FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

PROTOTIPO DE UN BUS DE SERVICIOS EMPRESARIALES ENTRE COOPERATIVAS DE AHORRO Y CRÉDITO UTILIZANDO MICROSOFT AZURE, CONSUMIBLE MEDIANTE UNA APLICACIÓN MÓVIL.

AUTOR

Juan José Ponce Torres

AÑO

2017

FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

PROTOTIPO DE UN BUS DE SERVICIOS EMPRESARIALES ENTRE COOPERATIVAS DE AHORRO Y CRÉDITO UTILIZANDO MICROSOFT AZURE, CONSUMIBLE MEDIANTE UNA APLICACIÓN MÓVIL.

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Sistemas de Computación e Informática

Profesor Guía

MSc. Carlos Andrés Muñoz Cueva

Autor

Juan José Ponce Torres

Año

2017

DECLARACIÓN DEL PROFESOR GUÍA

"Declaro haber dirigido este trabajo a través de reuniones periódicas con el estudiante, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación"

_____

Carlos Andrés Muñoz Cueva

Magister en Finanzas Empresariales

C. I. 1712981511

DECLARACIÓN DEL PROFESOR CORRECTOR

"Declaro haber revisado este trabajo, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación"

_____

Tannia Jacqueline Álava Freire

Magister Administración Tecnológica

C. I. 1706299169

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

"Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes."

_____

Juan José Ponce Torres

C. I. 1716729015

AGRADECIMIENTOS

Agradezco a Dios, por ser el motor que me impulsa a diario y me ha permitido obtener un logro más.

A la Universidad de las Américas por abrirme sus puertas y brindarme la oportunidad de realizarme profesionalmente.

A mis maestros, por su calidad humana y por compartir íntegramente sus conocimientos.

A mi profesor guía por su labor en este trabajo investigativo.

DEDICATORIA

A mis padres, por brindarme su apoyo incondicional y motivarme a siempre seguir adelante.

A mis hermanos, por impulsarme a ser mejor.

A mis Abuelitos por su amor y ejemplo.

RESUMEN

En el presente proyecto se desarrolló un prototipo de aplicativo móvil conectado a un Bus de Servicios Empresarial utilizando la tecnología de *Microsoft Azure*. El arquetipo tiene la finalidad de unir las cooperativas de ahorro y crédito pequeñas y medianas para que se pueda realizar transacciones monetarias, y se pueda potencializar los servicios que ofrecen las mismas. El aplicativo está conectado al bus de servicios a través de un servicio publicado en la misma plataforma de *Azure*, la cual se encarga de orquestar todas las transacciones para que se realicen correctamente

La aplicación móvil fue desarrollada utilizando *Xamarin Forms*. Tiene como funcionalidades principales transacciones entre cooperativas, pagos de servicios básicos, pago de tarjetas de crédito y soporta tener múltiples cuentas de distintas cooperativas asociadas a un mismo usuario. También, se creó simuladores de cooperativas de ahorro y crédito pequeñas que están conectadas al aplicativo móvil por el bus de servicios empresarial. Sirvieron para poder realizar transacciones ficticias, con la finalidad de probar el prototipo. Finalmente, para crear los datos usados en el simulador se publicó un sitio de administración que permite crear clientes, cuentas y servicios en los simuladores y puedan ser usados.

ABSTRACT

In this project, a prototype of a mobile application connected to a Business Services Bus was developed using Microsoft Azure technology. The archetype aims to unite small and medium-sized credit unions to enable monetary transactions to be carried out, and the services offered can be enhanced. The application is connected to the service bus through a service published in the same platform of Azure, which is in charge of orchestrating all the transactions so that they are carried out correctly

The mobile application was developed using Xamarin Forms. It has as main functionalities transactions between cooperatives, payments of basic services, payment of credit cards and supports having multiple accounts of different cooperatives associated to the same user. Also, small credit union simulators were created that are connected to the mobile application by the business services bus. They served to make fictitious transactions, to test the prototype. Finally, to create the data used in the simulator, a management site was created that allows the creation of clients, accounts and services in the simulators and can be used.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capitulo I. Introducción

1.1. Formulación del problema

El término bus de servicios empresarial fue utilizado por primera vez por *Roy W. Schulte* del Grupo Garner en 2002, y además fue mencionado en el libro "*The Enterprise Service Bus*" de David Chappel.

Actualmente existen empresas que ofrecen el servicio de integración de cajeros automáticos, o de sistemas de cobros, como por ejemplo *Banred*, pero los costos son elevados y solo los bancos grandes o medianos pueden acceder a este servicio. Por este motivo las cooperativas de ahorro y crédito pequeñas y medianas todavía realizan la integración de sus sistemas utilizando tecnologías obsoletas o incluso no cuentan con integración alguna.

En la cooperativa COOPROGRESO - Ecuador el uso de un ESB (Bus de Servicios Empresarial), sirve para integrar los servicios de cajeros automáticos con el Core bancario *DenariusOnline*. Otra de las ventajas es que puede hacer uso de la computación en la nube (*Cloud computing*), permite integrar sistemas de varias cooperativas de ahorro y crédito para crear un mejor mecanismo de comunicación que permita realizar transacciones de manera más eficiente, rápida y segura con menores costos debido a que se puede aprovechar economías de escala. (Jadeja, Modi 2012).

Cloud computing es una tecnología relativamente nueva que ofrece servicios a través de redes privadas virtuales sobre internet sin que el cliente tenga que preocuparse de comprar y mantener infraestructuras de servidores y almacenamiento adecuadas para sus aplicaciones e información. (Manheim, S., & Squillace, R, 2012)

1.2. Alcance

En el trabajo de titulación se propone desarrollar un aplicativo móvil utilizando *Xamarin Forms*. Va a estar conectado a un Bus de Servicios Empresarial que será desplegado en la plataforma de *Microsoft Azure*. Además, se propone implementar al menos dos simulaciones utilizando el lenguaje de programación C# en *Visual Studio*, para que realicen transacciones en línea y además sirvan para consultar el saldo disponible que tiene el cliente, pagar tarjetas de crédito y pago de facturas de servicios básicos. Todos los servicios estarán siempre disponibles porque se encontrarán publicados en la nube.

Se propone utilizar la tecnología de *Windows Azure*, para el despliegue de los servicios y del bus de servicios empresarial, ya que se cuenta con las licencias necesarias para el uso de esta plataforma. Además, los servidores que ofrece el sitio siempre están disponibles y permite administrar la infraestructura de una manera fácil y eficiente.

Adicionalmente se creará un sitio Web de administración de los datos de la aplicación y de los simuladores de las cooperativas de ahorro y crédito, con el fin de que se pueda generar datos para probar el prototipo. Se utilizará la arquitectura MVC y los diseños de *Bootstrap*, para la creación del sitio.

La limitación en este proyecto es que su aplicación va a estar conectada con simuladores y no con cooperativas reales debido a la dificultad para obtener acuerdos con las cooperativas debido a la sensibilidad de los datos.

1.3. Justificación

Actualmente la mayoría de cooperativas de ahorro y crédito pequeñas y medianas no cuentan con aplicaciones para teléfonos inteligentes en las que se pueda acceder a los servicios de una banca digital. Debido, a que las no cuentan con los recursos necesarios para la elaboración de este tipo de proyectos. Adicional, muchas cooperativas no disponen de un canal de integración para

realizar transacciones en línea, que les permita competir con los bancos grandes. Por esta razón se plantea como alternativa de solución la presente investigación.

Existen muchas ventajas sobre la aplicación que se propone crear para las cooperativas de ahorro y crédito pequeñas y medianas, ya que podrán ahorrar tiempo y dinero al cliente porque no tendrá que asistir presencialmente a una agencia, debido a que desde la aplicación móvil se podrá consultar el saldo disponible en sus cuentas, realizar transferencias en línea a distintas cooperativas, pagar varios servicios y se podrá tener varias cuentas de distintas cooperativas asociadas a un mismo usuario. Todo esto se lo podrá realizar desde cualquier lugar con un teléfono móvil inteligente.

Además, se brinda una oportunidad a las cooperativas para que puedan mejorar el servicio de integración de sus sistemas.

Esta propuesta es muy accesible para las cooperativas ya que no necesitan adquirir una infraestructura muy compleja y costosa para realizar la integración, debido a que el servicio va a estar disponible en la nube y se podrían conectar por medio de protocolos de internet seguros. Se aprovecha el concepto de economías de escala, se comparte la infraestructura, la aplicación y se podría cobrar dependiendo del número de transacciones que estas cooperativas de ahorro y crédito realicen.

1.4. Objetivos

1.4.1. Objetivo General

Desarrollar una aplicación móvil que se encuentre conectada a un bus de servicios empresarial integrado con simuladores de cooperativas de ahorro y crédito.

1.4.2. Objetivos específicos

- Definir una arquitectura de software de nivel empresarial que permita implementar la orquestación de transacciones en línea utilizando un bus de servicios empresarial como canal de comunicación entre cooperativas de ahorro y crédito y aplicaciones móviles para el cliente final.
- Desarrollar en la aplicación móvil la consulta de movimientos, transferencias monetarias entre cooperativas, pago de servicios básicos, pago de tarjetas de crédito y un inicio de sesión de dos formas de autenticación de usuarios.
- Crear simuladores de cooperativas de ahorro y crédito, que estén disponibles a través de un servicio Web, en las que se pueda realizar peticiones de información ficticia de clientes, de sus cuentas y de sus movimientos bancarios.
- Implementar en la arquitectura y en la aplicación móvil mecanismos de seguridad de acuerdo con lo establecido por los organismos de control del estado ecuatoriano.

Capítulo II. Marco Teórico

2.1. Bases Teóricas

2.1.1. Bus de Servicios Empresarial

ESB se encarga de conectar las aplicaciones en una arquitectura orientada a servicios (SOA). Es una plataforma de integración de servicios que permite la comunicación, conectividad, transformación, portabilidad y seguridad basada en estándares en la que se agrega mensajería, conexión a servicios Web, transformación de datos y enrutamiento inteligente para poder enlazar a varios servicios (Process OnLine, 2016).

Es una aplicación de middleware que brinda un servicio de transporte seguro y distribuido, en el que se almacena información que se desea enviar a través de la mensajería hasta que se garantice que el servicio cliente ha recibido los mensajes, incluso teniendo problemas de conectividad (Process OnLine, 2016).

Además, como ya se mencionó anteriormente el bus de servicios empresarial puede transformar los datos de un formato a otro, para que siempre se pueda satisfacer la compatibilidad de los formatos entre varios servicios (Process OnLine, 2016).

No es necesario saber la ubicación de los servicios porque ESB localiza el servicio cuando lo invoca, y en caso de que dicho servicio ya no esté disponible por cambio de ubicación o fallo en el funcionamiento, ESB puede determinar la nueva ubicación o alerta sobre el fallo detectado en el servicio (Process OnLine, 2016).

Es una herramienta que permite orquestar servicios, en el que se puede automatizar actividades para que se puedan construir servicios que agreguen valor al negocio (Process OnLine, 2016).

2.1.2. Bus de Servicios de Azure

El Bus de Servicios de *Azure* es un servicio gestionado para la creación de aplicaciones seguras y siempre conectadas, puede ser local o en la nube, que aprovechan los patrones de mensajería. El bus de servicio se utiliza a menudo como un componente clave en arquitecturas de soluciones consistentes, proporcionando mensajería asíncrona integrada con recursos adicionales como por ejemplo la base de datos SQL, aplicaciones Web y aplicaciones alojadas en Azure Virtual Machines. El Bus de Servicio cuenta con cuatro patrones de comunicación diferentes: Colas, Tópicos, *Relays* y *Event Hubs* (Collier y Shahan, 2016, p. 478).

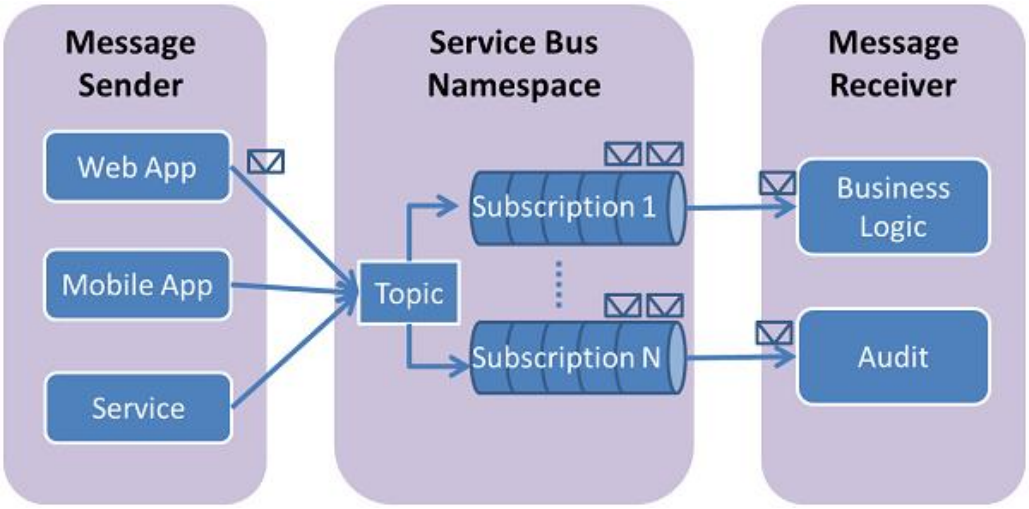


Figura 1. Servicios que ofrece *Windows Service Bus*.

Tomada de (Microsoft,2017).

2.1.3. Autenticación y autorización en Bus de servicios Azure

La autenticación de las aplicaciones en *Azure Service Bus* se realiza mediante *Shared Access Signature* (SAS) o por el control de acceso de *Active Directory* de Azure. Cuenta con un servicio de autenticación de firma, el cual permite a las aplicaciones autenticarse en el Bus de servicio mediante una clave de acceso (Collier y Shahan, 2016, p. 478).

2.1.4. Colas

Las colas proveen un método simple de entrega de mensajería porque se encuentran organizadas con la estructura de datos de primero entra, primero en salir. Los mensajes son enviados desde el emisor hacia el receptor, una vez recibidos son procesados para poder devolver alguna respuesta al emisor. Se puede tener múltiples emisores y receptores por lo que este tipo de mensajería es recomendable para el desarrollo de aplicaciones en la nube (Collier y Shahan, 2016, p. 478).

Las colas tienen un patrón asíncrono, solo se puede enviar mensajes desde uno de los extremos de la cola, y no de ambos. Permite que, si el receptor no está disponible el mensaje se quede almacenado por un tiempo en la cola, hasta que pueda ser procesado o hasta que caduque y se descarte (Collier y Shahan, 2016, p. 478).

Otra ventaja de la capa de comunicación es que se encuentra desacoplada entre emisores y receptores. Esto permite efectuar modificaciones de estos componentes, haciendo transparentes los cambios sin que afecte a la comunicación de los elementos que conforman el esquema de mensajería (Collier y Shahan, 2016, p. 478).

2.1.5. *Cloud Computing*

Es una alternativa a un centro de datos local porque el proveedor de los servicios de *Cloud Computing* es el responsable de la adquisición, mantenimiento y funcionamiento del *Hadware* de infraestructura. La mayor ventaja es que ofrece una variedad de servicios en donde solo se alquila el *Hadware* y el software que se necesita. Esto ayuda a las empresas porque les ahorra tiempo, espacio, dinero y dificultad en la adquisición y configuración de estos equipos. Además, estos servicios tienen la flexibilidad de economías de escala lo que permite que se arriende en ciertas épocas más capacidad o menos según lo requiera el cliente, y esto podría ahorrar muchos recursos (Collier, y Shahan, 2016, p. 4).

