Universitat de les Illes Balears

escola politècnica superior

Treball Final de Grau

Cupons Electrònics

ANTONI GORNALS LLITERAS

Tutor
Pep Lluís Ferrer Gomila

Escola Politècnica Superior
Universitat de les Illes Balears
Palma, 1 de juliol de 2018

SUMARI

ÍNDEX DE FIGURES

ACRÒNIMS

ACID Atomicity, Consistency, Isolation and Durability

JSON JavaScript Object Notation

API Application Programming Interface

MVC Model-View-Controller

JSP Java Server Pages

SPA Single Page Application

ORM Object Relational Mapping

SQL Structured Query Language

HTTP HyperText Transfer Protocol

DSA Digital Signature Algorithm

CA Certificate Authority

RSA Rivest, Shammir and Adleman

WSN Wireless Sensor Network

WPAN Wireless Personal Area Network

TFG Treball Final de Grau

RESUM

Amb els avanços de la tecnologia mòbil, les aplicacions de comerç electrònic cada vegada més van a la plataforma mòbil. Aquests avanços han donat peu a una evolució en els cupons tradicionals en paper. Ara, els cupons també han passat a la tecnologia mòbil pel que es presenta el concepte de *m-coupon* o cupó mòbil.

El que tracte aquest Treball Final de Grau (TFG) és una implementació d'aquest nou concepte de cupó mòbil, amb la principal particularitat que aquest presenta un sistema on es comparteixen els cupons entre els mateixos usuaris.

El que s'ha proposat és un esquema on el fabricant té una base de dades amb diferents usuaris registrats, on aquests tendran accés a les diferents ofertes de cupons disponibles. Una vegada un dels usuaris es disposa a obtenir una d'aquestes ofertes, aquest podrà fer ús del cupó per obtenir un benefici o servei, per altra banda, també tendrà l'opció de compartir el cupó amb un altre usuari, tant aquest estigui o no ho estigui registrat al sistema. Hem de tenir en compte que un cupó entregat per un usuari registrat al sistema sempre obtindrà més beneficis que no una entrega d'un usuari que no estigui registrat al sistema.

Apart de les funcionalitats que ens proporciona el sistema, aquestes s'han de dur a terme d'una manera segura i eficient que s'anirà explicant al llarg del document.

CAPÍTOL 1

INTRODUCCIÓ

Les noves tecnologies Web i l'ús d'Internet han avançat a passos gegantescs. Aquest fet ha conduït a canviar la manera en què es fa el negoci en general, i conseqüentment en el comerç en particular. Les empreses han hagut d'oferir noves oportunitats als consumidors i consumidores, per adequar-se a aquesta nova demanda, damunt les plataformes mòbils. Aquest nou enfocament condueix a parlar del concepte *Mobile Commerce*.

S'està transformant el model que es coneixia fins ara dins aquest sector de comerç, el *e-commerce*, però cal destacar que no l'està substituint, sols l'està complementant, ja que mentre aquest darrer engloba totes les transaccions que es realitzen a través d'Internet, el *m-commerce* inclou les compres i les ventes realitzades a través de telèfons mòbils intel·ligents. Aquest comerç afectarà significativament a les empreses i als seus clients, proporcionant grans beneficis a les dues parts.

Aquesta compra - venta de productes proporciona una gran flexibilitat en els serveis que s'ofereixen dins la plataforma mòbil. Un dels grans avantatges que ofereix és la geolocalització, la qual permet a les empreses oferir els seus serveis als clients segons en la ubicació en la que es trobin. Aquest nou concepte pot comportar un augment de les ventes i una expansió del mercat d'una forma senzilla i eficaç, ajustant-se a cada tipus de mercat, facilitant el procés de compra, millorant la imatge empresarial i facilitant la creació de nous clients fidels.

El possible augment de ventes va lligat a una reducció dels costs, ja que desapareixen els intermediaris entre client - proveïdor (personal de tenda, tendes físiques, material, etc). Permet que la comunicació entre client - proveïdor, i les relacions entre les seves comunicacions, tant internes com externes, millorin notòriament, ja que introdueix un accés senzill i directe de les parts implicades, fent sondejos electrònics als clients recopilant possibles millores de qualitat, i oferint actualitzacions immediates del producte que s'ofereix. Com a gran avantatge a destacar és que el producte demandat

sol oferir-se a un preu més baix per al client respecte al preu que s'ofereix en tenda. Cal mencionar que dins aquest comerç hi ha una major competència, ja que per a cada oferta existeix un ampli ventall de diferents opcions segons les restriccions del client. D'aquesta manera, és ell mateix qui té a la seva disposició els productes a on elegir i avaluar les diferents alternatives respecte el que vol.

Perquè sigui possible aquest model de comerç és necessari que els consumidors disposin de dispositius mòbils intel·ligents. Gràcies a la incorporació d'aquests dispositius els consumidors han obtingut grans avantatges respecte al comerç tradicional, i fins i tot, respecte al *e-commerce*. Principalment l'avantatge de la localització i el temps, ja que la majoria dels consumidors, en l'actualitat, duen els telèfon mòbil dins la butxaca la major part del temps i aquest fet permet realitzar transaccions en qualsevol lloc i moment.

Encara que hi hagi molts avantatges, s'ha de parlar també de les limitacions que existeixen dins *m-commerce* que poden frenar la compra per part dels consumidors. Es podria remarcar les dimensions de les pantalles dels dispositius, la lentitud de la connexió a Internet d'aquests mateixos, així com la potència limitada que tenen per processar les dades, la memòria o la limitació de la bateria.

Aquest comerç és un model nou que encara produeix desconfiança per part del client, respecte als pagaments o a l'enviament d'informació personal. Per poder contrarestar aquestes barreres de la compra *online*, les empreses ofereixen avantatges comercials instantànies, normalment relacionades amb uns preus més baixos, ofertes o promocions, i d'aquesta manera incentivar als consumidors a realitzar aquest tipus de compres i donar visibilitat i anar guanyant confiança per a aquest nou model, el *m-commerce*.

Una de les eines remarcables d'aquest tipus de comerç, i en la qual es basa aquest projecte, són els *m-coupon* (*Mobile Coupon*), que provenen dels cupons ja existents dins el comerç en el qual es basa el *m-commerce*, en el *e-commerce*, per tant, aquesta base és l'*e-coupon*.

1.1 *M-Coupons*

Un cupó mòbil és un document electrònic sol·licitat o entregat al telèfon mòbil, que pot ser intercanviat per obtenir un descompte econòmic o també per obtenir un producte o servei.

Normalment, els cupons els genera un fabricant o empresa que vol vendre o oferir una promoció d'alguns dels seus productes. Aquests es solen utilitzar per atreure clients potencials, ja sigui sols amb la promoció, o també creant algun tipus de sistema de fidelització.

Aquests cupons, habitualment, s'entreguen a una tenda física o virtual que està contractada pel fabricant que ha creat els diferents cupons. Així, el consumidor sols l'ha de presentar amb l'aplicació mòbil a les tendes que els validen i entregar el cupó quan fa una compra.

El seu ús presenta una sèrie d'avantatges:

- Genera una atracció automàtica per la compra dels diferents productes en promoció.
- Proporciona descomptes directes als possibles consumidors.
- Dóna la possibilitat d'obtenir informació sobre consumidors potencials.
- Facilita la distribució dels cupons.

Així i tot, les aplicacions mòbils presenten una sèrie d'inconvenients. Els desenvolupadors d'aquest *software* els han de tenir en compte, ja que han de satisfer totes les necessitats damunt una plataforma menys potent que un ordinador convencional. I no s'ha de deixar de costat el punt més important de tot *software*, i és que totes les comunicacions s'han de fer de manera fiable, segura i eficient.

En aquest treball fi de grau s'implementa l'esquema *Sharing secure m-coupon for peer-generated targeting via ewom communications* de *S.C. Hsueh i J.M. Chen* [1], en el qual un fabricant selecciona els clients específics de la base de dades que es troben registrats a la firma. El fabricant pot fer la selecció dels clients als quals vol dirigir el cupó amb una major rapidesa, ja que és ell qui elegeix el tipus de públic al què dirigirà el cupó específic. Una vegada feta la selecció, enviarà dos llibres de cupons al client, però, aquests dos llibres seran diferents. Un correspondrà als cupons que posseeix el client i del qual fa ús, i l'altre, correspondrà als cupons que vol compartir aquest client amb qualsevol altre client potencial que encara no estigui registrat a la firma.

Una vegada obtinguts els dos llibres de cupons, el client té dues possibilitats:

1. Entregar el cupó propi obtenint el descompte o servei descrit.
2. Compartir el cupó amb algun client no registrat a la firma, de forma que aquest obté un cupó però que proporciona menys descompte que el que obté el client registrat.

La segona possibilitat que es dóna al client correspon a una nova manera d'atreure clients, que és compartint els cupons entre ells mateixos directament. D'aquesta manera parlen entre ells i s'expandeixen més aviat les diferents ofertes disponibles de la firma.

Els cupons tenen un valor intrínsec, referint-se al marc econòmic, i per això solen ser objecte d'alteracions i falsificacions. Les limitacions que poden donar els dispositius mòbils en potència de computació i en espai d'emmagatzematge, fan que no es puguin implementar mecanismes de seguretat complexos. Com a possible solució a aquest problema, en aquest treball fi de grau, l'enfocament s'ha fet de manera que un cupó presenti el màxim de computacions lleugeres, com poden ser els *hash*, funcions de baix

còmput computacional, i útils a l'hora de verificar la validesa dels cupons, encara que també s'han hagut d'implementar alguns còmputs més pesats, però, sempre intentant reduir-los al màxim.

Per poder donar una solució al sistema de cupons esmentat, el desenvolupament s'ha hagut d'estructurar en dues parts completament diferenciades, una part de servidor i una altra de client. Pel que fa referència a la part del servidor, s'ha desenvolupat una part que emula totes les comunicacions necessàries en les quals hi participen totes les entitats involucrades en un sol servidor. Totes i cada una de les parts del servidor s'han implementat de manera modular i senzilla, ja que si es vol aplicar a la vida real sigui relativament fàcil separar les diferents entitats a nivell lògic.

Pel que fa a la part del client, el desenvolupament ha estat la implementació d'una aplicació mòbil que permet dur a terme totes les interaccions necessàries entre ell i el servidor, per tal d'obtenir, compartir i entregar els diferents cupons disponibles per aquest client.

Una vegada contextualitzat el projecte es podrà entendre, més detalladament, com han estat totes les passes, tant del disseny com de la implementació, per tal de poder treballar amb aquest nou model de cupons.

CAPÍTOL 2

ESTAT DE L'ART

En els darrers anys, com ja s'ha indicat, la plataforma mòbil ha evolucionat molt ràpidament. Degut a aquesta evolució cada vegada més han anat sorgint més implementacions de cupons mòbils que ofereixen diferents serveis. La idea d'aquest capítol és veure algunes d'aquestes implementacions per veure quines similituds i diferències es poden trobar.

2.1 Estudi de propostes de *M-Coupons*

El que s'ha pogut comprovar durant la recerca que s'ha fet sobre les diferents implementacions de cupons mòbils és que molts dels elements que fan referència al protocol són comuns en quasi tots els articles. Per altra banda, en el que més solen diferir és en la tecnologia utilitzada per dur a terme la comunicació, entre els diferents dispositius. Per tant, s'ha separat aquest capítol en els següents apartats:

- Tecnologies per a la comunicació.
- Entitats que participen en els protocols.
- Requeriments en els protocols.
- Variables presents en els cupons.

2.1.1 Tecnologies per a la comunicació

L'augment en l'interès per integrar els sistemes de cupons als dispositius mòbils intel·ligents ha provocat una anàlisi entre les tecnologies disponibles per a realitzar la connexió entre els dispositius mòbils i els dispositius per a la verificació dels cupons. Les tres tecnologies principals són:

NFC, *Near Field Communication*

Near Field Communication [2] és una tecnologia de comunicació sense fils, molt còmoda per als consumidors, ja que es pot fer la comunicació fora haver de fer cap tipus de configuració prèvia, sols apropant dos dispositius mòbils molt a prop un de l'altre.

Tot i que no és una tecnologia molt costosa, encara no és molt coneguda i no està disponible en tots els dispositius mòbils.

QR, *Quick Response Codes*

Quick Response Codes [3] és un sistema d'emmagatzematge d'informació en una matriu quadrada de punts, dissenyada per ser llegida amb la càmera d'un dispositiu mòbil. Es tracta d'un codi de barres bidimensional creat per una companyia japonesa, aquests codis contenen la informació codificada en una etiqueta *QR* on hi apareixen caràcters alfanumèrics que poden contenir text simple, adreces *web*, etc...

Tot i ser barata i utilitzada, el procés de lectura és molt més lent en comparació a la tecnologia *Near Field Communication*.

BLE, *Bluetooth Low Energy*

Bluetooth Low Energy [4], comercialitzat com *Bluetooth Smart*, és una tecnologia que utilitza Wireless Personal Area Network (WPAN). En comparació a les versions anteriors de *Bluetooth* clàssic, *Bluetooth Smart* ofereix un consum i un cost considerablement reduïts mentre manté un rang de comunicació similar. La nova versió també ha permès l'aparició d'un nou tipus de dispositius anomenats *beacons*, dispositius que envien senyals *Bluetooth* periòdiques on els dispositius que estan escoltant poden emprar per a determinar la distància a un altre *beacon*. El principal avantatge d'aquests nous dispositius és que no necessiten la interacció de l'usuari.

2.1.2 Entitats que participen en els protocols

Les entitats més comunes en els diferents articles de cupons electrònics són les següents:

- **Fabricant** o *Manufacturer*: entitat que ordena l'emissió dels cupons.
- **Distribuïdor** o *Issuer*: entitat que s'encarrega d'emetre i distribuir els cupons.
- **Comerciant** o *Merchant*: entitat on s'ha d'entregar cada un dels cupons.
- **Usuari** o *User*: usuari que pot obtenir i utilitzar els cupons.

2.1.3 Requeriments en els protocols

Els requeriments més comuns en les diferents implementacions són:

- **Integritat**: permet identificar si el contingut del cupó ha estat modificat.

- **Autenticitat**: l'usuari ha de poder verificar qui ha creat el cupó.
- **No rebuig en origen**: cap de les parts pot negar haver enviat un missatge que ha enviat.
- **No falsificable**: només les entitats autoritzades poden crear cupons vàlids.
- **No gastar excessivament**: els cupons sols es poden entregar les vegades acordades per les parts.
- **Reutilitzable** : el cupó es pot emprar més d'una vegada.
- **Transferible**: un usuari pot transferir cupons a altres entitats.

2.1.4 Variables presents en els cupons

Les variables presents en els cupons més comunes en les diferents implementacions són:

- **Número de sèrie**: permet identificar un cupó de manera unívoca en cada una de les fases.
- **Temps de validesa**: fins quan el cupó és vàlid.
- **Data d'emissió**: quan es va emetre el cupó.
- **Signatura digital de l'emissor**: per donar validesa al cupó.
- **Identificador del dispositiu**: identificador que ens permet identificar de manera única un dispositiu.
- **Informació del cupó**: informació que indica que ens proporcionaran els diferents cupons, normalment beneficis o serveis.

2.2 Anàlisi de propostes de *M-Coupons*

A continuació s'explicaran una mostra de les diferents propostes estudiades.

2.2.1 mCoupons: Aplicació basada amb *NFC, Near Field Communication*

A l'article de Sandra Dominikus i Manfred Aigner [5] es proposa un sistema de cupons mòbils on la principal característica és que utilitza la tecnologia de comunicació *Near Field Communication, NFC*. Per tant, el que cerca es que simplement apropant dos dispositius un a l'altre es pugui emetre, lliurar o compartir un cupó. La idea darrera aquest sistema és que l'usuari tengui el mòbil amb la tecnologia *NFC* activada i simplement acostant-se a un dispositiu *NFC* passiu, en aquest cas anomenat *Activator*, l'usuari obtengui el cupó.

Un dels problemes que presenta aquest protocol és que encara no tots els dispositius són compatibles amb la tecnologia *NFC*.

2.2.2 Sistema basat amb *NFC* i WSN

A l'article de Sun-Kuk Noh Seong-Ro Lee i DongYou Choi [6] trobam com a particularitat d'utilització, a part de la tecnologia *NFC*, és el *NFC* basat amb *LBS, Location Based System* damunt la xarxa WSN. Aquest sistema *LBS* és un servei que ajuda a detectar les localitzacions dels dispositius que estan dins la xarxa *WSN*. Aquest tipus de sistema ens pot proporcionar informació útil, com per exemple, localitzacions de diferents comerciants per comprar algun producte en concret. El que es cerca principalment en aquest article és millorar el sistema d'entrega de serveis amb la finalitat d'atreure més clients.

