SISTEMA D'AUTENTICACIÓ ONE-TIME PASSWORD (OTP) PER A MÒBILS

Memòria del projecte

d'Enginyeria en Informàtica

realitzat per

Natalia Miguel Álvarez

i dirigit per

Helena Rifà Pous

Bellaterra, 13 de Juny de 2008

El sotasignat, Helena Rifà Pous

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva

direcció per Natalia Miguel Álvarez

I per tal que consti firma la present.

Signat: Helena Rifà Pous

Bellaterra, 13 de Juny de 2008

Índex

Índex de Figures

1. Introducció

El número d'identitats i credencials usades per l'autenticació en els serveis existents en Internet ha augmentat considerablement. Avui en dia, els sistemes d'autenticació requereixen un nom i una contrasenya estàtica. Un usuari s'ha de recordar de totes les contrasenyes dels diferents serveis en els que està registrat i, com a conseqüència, ja sigui per comoditat o per inconsciència, aquestes contrasenyes poden acabar sent sempre les mateixes. També existeix una limitació en la mobilitat i, per tant, en la seguretat. Els sistemes actuals només permeten guardar les contrasenyes en el propi ordinador personal, dispositiu que pot mirar una tercera persona, i si volem utilitzar-les en un altre lloc hem de fer ús de la nostra memòria.

Amb l'aplicació que volem realitzar, estalviarem als usuaris haver de recordar-se de tots els serveis en els que s'han registrat així com de totes les contrasenyes escollides per poder-hi tenir un accés. De la mateixa manera evitarem l'ús de les contrasenyes estàtiques i augmentarem la seguretat en les connexions a través d'Internet. Tot això amb una aplicació senzilla, fàcil i ràpida d'usar instal·lada en el telèfon mòbil, aparell que s'està fent indispensable en la societat actual.

1.1. Objectiu/s del projecte

Aquest projecte consisteix en fer l'anàlisi, disseny i implementació d'un sistema d'autenticació a través de contrasenyes d'un sol ús (One Time Password -OTP-) per a dispositius mòbils.

Un usuari instal·larà el programa que genera les contrasenyes On-Time Password al seu mòbil. Mitjançant un número o nom identificatiu podrà generar una contrasenya que utilitzarà per identificar-se en una pàgina web i poder tenir privilegis a l'hora d'interactuar amb ella. El nostre client (dispositiu mòbil) generarà a partir d'una llavor, la contrasenya adient. Gràcies a aquesta llavor podrem obtenir una contrasenya diferent en cada ocasió. Quan l'usuari introdueixi el seu nom identificatiu a la pàgina web, en el servidor es generarà la mateixa contrasenya a partir de la mateixa llavor que haurà d'estar sincronitzada a la del mòbil. Així comprovarem la validesa de l'autenticació.

OTP és l'acrònim en anglès de One-Time Password. Aquesta contrasenya té el propòsit de dificultar l'accés no autoritzat a recursos protegits com, per exemple un compte d'usuari en un sistema de computació. Es genera una nova contrasenya cada cop que es necessiti. Amb això reduïm el risc que algú sense autorització descobreixi la contrasenya i l'usi. Són immunes als atacs que es poden fer a les contrasenyes estàtiques.

En aquest projecte s'utilitzarà el telèfon mòbil (Figura 1.1) com a dispositiu hardware capaç de generar una clau d'accés en funció de determinats paràmetres. Ha de ser de fàcil ús perquè pugui ser utilitzat per usuaris no-tècnics, integrable en els sistemes actuals de que disposen els dispositius mòbils i que sigui segur per evitar els problemes de seguretat abans esmentats.

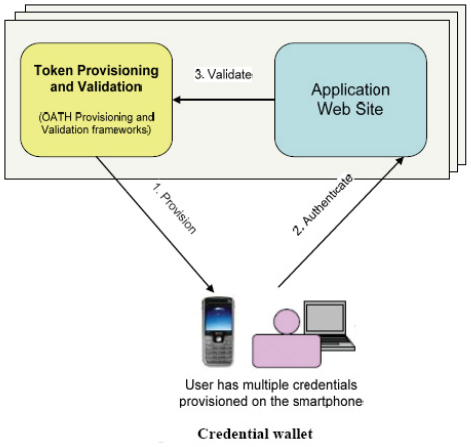


Figura 1.1. Utilització d'un telèfon mòbil per autenticar-se

Per a la realització del projecte, intentarem seguir les pautes de l'OATH (Initiative for Open Authentication) [1]. La iniciativa per l'autenticació oberta OATH resulta d'un treball col·laboratiu amb els líders de la industria de les TI per poder proporcionar una referència universal de l'arquitectura d'autenticació forta a través de tots els usuaris i tots els dispositius, sobretot les xarxes. Utilitzant estàndards oberts, OATH oferirà més opcions de hardware i, per tant, menor cost degut a una major competència entre fabricants. Permetrà als clients substituir diferents sistemes de seguretat propietari existents que tenen una elevada complexitat i cost.

El treball es basa els següents principis:

- **Especificació oberta i lliure**: OATH intenta establir una especificació oberta i lliure per l'autenticació forta basada en els estàndards oberts en tot el possible.
- **Dispositius innovadors**: especificar components de baix cost i dispositius d'autenticació multifunció. Transformar els dispositius mòbils actuals en dispositius d'autenticació forta.
- **Suport a la plataforma nativa**: facilitar el suport natiu per dispositius d'autenticació forta dels usuaris i en l'aplicació i gestió de plataformes d'identitat.
- **Mòduls interoperables**: permetre les millors solucions a través d'un marc d'interoperabilitat dels components. D'aquesta manera l'usuari de l'organització serà capaç de desplegar mòduls i dispositius de diferents proveïdors.

1.2. Planificació

En aquest apartat veurem la planificació temporal del projecte. A principis de gener es va realitzar una planificació (Figura 1.2) que ha quedat modificada (Figura 1.3) per diversos problemes en la realització del treball.

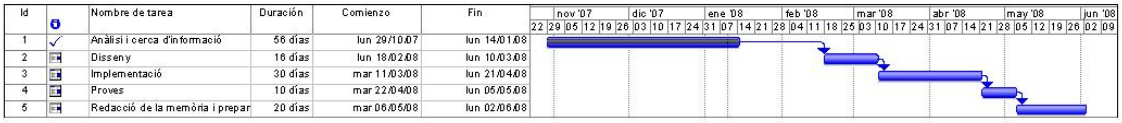
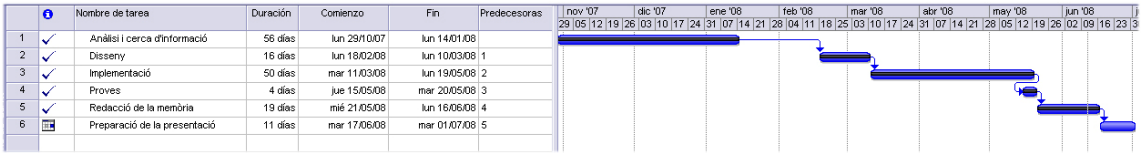


Figura 1.2. Primera planificació temporal del projecte



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Anàlisi i cerca d'informació	56 días	lun 29/10/07	lun 14/01/08	
2	Disseny	16 días	lun 18/02/08	lun 10/03/08	1
3	Implementació	50 días	mar 11/03/08	lun 19/05/08	2
4	Proves	4 días	jue 15/05/08	mar 20/05/08	3
5	Redacció de la memòria	19 días	mié 21/05/08	lun 16/06/08	4
6	Preparació de la presentació	11 días	mar 17/06/08	mar 01/07/08	5

Figura 1.3. Planificació temporal final del projecte

A continuació detallem les tasques en les que s'ha dividit el projecte a partir de la planificació final:

1. **Anàlisi i cerca d'informació:**
   En aquesta primera tasca s'ha realitzat un primer estudi de viabilitat del projecte gràcies a la cerca d'informació sobre el tema i el seu posterior anàlisi.

2. **Disseny:**
   Realització el diagrama UML i el protocol per la implementació del programa.

3. **Implementació:**
   Elaboració de les versions adients del programa. Aquesta tasca havia de durar uns 30 dies però es va allargar fins els 50 degut a diversos problemes amb algunes incompatibilitats entre llibreries i telèfons mòbils.

4. **Proves:**
   Passar diversos tests al codi que ens dóna l'anterior tasca i que detecti la major part possible d'errades. Al durar més temps la tasca d'implementació vam haver de solapar aquesta amb la de proves deixant un dia més per les proves de la implementació final. La durada també es va veure reduïda ja que al final no van caldre tants dies per fer-ho.

5. **Redacció de la memòria:**
   Temps per elaborar la memòria on queden redactats tots els passos explicats. Com les tasques anteriors es van retardar, aquesta també. Tanmateix, el dia de finalització havia de ser el mateix.

Les tasques 4 i 5 al final es van dividir en dos ja que el seu ordre de realització era important degut els retards en la planificació inicial.

2. Estat de l'art

En l'actualitat, existeixen diferents solucions per l'autenticació remota d'usuaris com poden ser les paraules de pas, els sistemes repte/resposta o els certificats basats en criptografia de clau pública.

El sector bancari ha sigut, junt amb l'acadèmic, el que més aviat va acollir els conceptes de criptografia de clau pública i de certificació digital com a components de seguretat. El sector financer també ha sigut promotor de la criptografia de clau simètrica des de molt abans, adoptant el DES (Data Encryption Standard) [2] com a base dels seus sistemes de xifrat en tots els dispositius relacionats amb mitjans de pagament com poder ser els caixers automàtics.

En canvi, en el desplegament del canal Internet, la relaxació excessiva en l'adopció de mesures de seguretat ha retardat el consens en l'adopció d'estàndards comuns fins que s'han produït una multitud d'incidents associats als termes com "phishing", "pharming", "keyloggers" i "trojans" que estan creant una veritable alarma social i posen en perill no només la credibilitat del nou canal, sinó també del bon nom de la banca en el seu conjunt, amb la pèrdua de conceptes com el de "confiança", que són la base del negoci bancari.

Per això s'han de prendre certes mesures. Cada entitat les escull com vol: unes amb missatges a mòbil, unes altres amb targetes amb coordenades, algunes amb teclats a la pantalla (cada cop menys degut a la seva poca seguretat), altres amb dispositius de clau (OTP) i les més sofisticades amb certificats electrònics.

Diferents bancs en Espanya estan donant suport a aquest mètode de seguretat. *La Caixa* ha creat *CaixaProtect®* que es basa en la creació d'una OTP que es enviada al mòbil en forma de missatge de text. *Bancaja* i *Caixa Galícia* també estan començant a utilitzar aquest servei. Això dóna més tranquil·litat a l'usuari a l'hora de comprar a través d'Internet (e-comerç) i no li reporta cap cost extra. Alhora ofereix un major control de les operacions realitzades ja que també avisa, amb missatge de text, de les operacions fetes amb imports molt elevats.

Hi ha moltes empreses que es dediquen a fabricar aquests tipus d'aparells per a la generació de contrasenyes d'un sol ús:

Aladdin [3] ens ofereix eTokens en forma de pendrive (Figura 2.1) i d'smartcard. Poden ser utilitzar en la majoria de sistemes operatius (Windows, Linux i Mac OS); suporten les següents API's i estàndards: PKCS#11 v2.01, CAPI (Microsoft Crypto API), Siemens/Infineon APDU commands, PC/SC, X.509 v3 certificate storage, SSL v3, IPSec/IKE; utilitzen els següents algorismes de seguretat: RSA 1024-bit / 2048-bit, DES, 3DES (Triple DES), SHA1; l'algorisme de seguretat de l'OTP es basa en tots els casos en les especificacions de l'OATH i, segons el dispositiu que tinguem, aquest pot portar alguns gigabytes de memòria flash. La bateria és molt duradera.

Figura 2.1. eTokens d'Aladdin

EnTrust [4] disposa de l'IdentityGuard Mini Token semblant als d'Aladdin però sense funcionar per USB (Figura 2.2). Es basa en les especificacions de l'OATH i suporta els algorismes DES/3DES amb sincronització de temps, és molt petit, la bateria dura entre sis i vuit anys i és submergible fins a un metre. Utilitza un software creat per ells, l'Entrust IdentityGuard 8.1.

Figura 2.2. Dispositius Entrust

Figura 2.3. Dispositius ActivCard

ActivCard [5] fabrica dispositius token i calculadores generadores d'OTP (Figura 2.3). *ipsCa* [6] és la companyia especialitzada en serveis de certificació digital i firma electrònica que ha començat la comercialització d'aquests tokens. Les especificacions son molt semblants a les dels tokens anteriors.

La desavantatge d'aquests dispositius és que l'usuari pot tenir un de diferent per cada servei a Internet en el que necessiti autenticar-se. A més, és necessari que sempre els porti i això pot resultar molt incòmode. Per aquest motiu s'han començat a implementar aplicacions per telèfons mòbils. Malgrat això, no hi ha gaire projectes realitzats. Existeixen diferents protocols d'autenticació [7] i [8] i alguna aplicació implementada [9]. En cinquè capítol repassarem el funcionament dels dos protocols.

3. Tecnologies

En aquest capítol analitzarem les tecnologies necessàries per fer les aplicacions, tant del telèfon mòbil com les del servidor.

3.1. Tecnologies disponibles per l'aplicació del mòbil

3.1.1. Java 2 Platform, Micro Edition (J2ME)

Java Platform, Micro Edition (Java ME) [10] proveeix un entorn robust i flexible per aplicacions que funcionen en dispositius mòbils i altres sistemes encastats (embedded) com poden ser els telèfons mòbils, les personal digital assistants (PDAs), descodificadors de televisió o impressores. J2ME inclou interfícies d'usuari flexibles, seguritat robusta, protocols de funcions de xarxa i suport per aplicacions en xarxa i off-line que es poden descarregar dinàmicament. Les aplicacions basades en J2ME són compatibles en molts dispositius.

Aquesta edició de Java utilitza una màquina virtual que requereix pocs Kilobytes de memòria per funcionar i un petit i ràpid col·lector de brossa (garbage collector). J2ME representa una versió simplificada de J2SE (Java Standard Edition) tal com es pot veure a la Figura 3.1. Sun, que va ser el creador de Java, va separar aquestes dues versions ja que J2ME estava pensat per dispositius amb limitacions de procés i capacitat gràfica.

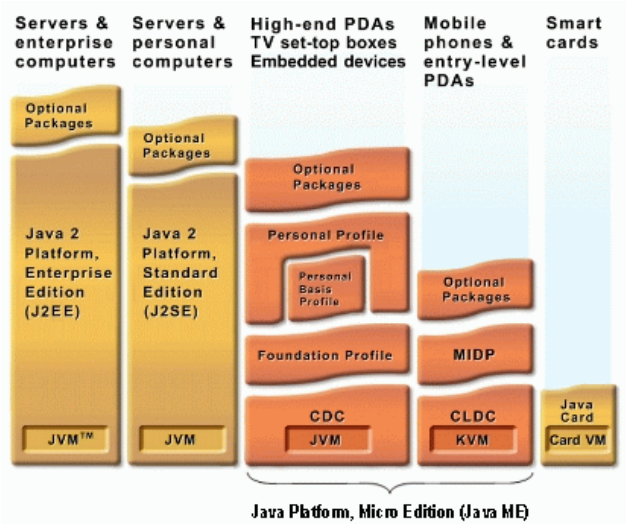


Figura 3.1.Components de la tecnologia Java ME

Un entorn d'execució determinat de J2ME es composa d'una selecció d' una configuració, un perfil, una màquina virtual i paquets opcionals.

3.1.1.1. Configuracions

Amb el temps la plataforma Java ME ha estat dividida en dues configuracions bàsiques, la primera s'ajusta als petits dispositius mòbils (Connected Limited Device Configuration) i, la segona, que s'orienta més cap a dispositius com els telèfons intel·ligents i els descodificadors (Connected Device Configuration).

Les característiques típiques dels dispositius que s'ajusten a cada configuració són:

CLDC (Connected Limited Device Configuration)

- Disposar entre 160 Kb i 512 Kb de memòria total disponible. Com a mínim s'ha de disposar de 128 Kb de memòria no volàtil per la Màquina Virtual i les biblioteques i 32 Kb de memòria volàtil per la Màquina Virtual en temps d'execució.
- Processador de 16 o 32 bits amb almenys 25 MHz de velocitat.
- Oferir un baix consum.
- Tenir connexió a algun tipus de xarxa amb connexió intermitent i ample de banda limitat (uns 9600 bps).

CDC (Connected Device Configuration)

- Processador de 32 bits.
- Disposar de 2Mb o més de memòria total (RAM i ROM).
- Connexió de xarxa persistent i amb un gran ample de banda.

3.1.1.2. Perfils

El perfil J2ME defineix les APIs (Application Programming Interface) que controlen el cicle de vida de l'aplicació, de la interfície d'usuari, etc. El conjunt d'APIs incloses en un perfil està orientat a un àmbit d'aplicació determinat i donen a la configuració una funcionalitat específica.

Per la configuració CDC tenim els següents perfils: Foundation Profile, Personal Profile i RMI Profile i, per la configuració CLDC: PDA Profile i Mobile Information Device Profile (MIDP).

