max Documentation

Release 3.6

UPCnet, SLU

May 24, 2014

Contents

MAX és un sistema de recollida i visualització de registres d'activitat generada per usuaris i aplicacions així com les interactuacions entre usuaris i entre usuaris i aplicacions.

El MAX registra l'activitat de dues maneres: Activa i passiva. Quan ho fa de manera passiva, els usuaris i aplicacions que generen activitat, usen l'API del MAX per registrar-hi activitat. Quan ho fa de manera activa, el MAX va a buscar l'activitat al sistema que la genera a través d'unes regles predefinides, per exemple, l'activitat generada per un compte de Twitter i un hashtag associat.

Tant els usuaris com les aplicacions que puguin interactuar amb el MAX poden ser d'origen en escenaris corporatius (interns) o en canvi ser totalment externs al sistema.

CHAPTER 1

Continguts

1.1 Instal·lació

En aquest apartat s'indiquen totes les passes per tal de posar en marxa una instància de tota la pila de software i serveis necessaris per al funcionament d'una instància del servei MAX.

1.1.1 Requisits

Cal satisfer una sèrie de requisits previs que es detallen a continuació.

1.1.2 Sistema operatiu

Actualment, degut a limitacions tecnològiques, el entorn del projecte només pot executar-se sobre màquines *nix. En aquests documents suposarem una màquina basada en una distribució Ubuntu Linux 11.04 (Natty). Per la instal·lació per altres distribucions cal substituïr els noms dels paquets pels corresponents a la distribució d'instal·lació.

Software base i llibreries:

> GCC i eines de compilació Python 2.7 (altres versions inferiors no són compatibles, mentre que superiors (>3) no han estat testades) Python Distribute (a.k.a setuptools) Git Erlang, xsltproc, zip (per la compilació del motor de cues)

1.1.3 Instal·lació de software base i llibreries

Presuposem que instal·lem el MAX i el software relacionat amb ell en la carpeta /var/pyramid:

```
$ mkdir /var/pyramid
$ cd /var/pyramid
```

Instal·lem els següents paquets base:

```
$ sudo apt-get install wget build-essential git-core
```

Instal·lació de Python 2.7 (compilat), es recomana compilar-lo per partir d'una copia neta i sense llibreries de sistema innecessàries. Seguirem aquestes comandes per instal·lar-lo:

```
$ sudo apt-get install libreadline5-dev libsqlite3-dev zlib1g-dev libbz2-dev
$ wget http://python.org/ftp/python/2.7.2/Python-2.7.2.tgz
$ tar xvfz Python-2.7.2.tgz
```

```
$ cd Python-2.7.2
$ ./configure --prefix=/var/pyramid/python2.7
$ make
$ make install
```

Instal·lació de Python Distribute:

```
$ wget http://python-distribute.org/distribute_setup.py
$ sudo python2.7 distribute_setup.py
```

Instal·lació de Erlang i llibreries:

```
$ sudo apt-get install erlang xsltproc zip
```

1.1.4 Instal·lació de llibreries accés LDAP

És necessari les últimes llibreries de OpenLDAP de les distribucions corresponents, en cas d'Ubuntu:

```
$ apt-get install libsasl2-dev libldap2-dev libssl-dev
```

1.1.5 Instal·lació de MAX

El MAX utilitza un sistema de montatge (descàrrega, compilació i configuració) automàtic anomenat zc.buildout.

Aquest sistema utilitza un fitxer (buildout.cfg) de configuració que conté tota la informació necessària per a construir un entorn de MAX complet i configurat.

Per començar ens baixarem de Github el codi executant la següent ordre:

```
$ git clone git://github.com/UPCnet/maxserver.git
```

Fet això, ens ha d'haver creat un nou directori maxserver. Accedim a ell:

```
$ cd maxserver
```

Executem el muntatge amb aquestes ordres:

```
$ /var/pyramid/python2.7/bin/python bootstrap.py
$ ./bin/buildout
```

Esperem fins a que finalitzi. Un cop finalitzat, tindrem tots els serveis llestos per ser arrencats.

1.2 Configuració

1.2.1 Python

Cal definir la codificació del Python que estem utilitzant per defecte:

```
$ vi /var/pyramid/python2.7/lib/python2.7/sitecustomize.py
```

Afegir a aquest arxiu:

```
import sys
sys.setdefaultencoding('utf-8')
```

1.2.2 Gunicorn

Cal crear l'usuari **pyramid** al sistema, que serà l'usuari amb el que s'executi els processos de Gunicorn:

```
$ adduser pyramid
```

1.2.3 RabbitMQ

El servidor de cues s'ha de configurar creant un usuari amb el que farem les operacions, creant un vhost per les nostres cues i assignant-li els permissos necessaris a l'usuari sota el context d'aquest vhost:

```
$ ./bin/rabbitmq-server -detached
$ ./bin/rabbitmqctl add_user  ``username`` ``password``
$ ./bin/rabbitmqctl add_vhost ``nom_maquina``
$ ./bin/rabbitmqctl set_permissions -p ``nom_maquina`` ``username`` ".*" ".*" ".*"
```

1.2.4 Celery

El procés que s'ocupa de llençar els workers de les tasques de les cues (`celeryd`), es configura a través d'un módul de Python que es troba directament en el directori src/max/maxrules:

```
BROKER_HOST = "localhost"
BROKER_PORT = 5672
BROKER_USER = ""
BROKER_PASSWORD = ""
BROKER_VHOST = ""

CELERY_RESULT_BACKEND = "amqp"
CELERY_IMPORTS = ("maxrules.tasks",)

# MongoDB config
mongodb_url = "mongodb://localhost"
mongodb_db_name = "max"
```

També conté la configuració de la base de dades MongoDB ja que els workers han de poder accedir-hi.

1.2.5 Ajustos de sistema

Per tal d'optimitzar el sistema cal ajustar alguns paràmetres de funcionament.

Pujar el número de file descriptors (FDs):

```
$ vi /etc/security/limits.conf
```

i posar la següent configuració:

```
# /etc/security/limits.conf
#
root soft nofile 10000
root hard nofile 10000

# Nginx
www-data soft nofile 10000
www-data hard nofile 10000
```

```
# Gunicorn/Pyramid
pyramid soft nofile 10000
pyramid hard nofile 10000
```

La base de dades ha d'estar en una partició ext4.

(http://www.mongodb.org/display/DOCS/Production+Notes)

General Unix Notes

> Turn off atime for the data volume Set file descriptor limit and user process limit to 4k+ (see etc/limits and ulimit) Do not use large VM pages with Linux (more info) Use dmesg to see if box is behaving strangely Try to disable NUMA in your BIOS. If that is not possible see NUMA Minimize clock skew between your hosts by using ntp; linux distros usually include this by default, but check and install the ntpd package if it isn't already installed

1.3 Funcionalitats

Aquestes són les característiques bàsiques del MAX:

> API de serveis web RESTful Aquests serveis parlen JSON tant d'entrada com de sortida Compleix els estàndards definits pel protocol activitystrea.ms L'autenticació de l'API REST es fa a través d'un sistema d'autenticació basat en oAUTH 2.0 (draft 23) L'API e interfície web estan implementats en un framework lleuger i escalable (Pyramid) Cloud friendly, es poder de manera fàcil ser desplegat al núvol i per lo tant, És fàcilment atomitzable i es podet encapsular en màquines virtuals molt petites És escalable horitzontalment basant-se en les dues premises anteriors

En quant a funcionalitats pròpies com a fil d'activitat, el MAX suporta (a la versió 3.0):

> Afegir activitat Afegir comentaris a una activitat Subscriure's (follow) a un context (URL) Eliminar subscripcions Generació d'activitat sota un context (URL) Cerques per context i per usuari
>
> Seguir (follow) a usuaris i deixar de seguir a usuaris Compartir (share) una activitat Marcar activitat com a m'agrada (like) Suport per a la creació d'un perfil bàsic d'usuari en el que es guardarà informació relacionada amb l'usuari i les seves subscripcions

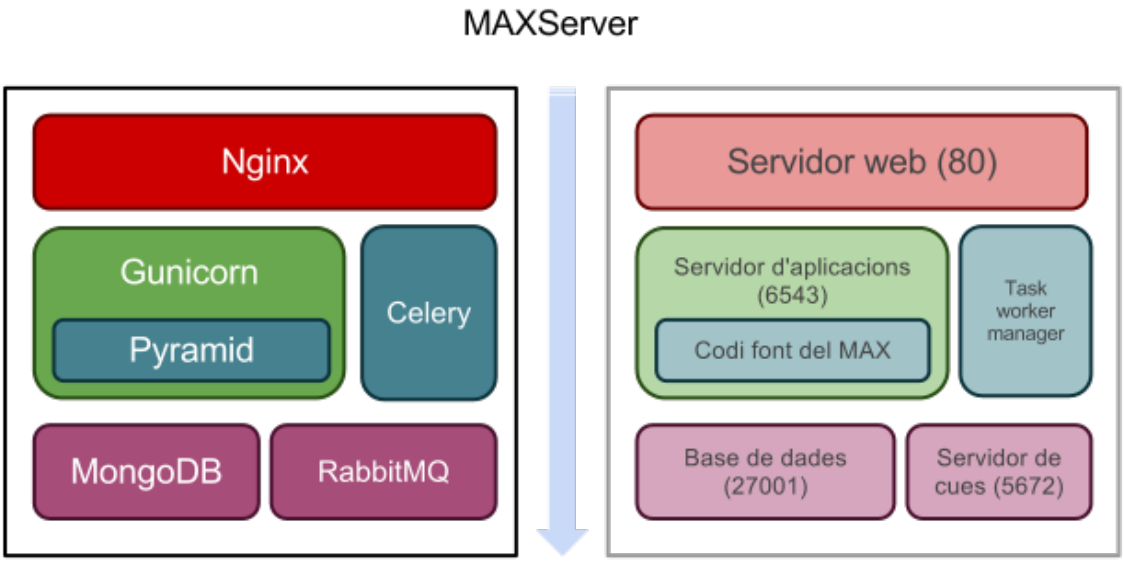
1.4 Arquitectura

Aquest document explica la proposta d'arquitectura del servei MAX (Motor d'activitat i subscripcions) que s'engloba dins del projecte som.UPC. El servei es composa de diferents capes que comprenen des del servidor web públic fins a la capa de persistència. A continuació passem a descriure cadascun d'ells i les seves funcions.

1.4.1 Nginx

Farem servir Nginx com a servidor web públic del servei. Per servir peticions aquest servidor utilitza l'aproximació asíncrona basada en events que proporciona un gran rendiment sota grans càrregues en contrast als servidors basats en fils de procés (threads) com Apache.

Serà la primera porta d'entrada al servei i proporcionarà robustesa a part de la capacitat d'absorbir grans quantitats de tràfic entrant.



1.4.2 Gunicorn

És el servidor d'aplicacions Python compatibles amb WSGI i el motor de la nostra aplicació. Està basat en un model semblant a l'utilitzat a Nginx, centrant-se amb donar un alt rendiment i ser molt lleuger.

1.4.3 Pyramid

El codi base del MAX està implementat amb el framework de desenvolupament Pyramid que està basat en Python. Aquest framework també te fama de ser molt lleuger (no supera les 5000 línies de codi) i optimitzat per tenir un alt rendiment i traça de memòria molt petita.

