritiene che le modifiche evidenziate siano corrette, può procedere con la fase Promote 2 e riportare il progetto aggiornato in stato Check-In, altrimenti può annullare l'aggiornamento cliccando sull'apposito pulsante.

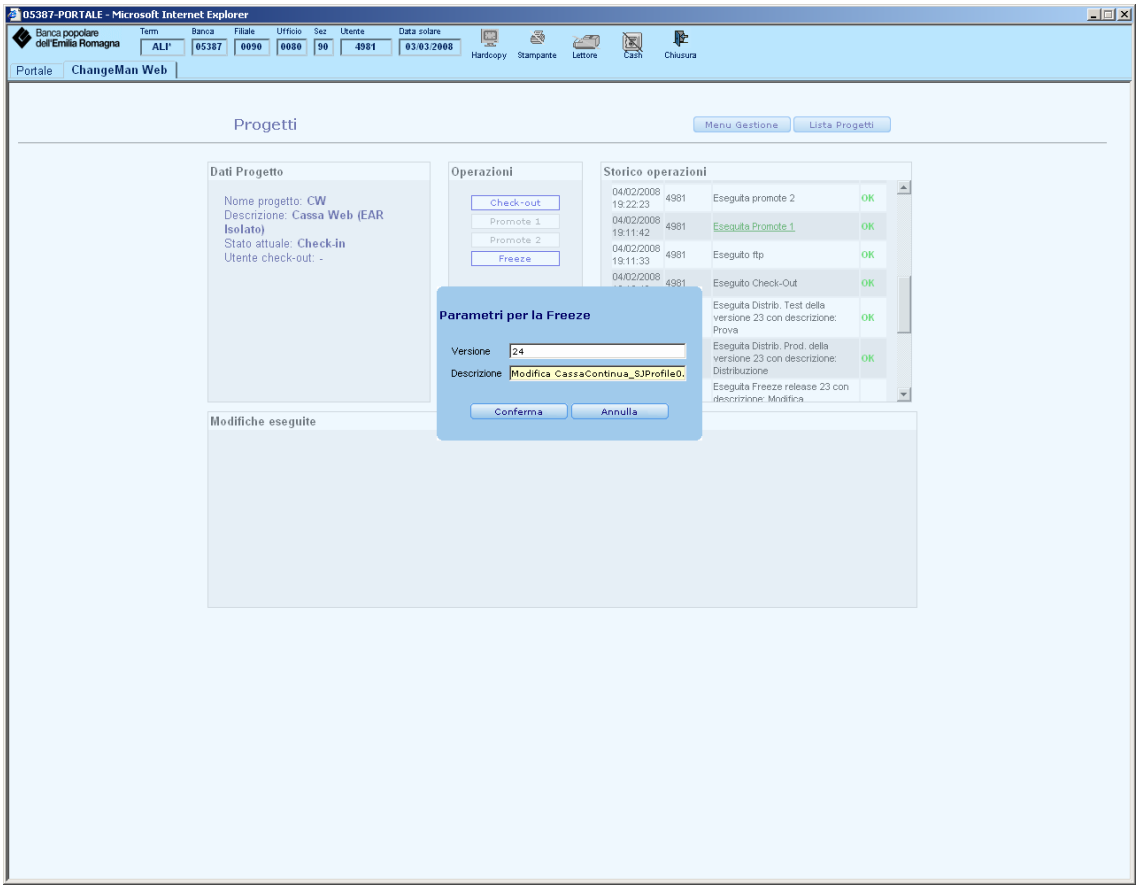


Figura 25

Cliccando sul bottone **Freeze** si apre una maschera che propone la nuova versione del progetto, e consente di inserire una descrizione in cui indicare i dati identificativi di tale versione come, ad esempio, le modifiche più significative apportate. Tale operazione è strettamente necessaria in quanto è possibile distribuire ed installare soltanto le Freeze.

