Màster Universitari en Programari Lliure

Projecte Final de Màster d'orientació professional
de l'àrea d'especialització "Administració de web
i de comerç electrònic"

IMPLEMENTACIÓ D'UN PROVEÏDOR DE SERVEI PAPI (PoA) LLEUGER EN PYTHON

Autor: **Javier Peña Blázquez**
Consultor: **Manel Zaera Idiarte**
Tutor extern: **Daniel García Franco**

Maig de 2013

Resum

Els **Sistemes *Single Sign-On*** o **Sistemes de Login Únic** (SSO) són sistemes d'autenticació que permeten als usuaris accedir a diversos recursos amb una sola instància d'identificació, es a dir, només és necessari autenticar-se una vegada per accedir a tots els serveis oferts.

PAPI és un protocol obert desenvolupat per RedIRIS que es fa servir per a implantar SSO. Es tracta d'un sistema d'autenticació i autorització que proporciona mecanismes per realitzar el control d'accés a aquells recursos que necessiten protegir certa informació i que estan distribuïts en una xarxa. El seu principal objectiu és mantenir l'autenticació com una qüestió local a l'organització a la qual pertany l'usuari, mentre que els recursos realitzen el procés d'autoritzar els accessos d'aquests usuaris.

PAPI, igual que qualsevol altre protocol SSO, defineix dos components principals, els quals en terminologia PAPI són coneguts com:

- **Servidor d'Autenticació (AS).** És el proveïdor d'identitat (IdP) i s'encarrega del procés d'autenticar als usuaris.
- **Punt d'Accés (PoA).** És el proveïdor de servei (SP) i s'encarrega de l'autorització o control d'accés als recursos protegits.

Així doncs, en PAPI, un PoA és un agent que permet protegir una aplicació web i integrar-la dintre d'un Sistema de Login Únic. A més, el PoA permet a les aplicacions web identificar als usuaris de forma centralitzada mitjançant la realització d'una petició d'autenticació al proveïdor d'identitat del SSO.

El protocol PAPI disposa d'implementacions software de referència realitzades en Perl, Java i PHP, pero fins ara la comunitat que fa servir Python com a llenguatge per al desenvolupament d'aplicacions web no disposava d'una implementació en Python d'aquest protocol.

L'objectiu principal d'aquest projecte ha estat realitzar el desenvolupament en Python d'un PoA lleuger, implementant la part del protocol PAPI que correspon al proveïdor de servei. El resultat d'aquesta implementació ha estat una llibreria Python que es farà servir des de les aplicacions web i que proporcionarà a aquestes una interfície senzilla per a realitzar la identificació dels usuaris i un control d'accés a la pròpia aplicació.

Paraules clau:
Sistemes de login únic, PAPI, proveïdor de servei, PoA, Python

Taula de continguts

Capítol 1 - Introducció

1.1 Proveïdor d'Accés a Punts d'Informació (PAPI)

Aquesta secció descriu l'evolució de PAPI des de la seva concepció com a producte fins a l'estat actual en el qual diverses implementacions del comportament del producte PAPI han provocat que PAPI es vegi com un protocol.

1.1.1 PAPI origen

PAPI va sorgir per la necessitat que tenien les biblioteques virtuals d'oferir accés per als seus usuaris als recursos digitals que aquestes van començar a adquirir a la fi dels anys 90. Els proveïdors d'aquests recursos realitzaven el control d'accés única i exclusivament per la IP d'origen dels usuaris. Aquest fet representava una trava important per a la comunitat científica o investigadora espanyola, ja que reduïa la mobilitat del personal, el qual solament disposava d'accés als recursos bibliogràfics des del seu lloc de treball habitual.

Arran d'aquesta situació va néixer el proxy de re-escriptura, per solucionar el problema de la mobilitat, la qual cosa va plantejar el problema del control d'accés per usuari a cada recurs, és a dir, ara que els usuaris podien accedir des de qualsevol ubicació, les biblioteques virtuals havien de controlar que accedien als seus recursos solament aquells usuaris amb permís per a tal efecte. Atès que el nombre de proveïdors de recursos era molt elevat, realitzar un control d'accés en cadascun d'ells suposava un esforç inabastable.

Per solucionar aquest nou problema, des de **RedIRIS**, Diego R. López i Rodrigo Castro van proposar un model d'autenticació i autorització que separava l'autenticació del control d'accés i que era ubic (per solucionar el problema de l'accés restringit a la IP d'origen). Aquest model va acabar convertint-se en una Infraestructura d'Autenticació i Autorització, on l'autenticació es realitzava en un únic punt central sota el control de cada organització (en aquest cas de cada biblioteca), ja que era l'organització la que posseïa les dades d'identitat i els atributs que determinaven el nivell d'accés d'un usuari. D'altra banda, l'autorització es realitzava en cada recurs al que es desitjava accedir mitjançant una “asserció” d'identitat que li arribava des del punt central d'autenticació, sobre la base de la identitat, els atributs rebuts de l'usuari i les regles de control d'accés que es defineixin per a cada recurs.

La infraestructura abans descrita s'implementà en RedIRIS en 2002 i se li donà el nom de **PAPI**, la qual es desplegà en el CSIC (Consell Superior de Recerques Científiques) convertint-se en la plataforma d'accés als recursos electrònics de la biblioteca virtual del CSIC.

1.1.2 PAPI evolució

Amb l'aparició dels conceptes de ***Single Sing-On*** (SSO) i federacions d'identitat digital (els quals ja defineix PAPI des de principis del 2000), PAPI evolucionà durant anys des d'una plataforma d'ús exclusiu per a les biblioteques virtuals, cap a una eina extremadament útil per realitzar el control d'accés a les aplicacions web. Aquest sistema permetia que l'usuari sols hagués d'autenticar-se una vegada per poder accedir a qualsevol aplicació que estigués protegida per PAPI, sempre dins del mateix entorn controlat.

A causa que la tecnologia o llenguatges de programació en els quals es desenvolupen les aplicacions web són molt diversos, RedIRIS començà a realitzar implementacions del comportament de PAPI (el qual originalment es desenvolupa en Perl com un mòdul intern del servidor web Apache) en altres llenguatges de programació com són PHP, Java, .Net.

En aquest punt és on es comença a parlar de PAPI com a protocol, ja que per realitzar aquestes altres implementacions (les quals es realitzen ja per persones diferents als autors originals) va ser necessari publicar amb molt més detall el comportament de PAPI, passant a crear-se la definició de PAPI independent de la seva implementació, és a dir, es crea el concepte de protocol PAPI. Si bé és veritat que els seus autors l'havien documentat com a tal des del principi, aquesta documentació estava molt lligada a la implementació original realitzada en Perl.

1.1.3 PAPI arquitectura i documentació

El propòsit d'aquest epígraf no és detallar per complet l'arquitectura lògica que defineix PAPI, sinó oferir una visió global, establir la terminologia i proporcionar els enllaços a la documentació oficial de PAPI.

PAPI s'ha convertit en un protocol obert que es fa servir per a implantar Sistemes de Login Únic (SSO). Es tracta d'un sistema d'autenticació i autorització que proporciona mecanismes per realitzar el control d'accés a aquells recursos que necessiten protegir certa informació i que estan distribuïts en una xarxa. El seu principal objectiu és mantenir l'autenticació com una qüestió local a l'organització a la qual pertany l'usuari, mentre que els recursos realitzen el procés d'autoritzar els accessos d'aquests usuaris [1, 2, 3].

Així doncs, PAPI és un sistema que facilita l'accés, a través d'Internet, a recursos d'informació que estan restringits a usuaris autoritzats. Els mecanismes d'autenticació emprats per identificar als usuaris s'han dissenyat per ser el més flexibles possible, permetent que cada organització empri un esquema d'autenticació propi, mantenint així les dades dins del seu propi àmbit, alhora que els proveïdors d'informació disposen de dades suficients per realitzar estadístiques. A més, els mecanismes de control d'accés són transparents per a l'usuari i compatibles amb els navegadors comunament emprats en qualsevol sistema operatiu. Atès que PAPI empra procediments HTTP estàndar, el seu ús per proveir serveis d'identitat digital i control d'accés no requereix de cap maquinari o programari específic, garantint als usuaris l'accés a qualsevol recurs d'informació al que tinguin dret.

L'arquitectura d'un sistema PAPI pot constar de tres components diferents, com es pot observar a la Figura 1:

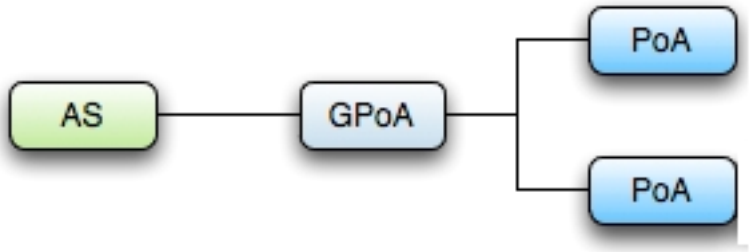


Figura 1: Arquitectura d'un sistema PAPI

- **Servidor d'autenticació** o **AS** de l'anglès *Authentication Server.* En la terminologia consensuada de SSO es coneix com **Proveïdor d'Identitat** o **IdP** de l'anglès *Identity Provider*. S'encarrega del procés d'autenticar a l'usuari, validant la seva identitat i associant-li una sèrie d'atributs perquè presente en els recursos protegits. Pot utilitzar diversos mètodes per autenticar als usuaris: consultar un fitxer de text, una base de dades, un sistema LDAP, etc.

- **Punt d'Accés** o **PoA** de l'anglès *Point of Access*. En la terminologia consensuada de SSO aquest component es conegut com **Proveïdor de Servei** o **SP** de l'anglès *Service Provider.* Realitza l'autorització de l'accés a un recurs protegit comprovant l'autenticació de l'usuari i els seus atributs. Així doncs, un PoA garantirà l'accés al recurs que protegeix sempre que algun servidor d'autenticació en el qual confia li proporcioni els atributs que indiquen que l'usuari s'ha autenticat prèviament, i utilitzant la informació addicional que l'AS li pot proporcionar per a l'autorització.

- **Grup de Punts d'Accés** o **GPoA** de l'anglès *Group Point of Access***.** Permet centralitzar les polítiques d'autorització d'una organització en un únic punt, al qual preguntaran els PoAs d'aquesta organització. Així doncs, permet personalitzar la quantitat d'informació de l'usuari que es proporciona als PoA, interposant-se entre aquests i l'AS, de manera que cada PoA rep exclusivament la informació necessària per a la seva autenticació i autorització. Es tracta d'un agregador de PoAs, en els quals aquests confien en lloc de l'AS, la qual cosa permet agilitar la configuració dels PoAs, centralitzant-la en aquest GPoA.

Pel que fa a la documentació de PAPI, en l'actualitat existeixen dos llocs web on hi ha àmplia informació sobre el protocol, el primer d'ells és http://papi.rediris.es/ mantingut per RedIRIS, institució on es va desenvolupar i es va promoure PAPI durant molt temps, i el segon és http://www.papisoftware.net mantingut per la comunitat que va desenvolupar PAPI i que no està lligada en l'actualitat a RedIRIS. Tots dos llocs disposen de documentació fiable i actualitzada de PAPI.

La documentació del protocol en castellà es pot trobar en el següent enllaç:

http://www.papisoftware.net/doku.php?id=documentacion:protocol

Per a una documentació del protocol més lligada a les implementacions originals de PAPI, es pot consultar el següent document:

http://papi.rediris.es/rep/papi_protocol_detailed.pdf

1.1.4 Versió lleugera de PAPI

PAPI en néixer com a producte, i en consolidar-se posteriorment com a protocol de SSO, és creat amb una sèrie de característiques que no són essencials per al protocol i que dificulten molt o si més no fan que la seva implementació sigui molt costosa, així com el seu desplegament i posterior ús per tercers.

És per això que recentment s'han realitzat implementacions fàcils, simples o lleugeres del protocol PAPI, on es deixa llibertat al implementador per prendre decisions de disseny segons el seu criteri en punts on el protocol no és molt rígid.

Aquests punts de lliure disseny són:

- Utilització o no del component GPoA. La tendència actual és integrar el GPoA dins o molt prop de l'AS .
- Ús de la cookies Hcook i Lcook. La tendència actual és no utilitzar-les i mantenir la mateixa informació amb una cookie de sessió d'usuari .
- Implementació del control d'accés en funció dels atributs rebuts. La tendència actual és cedir aquesta responsabilitat a l'aplicació més que al PoA .
- Implementació d'URL de Pas o zones de l'aplicació no protegides. La tendència actual és evitar aquest tipus de comportaments que poden donar lloc a confusió
- Ignorar el comportament WAYF (*Where are you from*) en els PoAs, cedint aquesta responsabilitat als GPoAs .

En aquest projecte s'ha implementat un PoA lleuger seguint aquestes noves tendències. En l'apartat dedicat a l'anàlisi del sistema es descriu amb més detall quin és el comportament del PoA lleuger que s'ha desenvolupat.

1.2 Objectius del projecte

Els principals objectius que es van definir a l'inici d'aquest projecte van ser els següents:

- **Analitzar el protocol PAPI.** Estudiar detalladament el funcionament del protocol PAPI, posant especial èmfasi en la definició dels missatges del PoA, ja que la llibreria ha d'implementar les peticions que realitza aquest agent i processar les respostes que rep.

- **Dissenyar i programar una biblioteca en Python que implementi un PoA de PAPI**. Com ja s'ha comentat, el protocol PAPI disposa d'implementacions software de referència realitzades en Perl, Java i PHP, pero fins ara la comunitat que fa servir Python com a llenguatge per al desenvolupament d'aplicacions web no disposa d'una implementació en Python d'aquest protocol. El principal objectiu d'aquest projecte es desenvolupar una llibreria que implementi el comportament d'un PoA del protocol PAPI. Aquesta llibreria s'ha d'implementar en el llenguatge de programació Python i podrà ser utilitzada pels programadors que empren aquest llenguatge en el desenvolupament de les seves aplicacions web.

1.3 Estat de la tècnica

Originalment PAPI es va desenvolupar en Perl com un mòdul intern del servidor web Apache. Aquest tipus de PoA permet usar els mecanismes propis del servidor web per permetre l'accés a l'aplicació o denegar-lo utilitzant una simple pàgina d'error *HTTP 403 Forbidden*.

Posteriorment, a causa de la gran varietat de llenguatges de programació que poden ser utilitzats per al desenvolupament d'aplicacions web, es van realitzar implementacions del comportament de PAPI en Java, PHP, C i .Net.

Actualment existeixen un gran nombre d'implementacions actives i estables de cadascun dels components de PAPI, que es distribueixen independentment encara que sota una mateixa versió de la distribució oficial. L'última versió estable és PAPI 1.5 i la següent taula mostra què components s'inclouen dins de la mateixa [1]:

	Versión	Servidor de autenticación	GPoA	PoA	Proxy PAPI	Tecnología
AuthServer	1.5	X				Perl
PAPI::PoA	1.5		X	X	X	
easyGPoA	1.0		X			PHP
easyPoA	1.0			X		
icGPoA	1.0	X	X			
phpPoA	2.0			X		
OpenPAPIFilter	1.0			X		Java
OpenPAPI-Realm	1.0			X		
jicGPoA	1.0	X	X			
PAPIFilter	1.0			X		
mod_papi	1.0			X		C

Figura 2: Taula amb les implementacions existents del protocol PAPI

En tots els casos es tracta d'aplicacions amb llicències lliures associades (GPL o LGPL).

No obstant això, el protocol PAPI no disposa de cap component implementat en Python. Per tant, en aquest projecte es preten realitzar el desenvolupament en Python d'un PoA PAPI, per tal que la comunitat que fa servir aquest llenguatge per al desenvolupament d'aplicacions web puga integrar-les en un sistema PAPI.

1.4 Programari lliure utilitzat en el projecte

En aquest apartat es citen els principals components de programari lliure utilitzats durant la realització del projecte:

- Sistema operatiu **GNU/Linux**, distribució **Debian 6**. És el S.O que s'ha utilitzat per dur a terme tant el desenvolupament de la llibreria com la documentació de tot el projecte.

- Llenguatge de programació **Python**. Ha estat el llenguatge que s'ha utilitzat per implementar tota la llibreria. En concret s'ha utilitzat la versió 2.6.

- Entorn de desenvolupament integrat **Eclipse** més mòdul **PyDev**. És l'IDE i el plug-in que s'han utilitzat per implementar tot el codi del projecte.

- Llibreria **M2Crypto**. És una llibreria per Python que ha estat necessària per utilitzar algunes de les funcions criptogràfiques que incorpora.

- Servidor web **Apache**. És el servidor web que s'ha utilitzat per provar el funcionament de la llibreria desenvolupada.

- Navegadors **Firefox** i **Google Chrome**. Són els navegadors web que s'han utilitzat per realitzar proves de funcionament de la llibreria.

- **Umbrello**. És l'eina de modelatge que s'ha utilitzat per realitzar els diagrames UML de les fases d'anàlisis i disseny del projecte.

- **Planner**. És l'enina que s'ha utilitzat per a crear els diagrames de Gantt.

- **OpenOffice Writer**. És la processador de textos que s'ha utilitzat per redactar la major part de la documentació relacionada amb aquest projecte.

- **Gimp**. S'ha utilitzat per manipular algunes de les imatges d'aquesta memòria.

- **EasyPoA**. Es tracta d'una llibreria que implementa un PoA lleuger en PHP. Ha estat desenvolupada pel tutor extern del projecte. Ha sigut de molta utilitat de cara a implementar algunes de les parts del codi d'aquest projecte.

1.5 Feina de l'estudiant

Durant la realització d'aquest projecte les labors dutes a terme per l'estudiant han estat diverses i les podem agrupar com segueix:

- En primer lloc, es va realitzar un “pla de projecte” amb la finalitat d'organitzar temporalment les diferents tasques que s'havien de dur a terme.

- A continuació es va elaborar un estudi de viabilitat del projecte.

- Posteriorment es va procedir a estudiar detalladament el funcionament del protocol PAPI, consultant i estudiant les fonts citades en l'apartat 1.1.3 i la documentació proporcionada pel tutor extern.

- Una vegada estudiat el protocol, es va procedir a realitzar l'anàlisi de la biblioteca a desenvolupar. Per a això es van definir tots els casos d'ús i requisits que havia de complir.

- Seguidament es va passar a la fase de disseny del sistema, on es van elaborar diferents tipus de diagrames UML per obtenir els models i especificacions que defineixen el sistema.

- A continuació es va procedir a desenvolupar la llibreria i es van realitzar diferents proves de funcionament. En aquest cas, a més de l'anàlisi i el disseny realitzat, també va ser molt útil la consulta del codi de la llibreria EasyPoA implementada en PHP.

- Finalment, es va procedir a integrar la llibreria en un entorn SSO real.

A més de tot l'anterior, l'alumne ha mantingut una comunicació constant amb el tutor extern, a qui se li han anat plantejant dubtes i qüestions de diferent índole durant tot el projecte. També s'ha mantingut informat al consultor de la UOC dels avanços que s'anaven realitzant.

1.6 Estructura de la memòria

La resta d'aquest document presenta la següent estructura:

- En el capítol 2 s'inclou el pla de treball que es va elaborar inicialment. En aquest pla apareix la divisió del projecte en tasques i la planificació temporal de les mateixes, fent ús d'un diagrama de Gantt.

- El capítol 3 presenta l'estudi de viabilitat del projecte.

- En el capítol 4 s'inclou l'anàlisi del sistema, amb la definició detallada dels requisits i els casos d'ús.

- En el capítol 5 es troba el disseny del sistema, amb la definició de l'arquitectura del sistema i la revisió dels casos d'ús.

- El capítol 6 es centra a descriure les fases de desenvolupament, implantació i manteniment del projecte.

- Finalment, el capítol 7 inclou les conclusions, objectius aconseguits, les possibilitats d'ampliació i les principals dificultats trobades.

Capítol 2 – Pla de treball

En aquest capítol es presenta el pla de treball que es va elaborar a l'inici del projecte i que s'ha anat seguint durant la seva realització. Aquest pla inclou les tasques en què es divideix el projecte, les fites marcades, la temporalització de cada tasca (establint la seva durada i precedències) i el càlcul total d'hores de feina que comporta. Per a això s'ha creat un diagrama de Gantt.

2.1 Divisió del projecte en tasques

A continuació s'enumeren les diferents tasques en les quals s'ha decidit dividir el projecte:

1. **Anàlisi del protocol PAPI.**

   1.1 **Estudi del protocol PAPI**. Estudi en profunditat de la documentació tècnica del protocol PAPI per comprendre completament el seu funcionament. Com ja s'ha comentat abans, s'ha de prestar especial atenció a la definició dels missatges que envia i rep un agent PoA en PAPI.

   1.2 **Documentació del protocol PAPI.** Documentar detalladament com funciona el protocol PAPI, creant per aquest fi uns diagrames de seqüència on es descrigui el comportament dels diferents agents involucrats en aquest protocol. També s'han de crear diagrames per descriure el diversos casos de ús.

2. **Disseny de l'estructura de la biblioteca.**

   2.1 **Disseny de les classes.** Realització del disseny de les classes que compondran la llibreria. Cal tenir en compte que la biblioteca ha de proporcionar una interfície ben definida perquè pugui ser utilitzada sense problemes pels desenvolupadors.

3. **Implementació de la biblioteca.**

   3.1 **Desenvolupament de les classes que conformen la biblioteca.** A partir del disseny realitzat en la fase anterior, s'han d'implementar utilitzant el llenguatge de programació Python les diferents classes que conformen la biblioteca. Per a això, s'ha de muntar prèviament l'entorn de desenvolupament (configuració de l'IDE, instal·lació i configuració d'un servidor web amb suport per Python...).

   3.2 **Proves de funcionament i rendiment.** Preparació d'un entorn de proves que validi el correcte funcionament de la biblioteca. Si és possible també es realitzaran proves de rendiment de la implementació realitzada.

3.3 **Documentació de la biblioteca.** Documentació tant de la biblioteca desenvolupada com de les proves realitzades. A més, elaboració d'una guia d'ús perquè els programadors sàpiguen com utilitzar aquesta llibreria.

4. **Implantació de la biblioteca en un entorn real.**

4.1 **Implantació de la llibreria en un SSO.** Implementació d'una petita aplicació web en Python que faci ús de la biblioteca desenvolupada i que es pugui desplegar dins d'una infraestructura SSO que utilitzi el protocol PAPI.

2.2 Divisió temporal de les tasques

Durada i precedència entre les actividats

En la següent taula es mostra la durada establerta per a les diferents tasques del projecte i les dependències que existeixen entre elles.

