

UNIVERSIDAD ANDRÉS BELLO

Facultad de Ingeniería

Ingeniería en Computación e Informática

**MEJORA DE SEGURIDAD A SERVIDOR XMPP OPENFIRE IMPLEMENTANDO MECANISMO DE FIRMADO DE FORMULARIOS AL SISTEMA DE REGISTRO EN BANDA.**

Tesis de pregrado para optar al título de Ingeniero en Informática.

Marcelo Antonio Aros Aros

Profesor guía Romina Débora Torres Torres

Viña del Mar de Chile, 2018.

Agradecimientos

…

# Abreviaturas.

* N.L.A. No Listo Aún.
* IoT: Internet of Things o Internet de las cosas.
* IETF: Internet Engineering Task Force.
* XMPP: Extensible Messaging and Presence Protocol (Protocolo extensible de mensajería y comunicación de presencia).
* JSF: Jabber Software Foundation (Fundación de Software Jabber)
* XSF: XMPP Standards Foundation (Fundación de estándares de XMPP)
* XEP: XMPP Enhancement Proposals (Propuestas de mejoras a XMPP)
* JID: Jabber ID.
* RFC: Request for Comments
* PO: Product Owner
* TCP: Transmission Control Protocol o Protocolo de Control de Transmisión
* UDP: User Datagram Protocol
* SPX: Internetwork Packet Exchange/Sequenced Packet Exchange
* RPC: Remote procedure call. Llamada a procedimiento remoto.x
* SNMP: Simple Network Management Protocol o Protocolo Simple de Administración de Red
* SMTP: Simple Mail Transfer Protocol o Protocolo para transferencia simple de correo
* NNTP: Network News Transport Protocol o Protocolo para la Transferencia de Noticias en Red
* FTP: File Transfer Protocol o Protocolo de Transferencia de Archivos
* SSH: Secure Shell o Intérprete de Órdenes seguro
* HTTP: Hypertext Transfer Protocol o Protocolo de Transferencia de Hipertexto
* CIFS: Common Internet File System o Sistema Común de Archivos de Internet
* IM: Instante Mensajería instantánea

# Índice de contenidos

[Abreviaturas. 2](#_Toc465070651)

[Índice de contenidos 3](#_Toc465070652)

[Índice de imágenes 5](#_Toc465070653)

[Índice de Tablas 5](#_Toc465070654)

[Resumen. 6](#_Toc465070655)

[1. INTRODUCCIÓN 7](#_Toc465070656)

[2. FUNDAMENTACIÓN 8](#_Toc465070657)

[2.1. Contextualización el proyecto 8](#_Toc465070658)

[2.1.1. XMPP 8](#_Toc465070659)

[2.1.1.1. Protocolo 8](#_Toc465070660)

[2.1.1.2. Historia 8](#_Toc465070661)

[2.1.1.3. XSF 11](#_Toc465070662)

[2.1.1.4. XEPs 11](#_Toc465070663)

[2.1.1.5. Funcionamiento 14](#_Toc465070664)

[Direccionamiento 15](#_Toc465070665)

[2.1.1.1.5. Desentralizado 15](#_Toc465070666)

[2.1.1.6. Usos 16](#_Toc465070667)

[2.1.1.7. Seguridad 17](#_Toc465070668)

[2.2. Problema 18](#_Toc465070669)

[2.2.1. Objetivo General 18](#_Toc465070670)

[2.2.2. Objetivos Específicos 19](#_Toc465070671)

[2.2.3. Restricciones 19](#_Toc465070672)

[2.2.4. Requerimientos No Funcionales 19](#_Toc465070673)

[2.3. Alternativas de Solución 19](#_Toc465070674)

[2.3.1. Alternativa 1: Implementar VPN 19](#_Toc465070675)

[2.3.2. Alternativa 2: CREAR XEP, PLUGIN Y LIBRERÍA QUE IMPLEMENTEN OAUTH 2.0 19](#_Toc465070676)

[2.3.3. Alternativa 3: Crear Plugin y librería que implemente la Xep-0348 la cual implementa Oauth 19](#_Toc465070677)

[2.3.4. Comparativa de Alternativas 20](#_Toc465070678)

[2.3.5. Elección de Alternativa 20](#_Toc465070679)

[2.4. Solución Propuesta 20](#_Toc465070680)

[2.4.1. Factibilidad Técnica 21](#_Toc465070681)

[2.4.1.1. Lenguajes de programación y Herramientas 21](#_Toc465070682)

[2.4.1.2. Hardware 21](#_Toc465070683)

[2.4.2. Alcances de solución 21](#_Toc465070684)

[2.4.3. Limitaciones de la solución 22](#_Toc465070685)

[2.4.4. Esquema de Solución 23](#_Toc465070686)

[2.4.4. Factores de éxito 23](#_Toc465070687)

[3. MATERIALES Y MÉTODOS 23](#_Toc465070688)

[3.1. METODOLOGÍA DE DESARROLLO 23](#_Toc465070689)

[3.1.1. ROLES Y RESPONSABILIDADES 25](#_Toc465070690)

[3.2. AMBIENTE DE DESARROLLO 26](#_Toc465070691)

[3.2.1. Gestión de la Configuración 26](#_Toc465070692)

[3.2.1.1. Control de Cambios 26](#_Toc465070693)

[3.2.1.2. Control de versiones 26](#_Toc465070694)

[3.2.1.3. Entregables 26](#_Toc465070695)

[3.2.2. Lenguajes de Programación 27](#_Toc465070696)

[3.2.3. Herramientas de Desarrollo 27](#_Toc465070697)

[3.3. Ambiente de Producción 27](#_Toc465070698)

[3.4. Ambientes de Pruebas 28](#_Toc465070699)

[3.4.1. Elementos deL ambiente 28](#_Toc465070700)

[3.4.2. Descripción general de las pruebas a realizar. 28](#_Toc465070701)

[3.5. Plan de Proyecto 28](#_Toc465070702)

[3.5.1. Gestión de las Comunicaciones 29](#_Toc465070703)

[3.5.1.2. Gestión de la Resolución de Conflictos 29](#_Toc465070704)

[3.5.2 Gestión de Riesgos 31](#_Toc465070705)

[3.5.2.1. Identificación de Riesgos 31](#_Toc465070706)

[3.5.2.2. Estrategias de Mitigación 32](#_Toc465070707)

[3.5.2.3. Estrategias de Contingencia 32](#_Toc465070708)

[3.5.3. Gestión de Requerimientos 33](#_Toc465070709)

[4. RESULTADOS Y DISCUSIÓN 33](#_Toc465070710)

[4.1. Historias de Usuario 34](#_Toc465070711)

[5. CONCLUSIONES 34](#_Toc465070712)

[5.1. Post-Mortem 34](#_Toc465070713)

[5.2. Trabajos Futuros 34](#_Toc465070714)

[Referencias 34](#_Toc465070715)

# Índice de imágenes

[Ilustración 1: Ciclo de vida de un XEP 13](#_Toc465151410)

[Ilustración 2: Direccionamiento 16](#_Toc465151411)

[Ilustración 3: «Stanzas» elementales 16](#_Toc465151412)

[Ilustración 4: XMPP Descentralizado 18](#_Toc465151413)

[Ilustración 5: OAuth 20](#_Toc465151414)

[Ilustración 6: Esquema se Solución 25](#_Toc465151415)

[Ilustración 7: Metodología de desarrollo de proyecto - SCRUM 26](#_Toc465151416)

[Ilustración 8: Metodología de desarrollo 27](#_Toc465151417)

# Índice de Tablas

[Tabla 1: Requerimientos No Funcionales 21](#_Toc465151398)

[Tabla 2: Comparativa entre Alternativas de Solución 22](#_Toc465151399)

[Tabla 3: Roles SCRUM 27](#_Toc465151400)

[Tabla 4: Roles y responsabilidades de metodología 28](#_Toc465151401)

[Tabla 5: Estructura de control de versiones Git. 28](#_Toc465151402)

[Tabla 6: Entregables 29](#_Toc465151403)

[Tabla 7: Herramientas de desarrollo 29](#_Toc465151404)

[Tabla 8: Tipos de Comunicación 31](#_Toc465151405)

[Tabla 9: Resolución de conflictos 32](#_Toc465151406)

[Tabla 10: Identificación de riesgos. 33](#_Toc465151407)

[Tabla 11: Estrategias de Mitigación 34](#_Toc465151408)

[Tabla 12: Estrategias de contingencia 35](#_Toc465151409)

# Resumen.

Una definición acertada del concepto Internet de las Cosas, es: «Internet of Things, o IoT, es lo que obtenemos cuando conectamos Cosas, que no son operadas por humanos, a Internet». Estas cosas, pueden ser sensores: de movimiento, temperatura, humedad, etc. Actuadores­: cercos electromecánicos, pantallas de información, equipos de aire acondicionado, etc. En este contexto, existen variados protocolos de comunicaciones, donde cada uno plantea propuestas interesantes, en torno a la definición mencionada de IoT.

El Protocolo extensible de mensajería y comunicación de presencia (XMPP), mejor conocido por XMPP, es un protocolo abierto y extensible basado en Lenguaje de Marcado Extensible (XML), permite el intercambio casi en tiempo real de datos estructurados pero extensibles entre dos o más entidades de red, debido a esto, originalmente ideado para mensajería instantánea. Este protocolo salió vencedor, luego de ser comparados en características útiles para el IoT con otros protocolos de comunicación, como lo son CoAP, MQTT y HTTP. Sin embargo, al momento de su evaluación aún presentaba vulnerabilidades de importantes de seguridad en el mecanismo de registro de identidades en-Banda, frente a esta problemática. Lo cual, eventualmente permitiría a hackers registrar identidades indiscriminadamente, de modo que, sobrecargaría las bases de datos, bajaría el servidor, y podría interactuar con identidades de la red XMPP, como lo son sensores, actuadores, y controladores, todo esto bajo la anonimidad. Este problema se demostró, creando un bot, el cual crea identidades virtualmente ilimitadas en una red vulnerable con un servidor XMPP Openfire. Para solucionar la problemática descrita, se modificaron y añadieron componentes al servidor XMPP Openfire, y la librería cliente XMPP SMACK de modo que, se corrigieron las vulnerabilidades en el registro en-Banda por medio de implementación de un sistema firmado de formularios, mediante el mecanismo OAuth 1.0 modificado. De esta forma, registrar de forma segura identidades – Cosas – en la red XMPP-IoT.

# 1. INTRODUCCIÓN

Desde el mismo origen del internet, lo que se ha buscado es impulsar las comunicaciones o en su defecto el intercambio de información, lo cual ha motivado a diferentes personas o grupos de trabajos, diseñar estándares de comunicación entre dispositivos de cómputo, dicho de otro modo, diseñar protocolos de comunicaciones entre computadores. Ya sean estos como lo son TCP, UDP, SPX, en la capa de transporte, según el modelo OSI. O también, NetBios, RPC, SSL, en la capa 5 de sesión. De igual modo, tenemos en la capa siete de aplicación, una serie de protocolos, como, por ejemplo, SNMP, SMTP, NNTP, FTP, SSH, HTTP, CIFS. Centrándonos en esta última capa también tenemos a XMPP, un protocolo inicialmente desarrollado para IM mensajería instantánea, el cual ha pasado por una serie etapas de adiciones y modificaciones de sus especificaciones.

XMPP ha estado en contante mejora a lo largo de los años gracias al aporte constante de la gran comunidad en torno al protocolo y la XMPP Software Fundation, que se dedica a estandarizar el proyecto, se ha logrado multiplicar las áreas de uso de este protocolo. Dentro de las cuales está, primeramente, la mensajería instantánea, para lo cual este protocolo se pensó originalmente, luego grupos de chat, control de sistemas, VoIP e IoT. Esta última área es la que nos interesa en el marco del presente documento. Ya que XMPP fue pensado inicialmente como un protocolo de comunicaciones tiene vulnerabilidades abiertas cuando hablamos de algún área que no ha sido originalmente diseñado. El problema en resumidas cuentas radica en que la creación de identidades sólo está regulada por CAPTCHAS en las especificaciones existentes de protocolo, esto evita la creación de identidades por medio de bots maliciosos, sin embargo, no delimita a los usuarios maliciosos que deseen crear identidades. Por esta razón Peter Waher, ha creado especificaciones de Extensiones del protocolo con él fin de dar una solución adecuada a la problemática de la seguridad en XMPP con IoT como área de uso.

En el presente documento, se presenta la historia de XMPP, como este funciona, la manera que se utiliza en el internet de las cosas y las mejoras que se proponen para mejorar la seguridad del protocolo.

# 2. FUNDAMENTACIÓN

## 2.1. Contextualización el proyecto

### 2.1.1. XMPP

#### 2.1.1.1. Protocolo

XMPP es un protocolo libre de comunicaciones, según el estándar de software libre. Este funciona sobre la capa siete, de aplicación de acuerdo con el modelo OSI. Este protocolo, fue pensado para ser utilizado en mensajería instantánea (IM por sus siglas en ingles), está basado en el metalenguaje XML (eXtensible Markup Language).

Su nombre es acrónimo sus siglas en inglés:

* P – Protocolo

XMPP es un protocolo; un conjunto de estándares que permiten a los sistemas que se comuniquen entre sí.

* P – Presencia

El indicador de presencia les dice a los servidores si un cliente está en Online[[1]](#footnote-1) / Offline[[2]](#footnote-2) / Busy[[3]](#footnote-3). En términos técnicos, la presencia determina el estado de una entidad XMPP; en términos simples si se está listo para recibir mensajes o no.

* M – Mensajería

La parte de “mensajería” de XMPP es la parte que se ve; el mensaje instantáneo (IM) enviado entre clientes. XMPP ha sido diseñado para enviar todos los mensajes en tiempo real, utilizando un mecanismo de envío muy eficiente; mientras que a menudo los mecanismos basados en web hacen muchas peticiones innecesarias lo que produce una carga de la red, y por lo tanto no son en tiempo real.

* X – eXtensible

XMPP está definido en un estándar abierto, usando el mismo enfoque para sistema para el desarrollo. En otras palabras, se ha diseñado para crecer y adaptarse a los cambios

Entre las definiciones formales están las realizadas por la IETF, que es la RFC 6120[[4]](#footnote-4), la cual describe el núcleo (core) de XMPP, y la RFC6121[[5]](#footnote-5) describe el comportamiento XMPP en la mensajería instantánea (IM por sus siglas en inglés).

#### 2.1.1.2. Historia de XMPP

**1998**:

Jeremie Miller anuncia la existencia de Jabber; una tecnología abierta para la mensajería instantánea y presencia. A lo largo del año de desarrollo se mueve rápidamente en un servidor de código abierto, jabberd, varios clientes de código abierto y bibliotecas de códigos y protocolos open-wire para la transmisión XML en tiempo real; así como extensiones de mensajería instantánea. Estos mismos protocolos (con varias mejoras y ampliaciones) todavía están en uso hoy en día.

**2000**:

Febrero: La IETF, publica un modelo y un conjunto de requisitos para los sistemas de mensajería instantánea y la presencia, realizado por el grupo de trabajo de Instant Messaging and Presence Protocol (IMPP).

Mayo: La versión 1.0 del servidor jabberd se libera y se estabilizan los protocolos Jabber de base (de transmisión XML, mensajería, presencia, listas de contactos, etc.).

Octubre: jabberd 1.2 se libera y el protocolo de devolución de llamada del servidor introducido para evitar la falsificación de direcciones debido al rápido crecimiento de la red de servidores Jabber

**2001**:

Agosto: Comienza la formación de la Jabber Software Foundation «JSF» la cual, tiene la función de coordinar el creciente número de proyectos de código abierto y las entidades comerciales que construyen y/o usan las tecnologías Jabber. El punto clave de la JSF es la gestión de los protocolos que se utilizan dentro de la comunidad Jabber/XMPP, mediante la documentación de los protocolos existentes y el desarrollo de extensiones de protocolo a través de un proceso de estándares abiertos.

**2002**:

Octubre: La formación del Grupo de Trabajo XMPP es aprobado por el Grupo de Dirección de Ingeniería de Internet (IESG: Internet Engineering Steering Group). Como resultado, JSF contribuye formalmente los protocolos Jabber de protocolo base al Internet Standards Process y asigna el control de cambios sobre los protocolos del IETF, bajo el nombre Extensible Messaging and Presence Protocol (XMPP).

Noviembre: Primera reunión del Grupo de Trabajo XMPP se llevó a cabo en IETF 55 e incluye presentaciones de Jeremie Miller, Joe Hildebrand, y Peter Saint-Andre.

**2003**:

Grupo de Trabajo XMPP se dedica a trabajar esencialmente la formalización de la base del protocolo Jabber a fin de lograr su adaptación, como los protocolos de mensajería instantánea y presencia aprobados por la IETF; revisiones posteriores realizadas en los protocolos básicos se centran en la mejora de la seguridad y la internacionalización.

Jabber Inc. presenta un Aviso de derechos de propiedad intelectual a la IETF en relación con la marca comercial Jabber.

**2004**:

Octubre: El IETF publica el RFC 3920[[6]](#footnote-6) y RFC 3921[[7]](#footnote-7) que define la funcionalidad básica de XMPP como estándares propuestos.

Tras la publicación de los documentos RFC XMPP, el IETF anuncia la conclusión del Grupo de Trabajo XMPP. Sin embargo, el desarrollo de nuevas extensiones XMPP continúa en la JSF.

**2005**:

Agosto: Implementación y despliegue de servicios basados ​​en XMPP a gran escala continúa, destacan por el lanzamiento de Google Talk.

**2006**:

Junio-Agosto: El JSF participa en el Google Summer of Code.

Octubre: La JSF cambia el nombre de "Jabber Enhancement Proposals" (JEPs) a los " XMPP Extension Protocols" (XEPs) y los mueve al sitio xmpp.org.

Diciembre: El JSF entra en una asociación con StartCom, para ofrecer certificados digitales gratis a los administradores de servidores Jabber / XMPP a través de la Autoridad de Certificación Intermedio XMPP.

**2007**:

Enero: La «Jabber Software Foundation» (JSF) se renombra a «XMPP Standards Foundation» (XSF) para describir con mayor precisión su enfoque en el desarrollo de extensiones de protocolo abierto XMPP con las especificaciones de la base de la IETF, de software de código abierto.

**2011**:

Febrero: El XSF mantiene la décima Cumbre XMPP en Bruselas, que comparte el edificio con una reunión intermedia del Grupo de Trabajo de la IETF XMPP, con un enfoque especial en la internacionalización.

Marzo: El IETF publica actualizaciones a el RFC 6120[[8]](#footnote-8) y RFC 6121[[9]](#footnote-9) la a de la definición del núcleo de XMPP.

**2012**:

Febrero: El XSF completa revisiones significativas de la multi-usuario de la extensión de chat XMPP XEP-0045[[10]](#footnote-10).

**2014**:

Enero: El XSF tiene la decimoquinta Cumbre XMPP en Bruselas, Bélgica.

Mayo: Los operadores de los servidores de la red XMPP pública, comienzan a actualizar de forma permanente el cifrado de las comunicaciones.

Septiembre: El XSF tiene la decimosexta Cumbre XMPP en Berlín, Alemania.

Octubre: El IETF publica el RFC 7395[[11]](#footnote-11) que define XMPP sobre WebSocket.

**2015**:

Febrero: El XSF mantiene la XVII Cumbre XMPP en Bruselas, Bélgica.

Junio: El IETF publica RFC 7590[[12]](#footnote-12) la actualización del uso de Transport Layer Security (TLS) en XMPP.

Septiembre: El IETF publica RFC 7622[[13]](#footnote-13) la modernización de la definición de direcciones XMPP.

#### 2.1.1.3. XSF

La Fundación de Estándares XMPP, también conocida como la XSF por sus siglas en inglés y anteriormente fue conocida por el nombre de Fundación de Software Jabber JSF, en la actualidad es una organización independiente de desarrollo sin fines de lucro, cuya misión principal es la de definir protocolos de extensión de XMPP, esto, en colaboración con IETF.

La XSF también proporciona información e infraestructura para la comunidad mundial de desarrolladores de Jabber / XMPP, proveedores de servicios y usuarios finales. Además, la XSF administra el programa de licencia de marca Jabber.

El núcleo de la XSF está compuesto por miembros electos. Que, a su vez, eligen un Consejo (la dirección técnica) y la Junta (Dirección de empresarial). El Consejo también nombra a varios equipos de trabajo, por ejemplo, el equipo redactor y el equipo de infraestructura, así como el director ejecutivo, tesorero y secretario.