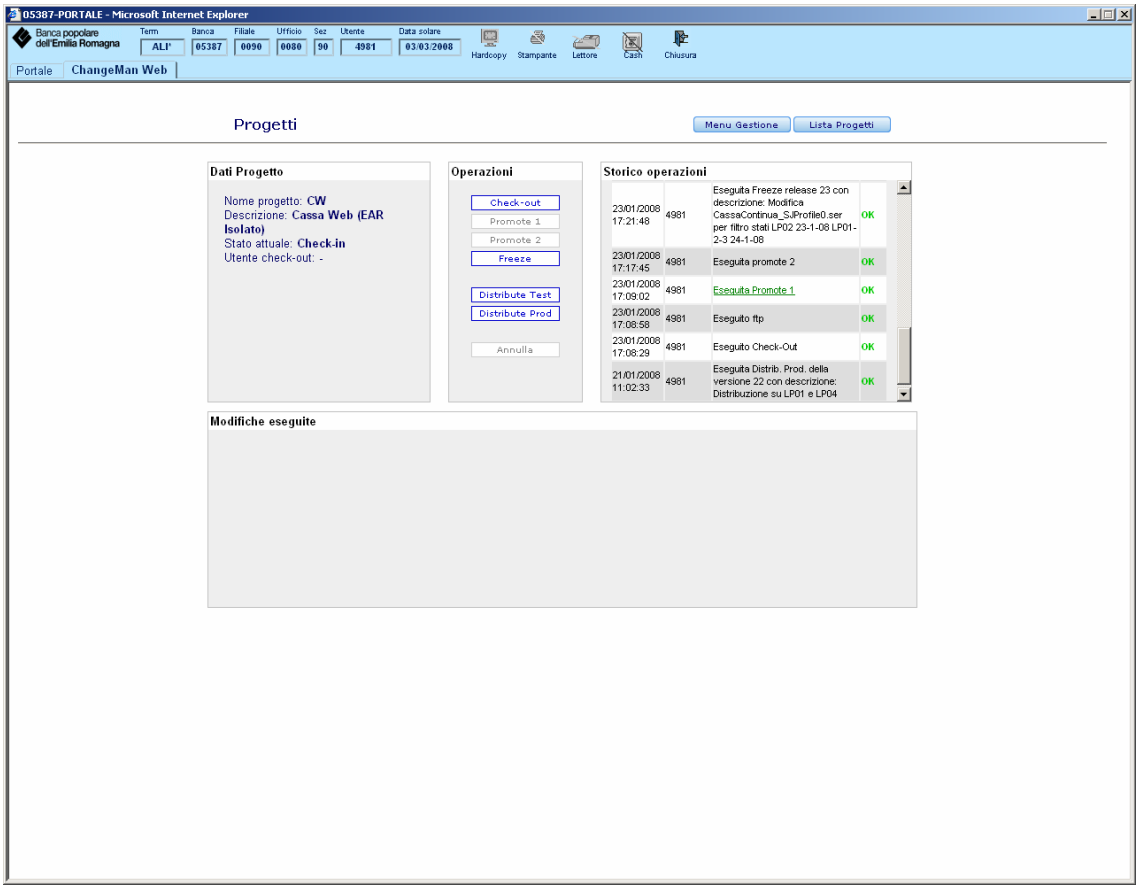


Figura 26

L'immagine sopra raffigurata evidenzia la composizione dello storico per le prime fasi finora descritte. Per aiutare l'utente che vuole ricostruire la storia delle installazioni di un determinato progetto, sono stati inseriti per ogni operazione le seguenti informazioni:

data e ora di esecuzione
matricola dell'utente che l'ha effettuata
descrizione
esito, positivo (OK)o negativo (KO)

CONCLUSIONI E SVILUPPI FUTURI

In questa sede vengono tratte le conclusioni sul lavoro svolto, evidenziando i vantaggi e le possibilità di sviluppo che caratterizzano lo strumento realizzato.

E' importante sottolineare come il requisito di **semplicità d'utilizzo** richiesto sia stato raggiunto: l'interfaccia Web rende lo strumento immediatamente utilizzabile da parte di qualsiasi utente, che può in questo modo operare in maniera autonoma anche se non conosce nel dettaglio l'ambiente su cui è installata l'applicazione.

La **sicurezza** del sistema è garantita mediante il sistema di autenticazione dell'utente all'interno del Portale aziendale integrato con RACF, per impedire l'esecuzione di attività indesiderate da parte di operatori non autorizzati, ed integrandosi nel sistema informativo esistente. Naturalmente è possibile definire più profili utente differenziando i privilegi all'interno di RACF, in questo modo gli utenti potranno accedere soltanto alle funzionalità di competenza (ad esempio, un utente può aggiornare un progetto ed installarlo in ambiente di test, mentre un altro non può aggiornarlo ma può installarlo sia in ambiente di test sia in produzione). L'utente con profilo di amministrazione può a sua volta gestire l'associazione utenti-progetti, ovvero consentire la visibilità dei progetti ai soli utenti indicati.

Per quanto detto, è possibile affermare che lo strumento realizzato risponde alle specifiche iniziali del progetto.

L'utilizzo dello strumento è già iniziato all'interno dell'azienda e, attualmente, l'applicazione è utilizzata sia dalle persone dell'Ufficio Analisi e Pianificazione sia da quelle dell'Ufficio Sistemisti. Il giudizio pervenuto dagli utilizzatori dello strumento è stato positivo per i vantaggi sopra citati, ed anche per la possibilità di avere a disposizione la "storia" delle versioni e delle installazioni degli applicativi Web.

Oltre ai vantaggi direttamente percepibili da parte degli utenti utilizzatori, lo strumento ha sicuramente reso più veloce e sicuro il processo di installazione, risparmiando il tempo di installazione manuale e soprattutto evitando possibili errori umani di configurazione.

Dati i buoni risultati ottenuti l'azienda ha ipotizzato di **estendere l'uso del sistema anche ad applicazioni installate su altre piattaforme**, e tale attività potrà essere effettuata

senza stravolgere il lavoro svolto, in quanto il progetto è stato realizzato in modo parametrico e modulare.

APPENDICE 1 - COME IL PORTALE CHIAMA LE APPLICAZIONI CHE LANCIA

Esempio

```
http://host/context/page?ABI=xxxx&FILIALE=xxxx&UFFICIO=xxxx&SEZI
ONE=xx&TERMID=xxxx&UTENTE=xxxx&RUOLO=&NOTE=&FRAME=window.top.nul
l&FRAMESERVIZI=window.top.null&IDAPPL=CHANGEMANWEB&HOST=xxxx&POR
TA=xx&CONTESTO=/portale&SERVLETCHIUSURA=PortaleServletChiudiAppl
Remota&TOKEN=3857476C7063624A30574854355065596B64666B6C5066552B4
F505A706E352B46513D3D0A&LUOGO=DESCRIZIONE_UFFICIO_UTENTE
```

L'applicazione lanciata deve estrapolare alcune informazioni ricevute in coda all'url e usarle per comporre un tracciato "xml" per richiamare l'url del portale, che verifica e risponde se è stato proprio lui a effettuare la chiamata.

Segue la descrizione puntuale del processo di validazione.

Servizi del Portale mediante chiamate a servlet

Il portale mette a disposizione delle applicazioni servizi mediante i quali:

- verificare che l'applicazione sia stata attivata dal portale;
- richiedere informazioni relative ad un terminale;
- richiede informazioni relative ad una applicazione attiva;
- richiedere informazioni relativa ad un terminale ed alla applicazione.

I servizi sono disponibili sia mediante:

- **istanziazione di classi Java e chiamate al metodo di gestione del servizio** per le applicazioni dello stesso contesto del portale;
- **chiamata http ad una servlet del portale con passaggio dati in formato xml** per le applicazioni non java oppure che non siano nello stesso contesto del portale.

Si descrivono i servizi disponibili mediante chiamate http a servlet del portale.