1.4.4 MongoDB

La capa de persistència està implementada amb una base de dades NoSQL anomenada MongoDB. Està optimitzada per a tindre un alt rendiment, no està lligada a la especificació prèvia d'un esquema i està orientada al document. Aquest documents que guarda són documents JSON. A més, aquesta base de dades permet la implementació molt senzilla d'alta disponibilitat i tolerància a fallides, així com balanceig de càrrega entre diferents nodes.

1.4.5 Celery

XXX

1.4.6 RabbitMQ

XX

1.4.7 Escalabilitat horitzontal

L'arquitectura proposada i gràcies a les funcionalitats de clusterització de la MongoDB és extremadament escalable. A més, permet l'encapsulament d'aquesta en màquines virtuals molt petites i en cas de necessitat del servei, es poden aprovisionar tantes com es vulguin per poder dotar al servei de més potència de càrrega i augmentar el seu rendiment.

1.5 Introducció a conceptes i convencions

En aquest document introduïem alguns conceptes relacionats amb l'arquitectura REST, orientats a entendre la implementació dels serveis al MAX. Aquí s'expliquen conceptes que o bé son comuns a tots els serveis, o ajuden a entendre la filosofia i la estructura darrera d'aquests.

1.5.1 Conceptes previs

La api REST està organitzada en recursos, identificats per una URI concreta. Cada recurs és capaç de subministrar diferents respostes en funció amb el mètode que utilitzem per accedir a ell. En termes generals, cada recurs exposa 4 mètodes que implementen el que es coneix com a UI (Uniform interface): GET, POST, PUT i DELETE. Aquests mètodes és poden interpretar sota l'acrònim que s'usa normalment de CRUD (Create, Read, Update, Delete), sent la correspondència la següent:

GET Llegir element(s) d'un recurs

HEAD Informació sobre un recurs.

POST Crear un element dins un recurs

PUT Modificar un element existent d'un recurs

DELETE Eliminar un element d'un recurs

Els recursos poden imlemenetar un o diversos dels mètodes de la UI, segons apliqui per el significat de cada recurs, complint sempre amb el significat genèric del mètode utilitzat. Els diferents serveis que hi han implementats, a part de la UI, comparteixen la forma en que expressen el resultat. Hi ha diferents codis d'estat , segons els estàndards HTTP, que es retornen com a codi d'estat en la resposta del servei, juntament amb les dades asociades a ca:

1.5.2 Codis d'estat

Tots els serveis indiquen el resultat de l'operació amb un codi d'estat de la resposta HTTP que pot ser un dels següents.

200 OK El servidor ha pogut processar el servei i donar una resposta correctament. Si estem fent un POST, i l'element que voliem crear ja existeix, aquest és el codi que rebrem.

201 CREATED S'ha creat una entitat nova al recurs

400 BAD REQUEST Alguns dels paràmetres requerits manquen o són erronis

401 UNAUTHORIZED Manca alguna paràmetre d'autenticació o aquesta no és vàlida

404 NOT FOUND El recurs solicitat no existeix, en aquest cas, comprovar la URI

500 INTERNAL SERVER ERROR Només en cas de apareixer algun bug o cas no tractat

501 NOT IMPLEMENTED En recursos existents, indica que aquell mètode encara no esta implementat o que no aplica per el tipus de recurs solicitat

Exceptuant els codis 200 i 201, que en el cos de la resposta es retorna l'object sobre el qual s'està actuant, en la resta de casos s'inclou un text descriptiu de l'error, que ens pot indicar la causa exacte del mateix.

1.5.3 Accés a recursos

Els recursos que anomenarem "arrel" (a partir d'ara col.leccions), són identificats per parts de la URI que no canvien, i són expresades en plural, o amb una paraula que impliqui un conjunt. Aquestes col·leccions contenen 1 o més elements (a partir d'ara entitats).

Per exemple:

```
GET /people
```

Ens llista totes les entitats del recurs people i de la mateixa manera:

```
GET /people/mindundi
```

Ens llista totes les dades de l'entitat mindundi. En aquest cas a la URI podem llegir un recurs arrel i una entitat concreta d'aquest recurs, però podem tenir altres recursos aniuats, representant un subconjunt d'entitats filtrades segons el recurs pare com per exemple:

```
GET /people/mindundi/activities
```

Les URI's dels serveis REST les podem interpretar quasibé sempre llegint de dreta a esquerra, agrupant els diferents identificadors d'entitats concretes amb el seu recurs. D'aquesta manera, l'ultim exemple el podem interpretar com a:

```
"Activitats de la persona amb nom mindundi"
```

En les col.leccions, els elements poden venir limitats per una cerca (per defecte 10 elements), i s'han de consultar de manera iterativa. Per saber quants elements tenim en total en una col·lecció, podem fer:

GET /people/mindundi/activities

i consultar la capçalera `X-totalItems` de la resposta.

1.5.4 Creació d'elements nous

Per afegir una entitat a un recurs concret, ens podem trobar dos casos, depenent del recurs. Ens podem trobar que necessitem indicar el paràmetre que es convertira en l'identificador únic de l'entitat, o que el sistema ja el generi automàticament i per tant no ens calgui proporcionar-lo. En ambdos casos, per afegir entitats a un recurs, sempre utilitzarem el mètode POST, per exemple:

```
POST /people/mindundi
```

Ens afegirà l'usuari mindundi al recurs. Per altra banda:

```
POST /people/mindundi/activities
```

Ens afegirà una activitat, generant l'identificador automàticament. En la documentació detallada de cada recurs, especifica quin cas aplica per cada un, i quins paràmetres i/o estructures de dades se li han de proporcionar.

1.5.5 Accedir a elements concrets

Les URI's per accedir a un element concret d'un recurs, coincidiran amb la sintaxis de les URI's per afegir un element especificant l'identificador. La diferència estarà en el mètode d'accés.

D'aquesta manera fent:

```
GET /people/mindundi
```

Obtenim l'usuari mindundi que en la secció anterior haviem creat. La URI és la mateixa, els paràmetres els mateixos (en aquest cas no n'hi han), només canvia el mètode HTTP usat.

1.5.6 Format dels paràmetres

Els serveis accepten paràmetres en format JSON quan s'envien a través del cos de la resposta, cas en el qual s'ha de incloure la capçalera 'content-type' indicant el valor 'aplication/json'.

Els serveis també accepten pas de paràmentres en el format 'application/form-url-encoded', per exemple les peticions GET que no tenen cos, obligatòriament han de utilitzar aquest format.

Per estructures complexes amb més d'un nivell utilitzarem sempre JSON, mentre que per llistes de paràmetres clau=valor, podem utilitzar el tradicional 'application/form-url-encoded'

Per paràmetres que representen llistes de valors i que s'han de passar via GET, hem d'assegurar que el format en que es genera la llista de paràmentres compleixi el bàsic de HTTP, ja que hi han multiples convencions de com fer-ho.

Per exemple, si tenim el camp 'context' que té 2 valors 'A' i 'B', els paràmetres de la petició han repetir la clau 'context' tantes vegades com valors hi hagi:

```
?context=A&context=B
```

1.5.7 Adreces canòniques per recursos amb múltiples URI's

Hi a alguns casos en que un recurs te més d'una possible URI, majoritàriament determinat per que al crear l'element necessitem donar algun paràmetre que és coherent que estigui inclos en la URI, però que per posteriors accessos al recurs no te sentit. Posem un exemple a continuació per veure-ho més clar:

Per Afegir una activitat:

```
POST /people/mindundi/activities
```

Pasant el contingut de l'activitat amb una estructura JSON. Aquesta activitat, tot i haver-la creat nosaltres, es pot donar el cas que algú que ens estigui seguint vulgui compartir aquesta activitat. Supsant que la activitat creada tingues un nom "activitat1", seguint la sintaxis del rest, podem accedir a la activitat a través de

```
GET /people/mindundi/activities/activitat1
```

Tot i així si el que volem es compartir l'activitat amb algú altre, no te sentit indicar qui és el creador de l'activitat, ja que totes les activitats estan a la mateixa "saca". Podem accedir a la activitat de la següent manera:

```
GET /activities/activitat1
```

que és equivalent a la forma anterior. Això ens dona una URI (canònica) simplificada per dur a terme accions com per exemple la de compartir, que seria de la següent forma:

```
POST /activities/activitat1/shares
```

on shares representa el conjunt de vegades que s'ha compartit la activitat

1.6 Permisos

Es defineixen una serie de permisos i comportaments per defecte en relació als permisos del sistema i de les entitats del sistema.

1.6.1 General

Els permisos de modificació de camps i d'eliminació d'objectes estan subjectes a la propietat dels objectes. La propietat

- creator
- actor
- owner
- El creator te el mateix comportament a tots els objectes i identifica l'usuari que estava autenticat via oauth en el moment

de la creació de l'objecte. Aquest camp no es pot modificar - L'actor identifica la persona real que està creant un objecte, i que podria ser diferent del creator, en el cas que un administrador estés creant activitat en nom d'una altra persona. L'actor tampoc es pot modificar. - El owner identifica qui és el propietari, és a dir qui té el poder per poder-lo modificar i eliminar. En el moment de creació de l'objecte l'owner és el mateix que el creador, excepte el el cas de crear activitats, que serà l'actor. L'owner pot canviar en determinades ocasions, com per exemple al marxar l'owner original d'una conversa, l'owner passarà a ser un altre dels components de la conversa.

1.6.2 Contextos

Els permisos dels contextos serveixen per determinar quines accions poden dur a terme els usuaris sobre ell. Quan es crea un contexte se li especifica un valor per cada permis, que determinara que pot cada usuari que interactui amb el contexte. Els permisos de cada usuari es posaran en el moment de la subscripció segons el valor escollit en el contexte, poden ser modificats individualment per ususari més endavant:

READ

Defineix qui pot llegir l'activitat escrita al context

- subscribed - Tots els usuaris subscrits al context poden llegir
- public - Qualsevol usuari sense esta subscrit al context el pot llegir

WRITE

Defineix qui pot afegir activitat al context

- subscribed - Tots els usuaris subscrits al context poden escriure al context
- restricted - Només el usuaris subscrits i autoritzats explicitament poden escriure al context
- public - Qualsevol usuari sense esta subscrit al context poden escriure al context

SUBSCRIBE

Defineix qui es pot subscriure al context

- restricted - Només els usuaris administradors poden efectuar subscripcions al context
- public - Qualsevol usuari pot subscriure's al context (i dessubscriure's)

UNSUBSCRIBE

Defineix qui es pot dessubscriure d'un context. Aquest permís es dona automàticament en cas que el permís subscribe sigui pub

- restricted - Només els usuaris administradors poden dessubscriure usuaris del context
- public - Qualsevol usuari pot dessubscriure's del context previament subscrit

DELETE

Defineix qui podrà borrar activitats d'un contexte. Per defecte el permís es restricted.

- restricted - Només els propietaris de l'activitat i usuaris autoritzats expresament poden esborrar activitats.
- subscribed - Qualsevol persona subscrita al context pot esborrar qualsevol activitat

Si no s'especifiquen en la creació del context, els permisos per defecte són:

read='public', write='public', subscribe='public', invite='public'

1.7 Autenticació OAuth2

Aquest document mostra quin és el flux oauth2 emprat per l'autenticació dels serveis webs del MAX, així com quin és el mecanisme triat per transportar els paràmetres oauth dins les peticions del servei.

Amb la implementació actual, els tokens generats per servidor OAuth no caduquen mai, i s'han d'invalidar manualment amb un webservice si és necessita.

