QANTIQA: MICROBLOGGING DESCENTRALITZAT

RUBÈN-DARIO CASTAÑÉ CARMONA

Aplicacions i sistemes distribuïts

Enginyeria Tècnica en Informàtica de Sistemes

Universitat Oberta de Catalunya

Carles Pairot Gavaldà

Març 2010

Rubèn-Dario Castañé Carmona: *Qantiqa: microblogging descentralitzat*

TUTOR:
Carles Pairot Gavaldà

DATA:
Març 2010

Qantiqa és un projecte de microblogging descentralitzat que cerca explorar les possibilitats de l'aplicació del paradigma d'aplicacions distribuïdes en el món del microblogging, basant-se en el middleware EasyPastry i modelant-lo sobre el model establert per Twitter.

Aquest document detalla tots els aspectes del desenvolupament inicial, tals com la recollida de requeriments, anàlisi, disseny i planificació de totes les fases corresponents.

ÍNDEX

ÍNDEX DE FIGURES

Part I

INTRODUCCIÓ A QANTIQA

1 JUSTIFICACIÓ

La Xarxa ha estat origen de nous mecanismes de comunicació com el correu electrònic, la missatgeria instantània, les BBS[1], els fòrums, etc. Alguns estan en desús, altres evolucionen i originen noves eines que revolucionen la vida de milers d'usuaris.

Un clar exemple és el blog, eina de publicació per excel·lència. Versàtil, disponible sota múltiples plataformes i accessible, qualsevol pot iniciar-ne un per tractar els seus temes d'interès.

Algunes de les plataformes més conegudes son Blogger, LiveJournal i WordPress.

Un dels formats més emprats és el microblogging (o tumblelog), aparegut a mitjans d'aquesta darrera dècada, el qual redueix el blog a la mínima expressió.

La causa d'aquest augment ha estat originat en Twitter, aparegut el 2006 i actual referència del microblogging. Twitter ho redueix a la mínima expressió, limitant-nos a 140 caràcters, tal i com un SMS.

Tot i semblar no tenir massa sentit el publicar frases breus, s'ha generat un ecosistema ric en possibilitats, convertint-se en una eina de comunicació emprada per tot tipus d'usuaris, des de ciutadans de a peu fins institucions com la White House, passant per estrelles mediàtiques, empreses, etc.

I el seu ús s'ha estès fins ser part imprescindible la web en temps real, emprant-se per informar d'events que estan passant ara mateix.

Però Twitter està limitat per la seva infraestructura i decisions. Si està fora de servei, es perd una eina important de la que depèn molta gent, a més de ser la única dipositària de tota la informació transmesa.

Altres xarxes de microblogging o amb certs aspectes són Google Buzz, Yahoo Meme, Facebook, Identi.ca, Jaiku o Tumblr.

Per això, un següent pas està en estandarditzar aquesta nova eina, esforç iniciat per la iniciativa OStatus. Actualment té un esborrany del protocol, basat en altres d'existents, i ja implementat parcialment per altres plataformes.

Més enllà de l'estandardització, un altre pas seria anar cap a una infraestructura amb la mateixa funcionalitat però sense els inconvenients actuals, més oberta i descentralitzada, sense perdre la compatibilitat amb l'ecosistema ja creat.

Amb aquesta idea inicial plantegem la creació de Qantiqa i explorar com les característiques pròpies de les aplicacions distribuïdes poden beneficiar una eina d'aquestes característiques.

1 Bulletin Board System

2 OBJECTIUS

Els objectius de Qantiqa giren entorn de cinc principis:

- Compatibilitat
- Alta disponibilitat
- Autenticació
- Sense spam
- Facilitat de cerca

2.1 COMPATIBILITAT

L'ecosistema creat al voltant de Twitter és extremadament ric en possibilitats pels usuaris, sobretot pel que fa en aplicacions per web i d'escriptori.

Hi ha vàries aplicacions de l'últim tipus que són open source i que ens permeten disposar d'un client per Qantiqa simplement modificant les URLs a les que apunta.

Suportar les aplicacions de tercers implica implementar una API[1] com la que ofereix Twitter. Pel que fa a la primera versió del projecte, implementarem la API completa però limitant-nos en suportar l'autenticació HTTP, tot i que estigui en camí d'entrar en desús a favor d'OAuth.

Així, treballarem amb Buzzbird, un client multi-plataforma basat en XUL, el runtime de Firefox, Thunderbird i Songbird, entre altres projectes open source.

2.2 ALTA DISPONIBILITAT

La possibilitat de trobar-nos amb Twitter saturat o inaccessible no és pas tant remota com pugui semblar. Aquí és on la filosofia distribuïda permet assolir, teòricament, la disponibilitat total de qualsevol missatge, sobretot gràcies a la replicació de parells de clau/valor entre els veïns d'un node segons latència, i no per criteris geogràfics.

El model de xarxa amb "super nodes"és similar al del servei DNS, on res t'impedeix accedir als servidors directament per IP, però és més fàcil treballar amb dominis.

Al no estar tot concentrat en un servidor o un conjunt de servidors, tot i que es podria oferir "super nodes" per tal de facilitat l'accés a la xarxa, aquesta mateixa serà accessible en tot moment, sempre que sapiguem la IP d'algun node.

El projecte haurà d'establir les bases per tal que aquest plantejament sigui cert i funcional

1 Application Programming Interface

2.3 AUTENTICACIÓ

En tota xarxa on el contingut sigui generat per qualsevol usuari s'ha de garantir que ningú publiqui amb un usuari que no sigui el seu.

Aquest és un dels punts més importants, ja que en una xarxa oberta i descentralitzada és difícil identificar un usuari. En aquest projecte explorarem les possibilitats de l'autenticació en entorns d'aquestes característiques.

2.4 SENSE SPAM

Totes les comunicacions actuals són susceptibles de tenir ús abusiu de les mateixes per promocionar productes i altres (males) ofertes. A Internet ja estem acostumats al conegut spam, del que intentem desfernos mitjançat múltiples tècniques com filtres bayesians, whitelisting, etc.

Twitter mateix ofereix algunes eines bàsiques al respecte, sense arribar a la complexitat dels filtres, per reportar usuaris que facin un mal ús del servei.

Tot i així, també és conegut per tenir freqüent spam, inclòs per haver estat emprat per controlar botnets, com en els vells temps d'IRC.

Amb això en ment, Qantiqa cercarà establir mecanismes de control, propers a les idees exposades per Zed Shaw al seu projecte Utu, basat en els conceptes d'identitat, reputació i retribució.

2.5 FACILITAT DE CERCA

La utilitat de serveis com Twitter resideix en la facilitat de cercar què està passant al món, prendre-li el pols i participar-hi immediatament.

Per tant, el producte resultant d'aquest projecte haurà de proporcionar eines de cerca de missatges relacionats amb certes paraules clau.

D'aquesta manera no ens limitarem a enviar missatges i podrem accedir a qualsevol informació en, pràcticament, temps real.

També serà un dels punts més importants del projecte, ja que sense això quedaria reduït a un software de missatgeria instantània més proper a clamar al desert que a facilitar la comunicació entre els usuaris.

3 REQUISITS

3.1 DESCRIPCIÓ INICIAL

Qantiqa pretén ser una eina de microblogging, per tant, l'usuari l'emprarà per publicar petits fragments de text lliure, a partir d'ara *quarks*, limitats a 140 caràcters.

Aquests quarks poden contenir partícules especials, tals com:

@nom_usuari És una menció a un altre usuari, a partir del seu nom d'usuari, ja sigui per dirigir un missatge públic o fer-ne referència en un quark. Pot estar precedit d'altres partícules:

- *- o via* Indica que estem reenviant, modificat, el quark publicat per un usuari.
- *: o cc*[1] Notifica a la llista d'usuaris que s'inclou a continuació que el quark publicat pot ser del seu interès.
- *RQ* Indica que estem reenviant, sense modificar, el quark publicat per un usuari. També anomenat *requark*.
- *D* Envia un quark privat a l'usuari.
- *(Sense, a l'inici)* És un quark públic dirigit a l'usuari. Normalment s'empra per dirigir-lo específicament o respondre a un altre quark.