Un ejemplo de *Cloud Computing* puede ser *Azure* porque ofrece varios servicios en la nube, como por ejemplo la creación de servidores para desplegar aplicaciones web, o también permite la creación de servidores de bases de datos, adicional se puede crear máquinas virtuales y conectar remotamente a dichas máquinas. Además, ofrece servicios de conexión con aplicaciones como por ejemplo Bus de Servicios Empresarial, o servicios de notificaciones, etc. Pero estos servicios se cobran conforme la cantidad de transacciones que se realicen y el tiempo que se encuentran habilitados los servicios contratados. (Collier y Shahan, 2016, p. 4)

2.2. Metodología de desarrollo

2.2.1. *Scrum*

Scrum, es una metodología que se utiliza para proyectos de desarrollo de software, se caracteriza principalmente por ser ágil y flexible cuyo objetivo se basa principalmente en desarrollar las historias de usuario que son más importantes para el cliente. Scrum emplea un enfoque iterativo e incremental, de modo que cuando termina una iteración el cliente puede ir obteniendo pequeñas partes de software funcionando, hasta entregar el software completo. Por lo que Scrum es susceptible a cambios, porque en el cliente brinda retroalimentación durante el desarrollo del proyecto, específicamente al final de cada iteración. A diferencia de las metodologías continuas en donde se presenta todo el desarrollo solo al final del proyecto. Además, se debe mencionar que la experiencia es muy importante en esta metodología porque es útil para la planificación y estimación de los tiempos, basándose en las retroalimentaciones de cada iteración (Schwaber y Sutherland, 2013).

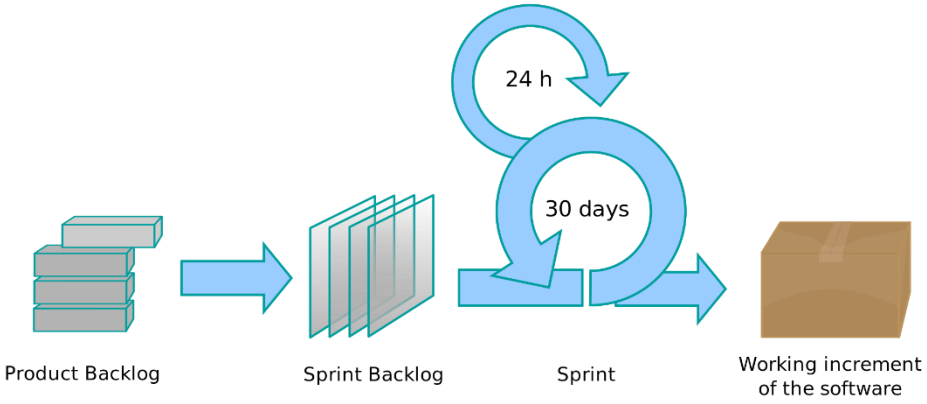


Figura 2. Imagen que explica *Scrum*.

Tomada de (Guerrero, 2016).

2.2.2. Equipo Scrum

El Equipo Scrum está compuesto por un Dueño de Producto (*Product Owner*), un Equipo de Desarrollo y un Scrum Master. Estos equipos se caracterizan porque son auto organizados debido a que internamente se decide cómo convertir los elementos de la lista de trabajo en funcionalidades del sistema (Schwaber y Sutherland, 2013).

Son multifuncionales porque en el equipo se encuentran todas las habilidades necesarias para elaborar el trabajo asignado. El equipo de desarrollo debe tener la suficiente cantidad de miembros para poder permanecer ágil y poder cumplir con el trabajo propuesto en el tiempo establecido (Schwaber y Sutherland, 2013).

2.2.3. Dueño del producto (*Product Owner*)

Se encarga de administrar el listado del producto (*Product Backlog*), y ayuda al equipo de desarrollo a entender la lista de trabajo, expresando de una forma clara y simple cada elemento que compone esta lista. Además, se encarga de

ordenar la lista de manera más eficiente para cumplir con los objetivos del trabajo propuesto. Optimiza el trabajo que desempeña el equipo de desarrollo para que se cumpla los tiempos establecidos (Schwaber y Sutherland, 2013).

2.2.4. *Scrum Master*

Es el encargado de asegurarse que el equipo entienda la metodología y la aplique correctamente. Además, es el líder del equipo *Scrum*, facilita a los miembros del equipo de desarrollo para que estos puedan continuar con su trabajo sin impedimentos ni restricciones. Ayuda al dueño del producto a ordenar correctamente el listado del trabajo incompleto para maximizar el rendimiento del equipo. La lista del trabajo pendiente se hace antes de cada iteración, para que cuando se lleven a cabo las reuniones se pueda presentar al equipo de desarrollo, y se planifique correctamente También es encargado de facilitar las reuniones de *Scrum*, cerciorándose de que sean productivas y que cumplan con su objetivo (Schwaber y Sutherland, 2013).

2.2.5. Equipo de desarrollo

Está conformado por profesionales que se encargan de la creación de incrementos de funcionalidades del sistema hasta poder obtener un producto completamente terminado, en donde se cumplan todas las “historias de usuario” solicitadas por el cliente. Cada incremento de software debe estar desarrollado para que se pueda poner en producción al final de cada *Sprint* (Schwaber y Sutherland, 2013).

El equipo de desarrollo se encarga además de seleccionar los elementos de trabajo que se compromete a realizar en cada iteración, también estima la complejidad de cada elemento que se encuentra en la lista de trabajo para cada *Sprint*. Cada miembro del equipo se auto asigna las tareas según su experiencia o su interés en desarrollar dicha tarea. Durante el *Sprint* el equipo trabaja de

manera conjunta para conseguir el objetivo planteado para dicha iteración (Schwaber y Sutherland, 2013).

2.2.6. *Sprint*

Es un bloque de tiempo de un periodo entre dos semanas hasta un mes, en el que se crea incrementos del producto, los cuales deben ser utilizables y desplegables. El *sprint* está compuesto de una reunión de planificación, *Scrum* diario, revisión y retrospectiva del *Sprint* Cada nuevo sprint comienza después de que finaliza el anterior. Durante la ejecución del *Sprint* no se puede afectar el objetivo establecido en la reunión de planificación. Pero si existiese el caso que los objetivos propuestos en el *Sprint* se vuelven obsoletos para la organización, existe la posibilidad de que el *Product Owner* cancele el *Sprint*. Los objetivos se vuelven obsoletos cuando la organización decide que el proyecto a realizarse ya no forma parte de los objetivos generales (*Schwaber y Sutherland*, 2013).

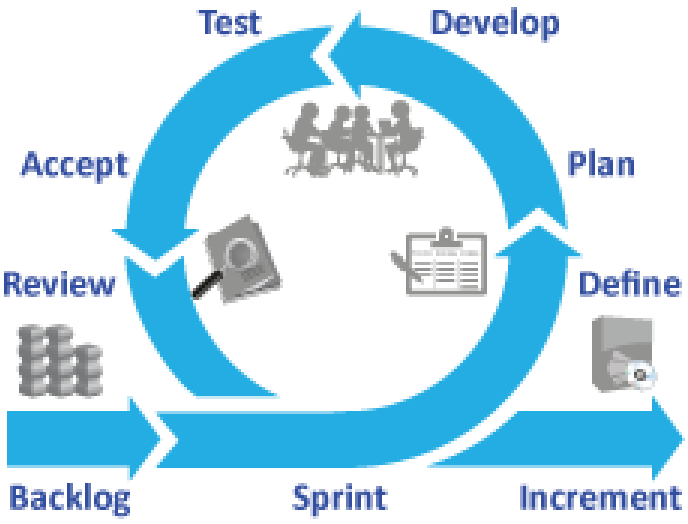


Figura 3. Imagen que muestra etapas de *Sprint*.

Tomada de (JamBuster, s. f).

2.2.7. Reunión de planificación del *Sprint*

Se planifica el trabajo a realizarse durante el *Sprint*. Debe estar reunido el equipo completo ya que es un trabajo colaborativo. Este evento tiene un máximo de duración de ocho horas en el caso de que sea un *Sprint* largo, en el caso de que sea más corto la duración es menor. El *Scrum Master* es el encargado de que la reunión se lleve a cabo y de que se respete los tiempos establecidos (*Schwaber y Sutherland*, 2013).

En la reunión se tratan dos temas especialmente. El primero: ¿Qué puede ser entregado en este *Sprint*?, en donde se plantea el objetivo del *Sprint*, y después se selecciona los elementos del *Product Backlog* que se necesitarían para poder cumplir el objetivo propuesto. Se toma en cuenta la capacidad proyectada por el equipo, la velocidad de desarrollo del equipo en la iteración anterior. Pero la cantidad de elementos seleccionados depende del Equipo de desarrollo ya que solo ellos se pueden evaluar si podrán lograr lo propuesto (Schwaber y Sutherland, 2013).

El segundo tema que se trata en esta reunión es: ¿Qué estrategia se utilizará para completar el trabajo seleccionado? Una vez elegido el objetivo del *Sprint* el equipo de desarrollo debe decidir cómo realizará la tarea para al final de la iteración obtener un producto “Terminado” y que sea utilizable por el cliente. Se debe descomponer las historias de usuario en unidades pequeñas para que puedan ser auto asignadas por los miembros del equipo de desarrollo (Schwaber y Sutherland, 2013).

Al final de la reunión el equipo de desarrollo podrá dar a conocer al *Product Owner* y al *Scrum Master* cómo desea completar el incremento que se propuso al inicio de la reunión. También se dará a conocer que tareas desea cada miembro del equipo realizar durante el desarrollo del proyecto (Schwaber y Sutherland, 2013).

2.2.8. *Scrum* Diario

Es una pequeña reunión diaria con una duración de 15 minutos. Se realiza a la misma hora y en el mismo lugar para facilidad de todos los asistentes. Sirve para que el equipo de desarrollo pueda actualizar sus actividades y realice una planificación para las próximas veinticuatro horas (Schwaber y Sutherland, 2013).

Cada miembro del equipo debe responder las siguientes preguntas sobre el desarrollo del *Sprint*:

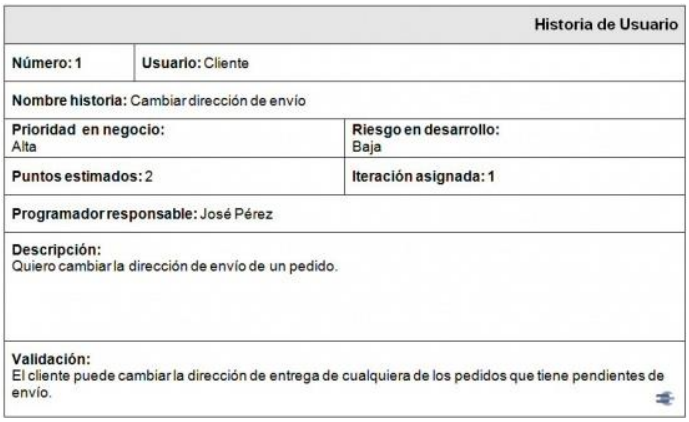
- ¿Qué hice ayer?
- ¿Qué haré hoy?
- ¿Veo algún impedimento que evité que se logré el objetivo del *Sprint*?

2.2.9. Revisión del *Sprint*

Es una reunión que se utiliza para presentar el trabajo completado durante el *Sprint*. Es útil porque facilita la retroalimentación de información y fomenta la colaboración. Entre los asistentes a este evento se encentran el equipo *Scrum* y los interesados clave del proyecto. El *Product Owner* es el encargado de empezar la reunión, indica que elementos del *Product Backlog* se lograron terminar y cuales están como no terminados. También da a conocer posibles fechas para la finalización del proyecto basándose en el progreso obtenido hasta el momento. Después el equipo de desarrollo explica el trabajo realizado y responde preguntas a los asistentes. Al final se colabora con información sobre lo que se debe hacer a continuación, esta información es útil para las reuniones de planificación de los siguientes *Sprint* (Schwaber y Sutherland, 2013).

2.2.10. Historia de usuario

Es una descripción de la funcionalidad que se debe agregar al software que se está desarrollando, cuyo objetivo es aportar valor al cliente. Debe contener un nombre breve y descriptivo. Una descripción de la funcionalidad que se desea realizar y criterio de validación en donde se describe las consideraciones para ser considerado terminado y aceptado. Además, se debe incluir información sobre los puntos de estimación, desarrollador asignado, la iteración a la que pertenece (Schwaber y Sutherland, 2013).



Historia de Usuario	
Número: 1	**Usuario:** Cliente
Nombre historia: Cambiar dirección de envío	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: José Pérez	
Descripción: Quiero cambiar la dirección de envío de un pedido.	
Validación: El cliente puede cambiar la dirección de entrega de cualquiera de los pedidos que tiene pendientes de envío.	

Figura 4. Ejemplo Historia de Usuario.

Tomada de (ScrumManager, 2013).

2.2.11. Lista de Producto (*Product Backlog*)

Es una lista que contiene todos los elementos necesarios para el desarrollo de un producto de software. Los elementos de la lista forman parte de todas las características, funcionalidades, y mejoras que se debe realizar sobre el *software* antes de su entrega. Cada elemento está compuesto por la descripción de la funcionalidad, el orden de importancia, la estimación y el valor. El *Product Owner* es el responsable de esta lista, de editar su contenido, de que siempre esté disponible para el equipo de desarrollo y de su orden (*Schwaber* y *Sutherland*, 2013).

La lista de producto constantemente está cambiando debido a que al principio solo se muestra lo que se entendió con la obtención de las historias de usuario.

A medida que se desarrolla el producto de *software*, esta va ir creciendo y mejorando por la retroalimentación que se recibe después de cada *Sprint* (Schwaber y Sutherland, 2013).

ToDo List

ID	Story	Estimation	Priority
7	As an unauthorized User I want to create a new account	3	1
1	As an unauthorized User I want to login	1	2
10	As an authorized User I want to logout	1	3
9	Create script to purge database	1	4
2	As an authorized User I want to see the list of items so that I can select one	2	5
4	As an authorized User I want to add a new item so that it appears in the list	5	6
3	As an authorized User I want to delete the selected item	2	7
5	As an authorized User I want to edit the selected item	5	8
6	As an authorized User I want to set a reminder for a selected item so that I am reminded when item is due	8	9
8	As an administrator I want to see the list of accounts on login	2	10
Total		**30**	

Figura 5. Ejemplo de *Product Backlog*.

Tomada de (Scrum Institute, s.f.).

2.2.12. Lista de pendientes del *Sprint* (*Sprint Backlog*)

Esta lista se crea una la reunión de planificación del *Sprint*, cuando el equipo *Scrum* ha decido cuales son los elementos de la Lista de Producto para ser desarrollado durante el próximo *Sprint*, por lo que se podría definir como un plan para entregar cumplir con el objetivo del *Sprint*. Esta lista es creada por el equipo de desarrollo, el cual decide el trabajo a realizarse durante la próxima iteración (*Schwaber* y *Sutherland*, 2013).

Además, esta lista es considerada como un plan detallado, que sirve para observar los cambios que aparecen durante el desarrollo del *Sprint*, y se puedan comprender en el *Scrum Diario*. El equipo de desarrollo es el encargado de modificar esta lista durante el *Sprint*, mientras que se avanza sobre el plan establecido en la reunión de inicio del *Sprint*, además se va obteniendo más experiencia sobre el trabajo realizado para conseguir el objetivo (*Schwaber* y *Sutherland*, 2013).

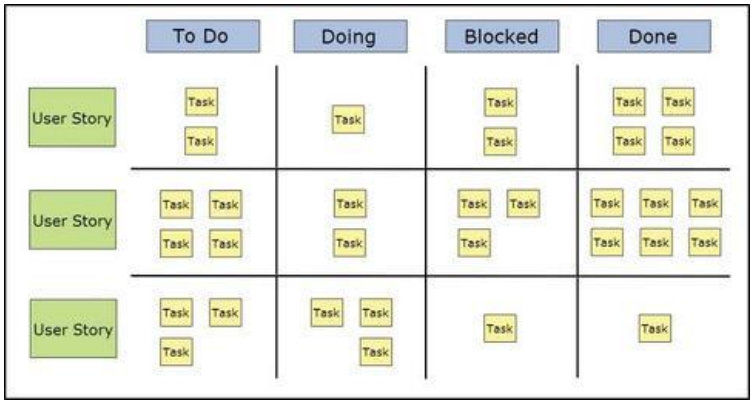


Figura 6. Ejemplo *Sprint Backlog*.