Ens centrarem en el perfil MIDP ja que és l'utilitzat per crear aplicacions per a telèfons mòbils:

MIDP

MIDP [11] va ser el primer perfil definit per aquesta plataforma, així com CLDC va ser la primera configuració definida per J2ME. Els tipus de dispositiu que s'adapten a les característiques per les quals està orientat són telèfons mòbils, buscapersones (pagers) o PDAs de gamma baixa amb connectivitat. La versió més actual és la 2.1 però la suportada per la majoria de mòbils actuals és la 2.0 que especifica els paquets següents:

- java.lang: Classes referents a la màquina virtual.
- java.util: Classes d'utilitat estàndard.
- javax.microedition.io: Conté les classes específiques per E/S.
- javax.microedition.lcui: Conté les classes per les GUI.
- javax.microedition.lcui.game: Una API de joc destinada a l'animació 2D.
- javax.microedition.media: Conté les classes base de la reproducció de música multimèdia.

- javax.microedition.media.control: Defineix els tipus específics de control per un reproductor.
- javax.microedition.midlet: Conté les classes bàsiques per les aplicacions Java ME.
- javax.microedition.pki: APIs autenticades per connexions segures.
- javax.microedition.rms: Proveeix una forma persistent d'emmagatzematge per les aplicacions Java ME.

Per un entorn CLDC i MIDP, igual que el de la Figura 3.2, que és com estan implementats quasi tots els dispositius mòbils actuals, es genera un MIDlet. Un MIDlet és l'aplicació creada pel desenvolupador de software Java ME, com pot ser un joc, una aplicació per entorns de negoci o altres característiques de mòbil.

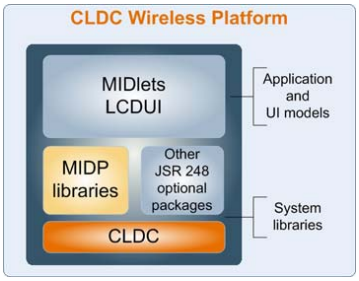


Figura 3.2. Arquitectura de la configuració CLDC

3.1.1.3. Màquina virtual

KVM

El seu nom prové de Kilobyte ja que fa referència a la baixa ocupació de memòria, entre 40 KB i 80 KB. És una implementació de Màquina Virtual reduïda i especialment orientada a dispositius amb baixes capacitats computacionals i de memòria. Està escrita en C i va ser dissenyada per ser:

- Petita, amb carrega de memòria entre 40 i 80 KB depenen de plataforma i opcions de compilació.
- Alta portabilitat.
- Modulable.
- El més completa i rapida possible.

Aquesta màquina virtual és la que tenen els dispositius amb configuració CLDC com són els mòbils o PDA's.

CVM

La CVM (Compact virtual Machine) és la màquina virtual de referència per les configuracions CDC, explicats en el següent punt. Està orientada a dispositius Electronics amb processadors de 32 bits de gamma alta i al voltant de 2 MB o més de memòria RAM.

3.1.1.4. Paquets opcionals

Existeixen molts paquets opcionals, com poden ser el JSR 82 per les protocol Bluetooth o, el JSR 177 per les connexions segures i autenticades. Ens centrarem en aquest últim paquet.

Secure and Trust Services API (SATSA)

El propòsit d'aquest JSR [12] és especificar una col·lecció d'APIs que proveeixen serveis de seguretat i autenticació per integrar Elements de Seguretat (SE). Un SE, component en un dispositiu J2ME, proveeix dels següents beneficis:

- Emmagatzematge segur per protegir dades, com claus privades d'usuaris, certificats de clau pública, informació personals, etcètera.
- Operacions criptogràfiques per suportar protocols de pagament, integritat de dades i confidencialitat de les dades.
- Un entorn d'execució segur per implementar característiques de seguretat. Les aplicacions J2ME poden dependre d'aquestes característiques per manipular serveis afegits com la identificació i autentificació d'usuaris o un pagament bancari.

L'API en aquesta especificació està definida per quatre paquets opcionals que poden ser implementats independentment:

- **SATSA-APDU**

  Defineix una API per suportar comunicacions amb aplicacions targeta intel·ligent utilitzant el protocol APDU (Application Protocol Data Unit). Per a això s'ha de crear una APDUConnection que serà la responsable d'intercanviar les APDUs codificades en el format ISO7816-4.

- **SATSA-JCRMI**

  Defineix un client per una Java Card RMI que permet a una aplicació J2ME invocar un mètode d'un objecte Java Card remot. El paquet JCRMI permet la comunicació amb una targeta intel·ligent fent ús del protocol del mateix nom. Aquest protocol permet que les aplicacions que no estan emmagatzemades en la targeta utilitzin objectes que sí ho estan.

- **SATSA-PKI**

  Defineix una API que suporta a nivell d'aplicació la firma d'una signatura digital (però no la verificació) i la gestió bàsica de credencials d'usuaris. Permet a les aplicacions J2EM sol·licitar a la targeta intel·ligent operacions de signatura digital. Inclou alguns mètodes de gestió bàsica de certificats. Una aplicació J2ME utilitza el mètode CMSMessageSignatureService per a signar missatges amb una clau privada. Aquests missatges poden ser signats amb l'objectiu d'aconseguir autenticació o no repudi.

- **SATSA-CRYPTO**

  Defineix un subconjunt de l'API criptogràfica de J2SE. Aquest paquet proveeix a les aplicacions J2ME d'una sèrie d'eines criptogràfiques que els permeten realitzar operacions com resum digital, signatura digital o xifrat.

3.1.2. J2MEMicroDB

J2MEMicroDB [13] és un projecte *Open Source* desenvolupat a la Universitat Politècnica de Catalunya, que té com a principal objectiu proporcionar als programadors de J2ME APIs per gestionar una base de dades relacional (limitada) en un dispositiu mòbil.

Aquest projecte consta de diverses parts:

- J2MELSDATALIB
- J2MESDLIB
- DBAccessRemot
- J2ME SQL Engine

La llibreria utilitzada ha sigut la J2MESDLIB que permet treballar amb taules relacionals en dispositius mòbils. Aquesta API proporciona els mètodes necessaris per poder crear, modificar i esborrar taules i el contingut de les mateixes.

3.1.3. Lightcrypto

LightCrypto [14] és un conjunt de classes Java *Open Source* amb rutines criptogràfiques bàsiques usant l'API de BouncyCastle de baix pes (BouncyCastle lightweight API).

Utilitza una petita quantitat de memòria per poder-la fer servir amb Java 2 Micro Edition, applets de navegadors, en petits dispositius Java com les PDA's o telèfons mòbils o en altres situacions on el JCE (Java Criptographic Extension) és massa pesat pel treball a realitzar.

Té les següents característiques:

- Crea digests (resums) de cadenes de caràcters i fitxers amb el algorismes hash MD5 o SHA.
- Xifra/Desxifra cadenes de caràcters i fitxers amb PBE (Password-Based Encryption).
- Xifra/Desxifra cadenes de caràcters i fitxers usant l'algorisme AES (implementació de baix pes) amb qualsevol longitud de clau (per defecte són 128 bits) en el mode CBC (Cipher-Block Chaining) amb el padding PKCS12.
- Xifra/Desxifra qualsevol flux de dades d'entrada (com els dels sockets) amb el xifratge de flux de dades RC4.
- Crea MACs (Message Authentication Code) de texts i fitxers amb HMAC (HMAC-SHA1) o CBC-BlockCipherMAC amb un vector d'inicialització IV.
- Xifra claus simètriques emmagatzemades i protegeix amb una paraula de pas (PBE).
- Les funcions de hash/xifrar/desxifrar/PBE/mac/hmac dades estan a HSQLDB, que és un motor de base de dades encastable usant consultes SQL.

3.2. Tecnologies disponibles pel servidor

3.2.1. Java 2 Standard Edition

Java 2 Standard Edition, Java Development Kit (JDK) i Standard Development Kit (SDK) [15] són noms pel mateix component i inclouen l'API de Java, el JRE (Java Virtual Machine), el compilador de Java i altres funcionalitats definides per *Sun Microsytems*.

A la Figura 3.3 podem veure on estan aquestes parts dins de l'arquitectura de J2SE.

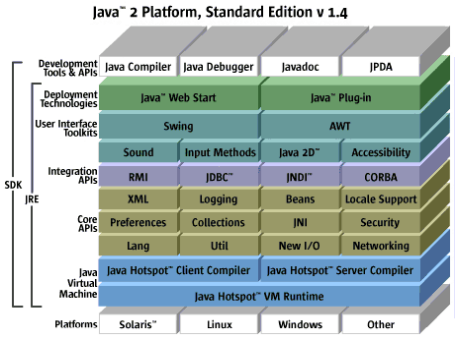


Figura 3.3. Arquitectura de J2SE

De les diferents APIs, n'hi ha una que mereix una atenció especial:

JDBC

JDBC [16] és l'acrònim de Java Database Connectivity, una API que permet l'execució d'operacions sobre bases de dades des del llenguatge de programació Java independentment del sistema d'operació on s'executi o de la base de dades a la qual s'accedeix utilitzant el dialecte SQL del model de base de dades que s'utilitzi.

La API JDBC es presenta com una col·lecció d'interfícies Java i mètodes de gestió de controladors de connexió cap a cada model específic de base de dades (Figura 3.4). Aquest controlador forma un conjunt de classes que implementen les interfícies Java i que utilitzen els mètodes de registre per a declarar els tipus de localitzadors a la base de dades (URL) que poden controlar.

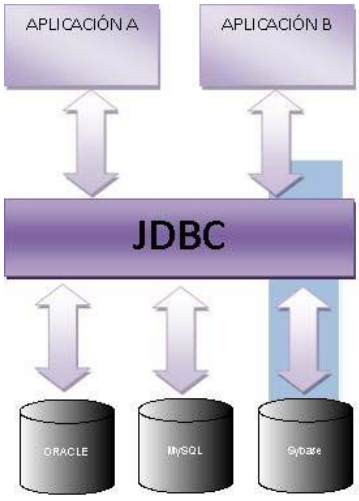


Figura 3.4. Arquitectura JDBC

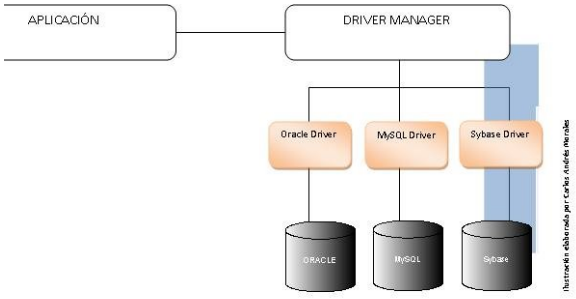


Figura 3.5. Connexió a la base de dades

Per a utilitzar una base de dades particular, l'usuari executa el seu programa juntament amb la llibreria de connexió apropiada al seu model i accedeix a ella establint una connexió, com es veu a la Figura 3.5. Per a això el proveeix dels paràmetres de connexió específics. A partir d'allà, pot realitzar qualsevol tipus de tasques amb la base de dades a les quals tingui permís: consultes, actualitzacions, creat modificat i esborrat de taules, execució de procediments emmagatzemats en la base de dades, etc.

JDBC compleix el seu objectiu mitjançant un conjunt d'interfícies de Java, cadascuna implementada de manera diferent per diferents distribuïdors. El conjunt de classes que la componen es denomina el controlador JDBC.

3.2.2. JSP (Java Server Pages)

JavaServer Pages (JSP) [17] és una tecnologia Java que permet generar contingut dinàmic per web en forma de documents HTML, XML o d'altres tipus. Va ser creat per Sun Microsystems.

L'ús més comú dels servlets és generar pàgines Web de forma dinàmica a partir dels paràmetres de la petició que enviï el navegador Web.

JSP pot considerar-se com una manera alternativa i simplificada de construir servlets. La paraula servlet ve derivada d'*applet* que es referia a petits programes escrits en Java que s'executen en el context d'un navegador web. La diferència és que el servlet s'executa en un servidor. El seu ús més comú és generar pàgines web de forma dinàmica a partir dels paràmetres de la petició que enviï el navegador web. La forma de fer-ho és incrustant aquest codi Java dins del codi HTML.

Les JSP's Ens permeten la utilització de codi mitjançant scripts. A més, es possible usar algunes accions JSP predefinides a partir d'etiquetes. Aquestes poden ser enriquides amb la utilització de Llibreries d'Etiquetes (TagLibs) externes o, inclús, personalitzades. Pot considerar-se una manera alternativa i simplificada de construir servlets.

El funcionament general d'aquesta tecnologia és que el Servidor d'Aplicacions interpreta el codi contingut en la pàgina JSP per construir el codi Java del servlet a generar. Aquest servlet serà el que generi el document (típicament HTML) que es presentarà en la pantalla del Navegador de l'usuari.

La principal avantatge de JSP en vers altres llenguatges és que Java és un llenguatge de propòsit general perfecte pel món web i apte per crear classes de la lògica de negoci. Una altra és que JSP hereta la portabilitat de Java i es possible executar les aplicacions en múltiples plataformes sense canvis.

3.2.3. Apache Tomcat

Tomcat funciona com un contenidor de servlets desenvolupat sota el projecte Jakarta en la Apache Software Fundation [18]. Implementa les especificacions dels servlets i de JavaServer Pages (JSP) de Sun Microsystems.

Tomcat és un servidor web, gràcies a Apache, amb suport de servlets i JSPs. Inclou el compilador Jasper que compila JSPs convertint-les en servlets. Pot funcionar com servidor web per sí mateix per entorns amb alt nivell de tràfic i alta disponibilitat. Com va ser escrit en Java, funciona en qualsevol sistema operatiu que disposi de la màquina virtual Java.

3.2.4. MySQL

MySQL [19] és un sistema de gestió de base de dades relacional, multithread i multiusuari que utilitza el llenguatge SQL (*Structured Query Language*). Pertanyia a l'empresa sueca *MySQL AB* però va ser comprada per *Sun Microsystems* aquest any.

MySQL ha esdevingut molt popular gràcies a la seva velocitat en executar consultes en l'elaboració d'aplicacions web en l'entorn del programari lliure. Es pot fer ús en aplicacions de tota mena de forma lliure i gratuïta sota les condicions de la llicència GPL (no comercial).

És un dels components de l'arquitectura LAMP i XAMP (Linux, Windows – Apache – MySQL - PHP) que és una plataforma per a la construcció de llocs web utilitzant programari lliure [20].

3.3. Eines de desenvolupament

3.3.1. Sun Java Wireless Toolkit for CLDC

J2ME Wireless Toolkit [21] és un conjunt d'eines per crear aplicacions Java que corre en dispositius obeint a la especificació Java Technology for the Wireless Industry (JTWI, JSR 185), que defineix la plataforma estàndard de la indústria para la següent generació de telèfons mòbils amb la tecnologia Java habilitada, i l'especificació Mobile Service Architecture (MSA, JSR 248) [22]. Consisteix en unes eines de construcció, utilitats i un emulador de dispositius. La última versió ens ofereix les següents APIs:

- Mobile Service Architecture (JSR 248)
- Java Technology for the Wireless Industry (JTWI) (JSR 185)
- Connected Limited Device Configuration (CLDC) 1.1 (JSR 139)
- Mobile Information Device Profile (MIDP) 2.0 (JSR 118)
- PDA Optional Packages for the J2ME Platform (JSR 75)

- Java APIs for Bluetooth (JSR 82)
- Mobile Media API (MMAPI) (JSR 135)
- J2ME Web Services Specification (JSR 172)
- Security and Trust Services API for J2ME (JSR 177)
- Location API for J2ME (JSR 179)
- SIP API for J2ME (JSR 180)
- Mobile 3D Graphics API for J2ME (JSR 184)
- Wireless Messaging API (WMA) 2.0 (JSR 205)
- Content Handler API (JSR 211)
- Scalable 2D Vector Graphics API for J2ME (JSR 226)
- Payment API (JSR 229)
- Advanced Multimedia Supplements (JSR 234)
- Mobile Internationalization API (JSR 238)
- Java Binding for the OpenGL(R) ES API (JSR 239)

A part de poder fer aplicacions per CLDC 1.1 i MIDP 2.0, també es poden desenvolupar aplicacions per CLDC 1.0 i MIDP 1.0 que són versions anteriors.

Sun Java Wireless Toolkit 2.5.2 inclou totes les característiques de desenvolupament avançat que ja existien en les versions anteriors com els MIDlets signats, la gestió de certificats i l'emulació over-the-air (OTA) integrada.

3.3.2. Netbeans

Netbeans IDE (Integrated Development Environment) [23] és un entorn de desenvolupament, una eina per programadors pensada per escriure, compilar, depurar i executar programes. Està escrit en Java però pot servir per qualsevol altre llenguatge de programació. És un projecte de codi obert creat per Sun Microsystems l'any 2000.

Dintre de les diferents tecnologies que ofereix cal destacar aquestes tres: **Mobility Pack** que ens ofereix totes les avantatges de les Wireless Toolkits, la possibilitat de fer aplicacions webs amb **JSP** i inclou la instal·lació del servidor **Apache Tomcat** per fer córrer aquestes aplicacions.

S'ha de tenir instal·lades les Wireless Toolkits perquè els emuladors de telèfons mòbils funcionin correctament.

4. Anàlisi

En aquest capítol analitzarem els requisits necessaris per la implementació del nostre sistema en una arquitectura client/servidor. Tant per un com per l'altre hem de tenir unes certes consideracions a l'hora d'implementar els seus respectius programes. Tot seguit elaborarem una llista amb els requisits per cada dispositiu.

4.1. Anàlisi del client

El client interactuarà amb dos dispositius: amb el seu telèfon mòbil i amb un navegador Web al que accedirà a través d'un ordinador personal o des del mateix mòbil.