Dentro de las bondades que ofrece la XFS, es que cualquier persona que tenga interés en XMPP y sus protocolos, puede ser parte de la comunidad XMPP, y eventualmente postular para ser miembro de la XSF.

De manera resumida, la Fundación de Estándares XMPP, tiene como función esencial la mantención y actualización del mismo protocolo, ya que crea las instancias en la que los interesados en corregir o crear nuevas funcionalidades en XMPP, puedan hacerlo. Generando cambios o nuevos XEPs, los cuales una vez siendo publicados, sean utilizados por la comunidad de administradores de servidores de XMPP en el mundo. De este modo, los clientes XMPP sean actualizados para incorporar dichas mejoras o nuevas funcionalidades. Estas especificaciones de funcionalidades son conocidas por el nombre de XEPs, Protocolos de extensión de XMPP, que son descritas en la sección que está a continuación.

#### 2.1.1.4. XEPs

La palabra XEP es un acrónimo que hace referencia a «XMPP Extension Protocols», cuyo significado en castellano es «Protocolos de Extensión de XMPP». En definitiva, las XEP son especificaciones de protocolos de extensión de XMPP, los cuales contienen una detallada descripción de cómo implementar una funcionalidad particular. Más, **no** son implementaciones, sino sólo la especificación del protocolo de extensión.

Los procesos que describen el ciclo de vida de los XEPs y como la XSF incide en ellos, están descritos en el XEP-0001[[14]](#footnote-14).

El avance de un XEP través del proceso de estandarización de la XSF depende de tres factores:

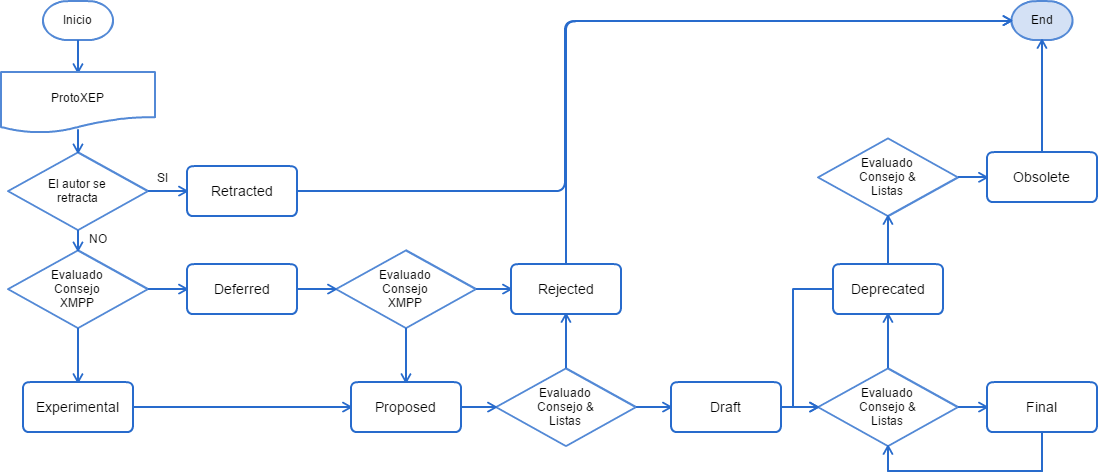
* Cierto consenso en las listas de discusión pública de la XSF.
* Ejecución de código en los clientes XMPP, servidores y bibliotecas.
* La aprobación formal por el Consejo XMPP.

La creación y posterior solicitud de publicación de una XEP, la puede hacer cualquier persona que haya creado un Protocolo de Extension para XMPP. Para realizar esta solicitud primero hay que realizar lo siguiente:

* Escribir la propuesta en formato XML siguiendo las directrices que se describen en el XEP-0143[[15]](#footnote-15): Directrices para los autores de protocolos de extensión XMPP.
* Antes de presentar su idea, leer y entender la política de derechos de propiedad intelectual de la XSF.
* Enviar un email con el archivo XML (o una URL para el archivo) al «Equipo Editor» (Editor Team) con una línea de asunto de "ProtoXEP: [Titulo del XEP]".

Luego de esto, el ProtoXEP entra al proceso, este es mostrado en la Figura 1.

Figura 1: Ciclo de vida de un XEP



Los posibles estados para un XEP están descritos a continuación:

* **Experimental**  
  Un XEP de cualquier tipo está en el estado «Experimental» después de que ha sido aceptado por el «Consejo XMPP» y publicado por la Fundación de Estándares XMPP (XSF) pero antes de que haya avanzado por el proceso a un estado de «Active» o «Draft»
* **Proposed**  
  Un XEP de cualquier tipo está en el estado de «Proposed», en castellano propuesto, mientras está en última convocatoria o en estudio por el Consejo de XMPP para el avance desde «Experimental» a «Draft» o «Active».
* **Draft**  
  Un XEP está en el estado de «Draft» después de que ha sido objeto de amplia discusión y revisión técnica en la lista estándares y ha sido votado positivamente para seguir su camino hacia la estandarización por el Consejo XMPP.
* **Final**  
  Un XEP alcanza el estado de «Final» después de que ha pasado por lo menos seis meses en estado de «Draft», además se ha implementado en al menos dos bases de código independientes, ha sido votado positivamente para su avance por el Consejo XMPP.  
  Una vez que un XEP ha avanzado a un estado de «Final», se hará todo lo posible para limitar el alcance de las modificaciones; en particular, no se harán cambios incompatibles. Sin embargo, las modificaciones limitadas se pueden realizar, siempre y cuando sean opcionales las modificaciones en los servidores y/o clientes que usan dicha XEP.  
  Por lo tanto, un protocolo «Final» es seguro para el despliegue de aplicaciones.
* **Active**  
  Un XEP que está en el proceso de estandarización se hace avanzar a un estado de «Active» después de que ha sido elegido para avanzar en el proceso de estandarización por el Consejo XMPP.
* **Deferred**  
  Un XEP Experimental de cualquier tipo se cambia al estado «Deferred» si no se ha actualizado en doce meses.
* **Retracted**  
  Un XEP está en el estado «Retracted» si el autor ha pedido al Editor de extensiones XMPP eliminar el XEP del proceso de estandarización de la XSF.
* **Rejected**  
  Un XEP está en el estado «Rejected» si el Consejo XMPP ha lo ha estimado inaceptable y ha votado para no moverlo hacia adelante en el proceso de estandarización.
* **Deprecated**  
  Un XEP está en el estado en «Depreated» si el Consejo XMPP ha determinado que el protocolo definido en él no está actualizado y que las nuevas implementaciones ya no se le utiliza (por ejemplo, porque ha sido reemplazado por un protocolo más moderno).
* **Obsolete**  
  Un XEP cambia de «Depreated» a «Obsolete» si el Consejo XMPP ha determinado que el protocolo definido en este documento ya no se ejecutará o no será desplegado.

#### 2.1.1.5. Funcionamiento

El RFC 3920, Extensible Messaging and Presence Protocol (XMPP) core: Describe la mensajería Cliente-servidor mediante el uso de flujos XML.

Los flujos de datos XML que utiliza XMPP para diversos fines son los listados a continuación:

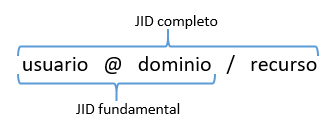
* Presence: Se encarga de suscribirse y desuscribirse a las «presencias» de otras entidades.
  + Envía constantemente la «presencia» de la entidad, para informar de esta y mantener la conexión persistente.
  + Se puede establecer un estado personalizado de las «presencia».
* message
  + Tipos:
    - Chat: mensaje en el contexto de una conversación uno-a-uno con historial de conversación.
    - Error: Ha producido un error relacionado con un mensaje previo enviado por el remitente.
    - Groupchat: El mensaje se envía en un contexto chat multiusuario. Es decir, a varios usuarios al mismo tiempo.
    - Headline: Mensaje probablemente autogenerado, es decir, puede ser noticias, información del mercado, RSS Feed, etc. A este mensaje no se espera respuesta.}
    - Normal: Es un mensaje fuera de contexto de una conversación uno-a-uno o de un grupo de chat, a la que se espera que el receptor responda.
  + Elementos hijos:
    - Subject: Contiene datos de carácter XML legible que especifica el tema del mensaje.
    - Body: Contiene datos de carácter XML legible por humanos que especifica el contenido textual del mensaje; este elemento hijo se incluye normalmente, pero es opcional.
    - Thread: Contiene datos de caracteres XML no legibles especificando un identificador que se utiliza para el seguimiento de un hilo de conversación (a veces conocido como un "sesión de mensajería instantánea") entre dos entidades. El valor del elemento <thread/> es generada por el emisor y deberá ser copiada de nuevo en todas las respuestas.
* iq: Info and Query, en español es, Información y Consulta
  + estrofas IQ proporcionan un mecanismo de petición-respuesta estructurada. La semántica básica de este mecanismo, por ejemplo, que el atributo 'id' es requerido, se definen en XMPP-CORE, mientras que la semántica específicos necesarios para completar los casos de uso particulares se definen en todos los casos por un espacio de nombres extendida (tenga en cuenta que la 'Jabber: cliente' y 'Jabber: servidor' espacios de nombres no definen ningún hijo de IQ estrofas que no sea el común <error />).

Una conexión XMPP puede ser autentificada mediante SASL[[16]](#footnote-16) (Opcional) y cifrada mediante SSL/TLS[[17]](#footnote-17).  
A grosso modo, SASL es un Framework para autenticación y autorización en internet. Sin embargo, este Framework no está cifrado, por ende, para cifrar los datos se puede ayudar de SSL[[18]](#footnote-18) o en su defecto TLS. Para ver más información sobre SASL, se puede revisar el RFC4422[[19]](#footnote-19) y para RLS el RFC5246[[20]](#footnote-20).

##### Direccionamiento

Para localizar una identidad en XMPP debemos utilizar direccionamiento en el caso de este protocolo tiene el nombre de «Jabber ID (JID)», el cual está compuesto por «nombre de usuario», «dominio» y un «recurso». En la Figura 2, podremos ver gráficamente como es un JID.

Figura 2: Direccionamiento en XMPP.



##### Stanzas

Las «Stanzas» son flujos XML, los cuales viajan por medio HTTP/S o SOCKETS interconectados obedeciendo a patrón cliente-servidor.

En la Figura 3, podemos visualizar las stanzas básicas.

Figura 3: Stanzas fundamentales en XMPP.



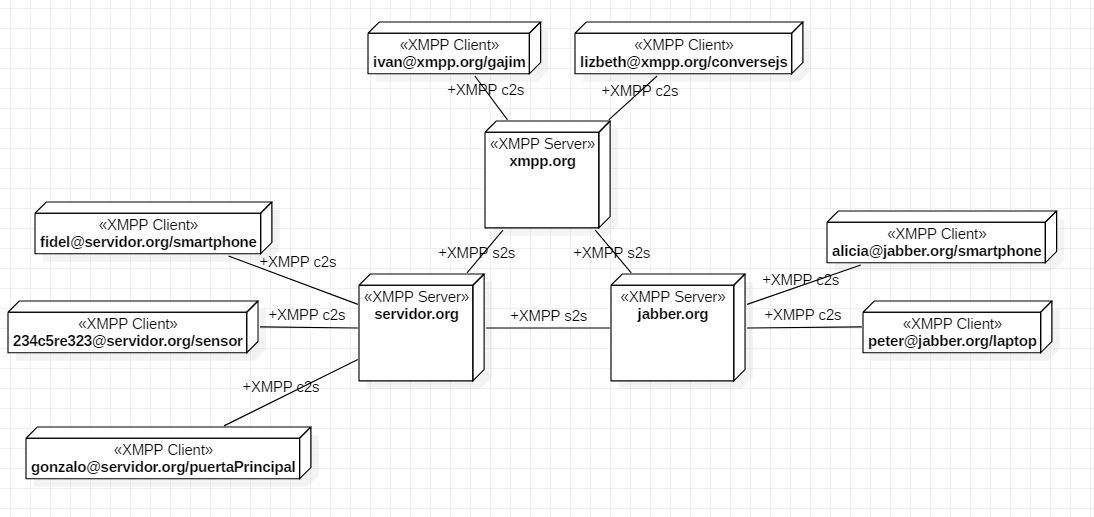
* En las «Stanzas» Todas tienen direcciones to='JID' and from='JID‘
  + «To» da el destino
  + «From» es agregado por servidor local.
* Cada «stanza» se rutea por separado
* Todo el contenido de la «stanza» es entregado.
* Different types for delivery semantics
  + <message/>: una dirección, un destinatario
  + <presence/>: una dirección, publica a muchos
  + <iq/>: «info/query» , request/response
* **Stream:stream:** Es utilizado para iniciar la conexión al servidor por TCP Socket.
  + El cliente envía el stream tag de inicio.
    - <?xml version='1.0'?>  
      <stream:stream xmlns="jabber:client"   
      to="binarylamp.cl" version="1.0“  
      xmlns:stream="http://etherx.jabber.org/streams" xml:lang="es" >
  + El servidor responde
    - <?xml version='1.0' encoding='UTF-8'?>  
      <stream:stream xmlns:stream="http://etherx.jabber.org/streams" xmlns="jabber:client" from="binarylamp.cl" id="7whd1lwsxk" xml:lang="es" version="1.0">
* **stream:features**: Es enviado por el servidor, con el fin para informar con que características cuenta este.
  + <stream:features>  
     <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>  
     <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>  
     <mechanism>DIGEST-MD5</mechanism>  
     </mechanisms>  
     <compression xmlns='http://jabber.org/features/compress'>  
     <method>zlib</method>  
     </compression>  
    </stream:features>
* **presence**: Tiene la función de informar el estado de la identidad.
  + <presence>  
     <show>dnd</show>  
     <status>Meeting</status>  
     <priority>1</priority>  
    </presence>
  + Show: chat, available, away, xa, dnd
  + Status: Texto leíble por humanos.
  + Priority: ¿Cuáles recursos requieren «mayor disponibilidad»?

##### 2.1.1.1.5. Desentralizado

Como se ha mencionado el protocolo XMPP tiene la facultad de permitir la mensajería instantánea (IM) entre los usuarios de manera accesible y menos compleja posible para ellos. De este modo, posee una arquitectura Cliente-Servidor descentralizada, es decir, la comunicación entre clientes no hay necesidad que pase por un único servidor central, sino que cualquier usuario tiene la capacidad de crear su propio servidor XMPP y adherirse a la red, teniendo un mínimo de seguridad, esto ayuda a que no exista una sobrecarga de la conexión (pensando en que XMPP está pensado para una gran cantidad de usuarios que requieren de la conexión sea rápida y confiable, como debe ser la IM), por otro lado, esto también puede significar una desventaja, ya que provoca que no exista un único punto de falla.

Como ayuda tenemos la gráfica de la Figura 4.

Figura 4: Federalización - descentralización - en XMPP.



#### 2.1.1.6. Usos

XMPP no solo se usa para mensajería instantánea, sino también para grupos de chat, Gamming, Control de Sistemas, Middleware, Cloud Computing, Data Sindicación, Voz sobre IP (VoIP), videoconferencias, Identity services y IoT, este último es en el ámbito que nos incumbe respecto al documento y proyecto.

Existen XEPs diseñados especialmente para este fin, dentro de las cuales están los siguientes:

* XEP-0322[[21]](#footnote-21): Efficient XML Interchange (EXI) Format
* XEP-0323[[22]](#footnote-22): Internet of Things – Sensor Data
* XEP-0324[[23]](#footnote-23): Internet of Things – Provisioning
* XEP-0325[[24]](#footnote-24): Internet of Things – Control
* XEP-0326[[25]](#footnote-25): Internet of Things – Concentrators

Cuando mencionamos XMPP IoT queremos decir que un JID es asignado a una «Things» o a una «estación de Things». Por ende, se tiene la opción de consultar datos, tal como si fuese un cliente más XMPP, por ende, el estará escuchando con el socket abierto, de modo, que pueda responder solicitudes que se vayan a realizar a este.

#### 2.1.1.7. Seguridad

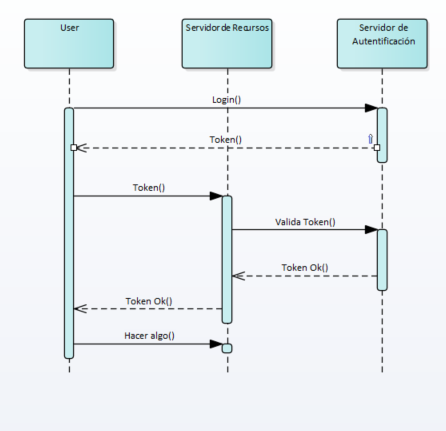
El principal problema de XMPP cuando nos referimos en el contexto del Internet de las Cosas, es el siguiente: debido a que este protocolo fue concebido pensando originalmente en Mensajería Instantánea, el XEP que se encarga de la creación automática de identidades, es decir, el XEP-0077: In-Band Registration, el único filtro para la creación de cuentas son los CAPTCHAs, los cuales se especifican en el XEP-0158: CAPTCHA Forms. Sin embargo, esta XEP está pensado para velar contra bots maliciosos, no pensado en usuarios maliciosos los cuales para burlar el bot escribiendo el código de verificación del CAPTCHA. Por ende, creando la identidad de un algún «Thing» y así burlar la seguridad del sistema.

Con la información que contamos en la actualidad, sabemos que los mecanismos antibots que usan CAPTCHAs pueden se vulnerados, mediante visión computacional. [falta referencia]

Para solucionar este problema uno de los Stakeholders involucrados en este proyecto, Peter Waher, creó el «XEP-0348: Signing Forms», es decir, la especificación del protocolo de extensión, el cual, utiliza «OAuth 1.0»[[26]](#footnote-26), para validad que lo agentes creadores de cuentas, sean robots o humanos, tengan autorización para crear identidades en el servidor XMPP.

El funcionamiento de OAuth 1.0 es explicado mediante la Figura 5.

Figura 5: Diagrama de secuencia de OAuth 1.0.



De este modo, con la «XEP-0348[[27]](#footnote-27): Signing Forms» podemos solucionar la vulnerabilidad existente en el contexto de IoT de la «XEP-0077[[28]](#footnote-28): In-Band Registration».

## 2.2. Problema

* El XEP-0077: In-Band Registration abre una brecha de seguridad, al permitir que ninguno o todos los clientes creen identidades en la red XMPP-IoT.
* En la actualidad, si se activa el mecanismo de In-Band Registration en conjunto con el XEP-0158: CAPTCHA Forms, no se cierra la brecha de seguridad, ya que hoy mediante visión computacional, se pueden resolver los CAPTCHAS.
* Un usuario o bot malicioso, fácilmente podría crear cantidad virtualmente ilimitadas en un servidor XMPP orientado al IoT. Vulnerando así la seguridad de toda la red XMPP-IoT.

### 2.2.1. Objetivo General

* **OG**: Solucionar vulnerabilidad de seguridad existente en la XEP-0077 dentro del contexto XMPP-IoT.

### 2.2.2. Objetivos Específicos

* **OE-01**: Reducir a cero los intentos de penetración exitosos de usuarios maliciosos a la red XMPP, que utilizan la vulnerabilidad de la XEP-0077.
* **OE-02**: Reducir a 0% la creación de identidades en un servidor XMPP-IoT por parte de usuarios, dispositivos o bots no autorizados, por medio de la vulnerabilidad en el registro en banda.
* **OE-03**: Permitir el registro de nuevas identidades sólo a los clientes autorizados.

### 2.2.3. Restricciones

* Documentación del código debe ser en inglés.
* Licencia de plugin y extensiones deben ser de licencia Apache 2.0, MIT, o similar.

### 2.2.4. Requerimientos No Funcionales

En la Tabla 1, podemos apreciar los Requerimientos No Funcionales.

Tabla 1: Requerimientos No Funcionales

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador | Título | Importancia | Complejidad | Atributo de Calidad Asociado |
| RNF01 | El código debe suficientemente claro para ser entendido y modificable por terceros. | Alta | Media | Mantenibilidad |
| RNF02 | El código fuente debe ser escalable. | Alta | Alta | Mantenibilidad |
| RNF03 | El software debe extender la seguridad del servidor. | Alta | Alta | Seguridad |

## 2.3. Alternativas de Solución

### 2.3.1. Alternativa 1: Implementar VPN

Esta consiste en cerrar a internet los puertos que utiliza XMPP, e implementar una capa de seguridad VPN para hacer uso del servidor, de este modo, disminuir considerablemente el riego de sufrir ataques maliciosos.

