# A.1: Desarrollo de los SPRINTS

## A.1.1 Sprint 1

Los Sprints anteriores, es decir, el 0a, 0b, y 0c consistieron en su mayoría en mitigaciones de riegos, de estas mayormente técnico, particularmente relacionados con conocimientos electrónicos, como también el conocimiento del protocolo de comunicaciones XMPP. Ya con mayor conocimiento electrónico, y sobre el protocolo XMPP podemos comenzar a desarrollar las modificaciones a «Openfire» y la codificación de la extensión para la librería «Smack».

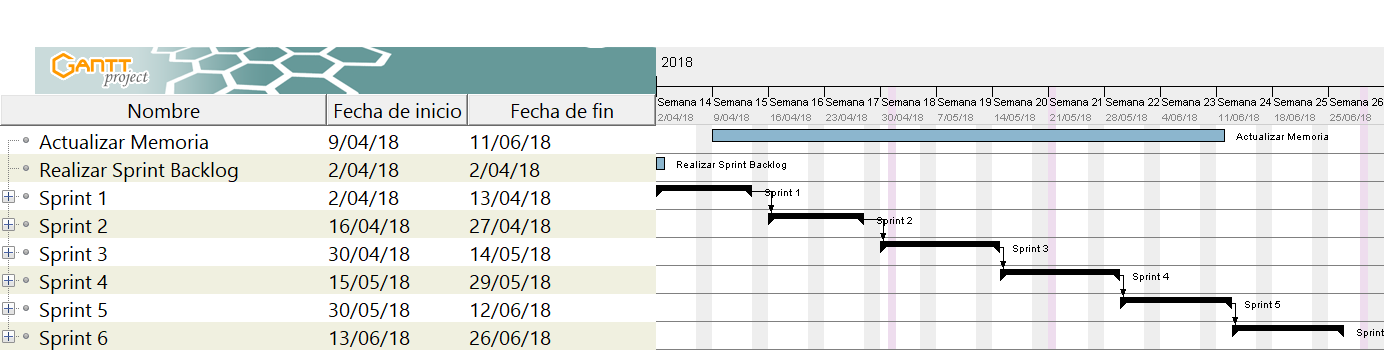
Durante este sprint, se comenzó la modificación de servidor XMPP Openfire.  
Los plazos del sprint tuvieron un leve retraso debido a que se gatillaron los riegos OF0348-RISK-04 y el OF0348-RISK-05, los cuales tienen que ver con estimaciones de tiempo, y disponibilidad de equipo de trabajo, por ende, se aplicaron las medidas de contingencia, es decir, se inyectaron mas horas hombre para este fin.

La reunión «Sprint Review» no se pudo realizar. Si bien se realizaron las pruebas de aceptación, pero no las realizo el «Product Owner». Esto se realizó, debido al profundo conocimiento del problema de equipo de testing.

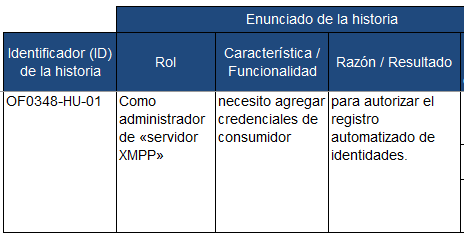
### A.1.1.1 Planificación

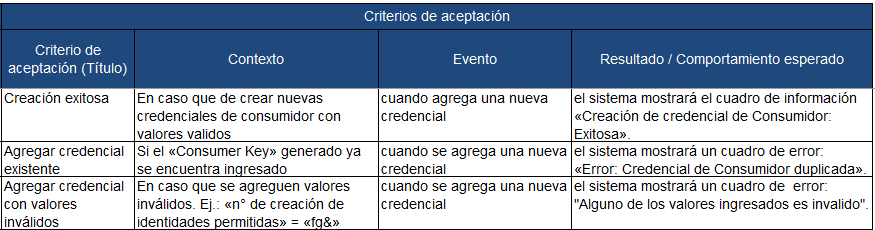
La planificación será mediante una carta Gantt. Obedeciendo así la metodología de gestión del proyecto. Realizaremos tres Sprint para llegar a la culminación del proyecto, generando el producto de software esperado. Debemos mencionar que previo al desarrollo de los Sprint, se confeccionó un «Product Backlog» en donde se añaden todas las tareas a realizar en el presente proyecto, las cuales se agruparán en tres conjuntos, donde cada uno de estos se asigna a un «Sprint».

A continuación, en la imagen se puede ver el grafico de la carta Gantt de planificación:

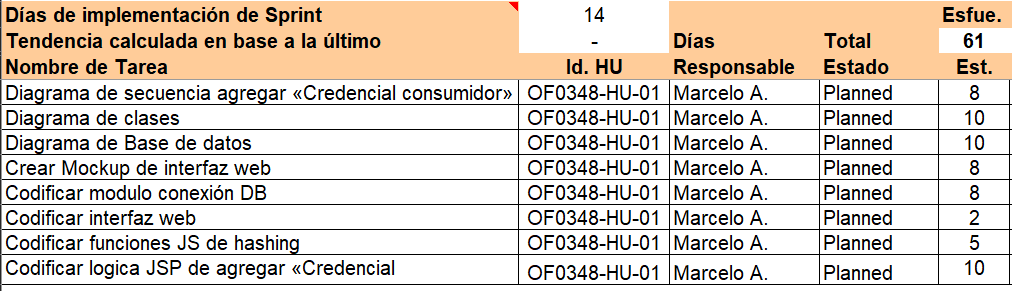


Como se ha mencionado anteriormente, se ha generado un «Product Backlog», el cual, contiene las Historias de Usuario a realizar durante el proyecto, este se puede ver en la imagen XXXX (Imagen product Backlog). A continuación, veremos las Historias de Usuario seleccionadas para realizar durante el presente Sprint.

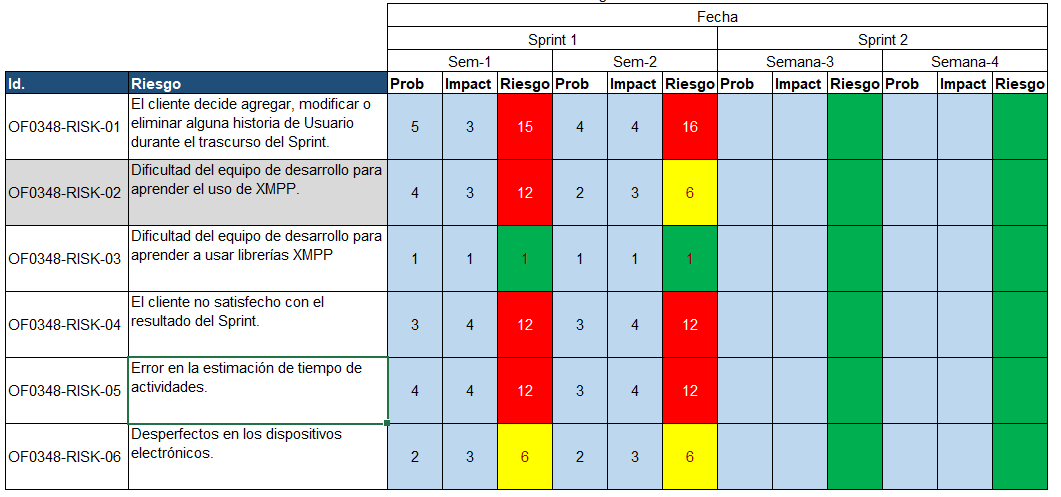


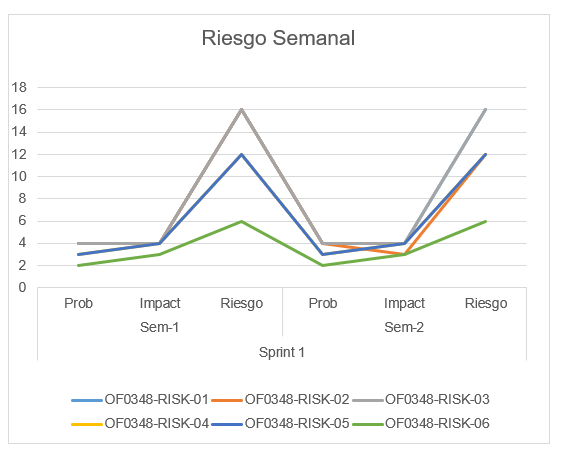


Ya seleccionadas las Historias de Usuario a desarrollar, las desglosamos en taras en la tabla XXXXX , a continuación.