#identificador Etiqueta per agrupar quarks dins d'un tema. Permeten detectar tendències dins de la xarxa. També anomenades *hash tags*.

L'usuari podrà seguir a un altre, en serà *follower*. Si el seguiment és recíproc, ambdós usuaris seran *friends*. Per tal de localitzar altres usuaris, emprarà el cercador d'usuaris.

I continuant amb la cerca, l'usuari es mantindrà al dia cercant certes paraules clau, per exemple, hash tags o qualsevol cadena de text que vulgui. La cerca serà actualitzable, per veure si han aparegut nous quarks.

L'usuari sempre estarà identificat amb les seves credencials, associades al seu nom d'usuari, pel que haurà de fer login a l'aplicació client per publicar quarks. A més, tindrà certa informació personal publicada que el permeti ser cercat.

Finalment, l'usuari tindrà la possibilitat de reportar usuaris que estiguin abusant del sistema, mitjançant un simple mecanisme que propagarà l'avís d'abús corresponent. Si l'usuari reportat és freqüent, els seus missatges seran alentits o rebutjats directament.

3.2 CASOS D'ÚS

En aquest projecte es presenten inicialment els següents casos d'ús:

Identificar usuari Validar el compte d'usuari en la xarxa.

Enviar quark Publicar un nou quark, ja sigui original, un requark o una resposta a un altre quark, però sempre públic.

Enviar quark privat Enviar un quark privat a un usuari determinat de la xarxa.

Seguir usuari Afegir-se a la llista de followers d'un usuari. Si és recíproc, els dos usuaris es consideren friends.

Cercar usuari Cercar usuaris en la xarxa que puguin ser coneguts, identificats per informació personal publicada.

Cercar quarks A partir de paraules clau, cercar quarks que les continguin i, per tant, estiguin relacionats amb la cerca.

Reportar usuari Propagar avís d'abús per part de l'usuari reportat a tota la xarxa.

4 DESCRIPCIÓ DE LA MEMÒRIA PFC

Aquesta memòria està composta per diferents capítols:

Introducció a Qantiqa Descripció general del projecte i establiment de les bases necessàries per la creació de la memòria, així com del producte final.

Planificació Descripció de la organització i timing del projecte, així com estimació de costos i detalls sobre els productes finals.

Anàlisi i Disseny Documentació sobre l'anàlisi i disseny realitzat del PFC.

Implementació Documentació sobre la implementació tècnica del PFC.

Conclusió Cloenda de la memòria PFC on es plasmen les impressions sobre la experiència del projecte.

5 ESTAT DE L'ART

Qantiqa es basa en diferents frameworks i tecnologies de l'ecosistema Java, presents en les diferents capes del seu nucli.

5.1 XARXA

Com eina descentralitzada, empra EasyPastry, framework de xarxa P2P descentralitzada i amb enrutament basat en claus (KBR) construït sobre Pastry.

Els nodes de la xarxa Pastry formen una capa de xarxa auto-organitzada a Internet, adaptant-se a l'arribada, marxa i fallida dels nodes. Així, cada usuari forma part de la xarxa, conformant el teixit actiu de Qantiqa, amb tota la informació pública (com Twitter) distribuïda per la mateixa, garantint l'accés als missatges dels usuaris encara que estiguin desconnectats.

Relacionat amb la xarxa, els missatges que hi viatgen són missatges serialitzats de Google Protocol Buffer (protobuf) , assolint alta consistència amb el seu ús, ja que tots els missatges tenen un esquema i són eficients en mida, minimitzant el tràfic i maximitzant la transmissió.

Són un mecanisme de serialització de dades, com XML, però com s'ha indicat, més eficient en mida i simple. A partir dels esquemes es genera el codi que gestiona el missatges (qantiqa-common/im.dario.qantiqa.common.protocol.Protocol), el qual s'adaptarà fàcilment a futures evolucions dels mateixos, ignorant els camps que no conegui. Així, teòricament es facilita la existència temporal de versions de software anteriors a la més recent, sense trencar-ne la compatibilitat.

Tot i així, els protobuf es poden convertir a altres formats, tals com XML i JSON, permetent una major interacció amb altres sistemes i clients, un requisit important de Qantiqa, ja que pretén mantenir-se compatible amb tots els clients Twitter, i aquests consumeixen la informació de l'API en format JSON, principalment, encara que alguns també ho fan en XML.

Així, amb un esforç mínim es garanteix que els missatges són consistents, transformables directament als formats requerits i fent ús de la experiència de Google en la optimització de comunicacions en xarxa.

5.2 WEB

5.2.1 *Infraestructura*

A diferència de la xarxa, altres parts del projecte fan servir models més convencionals, tal com la Web, coneguda per tots els usuaris que en fan ús diàriament.

Concretament, en Higgs està basat en un altre producte de Google: Google App Engine. Es tracta d'un sistema d'allotjament d'aplicacions Python i Java que empra els recursos de les infraestructures de Google,

especialment dissenyat per ser ràpid, segur i estàndard, facilitant el desenvolupament.

Les aplicacions funcionen en un "sandbox" amb un accés limitat al sistema operatiu en el que resideix, tot i que estan distribuïdes en molts servidors que actuen com un de sol, alhora que facilita aquesta distribució, al garantir que funcionen de forma independent al hardware i altres elements dels servidors.

Pel que fa a qantiqa-core, fa ús del servidor d'aplicacions Jetty, també emprat per Google a GAE, per la seva fàcil inclusió dins del software.

5.2.2 *Interfície i lògica HTTP*

Tant la API com Higgs empren el framework Play! per gestionar les peticions HTTP, així com resoldre-les, tant en forma de resposta XML, JSON o HTML.

Play! és un framework web "RESTful" que l'acosten més als frameworks web de Ruby, Python o PHP, tots llenguatges dinàmics, que als de Java, ja que permet un cicle de desenvolupament àgil, sense haver de compil·lar el codi font, així com oferint una arquitectura Model-Vista-Control·lador i altres eines habituals en el desenvolupament web, com:

- Gestió d'errors millorada, amb informació addicional.
- Tests unitaris integrats.
- Motor de plantilles.
- Etc.

El desenvolupament d'aplicacions webs amb Play! és molt senzill, amb una relació directa entre URLs i classes dins del codi font del projecte, pel que era una opció obligada per minimitzar l'esforç necessari, sense haver de reinventar la roda gestionant els aspectes típics de Servlets i servidors d'aplicacions Java.

5.3 EINES DE DESENVOLUPAMENT

A part de les eines de Play!, en el desenvolupament de Qantiqa s'ha optat per gestionar les dependències amb Maven.

Maven és una eina de gestió de projectes, dels quals les dependències en són una part important. Per això, existeix una infraestructura distribuïda de servidors HTTP que allotgen JARs dels diferents projectes de software lliure emprats més freqüentment. I pels que no ho són tant, s'ha configurat un repositori Maven amb Nexus, un gestor de repositoris Maven.

T

ot i així, les dependències de Play! són masses per allotjar-les al repositori personal, pel que es facilita un script d'instal·lació en el repositori local de Maven.

D'aquesta manera, s'aconsegueix simplificar la gestió de les dependències, quedant automatitzada l'adquisició de les mateixes amb les versions necessàries.

Finalment, Qantiqa es recolza en diferents projectes lliures d'Apache per completar la seva funcionalitat amb biblioteques provades per milers de programadors durant anys.

Part II

PLANIFICACIÓ

6 PLANIFICACIÓ

6.1 FASES DEL PROJECTE

El projecte presenta cinc etapes principals, seguint un model de desenvolupament iteratiu que s'anirà repetint periòdicament durant el projecte:

- Cerca de referències
- Anàlisi
- Disseny
- Implementació
- Documentació

En l'apartat Cronograma es detallen les fites, tasques i dates límits en el diagrama Gantt corresponent.