El problema que presenta aquest protocol, és que també utilitza la tecnologia *NFC*, que com hem dit abans aquesta no està disponible en tots els dispositius.

2.2.3 Arquitectura per *M-Coupons* anònims

A l'article de Alberto Bartoli i Eric Medvet [7] ens trobam amb una arquitectura que cerca que el client no hagi de instal·lar cap aplicació dedicada a aquest protocol, sino que s'utilitzi alguna aplicació de confiança per així evitar complicacions en implementar una aplicació totalment nova.

Pel que respecte a l'anonimat dels cupons, simplement es basa en què un usuari obté un cupó i aquest el pot compartir amb qualsevol usuari, l'anonimat s'aconsegueix quan qualsevol usuari pot entregar aquest cupó, per tant, des del punt de vista del servidor no es sap qui ha entregat el cupó.

Aquest protocol té la particularitat de què utilitza la tecnologia de comunicació per codi *QR*, que s'ha esmentat anteriorment. Per tant, els problemes que presenta aquest protocol estan relacionats amb la velocitat que aquesta tecnologia proporciona.

CAPÍTOL 3

PROTOCOL M-COUPONS: SHARING SECURE M-COUPONS FOR PEER-GENERATED TARGETING VIA EWOM COMMUNICATIONS

Normalment els cupons s'envien al mòbil de l'usuari directament. A la proposta de Sue-Chen Hsueh i Jun-Ming Chen[1], el que es pretén és que sigui de tipus *peer-generated targeting*, és a dir, que un usuari que estigui registrat pugui reenviar el seu cupó a un altre usuari, fins i tot si no està registrat a la plataforma. Així la comunicació envers de ser d'empresa a usuari, es pot fer directament d'usuari a usuari. Aquesta proposta és la que és objecte d'implementació en aquest TFG.

Com ens defineix l'article, dos paràmetres essencials a conèixer del protocol són els següents:

- **Cadena** X: Aquest paràmetre fa referència al nombre de vegades que un cupó podrà ser entregat. Aquesta és una cadena que pertany a l'usuari que hagi obtingut el cupó, i es podrà entregar x vegades.
- **Cadena** Y: Apart de la cadena X, quan un usuari obté un cupó, també es proporciona una cadena Y. Aquesta serveix per compartir el cupó amb altres usuaris. Anàlogament a l'anterior, el cupó compartit es podrà entregar y vegades.

Aquestes són les dues maneres d'entregar els cupons. La diferència entre elles, apart del nombre de vegades que es pugui entregar, és que no tendran el mateix descompte. Una entrega que correspon a la cadena X tendrà més descompte que una de la cadena Y, en el cas de què el cupó estigui relacionat en obtenir algun tipus de descompte.

El fet que els cupons puguin obtenir-se entre els mateixos usuaris, pot fer que les ventes augmentin considerablement, ja que els usuaris que reben aquests cupons compartits provenen d'una part coneguda, en canvi, si els cupons provenen del fabricant

directament poden ser considerats *spam*. Per altra banda, aquest canvi, fa que el possible cost de *marketing* que tengui una fabricant, baixi de manera significativa, ja que ho fan els mateixos usuaris entre ells.

3.1 Requeriments del protocol

Els requeriments que aquest protocol complirà són els següents:

- **Confidencialitat**: totes les comunicacions que es considerin que transporten informació sensible es xifren amb la clau pública del receptor.
- **Autenticació**: per aconseguir aquest requeriment, s'han implementat firmes digitals per poder autenticar l'origen dels missatges, per altra banda s'ha implementat una fase de *log in* per autenticar als usuaris.
- **Integritat**: per solucionar aquest requeriment, s'han utilitzat les funcions de *hash*, funcions que ens permeten aconseguir la integritat de la informació.
- **No reutilitzable**: cada vegada que es fa una entrega d'un cupó, es verifica que no s'hagi entregat anteriorment i es descarta perquè no es pugui tornar a entregar si l'entrega s'ha efectuat correctament.
- **No falsificable**: sols les parts que estan autoritzades podran generar nous cupons vàlids.
- **Data d'expiració**: cada cupó tendrà un temps de vida determinat que s'haurà de respectar.
- **No rebuig en origen**: una vegada creat el cupó el fabricant no podrà negar haverlo creat, així com moltes de les comunicacions aniran acompanyades d'una firma de l'origen perquè el receptor pugui verificar que efectivament és ell i no ningú més.

3.2 Actors del protocol

3.2.1 Fabricant

El fabricant és qui ven bens als usuaris, i les seves funcions són:

- **Autoritzar qui seran els distribuïdors** encarregats de fer les entregues dels cupons.
- **Identificar membres potencials**, per maximitzar el profit.
- **Donar d'alta als usuaris**, i així, podrà identificar a tots els usuaris.
- **Inicialitzar les ofertes**, ja que és l'encarregat de generar les ofertes que estaran disponibles pels usuaris.
- **Verificar si s'ha entregat un cupó**, és a dir, una vegada s'ha entregat un cupó el fabricant marca el cupó perquè no pugui tornar a ser entregat.

3.2.2 Distribuidor

El distribuïdor s'encarrega de:

- **Entregar els cupons als usuaris**. Una vegada el fabricant ha generat l'oferta i un usuari vol una d'aquestes ofertes, és el distribuïdor l'encarregat de proporcionar el cupó.
- **Verificar els cupons**, és a dir, comprovar si el cupó entregat és vàlid i enviar la confirmació al fabricant perquè aquest marqui el cupó com a entregat.

3.2.3 Comerciant

El comerciant s'encarrega de:

- **Fer l'entrega dels bens**. Són els encarregats de proporcionar el bé adquirit per l'usuari.
- **Atendre als usuaris que volen fer una entrega d'un cupó**.

3.2.4 Comunitat d'usuaris

Deim que hi ha una comunitat d'usuaris perquè inclou els usuaris registrats i els usuaris no registrats, és a dir, els qui no s'han registrat al fabricant. La funcionalitat addicional que tenen els usuaris registrats és que poden compartir el cupó amb un usuari, fins i tot si no està registrat al fabricant.

3.3 Fases del protocol

Una vegada definides les entitats, resumim els participants de les fases del protocol, que seran:

- **Fabricant** (F)
- **Distribuïdor** (I)
- **Comerciant** (R)
- **Usuari Registrat** (U)
- **Usuari No Registrat** (NU)

Les fases les dividirem tenint en compte les dues entitats principals. Per una banda tenim els usuaris registrats (U) que participen en les següents quatre fases:

1. **Fase d'Emissió**: fase on es fa l'entrega dels cupons als usuaris registrats (U).
2. **Fase de Lliurament**: fase on un usuari registrat (U) fa ús d'un dels cupons que té disponibles.

3. **Fase de Neteja**: fase seguida del lliurament on el fabricant marca el cupó com a entregat.

4. **Fase de Compartiment**: fase on un usuari registrat (U) comparteix un cupó amb un altre usuari no registrat (NU)

Per altra banda tendrem que els usuaris no registrats (NU) participen en dues fases diferents:

1. **Fase de Lliurament**: fase quan un usuari no registrat (NU) farà entrega d'un cupó compartit.

2. **Fase de Registre**: fase on un usuari no registrat (NU) podrà registrar-se a un fabricant per obtenir els cupons oferts als usuaris registrats (U).

Una vegada introduïdes les fases i els diferents actors en que constarà el protocol, introduirem de manera breu com es fa la verificació d'aquests cupons, ja que aquest concepte és clau per entendre la resta de fases.

3.3.1 Verificació per cadena de *Hash*

Aquest protocol utilitza el sistema de verificació amb funcions de *hash*, com es pot veure al diagrama següent.

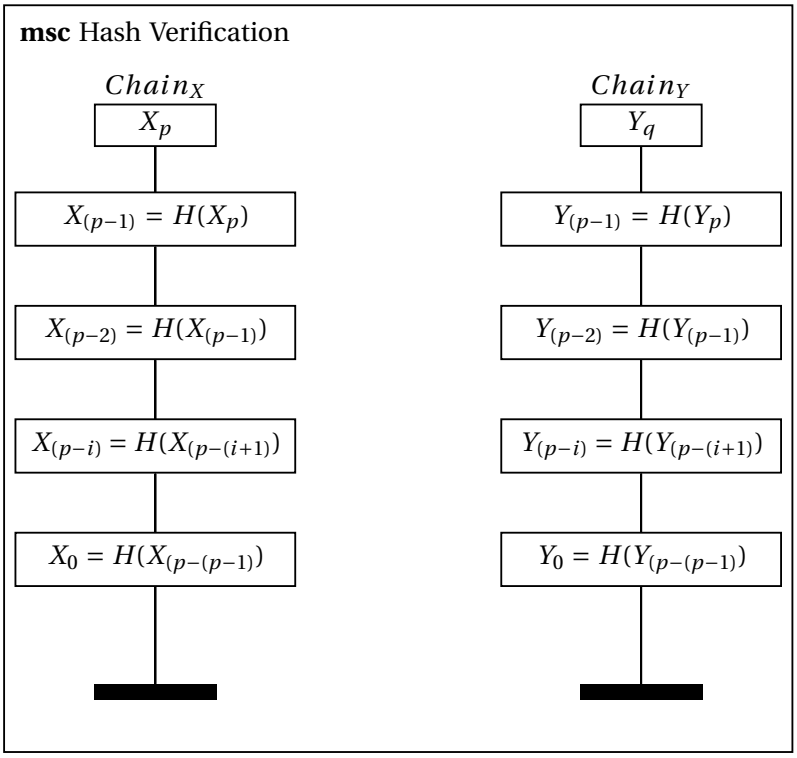

Figura 3.1: Seqüència de la creació dels valors arrel per a la verificació dels cupons.

Per explicar-ho d'una manera més clara, un cupó primerament consta de dos nombres p i q, aquests són els nombres que fan referència a les vegades que es podrà entregar cada un dels cupons, on p és el nombre de vegades que podrà entregar el cupó un usuari registrat i q és el nombre de vegades que es podrà entregar un cupó compartit. Per generar la cadena X i Y, el que es fa és calcular la funció de *hash* tantes vegades com marquin els valors p i q. Per tant, direm que els valors arrel X_0 està format per p vegades la funció de *hash* sobre el primer nombre aleatori X, i el mateix farem per la cadena Y. Aquest valor arrel permetrà al fabricant verificar si el cupó entregat es vàlid o no.

Els usuaris que vulguin entregar un cupó el que hauran de generar és la funció de *hash* tantes vegades com vegades ja hagi entregat el cupó, anomenat X_i o Y_i. Una vegada s'ha entregat el cupó, el fabricant, per comprovar que el cupó és vàlid, haurà de generar la funció de *hash* tantes vegades com faltin per arribar al valor arrel X_0, per verificar que l'origen és el vàlid.

3.3.2 Fases del protocol per a Usuaris Registrats

Dins aquest apartat definirem les diferents fases en les quals participen els usuaris registrats.

Fase d'Emissió

Com s'ha dit abans, aquesta fase defineix com un usuari registrat (U) ho farà per obtenir els cupons sobre les ofertes disponibles.

A continuació exposam la informació necessària per aquesta fase:

- ID_u: identificador únic de l'usuari registrat (U), assignat durant el procés de registre.
- p: nombre de vegades que es podrà utilitzar un cupó propi.
- q: nombre de vegades que es podrà utilitzar un cupó compartit.
- X_0: és el valor arrel que ens serveix per verificar si un cupó propi és vàlid.
- Y_0: és el valor arrel que ens serveix per verificar si un cupó compartit és vàlid.
- *EXD*: paràmetre que fa referència a la data d'expiració del cupó.
- *Inf*: la variable que consta d'informació rellevant del cupó, $Inf = EXD, p, q, X_0, Y_0, ID_u$.
- *SN*: número de sèrie únic per identificar un cupó.
- *MC*: és la variable que identifica el cupó, i aquesta està composta per $MC = X_0, Y_0, EXD, SN, p, q, ID_u$.
- S_u, S_i, S_f: firmes digitals, indicant quina de les parts l'ha generada.

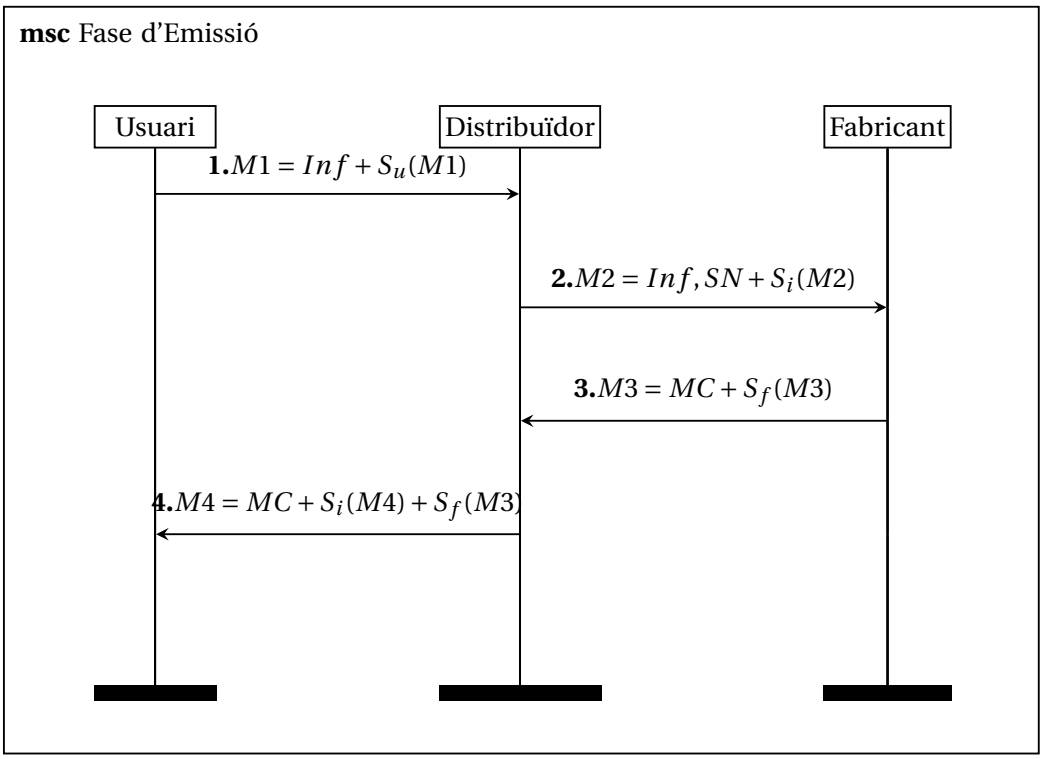

Figura 3.2: Fase d'Emissió

Protocol: *Fase d'Emissió*

1. U selecciona una de les ofertes disponibles per obtenir un cupó. Hem de tenir en compte que una oferta estarà formada per p, q i EXD. Abans de que U enviï el missatge haurà de dur a terme les següents passes:
   a) U agafa els valors de p i q per generar els valors arrel X_0 i Y_0 respectivament.
   b) Una vegada calculats els valors arrel, genera la variable Inf i, amb la clau privada xifra Inf per obtenir la firma digital.
   c) Tot seguit, envia el missatge a I.
2. Una vegada I ha rebut el missatge de U, primer de tot comprova que la firma és correcte. Un cop confirmat, genera SN i prepara el pròxim missatge a enviar, compost pel missatge anterior sencer, SN i la firma digital de I de tot el missatge a enviar a F.

3. Un cop F ha rebut el missatge de I, el primer que farà serà verificar les dues firmes tant la de I com la de U. Un vegada confirmades les firmes amb èxit, F genera la variable MC que serà el que enviarà a I, juntament amb la firma digital sobre MC.

4. I rep el missatge de F i primerament verifica la firma digital de F, una vegada verificada amb èxit, guarda la variable MC a la seva base de dades. Posteriorment, envia el missatge compost per MC, SN, la firma digital de F sobre MC i la firma digital de I sobre SN a U.

5. Finalment, U rep el missatge i simplement comprova que les dues firmes siguin correctes, confirmant així que ha obtingut el cupó correctament.

Figura 3.3: Intercanvi d'informació en la fase d'emissió.

Fase de Lliurament

La fase de lliurament defineix com un usuari registrat (U) farà l'entrega dels cupons que té disponibles.