Activitat	Durada	Dependències
1. Anàlisi del protocol PAPI	**40h**	**-**
1.1. Estudi del protocol PAPI	20h	-
1.2. Documentació del protocol PAPI	20h	1.1
2. Diseny de l'estructura de la biblioteca	**30h**	**1.2**
2.1. Diseny de les classes	30h	1.2
3. Implementació de la biblioteca	**100h**	**2.1**
3.1. Desenvolupament de las classes que conformen la biblioteca	60h	2.1
3.2. Proves de funcionament y rendimient	20h	3.1
3.3. Documentació de la biblioteca	20h	3.1, 3.2
4. Implantanció de la biblioteca en un entorn real	**15h**	**3.3**
4.1 Implantació de la llibreria en un SSO	15h	3.3

Com es pot observar, la durada total de les tasques s'ha ajustat a 185 hores, que és el temps que s'ha de dedicar a les pràctiques externes.

Fites

Perquè el consultor extern pugui realitzar un correcte seguiment del treball realitzat, s'ha decidit establir una fita al final de cadascuna de les quatre fases, més la fita corresponent al final del projecte.

Diagrama de Gantt

En el següent diagrama de Gantt es mostren les tasques en què s'ha descompost el projecte, la durada de cadascuna d'elles, les dependències entre elles i les fites establertes. S'ha decidit que cada dia es dedicarà 1 hora efectiva a la realització del projecte de pràctiques externes. A més, s'ha establert una setmana de treball on els dies laborables són de dilluns a dissabte.

Nombre	Trab...
Anàlisi del protocol PAPI	**40d**
Estudi del protocol PAPI	20d
Documentació del protocol PAPI	20d
Fita 1	
Disseny de l'estructura de la biblioteca	**30d**
Disseny de les classes	30d
Fita 2	
Implementació de la biblioteca	**100d**
Desenvolupament de les classes que conformen la biblioteca	60d
Proves de funcionament i rendiment	20d
Documentació de la biblioteca	20d
Fita 3	
Implantació de la biblioteca en un entorn real	**15d**
Implantació de la llibreria en un SSO	15d
Fita 4	
Fita 5 - Entrega final	

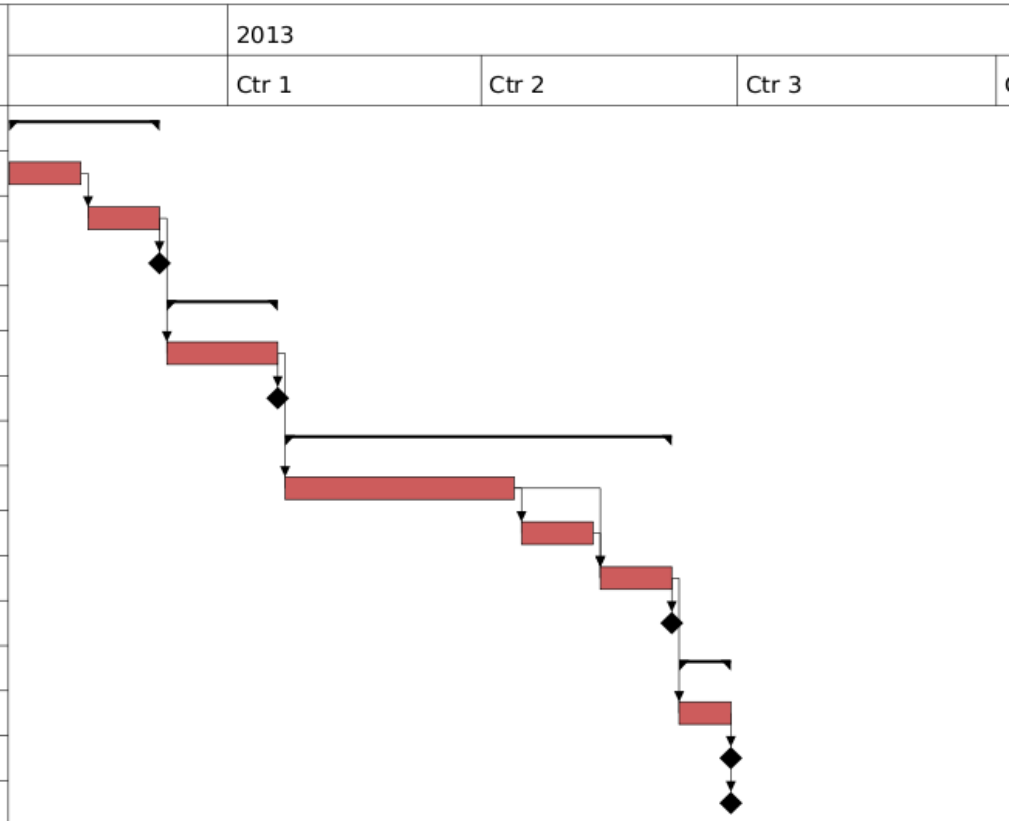


Figura 3: Diagrama de Gantt del projecte

Finalment, en la següent taula es poden observar les dates d'inici i fi planificades de cada tasca sobre la base de la durada assignada a cadascuna d'elles i del calendari laboral establert.

WBS	Nombre	Inicio	Fin	Duración
1	**Anàlisi del protocol PAPI**	**oct 15**	**dic 7**	**40d**
1.1	Estudi del protocol PAPI	oct 15	nov 9	20d
1.2	Documentació del protocol PAPI	nov 12	dic 7	20d
2	Fita 1	dic 7	dic 7	N/D
3	**Disseny de l'estructura de la biblioteca**	**dic 10**	**ene 18**	**30d**
3.1	Disseny de les classes	dic 10	ene 18	30d
4	Fita 2	ene 18	ene 18	N/D
5	**Implementació de la biblioteca**	**ene 21**	**jun 7**	**100d**
5.1	Desenvolupament de les classes que conformen la biblioteca	ene 21	abr 12	60d
5.2	Proves de funcionament i rendiment	abr 15	may 10	20d
5.3	Documentació de la biblioteca	may 13	jun 7	20d
6	Fita 3	jun 7	jun 7	N/D
7	**Implantació de la biblioteca en un entorn real**	**jun 10**	**jun 28**	**15d**
7.1	Implantació de la llibreria en un SSO	jun 10	jun 28	15d
8	Fita 4	jun 28	jun 28	N/D
9	Fita 5 - Entrega final	jun 28	jun 28	N/D

Figura 4: Taula amb la duració i les dates de inici i fi de cada tasca

2.3 Seguiment i control del projecte

Durant l'execució del projecte es va realitzar un seguiment per comprovar el grau de compliment dels objectius establerts i de la planificació inicial realitzada. Encara així, certes dificultats van provocar retards en la finalització d'algunes tasques. Les principals variacions que ha sofert el projecte respecte de la planificació inicial són les següents:

- Retard de 30 hores en la implementació de la biblioteca. Això va ser a causa de les dificultats que van sorgir per implementar sessions d'usuari en Python, i per les dificultats per trobar una llibreria criptogràfica per Python que complís amb les necessitats requerides pel protocol PAPI (xifrar un text amb la clau privada i desxifrar-lo amb la clau pública).
- A més, la data de lliurament final va ser establerta per al 3 de Juny, en lloc de per al 28 de Juny com s'hi havia previst.

Tot això va provocar que durant els últims dos mesos (Abril i Maig) l'alumne hagués de dedicar més d'una hora diària per finalitzar les fites 3 i 4 i el lliurament final dins dels terminis establerts.

Capítol 3 – Estudi de viabilitat

En el present capítol es realitza l'estudi de viabilitat del projecte a desenvolupar. En aquest estudi s'han tingut en compte en tot moment les característiques peculiars que presenta aquest projecte. És necessari destacar que la solució a desenvolupar ha vingut directament proposada per l'empresa de pràctiques externes en la definició del projecte. En concret es tracta de desenvolupar una llibreria que implementi el comportament específic d'un component del protocol PAPI, en un llenguatge de programació determinat com és Python.

3.1 Establiment de l'abast del sistema

En primer lloc es realitzarà una **descripció general** de les necessitats que s'han de resoldre amb la realització d'aquest projecte.

Com ja s'ha comentat, l'objectiu fonamental d'aquest projecte és dissenyar i programar una llibreria que implementi el comportament d'un agent *PoA* del protocol PAPI. Aquesta llibreria s'implementarà en el llenguatge de programació Python i podrà ser utilitzada pels programadors que utilitzen aquest llenguatge en el desenvolupament de les seves aplicacions web.

Així doncs, els programadors podran utilitzar aquesta llibreria per integrar les seves aplicacions web desenvolupades en Python en un Sistema de Login Únic basat en PAPI. Posteriorment, quan els usuaris intentin accedir a un recurs protegit, aquesta llibreria s'encarregarà de comprovar l'autenticació de l'usuari i d'autoritzar l'accés a aquests recurs.

A continuació es mostra un diagrama amb les accions que involucren la intervenció de la llibreria a desenvolupar:

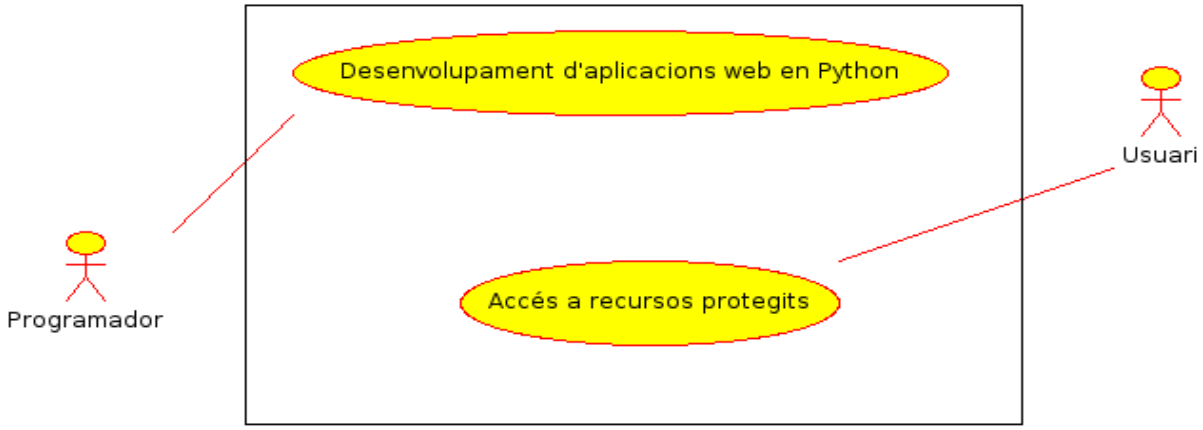


Figura 5: Diagrama amb les accions que involucren la intervenció de EasyPyPoA

Des del punt de vista **econòmic**, la realització d'aquest projecte ha d'implicar la menor despesa possible (a poder ser despesa nul·la), ja que es tracta d'un projecte a desenvolupar per un estudiant d'universitat que no va a percebre cap tipus compensació econòmica per la seva realització.

Des del punt de vista **tècnic**, malgrat que es tracta d'un projecte complex que requereix del coneixement de diverses tecnologies i protocols (protocols PAPI i HTTP, tecnologia de xifrat, programació web en Python, etc), la persona encarregada del mateix és personal qualificat que gaudeix i afronta aquest tipus de reptes tècnics sense molts problemes.

Des del punt de vista **legal**, cal tenir en compte que el projecte es desenvolupa com a part d'una assignatura del Master Universitari en Programari Lliure. Per tant, serà indispensable que el projecte i la seva documentació acabin publicant-se sota algun tipus de llicència lliure, ja que és un requisit imposat per la universitat. Per tant, les tecnologies a utilitzar per al desenvolupament de la llibreria també hauran de ser lliures.

Des del punt de vista **operatiu**, el desenvolupament d'aquest projecte no suposa cap problema o inconvenient per al funcionament de cap empresa o entitat.

D'altra banda, el desenvolupament d'aquesta llibreria pot afectar a **altres projectes** de tecnologies de la informació ja en curs o que es pensen posar en marxa, de la següent manera:

- Aquelles empreses o institucions que posseeixin aplicacions web programades en Python tindran la possibilitat de controlar l'accés a recursos protegits utilitzant una infraestructura basada en el protocol PAPI per implantar un sistema SSO.

- Aquelles empreses que decideixin emprendre un nou projecte per implantar un sistema SSO utilitzant el protocol PAPI, ara tindran la possibilitat de fer-ho programant les seves aplicacions web en Python, ja que fins ara només podien fer-ho en PHP, Perl, o Java.

Finalment, com el projecte no es desenvolupa per a cap empresa en concret, sinó que es tracta d'una llibreria que es pot utilitzar en multitud d'entitats, els departaments que es veuran afectats per la seva implantació variaran d'unes institucions a unes altres.

3.2 Estudi de la situació actual

Com ja s'ha comentat, el projecte a desenvolupar no està destinat a ser utilitzat en una sola empresa, sinó que es tracta d'una llibreria genèrica la finalitat de la qual és que qualsevol entitat que desitgi implantar un Sistema de Login Únic fent ús del protocol PAPI, pugui fer-ho utilitzant aplicacions web implementades en Python. Així doncs, en l'estudi de la situació actual no té sentit analitzar la situació en la qual es troba el sistema d'informació de cap empresa, ja que la llibreria no està destinada a ser implantada en alguna empresa en particular.

No obstant això, si que hem de tenir en compte que la llibreria implementa un dels agents que componen la infraestructura d'un sistema PAPI, en concret l'agent PoA. Així doncs, quan una empresa o institució decideixi utilitzar la llibreria que anem a desenvolupar, serà perquè desitja instal·lar i configurar una arquitectura PAPI de serveis web.

En un sistema PAPI hauran d'existir com a mínim els següents sistemes [1]:

- Un **Servidor d'Autenticació** o AS. Serà el sistema encarregat d'autenticar als usuaris en un únic punt de la xarxa, de la forma que es desitgi, i de recol·lectar informació sobre els mateixos per proporcionar-la als punts d'accés. Aquest sistema emet assercions o atributs amb la informació necessària per verificar la identitat dels usuaris. Per dur a terme el procés d'autenticació es pot consultar un fitxer amb els usuaris, consultar una bases de dades, utilitzar un sistema LDAP, etc. Cada entitat haurà de decidir el procediment que segueix per a l'autenticació dels usuaris.

- Els **Punts d'Accés** o PoA. Són els encarregats de protegir l'accés a un recurs concret. Utilitzen les assercions o atributs dels AS per verificar la identitat dels seus usuaris i obtenir informació addicional sobre els mateixos. Els PoA poden funcionar de múltiples formes, depenent dels requisits de l'aplicació que es vol protegir. En el nostre cas concret, si una entitat tria la nostra llibreria per implantar un PoA, l'autenticació dels usuaris es durà a terme des d'una aplicació escrita en Python i mitjançant una simple cridada a una funció.

També és possible que existeixin *agrupacions de punts d'accés* o *GPoA*. Aquest sistema seria un PoA que no protegiria a cap recurs en concret, sinó que donaria accés a un grup de PoAs amb característiques comunes de validació. Aquest sistema permet personalitzar la quantitat d'informació de l'usuari que es proporciona als PoA, interposant-se entre aquests i l'AS.

Així doncs, quan una entitat decideixi implantar una arquitectura PAPI utilitzant la nostra llibreria, haurà d'analitzar les característiques concretes dels sistemes on vagi a instal·lar cadascun dels components anteriors. Les necessitats de cada sistema dependran del programari que es decideixi instal·lar. Per exemple, per instal·lar el servidor d'autenticació PAPI desenvolupat originalment per RedIRIS serà necessari que el sistema disposi del servidor web Apache amb el suport per Perl. A més, si desitgem dur a terme l'autenticació dels usuaris usant un sistema LDAP, també serà necessari instal·lar el programari necessari.

Per poder utilitzar la nostra llibreria, el sistema on vagin a executar-se les aplicacions web haurà de disposar d'un servidor web amb intèrpret per Python.

Si analitzem l'estat del protocol PAPI en l'actualitat, podem observar que la comunitat disposa d'implementacions software de referència realitzades en Perl, Java, PHP, C i .Net, però no existeixen implementacions per a altres llenguatges de programació. Així doncs, un possible punt de millora per a aquest protocol seria realitzar implementacions dels diferents components en més llenguatges de programació. El principal objectiu d'aquest projecte serà doncs intentar contribuir en la millora de PAPI desenvolupant una llibreria

que implementi un agent PoA en Python i que permeti a la comunitat que fa servir aquest llenguatge per al desenvolupament d'aplicacions web que pugui fer ús d'aquest protocol en les seves aplicacions.

3.3 Definició dels requisits del sistema

A continuació es descriuen de manera general els **requisits** que haurà de complir el projecte. En cada requisit s'inclou una qualificació que representa la seva prioritat (nombre entre 0 i 100).

Requisits tècnics:

- (100) La llibreria haurà de proporcionar la funcionalitat pròpia d'un agent PoA del protocol PAPI.
- (100) La llibreria haurà de desenvolupar-se en Python perquè els programadors que utilitzen aquest llenguatge per al desenvolupament de les seves aplicacions web puguin fer ús de la mateixa.
- (90) La llibreria haurà de funcionar en qualsevol servidor web que tingui suport per Python.
- (90) Atès que la llibreria enviarà i rebrà dades sensibles (dades d'identitat i assercions d'autenticació dels usuaris) s'hauran d'utilitzar les tecnologies de xifrat adequades per transmetre aquesta informació.
- (70) La llibreria haurà de complir amb totes les convencions de codificació de programes establertes en la “Style Guide for Python Code” .[1]

Requisits operatius:

- (90) La llibreria haurà d'oferir una interfície ben definida que permeti als desenvolupadors utilitzar-la sense problemes.
- (80) La llibreria haurà d'anar acompanyada de la documentació necessària perquè els desenvolupadors aprenguin a utilitzar-la.
- (80) La llibreria haurà de seguir una estructura clara i ordenada que faciliti el manteniment de la mateixa i les possibles aportacions d'altres desenvolupadors.

Requisits legals:

- (100) La llibreria i la seva documentació s'hauran de publicar sota una llicència lliure.

1 http://www.python.org/dev/peps/pep-0008/

- (80) Totes les biblioteques utilitzades en el desenvolupament de la llibreria també hauran de tenir associada una llicència lliure.

Requisits econòmics:

- (70) Les despeses econòmiques associades al desenvolupament d'aquesta llibreria hauran de ser el més baixes possibles.

3.4 Estudi de les alternatives de solució

Com ja s'ha comentat anteriorment, en l'actualitat la comunitat que utilitza Python com a llenguatge per al desenvolupament d'aplicacions web no disposa d'una implementació en aquest llenguatge que permeti utilitzar el protocol PAPI.

Per tant, l'única **alternativa de solució** existent és desenvolupar una llibreria que permeti fer ús d'aquest protocol en el llenguatge Python. A més, com un requisit indispensable del projecte és que la llibreria es publiqui sota una llicència lliure, tampoc existeix la possibilitat de pensar en un desenvolupament privat. En tractar-se d'una solució feta a mesura, podrà complir perfectament amb tots els requisits tècnics i operatius establerts en l'apartat anterior. A més, com la llibreria es publicarà sota una llicència lliure i per al seu desenvolupament també s'utilitzaran eines lliures, els requisits legals i econòmics es compliran sense problemes.

3.5 Valoració de les alternatives

Análisi de costos/beneficis del projecte

En principi, per desenvolupar l'única alternativa de solució proposada no és necessari realitzar cap desemborsament econòmic, ja que:

- El llenguatge de programació Python té associada la llicència PSFL (Python Software Foundation License) que és una llicència de programari lliure permissiva, a l'estil de la llicència BSD.

- El llenguatge de programació Python disposa de totes les biblioteques lliures necessàries per al desenvolupament d'aquest projecte (llibreries per a encriptació, desenvolupament d'aplicacions web, etc)

- Existeixen servidors web lliures com Apache que tenen suport per interpretar aplicacions web realitzades en Python.

- Existeixen multitud de frameworks lliures per al desenvolupament d'aplicacions en Python.

- També existeix programari lliure per a la realització de les proves unitàries.

No obstant això, en tractar-se d'un desenvolupament a mesura, si que serà necessari dedicar diverses hores de treball per al seu disseny i implementació. En el capítol 2, on s'ha elaborat el “Pla de treball”, s'ha estimat que la realització d'aquest projecte suposarà un temps aproximat de 185 hores de treball.

Riscos del projecte

Els principals riscos associats a l'alternativa de solució contemplada són els següents:

1. Evolució del llenguatge Python cap a una nova versió del llenguatge que no sigui compatible cap a enrere.

2. Desaparició de la comunitat que manté el projecte Python o alguna de les llibreries utilitzades.

3. En estranyes situacions es podria donar algun cas de falta de suport d'alguna llibreria o del propi llenguatge Python.

Pal·liació de riscos del projecte

1. Si aquesta situació es dona, serà necessari desenvolupar una nova llibreria que siga compatible amb la nova versió del llenguatge. A més, s'haurà de mantenir durant un temps l'evolució de les dues versions de la llibreria.

2. Valorant l'estabilitat i l'abast de la comunitat formada entorn al llenguatge de programació Python, és molt probable que si l'equip principal de desenvolupament desapareix altres membres de la comunitat segueixin endavant amb el projecte a causa del gran nombre d'usuaris que l'utilitzen arreu de tot el món

3. Es podria contractar el suport d'una empresa externa que es comprometi a resoldre els possibles problemes que puguin sortir. No obtant això, en els portals web oficials del llenguatge Python i de les llibreries necessaries, existeix abundant informació que permet resoldre la major part dels problemes.

3.6 Selecció de la solució

Com ja hem comentat abans, per a aquest projecte sols s'ha definit una alternativa de solució possible, i per tant només hi ha una possibilitat d'elecció. A més, la proposta de solució venia ja suggerida per l'empresa de pràctiques externes.

A més, es tracta d'una solució que compleix amb tots els requisits definits i que pràcticament no presenta cap risc que posi en perill el desenvolupament del projecte. Per tant, es decideix desenvolupar una nova llibreria que implementi el comportament d'un PoA del protocol PAPI, en el llenguatge de programació Python.

Capítol 4 – Anàlisi del sistema

Per dur a terme l'anàlisi del sistema s'especificarà detalladament l'única alternativa de solució que s'ha presentat i s'ha donat per vàlida durant l'estudi de viabilitat del projecte. Cal recordar que es tracta de desenvolupar una nova llibreria que implementi el comportament d'un PoA del protocol PAPI, en el llenguatge de programació Python.

4.1 Definició del sistema

En aquest apartat es descriu la llibreria a desenvolupar amb més nivell de detall del que es va fer en l'estudi de viabilitat. A més, s'estableix com s'ha de comunicar aquesta llibreria amb altres sistemes i quins són els usuaris més representatius en el seu ús.

En primer lloc, per descriure els aspectes més importants de la llibreria a desenvolupar es determinen els requisits exactes del sistema prenent com a punt de partida la descripció dels requisits feta en l'estudi de viabilitat.

Requisits exactes del sistema

La llibreria ha de complir els requisits següents:

- La llibreria ha d'implementar la funcionalitat pròpia d'un agent PoA del protocol PAPI. És a dir, la llibreria ha d'implementar la part del protocol PAPI que es correspon al proveïdor de servei.