4.1.1. Telèfon mòbil

Volem crear un programa capaç de generar contrasenyes d'un sol ús. Es consideraran els següents punts:

4.1.1.1. Llenguatge de programació

Cal un llenguatge de programació de baix pes específic per a dispositius mòbils.

Les memòries internes dels telèfons mòbils actuals no tenen una gran capacitat. Això passa tant a la ROM, aquella on guardem totes les dades que volem mantenir, com a la RAM, la que utilitzem per compilar i executar els programes.

Existeixen diferents llenguatges de programació especials per aquest tipus de dispositius, per exemple, el Java 2 Micro Edition o el Borland C++ Mobile Edition. El Borland C++ ME està dissenyat per fer programes per ser executats en els sistemes operatius Symbian o Widows Mobile. A nosaltres ens interessa que l'aplicació pugui ser executada en qualsevol tipus de sistema operatiu de qualsevol telèfon mòbil.

4.1.1.2. Base de dades

Necessitem generar una petita base de dades per guardar les dades per crear la contrasenya. Aquesta base de dades s'ha de poder guardar a la memòria del dispositiu mòbil i ha d'estar protegida de l'accés d'usuaris il·legítims ja que contindrà dades molt sensibles.

Una manera interessant de fer-ho és que només es pugui fer operacions amb les taules havent introduït prèviament el PIN (Personal Identification Number) del mòbil, sinó podem xifrar aquestes dades.

4.1.1.3. Llibreries criptogràfiques

Calen llibreries específiques que implementin algorismes criptogràfics per a la generació de claus i el xifratge de les dades.

Les dades introduïdes a la taula, com les contrasenyes, i la One-Time Password han d'anar xifrades per tal que siguin segures. Usarem diferents algorismes per cada operació. Necessitarem una llibreria que tingui almenys un algorisme hash (MD5 o SHA) i un altre de clau simètrica (DES o AES) o sinó de clau asimètrica (RSA). En cas de no trobar cap llibreria amb aquestes característiques buscaríem una altra que fos de matemàtica modular per poder implementar mètodes capaços de xifrar dades.

4.1.1.4. Funcionalitat

La nostra aplicació ha de poder funcionar en tots els telèfons mòbils actuals i en tots els sistemes operatius.

4.1.1.5. Entorn "user-friendly"

Ha de tenir un entorn amigable per l'usuari, fàcil i senzill d'utilitzar.

4.1.2. Navegador Web

A través del navegador Web, l'usuari podrà tenir accés a la pàgina Web on s'haurà d'autenticar.

4.2. Anàlisi del servidor

Ens centrarem en la implementació d'una pàgina web on qualsevol usuari es pugui registrar i un cop fet això, es pugui autenticar fent servir la OTP generada pel seu mòbil. Per això, tindrem en compte aquests fets:

4.2.1. Base de dades

Igual que amb el client, cal generar una base de dades per poder guardar les dades necessàries per una correcte registre i autenticació de l'usuari.

Hi ha diferents eines: Oracle, PosgreSQL o MySQL.

La nostra base de dades serà molt petita perquè tenim poques coses a emmagatzemar. Com en el cos del telèfon mòbil del client, l'accés a aquesta base de dades ha d'estar protegida.

4.2.2. Servidor d'aplicacions

Hem de tenir en compte els errors del servidor que ens proveeix la pàgina web (caiguda del servidor, retards en la connexió, ...) ja que hem de poder accedir-hi a través de qualsevol lloc gràcies a una connexió a Internet.

4.2.3. Entorn "user-friendly"

La pàgina web a implementar ha de ser fàcil d'utilitzar i que sigui visible correctament per qualsevol dels navegadors que existeixen avui en dia, per exemple: Mozilla Firefox, Internet Explorer , Opera o Safari, així com en qualsevol sistema operatiu.

5. Arquitectura i disseny

Per començar a dissenyar el sistema hem de tenir en compte que es basa en un model client/servidor. El client usarà una aplicació instal·lada en un dispositiu mòbil i un navegador Web que tindrà al darrera un servidor que proporcionarà una pàgina Web. Però per poder establir una sèrie de comunicacions entre els dos dispositius hem de definir un protocol. Per crear el nostre protocol ens fixarem en dues solucions.

5.1. Protocols de referència

5.1.1. Autenticació Web amb l'ús d' un telèfon mòbil amb Bluetooth i de tres servidors

Aquesta solució [7] combina el sistema d'autenticació web/mòbil, és a dir, permet l'autenticació d'un usuari en una pàgina web gràcies a una contrasenya d'un sol ús donada per un telèfon mòbil.

El mecanisme bàsic d'autenticació està integrat amb un procés repte/resposta i una OTP. El repte donat pel servidor d'autenticació ha d'autenticar un dispositiu mòbil. Aquest dispositiu pot comunicar-se amb altres parts involucrades, com pot ser un ordinador personal, via una connexió Bluetooth.

El sistema consta de tres servidors diferents: un servidor Web, un d'autenticació i un de suport. El dispositiu mòbil estableix una connexió segura via Bluetooth amb l'ordinador personal per poder fer totes les operacions d'autenticació. Després de la connexió, l'usuari, prèviament registrat al servei Web, utilitza un nom i una contrasenya que només saben ell i el servidor Web per fer una primera autenticació (Figura 5.1).

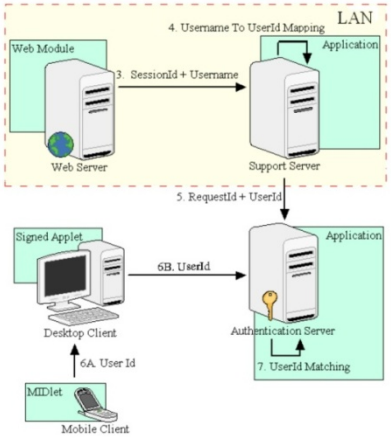


Figura 5.1. Configuració de la primera l'autenticació

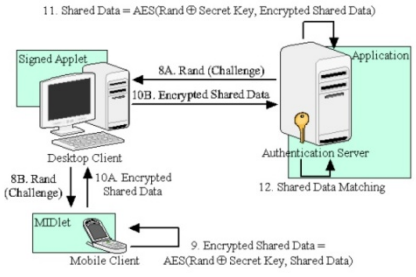


Figura 5.2. Combinació Repte/Resposta Web/Mòbil

Tot seguit, el servidor d'autenticació envia un repte que passa per l'ordinador personal i va directament al telèfon mòbil mitjançant la connexió abans establerta. Aquest últim respon amb unes dades xifrades que seran verificades pel mateix servidor (Figura 5.2).

Per últim, el servidor d'autenticació genera una contrasenya temporal que serà processada pel mòbil, obtenint la OTP, i enviada a través de l'ordinador personal al servidor Web. Aquest serà l'encarregat d'enviar-ho al servidor de suport. Alhora, el servidor d'autenticació haurà enviat la mateixa informació, OTP, que el mòbil també cap al servidor de suport, que serà l'encarregat de verificar que aquestes dues contrasenyes siguin les mateixes per acabar el procés d'autenticació de l'usuari (Figura 5.3).

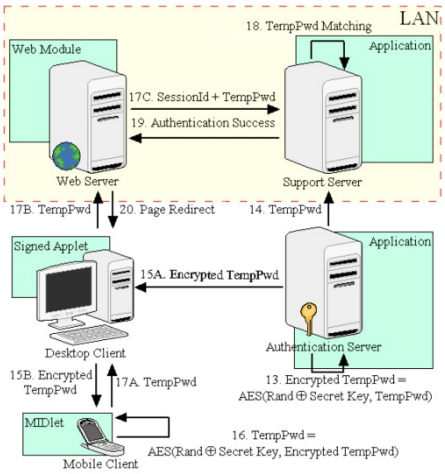


Figura 5.3. Combinació Web/Mòbil OTP

5.1.2. Autenticació Web amb l'ús d'un telèfon mòbil amb Bluetooth, un servei GSM i un servidor

La solució [8] proposada combina el principi simple i segur de la OTP amb la mobilitat d'un telèfon mòbil GSM. Està basada en un protocol simple de repte/resposta.

Disposarem d'un telèfon mòbil, un ordinador personal i un servidor que farà tant de servidor Web com de servidor d'autenticació.

Quan un usuari prèviament registrat al servei Web vol connectar-se, indica el seu nom d'usuari al formulari adient de la pàgina Web des d'un ordinador personal i, el servidor, li enviarà un repte al seu telèfon mòbil en forma de missatge de text. L'aplicació instal·lada al mòbil generarà una OTP mitjançant el repte rebut i l'enviarà cap el servidor gràcies a la connexió Bluetooth establerta entre el mòbil i l'ordinador personal. El servidor verificarà que la OTP enviada sigui correcte i autenticarà l'usuari (Figura 5.4).

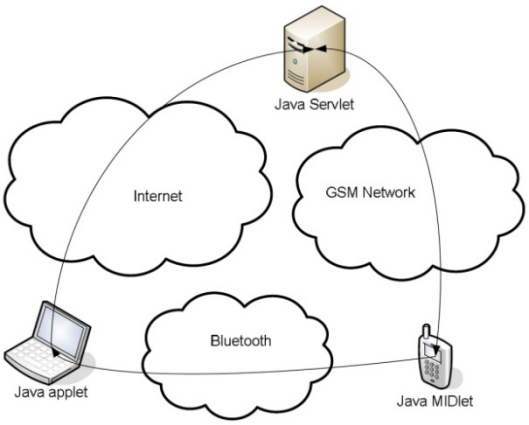


Figura 5.4. Arquitectura de components

5.1.3. Avaluació dels protocols: Anàlisi i característiques

Després de l'explicació d'aquests dos protocols anem a analitzar-los. Veurem les avantatges, els inconvenients i el que ens pot interessar a nosaltres a l'hora d'escriure el nostre protocol.

Tots dos tenen en comú una connexió Bluetooth entre un ordinador personal i el telèfon mòbil. D'aquesta manera s'aconsegueix una transmissió molt segura de les dades i s'evita haver de mostrar dades per pantalla. A part d'això, es facilita la tasca d'autenticació a l'usuari ja que no ha d'introduir cap dada a cap dispositiu i també evitem qualsevol errada humana. Però té l'inconvenient que, tant el telèfon mòbil com l'ordinador personal, han de disposar del hardware Bluetooth obligatòriament. Malgrat que avui en dia la majoria de telèfons mòbils disposen d'aquest mòdul i que afegir-ho a un ordinador personal és molt econòmic, no és un sistema que estigui totalment integrat. Això faria que l'aplicació implementada no funcionés per tots els mòbils que encara hi ha al mercat. A més a més, el fet d'utilitzar Bluetooth limita l'accés en ordinador públics: col·legis, universitats, biblioteques, etc, perquè hi ha moltes implementacions que tenen vulnerabilitats de seguretat on la transmissió de les dades i el contingut del telèfon mòbil poden ser visibles [22] i [23].

En quant els servidors, el primer protocol utilitza tres i el segon només un. Haver de mantenir tres servidors és costós i, una avaria en qualsevol d'un d'ells, faria difícil trobar en qual d'ells ha sigut i més lenta la seva reparació. Millor utilitzar un de sol.

El segon protocol usa una xarxa GSM per l'enviament de missatges de text per les dades sensibles que han de ser introduïdes al mòbil. Per poder fer aquests enviaments s'ha de contractar la xarxa GSM i s'ha de pagar. És una forma molt segura d'enviar dades ja que ens assegurem de que qui rep les dades és el propietari del telèfon però ens faria augmentar el cost del sistema perquè contractar la xarxa que volem no és de franc.

En resum, per realitzar el nostre protocol ens quedarem amb les idees següents: no utilitzarem Bluetooth, l'usuari introduirà les dades manualment; només utilitzarem un servidor que serà Web i d'autenticació; i usarem una xarxa GSM per l'enviament de dades sensibles o, sinó, algun mètode més econòmic com el correu postal o una trucada telefònica.

5.2. Protocol de comunicació per a dispositius mòbils basat en l'autenticació OTP

La nostra solució es basa en un protocol repte/resposta. Els elements a utilitzar seran els següents: un dispositiu mòbil (telèfon), un ordinador personal i un servidor Web que permetrà l'autenticació (Figura 5.5). La comunicació entre el servidor i l'ordinador es farà a través d'una connexió a Internet i, l'ordinador i el telèfon mòbil es comunicaran de forma manual. L'usuari haurà d'introduir les dades manualment.

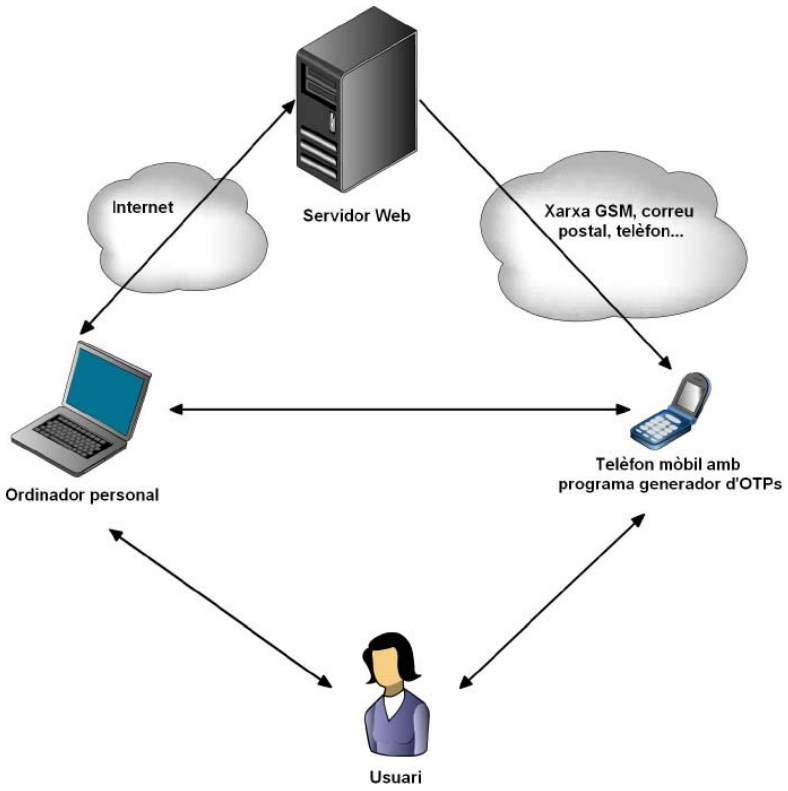


Figura 5.5. Arquitectura de components del protocol implementat

L'usuari ha de registrar-se, escollint un íd. d'usuari i una clau, dins d'un servei Web a través d'un formulari. El servidor es guardarà l'íd. (user_id), la clau (clau_web) que serveix per fer una primera autenticació, i una clau inicial (clau_inicial) creada per ell amb la qual generarem les contrasenyes d'un sol ús. A l'hora, el nom del servei Web (servei_id) i la mateixa clau inicial (clau_inicial) seran enviades a l'usuari en forma de missatge de text, correu postal, telèfon, etc., per a ser introduïdes a l'aplicació del telèfon mòbil.

Un cop registrat, l'usuari podrà autenticar-se en el servei Web. Haurà d'introduir el seu íd. d'usuari i la clau Web per la primera autenticació. Si és correcte, el servidor donarà a l'usuari un repte per a introduir al mòbil per tal de generar la contrasenya d'un sol ús. El servidor farà els mateixos passos per a la generació de la OTP i s'encarregarà de comprovar si la OTP introduïda per l'usuari és la correcte. Ambdós actualitzen la clau inicial de la mateixa manera per tal de mantenir la sincronització.

A continuació, veurem el procediment en detall:

REGISTRE

1) L'usuari es registra en un servei web. Introdueix en un formulari:
   - user_id
   - clau_web

   El servidor li dóna per un canal no web (SMS, correu postal, telèfon, en mà, ...):
   - servei_id
   - clau_inicial

   El servidor crea una entrada per aquest usuari en el seu registre:
   - user_id
   - clau_web
   - clau_inicial

2) L'usuari entra l'id del servei i la clau inicial en el seu mòbil.
   Es crea registre en el mòbil amb aquestes dades:
   - servei_id
   - clau_inicial

ACCÉS

1) L'usuari entra a la web i envia el seu user_id i la seva clau_web.

2) El servidor contesta amb un repte.

3) L'usuari introdueix el repte rebut en el mòbil.
   El mòbil opera:
   - clau_sessio = clau XOR repte
   - AES del servei_id amb la clau de sessió
   - Xifrat amb la clau_sessio

   El caràcters mostrats per pantalla seran la OTP.

   El mòbil actualitza la clau interna:
   - clau_inicial = clau_sessio

4) L'usuari envia OTP al servidor.

5) El servidor fa les operacions anàlogues al mòbil i comprova si el que ha rebut és correctes. Si és així, actualitza la clau:
   - clau_inicial = clau_sessio

A la Figura 5.6 podem veure el diagrama UML del protocol proposat.