Il passaggio dati avviene in formato xml ed ha le seguenti caratteristiche:

- i dati sono racchiusi fra il tag PORTALE (es. **<PORTALE>**…..**</PORTALE>**);
- i tag in risposta possono essere o meno presenti, e, se presenti, possono essere vuoti;

- non è prevista la gestione di attributi dei tag;
- in risposta il tag ESITO è obbligatorio, e se diverso da zero indica un errore: in questo caso gli altri tag possono essere presenti in parte oppure del tutto assenti.

Validazione avvio applicazione, prima chiamata

Il servizio esegue la validazione dell'applicazione la prima volta in assoluto (tipicamente all'avvio); richiede l'attivazione della servlet:

```
/portale/PortaleServletControllo
```

fornendo una stringa xml contenente i parametri (obbligatori) descritti nella tabella seguente:

NOME TAG	VALORE DEL TAG
TIPOCONTROLLO	Fisso al valore **A**
BANCA	fornito dal portale all'avvio dell'applicazione come parametro *ABI*
TERMID	fornito dal portale all'avvio dell'applicazione come parametro *TERMID*
TOKEN	fornito dal portale all'avvio dell'applicazione come parametro *TOKEN*
IDAPPL	fornito dal portale all'avvio dell'applicazione come parametro *IDAPPL*

e ritorna la stringa xml contenente i parametri descritti nella tabella seguente:

NOME TAG	VALORE DEL TAG
TIPOCONTROLLO	Fisso al valore **A**
BANCA	Valore fornito in fase di chiamata
TERMID	Valore fornito in fase di chiamata
TOKEN	Valore fornito in fase di chiamata
APPLID	Valore fornito in fase di chiamata
ESITO	Codice che identifica l'esito della validazione; assume i seguenti significati: • **0**: verifica conclusa correttamente; • **altri valori**: verifica non conclusa correttamente, il tag MESSAGGIO contiene una descrizione dell'errore riscontrato.
MESSAGGIO	Descrizione dell'errore

Si rammenti che questo servizio può essere richiamato solamente una volta, normalmente all'avvio dell'applicazione.

Esempio di richiesta validazione:

```
<PORTALE><TIPOCONTROLLO>A</TIPOCONTROLLO><BANCA>05387</BANCA><ID
APPL>A17</IDAPPL><TERMID>WEB*</TERMID><TOKEN>c2gvUFF1M05wYjg0Z0s
xVnA4NndqZz09DQo=</TOKEN></PORTALE>
```

Esempio di risposta alla richiesta precedente:

```
<PORTALE><TIPOCONTROLLO>A</TIPOCONTROLLO><BANCA>05387</BANCA><TE
RMID>WEB*</TERMID><IDAPPL>A17</IDAPPL><TOKEN>c2gvUFF1M05wYjg0Z0s
xVnA4NndqZz09DQo=</TOKEN><ESITO>0</ESITO><MESSAGGIO></MESSAGGIO>
</PORTALE>
```

Validazione avvio applicazione, chiamate successive

Il servizio che segue esegue la validazione dell'applicazione le volte successive alla prima; richiede l'attivazione della servlet:

```
/portale/PortaleServletControllo
```

fornendo una stringa xml contenente i parametri (obbligatori) descritti nella tabella seguente:

NOME TAG	VALORE DEL TAG
TIPOCONTROLLO	Fisso al valore **S**
BANCA	fornito dal portale all'avvio dell'applicazione come parametro *ABI*
TERMID	fornito dal portale all'avvio dell'applicazione come parametro *TERMID*
TOKEN	fornito dal portale all'avvio dell'applicazione come parametro *TOKEN*
APPLID	fornito dal portale all'avvio dell'applicazione come parametro *IDAPPL*

e ritorna la stringa xml contenente i parametri descritti nella tabella seguente:

NOME TAG	VALORE DEL TAG
TIPOCONTROLLO	Fisso al valore **S**
BANCA	Valore fornito in fase di chiamata
TERMID	Valore fornito in fase di chiamata
TOKEN	Valore fornito in fase di chiamata
APPLID	Valore fornito in fase di chiamata
ESITO	Codice che identifica l'esito della validazione; assume i seguenti valori: • **0**: verifica conclusa correttamente; • **altri valori**: verifica non conclusa correttamente, il tag MESSAGGIO contiene una descrizione dell'errore riscontrato.
MESSAGGIO	Descrizione dell'errore

Si rammenti che questo servizio può essere richiamato più volte dall'applicazione, ma solamente dopo avere fatto la prima chiamata di validazione descritta nel paragrafo precedente.

Esempio di richiesta di validazione:

```
<PORTALE><TIPOCONTROLLO>S</TIPOCONTROLLO><BANCA>05387</BANCA><ID
APPL>A17</IDAPPL><TERMID>WEB*</TERMID><TOKEN>c2gvUFF1M05wYjg0Z0s
xVnA4NndqZz09DQo=</TOKEN></PORTALE>
```

Esempio di risposta alla richiesta di validazione:

```
<PORTALE><TIPOCONTROLLO>S</TIPOCONTROLLO><BANCA>05387</BANCA><TE
RMID>WEB*</TERMID><IDAPPL>A17</IDAPPL><TOKEN>c2gvUFF1M05wYjg0Z0s
xVnA4NndqZz09DQo=</TOKEN><ESITO>0</ESITO><MESSAGGIO></MESSAGGIO>
</PORTALE>
```

Chiamata per reperire cognome e nome

http://host/portale/PortaleServletServizi?<PORTALE><SERVIZIO>TERMINALE</SERVIZI
O><BANCA>05387</BANCA><TERMID>XXXX</TERMID></PORTALE>

risultato

<PORTALE><BANCA>05387</BANCA><UFFICIO>0080</UFFICIO><FILIALE>0090</FI
LIALE><PDLINIZIALE>000</PDLINIZIALE><SEZIONE>00</SEZIONE><BANCAINIZI