A continuació exposam la informació necessària per aquesta fase:

- ID_r: identificador únic de cada un dels comerciants.
- *Nonce*: és un nombre aleatori que serveix al comerciant per inicialitzar la comunicació amb U i poder comprovar que realment és amb ell amb qui s'intercanvia informació.
- X_i: és el cupó que s'està entregant.
- i: és el valor de la cadena que s'està entregant.
- $R_i d$: és una variable que està formada per la funció de *hash* dels valors X_0, X_i i ID_u.
- Etiqueta (L): és la variable composta per la funció de *hash* dels valors ID_r i el *nonce*, variable que ens serveix per fer comprovacions d'autenticació de U.
- PU_i: xifrat amb la clau pública del distribuïdor.
- S_u, S_r, S_i: firmes digitals, indicant quina de les parts l'ha generada.

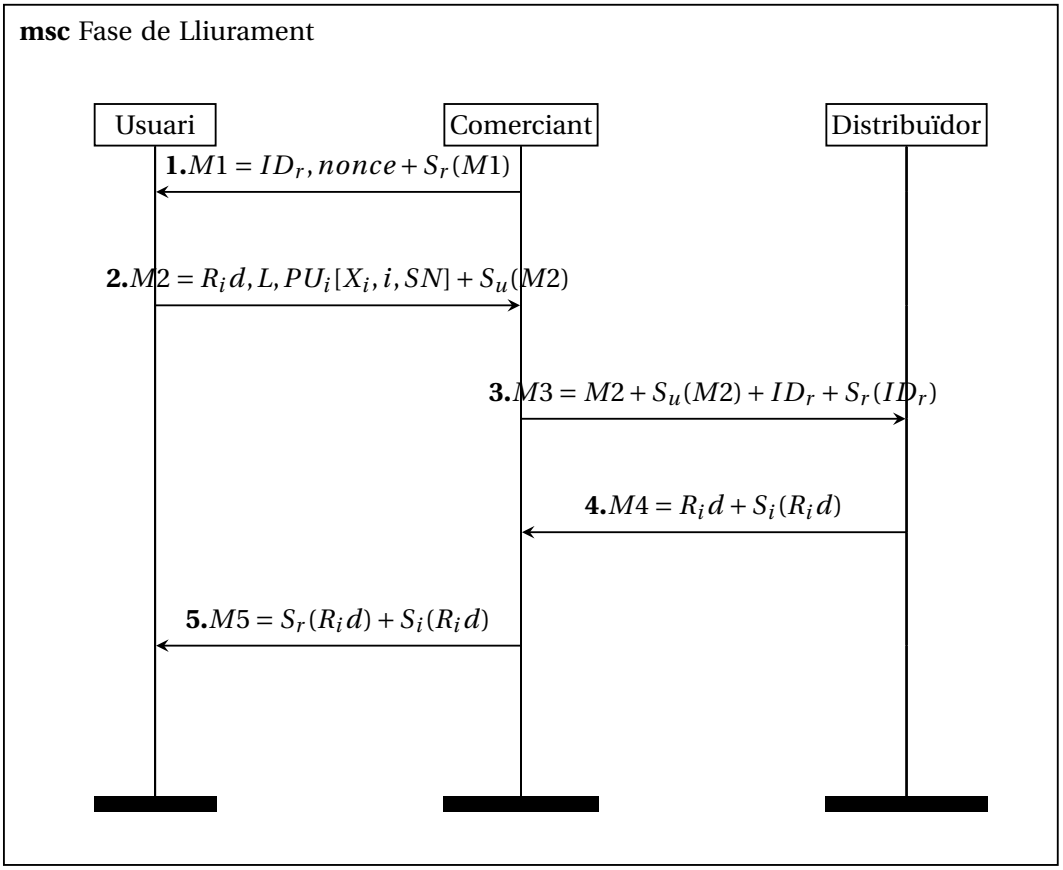

Figura 3.4: Fase de Lliurament.

Protocol: ***Fase de Lliurament***

1. R inicia la comunicació amb U indicant quin és el seu ID_r juntament amb el $nonce$. Una vegada generats els dos valors, es firmen i també s'envien a U.

2. U rep la firma digital de R i comprova que efectivament és R qui envia el missatge. Una vegada verificada la firma digital amb èxit, U haurà de dur a terme les següents passes:

   a) Primer generar el valor $R_i d$ que consta de la funció de *hash* sobre els valors X_0, X_i i ID_r.

   b) A continuació es genera L, que es calcularà amb la funció de *hash* sobre ID_r i el $nonce$.

c) Per finalitzar, el missatge que envia U a R, per continuar amb l'entrega, seran els valors $R_i d$, L i els valors X_i, i i SN xifrats amb la clau pública de I perquè sols ho pugui desxifrar I, juntament amb la firma de U sobre el missatge sencer perquè R pugui comprovar que efectivament U és qui ha enviat el missatge.

3. Tot seguit, R comprova la firma de U del missatge rebut. Una vegada comprovada satisfactòriament, amb el $nonce$ que havia generat prèviament, genera L i comprova que es correspon amb l'etiqueta rebuda. Una vegada acabades les dues comprovacions correctament, simplement reenvia el missatge rebut de U sencer amb la firma de tot el missatge a I.

4. I valida les firmes de R i de U, i una vegada validades segueix amb les següents passes:

   a) Desxifra amb la seva clau privada els valors X_i,i i SN que ha xifrat U.

   b) Comprova que X_i no hagi estat entregat abans.

   c) Genera R_{id} i comprova que sigui igual al que ha rebut.

   d) Una vegada finalitzades les comprovacions amb èxit, I genera la firma sobre el valor R_{id} i l'envia juntament amb el R_{id} generat anteriorment a R.

5. Una vegada R rep el missatge, verifica la firma de I, comprova que els dos R_{id} generats per U i R siguin correctes. Si tot és correcte, s'envia la confirmació a U de què ha estat entregat el cupó correctament, i com a confirmació s'envien les firmes de I i R a U.

6. U verifica les firmes de I i R per completar l'entrega del cupó i així pot saber que en la pròxima entrega ha de calcular una funció de *hash* més sobre el cupó.

Figura 3.5: Intercanvi d'informació en la fase de lliurament.

Fase de Neteja

La fase de neteja defineix com seran marcats cada un dels cupons una vegada s'hagin entregat, perquè no es puguin tornar a entregar.

A continuació exposam la informació necessària per aquesta fase:

- S_r, S_i, S_f: firmes digitals, indicant quina de les parts l'ha generada.

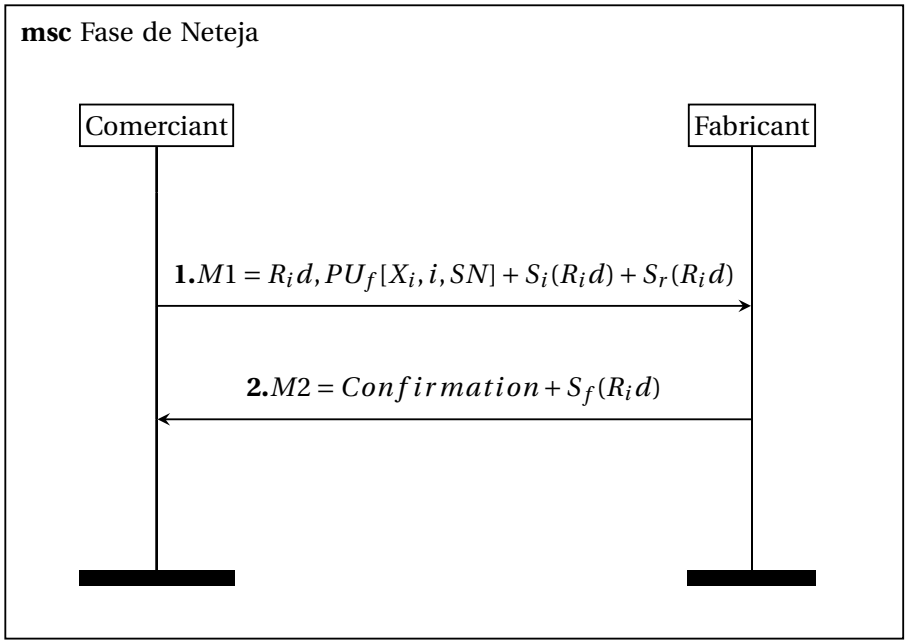

Figura 3.6: Fase de Neteja.

Protocol: *Fase de Neteja*

1. Tot seguit de què la fase de lliurament hagi finalitzat correctament, es duu a terme automàticament la fase de neteja començant pel R que prepara el missatge compost per R_{id}, i les variables X_i, i i SN xifrades amb la clau pública de F i les dues firmes digitals de R i del I, aquesta darrera enviada durant la fase de lliurament, ambdues generades sobre el valor $R_i d$.
2. Una vegada F rep el missatge, l'únic que fa és comprovar que les dues firmes digitals enviades per R siguin correctes. Una vegada comprovat que les firmes són correctes, F actualitza el valor de la cadena i envia el missatge de confirmació format per la seva firma digital sobre R_{id}.
3. Una vegada R rep la confirmació simplement verifica la firma enviada per F, per tenir en compte que s'ha marcat correctament com a cupó entregat.

Figura 3.7: Intercanvi d'informació en la Fase de Neteja.

Fase de Compartiment

Com s'ha explicat anteriorment, quan un usuari obté un cupó, aquest inclou tant la $Chain_X$ com la $Chain_Y$. En aquesta fase es defineix com s'enviarà la $Chain_Y$ al possible nou usuari.

A continuació exposam la informació necessària per aquesta fase:

- q: és el nombre de vegades que es podrà entregar un cupó compartit.
- Y: és el valor origen del qual es faran q vegades la funció de *hash* per obtenir el valor arrel Y_0.

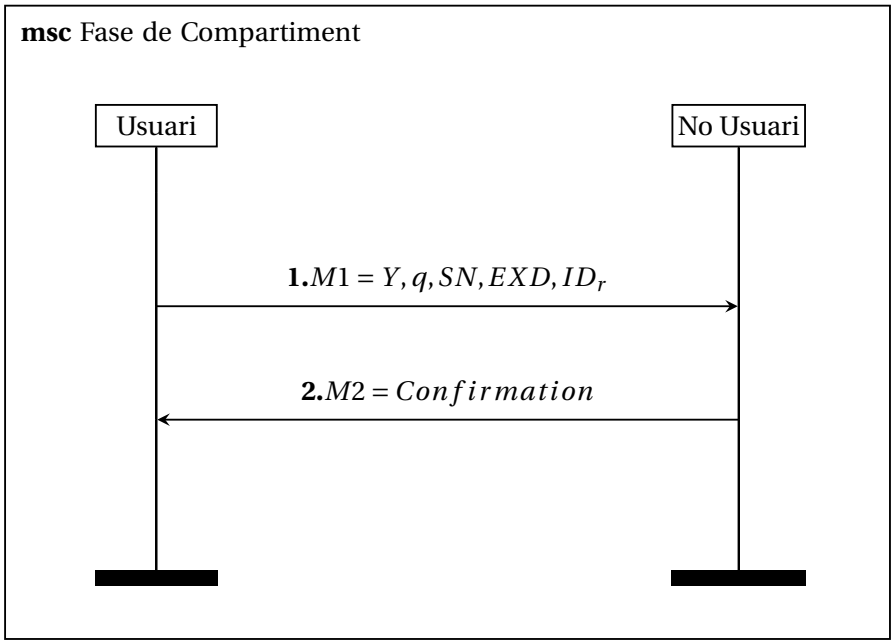

Figura 3.8: Fase de Compartiment.

Protocol: *Fase de Compartiment*

1. Quan U vol compartir un cupó amb un NU, l'únic que haurà d'enviar són els valors Y, q, EXD, SN i ID_r.
2. Una vegada NU ha rebut el missatge ja podrà obtenir la $Chain_y$, fent ús de la funció de *hash* q vegades sobre el valor Y. Una vegada NU ha rebut el cupó correctament simplement enviarà un missatge de confirmació.
3. U rep la confirmació de què el cupó s'ha compartit correctament.

Figura 3.9: Intercanvi d'informació en la fase de compartiment.

3.3.3 Fases del protocol per a Usuaris No Registrats

Dins aquest apartat definirem les dues fases que poden dur a terme usuaris que no estan registrats a la plataforma.

Fase de Lliurament per *NU*

Aquesta fase defineix com els usuaris que no estan registrats a la plataforma faran entrega dels possibles cupons que els hi hagi compartit un usuari registrat.

A continuació exposam la informació necessària per aquesta fase:

- Y_i: és l'única diferència que presenta en front a la fase de lliurament d'usuaris registrats. Y_i fa referència al cupó compartit que s'està entregant en aquest moment.

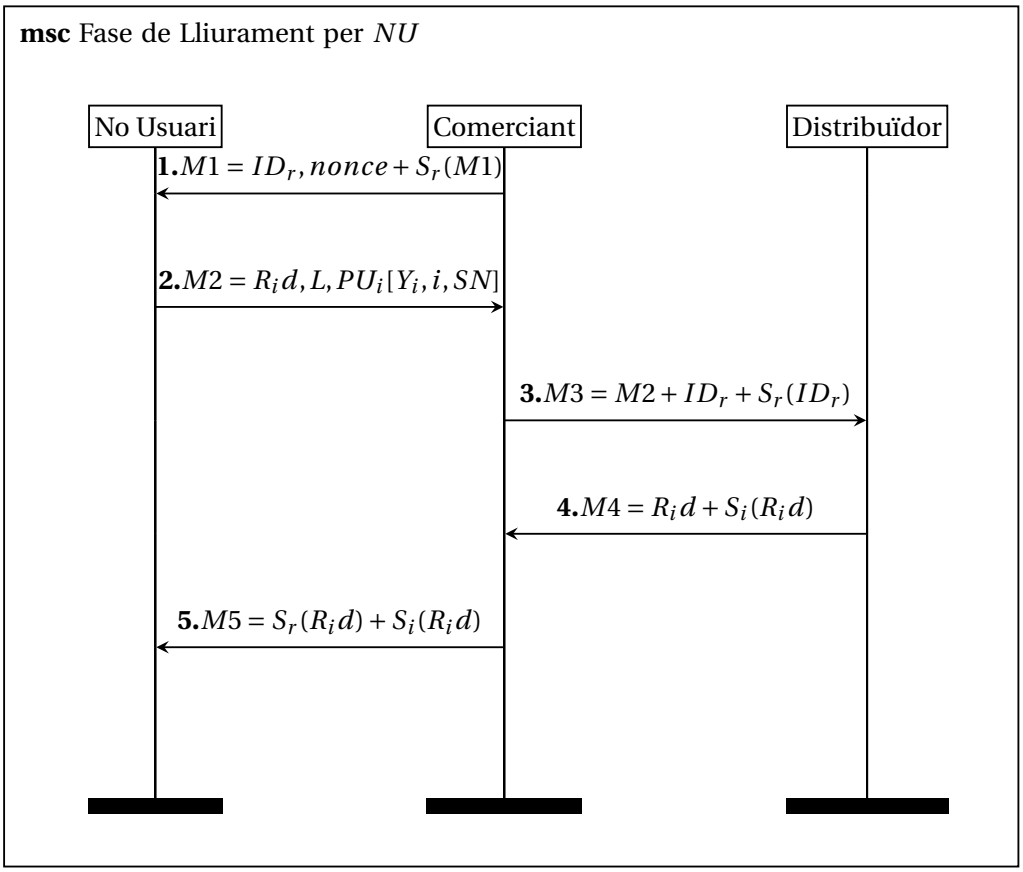

Figura 3.10: Fase de Lliurament per *NU*.

Protocol: Fase de Lliurament per NU

1. R inicia la comunicació amb NU indicant quin es el seu ID_r juntament amb el $nonce$. Una vegada generats els dos valors, es firmen i també s'envien a NU.

2. NU rep la firma digital de R i comprova que efectivament és R qui envia el missatge. Una vegada verificada la firma digital amb èxit, NU haurà de dur a terme les següents passes:

   a) Primer generar el valor R_{id} que consta de la funció de *hash* sobre els valors Y_0, Y_i i ID_r.

   b) A continuació es genera L que es calcula amb la funció de *hash* sobre ID_r i el $nonce$.

   c) Per finalitzar, el missatge que envia NU a R per continuar amb l'entrega, són els valors R_id, L i els valors Y_i, i i SN xifrats amb la clau pública de I perquè sols ho pugui desxifrar I.

3. Tot seguit, R verifica el $nonce$ que havia generat prèviament és el mateix que l'enviat per NU, genera L i comprova que es correspon amb l'etiqueta rebuda. Una vegada fetes les dues comprovacions correctament, simplement reenvia el missatge rebut de NU sencer amb la firma de tot el missatge a I.