Tomada de (Li, 2016).

2.2.13. *Burndown Chart*

Este gráfico permite resumir todo el trabajo del equipo que queda pendiente. Porque siempre tiene información actualizada. Es una herramienta que se utiliza para que el equipo pueda estar auto organizado. Además, a largo plazo sirve para obtener la velocidad con la que se está completando las tareas asignadas. Existen variaciones de este grafico como por ejemplo se puede separar el trabajo restante por cada miembro del equipo de desarrollo (*Schwaber* y *Sutherland*, 2013).

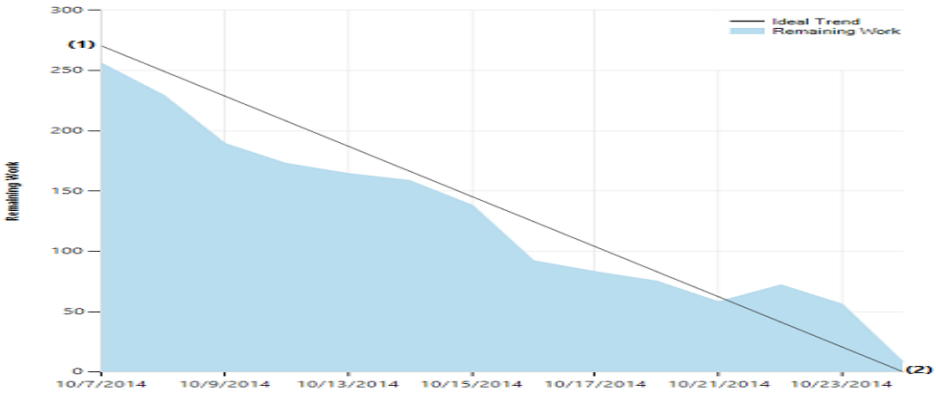


Figura 7. Ejemplo de *Burndown Chart*.

Tomada de (Microsoft, 2017).

2.3. Herramientas de desarrollo

2.3.1. *Microsoft Project*

Microsoft Project es una herramienta que simplifica la administración de proyectos y recursos (PPM). Además, sirve para planificación integrada y ayuda a realizar un seguimiento de los proyectos y a mantenerlos organizados (Microsoft, 2017).

Este *software* cuenta con una ruta critica el cual permite que se analice la secuencia con la que se deben realizarse las tareas y las dependencias que tienen unas de otras. Además, ofrece un control del proyecto, el cual permite a los administradores inspeccionar los cambios que han existido desde el inicio del proyecto hasta la finalización. También crea un diagrama de *Gantt* en el que se compara el tiempo con las actividades ingresadas en el proyecto. Adicional en este software se puede asignar recursos a cada una de las tareas creadas en el proyecto y de esta manera se puede llevar una adecuada distribución de los recursos con los que se cuenta para el proyecto, a cada recurso se le puede asignar un costo, y se puede calcular el costo total del proyecto (Microsoft, 2017).

2.3.2. *Visual Studio*

Visual Studio es un entorno de desarrollo integrado que soporta varios lenguajes de desarrollo como por ejemplo C++, C#, *Visual Basic* .NET, F#, *Java*, *Python*, *Ruby*, y *PHP*. Este entorno permite crear excelentes aplicaciones para dispositivos móviles u ordenadores de escritorio, ya sea en web o en la nube. Además, se puede escribir código para *iOS*, *Android* y *Windows* desde el mismo IDE. Además, cuenta con características adicionales como por ejemplo *IntelliSense*, que permite facilidad para el desarrollo de aplicaciones. *Visual Studio* aumenta la productividad y le facilita el trabajo, tanto si lo lleva a cabo por sí solo como si forma parte de un equipo (MSDN Library, 2016).

Visual Studio permite ampliar las posibilidades de desarrollo de aplicaciones, como por ejemplo se puede agregar complementos como *Xamarin* que sirve para hacer aplicaciones móviles multiplataforma, además se puede agregar herramientas para el desarrollo de videojuegos como *Unity*, o *Apache Cordova* para desarrollo de código base común para *HTML*, *CSS* y *JavaScript*, etc. También *Visual Studio* permite conectar las aplicaciones a servicios externos como por ejemplo *Azure* (MSDN Library, 2016).

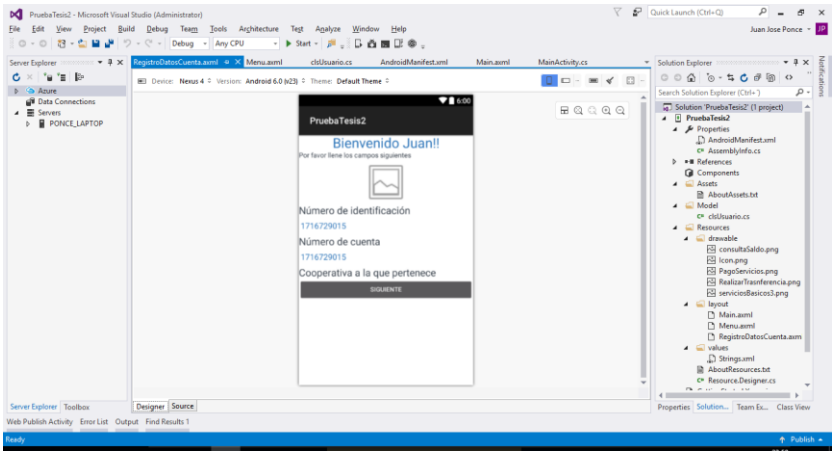


Figura 8. Entorno de desarrollo *Visual Studio*

2.3.3. *Xamarin Studio*

Es un entorno para el desarrollo de aplicaciones móviles nativas en *Android*, *IOS* y *Windows*. Se puede utilizar el lenguaje C# como código base para el desarrollo de las aplicaciones. Y cuando se compila la aplicación esta se traduce al lenguaje nativo de cada plataforma (MSDN Library, 2016).

Las aplicaciones *Xamarin* son construidas con controles e interfaces estándares. Estas aplicaciones pueden acceder a todas las funcionalidades expuestas por la plataforma del dispositivo, es decir que se puede acceder a la cámara, al GPS, a los sensores de huellas y demás funcionalidades adicionales que ofrecen los teléfonos inteligentes (MSDN Library, 2016).

Otra de las mayores ventajas de este IDE es que permite compartir código, es decir que se crea una aplicación compartida, con las funcionalidades básicas del teléfono y se puede compilar la aplicación para *Android*, *IOS* o *Windows* sin necesidad de reescribir el código nuevamente (MSDN Library, 2016).

2.3.4. *Microsoft Azure*

Microsoft Azure es una plataforma en la nube de *Microsoft* que ofrece una amplia colección de servicios integrados para que los desarrolladores y el personal encargado de manejar la infraestructura puedan facilitar su trabajo. Este portal admite la creación, implementación y administración de infraestructura para un centro de datos. *Azure* puede proporcionar una plataforma para crear aplicaciones que puedan aprovechar y satisfacer las necesidades de cualquier negocio. Es aprovechado por empresas globalizadas que tienen varias sucursales en lugares remotos y necesitan tener su infraestructura conectada en todo momento (Tulloch, Windows, 2013).

Además, brinda herramientas y tecnologías de código abierto, porque permite soportar una amplia gama de sistemas operativos, lenguajes de programación, marcos, bases de datos y dispositivos, que hacen de esta plataforma única para el manejo de la infraestructura y la conexión de servicios con aplicaciones (Tulloch, Windows, 2013).

2.3.5. Lenguaje de Programación C#

Es un lenguaje de programación que pertenece a la Plataforma .NET. Es un Lenguaje Común en Tiempo de Ejecución (CLR, *Common Language Runtime*). Es considerado como el principal lenguaje de esta plataforma para escribir aplicaciones (MSDN *Library*, 2016).

C# deriva de C y C++, es un lenguaje actual, amigable con el usuario, y orientado a objetos, se podía decir que este lenguaje es una evolución de C++ en las áreas de clases, *namespaces*, sobrecarga de métodos y manejo de excepciones. Se considera menos complejo que C++, además es más fácil de utilizar y menos propenso a errores. Es muy fácil de aprender, y se recomienda como un lenguaje para principiantes, ya que se puede entender fácilmente toda la teoría de orientación a objetos. Este lenguaje permite realizar juegos de video con *Unity*, aplicaciones web utilizando MVC o ASPX, Conectarse a servicios en la nube, o incluso realizar aplicaciones móviles con *Xamarin Studio* (MSDN *Library*, 2016).

2.3.6. SQL

Es un lenguaje de consulta estructurado, es considerado como un lenguaje de alto nivel para motores de bases de datos relacionales. Fue creado por el Instituto Americano de Normalización. Es por lo que la mayoría de DBMS como *SQL Server*, *Oracle o MySQL* utilizan este leguaje para las sentencias (Microsoft, 2012).

SQL tiene tres tipos de sentencias: Lenguaje de Definición de Datos (DDL), sirve para la declaración de objetos que se van a utilizar en la sentencia. Lenguaje de manipulación de datos, son sentencias que se usan para manipular los datos que están almacenados en la base. Lenguaje de control de datos, que se utiliza para controlar las funciones de administración como atomicidad o seguridad. (Microsoft, 2012).

SQL Server, no solo es un motor de base de datos, ya que cuenta. Con una colección de componentes que puede implementar para formar una plataforma de información que puede encontrarse en un servidor local o en la nube. Existen dos tipos de estas plataformas, primero aquellas que son utilizadas para administrar datos y las que le ayudan a ofrecer inteligencia de negocios (BI) (*Microsoft*, 2012).

2.3.7. Android

Este sistema operativo fue programado en base al *Kernel* de *Linux*. Se creó con la finalidad de que pueda ser utilizado en *smartphones*, *tablets*, relojes inteligentes, incluso en automóviles (Gironés, 2012).

Se puede desarrollar aplicaciones en lenguaje Java en la versión *Dalvik. Android* permite desarrollar aplicaciones móviles en las que se pueda acceder a las funciones que ofrecen los dispositivos móviles como por ejemplo GPS, acceso a la agenda de contactos, o a la red de internet, etc. (Gironés, 2012).

Android es libre porque no hay que pagar para poder utilizarlo. Es decir que si un fabricante de teléfonos requiere utilizarlo puede hacerlo, incluso podría mejorar el sistema y modificarlo para acoplarlo al teléfono. El desarrollo de las aplicaciones también tiene costos bajos y utiliza *Java* como lenguaje de programación que es bastante simple, es por lo que es uno de los sistemas operativos preferidos por los programadores de aplicaciones móviles, adicional si se desea se puede utilizar el IDE propio que es *Android Studio*, que igual no tiene ningún costo. Pero también existen entornos de desarrollo como *Xamarin Studio* que permite desarrollar aplicaciones en .NET y *Java* (Gironés, 2012).

2.3.8. *Bootstrap*

Según el sitio web de *Bootstrap*, este es el HTML, CSS y *JavaScript framework* para desarrollar aplicaciones responsivas, y el primero desarrollando proyectos móviles en la Web. En su sitio Web se puede encontrar mucha información de cómo empezar a utilizar este marco de referencia. Como instalarlo e incluirlo en el proyecto de aplicación Web para que los formularios y los objetos en general que ocupa el sitio sean responsivos y el diseño cambie acorde con *Bootstrap* (Lambert, 2015).

2.3.9. MVC

Modelo Vista Controlador (MVC) es una arquitectura de software que se utiliza para la creación de sistemas Web principalmente, debido a que se trata de separar la interfaz de usuario con manipulación de los datos (Álvarez, 2014).

Se utiliza cuando se desea que el ciclo de vida este mejor aplicado, por ejemplo, que exista mayor facilidad en el mantenimiento del código, que pueda existir reutilización de métodos y clases y los conceptos de cada capa estén separados adecuadamente (Álvarez, 2014).

El modelo separa en tres capas distintas las aplicaciones Web. La primera capa es el Modelo, que es el lugar en donde se ubican los datos que se utilizan para el aplicativo, en el modelo estarán ubicados los métodos para la manipulación de los datos, es decir las funciones CRUD (crear, eliminar, modificar y leer) (Álvarez, 2014).

La segunda capa es la vista en donde se ubica todo el código HTML, PHP, etc. Esto dependerá de las necesidades que requiera el diseñador, principalmente esto es una ventaja porque cuando se trabajan en equipos grandes y se dividen las tareas entre los que realizan el *Front End* y los que programan el *backend*, se puede trabajar en distintos archivos y se puede ir enlazando las vistas con las otras capas. Esta capa trabaja directamente con el modelo ya que se requiere mostrar los datos en pantalla, y ejecutar las funciones de cada botón (Álvarez, 2014).

La tercera capa son los controladores, es la capa que tiene el código para realizar todas las acciones solicitadas por el cliente a través de la vista, es decir que existe un enlace entre la interfaz de usuario y el modelo de datos. Esta capa no manipula ningún dato, ni tiene como salida información para mostrar en pantalla. Sino que envía los datos desde la interfaz de usuario hacia el modelo, y viceversa (Álvarez, 2014).

2.3.10. Algoritmo de encriptación *Rijndael*

Fue elegido por el instituto nacional de estándares de tecnología como el candidato para el estándar de encriptación avanzado (AES) que podría remplazar al estándar de encriptación de datos (DES). Este algoritmo fue diseñado por *Vincent Ritjmen* and *Joan Daemen*, del cual se deriva el nombre de este algoritmo. Es una nueva generación de algoritmos que soportan llaves de encriptación de hasta 256 bits, y divide los datos hasta en bloques de 128 bits (Pitchaiah y Daniel, 2012).

El algoritmo se basa en un conjunto de rondas, las cuales son conjuntos de cuatro funciones matemáticas diferentes que no se pueden reversar. El resultado de cada una de estas funciones se las conoce como estado, y estas son mapeadas a una matriz de estado. La matriz de estado sufre cuatro transformaciones por ronda (*ByteSub, ShitRow, MixColumn, AddRoundKey*) dando como resultado un texto cifrado (Pitchaiah y Daniel, 2012).

Capítulo III. Desarrollo del proyecto

3.1. Arquitectura

La arquitectura que se utilizará para el desarrollo de este proyecto se basa exclusivamente en que todos los componentes estarán enlazados al bus de servicios empresarial, el cual será el medio por el cual se comunicaran todas las partes del sistema. Como se muestra en el siguiente diagrama, el aplicativo móvil está conectado a un servicio móvil publicado en la nube utilizando *REST* por un canal seguro (HTTPS) que impide que atacantes puedan interceptar los mensajes que se transmiten. También, los simuladores se comunican por servicios *REST* en canales seguros con el Bus de Servicios Empresarial.

Cuando un cliente realiza una petición de consulta de movimientos, saldos, facturas de los servicios o transferencias de dinero, el aplicativo envía la petición al servicio móvil que se encuentra en la nube, y este que contiene algoritmos de encriptación *Rinjdael* prepara un mensaje para que sea enviado a través del ESB, y lo encripta usando llaves para que en el caso de que un atacante intermedio este escuchando tráfico no pueda descifrar el contenido del mensaje y los datos del cliente no sean vulnerados.

Los simuladores de cooperativas de ahorro y crédito siempre estarán escuchando las colas de entrada, hasta que se encuentre un mensaje, el cual es desencriptado y después es atendido, y según el tipo de operación solicitada. Al final se envía una respuesta encriptada al cliente con el mismo algoritmo de encriptación del mensaje, por una cola de respuesta.

Cuando el mensaje de respuesta llega al servicio móvil, los datos se envían a la aplicación móvil para que puedan ser mostrados en la pantalla del dispositivo. En el caso de que sean consultas, se mostrarán los datos solicitados por el cliente y el caso de que sean operaciones bancarias se mostrará el resultado de la operación.