1.7.1 Descripció del flux Oauth

A continuació una definició descriptiva del flux de petició de tokens:

- Obtenim el nom d'usuari i el password de l'usuari de qual volem obtenir el token
- Contactem amb el servei oauth, i li proporcionem les dades per obtenir el token
- El servei oauth ens proporciona un token per l'usuari indicat

Un cop tenim el token, per tal d'utilitzarlo:

- Fem una crida a un servei del MAX, indicant l'usuari i el token
- El MAX al rebre la petició, comprova al servidor oauth la validesa del token
- Si el token és valid, executa el servei i retorna la resposta

1.7.2 Crides al sistema oauth

`POST /token/`

Demana un token per un usuari i un scope concrets segons els següents paràmetres passats en format 'application/form-urlencoded'

:`grant_type` : La cadena de text 'password', que indica que utilitzara un usuari LDAP per validar :`client_id` : en el nostre cas, l'id és 'MAX' :`scope`: Cadena de text que indica desde on es genera l'activitat, de moment l'única permesa pel max és 'widgetcli', per tant hem d'usar aquesta :`username`: nom d'usuari pel que volem obtenir el token :`password`: contrasenya de l'usuari

Retorna un codi de resposta HTTP 200 i una estructura JSON de la forma:

```
{  "oauth_token": "4d53575d0d9582839c510b3302ac1f2c", "scope": "widgetcli", "fresh": false  }
```

on 'fresh' ens indica si el token s'acava de generar o ja n'existia un de vàlid per aquest usuari i scope concrets. Davant d'un error, el servei retorna un codi d'error HTTP 400, i una estructura JSON indicant l'error, per exemple:

```
{  "error": "invalid_grant"   }
```

POST /checktoken
Comprova la validesa d'un token per un usuari i un scope concrets segons els següents paràmetres passats en format 'application/form-url-encoded'

:oauth_token : El token que volem validar :user_id : Nom d'usuari al que pertany el token :scope: Cadena de text que indica desde on es genera l'activitat, de moment l'única permesa pel max és 'widgetcli', per tant hem d'usar aquesta. Ha de ser la mateixa que es va especificar al moment de demanar el token.

Retorna un codi de resposta HTTP 200 si el token és valid i un HTTP 401 si no ho és.

1.7.3 Us de OAuth amb MAX

Per poder indicar al max que ens volem autenticar utilizant OAuth, ho farem a través de 3 capçaleres HTTP pròpies, amb les que proporcionarem les dades al MAX perquè pugui fer la validació del token. Tota petició a un servei del max que requereix de autenticació OAuth ha de contenir:

- `X-Oauth-Token`: amb el valor del token que volem validar
- `X-Oauth-Username`: amb el nom de l'usuari al qual pertany el token
- `X-Oauth-Scope`: scope amb el que es va generar el token

1.8 API REST

Aquest document exposa els serveis web públics creats pel motor d'activitat i subscripcions. Aquests estan organitzats per recursos i les operacions sobre ells. Tots els serveis web són de tipus RESTful i l'intercanvi d'informació es realitza en format JSON.

Els paràmetres indicats a les seccions Query parameters, poden ser de 3 tipus:

REST Són obligatoris ja que formen part de la URI

Requerits Són obligatoris, però formen part de l'estructura JSON que s'envia amb el cos de la petició. En el cas de peticions GET, aquesta estructura equival a paràmetres de la URL. (i.e. ?context=ab0012313...)

Opcionals Com el nom indica, no son obligatoris, indiquen alguna funcionalitat extra

Tots els serveis requereixen autenticació oAuth en cas de que no s'especifiqui el contrari.

Les respostes que retorna el sistema, en cas que siguin 1 o múltiples resultats (Associats a col.leccions o entitats) seràn, o bé un sol objecte JSON en cas de l'accés a una entitat concreta, o una llista o array de resultats:

```
[
   { ... },
   { ... }
]
```

1.8.1 Usuaris

Operacions sobre el recurs *usuari* del sistema.

GET /people
Retorna el resultat d'una cerca d'usuaris del sistema en forma de llista de noms d'usuaris per l'ús de la UI.

Paràmetres JSON

- `username`: El filtre de cerca d'usuaris.

Exemple de petició:

```
{"username": "messi"}
```

Resposta esperada:

```
[
    {
        "username": "messi",
        "displayName": "messi",
        "id": "519b0000000000000000000",
        "objectType": "person"
    }
]
```

Codis d'error:

- 200 petició executada correctament

Success Retorna la llista d'usuaris que compleix la query especificada.

PUT /people/{username}

Modifica un usuari del sistema pel seu posterior us si existeix. En cas de que l'usuari no existeixi retorna un error. La llista de paràmetres actualitzables de moment es limita a `displayName` i a `twitterUsername`.

Query Parameters

- **username** – (REST) L'identificador de l'usuari
- **displayName** – (Opcional) El nom real de l'usuari al sistema
- **twitterUsername** – (Opcional) El nom d'usuari de Twitter de l'usuari

Exemple de petició

```
{"displayName": "Lionel Messi", "twitterUsername": "messi10oficial"}
```

Resposta esperada:

```
{
    "username": "messi",
    "iosDevices": [],
    "displayName": "Lionel Messi",
    "talkingIn": [],
    "creator": "test_manager",
    "androidDevices": [],
    "following": [],
    "subscribedTo": [],
    "last_login": "2000-01-01T00:01:00Z",
    "published": "2000-01-01T00:01:00Z",
    "owner": "messi",
    "twitterUsername": "messi10oficial",
    "id": "519b0000000000000000000",
    "objectType": "person"
}
```

Success

Retorna un objecte `Person` amb els paràmetres indicats modificats.

Error

```
{"error_description": "Unknown user: messi", "error": "UnknownUserError"}
```

POST /people/{username}

Crea el perfil propi (el de l'usuari que executa) d'usuari remotament al sistema pel seu posterior ús si no existeix. En cas de que l'usuari ja existis, el retorna canviant el codi d'estat HTTP en funció de l'acció realitzada.

Query Parameters

- **username** – (REST) L'identificador del nou usuari al sistema
- **displayName** – (Opcional) El nom real (de pantalla) de l'usuari al sistema

Cos de la petició

```
{"username": "neymar", "displayName": "Neymar JR"}
```

Resposta esperada

```
{
    "username": "neymar",
    "iosDevices": [],
    "displayName": "Neymar JR",
    "talkingIn": [],
    "creator": "neymar",
    "androidDevices": [],
    "following": [],
    "subscribedTo": [],
    "last_login": "2000-01-01T00:01:00Z",
    "published": "2000-01-01T00:01:00Z",
    "owner": "neymar",
    "id": "519b0000000000000000000",
    "objectType": "person"
}
```

Success

Retorna un objecte `Person`.

GET /people/{username}

Retorna la informació d'un usuari del sistema. En cas de que l'usuari no existeixi retorna l'error especificat.

Query Parameters

- **username** – (REST) L'identificador de l'usuari

Exemple de petició

Aquesta petició no necessita cos.

Resposta esperada:

```
{
    "username": "messi",
    "iosDevices": [],
    "displayName": "Lionel Messi",
    "talkingIn": [],
    "creator": "test_manager",
    "androidDevices": [],
    "following": [],
    "subscribedTo": [],
    "last_login": "2000-01-01T00:01:00Z",
```

```
    "published": "2000-01-01T00:01:00Z",
    "owner": "messi",
    "twitterUsername": "messi10oficial",
    "id": "519b0000000000000000000",
    "objectType": "person"
}
```

Success

Retorna un objecte `Person`.

Error

```
{"error_description": "Unknown user: messi", "error": "UnknownUserError"}
```

`GET /people/{username}/avatar`

Retorna l'avatar (foto) de l'usuari del sistema. Aquest és un servei públic.

Query Parameters

- **username** – (REST) L'identificador de l'usuari

Success Retorna la imatge pel seu ús immediat.

`POST /people/{username}/avatar`

Permet a l'usuari del sistema pujar la seva imatge del seu perfil (avatar).

Query Parameters

- **username** – (REST) L'identificador de l'usuari

Cos de la petició

La petició ha d'estar feta mitjançant multipart/form-data amb les capçaleres corresponents d'oAuth en aquest endpoint.

Success Retorna un codi **201** (Created)

`POST /people/{username}/device/{platform}/{token}`

Afegeix un token de dispositiu al perfil de l'usuari. Aquest token és el que identifica el dispositiu per a que se li puguin enviar notificacions push.

Query Parameters

- **username** – (REST) L'identificador del nou usuari al sistema
- **platform** – (REST) El tipus de plataforma
- **token** – (REST) La cadena de text que representa el token

Cos de la petició

Aquesta petició no necessita cos.

Resposta esperada

```
{
    "username": "messi",
    "iosDevices": [
        "12345678901234567890123456789012"
    ],
    "displayName": "Lionel Messi",
    "talkingIn": [],
```

```
        "creator": "test_manager",
        "androidDevices": [],
        "following": [],
        "subscribedTo": [],
        "last_login": "2000-01-01T00:01:00Z",
        "published": "2000-01-01T00:01:00Z",
        "owner": "messi",
        "twitterUsername": "messi10oficial",
        "id": "519b0000000000000000000",
        "objectType": "person"
    }
```


Success

Retorna un objecte `Person`.

`DELETE /people/{username}/device/{platform}/{token}`

Esborra un token de dispositiu al perfil de l'usuari. Aquest token és el que identifica el dispositiu per a que se li puguin enviar notificacions push.

Query Parameters

- **username** – (REST) L'identificador del nou usuari al sistema
- **platform** – (REST) El tipus de plataforma
- **token** – (REST) La cadena de text que representa el token

Cos de la petició

Aquesta petició no necessita cos.

Resposta esperada

Retorna un codi HTTP 204 (deleted) amb el cos buit

Success

Retorna un objecte `Person`.

1.8.2 Activitats de l'usuari

Representa el conjunt d'activitats creades per un usuari i permet tant llistar-les com crear-ne de noves.

`POST /people/{username}/activities`

Genera una activitat en el sistema. Els objectes d'aquesta activitat són els especificats en el protocol activitystrea.ms.

Query Parameters

- **username** – (REST) Nom de l'usuari que crea l'activitat
- **contexts** – (Opcional) Per fer que una activitat estigui associada a un context determinat fa falta que enviem una llista d'objectes *context* (sota la clau `contexts`) (ja que teòricament, podem fer que l'activitat estigui associada a varis contexts a l'hora), indicant com a `objectType` el tipus `uri` i les dades del context com a l'exemple.
- **object** – (Requerit) Per ara només suportat el tipus `objectType` *note*. Ha de contindre les claus `objectType` i `content` el qual pot tractar-se d'un camp codificat amb HTML, amb tags restringits.

Exemple de petició

```
{
    "object": {
        "objectType": "note",
        "content": "Testejant la creació d'un canvi d'estatus"
    }
}
```

Resposta esperada:

```
{
    "generator": null,
    "creator": "messi",
    "favoritesCount": 0,
    "object": {
        "content": "Testejant la creaci\u00f3 d'un canvi d'estatus",
        "objectType": "note"
    },
    "lastComment": "519b00000000000000000002",
    "actor": {
        "username": "messi",
        "displayName": "Lionel Messi",
        "objectType": "person"
    },
    "published": "2000-01-01T00:01:00Z",
    "verb": "post",
    "likes": [],
    "favorites": [],
    "replies": [],
    "owner": "messi",
    "likesCount": 0,
    "id": "519b00000000000000000000",
    "objectType": "activity"
}
```

Success

Retorna un objecte del tipus `Activity`.