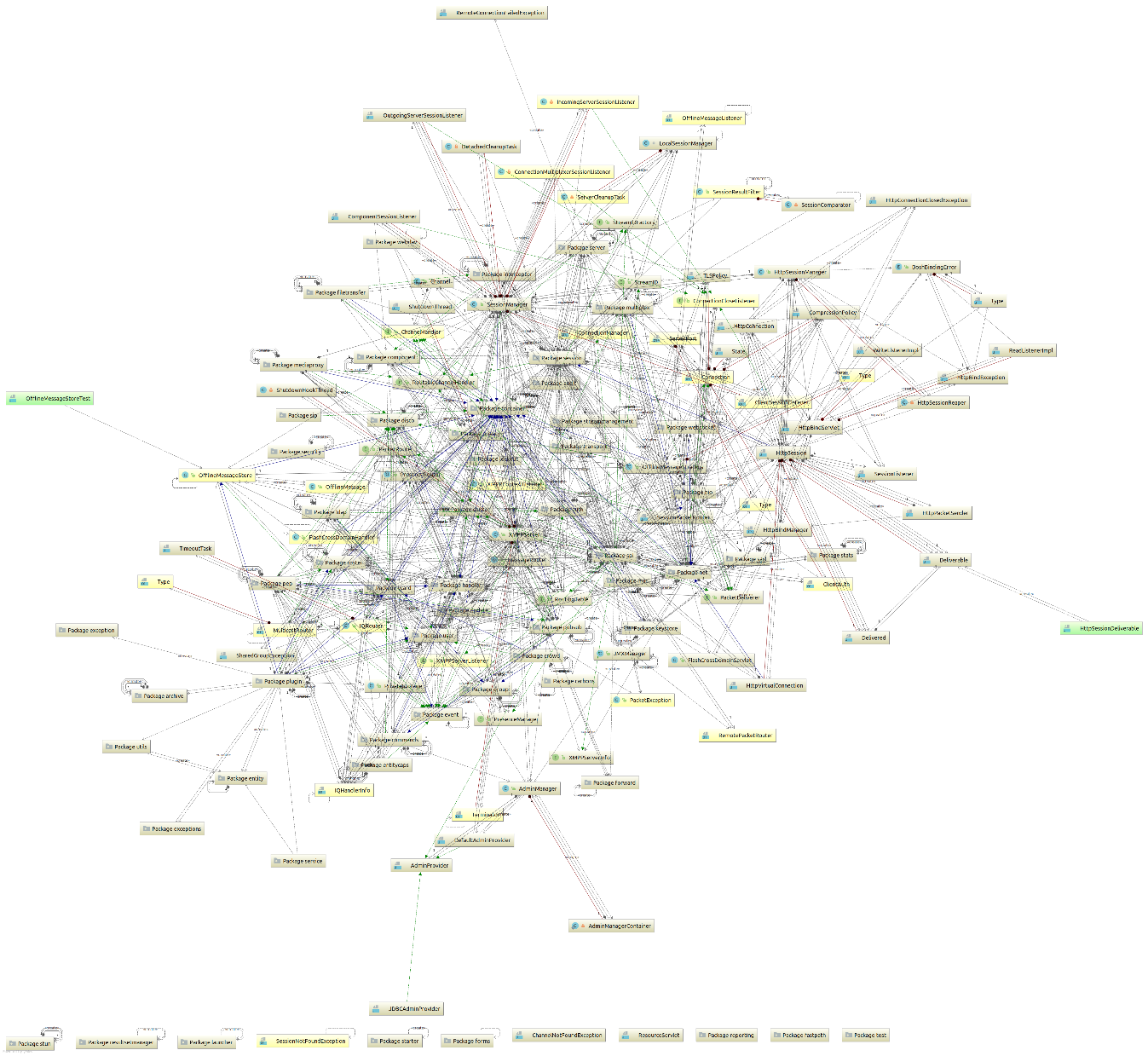
A continuación, en la Tabla XXXX, se detallan los riesgos identificados para el Sprint, y seguido el Diagrama XXXX con la evolución de estos.



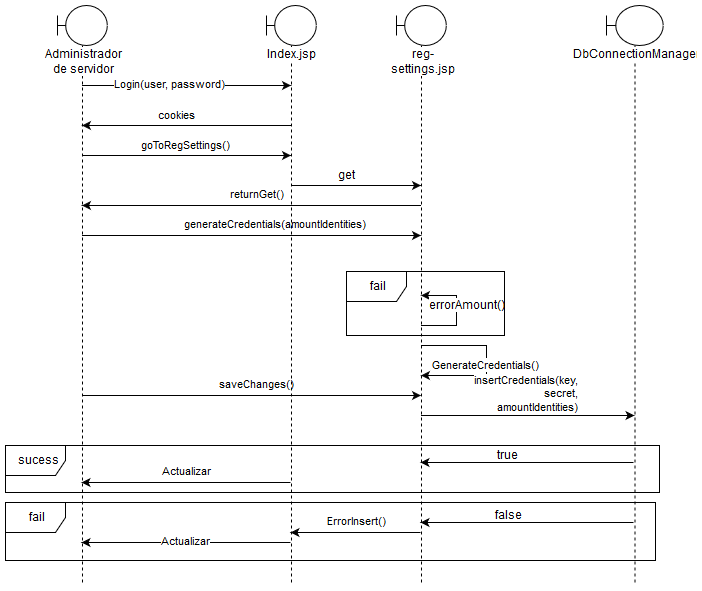


### A.1.1.2 Diseño

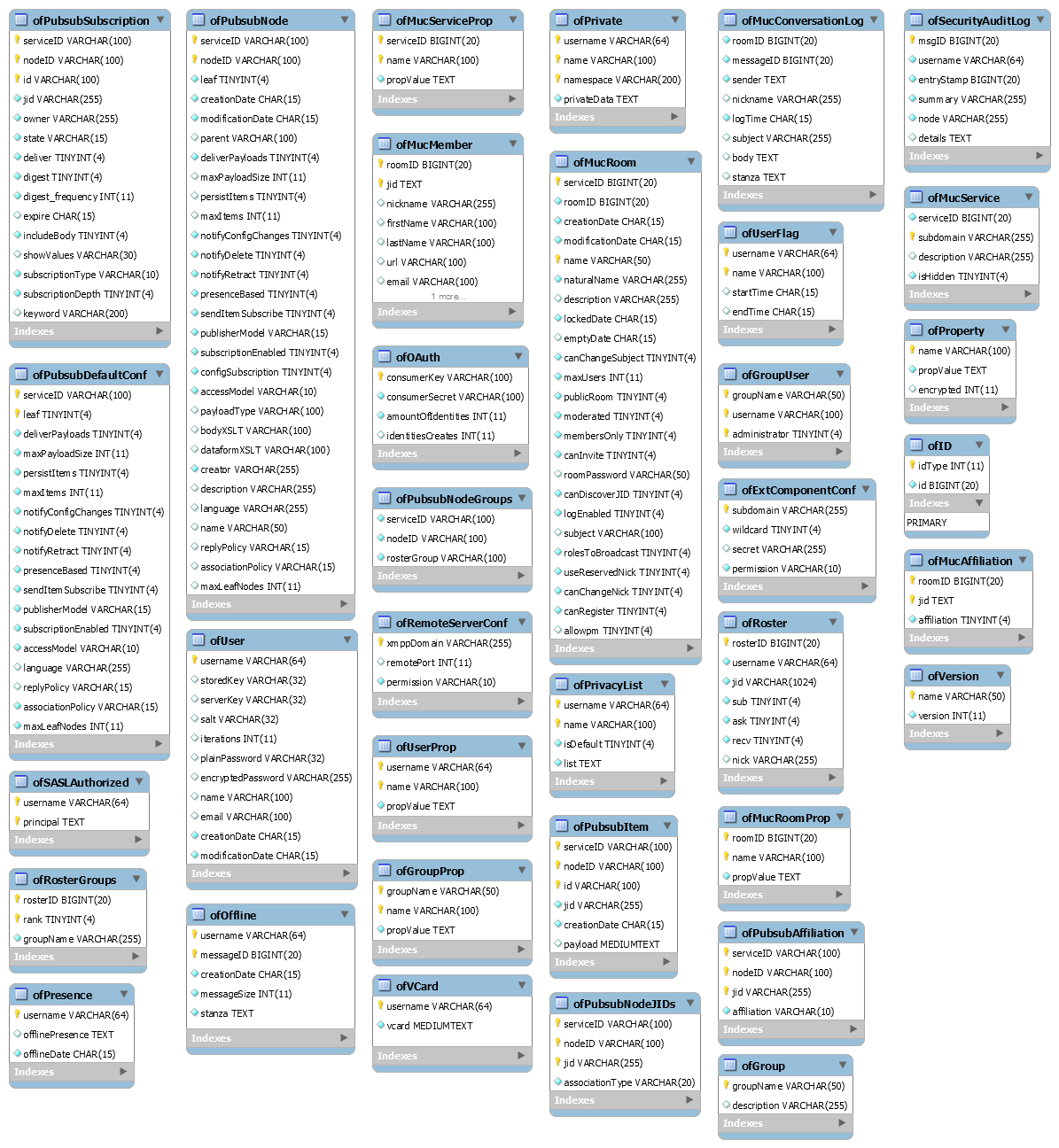
A partir de la «XEP-0348: Signing Forms» en conjunto con la «XEP-0077: Registration In-Band», se procedió a identificar las entidades que interactúan en el proceso de registro de nuevas cuentas en el servidor, las cuales se grafican mediante un diagrama de clases el cual está en la Imagen X, remarcando las clases que serán intervenidas.