4. I valida la firma de R, i una vegada validada, segueix amb les següents passes:

   a) Desxifra amb la seva clau privada els valors Y_i,i i SN que ha xifrat NU.

   b) Comprova que Y_i no hagi estat entregat abans.

   c) Genera R_id i comprova que sigui igual al que ha rebut.

   d) Una vegada finalitzades les comprovacions amb èxit, I genera la firma sobre el valor R_{id} i l'envia juntament amb el R_{id} generat anteriorment a R.

5. Una vegada R rep el missatge, verifica la firma de I i comprova que els dos R_id generats per NU i R siguin correctes. Si tot és correcte, s'envia la confirmació a NU de què ha estat entregat el cupó correctament. Com a mètode de confirmació s'envien les firmes de I i R a NU.

6. NU verifica les firmes de I i R per completar l'entrega del cupó i així pot saber que en la pròxima entrega ha de calcular una funció de *hash* més sobre el cupó.

Figura 3.11: Intercanvi d'informació a la fase de lliurament per *NU*.

Fase de Registre

Aquesta fase defineix quina serà la comunicació per dur a terme el registre d'un nou usuari.

A continuació exposam la informació necessària per aquesta fase:

- *Uname*: el *Username* és la variable que utilitzarà l'usuari nou per dur a terme el *log in*.
- *Pwd*: la *Password* és la contrasenya de l'usuari que utilitzarà per dur a terme el *log in*.
- PK_f: xifrat amb la clau pública del fabricant.
- S_{nu}: firma generada pel nou usuari.

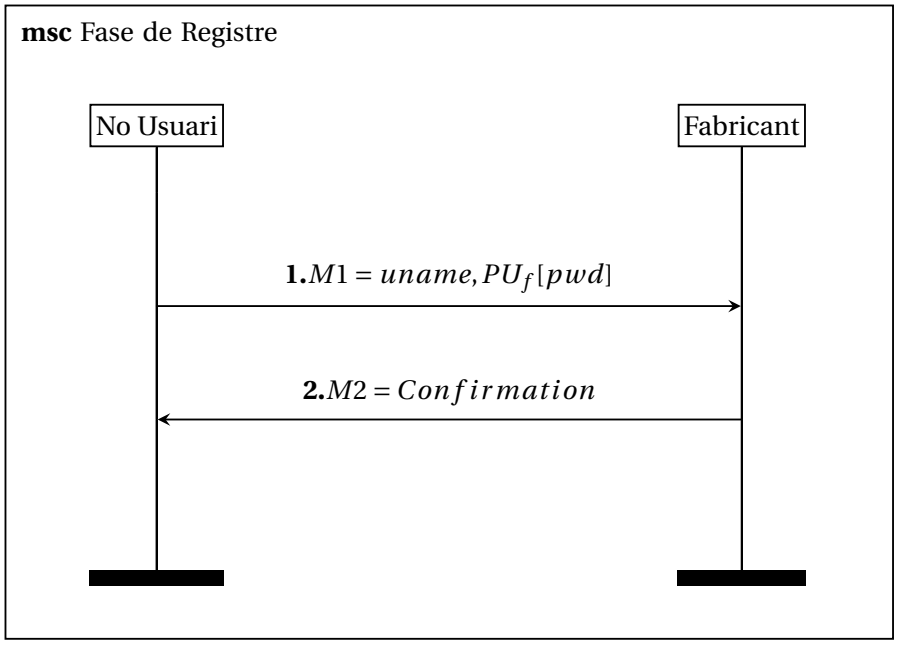

Figura 3.12: Fase de Registre

Protocol: *Fase de Registre*

1. Quan un *NU* es vol registrar l'únic que ha d'omplir és un petit formulari on sols ha d'indicar quin serà el seu nom d'usuari i la seva contrasenya. Per tant, el missatge que s'envia al *F* està compost per aquest *uname*, la *pwd* que ha d'anar xifrada amb la clau publica del *F* perquè sols la pugui desxifrar el mateix *F*.

2. *F* rep el missatge, desxifra la *pwd* per poder emmagatzemar les dades del nou usuari. Una vegada tota la informació queda emmagatzemada ja pot enviar la confirmació a *NU* de què passa a ser un usuari registrat, amb un missatge de confirmació.

3. El nou usuari registrat rep el missatge de confirmació de què ja és un nou usuari de la plataforma.

Figura 3.13: Intercanvi d'informació en la fase de registre

CAPÍTOL 4

DISSENY

En el capítol del disseny ens centrarem amb el que s'ha decidit implementar, i com s'ha fet l'estructura, les tecnologies i perquè aquestes i no unes altres. Per fer-ho d'una manera més organitzada parlarem de les diferents decisions preses separades en les dues parts de la implementació:

1. **Servidor**: consta de la part del *back-end* i la part de *front-end*, aquestes dues parts són les que s'han implementat primer per a la posterior implementació del client.

2. **Client**: una vegada tenim el servidor podem passar a la part del client per dur a terme les peticions al servidor i obtenir les diferents funcionalitats de la plataforma.

Abans de parlar de cada una de les diferents fases, primer parlarem de la seguretat que presentarà la plataforma. Ho explicarem de manera general perquè s'aplica el mateix en les dues parts.

4.1 Seguretat

4.1.1 Algoritme de *hash*

Com ja s'ha dit, durant la definició del protocol utilitzam un sistema de validació per cadenes de *hash* [8]. Aquestes cadenes no són més que un seguit de funcions de *hash*. Les funcions de *hash*, o també anomenades funcions resum, són funcions que tenen com entrada, normalment, una cadena de caràcters, i com a sortida s'obté una cadena de caràcters totalment diferent i de longitud fixa. Les funcions de *hash* presenten un conjunt de propietats interessants per al nostre protocol, que són les següents:

- **Baix cost computacional**: calcular la funció de *hash* requereix de poc cost computacional.

- **Compressió**: una funció de *hash* comprimeix les dades a una longitud fixa.

- **Uniforme**: per una funció de *hash* sobre un valor X, el resultat de la funció sempre serà el mateix, sempre i quan es faci damunt X i amb el mateix algoritme. Un petit canvi suposa un canvi total en el resultat obtingut.

- **Unidireccional**: a partir d'un resultat d'una funció de *hash* no ha de ser possible saber el valor de l'entrada.

Per implementar aquest tipus de funcions necessitam elegir algun dels algoritmes disponibles. Dues de les més importants són les següents:

- ***MD5*** [9]: *Message-Digest Algorithm* és un algoritme de resum criptogràfic de 128 *bits*. El problema d'aquest algoritme és la seguretat que presenta, ja que s'ha comprovat que és vulnerable a atacs de força bruta perquè la longitud de 128 *bits* és massa petita.

- ***SHA-256*** [10]: *Secure Hash Algorithm*. La diferència principal sobre *MD5* és la longitud de sortida, que és de 256 bits en front als 128 bits que presenta *MD5*. Per tant, el problema de seguretat que presenta *MD5* no existeix a *SHA-256*.

Donat el problema de seguretat que presenta l'algoritme *MD5*, per al projecte s'ha elegit *SHA-256*. D'aquesta manera podrem aconseguir un dels requeriments de seguretat que és la integritat de les dades, ja que si dos resultats d'una funció de *hash* donen resultats distints quan en teoria era la mateixa informació, vol dir que la informació rebuda ha estat modificada.

4.1.2 Sistema de xifratge

Durant la definició del protocol s'ha dit que utilitzam un algoritme de clau pública [11]. Aquest consta de dues claus per protegir els missatges. Una de les claus, com el seu propi nom indica, es pública disponible per a tothom, i una altra és privada i el propietari l'ha de guardar de manera que ningú tengui accés a ella.

S'utilitza un algoritme d'aquestes característiques perquè ens proporciona confidencialitat, ja que si xifram un missatge amb la clau pública de x ens asseguram que sols x podrà desxifrar el missatge amb la seva clau privada. Per altra banda, també assolim el no repudi en origen, ja que si x xifra el missatge amb la seva clau privada, el destinatari sols podrà desxifrar el missatge amb la clau pública de x, i per tant, el destinatari podrà estar segur de què el missatge ha estat enviat per x.

Hi ha diversos algoritmes de clau pública, com per exemple:

- ***ElGamal*** [12]: és un algoritme de clau asimètrica basat en el problema matemàtic del logaritme discret. Aquest algoritme tant es pot utilitzar per generar firmes digitals com per dur a terme el xifrat i desxifrat.

- **Rivest, Shammir and Adleman (RSA)** [13]: és el sistema criptogràfic de clau pública més utilitzat i és vàlid per xifrar, desxifrar i per firmar digitalment. La seguretat d'aquest algoritme es basa en el problema de la factorització del producte de dos nombres primers grans elegits a l'atzar.
- **Digital Signature Algorithm (DSA)** [14]: és un estàndard del Govern Federal dels Estats Units per dur a terme firmes digitals. Aquest no serveix per xifrar i desxifrar informació, sols per firmar. Aquest algoritme requereix de molt més temps de còmput que no altres algoritmes com RSA.

D'aquests algoritmes que s'han plantejat s'ha optat per RSA ja que és un dels més robusts, té una comunitat molt gran i té l'avantatge de presentar una gran quantitat de llibreries escrites amb el llenguatge *java*.

El sistema de distribució de claus elegit és el de certificats seguint l'estàndard *X.509*. Aquest estàndard segueix una ruta de validació composta per una *Certificate Authority (CA)* que és una entitat que emet certificats digitals per l'ús de tercers. Els certificats de clau pública tenen el següent contingut [15]:

- Certificat
  - Versió.
  - Número de sèrie.
  - *ID* del algoritme utilitzat per la CA per firmar (normalment RSA o DSA).
  - Emissor (CA).
  - Vàlid fins:
    - No abans de.
    - No després de.
  - Subjecte, expressat amb notació DN (*Distinguished Name*), compost per:
    - CN(*Common Name*).
    - OU(*Organizational Unit*).
    - O (*Organization*).
    - C (*Country*).

    El subjecte tant pot ser una persona, un servidor o fins i tot un servei.
  - Informació de clau pública del subjecte:
    - Algoritme de clau pública utilitzat.
    - Clau pública del subjecte.
  - Identificador únic de l'emissor (opcional).
  - Identificador únic del subjecte (opcional).
  - Extensions (opcionals).
- Algoritme utilitzat per firmar el certificat.
- Firma digital del certificat.

El programari utilitzat per crear aquests certificats és *OpenSSL*, que és un programari lliure que consisteix en un paquet d'eines relacionades amb la criptografia que inclou una funció específica per a la generació de certificats RSA.

4.2 Servidor

La primera part que s'ha d'implementar és el servidor. A continuació es mostrarà quin és el disseny d'aquest, així com les tecnologies que s'han decidit utilitzar per dur a terme les diferents funcionalitats.

4.2.1 Disseny de la implementació

El següent diagrama ens mostra on s'ubicarà la part del servidor en l'estructura general del projecte.

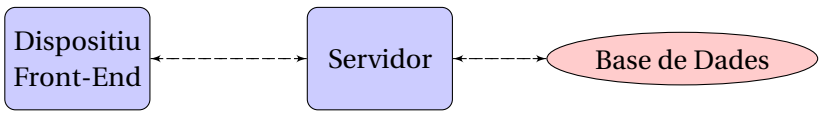

Figura 4.1: Estructura general.

Com podem veure dins l'estructura sols veim un servidor, però n'hi hauria d'haver un per cada entitat. El que farem és emular totes les entitats dins un sol servidor, de manera que la part lògica sigui fàcil de separar per una posterior implementació en el món real. El que tendrem és un dispositiu *front-end*, com podria ser una plana *web*, que durà a terme una sèrie de peticions i la informació serà processada pel servidor. Per altra banda, tendrem la base de dades encarregada de l'emmagatzematge de la informació ja processada.

4.2.2 Tecnologies utilitzades

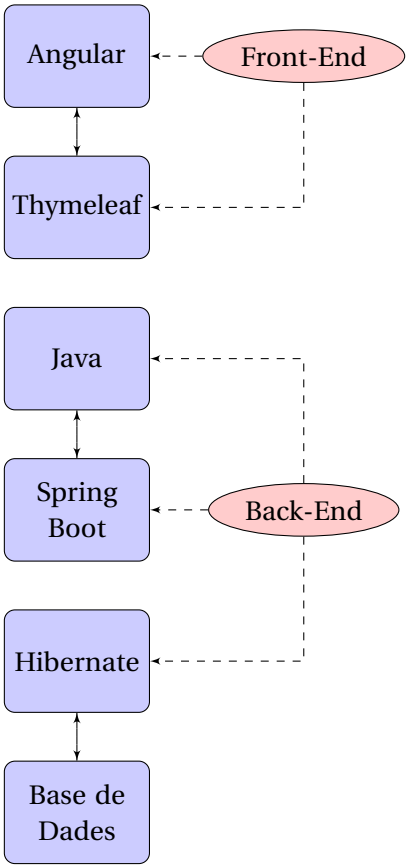

Figura 4.2: Estructura de les tecnologies en el servidor.

A la figura 4.2 s'indiquen les tecnologies que s'utilitzaran en el servidor, tenint en compte que la part de *Front-End* fa referència al dispositiu *front-end* esmentat en l'apartat anterior i el *Back-End* fa referència a la part del servidor. Per explicar-ho d'una forma més clara, es separarà l'explicació en els dos apartats de *Front-End* i *Back-End*, i en cada un d'ells s'explicarà en més detall cada una de les tecnologies utilitzades i el perquè d'elles.

Back-End

Dins l'apartat de *Back-End* tenim els tres elements principals següents:

- **Java** [16]: és un llenguatge de programació d'alt nivell molt utilitzat, que ofereix concurrència i és orientat a objectes. *Java* és un llenguatge interpretat, és a dir, necessita un programa intèrpret per a ser executat, i per això pot semblar lent en comparació amb altres llenguatges, però ofereix un índex de re-utilització de codi molt elevat. És un llenguatge flexible i potent, a part de ser un llenguatge multi-plataforma. Les característiques principals d'aquest llenguatge són:

1. **Senzill**: s'ha creat perquè sigui un llenguatge senzill amb una sintaxi elegant. Únicament consta de tres tipus de dades primàries, eliminant els punters i l'herència múltiple.
2. **Orientat a objectes**: segueix els paradigmes de la programació orientada a objectes, ja que la programació amb *java* es centra en la manipulació, creació i construcció d'objectes.
3. **Distribuït**: permet la construcció d'aplicacions distribuïdes per mitjà d'una col·lecció específica de classes.
4. **Robust**: és un llenguatge robust i fiable; s'ha escrit pensant en poder verificar errors i està molt tipificat.
5. **Segur**: té pocs problemes de seguretat, característica molt important en les aplicacions distribuïdes d'Internet.
6. **Arquitectura neutral**: és independent de la plataforma final on s'executarà el programa.
7. **Portable**: és un llenguatge d'alt nivell i de plataforma independent, i això li dóna portabilitat.
8. **Alt rendiment**: els compiladors *java* han anat millorant les seves prestacions.
9. **Concurrent**: *java* permet l'execució de múltiples fils d'execució, o diverses tasques de forma simultània.
10. **Dinàmic**: en temps d'execució, l'entorn *java* es pot ampliar mitjançant enllaços a classes que poden estar localitzades en servidors remots o en xarxa.

- ***Spring Framework*** [17]: per poder explicar que és *spring boot*, que és el que veim al diagrama, primer haurem d'explicar el *spring framework* ja que *spring boot* sols és un dels mòduls que ofereix aquest *framework*. *Spring* és un *framework* basat en tecnologies estàndard *JavaEE*, i es caracteritza principalment per la injecció de dependències o l'ús d'objectes convencionals *POJOs, Plain Old Java Objects*, com objectes de negoci. Aquestes idees van permetre un desenvolupament senzill, ràpid i unes aplicacions més lleugeres.

  Per concretar, *spring* ens proporciona tres elements principals:

  1. **Serveis *enterprise***: podem crear de manera senzilla un objecte transaccional, que el seu accés estigui restringit, o que sigui accessible de manera remota i transparent per al desenvolupador, etc. Sense haver d'escriure el codi manualment, en molts de casos sols és necessari etiquetar l'objecte.
  2. **Estereotips configurables**: als objectes de la nostra aplicació es poden etiquetar les classes indicant si pertanyen a la capa d'accés a les dades o a la capa de negoci. Es diu que són configurables perquè podem definir els estereotips com desitgem, per exemple, *@Service* indica que és un objecte

que pertany a la capa de negoci, i per altra banda, *@Repository* indica que és un objecte *DAO*, un objecte que pertany a la capa d'accés a les dades.