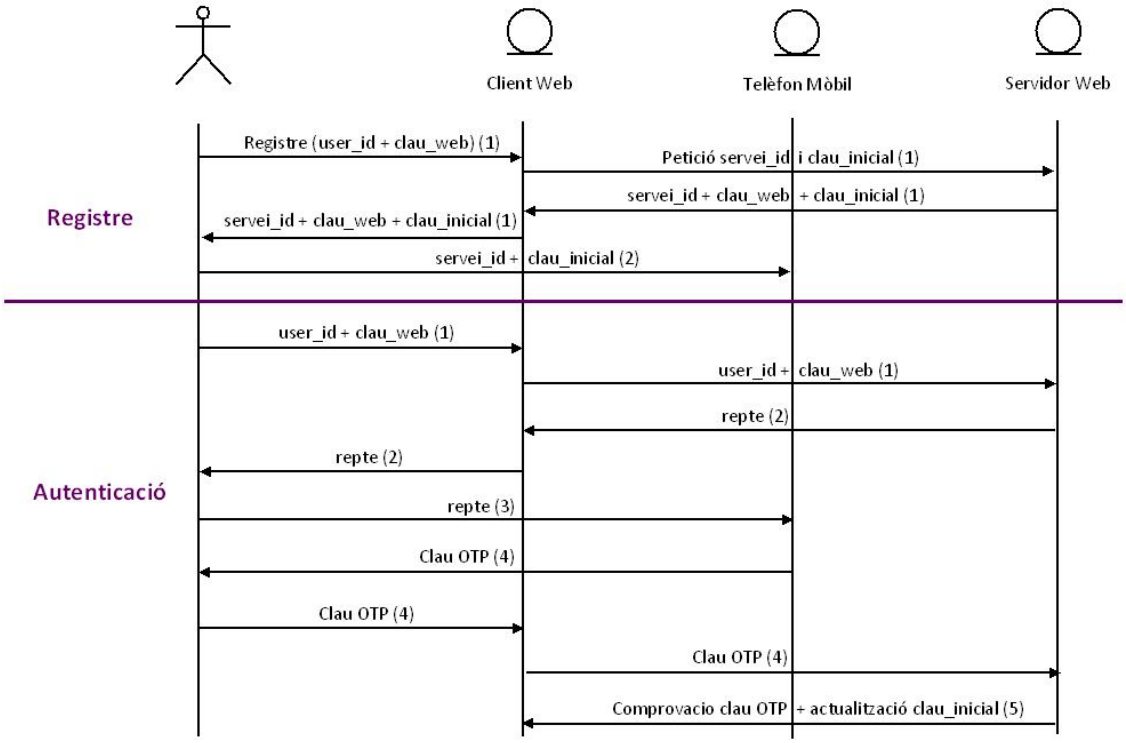


Figura 5.6. Diagrama UML del protocol

Especificacions del protocol:

- **Codificació i mida de les dades mostrades o introduïdes al dispositiu mòbil**

  Totes les claus donades per introduir en el telèfon o generades per ell mateix estaran codificades en Base 32 i tindran sis caràcters. Aquestes són la clau inicial donada pel servidor per introduir al telèfon, el repte i la OTP.

  Els caràcters codificats en codi ASCII (American Standard Code for Information Interchange), que consta de 256 caràcters diferents, es formen de vuit en vuit bits, és a dir, cadascun ocupa un byte. La codificació en Base 32 [24] es basa en la generació de caràcters de 5 en 5 bits. Això es tradueix en un canvi d'alfabet composat per totes les lletres de a la A a la Z, i els números del 2 al 7, ja que el 0 i el 1 poden portar confusió amb les lletres O i I.

  Amb aquesta codificació aconseguim reduir l'alfabet i evitar possibles caràcters difícils d'introduir al telèfon mòbil. Hem de pensar en la configuració del teclat del que disposen, on les lletres i els números generats per un alfabet en Base 32 són les més fàcils de teclejar.

  Donarem només sis caràcters per tal de fer la introducció i visualització de les cadenes de caràcters més còmode per l'usuari ja que les pantalles dels mòbils, malgrat cada cop es facin més grans, mesuren de mitjana unes dues polzades aproximadament.

- **Ús de la clau web**

  L'usuari quan es registra escull el seu íd. i una clau, anomenada clau web. Amb aquesta clau web es fa una primera autenticació amb la qual podem evitar que usuaris no autoritzats facin un atac de denegació de servei (Denial of service -DoS-). Si es donessin els reptes directament quan l'usuari introduís el seu íd., un usuari il·legítim podria estar demanant reptes per tots els usuaris i bloquejat l'accés al servei a qualsevol d'ells ja que hi haurà un límit de tres reptes per una correcta autenticació.

  Per altra banda, també ens permet donar de baixa l'usuari d'una manera ràpida i eficient si aquest perd el telèfon mòbil.

- **Emmagatzematge de les claus a les bases de dades**

  El client i el servidor guarden a les seves taules corresponents el hash de la clau inicial. El resultat d'aplicar una funció hash a una cadena de bytes és una cadena de bytes molt més gran on, per entrades diferents tindrem la mateixa mida en la sortida. L'aplicació de la funció és molt fàcil de calcular però trobar la seva inversa no, és a dir, a partir de la sortida és impossible trobar l'entrada i potser que a partir de dos entrades diferents tinguem dues sortides iguals. D'aquesta manera protegirem la nostra clau inicial generada pels poder fer els càlculs de la OTP.

  Utilitzarem l'algorisme SHA (Secure Hash Algorithm). SHA és un sistema de funcions hash criptogràfiques relacionades de l'Agencia de Seguretat Nacional dels Estats Units [27]. Existeixen dos variants del SHA: SHA-1 i SHA-2. SHA-1 produeix una sortida resum de 160 bits d'un missatge que pot tenir una mida màxima de 2^{64} bits i es basa en principis similars al usats pel disseny dels algorismes de resum MD4 i MD5. SHA-2 inclou quatre variants que es basen en el disseny inicial, una mica modificat, i en rangs de sortida incrementats: 224, 256, 384 i 512 bits. La família SHA és molt més difícil de trencar que, per exemple, el MD5. Han aconseguit trencar SHA-1 en 2^{63} operacions, que igualment segueix sent un número alt. Malgrat això, aquesta família d'algorismes és la més robusta que existeix avui en dia.

- **Generació de la contrasenya d'un sol ús**

  Un cop introduït el repte en l'aplicació per a la generació de la contrasenya d'un sol ús necessitem aplicar un algorisme de xifratge de clau simètrica o asimètrica. Pel nostre cas és més adient la primera opció. Dels diferents algorismes existents escollirem l'AES (Advanced Encryption Standard) que es basa en un esquema de xifratge per blocs i va ser adoptat pel govern d'Estats Units com un estàndard [28]. És un sistema molt segur ja que és molt difícil arribar a esbrinar el text xifrat sense disposar de la clau amb el que ha estat codificat. Es necessita una capacitat de càlcul molt gran per arribar a fer-ho. El sistema de blocs a utilitzar serà el mode Cipher FeedBack (CFB) [29] que realimenta el text xifrat, o part del mateix, per ser novament xifrat operant el resultat xor amb el text en clar per obtenir el següent bloc de text xifrat (Figura 5.7). Necessitarem un vector d'inicialització.

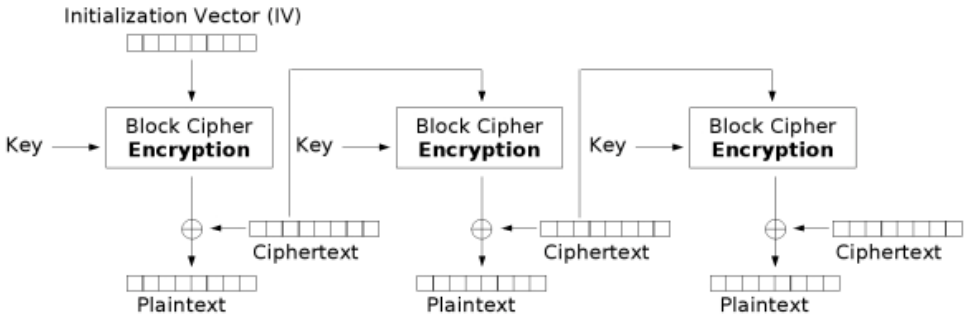


Figura 5.7. Mode de blocs Cipher FeedBack

La clau de sessió usada per xifrar serà un XOR del codi binari de la clau inicial i el resultant d'aplicar l'algorisme hash SHA al repte. L'aplicació de la funció hash, a part de donar-nos més seguretat en el procediment, ens donarà una sortida de bits d'igual mida que la clau inicial.
El XOR és un tipus d'encriptació molt senzill però que fa que aquesta clau sigui encara més robusta.

La clau de sessió final haurà d'estar truncada a 128 bits que és la mida mínima requerida per aplicar l'algorisme de l'AES.

El text a xifrar serà el nom d'íd. del servei emmagatzemat.

5.3. Disseny de les aplicacions

Abans d'explicar el disseny, tindrem en compte una limitació. Quan un usuari es registra cal enviar-li el nom del servei i la clau inicial via SMS, correu postal, telèfon..., com per poder enviar SMS s'ha de contractar un servei GSM, el correu postal triga dies i, per fer les proves necessàries no cal trucar per telèfon, aquestes dades seran mostrades per pantalla des de l'aplicació del servidor.

5.3.1. Aplicació del client (Telèfon mòbil)

En aquest apartat veurem com dissenyar l'aplicació pel dispositiu mòbil per tal que aquest arribi a fer les operacions descrites al protocol esmentat prèviament. Hi haurà una classe per la implementació del MIDlet i classes per la gestió de la base de dades i el xifratge, tot això sota una interfície gràfica. El llenguatge de programació usat serà el J2ME que ens garanteix que l'aplicació serà executable en qualsevol dispositiu mòbil que tingui instal·lat una plataforma Java.

5.3.1.1. Interfície gràfica

L'únic requisit de la nostra interfície gràfica és que sigui "user-friendly", és a dir, fàcil d'utilitzar per l'usuari, intuïtiu. No es necessita, ni es vol, un disseny carregat sinó el contrari, un disseny senzill. L'aparença de les pantalles tindrà un esquema com el de la Figura 5.8.

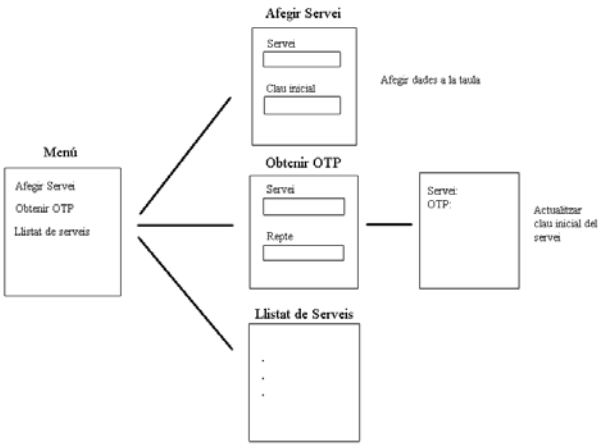


Figura 5.8. Aparença de les pantalles de l'aplicació

Per començar hi haurà un menú per poder escollir les diferents opcions: afegir un servei, obtenir una OTP o llistar els serveis disponibles. Tant en l'opció d'afegir servei com en la d'obtenir OTP disposarem de dos camps que pugui omplir l'usuari mitjançant el teclat del telèfon mòbil. Ens caldrà el nom del servei i la clau inicial o el repte, segons l'opció. Quan haguem introduït les dades per la OTP, mostrarem una pantalla amb aquesta contrasenya d'un sol ús. En el cas del llistat de serveis, només es mostrarà una llista amb el nom dels íd.'s dels serveis emmagatzemats. Així l'usuari no s'haurà de recordar de tots ells sinó que podrà consultar en qualsevol moment on ha estat registrat.

Tots els camps poden ser introduïts tant en majúscula com en minúscula. Així evitarem possibles fallides a l'hora d'introduir text per teclat i l'usuari no s'haurà de preocupar d'anar canviant el format dels caràcters fent més còmode aquesta part.

5.3.1.2. Classes J2ME

Per poder fer totes les funcions necessàries dins de la nostra aplicació, crearem tres classes: GestioTaules, GestioClaus i Conversions (Figura 5.9).

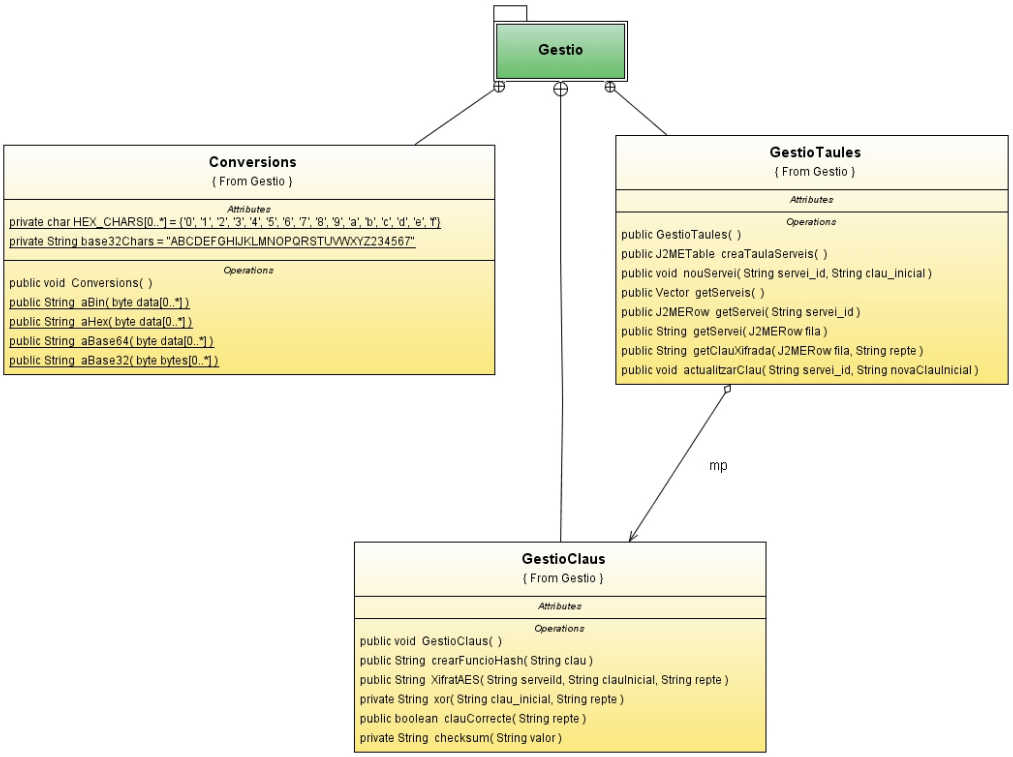


Figura 5.9. Diagrama de classes Java implementades al client

5.3.1.2.1. Classe GestioTaules

En aquesta classe implementarem tots els mètodes necessaris per a crear i gestionar la base de dades. Podrem generar la base de dades, afegir nous serveis amb les seves claus inicials corresponents, buscar segons el servei introduït per l'usuari, la clau inicial i generar la contrasenya d'un sol ús, buscar els diferents serveis per poder-los mostrar per pantalla i, actualitzar el camp de la clau inicial amb un nou valor un cop generada la OTP. Per dur a terme tots aquests mètodes, hem de buscar una llibreria adient. Escollirem entre dues opcions: J2MEMicroDB o OpenBaseMovil.

OpenBaseMovil [30], a part d'oferir-nos la possibilitat de generar una base de dades, també inclou classes per implementar el protocol de blueetooth, algorismes criptogràfics i interfícies gràfiques. Només volem una llibreria especial que implementi la primera opció. Ens decantarem J2MEMicroDB ja que es tracta d'una llibreria petita que té les classes i mètodes suficients per crear, consultar, modificar i esborrar una base de dades amb poques taules. El seu pes també és més reduït cosa que ens interessa ja que necessitem una aplicació de pocs bytes.

Tanmateix, les taules de la base de dades han de guardar-se en un lloc segur perquè ningú pugui d'accedir-hi ja que les dades que contenen són molt sensibles. Una manera interessant de fer-ho és que només es pugui fer operacions amb les taules havent introduït prèviament el PIN (Personal Identification Number) del mòbil. Sinó podem xifrar aquestes dades.

La taula de la base de dades no ha de ser accessible per ningú que no estigui autoritzat. L'usuari no ha de poder tenir accés a les dades si no es que abans vàlida la seva autenticitat. Una opció robusta és que, quan l'usuari vulgui fer alguna acció amb aquesta taula, se li demani el PIN del mòbil. Per poder-ho fer, la targeta SIM del mòbil necessita tenir un mòdul WIM (WAP Identity Module).

El WAP Identity Module (WIM) [31] s'utilitza per fer WTLS (Wireless Transport Layer Security) i funcions de seguretat a nivell d'aplicació i sobretot, per guardar i processar informació necessària per d'identificació i l'autenticació de l'usuari. La funcionalitat està basada en el requeriment de dades sensibles, especialment claus, que poden estar guardades en la WIM, i totes les operacions on aquestes claus estan implicades. Un exemple d'implementació de la WIM són les targetes intel·ligents (smart cards). En un telèfon mòbil és la targeta SIM (Subscriber Identity Module).

La forma en què un telèfon i una targeta intel·ligent interactuen s'especifica com un protocol de repte/resposta utilitzant APDU (Application Protocol Data Units) específics per aquesta aplicació. L'especificació està basada en les series ISO7816 dels estàndards de les targetes intel·ligents i està relacionat amb les especificacions GSM (Global System Communications).