- El PoA PAPI implementat ha de ser '**lleuger**' [5]. Per aconseguir que la llibreria sigui un component lleuger, s'han de restringir determinades característiques comunes dels PoAs, tal i com es va introduir a l'apartat 1.1.4. En concret, aquelles característiques que impliquen que el PoA siga lleuger són les següents:

    - No s'ha d'implementar el missatge *ATTREQ* del protocol PAPI, per la qual cosa el PoA desenvolupat només ha de poder comunicar-se amb GPoAs, i no amb un Servidor d'Autenticació directament (AS).

    - El PoA desenvolupat tampoc ha de tenir la capacitat de realitzar filtres d'accés basats en les dades rebudes en les assercions, de manera que la responsabilitat de realitzar el control d'accés recaurà en l'aplicació protegida.

    - No s'han d'utilitzar les cookies Hcook i Lcook. S'ha de mantenir la mateixa informació amb una cookie de sessió d'usuari .

- La llibreria ha de ser desenvolupada en el llenguatge de programació Python, ja que la finalitat principal d'aquest projecte és que la comunitat que usa Python com a llenguatge per al desenvolupament d'aplicacions web, disposi d'un PoA PAPI implementat en aquest llenguatge que permeti protegir les seves aplicacions.

- La llibreria ha de funcionar en qualsevol servidor web que tingui suport per Python. Per tant, per poder fer ús de la llibreria l'equip haurà de tenir instal·lat el llenguatge Python i un servidor web amb suport per al mateix. Si la nostra llibreria fa ús d'alguna biblioteca diferent de la biblioteca estàndard, s'especificarà en la documentació com un requisit per al funcionament del PoA desenvolupat.

- Tal com s'especifica en el protocol PAPI, la comunicació entre el PoA i el GPoA s'ha de dur a terme fent ús del protocol HTTP. En concret, s'utilitza el mètode GET per enviar els missatges, i les respostes pròpies d'HTTP per a enviar les contestacions.

- Com la llibreria enviarà i rebrà dades sensibles (dades d'identitat i assercions d'autenticació dels usuaris) s'han d'utilitzar tecnologies de xifrat adequades per transmetre aquesta informació. En concret, tal com especifica el protocol PAPI, per a la transmissió de les dades sensibles s'utilitzarà el sistema criptogràfic de clau pública RSA.

- La llibreria ha d'oferir una interfície ben definida que permeti als desenvolupadors utilitzar-la sense problemes. Per fer-ho s'han de definir classes que proporcionin els mètodes públics necessaris per comprovar si un usuari té accés a l'aplicació web protegida.

- La llibreria ha de seguir una estructura clara i ordenada que faciliti el manteniment de la mateixa i les possibles aportacions d'altres desenvolupadors.

- La llibreria ha d'anar acompanyada de la documentació necessària perquè els desenvolupadors aprenguin a utilitzar-la. A més s'inclouran exemples d'ús.

- La llibreria ha de complir amb totes les convencions de codificació de programes establertes en la "*Style Guide for Python Code*" .

- La llibreria i la seva documentació s'han de publicar sota una llicència lliure. En concret, per al programari s'ha decidit que s'utilitzarà la llicència *Apache 2 License* i per a la documentació una llicència *Creative Commons BY SA*.

- Totes les biblioteques i ferramentes utilitzades en el desenvolupament de la llibreria també han de tenir associada una llicència lliure.

D'altra banda, per definir el sistema també és necessari establir l'entorn tecnològic del projecte.

Entorn tecnològic del sistema

L'entorn tecnològic que s'utilitzarà en el desenvolupament de la llibrería serà el següent:

- Sistema operatiu: **GNU/Linux**, distribució **Debian 6**. La llibreria es desenvoluparà i provarà fent ús d'aquest sistema operatiu. No obstant això, una vegada implementada també es realitzaran proves en altres sistemes operatius com Windows.

- Servidor web: **Apache**. Per provar la llibreria serà necessari un servidor web amb suport per Python com Apache. S'ha decidit utilitzar aquest servidor perquè és el més estès, encara que també es realitzaran proves en altres servidors web.

- Desenvolupaments a mida. Com ja s'ha comentat, la llibreria es desenvoluparà per complet fent ús del llenguatge de programació **Python**.

- Entorn de desenvolupament. S'utilitzarà algun IDE lliure com **Eclipse** amb **PyDev** per desenvolupar la llibreria. Per al desenvolupament de les aplicacions web en Python que facin ús de la llibreria implementada es pot utilitzar algun framework web com Django.

Per completar la descripció del sistema, a continuació es fa referència al conjunt d'estàndards i normes que cal seguir en la implementació de la llibreria:

Normes que es poden seguir en la implementació

Les normes i estàndards que cal seguir en la implementació de la llibreria son els següents:

- Tot el programari necessari per a la implementació de la llibreria s'instal·larà i configurarà seguint les instruccions donades a la pàgina oficial de cada projecte o des dels repositoris de la distribució GNU/Linux escollida.

- Com ja s'ha comentat, la llibreria ha de complir amb totes les convencions de codificació de programes establertes en la "*Style Guide for Python Code*" .

- A més, la llibreria ha de seguir una estructura clara i ordenada que faciliti el seu manteniment i les possibles aportacions d'altres desenvolupadors. També s'han d'incloure comentaris que documentin el codi de forma completa: descripció de classes, mètodes, atributs, etc.

Una vegada descrit el sistema, el següent pas consisteix a identificar als usuaris que intervenen en la definició de requisits i en l'acceptació definitiva. En aquest cas concret, cal tenir en compte les peculiars característiques d'aquest projecte, que no està destinat a ser implantat en cap empresa concreta, i que es desenvolupa com a projecte final d'un màster universitari.

Identificació d'usuaris

El personal involucrat en la definició de requisits i acceptació de la solució final és:

- El **consultor extern**: Daniel García. Director tecnològic de l'empresa PRiSE dedicada a oferir serveis en matèria de Seguretat, Privacitat i Identitat Digital. Ha desenvolupat projectes similars a la llibreria que es pretén implementar, per la qual cosa té un alt grau de coneixement del protocol PAPI.

- El **consultor de la UOC**: Manel Zaera. Encarregat de revisar tot el treball dut a terme.

4.2 Establiment de requisits

En aquest apartat es completaran els requisits definits anteriorment. Després d'estudiar i d'analitzar detalladament tota la documentació tècnica del protocol PAPI i tenint en compte les consideracions realitzades per part del consultor extern, s'ha arribat a un nivell de comprensió elevat pel que fa al funcionament del protocol PAPI i a les funcionalitats que ha d'incorporar la llibreria. Així doncs, en aquest apartat es realitzarà una descripció tècnica detallada dels requisits que ha de complir la llibreria a desenvolupar.

Com hem comentat abans, la llibreria ha d'implementar la funcionalitat pròpia d'un agent PoA del protocol PAPI. No obstant això, la llibreria creada ha de ser 'lleugera', per la qual cosa s'han de restringir determinades característiques comunes dels PoAs. En l'apartat anterior ja s'han indicat quines són les funcionalitats que no han de ser implementades per tal d'aconseguir una llibreria 'lleugera':

- No s'ha d'implementar el missatge *ATTREQ* del protocol PAPI, per la qual cosa el nostre PoA només podrà parlar amb GPoAs, i no tindrà la capacitat d'interrogar a un AuthServer sobre els usuaris.

- El PoA implementat no ha de ser capaç de realitzar filtres d'accés basats en les dades rebudes en les assercions, per la qual cosa la responsabilitat de realitzar el control d'accés recaurà en l'aplicació protegida.

- Les dades dels usuaris autenticats es mantindran en una cookie de sessió.

Així doncs, el PoA implementat només ha de generar/processar els missatges especificats en la documentació del protocol PAPI que:

- S'encarreguen de la petició d'autorització: *CHECK* i *CHECKED.* D'aquesta forma, la llibreria s'encarregarà de generar missatges *CHECK* per comprovar si els usuaris tenen drets d'accés al recurs protegit; i de processar missatges *CHECKED* de forma lleugera, amb la finalitat de realitzar el control d'accés.

- S'encarreguen del procés de *logout*: *PAPISIGNOFFREQ*, *PAPILOGGEDOUT* i *PAPILOGOUT*. El PoA s'encarregarà de generar missatges *PAPISIGNOFFREQ* per indicar al GPoA que ha començat el procés de *logout*, i com a resposta procesarà missatges *PAPILOGGEDOUT.* A més, s'encarregarà de procesar els missatges *PAPILOGOUT* que li arribin d'un GPoA, quan un altre component hagi començat el procés de *logout*, responent amb un missatge *PAPILOGGEDOUT*.

Aquests són els principals requisits funcionals que ha de complir la llibreria a desenvolupar. A continuació s'especifiquen tots els detalls tècnics relacionats amb els mateixos [1, 2].

Requisit "Comunicació entre el PoA i el GPoA"

En primer lloc descriurem com s'ha de dur a terme la comunicació entre els diferents components del protocol PAPI (AS, GPoA i PoA). Per comunicar-se entre ells, els components de PAPI intercanvien missatges entre si. No obstant això, ho fan d'una manera una mica especial, ja que aquests missatges no són enviats directament entre els diferents components, sinó que la comunicació es duu a terme a través del navegador web de l'usuari. Els missatges del protocol PAPI es transmeten utilitzant mètodes GET o POST sobre HTTP mentre que, d'altra banda, les respostes són redireccions 302 d'HTTP, incloent a més cookies en el cas que l'usuari hagi d'emmagatzemar algun tipus d'informació útil per al protocol.

Vegem a través d'un exemple com es duu a terme la comunicació entre els diferents components. Suposem que un usuari accedeix a un component anomenat "*Web A*". Si *Web A* vol enviar un missatge a un altre component, per exemple anomenat "*Web B*", el navegador realitzaria les següents peticions:

1. El navegador envia una petició al component *Web A*:

```
http://web/A?arg1=X&arg2=Y
```

2. El component *Web A* retorna la següent resposta:

```
HTTP/1.1 302 Found
Date: Sun, 07 Dec 2012 16:21:02 GMT
Server: Apache/1.3.31 (Unix)
Location: http://web/B?arg3=Z
Transfer-Encoding: chunked

0
```

3. Finalment, el navegador de l'usuari, en rebre un missatge de resposta 302 (missatge de redirecció), torna a enviar automàticament una nova petició al component indicat en la capçalera *Location*, sent en aquest cas:

```
http://web/B?arg3=Z
```

En la Figura 6 es pot veure un diagrama de seqüència on es poden observar els missatges intercanviats entre els diferents components citats en l'exemple.

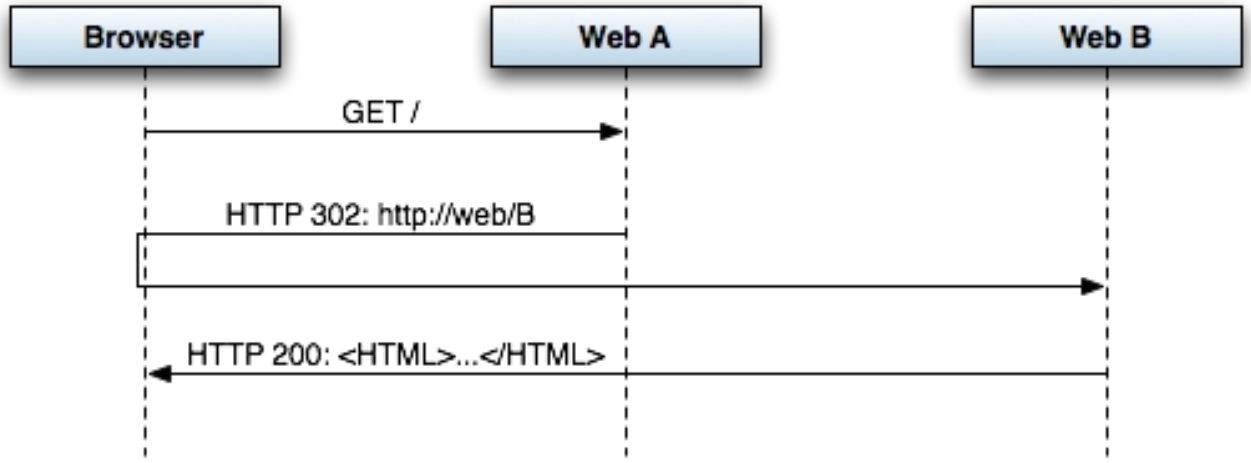


Figura 6: Comunicació entre els distints components del protocol PAPI

Abans de definir com s'ha de comportar el nostre PoA quan un usuari intenta accedir a un recurs protegit, vegem en el diagrama de la Figura 7 el conjunt de missatges que defineixen el protocol PAPI durant el procés d'accés a un recurs:

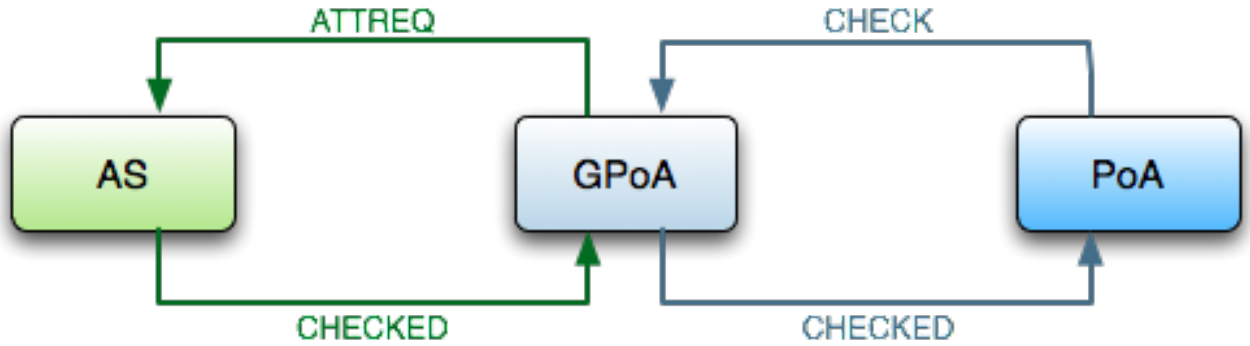


Figura 7: Missatges que defineixen el protocol de PAPI durant l'accés a un recurs protegit

Com es pot observar, en el nostre cas el PoA només es comunicarà amb el GPoA a través dels missatges *CHECK* i *CHECKED*. D'altra banda, el GPoA i l'AS es comunicaran utilitzant els missatges *ATTREQ* i *CHECKED*.

Requisit "Accés a un recurs protegit i petició d'autorització: CHECK - CHECKED"

Ara anem a analitzar quin ha de ser el comportament de l'agent PoA que hem d'implementar en els diferents casos que es poden donar:

1. Suposem que un usuari intenta accedir a una aplicació web protegida sense proporcionar cap asserció al PoA. Es tractarà doncs d'un usuari no autenticat. En aquest cas, el PoA enviarà un missatge *CHECK* al GPoA per comprovar si l'usuari té potencialment drets d'accés en aquest PoA. El GPoA, després de dur a terme

les accions oportunes, respondrà amb un missatge *CHECKED*. En cas que l'usuari tingui drets d'accés, el PoA obtindrà informació d'autenticació i una asserció d'atributs. En cas que el GPoA vulgui indicar que l'usuari no té autorització per accedir a aquest PoA o per assenyalar que ha ocorregut algun tipus d'error durant la seva identificació, en l'asserció retornada s'indicarà que s'ha produït un error. En la Figura 8 es poden observar els missatges intercanviats entre el navegador de l'usuari, el PoA i el GPoA en aquest cas.

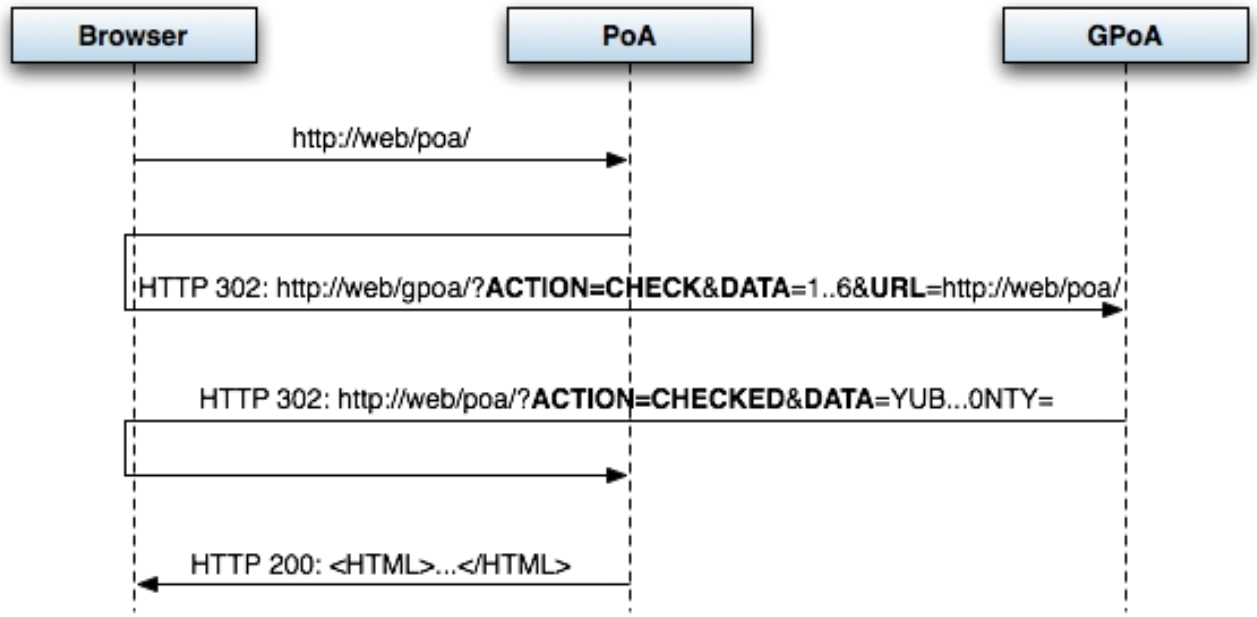


Figura 8: Comunicació CHECK-CHECKED entre el PoA i el GPoA

Missatge CHECK

El format del missatge és:

```
http://%gpoa-url%?ACTION=CHECK&DATA=%value%&URL=%PoA-url%
```

On,

- **gpoa-url**: és l'adreça web del GPoA al que se li mana el missatge.
- **ACTION**: és el tipus de missatge que s'envia al GPoA, CHECK en este cas.
- **DATA**: és una cadena alfanumèrica que ha de ser retornada en la resposta.
- **URL**: és l'adreça de resposta a la qual ha de fer la redirecció 302 d'HTTP el GPoA.

Resposta CHECKED a un CHECK

El GPoA enviarà un missatge *CHECKED* com a resposta al missatge anterior. En el cas que l'usuari tingui drets d'accés al PoA, se li enviarà una resposta correcta amb el següent format:

```
http://%PoA-url%?ACTION=CHECKED&DATA=%value%
```

On,

- **PoA-url**: és l'adreça web que el PoA ha enviat en el camp URL del missatge *CHECK*.
- **ACTION**: és el tipus de resposta que envia, en aquest cas *CHECKED*.
- **DATA**: és el camp que conté la informació que requeria el PoA. El valor de DATA s'obté després de passar a base 64 el valor obtingut de xifrar la següent cadena de text amb la clau privada RSA del GPoA:

  `userAssertion:expiryTime:currentTime:KEY`

  On,

  - **userAssertion**: és l'asserció de l'usuari pel PoA que ha enviat la petició.
  - **expiryTime**: és el temps en segons des de l'1 de Gener de 1970 en el qual l'asserció deixa de ser vàlida.
  - **currentTime**: és el temps en segons des de l'1 de Gener de 1970 en què ha estat generada l'asserció.
  - **KEY**: és el valor del paràmetre DATA que es va rebre en el missatge *CHECK*.

  El format de l'asserció és el següent:

  `%LLISTA_ATRIBUTS%@%AS-ID%`

  La llista d'atributs (%LLISTA_ATRIBUTS%) en una asserció no té un format definit, encara que la informació que conté segueix la següent estructura:

  Nom d'atribut - Llista de valors de l'atribut

  Dins d'una mateixa federació o *Single Sign-On* ha d'establir-se un acord sobre els separadors de cadascun dels elements que componen l'asserció. Es recomana utilitzar els següents:

  - Separador d'atributs: ,
  - Separador atribut-valor: =
  - Separador de valors en un atribut: |

  D'aquesta forma quedaria una asserció de la següent manera:

  `name1=valorA,name2=valorB|valorc@PAPI_AS_ID`

En cas que el GPoA vulgui indicar que l'usuari no té autorització per accedir a aquest PoA o per indicar que ha ocorregut algun tipus d'error durant la seva identificació el valor de *userAssertion* ha de ser '**ERROR**'.

D'altra banda, evidentment el PoA haurà de desxifrar la cadena continguda en el camp DATA del missatge *CHECKED* si vol tenir accés a les dades rebudes. Per a això haurà d'utilitzar la clau pública del GPoA.

2. Suposem ara que un usuari intenta accedir a una aplicació web protegida, i proporciona al PoA una asserció que indica que ja s'ha autenticat prèviament. En aquest cas, el PoA haurà de comprovar que l'asserció és vàlida verificant que no ha expirat i que l'usuari està autoritzat per accedir a aquest recurs. En cas que sigui vàlida l'usuari podrà accedir al recurs protegit. Si la sessió ha expirat s'haurà d'enviar un nou missatge *CHECK* al GPoA. Si es tracta d'un usuari no autoritzat es denegarà l'accés al recurs protegit.

Requisit "Procés de *Logout*"

Anem a analitzar quin ha de ser el comportament de l'agent PoA que hem d'implementar en els diferents casos que es poden donar:

1. Quan un usuari desitja sortir d'una aplicació protegida per un PoA PAPI, en la qual té una sessió iniciada, ha d'iniciar el procés de Logout. En aquest cas, el PoA ha d'enviar un missatge *PAPISIGNOFFREQ* al GPoA. Aquest missatge s'utilitza per inicialitzar la fase de Logout en PAPI. L'objectiu és que un PoA pugui indicar a aquell component que li va proveir d'informació de l'usuari que vol iniciar el procés de *Logout*. En la Figura 9 es poden observar els missatges intercanviats entre el navegador de l'usuari, el PoA i el GPoA en aquest cas.