Cada simulador cuenta con su propia base de datos, en los que se almacena datos de los clientes, cuentas, movimientos y tarjetas de crédito. La base de datos del simulador de servicios básicos almacena información sobre los clientes, los suministros y las facturas que cada cliente tiene pendiente el pago. Mientras que la base de datos del servicio móvil se tiene almacenado los datos de los usuarios que se han registrado en la aplicación usando las credenciales de Google, datos de las cooperativas que se han registrado, nombre de las colas que utiliza cada cooperativa para la comunicación, catálogos de la aplicación y un log de transacciones.

Las colas están configuradas para tener un ancho de banda de hasta 16 GB en mensajes esperando a ser procesados por los simuladores. También cada mensaje tiene un tiempo de vida de hasta 30 segundos antes de que sea destruido, con la finalidad de evitar ataques de denegación de servicios.

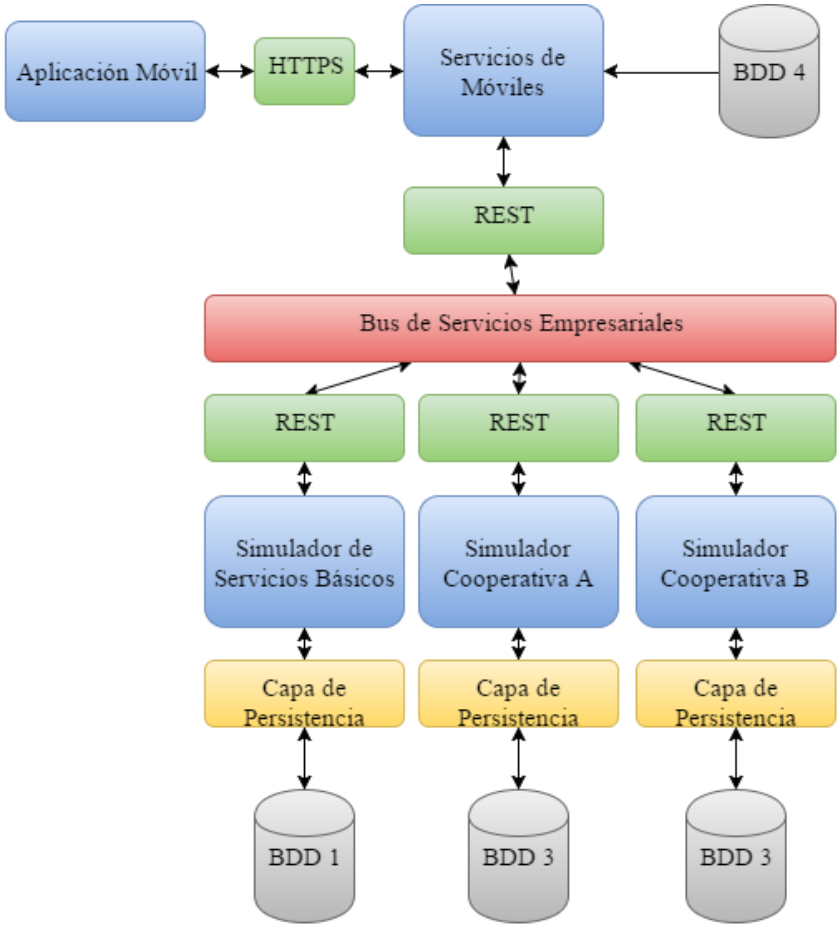


Figura 9. Arquitectura planteada

Para el desarrollo de la aplicación móvil se va a utilizar *Xamarin*, que es una tecnología de .Net que sirve para poder realizar aplicaciones tanto en *Android*, *IOS* o *Windows Phone*. Por propósitos del prototipo se realizará la aplicación solo para el sistema operativo de *Android*.

Para el desarrollo de los simuladores, y de los servicios móviles se utilizará el entorno de desarrollo *Visual Studio* y el lenguaje de programación C#. Para la creación y manejo de las bases de datos se utilizará *SQL Server*, ya que es acoplable con las tecnologías de desarrollo de los simuladores y además se puede publicar en *Azure* con mucha facilidad.

Finalmente, para el despliegue del bus de servicios se seleccionó la plataforma de *Microsoft Azure* ya que cuenta con un ESB propio entre sus herramientas. Además, se puede administrar el resto de la infraestructura fácilmente desde una misma consola.

Para ofrecer una mejor seguridad, se creó un *PIN* para el ingreso a la aplicación, cada vez que se cierra y se vuelve a abrir.

Según la Norma de Riesgo Operativo, en el artículo que trata sobre la Banca Móvil, que se encuentra detallada en el Anexo 1 establece que “las instituciones del sistema financiero que presten servicios a través de banca móvil deberán sujetarse en lo que corresponda a las medidas de seguridad dispuestas en los sub-numerales 4.3.5 y 4.3.8.

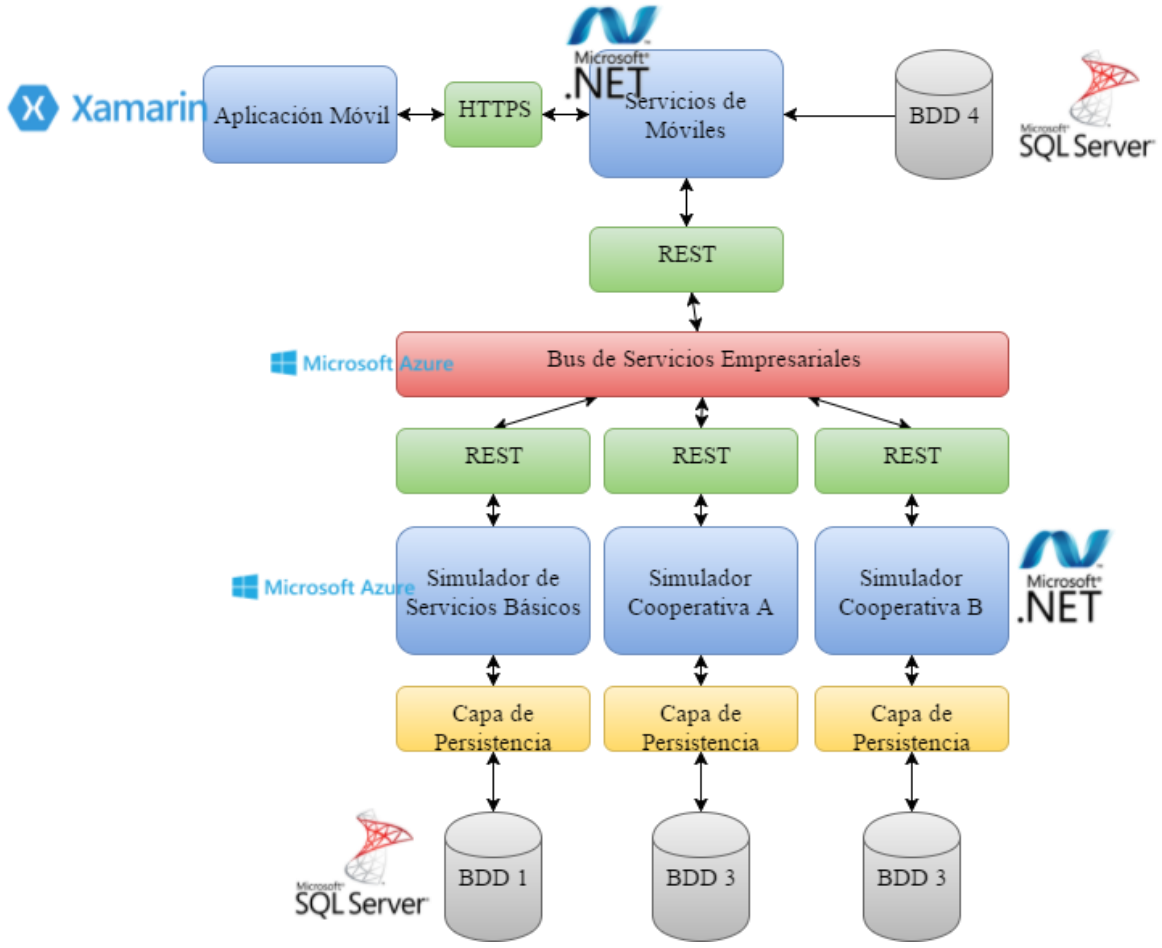


Figura 10. Arquitectura con herramientas tecnológicas utilizadas.

3.2. Aplicación de la Metodología

Para el desarrollo del proyecto se va a utilizar la metodología de *Scrum*. El proyecto está dividido en seis *Sprint*, en los que se ira realizando avances del proyecto, hasta concluir finalmente con la aplicación móvil, el sitio web de administración, los simuladores y la conexión con el bus de servicios empresarial. Se ha dividido en diez y seis diferentes historias de usuario, que incluye desarrollo y documentación del proyecto.

Tabla 1.

Product Backlog del Proyecto.

Tipo de Ítem	**Nombre de la Tarea**	**Horas de trabajo**	**Numero de Sprint**	**Prioridad**
Sprint	Sprint #1	0 horas	1	18
Ítem de trabajo	*Login* con Google	10 horas	1	5
Ítem de trabajo	Pantalla de registro	10 horas	1	3
Ítem de trabajo	Menú principal	10 horas	1	5
Ítem de trabajo	Desarrollo del Marco Teórico en documentación	12 horas	1	5
Sprint	Sprint #2	0 horas	2	20
Ítem de trabajo	Pantalla de consulta de movimientos	8 horas	2	3
Ítem de trabajo	Pantalla de pago de servicios	8 horas	2	3
Ítem de trabajo	Pantalla de Transferencias	8 horas	2	3
Ítem de trabajo	Pantalla para agregar Cuenta	8 horas	2	3
Ítem de trabajo	Pantalla de pin de ingreso	8 horas	2	3

Ítem de trabajo	Avance de documentación del proyecto	8 horas	2	5
Sprint	Sprint #3	0 horas	3	34
Ítem de trabajo	Conexión de aplicación móvil con *Azure*	16 horas	3	5
Ítem de trabajo	Creación de simulador de servicios básicos	10 horas	3	8
Ítem de trabajo	Creación de simuladores de cooperativas	10 horas	3	13
Ítem de trabajo	Avance de documentación del proyecto	10 horas	3	8
Sprint	Sprint #4	0 horas	4	29
Ítem de trabajo	Creación y configuración del bus de servicios empresarial	24 horas	4	13
Ítem de trabajo	Conexión de los simuladores con el bus de servicios empresarial	16 horas	4	8
Ítem de trabajo	Conexión del bus de servicios empresariales con el aplicativo móvil.	8 horas	4	8
Sprint	Sprint #5	0 horas	5	24
Ítem de trabajo	Seguridad de la aplicación, cifrado de datos.	6 horas	5	5

Ítem de trabajo	Creación de un sitio web para la administración de los simuladores	8 horas	5	3
Ítem de trabajo	Creación de un sitio web para administración de la aplicación	12 horas	5	8
Ítem de trabajo	Avance de documentación del proyecto	10 horas	5	8
Sprint	Sprint #6	0 horas	6	16
Ítem de trabajo	Realización del capítulo IV y V del documento del proyecto	20 horas	6	8
Ítem de trabajo	Pruebas del funcionamiento del proyecto	20 horas	6	8

3.2.1. *Sprint* 1

En la primera iteración se realizará la introducción y el marco teórico de la documentación del proyecto. En los que abarca toda la presunción necesaria para entender cómo se va a elaborar este proyecto.

En este *Sprint* además se empieza con el desarrollo de la aplicación móvil. Para empezar esta actividad, se requiere que se encuentre instalado el entorno de desarrollo *Visual Studio Professiona*l.

Adicional se debe instalar *Xamarin Studio* para *Android*, para que se pueda diseñar las pantallas para este sistema operativo.

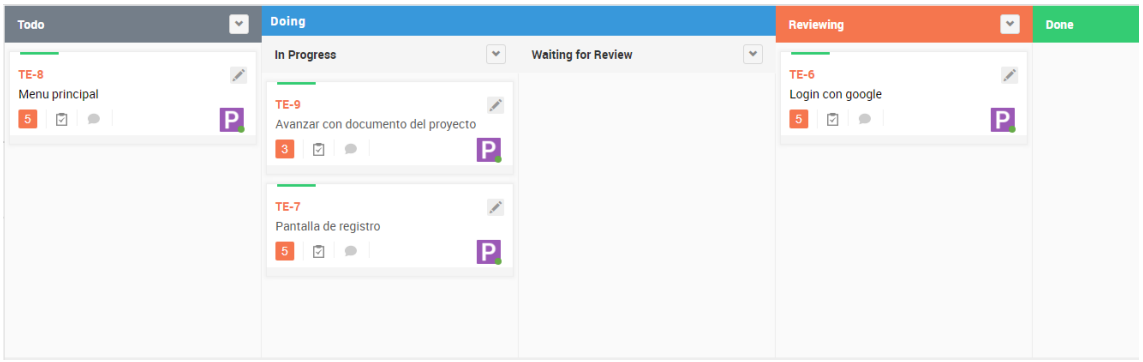


Figura 11. Tablero *Scrum* de primer *Sprint*

Tabla 2.

Primer Sprint Backlog

Tipo Ítem	**Sprint**	**Nombre de la tarea**	**Trabajo**	**Inicio**	**Fin**
Ítem de trabajo	1	*Login* con Google	10 horas	Lunes 6/2/17	Miércoles 8/2/17
Ítem de trabajo	1	Pantalla de registro	10 horas	Jueves 8/2/17	Viernes 10/2/17
Ítem de trabajo	1	Menú principal	10 horas	Viernes 10/2/17	Jueves 14/2/17
Ítem de trabajo	1	Realizar Introducción y Marco Teórico del documento.	12 horas	Martes 14/2/17	Jueves 16/2/17

Tabla 3.

Primera historia de usuario

Historia de usuario	
Número de historia de usuario: 1	**Usuario:** Cliente
Nombre de la historia: Inicio de sesión con Google	

Prioridad de negocio: Alta	**Riesgo de desarrollo:** Baja
Puntos Estimados: 1	**Iteración asignada:** 1
Programador Responsable: Juan José Ponce	

Descripción:

- Cuando arranque la *app*, se permita ingresar utilizando las credenciales de Google. En caso de que ya se encuentre registrado el cliente en el teléfono, no va a ser necesario que se vuelva a ingresar los datos de autenticación, porque estarán almacenados en una base de datos. Además, si el cliente tiene varias cuentas de Google ingresadas en el teléfono se permitirá seleccionar con cuál de ellas se desea ingresar a la *App*, ya que posteriormente quedará registrada la cuenta con la que se ingresó, y no se volverá a pedir credenciales.
- Por motivos de seguridad es mejor no almacenar contraseñas ni usuarios en la base de datos de la aplicación, es por lo que se ha optado por utilizar el sistema de autenticación de Google, con el que se evitan los administradores de la aplicación tener que proteger información sensible.

Validación:

- Para validar este requerimiento primero se debe iniciar en el teléfono y registrar una cuenta de Google, luego intentar ingresar a la aplicación, y no debe pedir contraseña, porque ya está almacenada en el teléfono.
- Se debe validar que cuando el teléfono no tenga registrada ninguna cuenta se solicite registrar una nueva cuenta para ingresar a la *App*.
- Se debe registrar varias cuentas en el teléfono para que cuando se ingrese a la aplicación, se solicite seleccionar la cuenta que se usará para el ingreso a la aplicación móvil.
- Una vez consultados los datos de Google, por medio del Id de registro valida si el usuario se encuentra registrado en la aplicación, en caso contrario este direccionará a la pantalla de registro de clientes, y ahí se almacena información relevante del cliente como por ejemplo nombre, género, estado civil y fecha de nacimiento. Estos datos son tomados de los que el cliente registro en su cuenta de Google.

