UNIVERSITA' DI PISA

FACOLTA' DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

PROGETTO ED IMPLEMENTAZIONE DI APPLICAZIONI WEB SICURE CON DISPOSITIVI MOBILI

Relatore: Prof. G. Dini

Correlatore: Prof. M. Avvenuti

Candidato: Federico Mori

anno accademico 2004–2005

Ai miei genitori,

a mia sorella e mio fratello,

alla mia ragazza,

che mi hanno sostenuto

in tutti questi anni.

Ringraziamenti

Un particolare ringraziamento va al Prof. Gianluca Dini e all'Ingegnere Nicola Provenzano per l'aiuto e i tanti consigli dati durante lo sviluppo della tesi. Ringrazio anche tutti i ragazzi che come me hanno lavorato in questi mesi nel laboratorio Perlab. Ringrazio infine, ma non per questo ultimi, tutti gli amici e i parenti che mi hanno seguito e sostenuto durante questo lungo iter universitario.

Sommario

Elenco delle Figure

Prefazione

La diffusione di Internet rende possibile la comunicazione tra utenti dotati di calcolatori dislocati in ogni parte del mondo senza aver bisogno di particolari procedure di collegamento, senza richiedere particolari attrezzature e, soprattutto, senza dover sostenere costi di trasmissione elevati. Oggi la rete Web è divenuta il mezzo di comunicazione e trasmissione dati tra computer maggiormente diffuso ed utilizzato: quotidianamente se ne fa uso per scopi ludici o interessi professionali. È ormai pratica comune utilizzare il Web per transazioni bancarie o altro tipo di operazioni delicate, veicolando, quindi, informazioni sensibili.

Il problema da affrontare oggi è quello della sicurezza delle trasmissioni in quanto, proprio per la sua larghissima diffusione, il rischio di usi impropri della rete è in costante agguato.

Nella catena di comunicazione vengono identificati tre componenti principali soggetti a tale tipo di problemi e su cui bisogna prestare attenzione: l'identità del mittente, l'integrità dei dati e l'identità del destinatario. Eccetto gli ultimi due aspetti, già analizzati e risolti in modo soddisfacente con le correnti tecnologie, l'attenzione del nostro lavoro si è soffermata sull'identità del mittente identificandolo come anello debole della catena. Per definizione infatti la postazione client è priva di qualsiasi tipo di garanzia potendo essere soggetta a virus, manomissioni esterne ed numerosi altri tipi di intrusioni. Spesso, a causa di poche o addirittura privo di conoscenze informatiche, l'utente non è in grado di gestire la macchina con cui opera, non è cosciente dello stato in cui il proprio sistema si trova e non è cosciente dei security risks a cui è esposto.

La garanzia dell'identità del soggetto che invia un messaggio serve a determinare che il messaggio proviene proprio da colui che egli dice di essere. La questione non è semplice quando i due soggetti della comunicazione non si conoscono direttamente e comunque non sono fisicamente vicini (in tal caso basterebbe l'esibizione di un documento d'identità). Scopo del nostro lavoro viene così ad essere quello di garantire l'identità del soggetto che utilizza delle credenziali elettroniche/informatiche.

Per fare questo abbiamo introdotto un ulteriore vincolo, un oggetto personale in possesso dell'utente, che permettesse di aumentare il livello di sicurezza del sistema garantendo l'univocità dell'utente. La scelta è ricaduta, vista anche la sua notevole diffusione sul telefono cellulare.

Il sistema sviluppato integra l'utilizzo di entrambi gli ambienti, rete Web e telefonia cellulare.

Con questo lavoro si è voluto sviluppare una Web Application che assicurasse l'autenticità di un utente, facendo uso di tecnologie ampiamente utilizzate come la piattaforma J2EE, il Web Server Container Tomcat e molte altre, coniugandole con strumenti di comunicazione mobile notevolmente diffusi. La prerogativa della nostra Web Application, oltre che a fornire un insieme di servizi per l'utente, è migliorare il livello di sicurezza. Tale processo viene solitamente implementato con l'utilizzo di moduli di autenticazione che adottano sistemi avanzati di sicurezza sulle reti. L'approccio scelto nel sistema sviluppato si articola in due fasi:

una prima fase di autenticazione con una password in possesso dell'utente ed eventualmente modificabile; ed una seconda fase di autenticazione con una password "*OneTime*", password di breve durata, monouso, generata automaticamente dal sistema ed inviata sul dispositivo cellulare.

Il lavoro svolto si suddivide in una prima parte dove si analizza l'uso ed il funzionamento del gateway SMS Kannel, in particolare implementando un'interfaccia amministrativa. Successivamente viene progettata ed implementata la web application "**VotiOnline**" che utilizza il gateway stesso.

Nella relazione seguente si cerca di descrivere accuratamente il sistema, dai componenti realizzati alle scelte implementative effettuate.

Nel Capitolo 1 vengono analizzate la problematiche sulla sicurezza in rete e viene fornita una visione d'insieme del sistema sviluppato specificando i concetti base della autenticazione e delle password OTP.

Il Capitolo 2 presenta il dispositivo gateway Kannel analizzando i suoi componenti ad alto livello e descrivendo la progettazione e l'implementazione di una interfaccia amministrativa.

Nel Capitolo 3 viene mostrata la progettazione ad alto livello della Web Application VotiOnline e successivamente tutte le architetture utilizzate tra cui *J2EE*, *JAAS, MVC, JSP*.

Il Capitolo 4 mostra l'implementazione dell'applicazione VotiOnline.

Infine nel Capitolo 5 vengono riportate le conclusioni del nostro lavoro.

Capitolo 1

INTRODUZIONE AL SISTEMA

Con l'aumento delle aziende, degli enti pubblici e dei privati sul web con un proprio sito, o semplicemente collegati a internet, è in costante e rapido aumento la problematica della sicurezza in rete.

Internet e il web risultano estremamente vulnerabili ad attacchi di varia natura; cresce quindi la domanda di servizi web sicuri.

Il World Wide Web è fondamentalmente un'applicazione client/server che gira su Internet e inter-reti TCP/IP. Classificare le minacce alla sicurezza web consiste nella localizzazione della minaccia: server web, browser web e traffico di rete fra browser e server.

I browser, client Web, sono motori multiprotocollo tra i quali HTTP, protocollo che permette di recuperare ed interpretare documenti in formato HTML. La maggior parte delle transazioni web comporta l'uso dell'HTTP per il recupero di documenti HTML. Tipicamente una sessione http si compone di un comando GET o POST che specifica un URL seguito da un numero

facoltativo di righe. Il server può richiedere all'utente un'autenticazione per un particolare servizio, e lo fa respingendo la richiesta e visualizzando all'utente un form dove inserire le proprie credenziali; inseriti i dati il browser riprova la connessione inviando i dati immessi. Da questa complessa concatenazione di passaggi di informazioni si può capire come possano formarsi molti rischi:

- Intercettazione di dati trasmessi tramite la rete.
- Manipolazione di dati negli elementi intermedi della rete (come i router).
- IP address spoofing dove l'host attaccante finge di essere uno affidabile inviando pacchetti con l'indirizzo di questo.
- DNS spoofing di server DNS fidati.
- IP source routine.

In quanto la navigazione non è in genere anonima, poiché la maggior parte delle connessioni non è crittata, le informazioni possono essere intercettate ed eventualmente modificate da un malintenzionato. I comandi Telnet, rlogin, rcp, rsh hanno parecchi punti deboli quanto a sicurezza: tutte le comunicazioni sono in chiaro e nessuna autenticazione viene svolta dalla macchina. Questi comandi sono suscettibili di intercettazione e allo spoofing dell'indirizzo IP. Ricordiamo che per "spoofing" si intende la sostituzione dell'indirizzo reale con uno falsificato.

Simili informazioni possono essere ottenute dalla cache del browser o dal file della cronologia sull'host di un client. Quindi le informazioni di autenticazione come altre possono trovarsi sotto gli occhi di persone non autorizzate.

Uno dei metodi più semplici per la protezione dei dati consiste nell'assicurarsi che la connessione tra il client e il server dei Web service sia protetta. A tale scopo, esistono diverse tecniche a seconda della portata della rete ed il profilo di attività delle interazioni. Tre tra le tecniche disponibili più diffuse vi sono:

- regole basate sul firewall.
- SSL (Secure Sockets Layer).
- reti private virtuali (VPN, Virtual Private Networks).

Se si conosce esattamente quali sono i computer che devono accedere al servizio, è possibile utilizzare le regole del firewall per limitare l'accesso ai soli computer con indirizzo IP conosciuto. Questa tecnica risulta particolarmente utile quando si desidera limitare l'accesso ai computer su una rete privata, ad esempio una rete LAN/WAN aziendale, e si desidera mantenere segreto (crittografato) il contenuto dei messaggi. Firewall possono fornire regole basate su criteri avanzati che offrono limitazioni differenti a client diversi basandosi sull'origine o sull'identità. Ciò risulta particolarmente utile quando, ad esempio, client diversi hanno accesso a funzionalità (metodi) differenti per gli stessi Web service.

È possibile utilizzare SSL per stabilire connessioni protette su reti non attendibili (come Internet). SSL crittografa e decrittografa i messaggi scambiati tra il client e il server. Crittografando i dati, i messaggi vengono protetti dalla lettura durante il loro trasferimento. SSL crittografa un messaggio dal client e quindi lo invia al server. Una volta ricevuto dal server, SSL decrittografa il messaggio e verifica che sia stato inviato dal mittente corretto (questo processo è conosciuto come autenticazione). Il server, o il client e il server, potrebbero essere in possesso dei certificati utilizzati come parte del processo di autenticazione, fornendo funzionalità di autenticazione

oltre alla crittografia di connessione. Per quanto sia un metodo particolarmente efficace per la creazione di comunicazioni protette, SSL presenta costi di prestazione da tenere in considerazione.

Una rete privata virtuale (VNP) è un'estensione di una rete privata che connette reti condivise o pubbliche come Internet. Una VPN consente di inviare dati tra due computer in una connessione protetta. Sebbene sia simile a SSL, una VPN è una connessione point-to-point a lungo termine. Ciò la rende efficace e particolarmente adatta ai Web service, ma per ottenere tale efficienza richiede che venga stabilita una connessione a lungo termine ed in costante esecuzione.

Altrettanto importante è garantire un elevato livello di sicurezza, oltre che sul canale di comunicazione, anche direttamente sul client/utente. Qualsiasi sia la protezione presente sulla rete di comunicazione è fondamentale che anche la sorgente dei dati sia protetta.

Uno scenario possibile potrebbe riguardare un utente in possesso di password per l'accesso al proprio conto corrente bancario. Nel caso in cui questa password venisse rubata o copiata da un altro utente, qualsiasi tipo di protezione sul canale di comunicazione sarebbe inutile. Verrebbe infatti a mancare la garanzia di attendibilità della fonte dei dati.

Sorge così il problema di dover assicurare il sistema sull'autenticità dell'utente in possesso delle credenziali di sicurezza.

La tesi proposta realizza una Web Application che affronta tale problematica utilizzando due livelli di autenticazione:

oltre a quello tradizionale che si avvale di password memorizzate e trasmesse via http, il sistema è costituito da un secondo livello di sicurezza non più

legato alla rete web ma al mondo dei dispositivi mobili, in particolare legato al telefono cellulare.

Il lavoro svolto si è suddiviso così in due parti.In una prima fase si sono affrontate le problematiche relative all'infrastruttura di Gateway SMS per telefoni cellulari, con particolare attenzione allo sviluppo di una interfaccia di amministrazione e controllo del Gateway. In una seconda fase si è realizzata una Web Application in grado di utilizzare i due livelli di autenticazione e di interfacciarsi con il Gateway SMS.