ALE>05387</BANCAINIZIALE><PDL>000</PDL><FILIALEINIZIALE>0000</FILIALEINIZIALE><LAN>000</LAN><ESITO>0</ESITO><RUOLOOPER>ANALISI</RUOLOOPER><PROFILOOPER>CENTRALE</PROFILOOPER><BLOCCO>L</BLOCCO><MESSAGGIO></MESSAGGIO><NOMEOPER>BATTILANI FILIPPO</NOMEOPER><CODOPER>XXXX</CODOPER><TERMID>XXXX</TERMID></PORTALE>

Chiusure delle applicazioni in altri contesti

E' possibile richiamare una servlet del portale per chiudere la pagina che contiene l'applicazione nel portale e ripassare il controllo alla prima pagina del portale, la documentazione è la seguente:

La chiusura delle applicazioni attive **sotto il controllo del portale** ed attive in contesti diversi da quello del portale può avvenire:

- Premendo il tasto destro del mouse sopra la descrizione dell'applicazione e selezionando l'elemento 'chiusura' del menu;
- Mediante chiamata ad una servlet del portale utilizzando alcuni parametri forniti dal portale in fase di avvio dell'applicazione. Tali parametri sono i seguenti:
    - **HOST**: nome dell'host in cui risiede il portale;
    - **PORTA**: nome della porta in cui risiede il portale;
    - **CONTESTO**: contesto in cui risiede il portale;
    - **SERVLETCHIUSURA**: nome della servlet del portale che effettua la chiusura;
      ovvero `PortaleServletChiudiApplRemota`
    - **ABI**: codice assegnato alla Banca;
    - **TERMID**: codice terminale;
    - **IDAPPL**: codice dell'istanza dell'applicazione assegnata dal portale.

Qualora l'applicazione remota decida di chiudersi, dovrà notificare tale decisione effettuano una chiamata all'url:

`http:[HOST]:[PORTA][CONTESTO][SERVLETCHIUSURA]` fornendo come parametri:

- ABI=*[ABI]*
- TERMID=*[TERMID]*
- IDAPPL=*[IDAPPL]*

Dove *[parametro]* è il contenuto del parametro fornito dal portale in fase di attivazione dell'applicazione.