Error

En cas de que l'usuari actor no sigui el mateix usuari que s'autentica via oAuth

```
{u'error_description': u"You don't have permission to access xavi resources", u'error': u
```

En cas que l'usuari no existeixi

```
{"error_description": "Unknown user: messi", "error": "UnknownUserError"}
```

Tipus d'activitat suportats:

- *note* (estatus d'usuari)

Tipus d'activitat projectats:

- *File*
- *Event*
- *Bookmark*
- *Image*

- *Video*
- *Question*

En el cas que volguem lligar l'activitat a un context en concret, suposant que l'usuari ha estat previament subscrit a aquest context.

Exemple de petició

```
{
    "contexts": [
                    {
                        "url": "http://atenea.upc.edu",
                        "objectType": "context"
                     }
                ],
    "object": {
        "objectType": "note",
        "content": "<p>[A] Testejant la creació d'un canvi d'estatus a un context</p>"
    }
}
```

Resposta esperada:

```
{
    "generator": null,
    "creator": "messi",
    "contexts": [
        {
            "url": "http://atenea.upc.edu",
            "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
            "tags": [
                "Assignatura"
            ],
            "displayName": "Atenea",
            "objectType": "context"
        }
    ],
    "favoritesCount": 0,
    "object": {
        "content": "[A] Testejant la creaci\u00f3 d'un canvi d'estatus a un context",
        "objectType": "note"
    },
    "lastComment": "519b00000000000000000002",
    "actor": {
        "username": "messi",
        "displayName": "Lionel Messi",
        "objectType": "person"
    },
    "published": "2000-01-01T00:01:00Z",
    "verb": "post",
    "likes": [],
    "favorites": [],
    "replies": [],
    "owner": "messi",
    "likesCount": 0,
    "id": "519b00000000000000000000",
    "objectType": "activity"
}
```

GET /people/{username}/activities

Llista totes les activitats de tipus post generades al sistema per part d'un usuari concret.

Query Parameters

- **username** – (REST) Identificador d'usuari que crea l'activitat

Exemple de petició

Aquesta petició no necessita cos.

Resposta esperada:


```
[
    {
        "favorited": false,
        "liked": false,
        "generator": null,
        "contexts": [
            {
                "url": "http://atenea.upc.edu",
                "displayName": "Atenea",
                "objectType": "context",
                "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
                "tags": [
                    "Assignatura"
                ]
            }
        ],
        "favoritesCount": 0,
        "object": {
            "content": "[A] Testejant la creaci\u00f3 d'un canvi d'estatus a un context",
            "objectType": "note"
        },
        "lastComment": "519b00000000000000000002",
        "actor": {
            "username": "messi",
            "displayName": "Lionel Messi",
            "objectType": "person"
        },
        "published": "2000-01-01T00:01:00Z",
        "verb": "post",
        "likes": [],
        "favorites": [],
        "replies": [],
        "deletable": true,
        "objectType": "activity",
        "id": "519b00000000000000000000",
        "likesCount": 0
    },
    {
        "favorited": false,
        "liked": false,
        "generator": null,
        "favoritesCount": 0,
        "object": {
            "content": "Testejant la creaci\u00f3 d'un canvi d'estatus",
            "objectType": "note"
        },
        "lastComment": "519b00000000000000000002",
        "actor": {
```

```
            "username": "messi",
            "displayName": "Lionel Messi",
            "objectType": "person"
        },
        "published": "2000-01-01T00:01:00Z",
        "verb": "post",
        "likes": [],
        "favorites": [],
        "replies": [],
        "deletable": true,
        "objectType": "activity",
        "id": "519b00000000000000000000",
        "likesCount": 0
    }
]
```

Note: En l'ultima resposta esperada hi han tres entrades les dues activitats que hem generat fins ara (amb context, i l'altre sense) i l'activitat que es genera quan es subscriu un usuari a un context, que es tracta com una activitat més.

Success

Retorna una col·lecció d'objectes del tipus `Activity`.

Error

En cas de que l'usuari actor no sigui el mateix usuari que s'autentica via oAuth

```
{u'error_description': u"You don't have permission to access xavi resources", u'error': u
```

En cas que l'usuari no existeixi

```
{"error_description": "Unknown user: messi", "error": "UnknownUserError"}
```

1.8.3 Activitats d'un contexte

Torna el conjunt d'activitats generades pels usuaris del sistema a un context. L'usuari que fa la petició ha de tindre permisos de lectura com a mínim en el context requerit, de lo contrari se li denegarà l'accés. Típicament s'utilitza per recuperar totes les activitats que els usuaris han associat a un context concret.

GET /contexts/{hash}/activities

Llistat de totes les activitats del sistema, filtrada sota algun criteri

Query Parameters

- **hash** – (REST) El hash (sha1) de la URL del context
- **sortBy** – (Opcional) Tipus d'ordenació que s'aplicarà als resultats. Per defecte és `activities`, i te en compte la data de publicació de l'activitat. L'altre valor possible és `comments` i ordena per la data de l'últim comentari a l'activitat.

```
{"context": "e6847aed3105e85ae603c56eb2790ce85e212997"}
```

Resposta esperada:

```
[
    {
        "favorited": false,
        "liked": false,
        "generator": null,
        "contexts": [
            {
                "url": "http://atenea.upc.edu",
                "displayName": "Atenea",
                "objectType": "context",
                "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
                "tags": [
                    "Assignatura"
                ]
            }
        ],
        "favoritesCount": 0,
        "object": {
            "content": "[A] Testejant la creaci\u00f3 d'un canvi d'estatus a un context",
            "objectType": "note"
        },
        "lastComment": "519b00000000000000000002",
        "actor": {
            "username": "messi",
            "displayName": "Lionel Messi",
            "objectType": "person"
        },
        "published": "2000-01-01T00:01:00Z",
        "verb": "post",
        "likes": [],
        "favorites": [],
        "replies": [],
        "deletable": true,
        "objectType": "activity",
        "id": "519b00000000000000000000",
        "likesCount": 0
    }
]
```


Success Retorna una col·lecció d'objectes del tipus `Activity`.

1.8.4 Timeline

Representa el flux d'activitat global de l'usuari, que comprèn les activitats que ha creat, les activitats de les persones a qui segueix i les activitats generades sota els contexts concrets al qual està subscrit, directa o indirectament.

GET /people/{username}/timeline

Llistat de totes les activitats del timeline de l'usuari. Actualment filtra les activitats i només mostra les de tipus *post*.

Query Parameters

- **username** – (REST) Nom de l'usuari que del qual volem el llistat
- **sortBy** – (Opcional) Tipus d'ordenació que s'aplicarà als resultats. Per defecte és `activities`, i te en compte la data de publicació de l'activitat. L'altre valor possible és `comments` i ordena per la data de l'últim comentari a l'activitat.

Exemple de petició

Aquesta petició no necessita cos.

Resposta esperada:


```
[
    {
        "favorited": false,
        "liked": false,
        "generator": null,
        "contexts": [
            {
                "url": "http://atenea.upc.edu",
                "displayName": "Atenea",
                "objectType": "context",
                "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
                "tags": [
                    "Assignatura"
                ]
            }
        ],
        "favoritesCount": 0,
        "object": {
            "content": "[A] Testejant la creaci\u00f3 d'un canvi d'estatus a un context",
            "objectType": "note"
        },
        "lastComment": "519b00000000000000000002",
        "actor": {
            "username": "messi",
            "displayName": "Lionel Messi",
            "objectType": "person"
        },
        "published": "2000-01-01T00:01:00Z",
        "verb": "post",
        "likes": [],
        "favorites": [],
        "replies": [],
        "deletable": true,
        "objectType": "activity",
        "id": "519b00000000000000000000",
        "likesCount": 0
    },
    {
        "favorited": false,
        "liked": false,
        "generator": null,
        "favoritesCount": 0,
        "object": {
            "content": "Testejant la creaci\u00f3 d'un canvi d'estatus",
            "objectType": "note"
        },
        "lastComment": "519b00000000000000000002",
        "actor": {
            "username": "messi",
            "displayName": "Lionel Messi",
            "objectType": "person"
        },
        "published": "2000-01-01T00:01:00Z",
        "verb": "post",
```

```
            "likes": [],
            "favorites": [],
            "replies": [],
            "deletable": true,
            "objectType": "activity",
            "id": "519b00000000000000000000",
            "likesCount": 0
        }
    ]
```

Success

Retorna una col·lecció d'objectes del tipus `Activity`.

1.8.5 Comentaris d'una activitat

Representa el conjunt de comentaris fets a una activitat.

`POST /activities/{activity}/comments`

Afegeix un comentari a una activitat ja existent al sistema. Aquest servei crea el comentari pròpiament dit dins de l'activitat i genera una activitat nova del tipus *comment* (l'usuari ha comentat l'activitat...)

Query Parameters

- **activity** – (REST) Ha de ser un identificador vàlid d'una activitat existent, per exemple: 4e6eefc5aceee9210d000004
- **object** – (Requerit) El tipus (`objectType`) d'una activitat comentari ha de ser *comment*. Ha de contindre les claus `objectType` i `content`.

Exemple de petició

```
{
    "object": {
        "objectType": "comment",
        "content": "<p>[C] Testejant un comentari nou a una activitat</p>"
    }
}
```

Resposta esperada:

```
{
    "generator": null,
    "creator": "messi",
    "favoritesCount": 0,
    "object": {
        "content": "[C] Testejant un comentari nou a una activitat",
        "inReplyTo": [
            {
                "contexts": [],
                "id": "519b00000000000000000000",
                "objectType": "note"
            }
        ],
        "keywords": [
            "testejant",
            "comentari",
            "nou",
            "una",
```

```
                "activitat"
            ],
            "objectType": "comment"
        },
        "lastComment": "519b00000000000000000002",
        "actor": {
            "username": "messi",
            "displayName": "Lionel Messi",
            "objectType": "person"
        },
        "published": "2000-01-01T00:01:00Z",
        "keywords": [],
        "verb": "comment",
        "likes": [],
        "favorites": [],
        "replies": [],
        "owner": "messi",
        "likesCount": 0,
        "id": "519b00000000000000000000",
        "objectType": "activity"
    }
```

Success

Retorna l'objecte `Activity` del comentari.

`GET /activities/{activity}/comments`
Llista tots els comentaris d'una activitat

Query Parameters

- **activity** – (REST) ha de ser un identificador vàlid d'una activitat existent, per exemple: 4e6eefc5aceee9210d000004

Exemple de petició

Aquesta petició no necessita cos.

Resposta esperada:

```
[
    {
        "actor": {
            "username": "messi",
            "displayName": "Lionel Messi",
            "objectType": "person"
        },
        "content": "[C] Testejant un comentari nou a una activitat",
        "deletable": true,
        "published": "2000-01-01T00:01:00Z",
        "id": "519b00000000000000000000",
        "objectType": "comment"
    }
]
```

Success

Retorna una col·lecció d'objectes del tipus `Comment`

1.8.6 Subscripcions

GET /contexts/public

Dona una llista de tots els contextes als qual un usuari es pot subscriure lliurement

Exemple de petició

Aquesta petició no necessita cos.