1.1 L'Autenticazione

L'autenticazione è il processo del *provare* la propria identità. Si tratta di una fase distinta dall'asserire la propria identità (fase di identificazione) e dal decidere quali privilegi derivino da essa (fase di autorizzazione). Dal punto di vista della sicurezza in rete l'autenticazione risulta la fase più complessa.

L'autenticazione è basata su uno, due o tre fattori:

- Qualcosa che si *conosce*
- Qualcosa che si *possiede*
- Qualcosa che si *è*

Il primo fattore riguarda password, PIN ecc., il secondo riguarda carte bancarie e dispositivi di autenticazione (per esempio dispositivi mobili), e il terzo fa riferimento ad attributi di tipo biologico.

Le soluzioni per l'autenticazione possono coinvolgere un processo con uno o più fattori; le applicazioni più semplici usano un solo fattore, quelle più delicate almeno due.

Le password, che ricadono nella categoria del "qualcosa che si sa", hanno si il vantaggio di essere utilizzate senza l'utilizzo di alcun strumento particolare, ma anche uno svantaggio perché ciò che si sa può essere detto ad altri, o catturato, oppure indovinato.

Nonostante l'aumento del livello di sicurezza legato all'utilizzo di token hardware è possibile che un utente non sia ben disposto a farne uso. Per la nostra applicazione il token hardware coinvolto nel processo di autenticazione è un dispositivo mobile entrato a far parte da diversi anni della vita comune di un gran numero di persone, il telefono cellulare (Figura 1.1).

Questo permette all'applicazione di essere facilmente utilizzata da qualsiasi utente, con conoscenze più o meno approfondite nel campo informatico, e di potersi adattare a tutte le realtà presenti sul territorio, grazie alla diffusione del telefono cellulare.

Il telefono cellulare entra a far parte del sistema di sicurezza come veicolo per le password monouso a tempo generate dall'applicazione.

1.2 Password monouso a tempo

Il concetto di one time password nasce dalla precaria sicurezza nella comunicazione tra client/server via http.

Come dice il nome, le OTP una volta generate, successivamente vedremo come e da chi, vengono utilizzate e la loro vita dura per il tempo strettamente necessario al processo di autenticazione dopodichè vengono cancellate. Vedi Figura 1.1.Queste password offrono una difesa molto forte contro le intercettazioni, i comandi telnet compromessi e anche contro la pubblicazione delle sessioni di login.Proprio le OTP costituiscono, nel sistema presentato, il secondo livello di sicurezza. Le OTP una volta generate vengono inviate al cellulare, token hardware della nostra applicazione, dell'utente che si deve autenticare. In seguito al loro utilizzo e alla conseguente affermativa autenticazione vengono cancellate in modo che nessun altro utente possa "loggarsi" utilizzando una "vecchia" password.

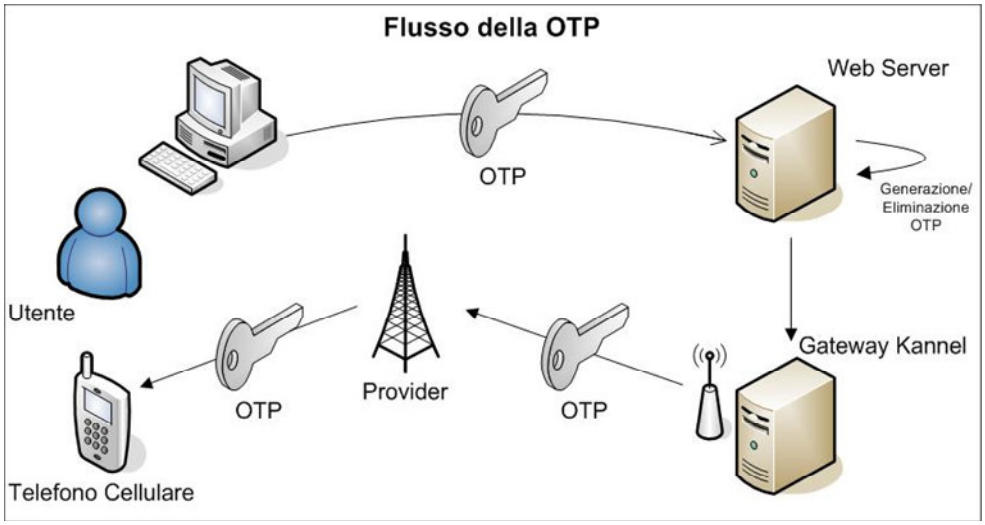

Figura 1.1 Flusso della password one time

Capitolo 2

GATEWAY KANNEL

La larga diffusione dei dispositivi cellulari ha permesso un processo di alfabetizzazione di massa della popolazione inaspettato. È comune, ormai, trovare persone delle età più disparate fare uso del cellulare per consultare servizi di segreteria telefonica, o inviare sms per comunicare con gli amici. Il cellulare è diventato uno strumento al centro d'ogni tipo di attività, dall'intrattenimento musicale a quello ludico, dall'attività lavorativa a quella personale.

I punti di forza di questo dispositivo risiedono nella semplicità con cui può essere avviata una comunicazione e nell'immediatezza insita nella scrittura dei messaggi sms: semplici messaggi di testo che posso essere inviati ad un prezzo irrisorio.

Il Gateway, è un'infrastruttura di telecomunicazioni e costituisce un ponte tra l'applicazione ed il cliente, dotato di telefono cellulare.

2.1 Kannel - SMS Gateway

Kannel è un progetto Open Source nato su iniziativa di Wapit Ltd[1]. Il progetto è nato nel Luglio del 1999 con lo scopo di implementare un gateway WAP ed SMS. Le esigenze erano dettate dalla necessità di iniziare a sviluppare servizi avanzati basati sulla piattaforma WAP in un contesto in cui le soluzioni esistenti erano poche e dal costo elevato.

Kannel fu sviluppato per supportare servizi di gateway sia per WAP che per SMS, in quanto i dispositivi in grado di utilizzare WAP erano ancora troppo pochi, situazione che nel tempo è migliorata, anche se, tuttora, sono moltissimi i dispositivi che non supportano WAP.

2.1.1 Architettura di Kannel

Kannel, come specificato in Figura 2.1, è dotato di interfacce per la comunicazione con tre tipologie differenti di agenti:

- SMS Centers: operatori di telefonia, come possono essere Vodafone o Telecom, con protocolli di comunicazione differenti
- Server HTTP: server in grado di operare su messaggi Wap o SMS, allo scopo di implementare servizi avanzati per dispositivi mobili
- Telefoni WAP

[1] Wapit Ltd è leader mondiale nell'ambito della progettazione e sviluppo di tecnologie ed applicazioni mobile

Gli SMS Center operano utilizzando vari protocolli di comunicazione, molti dei quali sono supportati da Kannel e vanno specificati in fase di configurazione.

Perché Kannel si renda disponibile al ricevimento di messaggi SMS, da parte di un utente dotato di telefono cellulare, è necessario stabilire un account presso l'SMS Center, richiedendolo all'operatore di rete mobile.

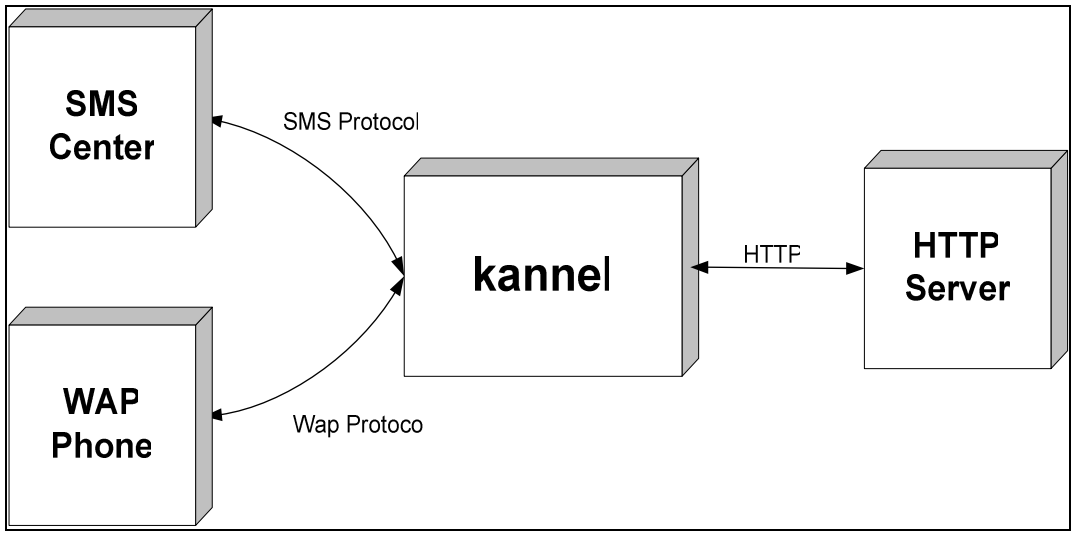

Figura 2.1 Interfacce di comunicazione per Kannel

Ad ogni account, acquisito presso un SMS Center, corrisponde un numero telefonico, cui può fare riferimento un utente e con il quale vengono spediti i messaggi SMS da Kannel.

Kannel riceve ed invia dati sulle interfacce operando in multitasking, in particolare tutti i dati ricevuti in ingresso sono memorizzati in alcune code interne e processati appena possibile.

2.1.2 Organizzazione in Box

Internamente Kannel opera utilizzando tre tipologie differenti di processi, Figura 2.2, chiamati box, ognuno dei quali serve una differente tipologia di interfaccia:

- Il *Bearerbox*, si occupa di fare da interfaccia con il mondo esterno, riceve i messaggi SMS dagli SMS Center, li esamina per controllare se sono semplici SMS o contengono pacchetti WAP, quindi li instrada verso il box più adatto.
- Il *Smsbox*, implementa le funzionalità necessarie ad un SMS Gateway, riceve i messaggi SMS, li interpreta come richieste di servizio e risponde ad essi nel modo opportuno
- Il *Wapbox*, implementa il protocollo WAP e WAP Push

Ci può essere una sola istanza del Bearerbox, ma più istanze di Smsbox o Wapbox, questa scelta deriva dalla necessità di utilizzare un unico modulo come interfaccia con gli SMS Center.

Il compito del Bearerbox è quello di ricevere i messaggi in ingresso, inserirli in opportune code e smistarli, poi, al rispettivo box.

Tutti i box operano in multithreading, quindi, ad esempio, nel caso del Bearerbox ci sono più thread, uno per il protocollo WAP, uno per SMS ed altri per la gestione delle code.