### 2.3.2. Alternativa 2: CREAR XEP, PLUGIN Y LIBRERÍA QUE IMPLEMENTEN OAUTH

Esta alternativa consiste en crear especificación de un Protocolo de Extension XMPP, que la Concilio de la XFS lo apruebe para que pase a «Draft» y posteriormente a «Experimental». Mientras se realiza este proceso, desarrollar un plugin para el servidor «OpenFire» y una Extension para la Librería «Smack» basados en el Proto-XEP.

### 2.3.3. Alternativa 3: modificar Servidor XMPP y librería XMPP para que implementen la Xep-0348 la cual tiene el mecanismo Oauth

Esta alternativa, consiste es utilizar como guía y referencia el XEP-0348: Signing Forms. El cual implementa OAuth 1.0 para delegar autorización para la creación de identidades en la red XMPP-IoT. Todo esto encriptado por medio SSL/TLS.

### 2.3.4. Comparativa entre Alternativas

A continuación, se presentan las alternativas para dar solución a la problemática planteada en las secciones anteriores. La Tabla 2 presenta las ventajas y desventajas de las alternativas previamente nombradas.

Tabla 2: Comparación de alternativas de solución.

|  |  |  |
| --- | --- | --- |
|  | Ventajas | Desventajas |
| Alternativa 1 | Escasa codificación. | Implementar en todos los clientes XMPP, una capa de seguridad mediante un cliente VPN.  Existen vulnerabilidades no corregidas. |
| Alternativa 2 | Tecnología modera.  Seguridad en el intercambio de información.  Permite regular el acceso a funcionalidades del software. | La creación y estandarización de un XEP toma demasiado tiempo, aproximadamente un año y medio.  Desconocimiento de parte del equipo de programadores desarrollar utilizando XMPP. |
| Alternativa 3 | Seguridad en el intercambio de información.  Permite regular el acceso a funcionalidades del software. | La XEP-0348 está diseñada y está se encuentra en estado «Experimental».  La tecnología usada es relativamente antigua, mas no desactualizada.  Desconocimiento de parte del equipo de programadores desarrollar utilizando XMPP. |

### 2.3.5. Elección de Alternativa

Para realizar a elección de una alternativa de solución, se tomó en consideración la problemática, los objetivos, los requerimientos no funcionales de proyecto, más las restricciones del interesado, es decir Peter Waher.

Tomando en consideración todos estos factores, se llegó a la conclusión que la mejor alternativa es la tres, ya que cumple con realizar y satisfacer lo solicitado en el tiempo requerido.

## 2.4. Solución Propuesta

Lo necesario para solucionar la vulnerabilidad de seguridad existente en la «XEP-0077: In-Band Registration», es utilizar la «XEP-034: Signing Forms», este para este fin es necesario:

1. Modificar el servidor XMPP «OpenFire».
2. Crear extensión para librería XMPP «SMACK».

De este modo, se realizarán las modificaciones y agregaciones al servidor XMPP Openfire, el cual, quedará habilitado para atender las nuevas de consultas que se enviarán desde un cliente, para la creación de identidades utilizando los mecanismos de registro en banda en conjunto el firmado de formularios.

### 2.4.1. Factibilidad Técnica

#### 2.4.1.1. Lenguajes de programación y Herramientas

El servidor XMPP que utilizaremos será «OpenFire», el cual funciona sobre la máquina virtual de Java. Cuenta con una consola de administración web. En donde es posible gestionar cuentas, permisos, grupos, salas de conferencias, bloqueos y lo que es relevante en el presente proyecto, habilitar o deshabilitar características, ej.: In-Band Registration.

Las modificaciones del servidor «OpenFire» y la extensión para la librería «Smack» serán codificadas, programando en el lenguaje de programación Java versión 1.8, ya que el servidor XMPP que se seleccionó opera sobre esta versión de la máquina virtual de Java.  
La IDE que utilizaremos para codificar en Java será «IntelliJ IDEA» versión 2017.2, una suite completa que permite integración perfecta con la máquina virtual de Java con gestores de dependencias java como lo son Maven y Gradle.

Se utilizará el lenguaje de programación C#, para desarrollar un cliente XMPP con la finalidad de mitigar el riesgo técnico relacionado con el nivel de conocimiento de XMPP. Para desarrollar este cliente se utilizará el entorno de desarrollo gratuito de Microsoft, es decir, Visual Studio 2015 Community Edition.

#### 2.4.1.2. Hardware

Para la creación de toda la solución debemos tener en cuenta lo siguientes requerimientos de hardware.

Entorno de desarrollo

* Laptop
  + Procesador i5 7ma GEN.
  + 8 GB RAM
  + 20 GB de almacenamiento disponible
* Servidor
  + SO: Debian Server / Ubuntu Server / CentOS / RedHat / Fedora Server
  + 4 GB RAM
  + 4 GB de almacenamiento disponible
  + MySQL Server / HSQL / Postgres SQL / MS SQL SERVER
  + Registro DNS STUN y TURN correctamente configurados.
  + IP Publica
  + Configuración TLS valida
  + Puertos abiertos: 5222, 5223, 9090, 9091, 7070, 7443, 7777
* Thing
  + Raspberry Pi 3
  + Tarjeta SD.
  + Protoboard.
  + Sensor de temperatura DFRobot LM35.
  + Sensor Humedad DFRobot humdity sensor.
  + Arduino UNO r3
* Transversal
  + Conexión a Internet.

### 2.4.2. Alcances de solución

Con el propósito de cumplir con los objetivos propuestos, el proyecto brindará una solución al problema que:

* Proporcionará la interfaz para utilizar el mecanismo descrito en la XEP-0348, tanto en cliente como en el servidor.
* Filtrará que clientes puede pueden crear identidades.
* Sólo dispositivos autorizados puedan crear nuevas identidades en la red XMPP-IoT.
* Se modificará el servidor Openfire generando una interfaz empotrada en la consola web servidor.

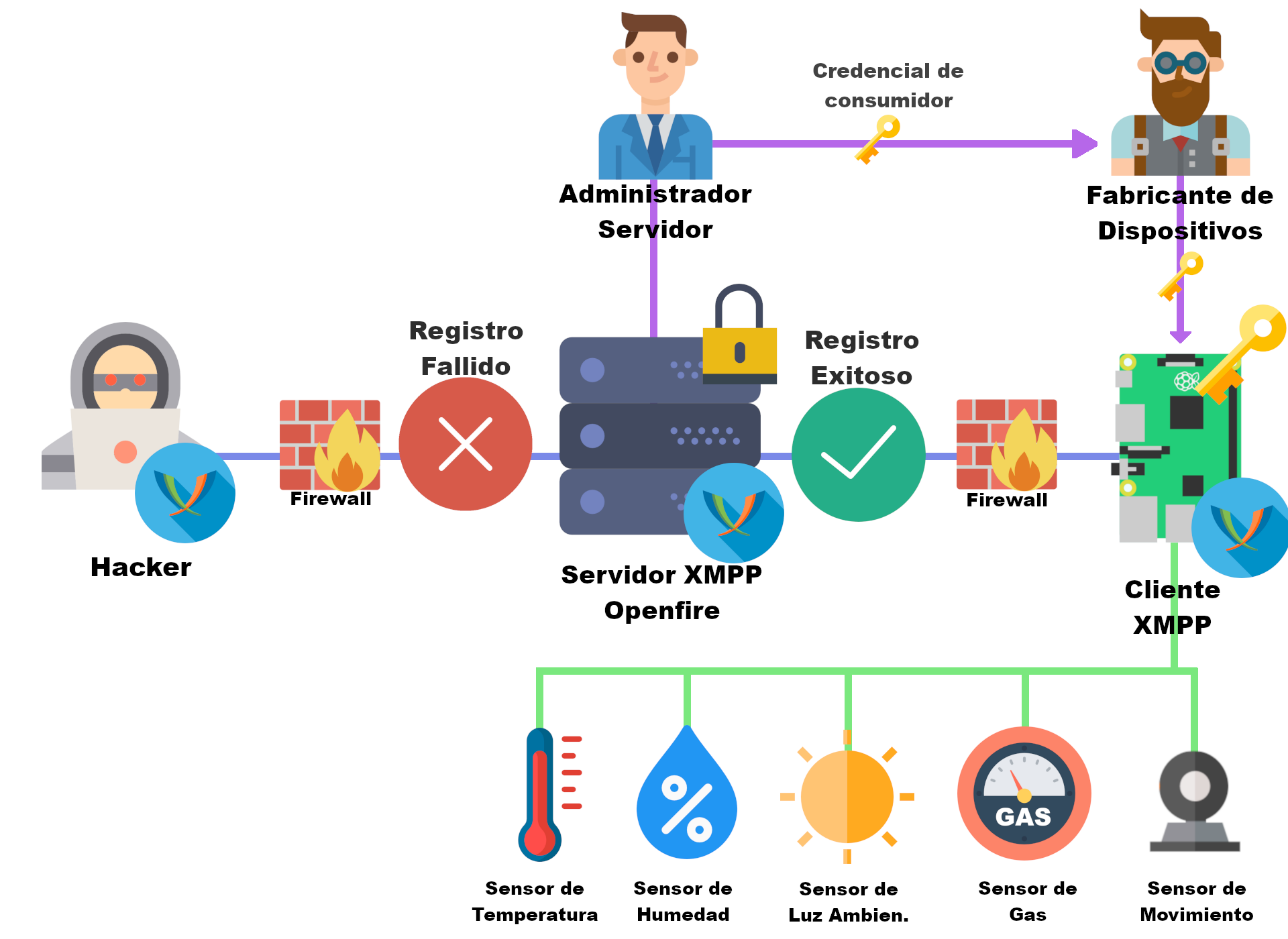
### 2.4.3. Limitaciones de la solución

* Sólo se modificará el servidor «OpenFire» y una extensión de la librería «Smack».

### 2.4.4. Esquema de Solución

En la Figura 6, se logra visualizar, el esquema de la solución planteado en nuestra propuesta. Es decir, que el administrador crea «credenciales de consumidor», donde le asigna una cantidad permitida de registro de nuevas identidades a dichas credenciales. Luego, esta es entregada al interesado en crear identidades en la red XMPP-IoT, en este caso «Fabricante de Dispositivos» quien la instala en los dispositivos, que serán clientes del servidor XMPP. De este modo, exclusivamente, los clientes que posean estas credenciales podrán registrar identidades en la red, y así el «Hacker», ni los bot maliciosos podrán realizar registros en la red XMPP-IoT, como antes podían hacerlos.

Figura 6: Esquema de solución.



### 2.4.5. Factores de éxito

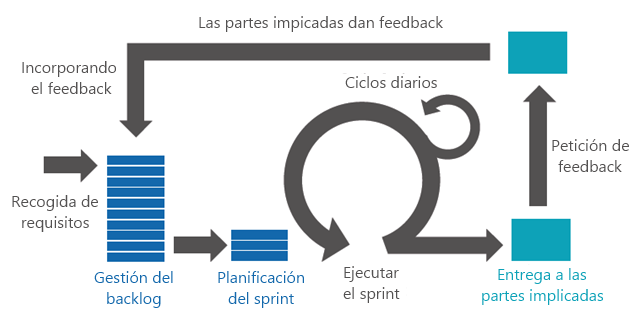
Para que el proyecto se considere exitoso se deberá cumplir con los requerimientos no funcionales, además de que se cumpla el objetivo general y los específicos.

# 3. MATERIALES Y MÉTODOS

## 3.1. METODOLOGÍA DE DESARROLLO

La metodología escogida para el desarrollo del proyecto es SCRUM, debido a que ésta permite al equipo trabajar de manera colaborativa para la obtención del mejor resultado posible. Los procesos SCRUM están enfocados a que el equipo de desarrollo pueda ir reaccionando ante las observaciones y posibles requisitos cambiantes de los interesados, gracias a que se debe realizar entregas cortas, idealmente cada cuatro semanas. Este tiempo de desarrollo es llamado «Sprint» o «Iteración». Para el proyecto se utilizarán Sprints de tres semanas. En la Figura 7 se muestra el proceso de desarrollo que utiliza SCRUM.

Figura 7: Metodología de desarrollo de proyecto - SCRUM.



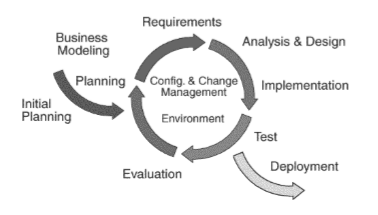
Se generará un Product Backlog (Pila del producto), el cual contendrá todas las historias de Usuario (HU), identificadas y priorizadas por el Product Owner. En este documento se mantienen los requerimientos de sistema, con los cuales se irán realizando los Sprint. Para cada Sprint se tendrá un Sprint Backlog, el cual contendrá las HU a desarrollar en cada iteración, y cada HU contendrá las tareas asociadas.

Se utilizará la herramienta GitHub, en la cual usaremos «issues» para seguir defectos, mejoras y tareas.

Al inicio y final de cada Sprint se agendarán reuniones documentadas con el Product Owner para decidir que HU se desarrollarán, para que éste pruebe las nuevas funcionalidades y se converse sobre cómo se llevó a cabo el Sprint. Adicionalmente se deberá aprobar el entregable, o en su defecto rechazar el documento.

En cada Sprint se realizará el ciclo de la Imagen 8, el que comprende las fases de: Planificación, Análisis y Diseño, Implementación, Pruebas y Despliegue. Al término de cada Sprint se realizará lo que se conoce por «Sprint Retrospective» en se analizar cuáles fueron las errores y aciertos del presente Ciclo, para corregir los errores en los siguientes Sprint.

Figura 8: Metodología de desarrollo.



Dentro de las razones por las cuales hemos elegidos las metodologías antes mencionadas, están:

* Los pueden ser cambiables, por lo que se requiere una metodología que permita y regule los cambios a la planificación.
* La necesidad de constante retroalimentación, para así validar el producto directamente con el Product Owner.
* Tener un producto con el cual el cliente se sienta satisfecho.

### 3.1.1. ROLES Y RESPONSABILIDADES

En la Tabla 3 se grafican los roles y responsabilidades de SCRUM de los involucrados.

Tabla 3: Roles en metodología SCRUM.

|  |  |  |
| --- | --- | --- |
| Rol | Encargado | Responsabilidades |
| Scrum Master | Marcelo Aros | Asegurar cumplimiento de reglas y procesos |
| Product Ower | Romina Torres | Generación de historias de usuario  Validación de Sprints |
| Team | Marcelo Aros | Desarrollo del proyecto |
| Stakeholders | Peter Waher | Supervisión y orientación |

En la Tabla 4 se presentan los roles y responsabilidades en contexto de la metodología de programación.

Tabla 4: Roles y responsabilidades de metodología SCRUM.

|  |  |  |
| --- | --- | --- |
| Rol | Encargado | Responsabilidades |
| Director de Proyecto | Marcelo Aros | Coordinar Reuniones con cliente  Realizar seguimiento del progreso |
| Cliente | Romina Torres | Define especificaciones  Realiza pruebas de aceptación |
| Programador | Marcelo Aros | Codificación.  Realización de pruebas unitarias. |
| Encargado de pruebas | Marcelo Aros | Orientar al cliente en las pruebas de aceptación  Ejecutar pruebas funcionales y publicar resultados. |

## 3.2. AMBIENTE DE DESARROLLO

### 3.2.1. Gestión de la Configuración

#### 3.2.1.1. Control de Cambios

Para el control de cambios se ha decidido utilizar el software GitHub, herramienta web para la gestión de proyectos mediante la utilización de tickets o incidencias. Este se encuentra en la dirección https://github.com/MarceloAros/Openfire/issues GitHub se utilizará para realizar seguimiento de defectos, mejoras y tareas a realizar.

#### 3.2.1.2. Control de versiones

De igual modo que para el control de cambios se decidió utilizar GitHub herramienta destinada a gestionar todas los repositorios y versiones de los documentos y archivos del proyecto. El cual es gratuito, además de ser abierto para cualquier persona, las direcciones son las siguientes:

* <https://github.com/MarceloAros/Openfire>
* <https://github.com/MarceloAros/Smack>
* <https://github.com/MarceloAros/XEP0348-test>
* <https://github.com/MarceloAros/OF0348-Docs>

Utilizaremos la estructura propia de Git, la cual describimos en la Tabla 5 que está a continuación.

Tabla 5: Estructura de control de versiones Git.

|  |  |
| --- | --- |
|  | Descripción |
| Master | Código fuente «principal» |
| Branch | Entorno de trabajo aislado del «Master» |
| Releases | Etiquetas de liberaciones |

Para las líneas base y cada liberación del producto se definieron las siguientes políticas de liberación:

* Las pruebas de aceptación deben haber sido 100% aprobadas.
* Las pruebas unitarias deben pasar en un 100%.
* Los ítems de configuración contenidos en cada línea base serán los siguientes:
  + Ejecutable
  + Código Fuente
  + Notas de Instalación
  + Notas de Liberación
  + Manual de Usuario
  + Memoria de Proyecto
  + Sprint Backlog
  + Product Backlog

#### 3.2.1.3. Entregables

Tabla 6: Detalles de liberaciones y entregables.

|  |  |  |
| --- | --- | --- |
| Entrega | Versión de Producto(s) | Fecha |
| Sprint #0-A | MitigationApp 0.1 | 30 de abril de 2017 |
| Sprint #0-B | MitigationApp 0.2 | 30 de mayo de 2017 |
| Sprint #0-C | MitigationApp 0.3 | 30 de junio de 2017 |
| Sprint #1 | Openfire 4.3.1-beta1 | 13 de abril de 2018 |
| Sprint #2 | Openfire 4.3.1-beta2 | 27 de abril de 2018 |
| Sprint #3 | Openfire 4.3.1.-beta3 | 14 de mayo de 2018 |
| Sprint #4 | Openfire 4.3.1.-beta3.2 SMACK 4.3.0-alpha.1 | 29 de mayo de 2018 |
| Sprint #5 | Openfire 4.3.1.-beta3.1 SMACK 4.3.0-alpha.1.2 | 16 de junio de 2018 |

### 3.2.2. Lenguajes de Programación

Para el desarrollo del código fuente se utilizará Java para modificación del servidor XMPP «OpenFire» y para librería «Smack», como también C# para la creación de un cliente simple, Processing (Arduino) para la conexión de sensores de temperatura y humedad por medio de I2C. La gran cantidad de lenguajes de programación es debido a que necesitamos interconectar diversos dispositivos.

Las librerías más importantes que estamos utilizando se encuentran:

* Java
  + Igniterealtime SMACK
  + Firmata4j
  + Xpp3
* C#
  + Waher.Networking
  + Waher.Things
  + Waher.Runtime
  + Waher.Content
  + Waher.Events

### 3.2.3. Herramientas de Desarrollo

A continuación, se encuentra la Tabla 7, la cual grafica cuales son nuestras herramientas de desarrollo con su funcionalidad.

Tabla 7: Herramientas de desarrollo.

|  |  |
| --- | --- |
| Nombre de herramienta | Descripción |
| Intellij IDEA | IDE Java |
| Android Studio | IDE Android y Java |
| Gradle | Gestor de dependencias Java |
| Maven | Gestor de dependencias Java |
| Visual Studio | IDE de desarrollo y testing para C# y VB |
| Arduino IDE | IDE de desarrollo para programar en Process (Arudino). |
| Bitvise SSH | Cliente SSH y SFT, se utiliza para conectarse a Raspberry Pi e intercambiar archivos. |
| GitHub | Cliente Git para GitHub |

## 3.3. Ambiente de Producción

El producto será utilizado en sistemas con las siguientes características.

* Raspberry PI 3
  + SO: Rapsbian o Windows Core Edition
    - JRE
  + Things
* Servidor
  + SO: Debian Server / Ubuntu Server / CentOS / RedHat / Fedora Server
  + Dominio configurado.
  + CPU: igual o superior a Intel i3-3220
  + RAM: 2 GB
  + Almacenamiento: 10 GB disponibles
  + IP publica
  + Registros DNS SVR configurados
* Cliente (con librería implementada)
  + SO: Windows, Linux o OSX
    - JRE
  + CPU: igual o superior a Intel Celeron G1610
  + RAM: 1GB
  + Almacenamiento: 500 MB disponibles.