Resposta esperada:


```
[
    {
        "displayName": "Atenea",
        "tags": [
            "Assignatura"
        ],
        "url": "http://atenea.upc.edu",
        "published": "2000-01-01T00:01:00Z",
        "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
        "permissions": {
            "write": "public",
            "subscribe": "public",
            "read": "public",
            "invite": "subscribed"
        },
        "id": "519b0000000000000000000",
        "objectType": "context"
    },
    {
        "displayName": "Atenea A",
        "tags": [
            "Assignatura"
        ],
        "url": "http://atenea.upc.edu/A",
        "published": "2000-01-01T00:01:00Z",
        "hash": "90c8f28a7867fbad7a2359c6427ae8798a37ff07",
        "permissions": {
            "write": "public",
            "subscribe": "public",
            "read": "public",
            "invite": "subscribed"
        },
        "id": "519b0000000000000000000",
        "objectType": "context"
    }
]
```


Success

Retorna un objecte del tipus `Activity`.

POST /people/{username}/subscriptions

Subscriu l'usuari a un context determinat. El context al qual es vol subscriure l'usuari ha de ser de tipus public, sinó obtindrem un error d'autorització `401 Unauthorized`

Query Parameters

- **username** – (REST) L'identificador de l'usuari al sistema.
- **contexts** – (Requerit) Tipus d'objecte al qual ens volem subscriure, en aquest cas del tipus

context. Hem de proporcionar un objecte amb les claus `objectType` i el valor *context*, i la dada `url` del context.

Exemple de petició

```
{
    "object": {
        "objectType": "context",
        "url": "http://atenea.upc.edu/A"
    }
}
```

Resposta esperada:

```
{
    "generator": null,
    "creator": "messi",
    "favoritesCount": 0,
    "object": {
        "url": "http://atenea.upc.edu/A",
        "objectType": "context"
    },
    "lastComment": "519b00000000000000000002",
    "actor": {
        "username": "messi",
        "displayName": "Lionel Messi",
        "objectType": "person"
    },
    "published": "2000-01-01T00:01:00Z",
    "keywords": [],
    "verb": "subscribe",
    "likes": [],
    "favorites": [],
    "replies": [],
    "owner": "messi",
    "likesCount": 0,
    "id": "519b000000000000000000",
    "objectType": "activity"
}
```

Success

Retorna un objecte del tipus `Activity`.

Error

En cas que l'usuari no existeixi

```
{ "error_description": "Unknown user: messi", "error": "UnknownUserError" }
```

Representa el conjunt de contextes als quals esta subscrit un usuari.

`GET /people/{username}/subscriptions`
Torna totes les subscripcions d'un usuari

Query Parameters

- **username** – (REST) L'identificador de l'usuari al sistema

Exemple de petició

Aquesta petició no necessita cos.

Resposta esperada:


```
[
    {
        "displayName": "Atenea",
        "tags": [
            "Assignatura"
        ],
        "url": "http://atenea.upc.edu",
        "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
        "objectType": "context",
        "permissions": [
            "read",
            "write",
            "unsubscribe",
            "invite"
        ]
    },
    {
        "displayName": "Atenea A",
        "tags": [
            "Assignatura"
        ],
        "url": "http://atenea.upc.edu/A",
        "hash": "90c8f28a7867fbad7a2359c6427ae8798a37ff07",
        "objectType": "context",
        "permissions": [
            "read",
            "write",
            "unsubscribe",
            "invite"
        ]
    }
]
```


DELETE /people/{username}/subscriptions/{hash}

Elimina la subscripció d'un usuari, si l'usuari té permis per dessubscriure's. NO esborra les activitats que s'hagin creat previament al context del qual ens hem dessubscrit. Tot i que les activitats que queden a la base de dades no es poden consultar directament, en el timeline de un usuari coninuarà veient les activitats que va crear ell.

Query Parameters

- **username** – (REST) L'identificador de l'usuari al sistema.
- **hash** – (REST) El hash del context la subscripció al qual es vol esborrar. Aquest hash es calcula fent una suma de verificació sha1 dels paràmetres del context

Exemple de petició

Aquesta petició no te cos.

1.8.7 Missatges i converses

El MAX implementa des de la seva versió 3.0 la funcionalitat de missatgeria instantània asíncrona entre els seus usuaris.

- Les converses tenen un limit de 20 participants.
- Les converses tenen un propietari, que és l'usuari que va crear la conversa.

- El propietari de la conversa pot afegir més gent a la conversa.
- El propietari de la conversa pot fer fora usuaris de la conversa.
- El propietari de la conversa *NO* pot marxar d'una conversa
- Els participants d'una conversa poden marxar sempre que vulguin de la conversa, els seus missatges no s'esborren

Aquests són els serveis associats.

`POST /conversations`

Crea una conversa nova, hi subscriu tots els participants especificats, i afegeix el missatge a la conversa.

Query Parameters

- **contexts** – (Requerit) Tipus d'objecte al qual ens volem subscriure (en aquest cas `conversation`). Hem de proporcionar un objecte amb les claus `objectType` i el valor `conversation`, i la llista de `participants` com a l'exemple
- **object** – (Requerit) Tipus d'objecte de la conversa. Hem de proporcionar un objecte (per ara només es permet el tipus *note*) i el contingut amb les dades `content` amb el cos del missatge propiament dit

Exemple de petició

```
{
    "contexts": [
        {
            "objectType":"conversation",
            "participants": ["messi", "xavi"]
        }
    ],
    "object": {
        "objectType": "note",
        "content": "Nos espera una gran temporada, no es cierto?"
    }
}
```

Resposta esperada:

```
{
    "generator": null,
    "creator": "messi",
    "contexts": [
        {
            "displayName": "messi, xavi",
            "id": "519b00000000000000000000",
            "objectType": "conversation"
        }
    ],
    "object": {
        "content": "Nos espera una gran temporada, no es cierto?",
        "keywords": [
            "nos",
            "espera",
            "una",
            "gran",
            "temporada",
            "cierto",
            "messi"
        ],
```

```
            "objectType": "note"
        },
        "replies": [],
        "actor": {
            "username": "messi",
            "displayName": "Lionel Messi",
            "objectType": "person"
        },
        "keywords": [],
        "verb": "post",
        "published": "2000-01-01T00:01:00Z",
        "owner": "messi",
        "id": "519b0000000000000000000",
        "objectType": "message"
    }
```

Success

Retorna l'objecte `Message` (activitat).

`GET /conversations/{hash}/messages`
Retorna tots els missatges d'una conversa

Query Parameters

- **hash** – (REST) El hash de la conversa en concret. Aquest hash es calcula fent una suma de verificació sha1 de la llista de participants (ordenada alfabèticament i sense espais) de la conversa

Exemple de petició

Aquesta petició no te cos.

Resposta esperada:

```
[
    {
        "generator": null,
        "contexts": [
            {
                "displayName": "messi, xavi",
                "id": "519b0000000000000000000",
                "objectType": "conversation"
            }
        ],
        "object": {
            "content": "Nos espera una gran temporada, no es cierto?",
            "objectType": "note"
        },
        "replies": [],
        "actor": {
            "username": "messi",
            "displayName": "Lionel Messi",
            "objectType": "person"
        },
        "verb": "post",
        "published": "2000-01-01T00:01:00Z",
        "id": "519b0000000000000000000",
        "objectType": "message"
    }
]
```

Success

Retorna una llista d'objectes `Message`

GET /conversations
Retorna totes les converses de l'actor que faci la petició

Exemple de petició

Aquesta petició no te cos.

Resposta esperada:


```
[
    {
        "displayName": "xavi",
        "creator": "messi",
        "messages": 1,
        "participants": [
            {
                "username": "messi",
                "displayName": "Lionel Messi",
                "objectType": "person"
            },
            {
                "username": "xavi",
                "displayName": "xavi",
                "objectType": "person"
            }
        ],
        "lastMessage": {
            "content": "Nos espera una gran temporada, no es cierto?",
            "published": "2000-01-01T00:01:00Z"
        },
        "published": "2000-01-01T00:01:00Z",
        "owner": "messi",
        "permissions": {
            "read": "subscribed",
            "write": "subscribed",
            "unsubscribe": "public",
            "invite": "restricted",
            "subscribe": "restricted"
        },
        "id": "519b0000000000000000000",
        "objectType": "conversation"
    }
]
```


Success

Retorna una llista d'objectes del tipus `Conversation`.

GET /conversations/{id}
Retorna una conversa

Query Parameters

- **id** – (REST) L'identificador d'una conversa. el podem obtenir en la resposta al crear una conversa nova, o en la llista de converses d'un usuari.

Exemple de petició

Aquesta petició no te cos.

Resposta esperada:

```
{
    "displayName": "xavi",
    "creator": "messi",
    "participants": [
        {
            "username": "messi",
            "displayName": "Lionel Messi",
            "objectType": "person"
        },
        {
            "username": "xavi",
            "displayName": "xavi",
            "objectType": "person"
        }
    ],
    "published": "2000-01-01T00:01:00Z",
    "owner": "messi",
    "permissions": {
        "read": "subscribed",
        "write": "subscribed",
        "unsubscribe": "public",
        "invite": "restricted",
        "subscribe": "restricted"
    },
    "id": "519b000000000000000000",
    "objectType": "conversation"
}
```

Success

Retorna un objecte del tipus `Conversation`.

`PUT /conversations/{id}`
Modifica una conversa

Query Parameters

- **id** – (REST) L'identificador d'una conversa. el podem obtenir en la resposta al crear una conversa nova, o en la llista de converses d'un usuari.
- **displayName** – El nom visible de la conversa, només visible en converses de més de 2 participants.

Exemple de petició

```
{
    displayName: 'Nou nom'
}
```

Resposta esperada:

```
{
    "displayName": "xavi",
    "creator": "messi",
    "participants": [
        {
            "username": "messi",
            "displayName": "Lionel Messi",
            "objectType": "person"
```

```
            },
            {
                "username": "xavi",
                "displayName": "xavi",
                "objectType": "person"
            }
        ],
        "published": "2000-01-01T00:01:00Z",
        "owner": "messi",
        "permissions": {
            "read": "subscribed",
            "write": "subscribed",
            "unsubscribe": "public",
            "invite": "restricted",
            "subscribe": "restricted"
        },
        "id": "519b0000000000000000000",
        "objectType": "conversation"
    }
```

Success

Retorna un objecte del tipus `Conversation`.

`POST /conversations/{hash}/messages`

Crea un missatge nou a una conversa ja existent

Query Parameters

- **hash** – (REST) El hash de la conversa en concret. Aquest hash es calcula fent una suma de verificació sha1 de la llista de participants (ordenada alfabèticament i sense espais) de la conversa

Exemple de petició

```
{
    "object": {
        "objectType": "note",
        "content": "M'agrada Taradell!"
    }
}
```

Resposta esperada:

```
{
    "generator": null,
    "creator": "messi",
    "contexts": [
        {
            "displayName": "messi, xavi",
            "id": "519b0000000000000000000",
            "objectType": "conversation"
        }
    ],
    "object": {
        "content": "M'agrada Taradell!",
        "keywords": [
            "agrada",
            "taradell",
            "messi"
```

```
            ],
            "objectType": "note"
        },
        "replies": [],
        "actor": {
            "username": "messi",
            "displayName": "Lionel Messi",
            "objectType": "person"
        },
        "keywords": [],
        "verb": "post",
        "published": "2000-01-01T00:01:00Z",
        "owner": "messi",
        "id": "519b00000000000000000000",
        "objectType": "message"
    }
```


Success

Retorna l'objecte `Message` (activitat).