El paquet opcional SATSA-APDU ens permet establir una connexió entre l'aplicació i la targeta SIM del mòbil. Un altre paquet opcional, el SATSA-PKI, té el mètode CMSMessageSignatureService per a signar missatges amb una clau privada emmagatzemada en un mòdul WIM. Aquests missatges poden ser signats amb l'objectiu d'aconseguir autenticació o no repudi. L'autorització per a usar una clau concreta del mòdul WIM vindrà donada per la política de seguretat d'aquest mòdul. Podria requerir que s'insereixi el PIN del mòbil.

No obstant, hem de tenir en compte unes altres consideracions. El mòdul WIM ha d'estar a la targeta SIM del mòbil però, després de preguntar a les diferents companyies telefòniques disponibles, ens van informar que les targetes SIM actuals no disposen d'aquest mòdul. Tanmateix, els telèfons mòbils més venuts a dia d'avui no inclouen una plataforma de Java amb l'API de SATSA.

Segons les especificacions de dispositius de la marca Nokia® [32], SATSA només es troba disponible a partir dels mòbils Nokia® de tercera generació de la sèrie 60 (S60 3rd edition). A part de Nokia®, telèfons mòbils de gamma alta d'altres companyies també l'utilitzen com, per exemple, Samsung® o LG® [33]. Tot i això, són molts pocs i els més venuts són els de la sèrie 40, anterior a la 60, que no permeten l'ús de SATSA.

S'ha realitzat una altra comparativa amb els mòbils Sony-Ericsson® [34] i, els telèfons més venuts, i dels que n'hi ha més, utilitzen la plataforma Java 7 que tampoc inclou l'API de SATSA. L'aplicació ha de poder funcionar en tots els telèfons mòbils actuals.

Com la majoria de mòbils no incorporen l'última plataforma de Java, la número 8. No podrem fer la protecció de la taula de la base de dades desitjada.

La classe GestioTaules tindrà els següents mètodes:

- **public J2METable creaTaulaServeis()**
  Crea la base de dades amb la taula "taulaServeis" que consta de dos columnes: una per l'íd. del servei anomenada "servei_id", i una altre per la clau inicial anomenada "clau_inicial". Totes dues contindran cadenes de caràcters. La columna "servei_id" serà la clau de la taula.

- **public void nouServei(String servei_id, String clau_inicial)**
  Afegeix nous serveis amb les seves claus inicials. Comprovem que existeix la taula "taulaServeis" i si no existeix la creem. Un cop feta la comprovació anterior, carreguem la taula i afegirem una fila nova amb el servei i la clau inicial indicats. La clau inicial introduïda serà el resultat d'aplicar una funció hash.
  Vigilarem que el servei no existeixi ja a la taula per evitar tenir dades duplicades en la mateixa. En el cas d'intentar introduir un servei ja afegit, avisarem a l'usuari.
  El nom de l'íd. del servei l'introduirem en majúscules per facilitar la feina a l'hora de fer cerques i evitar possibles errades de part de l'usuari.

- **public Vector getServeis()**
  Busca a la taula totes les files amb serveis i les retorna dins d'un vector.

- **public J2MERow getServei(String servei_id)**
  Retorna la fila corresponent al nom de servei indicat. Vigilarem de buscar l'íd. del servei en lletres majúscules ja que prèviament ha sigut introduït així.

- **public String getServei(J2MERow fila)**
  Retorna el contingut emmagatzemat en la fila indicada.

- **public String getClauXifrada(J2MERow fila, String repte)**
  Donada una fila, agafarem el valor dels seus camps "servei_id" i "clau_inicial" per, amb el repte, aplicar el xifrat AES i obtenir la contrasenya d'un sol ús, la qual retornarem.

- **public void actualitzarClau(String servei_id, String novaClauInicial)**
  Actualitza la clau inicial de la fila indicada per l'íd. del servei.

5.3.1.2.2. Classe GestioClaus

En aquesta classe implementarem els mètodes relatius al xifratge de dades. Entre aquests diferents mètodes hi haurà un per aplicar la funció hash i un altre pel xifrat i generació de la OTP. Per realitzar el mètode que implementa la OTP, ens caldrà fer un altre que faci el xifratge XOR.

També crearem un mètode per comprovar que l'usuari introdueix al telèfon mòbil les dades més importants correctament, com són la clau inicial i el repte. Això ho farem fent un checksum que consistirà en calcular la funció hash dels quatre primers caràcters i comprovant que els dos últims són iguals al resultat d'aplicar la funció.

Per fer totes aquestes funcions necessitem una llibreria que implementi mètodes criptogràfics. El paquet opcional SATSA de J2ME es podria fer servir però l'aplicació no funcionaria en la majoria de mòbils, així que ens decantarem per fer ús de la llibreria criptogràfica *BouncyCastle*, en concret el conjunt de classes *LightCrypto* que tenen una versió especial per a dispositius mòbils.

Mètodes existents a la classe:

- **public String crearFuncioHash(String clau)**
  Mètode que aplica una funció hash a una cadena de caràcters i ens retorna el resultat. La mida final d'aquest resultat serà de 16 bits a fi de reduir la cadena de caràcters retornada ja que estarà codificada en Base 32.
  L'algorisme hash que aplicarem serà el de SHA-1.

- **private final String xor(String clau_inicial, String repte)**
  Donades dos cadenes de caràcters en format binari, farem la funció xor entre les dues i retornarem el resultat en el mateix format que l'entrada.
  La funció xor la calcularem així:

Entrada		Sortida
A	**B**	**A XOR B**
0	0	0
0	1	1
1	0	1
1	1	0

- **public String xifratAES(String serveiId, String clauInicial, String repte)**
  Mètode que s'encarrega de generar la contrasenya d'un sol ús. Després de fer el xor amb la clau inicial i el repte, el resultat serà la clau. També crearem un vector de bytes de dades amb el repte per poder fer el xifratge de blocs CFB. El text pla a xifrar serà el nom del servei.
  Tot això ens retornarà una cadena de caràcters codificada en Base 32. Serà la OTP.

- **private String checksum (String valor)**
  Donada una cadena de caràcters, li aplica una funció MD5. La cadena d'entrada ha de ser de quatre caràcters i donarem una altra de sortida d'un. Codificarem la sortida en Base 32 i obtindrem el resultat final que serà una cadena de caràcters de dos elements.

- **public boolean clauCorrecte(String repte)**
  Aplicarem el mètode checksum per comprovar si el repte rebut és correcte. Donats els quatre primers caràcters del repte, el resultat d'aplicar checksum ha de ser igual als dos últims caràcters del mateix. Si la comparació és correcte retornarem true, sinó false.

5.3.1.2.3. Classe Conversions

La classe conversions contindrà tots els mètodes per fer els canvis de base. Com vam comentar anteriorment, totes les claus i reptes estaran codificats en Base 32. També ens caldrà tenir un mètode que passi els bytes resultants d'aplicar la funció hash a binari per poder fer el xifratge XOR. Aprofitarem per crear altres mètodes que implementin altres canvis de base per poder tenir més flexibilitat a l'hora de realitzar l'aplicació. Seran els canvis a hexadecimal i base 64.

Mètodes continguts a la classe:

- **public static final String aBin(final byte[] data)**
  Donat un array de bytes retornem el seu valor codificat en binari.

- **public static final String aHex(final byte[] data)**
  Donat un array de bytes, retornem el seu valor codificat en hexadecimal.

- **public static final String aBase64(final byte[] data)**
  Donat un array de bytes, retornem el seu valor codificat en Base 64.

- **public static String aBase32(final byte[] bytes)**
  Donat un array de bytes, retornem el seu valor codificat en Base 32.

5.3.1.3. Classe MIDlet

El diagrama de classes de la classe *GeneradorOTP*, que és la que implementa el nostre MIDlet serà com el de la Figura 5.10.

Figura 5.10. Diagrama de classes del MIDlet

5.3.1.3.1. Classe GeneradorOTP

La classe GeneradorOTP tindrà tots els mètodes necessaris per crear l'aplicació. Aquesta classe hereta de la classe MIDlet i implementa la interfície CommandListener. En concret, implementarem el mètode CommandAction (Command c, Displayable d) que indica que ha ocorregut en una pantalla (Displayable) un event d'una comanda (command). Les comandes indiquen les accions que tenen les tecles d'un dispositiu mòbil en relació a les pantalles, per exemple les d'acceptar, sortir o tornar. Els "Displayables" són els elements que formen les pantalles: formularis, llistes, alertes, caixes de text,... Donat un "Displayable", aquest esperarà a un event de comanda per mostrar la pantalla següent. Caldrà implementar els mètodes per crear cada pantalla i el seu contingut així com uns altres per crear els events de les comandes.

Tindrem "Displayables" diferents per cada pantalla, igual que els de la Figura 5.8. Seran els següents:

- **Menú:**
  Haurà un llistat d'opcions sel·leccionables i una comanda de sortida de l'aplicació.

- **Afegir Servei:**
  Formulari que contindrà dos caixes de text, una per introduir l'íd del servei i un altre per la clau inicial. Tindrà dues comandes, una per tornar al menú i una altra per acceptar les dades introduïdes. Caldrà controlar que aquestes dades siguin correctes: els dos camps han de tenir caràcters escrits i, la clau inicial ha de tenir exactament sis caràcters i que el checksum d'aquests sigui correcte.

- **Obtenir OTP:**
  Aquest formulari serà igual que el d'afegir servei però la segona caixa de text serà per introduir el repte. Controlarà que les dades siguin correctes també de la mateixa manera que afegir servei. Tindrà dues comandes, una per tornar al menú i una altra per acceptar les dades introduïdes.

- **OTP:**
  Formulari que mostrarà dos elements de text amb el nom de l'íd. del servei i la contrasenya d'un sol ús. Hi haurà una comanda per poder tornar al menú.

- **Llistat de serveis disponibles:**
  Llistat amb els diferents serveis. Hi haurà una comanda per poder tornar al menú.

5.3.2. Aplicació del servidor (Pàgina Web)

Aquí veurem com dissenyar la pàgina Web d'un servei per fer el registre i autenticació d'un usuari. Tindrem les mateixes classes però amb alguns mètodes diferents i, a més a més, els arxius HTML i JSP per donar forma a la nostra pàgina.

5.3.2.1. Interfície gràfica

Igual que l'aplicació del client, la pàgina web que crearem ha de ser "user-friendly" i senzilla. L'esquema seguit serà com el de la Figura 5.11.

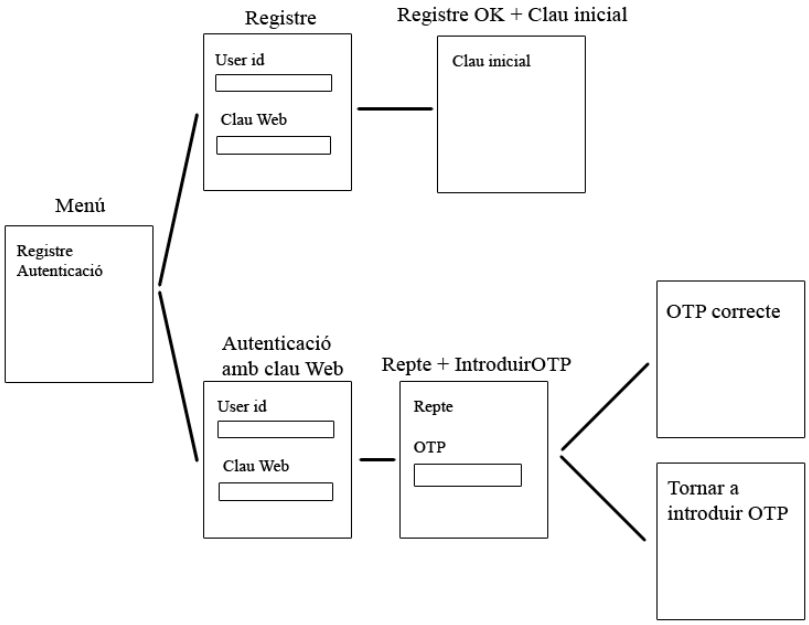


Figura 5.11. Aparença de les pantalles del servidor

A través del menú accedirem a les opcions de registre i autenticació. En el registre, l'usuari escollirà una íd. i una contrasenya, que no variarà, i els introduirà en un formulari. Si el registre és satisfactori, l'aplicació mostrarà la clau inicial a introduir al mòbil. Un cop fet això, l'usuari ja es podrà autenticar. Haurà d'introduir en un altre formulari l'íd. i la contrasenya escollits en el registre i, si la primera autenticació és correcte, se li mostrarà per pantalla el repte a introduir al mòbil per generar la OTP. Un cop generada, l'introduirà en el formulari pertinent de la pàgina Web i es faran les tasques de verificació. L'usuari té tres intents per introduir una OTP abans que se li doni un altre repte.

Igual que amb el client, tots els camps poden ser introduïts tant en majúscula com en minúscula per facilitar l'usuari la introducció de dades als formularis.

5.3.2.2. Pàgina Web

La tecnologia utilitzada per fer la pàgina Web serà Java Server Pages (JSP). Necessitem poder incrustar codi Java en el HTML per tal de poder fer totes les operacions requerides. Dividirem els JSP's en dos grups, registre i autenticació, que accedirem a ells a través d'un menú.

5.3.2.2.1. Menú

El menú seran dos links que redireccionaran l'usuari cap al registre o l'autenticació. Al finalitzar qualsevol de les dues operacions sempre tornarem al menú.

5.3.2.2.2. Registre

El registre tindrà dos pàgines JSP:

- **registre.jsp**
  Aquí tindrem la primera part del registre. Constarà d'un formulari on l'usuari introduirà l'íd. i contrasenya escollits. L'íd no podrà existir a la base de dades, si no és així l'haurem d'avisar perquè escolli un nom nou, i la contrasenya serà qualsevol que l'usuari desitgi.

- **registreOK.jsp**
  Un cop introduïdes les dades anteriors, si tot és correcte, crearem l'entrada a la base de dades per aquest usuari i haurem de mostrar per pantalla l'íd. del servei i la clau inicial, que ha sigut creada aleatòriament, per introduir a l'aplicació del client.

5.3.2.2.3. Autenticació

L'autenticació tindrà quatre pàgines JSP:

- **autenticacio.jsp**
  Aquesta serà la primera part de l'autenticació. Mostrarem un formulari per pantalla perquè l'usuari introdueixi l'íd. i la contrasenya escollits en el registre.

- **comprovacioUser.jsp**
  A partir de les dades anteriors, verificarem que existeixen a la base de dades i que són correctes o no. Si no són correctes, avisarem l'usuari i haurà de tornar a introduir les dades. Si sí que ho són, mostrarem un link per obtenir el repte.

- **repte.jsp**
  Generarem un repte aleatori i el mostrarem per pantalla. A sota, tindrem el formulari per introduir la OTP que haurà generat l'usuari en l'aplicació del telèfon mòbil.

- **autenticacioOK.jsp**

  Si la OTP introduïda és correcte, finalitzarem l'autenticació, sinó l'usuari tindrà tres intents més per introduir-la abans de que generem un altre repte. Això ho farem per si hi ha hagut un fallo al posar les dades. Els reptes els anirem emmagatzemant en una taula per tenir un control sobre ells. Donarem fins a tres reptes abans de bloquejar el servei per aquell usuari.

5.3.2.3. Classes J2SE

Per poder fer totes les funcions necessàries a la pàgina Web, crearem tres classes, que tindran el mateix nom que les del client: GestioTaules, GestioClaus i Conversions (Figura 5.12). Com les algunes passes a fer al servidor són iguals a les del client, podrem aprofitar molts mètodes ja implementats. El fet d'usar Java per les dues aplicacions ens dóna una versatilitat i reaprofitament de codi molt gran.

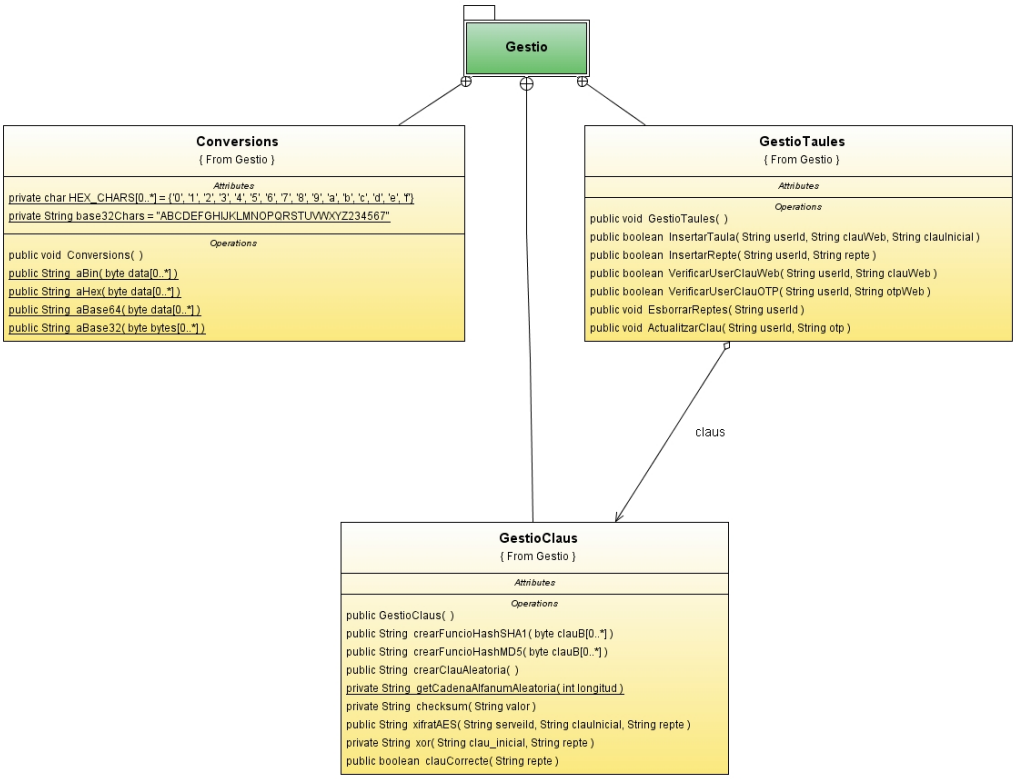


Figura 5.12. Diagrama de classes Java implementades al client

5.3.2.3.1. Classe GestioTaules

En aquesta classe implementarem tots els mètodes necessaris per a crear i gestionar la base de dades. La tecnologia que usarem serà MySQL. Necessitem generar una base de dades petita que constarà de dues taules, una per emmagatzemar les dades de l'usuari (íd., clau Web i clau inicial) i una altra per guardar els reptes que li anem donant.