6.2 ESTIMACIÓ DE COSTOS

Suposant un cost de 10 € a l'hora i considerant que el projecte complet consta de 378 hores, com es pot veure en l'arxiu adjunt d'OpenProj, els costos són:

- Temps: 378 hores.
- Econòmic: 3.780 €.

Tot i així, emprant COCOMO com eina d'estimació, a partir dels resultats oferts per ohloh per Qantiqa, el projecte correspondria a:

- Número de línies: 34.099 línies.
- Estimació d'esforç: 8 anys-persona
- Sou (aproximat): 24.000€ (28.725$, canvi a data de 6 de Juny de 2010: 1,20$)
- Cost econòmic: 191.233€ (228.887$)

Aquesta estimació es pot consultar en línia a http://www.ohloh.net/p/qantiqa.

6.3 RESULTAT FINAL

El projecte entrega diferents productes:

- qantiqa-core: aplicació Java de consola, executable com dimoni. Incorpora un Jetty per fer de servidor HTTP i publicar el projecte qantiqa-api, en format WAR.
- qantiqa-higgs: aplicació Java per Google App Engine, gestionable mitjançant Play! però amb possibilitat de publicar-se a Internet.

6.4 CRONOGRAMA

El cronograma del projecte (Gantt) es troba en la pàgina següent.

6.4.1 *Desviacions respecte a la planificació inicial*

S'han respectat les dates d'entrega, a excepció de la PAC 3 per motius personals. La dedicació total al projecte ha estat la meitat prevista, fet que ha impedit assolir tots els objectius plasmats en l'anàlisi, però entregant un producte funcional.

Com que la dedicació ha estat molt influenciada per la vida personal, aquesta és molt aleatòria i caòtica, pel que no s'inclou el cronograma, ja que no seria pragmàtic reconstruir una activitat inconsistent. Es podria fer a partir dels registres del repositori de codi, però es pot afirmar que el 70% de les hores s'ha destinat a la programació, mentre que la resta s'ha destinat a la elaboració de la documentació.

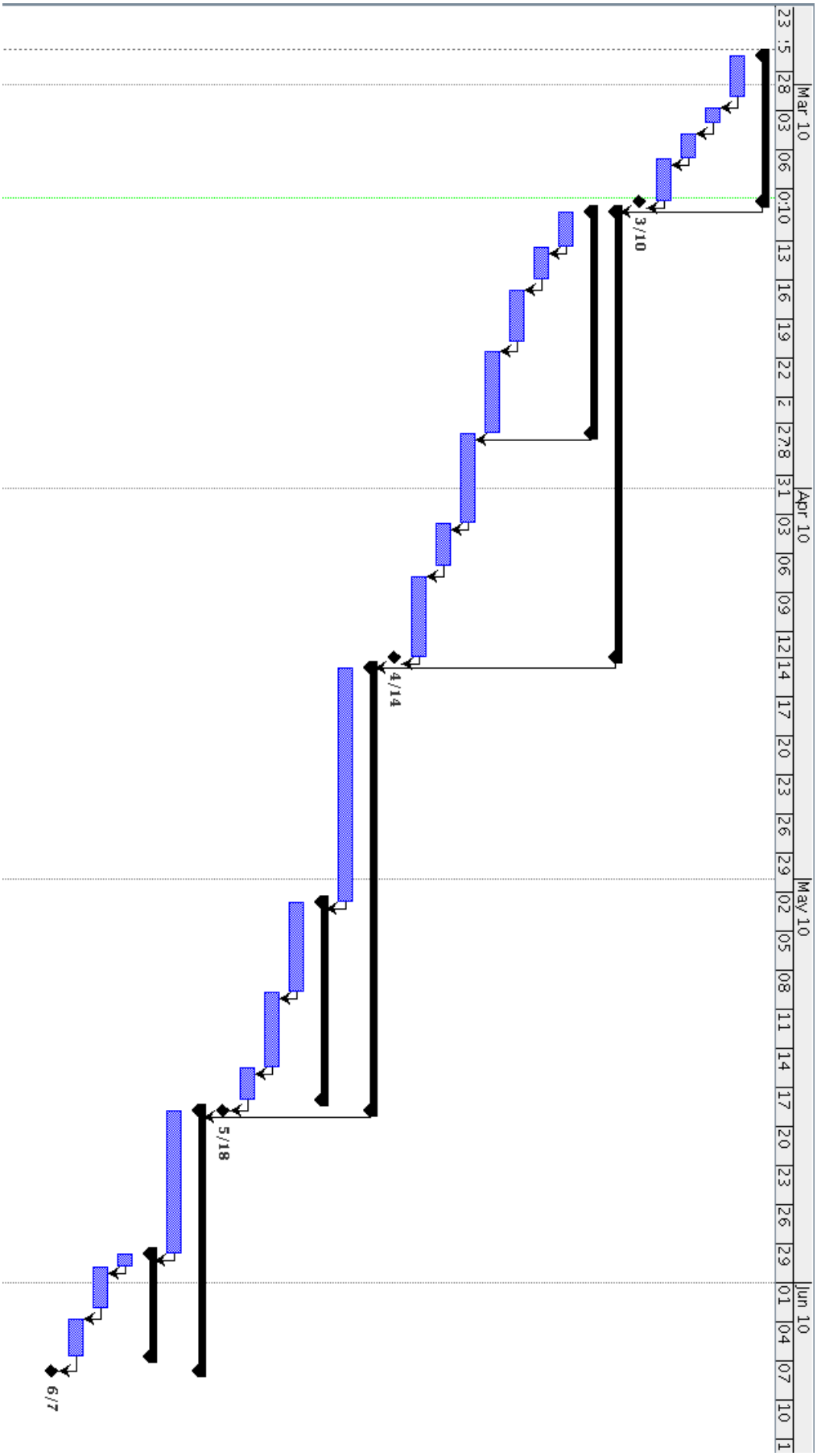

Figura 1: Cronograma

Part III

ANÀLISI I DISSENY

7 ANÀLISI I DISSENY

L'anàlisi del projecte es centrarà en quatre àrees principals, les quals comprenen la totalitat del mateix:

- API REST del node: oferirà la funcionalitat pel client.
- Autenticació
- Sistema d'identitat, reputació i retribució (antispam)
- Cerca

És important tenir en compte que partim d'eines ja creades, com la part client, pel que es modelarà la part servidora, així com la interacció amb el client i la xarxa distribuïda.

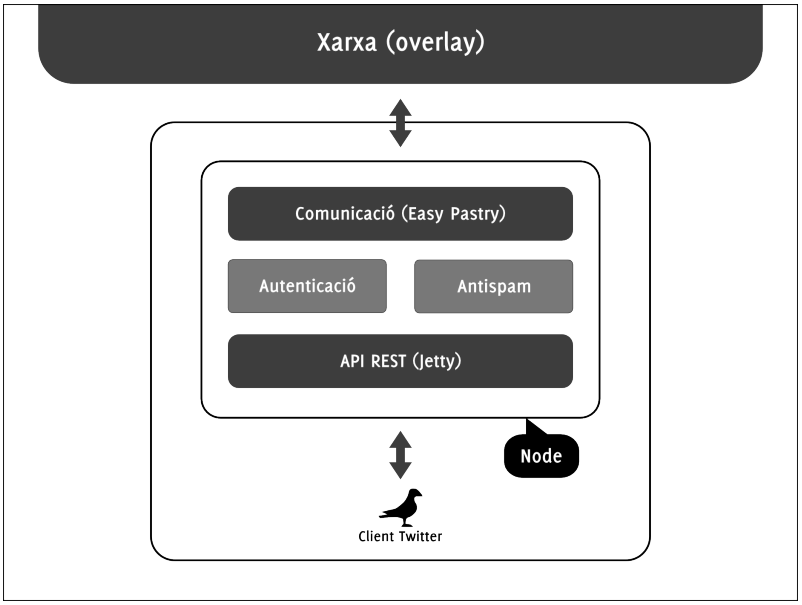

Figura 2: Esquema d'arquitectura

En el detall dels punts anteriorment esmentats relacionarem aquests amb els casos d'ús descrits en la recollida de requisits.