3. **Injecció de dependències**: aquesta característica ens permet que un objecte que ens proporciona un servei, pugui accedir a un objecte client que aquest necessita d'una forma senzilla; per exemple, que un objecte de la capa de presentació es pugui comunicar amb un de la capa de negoci.

Dins el *framework* de *spring* s'han anat creant diferents mòduls per donar forma a aquest *framework*, per exemple , *Spring Data*, *Spring Security*, etc.

En aquest projecte s'han utilitzat alguns d'aquests mòduls. Un d'ells és *spring boot* que ens ajuda a compilar el projecte utilitzant els diferents mòduls de *spring* d'una manera àgil i evitar al màxim la configuració.

El mòdul de *spring boot* ens presenta una sèrie de característiques interessants per al projecte:

1. **Evitar la configuració**: en lloc d'estar escrivint la configuració necessària i haver de validar si és correcte, *spring boot* proporciona les configuracions necessàries per defecte en molts d'escenaris.
2. **Gestió de dependències**: com s'ha dit abans, el *framework spring* ens proporciona injecció de dependències, però en ocasions sorgeixen problemes amb el tema de les versions d'aquestes o les dependències transitives. Per tant, *spring boot* fa un anàlisi d'aquestes dependències i les posa en ordre per evitar la duplicitat i compila els diferents mòduls *spring* amb el menor nombre de dependències possibles.

- ***Hibernate*** [18]: és una de les dependències que es pot implementar fàcilment en *spring boot*, és una solució, de codi obert, implementada per al mapeig d'objectes relacionals *Object Relational Mapping (ORM)* en aplicacions *java* sobre una base de dades relacional. El propòsit principal és alliberar al programador d'un seguit de tasques pròpies de la persistència de dades relacional.

  La funcionalitat bàsica de *Hibernate* és la del mapeig de classes *java* en taules de la base de dades. *Hibernate* també proporciona un llenguatge de consulta (*Hibernate Query Language, HQL*) i proporciona facilitats per a la recuperació de dades. Per fer-ho, genera les peticions *SQL* i delega al desenvolupador la gestió manual del resultat de la consulta.

En definitiva, *spring* és un *framework* programat en *java* que, entre altres coses, ens facilitarà la implementació d'aplicacions *web* basades en l'arquitectura *Model-View-Controller (MVC)* i serveis *RESTful*. Aquests serveis es caracteritzen per una arquitectura client-servidor i estan dissenyats per utilitzar un protocol de comunicació *stateless* com *HyperText Transfer Protocol (HTTP)*.

Per tant, la combinació triada pel projecte, és el llenguatge de programació *java* juntament amb el *framework spring*. De totes formes, abans de triar aquesta combinació es van plantejar altres *frameworks*, així com altres llenguatges de programació, que possiblement també haguessin servit per desenvolupar la tasca, com són els següents:

- ***Ruby on Rails*** [19]: és un *framework* d'aplicacions web de codi obert, escrit amb el llenguatge de programació *ruby*, i també segueix molt bé l'arquitectura MVC. Aquest es basa en dos principis fonamentals que són la no repetició de codi redundant i la configuració automàtica del sistema.
- ***Django*** [20]: és un *framework* per al desenvolupament d'aplicacions *web*, és de codi obert i escrit en el llenguatge de programació *python*. També respecta molt bé l'arquitectura MVC. Com a principis fonamentals contempla la no repetició de codi, el desenvolupament ràpid de les aplicacions i l'escalabilitat.
- ***Meteor*** [21]: és un *framework* també de desenvolupament d'aplicacions *web*, escrit amb *JavaScript* i de codi obert. Aquest s'integra molt bé amb bases de dades *No SQL*.

Finalment, s'ha triat *spring* perquè en general proporciona els mateixos avantatges que presenten altres *frameworks*, amb un avantatge afegit que és que està escrit amb el llenguatge de programació *java*, i aquest és amb el que esteim més familiaritzats.

Per altra banda, també és important esmentar quina ha estat l'eina d'automatització del procés *build*. Aquest procés bàsicament consta de la compilació, proves o *tests*, empaquetament, etc. L'eina que s'ha triat és *Gradle* [22], que comparteix els principals avantatges que proporcionen les altres eines com són *Maven* i *Ant*. Però apart, ens proporciona altres avantatges afegits, que són:

1. Flexibilitat.
2. Pot fer feina damunt més d'un llenguatge, no sols *java*.
3. Utilitza DSL (*Domain-specific language*), llenguatge senzill i clar.
4. Realitza gestió de dependències.
5. Utilitza *Groovy*, que és un llenguatge de programació orientat a objectes que utilitza la plataforma *java*, i bàsicament constitueix una alternativa al llenguatge *java* per executar-se en la màquina virtual de *java*.

Com hem dit s'ha triat *Gradle*, a part dels seus avantatges, també pels inconvenients que presenta l'altra eina plantejada *Maven*:

- Estructura molt rígida.
- Fitxer de compilació amb molt de codi.
- Utilitza el format *XML* per a la configuració.

Base de Dades

Per la base de dades es va optar per *HSQLDB* [23] (*Hyper SQL Database*) que és una base de dades feta totalment amb el llenguatge de programació *java*. Aquesta presenta unes característiques interessants per al projecte:

- Senzilla.
- Ràpida.
- Implementada amb *java*.
- Permet executar-se en memòria.
- Utilitza el llenguatge *Structured Query Language (SQL)*.

Que permeti executar-se en memòria és un gran avantatge ja que no hem d'instal·lar cap programari addicional, i a més, ens permet compartir l'arxiu de la base de dades fàcilment i d'aquesta manera fer còpies de seguretat de manera senzilla.

Utilitza el llenguatge *SQL*, i per tant, una vegada es vulgui moure la base de dades a un sistema més estable es podrà fer d'una manera senzilla i ràpida. *SQL* [24] és un llenguatge específic que dóna accés a un sistema gestor de base de dades relacionals. És un llenguatge molt utilitzat i amb una comunitat gran. Aquest llenguatge requereix de taules fixes d'on extreure la informació a través de diferents consultes.

A part del llenguatge *SQL* també es va plantejar una altra opció interessant, que és la següent:

- ***NoSQL***: també és un sistema gestor de base de dades, però en aquest cas les dades no requereixen d'estructures fixes com el llenguatge *SQL*. Les taules normalment no soporten operacions de *JOIN* molt utilitzades al llenguatge *SQL* per relacionar informació. Aquest sistema durant els darrers anys ha anat agafant força però encara enfora de ser tant utilitzat com el llenguatge SQL.

Ens hem decantat pel llenguatge SQL ja que també és el llenguatge en el qual tenim més experiència i ens proporciona totes les característiques necessàries per dur a terme el projecte. A part, és més fàcil la implementació d'un sistema gestor de bases de dades amb el llenguatge *SQL* que no un sistema basat en el llenguatge *No SQL*.

Com s'ha comentat anteriorment, per dur a terme l'accés a la base de dades es farà a través de la dependència *Hibernate*.

A continuació veurem el diagrama entitat-relació de la base de dades, és a dir, com es relacionen les diferents entitats.

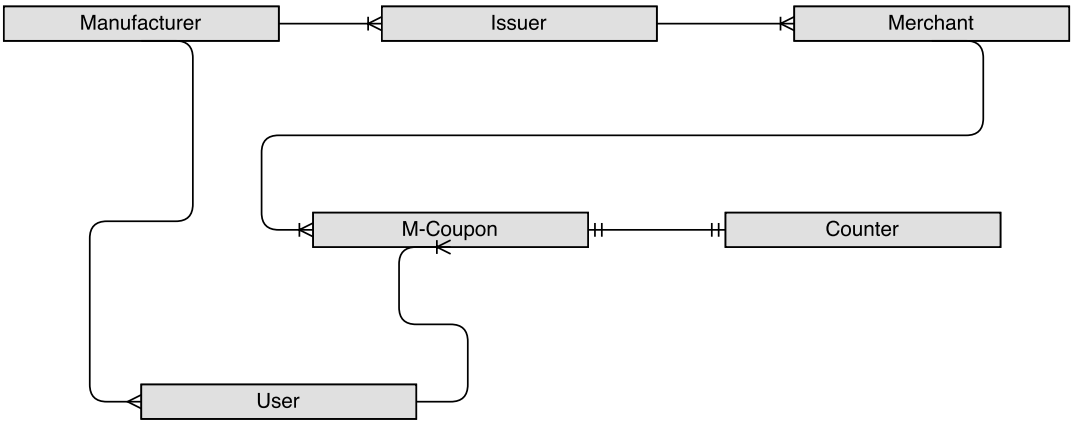

Figura 4.3: Disseny de la Base de dades en el servidor.

Cada una de les sis taules de la figura 4.3 emmagatzema la següent informació:

- ***Manufacturer* o Fabricant**: conté informació sobre els diferents fabricants, com per exemple, els usuaris que estan registrats a la plataforma i els distribuïdors que té associats per a la distribució dels cupons.
- ***Issuer* o Distribuïdor**: conté informació sobre els diferents distribuïdor. En aquesta taula podem trobar, entre altres coses, el fabricant al qual està associat el distribuïdor i els diferents comerciants associats al distribuïdor.
- ***Merchant* o Comerciant**: conté informació sobre els diferents comerciants. Aquesta taula conté el distribuïdor al qual està associat el comerciant.
- ***M-Coupon* o Cupó**: conté informació sobre les diferents característiques dels cupons, com per exemple, el comerciant al qual s'ha d'entregar i a quin usuari pertany el cupó.
- ***Counter* o Comptador**: aquesta taula és l'encarregada de guardar el nombre de vegades que s'ha entregat un cupó, i per tant és com una característica més del cupó.
- ***User* o Usuari**: conté informació relacionada amb els usuaris que s'han registrat a la plataforma.

Front-End

Dins l'apartat de *Front-End* tenim dos elements principals:

- ***Angular*** [25]: és un *framework Javascript* i de codi obert, creat per *Google*, i destinat a la creació d'aplicacions *web* modernes del tipus *Single Page Application (SPA)*. *SPA* és una solució per oferir pàgines *web* d'una manera més immediata, ràpida i fluida. Ho fa intentant dur cada vegada més característiques al navegador, és a dir, al lloc del client, envers de dependre tant del servidor.

- ***Thymeleaf*** [26]: és un sistema de plantilles escrit amb *java* que s'adapta molt bé al MVC. Juntament amb *spring* es pot utilitzar per reemplaçar completament els arxius *Java Server Pages (JSP)*. L'objectiu principal és permetre la creació de plantilles d'una forma elegant i amb un codi ben formatat, perquè després es puguin visualitzar correctament als diferents navegadors.

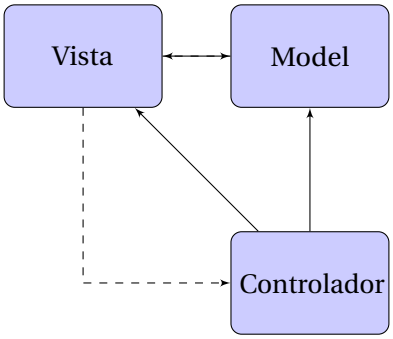

Figura 4.4: Model-Vista-Controlador.

En definitiva el que més ens interessa es poder implementar de manera eficient i senzilla una arquitectura de MVC [27]. Aquesta arquitectura de programari està composta per tres components principals diferents. Per una banda defineix els components per la representació de la informació i per una altra la interacció de l'usuari. Aquesta arquitectura es basa en idees de reutilització de codi i la separació de conceptes, característiques que permeten facilitar la creació i el manteniment de les aplicacions. Els tres components són:

- **Model**: és la representació de la informació amb la qual el sistema opera, i gestiona tots els accessos a la informació, com consultes o actualitzacions. El model envia a la vista la part de la informació que ha de ser mostrada i les peticions d'accés o manipulació d'informació arriben al model a través del controlador.

- **Vista**: presenta el model en un format adequat per interactuar. Requereix d'un model per saber com ha de representar la informació.

- **Controlador**: aquest respon als esdeveniments i invoca peticions al model quan es fa una sol·licitud d'informació. També pot enviar comandes a la seva vista associada si es sol·licita un canvi de forma en què es representa el model. Es pot dir que és l'intermediari entre la vista i el model.

4.3 Client

Una vegada tenim el disseny del servidor, explicarem quin serà el disseny del client. El client serà qui farà les peticions al servidor.

4.3.1 Disseny de la implementació

A la pròxima figura podem veure el disseny que serà implementat al client.

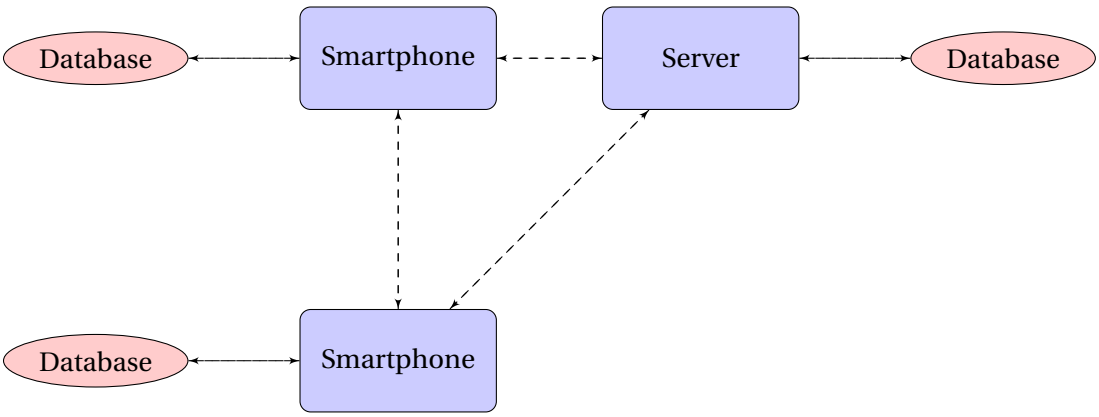

Figura 4.5: Diagrama del client.

A diferència del diagrama del servidor, podem veure com apareix una base de dades nova, que és a la que accedirà l'aplicació del client. A més, apareix la comunicació entre els diferents dispositius mòbils, bàsicament per dur a terme la compartició de cupons.

Per altra banda, tenim la connexió amb el servidor que es farà a través d'una *Application Programming Interface (API)* on hi trobarà les funcions necessàries per dur a terme l'entrega i l'obtenció de cupons. S'ha d'indicar que aquestes funcions seran executades mitjançant comunicacions *HTTP*. Una *API* és defineix com un subconjunt de subrutines, funcions i procediments que ofereixen certa informació; funciona com una capa d'abstracció.

4.3.2 Tecnologies utilitzades

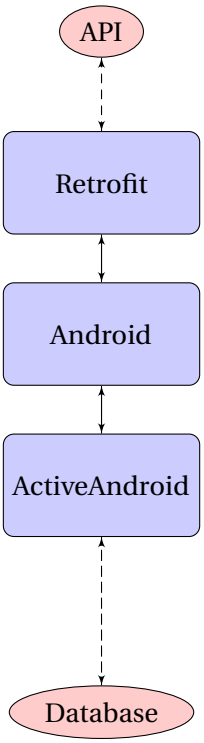

Figura 4.6: Estructura de les tecnologies en el client.

Primer de tot, s'ha de prendre la decisió sobre quin sistema operatiu s'establirà l'aplicació del client. Les opcions que es van plantejar són les següents:

- ***Android*** [28]: és un sistema operatiu basat en el nucli *Linux*, dissenyat específicament per a dispositius mòbils amb pantalla tàctil. És un sistema operatiu de codi obert i té una gran quota de mercat.

- ***iOS*** [29]: és un sistema operatiu, també dissenyat específicament per a dispositius mòbils. Presenta el problema de què és programari propietari i en conseqüència no té tanta quota de mercat.

- ***Windows Phone*** [30]: és un sistema operatiu dissenyat específicament per a dispositius mòbils. També presenta el problema de que és programari propietari. Aquest pràcticament queda descartat ja que la seva quota de mercat és molt baixa.

El sistema operatiu elegit és *android*, ja que com hem dit abans té una gran quota de mercat pel que fa als dispositius mòbils. Un dels avantatges que també ens va dur a prendre aquesta decisió és que, entre altres, utilitza el llenguatge de programació *java*, que ens interessa per tota la interoperabilitat de la plataforma.

Una de les eleccions que no s'ha descartat és *iOS*, però per desenvolupar una aplicació amb aquest sistema operatiu és necessari el coneixement d'altres llenguatges de

programació com *Objective-C*.