La primera taula, anomenada *useridtaula*, tindrà tres columnes: User_id, clau_web i clau_inicial. La segona, anomenada *taulareptes*, tindrà quatre columnes: User_id, repte1, repte2 i repte3. En aquesta taula gestionarem els reptes donats a l'usuari.

Els mètodes implementats a la classe seran:

- **public boolean InsertarTaula(String userId, String clauWeb, String clauInicial)**
  Mètode on afegirem a la taula les dades del registre de l'usuari. Farem una connexió a la base de dades. Abans d'afegir res, comprovarem que l'usuari a introduir no existeix. Si existeix retornarem true, és a dir, que hi ha hagut un error, si no existeix crearem una fila nova a les dues taules. La taula *useridtaula* tindrà tots els camps complets, la taula *taulareptes* tindrà el nom de l'usuari i els reptes estaran inicialitzats amb un zero. En aquest cas, retornarem false.
  La clau web guardada a la taula serà el resultat d'aplicar una funció hash SHA a la clau web introduïda per l'usuari. La clau inicial final serà el resultat d'aplicar la mateixa funció hash que la clau web, a la clau inicial generada aleatòriament, igual que en el client.

- **public boolean InsertarRepte(String userId, String repte)**
  En aquest mètode afegirem els reptes a la taula *taulareptes*. Cada cop que li donem a l'usuari un repte nou, aquest serà guardat en la posició del vector de reptes creat: repte1, repte2 o repte3, que són diferents columnes de la taula.
  Un cop establerta la connexió amb la base de dades, si a la columna del repte hi ha un zero, podrem guardar el repte corresponent, sinó agafarem la següent posició que contingui un zero. Si l'operació s'ha realitzar correctament, és a dir, que tenim posicions lliures per guardar un repte, retornarem false, que no ha hagut error, sinó retornarem true.

- **public boolean VerificarUserClauWeb(String userId, String clauWeb)**
  Aquí comprovarem que la clau web introduïda per un usuari és correcte i igual a l'afegida a la taula de la base de dades. Retornarem true si ha hagut algun error, false si no hi ha hagut cap.

- **public boolean VerificarUserClauOTP(String userId, String otpWeb)**
  Mètode per comprovar que la OTP introduïda per l'usuari, generada per l'aplicació del telèfon mòbil, és igual a la generada pel servidor. Les dues contrasenyes d'un sol ús estaran calculades per les mateixes funcions criptogràfiques però hem de tenir en compte els reptes generats.
  Cada cop que l'aplicació del mòbil calcula una OTP, automàticament actualitzem la clau inicial. Això és possible perquè l'operació s'ha realitzat amb èxit. Però, en el servidor, fins que no comprovem que les dues OTP que tenim són iguals, no podem dir que la operació hagi estat correcte. Llavors, el que farem per no tenir confusions és, a partir del segon repte donat, calcular les claus de sessió usades per generar la OTP tenint en compte els reptes anteriors guardats a la taula de reptes. La clau de sessió no és res més que la OTP amb la que actualitzem la clau inicial. Així simularem la actualització de la clau inicial però sense sobreescriure la clau inicial original.
  Retornarem false si no ha hagut cap error, true en cas contrari.

- **public void EsborrarReptes(String userId)**
  Mètode encarregat de tornar a inicialitzar a zeros tots els reptes. Quan l'usuari ha estat autenticat correctament amb la OTP, hem d'esborrar tots els reptes existents a la taula.

- **public void ActualitzarClau(String userId, String otp)**
  Mètode per actualitzar la clau inicial un cop l'autenticació ha finalitzat.

5.3.2.3.2. Classe GestioClaus

En aquesta classe implementarem els mètodes relatius al xifratge de dades. Entre aquests diferents mètodes hi haurà dos per aplicar una funció hash i un altre pel xifrat i càlcul de la OTP. Per realitzar el mètode que implementa la OTP, ens caldrà fer un altre que faci el xifratge XOR. Són les mateixes operacions que en el client i tindrem les mateixes consideracions.

Utilitzarem la mateixa llibreria BouncyCastle: *LightCrypto* però per la versió de Java SE.

Mètodes implementats a la classe:

- **public String crearFuncioHashSHA1(byte[] clauB)**
  Mateixa implementació que el mètode del client *crearFunciόHash*.

- **public String crearFuncioHashMD5(byte[] clauB)**
  Mateixa implementació que el mètode *crearFuncioHashSHA1* però el càlcul serà amb la funció hash MD5.

- **private static String getCadenaAlfanumAleatoria(int longitud)**
  Mètode encarregat de crear i retornar una cadena de caràcters, entre la A i la Z i el 2 i 9, aleatòria. Agafem aquest rang de lletres i números ja que és el mateix rang que el de la codificació en Base 32.

- **private String checksum (String valor)**
  Retornarem dos caràcters codificats en Base 32 que seran el resultat d'aplicar l'algorisme hash MD5 a una cadena de quatre caràcters.

- **public String crearClauAleatoria()**
  Mètode que retornarà una cadena de sis caràcters aleatoris en Base 32. L'utilitzarem per crear la clau inicial i els reptes.

- **public String xifratAES(String serveiId, String clauInicial, String repte)**
  Mateixa implementació que el mètode del client *xifratAES*.

- **private final String xor(String clau_inicial, String repte)**
  Mateixa implementació que el mètode del client *xor*.

5.3.2.3.3. Classe Conversions

La classe Conversions del servidor consta dels mateixos mètodes que la classe Conversions del client.

6. Implementació

En aquest capítol veurem les eines, modificacions i configuracions utilitzats en la implementació de les aplicacions del client i del servidor. Al final mostrarem una sèrie de figures que explicaran visualment totes les passes que s'han de donar en totes dues aplicacions per realitzar el protocol d'autenticació.

6.1. Aplicació del client

Per implementar l'aplicació que anirà instal·lada al telèfon mòbil hem fet servir el *Sun Java Wireless Toolkit for CLDC* i *Netbeans*. Aquests dos programes ens proporcionen les eines necessàries per fer un bon desenvolupament i simulació d'una aplicació en J2ME.

Com ja hem comentat anteriorment, l'aplicació creada a d'ocupar pocs bytes. Hi ha certs telèfons mòbils que no admeten executables més grans de un megabyte [35]. La llibreria que necessitem i que ocupa més espai és la *BouncyCastle*. En comptes d'afegir al projecte la llibreria sencera, eliminarem les classes que no fem servir vigilant de no treure cap que utilitzin les que ens importen. D'aquesta manera aconseguim reduir l'executable *.jar* de 1.2 MB a 742 KB. Mida més que suficient perquè no tinguem problemes en cap dispositiu mòbil.

A part d'haver de reduir la llibreria *BouncyCastle*, a l'hora de compilar s'ha de fer amb ofuscació. L'ofuscació serveix per eliminar classes no desitjades d'APIs que són massa grans, com la *BouncyCastle* de baix pes, *LightCrypto*. A més, gràcies a l'ofuscació podem evitar els intents d'enginyeria inversa i evitar que qualsevol persona pugui veure el codi font de l'aplicació sense consentiment previ. Netbeans ens proveeix d'una eina d'ofuscació, anomenada *ProGuard*, i ens deixa configurar-la sobre diferents nivells. En el nivell "Off" la desactivem; el nivell 1 és per editar-ho com nosaltres vulguem; el nivell 2 per ofuscar camps privats i mètodes; els nivells 3 i 4 per el mateix que el segon però afegint-hi els camps per defecte i les classes; dels nivells 5 al 8 obtenim una ofuscació que serveix principalment per llibreries; i, l'últim nivell serveix per ofuscar tot excepte les classes públiques del MIDlet i està dissenyat principalment per aplicacions. Nosaltres utilitzarem l'ofuscació en el nivell 8 que és especial per llibreries i ens ajuda a la reducció de mida de l'aplicació final. El codi de l'ofuscació seria el següent:

```
Level 8 - Everything except public fields and methods of public classes.
Mainly for libraries or applications.

Obfuscator arguments:
  -dontusemixedcaseclassnames
  -defaultpackage ''
  -overloadaggressively
  -keep public class ** {
    public *;
  }
```

6.2. Aplicació del servidor

Per implementar la pàgina Web hem fet servir *Netbeans* i per poder-la compilar i visualitzar el servidor *Apache Tomcat*.

Netbeans ens dóna l'opció d'instal·lar una versió bastant nova de *Tomcat* i ens facilita la compilació de les JSPs fent ell totes les operacions de configuració i visualització del servidor. A part, ens deixa plena llibertat per configurar els arxius tant de la pàgina Web com del servidor. També podem instal·lar la versió que vulguem de *Tomcat* i fer-la córrer igual que la versió que instal·la *Netbeans*.

Un cop implementada la pàgina Web i amb correcte funcionament, configurarem el servidor *Tomcat* per poder fer connexions segures, és a dir, per usar el protocol HTTPS. HTTPS [36] és un protocol de xarxa basat en el protocol HTTP destinat a la transferència segura de dades d'hipertext, és a dir, és una versió segura de HTTP. El sistema HTTPS utilitza un xifrat basat en les Secure Socket Layers (SSL) [37] per crear un canal xifrat més apropiat pel tràfic d'informació sensible que el protocol HTTP. Però per fer tot això ens caldrà crear un certificat. El mateix programari de desenvolupament de Java (SDK) ens ofereix una eina molt útil i senzilla per crear certificats, és la keytool [38]. Les passes a seguir per crear-lo són les de la Figura 6.1. Obtindrem un arxiu amb el nom *.keystore* i el col·locarem en l'arrel dels arxius del servidor. El tros a afegir en l'arxiu *server.xml* del servidor serà:

```
<Connector SSLEnabled="true" acceptCount="100" clientAuth="false"
           disableUploadTimeout="true" enableLookups="true"
           keystoreFile=".keystore" keystorePass="password"
           maxSpareThreads="75" maxThreads="200" minSpareThreads="5"
           port="8443" scheme="https" secure="true" sslProtocol="TLS"/>
```

Si volem fer una connexió segura a la pàgina Web, l'haurem de cridar posant HTTPS i serà pel port 8443. Un connexió normal seria amb HTTP i el port 8080.

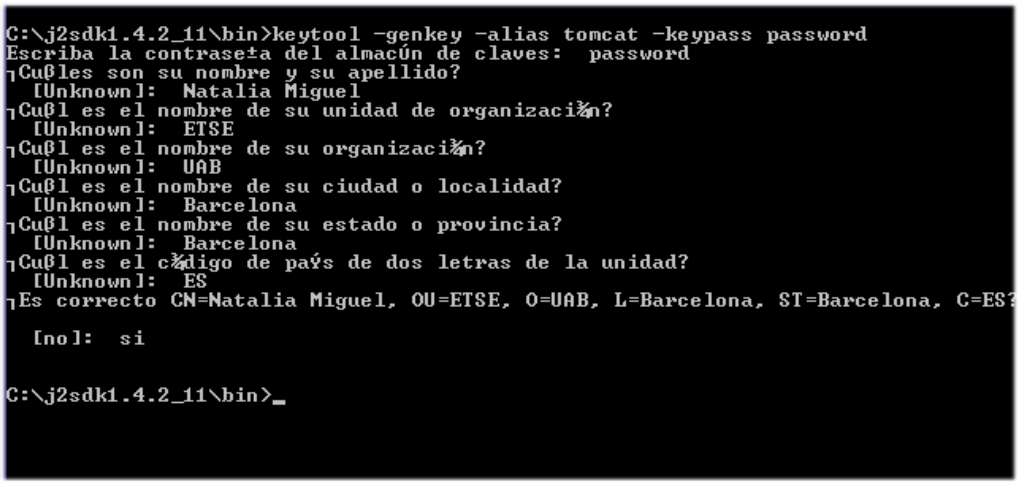


Figura 6.1. Passes per crear un certificat amb keytool de Java

6.3. Visualització de les aplicacions

Per visualitzar les passes de registre i autenticació mostrarem les figures de forma endreçada: de dalt a baix i d'esquerra a dreta.

A través de l'ordinador personal que té connexió a Internet, obrim el navegador web i anem a l'adreça del servei pel qual volem tenir accés. Com és una connexió segura i té un certificat web, com el de la figura 6.2, que no està emmagatzemat en el navegador, l'haurem d'acceptar comprovant abans les dades del mateix, verificant que tot és correcte i que amb qui fem la connexió és realment qui nosaltres volem.

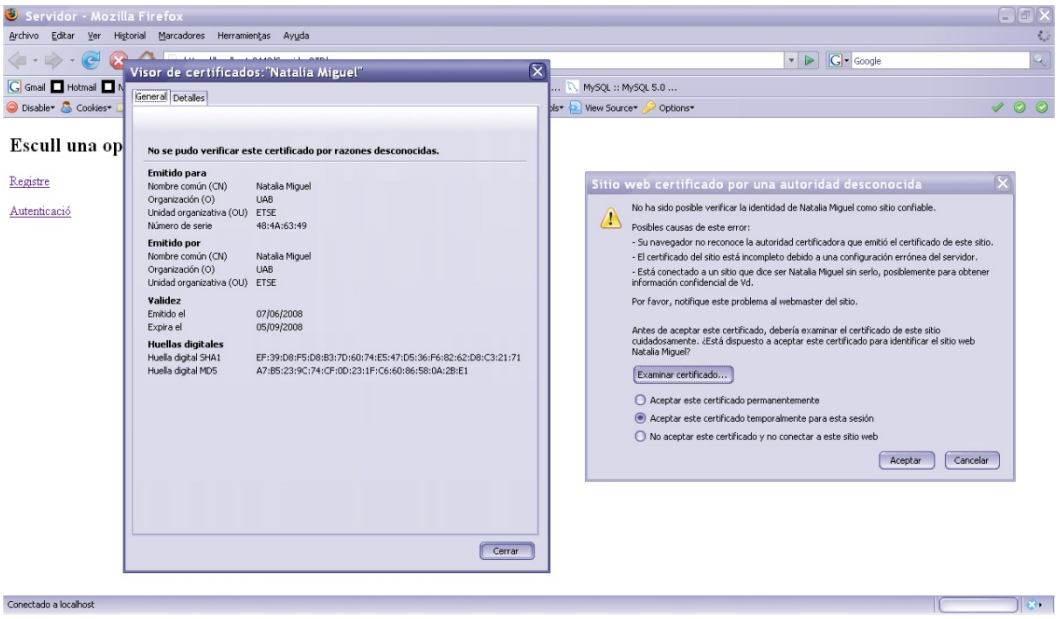


Figura 6.2. Connexió HTTPS

Un cop acceptat el certificat, escollirem l'opció Registre de les dues que apareixen al menú inicial (Figura 6.3).

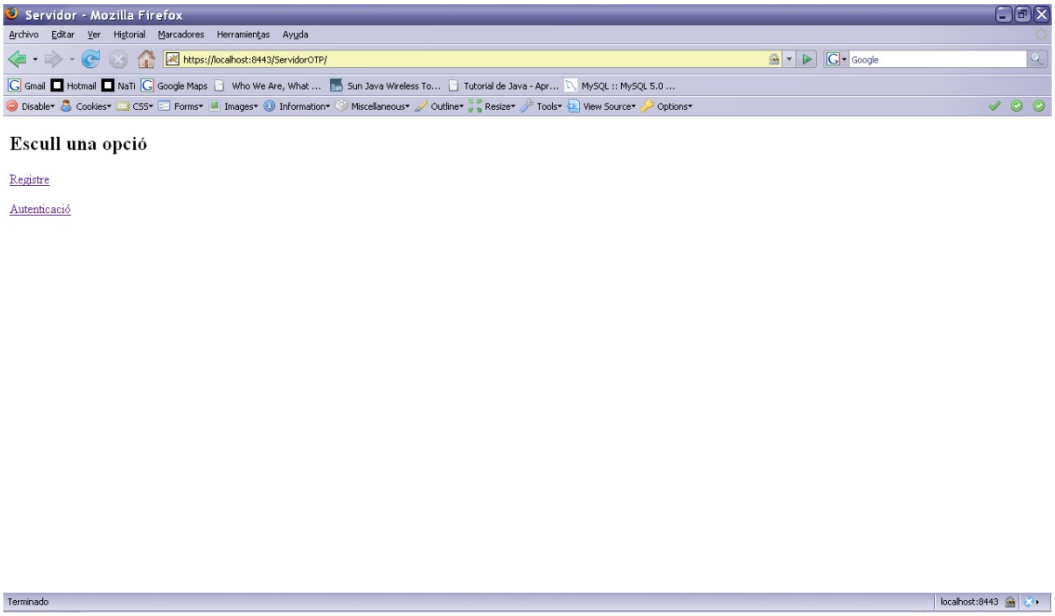


Figura 6.3. Menú del servidor

El link ens portarà a un formulari de registre com el de la Figura 6.4. Caldrà introduir el nom d'usuari i la clau web escollits.