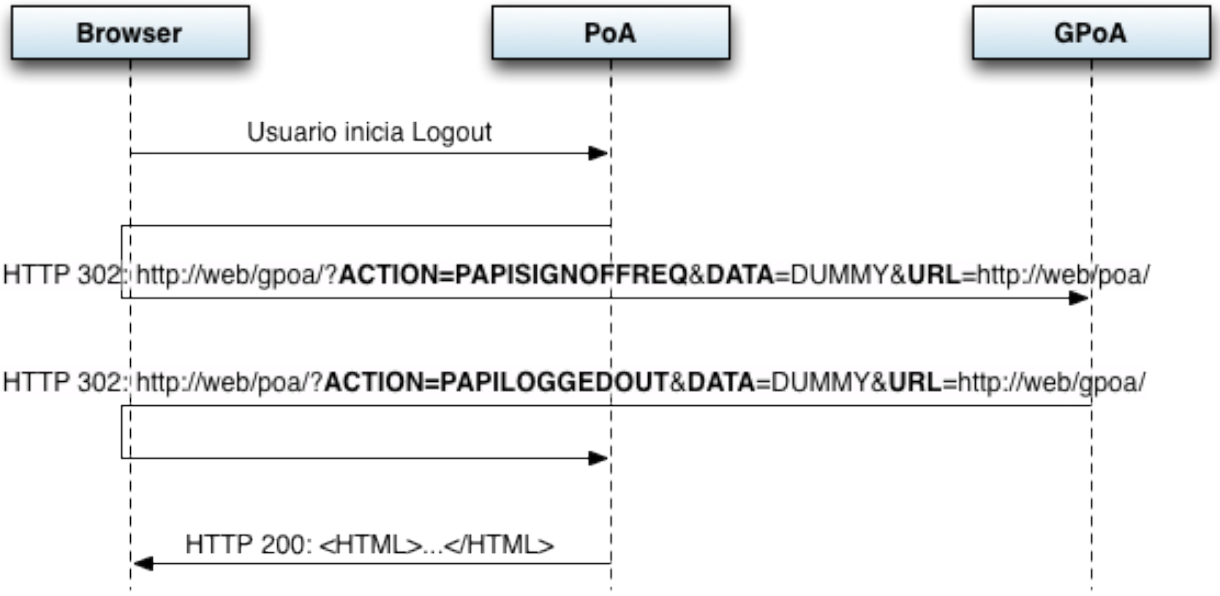


Figura 9: Comunicació PAPISIGNOFFREQ-PAPILOGGEDOUT entre el PoA i el GPoA

Missatge PAPISIGNOFFREQ

El format del missatge és:

`http://%elem-url%?ACTION=PAPISIGNOFFREQ&DATA=DUMMY&URL=%PoA-url%&POA=%PoA-id%&PAPIOPOA=%PoA-url%`

On,

- **elem-url**: és l'adreça web del GPoA al que se li mana el missatge.
- **ACTION**: és el tipus de missatge que s'envia al GPoA, *PAPISIGNOFFREQ* en este cas.
- **DATA**: és una cadena que no té cap funció en el missatge.
- **URL**: és la URL de tornada on enviar el missatge *PAPILOGGEDOUT*.
- **POA**: es l'ID del PoA. [PARÁMETRO OPCIONAL]
- **PAPIOPOA**: és la URL del PoA original que va començar el procés de *LogOut* [*PARÀMETRE OPCIONAL*]

Resposta PAPILOGGEDOUT a un PAPISIGNOFFREQ

El GPoA envia un missatge *PAPILOGGEDOUT* com a resposta al missatge anterior:

`http://%elem-url%?ACTION=PAPILOGGEDOUT&DATA=DUMMY&URL=%GPoA-url%&PAPIOPOA=%PoA-url%`

On,

- **elem-url**: és l'adreça web del PoA (proveïdor de servei) al que se li mana el missatge.
- **ACTION**: és el tipus de missatge que s'envia com a resposta, en este cas *PAPILOGGEDOUT*
- **DATA**: és una cadena que no té cap funció en el missatge.
- **URL**: és la URL d'on ve el missatge de resposta (de cara a identificar qui t'està dient que ha tancat la sessió). [*PARÀMETRE OPCIONAL*]
- **PAPIOPOA**: és la URL del PoA original que va començar el procés de *LogOut* [*PARÀMETRE OPCIONAL*]

2. Una altra situació pot donar-se quan l'usuari tanca la seva sessió en una altra aplicació integrada en el SSO, la qual cosa provoca que el IdP PAPI enviï un missatge de *Logout* a la resta de PoAs PAPI perquè aquests tanquin la seva sessió i indiquin a l'aplicació que ha de tancar la seva sessió. En aquest cas, el GPoA envia un missatge *PAPILOGOUT*. Aquest missatge s'utilitza per indicar als components que se li hagi enviat informació d'usuari que han de tancar la sessió de l'usuari. D'aquesta forma, aquest missatge només serà possible d'un GPoA a un PoA. Finalment, els PoA envien un missatge *PAPILOGGEDOUT* com a resposta al missatge anterior. En la Figura 10 es poden observar els missatges intercanviats

entre el navegador de l'usuari, el PoA i el GPoA en aquest cas.

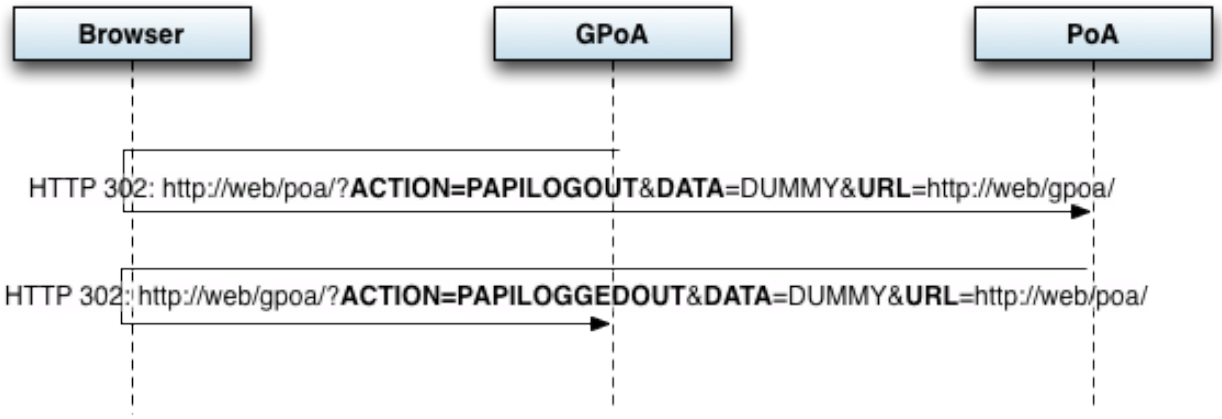


Figura 10: Comunicació PAPILOGOUT-PAPILOGGEDOUT entre el PoA i el GPoA

Missatge PAPILOGOUT

El format del missatge és:

```
http://%elem-url%?ACTION=PAPILOGOUT&DATA=DUMMY&URL=%url
%&PAPIOPOA=%PoA-url%
```

On,

- **elem-url**: és l'adreça web del PoA al que se li mana el missatge.
- **ACTION**: és el tipus de missatge que s'envia, en este cas *PAPILOGOUT.*
- **DATA**: és una cadena que no té cap funció en el missatge.
- **URL**: és la URL de tornada on enviar el missatge *PAPILOGGEDOUT*.
- **PAPIOPOA**: és la URL del PoA original que va començar el procés de *LogOut* [*PARÀMETRE OPCIONAL*]

Resposta PAPILOGGEDOUT a un PAPILOGOUT

El PoA envia un missatge *PAPILOGGEDOUT* com a resposta al missatge anterior:

```
http://%elem-url%?ACTION=PAPILOGGEDOUT&DATA=DUMMY&URL=%GPoA-
url%&PAPIOPOA=%PoA-url%
```

On,

- **elem-url**: és l'adreça web del PoA al que se li mana el missatge.
- **ACTION**: és el tipus de missatge que s'envia com a resposta, en este cas *PAPILOGGEDOUT*
- **DATA**: és una cadena que no té cap funció en el missatge.
- **URL**: és la URL d'on ve el missatge de resposta (de cara a identificar qui

t'està dient que ha tancat la sessió). [*PARÀMETRE OPCIONAL*]

- **PAPIOPOA**: és la URL del PoA original que va començar el procés de *LogOut* [*PARÀMETRE OPCIONAL*]

Escenari complet de *Logout*

En el següent diagrama es pot apreciar un escenari complet de *LogOut* on un PoA inicia el procés de *LogOut* i es propaga per tot el Single Sign-On fins que rep un missatge de confirmació que tot ha anat correctament:

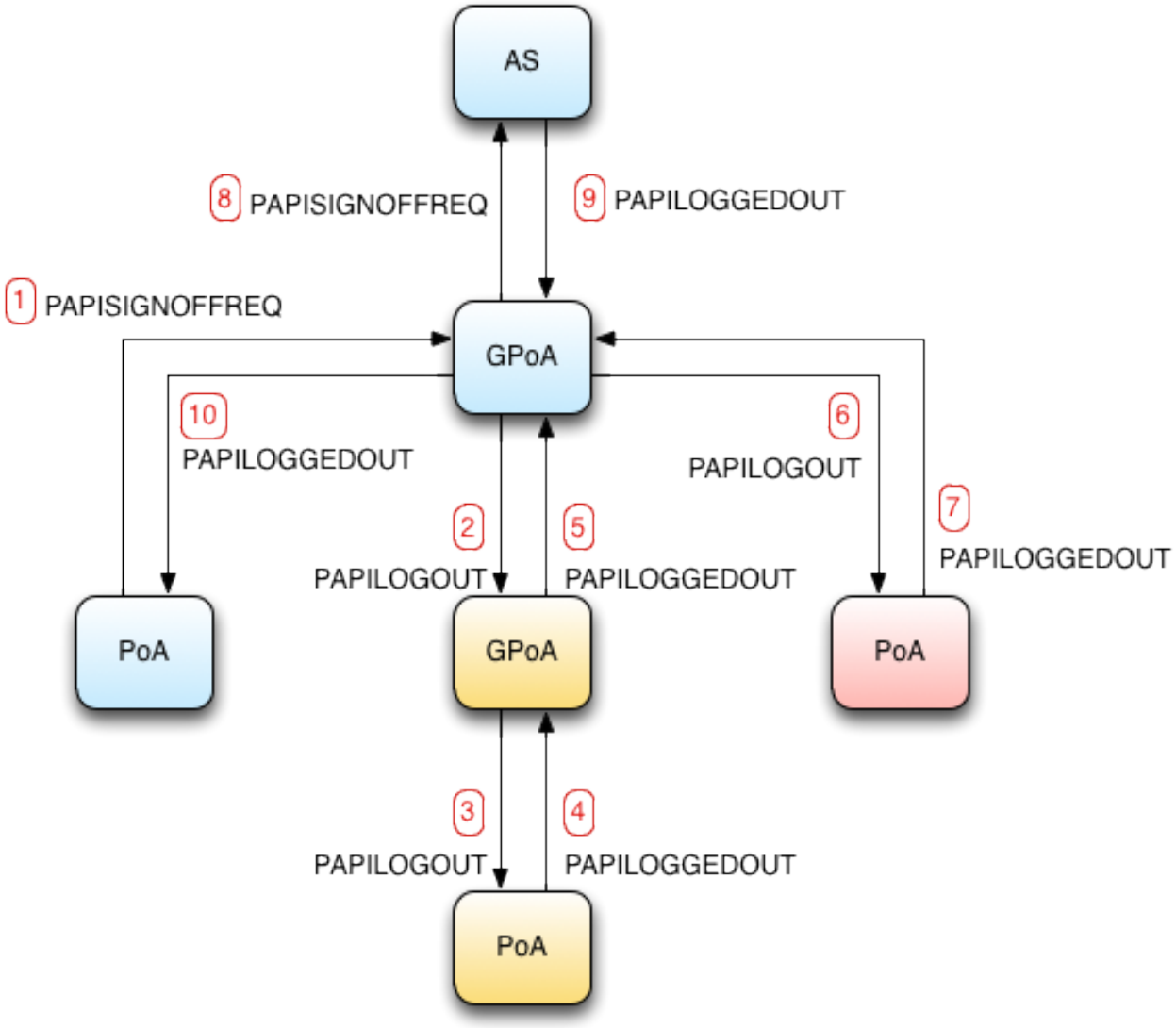


Figura 11: Escenari complet de LogOut

Una vegada descrits els principals requisits funcionals que ha de complir la llibreria, a continuació s'especifiquen el casos d'ús d'aquests requisits, per tenir una visió clara de la descripció del problema i de la interacció dels usuaris amb el sistema.

Cas d'ús "Protecció i accés a un recurs protegit"

Els usuaris de la institució que protegeixi els seus recursos fent ús d'una arquitectura PAPI, intentaran accedir a ells des de fora de la institució i a través d'Internet. En aquest moment, el PoA haurà de protegir l'accés a aquests recursos. Si l'usuari no s'ha autenticat prèviament, el PoA es posarà en contacte amb el GPoA per obtenir les assercions de l'usuari i verificar la seva identitat. En cas que tot sigui correcte, el PoA donarà permís a l'usuari perquè accedeixi al recurs protegit. Per la seva banda, els programadors utilitzaran la llibreria desenvolupada per protegir aplicacions web escrites en Python que es trobin dins d'un SSO basat en PAPI.

Context: Un usuari desitja accedir per primera vegada a una aplicació protegida per un PoA lleuger PAPI.

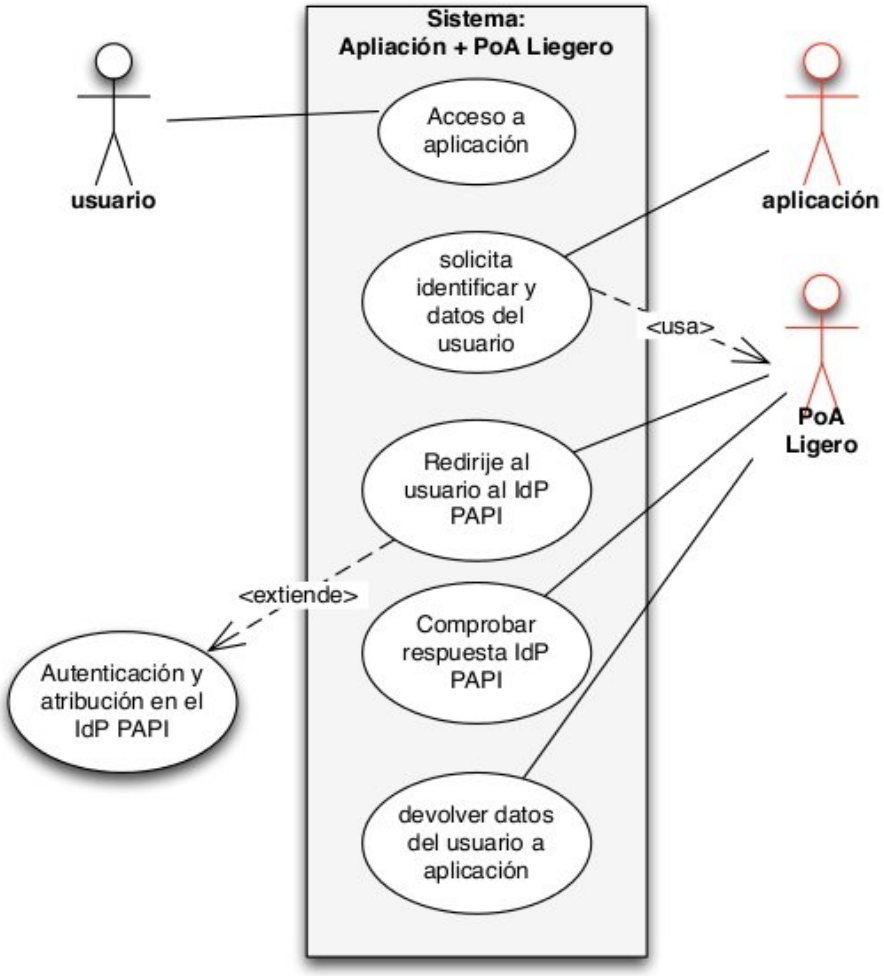


Figura 12: Cas d'ús procés de login quan un usuari intenta accedir per primera vegada a un recurs protegit

Context: Un usuari desitja accedir de nou a una aplicació protegida per un PoA lleuger PAPI. En aquest cas l'usuari ja s'ha autenticat previament.

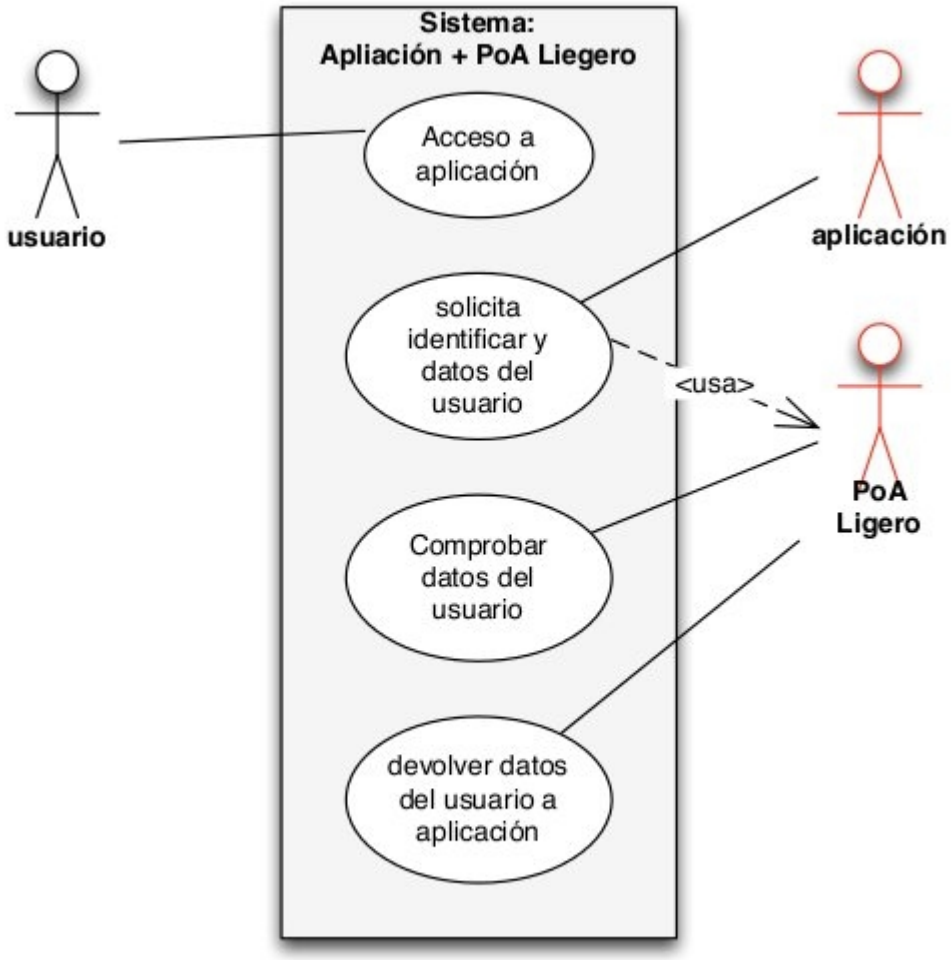


Figura 13: Cas d'ús accés a un recurs protegit per part d'un usuari ja autenticat

Cas d'ús "Procés de *logout* en PAPI"

Quan un usuari desitja sortir d'una aplicació protegida per un PoA PAPI, en la qual té una sessió iniciada, ha d'iniciar el procés de Logout. En els següents diagrames de casos d'ús es poden observar les dues situacions que es poden donar per a que un PoA PAPI sol·liciti a l'aplicació protegida tancar la sessió que es manté establerta.

Context: Un usuari desitja sortir d'una aplicació protegida per un PoA lleuger PAPI, en la qual té una sessió iniciada.

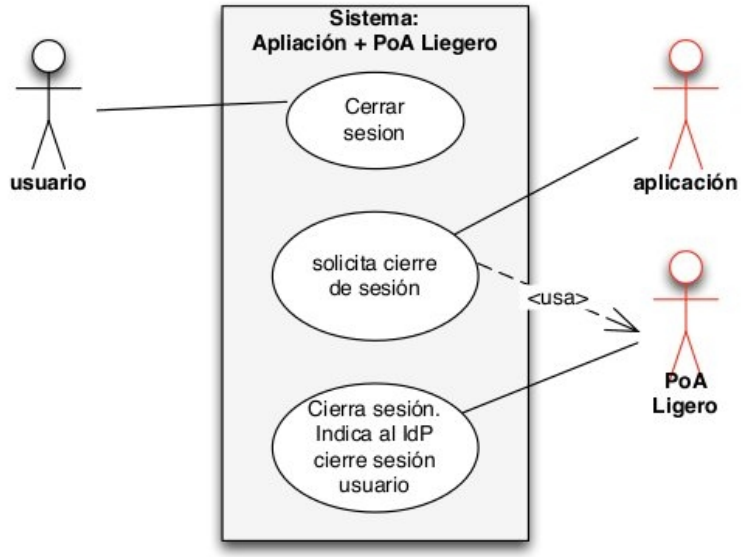


Figura 14: Cas d'us sol·licitud de tancament d'una sessió per part de l'usuari

Context: L'usuari tanca la seva sessió en una altra aplicació integrada en el SSO, la qual cosa provoca que el IdP PAPI enviï un missatge de Logout al PoA lleuger PAPI perquè aquest tanqui la seva sessió i indiqui a l'aplicació que ha de tancar la seva sessió.

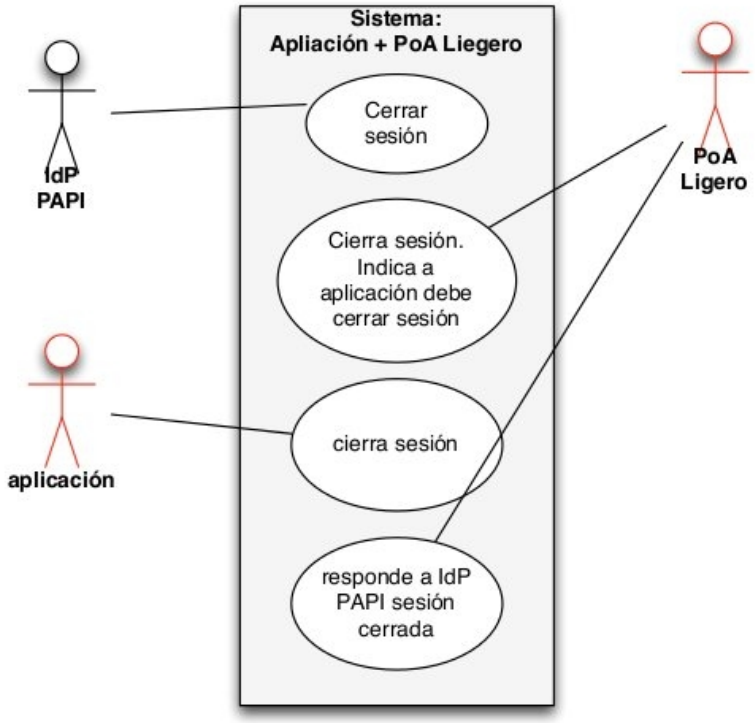


Figura 15: Cas d'ús tancament d'una sessió iniciat en una altra aplicació integrada en el SSO

En aquesta secció s'han descrit detalladament les característiques més importants dels missatges que envia i rep un agent PoA, que és el component de PAPI que s'ha d'implementar en la llibreria. Si es desitja conèixer més informació sobre el format i la funció dels missatges que s'intercanvien la resta de components és possible accedir a la secció de "*Documentació*" de la URL http://www.papisoftware.net o d'http://papi.rediris.es.