## 3.4. Ambientes de Pruebas

### 3.4.1. Elementos deL ambiente

* Arduino UNO Rev3
  + Sensor de temperatura
  + Sensor de humedad
* Raspberry Pi Modelo B
  + Servidor XMPP OpenFire 4.0.3
    - XEP-0348 implementado.
    - XEP-0077 implementado.
* Computador Laptop
  + SO: Windows
  + Procesador Intel i5-6400
  + Memoria RAM: 8 GB
  + Almacenamiento 1 TB
* Servidor
  + SO: Ubuntu server
  + Core: 1
  + RAM: 1 Gb
  + Almacenamiento: 64 GB SSD
  + Dominio: binarylamp.cl

### 3.4.2. Gestion de las pruebas.

Se deben generar pruebas continuas, frecuentemente repetidas y automatizadas para asegurar que el código funcione y haga lo que se supone debe hacer. Por esta razón las pruebas se programarán «Junit4», una librería Java para realizar pruebas unitarias, que ofrece funcionalidades para poder realizar y automatizar las pruebas de manera simple.

Para cada Sprint se realizará un nuevo plan de pruebas, con el objetivo de cumplir con todo lo requerido por las tareas de dicho Sprint. Se debe contar con pruebas unitarias, de aceptación y de sistema, como se especifica en las siguientes subsecciones.

Según las metodologías escogidas, las pruebas unitarias se realizarán al finalizar el desarrollo de cada módulo. Las pruebas de integración y regresión se realizarán al agregar un componente nuevo al sistema, mientras que las pruebas de sistema se realizarán después de la realización de todas las HU de la iteración. Finalmente, las pruebas de aceptación las realizará el Product Owner en la reunión de Sprint Review.

Las pruebas unitarias programadas se reutilizarán para efectuar pruebas de integración, consistentes en la automatización de la utilización de las pruebas unitarias cada vez que se agregue un componente al programa.

Se utilizará el método de bottom-up, el cual consiste en empezar las pruebas de los componentes de más bajo nivel, para luego ir subiendo y repitiendo el proceso hasta llegar a los componentes más complejos y finalizar las pruebas de integración.

Cada vez que se hagan cambios al sistema se deberá ejecutar pruebas de regresión para asegurar que dichos cambios no han afectado al funcionamiento general del software.

Para asegurar que nuevas funcionalidades se comportan de manera correcta al ser unidas a los demás componentes se debe realizar pruebas de regresión, en las cuales se reutilizan nuevamente las pruebas unitarias para asegurarse de que los componentes antiguos no están fallando debido a las nuevas funciones.

Las pruebas de sistema se realizarán una vez estén funcionando las partes principales del software, de manera de poder determinar si se van cumpliendo los objetivos propuestos.

Por último, el cliente o el director de proyecto realizará las pruebas de aceptación para concluir cada Sprint, en el cual probará que los criterios de aceptación para las historias de usuario desarrolladas se cumplieron, si no pasan se refactorizará, hasta que se cumplan las pruebas de aceptación.

## 3.5. Plan de Proyecto

### 3.5.1. Gestión de las Comunicaciones

En la Tabla 8 podemos visualizar el contenido de cada una, los tipos de comunicación, los interesados involucrados, y el formato correspondiente.

Tabla 8: Gestión de las comunicaciones - tipos de comunicación.

|  |  |  |  |
| --- | --- | --- | --- |
| Información | Interesados | Formato | Contenido |
| Sprint Planning | * Product Owner, * Scrum Master * Team | Reunión en oficina del PO, vestimenta casual. | Se discute sobre historias de usuario a abordar en el siguiente Sprint según priorizaciones del Product Owner. También se discuten los criterios de aceptación |
| Daily Sprint Meeting | * Scrum Master * Team | Reunión diaria ya sea por algún método virtual o presencial. Duración de 15 minutos máximo. | El equipo se alinea para trabajar de manera coordinada. Se discute lo que se realizó el día anterior y lo que se realizara el presente día. |
| Demostración | * Product Owner * Scrum Master * Team | Reunión semiformal en oficina del PO, vestimenta semi-formal. | Se presentan las historias de usuario realizadas a través de la presentación del producto. |
| Presentación Hitos | * Product Owner * Scrum Master * Team * Comisión | Presentación en sala 216, Universidad Andrés Bello, vestimenta semi-formal. | Se realiza presentación de lo que se llevó a cabo durante el Sprint. |
| Cambios en Documentos | * Scrum Master * Team | Issue GitHub | Cambios que se realizaron y razón. |
| Tareas Por Realizar | * Scrum Master * Team | Issue GitHub | Tareas por realizar, con fecha tope y descripción. |

#### 3.5.1.2. Gestión de la Resolución de Conflictos

Para los conflictos que se presenten durante la ejecución del proyecto se utilizarán las siguientes estrategias estipuladas en la Guía del PMBOK 4ª Edición:

1. Apartarse/Eludir: Retirarse de una situación de conflicto real o potencial.
2. Suavizar/Reconciliar: Hacer hincapié en los puntos de acuerdo más que en las diferencias.
3. Consentir: Buscar soluciones que aporten un cierto grado de satisfacción a todas las partes.
4. Forzar: Imponer su propio punto de vista a costa de los demás; ofrece únicamente soluciones de tipo ganar-perder.
5. Colaborar: Incorporar múltiples puntos de vista y visiones a partir de perspectivas diversas; conduce al consenso y al compromiso.
6. Confrontar/Resolver problemas: Tratar un conflicto como un problema que debe resolverse mediante el examen.

En la Tabla 9 se presentan las estrategias a utilizar para cada posible conflicto detectado.

Tabla 9: Resolución de conflictos.

|  |  |
| --- | --- |
| Conflicto | Estrategia |
| Diferencia en disponibilidad de horario para reuniones. | Colaborar para llegar a un acuerdo de horario. |
| Product Owner intenta crear requisitos no factibles. | Colaborar trabajando en conjunto a fin de llegar a un acuerdo. |

### 3.5.2 Gestión de Riesgos

Para la Gestión de los Riesgos, éstos se separarán en cuatro tipos:

1. Riesgos Técnicos
2. Riesgos Externos
3. Riesgos Organizacionales
4. Riesgos de Dirección de Proyecto

Cada uno de estos riegos potenciales, tendrá un valor de probabilidad, el cual representa la probabilidad de que un hecho ocurra y un valor de impacto, que hace referencia a incidencia de este sobre el proyecto.

Tabla 10: Asignación de riesgos.

|  |  |
| --- | --- |
| Titulo | Valor numérico |
| Muy Bajo | 1 |
| Bajo | 2 |
| Medio | 3 |
| Alto | 4 |
| Muy Alto | 5 |

Tabla 11: Matriz de riesgos.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Probabilidad | Muy Alto | 5 | 10 | 15 | 20 | 25 |
| Alto | 4 | 8 | 12 | 16 | 20 |
| Medio | 3 | 6 | 9 | 12 | 15 |
| Bajo | 2 | 4 | 6 | 8 | 10 |
| Muy bajo | 1 | 2 | 3 | 4 | 5 |
|  | | Muy Bajo | Bajo | Medio | Alto | Muy Alto |
| Impacto | | | | |

#### 3.5.2.1. Identificación de Riesgos

En la Tabla 12 podemos ver los riesgos identificados del proyecto.  
El Id. de cada riesgo se encuentra conformado por el código de proyecto, es decir, «OP0348-» seguido de la palabra «RISK-» acompañado del número identificador del riesgo.

Tabla 12: Identificación de riesgos.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Tipo | Riesgos | Probabilidad | Impacto | Valor de riesgo |
| OF0348-RISK-01 | Técnico | El cliente decide agregar, modificar o eliminar alguna historia de Usuario durante el trascurso del Sprint. | Alto | Alto | 16 |
| OF0348-RISK-02 | Técnico | Dificultad del equipo de desarrollo para aprender el uso de XMPP. | Alta | Muy Alto | 20 |
| OF0348-RISK-03 | Técnico | Dificultad del equipo de desarrollo para aprender a usar librerías XMPP | Alta | Alto | 16 |
| OF0348-RISK-04 | Externo | El cliente no satisfecho con el resultado del Sprint. | Media | Alto | 12 |
| OF0348-RISK-05 | Dirección de proyecto | Error en la estimación de tiempo de actividades. | Media | Alto | 12 |
| OF0348-RISK-06 | Externo | Desperfectos en los dispositivos electrónicos. | Bajo | Medio | 6 |

#### 3.5.2.2. Estrategias de Mitigación

En la Tabla 13 se muestra la mitigación correspondiente a cada riesgo identificados por su respectivo ID.

Tabla 13: Estrategias de mitigación de riesgos.

|  |  |  |
| --- | --- | --- |
| ID | Riesgo | Mitigación |
| OF0348-RISK-01 | El cliente decide agregar, modificar o eliminar alguna historia de Usuario durante el trascurso del Sprint. | En la «Sprint Planning» se hace hincapié al Product Owner que cualquier modificación a lo que está en la pila del Sprint Backlog durante el Sprint se deberá ver en el Sprint Review, es decir, dejarlo para próximos Sprints. |
| OF0348-RISK-02 | Dificultad del equipo de desarrollo para aprender el uso de XMPP. | Se realizará un «Sprint #0» en el cual se desarrollará un prototipo con las funcionalidades de XMPP. |
| OF0348-RISK-03 | Dificultad del equipo de desarrollo para aprender a usar librerías XMPP | Se realizará un «Sprint #0» en el cual se desarrollará un prototipo con las funcionalidades de las librerías de Clayster y/o WaherData. |
| OF0348-RISK-04 | El cliente no satisfecho con el resultado del Sprint. | Se diseñarán criterios y pruebas de aceptación en el inicio de cada sprint, los cuales él cliente deberá aceptar. |
| OF0348-RISK-05 | Error en la estimación de tiempo de actividades. | Durante la «Sprint Planning» se realizará un análisis de las horas destinadas al análisis, diseño/desarrollo y pruebas. |
| OF0348-RISK-06 | Desperfectos en los dispositivos electrónicos. | Se realizarán conexiones y manipulaciones de la electrónica con los dispositivos desconectados de la toma de electricidad, además eliminando la estática del sujeto que intervenga los dispositivos. |

Tabla 14: Estrategias de Mitigación

#### 3.5.2.3. Estrategias de Contingencia

En la Tabla 15 se muestran la estrategia de contingencia para cada riesgo.

Tabla 15: Estrategias de contingencia para los riesgos.

|  |  |  |
| --- | --- | --- |
| ID | Riesgo | Contingencia |
| OF0348-RISK-01 | El cliente decide agregar, modificar o eliminar alguna historia de Usuario durante el trascurso del Sprint. | Se gestiona la Historia de Usuario requerida siempre y cuando esta no tenga un impacto mayor a 1000. Como resultado, se debe volver a analizar los tiempos requeridos para la finalización del Sprint. Si fuera necesario, se deberá modificar dicha planificación. |
| OF0348-RISK-02 | Dificultad del equipo de desarrollo para aprender el uso de XMPP. | Coordinar reuniones con desarrollador senior con conocimientos avanzados en XMPP, que además es un «Stakeholder» del proyecto, Peter Waher, para solucionar dudas e inconvenientes, de esta manera evitar el estancamiento en el proyecto. |
| OF0348-RISK-03 | Dificultad del equipo de desarrollo para aprender a usar librerías XMPP | Se coordinará reuniones con desarrollador senior quien es el creador de dichas librerías, Peter Waher, para solucionar dudas y así evitar el estancamiento en el proyecto. |
| OF0348-RISK-04 | El cliente no satisfecho con el resultado del Sprint. | Se examinará la posibilidad de integrar las Historias de Usuario con las que no está satisfecho en un siguiente Sprint, con las modificaciones acorde a lo requiera el cliente. En caso de no ser posible, se le informará al cliente. En caso de un Sprint, se seguirá con el siguiente Sprint. En caso del producto final, se entregará en ese estado. |
| OF0348-RISK-05 | Error en la estimación de tiempo de actividades. | Se le inyectará mayor cantidad de HH al Sprint, de modo que se pueda solucionar este error. En su defecto, si no pudiese, se modificará la planificación de acuerdo al atraso existente. |
| OF0348-RISK-06 | Desperfectos en los dispositivos electrónicos. | Se utilizarán los repuestos, si estos fallan, se adaptará un Arduino para que cubra esta funcionalidad, esto es lo que respecta a sensores. Si falla cualquier otro dispositivo, se deberá comprar el repuesto de este. |

### 3.5.3. Gestión de Requerimientos

Los requerimientos en el proyecto serán acordados entre el «Product Owner» y el «Scrum master» por medio de la gestión de Historias de Usuario. Asimismo, el Product Backlog se desarrollará con todas las Historias de Usuario identificadas, y se utilizará «Planning Poker» para la estimación de esfuerzo y priorización.

Al momento de realizar la reunión Sprint Planning el Product Owner presenta las Historias de usuario asociadas al Product Backlog, de acuerdo con la prioridad asociada a esta, así el equipo determinará cuales se comprometerán a desarrollar en la duración del Sprint.

Durante el trascurso del proyecto se podrán gestionar las Historias de Usuario, es decir, agregar, modificar y eliminar alguna de estas. Esto sólo se puede realizar en la finalización del Sprint, de otro modo, estos cambios afectarán a la planificación Sprint en curso, generando una incertidumbre en el tiempo requerido para la conclusión de este.

## Proceso de Sprints

Luego de considerar toda la información expuesta en este capítulo, el proceso a seguir en cada Sprint se explica con detalle a continuación.

1. Se realiza la reunión «Sprint Planning», en la cual se encuentran presentes el Product Owner y los integrantes del equipo de desarrollo. Se procede a establecer el objetivo del Sprint, se otorga una prioridad a cada Historia de Usuario presentes en el Product Backlog. Se acuerda cuales Historias de Usuarios serán desarrolladas, además se establecen Criterios de aceptación para estas, los cuales serán la base de las pruebas de aceptación que se realizarán al término del Sprint.
2. Se genera el Sprint Backlog, en el cual se especifican todas las tareas a realizar durante el Sprint, además a cada una se una estimación de tiempo.
3. Durante que el Sprint se está ejecutando, se realiza un desarrollo psuedo-cascada. Comenzando por Analisis, seguido de Diseño, Codificación y Pruebas.
4. Eventualmente si el Product Owner solicita algún cambio, este se agrega como ticket a GitHub, el cual debe ser priorizado en el Product Backlog para el siguiente Sprint.
5. Los defectos, se agregan una vez encontrados como un ticket de «defecto» a GitHub, el cual deberá ser solucionado dentro del trascurso del Sprint.
6. Al momento de finalizar el Sprint se realiza la Reunión Sprint Retrospective, en la cual participa el Product Owner junto con el equipo de desarrollo. Se acepta el Sprint luego de haber aprobado las pruebas de aceptación. De no ser así se procederá a hacer una replanificación.

# 4. RESULTADOS Y DISCUSIÓN

El presente capítulo presenta las Historias de Usuario obtenidas, sus criterios de aceptación, la trazabilidad entre éstas, los requerimientos no funcionales y los objetivos. Además, se presentarán todos los resultados obtenidos durante el desarrollo del proyecto y se discutirá sobre el cumplimiento de los objetivos y requerimientos no funcionales.

## 4.1. Historias de Usuario

En la Tabla 16 se aprecian las historias de usuario generadas en proyecto OF0348.

Tabla 16: Historias de usuario.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Enunciado de la historia | | |
| Identificador (ID) de la historia | Rol | Característica / Funcionalidad | Razón / Resultado |
| OF0348-HU-01 | Como administrador de «servidor XMPP» | necesito agregar credenciales de consumidor | para autorizar el registro automatizado de identidades. |
|
|
| OF0348-HU-02 | Como administrador de «servidor XMPP» | necesito modificar credenciales de consumidor | para cambiar la cantidad de identidades asociadas a una credencial. |
|
| OF0348-HU-03 | Como administrador de «servidor XMPP» | necesito visualizar las «credenciales de consumidor», las identidades permitidas y las identidades utilizadas | para identificar el uso que cada consumidor hace de la red XMPP. |
|
| OF0348-HU-04 | Como administrador de «servidor XMPP» | necesito visualizar cuantas y que identidades han sido creadas con una «credencial de consumidor» | para llevar un registro de las identidades creadas asociadas a una credencial de consumidor». |
|
| OF0348-HU-05 | Como administrador de «servidor XMPP» | necesito poder eliminar una identidad creada por una «credencial de consumidor» | para tener la capacidad de eliminar un cliente malicioso que está utilizando cierta identidad. |
|
|
| OF0348-HU-06 | Como administrador de «servidor XMPP» | necesito permitir el registro identidades (cuentas) únicamente que incluyan «Credenciales de Consumidor» validas en el formulario de registro | a fin de evitar el ingreso a bots, usuarios maliciosos y no autorizados, mediante la vulnerabilidad de la XEP-0077. |
|
|
| OF0348-HU-07 | Como «Thing» | debo poder registrarme automáticamente en «Openfire» usando credenciales de consumidor | para ser mejorar la seguridad de la comunicación y el servidor. |
|
|
| OF0348-HU-08 | Como usuario de red XMPP-IoT | quiero poder registrar una identidad e ingresar a la red | Para leer o escribir valores de mis dispositivos (Things) |
|
|

Por cada una de las historias de usuario generadas, también se establecieron criterios de aceptación para estas, los cuales se ven en la Tabla 17.