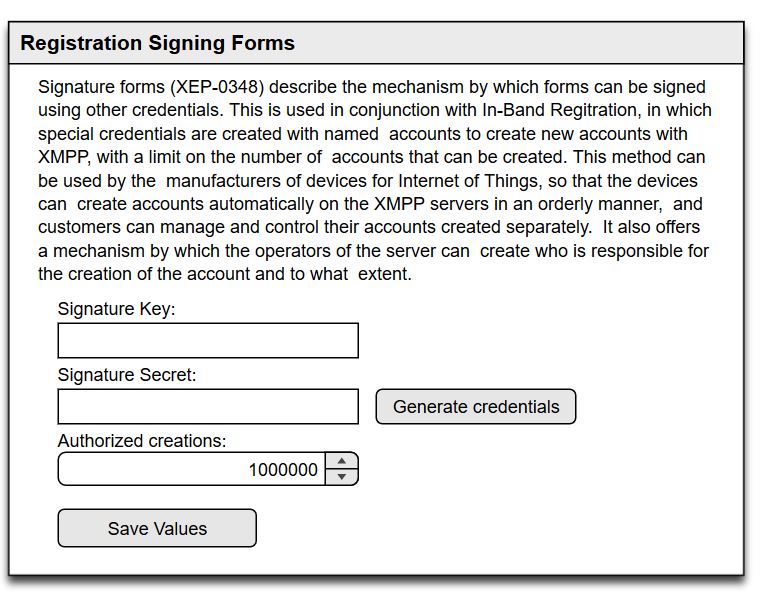
Ya identificadas las clases identificadas en el proceso, se realiza un diagrama de secuencia, que se puede apreciar en el diagrama XXXX



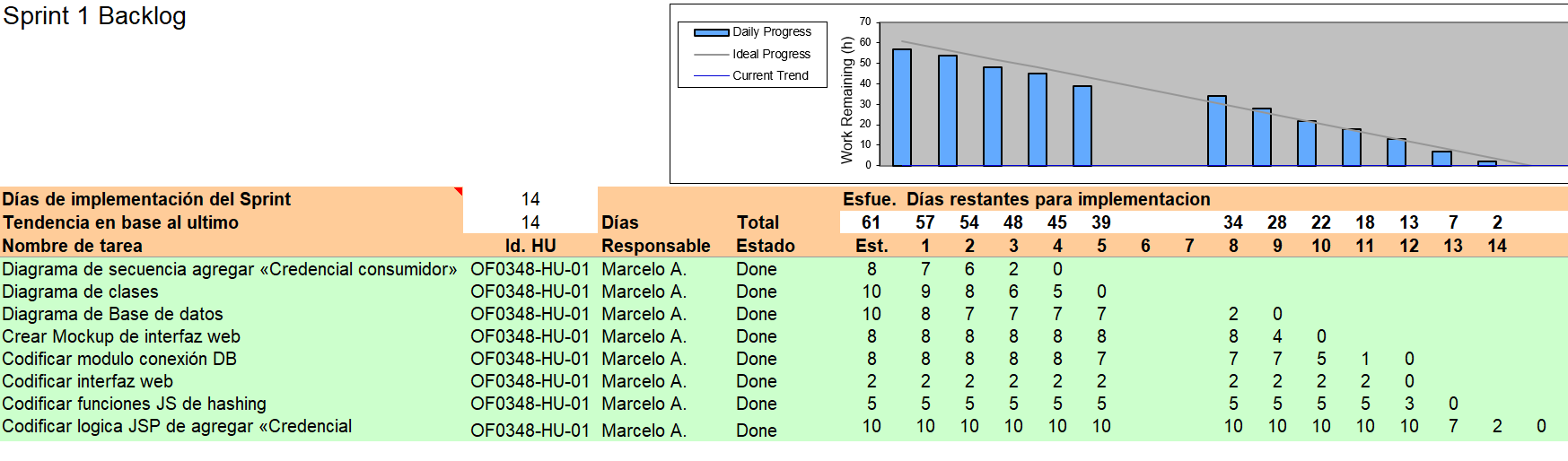
En cuanto al diseño de la base de datos, se explica mediante el siguiente diagrama de Entidad-Relación mostrado a continuación, en el grafico XXXXX



También se realizó un mockup de las modificaciones al front-end de «Openfire» para agregar la funcionalidad de agregar «Credenciales de Consumidor», a modo de prototipo no funcional, con el fin de que se puedan apreciar futuros cambios. El mockup se puede ver en el diagrama XXXXX