APPENDICE 2 - SHELL UNIX REALIZZATA PER GENERARE IN MODO DINAMICO GLI SCRIPT DI INSTALLAZIONE DELLE APPLICAZIONI WEB SU PIATTAFORMA WEBSPHERE

```
#*************************************************************
#* ...CreaShellDeploy.sh
#*
#* Goal: creare dinamicamente cinque SHELL per:
#* - stoppare l'applicazione se già installata (stop{PROJECT_NAME}.sh)
#* - disinstallare l'applicazione se già installata (uninstall{PROJECT_NAME}.sh)
#* - installare l'applicazione (install{PROJECT_NAME}.sh)
#* - startare l'applicazione (start{PROJECT_NAME}.sh)
#* - eventualmente salvare le configurazioni (saveconfig{PROJECT_NAME}.sh)
#*
#* Parametri:  $1 Path del file .INSTALLA
#*             $2 Nome Progetto
#*             $3 NODE Installazione
#*             $4 SERVER Installazione
#*             $5 SQL Statico (SI/NO)
#*             $6 CONTEXTROOT (Solo per WAR)
#*             $7 PATHAPPLSRV
#*             $8 DB2_DATASOURCE_NAME_ORIG
#*             $9 FAIMAPRES
#*             $10 VH
#*************************************************************
# Controllo parametri necessari
#*************************************************************
if test -n $1 -a -n $2 -a -n $3 -a -n $4 -a -n $5 -a -n $6 -a -n $7
then
  PATH_INSTALLA=$1
  F_INSTALLA=$1/$2.installa
  NODE=$3
  SERVER=$4
  SQLSTATICO=$5
  CONTEXT=$6
  PATHAPPLSRV=$7
  DB2_DATASOURCE_NAME_ORIG=$8
  FAIMAPRES=$9
  VH=${10}
  filelog=/serena/shell/log/log_esecuzione.log
else
  echo `date`"ERRORE nel lancio della script." >> ${filelog}
  echo `date`"Usage: CreaShellDeploy.sh path_file nome_file node server sqlstatico
contextroot pathapplsrv" >> ${filelog}
  exit 1
fi
```

```
#*************************************************************
# Controllo esistenza e leggibilità del file di parametri {PROJECT_NAME}.INSTALLA
#*************************************************************
if [ ! -r ${F_INSTALLA} ]
then
  echo `date`"Manca il file ${F_INSTALLA} o mancano i permessi di lettura" >> ${filelog}
  exit 2
else
  PROJECT_NAME=`cat ${F_INSTALLA} | awk -F: '{ print $1 }'`
  APPL_PREFIX=`cat ${F_INSTALLA} | awk -F: '{ print $2 }'`
  APPL_SUFFIX=`cat ${F_INSTALLA} | awk -F: '{ print $3 }'`
  EAR_FILE=`cat ${F_INSTALLA} | awk -F: '{ print $4 }'`
  DBRM_FILE=`cat ${F_INSTALLA} | awk -F: '{ print $5 }'`
  DB2_DATASOURCE_NAME=`cat ${F_INSTALLA} | awk -F: '{ print $6 }'`
  CICS_DATASOURCE_NAME=`cat ${F_INSTALLA} | awk -F: '{ print $7 }'`
  FILE_BIND=`cat ${F_INSTALLA} | awk -F: '{ print $8 }'`
  PATH_EAR_DBRM=`cat ${F_INSTALLA} | awk -F: '{ print $9 }'`
fi
#*************************************************************
# variabili contenenti caratteri particolari
#*************************************************************
DOLLARO="$"
APICE="'"
APICESTORTO='`'
GRAFFAOPEN="{"
GRAFFACLOSE="}"
SLASH="/"
PUNTOVIRGOLA=";"
UGUALE="="
PIPE="|"
#*************************************************************
# calcolo del nome dell'ear/war senza suffisso
#*************************************************************
EAR_NAME=` echo ${EAR_FILE} | awk -F. '{ print $1 }'`
TIPOFILE=` echo ${EAR_FILE} | awk -F. '{ print toupper($2) }'`
echo `date`"depl_$EAR_NAME" >> ${filelog}
echo `date`"depl_$TIPOFILE" >> ${filelog}
#*************************************************************
# cancellazione dei file unistall${PROJECT_NAME}.sh,
# saveconfig${PROJECT_NAME}.sh,
# install${PROJECT_NAME}.sh,
# creaInstall${PROJECT_NAME}.sh, temp1${PROJECT_NAME}.txt,
# temp2${PROJECT_NAME}.txt, temp3${PROJECT_NAME}.txt,
# outputCreaInstall${PROJECT_NAME}, start${PROJECT_NAME}.sh,
# stop${PROJECT_NAME}.sh
#*************************************************************
if test -f uninstall${PROJECT_NAME}.sh
then
  rm uninstall${PROJECT_NAME}.sh
fi
```

```
if test -f install${PROJECT_NAME}.sh
then
  rm install${PROJECT_NAME}.sh
fi
if test -f saveconfig${PROJECT_NAME}.sh
then
  rm saveconfig${PROJECT_NAME}.sh
fi
if test -f creaInstall${PROJECT_NAME}.sh
then
  rm creaInstall${PROJECT_NAME}.sh
fi
if test -f temp1${PROJECT_NAME}.txt
then
  rm temp1${PROJECT_NAME}.txt
fi
if test -f temp2${PROJECT_NAME}.txt
then
  rm temp2${PROJECT_NAME}.txt
fi
if test -f temp3${PROJECT_NAME}.txt
then
  rm temp3${PROJECT_NAME}.txt
fi
if test -f temp3${PROJECT_NAME}.txt
then
  rm outputCreaInstall${PROJECT_NAME}
fi
if test -f start${PROJECT_NAME}.sh
then
  rm start${PROJECT_NAME}.sh
fi
if test -f stop${PROJECT_NAME}.sh
then
  rm stop${PROJECT_NAME}.sh
fi
```

```
#*************************************************************
# creazione di uninstall${PROJECT_NAME}.sh e stop${PROJECT_NAME}.sh
# con verifica se l'applicativo è già installato
#*************************************************************
PATH=$PATH:${PATHAPPLSRV}
echo 'PATH=$PATH:'${PATHAPPLSRV} > uninstall${PROJECT_NAME}.sh
echo 'PATH=$PATH:'${PATHAPPLSRV} > stop${PROJECT_NAME}.sh
if [ ${TIPOFILE} = "EAR" ]
then
       APPL=${EAR_NAME}
       echo APPL=${EAR_NAME} >> uninstall${PROJECT_NAME}.sh
       echo APPL=${EAR_NAME} >> stop${PROJECT_NAME}.sh