Com s'ha comentat abans, les connexions es faran utilitzant connexions HTTP. Així, per dur a terme fàcilment aquestes connexions utilitzam la llibreria *Retrofit* [31], que ens permet convertir una API en una interfície *java*. A més, ens permet obtenir respostes JavaScript Object Notation (JSON) i convertir-les en objectes *java* automàticament.

Un altre aspecte a contemplar és com farem la compartició de cupons, sobretot quina tecnologia de comunicació s'utilitzarà per transferir aquest cupó. En aquest cas s'han contemplat diferents tecnologies com per exemple les parlades en capítols anteriors. Entre les diferents tecnologies, s'ha triat fer-ho amb la tecnologia *bluetooth*. Aquesta tecnologia és molt utilitzada i es troben llibreries fàcilment per implementar una comunicació senzilla.

Base de Dades

Per la base de dades es va optar per la solució *SQLite* [32]. Aquest és un sistema de gestió de bases de dades relacional que permet obtenir Atomicity, Consistency, Isolation and Durability (ACID), una petita llibreria escrita amb el llenguatge de programació *C*. Presenta les següents característiques:

- **Atomicitat**: ens garanteix que les transaccions són completes, és a dir, si una operació consta de més d'un pas es faran tots els passos o no se'n farà cap.
- **Consistència** (*Integritat*): ens assegura que sols el que es comença es pot acabar, és a dir, qualsevol transacció durà a la base de dades a un estat vàlid.
- **Aïllament**: aquesta propietat ens assegura que una operació no ens pot afectar a altres. Per exemple, que dues transaccions sobre una mateixa informació siguin independents i no generin cap tipus d'error.
- **Durabilitat** (*Persistència*): aquesta propietat ens assegura que una vegada realitzada l'operació, aquesta persistirà i no es podrà desfer, fins i tot, si falla el sistema.

A part de complir les característiques esmentades, permet crear una base de dades que s'executa dins la mateixa aplicació i fins i tot implementa la major part del llenguatge *SQL*. Tot i així, s'ha de triar de quina manera es faran els accessos a la base de dades, ja que si hem de programar totes les consultes *SQL* resulta molt tediós. Per als accessos a la base de dades utilitzarem la llibreria *Active Android* [33], que és una llibreria que actua com a *ORM*, que ens permet fer consultes a la base de dades d'una manera molt senzilla i eliminant gran quantitat d'errors.

En la següent figura podem veure el disseny de la base de dades i com es relacionen les diferents entitats.

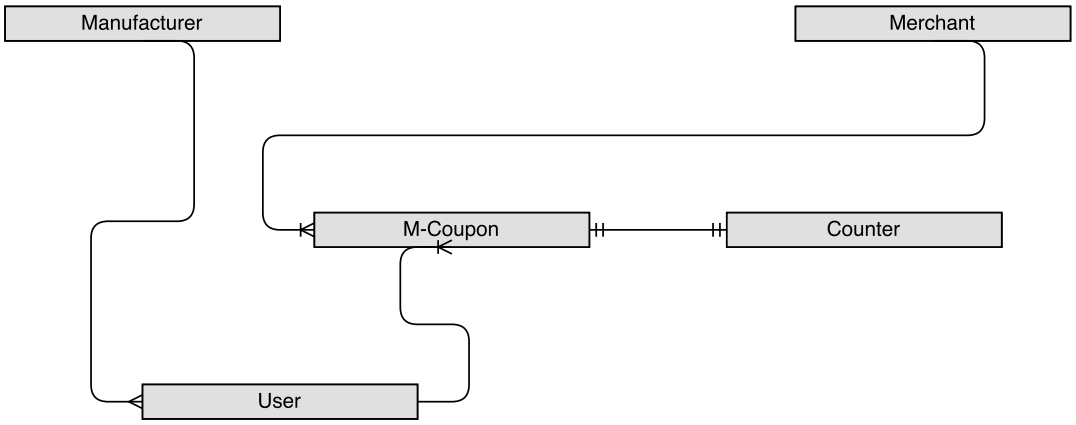

Figura 4.7: Disseny Base de dades del client

Com podem veure, a diferència de la base de dades del servidor, sols manca una de les taules, el distribuïdor, ja que pel que respecta a l'usuari no és necessari saber qui és el distribuïdor ni cap informació relacionada amb ell. Per la resta, la base de dades és pràcticament idèntica, i les taules ens serveixen per guardar el mateix tipus d'informació, però amb menys detall.

CAPÍTOL 5

IMPLEMENTACIÓ

Una vegada tenim el disseny definit, s'ha de passar a la implementació de tot l'esmentat al capítol anterior. Per fer-ho d'una manera estructurada s'explicarà de la mateixa manera en què s'ha separat el disseny, començarem amb la seguretat, posteriorment explicarem el servidor i acabarem amb la implementació del client.

5.1 Seguretat

Per implementar els requeriments necessaris en el protocol s'ha utilitzat una de les llibreries *java* que s'han esmentat anteriorment. S'han hagut d'utilitzar metodologies de control de versions, per tenir un control sobre el codi implementat.

5.1.1 Package Crypto

La llibreria encarregada de proporcionar la seguretat al protocol, és una llibreria escrita amb el llenguatge de programació *java* que conté les funcions necessàries per dur a terme la implementació dels requeriments de seguretat. Aquesta llibreria consta de dues classes principals:

- ***Cryptography***: és la classe que conté totes les funcions criptogràfiques necessàries.
- ***Base64***: és la classe que s'encarrega de fer la codificació en base 64.

A continuació es fa un recull de les funcions més utilitzades d'aquesta llibreria:

```
public void initPrivateKey(String filename) {
privateKey = readPrivateKey(filename);
}

private PrivateKey readPrivateKey(String filename){
PKCS8EncodedKeySpec keySpec =
new PKCS8EncodedKeySpec(readFileBytes(filename));
KeyFactory keyFactory = KeyFactory.getInstance("RSA");
return keyFactory.generatePrivate(keySpec);
}
```

El conjunt de funcions anterior ens permet dur a terme la inicialització de la clau privada, també en tenim un per la clau pública.

```
public String encryptWithPublicKey(String toEncrypt) {
byte[] secret = "".getBytes();
byte[] message = toEncrypt.getBytes("UTF8");
secret = encrypt(publicKey, message);
return new String(Base64.encodeToString(secret,Base64.DEFAULT));
}

private byte[] encrypt(PublicKey key, byte[] plaintext){

Cipher cipher =
Cipher.getInstance("RSA/ECB/OAEPWithSHA1AndMGF1Padding");

cipher.init(Cipher.ENCRYPT_MODE, key);
return cipher.doFinal(plaintext);
}
```

Una vegada tenim inicialitzada una de les claus, tan sigui privada com sigui pública, tenim les següents funcions per dur a terme el xifrat de la informació. Com podem veure s'utilitza l'algoritme *RSA*, amb el mode d'operació de xifratge per blocs *ECB* i amb el mètode de *padding OAEP*.

```
public String decryptWithPrivateKey(String toDecrypt) {
byte[] secret = Base64.decode(toDecrypt,Base64.DEFAULT);
byte[] recovered_message = "".getBytes();
String message = "";
recovered_message = decrypt(privateKey, secret);
message = new String(recovered_message, "UTF8");
return message;
}

private byte[] decrypt(PrivateKey key, byte[] ciphertext){

Cipher cipher =
Cipher.getInstance("RSA/ECB/OAEPWithSHA1AndMGF1Padding");

cipher.init(Cipher.DECRYPT_MODE, key);
return cipher.doFinal(ciphertext);
}
```

Així com tenim un conjunt de funcions per xifrar, també en tenim un per desxifrar.

```
public String getSignature(String toSign) {
String signature = "";
signature = sign(toSign, privateKey);
return signature;
}

private String sign(String plainText, PrivateKey privateKey) {
Signature privateSignature =
Signature.getInstance("SHA256withRSA");
privateSignature.initSign(privateKey);
privateSignature.update(plainText.getBytes("UTF-8"));
byte[] signature = privateSignature.sign();
return
new String(Base64.encodeToString(signature,Base64.DEFAULT));
}
```

Les dues funcions que presentam són les encarregades de dur a terme les firmes digitals. La clau inicialitzada per dur a terme la firma ha de ser la clau privada. Com hem explicat en el capítol anterior, per dur a terme la funció de *hash* utilitzam l'algoritme *SHA-256*.

```
public boolean getValidation(String toVerify,String signature) {
boolean validated = false;
validated = verify(toVerify, signature, publicKey);
return validated;
}

verify(String plainText,String signature,PublicKey publicKey){
Signature publicSignature =
Signature.getInstance("SHA256withRSA");
publicSignature.initVerify(publicKey);
publicSignature.update(plainText.getBytes("UTF-8"));

byte[] signatureBytes =
Base64.decode(signature.getBytes(),Base64.DEFAULT);

return publicSignature.verify(signatureBytes);
}
```

Per dur a terme la verificació de les firmes digitals, s'utilitzen les dues funcions anteriors. Utilitzam la clau pública de l'emissor per verificar que ha estat ell qui ho ha enviat. Una vegada desxifrada la firma, es calcula la funció de *hash* sobre la informació per comprovar que no ha estat modificada.

```
public static String hash(String text) {
String hash = "";
MessageDigest digest;
digest = MessageDigest.getInstance("SHA-256");
byte[] hashBytes = digest.digest(text.getBytes("UTF-8"));

hash =
new String(Base64.encode(hashBytes,Base64.DEFAULT)).replace("\n");

return hash;
}
```

Si el que volem fer és la funció de *hash* a soles utilitzarem l'algoritme mencionat *SHA-256*. Aquesta funció s'utilitza sobretot per calcular els valors arrel X_0 *i* Y_0 i també per verificar la validesa dels cupons.

5.1.2 Control de versions

Com s'ha comentat al principi del capítol, s'ha utilitzat una eina de control de versions, en aquest cas *Git* [34]. Aquest és un programari de control de versions que ens és d'ajuda a l'hora d'implementar tot el codi, ja que per controlar grans quantitats de codi és recomanable utilitzar un control de les versions. A més, ens proporciona les següents característiques interessants per al projecte:

- ***Backup***: ens serveix com a sistema de restauració, en cas de perdre part del codi, ja que tot el codi està guardat d'una manera estructurada al repositori *online*.

- **Desenvolupament no lineal**: ens permet que un grup de persones faci feina damunt el mateix projecte alhora, a més d'una manera ràpida ja que es pot fer feina en branques independents una de l'altra, amb facilitat d'integrar-les entre elles.

- **Gestió distribuïda**: cada un dels desenvolupadors pot tenir una còpia en local de tot el projecte i de l'historial de les diferents actualitzacions que ha sofert el projecte.

Com s'ha dit anteriorment, tot el projecte es guarda en un repositori *online*. Per això es va elegir la plataforma *GitHub*, que proporcionen un any de repositoris privats gratuïts als estudiants.

5.2 Servidor

La implementació del servidor es farà de manera estructurada seguint el següent esquema:

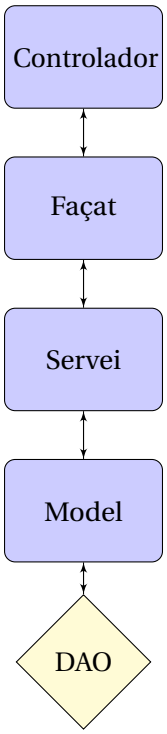

Figura 5.1: Diagrama de la implementació del Servidor.

- **Controlador**: és la capa més alta de l'estructura. Aquesta és on arriben totes les peticions generades per l'usuari i per on retornen les respostes a aquests.

- **Façat**: aquesta capa és l'encarregada de presentar les dades als controladors d'una manera estructurada. Ens serveix per separar la lògica de la presentació de les dades, ja que si no hauríem de tenir les dues capes juntes, serveis i façat alhora.

- **Servei**: és la capa que conté la part lògica del programa.

- **Model**: serveix per definir l'estructura de les dades de les diferents entitats.

- ***DAO***: *Data Access Object* és un component de *software* que subministra una interfície entre l'aplicació i la base de dades. Aquests permeten realitzar operacions amb les dades obtingudes fora mostrar cap tipus de informació.

Una vegada definides les capes del servidor, explicarem els components de cada una d'elles amb més detall.

5.2.1 Controladors

Els diferents controladors que hem definit en la part del servidor són:

- ***MerchantController* o Controlador del comerciant**: és el controlador que correspon a l'entitat del comerciant on figuren les diferents peticions entrants o sortides, com per exemple:

```
@RequestMapping(value="/merchant/url",method=RequestMethod.GET)
@ResponseBody
public String getParamsRedeem(){
return merchantService.initRedeemParamsMCoupon();
}
```

  El que defineix la funció anterior és quan un comerciant rep una petició **GET** del protocol HTTP a la direcció */merchant/url*, s'inicialitza una entrega d'un cupó que fa un usuari al comerciant.

- ***IssuerController* o Controlador del Distribuïdor**: és el controlador que correspon a l'entitat del distribuïdor on figuren les peticions relacionades amb l'assignació d'un usuari als diferents cupons.

- ***ManufacturerController* o Controlador del Fabricant**: és el controlador que correspon a l'entitat del fabricant on figuren les peticions relacionades amb el registre de nous usuaris a la plataforma, el *log-in* dels diferents usuaris a la plataforma i l'entrega de noves ofertes de cupons als diferents usuaris.

5.2.2 Façat

El façat és una capa que al final no s'ha implementat. La implementació d'aquesta capa es faria una vegada es volgués fer una implementació al mon real del projecte.

5.2.3 Serveis

Els serveis implementats són:

- ***CounterService* o Servei de comptador**: aquest servei conté la lògica de l'entitat comptador, és a dir, totes les funcions encarregades d'obtenir la informació sobre les vegades que s'ha entregat un cupó.
- ***MCouponService* o Servei dels cupons**: és el servei que conté la lògica encarregada de l'entitat cupó. S'encarrega de processar la informació que està relacionada amb els cupons.
- ***UserService* o Servei de l'usuari**: aquest servei conté la lògica de l'entitat usuari. Conté les funcions encarregades d'obtenir la informació relacionada amb els usuaris en les diferents etapes on participen.
- ***MerchantService* o Servei del comerciant**: aquest servei conté la lògica encarregada d'obtenir la informació dels diferents comerciants en les etapes en les qual participen.
- ***IssuerService* o Servei del Distribuïdor**: aquest servei conté la lògica encarregada de l'obtenció de informació relacionada amb el distribuïdor i les funcions necessàries per dur a terme les diferents fases del protocol on participa.
- ***ManufacturerService* o Servei del fabricant**: aquest servei conté la lògica encarregada d'obtenir la informació relacionada amb el fabricant i les funcions necessàries per dur a terme les diferents fases on participa.

5.2.4 Models

Per definir l'estructura de les diferents entitats s'ha duit a terme a través dels models. A continuació veurem amb detall cada un dels paràmetres dels que consta cada model.

- **Comptador**:
  - *Id*: identificador únic per cada comptador.
  - *LastHash*: conté el valor de la darrera funció *hash* que s'ha entregat de cada cupó.
  - *Counter*: conté el nombre de vegades que s'ha entregat un cupó.
  - *MCoupon*: identificador del cupó al qual fa referència el comptador.
- **Cupó**:
  - *Id*: identificador únic per cada un dels cupons.
  - *User Id*: usuari al qual pertany el cupó.
  - *Merchant Id*: identificador del comerciant al qual s'ha d'entregar el cupó.
  - *Counter*: objecte comptador al qual fa referència el cupó.
  - Y_0 i X_0: els dos valors arrel del cupó.

  - p i q: els dos valors que indiquen quantes vegades es farà la funció de *hash* sobre el cupó.
  - sn: número de sèrie únic per cada cupó.
  - exd: data en la que caduca el cupó.

- **Usuari**:
  - *Id*: identificador únic per cada un dels usuaris.
  - *Username*: identificador únic que serveix per dur a terme el *log in* a la plataforma.
  - *Password*: valor secret que juntament amb el *username* completen la fase de *log in*.
  - *M-Coupons*: llista de cupons de la que disposa l'usuari.
  - *Manufacturer Id*: identificador del fabricant al qual està registrat l'usuari.
  - X i Y: són les dues cadenes que s'utilitzaran per calcular els dos valors arrel.

- **Comerciant**:
  - *Id:* identificador únic per cada un dels comerciants.
  - *Issuer Id*: identificador del distribuïdor al qual està associat el comerciant.
  - *M-Coupons*: llistat dels diferents cupons els quals s'han d'entregar al comerciant.
  - *Name*: nom del comerciant.

- **Distribuïdor**:
  - *Id*: identificador únic de cada un dels distribuïdors.
  - *Merchants*: llista de comerciants que estan associats al distribuïdor.
  - *Manufacturer Id*: identificador del fabricant al qual està associat el distribuïdor.
  - *Name*: nom del distribuïdor.