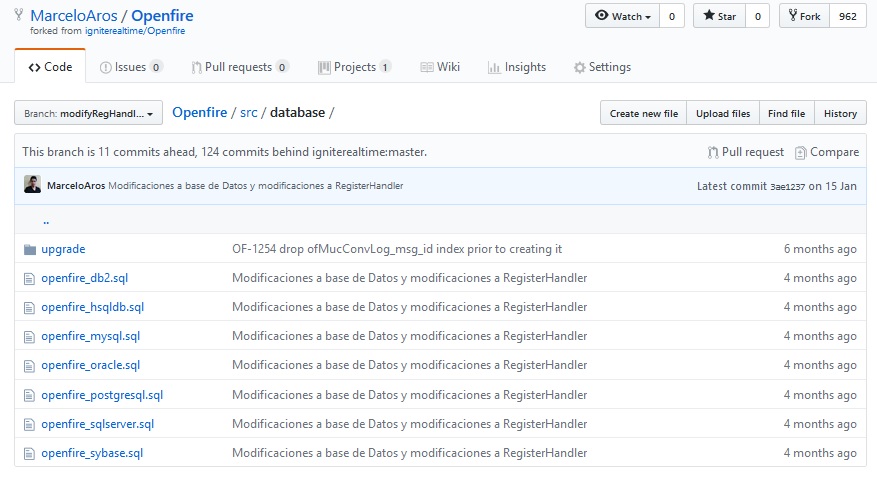
### A.1.1.3 Resultados

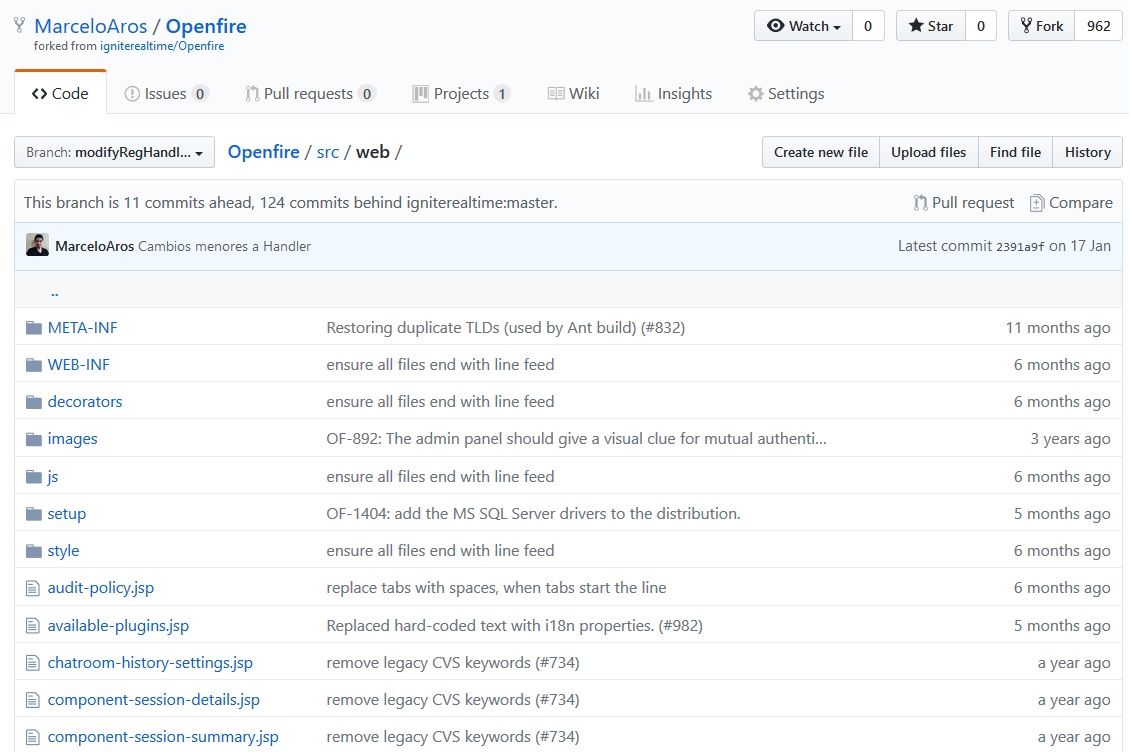


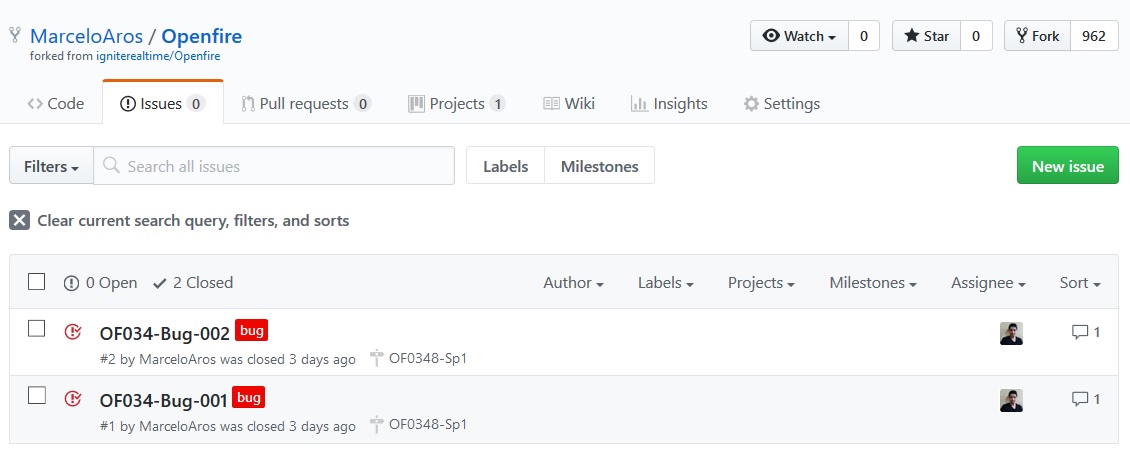
### A.1.1.4 Evidencia

A continuación, se aprecia la evidencia en Github, tanto de la documentación del proyecto, como el proyecto mismo.

En el trascurso del desarrollo del Sprint 1 se generaron 2 incidencias, estas tienen relación con las validaciones. La primera obedece al fallo al validar una cantidad de identidades permitida para una «credencial de consumidor». La segunda al permitir ingreso de credenciales de consumidor, sin haber sido generadas con anterioridad.







### A.1.1.5 Post-Mortem

**Problemas resueltos**

* El servidor «Openfire» es capaz de incorporar «Consumer Key», «Secret Key», cantidad de creación de identidades permitas, cantidad de creación de identidades utilizadas.

**Problemas futuros**

* Si bien las credenciales son agregadas correctamente en la base de datos del servidor «Openfire», mediante la consola de administración web, no es posible visualizar estos datos, ni llevar un registro y control sobre cuantas y que identidades ha creado un «Consumer Key». Esto se realiza en una de las historias de usuario asignadas al proyecto OF0348.

**Lecciones aprendidas**

* Con mejores estimaciones de tiempo y mejor manejo de los riesgos es mucho más factible cumplir con los plazos establecidos.

**Métricas**

* KLOC: Se finalizo este Sprint con 1.2 KLOC
* Cantidad de defectos: 2 defectos encontrados durante el desarrollo del sprint, todos solucionados, no quedó defecto pendiente de ser resuelto.
* Efectividad de pruebas: Se realizaron 3 pruebas de aceptación, las cuales pasaron.

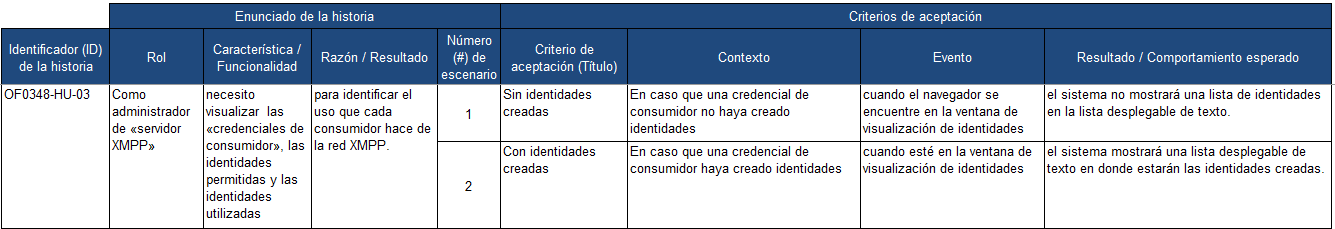
## A.1.2 Sprint 2

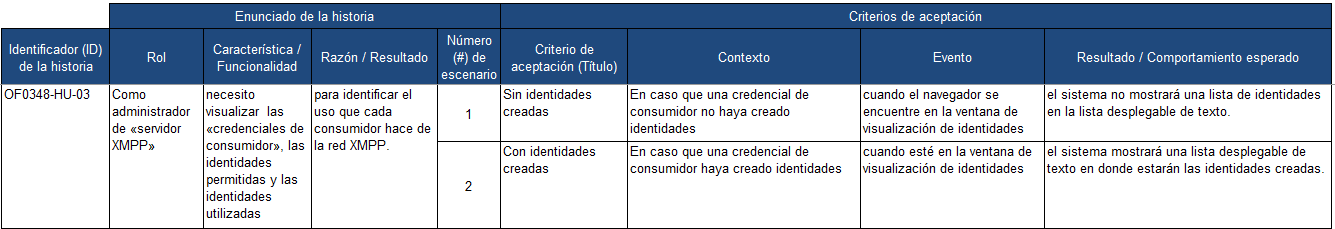
### A.1.2.1 Planificación

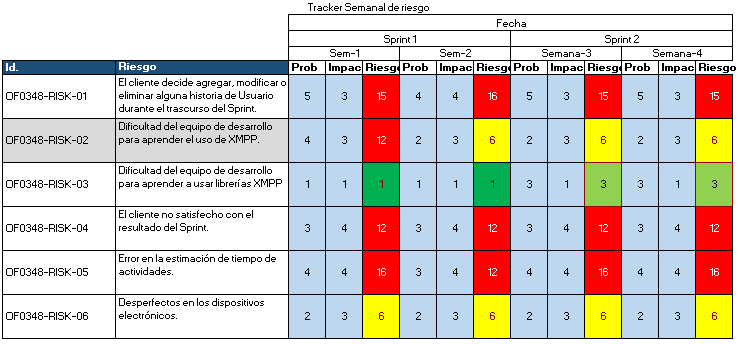
En el Sprint anterior se desarrollo el modulo de generación y almacenamiento de credenciales de consumidor con sus respectivas variables asociadas, es decir, cantidad de creación de identidades permitidas y cantidad de identidades actualmente creadas.

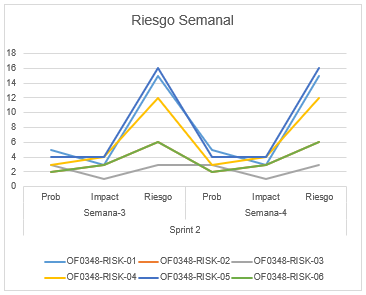
Siguiendo la planificación de carta Gantt entramos en plazo preestablecido al sprint 2.