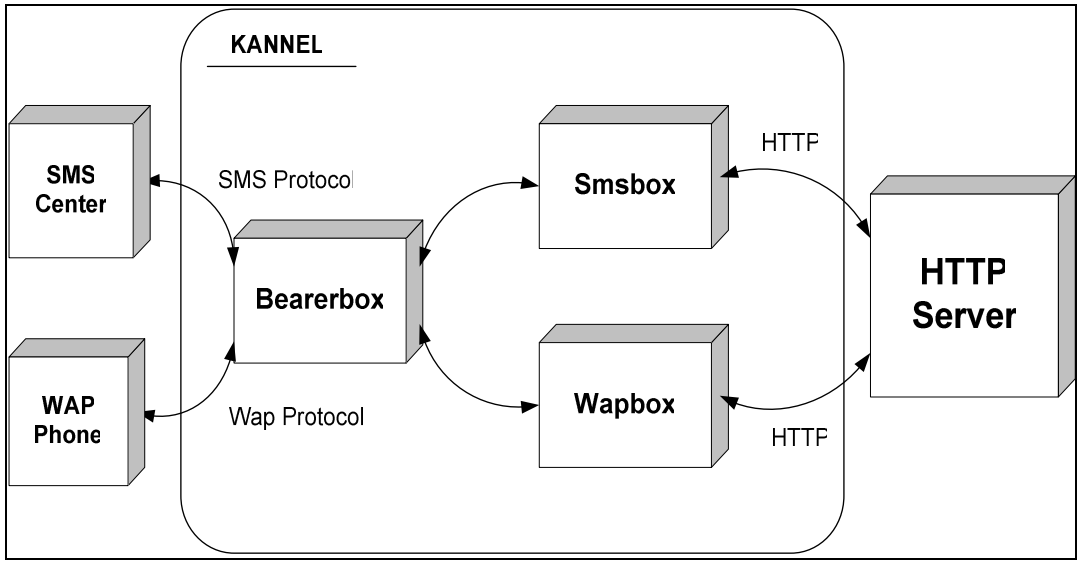

Figura 2.2 I box presenti in Kannel

2.1.3 Servizi SMS

Il bearerbox ha il compito di inoltrare i messaggi ricevuti su una della interfacce di ingresso all'opportuno box. L'Smsbox ha il compito di analizzare i messaggi SMS ed interpretarli come richieste di servizio.

Kannel offre la possibilità di implementare semplici servizi, utilizzando alcune routine interne, in questo caso il risultato del servizio richiesto, prodotto dall'elaborazione dell'Smsbox, viene spedito, indietro, all'Bearerbox che provvede ad inoltrarlo, sotto forma di SMS, al cliente specifico.

2.1.4 SMS Center

Gli SMS Center dal punto di vista di Kannel sono gli strumenti per connettersi alla rete mobile e per spedire o ricevere SMS.

Fondamentalmente, possiamo dividere gli SMS Center in due grandi categorie:

- SMS Center gestiti da operatori di telefonia mobile
- interfacce in grado di gestire la comunicazione mobile, fondamentalmente telefoni cellulari e modem GSM

Nel primo caso, il servizio è fornito direttamente dall'operatore: l'operatore di telefonia assegna al Centro Servizi un numero speciale, ad es. 48xxx, al quale risponde direttamente Kannel. Si tratta di un servizio molto più affidabile, primo perché richiesto espressamente, poi, anche, perché la connessione con l'operatore di telefonia, il più delle volte, avviene tramite link fisico. In questo caso, la connessione avviene mediante l'uso di modem e connessioni con protocolli UCP/EMI, o SMPP.

Nel secondo caso si utilizza una soluzione più flessibile, più economica, ma meno prestante. Praticamente l'idea è quella di utilizzare un dispositivo, tipo cellulare o modem GSM, dotati di Sim Card, per spedire e ricevere SMS e poi interfacciare questo dispositivo con Kannel per permetterne la gestione, solitamente con un cavo di tipo seriale.

2.2 Gateway Kannel e Password OTP

Il sistema è costituito da diversi sottosistemi fondamentali tra cui citiamo quelli di maggior pertinenza con la nostra problematica di sicurezza:

- Kannel
- Postazione di Controllo
- SMS Center

Il sottosistema Kannel è costituito dai due moduli *bearerbox* e *smsbox*, dove il primo si occupa di comunicare con gli SMS Center e con eventuali altri dispositivi di comunicazione, tipo modem GSM e il secondo è adibito alla gestione dei servizi SMS. Smsbox offre, inoltre, ulteriori operazioni di log, ed il servizio di SMS Push.

La Postazione di Controllo ha il compito di amministrazione del sottosistema Kannel avendo accesso a tutte le operazioni di gestione del sistema. Inoltre una Postazione di Controllo permette l'invio di SMS. Per l'utilizzo del servizio di SMS Push il Centro Servizi ha bisogno di essere registrato presso Kannel, come SendSMSUser.

La chiamata per l'invio di un sms, l'sms stesso nella logica del nostro sistema è costituito proprio dalla nostra password OTP, verso un destinatario deve quindi essere di questo tipo:

http://smsbox/cgi-bin/sendsms?username=foo&password=bar&to=3471234567&text=testo;

2.3 L'interfaccia Amministrativa

Per facilitare l'utilizzo del Gateway Kannel è stata implementata un' interfaccia Web (Figura 2.3) per la gestione ed il controllo dello stato del Gateway Kannel da parte dell'amministratore.

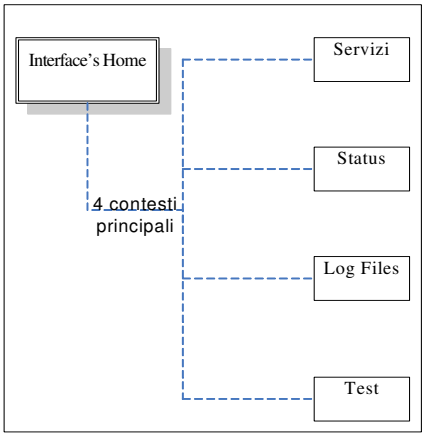

Figura 2.3 Contesti principali interfaccia

2.3.1 Struttura dell'interfaccia

L'utente che utilizza il portale amministrativo ha a disposizione quattro modalità di interazione col gateway(Figura 2.4):

- Uso dei servizi amministrativi
- Verifica stato del gateway
- Consultazione log files
- Fase di test del gateway

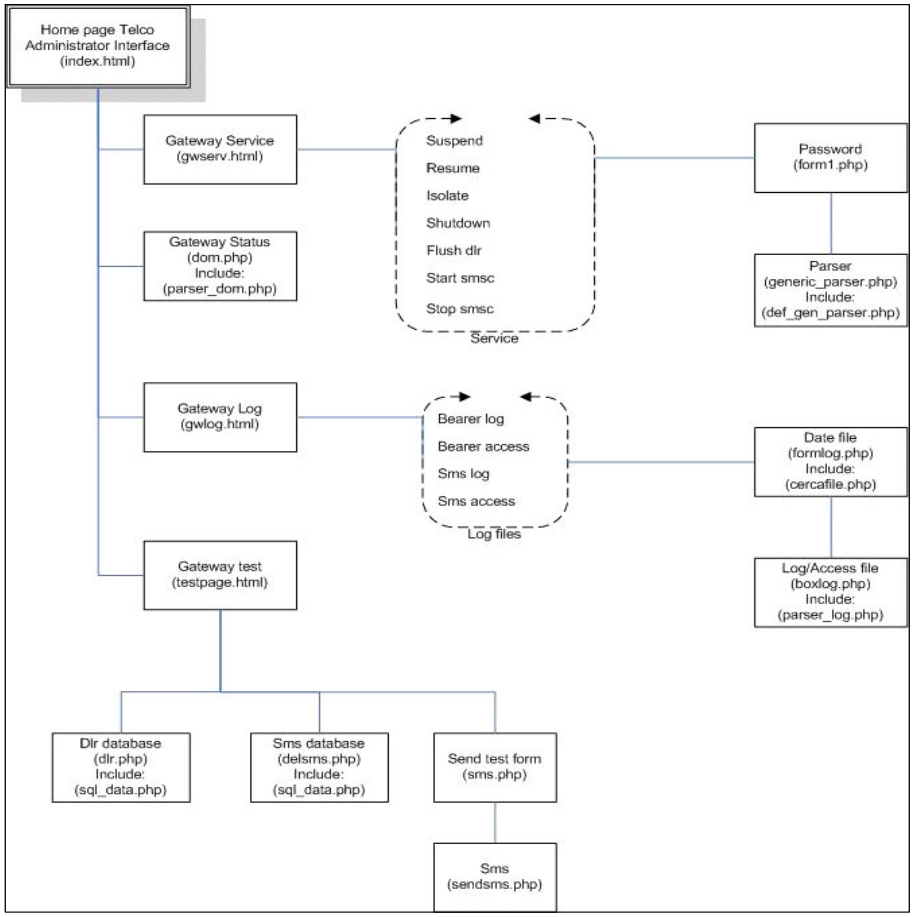

Figura 2.4 Organigramma interfaccia

2.3.1.1 Servizi amministrativi

In questa pagina l'amministratore può gestire direttamente il funzionamento del gateway scegliendo l'appropriato servizio; prima che questo venga eseguito è richiesta la password amministrativa (Figura 2.5).

Per ogni servizio fornito è presente anche un link informativo.

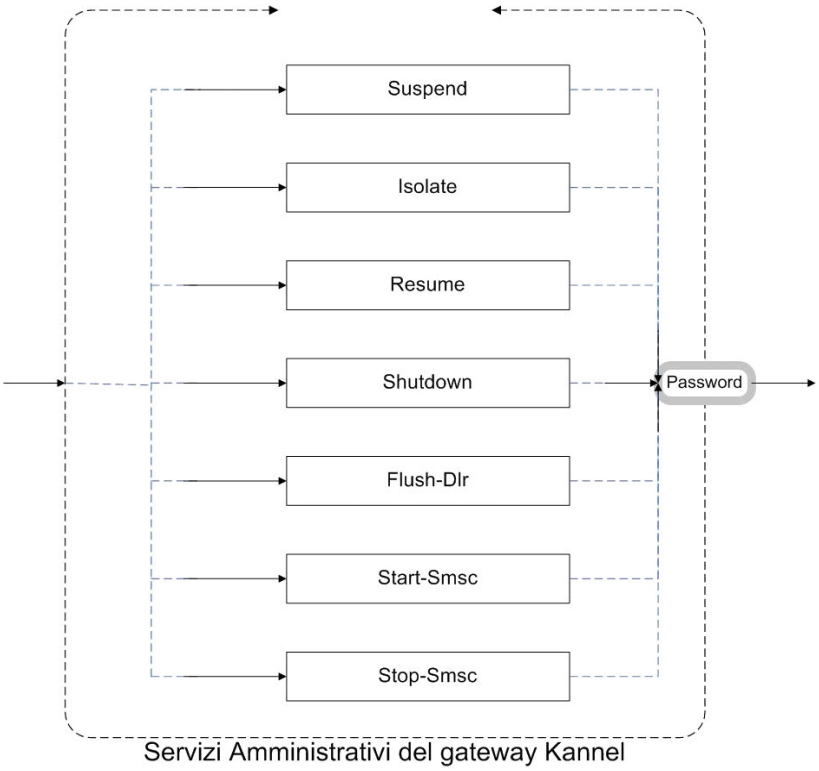

Figura 2.5 Servizi amministrativi

2.3.1.1.1 **Implementazione della pagina servizi**

La gestione dinamica delle pagine è garantita per mezzo di funzioni ***php***.

Il tipo di servizio è passato, dopo aver associato ognuno di essi ad una stringa in un array, da un form con metodo GET al file php *generic_parser.php* che si occupa di trasmettere il comando al gateway.

Allo stesso tempo viene effettuato un *parsing* delle informazioni fornite dal gateway, sulla riuscita o meno dell'operazione. Il *parsing* viene effettuato aprendo il file con una *fopen()* ed eseguendo su di esso una *fread()* fino alla fine del file (*feof()*). L'URL remoto fornito dal form per richiedere il servizio è aperto come un file con l'utilizzo della funzione *filesize_remote()*.