else
       APPL=${EAR_NAME}_war
       echo APPL=${EAR_NAME}_war >> uninstall${PROJECT_NAME}.sh
       echo APPL=${EAR_NAME}_war >> stop${PROJECT_NAME}.sh
fi
echo "GIA_INSTALL${UGUALE}${APICESTORTO}wsadmin.sh -c ${APICE}${DOLLARO}AdminApp
list${APICE} ${PIPE} grep -x ${APPL}${APICESTORTO}" >> uninstall${PROJECT_NAME}.sh
echo "GIA_INSTALL${UGUALE}${APICESTORTO}wsadmin.sh -c ${APICE}${DOLLARO}AdminApp
list${APICE} ${PIPE} grep -x ${APPL}${APICESTORTO}" >> stop${PROJECT_NAME}.sh

echo 'if test -z ${GIA_INSTALL}' >> uninstall${PROJECT_NAME}.sh
echo 'if test -z ${GIA_INSTALL}' >> stop${PROJECT_NAME}.sh
echo 'then' >> uninstall${PROJECT_NAME}.sh
echo 'then' >> stop${PROJECT_NAME}.sh
echo 'else' >> uninstall${PROJECT_NAME}.sh
echo 'else' >> stop${PROJECT_NAME}.sh
echo 'if [  ${APPL}  -eq  ${GIA_INSTALL} ]' >> uninstall${PROJECT_NAME}.sh
echo 'if [  ${APPL}  -eq  ${GIA_INSTALL} ]' >> stop${PROJECT_NAME}.sh
echo 'then' >> uninstall${PROJECT_NAME}.sh
echo 'then' >> stop${PROJECT_NAME}.sh
       UNINST1=`echo wsadmin.sh -conntpe SOAP -host localhost -c ${APICE}set appManager
[${DOLLARO}AdminControl queryNames
cell=${NODE}Network,node=${NODE},type=ApplicationManager,process=${SERVER},*]${PUNTOVIRG
OLA}${DOLLARO}AdminControl invoke ${DOLLARO}appManager stopApplication ${APPL}${APICE}`
         echo ${UNINST1} >> stop${PROJECT_NAME}.sh
              UNINST1=`echo wsadmin.sh -conntpe SOAP -host localhost -c
${APICE}${DOLLARO}AdminApp uninstall ${APPL}${PUNTOVIRGOLA}${DOLLARO}AdminConfig
save${APICE}`
       echo ${UNINST1} >> uninstall${PROJECT_NAME}.sh
echo 'fi' >> uninstall${PROJECT_NAME}.sh
echo 'fi' >> stop${PROJECT_NAME}.sh
echo 'fi' >> uninstall${PROJECT_NAME}.sh
echo 'fi' >> stop${PROJECT_NAME}.sh
```

```
#*************************************************************
# estrazione dell'elenco dei datasource presenti
# e creazione install${PROJECT_NAME}.sh
#*************************************************************
COMMD=`echo wsadmin.sh -c ${APICE}${DOLLARO}AdminApp taskInfo
${PATH_EAR_DBRM}/${EAR_FILE} MapResRefToEJB${APICE}`
echo `date`"${COMMD}" >> ${filelog}
echo  ${COMMD} > creaInstall${PROJECT_NAME}.sh

# Verifica se l'applicazione contiene Risorse da Mappare
# (RESOURCE=0 -> SI ------- RESOURCE=1 -> NO)
sh creaInstall${PROJECT_NAME}.sh > outputCreaInstall${PROJECT_NAME}
RESOURCE=`cat outputCreaInstall${PROJECT_NAME} | grep -in "There is no resource
references." | wc -l`
if [ ${FAIMAPRES} != "SI" ]
then
  RESOURCE=1
fi
echo `date`"RESOURC:${RESOURCE}" >> ${filelog}
echo `date`"File:${EAR_FILE}" >> ${filelog}
echo `date`"Name:${EAR_NAME}" >> ${filelog}
# Prima riga del file install${PROJECT_NAME}.sh
if [ ${RESOURCE} -eq 1 ] || [ ${TIPOFILE} = "WAR" ]
then
  echo `date`"INST-A" >> ${filelog}
  INST1=`echo wsadmin.sh -conntpe SOAP -host localhost -c ${APICE}${DOLLARO}AdminApp
install ${PATH_EAR_DBRM}/${EAR_FILE} ${GRAFFAOPEN}-appname ${APPL} '\'`
else
  echo `date`"INST-B" >> ${filelog}
  INST1=`echo wsadmin.sh -conntpe SOAP -host localhost -c ${APICE}${DOLLARO}AdminApp
install ${PATH_EAR_DBRM}/${EAR_FILE} ${GRAFFAOPEN}-appname ${APPL} -MapResRefToEJB
${GRAFFAOPEN} '\'`
fi

# Penultima riga del file install${PROJECT_NAME}.sh
if [ ${TIPOFILE} = "EAR" ]
then
  INST3=`echo -server ${SERVER} -node ${NODE} -cell
${NODE}Network${GRAFFACLOSE}${PUNTOVIRGOLA}${DOLLARO}AdminConfig save${APICE}`
else
  INST3=`echo -server ${SERVER} -node ${NODE} -cell ${NODE}Network -contextroot
\"/${CONTEXT}\"${GRAFFACLOSE}${PUNTOVIRGOLA}${DOLLARO}AdminConfig save${APICE}`
fi
echo `date`"INST3:${INST3}" >> ${filelog}
```

```
#*************************************************************
# creazione di start${PROJECT_NAME}.sh
#*************************************************************
# Avvio della applicazione - Ultima riga del file install${PROJECT_NAME}.sh
if [ ${TIPOFILE} = "EAR" ]
then
  INST4=`echo wsadmin.sh -conntpe SOAP -host localhost -c ${APICE}set appManager
[${DOLLARO}AdminControl queryNames
cell=${NODE}Network,node=${NODE},type=ApplicationManager,process=${SERVER},*]${PUNTOVIRG
OLA}${DOLLARO}AdminControl invoke ${DOLLARO}appManager startApplication
${EAR_NAME}${APICE}`
else
  INST4=`echo wsadmin.sh -conntpe SOAP -host localhost -c ${APICE}set appManager
[${DOLLARO}AdminControl queryNames
cell=${NODE}Network,node=${NODE},type=ApplicationManager,process=${SERVER},*]${PUNTOVIRG
OLA}${DOLLARO}AdminControl invoke ${DOLLARO}appManager startApplication
${EAR_NAME}_war${APICE}`
fi
#*************************************************************
# continua la creazione di install${PROJECT_NAME}.sh
#*************************************************************
if [ ${RESOURCE} -eq 0 ] && [ ${TIPOFILE} = "EAR" ]
then
  # Elimino le prime righe di output (di solito 11) perchè non contengono dati utili
  # e inserisco il risultato nel file temp1${PROJECT_NAME}.txt
    RIGHETOT=`cat outputCreaInstall${PROJECT_NAME}|wc -l`
  RIGHEINUTILI=`cat outputCreaInstall${PROJECT_NAME} | grep -in "The current contents of