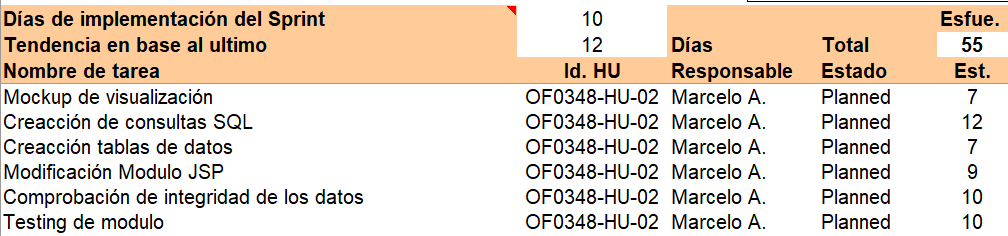
Durante el Sprint, se continuó añadiendo funcionalidades al servidor XMPP «Openfire», en particular el desarrollo de la historia de usuario OF0348-HU-03, la cual, hace uso de los datos almacenado en el desarrollo del sprint anterior, esto es, para finalmente visualizarlos de forma ordenada y coherente a lo almacenado, con la variable «oauth\_secret\_key» oculta. Se gatillaron los riegos OF0348-RISK-05, debido a la falta de experiencia con el desarrollo de «JavaServer Pages», por ende, se aplicó la medida de contingencia para dicho riesgo, la cual fue, inyectar más horas hombre.







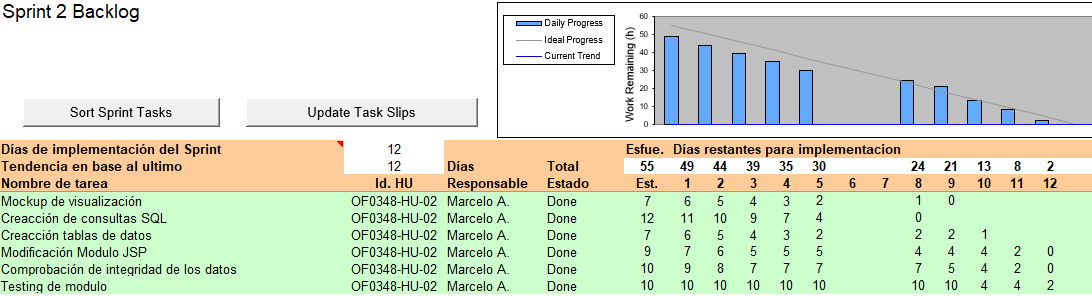




### A.1.2.2 Diseño

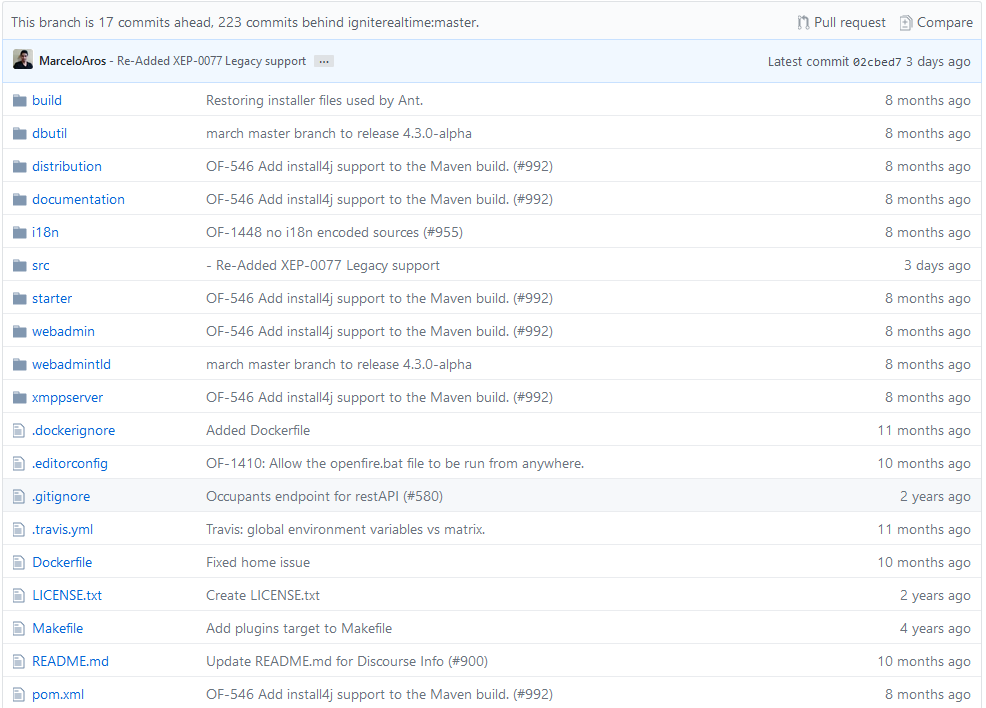
En el primer Sprint se logró identificar las clases involucradas en el desarrollo OP0348. En este Sprint se modificó la clase «IqRegisterHandler.class» y «reg-settings», como también se modificó los scripts iniciales de creación de las bases de datos.