7.1 API REST

Ens centrarem en las funcions de l'API emprades per Buzzbird, com a primera fase del projecte, garantint tota la funcionalitat del mateix.

Aquesta es centra al voltant dels següents mètodes, amb els paràmetres suportats indicats, ja que són els que empra Buzzbird, el client de referència escollit, i amb format de resposta JSON:

- Mètodes de cerca
  - search

    q — El text a cercar.

- Mètodes de línia de temps
  - statuses/home_timeline

    since_id — Retorna només els estatus (quarks) amb ID més gran que l'especificat.

    count — Especifica el número de quarks a recuperar. No pot ser més gran de 200.
  - statuses/mentions

    count — Especifica el número de quarks a recuperar. No pot ser més gran de 200.
  - statuses/user_timeline

    count — Especifica el número de quarks a recuperar. No pot ser més gran de 200.
  - statuses/retweeted_by_me
- Mètodes d'estatus
  - statuses/show

    id — El ID numèric del quark a recuperar.
  - statuses/update

    status — El text del teu quark. Necessita estar codificat en format URL. Els quarks majors de 140 caràcters provocaran un error 403. Els missatges duplicats respecte l'últim missatge de l'usuari també provocaran un error 403. Quan es produeixi un d'aquests casos, el cos de la resposta contindrà el motiu.

    in_reply_to_status_id — ID d'un quark existent, del que aquest és una resposta.[1]

    source — Nom de l'aplicació.
  - statuses/destroy

    id — El ID numèric del quark a eliminar.
  - statuses/retweet

    id — El ID numèric del quark a reenviar.

    source — Nom de l'aplicació.
- Mètodes d'usuari
  - users/show

    id — ID numèric o nom d'un usuari.

1 Serà ignorat si en el quark no s'especifica el nom de l'usuari del quark indicat en el text del que estem enviant.

- users/search

  q — Consulta a realitzar contra la cerca d'usuaris.

- Mètodes de missatges directes (quarks privats)
  - direct_messages

    count — Especifica el número de quarks a recuperar. No pot ser més gran de 200.

- Mètodes d'amistat
  - friendships/create

    id — ID numèric o nom d'un usuari.
  - friendships/destroy

    id — ID numèric o nom d'un usuari.
  - friendships/show

    source_id — ID numèric d'un usuari (és excloent amb el paràmetre source_id).

    source_screen_name — Nom d'un usuari (és excloent amb el paràmetre source_screen_name).

    target_id — ID numèric d'un usuari (és excloent amb el paràmetre target_id).

    target_screen_name — Nom d'un usuari (és excloent amb el paràmetre target_screen_name).

- Mètodes de compte
  - account/verify_credentials
  - account/profile_image[2]

    screen_name — Nom d'un usuari.

- Mètodes de preferits
  - favorites/create

    id — El ID numèric del quark a marcar com preferit.

  Totes les referències a tweet no seran substituïdes per quark a efectes de minimitzar els canvis necessaris al client. Tot i així, s'oferirà URLs alternatives amb la nomenclatura pròpia de Qantiqa per tal d'oferir una API coherent.

La resta de mètodes de l'API REST es deixen per una futura implementació, degut a la naturalesa open source del projecte.

2 No es tracta d'un mètode de l'API, però és necessari pels avatars dels usuaris.

7.1.1 *Casos d'ús*

Arribats a aquest punt, podem fer una relació un a un entre les mètodes exposats i un cas d'ús, però hi ha funcionalitats descrites que no s'empren de forma directa o individual.

Per això, partint de la llista de l'apartat 3.2 i els mètodes de l'API, la llista de casos d'ús a tractar queda així:

Enviar quark Publicar un nou quark, ja sigui original, un requark o una resposta a un altre quark, però sempre públic.

Enviar quark privat Enviar un quark privat a un usuari determinat de la xarxa.

Seguir usuari Afegir-se a la llista de followers d'un usuari. Si és recíproc, els dos usuaris es consideren friends.

Deixar de seguir usuari Esborrar-se de la llista de followers d'un usuari.

Marcar quark Marcar quark com preferit.

Reenviar quark Fer-se ressò d'un quark enviat per un altre usuari.

Esborrar quark Eliminar un quark creat per l'usuari.

Llegir quarks Llegir l'stream[3] de quarks, amb mencions, missatges privats i retweets/requarks.

Els casos d'ús *Identificar usuari*, *Cercar usuari*, *Cercar quarks* i *Reportar usuari* corresponen a les àrees següents.

Enviar quark i enviar quark privat

Ambdós casos d'ús són idèntics, però difereixen en un punt: la forma de dirigir el missatge. Això es veurà en la implementació, però un serà un missatge directe i l'altre un missatge dirigit al node del client, propagant-se als veïns.

Tot i aquesta diferència, considerem que el flux és el mateix i amb un únic diagrama es descriu ambdós casos d'ús.

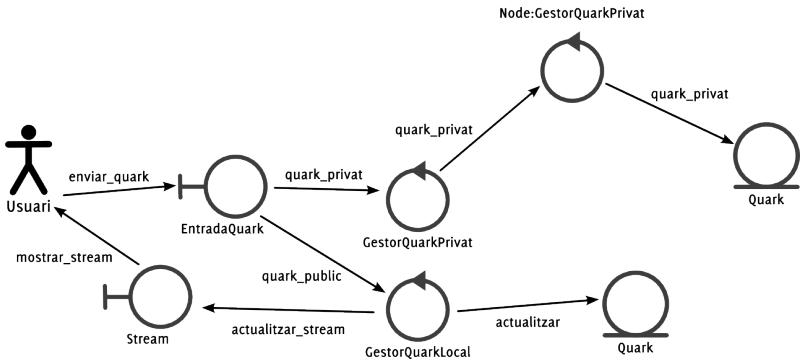

Figura 3: Cas d'ús "Enviar quark i enviar quark privat"

Es corresponen al mètode de l'API REST *statuses/update*.

3 També anomenat línia temporal o timeline.

Seguir i deixar de seguir usuari

Ambdós casos és troben totalment relacionats, tractant-se de casos excloents: o es segueix algú o no es segueix. Per això, també unifiquem el diagrama.

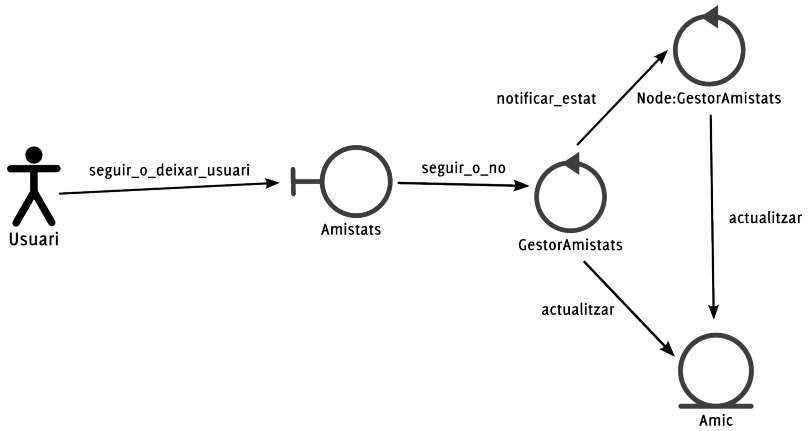

Figura 4: Cas d'ús "Seguir i deixar de seguir usuari"

Es corresponen als mètodes:

- *friendships/create*
- *friendships/destroy*

Marcar quark

Aquest cas és simple, corresponent al mètode *favorites/create*.