`POST /people/{username}/conversations/{id}`

Afegeix un usuari a una conversa. L'usuari propietari de la conversa és l'únic que ho pot fer. Hi ha un limit de 20 participants per conversa.

Query Parameters

- **username** – (REST) L'usuari que es vol afegir a la conversa
- **id** – (REST) L'identificador d'una conversa. el podem obtenir en la resposta al crear una conversa nova, o en la llista de converses d'un usuari.

Exemple de petició

Aquesta petició no te cos.

Resposta esperada:


```
{
    "generator": null,
    "creator": "messi",
    "favoritesCount": 0,
    "object": {
        "participants": [
            {
                "username": "messi",
                "displayName": "Lionel Messi",
                "objectType": "person"
            },
            {
                "username": "xavi",
                "displayName": "xavi",
                "objectType": "person"
            },
            {
                "username": "nouusuari",
                "displayName": "nouusuari",
                "objectType": "person"
            }
        ],
        "id": "519b00000000000000000000",
```

```
            "objectType": "conversation"
        },
        "lastComment": "519b00000000000000000002",
        "actor": {
            "username": "nouusuari",
            "displayName": "nouusuari",
            "objectType": "person"
        },
        "published": "2000-01-01T00:01:00Z",
        "keywords": [],
        "verb": "subscribe",
        "likes": [],
        "favorites": [],
        "replies": [],
        "owner": "nouusuari",
        "likesCount": 0,
        "id": "519b00000000000000000000",
        "objectType": "activity"
    }
```


Retorna un codi HTTP 201 (created) amb la subscripció, o un HTTP 401 (Unauthorized) si l'usuari no és el propietari. Si sobrepassem el límit obtindrem un HTTP 403 (Forbidden)

DELETE /people/{username}/conversations/{id}

Treu un usuari d'una conversa. Ho pot fer qualsevol participant de la conversa excepte el propietari.

Query Parameters

- **username** – (REST) L'usuari que es vol afegir a la conversa
- **id** – (REST) L'identificador d'una conversa. el podem obtenir en la resposta al crear una conversa nova, o en la llista de converses d'un usuari.

Exemple de petició

Aquesta petició no te cos.

Resposta esperada:

Retorna un codi HTTP 204 (deleted) amb el cos buit, o un HTTP 401 (Unauthorized) si l'usuari no és el propietari

DELETE /conversations/{id}

Elimina una conversa

Elimina una conversa i tots els seus missatges de forma permanent. L'usuari propietari de la conversa és l'únic que pot eliminarla.

Query Parameters

- **id** – (REST) L'identificador d'una conversa. el podem obtenir en la resposta al crear una conversa nova, o en la llista de converses d'un usuari.

Exemple de petició

Aquesta petició no te cos.

Resposta esperada:

Retorna un codi HTTP 204 (deleted) amb el cos buit, o un HTTP 401 (Unauthorized) si l'usuari no és el propietari

1.8.8 Contextos

Tot i que els serveis associats a contextos són majoritàriament d'accés restringit, els que són accessibles per usuaris normals estàn documentats aquí

GET /contexts/public
Dona una llista de tots els contextes als qual un usuari es pot subscriure lliurement

Exemple de petició

Aquesta petició no necessita cos.

Resposta esperada:


```
[
    {
        "displayName": "Atenea",
        "tags": [
            "Assignatura"
        ],
        "url": "http://atenea.upc.edu",
        "published": "2000-01-01T00:01:00Z",
        "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
        "permissions": {
            "write": "public",
            "subscribe": "public",
            "read": "public",
            "invite": "subscribed"
        },
        "id": "519b0000000000000000000",
        "objectType": "context"
    },
    {
        "displayName": "Atenea A",
        "tags": [
            "Assignatura"
        ],
        "url": "http://atenea.upc.edu/A",
        "published": "2000-01-01T00:01:00Z",
        "hash": "90c8f28a7867fbad7a2359c6427ae8798a37ff07",
        "permissions": {
            "write": "public",
            "subscribe": "public",
            "read": "public",
            "invite": "subscribed"
        },
        "id": "519b0000000000000000000",
        "objectType": "context"
    }
]
```


Success

Retorna un objecte del tipus `Context`.

GET /contexts/{hash}/avatar
Retorna la imatge que li correspon al context depenent del usuari de Twitter que te assignat. Si no en te cap, retorna una imatge estàndar. Per ara només està implementada la integració amb Twitter i dissenyat per quan un context vol *parlar* impersonat a l'activitat del seu propi context. Per exemple, una assignatura.

Aquest és un servei públic, no és necessaria la autenticació oauth.

Query Parameters

- **hash** – (REST) El hash del context en concret. Aquest hash es calcula fent una suma de verificació sha1 de la URL del context.

Success

Retorna la imatge del context.

1.9 API REST (restringida)

Aquesta és la llista de serveis REST que queda fora de la operativa d'usuari i de la UI. Està dissenyada per l'ús d'un usuari restringit d'aplicació pel qual prèviament se li ha d'haver donat permisos al sistema. Aquest usuari no te cap privilegi addicional sobre el sistema que el que s'exposa en la descripció de cada servei.

1.9.1 Usuaris

POST /people/{username}

Crea un usuari remotament al sistema pel seu posterior ús si no existeix. En cas de que l'usuari ja existis, el retorna canviant el codi d'estat HTTP en funció de l'acció realitzada.

Query Parameters

- **username** – (REST) L'identificador del nou usuari al sistema
- **displayName** – (Opcional) El nom real (de pantalla) de l'usuari al sistema

Cos de la petició

```
{"username": "messi", "displayName": "Lionel Messi"}
```

Resposta esperada

```
{
    "username": "messi",
    "iosDevices": [],
    "displayName": "messi",
    "talkingIn": [],
    "creator": "test_manager",
    "androidDevices": [],
    "following": [],
    "subscribedTo": [],
    "last_login": "2000-01-01T00:01:00Z",
    "published": "2000-01-01T00:01:00Z",
    "owner": "messi",
    "id": "519b0000000000000000000",
    "objectType": "person"
}
```

Success

Retorna un objecte `Person`.

1.9.2 Contexts

POST /contexts

Crea un context al sistema.

Query Parameters

- **objectType** – (Obligatori) De moment se suporta el tipus `context`
- **url** – La URL amb la qual s'identifica el context.
- **displayName** – (Opcional) El nom per mostrar a la UI. Si no s'especifica, es mostrarà la url
- **twitterHashtag** – (Opcional) El hashtag (#) que identifica els posts a Twitter com a posts del context. Tots els posts d'usuaris del sistema i amb permisos al context amb compta de twitter informada s'importaran i apareixeran a l'activitat del context.
- **twitterUsername** – (Opcional) El nom d'usuari de Twitter que aquest context té assignat. Qualsevol post fet a Twitter amb aquest usuari s'importarà i apareixerà a l'activitat del context com activitat (impersonat) del propi context.
- **permissions** – (Opcional) Els permisos i parametrització de seguretat relacionada amb el context. Per defecte els contextos són públics a tots els efectes.
- **tags** – (Opcional) Llista de tags per categoritzar un contexte

Cos de la petició

```
{
    "url": "http://atenea.upc.edu",
    "objectType": "context",
    "displayName": "Atenea",
    "tags": ["Assignatura"]
}
```

Resposta esperada

```
{
    "displayName": "Atenea",
    "creator": "test_manager",
    "url": "http://atenea.upc.edu",
    "tags": [
        "Assignatura"
    ],
    "published": "2000-01-01T00:01:00Z",
    "owner": "test_manager",
    "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
    "permissions": {
        "read": "public",
        "write": "public",
        "invite": "public",
        "subscribe": "public"
    },
    "id": "519b000000000000000000",
    "objectType": "context"
}
```

Success

Retorna l'objecte `Context`.

`GET /contexts`

Cerca un context al sistema

Tags (Opcional)

Cos de la petició

```
{
    "tags": "Assignatura"
}
```

Resposta esperada

```
[
    {
        "displayName": "Atenea",
        "creator": "test_manager",
        "url": "http://atenea.upc.edu",
        "tags": [
            "Assignatura"
        ],
        "published": "2000-01-01T00:01:00Z",
        "owner": "test_manager",
        "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
        "objectType": "context",
        "id": "519b00000000000000000000",
        "permissions": {
            "read": "public",
            "write": "public",
            "invite": "public",
            "subscribe": "public"
        }
    }
]
```

PUT /contexts/{hash}

Modifica un context al sistema. Els camps que es poden modificar queden descrits a continuació

Query Parameters

- **hash** – (REST) El hash del context en concret. Aquest hash es calcula fent una suma de verificació sha1 de la URL del context.
- **displayName** – (Opcional) El nom per mostrar a la UI.
- **twitterHashtag** – (Opcional) El hashtag (#) que identifica els posts a Twitter com a posts del context. Tots els posts d'usuaris del sistema i amb permisos al context amb compta de twitter informada s'importaran i apareixeran a l'activitat del context.
- **twitterUsername** – (Opcional) El nom d'usuari de Twitter que aquest context té assignat. Qualsevol post fet a Twitter amb aquest usuari s'importarà i apareixerà a l'activitat del context com activitat (impersonat) del propi context.
- **tags** – (Opcional) Llista de tags per categoritzar un contexte

Cos de la petició

```
{ "twitterHashtag": "assignatura1" }
```

Resposta esperada

```
{
    "twitterHashtag": "assignatura1",
    "displayName": "Atenea",
    "creator": "test_manager",
    "url": "http://atenea.upc.edu",
    "tags": [
        "Assignatura"
```

```
        ],
        "published": "2000-01-01T00:01:00Z",
        "owner": "test_manager",
        "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
        "objectType": "context",
        "id": "519b0000000000000000000",
        "permissions": {
            "read": "public",
            "write": "public",
            "invite": "public",
            "subscribe": "public"
        }
    }
```

Success

Retorna l'objecte `Context` modificat.

`GET /contexts/{hash}`

Retorna la informació d'un objecte `Context`.

Query Parameters

- **hash** – (REST) El hash del context en concret. Aquest hash es calcula fent una suma de verificació sha1 de la URL del context.

Cos de la petició

Aquesta petició no te cos.

Resposta esperada

```
{
    "twitterHashtag": "assignatura1",
    "displayName": "Atenea",
    "creator": "test_manager",
    "url": "http://atenea.upc.edu",
    "tags": [
        "Assignatura"
    ],
    "published": "2000-01-01T00:01:00Z",
    "owner": "test_manager",
    "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
    "permissions": {
        "read": "public",
        "write": "public",
        "invite": "public",
        "subscribe": "public"
    },
    "id": "519b0000000000000000000",
    "objectType": "context"
}
```

Success

Retorna un objecte del tipus `Context`.

`DELETE /contexts/{hash}`

Esborra un objecte `Context` i les subscripcions de tots els usuaris subscrits a aquell contexte NO esborra les activitats que s'hagin creat previament al context esborrat. Tot i que les activitats que queden a la base de dades no es poden consultar directament, en el timeline de un usuari coninuarà veient les activitats que va crear ell.

Query Parameters

- **hash** – (REST) El hash del context en concret. Aquest hash es calcula fent una suma de verificació sha1 dels paràmetres del context

Cos de la petició

Aquesta petició no te cos.

1.9.3 Subscripcions

POST /people/{username}/subscriptions

Subscriu l'usuari a un context determinat.

Query Parameters

- **username** – (REST) L'identificador de l'usuari al sistema.
- **contexts** – (Requerit) Tipus d'objecte al qual ens volem subscriure, en aquest cas del tipus *context*. Hem de proporcionar un objecte amb les claus `objectType` i el valor *context*, i la dada `url` del context.