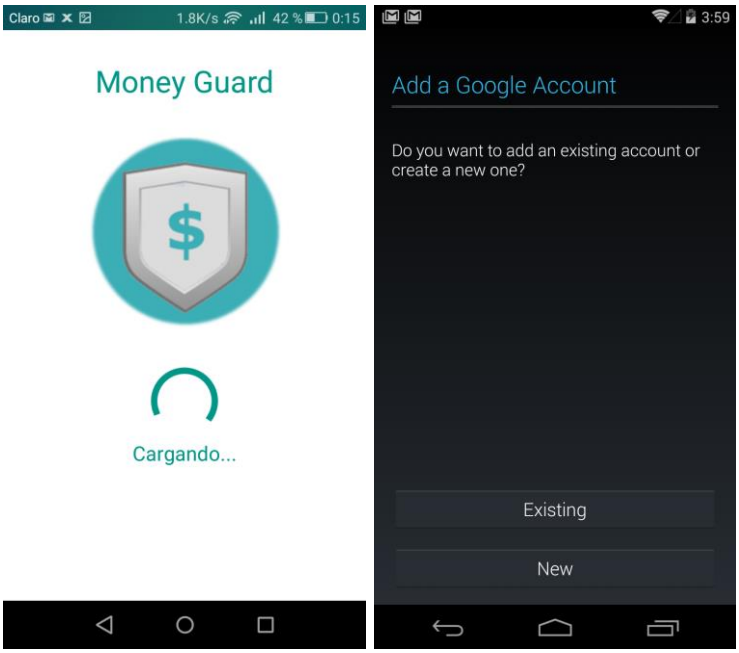


Figura 12. Ingreso a la aplicación utilizando Google *API*

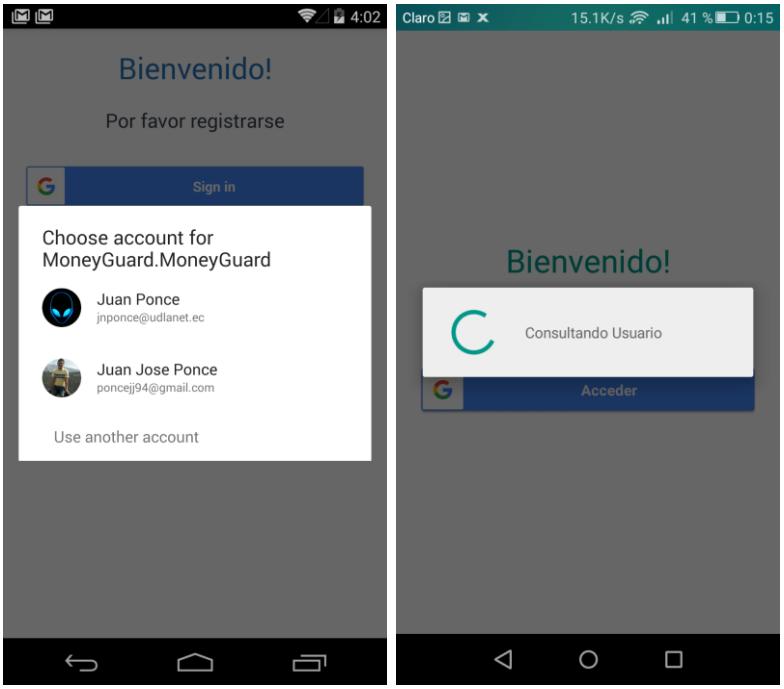


Figura 13. Ingreso a la *App* cuando usuario está registrado

Tabla 4.

Segunda historia de usuario

Historia de usuario	
Número de historia de usuario: 2	**Usuario:** Cliente
Nombre de la historia: Pantalla de registro	
Prioridad de negocio: Media	**Riesgo de desarrollo:** Baja
Puntos Estimados: 3	**Iteración asignada:** 1
Programador Responsable: Juan José Ponce	
Descripción: • Registrar el número de identificación del cliente, y la cooperativa de ahorro y crédito que se desee asociar a la cuenta del cliente. • Las opciones de las cooperativas de ahorro y crédito serán aquellas que se encuentren registradas en las bases de datos de la aplicación.	
Validación: • Para la validación de esta historia de usuario, se debe ingresar a la *App* con un usuario de Google que no se haya registrado anteriormente. Cuando se cumple esta condición, se direccionará a la pantalla de registro inicial de cliente. Caso contrario, se direccionará hacia la pantalla de ingreso de *PIN*. • Se debe tener una cuenta que la cooperativa debe estar dentro de las opciones que se muestran al cliente. Cuando se ingresa el número de identificación, se realice una consulta sobre los datos del cliente registrados la entidad y verifique la existencia del cliente. Cuando cumpla el requisito los datos de registro se almacenarán y se procederá con la creación del *PIN*. Caso contrario se presentará un error indicando que no existen cuentas asociadas con la identificación.	

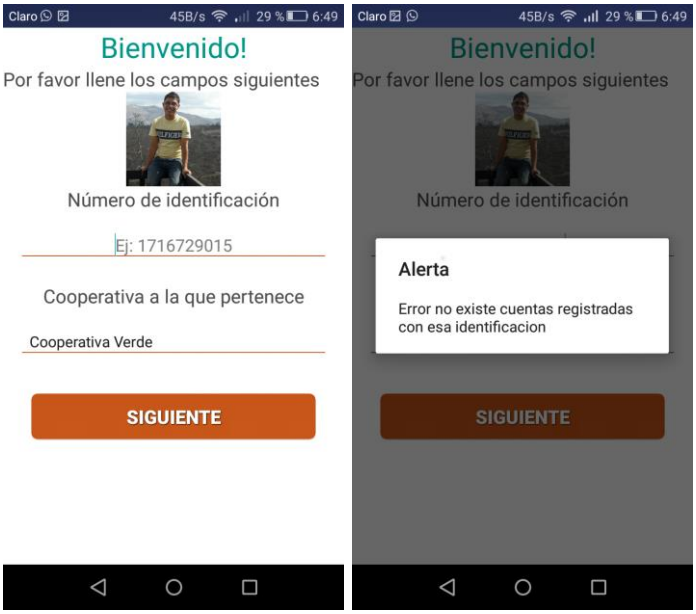


Figura 14. Pantalla de la *App* Móvil en el primer registro de cliente.

Tabla 5.

Tercera historia de usuario

Historia de usuario	
Número de historia de usuario: 3	**Usuario:** Cliente
Nombre de la historia: Menú principal	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Baja
Puntos Estimados: 5	**Iteración asignada:** 1
Programador Responsable: Juan José Ponce	

Descripción:
• La aplicación debe tener una pantalla que permita desplazarse a otras pantallas por medio de un menú con varias opciones:
  ○ Consulta de Movimientos.
  ○ Transferencias
  ○ Pago de servicios
  ○ Pago de tarjetas
  ○ Añadir cuentas
  ○ Salir	

• El menú principal debe mostrar al cliente todas las cuentas de las cooperativas que tenga asociadas a la cuenta de la aplicación móvil, y las tarjetas de crédito asociadas a las cuentas registradas. • Cuando se presione sobre una de las cuentas mostradas en el menú de inicio se debe mostrar el detalle de movimientos de dicha cuenta. • Cuando se presione sobre una de las tarjetas de crédito en el menú principal automáticamente se redireccionará a la pantalla de Pago de tarjetas.
Validación: • Para validar está función se debe ingresar a la aplicación, con un usuario registrado. Una vez ingresados los datos correctos se despliega la pantalla principal, la cual debe mostrar todas las cuentas asociadas al cliente y sus tarjetas de crédito. Si se presenta un error se mostrará un mensaje indicando que existió un error al cargar la pantalla. • Cuando se presiona sobe una de las cuentas del cliente se debe direccionar a la pantalla de consulta de movimientos y se debe desplegar los últimos diez movimientos que se realizó con dicha cuenta. • Cuando se presione sobre una de las tarjetas de crédito del cliente en la pantalla principal se debe desplegar el formulario para realizar el pago de la tarjeta. • Al lado derecho se encuentra el menú de opciones de todas las operaciones que puede realizar la aplicación.

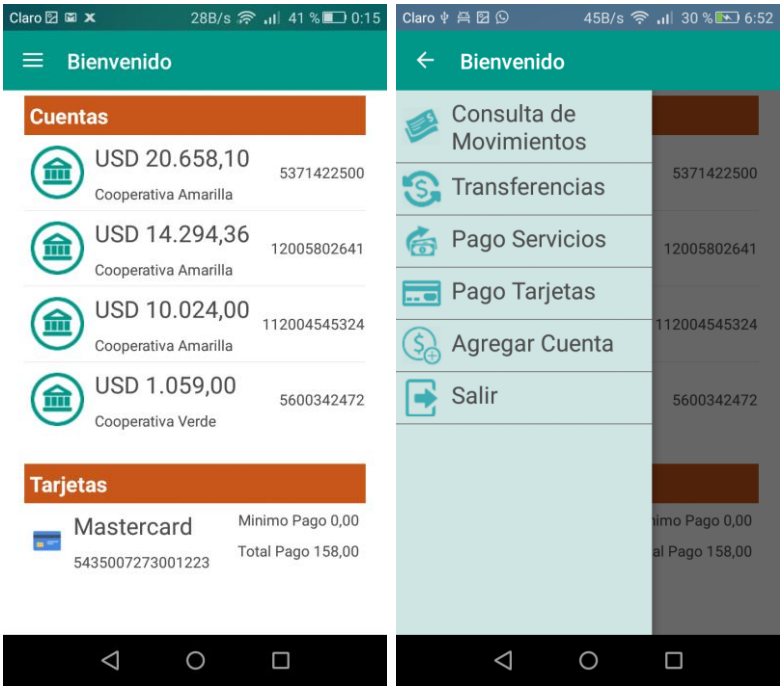


Figura 15. Pantalla de menú principal y menú lateral

3.2.2. *Sprint* 2

La segunda iteración consiste en continuar desarrollando este documento conforme se avanza con el desarrollo del prototipo.

Entre las actividades de desarrollo se va a realizar el diseño de la pantalla de consulta de saldos en las cuentas bancarias asociadas al cliente, pantalla de consulta de movimientos por cada cuenta que se selecciona, pantalla para realizar transferencias de valores, las cuales pueden ser a la misma cooperativa o a distintas, pantalla para el pago de servicios básicos, también se diseñará una pantalla para asociar más cooperativas a la cuenta del cliente y se agregará una autenticación de doble factor, que se utilizará un *PIN* para la validación de usuario.

Tabla 6.

Segundo Sprint Backlog del proyecto.

Tipo de Ítem	Número Sprint	Nombre	Trabajo	Inicio	Fin
Ítem de Trabajo	2	Pantalla de consulta de movimientos	8 horas	Lunes 20/2/17	Martes 21/2/17
Ítem de Trabajo	2	Pantalla de pago de servicios	8 horas	Martes 21/2/17	Jueves 23/2/17
Ítem de Trabajo	2	Pantalla de Transferencias	8 horas	Jueves 23/2/17	Viernes 24/2/17
Ítem de Trabajo	2	Pantalla para agregar Cuenta	8 horas	Viernes 24/2/17	Jueves 28/2/17
Ítem de Trabajo	2	Pantalla de pin de ingreso	8 horas	Martes 28/2/17	Miércoles 1/3/17
Ítem de Trabajo	2	Avance documentación del proyecto	8 horas	Miércoles 1/3/17	Viernes 3/3/17

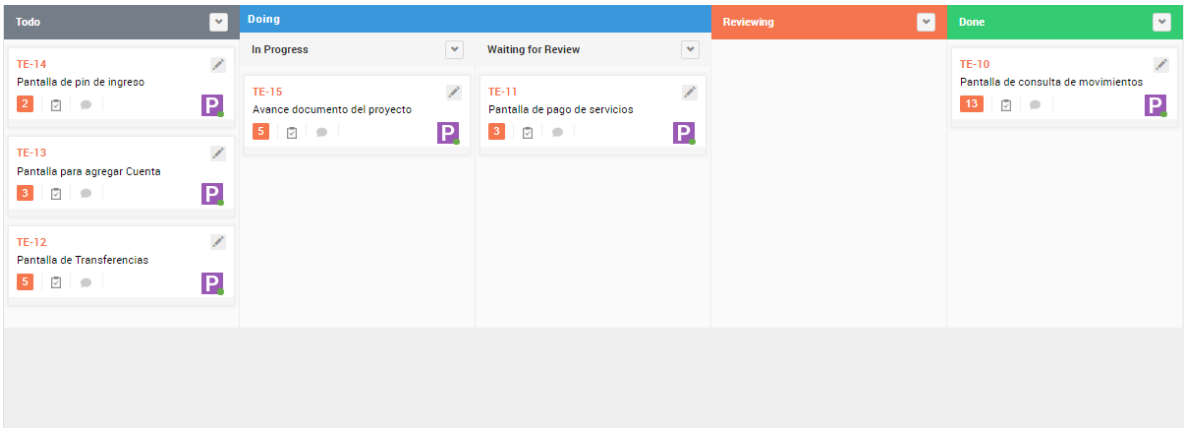


Figura 16. Tablero *Scrum* de segundo *Sprint*

Tabla 7.

Cuarta historia de usuario

Historia de usuario	
Número de Historia de usuario: 4	**Usuario:** Cliente
Nombre de la historia: Pantalla de consulta de movimientos	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Baja
Puntos Estimados: 5	**Iteración asignada:** 2
Programador Responsable: Juan José Ponce	
Descripción: • Se requiere que exista una pantalla de consulta de movimientos en la aplicación. • Está pantalla debe tener una cabecera, la cual muestra información importante como el número de cuenta, el nombre de la cooperativa a la pertenece, el saldo disponible actualmente. • Y el detalle debe mostrar datos como si fue un crédito o un débito, la fecha de la transacción, el valor, la descripción de la transacción y el saldo disponible después del movimiento bancario. • Se puede mostrar hasta los diez últimos movimientos realizados anteriormente con esa cuenta.	
Validación: • Para validar está función, el cliente debe ingresar a la pantalla de consulta de movimientos sea por la pantalla principal o por el menú lateral. Una vez que se seleccione la cuenta para consultar los movimientos, la pantalla debe cargar con los datos de la cabecera y se debe consultar el detalle inmediatamente a la cooperativa. También se debe validar que los datos mostrados en pantalla sean coherentes a los movimientos realizados en la cuenta.	

Figura 17. Pantalla de consulta de movimientos de la *App*.

Tabla 8.

Quinta historia de usuario

Historia de usuario	
Número de historia de usuario: 5	**Usuario:** Cliente
Nombre de la historia: Pantalla de pago de servicios	
Prioridad de negocio: Media	**Riesgo de desarrollo:** Baja
Puntos Estimados: 5	**Iteración asignada:** 2
Programador Responsable: Juan José Ponce	

• **Descripción:**
• En la aplicación debe existir una pantalla para realizar el pago de servicios básicos como:
  ○ Luz
  ○ Agua
  ○ Teléfono	

• Se desea que sea simple, que tan solo se deba seleccionar el tipo de servicio, y se ingrese el código de suministro o número de teléfono, y se presente la factura al cliente. • Cuando ya se ha mostrado la factura del cliente, debe existir la posibilidad de pagar el total de la factura, se presiona el botón y se muestra un formulario en el que se seleccionará la cuenta a la que se realizará el débito.
Validación: • Para validar esta funcionalidad, primero se debe tener un suministro válido registrado en el simulador de servicios básicos implementados. • Se debe tener una factura vigente, que tenga valores pendientes a cancelar. • Para realizar el pago el cliente debe tener saldo disponible en la cuenta bancaria de la que se va a realizar el débito.