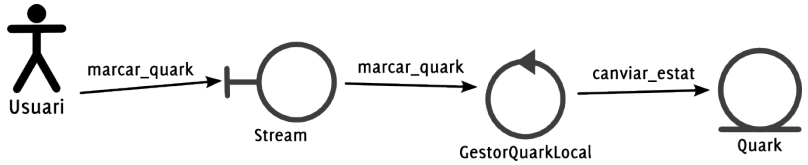

Figura 5: Cas d'ús "Marcar quark"

Reenviar quark

Es correspon al mètode *statuses/retweet*. És un mecanisme natural de Twitter que consisteix en tornar a publicar el contingut d'un estatus per tal de promocionar-ho als nostres followers.

Esborrar quark

Correspon al mètode *statuses/destroy*.

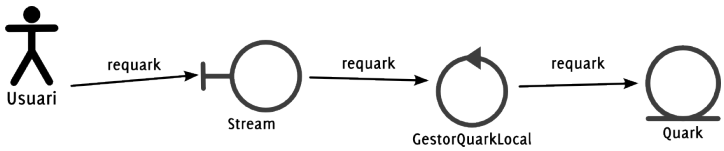

Figura 6: Cas d'ús "Reenviar quark"

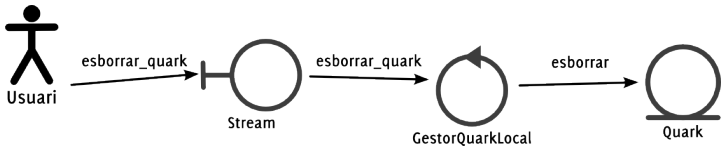

Figura 7: Cas d'ús "Esborrar quark"

Llegir quarks

Tot usuari té un stream de quarks que vol llegir, composat pels diferents tipus de quarks que ja hem vist en aquest document. Per això, aquest cas d'ús empra diferents mètodes de l'API:

- *statuses/home_timeline*
- *statuses/mentions*
- *statuses/user_timeline*
- *statuses/retweeted_by_me*
- *direct_messages*

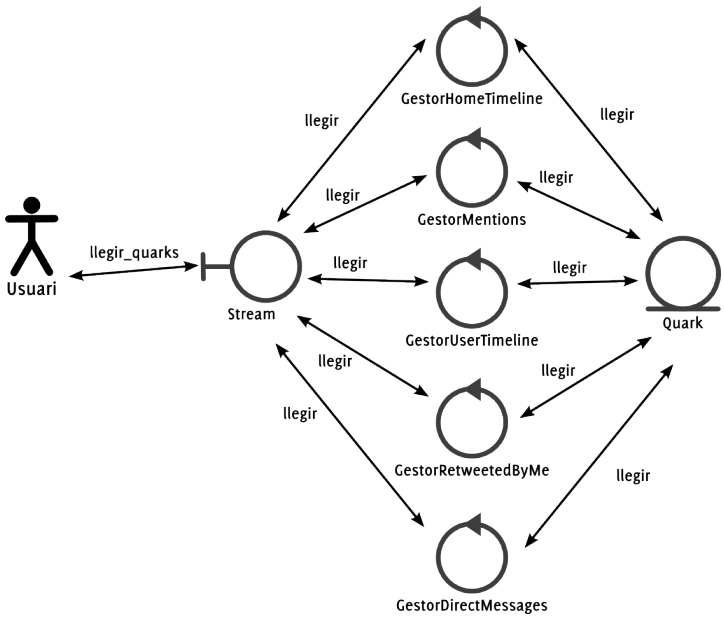

Figura 8: Cas d'ús "Llegir quarks"

7.2 AUTENTICACIÓ

Twitter ofereix dos tipus d'autenticació:

- Autenticació bàsica HTTP (usuari i contrasenya a la petició HTTP)
- Autenticació mitjançant OAuth

Per senzillesa i per cobrir les necessitats de la majoria de clients, Qantiqa implementarà l'autenticació bàsica HTTP, deixant el suport per OAuth per una futura implementació.

En el model client-servidor, l'autenticació es realitza a nivell del servidor, després de rebre les credencials corresponents al client. Però en les xarxes distribuïdes no disposem, a priori, de cap servidor central que realitzi aquesta funció, tornat molt difícil l'autenticació.

Així, en el paradigma distribuït, hem de recórrer als supernodes, és a dir, alguns nodes de l'overlay tindran funcionalitats especials.

7.2.1 *Supernodes*

Però no podem admetre qualsevol node com supernode, també anomenats gluons, ja que obriríem la porta a crackers[4] que accedissin a la xarxa amb versions modificades de Qantiqa i es fessin amb les credencials dels usuaris, alterant el contingut dels quarks, etc.

En el punt anterior deixem entreveure que una altra de les funcions dels gluons serà la de punt d'entrada a la xarxa. La naturalesa de Pastry ja fa necessari que un nou usuari entri a l'overlay a partir d'un node membre, pel que té sentit proporcionar els mecanismes d'autenticació en els gluons.

Per minimitzar les possibilitats de gluons modificats, hi haurà un webservice central (Higgs) situat a Google App Engine per l'autorització de gluons i la distribució de la llista oficial, aprofitant l'alta disponibilitat i seguretat de Google.

7.2.2 *Funcionament*

Tornant a l'usuari, quan aquest contacti un gluon, s'identificarà amb un nom d'usuari i usuari, el qual serà verificat. Si el nom ja estigués en ús, serà rebutjat.

En cas contrari, s'emmagatzemarà fora de l'overlay, en local al gluon i es replicarà cap als altres. La contrasenya es mantindrà en MD5 per evitar que algú les pugui capturar.

Aquesta manera de funcionar és més segura que la feta servir per Twitter amb la seva autenticació HTTP bàsica, que simplement concatena el nom d'usuari i la contrasenya separats pel caràcter ':' i ho codifica en Base64.

7.2.3 *Assumpcions*

En tot el presentat anteriorment estem assumint certs "dogmes", els quals descrivim a continuació:

4 Segons la definició del Jargon File.

1. La seguretat i l'alta disponibilitat dels datacenters de Google.
2. El baix percentatge de col·lisions de claus en els algoritmes MD5 i SHA-1, així com la dificultat d'esbrinar/generar les claus.
3. Els gluons no guarden cap mena d'auditoria dels accessos dels usuaris.
4. El codi font del client sempre romandrà lliure, però tota modificació enviada al repositori oficial serà revisada.

7.2.4 *Arquitectura*

Finalment, per l'autenticació i altres funcions, la xarxa presentarà una estructura com la del següent esquema:

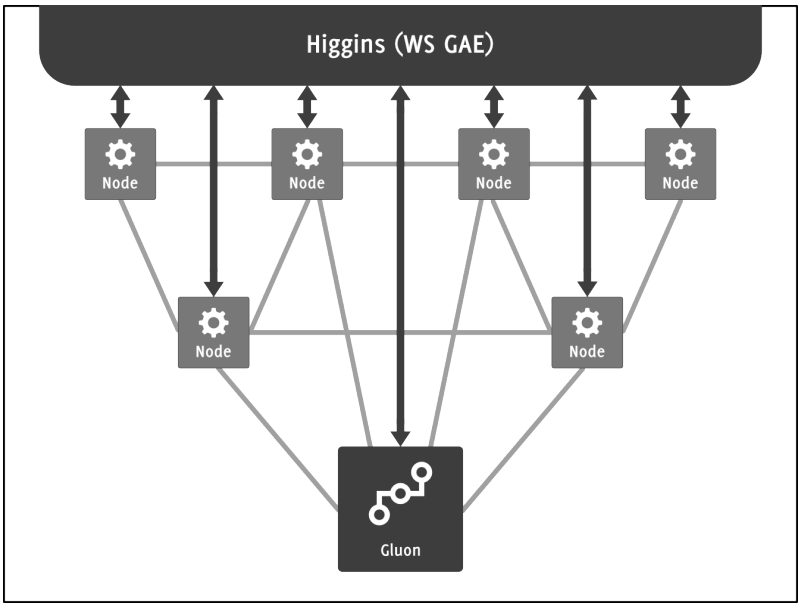

Figura 9: Arquitectura de la xarxa

7.2.5 *Casos d'ús*

Aquest àrea només té un cas d'ús, *Identificar usuari.* Com ja hem descrit, l'usuari s'identificarà mitjançant un login, amb el seu usuari i password[5].