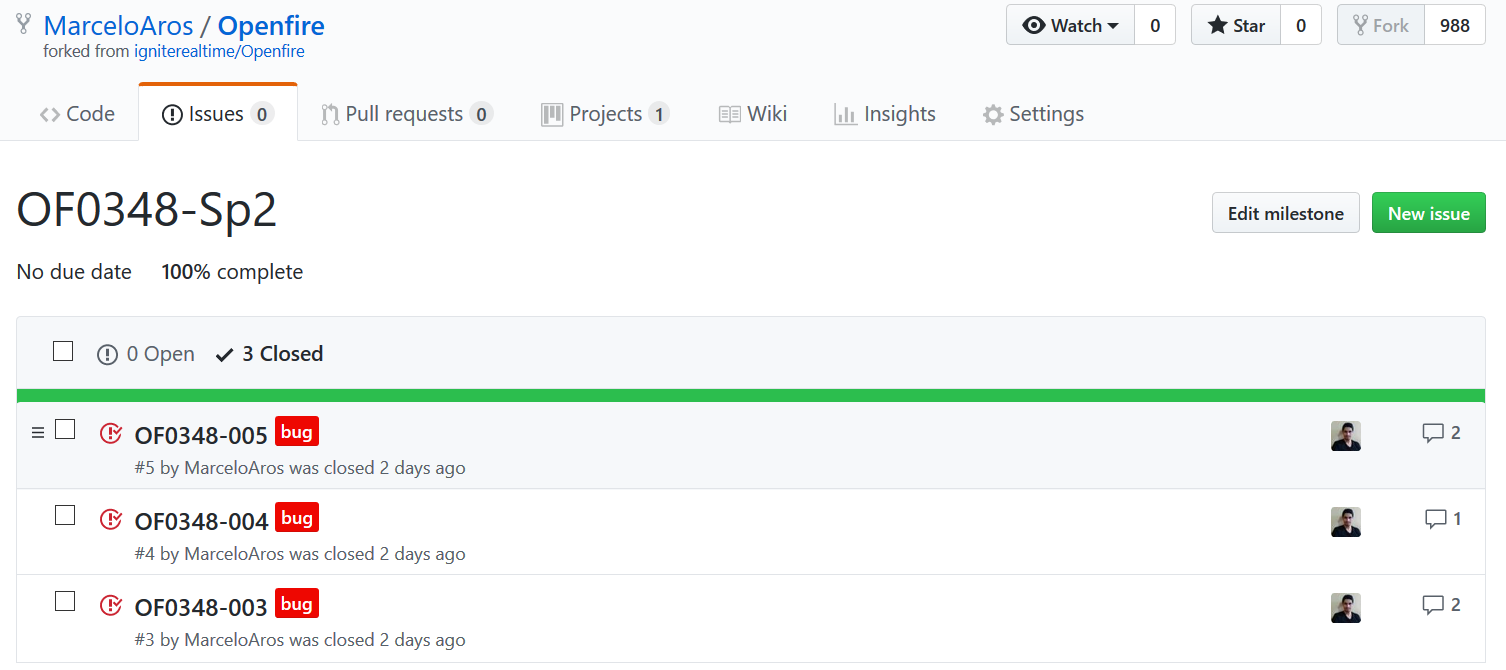
### A.1.2.3 Resultados

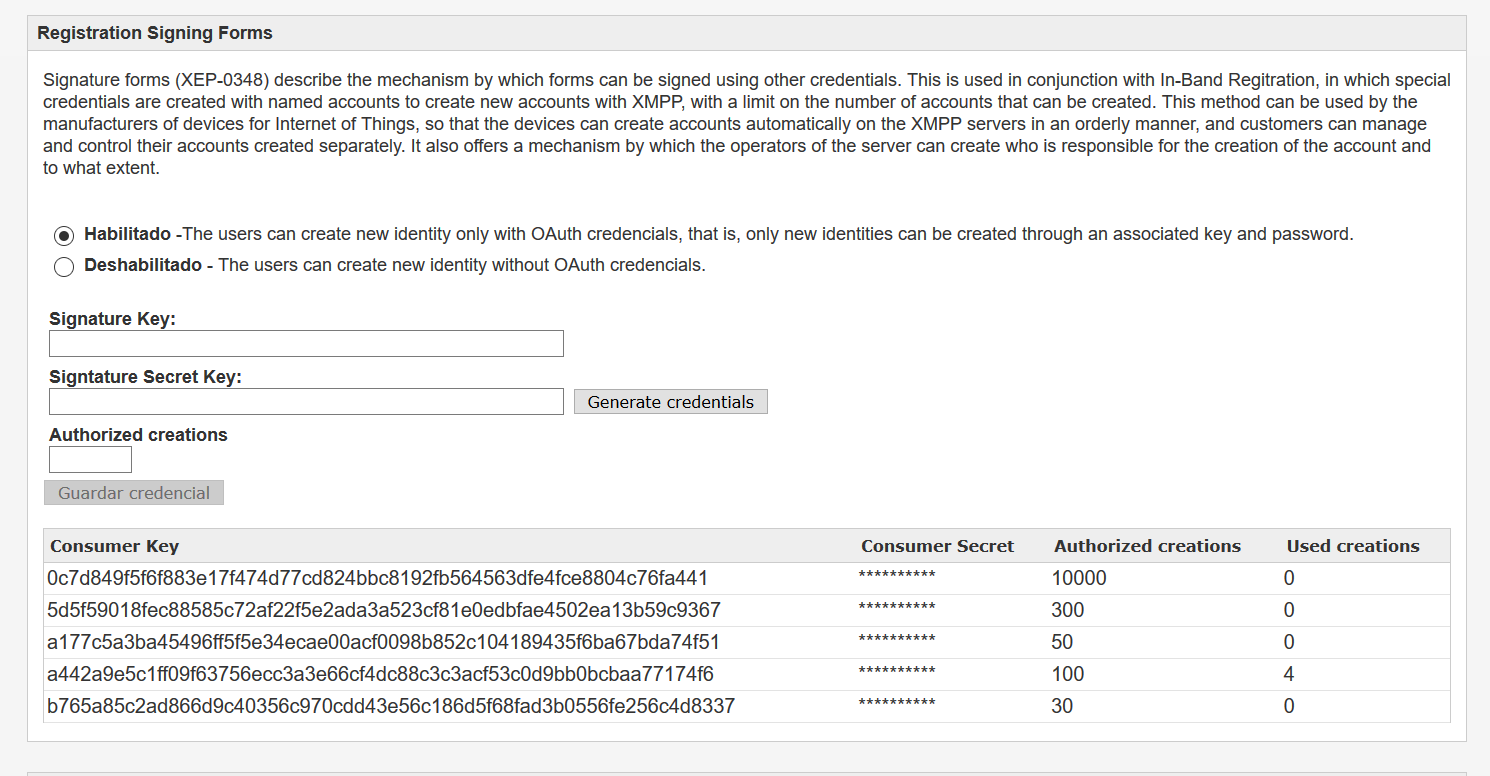


### A.1.2.4 Evidencia

La evidencia del código fuente se puede revisar en Github con los todos los commits realizados a la fecha.







### A.1.2.5 Post-Mortem

**Problemas resueltos**

* El servidor XMPP «Openfire» es capaz de visualizar las credenciales de consumidor que posee registradas, de este modo, se sabe cuantas identidades potencialmente se podrían registrar en la red XMPP-IoT.

**Problemas futuros**

* Si bien, en este momento se pueden ingresar muchas credenciales de consumidor al ambiente del servidor Openfire, no estamos aportando un valor agregado, ya que falta el grueso de la implementación que es la lógica detrás del registro de nuevas identidades, la cual debiese que seleccionar el «Product Owner» en la próxima reunión «Sprint Planning».

**Lecciones aprendidas**

* Aún se continua con problemas de estimación de tiempo, en este caso, se supuso que, debido a la aparente simpleza de la HU, iba a ser un desarrollo simple, sin embargo, el equipo de desarrollo tuvo grandes inconvenientes con JavaServer Pages, ya que no se adelanto el trabajo con esta tecnología, pero se asume que es una de las consecuencias de trabajar con una metodología ágil.

**Métricas**

* KLOC: Se finalizo este Sprint con 1012 LOC.
* Cantidad de defectos: Se detectaron 3 bug en etapas tempranas de desarrollo.
* Efectividad en pruebas: Se realizaron dos pruebas de aceptación las cuales pasaron.

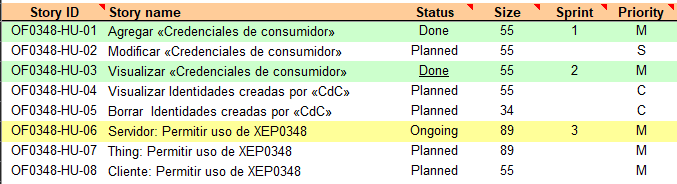
## A.1.3 Sprint 3

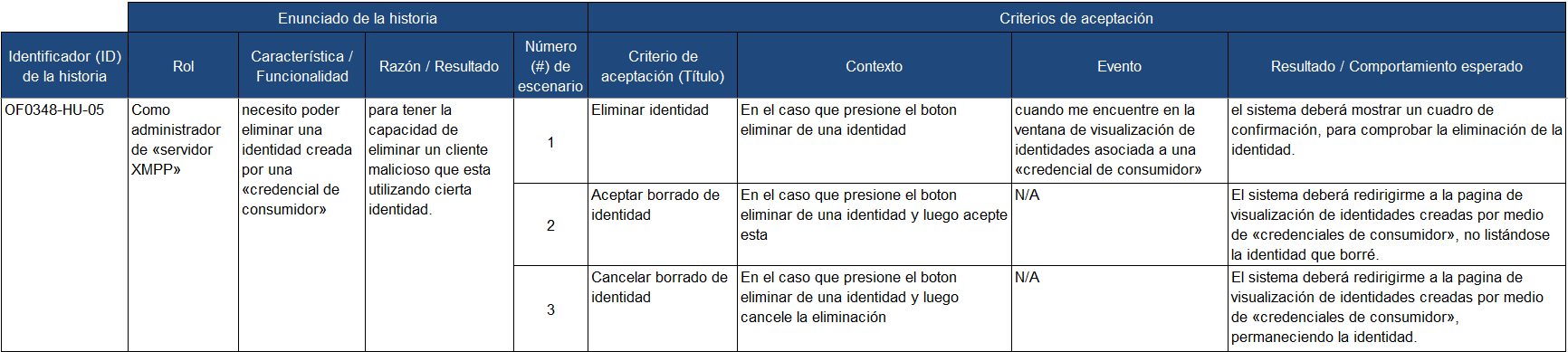
Luego de desarrollar la generación, almacenamiento y visualización de credenciales de consumidor.

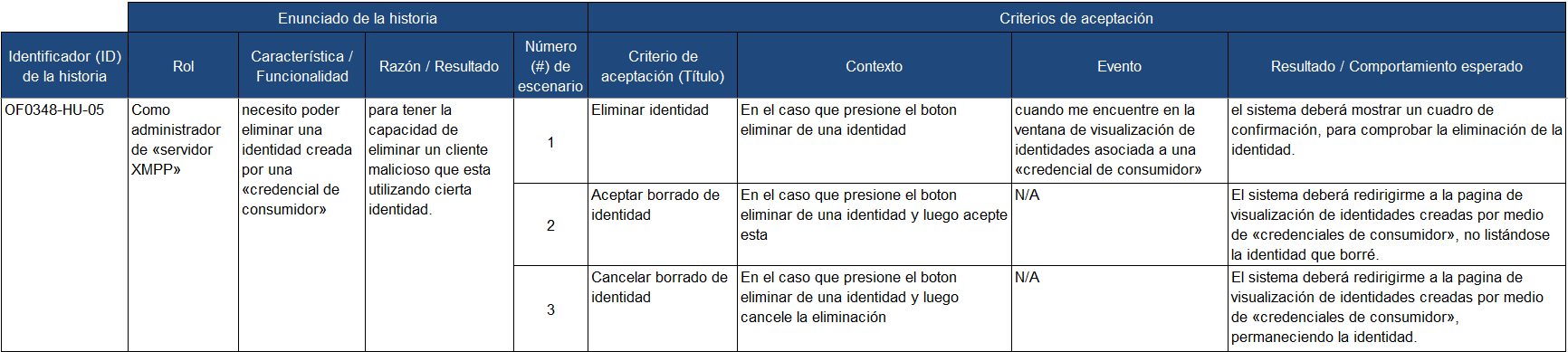
Durante este Sprint se procedió a dar inicio al desarrollo de la interacción lógica con clientes XMPP, es decir, el servidor Openfire crea conexiones a través de HTTPS para así enviar mediante «stanzas» XML, de acuerdo con lo establecido en el core de XMPP, datos de registro a los clientes.