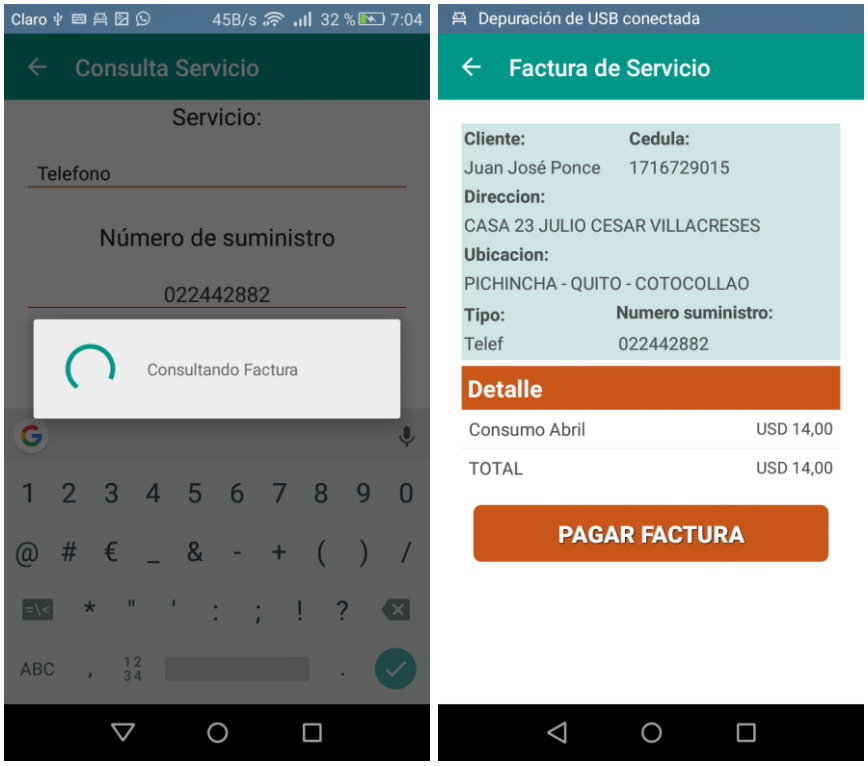


Figura 18. Pantalla de consulta de factura de servicios.

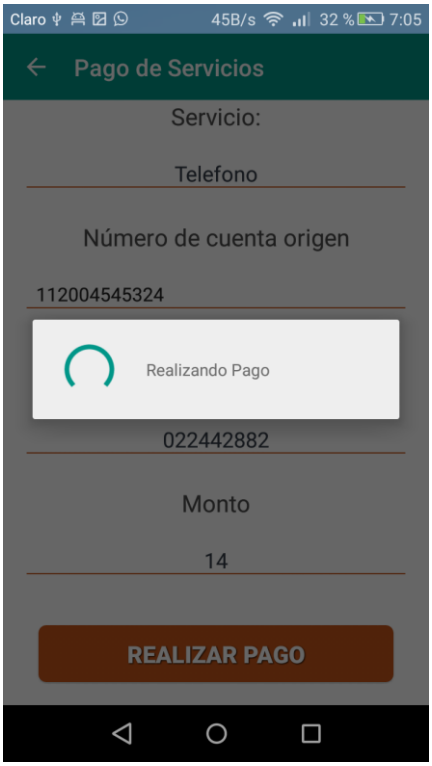


Figura 19. Pantalla de pago de servicios en la aplicación.

Tabla 9.

Sexta historia de usuario

Historia de usuario	
Número de historia de usuario: 6	**Usuario:** Cliente
Nombre de la historia: Pantalla de transferencias entre cuentas	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos Estimados: 5	**Iteración asignada:** 2
Programador Responsable: Juan José Ponce	
Descripción: • Se requiere que exista un formulario en el aplicativo móvil, para ingresar los datos requeridos y se pueda procesar una transferencia de dinero entre diferentes cuentas, que pueden existir en la misma cooperativa, o en distintas cooperativas.	

- En el formulario debe existir un campo para poder seleccionar la cooperativa destino, campo para ingresar el número de cuenta del beneficiario, el número de identificación beneficiario, seleccionar el tipo de cuenta (Ahorros o corriente), monto de la transferencia e ingresar una pequeña descripción de la transferencia.
- Está historia de usuario también contempla que se debe crear un Log en la base de datos de la aplicación en el que se guarde todas las transacciones realizadas por medio del aplicativo móvil con el fin de que se pueda obtener información estadística de estos datos.

Validación:

- Para validar esta historia de usuario se debe ingresar a la pantalla de transferencias, se debe ingresar los datos requeridos en el formulario sobre el cliente beneficiario, monto y descripción, si los datos son correctos se debe realizar la transferencia correctamente.
- La transferencia se realiza correctamente cunado se hizo un débito sobre la cuenta del emisor y un crédito sobre la cuenta del beneficiario. Y se procede a registrar el movimiento de los valores en ambas cuentas y en el log de transacciones propio del aplicativo móvil.
- En el caso de que los datos sean incorrectos se debe desplegar un mensaje de error indicando que no se puedo procesar la transferencia. En el caso de que se ha efectuado el débito sobre la cuenta del emisor se debe proceder con el reverso de dicho débito.
- En caso de que uno de los servicios de las cooperativas no esté disponibles, la operación tendrá que desplegar un mensaje de error indicando el inconveniente, ya que debe prevalecer la atomicidad en todas las transacciones que se realicen a través del servicio de la aplicación móvil.

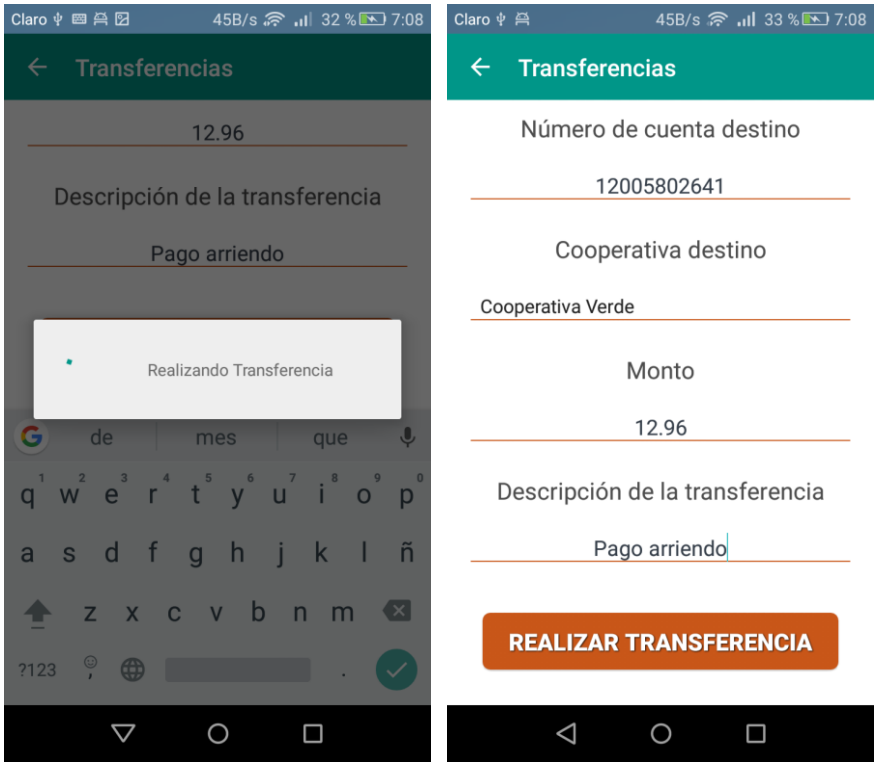


Figura 20. Pantalla para realizar transferencias a distintas cuentas.

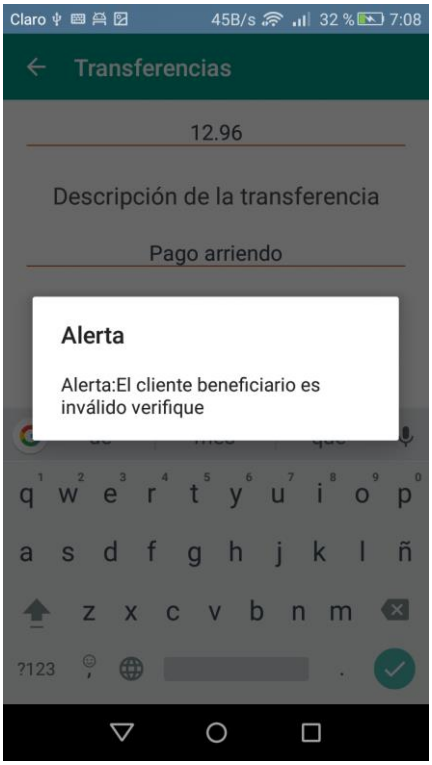


Figura 21. Pantalla de error en caso de transferencia no exitosa.

Tabla 10.

Séptima historia de usuario

Historia de usuario	
Número de historia de usuario: 7	**Usuario:** Cliente
Nombre de la historia: Pantalla para agregar cuentas asociadas al usuario	
Prioridad de negocio: Media	**Riesgo de desarrollo:** Baja
Puntos Estimados: 5	**Iteración asignada:** 2
Programador Responsable: Juan José Ponce	
Descripción: • Se requiere que exista una pantalla en el aplicativo móvil, para que se puedan asociar más cooperativas al cliente, y se pueda utilizar las cuentas registradas a su nombre para realizar todas las operaciones que se pueden hacer dentro de la *App*. • Se debe ingresar el número de identificación del cliente y la cooperativa que se desea asociar. • Los datos de la nueva cuenta asociada al cliente se almacenan en la base de datos de la aplicación, para que cuando el cliente vuelva a iniciar sesión pueda obtener su nueva configuración.	
Validación: • Para validar esta historia de usuario se debe ingresar a la pantalla de “Agregar Cuenta” utilizando el menú que se encuentra en la parte derecha, en está pantalla, se debe ingresar el número de identificación del cliente que inicio sesión con su usuario, y se debe seleccionar una cooperativa que actualmente no está registrada al cliente y si disponga de una cuenta vigente dentro de la cooperativa. • En caso de que no se cumpla con las condiciones de uso de la pantalla se desplegará un mensaje de alerta indicando el inconveniente generado. • Cuando se agrega una cuenta correcta en una cooperativa válida la pantalla se cierra y se direcciona a la pantalla principal de la aplicación,	

y se muestran las nuevas cuentas encontradas en la nueva cooperativa asociada.

Figura 22. Pantalla para asociar más cuentas a la cuenta del cliente.

Tabla 11.

Octava historia de usuario

Historia de usuario	
Número de historia de usuario: 8	**Usuario:** Cliente
Nombre de la historia: Pantalla PIN de validación	
Prioridad de negocio: Media	**Riesgo de desarrollo:** Baja
Puntos Estimados: 3	**Iteración asignada:** 2
Programador Responsable: Juan José Ponce	
Descripción: • Se requiere que exista una pantalla en el aplicativo móvil, para que esté implementada seguridad de doble factor, es decir que además de que el cliente se registre con su usuario de Google también debe conocer el	

PIN de registro para acceder a los datos de las cuentas dentro del aplicativo, este *PIN* debe ser registrado junto con los datos iniciales del usuario.

- En el caso de que el *PIN* necesite ser cambiado debe contactarse con algún administrador de la plataforma, ya que solo el personal tendrá acceso desde el sitio web a realizar esta operación.

Validación:

- Para validar esta historia de usuario, el cliente se debe registrar en la aplicación, después ingresar sus datos, tendrá que registrar un PIN para su futuro ingreso. Este *PIN* debe ser conocido únicamente por el cliente, ya que esto garantizará mayor seguridad.
- Cada vez que vuelva a ingresar a la aplicación se le pedirá al cliente en *PIN* registrado, en el caso de que este sea incorrecto se mostrará un mensaje al cliente.
- Cuando se ingresa el *PIN* correcto que está asociado al usuario, se dispara automáticamente una petición, de las cuentas del cliente, y se consulta en cada cooperativa de las que el cliente tenga asociadas cuentas bancarias.

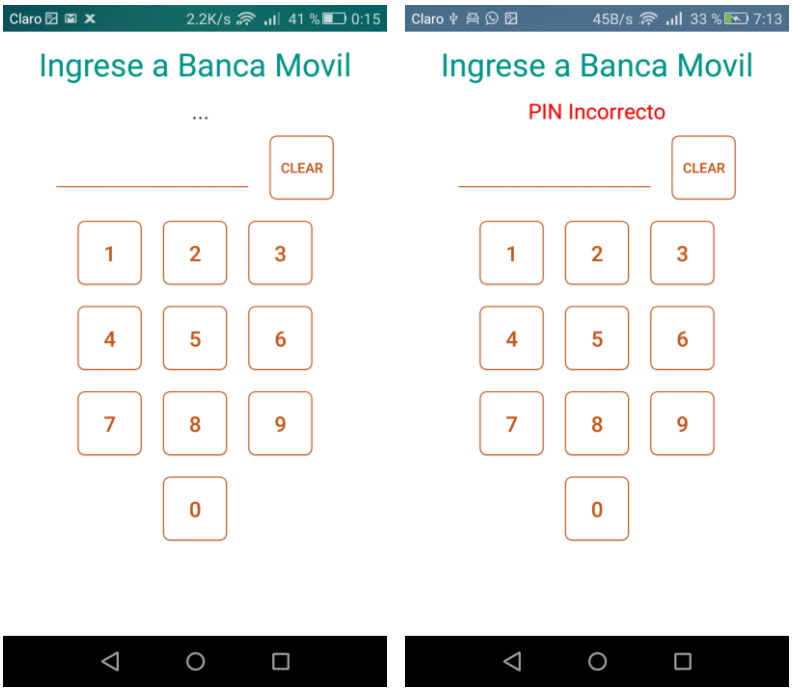


Figura 23. Pantalla de *PIN* de seguridad de doble factor.

3.2.3. Sprint 3

En este Sprint se propone realizar la conexión del aplicativo móvil con Azure, para que se pueda realizar las peticiones a un servidor publicado en la nube y que sea accesible desde internet. Además, en este Sprint se va a crear el bus de servicios empresarial de Azure que se encargará de realizar la comunicación entre los simuladores y el aplicativo móvil. También se desea crear las bases de datos y los servicios web que usarán los simuladores para la comunicación con el bus y el almacenamiento de los datos que se generen en los simuladores de las cooperativas.

Tabla 12.

Tercer Sprint Backlog del proyecto

Tipo de Ítem	**Número de Sprint**	**Nombre**	**Trabajo**	**Inicio**	**Fin**
Ítem de Trabajo	3	Conexión de aplicación móvil con Azure	16 horas	Lunes 6/3/17	Miércoles 9/3/17
Ítem de Trabajo	3	Creación de simulador de servicios básicos	10 horas	Miércoles 9/3/17	Lunes 13/3/17
Ítem de Trabajo	3	Creación de simuladores de cooperativas	10 horas	Lunes 13/3/17	Miércoles 15/3/17
Ítem de Trabajo	3	Avance del capítulo III del	10 horas	Miércoles 15/3/17	Viernes 17/3/17

		documento del proyecto			

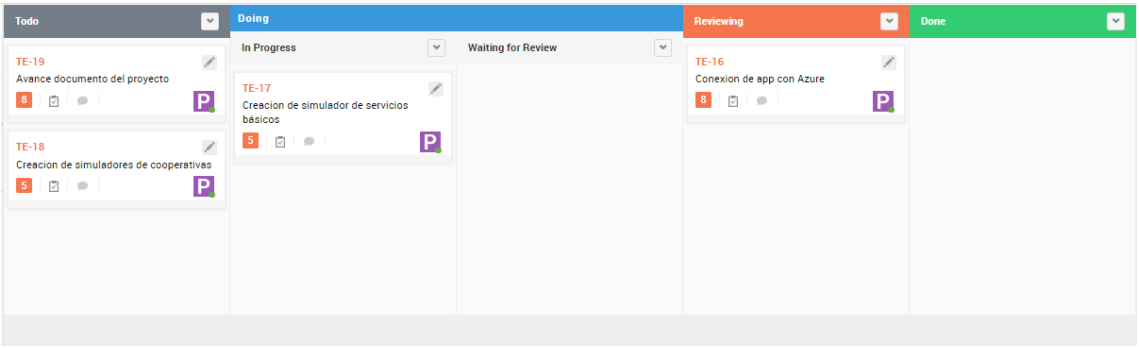


Figura 24. Tablero *Scrum* de tercer *Sprint*

Tabla 13.