Aquest operativa es correspon al mètode de l'API *account/verify_credentials*.

5 Segons l'eina, l'usuari podrà guardar les seves credencials.

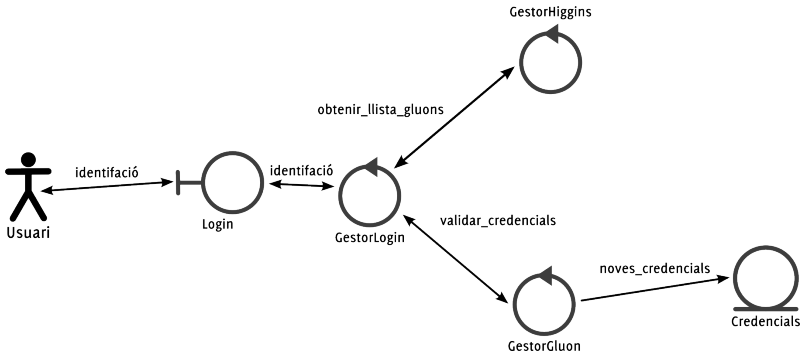

Figura 10: Cas d'ús "Identificar usuari"

7.3 ANTISPAM

NOTA: Funcionalitat no present en el producte final.

En tot sistema de comunicació poden aparèixer dos fenòmens habituals:

- Spam
- Trolls

A aquesta àrea seria més adequat dir-li "Control de plagues", però, ja que no hi ha cap sistema antitrolls, el terme més aproxima és el que encapçala aquesta secció.

Per spam comprenem tot aquell contingut que pretén vendre-nos quelcom o dirigir-nos a contingut enganyós o amb finalitats malicioses.

Però que entenem per trolls? Tothom s'ha trobat alguna vegada a una persona que cerca provocar als altres, propiciant discussions, ja sigui mitjançant l'insult o la reiteració de temes. Ara ja podeu imaginar que aquesta persona és un troll.

Davant d'aquests dos fenòmens s'han desenvolupat moltes solucions, englobades dins del software antispam, des dels filtres per e-mail[6] fins filtres per blogs[7].

Com que els trolls són difícilment controlables automàticament, també s'han desenvolupat molts mètodes de reputació. Hi ha diferents models, però tots es basen en la votació dels usuaris, tant en positiu com en negatiu. Alguns afegeixen un factor anomenat "karma" que s'ajusta segons els vots rebuts dins d'unes condicions.

En l'anterior punt hem introduït el concepte "reputació". Abans de continuar amb aquest, descriurem aquest i altres conceptes esmentats a l'apartat 2.4 a la pàgina 4.

7.3.1 *Els tres pilars*

Tot sistema antispam amb caire social es basa en tres conceptes fonamentals:

1. Identitat
2. Reputació

6 SpamAssassin, llistes negres, etc.
7 Akismet, filtres "HOYGAN", etc.

3. Retribució

La identitat ve determinada per l'ús de comptes registrats en el sistema, és a dir, cada aportació de l'usuari va identificada pel seu nom, lligat de forma indissoluble i consultable.

Amb la reputació entra en joc el sistema de votacions comentat prèviament, el qual és regulat per la comunitat sense intervenció d'administradors.

I la retribució és conseqüència del sistema de votacions, aplicat sobre la reputació de l'usuari. Tothom és igual en la xarxa i serà retribuït en proporció a la seva actitud. Si algú es dedica a molestar, la comunitat el votarà negatiu i patirà les conseqüències.

7.3.2 *Model optat*

Qantiqa opta per un model senzill de reputació basat en votacions a nivell de cada quark, la retribució del qual serà afegir un retràs sistemàtic en la propagació dels missatges del node associat a l'usuari.

Per la identitat ja s'ha descrit el sistema que la garantirà, completant els tres pilars bàsics que conformen el sistema antispam.

7.3.3 *Casos d'ús*

Aquesta àrea, relacionada amb l'API REST, ja que aquest sistema repercuteix en la propagació dels missatges, consta d'un únic cas d'ús, *Reportar usuari:*

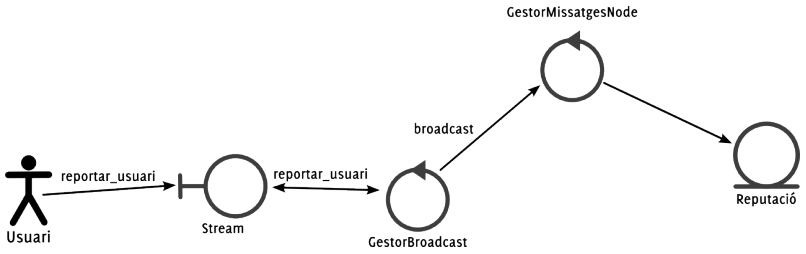

Figura 11: Cas d'ús "Reportar usuari"

En línies generals, quan un usuari reporta un altre, es llença un missatge de broadcast a tots els nodes possibles per tal que actuïn en conseqüència, segons l'algoritme que s'implementi per la propagació de missatges.

7.4 CERCA

La cerca és essencial per tal de socialitzar l'ús de Qantiqa, ja que sense la possibilitat de trobar quarks sobre temàtiques afins, no tindrem manera de trobar usuaris per seguir la seva activitat en línia.

Per això, treballarem amb un DHT d'índex, separant el contingut dels quarks en diferents paraules per emprar-les com clau d'aquest DHT. Si

ja existís alguna de les claus, s'actualitzarà afegint un identificador de quark més.

A l'hora de fer la cerca, el nostre node consultarà al DHT per cada paraula introduïda en la cerca i combinarà els resultats segons rellevància. Per fer-ho senzill, primer anirien els que continguin totes les paraules, després els que en continguin n-1 i successivament.

7.4.1 *Casos d'ús*

Les possibilitats de cerca es divideixen en dos grans grups:

- D'usuari.
- I de quarks.

Això es tradueix en dos casos d'ús addicionals, *Cercar usuari* i *Cercar quarks*, els quals unim en el següent diagrama, ja que la seva interfície serà la mateixa, seguint les pràctiques pròpies de Twitter.

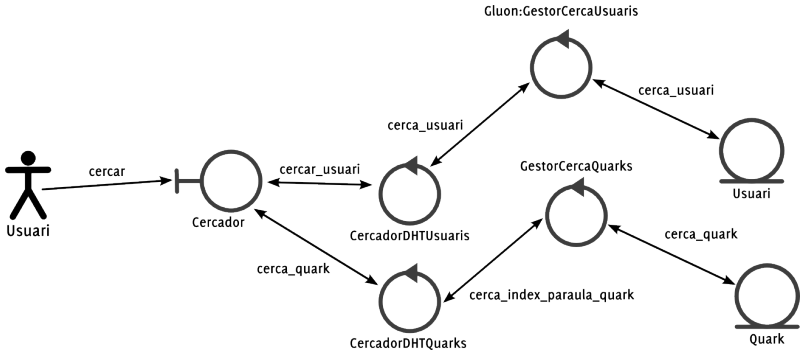

Figura 12: Cas d'ús "Cerques"

Per cercar quarks entenem el realitzar una cerca per paraules clau i obtenir una sèrie de quarks que continguin aquestes paraules.

El cas dels usuaris es realitzarà consultant contra un gluon, ja que són ells qui tenen les llistes d'usuaris actius de l'overlay.

Finalment, dins dels quarks es poden donar paraules amb significat especial, ja descrites en l'apartat 3.1 a la pàgina 5 i que es tindran en compte. Així tindrem els següents tipus de consulta per quarks:

- Contenint una o més paraules: *qantiqa*.
- D'un usuari: *from@im_dario*.
- Responent a un usuari, el quark comença per @usuari: *to@im_dario*.
- Mencionant un usuari, el quark conté @usuari: *@im_dario*.
- Contenint un hashtag: *#hashtag*.
- Originats per una aplicació determinada: *cerca+source:aplicació*.