2.3.1.2 Gateway status

Qui l'amministratore può visionare varie informazioni riguardanti il gateway:

- Versione del *Bearerbox*
- Stato corrente del Gateway
- Numero di *WDP* ricevuti, accodati e inviati
- Numero di *SMS* ricevuti, accodati e inviati
- Tipo di *boxes* presenti e loro stato
- Numero di *SMSC* presenti loro stato ed altre informazioni

2.3.1.2.1 **Implementazione della pagina di status**

Dopo aver aperto in lettura il file remoto XML, contenente tutte le informazioni riguardanti il gateway si effettua un *parsing xml* adottando la tecnica del Dom-Parser(Figura 2.6).

Quindi per creare il file html si fa uso della funzione *parsefile()* inclusa nel file *parser_dom.php* che con l'ausilio del tipo predefinito Domdocument genera prima la radice dell'oggetto dinamico e poi ricorsivamente visita tutto il documento xml creando creando figli per ogni tag incontrato.

Così facendo il file parser creato in php è scalabile verso il futuro potendosi adattare a qualsiasi modifica del file xml.

Il risultato ottenuto è una rappresentazione chiara della struttura del file xml che rispetta le corrispondenze logiche tra i tag.

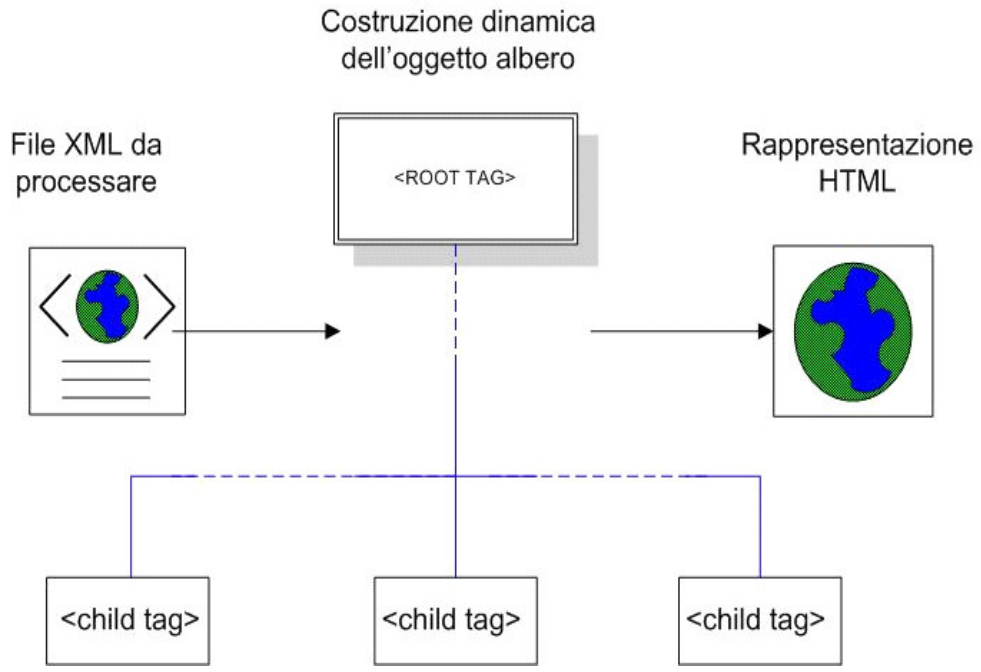

Figura 2.6 Parser Dom file xml remoto

2.3.1.3 File di log

Nella Log Files page è possibile rivedere i file log e access archiviati in var/log/kannel (Figura 2.7).

Si dà la possibilità di scegliere tra i vari file oltre in base all'estensione anche in base alla data di creazione. E' possibile visionare i file di log compressi con l'algoritmo gz.

2.3.1.3.1 **Implementazione della pagina files log**

Dopo aver scelto il link adatto ai propri interessi, scegliendo tra i due diversi box (sms e bearer) e tra le due diverse estensioni (log e access) è possibile selezionare la data del file usando una select in html e la funzione php *filemtime()*.

Successivamente con la funzione php *parsefile()* si effettua un parsine del file presente nella directory ordinando le varie informazioni all'interno di una tabella html. Le informazioni presenti nel file vengono organizzate in un array e stampate in colonne di una tabella html.

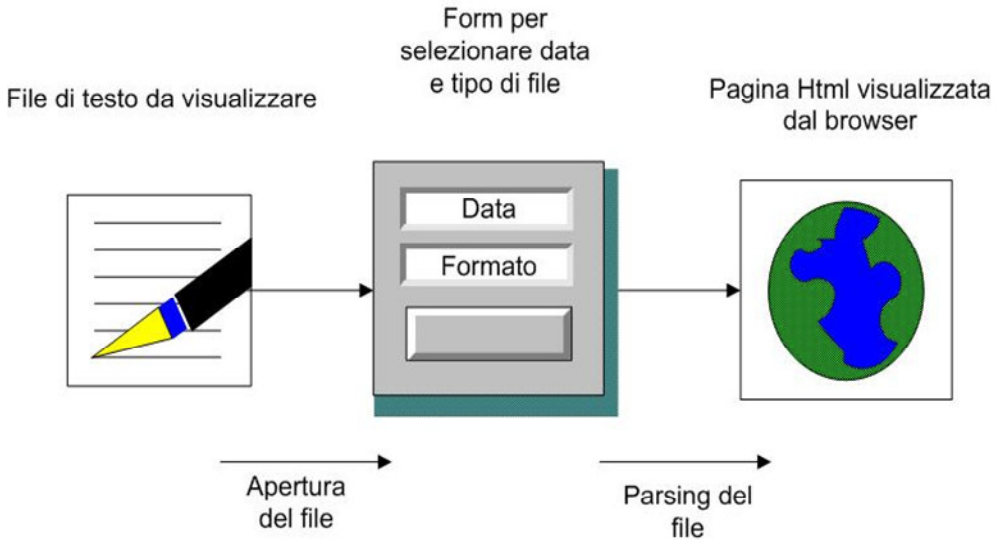

Figura 2.7 Scelta del file di log

2.3.1.4 Fase di test e accesso al database

Come ultima opportunità l'amministratore ha la possibilità di accedere al database e visionare tutte le informazioni riguardanti gli SMS inviati.

Ha anche la possibilità di effettuare di effettuare un test per l'invio di un sms dal gateway dopo aver compilato un apposito form (Figura 2.8).

2.3.1.4.1 **Implementazione della pagina di test**

Selezionato il database voluto si crea la connessione con questo. La connessione viene effettuata per mezzo della funzione *mysql_connect()* che apre una connessione con un database appunto mysql. Sempre per mezzo di codice php con la funzione *display_db_table()* si visualizzano le informazioni

contenute nel database all'interno di una tabella html. In un array vengono memorizzati i vari tipi di campo della tabella che successivamente verranno richiamati al momento in cui questa verrà costruita. La query sul database genera un insieme di stringhe che mano a mano vengono stampate all'interno della suddetta tabella.

Il form per effettuare il send test è generato anch'esso con codice php e con l'uso di javascript per dei pop-up informativi.

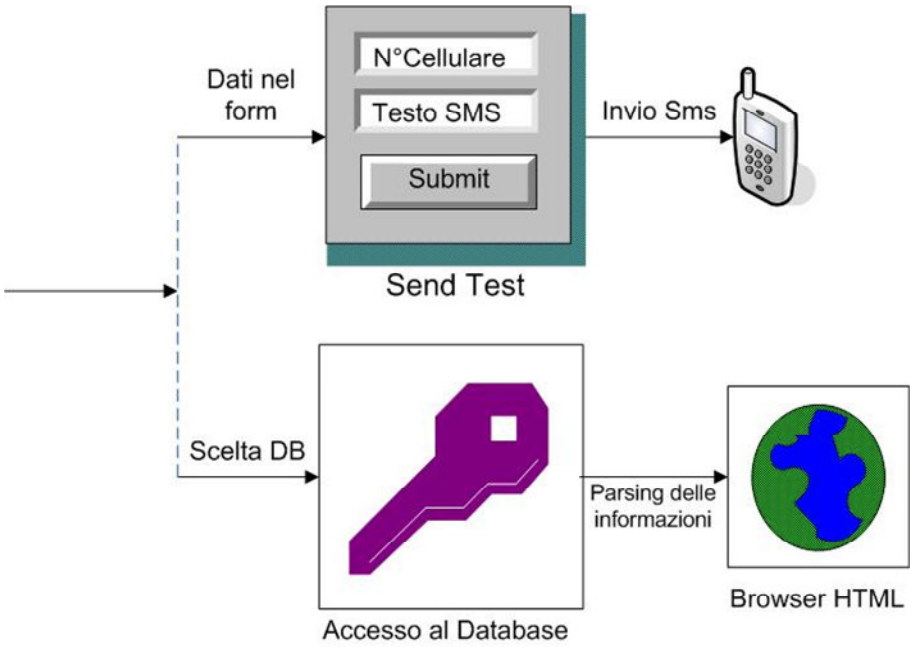

Figura 2.8 Fase di test e recupero informazioni da database mysql

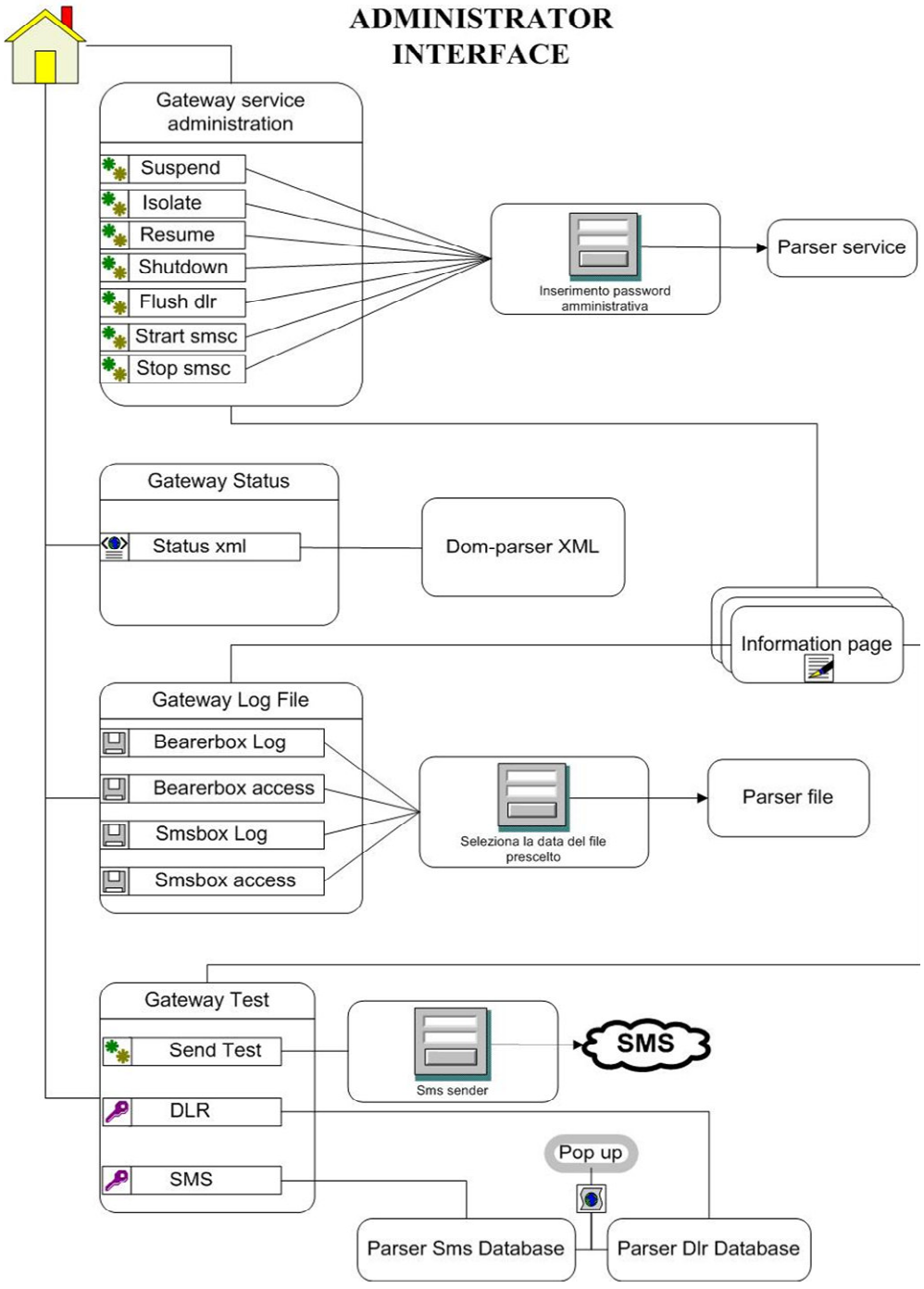

Figura 2.9 Interfaccia Amministrativa

Capitolo 3

WEB APPLICATION – VOTIONLINE

3.1 Requisiti di Progetto

La Web Application sviluppata fornisce un sistema informatizzato per la gestione degli statini rispettando particolari vincoli di sicurezza avvalendosi dell'ausilio di strumenti mobile.