Cos de la petició

```
{
    "object": {
        "objectType": "context",
        "url": "http://atenea.upc.edu"
    }
}
```

Resposta esperada

```
{
    "generator": null,
    "creator": "test_manager",
    "favoritesCount": 0,
    "object": {
        "url": "http://atenea.upc.edu",
        "objectType": "context"
    },
    "lastComment": "519b000000000000000002",
    "actor": {
        "username": "messi",
        "displayName": "messi",
        "objectType": "person"
    },
    "published": "2000-01-01T00:01:00Z",
    "keywords": [],
    "verb": "subscribe",
    "likes": [],
    "favorites": [],
    "replies": [],
    "owner": "messi",
    "likesCount": 0,
    "id": "519b000000000000000000",
    "objectType": "activity"
}
```

Success

Retorna un objecte del tipus `Activity`.

Error

En cas que l'usuari no existeixi

```
{ "error_description": "Unknown user: messi", "error": "UnknownUserError" }
```

`DELETE /people/{username}/subscriptions/{hash}`

Elimina la subscripció d'un usuari Esborra un objecte `Context` i les subscripcions de tots els usuaris subscrits a aquell contexte. NO esborra les activitats que s'hagin creat previament al context del qual ens hem dessubscrit. Tot i que les activitats que queden a la base de dades no es poden consultar directament, en el timeline de un usuari coninuarà veient les activitats que va crear ell.

Query Parameters

- **username** – (REST) L'identificador de l'usuari al sistema.
- **hash** – (REST) El hash del context la subscripció al qual es vol esborrar. Aquest hash es calcula fent una suma de verificació sha1 dels paràmetres del context

Cos de la petició

Aquesta petició no te cos.

1.9.4 Permisos a contexts

Sobre els objectes context es poden otorgar o revocar permisos a usuaris del sistema. Aquests permisos són qualsevol dels permisos definitsen profunditat en l'apartat de permisos. Els permisos otorgats amb aquest sistema són persistents, per tant un canvi de permisos en els valors per defecte del context no afecta als permisos afegits/denegats.

Per tornar un usuari am permisos afegits/denegats als valors per defecte del contexte, haurem de resetejar els permisos d'aquell usuari en aquell contexte.

Un usuari pot tenir un permis per defecte, i tot i aixi otorgar-li el permis de manera persistent, de manera que a partir de llavors, no el pot perdre exepte per eliminació explícita.

`PUT /contexts/{hash}/permissions/{username}/{permission}`

Afegeix els permisos per un context donat un identificador d'usuari i el permís que li vols donar, de forma persistent.

Query Parameters

- **hash** – (REST) El hash del context en concret. Aquest hash es calcula fent una suma de verificació sha1 de la URL del context.
- **username** – (REST) L'identificador del nou usuari al sistema
- **permission** – (REST) El permís que li volem otorgar a l'usuari

Cos de la petició

Aquesta petició no te cos.

Resposta esperada

```
{
    "twitterHashtag": "assignatura1",
    "displayName": "Atenea",
    "url": "http://atenea.upc.edu",
    "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
    "objectType": "context",
```

```
        "permissions": [
            "read",
            "write",
            "unsubscribe"
        ]
    }
```

Success

Si el permís ja estava otorgat, el codi HTTP de resposta és 200, si no, torna un 201. En el cos, torna la subscripció modificada.

DELETE /contexts/{hash}/permissions/{username}/{permission}

Denega els permis per un context donat un identificador d'usuari i el permís que li vols donar de manera persistent

Query Parameters

- **hash** – (REST) El hash del context en concret. Aquest hash es calcula fent una suma de verificació sha1 de la URL del context.
- **username** – (REST) L'identificador del nou usuari al sistema
- **permission** – (REST) El permís que li volem otorgar a l'usuari

Cos de la petició

Aquesta petició no te cos.

Resposta esperada

```
{
    "twitterHashtag": "assignatura1",
    "displayName": "Atenea",
    "url": "http://atenea.upc.edu",
    "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
    "permissions": [
        "read",
        "unsubscribe"
    ],
    "objectType": "context"
}
```

POST /contexts/{hash}/permissions/{username}/defaults

Reseteja els permisos d'un usuari en un contexte als per defecte d'aquell contexte. Això elimina qualsevol permis persistent afegit o denegat anteriorment.

Query Parameters

- **hash** – (REST) El hash del context en concret. Aquest hash es calcula fent una suma de verificació sha1 de la URL del context.
- **username** – (REST) L'identificador del nou usuari al sistema

Cos de la petició

Aquesta petició no te cos.

Resposta esperada

```
{
    "twitterHashtag": "assignatura1",
    "displayName": "Atenea",
    "url": "http://atenea.upc.edu",
```

```
        "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
        "permissions": [
            "read",
            "write",
            "unsubscribe"
        ],
        "objectType": "context"
    }
```

1.9.5 Activitats

POST /people/{username}/activities

Afegeix una activitat en nom d'un usuari qualsevol

Query Parameters

- **username** – (REST) El nom d'usuari en nom del qual es crearà l'activitat
- **contexts** – (Opcional) Per fer que una activitat estigui associada a un context determinat fa falta que enviem una llista d'objectes *context* (sota la clau `contexts`) (ja que teòricament, podem fer que l'activitat estigui associada a varis contexts a l'hora), indicant com a `objectType` el tipus `context` i les dades del context com a l'exemple
- **object** – (Requerit) Per ara només suportat el tipus (`objectType`) *note*. Ha de contindre les claus `objectType` i `content` que pot tractar-se d'un camp codificat amb HTML

Cos de la petició

```
{
    "contexts": [
                    {
                        "url": "http://atenea.upc.edu",
                        "objectType": "context"
                     }
                ],
    "object": {
        "objectType": "note",
        "content": "<p>[A] Testejant la creació d'un canvi d'estatus a un context</p>"
    }
}
```

Resposta esperada

```
{
    "generator": null,
    "creator": "test_manager",
    "contexts": [
        {
            "twitterHashtag": "assignatura1",
            "displayName": "Atenea",
            "tags": [
                "Assignatura"
            ],
            "url": "http://atenea.upc.edu",
            "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
            "objectType": "context"
        }
    ],
    "favoritesCount": 0,
```

```
        "object": {
            "content": "[A] Testejant la creaci\u00f3 d'un canvi d'estatus a un context",
            "objectType": "note"
        },
        "lastComment": "519b00000000000000000002",
        "actor": {
            "username": "messi",
            "displayName": "messi",
            "objectType": "person"
        },
        "published": "2000-01-01T00:01:00Z",
        "verb": "post",
        "likes": [],
        "favorites": [],
        "replies": [],
        "owner": "messi",
        "likesCount": 0,
        "id": "519b00000000000000000000",
        "objectType": "activity"
    }
```

POST /contexts/{hash}/activities

Afegeix una activitat en nom d'un context qualsevol

Query Parameters

- **hash** – (REST) El hash del context en nom del qual es crearà l'activitat
- **contexts** – (Requerit) Per fer que una activitat estigui associada a un context determinat fa falta que enviem una llista d'objectes *context* (sota la clau `contexts`) (ja que teòricament, podem fer que l'activitat estigui associada a varis contexts a l'hora), indicant com a `objectType` el tipus `context` i les dades del context com a l'exemple. En aquest cas d'ús el contexte especificat aquí ha de ser el mateix que l'especificat al paràmetre {hash}
- **object** – (Requerit) Per ara només suportat el tipus (`objectType`) *note*. Ha de contindre les claus `objectType` i `content` que pot tractar-se d'un camp codificat amb HTML.

Cos de la petició

```
{
    "contexts": [
                    {
                        "url": "http://atenea.upc.edu",
                        "objectType": "context"
                     }
                ],
    "object": {
        "objectType": "note",
        "content": "<p>[A] Testejant la creació d'un canvi d'estatus a un context</p>"
    }
}
```

Resposta esperada

```
{
    "generator": null,
    "creator": "test_manager",
    "contexts": [
        {
            "url": "http://atenea.upc.edu",
```

```
                "displayName": "Atenea",
                "tags": [
                    "Assignatura"
                ],
                "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
                "objectType": "context"
            }
        ],
        "favoritesCount": 0,
        "object": {
            "content": "[A] Testejant la creaci\u00f3 d'un canvi d'estatus a un context",
            "keywords": [
                "testejant",
                "creaci\u00f3",
                "canvi",
                "estatus",
                "context"
            ],
            "objectType": "note"
        },
        "lastComment": "519b00000000000000000002",
        "actor": {
            "url": "http://atenea.upc.edu",
            "hash": "e6847aed3105e85ae603c56eb2790ce85e212997",
            "displayName": "Atenea",
            "objectType": "context"
        },
        "published": "2000-01-01T00:01:00Z",
        "keywords": [
            "canvi",
            "creaci\u00f3",
            "estatus",
            "testejant",
            "context"
        ],
        "verb": "post",
        "likes": [],
        "favorites": [],
        "replies": [],
        "owner": "test_manager",
        "likesCount": 0,
        "id": "519b00000000000000000000",
        "objectType": "activity"
    }
```


1.10 Max UI

En aquest document descriurem els passos per integrar el widget del max, anomenat MaxUI, en aplicacions de tercers. MaxUI proporciona una interfície d'usuari configurable per a clients finals a través de la qual accedir a les funcionalitats del MAX.

El widget esta dissenyat per incorporar-se en qualsevol pagina html com a un contingut més, sense iframes, i sense interferir ni amb el comportament de la pàgina ni amb la càrrega d'aquesta. Si per la raó que sigui el widget no esta disponible o tarda molt en carregar, la pàgina que el conté no s'en veurà perjudicada. Al ser un component implementat en javascript, la càrrega de renderitzar-lo i de les peticions als serveis del max que realitza està suportada al 100% pel navegador.

1.10.1 Prerequisits

Per poder utilitzar el widget cal:

- Tenir instal·lat a l'entorn web jQuery 1.7.x o superior.