Tabla 17: Criterios de aceptación de historias de usuario.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Criterios de aceptación | | | | | |
| ID H.U. | Número de escenario | Criterio de aceptación (Título) | Contexto | Evento | Resultado / Comportamiento esperado |
| OF0348-HU-01 | 1 | Creación exitosa | En caso de que de crear nuevas credenciales de consumidor con valores validos | cuando agrega una nueva credencial | el sistema mostrará el cuadro de información «Creación de credencial de Consumidor: Exitosa». |
| 2 | Agregar credencial existente | Si el «Consumer Key» generado ya se encuentra ingresado | cuando se agrega una nueva credencial | el sistema mostrará un cuadro de error: «Error: Credencial de Consumidor duplicada». |
| 3 | Agregar credencial con valores inválidos | En caso de que se agreguen valores inválidos. Ej.: «n° de creación de identidades permitidas» = «fg&» | cuando se agrega una nueva credencial | el sistema mostrará un cuadro de error: "Alguno de los valores ingresados es invalido". |
| OF0348-HU-02 | 1 | Modificación exitosa | Si se modifica una credencial de consumidor con valores validos | cuando modifica una credencial | el sistema mostrará el cuadro de información "Modificación de credencial de Consumidor: Exitosa". |
| 2 | modificar valores de credencial con valores inválidos | En caso de que se modifiquen valores inválidos. Ej.: «n° de creación de identidades permitidas» = «fg&» | cuando se modifica credenciales | el sistema mostrará un cuadro de error: "Alguno de los valores ingresados es invalido". |
| OF0348-HU-03 | 1 | Sin identidades creadas | En caso de que una credencial de consumidor no haya creado identidades | cuando el navegador se encuentre en la ventana de visualización de identidades | el sistema no mostrará una lista de identidades en la lista desplegable de texto. |
| 2 | Con identidades creadas | En caso de que una credencial de consumidor haya creado identidades | cuando esté en la ventana de visualización de identidades | el sistema mostrará una lista desplegable de texto en donde estarán las identidades creadas. |
| OF0348-HU-04 | 1 | Sin identidades creadas con la «credencial de consumidor» | En el caso que no se hayan creado identidades | cuando acceda al visualizar las identidades creadas por medio de una «credencial de consumidor» | El sistema no mostrará identidades. |
| 2 | Con identidades creadas con la «credencial de consumidor» | En el caso que se hayan creado identidades | cuando acceda al visualizar las identidades creadas por medio de una «credencial de consumidor» | El sistema mostrará las identidades creadas con anterioridad con la «credencial de consumidor» seleccionada. |
| OF0348-HU-05 | 1 | Eliminar identidad | En el caso que presione el botón eliminar de una identidad | cuando me encuentre en la ventana de visualización de identidades asociada a una «credencial de consumidor» | el sistema deberá mostrar un cuadro de confirmación, para comprobar la eliminación de la identidad. |
| 2 | Aceptar borrado de identidad | En el caso que presione el botón eliminar de una identidad y luego acepte esta | N/A | El sistema deberá redirigirme a la página de visualización de identidades creadas por medio de «credenciales de consumidor», no listándose la identidad que borré. |
| 3 | Cancelar borrado de identidad | En el caso que presione el botón eliminar de una identidad y luego cancele la eliminación | N/A | El sistema deberá redirigirme a la página de visualización de identidades creadas por medio de «credenciales de consumidor», permaneciendo la identidad. |
| OF0348-HU-06 | 1 | Habilitar «Signing Forms» | En caso de que se habilite mediante un «radio button» la característica «Signing Forms» y se encuentre habilitada la característica «Registración In-Band» | cuando guarde los cambios en la ventana de configuración de registro | el sistema permitirá registrar «Things» o clientes XMPP mediante el uso de «credenciales de consumidor». |
| 2 | Deshabilitar «Signing Forms» | En caso de que se deshabilite mediante un «radio button» la característica «Signing Forms» y se encuentre habilitada la característica «Registración In-Band» | cuando guarde los cambios en la ventana de configuración de registro | el sistema permitirá registrar «Things» o clientes XMPP mediante el uso de «credenciales de consumidor». |
| 3 | Inhabilitar cambios a «Signing Forms» | En caso de que se encuentre deshabilitada la característica «Registration in-Band» | cuando se está visualizando la ventana de configuración de registro | el sistema me impedirá realizar cambios en la configuración de «Signing Forms» |
| OF0348-HU-07 | 1 | Registro exitoso mediante «Signing Forms» | En caso de que una «Thing» intente registrar una cuenta mediante «Registration in-Band» y «Signing Forms» en el servidor con método de firmado «HMAC-SHA256» con credenciales de consumidor validas | cuando en el servidor XMPP se encuentre habilitado el registro de identidades mediante registro en banda con el mecanismo de firmado de formularios y se encuentre la establecida la conexión con este. | se podrá realizar login en el servidor con el nuevo usuario. |
| 2 | Registro fallido mediante «Signing Forms» 1 | En caso de que una «Thing» intente registrar una cuenta mediante «Registration in-Band» y «Signing Forms» en el servidor con método de firmado «HMAC-SHA256» con credenciales de consumidor invalidas o nulas. | cuando en el servidor XMPP se encuentre habilitado el registro de identidades mediante registro en banda con el mecanismo de firmado de formularios y se encuentre la establecida la conexión con este. | Se atrapará el mensaje de error y se lanzará una excepción y no se podrá realizar login en el servidor XMPP con el nuevo usuario. |
| 3 | Registro fallido mediante «Signing Forms» 2 | En caso de que una «Thing» intente registrar una cuenta mediante «Registration in-Band» y «Signing Forms» en el servidor con metodo de firmado «HMAC-SHA256» con credenciales de consumidor validas | cuando en el servidor XMPP se encuentre deshabilitado el registro de identidades mediante registro en banda con el mecanismo de firmado de formularios y se encuentre la establecida la conexión con este. | Se atrapará el mensaje de error y se lanzará una excepción, mostrando además que la característica no está habilitada o no esta soportada y no se podrá realizar login en el servidor XMPP con el nuevo usuario. |
| OF0348-HU-08 | 1 | Registrar Thing | En el caso que registre un dispositivo por primera vez en la red XMPP-IoT | Cuando este posea una conexión a internet y el servidor XMPP-IoT posea habilitados los mecanismos «In-Band Registration» y «Signing Forms» | Deberá mostrar al usuario el nuevo JID de dispositivo en cuestión |
| 2 | Leer datos de  Thing Sensor | En el caso que el usuario registrado lea el/los valores del sensor con el comando $valor o ALL en la red XMPP-IoT | Cuando el dispositivo se encuentre conectado a internet y el servidor esté operativo. | El sensor deberá responder el $valor solicitado, y si no conoce este $valor deberá responder un mensaje de ayuda con los valores disponibles de consulta. |
| 3 | Escribir datos de Thing Actuador | En el caso que el usuario registrado solicite escribir de un actuador en la red XMPP-IoT | Cuando el dispositivo se encuentre conectado a internet y el servidor esté operativo. | El dispositivo actuador deberá escribir el valor y responder "escritura ok" en el caso de valores válidos, en caso contrario, deberá responder un mensaje de ayuda, explicando como escribir valores. |

Tabla 18: Matriz de trazabilidad.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID HU | RFN01 | RFN02 | RFN03 | Objetivo General | Objetivo Especifico 1 | Objetivo Especifico 2 | Objetivo Especifico 3 |
| OF0348-HU-01 | X | X | X | X | X | X | X |
| OF0348-HU-02 | X | X |  |  |  |  |  |
| OF0348-HU-03 | X | X |  |  |  |  |  |
| OF0348-HU-04 | X | X |  |  |  |  |  |
| OF0348-HU-05 | X | X | X | X | X | X | X |
| OF0348-HU-06 | X | X | X | X | X | X | X |
| OF0348-HU-07 | X | X | X | X | X | X | X |
| OF0348-HU-08 | X |  |  | X |  |  |  |

## 4.2. Plataforma XMPP-Iot OpenFire + SMACK















Las redes de Internet de las Cosas (IoT) pueden ser implementadas utilizando diversos protocolos de comunicaciones. Los más habituales al crear estas redes son: HTTP, CoAP, MQTT y XMPP, y en la selección de estos protocolos, dejamos de lado los medios de comunicación que, exclusivamente, son M2M (Machine to Machine) debido a que estos últimos, por su definición de comunicación exclusivamente entre maquinas, no entran en la categoría de protocolo IoT, propiamente tal. En la Tabla 19 se comparan los protocolos IoT antes mencionados, se expone las diversas características discriminatorias al momento de realizar la elección del mejor protocolo, según la necesidad. D que que la El que l, y

Tabla 19: Comparativa protocolos IoT más utilizados.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Característica | HTTP | CoAP | MQTT | XMPP |
|  | **[7]** | **[15]** | **[1]** | **[10] [14]** |
| Request /  Response | ✓ | ✓ | ✗ | ✓ |
| Publish /  Subscribe | ✗ | ✗ | ✓ | ✓ |
| Multicast | ✗ | ✓ | ✗ | ✓ |
| Events or  Push | ✓ | ✓ | ✓ | ✓ |
| Bypasses firewall | ✗ | (✓) | ✓ | ✓ |
| Federation | ✗ | ✗ | ✗ | ✓✓ |
| Authentication | ✓ | ✓ | ✓ | ✓ |
| Network Identity | (✓) | (✓) | ✗ | ✓ |
| Authorization | ✗ | ✗ | ✗ | ✓✓ |
| Encryption | ✓ | ✓ | ✓ | ✓ |
| End-to-end encryption | ✗ | ✗ | ✗ | ✓✓ |
| Compression | ✓ | ✗ | ✗ | ✓ |
| Streaming | ✓ | ✗ | ✓ | ✓ |
| Reliable messaging | ✗ | ✗ | ✓✓ | ✗ |
| Message  Queues | ✗ | ✗ | ✗ | ✗ |

En cada uno de los protocolos listados en la Tabla 19 existen desafíos en el área de seguridad. Por ejemplo, con HTTP se pueden crear sistemas altamente robustos y seguros, sin embargo, el problema de la topología de HTTP es un gran obstáculo. Debido a que los clientes se pueden conectar a los servidores, pero normalmente los servidores no se pueden volver a conectar a los clientes, ya que la mayoría de ellos se encuentran detrás de los firewalls.

CoAP [15], es un protocolo de transferencia RESTful para nodos y redes con restricciones. Es escalable, tiene la posibilidad de implementar Datagram Transport Layer Security (DTLS) [9], sin embargo, aun así posee una pobre capa de seguridad y el Bypass de Firewall es bastante limitado, lo cual hace que exponer servicios a través de Internet sea bastante engorroso, monetariamente costoso y complejo. Por otro lado, el protocolo Message Queuing Telemetry Transport (MQTT) está basado en el patrón de diseño Publish-Suscribe, a diferencia de CoAP realiza un eficaz bypass de los firewalls, y soporta encriptación SSL/TLS, características realmente importantes cuando hablamos de redes IoT, sin embargo, el protocolo MQTT tiene serias vulnerabilidades conocidas [11], a tal nivel de gravedad, que, incluso gobiernos han llamado a no usarlo y han prohibido su uso en sistemas gubernamentales.

Al final, esta listado el Protocolo extensible de mensajería y comunicación de presencia (XMPP - por sus siglas en inglés) [10] [14], este es un protocolo abierto basado en flujos XML para la comunicación en tiempo real. XMPP fue desarrollado en el año 1998 con el nombre Jabber, después estandarizado por la Internet Engineering Task Force (IETF) liberando así su primera versión oficial bajo el nombre de Jabber, para luego, en el año 2002, pasar a conocerse con el nombre XMPP. El protocolo es estandarizado por Fundación de Estándares XMPP (XSF - por sus siglas en ingles), quienes corrigen errores y extienden las funcionalidades de este protocolo, por medio de los que se conocen por Protocolos de Extension de XMPP (XEPs) [12]. Inicialmente fue desarrollado para mensajería instantánea, pero con el paso del tiempo, ha sido utilizado ampliamente para diversos fines iniciando desde IM, como para videojuegos, notificaciones Push, Base de datos descentralizadas, redes sociales, microblogging, sistemas de videoconferencias, y finalmente Internet de las Cosas. Es un protocolo muy versátil, probado, con una larga data, y escalable. Esto, sumado al soporte de XSF, es lo hace que XMPP sea un protocolo actualizado y seguro.

Como podemos apreciar en la Tabla 19, XMPP es el que cumple de mejor manera las características listadas para seleccionar un protocolo IoT con difusión en la red de Internet.

A pesar de lo mencionado, existe un problema habitual al momento de crear una red XMPP-IoT, que reside en brindar una experiencia transparente para el usuario final, el cual desea adicionar nuevas “cosas” (tipo Plug and Play), tales como sensores, actuadores, controladores o concentradores a la red. Lamentablemente, para este, el uso interoperable de Things de diversos fabricantes de dispositivos, no es viable, ya que cada uno de estos, utiliza protocolos distintos o redes cerradas. Para solucionar esta problemática existe en XMPP, el registro en banda, mecanismo descrito en el XEP-0077: In-Band Registration, el que permite la creación de nuevas identidades donde cada una representa un dispositivo - utilizando el mismo canal de comunicaciones para conexión a los servidores de la red XMPP-IoT, de este modo, cada fabricante puede crear identidades para sus dispositivos en la red, simplificando el control unificado de estos. El problema en este caso es que no hay control sobre cuantas identidades crea el fabricante, ni mucho menos un límite en la cantidad del registro de nuevas cuentas para los dispositivos de este último.

En efecto, es útil cuando hablamos de redes XMPP-IoT de un sólo fabricante. En este caso, la gran mayoría de los servidores, tienen implementado el XEP-0077, para realizar el registro en banda, usando el mismo protocolo XMPP, así se evita usar el método alternativo de crear una web API para crear nuevas identidades, añadiendo posibles vulnerabilidades extra a nuestra red y reduciendo la flexibilidad de esta. El problema fundamental con el Registro en banda, propiamente tal, es que deja totalmente expuesta nuestra red XMPP-IoT al ataque de usuarios/bots maliciosos, ya que pueden sobrecargar nuestros servidores, con solicitudes virtualmente ilimitadas de creación de nuevas cuentas, y los servidores no tienen la capacidad de discriminar las solicitudes benignas de las nocivas. Frente a tal desafío, en los inicios de XMPP, se desarrolló el XEP-0158: CAPTCHA Forms, que describe el mecanismo que opera en conjunto del registro en-banda, para añadirle CAPTCHAs a este último. Esta solución fue sumamente útil para proteger las redes XMPP de los ataques de bots maliciosos durante los primeros años de la popularidad de la mensajería instantánea, cuando el poder de cómputo de los ordenadores personales era bastante inferior al de la actualidad. Pero hoy, no es suficiente, debido a que por medio de visión computacional es relativamente sencillo resolver los CAPTCHAs [3].

De todos modos, cuando se utiliza el XEP-0077 en conjunto de CAPTCHAS Forms para filtrar a los usuarios maliciosos, las “cosas” - que en estricto rigor son bots - no pueden registrarse de forma automatizada en la red XMPP-IoT. Así que esta opción, si fuese segura, sabiendo que no lo es, no es una opción viable para crear dispositivos “plug and play”. Para este fin, es decir, para crear una red XMPP-IoT con la capacidad de registrar en-banda nuevas identidades, generalmente se activa el registro en banda, sin habilitar CAPTCHAS Forms. Lo cual, nos permite tener dispositivos “plug and play”. Esto, al mismo tiempo, abre una brecha en la seguridad de la red XMPP-IoT. Permitiendo que cualquiera - cliente inocuo o maligno, bot benigno o malicioso - pueda visualizar a nivel de red el servidor y esté plenamente facultado para registrar identidades en el servidor, lo que abre una brecha en la seguridad en la red XMPP-IoT.

Este trabajo posee las siguientes secciones. Sección 2 presenta la propuesta para disminuir los ataques de red y dar una alternativa de interoperabilidad a los fabricantes de dispositivos. Sección 3 describe como esta propuesta es implementada en un servidor y cliente XMPP para verificar con un experimento que efectivamente los ataques se reducen a cero. Sección 4 describe las conclusiones y trabajos futuros.

### 4.2.1. Propuesta

Como ya hemos visto en la Tabla 19, XMPP es más robusto en términos de seguridad que el resto de los protocolos comparados, ya que existen XEPs que tributan en esta área. Tal como lo es el protocolo de extensión de XMPP XEP-0348: Signing Forms[16], que describe el mecanismo para realizar el firmado de formularios, con credenciales que no tienen directa relación con la conexión - usando el algoritmo modificado de OAuth 1.0[8] para así concretar el registro de nuevas identidades en la red XMPP, esto es, usando el protocolo de extensión XEP-0077[13] que describe el mecanismo para realizar un registro en banda de nuevas cuentas en el servidor XMPP.

En términos prácticos, al unificar estas dos tecnologías, es decir, el registro en banda junto al proceso de firmado de formularios de registro se dota a la red XMPP-IoT de la capacidad de otorgar el permiso para registrar nuevas identidades a los fabricantes de dispositivos IoT (Cosas). Donde las credenciales para crear estas nuevas identidades en la red, no tienen relación con la conexión o ingreso a la red XMPP, estas toman el nombre de credenciales de consumidor que es el par “consumer key” y “consumer secret”, las que a su vez se asocian a un cantidad de registro de nuevas identidades permitidas en la red por dicha credencial, como por ejemplo: la consumer key=00000000 perteneciente al manufacturador de dispositivos BinaryLamp, sólo podrá crear 1000 identidades en el servidor XMPP-IoT, posibilitando, de este modo, la interoperabilidad de dispositivos IoT de distintas compañías, sacando provecho de la federalización de XMPP o mediante la concesión de credenciales de consumidor a los fabricantes de dispositivos IoT, para que estos puedan registrar nuevas identidades en la red, por medio de estas credenciales o dos opciones anteriores en su conjunto.

Para dar solución a la problemática expuesta, debemos seleccionar un servidor XMPP. En la actualidad existe una gran cantidad de servidores XMPP. Existen de uso libre o de pago, de código abierto o propietario y desarrollado en diversos lenguajes de programación, documentación variada, con una comunidad existente en torno al servidor, que operan sobre diferentes sistemas operativos y con diversas XEPs implementadas. *IoT Broker* (https://waher.se/Broker.md), servidor XMPP de uso no libre, no es open-source y no hay una comunidad incipiente en torno al proyecto. *AstraChat Isode M-Link*, cuyo uso es de pago, mientras que *ejabberd*, *Tigase* y *Openfire* son de uso libre. Los primeros dos poseen una comunidad también incipiente en torno al desarrollo de nuevas características. Por otro lado, tenemos *Openfire* que es de uso libre, open-source, esta codificado en lenguaje Java, funciona bajo los Sistemas Operativos Linux, macOS, Solaris y Windows, inicialmente desarrollado por la empresa Jive Software, quienes ayudaron ayudador a formar la gran comunidad que mantiene activo el proyecto en la actualidad, llamada Igniterealtime. La misma comunidad posee *SMACK*, una librería que tiene una potente API para crear clientes XMPP, la que ostenta las mismas características mencionadas del servidor.

El servidor XMPP Openfire incluye soporte completo de RFC XMPP, así como las extensiones más comunes. La Tabla 20 se detalla el nivel de soporte para los requisitos establecidos por XEP-0302: XMPP Compliance Suites 2012.

Tabla 20: Nivel de soporte de Openfire de los requisitos XMPP Compliance Suites 2012.

|  |  |
| --- | --- |
| Especificación | Soportado |
| RFC 6120: XMPP Core | Sí |
| RFC 6121: XMPP IM | Sí |
| RFC 7622: XMPP ADDR | Sí |
| XEP-0030: Service Discovery | Sí |
| XEP-0114: Jabber Component Protocol | Sí |

El cumplimiento de soporte de nivel avanzado incluye el conjunto básico completo, así como características más avanzadas de uso común para los clientes XMPP. La Tabla 21 se detalla el nivel de soporte para los requisitos establecidos por XMPP Compliance Suites 2012.

Tabla 21: Soporte XEPs avanzado de XMPP.

|  |  |
| --- | --- |
| Especificación | Soportado |
| XEP-0115: Entity Capabilities | Sí |
| XEP-0191: Blocking Command | No |
| XEP-0124: Bidirectional-streams Over Synchronous HTTP (BOSH) | Sí |
| XEP-0206: XMPP Over BOSH | Sí |
| XEP-0054: vcard-temp | Sí |
| XEP-0163: Personal Eventing Protocol | Sí |
| XEP-0045: Multi-User Chat | Sí |
| XEP-0198: Stream Management | Parcialmente |

En la Tabla 22 se listan los XEPs compatibles con Openfire. Se omiten los XEPs que solo requieren soporte del lado del cliente.

Tabla 22: XEPs soportados por Openfire.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Para crear el cliente usamos **SMACK**, el cual soporta las siguientes XEPs, listados en la Tabla 23.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Tabla 23: XEPs soportados por SMACK.

|  |  |  |
| --- | --- | --- |
| Especificación |  | Especificación |
| n/a-Google GCM JSON payload | XEP-0138-Stream Compression |
| n/a-Group Chat Invitations | XEP-0141-Data Forms Layout |
| n/a-Jive Properties | XEP-0163-Personal Eventing Protocol |
| XEP-0004-Data Forms | XEP-0184-Message Delivery Receipts |
| XEP-0012-Last Activity | XEP-0191-Blocking Command |
| XEP-0013-Flexible Offline Message Retrieval | XEP-0199-XMPP Ping |
| XEP-0016-Privacy Lists | XEP-0202-Entity Time |
| XEP-0022-Message Events | XEP-0203-Delayed Delivery |
| XEP-0030-Service Discovery | XEP-0206-XMPP Over BOSH |
| XEP-0033-Extended Stanza Addressing | XEP-0224-Attention |
| XEP-0045-Multi User Chat | XEP-0231-Bits of Binary |
| XEP-0047-In-Band Bytestreams | XEP-0280-Message Carbons |
| XEP-0048-Bookmarks | XEP-0296-Best Practices for Resource Locking |
| XEP-0049-Private Data | XEP-0308-Last Message Correction |
| XEP-0050-Ad-Hoc Commands | XEP-0313-Message Archive Management |
| XEP-0054-vcard-temp | XEP-0319-Last User Interaction in Presence |
| XEP-0055-Jabber Search | XEP-0323-Internet of Things - Sensor Data |
| XEP-0059-Result Set Management | XEP-0324-Internet of Things - Provisioning |
| XEP-0060-PubSub | XEP-0325-Internet of Things - Control |
| XEP-0065-SOCKS5 Bytestreams | XEP-0332-HTTP over XMPP transport |
| XEP-0071-XHTML-IM | XEP-0333-Chat Markers |
| XEP-0077-In-Band Registration | XEP-0334-Message Processing Hints |
| XEP-0079-Advanced Message Processing | XEP-0335-JSON Containers |
| XEP-0080-User Location | XEP-0347-Internet of Things - Discovery |
| XEP-0082-XMPP Date Time Profiles | XEP-0352-Client State Indication |
| XEP-0085-Chat State Notifications | XEP-0357-Push Notifications |
| XEP-0090-Time Exchange | XEP-0359-Stable and Unique Stanza IDs |
| XEP-0092-Software Version | XEP-0363-HTTP File Upload |
| XEP-0093-Roster Item Exchange | XEP-0372-References |
| XEP-0095-Stream Initation | XEP-0382-Spoiler Messages |
| XEP-0096-SI File Transfer | XEP-0384-OMEMO Multi End  Message and Object Encryption |
| XEP-0115-Entity Capabilities |
| XEP-0116-Jingle | XEP-0392-Consistent Color Generation |
| XEP-0122-Data Forms Validation | XEP-0394-Message Markup |
| XEP-0133-Service Administration | XEP-xxxx-Multi-User Chat Light |

En este trabajo proponemos desarrollar e implementar el mecanismo descrito en la especificación **XEP-0348: Signing Forms** sobre la **XEP-0077: In-band Registration**, tanto en el servidor como en el cliente, para que, de este modo, se pueda asignar a un cierto lote definido de clientes la facultad de crear identidades en la red XMPP-IoT. Con esto reduciremos a cero la creación de identidades por usuarios o bot maliciosos mediante la vulnerabilidad de la XEP-0077. En particular, para validar nuestra propuesta, realizaremos un experimento implementándola en el cliente *SMACK* y en el servidor *Openfire* tal como se aprecia en la Figura 9 y la Figura 10.