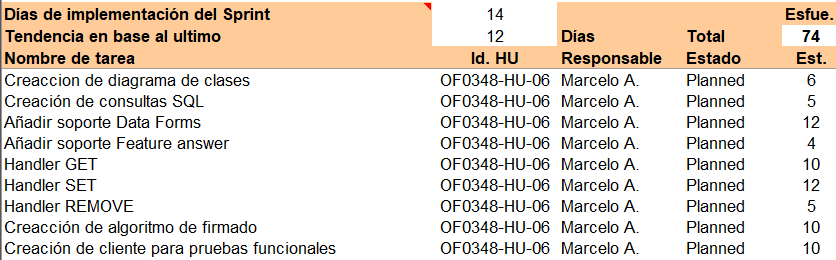
### A.1.3.1 Planificación

De acuerdo con la reunión de planificación «Sprint Planning» en la cual se seleccionó la OP0348-HU-06 para realizar con peso en números Fibonacci 89.









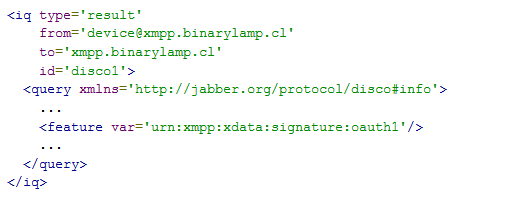
### A.1.3.2 Diseño

En este caso, todo el esfuerzo de desarrollo se centró en «IqRegisterHandler.class» que como el nombre dice es el encargado de capturar las solicitudes de creación de identidades de mediante el protocolo de extensión XMPP 0077. El gran problema al realizar el rediseño de esta clase fue su alto grado de complejidad, tanto así que es la segunda clase dentro de todas las que posee el servidor XMPP Openfire con mayor valor de complejidad ciclomática, con un , lo cual es sumamente alto, teniendo en cuenta para que una clase de muy alto riesgo el valor es de 50, y en este caso estamos, por poco, duplicando ese valor.

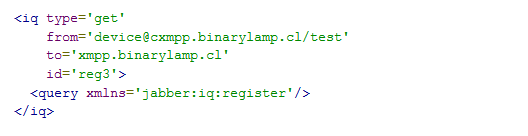
El servidor deberá preguntar por las «Features» por medio la consulta:



Y luego deberá capturar la respuesta del cliente, que estará conformada de acuerdo con la siguiente estructura:



Luego de que se valida que el cliente, tanto como el servidor soportan el XEP-0348, el cliente envía la «stanza» en la siguiente imagen:



Luego el host responde con la «stanza» con el formulario de registro en donde le solicita únicamente que el cliente complete cuatro valores, «username», «password», «oauth\_consumer\_key», y «oauth\_signature». También deja una serie de valores como opcionales, por ejemplo: «email», «first», «last», y el host envía envía «FORM\_TYPE», «oauth\_version», «oauth\_signature\_method», «oauth\_token», «oauth\_token\_secret», «oauth\_nonce» y «oauth\_timestamp». La «stanza» se puede ver a continuación:



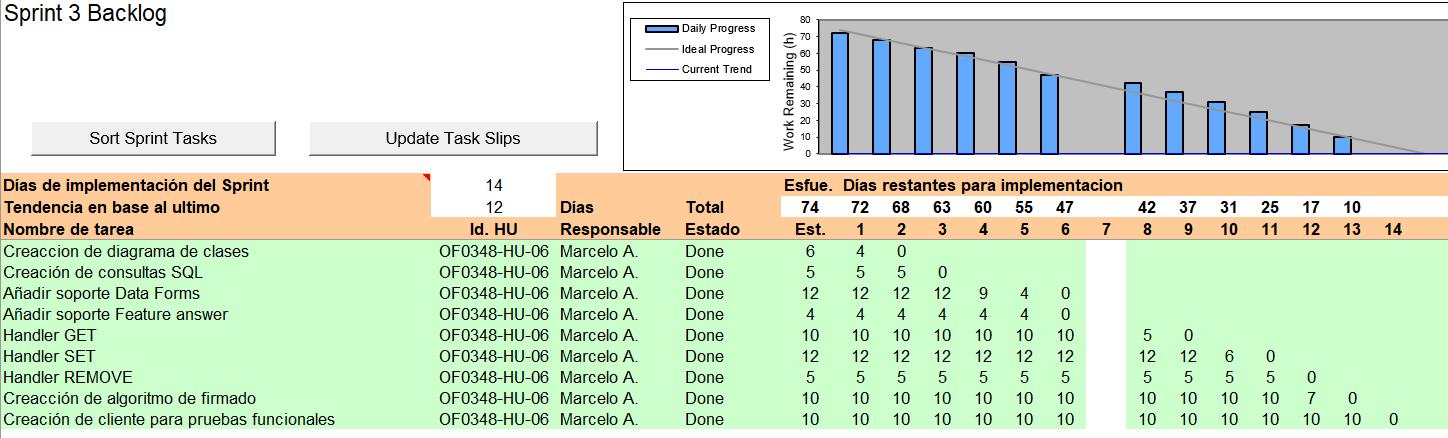
Luego el cliente recibe esta respuesta del servidor para luego comenzar la generación de «oauth\_signature» dependiendo del método firma elegido, será:

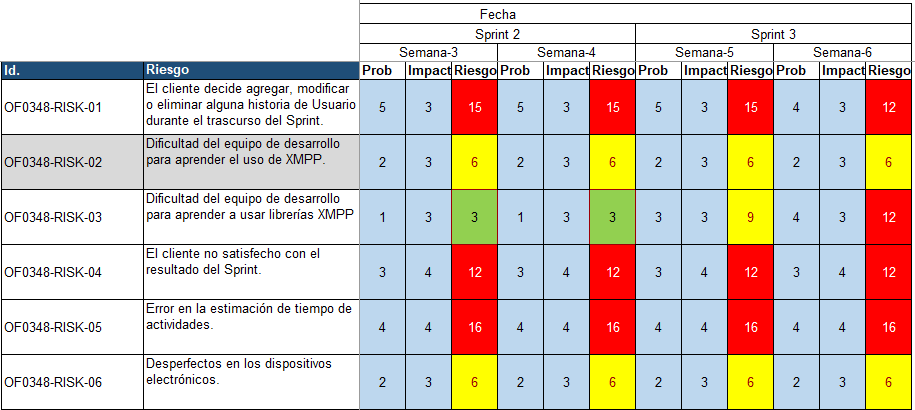
* HMAC-SHA1
* HMAC-SHA256
* RSA-SHA1
* PLAINTEXT

Por seguridad, se estableció como método de firmado único el HMAC-SHA256.