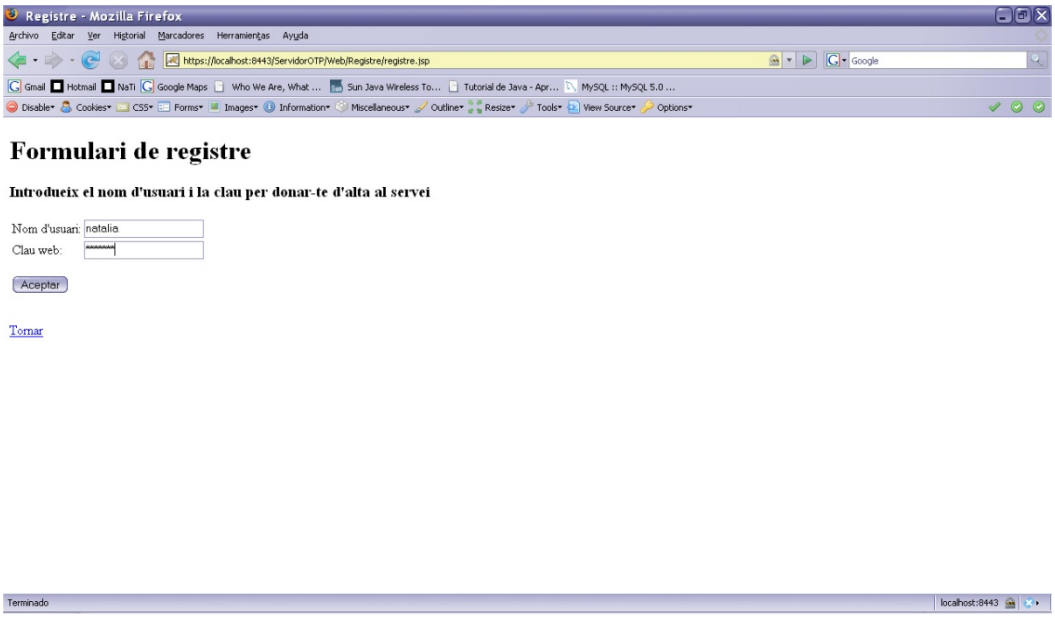


Figura 6.4. Formulari del registre

En quant s'emmagatzemen les dades a les taules corresponents de la base de dades, apareixeran per pantalla les dades (Figura 6.5) a introduir al telèfon mòbil. Tornarem al menú principal amb el link *Tornar*.

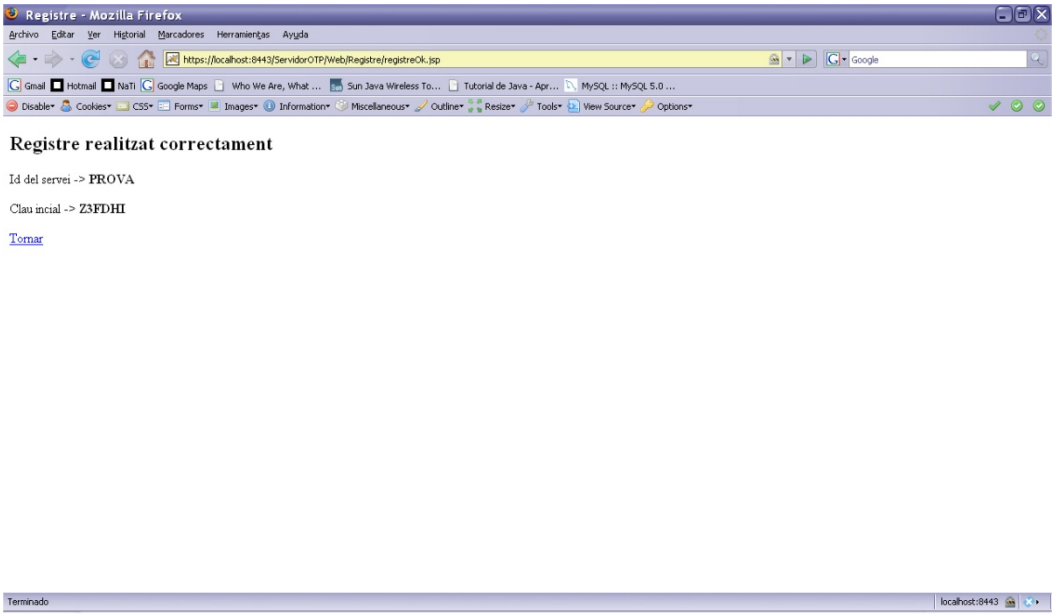


Figura 6.5. Usuari registrat correctament

Agafarem el telèfon mòbil i escollirem l'opció Afegir Servei (Figura 6.6) del menú principal de l'aplicació. Seguidament introduirem les dades (Figura 6.7) i, un cop introduïdes, si mirem la llista de serveis disponibles comprovarem que el servei ha estat introduït correctament (Figura 6.8).

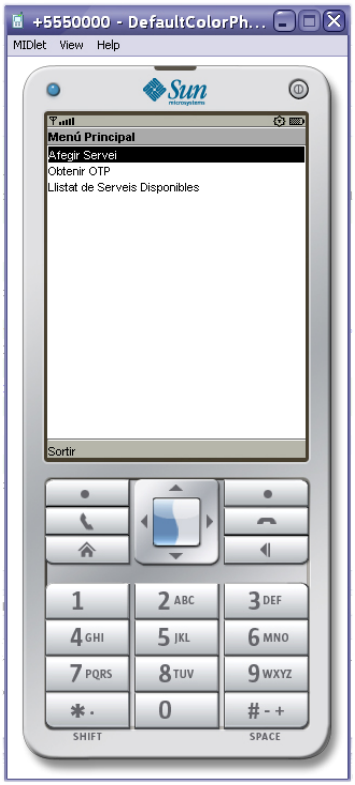


Figura 6.6. Menú del client

Figura 6.7. Formulari d'introducció de serveis i claus inicials

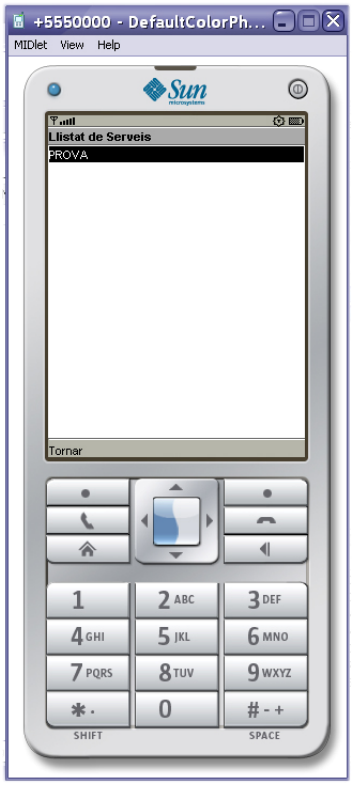


Figura 6.8. Llistat de serveis afegits

Després d'escollir l'opció Autenticació del menú principal (Figura 6.3) ens apareixerà un formulari on haurem d'introduir les dades utilitzades per fer el registre (Figura 6.9).

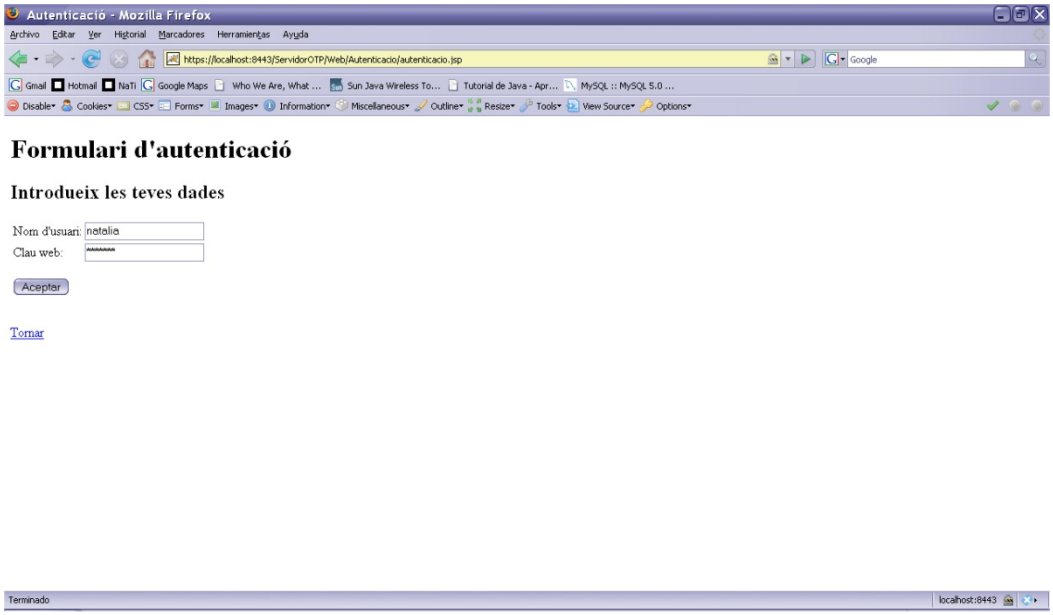


Figura 6.9. Formulari d'autenticació (Introducció del nom d'usuari i clau Web)

Si les dades introduides són correctes en apareixerà la pàgina de la Figura 6.10 per indicar-nos que tot ha estat bé. Si no ho fossin, el missatge seria d'error i hauriem de tornar a la pàgina anterior, la de la Figura 6.9.

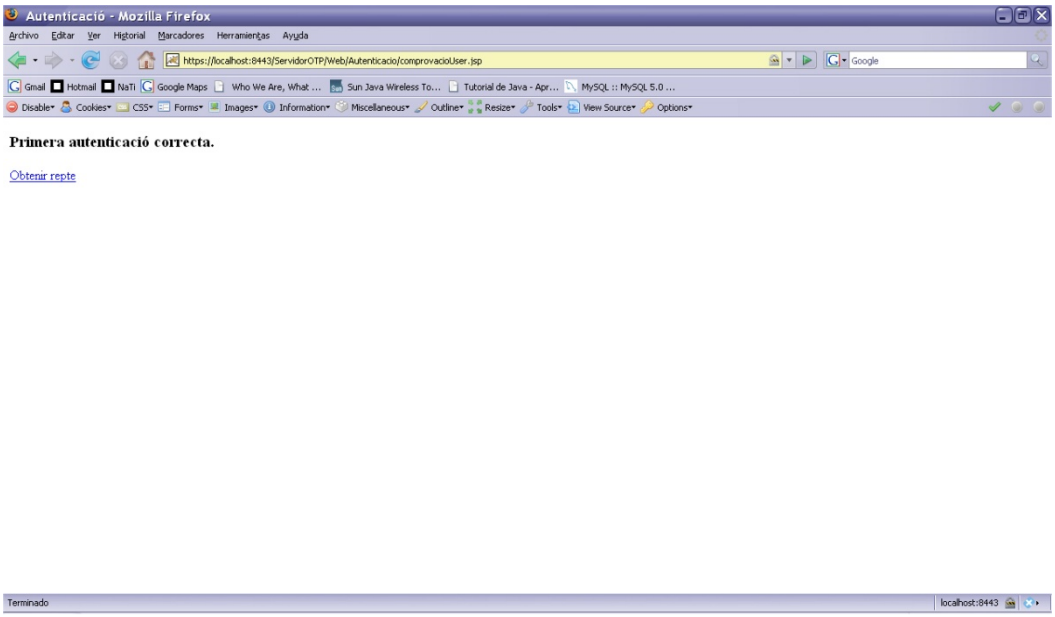


Figura 6.10. Primera autenticació (clau Web) correcta

Al clicar l'opció *Obtenir repte* (Figura 6.10) passarem a la pàgina de la Figura 6.11 i obtidrem el repte. Agafarem l'opció obtenir OTP del menú de l'aplicació del telèfon mòbil (Figura 6.12) i introduirem el nom del servei i el repte donat pel servidor (Figura 6.13). Es calcularà la OTP i ens sortirà el valor a introduir (Figura 6.14) al formulari de la pàgina Web (Figura 6.11).

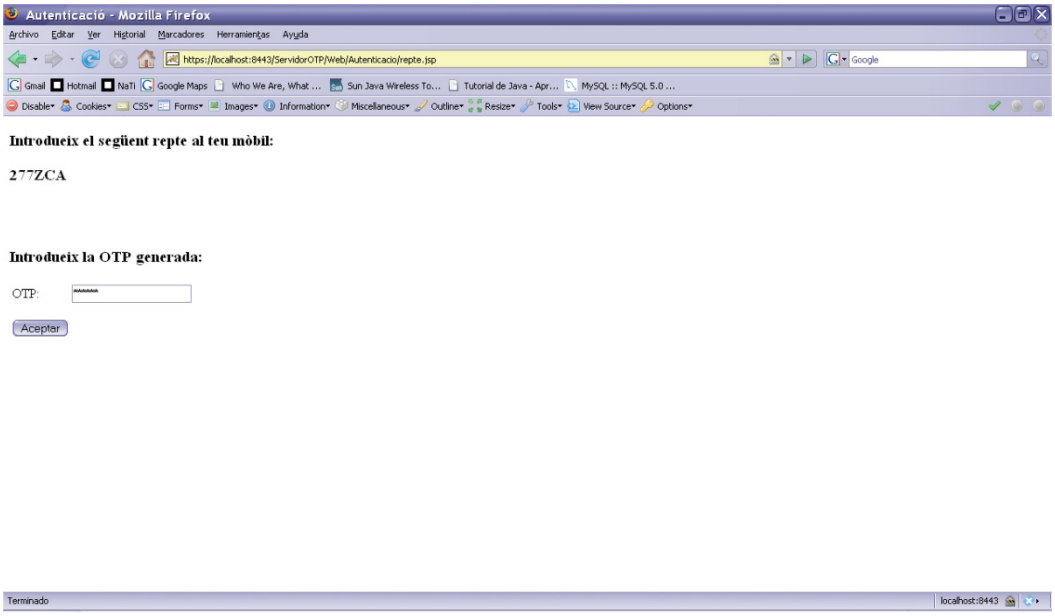


Figura 6.11. Visualització del repte i introducció de la OTP

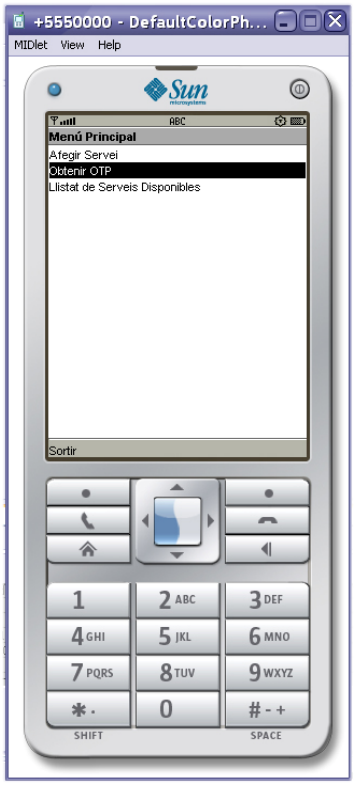


Figura 6.12. Escollir al menú l'opció d'obtenir OTP

Figura 6.13. Introducció del nom del servei i el repte

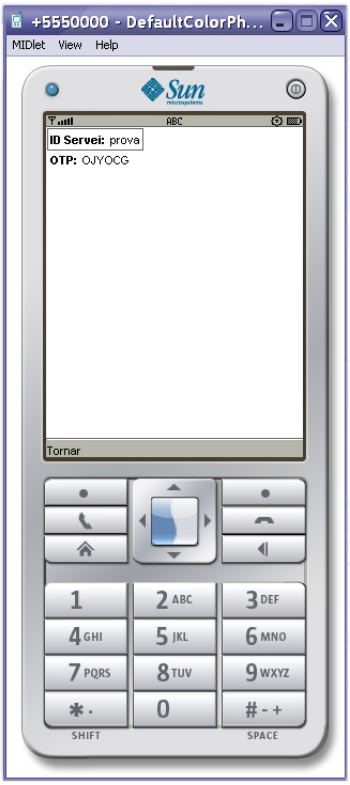


Figura 6.14. Visualització de la OTP

Si la OTP és correcte, un missatge ens ho confirmarà (Figura 6.15). Si no ho és, tindrem tres intents per tornar a introduir la OTP fins que el servidor ens proporcioni un altre repte. Un cop acabats els tres reptes que ens pot donar, l'accés al servei quedarà bloquejat.