Figura 9: Diagrama de despliegue.

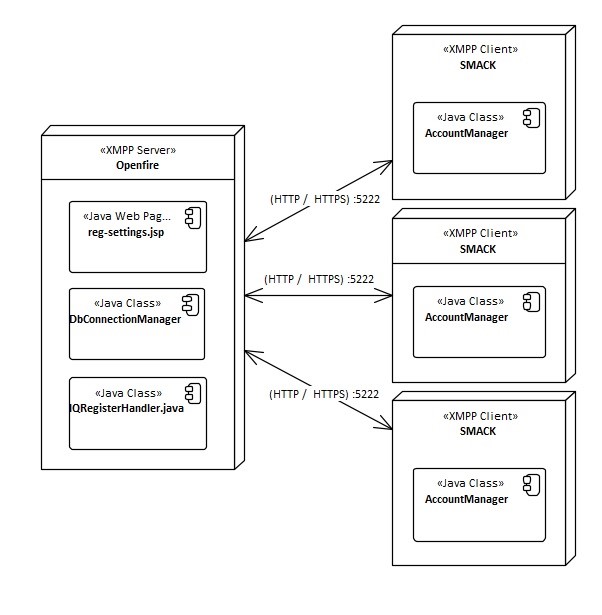
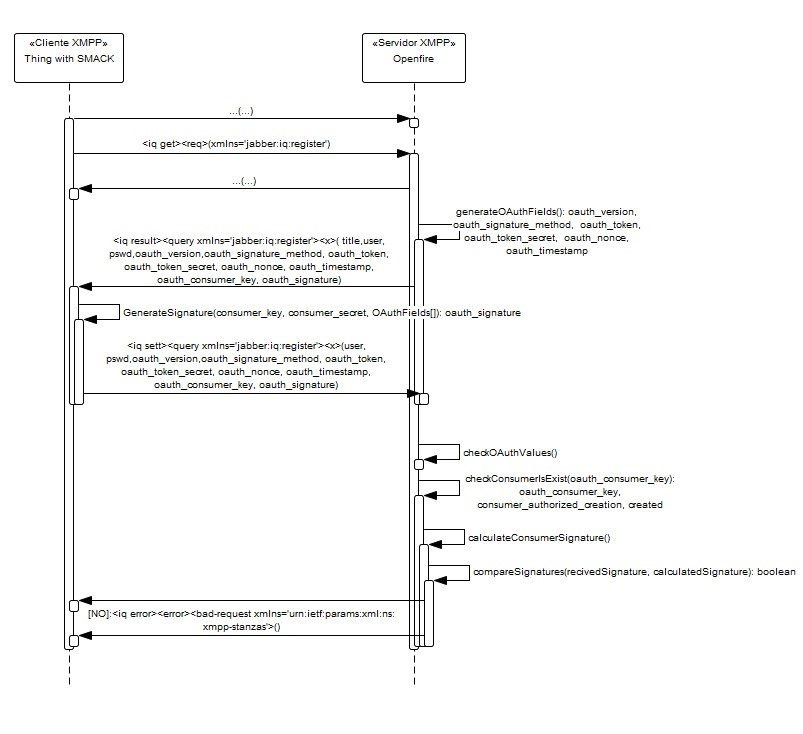


Figura 10: Diagrama de secuencia de Registration In-Band en conjunto con Signing Forms.



Para ello se planificó el desarrollo de este nuevo mecanismo en el servidor XMPP Openfire y en la librería SMACK, desarrollo que constó de cuatro liberaciones relevantes:

1. Bot que demuestra la vulnerabilidad existente en la XEP-0077 en el contexto de IoT.
2. Servidor Openfire con las mejoras para incorporar el XEP-0348.
3. Librería SMACK incorporando mejoras para implementando el XEP-0348.
4. Cliente Thing que utiliza la versión mejorada de SMACK, para validar la propuesta.

Inicialmente en el desarrollo, se codifico un pequeño bot, utilizando la librería SMACK, el cual tiene la facultad de crear, virtualmente, ilimitadas identidades en el servidor Openfire. Los resultados de este se pueden ver en la Sección 3. Después de la creación del bot, se continuo con el análisis, diseño, codificación y con las respectivas pruebas de la mejora del Servidor XMPP. Esto genero los resultados visualizados en la Figura 4 en donde se muestra el Diagrama de clases de Openfire, que permite visualizar las clases afectadas en el proyecto, incluyendo “IqRegisterHandler.class”, “Form.class”, “DataForm.class”, “DataForms.Item.class”, DbConnection.class y “regsettings”. Este último es responsable de la visualización de la consola web, que como su nombre indica, es el encargado de la configuración de los mecanismos de registro al interior del servidor Openfire. Para completar la explicación del funcionamiento de la implementación del XEP-0348 en Openfire, añadimos la Figura 11.

Figura 11: Diagrama secuencia desarrollo Openfire.

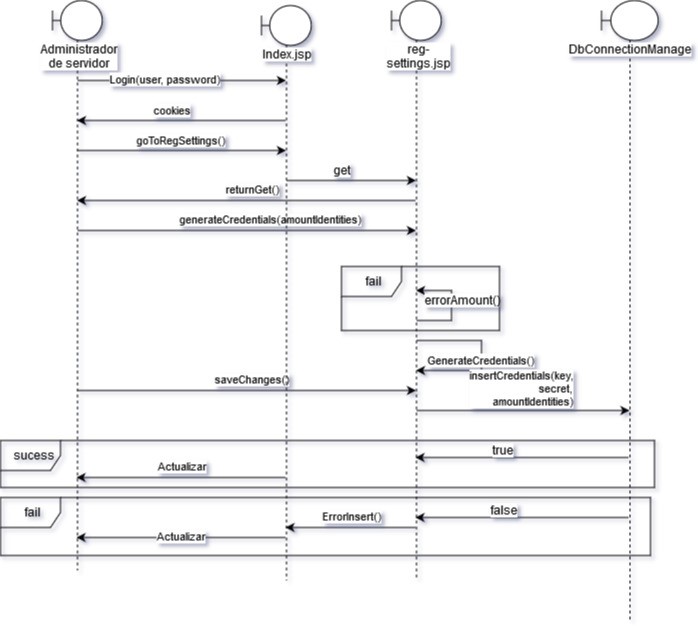
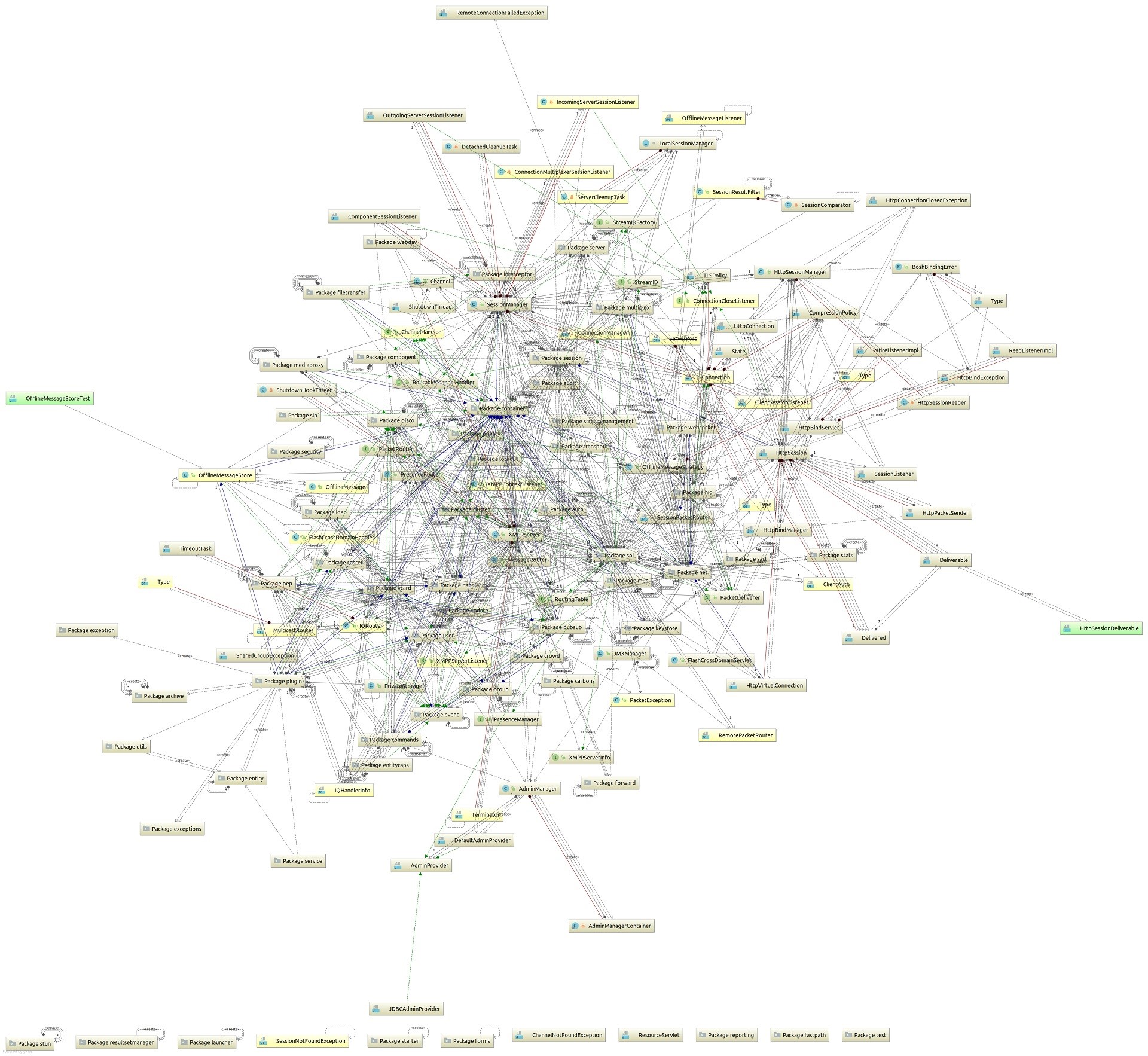
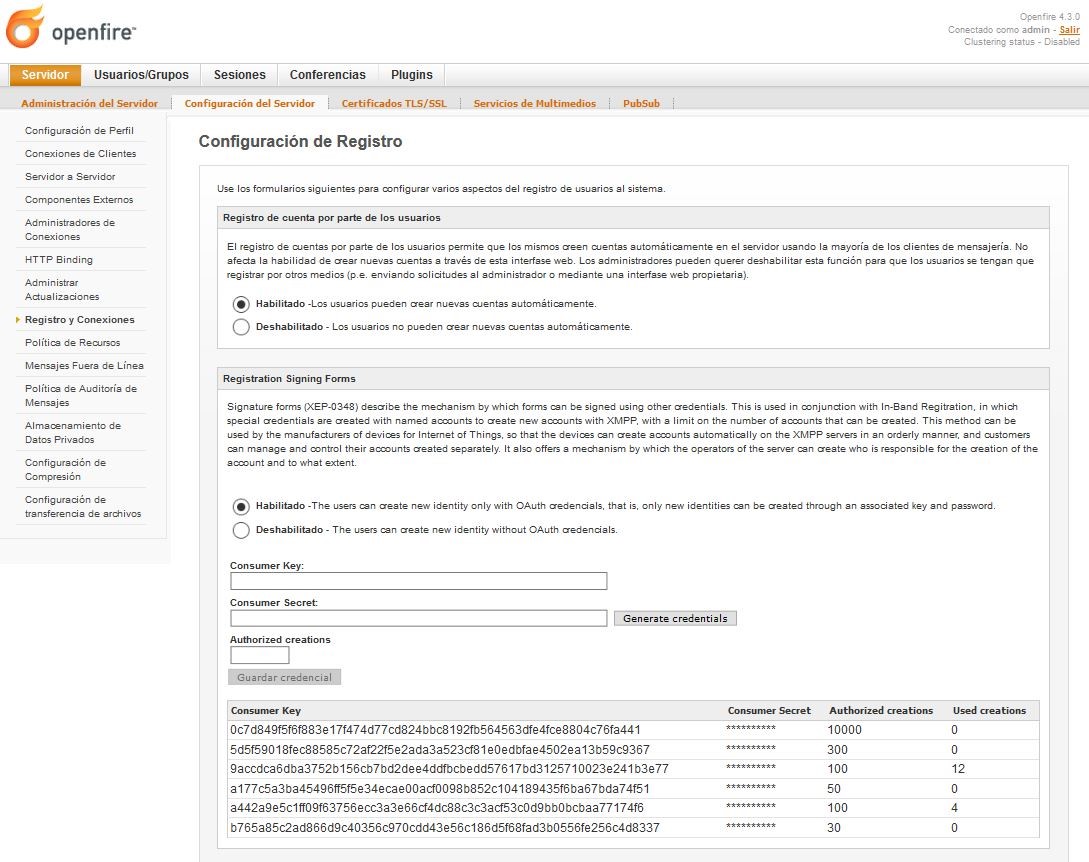


Figura 12: Diagrama de paquetes de Openfire.



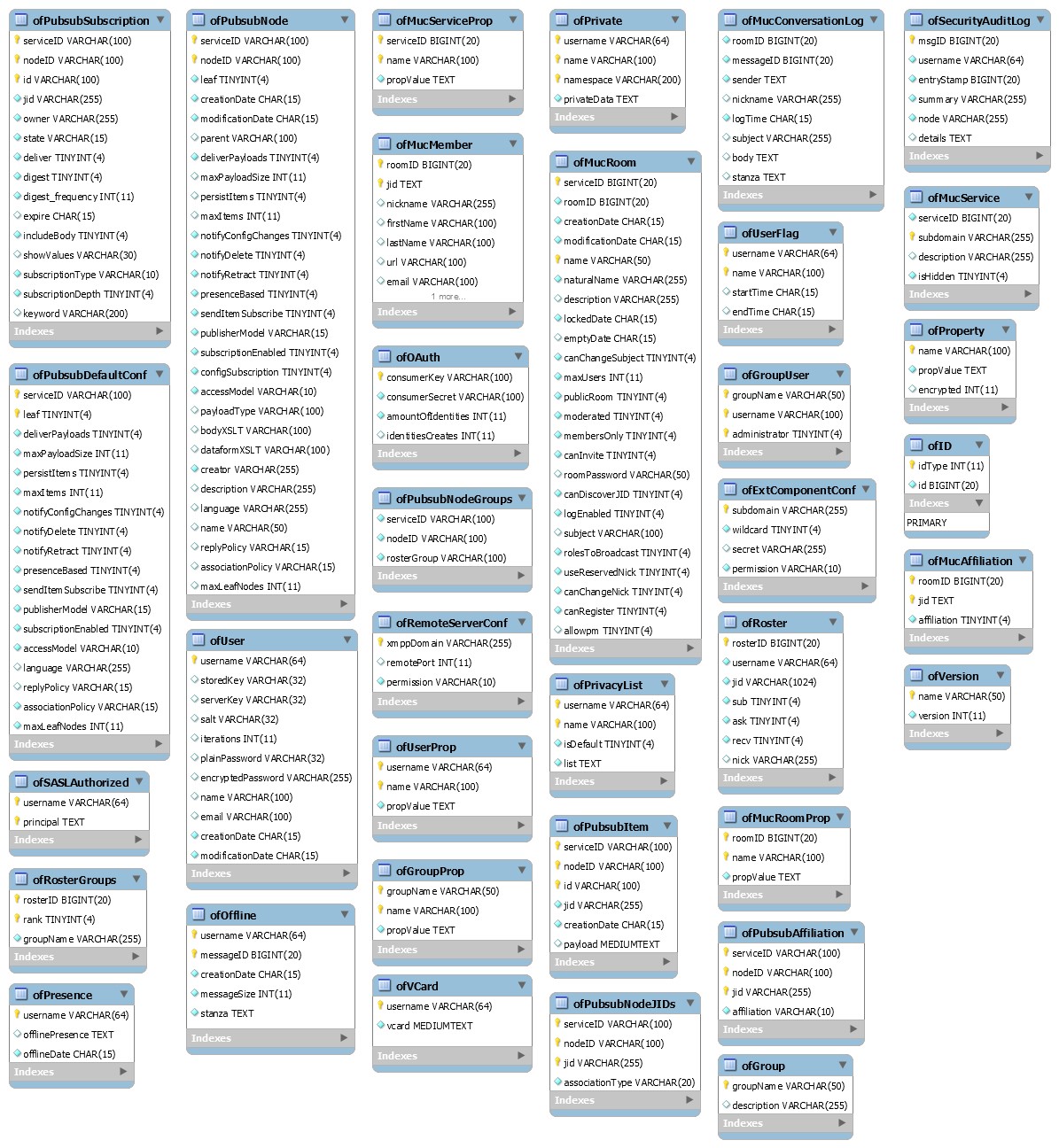
Luego del desarrollo, pruebas y refactoring, se puede ver el resultado visible en la consola web del servidor, esta última se aprecia en la Figura 13. Aquí se ve el panel de “Configuración de Registro” de la consola web del servidor Openfire, donde la segunda sección que lleva el título de “Registration Signing Forms”, es una de las partes desarrolladas. En esta zona, el administrador de la red XMPP puede habilitar o deshabilitar el uso del mecanismo descrito en el XEP-0348: Signing Forms, como también generar nuevas credenciales de consumidor, donde cada una consiste en “llave de consumidor”, el “secreto del consumidor” y el número “creaciones autorizadas” este representa la cantidad de nuevas identidades que se pueden crear utilizando estas credenciales. Lo que significa, que cuando se llega a límite de la cantidad permitida, no se podrá continuar creando más identidades con dicha credencial. Por lo tanto, si es que el dueño de esta desea continuar creando identidades, tendrá que solicitar al administrador de la red XMPP-IoT que le genere nuevas credenciales.

Figura 13: Consola web de Openfire.



También se agregaron tablas a la base de datos al servidor Openfire, esto se puede apreciar en la Figura 14. Cabe mencionar que la manera de crear las relaciones es mediante indexs, y no por relaciones SQL propiamente dicho. La tabla agregada es “ofOAuth” para los fines del proyecto.

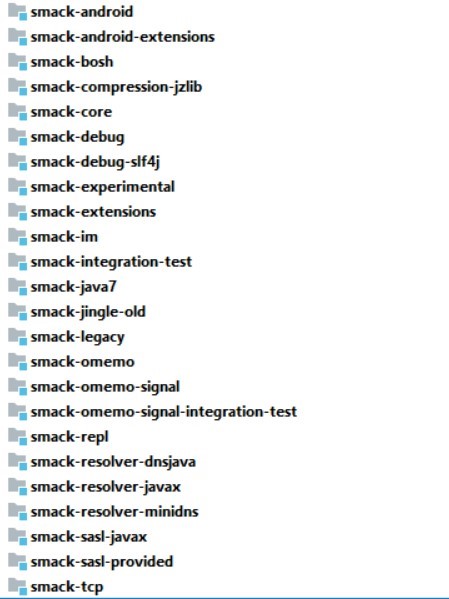
Figura 14: Diagrama de tablas de base de datos.



En el desarrollo de la mejora de Openfire, se escribieron 4595 líneas de código. La media de complejidad ciclomatica fue de 39. El valor es bastante alto, esto se debe a que la clase “IqRegHandler.class” posee un e(G) = 98, lo cual altera considerablemente la media en general. La clase inicialmente estaba puntada en 58, por ende, difícilmente se podía reducir la complejidad de esta añadiéndole funcionalidades.

El desarrollo de la librería SMACK, se inició identificando las entidades que posee el código fuente, el cual tiene en su interior varios subproyectos, que se pueden visualizar en la Tabla 24, examinando estos, identificamos que en “smack-extensions” es donde están las clases involucradas en la creación de nuevas identidades en la red XMPP-IoT.