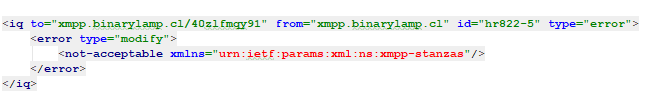
### A.1.3.3 Resultados

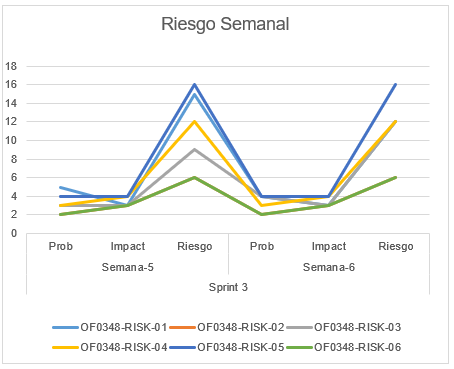
Luego de seguir de metodología de proyecto junto la de desarrollo, tenemos como la asignación de horas a cada tarea de la historia de usuario seleccionada para el desarrollo en este Sprint. Se puede apreciar el resultado en los gráficos e imagines a continuación:





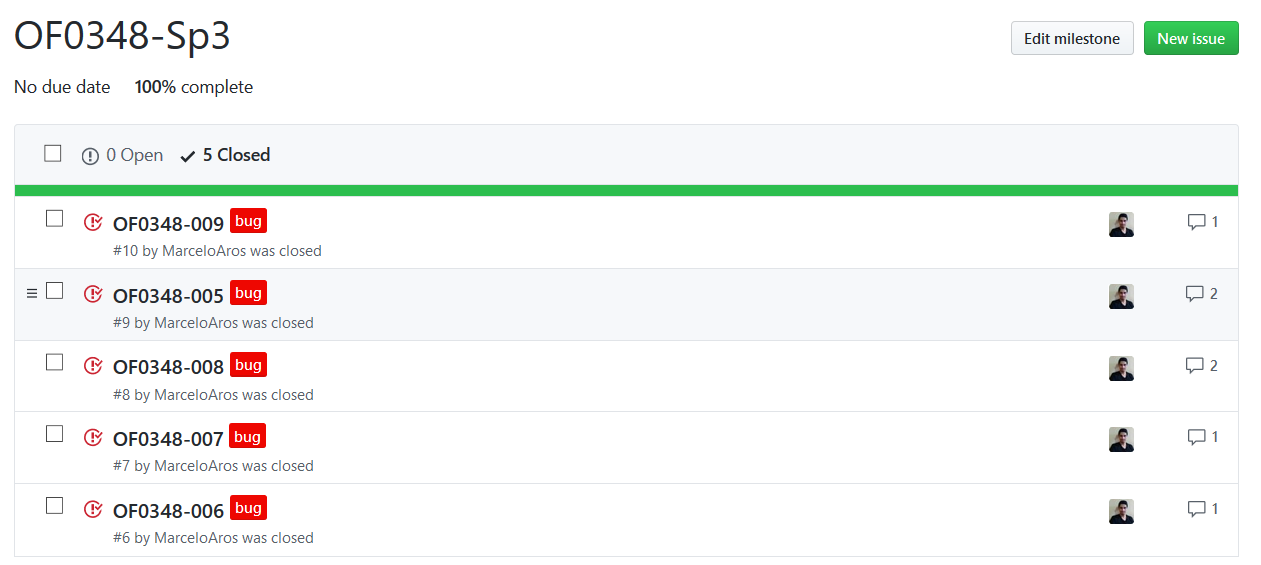


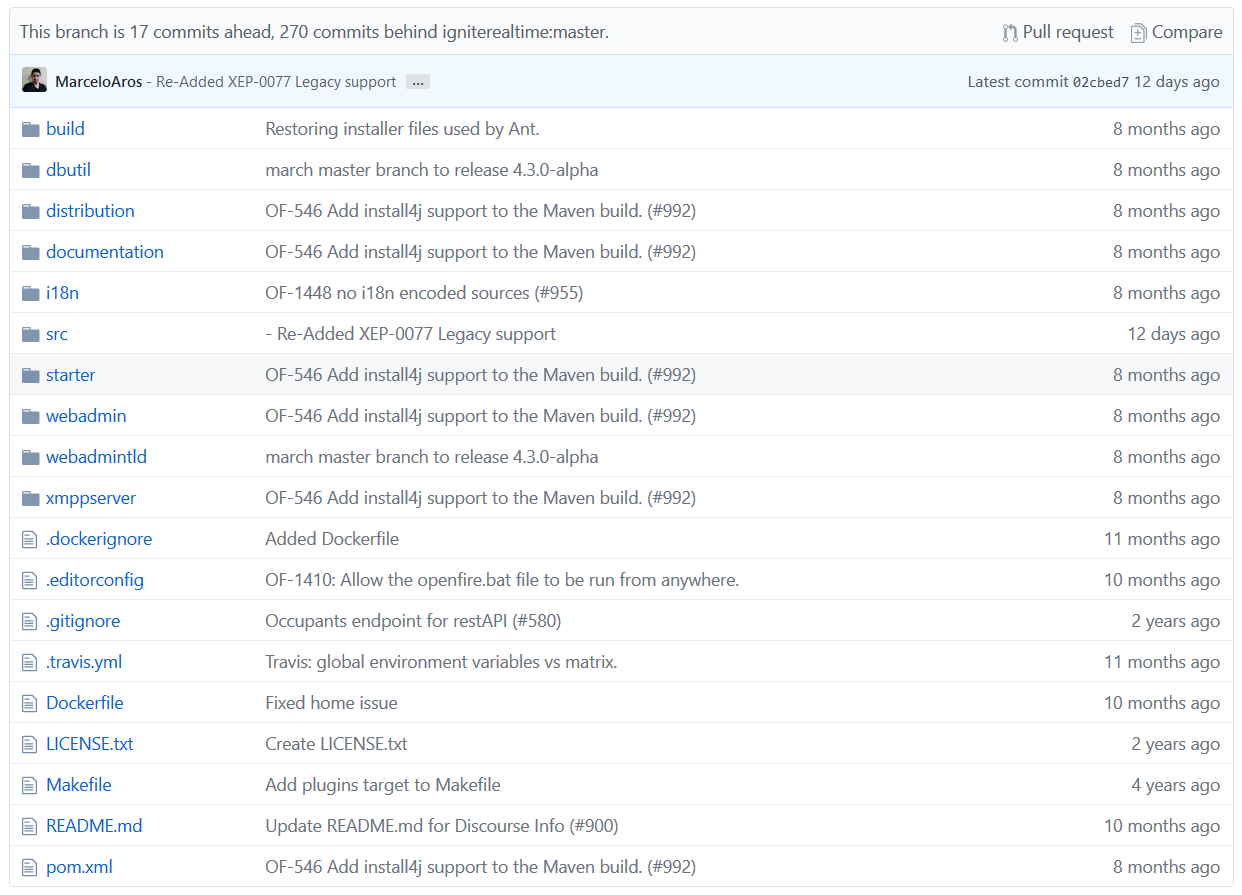




Como se puede apreciar en grafico de resultado este fue el Sprint que hubo más

### A.1.3.4 Evidencia





### A.1.3.5 Post-Mortem

**Problemas resueltos**

* Brindamos la solución al problema central del proyecto, el cual consistía en que el servidor XMPP «Openfire» pudiese responder a las solicitudes de registro de identidades mediante el uso del XEP-0348. De igual manera se brindó solución a los bugs asociados.

**Problemas futuros**

* Se ha completado el desarrollo asociado al servidor «Openfire». Pero aún queda el desarrollo correspondiente a librería «SMACK», el consiste en manera similar a la codificación de Servidor XMPP, es decir, agregar la funcionalidad de creación de identidades mediante la especificación XEP-0348.
* También tenemos las historias de usuario OF0348-HU-02 también la OF0348-HU-04 y la OF0348-HU-05.

Lecciones aprendidas

* Dentro de las consideraciones que debemos tener, esta la de evaluar mejor la gestión de riesgos al comenzar el proceso de un Sprint, pero investigando un poco más allá, se encontró que SCUM no determina de buena manera quien es el encargado de la gestión de riesgo, debido a que no puede gestionar ciertos riesgos intrínsecos al desarrollo de la metodología del proyecto, que en este caso es SCUM. Por ende, como los académicos no tienen consenso en este punto, se le asigno esta labor al director de proyecto.

**Métricas**

* KLOC: Se finalizo este Sprint con 2185 LOC
* Cantidad de defectos: Se detectaron 5 bugs en etapas tempranas de desarrollo.
* Efectividad en las pruebas: Se realizaron 3 pruebas de aceptación, de las cuales inicialmente pasaron 2, luego de realizar las correcciones necesarias, se logró que pasaron los 3 caso de prueba de manera exitosa.

## A.1.4 Sprint 4

### A.1.4.1 Planificación

### A.1.4.2 Diseño

### A.1.4.3 Resultados

### A.1.4.4 Evidencia

### A.1.4.5 Post-Mortem

## A.1.5 Sprint 5

### A.1.5.1 Planificación

### A.1.5.2 Diseño

### A.1.5.3 Resultados

### A.1.5.4 Evidencia

### A.1.5.5 Post-Mortem