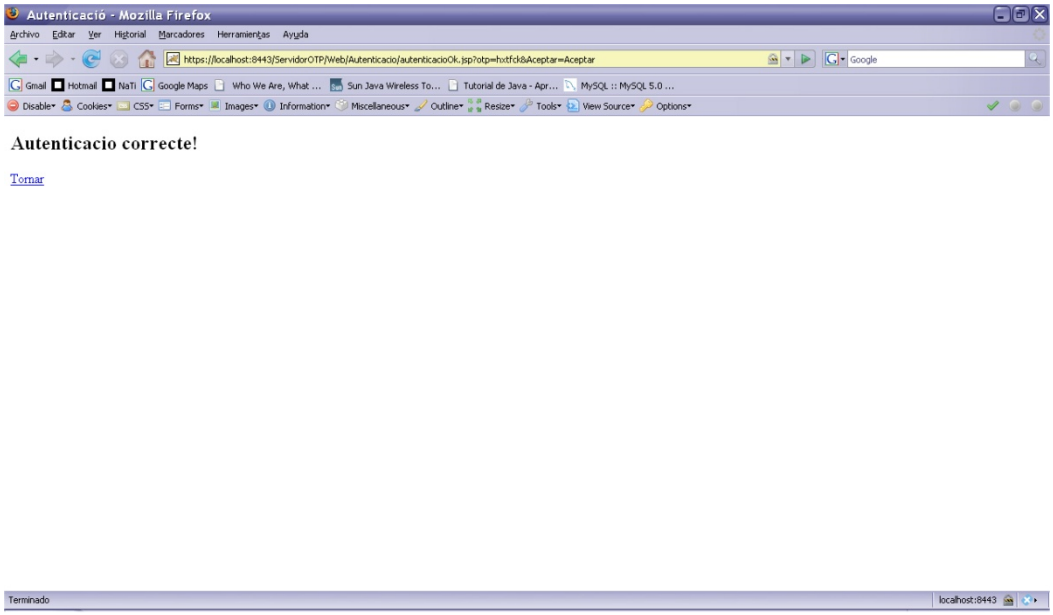


Figura 6.15. Autenticació amb la OTP realitzada

6.4. Avaluació de seguretat

En aquest apartat conté l'avaluació de seguretat de la solució. Estarà dividit en dues parts on s'explicaran els possibles atacs i les contramesures que es podrien aplicar o s'han aplicar per evitar-los.

6.4.1. Dispositiu mòbil

Possibles atacs:

1. Un atacant pot intentar robar un telèfon mòbil per fer-se passar per un usuari vàlid.
2. Un atacant pot intentar manipular el mòbil per aprendre la clau secreta (clau inicial).
3. Un usuari maliciós pot veure l'usuari del telèfon mòbil introduint la clau inicial o el repte.

Contramesures:

1. S'hauria de protegir l'aplicació amb una contrasenya que podria ser el número PIN del mòbil per evitar que l'usuari hagi de recordar una clau més.

2. La clau inicial guardada està xifrada amb un algorisme d'encriptació fort. Guardem el hash de la clau i no la clau en sí mateixa.

   La llibreria usada per generar la base de dades, guarda les taules en *RecordStore's* [39]. Un *Record* és el format donat a les dades guardades de forma persistent a la memòria del telèfon mòbil, és a dir, les dades no volàtils. Un *RecordStore's* consisteix en una col·lecció de *Record's* que es mantenen persistents en memòria a través d'invocacions múltiples del MIDlet. Usant la classe *RecordStore* i el mètode *setMode* en mode privat, podem aconseguir que cap altre aplicació del mòbil pugui tenir accés a la taula creada. Per fer tot això, hem de modificar la llibreria J2MEMicroDB.

   La modificació de la llibreria i el requeriment d'una contrasenya, fan que tant l'aplicació com la taula siguin totalment segurs. Per fer que l'aplicació ens demani una contrasenya cada cop que volem executar el MIDlet, farem una idea semblant a la del registre al servidor, un sistema usuari/contrasenya tots dos estàtics i guardats de forma segura com la taula.

3. Perquè ningú, excepte l'usuari vàlid, pugui veure dades sensibles a l'hora d'introduir-les al telèfon mòbil, és podrien usar dues tècniques conjuntes. La primera seria crear una connexió per Bluetooth entre el telèfon mòbil i l'ordinador personal de l'usuari i, la segona, contractar una xarxa GSM per enviar missatges de text i implementar en la nostra aplicació un mòdul que permetés rebre aquests missatges i verificar l'autenticitat dels mateixos.

6.4.2. Connexió a Internet

Possibles atacs:

1. Un atacant pot intentar fer-se passar per un usuari vàlid i esbrinar la contrasenya del servei.
2. Un atacant pot estar mirant quan mostrem per pantalla dades sensibles.
3. Un atacant pot segrestar la sessió per tenir un accés no autoritzat.
4. Un atacant pot estar "escoltant" en la connexió per intentar aprendre les contrasenyes o qualsevol dada sensible.
5. Un atacant pot muntar un atac Man-in-the-middle.

Contramesures:

1. Les contrasenyes d'un sol ús són impossibles d'esbrinar només sabent una ja que són totalment aleatòries. A part, tenim dues autenticacions abans de poder accedir a un servei.

2. Un cop aplicades les contramesures del dispositiu mòbil, l'atac que pot fer-se per culpa de mostrar dades per pantalla desapareixerà.

3. Els altres atacs els evitarem amb connexió segura amb SSL al servidor.

7. Conclusions

Aquest projecte tenia com a objectiu dissenyar una aplicació per a dispositius mòbils capaç de generar contrasenyes d'un sol ús per augmentar la seguretat en els accessos a determinats serveis. Un usuari ja no haurà de memoritzar totes les contrasenyes que tingui pels serveis als que estigui registrat, sinó que mitjançant el seu telèfon mòbil podrà generar una contrasenya aleatòria cada cop que es vulgui autenticar en cada un d'ells.

L'objectiu fixat ha estat realitzat amb èxit ja que s'ha aconseguit fer l'aplicació del mòbil i un servidor per a un servei. Ambdós estan totalment sincronitzats i, tan el registre com l'autenticació, es poden fer perfectament. Les dues aplicacions són molts senzilles i visualment molt simples però fan totes les funcions proposades, a més de ser totes dues molt fàcils d'usar per l'usuari.

Ha hagut certes complicacions en el desenvolupament. La majoria han sigut en el tema de les llibreries, sobretot la SATSA. És difícil trobar la documentació sobre els dispositius que poden fer-la servir i no hi ha gens d'informació sobre el mòdul WIM de la targeta SIM, cosa que ha fet perdre molt de temps. Trobar el nivell d'ofuscació per les llibreries *BouncyCastle* tampoc va ser ràpid. A part d'això es requerien uns coneixements bàsics sobre J2ME i JSP per la implementació de les dues aplicacions. La sincronització entre els dos dispositius i el control de les possibles fallides en el servidor, un cop ben pensats, es van solucionar ràpidament.

El nivell de seguretat de l'aplicació del client és mitjà, però amb les propostes explicades en l'apartat d'avaluació de seguretat, com la connexió Bluetooth o l'enviament de missatges de text per les dades semsibles, pren un nivell de seguretat molt alt.

Hem creat una aplicació oberta i lliure basada en estàndard oberts, destinada a dispositius innovadors (telèfon mòbil) que alhora és interoperable amb qualsevol altre aparell i que facilita l'autenticació dels usuaris als seus serveis. Amb totes aquestes característiques hem aconseguit un programa basat en els principis de l'OATH.

Per finalitzar, només dir cada dia hi ha més problemes en la seguretat de les dades dels usuaris i que gràcies a aplicacions com la realitzada i idees semblants ajudem augmentar la seguretat a Internet i poder fer transaccions a través de tot el món sense preocupar-nos.

7.1. Treball futur

Una part del treball futur, és la realització d'una pàgina web més visual i acurada mitjançant l'ús de fulls CSS (Cascade Style Sheets) i de Java Servlets, sense renunciar a que segueixi fàcil i intuïtiva de fer servir per l'usuari.

Una altra part seria contractar una xarxa GSM per l'enviament de missatges de text per les dades més sensibles, la clau inicial i el repte; poder establir una connexió per Bluetooth entre el telèfon mòbil i l'ordinador personal per tal de fer més còmode i alhora més segura la transmissió de les dades entre ambdós dispositius; i fer les modificacions de la llibreria usada per crear la base de dades del telèfon mòbil per protegir la taula amb les dades sensibles.

Bibliografia

[1] Initiative for open authentication, *OATH - initiative for open authentication | All users, all devices, all networks*. http://www.openauthentication.org/. [Consulta: 13 de Juny 2008]

[2] Menezes,Alfred J.; Van Oorschot, Paul C.; Vanstone, Scott A., *Handbook of Applied Cryptography*. New York: CRC Press, 1996. 780 p. ISBN:0849385237.

[3] Aladdin Knowledge Systems, Inc., *Aladdin Knowledge Systems - Protección Software, Seguridad Internet, Filtro de contenidos, Gestión de Identidad Digital*. http://www.aladdin.es/. [Consulta: 13 de Juny 2008]

[4] Entrust, *PKI, (Public Key Infrastructure), Outsourced and Managed PKI Software Services by Entrust. Entrust Provides Multi-Factor Authentication and Strong Two Factor Authentication for Online Transactions. Entrust's Internet Transaction Monitoring Platform provides a solution for Online Fraud Dectection*. http://www.entrust.com/. [Consulta: 13 de Juny 2008]

[5] Envoy Data Corporation. Active Indentity Solutions. http://www.itsecuritymall.com/actividentity/AAAserver.html. [Consulta: 13 de Juny 2008]

[6] ipsCA, IPS Certification Authority, S.L.; *ipsCA, Servicios de firma electronica y certificacion digital,Electronic Signature and Digital Signature Software Solutions - Wildcard SSL Certificate Wildcard Certificates SSL Certificates Free SSL Secure Server Certificate Branded SSL Certificate Authority Wildcard Certificates*. http://www.ipsca.com/. [Consulta: 13 de Juny 2008]

[7] Me, Gianluigi; Pirro, Daniel; Sarrecchia, Roberto. A mobile based approach to strong authentication on Web [Format pdf]. Italia: IEEE Computer Society, 2006. 5 p.

[8] Hallsteinsen, Steffen; Van Thanh, Do; Jørstad, Ivar. *Using the mobile phone as a security token for unified authentication* [Format pdf]. Noruega: IEEE Computer Society, 2007. 6 p.

[9] Straub, Matthias. *Mobile-OTP: Strong Two Factor Authentication with mobile phones*. Actualizació 14 d'abril de 2008. http://motp.sourceforge.net/. [Consulta: 10 de Juny 2008]

[10] Sun Microsystems, Inc., *Java ME Techonology*. http://java.sun.com/javame/technology/index.jsp. [Consulta: 10 de Juny 2008]

[11] Sun Microsystems, Inc. Overview MID (Profile). http://java.sun.com/javame/reference/apis/jsr118/. [Consulta: 11 de Juny 2008]

[12] Sun Microsystems, Inc. *Security and Trust Services API (SATSA)* [Format pdf]. Versió 1.0, Juliol 2004. 214 p.

[13] Casany, Maria José; Alier, Marc; Casado, Pablo. J2MEMicroDB. http://morfeo.upc.es/crom/course/view.php?id=5. [Consulta: 10 de Juny 2008]

[14] Van Ham, Gert, *Lightcrypto (lightweight cryptographic library)* http://jcetaglib.sourceforge.net/lightcrypto/. Octubre de 2003. [Consulta: 10 de Juny 2008]

[15] Sun Microsystems, Inc. *Java SE Techonology at a Glance*. http://java.sun.com/javase/technologies/index.jsp. [Consulta: 10 de Juny 2008]

[16] Sun Microsystems, Inc. *Java SE Technologies - Database*. http://java.sun.com/javase/technologies/database/. [Consulta: 10 de Juny 2008]

[17] Sun Microsystems, Inc. *JavaServer Pages Technology*. http://java.sun.com/products/jsp/. [Consulta: 10 de Juny 2008]

[18] The Apache Software Foundation, *Apache Tomcat*. http://tomcat.apache.org/. [Consulta: 10 de Juny 2008]

[19] Wikipedia, *MySQL*. http://es.wikipedia.org/wiki/MySQL. [Consulta: 10 de Juny 2008]

[20] Seidler, Kai 'Oswald', Apache Friends – XAMPP. Actualització 24 de desembre 2007. http://www.apachefriends.org/en/xampp.html. [Consulta: 10 de Juny 2008]

[21] Sun Microsystems, Inc. Sun Java Wireless Toolkit for CLDC Overview. http://java.sun.com/products/sjwtoolkit/overview.html. [Consulta: 10 de Juny 2008]

[22] Sun Microsystems, Inc. *The Mobile Service Architecture Specification*. http://developers.sun.com/mobility/midp/articles/msaintro/. [Consulta: 10 de Juny 2008]

[23] Netbeans, *Welcome to Netbeans*. http://www.netbeans.org/. [Consulta: 10 de Juny 2008]

[24] Bialoglowy, Marek. *Bluetooth Security Review*. Actualització 24 de maig de 2005. http://www.securityfocus.com/infocus/1830. [Consulta: 15 de Juny 2008]

[25] Tablado, Alberto Moreno. *Seguridad Mobile*. http://gospel.endorasoft.es/bluetooth/seguridad-bluetooth/blue-mac-spoofing.html. [Consulta: 15 de Juny 2008]

[26] Crockford, Douglas. *Base32 Encoding*. http://crockford.com/wrmg/base32.html. [Consulta: 10 de Juny 2008]

[27] Eastlake, D.; Jones, P.; *US Secure Hash Algorithm 1 (SHA1)*. http://www.ietf.org/rfc/rfc3174.txt. Setembre, 2001. 22 p. [Consulta: 10 de Juny 2008]

[28] Schaad, J.; Housley, R.; *Advanced Encryption Standard (AES) Key Wrap Algorithm*. http://www.networksorcery.com/enp/rfc/rfc3394.txt. Setembre, 2002. 41 p. [Consulta: 10 de Juny 2008]

[29] Eng, Asgaut. *Cipher FeedBack Mode*. Actualització 10 d'abril de 1996. http://www.pvv.ntnu.no/~asgaut/crypto/thesis/node16.html. [Consulta: 10 de Juny 2008]

[30] Elondra, *OpenBaseMovil*. Actualització 24 de Maig de 2008. http://www.openbasemovil.org/. [Consulta: 10 de Juny 2008]

[31] Wireless Application Forum, Ltd., *Wireless Application Protocol Identity Module Specification* [Format pfd]. Versió del 5 de novembre de 1999. 98 p.

[32] Nokia, *Forum Nokia - resources for mobile application developers*. http://www.forum.nokia.com. Actualització 2 de juny de 2008. [Consulta: 10 de Juny de 2008]

[33] S60, *S60 Phones*. http://www.s60.com/life/s60phones/browseDevices.do?edition=THIRD_EDITION&region=&manufacturer=&sortBy=MANUFACTURER_AND_DATE. [Consulta: 10 de Juny de 2008]

[34] Sony-Ericsson, *Java platforms and screen sizes for Sony Ericsson feature and mass-market phones* [Format pdf]. 2 p.

[35] Nokia, *Device Details – Nokia 6234*. http://www.forum.nokia.com/devices/6234. [Consulta: 10 de Juny de 2008]

[36] Rescorla, E., *HTTP over TLS*. http://www.ietf.org/rfc/rfc2818.txt. RTFM, Inc., 2000. 7 p. [Consulta: 14 de Juny 2008]

[37] Direks, T.; Allen, C., *The TLS protocol Version 1.0*. http://www.ietf.org/rfc/rfc2246.txt. Certicom, gener de 1999. 80 p. [Consulta: 14 de Juny 2008]

[38] Sun Microsystems, Inc. *Keytool – Key and Cretificate Management Tool*. http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html. [Consulta: 14 de Juny 2008]

[39] Gosh, Soma. *The J2ME record management system*. http://www.ibm.com/developerworks/library/wi-rms/. [Consulta: 14 de Juny 2008]

Firmat: Natalia Miguel Álvarez

Bellaterra, Juny de 2008

RESUM

Aquest projecte consisteix en fer l'anàlisi, disseny i implementació d'un sistema d'autenticació a través de contrasenyes d'un sol ús (One Time Password -OTP-) per a dispositius mòbils. Per evitar l'ús de contrasenyes estàtiques farem una aplicació per a telèfons mòbils capaç de generar contrasenyes aleatòries gràcies a uns paràmetres previs, així com de poder tenir un registre dels serveis on poden ser utilitzades. Partirem d'un protocol repte/resposta on l'usuari interactuarà amb el seu telèfon mòbil i un ordinador personal amb una connexió a Internet. Podrà registrar-se i, introduint certes dades al mòbil que li proporciona el servidor, podrà fer el procés d'autenticar-se per poder accedir a zones restringides del servei.

RESUMEN

Este proyecto consiste en hacer el análisis, diseño i implementación de un sistema de autenticación a través de contraseñas de un solo uso (One Time Password –OTP-) para dispositivos móviles. Para evitar el uso de contraseñas estáticas haremos una aplicación para teléfonos móviles capaz de generar contraseñas aleatorias gracias a unos parámetros previos, así como de poder tener un registro de los servicios donde pueden ser utilizadas. Partiremos de un protocolo reto/respuesta donde el usuario interactuará con su teléfono móvil y un ordenador personal con una conexión a Internet. Podrá registrarse y, introduciendo ciertos datos en el móvil que le proporcionará el servidor, podrá hacer el proceso de autenticarse para poder acceder a zonas restringidas del servicio.

ABSTRACT

This Project consists of the analysis, design and implementation of a One Time Password system for mobile devices. To avoid the use of static passwords, we will develop a mobile phone application capable of generating random passwords from previous parameters, and storing a register containing the services where they might be used. We will start from a challenge/response protocol. The user will interact through his mobile phone and a personal computer connected to the Internet. He will be able to register and, introducing certain data given from the server in his cell phone, he might authenticate himself to access the service's restricted zones.