I requisiti fondamentali a cui la web application deve rispondere sono (Figura 3.1):

1. Mantenimento della sicurezza delle informazioni:

   Effettuare l'autenticazione di 1° livello basata su password.

   Effettuare l'autenticazione di 2° livello basata su OneTime password.

2. Fornire i servizi di cui il client ha bisogno.

3. Chiudere la sessione e cancellare le informazioni di sessione.

3.1.1 Garanzie di sicurezza

La prerogativa del sistema è di garantire un alto livello di sicurezza nell'accesso ai servizi forniti impedendo così che dati presenti possano essere letti o modificati da utenti non autorizzati.L'alto livello di sicurezza è assicurato dalla presenza di un doppio strato di autenticazione che rende il sistema più sicuro nell'interazione con i browser degli utenti.

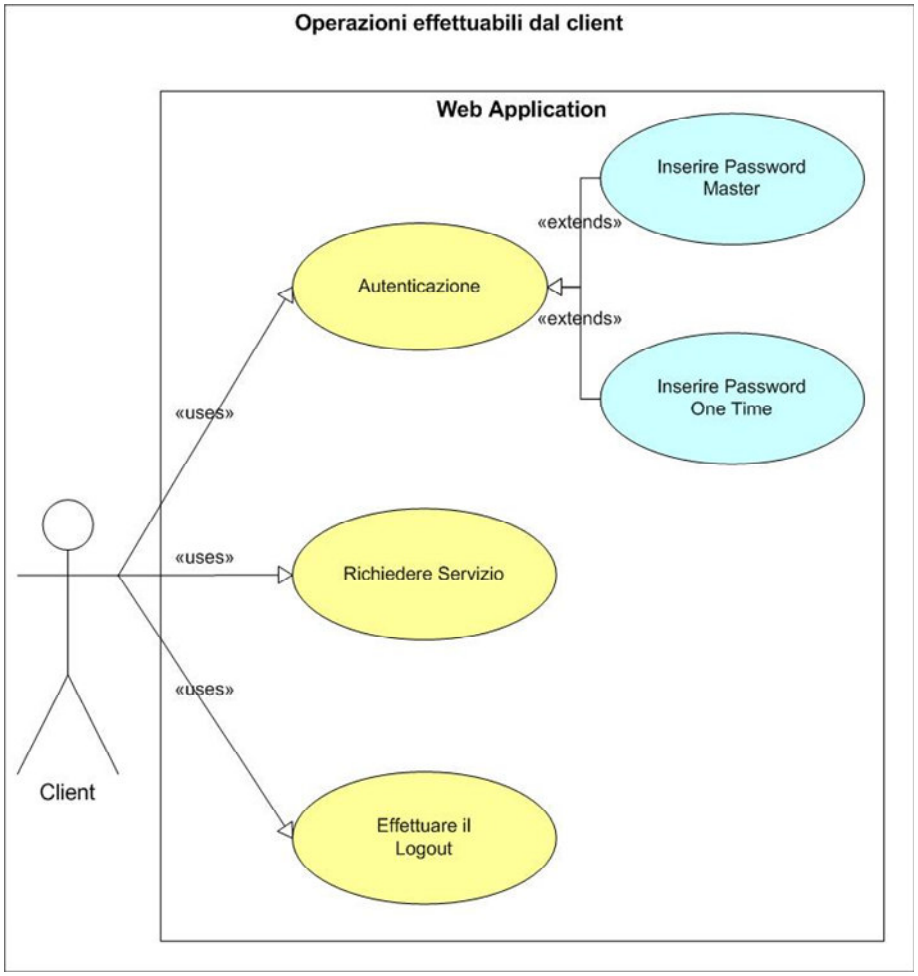

Figura 3.1 Casi d'uso dell'applicazione

3.1.2 Autenticazione di primo livello

Questo requisito, come scritto nel titolo del paragrafo, è la prima barriera di sicurezza che un utente incontra nell'accesso ai servizi (Figura 3.2).

In quanto tale è anche quel livello di sicurezza che più si espone ad eventuali attacchi di malintenzionati.

Si basa su un insieme di moduli implementati dalla Sun, il modello JAAS. Questo insieme di moduli sono stati rivisti e modificati per potersi meglio adattare alle specifiche del problema. Questo livello di sicurezza, con l'ausilio del database, ci permette di identificare univocamente un client non fornendo però certezza della corrispondenza tra le credenziali inserite e il client che si trova dall'altro lato della http-Request. Il problema risiede proprio nell'architettura della rete internet.

3.1.3 Autenticazione di secondo livello

Passato il primo livello di sicurezza si effettua la seconda e ultima autenticazione (Figura 3.2).

Il sistema genera la password one time e tramite il gateway kannel la invia la dispositivo mobile, telefono cellulare, dell'utente.

In questo modo le informazioni di sicurezza come la one time password, che ha un tempo di vita pari a quella dell'applicazione stessa, aumentano ulteriormente l'impenetrabilità del sistema ad eventuali malintenzionati..

La robustezza del secondo livello di autenticazione risiede proprio nell'utilizzo di dispositivi mobili (telefoni cellulari) che esulano dalla comunicazione http tra client e server.

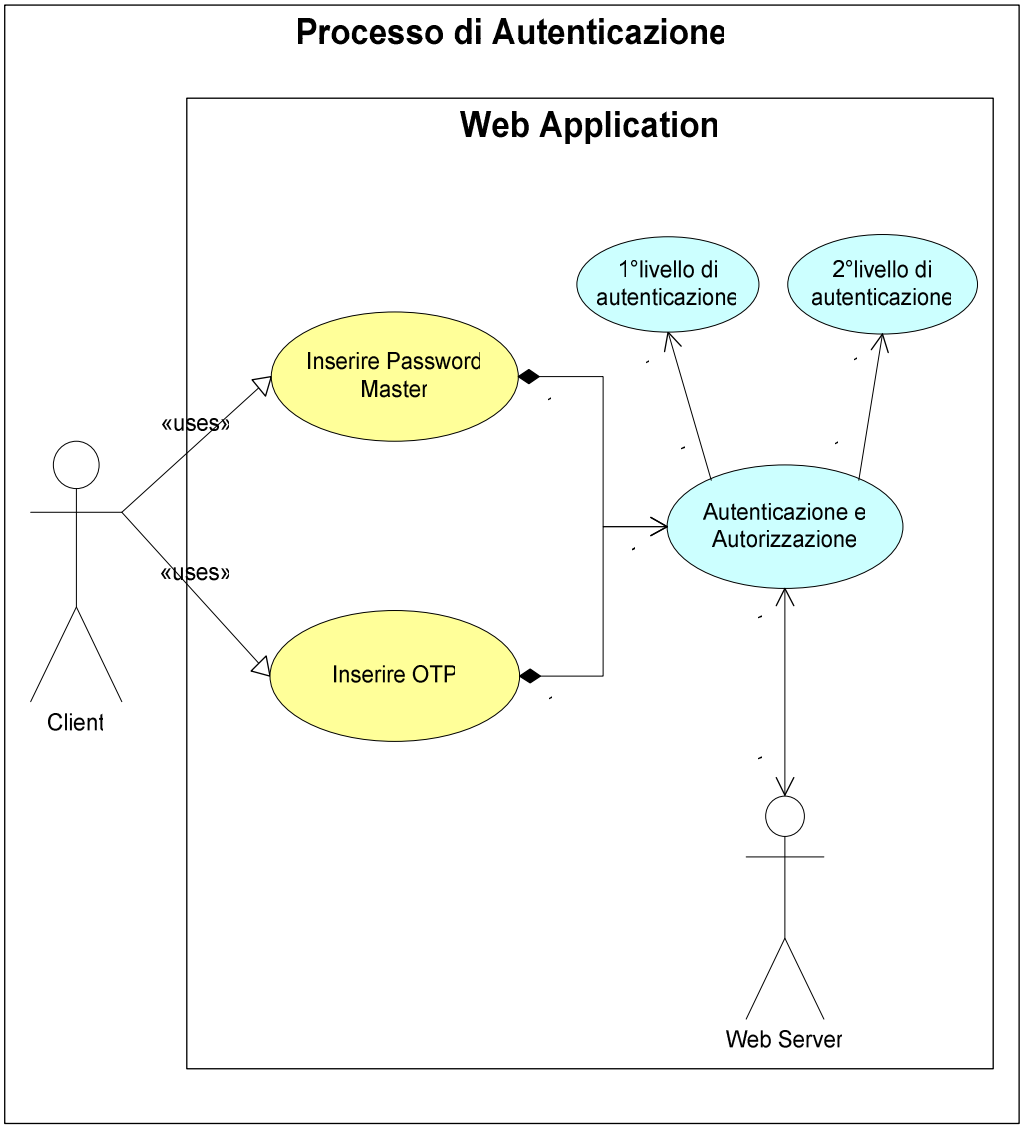

Figura 3.2 Casi d'uso sicurezza

3.1.4 Servizi Amministrativi

Giunti a questo punto l'utente è definitivamente autenticato e ha la possibilità di richiedere un servizio qualsiasi, tra quelli offerti.

Nella specifica Web Application l'implementazione dei servizi si è limitata a 4 anche se grazie alla flessibilità del modello utilizzato MVC la modularità ed espandibilità risulta essere una proprietà implicita ed esente di ulteriori accorgimenti.

I sottosistemi sono completamente disgiunti, questo garantisce una modifica degli stessi o riprogettazione nel completo rispetto del funzionamento del resto del programma.

I servizi offerti sono 4 (Figura 3.3):

- Ricerca di uno statino
- Inserimento di uno statino
- Modifica della Password Master
- Modifica del numero di cellulare

Le operazioni di modifica password e numero di cellulare vengono eseguite solo dopo aver autorizzato la modifica; l'autorizzazione viene effettua richiedendo nuovamente la password master.

La ricerca dello statino viene effettuata dopo aver selezionato l'appropriato campo su cui eseguire la query. Sia in caso affermativo che negativo vengono visualizzati i risultati.