- **Fabricant**:
  - *Id*: identificador únic de cada un dels fabricants.
  - *Issuers*: llista dels distribuïdors que estan associats al fabricant.
  - *Users*: llista d'usuaris que s'han registrat al fabricant.
  - *Name*: nom del fabricant.

5.2.5 DAO, *Data Access Object*

Com hem dit abans, s'ha definit la capa dels objectes encarregats d'accedir a la informació de la base de dades. Els *DAO* implementats són els següents:

- ***CounterRepository* o Repositori de comptadors**: s'ha definit una única funció que serveix per obtenir el comptador corresponent a un cupó, fent la cerca a través del cupó. A continuació podem veure un exemple de com s'implementa:

```
@Repository("countermcouponRepository")
public interface CounterMCouponRepository extends
PagingAndSortingRepository<CounterMCoupon, Long>{

CounterMCoupon findByMCoupon(MCoupon mCoupon);
}
```

- ***MCouponRepository* o Repositori de cupons**: al repositori dels cupons s'han definit dues funcions:
  - Obtenir un cupó a través del seu identificador.
  - Dur a terme la cerca de tots els cupons guardats en la base de dades i guardar-los en una llista de cupons.
- ***MerchantRepository* o Repositori de comerciants**: s'han definit dues úniques funcions:
  - Dur a terme la cerca del comerciant a través del seu nom.
  - Dur a terme la cerca de tots els comerciants a través de l'identificador del distribuïdor.
- ***IssuerRepository* o Repositori de distribuïdors**: sols s'ha definit una funció que s'encarrega de dur a terme la cerca del distribuïdor a través del nom.
- ***ManufacturerRepository* o Repositori de fabricants**: sols s'ha definit una funció que s'encarrega de dur a terme la cerca del fabricant a partir del nom.

5.3 Client

Una vegada el servidor està implementat, s'ha de separar tota la lògica corresponent a l'usuari, per dur-la a l'aplicació mòbil. Com s'ha comentat en capítols anteriors haurem d'implementar la base de dades de forma que es puguin emmagatzemar dades persistents en l'aplicació. La forma en que s'ha estructurat la implementació és la següent:

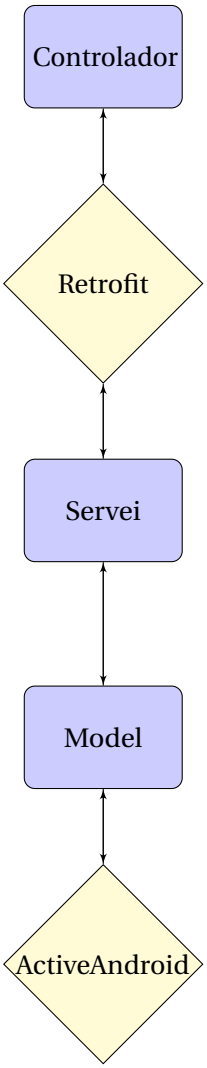

Figura 5.2: Diagrama de la implementació del client.

- **Controlador**: com tenim en el servidor, en la part del client també tenim els controladors encarregats d'enviar i rebre les diferents peticions a l'aplicació.
- ***Retrofit***: és la llibreria encarregada de dur a terme les connexions HTTP entre el servidor i l'aplicació.
- **Servei**: és la capa que conté la part lògica de l'aplicació.
- **Model**: és la capa que defineix l'estructura de les dades.
- ***ActiveAndroid***: és la llibreria encarregada de dur a terme la configuració de la base de dades.

5.3.1 Controladors

Els controladors, en el cas de l'aplicació mòbil, s'encarreguen de definir els diferents mètodes per dur a terme les peticions HTTP entre l'usuari i les diferents entitats que

resten al servidor. Els controladors definits són els següents:

- ***MerchantController* o Controlador del comerciant**: conté les diferents peticions que l'usuari necessita fer al comerciant en les diferents etapes. Per veure com s'han implementat veurem un exemple:

```
@POST("/merchant/getChallengeRedeem")
Call<String> getChallengeRedeem(@Body String json);

@GET("/merchant/getParamsRedeem")
Call<String> getParamsRedeem();
}
```

  En aquest cas podem veure dues peticions HTTP, una utilitzant el mètode *GET* i l'altre el mètode *POST*. La direcció a les quals van dirigides les peticions són les que fan referència al servidor, en aquest cas al controlador del comerciant. Per dur a terme aquestes peticions s'utilitza la llibreria esmentada anteriorment *Retrofit*.

- ***IssuerController* o Controlador del distribuïdor**: conté les diferents peticions que l'usuari ha de fer al distribuïdor en les diferents fases del protocol.

- ***ManufacturerController* o Controlador del fabricant**: conté les diferents peticions que l'usuari ha de fer al fabricant per completar les diferents fases del protocol.

5.3.2 *Retrofit*

Retrofit és la llibreria que s'encarrega de dur a terme les connexions HTTP, però per fer-ho necessitam la classe *AsyncTask* que ens serveix per executar diferents *threads* o fils en segon pla. Les *AsyncTask* generades són una per cada una de les funcions que ha de dur a terme l'aplicació, i són les següents:

- ***BuyCouponTask***: aquesta tasca ens serveix per dur a terme la connexió HTTP quan un usuari vol obtenir un nou cupó. La manera d'implementar-ho és la següent:

```
IssuerAPI issuerAPI = new Retrofit.Builder()
  .baseUrl(Constants.BASE_URL)
  .addConverterFactory(ScalarsConverterFactory.create())
  .build()
  .create(IssuerAPI.class);

Call<String> stringCall = issuerAPI.setUserCoupon(json.toString());
message = stringCall.execute().body();
```

  Bàsicament el que es fa en aquestes funcions és dur a terme la connexió HTTP, una vegada generada, preparar la petició per a obtenir un nou cupó, i enviar-la a través de la connexió generada.

- ***LoginTask***: conté les peticions per dur a terme la fase de *log in* en l'aplicació. Posteriorment el servidor verificarà si aquesta petició és correcta.

- ***OffersTask***: tasca que s'encarrega d'obtenir les diferents ofertes disponibles per als diferents usuaris.

- ***RedeemCouponTask***: tasca encarregada de generar les peticions per dur a terme una entrega d'un cupó.

- ***SignupTask***: tasca encarregada de generar les peticions necessàries per dur a terme el registre d'un usuari nou a la plataforma.

5.3.3 Serveis

Per a la lògica de l'entitat usuari no són necessaris tants de serveis. Els que s'han implementat són els següents:

- ***CounterService* o Servei del comptador**: conté les funcions encarregades d'obtenir la informació sobre les vegades que s'ha entregat un cupó.

- ***MCouponService* o Servei dels cupons**: conté la lògica per obtenir i processar la informació relacionada amb els cupons.

- ***UserService* o Servei de l'usuari**: conté les funcions encarregades d'obtenir i processar la informació dels diferents usuaris.

- ***BluetoothService* o Servei *Bluetooth***: és el servei encarregat de dur a terme la connexió *bluetooth* entre els diferents dispositius mòbils.

5.3.4 Model

Com ens ha passat amb els serveis, els models també s'han simplificat, degut a que sols s'han de definir les estructures de la part de l'entitat usuari. Els models són el comptador, cupó i usuari. No entrarem més en detall ja que els atributs són els mateixos que s'han definit al servidor.

5.3.5 Interfície d'Usuari

A continuació explicam com s'ha implementat la interfície d'usuari, on tenim les diferents activitats, que són les següents:

- ***LoginActivity***: és la primera activitat de la interfície. Aquesta sols conté un botó per dur a terme la fase de registre i un altre per dur a terme la fase de *log in* de les dades introduïdes en els camps que mostra l'activitat.

- ***SignupActivity***: en l'activitat anterior s'accedeix a la fase de registre, on es mostrarà aquesta activitat que conté un petit formulari on s'han d'introduir el *username* i la *password* que es desitja.

- ***MainActivity***: una vegada l'usuari ha duit a terme el *log in* correctament, s'engega l'activitat principal, que en la primera part ens mostrarà els diferents cupons que tenim disponibles per entregar. Per fer-ho més modular, l'activitat principal s'ha dividit en tres fragments on cada un conté la seva funcionalitat. Aquests són els següents:
  - ***MCouponList***: fragment principal on es mostren tots els cupons disponibles per entregar.
  - ***MCouponOfferList***: fragment que ens mostra totes les ofertes disponibles que té l'usuari.
  - ***MCouponShareList***: fragment que ens mostra tots els cupons que tenim disponibles per compartir amb altres dispositius.
- ***BluetoothActivity***: si el que es vol és compartir el cupó, es farà a través d'aquesta activitat. Aquesta ens deriva a la selecció de dispositius disponibles per dur a terme la connexió *bluetooth* entre els dispositius i posteriorment fer la compartició del cupó.
  - ***DeviceListActivity***: aquesta és l'activitat que deriva de l'anterior, i ens permet elegir a quin dispositiu *bluetooth* ens volem connectar.

5.3.6 Configuració

Les configuracions específiques per poder dur a terme totes les funcionalitats es recullen en l'arxiu *AndroidManifest.xml* de l'aplicació. A continuació s'explicarà quins han estat els canvis més significatius duits a terme.

Permisos

Els permisos que s'han hagut de configurar són els següents:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name=
"android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name=
"android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name=
"android.permission.READ_INTERNAL_STORAGE" />
<uses-permission android:name=
"android.permission.WRITE_INTERNAL_STORAGE" />
<uses-permission android:name="android.permission.BLUETOOTH" />
```

- ***Internet***: permís que ens permet dur a terme les diferents operacions de xarxa.
- ***Storage***: permisos que ens permeten la lectura i escriptura tant en la memòria interna com en la memòria externa.
- ***Bluetooth***: permís necessari perquè l'aplicació pugui accedir al *bluetooth*.

Base de dades

La configuració referent a la base de dades és la següent:

```
<meta-data android:name="AA_DB_NAME" android:value="MCoupon.db" />
<meta-data android:name="AA_DB_VERSION" android:value="10" />
```

Simplement són els dos camps per indicar quin és el nom de l'arxiu que conté la base de dades i la versió d'aquesta per si es fan modificacions a l'estructura.

Activitats

Dins l'arxiu *AndroidManifest.xml* també s'han de configurar les diferents activitats presents en l'aplicació. Es configuren de la següent manera:

```
<activity android:name=".activity.LogInActivity"
android:label="MCoupon">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity android:name=".activity.MainActivity" />
<activity android:name=".activity.SignUpActivity" />
<activity android:name=".activity.BluetoothActivity" />
<activity android:name=".activity.DeviceListActivity" />
```

CAPÍTOL 6

RESULTATS

En aquest capítol es veuran els resultats obtinguts una vegada implementades totes les fases i entitats del protocol. Per veure-ho ho farem des del punt de vista de l'aplicació mòbil, ja que és l'entitat que participa en totes les fases del protocol.

A la següent figura es mostra el digrama de flux corresponent a l'aplicació mòbil. Posteriorment s'explica cada una de les parts de manera visual amb el resultat obtingut.

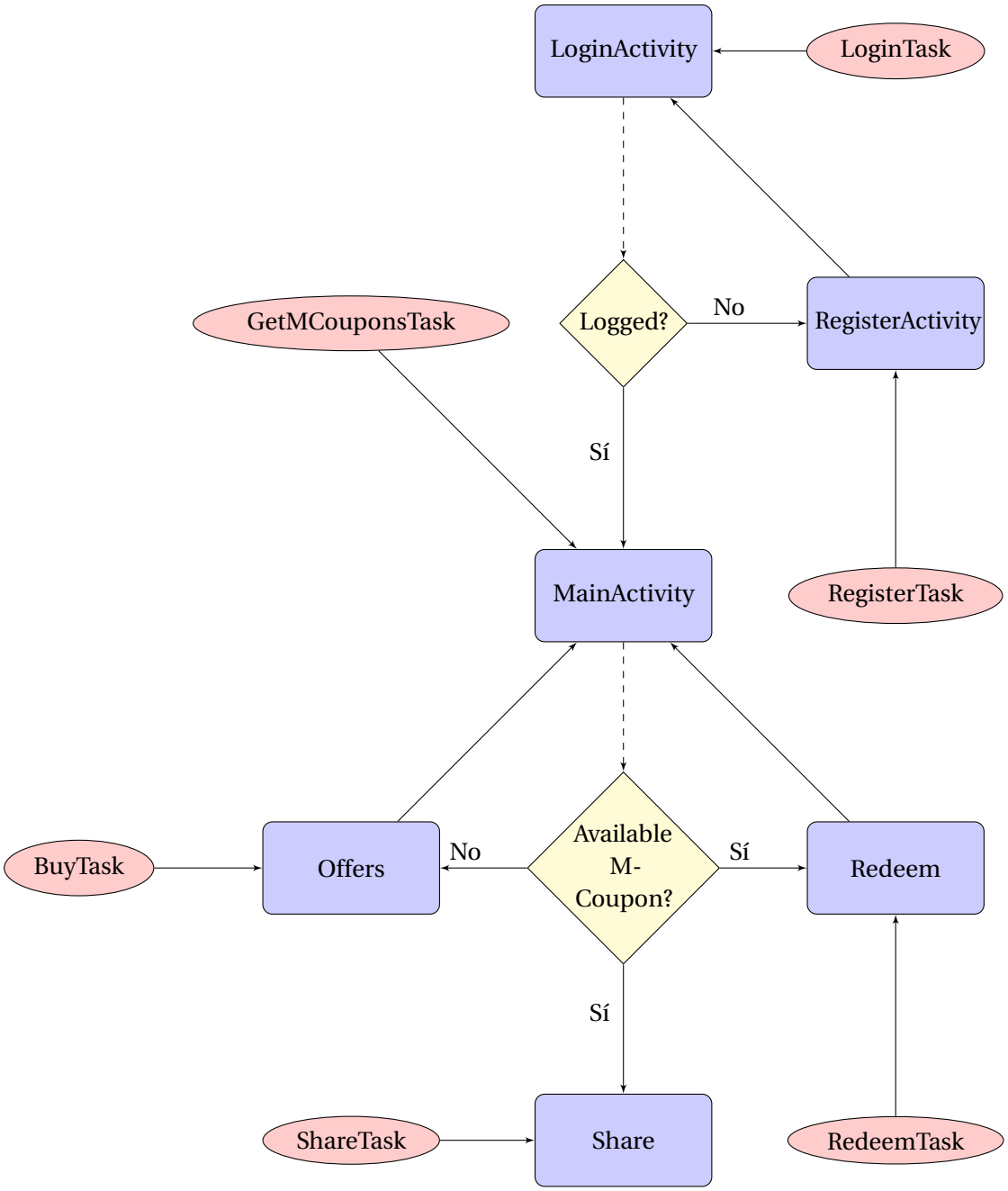

Figura 6.1: Diagrama de l'aplicació mòbil.

Com podem veure al diagrama, la primera activitat que ens trobam una vegada executam l'aplicació mòbil és el *LoginActivity*.

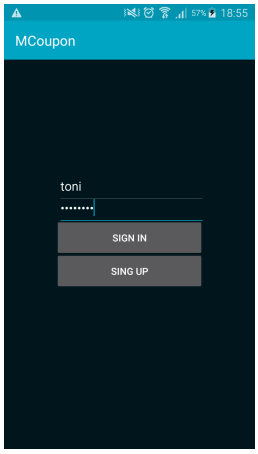

Figura 6.2: Interfície del *Log in*

Com podem veure a la figura 6.2, sols té un camp de *username* i *password* perquè els usuaris que estan registrats puguin dur a terme el *log in* directament, i per als usuaris que no estan registrats tenim l'opció de *register*, la qual ens derivarà a la *SignUpActivity*.

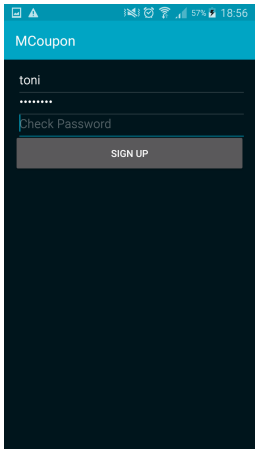

Figura 6.3: Interfície del registre.

La figura 6.3 consta d'un petit formulari, on s'ha d'indicar el *username* i la *password* que es vol utilitzar per dur a terme el *log in*. Per dur a terme el registre correctament s'ha implementat de manera que la *password* ha de ser de com a mínim de sis caràcters i s'ha d'escriure dues vegades en el formulari.

Una vegada l'usuari s'ha registrat ja està llest per dur a terme el *log in*, tornant arrere a l'activitat *LoginActivity*. S'han d'introduir les dades registrades en la fase de registre i duim a terme el *log in* correctament. Una vegada fet el *log in*, ens trobam amb l'activitat *MainActivity*. Aquesta ens mostrarà si tenim cap cupó disponible per entregar, i si no, aquesta estarà buida. Ho podem veure a la figura següent:

Figura 6.4: Interfície del *Main*.

L'activitat *MainActivity* consta de la pantalla principal on es poden veure els diferents cupons disponibles per a l'usuari. A la figura 6.4 podem veure com és buida degut a que l'usuari no ha obtingut cap cupó. Es pot obtenir un cupó anant a la pestanya de *Offer* que es troba en la barra d'eines superior.

Figura 6.5: Fragment de les ofertes.

Una vegada dedins ens trobam amb les diferents ofertes de cupons que ha generat el fabricant perquè estiguin disponibles per als usuaris. Si volem obtenir una d'aquestes ofertes simplement s'ha de pitjar a sobre d'una d'elles i ens trobam amb la pantalla següent:

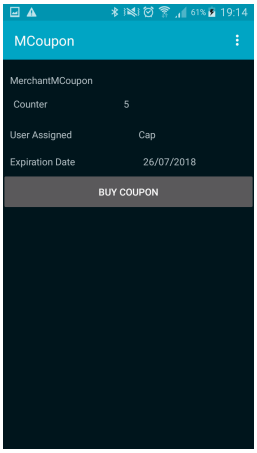

Figura 6.6: Fragment per obtenir el cupó.

Ens trobarem amb les diferents característiques que presenta el cupó juntament amb l'opció d'obtenir el cupó que és el que ens interessa. Una vegada triada l'opció d'obtenir el cupó l'hauríem de tenir en la pantalla principal, ja disponible per entregar-lo al comerciant assignat.

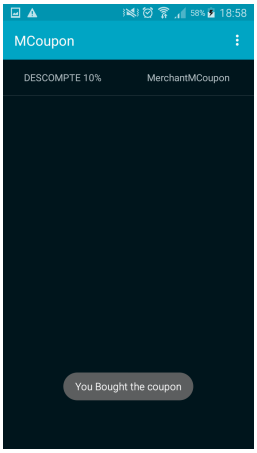

Figura 6.7: Cupó obtingut correctament.

Una vegada disposats a dur a terme l'entrega del cupó, simplement pitjant sobre el cupó disponible ens trobarem amb la pantalla anterior, on podem veure que ara l'usuari assignat a aquell cupó és el nostre. Per prosseguir amb la fase d'entrega, simplement s'ha de clicar sobre el botó d'entrega de cupó.

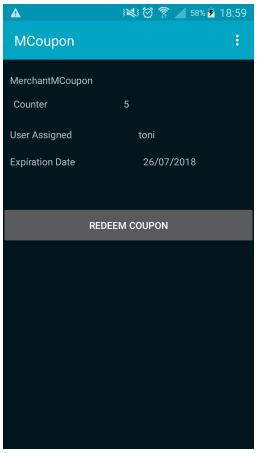

Figura 6.8: Fragment per entregar el cupó.

Una vegada entregat el cupó correctament ens trobarem amb la següent pantalla:

Figura 6.9: Primer cupó entregat correctament.

Per comprovar que sols es pot entregar el cupó p vegades, que és el nombre que s'ha vist abans d'obtenir el cupó, una vegada l'hem entregat p vegades, ens trobarem amb la pantalla següent, on desapareix l'opció d'entrega de cupó i ens avisa de que no es pot entregar més vegades.

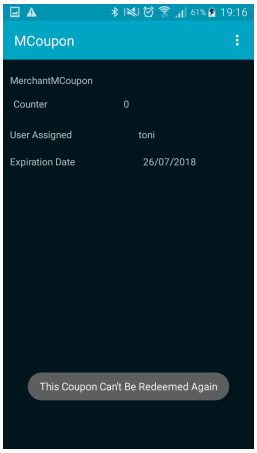

Figura 6.10: Cupó entregat el màxim de cops.

Per altra banda, una vegada hem obtingut el cupó, també tenim l'opció de compartir el cupó. Per dur a terme la compartició tenim la pestanya de *share* que també es troba en la barra d'eines superior. Una vegada dins l'activitat ens trobarem amb el cupó disponible per compartir, igual que a la fase d'entrega. Així que sols caldrà pitjar sobre el cupó disponible i ens trobarem amb la pantalla següent:

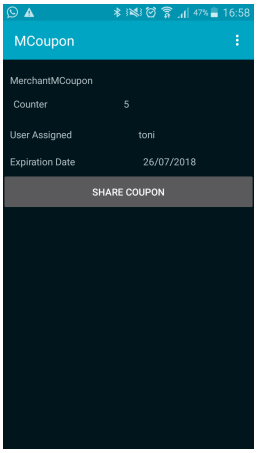

Figura 6.11: Interfície per compartir el cupó.

Una vegada disposats a compartir el cupó, el primer que ens apareixerà una pantalla on simplement hi ha un botó per enviar el cupó, i just abaix del nom de l'aplicació ens indica si esteim connectats amb algun dispositiu *bluetooth* o no. En el segon cas encara hem de dur a terme la connexió abans de compartir el cupó. Per dur-la a terme ens dirigim a la barra d'eines superior on tendrem l'opció de veure els dispositius disponibles per dur a terme la connexió.

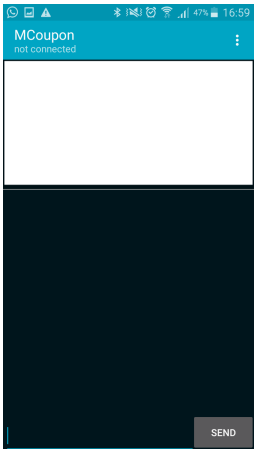

Figura 6.12: Interfície de *Share*.

En la pantalla següent podrem veure els diferents dispositius disponibles per dur a terme la connexió. En aquest cas elegirem el dispositiu *ToniGornalsF60*.

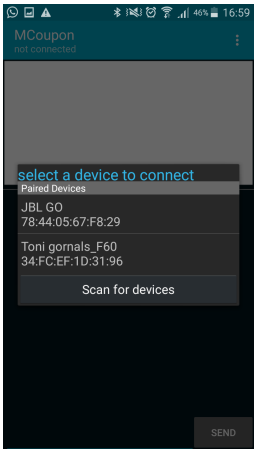

Figura 6.13: Fragment per dur a terme la connexió.

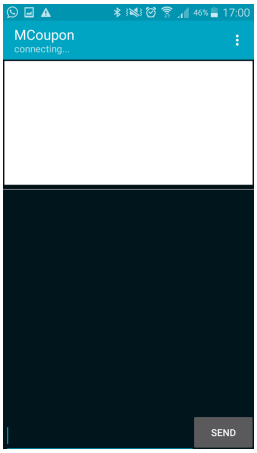

Figura 6.14: Connectant.

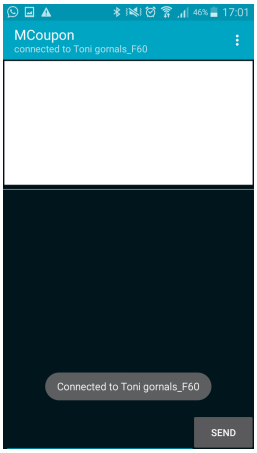

Figura 6.15: Connexió finalitzada al mòbil 1.

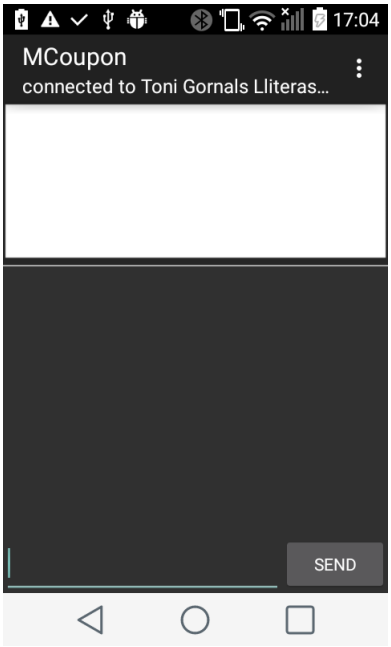

Figura 6.16: Connexió finalitzada al mòbil 2.

Una vegada finalitzada la connexió correctament, ens disposam a pitjar sobre el botó *send* per compartir el cupó amb l'altre usuari. Com s'ha pogut comprovar quan ha finalitzat la connexió els dos dispositius estan en la mateixa activitat de *share*. Aquest és un requeriment per dur a terme la connexió a través de l'aplicació.

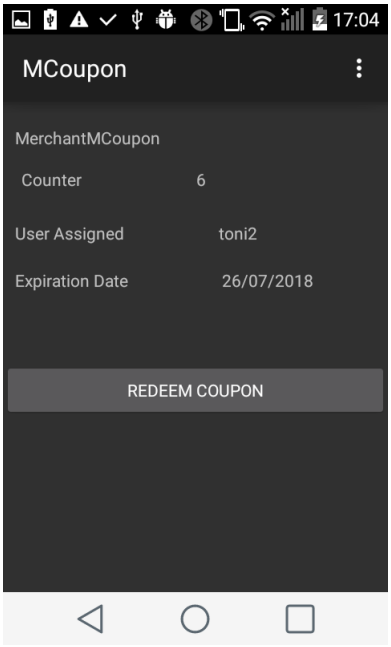

Figura 6.17: Cupó compartit.

Una vegada ha finalitzat el procés, en el segon dispositiu ens trobarem amb el cupó dis-

ponible per dur a terme l'entrega del cupó. Podem comprovar que les característiques del cupó són diferents. Ara el cupó es pot entregar q vegades i l'usuari assignat és el propietari del segon dispositiu.

CAPÍTOL 7

CONCLUSIONS

7.1 Anàlisi

Com s'ha dit Internet i les tecnologies *web* han canviat molt les maneres de fer negoci en general. Aquests canvis han permès als usuaris obtenir informació sobre els productes en què estan interessats d'una manera ràpida i senzilla. Els negocis s'han interessat molt en dur a terme aquests canvis en el seu model de negoci, degut a que els proporciona un augment de les ventes significatiu, i els permet dur a terme campanyes de publicitat dels seus productes amb molta més facilitat, rapidesa i efectivitat.

Per altra banda, la introducció dels dispositius mòbils en aquest tipus de comerç ha duit a una evolució del mateix, degut a què avui dia gran part de la població disposa d'un dispositiu mòbil en la seva butxaca. A l'hora de dur a terme aquest projecte s'ha tengut en compte la introducció massiva d'aquests dispositius mòbils al comerç. Per això tot el que fa referència a l'usuari s'ha duit a terme sobre una aplicació suportada per aquests tipus de dispositius.

L'article en que s'ha basat aquest document és *Sharing secure m-coupons for peer-generated targeting via eWOM communications* de *Sue-Chen Hsueh i Jun-Ming Chen* [1]. Aquest protocol presenta una nova forma de compartir cupons, ja que es poden compartir entre els mateixos usuaris, canvi que presenta grans avantatges sobre el procés de compra - venta de productes. Per altra banda, el protocol ens requereix d'implementar mesures de seguretat per poder assegurar que les comunicacions es fan de manera confidencial i integra.

S'han explicat les diferents fases que s'han duit a terme per al desenvolupament del projecte. Primer s'han definit les comunicacions necessàries entre les diferents entitats. Durant el desenvolupament han sorgit nous paràmetres a enviar o fins i tot paràmetres eliminats de les comunicacions proposades en l'article.

Tot seguit, s'ha explicat com s'han dissenyat els elements que comporta el protocol, principalment el servidor i el client o aplicació mòbil. Així, hem descrit tots els aspectes relacionats amb el disseny, incloent les tecnologies utilitzades i el perquè d'elles, així com la manera d'interconnectar-les. A continuació, s'ha explicat com s'han utilitzat en la implementació cada una de les tecnologies que s'han definit en el disseny, així com les entitats que participen en el protocol.

Finalment, s'ha demostrat quin ha estat el resultat d'aplicar totes les etapes anteriors a nivell pràctic, fent referència a cada una de les etapes durant el procés de registre, *log in* dels diferents usuaris i la compra, entrega i compartició dels cupons.

7.2 Línies Futures

Hi ha certs aspectes que es poden millorar, tant en la part del servidor com en l'aplicació mòbil. Alguns dels més importants són:

- La implementació de la tecnologia *NFC*. Aquesta tecnologia ens pot proporcionar rapidesa a l'hora de fer l'entrega i compartició dels cupons.
- Fer l'aplicació compatible per a sistemes *iOS*, i d'aquesta manera augmentar l'abast en el mercat de dispositius mòbils.
- Fer el desenvolupament final d'una aplicació *web* capaç de comunicar-se amb el protocol.
- Acabar de completar el *framework* per així afegir més funcionalitats.

BIBLIOGRAFIA

[1] S.-C. Hsueh and J.-M. Chen, “Sharing secure m-coupons for peer-generated targeting via ewom communications,” January 2010. 1.1, 3, 7.1

[2] Near field communication. [Online]. Available: https://nfc-forum.org 2.1.1

[3] Quick response code. [Online]. Available: https://en.wikipedia.org/wiki/QR_code 2.1.1

[4] Bluetooth low energy. [Online]. Available: https://www.link-labs.com/blog/bluetooth-vs-bluetooth-low-energy 2.1.1

[5] S. Dominikus and M. Aigner, “mcoupons: An application for near field communication (nfc),” September 2005. 2.2.1

[6] S.-K. N. S.-R. Lee and D. Choi, “Proposed m-payment system using near-field communication and based on wsn-enabled location-based services for m-commerce,” January 2014. 2.2.2

[7] A. Bartoli and E. Medvet, “An architecture for anonymous mobile coupons in a large network,” September 2016. 2.2.3

[8] Hash function. [Online]. Available: https://learncryptography.com/hash-functions/what-are-hash-functions 4.1.1

[9] Message digest 5. [Online]. Available: http://searchsecurity.techtarget.com/definition/MD5 4.1.1

[10] Secure hash algorithm. [Online]. Available: https://es.wikipedia.org/wiki/SHA-2 4.1.1

[11] Asymetric algorithm. [Online]. Available: https://www.genbetadev.com/seguridad-informatica/tipos-de-criptografia-simetrica-asimetrica-e-hibrida 4.1.2

[12] Elgamal encryption algorithm. [Online]. Available: http://www.iusmentis.com/technology/encryption/elgamal/ 4.1.2

[13] Rivest, shamir and adleman. [Online]. Available: http://searchsecurity.techtarget.com/definition/RSA 4.1.2

[14] Digital signature algorithm. [Online]. Available: https://en.wikipedia.org/wiki/Digital_Signature_Algorithm 4.1.2

[15] X.509 certificate structure. [Online]. Available: https://en.wikipedia.org/wiki/X.509 4.1.2

[16] Java language. [Online]. Available: https://docs.oracle.com/javase/7/docs/technotes/guides/language/index.html 4.2.2

[17] Spring framework. [Online]. Available: https://spring.io/ 4.2.2

[18] Hibernate. [Online]. Available: http://hibernate.org/ 4.2.2

[19] Ruby on rails. [Online]. Available: http://rubyonrails.org/ 4.2.2

[20] Django project. [Online]. Available: https://www.djangoproject.com 4.2.2

[21] Meteor. [Online]. Available: https://www.meteor.com 4.2.2

[22] Gradle. [Online]. Available: https://gradle.org 4.2.2

[23] Hyper sql database. [Online]. Available: http://hsqldb.org 4.2.2

[24] Structured query language. [Online]. Available: https://en.wikipedia.org/wiki/SQL 4.2.2

[25] Angular. [Online]. Available: https://angular.io/ 4.2.2

[26] Thymeleaf. [Online]. Available: http://www.thymeleaf.org/ 4.2.2

[27] Model, view and controller. [Online]. Available: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller 4.2.2

[28] Android operating system. [Online]. Available: https://www.android.com/ 4.3.2

[29] ios. [Online]. Available: https://en.wikipedia.org/wiki/IOS 4.3.2

[30] Windows phone. [Online]. Available: https://en.wikipedia.org/wiki/Windows_Phone 4.3.2

[31] Retrofit. [Online]. Available: http://square.github.io/retrofit 4.3.2

[32] Sqlite. [Online]. Available: https://www.sqlite.org/index.html 4.3.2

[33] Activeandroid. [Online]. Available: http://www.activeandroid.com 4.3.2

[34] Git. [Online]. Available: https://git-scm.com/ 5.1.2