La resta dels requisits que ha de complir la llibreria ja han estat establerts amb el suficient nivell de detall en l'apartat anterior.

4.3 Definició d'interfícies d'usuari

En aquesta fase de l'anàlisi s'han d'especificar com seran les diferents interfícies que hi haurà entre el sistema que descrivim i els usuaris. El primer pas en la definició d'aquestes interfícies es definir els perfils d'usuaris que utilitzaran el sistema.

Perfils d'usuaris

La biblioteca que s'ha de desenvolupar la utilitzaran principalment els programadors d'aplicacions web en Python. En concret, utilitzaran la llibreria per protegir l'accés a les aplicacions web que hagin d'estar integrades en una arquitectura PAPI. En general aquests usuaris tindran les característiques següents:

- Estaran familiaritzats amb la programació d'aplicacions web.
- Coneixeran tots els paradigmes propis de la programació orientada a objectes.
- Coneixeran el llenguatge de programació Python i sabran utilitzar-ho per programar aplicacions web.

Tenint en compte aquestes característiques, aquests usuaris no haurien de tenir cap tipus de problema a utilitzar de forma adequada la llibreria per protegir les seves aplicacions web.

D'altra banda, quan un usuari de la institució que utilitzi PAPI intenti accedir a un recurs protegit, les funcions de la llibreria seran executades. No obstant això, com tots els mecanismes de control d'accés són transparents per a l'usuari, aquest no mantindrà cap contacte directe amb la biblioteca.

A continuació s'especifiquen els principis generals de la interfície d'usuari.

Principis generals de la interfície d'usuari

La biblioteca tindrà les següents característiques:

- En tractar-se d'una biblioteca destinada a ser utilitzada pels programadors, haurà d'oferir una sèrie de funcions o mètodes públics que puguin ser utilitzats pels desenvolupadors des de les seves aplicacions web per protegir l'accés a les

mateixes. En principi, s'ha pensat a dissenyar una classe principal que implementi la funcionalitat del PoA. Així doncs, els desenvolupadors podran crear instàncies d'aquesta classe, la qual haurà d'oferir alguns mètodes públics com els següents:

- **checkAccess**: que permetrà comprovar si un usuari té accés a l'aplicació web protegida.

- **getUserData**: que permetrà recuperar les dades de l'asserció de l'usuari.

- La biblioteca anirà acompanyada d'un fitxer de configuració que tindrà el format propi dels arxius .INI. El programador podrà definir en aquest fitxer els paràmetres de configuració globals i particulars per a tots els recursos que es protegeixin amb aquesta biblioteca. Així doncs, el fitxer de configuració tindrà un aspecte similar al següent:

```
[DEFAULT]
Lcook_Timeout       = 60
URL_Timeout         = 200
GPoA_URL            = http://poaURL/GPoA
Pubkeys_Path        = /usr/local/apache2/papi/EasyPyPoA/keys
Attribute_Separator = ","
Value_Separator     = "="

[ServiceID_1]
Location = /path/to/service_ID_1/
```

  Com es pot observar, la idea és mantenir una secció per definir tots els paràmetres comuns per a tots els recursos protegits. Després, existirà una secció per cadascun dels recursos protegits amb els paràmetres de configuració específics per a cadascun d'ells.

- Quan es produeixi un error durant l'execució d'algun mètode, es llançarà una excepció perquè pugui ser capturada des de l'aplicació web que fa ús de la llibreria. El programador de l'aplicació web serà l'encarregat de mostrar a l'usuari el missatge d'error que consideri oportú.

Finalment, simplement cal indicar que per a aquest projecte concret no té sentit especificar com serà la interfície per a cadascun dels casos d'ús identificats, ja que les característiques de la mateixa han estat completament definides en els principis generals de la interfície d'usuari.

4.4 Especificació del pla de proves

Per acabar la fase d'anàlisi, es realitzarà l'especificació del pla de proves, que ens servirà per a establir si el sistema compleix els requisits establerts pels usuaris.

Es realitzaran proves de la biblioteca a diversos nivells:

- **Proves unitàries**, a fi de testar separadament cadascun dels mètodes que formaran la llibreria. Està previst utilitzar algun framework de proves unitàries com *unittest* o *doctest* que permeta definir proves per a cada mètode o funció implementada. Per exemple, es crearan proves unitàries per a comprovar el funcionament del mètode que ha de processar el missatge *CHECKED* rebut des d'un GPoA. Aquestes proves les realitzarà el programador de la llibreria, i en cada cas concret verificarà que s'estan obtenint els resultats esperats (que variaran en funció del mètode).

- **Proves d'integració**, a fi de testar el funcionament dels components actuant de forma coordinada. Una vegada la biblioteca estigui implementada, s'integrarà en un sistema SSO de prova basat en PAPI per comprovar el seu correcte funcionament. En aquest cas, les proves les realitzaran el programador de la biblioteca junt amb el consultor extern. L'empresa PRiSE s'ha ofert a proporcionar un entorn SSO on realitzar les proves. Aquesta prova permetrà comprovar si el PoA desenvolupat es capaç de comunicar-se correctament amb el GPoA i d'integrar-se amb la resta de componets.

- **Proves d'implantació**, a fi de testar el funcionament del sistema en el seu entorn d'operació. En aquest cas s'inclouran aplicacions web protegides amb la llibreria implementada dins d'un entorn SSO real i basat en PAPI. Aquestes proves també les realitzarà el programador de la llibreria junt amb el consultor extern. Per aquest fi, s'hauran d'implementar les aplicacions web Python que han de ser protegides per la biblioteca. També serà necessari, com en el cas anterior, disposar d'un entorn SSO basat en PAPI. Com a producte d'aquesta prova s'espera que es pugui accedir de forma correcta als recursos protegits per la llibreria implementada.

- **Proves d'acceptació**, a fi que els usuaris del sistema en validin el funcionament correcte. Per exemple, un usuari pot intentar accedir a diversos recursos protegits per la nostra biblioteca dins d'una arquitectura PAPI. En aquest cas, la prova l'ha de realitzar l'usuari final. Com a producte d'aquesta prova, l'usuari ha de poder accedir als recursos protegits de forma correcta.

Capítol 5 – Disseny del sistema

L'objectiu de la fase de disseny és obtenir els models i especificacions que defineixen el projecte a partir de l'anàlisi realitzada en la fase anterior. Les activitats dutes a terme en aquesta fase ens permetran determinar les especificacions de desenvolupament i integració, i definir l'entorn de proves i implantació necessaris per al seu correcte funcionament.

5.1 Arquitectura

La definició de l'arquitectura del sistema és el primer pas per a identificar- ne els components i dóna lloc a les fases de disseny següents, en les quals aprofundirem. L'objectiu és disposar d'un conjunt de documents i diagrames complets i concisos que siguin comprensibles per a la direcció i alhora serveixin de base per a aprofundir en el disseny del sistema.

A continuació, i abans de detallar més el disseny del sistema, caldrà definir les normes i els estàndards de disseny i construcció.

5.1.1 Definició de nivells d'arquitectura

Hi ha diverses maneres de veure o entendre l'arquitectura d'un sistema:

- **Arquitectura conceptual**: té el propòsit de dirigir l'atenció sobre els grans blocs que formen el sistema, sense entrar en detalls, i identificar les relacions entre aquests blocs. És molt útil per a comunicar a la direcció o als departaments no tècnics una visió global del sistema.

- **Arquitectura lògica**: afegeix detalls a l'anterior i incorpora la definició de les interfícies de comunicacions entre els components, la qual cosa permet als desenvolupadors de cada component treballar sense dependre l'un de l'altre.

Com a suport als diagrames, es poden utilitzar targetes **CRC** (*class responsibility collaborator*), que tenen les característiques següents:

- A la part superior figura el nom del component.

- A la columna esquerra s'ha de reflectir tot el que el component sap o fa sobre ell mateix. S'hi inclourà tot allò que creiem que és responsabilitat seva i la informació que haurà de mantenir.

- A la part dreta figuraran els components amb què es relaciona per poder dur a terme les responsabilitats de la part esquerra.

Definició de l'arquitectura

En primer lloc, per a expressar l'**arquitectura conceptual** del projecte que hem de desenvolupar, usem la notació UML per a crear un diagrama de components (Figura 16).

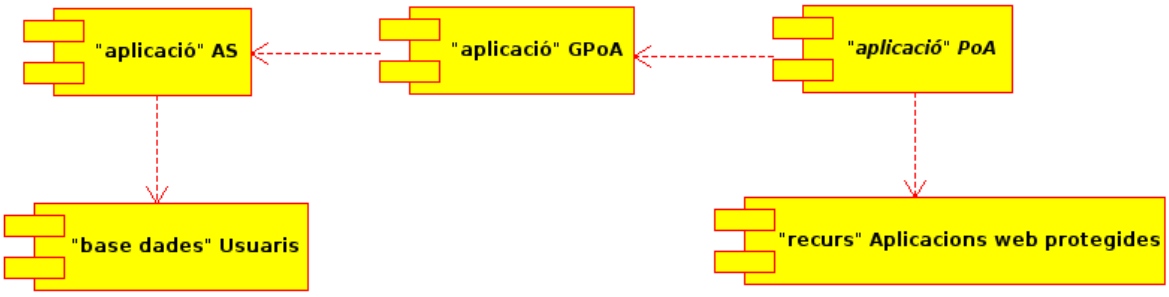


Figura 16: Diagrama UML de components del sistema PAPI

En el diagrama podem veure els components d'un sistema PAPI i els connectors que els uneixen. Aquests connectors indiquen que es produeix alguna mena de comunicació entre ells. Els diferents components en aquest diagrama s'identifiquen mitjançant els estereotips que els acompanyen. És important destacar que la llibreria que hem de desenvolupar sols implementa la funcionalitat d'un PoA, però necessita comunicar-se amb la resta de components d'un sistema PAPI per a complir amb la seva finalitat. En concret, s'ha de comunicar amb l'aplicació web protegida i amb el GPoA.

Una vegada consensuada aquesta visió general del sistema, aprofundim en les interfícies dels components per obtenir l'**arquitectura lògica** del sistema. Per a això, estenem el diagrama de components anterior detallant els processos de comunicació (Figura 17).

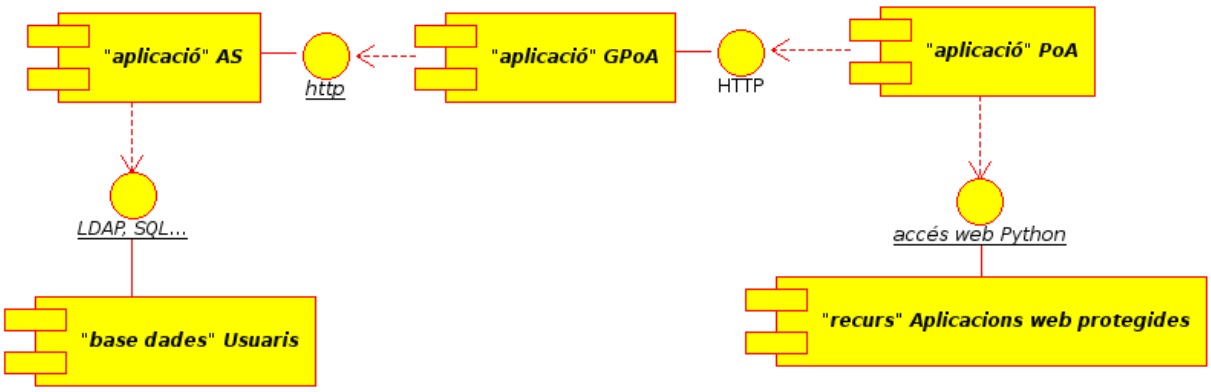


Figura 17: Diagrama UML de components amb interfícies del sistema PAPI

Com a suport a la generació del diagrama anterior, utilitzem targetes CRC.

PoA	
• Controla l'accés als recursos protegits. • Es comunica amb el GPoA per a obtener les assercions dels usuaris que volen accedir als recursos protegits. • Processa les respostes rebudes des del GPoA.	• Aplicacions web protegides • GPoA

GPoA	
• S'encarrega de donar accés a un grup de PoAs amb característiques comunes de validació. • Permet centralitzar les polítiques d'autorització d'una organització en un sol punt, al qual preguntaran els PoA. • S'encarrega d'enviar peticions d'assercions dels usuaris a l'AS.	• AS • PoA

AS	
• S'encarrega d'autenticar als usuaris en un únic punt de la xarxa. • Recol·lecta informació sobre els usuaris i la proporciona als GPoA. • Emet les assercions amb la informació necessaria per verificar la identitat dels usuaris. • Per a dur a terme l'autenticació dels usuaris pot consultar un fitxer de text, una bases de dates, un sistema LDAP...	• GPoA • Base de dades d'usuaris.

5.1.2 Especificació d'estàndards, normes de disseny i construcció

És important acordar amb les persones encarregades del disseny del sistema, i amb els equips que el construiran, unes normes que caldrà seguir en la notació de diagrames i documents. Aquestes normes poden derivar d'estàndards o recomanacions, o bé poden ser de creació i ús interns.

Definició del conjunt de normes i notacions

És convenient que tots els documents creats comparteixin unes característiques i mantinguin un format coherent. Per a això, després d'estudiar els estàndards i les recomanacions sobre el tema, s'arriba a les conclusions següents:

- **Documents de disseny**: aquests documents els han de poder consultar tant el personal tècnic implicat, com el no tècnic. S'acorda que es treballin en format OpenDocument i que la versió més recent estigui simultàniament en PDF per a la seva consulta. En aquest projecte concret, els documents de disseny s'inclouran en aquesta memòria.

- **Diagrames de disseny**: per als diagrames de disseny s'acorda usar la notació UML (http://www.uml.org/) en la seva versió 2.3, definida per l'Object Management Group (www.omg.org). Per a crear el diagrames s'utilitzarà la ferramente Umbrello UML Modeller.

- **Documentació tècnica**: la documentació tècnica serà possiblement la que més revisions patirà i contindrà també enllaços a documentacions de les eines usades, especificacions de programació (API), etc., per la qual cosa utilitzarem un format tan flexible com sigui possible i integrable amb les eines de desenvolupament que s'usin. Per a això, es decideix utilitzar DocBook (www.docbook.org), que ens permetrà: partir un document en diversos fitxers estructurats, incloure referències a altres documents, generar de forma senzilla diversos formats per a la seva visualització (PDF, HTML), possibilitat de separar el contingut i l'estructura del document del seu format, independència de l'editor utilitzat, etc.

Es pot destacar que en prendre les decisions s'ha donat importància a la implantació del format o notació en la indústria i a l'accessibilitat que té; és a dir, a la disponibilitat d'exemples i documentació, i també a un ampli conjunt d'eines que hi treballin amb ells.

5.1.3 Identificació de subsistemes

Aquest apartat està destinat a dividir el sistema que s'ha d'implantar en diversos subsistemes. No obstant això, per a aquest projecte en concret, i a causa de les seues peculiars característiques, no sembla convenient dividir els diferents sistemes identificats en diversos subsistemes. Aquesta decisió es basa en el fet que cada sistema ja té unes funcions molt concretes que ha de realitzar i que no són excessivament complexes o abundants. A més, cal tenir en compte que el projecte consisteix a desenvolupar només un dels components en què es divideix una infraestructura PAPI, en concret el proveïdor de servei o PoA. Així doncs, s'ha decidit prendre com a definitiva l'arquitectura que s'ha mostrat en els diagrames de l'apartat 5.1.1.

5.2 Revisió de casos d'ús

En aquest apartat es revisaran els casos d'ús definits en la fase d'anàlisi i es determinaran les operacions que hauran d'implementar les interfícies de cada un. Com el sistema que estem dissenyant està centrat en el desenvolupament, l'estudi incorpora la definició de les classes i, per tant, els diagrames representaran també la interacció.

Així doncs, durant aquesta fase establirem les característiques de tot el sistema, en revisarem els requisits i en dissenyarem les classes (amb els seus atributs, operacions i relacions). De manera natural durant el procés, també obtindrem el disseny de les proves que asseguraran el bon funcionament del sistema durant el desenvolupament i les seves condicions d'implantació.

5.2.1. Revisió dels subsistemes segons els casos d'ús

Com ja s'ha comentat anteriorment, la llibreria que s'ha de desenvolupar solament ha d'implementar la funcionalitat pròpia d'un PoA lleuger dins d'un sistema PAPI. Així doncs, tots els casos d'ús que es van identificar en la fase d'anàlisi estan relacionats amb funcions que ha de dur a terme aquest component. Per tant, el nostre projecte ha d'implementar el sistema identificat com a "**aplicació PoA**". A continuació es revisen els casos d'ús que es van definir.

- **Cas d'ús "Protecció i accés a un recurs protegit".** Els usuaris de l'entitat que protegeixi les seves aplicacions web fent ús del protocol PAPI, intentaran accedir a aquestes aplicacions des de fora de la institució i a través d'Internet. En aquest moment, el PoA haurà de protegir l'accés a aquests recursos. Si l'usuari no s'ha autenticat prèviament, el PoA es posarà en contacte amb el GPoA per obtenir les assercions de l'usuari i verificar la seva identitat. En cas que tot sigui correcte, el PoA donarà permís a l'usuari perquè accedeixi al recurs protegit. Aquest cas d'ús està relacionat amb els sistemes següents:
    - Sistema "aplicació PoA". És el sistema principal que ha de realitzar l'autorització o control d'accés als recursos protegits. Si l'usuari presenta les credencials adequades podrà accedir al recurs protegit. En cas contrari, el PoA li denegarà l'accés.
    - Sistema "aplicació GPoA". En cas que l'usuari no s'hagi autenticat prèviament en el sistema PAPI, el PoA haurà de posar-se en contacte amb el GPoA per obtenir les credencials que permetin a l'usuari accedir al recurs protegit.
- **Cas d'ús "Procés de *logout* en PAPI".** En aquest cas d'ús es poden diferenciar dues situacions:

- Quan un usuari que prèviament ha iniciat una sessió en el sistema PAPI desitja tancar-la, ha d'indicar-ho al PoA, per exemple, a través de l'accés a una aplicació web que enviï l'ordre corresponent al PoA. Aquesta ordre iniciarà la fase de *LogOut* en PAPI, per la qual cosa un missatge serà enviat entre el PoA i el GPoA. L'objectiu és que el PoA pugui indicar a aquell component que li va proveir d'informació de l'usuari que vol iniciar el procés de *LogOut*. Aquest cas d'ús està relacionat amb els sistemes següents:

  - Sistema “aplicació PoA”. En aquest cas, el PoA haurà de tancar la sessió de l'usuari i enviar un missatge de *LogOut* al GPoA perquè aquest propagui la petició de *LogOut* per tot el *Single Sign-On*.

  - Sistema “aplicació GPoA”. En aquest cas, el GPoA rebrà la petició de *LogOut* des d'un PoA i haurà de propagar-la per la resta del sistema PAPI.

- Quan un usuari tanca una sessió que tenia establerta en un PoA, aquesta petició de tancament de sessió ha de ser propagada a la resta de PoAs que componen el sistema PAPI. Així doncs, el GPoA ha d'encarregar-se d'enviar un missatge de tancament de sessió a tots els PoAs. Aquest missatge s'utilitza per indicar als components que se'ls hagi enviat informació d'usuari que han de tancar la sessió de l'usuari. Aquest cas d'ús està relacionat amb els sistemes:

  - Sistema “aplicació PoA”. En aquest cas, el PoA rebrà el missatge de *LogOut* per part del GPoA, i haurà de tancar la sessió de l'usuari.

  - Sistema “aplicació GPoA”. En aquest cas, el GPoA haurà d'enviar el missatge de *LogOut* a tots els PoAs del sistema PAPI que hagin rebut informació de l'usuari que està tancant la sessió.

Després de l'estudi detallat dels casos d'ús i dels sistemes, s'ha confeccionat el diagrama de classes del sistema PoA que s'ha de desenvolupar. Es pot veure aquest diagrama en la Figura 18.

Com es pot observar en el diagrama de classes, existeix una classe principal denominada **EasyPyPoA** que és la que implementa el component PoA de l'arquitectura PAPI. Aquesta classe ofereix una sèrie de mètodes públics que permetran a les aplicacions web implementades en Python interactuar amb el PoA. Vegem quins són:

- **check_access**: permetrà controlar l'accés a una aplicació web.

- **sign_off_requet**: permetrà tancar una sessió iniciada en un sistema PAPI.

- **get_user_data**: permetrà recuperar les dades de l'asserció d'un usuari.

- **get_issuer**: permetrà obtenir l'identificador del proveïdor d'identitat que va generar l'asserció amb les dades de l'usuari.

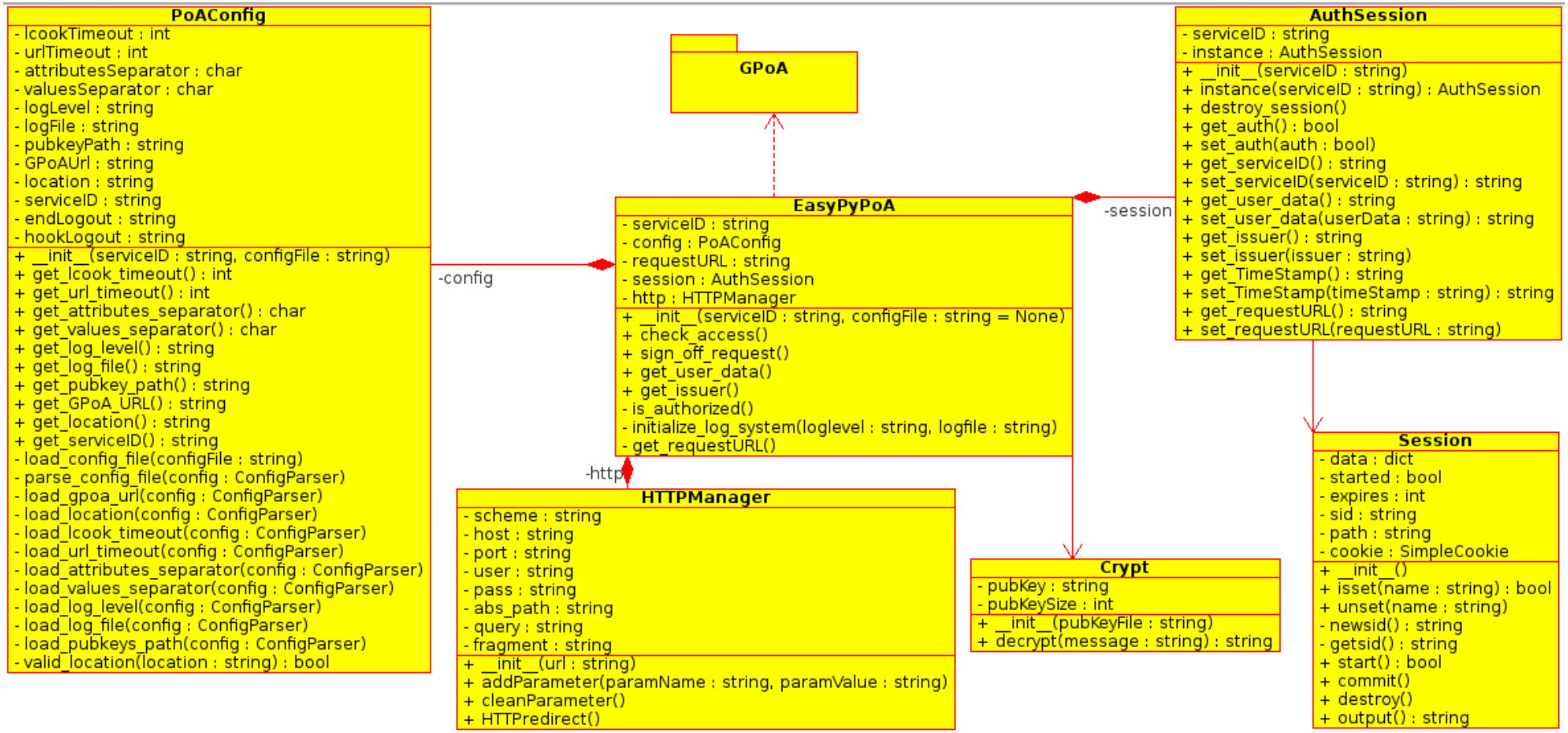


Figura 18: Diagrama de classes UML del sistema PoA

A més, com es pot veure en el diagrama, aquesta classe interactua amb el sistema que implementa al GPoA, i que serà el que proporcionarà al PoA les assercions amb les dades de cada usuari.