the task after running default bindings are" | awk -F: '{ print $1 }'`
  echo `date`"${RIGHEINUTILI}" >> ${filelog}
fi

if [ ${RESOURCE} -eq 0 ] && [ ${TIPOFILE} = "EAR" ]
then
  if test -z ${RIGHEINUTILI}
  then
    echo  'echo errore in fase di creazione file install' > install${PROJECT_NAME}.sh
    echo  'echo errore in fase di creazione file install' > uninstall${PROJECT_NAME}.sh
    echo  'echo errore in fase di creazione file install' > start${PROJECT_NAME}.sh
    echo  'echo errore in fase di creazione file install' > stop${PROJECT_NAME}.sh

    if test -f install${PROJECT_NAME}.sh
    then
      rm install${PROJECT_NAME}.sh
    fi
    if test -f saveconfig${PROJECT_NAME}.sh
    then
      rm saveconfig${PROJECT_NAME}.sh
    fi
    if test -f uninstall${PROJECT_NAME}.sh
    then
```

```
      rm uninstall${PROJECT_NAME}.sh
    fi
    if test -f start${PROJECT_NAME}.sh
    then
      rm start${PROJECT_NAME}.sh
    fi
    if test -f stop${PROJECT_NAME}.sh
    then
      rm stop${PROJECT_NAME}.sh
    fi
    echo `date`"ERRcreazione1-install${PROJECT_NAME}.sh" >> ${filelog}
  else
    if [  ${RIGHEINUTILI}  -eq  11 ]
    then
      RIGHEUTILI=`expr $RIGHETOT - $RIGHEINUTILI`
      #sh creaInstall${PROJECT_NAME}.sh|tail -n ${RIGHEUTILI} > temp1${PROJECT_NAME}.txt
      cat outputCreaInstall${PROJECT_NAME}|tail -n ${RIGHEUTILI} >
temp1${PROJECT_NAME}.txt

         # Sostituzione di "module: " con "module:" per eliminare lo spazio
         `sed 's/module: /module:/g' temp1${PROJECT_NAME}.txt >
temp2${PROJECT_NAME}.txt`
         cat temp2${PROJECT_NAME}.txt > temp1${PROJECT_NAME}.txt

         # Sostituzione di "EJB: ", "uri: ", "referenceBinding: ", "resRef.type: ",
"JNDI: "  con """
         `sed 's/EJB: /"/g' temp1${PROJECT_NAME}.txt > temp2${PROJECT_NAME}.txt`
         cat temp2${PROJECT_NAME}.txt > temp1${PROJECT_NAME}.txt
         `sed 's/uri: /"/g' temp1${PROJECT_NAME}.txt > temp2${PROJECT_NAME}.txt`
         cat temp2${PROJECT_NAME}.txt > temp1${PROJECT_NAME}.txt
         `sed 's/referenceBinding: /"/g' temp1${PROJECT_NAME}.txt >
temp2${PROJECT_NAME}.txt`
         cat temp2${PROJECT_NAME}.txt > temp1${PROJECT_NAME}.txt
         `sed 's/resRef.type: /"/g' temp1${PROJECT_NAME}.txt > temp2${PROJECT_NAME}.txt`
         cat temp2${PROJECT_NAME}.txt > temp1${PROJECT_NAME}.txt
         `sed 's/JNDI: /"JNDI:/g' temp1${PROJECT_NAME}.txt > temp2${PROJECT_NAME}.txt`
         cat temp2${PROJECT_NAME}.txt > temp1${PROJECT_NAME}.txt

         RIGHETOTtemp1=`cat temp1${PROJECT_NAME}.txt|wc -l`
         counter=0
         cat /dev/null > temp2${PROJECT_NAME}.txt

         #Per ogni riga concatenazione di doppio apice o inserimento di '} \' se vuota
         while [ ${counter} -lt ${RIGHETOTtemp1} ]
         do
           counter=`expr $counter + 1`
           echo $counter
           RIGAtemp=`cat temp1${PROJECT_NAME}.txt | head -n ${counter} | tail -n 1 | awk
'{ print $0 }'`
          if test -z ${RIGAtemp}
```

```
            then
              RIGAtemp2='} \'
            else
              RIGAtemp3=`echo ${RIGAtemp} | grep JNDI:`
              if test -z ${RIGAtemp3}
              then
                RIGAtemp2=${RIGAtemp}'" \'
              else
                RIGAtemp2='"jdbc/'${DB2_DATASOURCE_NAME}'" \'
              fi
            fi
            echo ${RIGAtemp2} >> temp2${PROJECT_NAME}.txt
          done

          # Sostituizione di "module: " con '{ "' per eliminare lo spazio
          `sed 's/module:/{ "/g' temp2${PROJECT_NAME}.txt > temp3${PROJECT_NAME}.txt`
          cat temp3${PROJECT_NAME}.txt > temp2${PROJECT_NAME}.txt

          echo 'PATH=$PATH:'${PATHAPPLSRV} > install${PROJECT_NAME}.sh
          echo ${INST1} >> install${PROJECT_NAME}.sh
          cat temp2${PROJECT_NAME}.txt >> install${PROJECT_NAME}.sh

          echo ${INST3} >> install${PROJECT_NAME}.sh

    echo 'PATH=$PATH:'${PATHAPPLSRV} > start${PROJECT_NAME}.sh
    echo ${INST4} >> start${PROJECT_NAME}.sh

          echo '#LA SAVE VIENE ESEGUITA NELLE SHELL DI INSTALLAZIONE E DISINSTALLAZIONE'
> saveconfig${PROJECT_NAME}.sh
          # Spostamento delle shell create nelle directory di lavoro
          mv install${PROJECT_NAME}.sh ${PATH_INSTALLA}'/'install${PROJECT_NAME}.sh
          mv uninstall${PROJECT_NAME}.sh ${PATH_INSTALLA}'/'uninstall${PROJECT_NAME}.sh
${PATH_INSTALLA}'/'saveconfig1${PROJECT_NAME}.sh
          mv saveconfig${PROJECT_NAME}.sh ${PATH_INSTALLA}'/'saveconfig${PROJECT_NAME}.sh
          mv start${PROJECT_NAME}.sh ${PATH_INSTALLA}'/'start${PROJECT_NAME}.sh
          mv stop${PROJECT_NAME}.sh ${PATH_INSTALLA}'/'stop${PROJECT_NAME}.sh

          chmod 655 ${PATH_INSTALLA}'/'install${PROJECT_NAME}.sh
          chmod 655 ${PATH_INSTALLA}'/'uninstall${PROJECT_NAME}.sh
          chmod 655 ${PATH_INSTALLA}'/'saveconfig${PROJECT_NAME}.sh
          chmod 655 ${PATH_INSTALLA}'/'start${PROJECT_NAME}.sh
          chmod 655 ${PATH_INSTALLA}'/'stop${PROJECT_NAME}.sh