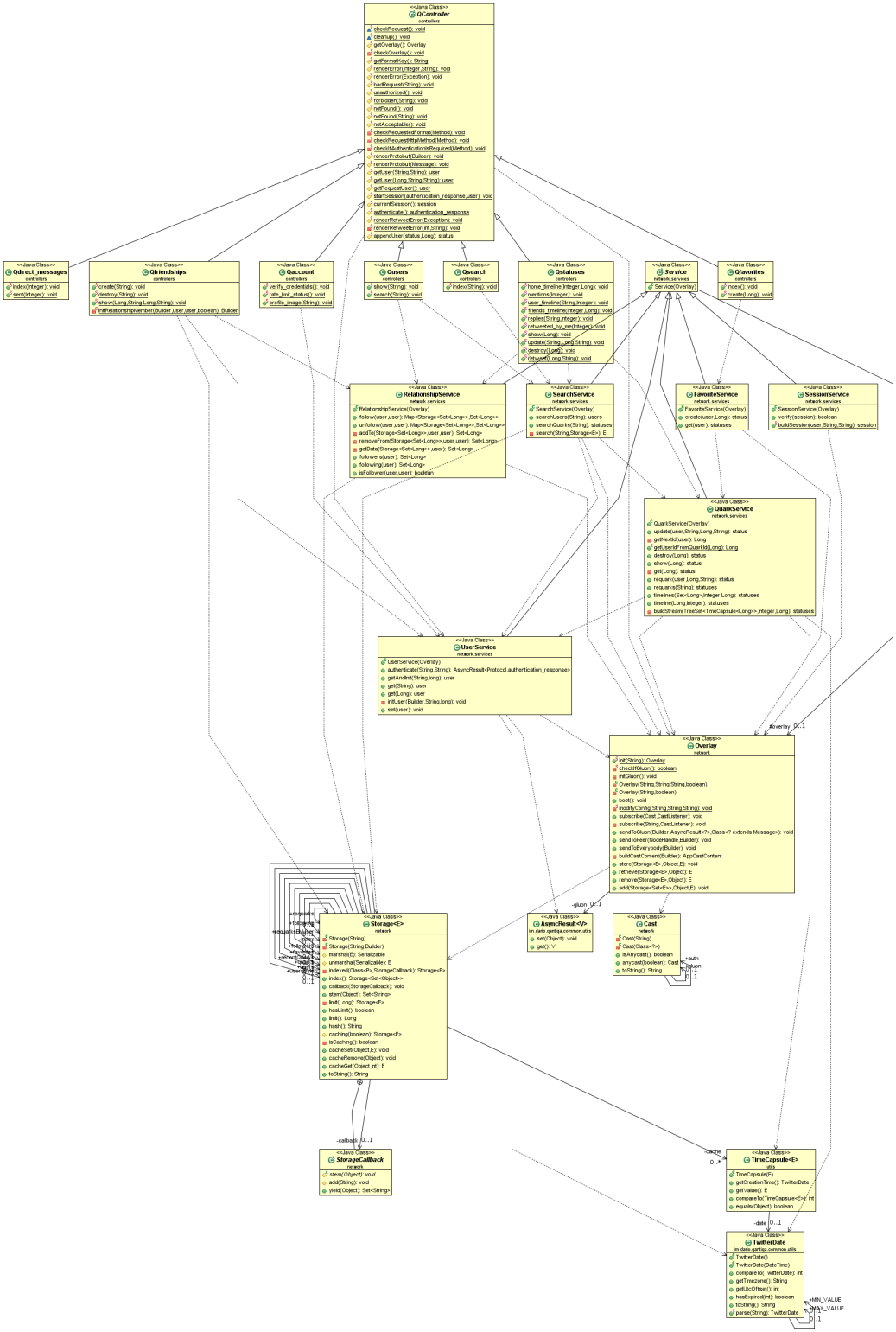

Figura 13: Diagrama de classes

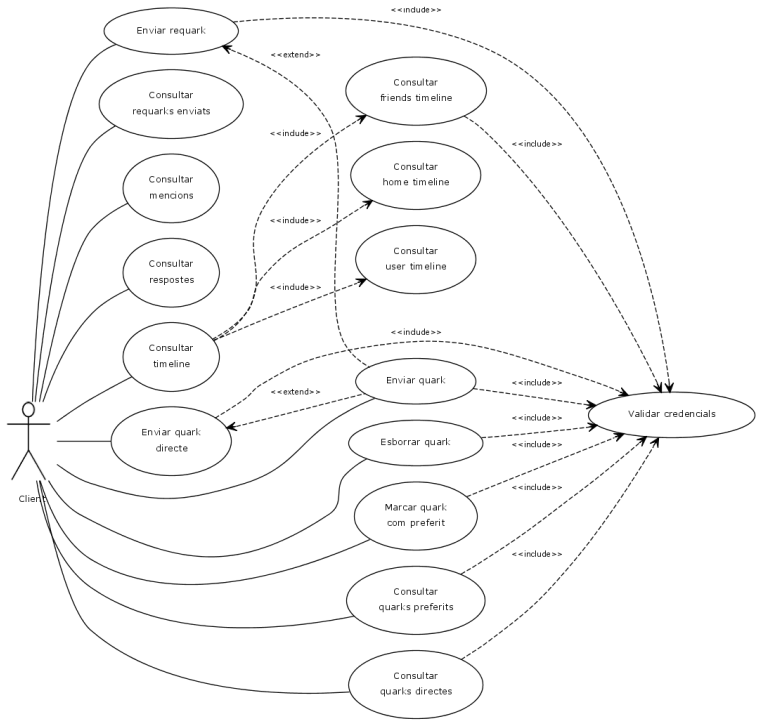

Figura 14: Casos d'ús centrats en els quarks

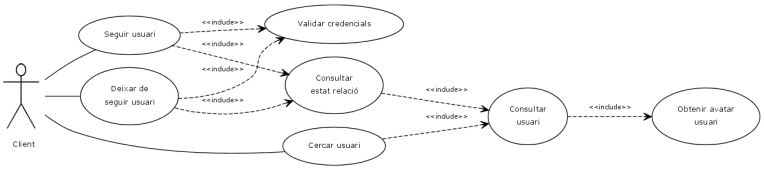

Figura 15: Casos d'ús centrats en altres activitats dels usuaris

Part IV

IMPLEMENTACIÓ

8 IMPLEMENTACIÓ

8.1 IMPLEMENTACIÓ

A nivell tècnic, la implementació de Qantiqa no ha patit de masses problemes, excepte per un parell.

Un de petit al fer servir classes que hereten de CastContent. Finalment s'ha optat per fer una classe "wrapper" al voltant de CastContent per passar els missatges protobuf per l'overlay.

L'altre és que Higgs sempre retorna identificadors múltiples de dos quan emmagatzema usuaris per futura referència.

Però, en general, les eines, biblioteques i frameworks emprats han funcionat bé, sense haver necessitat de fer "workaround" en el codi dels mateixos.

8.1.1 *Guia de desenvolupament*

En aquest apartat es detalla, pas a pas, com configurar l'entorn de desenvolupament per fer funcionar Qantiqa.

Dependències (no gestionades per Maven)

El projecte depèn dels següents frameworks/eines:

- JDK 6 (Java(TM) SE Runtime Environment (build 1.6.0_20-b02)):
  - Open SuSE: http://en.opensuse.org/Installing_Sun's_Java_on_SUSE_Linux
  - Debian (non-free)/Ubuntu (Universe): apt-get install sun-java6-jdk
  - Arch Linux: pacman -S jdk
  - Genèric: també es pot baixar i instal·lar amb un .bin o .rpm (per sistemes Red Hat).
    - Fedora: http://www.mjmwired.net/resources/mjm-fedora-f8.html#java
- Maven (versió 2.2.1): es pot descarregar i descomprimir o emprar els paquets per cada distribució o sistema operatiu:
  - Open SuSE: http://software.opensuse.org/search?baseproject=ALL&p=1&q=maven
  - Debian/Ubuntu: apt-get install maven2
  - Arch Linux: pacman -S maven
- Play! framework (versió 1.0.1): només cal descarregar i descomprimir en un directori del home. Requereix Python instal·lat (versió 2.6.x).