Al voltant de la classe EasyPyPoA s'han definit una sèrie de classes que permeten estructurar de forma clara el sistema PoA i que proporcionen diverses funcionalitats a la classe principal. A continuació es descriu breument la finalitat de cadascuna d'elles:

- **PoAConfig**: Aquesta classe s'encarrega de carregar i emmagatzemar la configuració del PoA a partir d'un fitxer de configuració.
- **HTTPManager**: Aquesta classe s'encarrega de realitzar les redireccions HTTP 302 utilitzades pel protocol PAPI mitjançant la injecció de la capçalera *Location* d'HTTP.
- **Crypt**: Proporciona suport criptogràfic al PoA.
- **AuthSession**: Realitza la gestió i encapsulació d'una sessió.
- **Session**: Permet crear i destruir sessions en Python.

A més, s'ha creat una classe per a cadascun dels diferents tipus de missatges que pot generar i/o rebre un PoA: *CHECK*, *CHECKED*, *PAPISIGNOFFREQ*, *PAPILOGGEDOUT* i *PAPILOGOUT*. Aquestes classes s'encarreguen de realitzar les accions que s'han de dur a terme quan el PoA genera o rep els diferents tipus de missatges. Això permet reduir la complexitat de la classe principal del PoA.

Adicionalment, durant el desenvolupament de les classes s'inclouran comentaris detallats en cadascun dels atributs i mètodes perquè quedi clarament especificada la seva funcionalitat.

5.2.2. Elecció d'alternatives de components i llicències més adequades

Una vegada realitzada la revisió dels casos d'ús del sistema PAPI, hem identificat que tot el sistema complet s'haurà de desenvolupar a mida. No obstant això, per a proporcionar suport criptogràfic al sistema s'utilitzarà una llibrería anomenada **M2Crypto**[2].

A més a més, l'estudi de les diferents alternatives existents en el mercat, juntament amb els casos d'ús que es volen satisfer, han donat com a resultat la taula següent, que resumeix els principals components que s'han d'utilitzar en la fase de desenvolupament.

Component	Paquet	Versió prevista	Llicència
Sistema operatiu	GNU/Linux	2.6	GPL
Llenguatge de programació	Python	2.6	Python Software Foundation License

2 https://pypi.python.org/pypi/M2Crypto

Component	Paquet	Versió prevista	Llicència
Servidor web	Apache	2.2	Apache Software License
Llibreria criptogràfica	M2Crypto	0.20	BSD-style license
IDE	Eclipse + Pydev	3.5/2.7	Eclipse Public License

D'altra banda, pel que fa a l'elecció de la llicència de desenvolupament, s'ha de tenir en compte que el projecte es desenvolupa com a part d'una assignatura del Master Universitari en Programari Lliure. Per tant, es indispensable que el projecte i la seva documentació es publiquen sota algun tipus de llicència lliure, ja que és un requisit imposat per la universitat. Per al programari s'ha decidit que s'utilitzarà la ***Apache License***. Es tracta d'una llicència estil BSD que ens permet mantenir el copyright sobre el nostre desenvolupament, i és coherent amb les llicències de la resta de components (M2Crypto). Permet que el destinatari del programari l'utilitzi, el copie, el modifique, el redistribueixi o el vengui. També ens permet la seva incorporació futura a un producte comercialitzable sota una llicència propietària. Per a la documentació s'utilitzarà una llicència ***Creative Commons BY SA***.

5.2.3. Especificacions de desenvolupament i proves

En aquest apartat s'estableixen les condicions i característiques de l'entorn de desenvolupament i es defineixen les proves necessàries que s'hauran de realitzar per assegurar el funcionament del sistema una vegada implantat. Aquestes proves s'intentaran definir com a proves unitàries, és a dir, proves amb el mínim nivell possible de dependència entre elles perquè permetin un desenvolupament del programari per components, deixant per a la fase final les proves d'integració.

Com ja s'ha comentat, l'objectiu principal d'aquest projecte és desenvolupar una llibreria en Python que implementi un PoA lleuger. D'aquesta forma, els programadors que utilitzin aquest llenguatge per al desenvolupament de les seves aplicacions web podran integrar aquestes aplicacions en un entorn PAPI. Així doncs, l'elecció del llenguatge de programació a utilitzar per al desenvolupament de la llibreria ha vingut establerta des de la definició inicial del projecte. En concret s'utilitzarà la versió 2 de **Python** per ser la més estesa actualment, encara que es deixa oberta la possibilitat d'adaptar en un futur la llibreria perquè també sigui compatible amb Python 3. Python és un llenguatge de script molt flexible i versàtil, té moltes llibreries de suport i una comunitat d'usuaris molt extensa. A més permet orientació a objectes.

A l'hora de determinar l'entorn de desenvolupament a utilitzar s'han considerat diverses alternatives com Emacs, Vim, Jedit, Gedit, etc. No obstant això, per la seva facilitat d'ús i la seva potència, juntament amb la possibilitat d'incorporació de moltes extensions ja existents, s'ha decidit utilitzar l'entorn de desenvolupament integrat **Eclipse** juntament amb l'extensió **PyDev**. Es tracta d'un entorn multiplataforma i molt potent que proporciona

funcions avançades de detecció de sintaxi, detecció d'errors, navegació avançada pel codi, autocompletat de codi, integració amb sistemes de control de versions, etc.

La resta d'especificacions de desenvolupament provenen de les decisions preses anteriorment, ja que el format de documentació, i també el marc de treball de les proves unitàries, sorgeixen de manera natural a partir del llenguatge de programació escollit.

- Marc de treball de proves unitàries: mòdul **unittest**[3].
- Documentació del desenvolupament: format **epydoc**[4]
- Documentació tècnica de les interfícies i el seu ús: **DocBook**.

Finalment, a continuació s'enumeren les principals proves unitàries, extretes de les funcionalitats i interfícies del sistema, que es realitzaran en cada una de les classes definides:

- **PoAConfig**:
  - Es proporcionarà com entrada fitxers amb una estructura invàlida (sense que siguin fitxers amb format INI) per comprovar que són detectats correctament.
  - Es proporcionaran fitxers amb seccions obligatòries no presents per comprovar que són detectats com a invàlids.
  - Es proporcionaran fitxers amb camps obligatoris no presents per comprovar que són detectats com a invàlids.
  - Es proporcionarà com entrada noms de fitxers de configuració no existents o sense permisos de lectura per comprovar que l'error és detectat.
  - Es proporcionaran com entrada fitxers de configuració on els diferents camps tinguin valors o formats invàlids. Per exemple, caràcters on s'esperin sencers i al revés, valors invàlids per a la directiva *Location*, paths no existents per a les directives de la clau pública del GPoA i del fitxer de log, etc.
  - També es proporcionaran fitxers de configuració vàlids per comprovar que són acceptats sense problema.

- **Crypt**:
  - Es proporcionaran com a entrada paths de fitxers de clau pública inexistents i sense permisos de lectura per comprovar que l'error és detectat.
  - Es proporcionaran com a entrada paths de fitxers de clau pública amb un format invàlid per comprovar que l'error és detectat.
  - Es proporcionaran com entrada textos mal xifrats per comprovar que l'error és detectat. Per exemple, es proporcionaran textos xifrats amb una clau privada que no es correspongui amb la clau pública del GPoA, es proporcionaran textos sense xifrar, es proporcionar textos ben xifrats però sense codificar en base64, etc.
  - També es proporcionaran com entrada textos ben xifrats, per comprovar que el textos obtinguts es corresponen amb els originals.

3 http://docs.python.org/2/library/unittest.html
4 http://epydoc.sourceforge.net/

- **HTTPManager**:
  - Es proporcionarà com a entrada diverses URLs mal formades per comprovar que l'error és detectat.
  - Es proporcionaran URL correctes i es provarà a realitzar diverses redireccions HTTP 302 per comprovar que s'executen de forma correcta.

- **Session**:
  - Es comprovarà que quan s'accedeix per primera vegada a un servidor web, aquest genera la cookie de sessió necessària i l'envia al navegador a través de la capçalera HTTP *Set-cookie*.
  - Es comprovarà que quan s'accedeix posteriors vegades, el navegador reenvia la cookie al servidor a través de la capçalera *Cookie*, i el servidor és capaç de carregar les dades que tenia emmagatzemats en la sessió.
  - Es provarà a crear i destruir les sessions, per comprovar que els processos es completen de forma correcta.
  - Es provarà a afegir i llevar dades de la sessió per comprovar que són emmagatzemades i destruides de forma correcta.

- **AuthSession**:
  - Es provarà a iniciar sessions a través d'aquesta classe.
  - Es comprovarà que és possible modificar i consultar els valors que s'han d'emmagatzemar en una sessió PAPI.

- **EasyPyPoA**:
  - Es comprovarà que és possible accedir a un recurs protegit després de passar pel procés d'autenticació en el sistema PAPI.
  - Es comprovarà que una vegada iniciada una sessió en el sistema, ja no és necessari tornar a autenticar-se en el sistema PAPI.
  - Es comprovarà que és possible iniciar el procés de tancament de sessió i que una vegada tancada la sessió ja no és possible accedir als recursos protegits sense passar pel procés d'autenticació.
  - S'intentarà accedir a un recurs protegit després d'autenticar-se en el sistema, però amb una sessió caducada, per comprovar que no és possible.

5.2.4. Requisits d'implantació

Els requisits d'implantació són els que ha de complir el sistema quan treballi en l'entorn real conjuntament amb la resta de sistemes (GPoA , AS i aplicacions web Python).

Com ja hem comentat anteriorment, el projecte a desenvolupar no està destinat a ser utilitzat en una sola empresa, sinó que es tracta d'una llibreria genèrica la finalitat de la qual és que qualsevol entitat que desitgi implantar un Sistema de Login Únic fent ús del protocol PAPI, pugui fer-ho utilizant aplicacions web implementades en Python. A més a més, la llibrería no està pensada per a que els usuaris finals del sistema interactuen directament amb ella. La llibrería la utilitzaran els programadors per protegir l'accés a les

aplicacions web que hagin d'estar integrades en una arquitectura PAPI. En general, aquests usuaris estaran familiaritzats amb la programació d'aplicacions web y coneixeran el llenguatge Python, per la qual cosa no haurien de tenir cap tipus de problema a utilitzar de forma adequada la llibreria per protegir les seves aplicacions web. Així doncs, es descarta la possibilitat de desenvolupar un pla de formació. No obstant això, la llibreria anirà acompanyada de la documentació tècnica adequada per a que els programadors la puguin consultar quan sigui necessari.

Pel que fa a la documentació del projecte, esta es publicarà amb una llicència lliure associada, per la qual cosa qualsevol persona tindrà accés a la mateixa i podrà consultar-la, copiar-la i modificar-la.

Des del punt de vista tecnològic, el PoA desenvolupat podrà executar-se en qualsevol màquina que tingui instal·lat el següent programari:

- El llenguatge de programació Python en la seva versió 2 (>=2.6)
- La llibreria criptogràfica M2Crypto.
- Un servidor web amb suport per interpretar aplicacions web Python com Apache 2.

Els requisits de maquinari seran mínims, i vindran determinats pel servidor web que es decideixi utilitzar, i per la versió de Python que s'instal·li.

Finalment, cal recordar que per al correcte funcionament del sistema, el PoA haurà de poder establir una comunicació a través d'HTTP amb un GPoA. El GPoA normalment estarà instal·lat en un altre servidor. Això haurà de ser tingut en compte, per exemple, a l'hora de configurar els tallafocs dels servidors.

Capítol 6 – Desenvolupament, implantació i manteniment del sistema

6.1 Fase de desenvolupament

L'objectiu de la fase de desenvolupament és la construcció ordenada del sistema del qual s'ha avaluat la viabilitat, s'ha analitzat i s'ha dissenyat. En aquest projecte concret s'ha seguit una metodologia tradicional, i l'inici del desenvolupament s'ha produït quan les fases anteriors s'han completat satisfactoriament i en la seva totalitat.

6.1.1 Planificació de les activitats de desenvolupament i integració del sistema

Una vegada arribats a aquest punt, ja tenim informació sobre què s'ha de desenvolupar, quines eines utilitzarem, en quin entorn, etc.

Les activitats que s'han de realitzar per completar la fase de desenvolupament i integració d'aquest projecte són les següents:

- **Concretar versions del programari i llibreries que s'utilitzaran**. En aquest cas s'ha decidit emprar la versió 2.6 de Python per a realitzar el desenvolupament, i la versió 0.20 de la llibreria criptogràfica M2Crypto.

- **Estudiar aquestes llibreries**. S'ha d'estudiat el funcionament i ús de la llibreria M2Crypto, així com els mòduls de la llibreria estàndar de Python que és necessari utilitzar.

- **Implantar l'entorn de desenvolupament**. En aquest cas s'ha d'instalat el llenguatge Python 2, la llibrería M2Crypto i l'OpenSSL. A més s'ha d'instal·lat i configurat l'IDE Eclipse amb el plugin PyDev, i un servidor web amb suport per a Python, que en aquest cas és Apache 2.

- **Desenvolupar les proves unitàries**. Per provar el funcionament de les diverses classes implementades.

- **Desenvolupar els components necessaris**. En aquest cas totes les classes que composen la llibreria.

- **Realitzar la documentació**.

- **Desenvolupar les proves d'integració del sistema**. S'ha de desenvolupar una

petita aplicació web en Python que faci ús de la llibreria implementada i que es pugui desplegar dins d'un SSO que utilitzi el protocol PAPI. Aquest punt és el que marcarà el nivell d'èxit del projecte.

- **Aprovar el sistema**.

Per a expressar la planificació del desenvolupament del projecte utilitzarem un diagrama de Gantt. El diagrama és el que es pot observar a la Figura 19. En el diagrama es poden veure totes les tasques que s'han enumerat, junt amb la seva durada prevista i les seves dependències.

Aquest diagrama permet seguir l'evolució del desenvolupament i ajustar les fites i dates estimades segons com vagi evolucionant.

6.2.2 Desenvolupament

L'objectiu d'aquesta fase és implantar l'entorn de desenvolupament seleccionat en l'equip que l'ha de dur a terme i l'ha d'executar, i generar el codi dels diferents components del sistema. A més s'han d'executar les proves unitàries i d'integració.

Així doncs, segons les decisions preses en les fases anteriors i a partir de la planificació del desenvolupament, en aquesta fase s'ha de realitzar el següent:

- **Instal·lació d'Eclipse i del plugin PyDev**. Eclipse s'instalarà des dels repositoris de la distribució GNU/Linux que s'està utilitzant (Debian 6). Per a la instal·lació i configuració del plugin PyDev en Eclipse es seguiran els passos establerts a la pàgina web oficial[5].

- **Establir un estil de codificació**. Com ja es va comentar, es seguiran totes les convencions de codificació de programes establertes en la "*Style Guide for Python Code*".

- **Generació del codi** a partir dels diagrames de classes, els casos d'ús revisats i les proves unitàries.

- **Execució de les proves unitàries** concurrentment amb el desenvolupament.

6.3.3 Documentació

L'objectiu d'aquesta fase és l'elaboració de la documentació d'usuari. Així doncs, sobre la base de les decisions preses pel que fa a aquesta qüestió en fases anteriors, relatives al seu format i disponibilitat, se n'ha de desenvolupar l'estructura i el contingut. Cal tenir en compte que la documentació, en aquest projecte en concret, estarà destinada a que els programadors tinguen una guia de com instal·lar, configurar i utilitzar la llibreria implementada. Així doncs, l'estil de redacció de la documentació final s'adequarà a aquest tipus de destinatari.

5 http://pydev.org/

Nombre	Trab...
Desenvolupament	**101d**
Concretar versions del programari i llibreries	1d
Estudiar llibreries	5d
Implantar l'entorn de desenvolupament	1d
Desenvolupar les proves unitàries	20d
Desenvolupar els components	50d
Realitzar la documentació	20d
Desenvolupar les proves d'integració	3d
Aprovar el sistema	1d

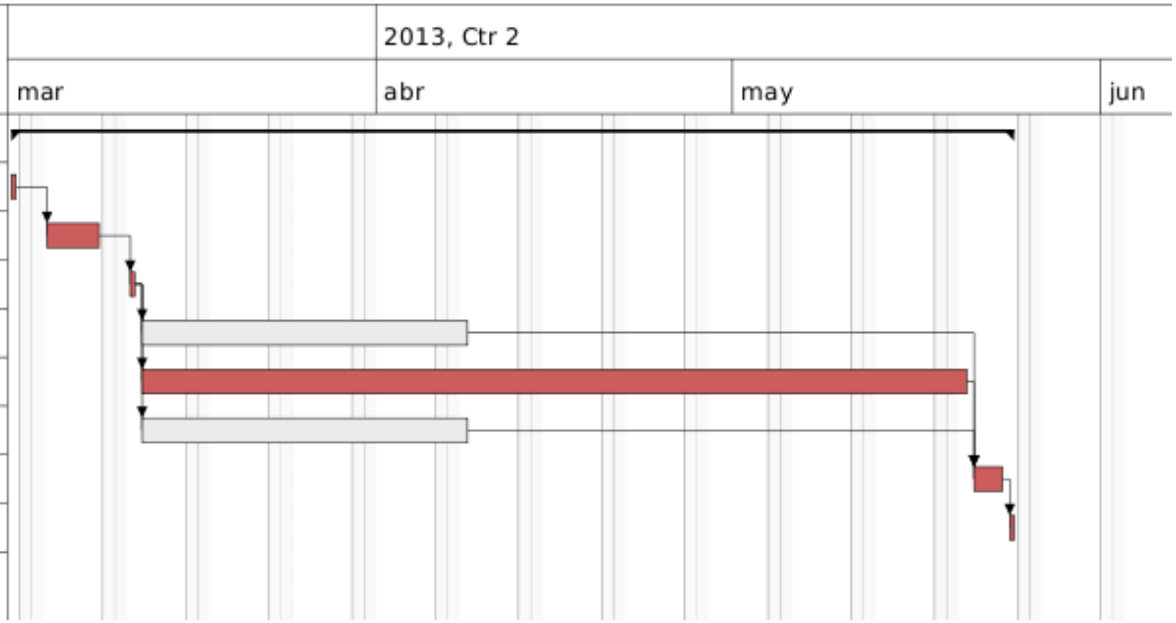


Figura 19: Diagrama de Gantt amb la planificació del desenvolupament

Les decisions preses en fases anteriors indiquen que la documentació d'usuari ha d'estar en format **DocBook**. A més, la documentació técnica continguda en el codi font es trobarà en format **Epydoc**.

6.2 Fase d'implantació

En aquest cas concret cal tenir en compte les peculiars característiques d'aquest projecte, que no està destinat a ser implantat en cap empresa concreta, sinó que és tracta d'una llibreria genèrica que es pot utilitzar en multitud d'entitats. Per tant, els usuaris que es veuran afectats per la seva implantació variaran d'unes institucions a unes altres.

No obstant això, si que s'han de realitzar proves d'integració a fi de testar el funcionament dels diferents components que formen una arquitectura PAPI actuant de forma coordinada (PoA, GPoA i AS). Una vegada la biblioteca estigui implementada, s'integrarà en un sistema SSO de prova basat en PAPI per comprovar el seu correcte funcionament. L'empresa PRiSE s'ha ofert a proporcionar un entorn SSO on realitzar les proves. Aquesta prova permetrà comprovar si el PoA desenvolupat es capaç de comunicar-se correctament amb el GPoA i d'integrar-se amb la resta de componets.

Així doncs, en aquest projecte la implantació consisteix en integrar la llibreria desenvolupada en un entorn SSO i en fer diferents tipus de proves per a comprobar que funciona de forma correcta. Considerarem doncs que aquest procediment es equivalent al pas a la producció del sistema en l'entorn en que operarà finalment.

6.2.1 Formació

A causa de les peculiars característiques d'aquest projecte, no s'elaborarà cap pla de formació als usuaris. Això és així, entre altres coses, perquè la llibreria implementada no interactuarà de forma directa amb l'usuari final, i aquest la utilitzarà de forma transparent.

El que sí es farà és elaborar la documentació tècnica necessària perquè els programadors aprenguin a instal·lar, configurar i utilitzar la llibreria que implementa al PoA. Aquesta documentació es pot trobar a l'annex 1 d'aquesta memòria.

6.2.2 Implantació del sistema i proves

Com s'ha comentat, en aquesta etapa s'han de dur a terme les actuacions necessàries per implantar la llibreria desenvolupada en un entorn SSO proporcionat per l'empresa PRiSE. Com a pas previ, s'haurà de comprovar que els recursos per complir amb els requisits mínims necessaris estan disponibles.

Amb tots els procediments documentats i el sistema provat, la implantació es limatará en aquest cas a realitzar la instal·lació de la llibreria en l'ordinador on es trobin els recursos que s'han de protegir mitjançant el PoA desenvolupat. Aquest ordinador haurà de tenir

comunicació amb el servidor on es trobi instal·lat el GPoA. Així doncs, s'hauran de realitzar les següents tasques:

- Instal·lar el programari necessari en el ordinador on s'implantarà el PoA (servidor web, python, openssl i llibreria M2Crypto)

- Instal·lar la llibrería que implementa el PoA en aquest ordinador.

- Configurar el PoA amb les dades adequades per al sistema SSO on estarà integrat.

Una vegada comprovada que la instal·lació s'ha dut a terme de forma correcta, s'executaran les proves d'implantació:

- Intent d'accés a una aplicació web protegida per part d'un usuari que no ha iniciat ninguna sessió al sistema PAPI.