Novena historia de usuario

Historia de usuario	
Número de historia de usuario: 9	**Usuario:** Cliente
Nombre de la historia: Conexión aplicación móvil con *Azure*	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos Estimados: 13	**Iteración asignada:** 3
Programador Responsable: Juan José Ponce	
Descripción: • Se requiere la conexión del aplicativo móvil con la nube de *Azure*, para que los datos que utiliza la aplicación móvil como por ejemplo los usuarios o las cooperativas registradas se encuentren registradas en bases de datos en la nube. • El servicio de bus empresarial que se propone como arquitectura. Se plantea un servicio de aplicativos móviles en *Azure*, el cual se encargarán de recibir todas las peticiones de los clientes que realicen desde sus dispositivos.	

- Se encargarán de orquestar las operaciones y direccionar hacia las distintas cooperativas que le pertenezca cada solicitud.
- El servicio tendrá acceso a las bases de datos propias del aplicativo, para la consulta de usuarios, consulta de cooperativas registradas, consulta de catálogos, inserción del log de transacciones realizadas e inserción en el log de errores que se presentan en el aplicativo.

Validación:

- Para validar esta historia de usuario la aplicación debe consultar los catálogos y las cooperativas asociadas al esquema para que esos datos se almacenen en memoria mientras se utiliza la aplicación.
- Se debe además poder realizar todas las operaciones de la aplicación sin presentar ningún error, se realizaría las transacciones y estas deberían haber quedado registradas en el log de transacciones.
- Al servicio está publicado en la nube se debería utilizar la aplicación en cualquier momento del día y está tendría que responder cualquier solicitud del cliente.

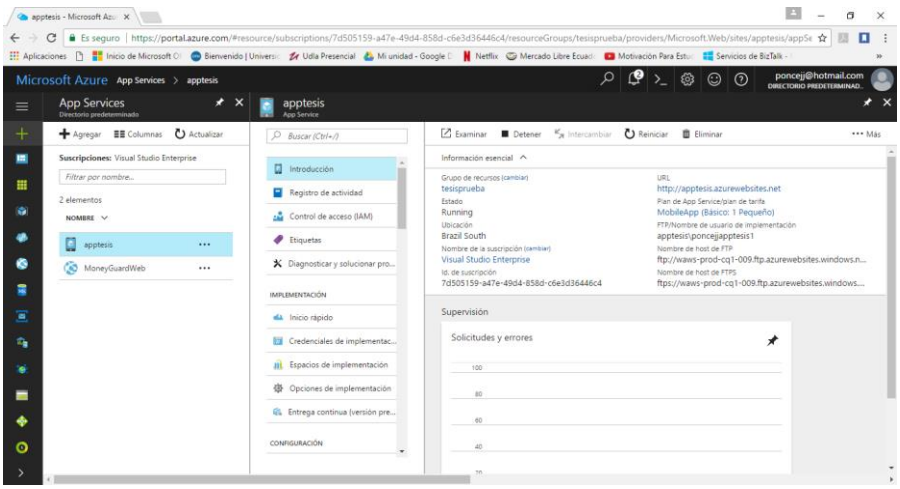


Figura 25. Servicio móvil en la plataforma de *Azure*

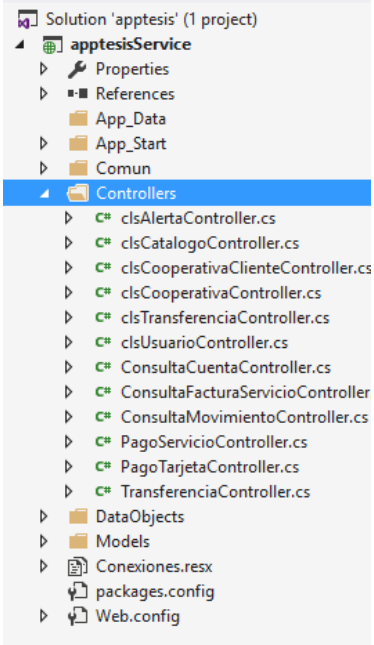


Figura 26. Imagen de la solución del servicio "AppTesis".

Tabla 14.

Decima historia de usuario

Historia de usuario	
Número: 10	**Usuario:** Cliente
Nombre de la historia: Creación de simuladores de cooperativas de ahorro y crédito en *Azure*	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos Estimados: 13	**Iteración asignada:** 3
Programador Responsable: Juan José Ponce	
Descripción: • Se requiere crear simuladores de cooperativas de ahorro y crédito, para que siempre estén escuchando solicitudes provenientes del bus de servicios empresarial, y se puedan procesar las peticiones, según la operación que se desee realizar. • Cada simulador debe contar con su propia base de datos, en donde se almacenará la información de sus clientes, cuentas, movimientos, etc.	

• Los simuladores deben estar publicados en Azure en un servicio en la nube. Al igual que sus bases de datos.
Validación: • Para validar esta historia de usuario, se debe poder realizar todas las operaciones implementadas sobre el simulador de la cooperativa, es decir: Consulta de cuentas, consulta de movimientos, realizar débitos, realizar créditos, pago de tarjetas.

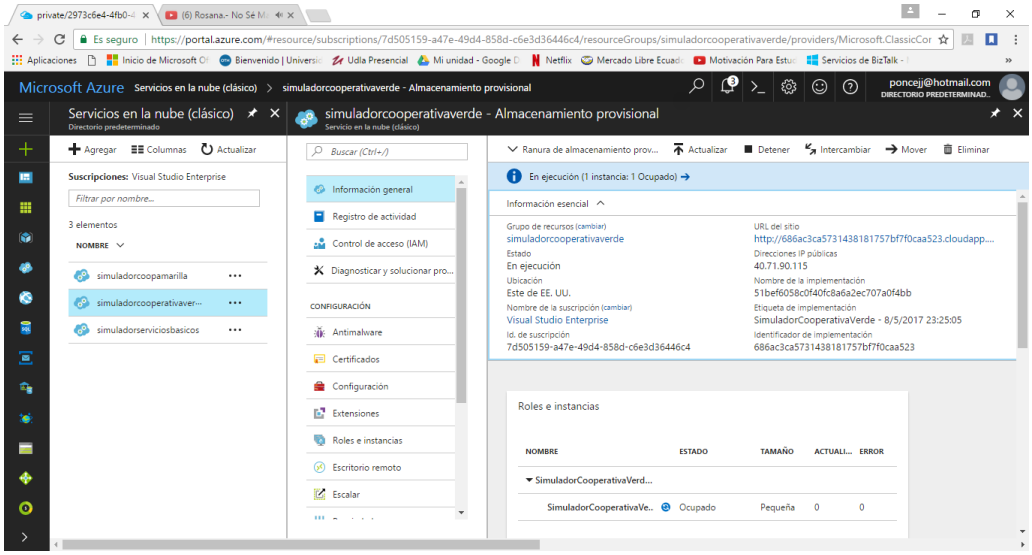


Figura 27. Simuladores publicados y en ejecución en *Azure*.

Tabla 15.

Onceava historia de usuario

Historia de usuario	
Número: 11	**Usuario:** Cliente
Nombre de la historia: Creación de simulador de servicios básicos en *Azure*	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos Estimados: 13	**Iteración asignada:** 3
Programador Responsable: Juan José Ponce	
• **Descripción:**	

• Se requiere crear un simulador de servicios básicos, para que se pueda utilizar en las pruebas de pago de servicios básicos utilizando la aplicación desarrollado. Siempre debe estar escuchando las peticiones asociadas con este servicio que sean provenientes del bus de servicios empresarial. Estas solicitudes deben ser procesadas por el simulador, y debe devolver una respuesta al aplicativo para poder conocer el estado de la transacción. • Este simulador debe contar con su propia base de datos, en donde se almacenará la información de sus clientes, suministros, datos de facturas, etc. • Los simuladores deben estar publicados en *Azure* en un servicio en la nube, al igual que sus bases de datos, para que se puedan utilizaren todo momento.
Validación: • Para validar esta historia de usuario, se debe poder consultar las facturas del cliente cuando se ingresa un número de suministro correcto que esté asociado al cliente. Además, se debe poder realizar el pago de los valores pendientes, y la factura deberá actualizarse a un estado de pagado.

3.2.4. Sprint 4

Está cuarta iteración se propone configurar el bus de servicios empresarial para que con las conexiones con los simuladores y el aplicativo móvil se pueda realizar la comunicación, y que los mensajes que viajen a través de este servicio puedan llegar a su destino.

Tabla 16.

Cuarto Sprint Backlog

Tipo Ítem	**Número Sprint**	**Nombre Tarea**	**Trabajo Actual**	**Inicio**	**Fin**
Ítem de Trabajo	4	Creación y configuración del bus de servicios empresarial	24 horas	Lunes 20/3/17	Lunes 27/3/17
Ítem de Trabajo	4	Conexión de los simuladores con el bus de servicios empresarial	16 horas	Lunes 27/3/17	Miércoles 30/3/17
Ítem de Trabajo	4	Conexión del bus de servicios empresariales con el aplicativo móvil.	8 horas	Miércoles 30/3/17	Viernes 31/3/17

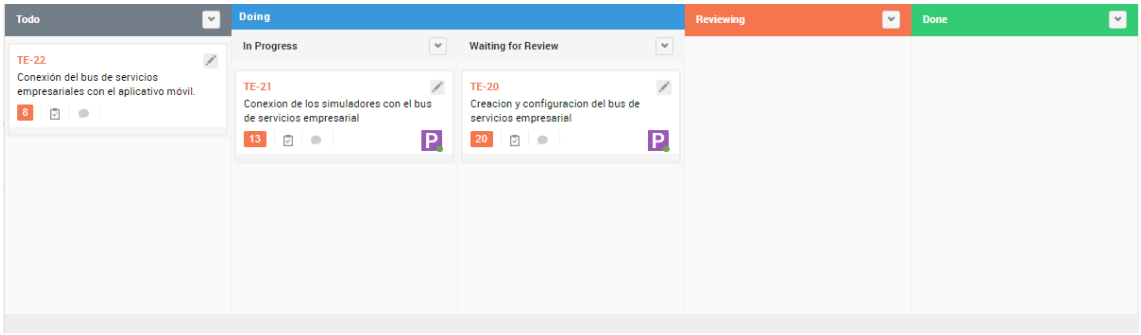


Figura 28. Tablero *Scrum* de cuarto *Sprint*

Tabla 17.

Doceava historia de usuario

Historia de usuario	
Número de historia de usuario: 12	**Usuario:** Cliente

<table>
<tr><td colspan="2">Nombre de la historia: Creación del bus de servicios empresarial en Azure</td></tr>
<tr><td>Prioridad de negocio: Alta</td><td>Riesgo de desarrollo: Baja</td></tr>
<tr><td>Puntos Estimados: 13</td><td>Iteración asignada: 4</td></tr>
<tr><td colspan="2">Programador Responsable: Juan José Ponce</td></tr>
<tr><td colspan="2">Descripción:
Se requiere crear y configurar un bus de servicios empresariales en la plataforma de Azure, según la arquitectura plantada para que sirva como medio de comunicación entre el servicio del aplicativo móvil y los simuladores creados en historias de usuario anteriores.
En el bus de servicios se crearán las colas para que los mensajes de petición y respuesta de cada usuario se almacenen temporalmente hasta que los simuladores los procesen y puedan dar una respuesta de vuela.</td></tr>
<tr><td colspan="2">Validación:
Para validar esta historia de usuario, se debe probar la comunicación que existe entre el aplicativo móvil y los simuladores, se puede crear peticiones desde la aplicación como por ejemplo la consulta de saldos en las cuentas, se debe enviar la petición y encolar el mensaje de respuesta para que se muestre en la interfaz del usuario.
Otra validación es ingresar a la plataforma de Azure, y verificar que este creado el bus de servicios, adicional deben existir las colas que se van a utilizar para cada uno de los simuladores.</td></tr>
</table>

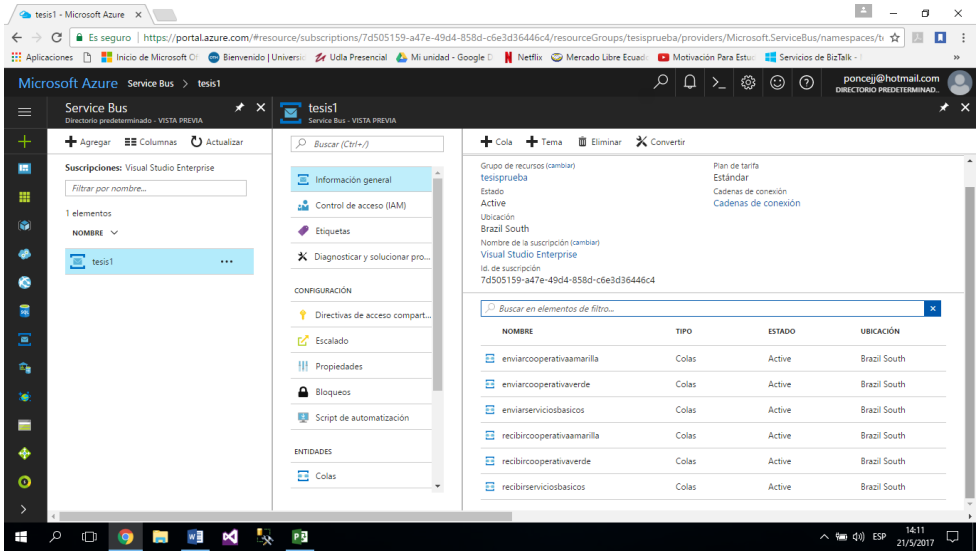


Figura 29: Implementación de Bus de Servicios Empresarial en *Azure*.

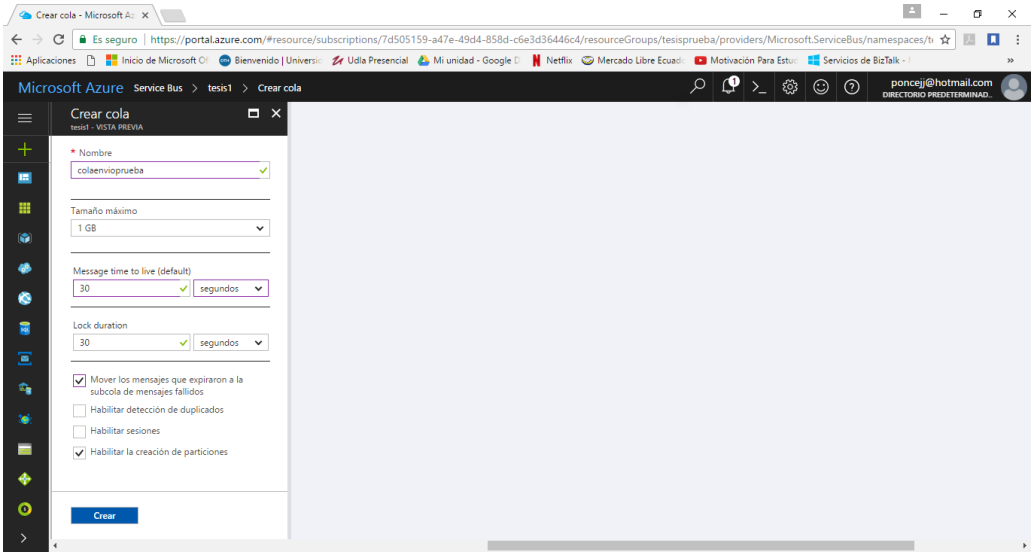


Figura 30. Creación de colas para ESB en *Azure*.

Tabla 18.