Tutti e quattro i servizi operano all'interno della stessa sessione e quindi sono eseguiti a nome dello stesso client precedentemente autenticato; quindi ogni operazione effettuata sul database è vincolata sempre allo stesso *subject*.

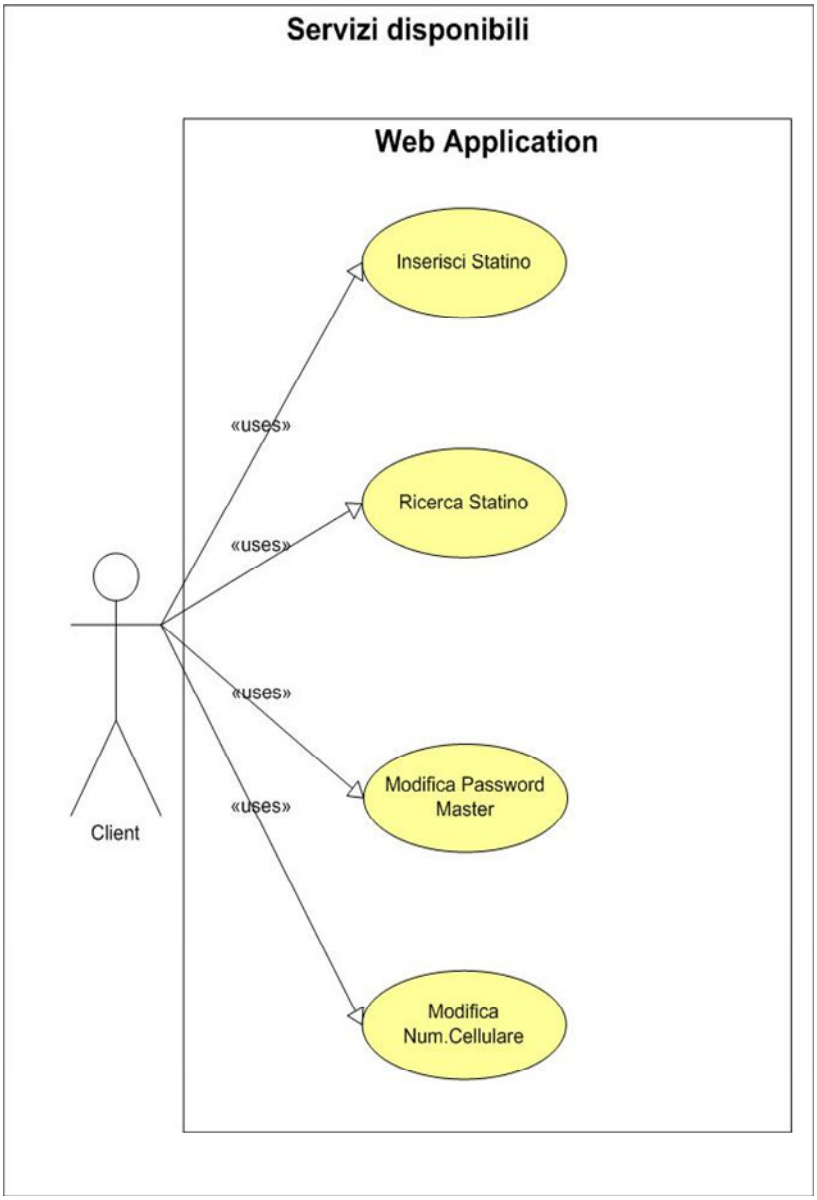

Figura 3.3 Casi d'uso servizi

3.1.5 Fine sessione

Una volta che l'utente ha utilizzato i servizi voluti deve effettuare il logout dalla Web. Application. Questo passo impone che tutte le informazioni di sessione, come password one time e credenziali del soggetto autenticato, vengano invalidate.

Così facendo al login successivo sarà generata una nuova password one time d'associare alla sessione. Le informazioni di sessione infatti durano per tutta la durata del timeout imposto in fase di progettazione.

3.2 Architetture utilizzate

3.2.1 Piattaforma J2EE

Il linguaggio di programmazione Java rappresenta la scelta ideale per la progettazione di un'architettura server-side. La portabilità è uno dei suoi punti fondamentali. La possibilità di avere un linguaggio che sia portabile su ogni tipo di piattaforma di sviluppo rappresenta un enorme vantaggio poiché uno sviluppatore può scrivere il componente un'unica volta e renderlo utilizzabile su ogni tipo di ambiente server-side esistente. Per dare quindi la possibilità a tutti di poter usufruire delle potenzialità di Java, la Sun ha prodotto una completa piattaforma di sviluppo chiamata J2EE (Java 2 Enterprise Editino).

La piattaforma J2EE ha aggiunto alla flessibilità, alla potenza, ed alla semplicità di utilizzo di Java una vera e solida prospettiva di business: la possibilità di realizzare applicazioni distribuite scalabili, transazionali a

componenti. Questo è possibile grazie all'integrazione nella piattaforma di una suite di servizi ed interfacce standard come JNDI, JTS, JMS, JCA e JAAS.

J2EE nasce inoltre fortemente orientata allo sviluppo di applicazioni internet ed intranet con l'introduzione di un set di tecnologie (Java Server Pages e Servlets) per l'integrazione del front-end internet/intranet con il back-end applicativo a componenti (Java Beans, Enterprise Java Beans-EJB)(Figura 3.4).

3.2.1.1 Architettura Multi-Tier

Ogni applicazione distribuita viene realizzata effettuando una suddivisione della stessa in layer (un layer rappresenta uno strato di suddivisione orizzontale).

Ciascun layer ha un ruolo particolare nell'ambito dello sviluppo dell'applicazione ed all'interno può contenere uno o più componenti; i layer sono semplicemente astrazioni e non corrispondono ad una distribuzione fisica dell'applicazione.

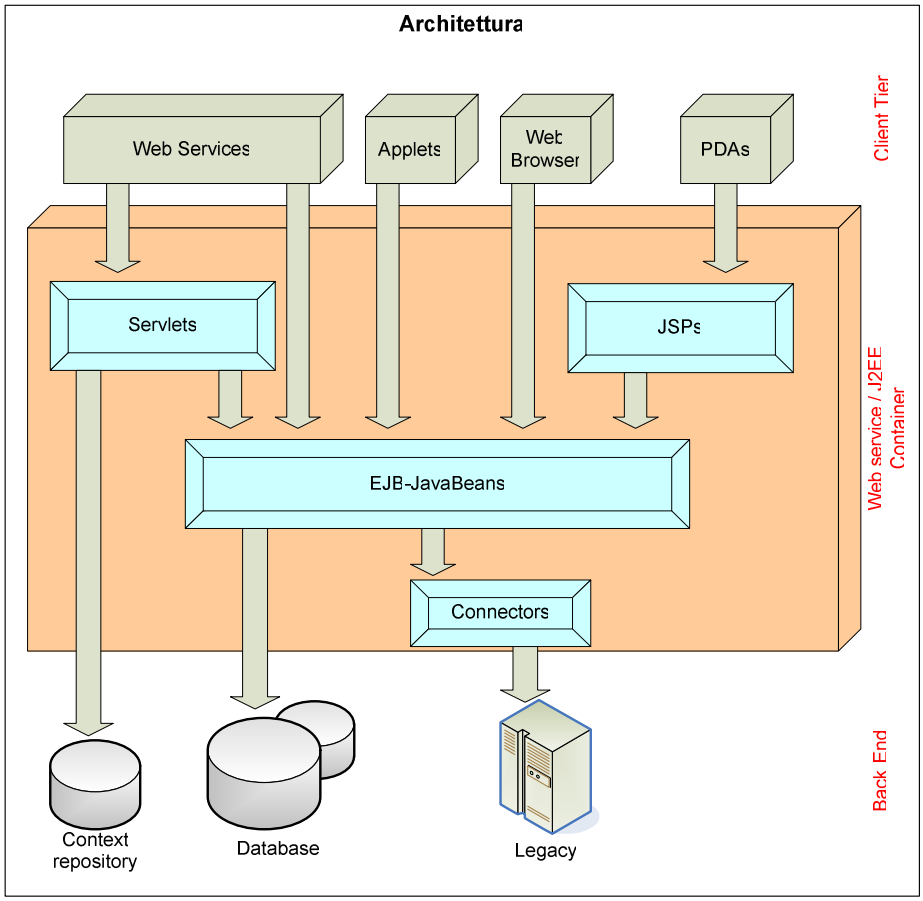

Figura 3.4 Architettura J2EE

Una tipica separazione in layer è la seguente:

- **Presentation Layer,** contiene i componenti che si occupano di gestire l'interfacciamento dell'applicazione verso l'utente. Per esempio in una web application i componenti utilizzati per il presentation layer potrebbero essere Java servlet, Java server pages, Java applet.
- **Business Logic Layer,** contiene i componenti che interagiscono tra di loro per risolvere un determinato problema dell'applicazione.

- **Data Layer,** è utilizzato dal business logic layer per effettuare i salvataggi di stato in maniera persistente. Un data layer è costituito essenzialmente da uno o più database.

Il vantaggio di una applicazione partizionata tramite layer è subito evidente: ogni layer venendo isolato dagli altri può essere gestito in maniera ottimale.

La separazione fisica dei layer rappresenta un'altra questione. In una architettura basata sul modello two-tier due layer sono fisicamente separati dal terzo, formando due tier (un tier rappresenta uno strato di suddivisione verticale. L'applicazione deve essere vista come un insieme di strati sovrapposti, ognuno dei quali è un tier.) fisicamente separati. In una architettura basata sul modello three-tier ogni layer è suddiviso in un tier separato dagli altri. In oguna di queste architetture la separazione tra i tier viene realizzata tramite physical boundaries (linee di demarcazione fisiche), come due macchine differenti, due processi differenti etc.

Quindi sono vari i componenti che vanno a costituire l'applicazione e possono essere suddivisi oltre che per il layer in cui si trovano anche per la ubicazione dove vengono eseguiti:

- Applicazioni client e applet sono componenti che vengono eseguiti sulla macchina client;
- Java servlet e JSP sono componenti web che vengono eseguiti sul server;
- EJB sono componenti business che vengono eseguiti sul server;

Il modello della sicurezza in J2EE, JAAS, permette di configurare un componente web o un enterprise bean in modo tale che le risorse del sistema siano accedute solo dagli utenti autorizzati.

3.2.2 JAAS (Java Authentication and Authorization Service)

Il Java Authentication and Authorization Service è un package opzionale introdotto con il Java 2 SDK v.1.3 e successivamente è stato integrato nel Java 2 SDK v.1.4.

Questa struttura è utilizzata per due scopi principali:

1. **Autenticazione** di utenti, per determinare in modo sicuro e fidato chi esegue il codice Java.
2. **Autorizzazione** di utenti, in modo da concedere loro i giusti permessi per eseguire le operazioni.

JAAS implementa una versione Java dello standard *Pluggable Authentication Model (PAM),* di conseguenza l'autenticazione viene fatta in modo pluggable e questo consente alle applicazioni di rimanere indipendenti dalla sottostante tecnologia di autenticazione.

Per autorizzare l'accesso alle risorse bisogna prima autenticare l'entità che la richiede, a tale entità viene associato il termine **Subject** il quale, prima di accedere alle risorse, deve essere autenticato.

Per autenticare un *Subject* JAAS lavora nel seguente modo (Figura 3.5):

1. l'applicazione istanzia il **LoginContext .**
2. il **LoginContext** richiama una **Configuration** per caricare i **LoginModule(s)** necessari per l'applicazione.
3. l'applicazione invoca il metodo *login()* del **LoginContext.**

4. il metodo *login()* invoca il **LoginModule** selezionato al passo 2 e attende che questo identifichi il soggetto. Se l'autenticazione ha successo il **LoginModule** associa al soggetto lo username e la password forniti dall'utente.
5. il **LoginContext** ritorna lo stato dell'autenticazione.
6. Se l'autenticazione si è conclusa con successo l'applicazione può recuperare il *Subject* dal **LoginContext**.