Tabla 24: Proyectos anidados al interior de la librería SMACK.



Luego de esto, se procedió a identificar las clases involucradas en el mecanismo descrito en el XEP-0077: In-Band Registration. Agrupando estas, restan las clases mostradas en la Figura 15, luego de las modificaciones quedó acorde al diagrama ilustrado en la Figura 16.

Figura 15: Diagrama de clases previo a la intervención de la librería SMACK.

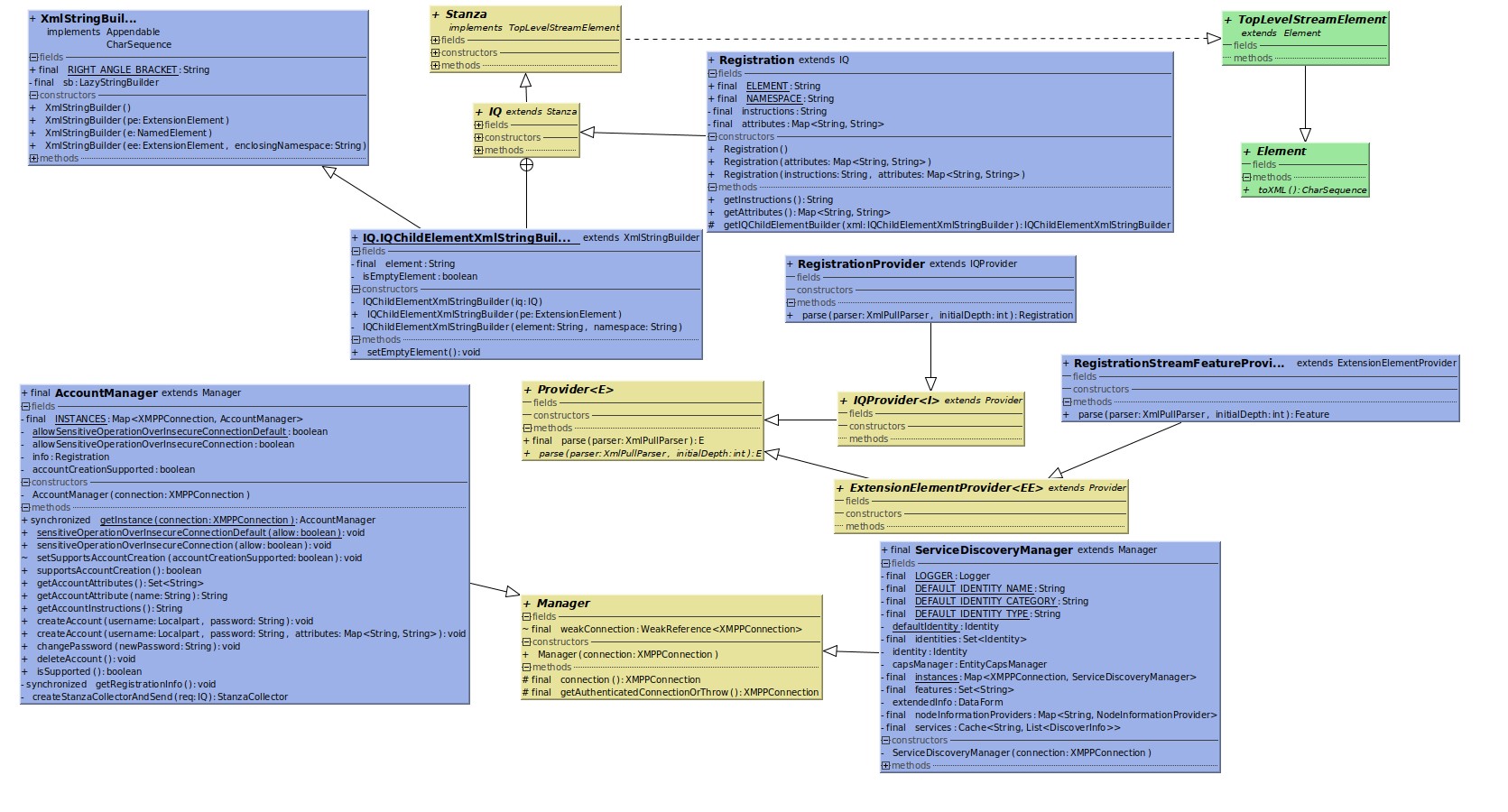
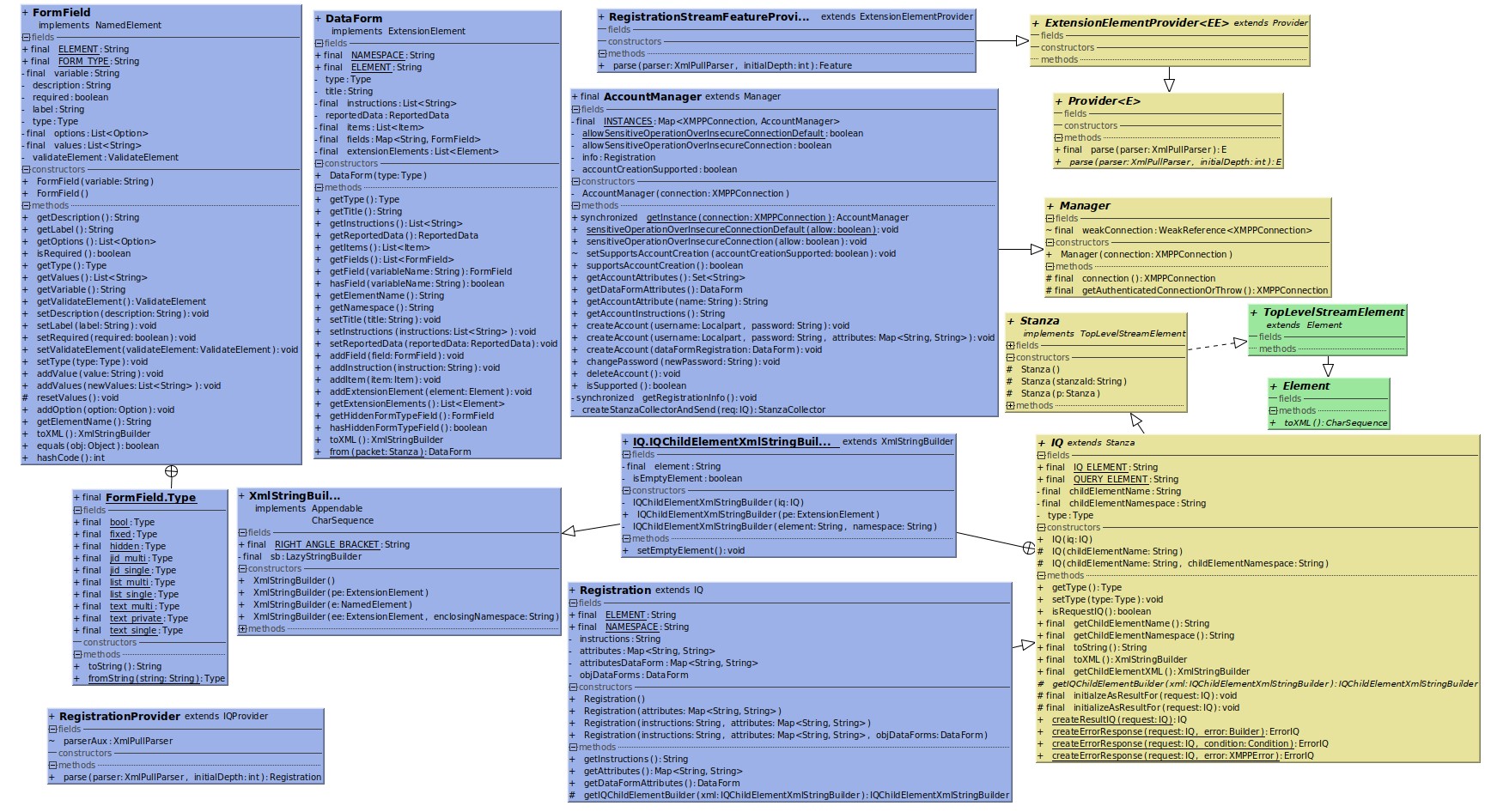


Figura 16: Diagrama de clases posterior a la intervención de la librería SMACK.



Finalmente, para depurar, en su conjunto, el sistema XMPP-IoT se creó un cliente usando la librería SMACK modificada, por ende, se incorporó incorporo las credenciales de consumidor. El cliente funciona como una Cosa que posee sensores de humedad, temperatura, gas, luz ambiente, movimiento y actuadores: interruptores de luces y un display LED de 16x2, mediante el cual veremos información relevante de nuestro dispositivo. El programa establece la conexión con el servidor Openfire, luego envía la stanzas para crear automáticamente una identidad en este, utilizando el registro en-banda en conjunto del mecanismo de firmado de formularios. El servidor Openfire, el que debido a las añadiduras se encuentra facultado para responder la solicitud, responde a esta, para finalmente crear la identidad en la red XMPP-IoT. Los detalles de las stanzas enviados en esta conversación entre el cliente y el servidor, se pueden apreciar en las figuras a continuación.

Figura 17: Consulta de Features soportadas de cliente a servidor.



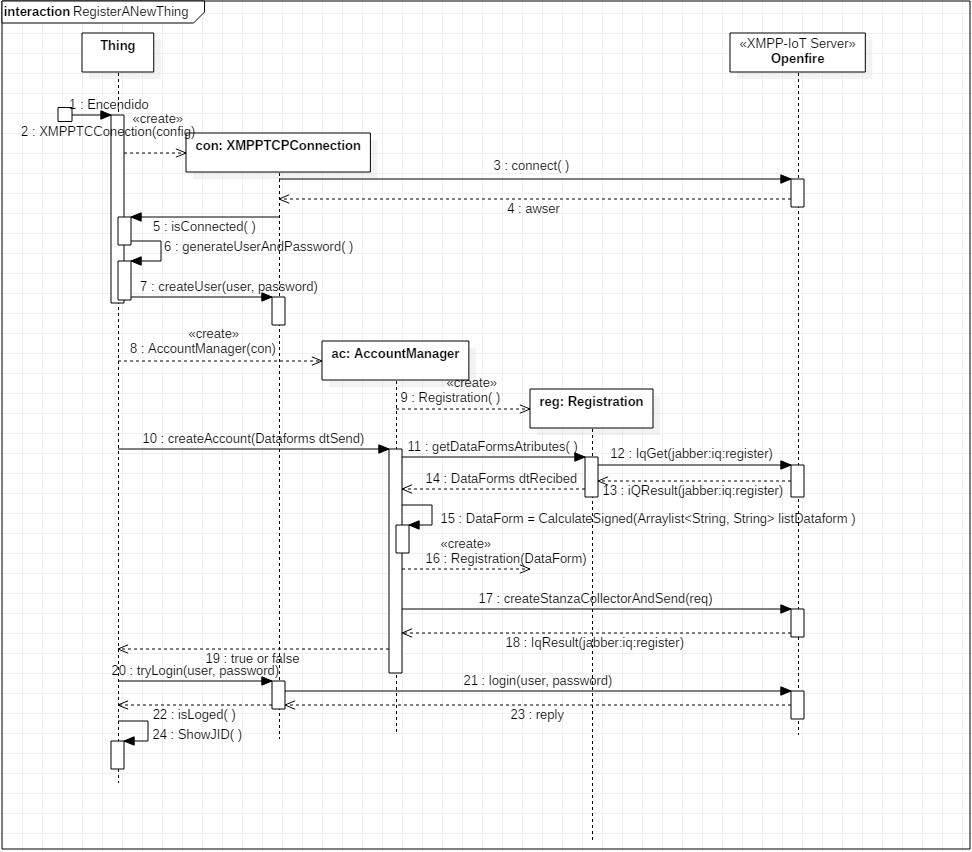
Figura 18: Respuesta de Features soportadas del servidor al cliente.



En el momento que se realiza la conexión con el servidor, el cliente XMPP se debe asegurar que el servidor soporte el registro en-banda utilizando formularios firmados, para eso el cliente envía la stanza en la Figura 17 y el servidor debe responder las “Features” que soporte, la que anuncia el soporte del XEP-0348, es “urn:xmpp:xdata:signature:oauth1”. En la Figura 18 se aprecia la stanza en donde el servidor responde todas las “Features” que soporta.

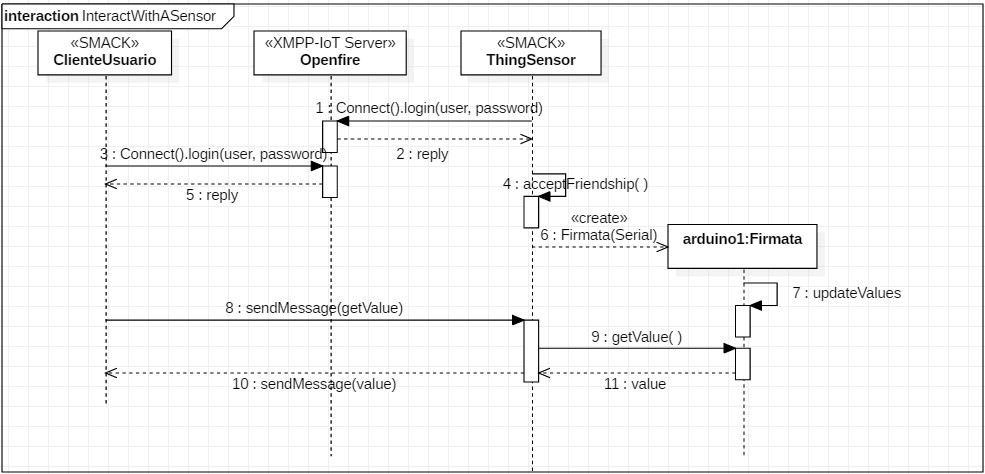
El proceso de creación de identidades también se puede entender, por medio del siguiente diagrama de secuencia en la Figura 19.

Figura 19: Diagrama de secuencia registro de una nueva identidad usando la librería SMACK modificada en un Thing, cliente XMPP.



En siguiente diagrama de interacción, en la Figura 20, se expresa las entidades que interactúan cuando un usuario solicita los valores percibidos - por medio de mensajes de chat - a un sensor particular.

Figura 20: Diagrama de secuencia de solicitud de lectura a un sensor.



De forma similar a la lectura de un sensor, se pueden escribir valores en un actuador. El proceso para realizar esto, se encuentra en el diagrama de la Figura 21.

Figura 21: Diagrama de secuencia de solicitud de escritura a un actuador.

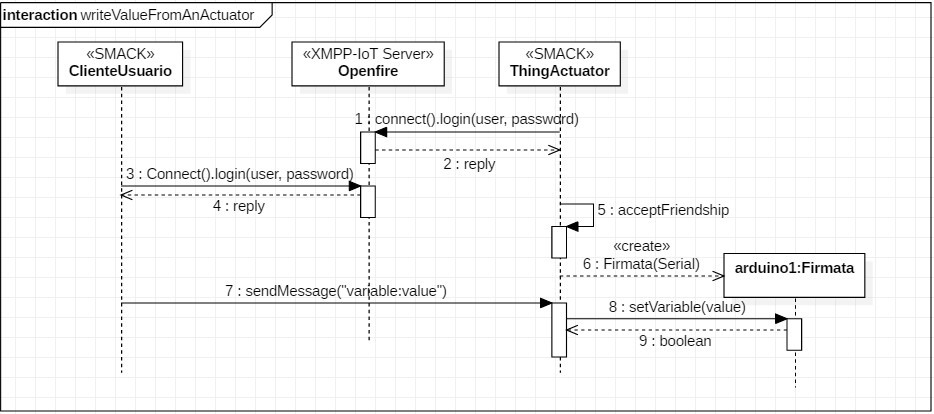


Figure 14: Escritura del valor en un actuador.

**Seguridad**

Con lo mencionado hasta el momento, se concluye el desarrollo de la mejora a la plataforma XMPP-IoT mediante la implementación de registro en-banda con firmado de formularios. Sin embargo, queda un tema pendiente, la seguridad en general. Si bien en esta publicación hablamos de seguridad por medio la delegación de responsabilidades para el registro de nuevas de identidades, usando el mecanismo con la XEP-0348: Signing Forms. De nada sirve que la red posea servidores con esta característica, si es existe la posibilidad que la identidad del servidor pueda ser fácilmente adulterada, es decir, que en el momento que un cliente XMPP se intenta conectar al servidor correspondiente, un oponente - hacker - puede realizar un ataque “Man in the Middle”, para suplantar la identidad de nuestro servidor, abriendo así la posibilidad de sustraer información sensible. Para minimizar el riesgo, una opción razonable es utilizar SSL/TLS, pero debido a los agujeros de seguridad en SSL producidos por la corta proyección al momento del diseño de este protocolo, lo deja vulnerable en todas sus versiones, desde la 1.0 a la 3.0. Debido a esta razón la mejor opción es usar TLS, y no cualquier versión, ya que también hay vulnerabilidades conocidas, las cuales, con un mediano poder de cómputo, pueden ser explotadas. La versión que utilizaremos será TLS 1.2 o la versión 1.1. TLS se puede dividir en 3 grandes fases o etapas, que consisten en:

1. Negociación de que algoritmo criptográfico se utilizará en la comunicación.
2. Intercambio de llaves públicas y la autentificación basada en certificados digitales.
3. Se cifra con criptografía simétrica el tráfico.

**Security Levels**

Cuando nos referimos a algoritmos criptográficos, cabe mencionar que cada uno proporciona diversos “puntos fuertes” de seguridad, que están en función de qué tipo de algoritmo y qué tamaño de clave este utilice. Los Security Levels o Bits of Security son usados para estimar la fortaleza del cifrado para proteger la información sensible, en función de fortaleza potencial histórica del adversario, es decir, permite cuantificar qué tan fuerte debe ser un algoritmo para ser catalogado como “algoritmo criptográficamente fuerte”. Generalmente está basado en el “mejor” ataque conocido sobre el algoritmo. Esto implica que cada vez el ataque avanza, también lo debe hacer el algoritmo y/o su clave.

Para cada algoritmo criptográfico existen diversas publicaciones que proporcionan recomendaciones y fórmulas para aproximarse al tamaño de clave adecuado para cada algoritmo. Si bien, estos estudios están disponibles, aún sigue siendo una tarea sumamente engorrosa seleccionar un algoritmo y tamaño de clave adecuado, debido que, para esto, se debe leer y comprender todos estos documentos. Como una solución ante tal mar de información, nace keylength.com, una web que pretende simplificar tal labor dando un resumen con los tamaños de clave recomendados basados en los Security Level, en donde uno de los factores discriminantes es el Bits of Security (BOS). A continuación, se puede ver la Tabla 25 y la Tabla 26 con un listado de los algoritmos comúnmente usados con sus respectivos valores recomendados y hasta cuando cada organismo recomienda usar dicho método.

Tabla 25: Recomendaciones Criptográficas ECRYPT del año 2018.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Protección | Simétrica | Factoring Modulus | Log. discreto  Llave Grupo | Curva Elíptica | Hash |
| Nivel estándar legado  No debería ser usado en sistemas nuevos | 80 | 1024 | 160 1024 | 160 | 160 |
| Protección a corto plazo  Seguridad durante al menos diez años (2018-2028) | 128 | 3072 | 256 3072 | 256 | 256 |
| Protección a largo plazo  Seguridad de treinta a cincuenta años (2018-2068) | 256 | 15360 | 512 15360 | 512 | 512 |

Tabla 26: Recomendaciones Criptográficas NIST del año 2016.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fecha | Security Strength | Algoritmos Simétricos | Factoring Modulus | Log. Discreto | | Curva Elíptica | Hash (A) | Hash (B) Ej.: HMAC |
| Llave | Grupo |
| (Legado) | 80 | 2TDEA | 1024 | 160 | 1024 | 160 | SHA-1 |  |
| 2016 - 2030 | 112 | 3TDEA | 2048 | 224 | 2048 | 224 | SHA-224  SHA-512/224  SHA3-224 |  |
| 2016 – 2030  & después | 128 | AES-128 | 3072 | 256 | 3072 | 256 | SHA-512/256  SHA3-256 | SHA-1 |
| 2016 – 2030 & después | 192 | AES-192 | 7680 | 384 | 7680 | 384 | SHA-384 SHA3-384 | SHA-224 SHA-512/224 |
| 2016 – 2030 & después | 256 | AES-256 | 15360 | 512 | 15360 | 512 | SHA-512 SHA3-512 | SHA-256 SHA-512/256  SHA-384  SHA-512  SHA3-512 |

Luego del análisis de los algoritmos y sus respectivas claves, considerando a un oponente con alto poder de cómputo, establecimos como protocolo criptográfico TLS Curva elíptica Diffie-Hellman RSA con AES256 SHA128-256 y como método de hashing HMAC SHA256 en la firma, en el mecanismo OAuth 1.0 modificado. En el primer caso en conjunto tiene un Security Strength de 256, al igual que OAuth, superando lo recomendado para la actualidad según el NIST [4] de EE. UU. y ECRYPT CSA de la Unión Europea [6] siendo ambas organizaciones posibles rivales, para tenerlos en consideración en caso de tener información sensible, según el criterio de ellos. En resumen, la plataforma XMPP-IoT está protegida por medio de los métodos criptográficos que podemos ver en la Tabla 27 y Tabla 28.