- Intent d'accés a una aplicació web protegida per part d'un usuari que sí ha iniciat una sessió previament al sistema PAPI.

- Intent d'accés a una aplicació web protegida per part d'un usuari que ha iniciat una sessió previament al sistema PAPI, però aquesta ha expirat.

- Intent d'accés a una segona aplicació web després d'haver accedit previament a un altra aplicació web protegida dins del mateix sistema PAPI.

- Intent de desconnexió després d'haver iniciat una sessió al sistema PAPI.

Una vegada revisats els resultats i contrastats amb els requisits, caldrà decidir si hi ha incidències per resoldre. Posteriorment, i segons la gravetat de les incidències registrades, decidirem si tornem a executar el pla de proves completament o parcialment.

Passos seguits per a realitzar la implantació en el SSO PAPI de PRiSE.

En aquest apartat es descriuen els passos que s'han seguit per a integrar EasyPyPoA en un entorn de prova SSO basat en PAPI. Com ja hem comentat, aquest entorn ha sigut proporcionat per l'empresa PRiSE. A continuació s'enumeren les principals dades que descriuen l'entorn utilitzat per a realitzar la implantació del sistema:

- **Recurs web a protegir**: http://javipenya.no-ip.biz:8080/EasyPyPoA/index.py
- **URL del GPoA**: http://lab.prise.es/adAS/PAPI/gpoa.php

Seguidament es descriuen els passos que s'han seguit per a realitzar la implantació:

- Primer de tot, a la màquina on havia d'estar el recurs web protegit es va instal·lar tot el programari necessari: servidor web Apache, Python2, Openssl i llibreria M2Crypto. L'ordinador executava el sistema operatiu Debian 6, per la qual cosa va

ser posible instal·lar tot el programari des dels repositoris. En concret, es va executar l'ordre:

```
aptitude install apache2 python2.6 openssl m2crypto
```

- Després es va descomprimir la llibreria EasyPyPoA i es va instal·lar en el sistema. Per a això es van executar les ordres:

```
tar xzvf EasyPyPoA-0.1.tar.gz
cd EasyPyPoA-0.1/
python setup.py install
```

- A continuació es va crear un directori destinat a guardar les aplicacions web protegides. En aquest cas es va crear l'estructura de directoris /web/EasyPyPoA/. En aquest directori es van copiar els scripts de prova *index.py* i *exit.py* inclosos en la llibreria EasyPyPoA:

```
mkdir /web/EasyPyPoA/
cp index.py exit.py /web/EasyPyPoA/
```

- Després es va configurar el servidor web Apache perquè en el directori creat al pas anterior es pugueren executar CGI scripts Python. A més, en aquest cas concret el servidor web vam fer que escoltara al port 8080. Per a això, al fitxer de configuració `/etc/apache2/ports.conf` es van agregar les següents directives:

```
NameVirtualHost *:8080
Listen 8080
```

- I per poder executar CGI scripts Python, al fitxer de configuració situat a `/etc/apache2/sites-available/default` es van anyadir les directives següents:

```
<VirtualHost *:8080>
     DocumentRoot /web
     Alias /EasyPyPoA/ /web/EasyPyPoA/
     <Directory "/web/EasyPyPoA">
          Options ExecCGI
          AddHandler cgi-script py
     </Directory>
     DirectoryIndex index.py
</VirtualHost>
```

- El següent pas va ser configurar la llibreria EasyPyPoA. Per a això, es va modificar el fitxer de configuració que durant l'instal·lació de la llibreria s'havia creat al directori `/usr/local/lib/python2.6/dist-packages/EasyPyPoA/conf/PAPIPoA.ini`. El fitxer va quedar de la següent manera:

```
[DEFAULT]
Lcook_Timeout          = 60
URL_Timeout            = 20
Current_Time_Windows   = 20
GPoA_URL               =  http://lab.prise.es/adAS/PAPI/gpoa.php
Attribute_Separator    = ","
Value_Separator        = "|"
LogLevel               = DEBUG
LogFile                = /var/log/apache2/EasyPyPoA/PAPIEasyPyPoA.log
End_Logout             = http://javipenya.no-ip.biz:8080

[TestEasyPoA]
Location = /EasyPyPoA/
```

- Després es va crear el directori destinat a emmagatzemar la clau pública del GPoA. En aquest cas:

```
mkdir /usr/local/python2.6/dist-packages/EasyPoA/conf/keys/
```

- I es va copiar en aquest directori la clau publica del GPoA, amb el nom de fitxer ***_GPoA_pubkey.pem***.

- Amb això, va quedar tot preparat per a comencar les proves. En primer lloc, es va provar a accedir a l'aplicació web protegida sense haver iniciat una sessió al sistema PAPI. Per a això, es va introduir al navegador la URL del recurs protegit [http://javipenya.no-ip.biz:8080/EasyPyPoA/index.py]. En la Figura 20 es poden observar les redireccions que es produeixen en intentar accedir al web recurs protegit. A més, com no s'ha iniciat una sessió prèvia, el GPoA ens mostra un formulari perquè ens autentiquem per primera vegada en el sistema:

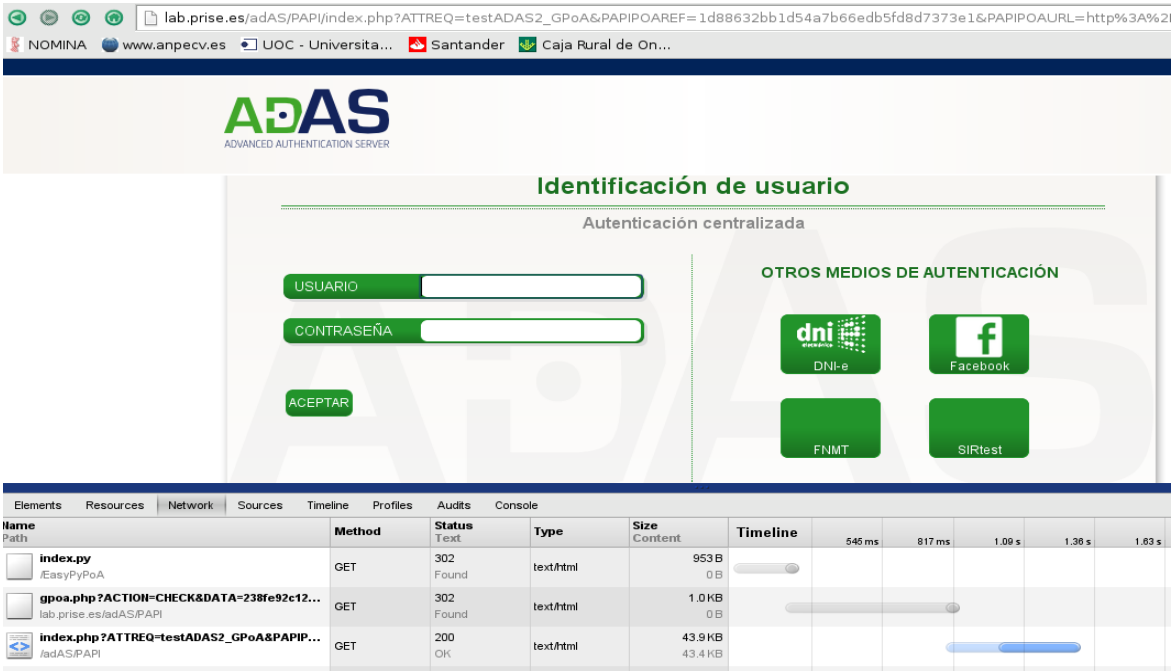


Figura 20: Intent d'accés a un recurs protegit sense tenir una sessió iniciada

- Una vegada introduït l'usuari i la constrasenya, el GPoA fa les comprovacions oportunes i envía una asserció amb les dades del usuari. En la Figura 21 es mostra com el PoA comprova les dades enviades per part del GPoA i dona accés al usuari mostrant les dades rebudes en l'asserció:

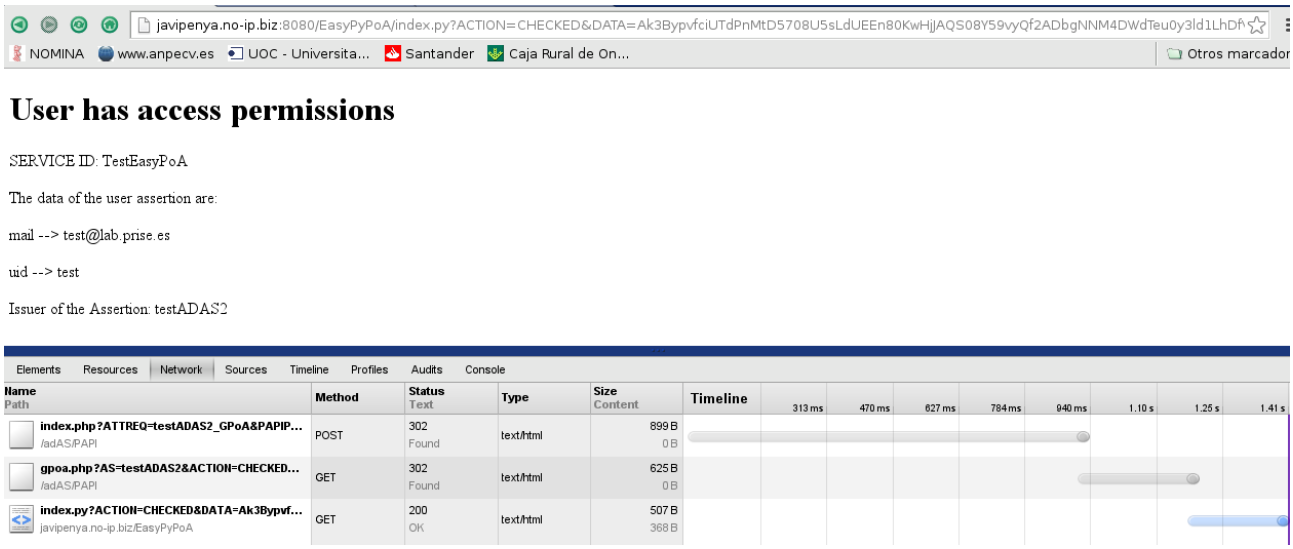


Figura 21: Accés al recurs protegit després d'autenticar-se al IdP/GPoA

- La següent prova va consitir en accedir al recurs web protegit després de dur a terme la prova anterior, es a dir, després d'haver iniciat una sessió al sistema PAPI. En aquest cas, com es pot observar a la Figura 22, el PoA no necessita redireccionar al GPoA, ja que li arriba una cookie amb la sessió previament iniciada, i dóna accés al usuari:

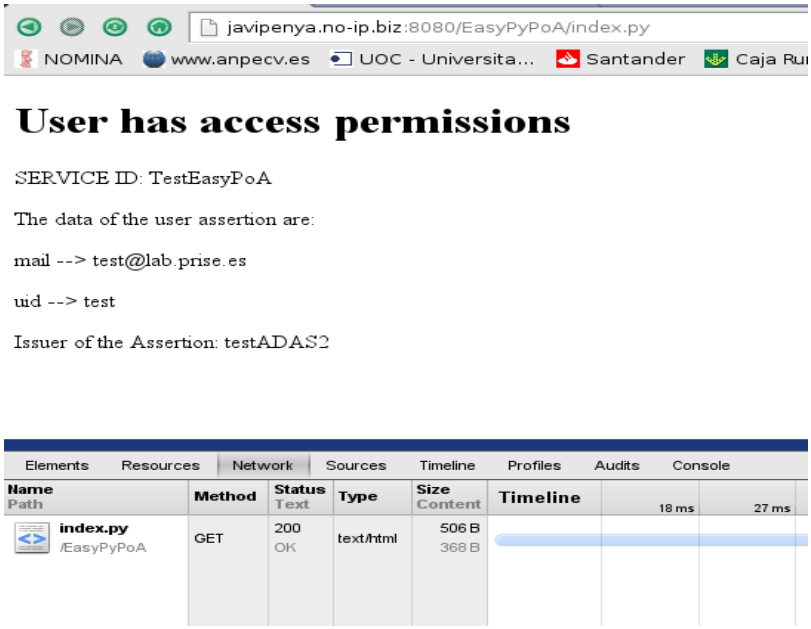


Figura 22: Accés a un recurs protegit després d'iniciar sessió al sistema PAPI

- Seguidament, es va provar a accedir al recurs web protegit després de deixar passar el temps necessari per a que expirara la sessió en el PoA (60 segons). En aquest cas, el PoA torna a enviar el missatge CHECK al GPoA, pero com en el GPoA la sessió encara no ha expirat, aquest torna les dades del usuari directament sense tenir que introduir el nom d'usuari i la contrasenya. Això es pot veure a la Figura 23:

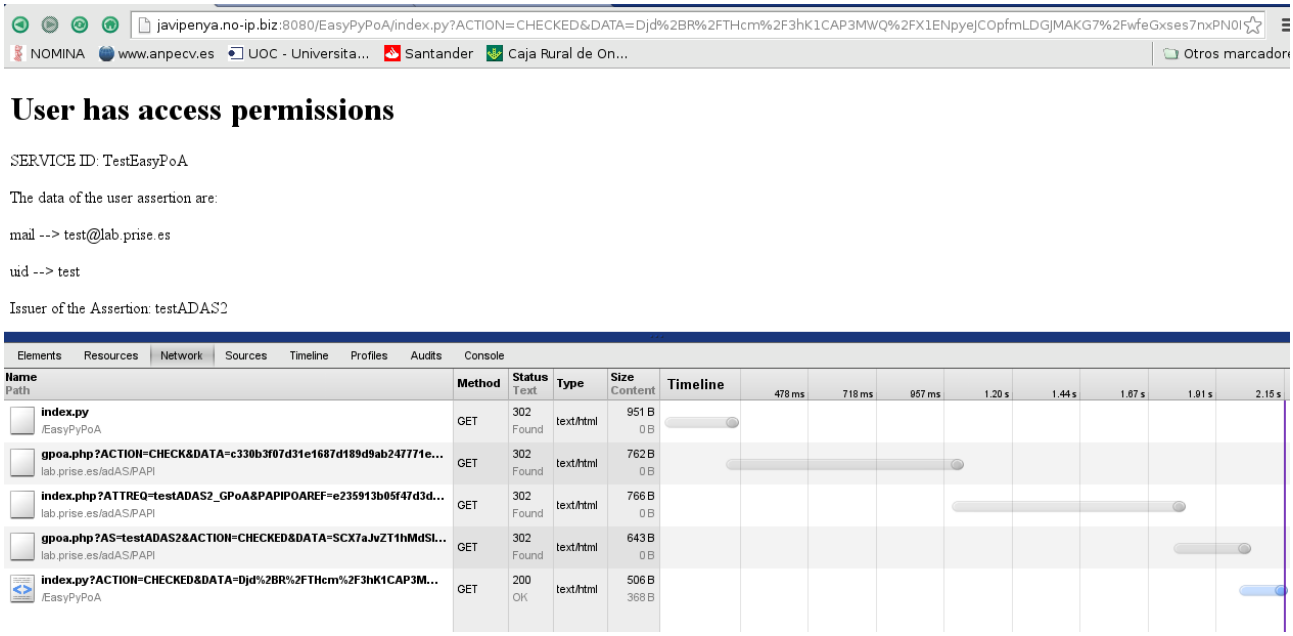


Figura 23: Accés al recurs protegit després de que expire la sessió al PoA

- Finalment, es va realitzar un intent de desconnexió després d'haver iniciat una sessió al sistema PAPI. Per a això, es va accedir a la direcció web [http://javipenya.no-ip.biz:8080/EasyPyPoA/exit.py] que és l'aplicació preparada per iniciar el proceś de Logout. Com es pot observar a la Figura 24, en aquest cas el PoA es posa en contacte amb el GPoA per indicar-li que s'ha iniciat el procés de Logout i el GPoA demana confirmació:

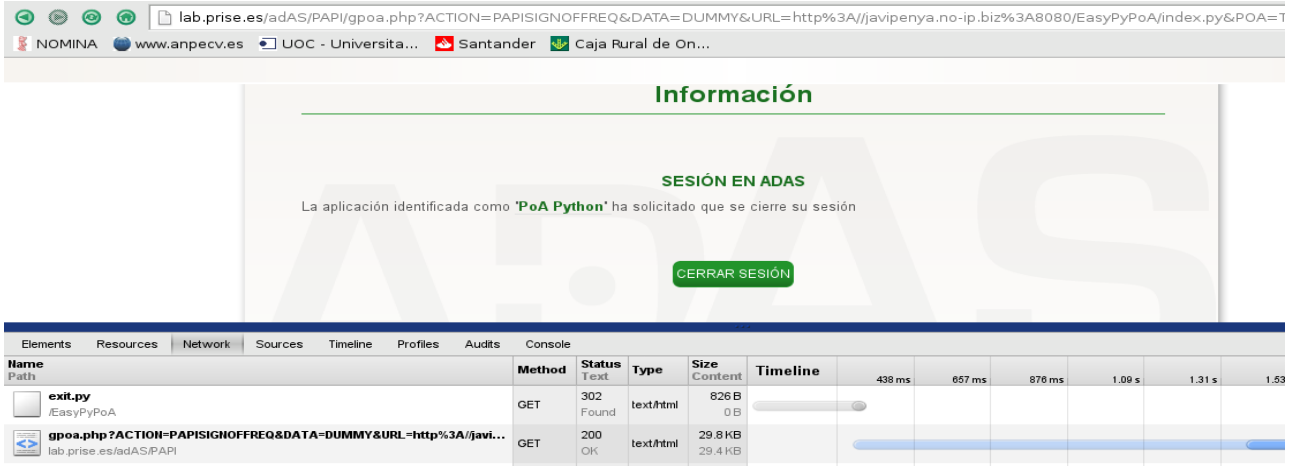


Figura 24: Procés de Logout. Confirmació del tancament de sessió al GPoA

- Una vegada confirmat el tancament de la sessió al GPoA, aquest contesta al PoA i el PoA redirigeix a la URL indicada en la configuració al paràmetre End_Logout, en aquest cas a [javipenya.no-ip.biz:8080]. Això es pot observar a la Figura 25:

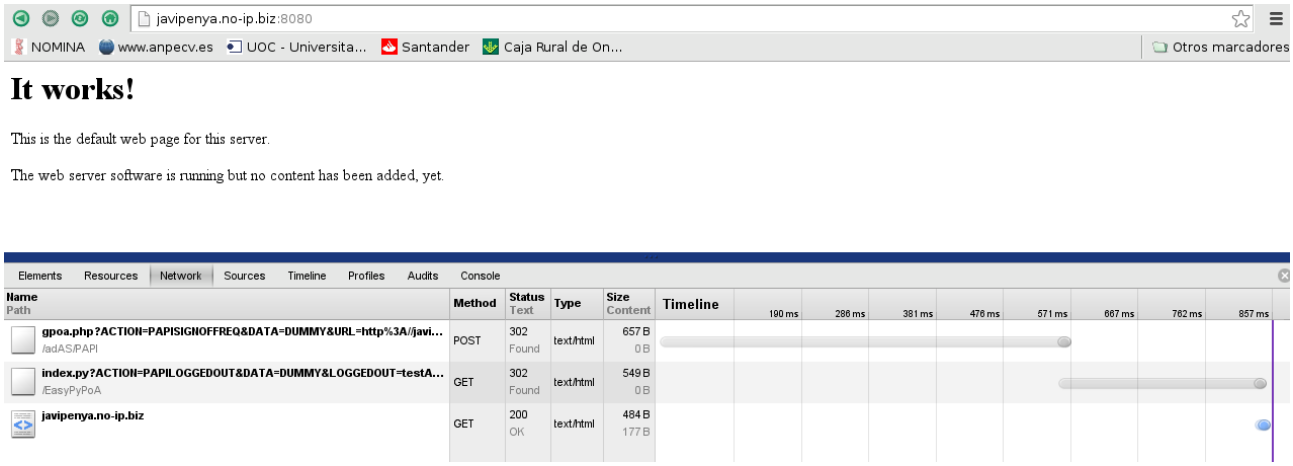


Figura 25: Final del procés de Logout iniciat al PoA

Després de totes les proves dutes a terme, podem afirmar que el procés d'implantació ha estat un èxit. Cal tenir en compte que s'ha utilitzat un GPoA desenvopat en PHP per l'empresa PRiSE i que el PoA és el programari desenvolupat per l'alumnen en Python. Tot i això, tots els components del sistema PAPI han aconseguit comunicar-se sense problemes i dur a terme totes les accions previstes.

6.2.3 Nivell de serveis

Si el resultat obtingut en les proves d'implantació és satisfactori, fixarem un nivell de servei complet en el SSO on el PoA ha estat implantat. D'aquesta manera, el servei ofert pel PoA serà bàsicament la capacitat de controlar l'accés a recursos protegits dins d'un sistema PAPI, i la capacitat d'iniciar una desconnexió del sistema per part de l'usuari.

6.2.4 Acceptació del sistema

Aquesta activitat consistirá a presentar al consultor extern de PRiSE i al tribunal d'avaluació de la UOC tota la documentació relativa a la finalització del proyecte, incloent-hi el codi font de la llibreria, els resultats de les proves, etc. La finalitat d'aquesta activitat es la acceptació definitiva del sistema.

6.3 Fase de manteniment

Com ja hem comentat, el sistema desenvolupat no està destinat a ser implantat en cap empresa concreta, sinó que es tracta d'una llibreria genèrica que es pot utilitzar en multitud d'entitats. Així doncs, el desenvolupador realitzarà un procés de manteniment de la llibreria que consistirà en anar provant-la en diferents entorns i en anar incloent millores o correccions d'errors en les diferents versions que es vagin publicant. De la mateixa manera, si el desenvolupador rep algun missatge indicant bugs en la llibreria, intentarà corregir-los i publicar una nova versió el més aviat possible.

A més, el desenvolupador posarà a disposició de la comunitat tot el codi font i la documentació elaborada, per la qual cosa qualsevol persona amb suficients coneixements tècnics podrà realitzar el manteniment de la llibreria en l'entitat o institució on sigui implantada. Si sorgeix qualsevol tipus de problema amb aquest manteniment, el desenvolupador es mostrarà disposat a ajudar en la mesura del possible.

En qualsevol cas, en tractar-se d'un projecte lliure les entitats sempre tindran la possibilitat de contractar a una tercera empresa que els proporcioni el suport necessari si el desenvolupador original no ofereix aquest servei o el temps de resposta no s'ajusta a les seves necessitats.

Capítol 7 – Conclusions

7.1 Objectius aconseguits

A la finalització d'aquest projecte, podem afirmar que s'han complert tots els objectius que es van marcar a l'inici del mateix ja que:

- D'una banda, l'alumne ha aconseguit un nivell de comprensió elevat sobre com funciona el protocol PAPI, i quin és el format dels missatges que utilitza.