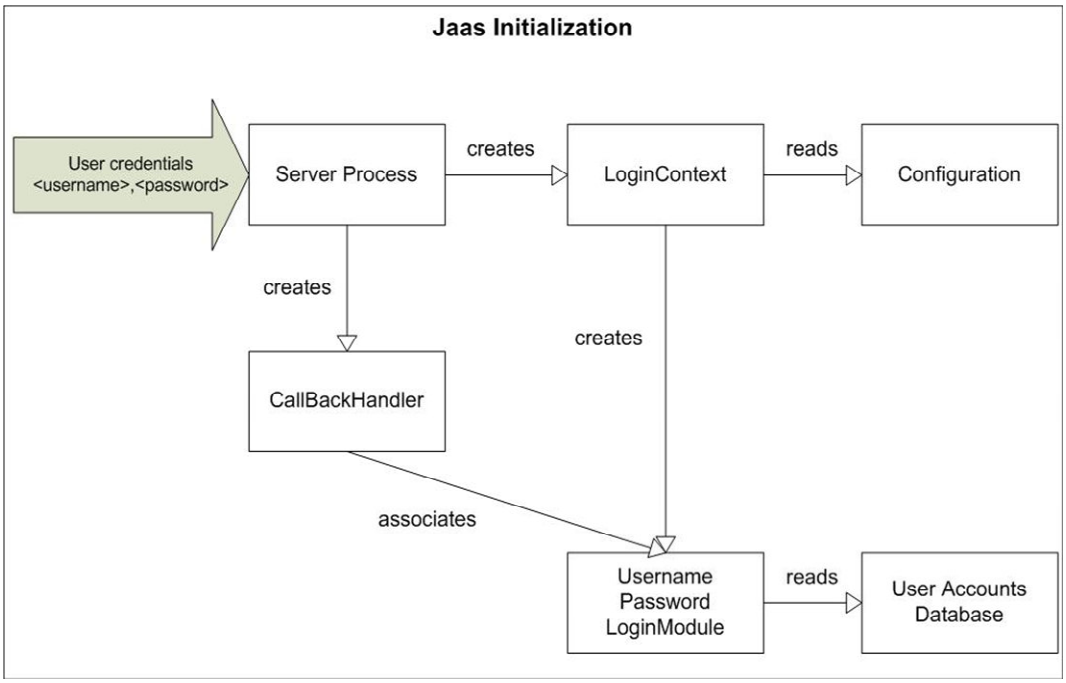

Figura 3.5 JAAS

Di rilievo è la presenza della **Configuration**, un nome al quale è associato il **LoginModule** voluto per l'applicazione specifica. E' infatti il **LoginModule** che definisce la tecnologia per effettuare l'autenticazione. Potendo richiamare diversi **LoginModule(s),** è possibile utilizzare definire diverse tecnologie senza dover modificare l'applicazione stessa.

Ad ogni *Subject* autenticato vengono associati una serie di **Principals** ognuno dei quali rappresenta un'identità per l'utente, ad esempio un utente puo avere un nome principale ed un numero di codice fiscale, così da poterlo distinguere da un altro *Subject*.

Dopo che l'utente è stato autenticato, il *subject* viene associato al contesto access-control per la fase di autorizzazione.

3.3 Modelli e strutture ad alto livello

Per l'implementazione del sistema sulla piattaforma J2EE è stato usato un particolare modello architetturale, il modello MVC.

L'uso del modello MVC è stato scelto dopo aver analizzato i requisiti e le specifiche del sistema. Inizialmente la scelta, ricaduta unicamente sull'utilizzo di Web Server, servlets e pagine JSP, ha evidenziato alcuni svantaggi. Vediamo insieme questi strumenti e la loro integrazione per giungere poi ad un unico modello unificante l'MVC.

3.3.1 Web Server e Web Applications

Un web server è un programma che, utilizzando il modello client/server e il protocollo HTTP, fornisce i file che costituiscono una pagina web agli utenti che ne fanno richiesta utilizzando un programma client: il browser. Ogni computer che contiene un sito web deve avere un programma web server.

La Web Application è una estensione dinamica del Web Server. Ci sono due tipi di web application:

- Orientata alla presentazione. Una Web Application orientata alla presentazione genera pagine Web dinamiche contenenti vari tipi di linguaggi di markup (Html, Xml etc) in risposta a richieste http.
- Orientata al servizio. Una Web Application orientata al servizio implementa il cuore di un raffinato Web Service. La Web Application orientata al servizio è spesso invocata da applicazioni orientate alla presentazione.

Nella Java 2 Platform, i *componenti Web* forniscono al Web Server possibilità di estensione dinamica. I *componenti Web* sono Java servlets e pagine JSP.

Questi *componenti web* sono supportati dai servizi di una piattaforma run-time chiamata *Web container*. Il *Web container* fornisce servizi come request dispatching, security, concurrency, gestione tempi di vita.

3.3.1.1 Ciclo di vita di una Web Application

Una Web Application consiste di componenti Web, di file statici come immagini e di classi e librerie. Il processo per creare ed eseguire le Web Application è diverso dalle tradizionali classi Java stand-alone.

Alcuni aspetti sul comportamento di una web-app possono essere configurati quando la web-app è già stata implementata. Le informazioni di configurazione sono mantenute in file di testo in XML chiamato *web application deployment descriptor*.

Il processo di creare, installare ed eseguire una Web application può essere riassunto nei seguenti passi:

- Implementare il codice dei web component (includendo possibilmente un descrittore di deploy).

- Collegare fra loro i vari web component della web application compresi i vari file statici(immagini etc).
- Installare la web application all'interno del web container.
- Accedere all'url che referenzia la web application.

3.3.1.2 Portabilità della Web Application

Per garantire la portabilità della web-app è necessario impacchettarla in un file *war* (Web Application Archive), simile ai package usati nelle comuni classi Java. In aggiunta ai Web Components , come adottato anche nella nostra soluzione per garantire un ulteriore livello di portabilità, è possibile inserire altri file:

- Classi utility server-side (database beans, shopping carts, ..etc). Queste classi si rifanno all'architettura JavaBean che vedremo più tardi.
- Contenuto static per la parte web presentation (file HTML, immagini, fogli di stile CSS..etc).
- Classi client-side (applets e utility class).

I Web Components ed il contenuto Web statico è spesso chiamato *Web resources*.

3.3.1.3 Configurare una Web Application

La web-app è configurata per mezzo di elementi contenuti all'interno dei Web application deployment descriptors. È possibile creare i descrittori sia manualmente con text editor oppure per mezzo di deploytool. Elementi fondamentali per un file descrittore sono:

- Prolog (fondamentale per un file XMl)
- Alias Path. Quando viene ricevuta una richiesta, il Web Server ha la necessità di sapere il percorso del Web component che deve gestirla. Questo è fatto mappando l'URL contenuto nella richiesta con un web component.

  http://<*host*>:8080/`context_root`/`alias_path`

- Context e Initialization Parameters. È possibile condividere oggetti nel contesto della web application. È possibile passare parametri al contesto o ad un Web Component. Per farlo è necessario aggiungere al descrittore gli elementi *context-param* e *init-param*.
- Filter mappings. Si usano filtri per decidere cosa filtrare di una richiesta alla web-app.
- Error mappings. È possibile mappare il codice di errore di ritorno di una http response o di una classe Java con una pagina di errore.
- Referencies to environment entries, resource environment entries, resources. Se la Web-app fà uso di riferimenti a database o alter risorse è possible inserirli nel descrittore dichiarando *env-entry* oppure *resource-env-ref* oppure *resource-ref*.

3.3.1.4 Deploying Web Applications

Ci sono diversi modi per fare il deploy permanente di un contesto di una web application su Tomcat mentre Tomcat è in funzione:

- Con il task deploy Ant:

```
<deploy url="url" path="mywebapp"
war="file:/path/to/mywebapp.war"
```

username="*username*" password="*password*" />

- Con un deploytool nel quale scegliere le operazioni da eseguire nel copiare il file war e nel creare il contesto nel web server container, Tomcat.

3.3.2 Servlets

Vediamo il primo dei due Web components di cui abbiamo accennato prima. L'uso di Java servlets forza una conversione delle richieste http in un object-oriented. Questa strategia favorisce lo sviluppatore a concentrarsi sulla applicazione stessa e non sui meccanismi http. Come per le richieste http, che utilizzano le cosìddette CGI (Common Gateway Interface) per estendere i meccanismi di richiesta e risposta , così Java utilizza gli strumenti servlets per risolvere le richieste e le risposte. Una richiesta è passata al server container, nel nostro caso a Tomcat, che si preoccupa di verificare la presenza della servlet richiesta; in caso negativo, cioè nessuna servlet registrata nel container (file web.xml) col nome presente nella richiesta, la richiesta è inviata al server http. È quindi il server container a gestire tutte le funzioni delle servlets. Il container crea, invoca e per ultimo distrugge la servlet.

Generalmente una servlet è una sottoclasse di javax.servlet.http.HttpServlet. Una servlet deve implementare quattro metodi, che sono invocati all'occorrenza dal container:

- **public void init(ServletConfig config) –** chiamato dal server container la prima volta che la servlet è creata e prima di processare ogni richiesta.

- **public void doGet(HttpServletRequest request, HttpServletResponse response) –** chiamato per processare una specifica richiesta che usa http GET protocol, genera una corrispondente risposta dinamica.
- **public void doPost(HttpServletRequest request, HttpServletResponse response) –** chiamato per processare una specifica richiesta che usa http POST protocol, genera una corrispondente risposta dinamica.
- **public void destroy() –** chiamato dal server container quando la servlet ha terminato il suo servizio, quando il server container viene chiuso o quando quando la web application viene dimessa.

3.3.3 JSP

Altro importante Web component della web application sono le pagine JSP. Le JSP (Java Server Page) sono "inside-out servlets" che rendono più facile la creazione ed il mantenimento di pagine dinamiche. Invece di mettere ciò che si vuole visualizzare all'interno di un comando Java print in una response http, nelle JSP tutto è scritto nella response http eccetto ciò che è posto all'interno di particolari statement Java. Le pagine JSP sono perciò più indicate per una logica di presentazione mentre le servlets per una logica computazionale.

Con le Jsp è possibile iniziare a scrivere la pagina in standard HTML e successivamente aggiungere le caratteristiche dinamiche utilizzando statement in Java o particolari tag JSP.

3.3.3.1 La scelta di Servlets e JSP

Rispetto alle CGI le servlets permettono un esecuzione che grava meno sul sistema; ad ogni richiesta http non viene iniziato un nuovo processo ma un nuovo Java thread, molto meno laborioso e pesante da elaborare rispetto ad un processo di sistema. Le servlets, a differenza delle CGI, tra una request e l'altra rimangono in memoria non aggravando la computazione della cache. Tra più servlets è possibile condividere informazioni e sessioni garantendo una migliore gestione e connessione ai database. Le servlets essendo un componente Java garantiscono la portabilità su qualsiasi macchina che monta una JVM.