Procediment

1. Descarregar codi del repositori SVN de Google Code.
2. Modificar $PATH per incloure el directori de Play!, per exemple, PLAY_HOME=~/software/play-1.0.1.
3. Instal·lar Play! en repositori Maven local, executant script "$QANTIQA_HOME/mvn_install_play_libs.sh $PLAY_HOME". $QANTIQA_HOME és el directori on s'ha descarregat el repositori SVN de Google Code. L'script genera les instruccions que s'han d'executar, es recomana redirigir la sortida a un fitxer i executar l'script resultant.
4. Amb Maven, instal·lar projectes al repositori local amb "cd $QANTIQA_HOME && mvn clean install"
5. Configurar qantiqa.ip a $QANTIQA_HOME/qantiqa-api/conf/application.conf amb la IP en la que es publicarà el port d'EasyPastry.
6. Importar $QANTIQA_HOME a Eclipse, important cada projecte inclòs. Són cinc projectes en total:
   a) qantiqa: contenidor de tots els projectes.
   b) qantiqa-api: nucli de l'aplicació (API).
   c) qantiqa-common: classes comunes a tots els projectes.
   d) qantiqa-core: projecte d'integracio de Qantiqa amb Jetty.
   e) qantiqa-higgs: webservice de gestió de gluons i usuaris.
7. A Eclipse, revisar variable de classpath M2_REPO a Window > Preferences > Java > Build Path > Classpath Variables. Ha de contenir la ruta al repositori local Maven (~/.m2/repository).
8. Arrencar Higgs i configurar:
   a) play run qantiqa-higgs
   b) Configurar gluon local (new gluon) a Higgs, amb secret extret de qantiqa-api/conf/application.conf
9. Per fer funcionar qantiqa-core (amb qantiqa-higgs arrencat sota Play!):
   a) cp ~/.m2/repository/im/dario/qantiqa/common/qantiqa-common/1.1-SNAPSHOT/qantiqa-common-1.1-SNAPSHOT.jar qantiqa-api/lib/
   b) play war qantiqa-api -o qantiqa-core/target/classes/war
   c) Si es vol emprar un client Twitter (provat amb Buzzbird i Spaz), es pot modificar /etc/hosts per redirigir les peticions a Twitter cap a Qantiqa amb:
      i. 127.0.0.1 twitter.com
      ii. 127.0.0.1 www.twitter.com
      iii. 127.0.0.1 search.twitter.com

d) Nota: sota qantiqa-core es publica la API pels ports 8080 i 8443, pel que es pot emprar xinetd per redirigir les peticions dels clients Twitter a Qantiqa, seguint les instruccions oficials de Jetty.

10. Per provar-ho sota Play!:
    a) play run qantiqa-higgs
    b) play run qantiqa-api
    c) Nota: es pot connectar amb Eclipse per depurar el codi als ports 8080 (API) i 8081 (Higgs).
    d) Nota: per fer proves, es pot emprar "curl -v -u usuari:password URL".

8.2 JOC DE PROVES

Les proves de Qantiqa empren el sistema de test unitaris integrats en Play! i, per tant, es necessita iniciar el projecte qantiqa-api amb "play test qantiqa-api".

Un cop arrencat, sempre amb Higgs en marxa també, ens dirigim a http://127.0.0.1:8080/@tests. Els tests també es poden executar des d'Eclipse, ja que són compatibles amb JUnit.

Els tests disponibles corresponen als casos d'ús descrits en aquesta memòria, tot i que alguns estan separats per evitar que la seva execució es faci en un ordre incorrecte:

ACCOUNTTEST Prova la verificació de credencials (usuari/password) i obtenir l'avatar de l'usuari.

DIRECTMESSAGESTEST Prova de la existència de la API per missatges directes, tot i que no està implementada.

FAVORITESTEST Prova els preferits del sistema, tant la seva obtenció com creació. La prova show() requereix que s'hagi enviat un quark (StatusesTest).

FRIENDSHIPSTEST Proves sobre les relacions entre usuaris. Hi ha la variant Destroy que permet provar si es pot esborrar una relació, ja que si es deixava en el test original s'executava abans de la creació.

MISCTEST Proves vàries sense relació directa amb les funcionalitats de l'API.

SEARCHTEST Prova de cerques en l'overlay de Qantiqa.

STATUSESTEST Proves entorn dels quarks, amb la prova derivada d'esborrar-ne un (Destroy). És el nucli de l'aplicació i cobreix totes les funcionalitats implementades.

USERSTEST Proves de cerca d'usuaris i obtenció de perfil (relacionada amb l'AccountTest).

És possible que alguna de les proves falli a la primera, ja que el recurs sol·licitat no s'ha creat, a l'espera de llençar el test adequat.

Part V

CONCLUSIÓ

9 CONCLUSIÓ

Qantiqa és un projecte ambiciós, amb moltes possibilitats d'ampliació en la direcció de les xarxes socials distribuïdes. Durant els mesos de desenvolupament han aparegut plataformes similars com Diaspora*, com resposta a les polítiques de privacitat de Facebook.

Sobre les xarxes socials actuals, Twitter és la més simple en funcionament, però fer Qantiqa m'ha demostrat que és molt més complexe del que sembla a simple vista. Tot i així, el seu èxit és extraordinari, pel que està per quedar-se durant molt de temps.

Per això i les circumstàncies personals durant el semestre, aquest projecte ha resultat ser d'una magnitut més gran de la que vaig preveure, però igual d'interessant i tot un repte, però complicat d'assolir totalment.

Així, apart de l'experiència adquirida i l'experimentació amb aplicacions descentralitzades, puc afirmar que el futur d'Internet passa per un funcionament pròxim al que he implementat, amb les dades de l'usuari en la seva màquina, i potser en la dels seus contactes pel que fa a la seva informació pública, així com un enfoc més concentrat en construir APIs que permetin crear noves funcionalitats.

10 GLOSSARI

QUARK Missatge públic, equivalent al tweet de Twitter.

REQUARK Reenviament d'un quark als seguidors de l'usuari qui ho efectua. Equivalent al retweet de Twitter.

HASHTAG Cadena de text emprada per agrupar quarks.

Rubén Mondéjar Andreu, Carles Pairot Gavaldà, and Pedro García López. EasyPastry : Middleware Abstraction for Structured P2P Development. http://ast-deim.urv.cat/easypastry/, Març 2009. [Online; 2010-03-07].

Michael Arrington. Twitter Can Be Liberated - Here's How. http://techcrunch.com/2008/05/05/twitter-can-be-liberated-heres-how/, Maig 2008. [Online; 2010-03-07].

Microformats. Twitter Syntax. http://microformats.org/wiki/twitter-syntax, Octubre 2009. [Online; 2010-03-08].

Microsyntax.org. Microsyntax.org Proposals. http://www.microsyntax.org/proposals. [Online; 2010-03-08].

E. Prodromou, B. Vibber, , J. Walker, and Z. Copley. OStatus 1.0 Draft 1. http://ostatus.org/sites/default/files/ostatus-1.0-draft-1-specification.html, Març 2010. [Online; 2010-03-07].

Zed Shaw. Utu: Saving The Internet (with hate). http://zedshaw.railsmachina.com/project.html, Febrer 2008. [Online; 2010-03-08].

Twitter, Inc. API Wiki. http://apiwiki.twitter.com/. [Online; 2010-03-07].

Wikipedia. Twitter — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/wiki/Twitter, Març 2010. [Online; 2010-03-07].