- D'altra banda, l'alumne ha aconseguit dissenyar i desenvolupar una llibreria que implementa el comportament d'un PoA PAPI lleuger utilitzant el llenguatge de programació Python. D'aquesta forma, gràcies a aquesta llibreria els desenvolupadors d'aplicacions web Python podran integrar les mateixes en un sistema Single Sign-On basat en PAPI.

Aquests són els dos objectius principals que es van establir a l'inici del projecte i, després dur a terme el desenvolupament i la implantació de la llibreria, podem afirmar que s'han aconseguit amb èxit.

7.2 Dificultats trobades

En aquest apartat es destaquen les principals dificultats que s'han trobat a l'hora de dur a terme aquest projecte:

- Durant la fase d'anàlisi del projecte, la principal dificultat va ser comprendre el que implicava que la llibreria hagués de ser 'lleugera'. En la documentació del protocol PAPI no s'especificava que parts del protocol ha d'incorporar una implementació del mateix pel fet de ser lleugera o no. Aquest problema va ser resolt gràcies a les respostes proporcionades pel tutor extern, que ja havia realitzat implementacions lleugeres de PAPI en altres llenguatges de programació. En la introducció i en la definició dels requisits s'han establert quins són aquestes característiques peculiars.

- Durant la fase d'implementació van sorgir altres dificultats més costoses de solucionar. D'una banda, en implementar una llibreria lleugera, tant el context de seguretat de PAPI com els paràmetres de la petició original s'havien de mantenir en una sessió d'usuari, i no utilitzant fitxers DB o bases de dades com fan les implementacions pesades del protocol. El problema va sorgir pel fet que Python no ofereix suport per a la gestió de sessions, com sí fan altres llenguatges més orientats al desenvolupament web com PHP. La solució a aquest problema va ser implementar una classe que imités en la mesura del possible la funcionalitat que es proporciona en PHP amb les seves varibles de sessió i les seves funcions '*session*' [10].

- D'altra banda, un altra dificultat va sorgir a causa de la peculiar forma que té el protocol PAPI de xifrar els missatges per garantir l'autenticitat i integritat dels mateixos [4]. En PAPI, els missatges es xifren amb la clau privada de l'emissor i es desxifren amb la seva clau pública (és una forma peculiar de signar els missatges sense fer ús de funcions hash). El problema és que en molts llenguatges de programació no existeixen llibreries criptogràfiques capaces de realitzar aquesta estranya operació de xifrat. La idea inicial era fer ús de la llibreria PyCrypto, però aquesta llibreria no permitia dur a terme aquesta operació. Després de molt buscar, es va donar amb la llibreria M2Crypto que sí que inclou un parell de mètodes permeten dur a terme aquest tipus d'operació de xifrat.

A més dels problemes citats van aparèixer molts uns altres, però la seva solució va ser molt més senzilla i ràpida, i en la majoria dels casos es van resoldre amb l'enviament i recepció d'alguns correus al tutor extern.

7.3 Possibilitats d'ampliació

Com s'ha comentat en l'apartat 7.1, s'han aconseguit tots els objectius que es van marcar a l'inici del projecte, la llibreria és completament funcional i compleix tots els requisits d'una llibreria lleugera que implementa el comportament d'un PoA de PAPI.

No obstant això, com també s'ha comentat ja, la llibreria s'ha desenvolupat utilitzant el llenguatge de programació Python 2. Una possibilitat d'ampliació seria modificar la llibreria perquè també fos compatible amb la nova versió de Python (Python 3) que no és compatible cap a enrere. En els inicis del projecte es va decidir implementar-la en Python 2 perquè actualment segueix sent la versió més estesa d'aquest llenguatge de programació. No obstant això, és probable que amb el temps cada vegada més usuaris comencin a utilitzar Python 3.

Un altra possibilitat d'ampliació d'aquest projecte seria implementar un GPoA lleuger en Python. La majoria de les implementacions existents de PAPI tenen tots dos components implementats, tant el PoA com el GPoA. Així doncs, en un futur es podria implementar en Python el component que falta en aquest cas. No obstant això, és important indicar que el nostre PoA pot interactuar amb PoAs desenvolupats en altres llenguatges de programació.

7.4 Conclusions

Amb la realització d'aquest projecte s'ha aconseguit dissenyar i desenvolupar una llibreria lleugera que implementa el comportament del component PoA del protocol PAPI en el llenguatge de programació Python. Així doncs, s'ha contribuït en un projecte de programari lliure com és PAPI, protocol de SSO desenvolupat per RedIRIS que s'utilitza àmpliament en els sistemes SSO de la comunitat academico-científica espanyola. En

concret, s'ha contribuït desenvolupant el component PoA en un llenguatge de programació, com és Python, per al qual no existia cap altra implementació. D'aquesta manera, a partir d'ara els desenvolupadors web podran utilitzar aquest llenguatge per integrar aplicacions web desenvolupades en Python en entorns SSO basats en PAPI.

Com a experiència personal, cal destacar que la realització d'aquest projecte ha suposat un gran repte per a l'alumne per diferents motius:

- D'una banda, s'ha hagut d'estudiar i comprendre a nivell tècnic un protocol desconegut per complet per a l'alumne com és PAPI. A més, l'alumne tampoc havia tingut cap experiència prèvia amb ningun altre sistema Single Sign-On.

- A més, malgrat que l'alumne sí que tenia experiència prèvia amb el llenguatge de programació Python, ha hagut d'utilitzar llibreries estàndard i de tercers amb les quals mai havia treballat abans com: M2Crypto, logging, Cookie, Cpickle, ConfigParser, etc, per la qual cosa s'ha hagut de realitzar molt treball de documentació durant la fase de desenvolupament.

- D'altra banda, malgrat l'existència de components PoA desenvolupats en altres llenguatges com EasyPoA en PHP, que han suposat un gran suport, el projecte havia de desenvolupar-se partint de zero, i el resultat final ha estat una llibreria amb més de 1500 línies de codi, la qual cosa pot donar una idea del seu volum i complexitat.

- Addicionalment, durant la realització del projecte l'alumne ha hagut de fer front a diverses dificultats i imprevists que han anat sorgint com els que s'han citat en l'apartat 7.2.

Tot això ha suposat una experiència enriquidora i satisfactòria per a l'estudiant, sobretot després d'haver aconseguit fer front a totes les dificultats del projecte.

També m'agradaria destacar el paper que ha tingut el programari lliure en la realització d'aquest projecte. Tots els programes que s'han utilitzat per al desenvolupament del mateix tenen associada una llicència lliure, la qual cosa demostra que en l'actualitat existeix un gran ventall de programes lliures que han adquirit un alt grau de maduresa i que permeten desenvolupar projectes com el present. IDEs com Eclipse, llenguatges de programació com Python, servidors web com a Apache, navegadors com Firefox, etc permeten realitzar aquest tipus d'afirmacions.

Finalment, també m'agradaria destacar el suport rebut per part del consultor de la UOC i del tutor extern de PriSE que sempre s'han mostrat disposats a ajudar-me en tot moment. Sense ells hagués estat practicament impossible dur a terme aquest projecte.

Capítol 8 – Referències bibliogràfiques

[1] http://www.papisoftware.net

[2] http://papi.rediris.es/

[3] The PAPI Development Team. "*A Detailed Description of the PAPI Protocol*". http://papi.rediris.es/rep/PAPI_Protocol_Detailed.pdf

[4] Rodrigo Castro Rojo. "PAPI COMO INFRAESTRUCTURA DE SEGURIDAD DISTRIBUIDA APLICADA A ENTORNOS DE FUSIÓN TERMONUCLEAR". Tesis doctoral. UNED 2010. http://e-spacio.uned.es:8080/fedora/get/tesisuned:IngInf-Rcastro/Documento.pdf

[5] www.prise.es

[6] http://creativecommons.org/

[7] Python Documentation. http://www.python.org/doc/

[8] M2Crypto Project. http://chandlerproject.org/Projects/MeTooCrypto

[9] http://www.apache.org/

[10] Python Sessions http://code.activestate.com/recipes/577524-python-sessions/?in=lang-python

Anexos

A.1 Manual d'usuari

A.1.1 Introducció

- Nom de la llibreria: **EasyPyPoA**
- Tecnologia: **Python 2**
- Compatibilitat PAPI: **PoA**
- Llicència: **Apache License**

La llibreria EasyPyPoa implementa el comportament d'un PoA de PAPI lleuger. Es tracta d'un programari senzill i fàcil de desplegar desenvolupat en Python.

A.1.2 Instal·lació

La llibreria EasyPyPoA requereix d'un servidor web que sigui capaç d'executar *cgi scripts* Python, com per exemple Apache 2. A més, és necessari que la màquina on s'executi tingui instal·lada la llibreria M2Crypto:

- http://chandlerproject.org/Projects/MeTooCrypto

També és possible descarregar la llibreria M2Crypto des dels repositoris que inclouen les diferents distribucions de GNU/Linux. Per exemple, en una distribució basada en Debian podríem executar la següent ordre per instal·lar M2Crypto:

```
aptitude install m2crypto
```

Per instal·lar EasyPyPoA, en primer lloc cal descomprimir el paquet descarregat sota un directori del servidor web. Per a això es pot executar la següent ordre:

```
tar xzvf EasyPyPoA-0.1.tar.gz
```

Una vegada descomprimit ens trobem amb els següents fitxers i directoris:

```
EasyPyPoA/
index.py
exit.py
setup.py
LICENSE.txt
README.txt
```

On,

- **EasyPyPoA/**: Aquest directori conté les classes Python que formen la llibreria.
- **index.py**: és un fitxer de test per provar el comportament de EasyPyPoA quan s'intenta accedir a un recurs protegit.
- **exit. py**: és un fitxer de test per provar el comportament de EasyPyPoA quan es vol tancar una sessió oberta en PAPI.
- **setup.py**: és el script Python que s'ha d'executar per instal·lar la llibreria en el sistema.
- **LICENSE.txt**: és el fitxer amb la llicència *Apache License*.
- **README.txt**: és el fitxer que inclou les instruccions per instal·lar correctament la llibreria.

A més, dins de directori **EasyPyPoA/** existeix un subdirectori anomenat **conf/** que conté l'arxiu de configuració de la llibreria que s'utilitzarà per defecte (**PAPIPoA.ini**).

Per instal·lar la llibreria en el sistema simplement cal accedir com a administrador al directori on es troben tots els fitxers anteriors i executar l'ordre:

```
python setup.py install
```

Aquesta ordre copiarà la llibreria EasyPyPoA en el directori del sistema on es troben les llibreries i mòduls Python desenvolupats per tercers (no pertanyents a la biblioteca estàndard). Per exemple, en la distribució Debian 6 acabaria copiant-se en el directori:

```
/usr/local/lib/python2.6/dist-packages/EasyPyPoA/
```

Una altra opció seria copiar la llibreria en qualsevol directori inclòs en el PYTHONPATH.

A.1.3 Configuració

Com s'ha comentat, per defecte EasyPyPoA busca el fitxer de configuració **PAPIPoA.ini** en el subdirectori *conf/* que es troba dins de la carpeta que conté les classes que formen la llibreria. No obstant això, com veurem en el següent apartat, també és possible crear el fitxer de configuració en qualsevol altre *path* del sistema, ja que al instanciar la classe EasyPyPoA és possible indicar-li on es troba el fitxer de configuració.

El fitxer de configuració PAPIPoA.ini té el format típic dels fitxers de configuració .INI. Existeix una secció especial denominada **[DEFAULT]** i una secció **[ServiceID]** per cada recurs que es desitgi protegir.

A continuació es mostra un exemple:

```
[DEFAULT]
Lcook_Timeout         = 60
URL_Timeout           = 5
Current_Time_Windows  = 2
GPoA_URL              = http://adas.example.com/adas/GPoA
Pubkeys_Path          = /usr/local/apache2/papi/EasyPyPoA/keys
Attribute_Separator   = ","
Value_Separator       = "|"
LogLevel              = DEBUG
LogFile               = /var/log/apache2/EasyPyPoA/PAPIEasyPyPoA.log
End_Logout            = http://www.example.com
Hook_Logout           = http://www.example.com/logout.py

[ServiceID_1]
Location              = /path/to/service_ID_1/

[ServiceID_2]
Location              = /path/to/service_ID_2/
GPoA_URL              = http://papi.example.net/PAPIGPoA

[ServiceID_3]
Location              = /path/to/service_ID_3/
GPoA_URL              = http://example.com/OddGPoA
Pubkeys_Path          = /usr/local/apache2/papi/EasyPyPoA/other_keys
```

Secció DEFAULT

La secció [DEFAULT] defineix tots els paràmetres comuns per a tots els recursos que es protegeixin amb la mateixa instància de EasyPyPoA.

Els paràmetres definits en la secció [DEFAULT] seran heretats per la resta de seccions (que es corresponen amb la configuració per a cada recurs), tret que siguin redefinits. La secció [DEFAULT] no és obligatòria.

Seccions de recursos

Aquestes seccions defineixen els recursos a ser protegits per EasyPyPoA, on *[ServiceID]* és l'identificador del recurs i el que proporciona nom a la secció.

Aquestes seccions han de tenir sempre el paràmetre *Location* que indica el path del recurs en el servidor web que es desitja protegir. La resta de paràmetres són opcionals, ja que es poden heretar de la secció [DEFAULT] o prendre els valors per defecte.

Paràmetres de configuració

- **Lcook_Timeout**: Defineix el temps de vida per a la sessió. És a dir, és el temps durant el qual no es tornaran a comprovar les credencials contra el IdP una vegada l'usuari ja s'hagi autenticat. Es mesura en segons. El seu valor per defecte és 3600.
- **URL_Timeout**: Defineix el temps màxim que pot passar entre que EasyPyPoA pregunta a un GPoA i aquest li contesta. Es mesura en segons i el seu valor per defecte és 10.
- **Current_Time_Windows**: Finestra de temps que permet el funcionament entre sistemes que tenen un desfasament de temps. Es mesura en segons i el seu valor per defecte és 10.
- **GPoA_URL**: URL del GPoA a qui pregunta EasyPyPoA si no té dades sobre l'usuari que tracta d'accedir al recurs.
- **Pubkeys_Path**: Ruta del directori on es troba la clau pública del GPoA. El fitxer ha de tenir el següent nom *_GpoA_pubkey.pem .* Si no s'indica el valor per defecte serà el directori *conf/keys/*, situat on es troben les classes que implementen la llibreria.
- **Attribute_Separator**: Caracter usat per separar els noms dels atributs dels seus valors en l'asserció enviada des del GPoA a EasyPyPoA.
- **Value_Separator**: Caracter usat per separar els diferents possibles valors dels atributs en l'asserció enviada des del GPoA a EasyPoA.
- **LogLevel**: Nivell de log de EasyPyPoA. Els diferents valors que pot tenir són: DEBUG, INFO, WARNING, ERROR i CRITICAL.
- **LogFile**: Ruta i nom del fitxer on s'enviaran els missatges de log de EasyPyPoA.
- **Location**: Indica el path del recurs en el servidor web que es desitja protegir.
- **End_Logout**: URL on l'usuari serà redirigit en finalitzar el procés de logout.
- **Hook_Logout**: Mètode o funció definit per l'usuari que serà invocada per EasyPyPoA quan es realitza un logout amb la intenció de permetre que la *Relying Party*, és a dir, l'aplicació que confia en EasyPyPoA faci també el logout.

A.1.4 Exemple d'us de EasyPyPoA

Per provar el funcionament de EasyPyPoA amb els scripts Python de prova inclosos en la llibreria (**index.py** i **exit.py**) hem de seguir els següents passos:

1. Preparar el servidor web perquè pugui executar scripts Python. Per exemple, en Apache es poden afegir al fitxer de configuració les següents directives:

```
    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
    <Directory "/usr/lib/cgi-bin">
      AllowOverride None
      Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
      Order allow,deny
      Allow from all
    </Directory>
```

D'aquesta forma, podem situar qualsevol aplicació web Python en el directori `/usr/lib/cgi-bin/` i accedir a ella. Per exemple, si incloem un script anomenat *test.py* en aquest directori i intentem accedir des de localhost haurem d'utilitzar l'adreça:

```
http://localhost/cgi-bin/test.py
```

2. Descomprimir el fitxer Easypypoa-0.1.tar.gz en un directori del servidor web.

```
tar xzvf EasyPyPoA-0.1.tar.gz
```

3. Accedir al directori que s'ha creat en extreure el fitxer comprimit (*EasyPyPoA-0.1/*) i executar com a administrador el fitxer ***setup.py*** com s'indica:

```
cd EasyPyPoA-0.1/
python setup.py install
```

Aquesta ordre copia la llibreria **EasyPyPoA** en el directori del servidor que conté les llibreries Python desenvolupades per tercers. Per exemple, en Debian serà copiada en un directori similar a `/usr/local/lib/python2.6/dist-packages/`. A partir d'aquest moment, els desenvolupadors ja poden importar la llibreria EasyPyPoA des de les seves aplicacions o scripts.

4. Crear un directori que serà el recurs a protegir. En aquest exemple crearem el directori EasyPyPoA dins del directori establert en la configuració del servidor web. En aquest cas es crearà el directori `/usr/lib/cgi-bin/EasyPyPoA`. En aquest directori s'han de copiar els scripts de prova *index.py* i *exit.py*.

```
mkdir /usr/lib/cgi-bin/EasyPyPoA/
cp index.py exit.py /usr/lib/cgi-bin/EasyPyPoA/
```

5. Configurar EasyPyPoA. Per a això podem editar el fitxer de configuració que ve per defecte amb la llibreria, o crear un fitxer de configuració nou. En aquest exemple s'edita el fitxer de configuració que ve per defecte en EasyPyPoA `/usr/local/python2.6/dist-packages/EasyPoA/conf/PAPIPoA.ini`. Per exemple:

```
[DEFAULT]
Lcook_Timeout          =  60
URL_Timeout            =  5
Current_Time_Windows   =  2
GPoA_URL               =  http://exemple.com/URL/gpoa.py
Pubkeys_Path = /usr/local/python2.6/dist-packages/EasyPoA/conf/keys/
Attribute_Separator    = ","
```

```
Value_Separator      = "|"
LogLevel             = DEBUG
LogFile =  /var/log/apache2/EasyPyPoA/PAPIEasyPyPoA.log
End_Logout           = http://exemple.com

[TestEasyPoA]
Location = /cgi-bin/EasyPyPoA/
```

6. Crear, en cas que no existeixi, el directori destinat a emmagatzemar la clau pública del GPoA. En aquest cas:

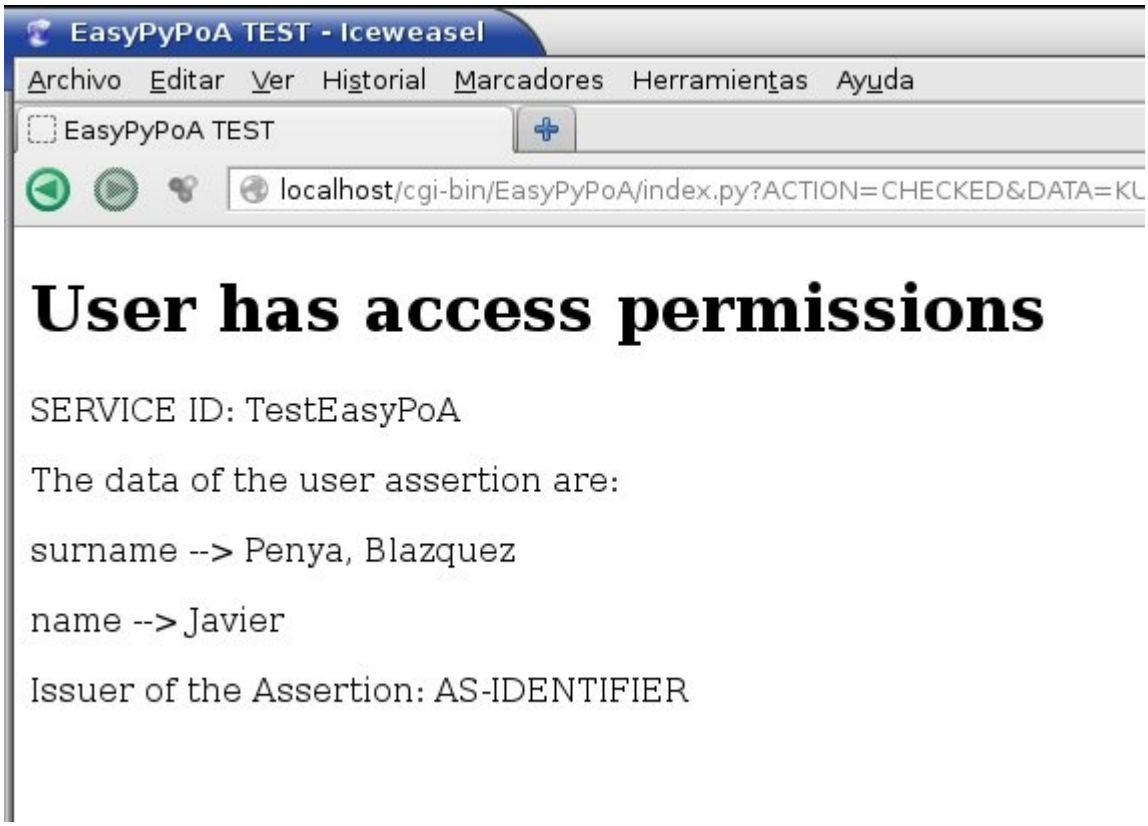
```
mkdir /usr/local/python2.6/dist-packages/EasyPoA/conf/keys/
```

7. Copiar en aquest directori la clau publica del GPoA la URL del qual hem indicat, amb el nom de fitxer ***_GPoA_pubkey.pem***.

8. Modificar el fitxer ***index.py*** que prèviament s'ha copiat al directori /usr/lib/cgi-bin/EasyPyPoA/, perquè instàncie un objecte EasyPyPoA amb els paràmetres adequats, sent el primer el *ServiceID* i el segon el *Path* al fitxer de configuració (si és que no es desitja usar el fitxer per defecte). Per exemple:

```
.
..
...
try:
    poa = EasyPyPoA('TestEasyPoA')
    if poa.check_access():
        body = '<body>\n'
        body += '<h1>User has access permissions</h1>\n'
        body += '<p>SERVICE ID: ' + poa.get_serviceID() + '</p>'
        body += '<p>The data of the user assertion are: </p>\n'
        userData = poa.get_user_data()
        for attr, values in userData.items():
              body += '<p>' + attr + ' --> '
                for v in values:
                    body += v + ', '
                body = body[:-2]
                body += '</p>'
        body += 'Issuer of the Assertion: ' + poa.get_issuer()
        body += '</body>\n</html>\n'
    else:
        body = '<body>\n'
            body += '<h1 style="color:red;">User has not access
permissions</h1>\n'
        body += '</body>\n</html>\n'
...
..
.
```

9. Ja solament resta accedir a la url del recurs que s'acaba de protegir. Per a això obrim el navegador i introduïm la url per accedir al script *index.py*. Per exemple:



10. Per tancar la sessió accedir al script *exit.py*.