Rispetto alle CGI le pagine JSP garantiscono la dinamicità dei propri contenuti. Rispetto alle pagine in php, che ugualmente generano pagine dinamiche , hanno il vantaggio di essere scritte in Java senza dover costringere il programmatore a usare nuove regole sintattiche. Rispetto ad una servlet la JSP, anch'essa tradotta e trattata come una servlet internamente al sistema applicazione, è più conveniente da scrivere suddividendosi in parte in codice html ed in parte in codice Java; la parte html può essere lasciata ad un programmatore esperto in html che utilizza un particolare tool e successivamente l'esperto in JSP può mettere mano al codice Java senza doversi preoccupare del resto.

Utilizzare unicamente questi due strumenti non permette però una completa portabilità e scalabiltà dell'applicazione stessa. Infatti ogni connessione con database dovrebbe essere fatta dalla servlet; questo comporta che l'aggiunta di un nuovo servizio corrisponda a una reimplementazione e ricompilazione di tutte le servlets coinvolte per modificare cicli, controlli e path.

3.3.4 JavaBean

Subentra così l'utilizzo di un altro strumento, i JavaBean. I JavaBean sono delle classi Java che definiscono un insieme di metodi di lettura e scrittura sui propri membri privati. Questo permette alle servlets sovrastanti di demandare ogni operazione sui dati ai Javabean.

"*JavaBean components are Java classes that can easily reused and composed together into applications."* È la definizione iniziale fornita da Sun Microsystem (nelle specifiche J2EE) che ci lascia capire di come nella progettazione dei JavaBean, la riusabilità e modularità siano concetti fondamentali.

I JavaBean sono delle classi Java che devono rispettare le seguenti regole:

- Avere un costruttore privo di argomenti o esserne addirittura privo,(quando una classe Java omette la dichiarazione di un costruttore, le viene fornito automaticamente uno privo di argomenti).

- Possono avere delle proprietà a cui è possibile accedere con i classici metodi:

  public void setPropertyName (PropertyType value);

  per impostare la proprietà, mentre per ottenere un valore:

  public PropertyType getPropertyName();

  I metodi di accesso a queste proprietà devono essere pubblici, ma una proprietà non è detto che debba per forza avere il metodo *set* e il metodo *get*.

3.3.5 Paradigma MVC

L'MVC (Model View Controller) è un architectural pattern e non un design pattern poiché i vari ruoli del pattern possono essere ricoperti da insiemi di classi anziché da singole classi.

Nell'MVC, il flusso dell'applicazione è mediato da un *Controller* centrale. Il *Controller* delega le richieste, nel nostro caso http requests, ad un *Handler* (gestore dell'evento). Gli *Handlers* sono legati ad un *Model*, ed ogni *handler* agisce da tramite tra la request ed il *Model* (Figura 3.6).

Il *Model* rappresenta, o incapsula, la logica business o lo stato dell'applicazione. Solitamente il controllo è indirizzato attraverso il *Controller* ad un appropriato *View*. Il cambiamento del flusso delle istruzioni e del controllo è determinato consultando il configuration file visto precedentemente nel paragrafo 3.3.1.3. Questo garantisce un lasco accoppiamento tra il *View* ed il *Model*, rendendo più semplice la creazione ed il mantenimento dell'applicazione.

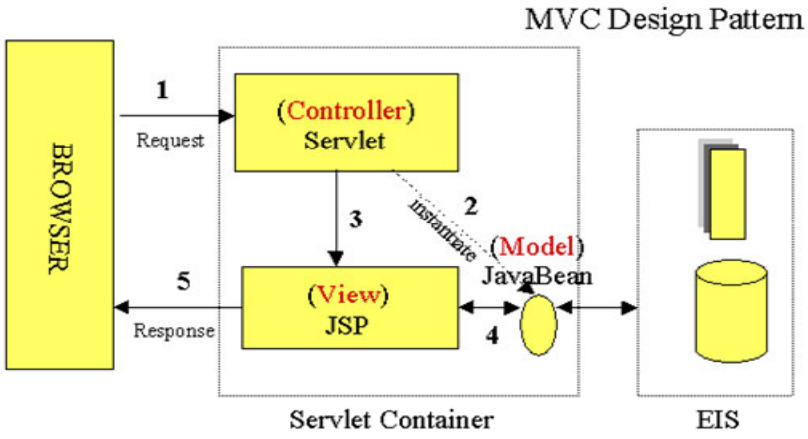

Figura 3.6 MVC pattern

3.3.5.1 Il Model

La porzione *Model* (Figura 3.7)di un sistema basato su MVC può essere spesso suddiviso in due maggiori sottosistemi:

- **Stato interno** del sistema.
- **Azioni** che possono essere adottate per modificare lo stato.

Molte applicazioni rappresentano lo stato del sistema con un set di uno o più JavaBean (3.3.4). Le proprietà del bean rappresentano i dettagli dello stato del sistema. In base alla complessità dell'applicazione, queste informazioni possono essere contenute all'interno del bean stesso oppure all'interno di altre risorse come database, EJB o altro.La gran parte delle applicazioni rappresentano il set delle possibili operazioni come metodi che possono essere richiamati in un bean.Una parte più piccola di queste applicazioni, invece integra all'interno delle classi *Action* questo set di operazioni. Questo può essere utile quando la logica di sviluppo è molto semplice ed il riutilizzo del logica di business non è contemplata.

3.3.5.2 Il View

La porzione *View* (Figura 3.7) è nella gran parte delle volte sviluppata utilizzando JavaServer Page (JSP) tecnology (3.3.3).

3.3.5.3 Il Controller

La porzione *Controller* (Figura 3.7) dell'applicazione è incentrata nel ricevere request da un client (tipicamente un utente che utilizza un web browser), decidere quale tipo di logica da utilizzare (Model), e successivamente delegare la responsabilità di visualizzare i risultati all'appropriato componente View.Il componente primario del *Controller* è una classe servlet , *ActionServlet*. Questa servlet è configurata definendo un set di *ActionMappings* (3.3.1.3).

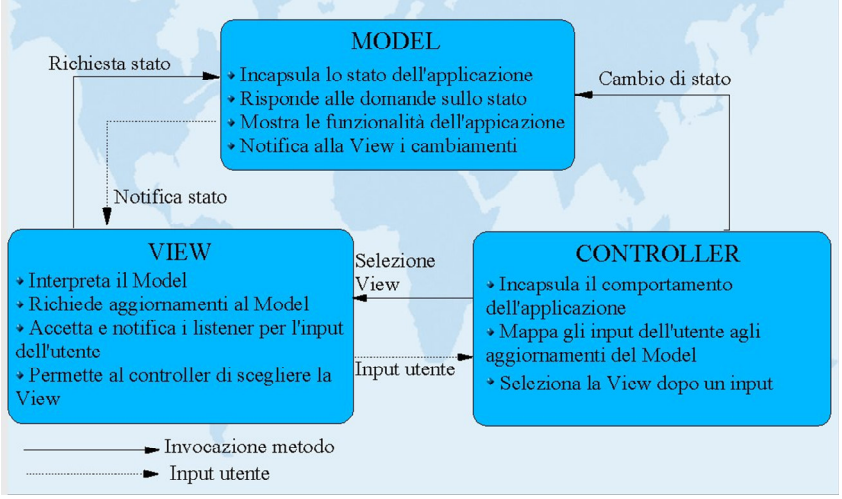

Figura 3.7 Flusso delle informazioni in MVC

3.4 Installazione del sistema

3.4.1 Architettura del sistema

Per implementare il sistema , sono stati scelti i seguenti componenti (Figura 3.8):

- sistema operativo open source **Linux**;
- **Java 2 Standard Edition,** http://java.sun.com/ (versione 1.4.2 06);
- **Apache Jakarta Tomcat,** http://jakarta.apache.org/tomcat/index.html, Servlet/JSP container (version 5.5);

- **Mysql** (versione) , database utilizzato dalla Web Application per la memorizzazione delle informazioni;
- **Eclipse,** un open source tool platform IDE per lo sviluppo nello specifico del codice Java;
- **Plugin Lombotz** per Eclipse http://www.jboss.org/;

Il codice utilizzato per i componenti dinamici della web application è il Java, e per questo è stato necessario installare sulla macchina il pacchetto J2SE per avere disponibile l'interprete java JVM (Java virtual machine) e tutto l'insieme delle API SUN.

Apache Jakarta Tomcat permette l'esecuzione della web application facendo da contenitore per pagine JSP e Servlet.

Mysql è un DBMS open source all'interno del quale vengono memorizzate le informazioni necessarie per l'autenticazione e per i servizi.

Eclipse è un ambiente di sviluppo integrato open source che ci ha permesso di generare pagine JSP e Servlet con codice Java. L'utilizzo del plugin Jboss per Eclipse, ci ha permesso di automatizzare la fase di compilazione e deploy della web application all'interno del Web server Tomcat.

In figura è mostrata l'architettura del sistema.

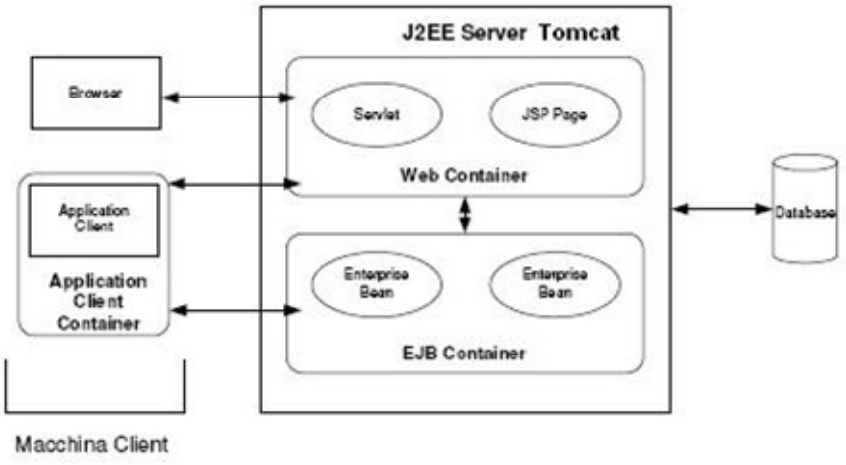

Figura 3.8 Architettura del sistema

3.4.2 Installazione e configurazione del software

Recuperare il pacchetto dei sorgenti di Apache Tomcat dal sito ufficiale http://jakarta.apache.org/site/downloads/downloads_tomcat-5.cgi, e decomprimerlo sulla macchina.

tar -xvzf jakarta-tomcat-5.5.9-admin.tar.gz

Recuperare il pacchetto dei sorgenti di J2SE dal sito ufficiale della SUN http://java.sun.com, ed eseguire l'installazione.

Recuperare il pacchetto dei sorgenti di Mysql dal sito ufficiale http://www.mysql.com, ed eseguire l'installazione.

tar -xvzf mysql-debug-5.0.4-beta-pc-linux-gnu-i686.tar.gz

Scaricare ed installare i sorgenti del Connector/J, driver Java, dal sito ufficiale http://dev.mysql.com/downloads/connector/j/3.1.html, necessari a commutare chiamate JDBC (Java Database Connectivity) nel protocollo di rete supportato da Mysql.

tar -xvzf mysql-connector-java-3.1.8a.tar.gz

E' possibile impostare particolari configurazioni di sicurezza nel file proprietà di sicurezza java.security nella cartella lib/security di J2RE. Per questa web application è necessario aggiungere la seguente riga di codice :

login.config.url.1=file:/home/allusers/jakart-tomcat-5.5.7/auth.conf

(i percorsi specificati sono chiaramente relativi alla macchina dove gira la web application)

Nel file auth.conf si specifica il package e quindi la classe LogModule necessaria per il modello Jaas: Login {Sicurezza.LogModule required debug=true; };