Tabla 27: Protocolos criptográficos utilizados.

|  |  |
| --- | --- |
| Protocolos | |
| SSLv2 | No |
| SSLv3 | No |
| TLSv1 | No |
| TLSv1.1 | Sí |
| TLSv1.2 | Sí |

Tabla 28: Métodos de cifrado utilizados.

|  |  |  |  |
| --- | --- | --- | --- |
| Conjunto de cifrado | Bitsize | Confidencialidad directa | Información |
| ECDHE-RSA-AES256-GCM-SHA384 | 256 | Sí | Curva: sect571r1 |
| ECDHE-RSA-AES256-SHA384 | 256 | Sí | Curva: sect571r1 |
| ECDHE-RSA-AES256-SHA | 256 | Sí | Curva: sect571r1 |

De igual manera, se configuró las entradas DNS SRV[2], para que la comunicación entre servidores y clientes fuese fiable, mediante las siguientes entradas DNS:

* xmpp-client. tcp.domain.tld. TTL IN SRV priority weight port target
* xmpp-server. tcp.domain.tld. TTL IN SRV priority weight port target

Realizadas todas estas acotaciones en cuanto a seguridad, podemos decir que, en cuanto al estado del arte en el contexto criptográfico, el sistema se encuentra a la vanguardia.

### 4.2.2. Validación

Como se describió con anterioridad, se ha codificado un bot-malicioso, que registrará identidades utilizando la vulnerabilidad existente en la XEP-0348

Tal como se ve en la Figura 1 el experimento consta de un servidor XMPP *Openfire* modificado con varios clientes XMPP *SMACK* modificados, estos últimos son:

* Un Cliente XMPP Thing-Sensor (TS–1) que posee un sensor de humedad y temperatura
* Un Cliente XMPP Thing-Sensor (TS–2) que posee un sensor de gas
* un Cliente XMPP Thing-Controlador (TC–1)

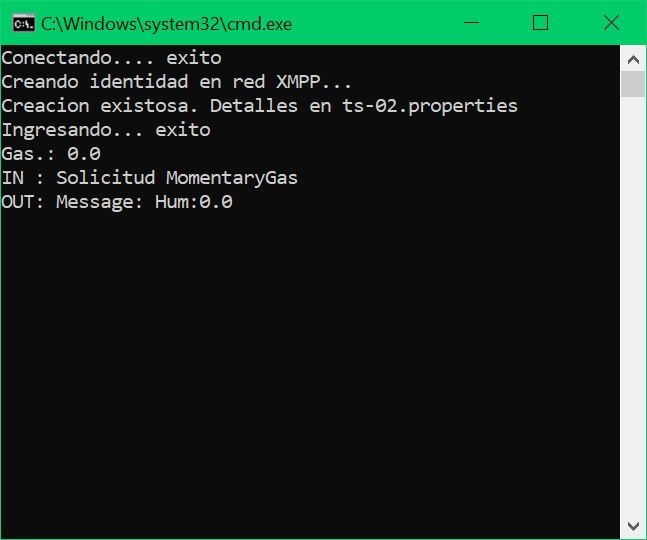
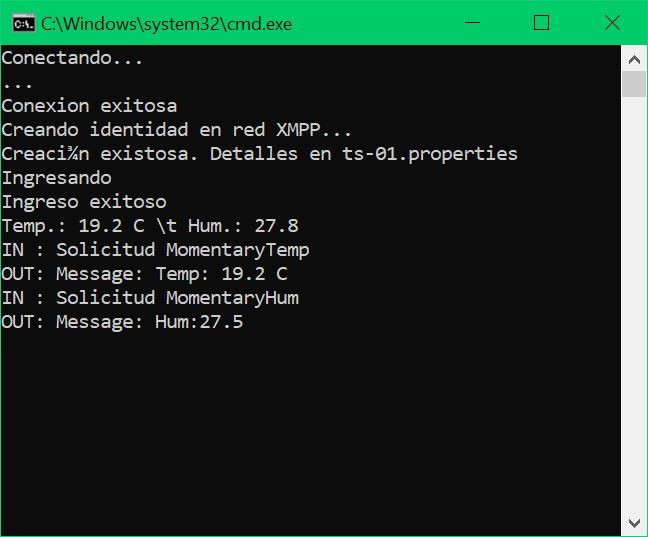
Ya realizada las modificaciones, el procedimiento es el siguiente:

1. Ingresar a la consola de administración web de *Openfire*.
2. Ingresar a la sección de registro.
3. Generar Credenciales de Consumidor, es decir, *ConsumerKey*, *Consumer Secret*.
4. Ingresar 3 como cantidad de creación de identidades permitida al *ConsumerKey*.

Para los clientes ingresamos el *ConsumerKey* y el *ConsumerSecret*, para que, de este modo, cuando se realice la conexión con el servidor, exista el intercambio de Stanzas con dataforms [5]. (está raro, no sé qué debería decir) este modo, se le delegue al cliente la facultad de crear identidades en el servidor XMPP Openfire. Así con el cliente TC–1 pudimos realizar consultas de (temperatura (?)) temperatura, humedad al cliente TS–1 y al gas a TS–2. Realizado esto, se creó y ejecuto el Bot malicioso en ambos servidores, tanto en el modificado como en el sin modificar. Los resultados se pueden apreciar en las siguientes Tablas:

Table 11: Test de penetración a servidor Openfire sin implementación

|  |  |  |
| --- | --- | --- |
| Tiempo (sec.) | Intentos de Creación por segundo | Identidades creadas exitosamente |
| 50 | 2 | 100 |
| 100 | 2 | 200 |
| 300 | 3 | 900 |



(a) Thing-Sensor 1 (b) Thing-Sensor 2

Figure 15: Retorno por consola de los clientes.

Table 12: Test de penetración a servidor Openfire implementado el XEP-0348

|  |  |  |
| --- | --- | --- |
| Tiempo (sec.) | Intentos de Creación por segundo | Identidades creadas exitosamente |
| 50 | 2 | 0 |
| 100 | 2 | 0 |
| 300 | 3 | 0 |

# 5. CONCLUSIONES

## 5.1. Post-Mortem

Con lo evidenciado en el presente documento, se puede dar crédito de que se logró llevar a cabo el desarrollo de un proyecto de larga duración, aplicando los conocimientos adquiridos a lo largo de toda la carrera de Ingeniería en Computación e Informática, a la par aprendiendo sobre la marcha acerca de cómo se deben llevar a cabo las metodologías en la realidad de un proyecto, con especial énfasis en los tiempos de estimación de tareas. Además, se pudo comprender la importancia de las buenas prácticas tanto en el desarrollo en el proyecto como del producto software, ya que, si no se lleva bien alguna de las tres, es muy fácil que se generen atrasos a las fechas propuestas en un inicio, como se apreció en más de un sprint del proyecto.

Se comprendió además la importancia del diseño y realización de una suite de pruebas, ya que debido a la falta de éstas durante el proyecto se generaron grandes retrasos al encontrar defectos. Además, se debe siempre tener en consideración la mantenibilidad del código, ya que siempre está la posibilidad de que una persona o equipo trabaje sobre el código de uno, y es importante que sea entendible y fácil de modificar para así generar un mejor ambiente en el ámbito del desarrollo de software. De igual manera es realmente importante, tener un sistema de control de la configuración, estable y seguro, para tener un respaldo del código fuente y las liberaciones correspondientes.

La mejora de seguridad a servidor XMPP Openfire y a la librería XMPP SMACK aporta al estado del arte, ya que se creo una solución que permite tanto a los usuarios intermediarios de la red XMPP-IoT como a los usuarios finales, registrar identidades en la red de forma segura y controlada. De esta forma se elimina la posibilidad que hackers puedan sobrecargar el servidor o la base de datos de este. De este modo, con la mejora de seguridad tanto, en el servidor Openfire y la librería SMACK estamos, brindando un punto de partida para crear sistemas de Internet de las Cosas de vanguardia.

## 5.2. Trabajos Futuros

Si bien, este proyecto ha sido un avance importante en cuanto al protocolo de comunicaciones XMPP en el contexto Internet de las Cosas, es sólo la punta iceberg en cuanto al potencial que posee el protocolo. Ya que, existen varias especificaciones que tributan en esta área. En la Tabla 17 se aprecia lod XEPs, que han sido desarrollados específicamente para los desafíos que plantea el Internet de las cosas.

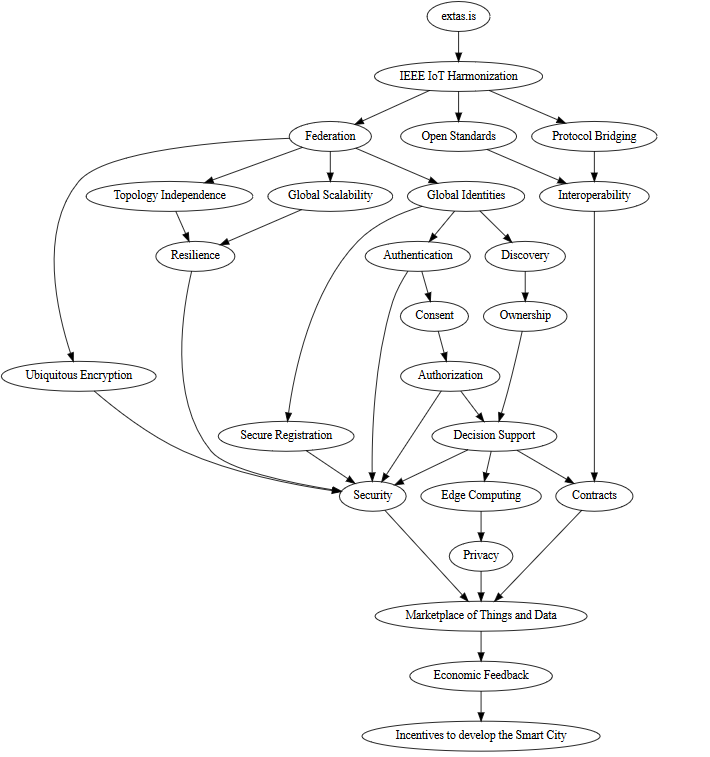
Tabla 23 XEPs IoT

|  |  |  |
| --- | --- | --- |
| Numero | Nombre | Descripción |
| XEP-0323 | IoT Sensor Data | Proporciona la arquitectura subyacente, las operaciones básicas y las estructuras de datos para la comunicación de datos del sensor a través de redes XMPP. Incluye un modelo de abstracción de hardware, eliminando cualquier detalle técnico implementado en tecnologías subyacentes. Este XEP es utilizado por todos los demás XEPs de redes de sensores. |
| XEP-0324 | IoT Provisioning & Security | Esta especificación Define cómo el aprovisionamiento, la administración de privilegios de acceso, etc., pueden implementarse de manera fácil y eficiente. |
| XEP-0325 | IoT Control | Define cómo controlar actuadores y otros dispositivos en redes de sensores. |
| XEP-0326 | IoT Concentrators | Define cómo manejar arquitecturas que contienen concentradores o servidores que manejan múltiples sensores. |
| XEP-0331 | Data Forms - Color Field Types | Define extensiones de cómo se pueden manejar los parámetros de color, en función de los Data Forms. |
| XEP-0336 | Data Forms - Dynamic Forms | Define extensiones de cómo se pueden crear formularios dinámicos, en función de los Data Forms Validation, Publishing Stream Initiation Requests y Data Forms Layout. |
| XEP-0347 | IoT Discovery | Define los aspectos peculiares del descubrimiento del sensor en redes de sensores. Además de descubrir sensores por JID, también define cómo descubrir sensores en función de la ubicación, etc. |
| XEP-0000 | IoT Events | Define cómo los sensores envían eventos, cómo se configuran la suscripción al evento, los niveles de histéresis, etc. |
| XEP-0000 | IoT Interoperability | Define directrices sobre cómo lograr la interoperabilidad en redes de sensores, publicando interfaces de interoperabilidad para diferentes tipos de dispositivos. |
| XEP-0000 | IoT Battery Powered Sensors | Define cómo manejar los peculiares relacionados con los dispositivos alimentados por batería y otros dispositivos disponibles intermitentemente en la red. |
| XEP-0000 | IoT Multicast | Define cómo los datos del sensor pueden ser multidifusión de maneras eficientes. |
| XEP-0000 | IoT PubSub | Define qué tan eficiente puede hacerse la publicación de datos de sensores en Internet of Things. |
| XEP-0000 | IoT Chat | Define cómo las interfaces de persona a máquina deben construirse usando mensajes de chat para que sean fáciles de usar, automatizables y consistentes con otras extensiones de IoT y arquitectura subyacente posible. |

Por lo tanto, queda bastante por desarrollar para crear un servidor XMPP-IoT robusto. Si bien, en la actualidad este servidor posee el mecanismo de registro en banda, para así tener la posibilidad de incorporar dispositivos «plug and play», también queda la labor de desarrollar un mecanismo de delegación de responsabilidades, registro de cosas, un sistema de aprovisionamiento, etceterá.

Hoy se están creando la gran especificación titulada por «IEEE 1451-99 - Standard for Harmonization of Internet of Things (IoT) Devices and Systems» por el equipo de «Devices and Systems Harmonization Working Group» que consiste la creación descriptiva de un ecosistema IoT seguro, interoperable, con identidades globales, registro seguro, rentable – para todos – y un más características que se pueden observar en la Ilustración 9. Cabe destacar el ítem en la ilustración que lleva el título «Secure Registration», que describe exactamente lo que hemos realizado en el presente trabajo.

Ilustración 9 IoT Harmonization of Internet of Things



# Referencias

IETF. (27 de Septiembre de 2016). *RFC 3268 - The Transport Layer Security (TLS) Protocol*. Obtenido de https://tools.ietf.org/html/rfc5246

IETF. (28 de Septiembre de 2016). *RFC 3920 - Extensible Messaging and Presence Protocol (XMPP): Core*. Obtenido de IETF: https://www.ietf.org/rfc/rfc3920.txt

IETF. (28 de Septiembre de 2016). *RFC 3921 - Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence*. Obtenido de IETF: https://www.ietf.org/rfc/rfc3921.txt

IETF. (2016 de Septiembre de 2016). *RFC 5790 - Use of Transport Layer Security (TLS) in the Extensible Messaging and Presence Protocol (XMPP)*. Obtenido de IETF: https://tools.ietf.org/html/rfc7590

IETF. (28 de Septiembre de 2016). *RFC 6120 - Extensible Messaging and Presence Protocol (XMPP): Core*. Obtenido de IETF: https://tools.ietf.org/html/rfc6120

IETF. (28 de Septiembre de 2016). *RFC 6121 - Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence*. Obtenido de IETF: https://tools.ietf.org/html/rfc6121

IETF. (27 de Septiembre de 2016). *RFC 7395 - An Extensible Messaging and Presence Protocol (XMPP) Subprotocol for WebSocket*. Obtenido de IETF: https://tools.ietf.org/html/rfc7395

IETF. (28 de Septiembre de 2016). *RFC 7622 - Extensible Messaging and Presence Protocol (XMPP): Address Format*. Obtenido de IETF: https://tools.ietf.org/html/rfc7622

Pressman, R. S. (2010). *Ingenería del Software. Un efoque practico* (Septima ed.). Mexico D.F.: The McGraw-Hill.

Project Management Institute. (2009). *Guia de los Fundamentos Para la Direccion de Proyectos* (Cuarta ed.). Project Management Institute.

Waher, P. (2016). *Learning Internet of Things.* Birmingham: Packt Publishing.

XSF. (26 de 09 de 2016). *XEP-0001: XMPP Extension Protocols*. Obtenido de XMPP : http://xmpp.org/extensions/xep-0001.html

XSF. (28 de Septiembre de 2016). *XEP-0004: Data Forms*. Obtenido de xmpp.org: http://xmpp.org/extensions/xep-0004.html

XSF. (28 de Septiembre de 2016). *XEP-0045: Multi-User Chat*. Obtenido de xmmp.org: http://xmpp.org/extensions/xep-0045.html

XSF. (28 de Septiembre de 2016). *XEP-0077: In-Band Registration*. Obtenido de xmpp.org: http://xmpp.org/extensions/xep-0077.html

XSF. (27 de Septiembre de 2016). *XEP-0143: Guidelines for Authors of XMPP Extension Protocols*. Obtenido de xmmp.org: http://xmpp.org/extensions/xep-0143.html

XSF. (28 de Septiembre de 2016). *XEP-0348: Signing Forms*. Obtenido de xmpp.org: http://xmpp.org/extensions/xep-0348.html

1. Online: En línea [↑](#footnote-ref-1)
2. Offline: Fuera de línea [↑](#footnote-ref-2)
3. Busy: Ocupado [↑](#footnote-ref-3)
4. Extensible Messaging and Presence Protocol (XMPP): Core <https://tools.ietf.org/html/rfc6120> [↑](#footnote-ref-4)
5. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <https://tools.ietf.org/html/rfc6121> [↑](#footnote-ref-5)
6. Extensible Messaging and Presence Protocol (XMPP): Core <https://www.ietf.org/rfc/rfc3920.txt> [↑](#footnote-ref-6)
7. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <https://www.ietf.org/rfc/rfc3921.txt> [↑](#footnote-ref-7)
8. Extensible Messaging and Presence Protocol (XMPP): Core <https://tools.ietf.org/html/rfc6120> [↑](#footnote-ref-8)
9. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <https://tools.ietf.org/html/rfc6121> [↑](#footnote-ref-9)
10. XEP-0045: Multi-User Chat <http://xmpp.org/extensions/xep-0045.html> [↑](#footnote-ref-10)
11. An XMPP Subprotocol for WebSocket<https://tools.ietf.org/html/rfc7395> [↑](#footnote-ref-11)
12. Use of Transport Layer Security (TLS) in the XMPP <https://tools.ietf.org/html/rfc7590> [↑](#footnote-ref-12)
13. Extensible Messaging and Presence Protocol (XMPP): Address Format <https://tools.ietf.org/html/rfc7622> [↑](#footnote-ref-13)
14. XEP-0001: XMPP Extension Protocols <http://xmpp.org/extensions/xep-0001.html> [↑](#footnote-ref-14)
15. XEP-0143: Guidelines for Authors of XMPP Extension Protocols <http://xmpp.org/extensions/xep-0143.html> [↑](#footnote-ref-15)
16. SASL: Simple Authentication and Security Layer. [↑](#footnote-ref-16)
17. TLS: The Transport Layer Security. [↑](#footnote-ref-17)
18. SSL: Secure Sockets Layer. [↑](#footnote-ref-18)
19. RFC4422: SASL <https://tools.ietf.org/html/rfc4422> [↑](#footnote-ref-19)
20. RFC5246: TLS <https://tools.ietf.org/html/rfc5246> [↑](#footnote-ref-20)
21. XEP-0322 <http://xmpp.org/extensions/xep-0322.html> [↑](#footnote-ref-21)
22. XEP-0323 <http://xmpp.org/extensions/xep-0323.html> [↑](#footnote-ref-22)
23. XEP-0324 <http://xmpp.org/extensions/xep-0324.html> [↑](#footnote-ref-23)
24. XEP-0325 <http://xmpp.org/extensions/xep-0325.html> [↑](#footnote-ref-24)
25. XEP-0326 <http://xmpp.org/extensions/xep-0326.html> [↑](#footnote-ref-25)
26. The OAuth 1.0 Protocol <https://tools.ietf.org/html/rfc5849> [↑](#footnote-ref-26)
27. XEP-0348: Signing Forms <http://xmpp.org/extensions/xep-0348.html> [↑](#footnote-ref-27)
28. XEP-0077: In-Band Registration <http://xmpp.org/extensions/xep-0077.html> [↑](#footnote-ref-28)