    else
          echo  'echo errore in fase di creazione file install' >
install${PROJECT_NAME}.sh
          echo  'echo errore in fase di creazione file install' >
saveconfig${PROJECT_NAME}.sh
          echo  'echo errore in fase di creazione file install' >
uninstall${PROJECT_NAME}.sh
          echo  'echo errore in fase di creazione file install' > start${PROJECT_NAME}.sh
```

```
          echo  'echo errore in fase di creazione file install' > stop${PROJECT_NAME}.sh

          if test -f install${PROJECT_NAME}.sh
          then
            rm install${PROJECT_NAME}.sh
          fi
          if test -f saveconfig${PROJECT_NAME}.sh
          then
            rm saveconfig${PROJECT_NAME}.sh
          fi
          if test -f uninstall${PROJECT_NAME}.sh
          then
            rm uninstall${PROJECT_NAME}.sh
          fi
          if test -f start${PROJECT_NAME}.sh
          then
            rm start${PROJECT_NAME}.sh
          fi
          if test -f stop${PROJECT_NAME}.sh
          then
            rm stop${PROJECT_NAME}.sh
          fi
          echo `date`"ERRcreazione2-install${PROJECT_NAME}.sh" >> ${filelog}
    fi
  fi
else
  echo 'PATH=$PATH:'${PATHAPPLSRV} > install${PROJECT_NAME}.sh
  echo ${INST1} >> install${PROJECT_NAME}.sh
  echo `date`"PrimaDi-MapWebModToVHProg${PROJECT_NAME}" >> ${filelog}
  echo `date`"VH${PROJECT_NAME} ${VH}" >> ${filelog}
  if [ ${VH} = "SI" ]
  then
          echo `date`"Inserimento-MapWebModToVHProg${PROJECT_NAME}" >> ${filelog}
          echo '-MapWebModToVH  '${GRAFFAOPEN}' \' >> install${PROJECT_NAME}.sh
          echo ${GRAFFAOPEN}' "'${EAR_NAME}'" \' >> install${PROJECT_NAME}.sh
          echo '"'${EAR_FILE}',WEB-INF/web.xml" \' >> install${PROJECT_NAME}.sh
          echo '"default_host" \' >> install${PROJECT_NAME}.sh
          echo ${GRAFFACLOSE}' \' >> install${PROJECT_NAME}.sh
          echo ${GRAFFACLOSE}' \' >> install${PROJECT_NAME}.sh
        fi

        if [ ${DB2_DATASOURCE_NAME_ORIG} != "NO" ]
        then
                echo '-MapResRefToEJB  '${GRAFFAOPEN}' \' >> install${PROJECT_NAME}.sh
                echo ${GRAFFAOPEN}' "'${EAR_NAME}'" \' >> install${PROJECT_NAME}.sh
                echo '"" \' >> install${PROJECT_NAME}.sh
                echo '"'${EAR_FILE}',WEB-INF/web.xml" \' >> install${PROJECT_NAME}.sh
                echo '"jdbc/'${DB2_DATASOURCE_NAME_ORIG}'" \' >>
install${PROJECT_NAME}.sh
                echo '"javax.sql.DataSource" \' >> install${PROJECT_NAME}.sh
```

```
                echo '"jdbc/'${DB2_DATASOURCE_NAME}'" \' >> install${PROJECT_NAME}.sh
                echo ${GRAFFACLOSE}' \' >> install${PROJECT_NAME}.sh
                echo ${GRAFFACLOSE}' \' >> install${PROJECT_NAME}.sh
  fi

  echo ${INST3} >> install${PROJECT_NAME}.sh
  echo 'PATH=$PATH:'${PATHAPPLSRV} > start${PROJECT_NAME}.sh
  echo ${INST4} >> start${PROJECT_NAME}.sh

  echo '#LA SAVE VIENE ESEGUITA NELLE SHELL DI INSTALLAZIONE E DISINSTALLAZIONE' >
saveconfig${PROJECT_NAME}.sh
#*************************************************************
# Spostamento delle shell create nelle directory di lavoro
#*************************************************************
  echo PERCORSO:`pwd`
  mv install${PROJECT_NAME}.sh ${PATH_INSTALLA}'/'install${PROJECT_NAME}.sh
  mv uninstall${PROJECT_NAME}.sh ${PATH_INSTALLA}'/'uninstall${PROJECT_NAME}.sh
  mv saveconfig${PROJECT_NAME}.sh ${PATH_INSTALLA}'/'saveconfig${PROJECT_NAME}.sh
  mv start${PROJECT_NAME}.sh ${PATH_INSTALLA}'/'start${PROJECT_NAME}.sh
  mv stop${PROJECT_NAME}.sh ${PATH_INSTALLA}'/'stop${PROJECT_NAME}.sh

  chmod 655 ${PATH_INSTALLA}'/'install${PROJECT_NAME}.sh
  chmod 655 ${PATH_INSTALLA}'/'uninstall${PROJECT_NAME}.sh
  chmod 655 ${PATH_INSTALLA}'/'saveconfig${PROJECT_NAME}.sh
  chmod 655 ${PATH_INSTALLA}'/'start${PROJECT_NAME}.sh
  chmod 655 ${PATH_INSTALLA}'/'stop${PROJECT_NAME}.sh
fi
#*************************************************************
# Eliminazione dei file temporanei
#*************************************************************
if test -f creaInstall${PROJECT_NAME}.sh
then
  rm creaInstall${PROJECT_NAME}.sh
fi
if test -f temp1${PROJECT_NAME}.txt
then
  rm temp1${PROJECT_NAME}.txt
fi
if test -f temp2${PROJECT_NAME}.txt
then
  rm temp2${PROJECT_NAME}.txt
fi
if test -f temp3${PROJECT_NAME}.txt
then
  rm temp3${PROJECT_NAME}.txt
fi
if test -f outputCreaInstall${PROJECT_NAME}
then
  rm outputCreaInstall${PROJECT_NAME}
fi
```

Bibliografia

- Sito ufficiale di Java della Sun Microsystem: http://java.sun.com
- "Java 2 Enterprise Edition" documentazione disponibile al sito http://java.sun.com/j2ee/docs.html
- Sito ufficiale Banca Popolare dell'Emilia Romagna
- Schede tecniche Banca Popolare dell'Emilia Romagna
- "Progetto di Basi di Dati Relazionali" Sonia Bergamaschi. Domenico Beneventano, Maurizio Vincini, Pitagora Editrice Bologna
- "Javascript, La guida" David Flanagan, Apogeo
- "JSP Java Server Pages, Guida di riferimento" James Goodwilll, Apogeo