Si no disposem de jQuery, el podem instal·lar localment o bé agafar-lo de qualsevol CDN disponible a la xarxa, per exemple:

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
```

- Disposar d'algun mecanisme per generar javascript dinàmic:

Com a mínim hi haurà un paràmetre que s'ha de configurar, que depèn de l'usuari autenticat a la web on es mostrarà el widget. Es per això que el nostre sistema ha de disposar d'algun mètode per incorporar fitxers *.js* generats partir de plantilles.

- Disposar d'algun mecanisme per interceptar la autenticació de l'usuari a l'aplicació hoste.
- Els usuaris del sistema ha de ser usuaris de l'LDAP UPC.

El servidor de OAUTH que s'utilitza per autenticar els usuaris del widget, te de base els usuaris del LDAP.

1.10.2 Instal·lació del widget

La instal·lació del widget consisteix en les següents parts:

- Incorporació del *contenidor* html on es renderitzadà el widget
- Incorporació del *loader* que descarregarà i inicialitzarà el widget
- Incorporació dels *css* del widget

Contenidor

El *contenidor* és l'element del DOM de la pàgina destí on es vol mostrar el widget. Hem de reservar un espai amb un element `<div>` al qual li hem de fixar la amplada en pixels ja sigui a través dels css existents en la pàgina, o bé amb un `style` *inline*. L'amplada pot ser qualsevol, però per una millor experiència d'usuari es recomana d'un mínim de 300px:

```
<div id="container" style="width:300px"> </div>
```

Aconsellem proporcionar identificar inequívocament el contenidor amb un `id=nom`, per facilitar la inicialització del widget. Si no es proporciona un identificador directament al contenidor, s'ha de poder construir un selector apropiat que retorni únicament el contenidor, per exemple:

```
<div id="principal">
  <div class="columna primera">
   XXXX
  </div>
  <div class="columna segona"></div>
</div>
```

Suposant que l'exemple és codi html ja existent, que volem aprofitar i no podem/volem inserir nous elements html, podríem referenciar el contenidor de manera única amb el selector css `#principal .columna.segona`

Loader

El *loader* és el següent codi javascript encarregat de instanciar de manera asíncrona el widget en la nostra aplicació:

```
// 1 - Inicialitzar variable global
window._MAXUI = window._MAXUI || {}

// 2 -  Inicialitzar widget, quan estigui disponible
window._MAXUI.onReady = function() {
    var settings = {  }

    intervalID = setInterval(function(event) {
        if ($().maxUI) {
            clearInterval(intervalID)
            $('#activityStream').maxUI(settings)
        }
    }, 30)
}


// 3 - Descarregar codi del widget
(function(d){
    var mui_location = 'http://rocalcom.upc.edu/maxui/maxui.js'
    var mui = d.createElement('script'); mui.type = 'text/javascript'; mui.async = true;
    mui.src = mui_location
    var s = d.getElementsByTagName('script')[0]; s.parentNode.insertBefore(mui, s);
}(document))
```

i consta de 3 porcions de codi que s'han d'incorporar a la resta de javascript de la nostra pàgina. Passem a descriure les diferents parts:

1. **Inicialitzar variable global**

El widget utilitza aquest variable, de tipus `Object` de javascript, on es poden emmagatzemar dades en format *clau-valor*. Aquesta variable és accessible com a global, a través de `_MAXUI` o `window._MAXUI` indistintament, i proporciona un lloc on emmagatzemar altres variables globals o configuracions, sense risc d'entrar en conflicte de noms amb altres variables existents. Aquesta primera part s'assegura de crear la variable si no existeix i donar-li un valor per defecte

2. **Inicialitzar widget**

Aquí definim una funció `onReady`, que el propi widget s'encarrega d'executar un cop s'ha completat la descàrrega en el següent pas. Dins d'aquesta funció és on definirem sobre quin *contenidor* hem d'inicialitzar el widget (`#activityStream` a l'exemple), i li passarem els paràmetres de configuració oportuns.

3. **Descarregar codi del widget**

Per últim, injectem en el codi de la pàgina l'ordre per descarregar de manera asíncrona el codi del maxui. La ubicació d'aquest codi pot ser remota com a l'exemple, que el descarrega de `http://rocalcom.upc.edu/maxui/maxui.js`, o bé el podeu ubicar als vostres servidors. **ULL!** Si l'ubiqueu als vostres servidors, les imatges que utilitza el widget les continuara agafant del servidor del qual heu descarregat el maxui.js. En cas que volguéssiu hostatjar les imatges, haureu de substituir manualment la url al maxui.js.

Note: **IMPORTANT** S'ha de respectar l'ordre de les 3 parts quan incorporem el codi als fitxers javascripts de la pagina.

1.10.3 CSS

Cal incorporar els css dels qual depèn el widget a cadascuna de les pàgines on se'l vulgui renderitzar. Per fer-ho, inclourem el següent codi al `<head>` de la pàgina:

```
<link rel="stylesheet" type="text/css" href="http://rocalcom.upc.edu/maxui/maxui.css">
```

o bé, tal com hem explicat anteriorment amb el `maxui.js`, el podem hostatjar localment en els nostres servidors, i de mateixa manera, haurem de tenir en compte la reescriptura de les urls de les imatges que hi ha al css.

1.10.4 Configuració del widget

Per configurar el widget, prepararem una variable javascript, on especificarem els paràmetres amb els quals volem inicialitzar el widget. Aquí tenim una mostra, a tall d'exemple per veure una representació dels diversos valors que pot prendre, en mode timeline:

```
var settings = {
       'language': 'ca',
       'username' : 'nom.cognom',
       'oAuthToken' : '01234567890abcdef01234567890abcd',
       'oAuthGrantType' : 'password',
       'maxServerURL' : 'https://rocalcom.upc.edu',
       'activitySource': 'timeline'
       }
```

i un altra exemple en mode context:

```
var settings = {
       'language': 'ca',
       'username' : 'nom.cognom',
       'oAuthToken' : '01234567890abcdef01234567890abcd',
       'oAuthGrantType' : 'password',
       'maxServerURL' : 'https://rocalcom.upc.edu',
       'readContext': 'http://foo.com/bar',
       'writeContexts': ['http://foo.com/bar/cel', 'http://foo.com/bar/cel/ona]''
       'activitySource': 'activities'
       }
```

A continuació detallarem els diferents paràmetres que es poden utilitzar, quins són obligatoris, i el tipus de valor que s'espera en cada un d'ells:

Paràmetres referents al MAX

- `username` (obligatori) - Nom d'usuari del MAX (El mateix que el LDAP *nom.cognom*)
- `oauthToken` (obligatori) - token oAuth de l'usuari del MAX
- `maxServerURL` (obligatori) - URL absoluta del servidor max a utilitzar
- `maxTalkURL` (obligatori) - Si desde el servei MAX no s'indica el contrari, és el mateix que `maxServerURL` acavat amb `/max`
- `readContext` (obligatori) - URI del context del qual volem mostrar-ne les activitats.
- `writeContexts` - `default: []` - Llista d'URIS de contextos alternatius on es publicaran les activitats. El context especificat a * `readContext`, formara sempre part automàticament d'aquesta llista.
- `activitySource` (obligatori)- Font de l'activitat. Pot ser `timeline` o `activities`.

- `activitySortOrder` - `default: activities` - Ordre que s'aplicarà a les activitats tant en mode timeline com en mode activities. Si és `activities` la ultima activitat generada sortirà la primera. SI és `comments` la ultima activitat on s'hagi fet un comentari sortirà la primera.
- `generatorName` (obligatori) - Nom que s'adjuntarà a les activitats generades des del widget, representant l'orígen de les activitats. Típicament serà el nom de l'aplicació on s'ha instal·lat el widget.

Paràmetres de la UI

- `UISection` - `default: timeline` - Secció a mostrar al inicialitzar el widget. Hi han dues opcions `timeline` per mostrar el fil d'activitat, i `conversations` per mostrar les converses privades.
- `avatarURLpattern` - Si no està especificat, el widget intentarà obtenir les imatges dels usuaris del propi max. Si l'aplicació vol utilitzar les seves propies imatges, pot proporcionar una url on es pugui proporcionar un paràmetre `{1}` amb el nom d'usuari, i que retorni la imatge de l'usuari o una imatge genèrica si no existeix l'usuari, d'una forma similar a algun d'aquests exemples:

  ```
  http://laMevaAplicacio.com/fotos/{1}
  http://laMevaAplicacio.com/fotos?usuari={1}
  ```

- `disableTimeline` - `default: false` - Posar-ho a `true` per deshabilitar el fil d'activitat
- `disableConversations` - `default: false` - Posar-ho a `true` per deshabilitar les converses
- `language` - `default: en` - Idioma de la interfície, disposa dels literals traduïts en Català (ca), Anglès (en) i Castellà(es).
- `literals` - Objecte javascript per definir literals personalitzats per l'aplicació. Hi ha dos casos d'ús:
  - Literals per un idioma que no *existeix per defecte*: S'han d'especificar **tots**
  - Literals per un idioma que *ja existeix*: S'han d'especificar només els que es volen sobreescriure.
    Els literals disponibles són:

```
{'new_activity_text': 'Escriu alguna cosa...',
 'activity': 'activitat',
 'conversations': 'converses',
 'conversations_list': 'llista de converses',
 'new_conversation_text': 'Cita a @algú per iniciar una conversa',
 'new_activity_post': "Publica",
 'toggle_comments': "comentaris",
 'new_comment_text': "Comenta alguna cosa...",
 'new_comment_post': "Comenta",
 'load_more': "Carrega'n més",
 'context_published_in': "Publicat a",
 'generator_via': "via",
 'search_text': "Busca...",
 'and_more': "i més...",
 'new_message_post':'Envia el missatge',
 'post_permission_unauthorized': 'No estàs autoritzat a publicar en aquest contexte',
 'post_permission_not_here': "No estas citant a @ningú"
}
```

Altres Paràmetres

- `maxRequestsAPI` - `default: jquery` - Api a utilitzar per les peticions al servidor MAX. Actualment només suporta jquery en aquesta versió.
- `enableAlerts` - `default: false` - Booleà per activar finestres emergents d'alerta quan succeeixi algun error. Útil per a depurar errors.

La lectura/escriptura de les activitats d'un contexte, venen donades pels permisos de subscripcio atorgats en el moment de subscriure l'usuari, i dels permisos per defecte del context.

1.10.5 Autenticació

La autenticació del widget es fa mitjançant un token oauth que s'ha de demanar al servidor https://oauth.upc.edu. Per demanar aquest token s'ha de fer la petició corresponent al servidor, i injectar el token juntament amb el nom d'usuari als paràmetres de configuració explicats anteriorment.

Com que es necessita tenir accés a les credencials de l'usuari per sol·licitar el token oauth, actualment el mètode vigent, implica que l'aplicació ha de implementar en el seu procés de login les següent accions en el moment que disposa del password de l'usuari:

- Demanar el token oAuth i emmagatzemar-lo en les bases de dades pròpies de l'aplicació, amb l'objectiu de només demanar-lo la primera vegada que un usuari es connecta a l'aplicació.
- Crear l'usuari al max, i subscriure'l als contextes oportuns si s'escau.

1.10.6 CORS - Cross Origin Resource Sharing

Les crides al MAX que es fan des del widget es van via peticions XHR des del navegador. Degut a restriccions de seguretat, per defecte els navegadors no permeten que una crida XHR interactuï amb dominis diferents del qual s'ha accedit. Per exemple, si hem carregat l'aplicació a `http://www.foo.com`, no podrem fer crides XHR a `http://www.bar.com`.

Per superar aquest obstacle, s'ha implementat l'estàndar CORS que permet fer aquestes accions, però no tots els navegadors ho suporten. De moment el sistema de reserva per tal d'assegurar el funcionament del widget en navegadors antics, necessita de dues coses:

- Definir una url continguda en el servidor de l'aplicació que fagi proxy de les peticions cap a la url del servidor MAX: Per exemple:

```
- Aplicació a http://www.foo.bar
- Servidor  MAX http://www.max.com
- http://www.foo.bar/max --> http://www.max.com
```

- Configurar el widget perquè utilitzi el redireccionament en casos que el navegador no suporti CORS:

```
{
 'maxServerURLAlias' : 'http://www.foo.bar/max'
}
```

1.10.7 Depuració d'errors

A part del paràmetre `enableAlerts` de la configuració, per poder esbrinar la causa de que no s'inicialitzi el widget, recomanem utilitzar les eines de desenvolupament natives disponibles en algunes navegadors com *Google Chrome* o plugins com *firebug* pe al *Firefox*. Bàsicament ens haurem de fixar en possibles errors javascript que aparegui a la consola d'errors, i a peticions XHR fallides. En aquest segon cas, ens interessara fixar-nos el el missatge d'error en format JSON que haurà retornat la petició fallida.