Treceava historia de usuario

Historia de usuario	
Número de historia de usuario: 13	**Usuario:** Cliente

<table>
<tr><td colspan="2">Nombre de la historia: Conexión de los simuladores con el bus de servicios empresarial.</td></tr>
<tr><td>Prioridad de negocio: Alta</td><td>Riesgo de desarrollo: Baja</td></tr>
<tr><td>Puntos Estimados: 13</td><td>Iteración asignada: 4</td></tr>
<tr><td colspan="2">Programador Responsable: Juan José Ponce</td></tr>
<tr><td colspan="2">Descripción:
Se requiere conectar el bus de servicios empresarial con los simuladores de las cooperativas de ahorro y crédito para que puedan recibir peticiones desde el aplicativo móvil.
Las colas deben ser configuradas para que el tiempo de vida de un mensaje dentro de la cola no sea mayor a 30 segundos, después será eliminado. Así se evitará que se sature la cola.</td></tr>
<tr><td colspan="2">Validación:
Para validar la funcionalidad implementada la aplicación debe poder realizar las operaciones de consulta, inserción y modificación de datos. Cuando se consulta los estados de cuenta, o el detalle de los movimientos se debe observar una respuesta correcta en la aplicación, cuando el usuario realiza un pago se debe registrar ese movimiento, etc. Si se efectúan correctamente todas las operaciones significa que funciona correctamente la conexión de las colas del bus de servicios empresarial con los simuladores.</td></tr>
</table>

```
1 reference
public void ProcesarCola()
{
    ClientEnvio = QueueClient.CreateFromConnectionString(connectionString, ColaEnvio);
    clsProcesadorMensajeria objProcesadorMensajeria = new clsProcesadorMensajeria();

    Console.WriteLine("Starting processing of messages");

    while(true)
    {
        BrokeredMessage message = ClientEnvio.Receive(TimeSpan.FromSeconds(1));
        if(message != null)
        {
            try
            {
                string strRespuesta =  objProcesadorMensajeria.procesarMensaje
                    (message.Properties["Tipo"].ToString(),message);
                if (strRespuesta != null)
                {
                    Console.WriteLine("Se recibio un mensaje...");
                    ProcesarRespuesta(strRespuesta);
                    message.Complete();
                }
            }
            catch(Exception ex)
            {
                Console.WriteLine("Error al procesar consulta cuenta cliente: " + ex.Message);
                message.Abandon();
            }
        }
    }
}
```

Figura 31. Código de procesamiento de mensajes por colas de ESB.

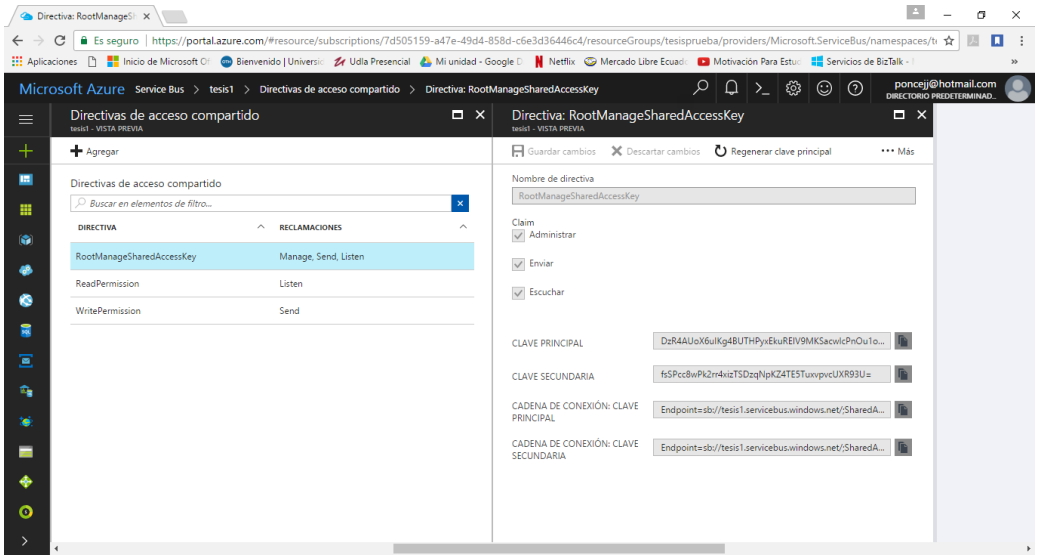


Figura 32. Cadena de conexión del Bus de Servicios Empresarial.

Tabla 19.

Catorceava historia de usuario

Historia de usuario	
Número de historia de usuario: 14	**Usuario:** Cliente
Nombre de la historia: Conexión del bus de servicios empresariales con el aplicativo móvil.	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Baja
Puntos Estimados: 13	**Iteración asignada:** 4
Programador Responsable: Juan José Ponce	
Descripción: • Para el funcionamiento del aplicativo se debe conectar el bus de servicios empresariales con el servicio que está publicado en la nube. De este modo las operaciones implementadas podrán ejecutarse en cada una de las cooperativas.	
Validación • Para validar esta historia de usuario el cliente tendría de ejecutar las operaciones de las pantallas antes mencionadas, sin que errores de ejecución en el *backend* se muestren.	

```
0 references
public string Get(string strNombreColaEnvio, string strNombreColaRespuesta,
    string strTipo, string strIdentificacionCliente, string strContrasena)
{
    QueueClient client = QueueClient.CreateFromConnectionString(this.connectionString, strNombreColaEnvio);
    string strMensajeRecibido = string.Empty;
    string result;
    try
    {
        BrokeredMessage message = new BrokeredMessage("Mensaje de envio");
        message.Properties["Tipo"] = strTipo;
        message.Properties["strIdentificacionCliente"] = strIdentificacionCliente;
        message.Properties["strContrasena"] = strContrasena;
        client.Send(message);
        strMensajeRecibido = clsFuncionesGenerales.RecibirRespuesta(strNombreColaRespuesta, strTipo);
    }
    catch (Exception ex)
    {
        result = "Error al obtener las cuentas registradas por usuario: " + ex.Message;
        return result;
    }
    result = strMensajeRecibido;
    return result;
}
```

Figura 33. Código para enviar mensajes desde aplicativo móvil.

3.2.5. *Sprint* 5

En la quinta iteración se prevé que se realizará todas las configuraciones y desarrollo necesario para poder brindar seguridad a la aplicación móvil, y a la comunicación de los paquetes de datos que se envíen y se reciban por este canal, ya que, al ser una propuesta para las cooperativas de ahorro y crédito, debe cumplir con ciertos estándares de seguridad.

Además, se desea configurar las colas del bus de servicios empresarial para que se pueda demostrar que el prototipo soporta una carga considerable, y que el servicio no se va a saturar.

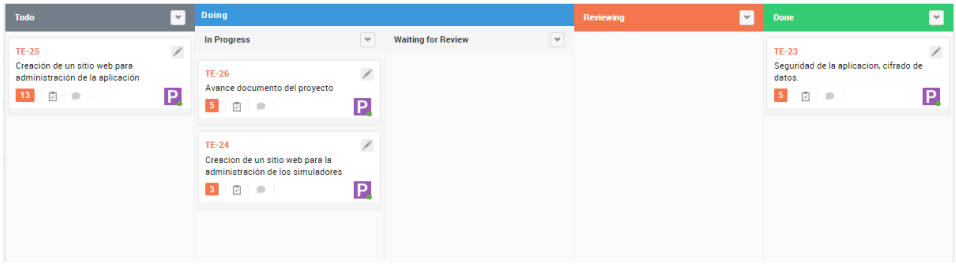


Figura 34. Tablero *Scrum* de quinto *Sprint*.

Tabla 20.

Quinto Sprint Backlog

Tipo de Ítem	**Número de Sprint**	**Nombre**	**Trabajo Actual**	**Inicio**	**Fin**
Ítem de Trabajo	5	Seguridad de la aplicación, cifrado de datos.	6 horas	Lunes 3/4/17	Martes 4/4/17
Ítem de Trabajo	5	Creación de un sitio web para la administración de los simuladores	8 horas	Martes 4/4/17	Miércoles 5/4/17

Ítem de Trabajo	5	Creación de un sitio web para administración de la aplicación	12 horas	Miércoles 5/4/17	Viernes 7/4/17
Ítem de Trabajo	5	Avance de documentación del proyecto	10 horas	Lunes 10/4/17	Viernes 14/4/17

Tabla 21.

Quinceava historia de usuario

Historia de usuario	
Número de historia de usuario: 15	**Usuario:** Oficial de seguridad
Nombre de la historia: Seguridad de la aplicación, cifrado de datos.	
Prioridad de negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos Estimados: 13	**Iteración asignada:** 5
Programador Responsable: Juan José Ponce	
Descripción: • El oficial de seguridad requiere que los datos que entran y salen de la aplicación móvil, necesitan estar cifrados, porque no se puede garantizar la seguridad de los canales de comunicación, entre el dispositivo móvil y los simuladores de las cooperativas de ahorro y crédito pequeñas. Es por lo que se debe implementar algoritmos de encriptación mediante una llave de cifrado y descifrado para cumplir con este requerimiento. La llave para el cifrado de los datos debe ser única por cada usuario de la aplicación, es por lo que se sugiere que sea un valor único de cada dispositivo.	
Validación: • Para validar esta funcionalidad, el *CISO* debe realizar pruebas de la aplicación, en el que se grabe un log con los datos cifrados y con los datos descifrados para validar que los datos enviados y recibidos se encuentran protegidos.	

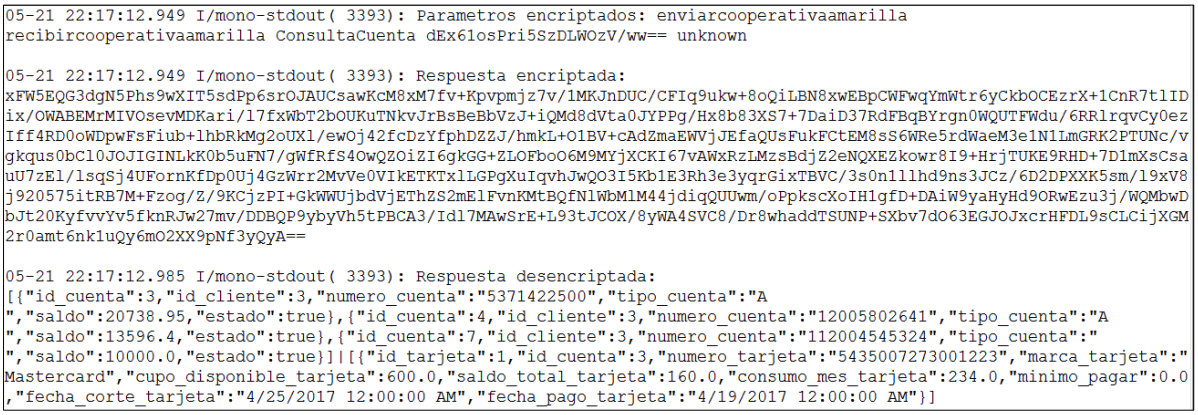


Figura 35. Consulta de cuenta (Datos encriptados y desencriptados).

```
12 references
internal static string Encriptar(string strTextoEncriptar, string strPassword)
{
    var algorithm = ObtenerAlgorithmo(strPassword);

    if (strTextoEncriptar == null || strTextoEncriptar == "") return "";

    byte[] bytEncriptados;
    using (ICryptoTransform encryptor = algorithm.CreateEncryptor(algorithm.Key, algorithm.IV))
    {
        byte[] bytAEncriptar = Encoding.UTF8.GetBytes(strTextoEncriptar);
        bytEncriptados = EncritacionMemoria(bytAEncriptar, encryptor);
    }
    return Convert.ToBase64String(bytEncriptados);
}
```

Figura 36. Código utilizado para la encriptación de los datos.

```
4 references
internal static string Desencriptar(string strTextoEncriptar, string strPassword)
{
    var algorithm = ObtenerAlgorithmo(strPassword);

    if (strTextoEncriptar == null || strTextoEncriptar == "") return "";

    byte[] descryptedBytes;
    using (ICryptoTransform decryptor = algorithm.CreateDecryptor(algorithm.Key, algorithm.IV))
    {
        byte[] encryptedBytes = Convert.FromBase64String(strTextoEncriptar);
        descryptedBytes = EncritacionMemoria(encryptedBytes, decryptor);
    }
    return Encoding.UTF8.GetString(descryptedBytes);
}
```

Figura 37. Código utilizado para desencriptar los datos.

Tabla 22.

Dieciseisava historia de usuario

Historia de usuario	
Número de historia de usuario: 16	**Usuario:** Administrador de la aplicación móvil
Nombre de la historia: Creación de un sitio web para la administración de la aplicación móvil.	
Prioridad de negocio: Media	**Riesgo de desarrollo:** Baja
Puntos Estimados: 8	**Iteración asignada:** 5
Programador Responsable: Juan José Ponce	
Descripción: • El administrador de la aplicación móvil requiere un sitio para poder ver obtener datos relevantes sobre el uso de los servicios, es decir saber cuántas transacciones se realizaron cada día utilizando la aplicación móvil. Además, desea saber todas las notificaciones relevantes sobre el funcionamiento del servicio que conecta a la aplicación con el bus de servicios. • En el sitio además debe existir pantallas para la administración de los usuarios que se encuentren registrados, para poder cambiar datos como el pin de ingreso, o eliminar cooperativas asociadas al cliente y este ya no desee que se muestre información en su cuenta. • Se debe poder administrar todas las cooperativas que están asociadas al esquema, cambiar los nombres de las colas, o la URL del bus que utiliza para cooperativa para la comunicación con el servicio móvil. • Los catálogos que utiliza el aplicativo en donde se encuentran opciones como el tipo de cedula, o el tipo de cuenta se debe poder agregar contenido, modificar o eliminar.	
Validación:	

- Para validar esta funcionalidad el administrador del sitio debe ingresar a la URL donde está publicado el sitio y validar cada una de las pantallas. Para la pantalla de catálogos se debe crear un nuevo catálogo, modificar uno existente y eliminar el catalogo creado. Se debe ir mostrando en la pantalla todos los cambios que se realizan con los catálogos.
- Para la pantalla de usuarios, solo se puede modificar y eliminar usuarios ya que no se puede crear nuevos porque es necesario que este proceso se realice desde la aplicación móvil. Para eliminar un usuario primero se debe eliminar todas las asociaciones que tenga el usuario con las cooperativas, porque en la base de datos existe relación entre las tablas. Igual todos los cambios que se realicen sobre los usuarios se deben mostrar en las pantallas de visualización de los datos.

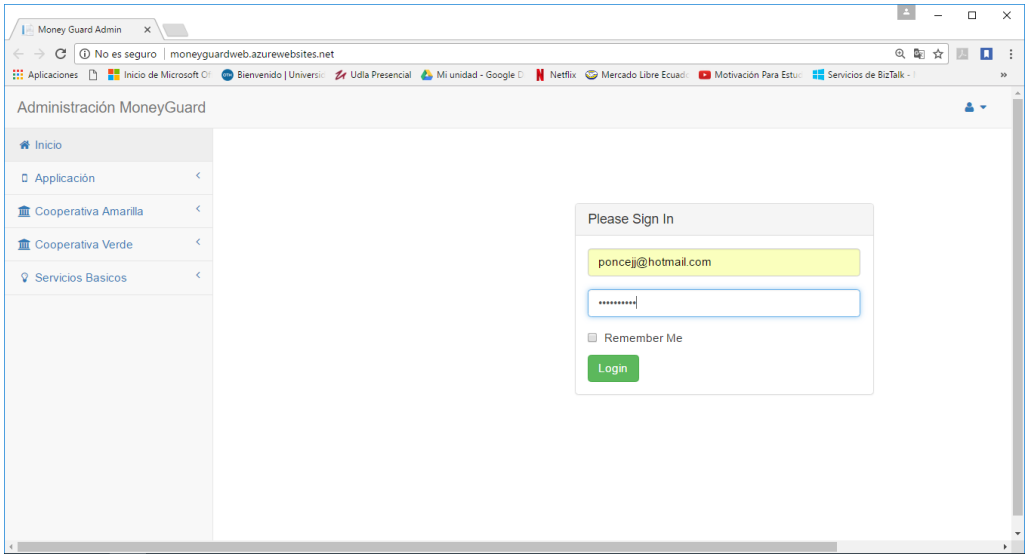


Figura 38. Pantalla para iniciar sesión en el sitio de administración.

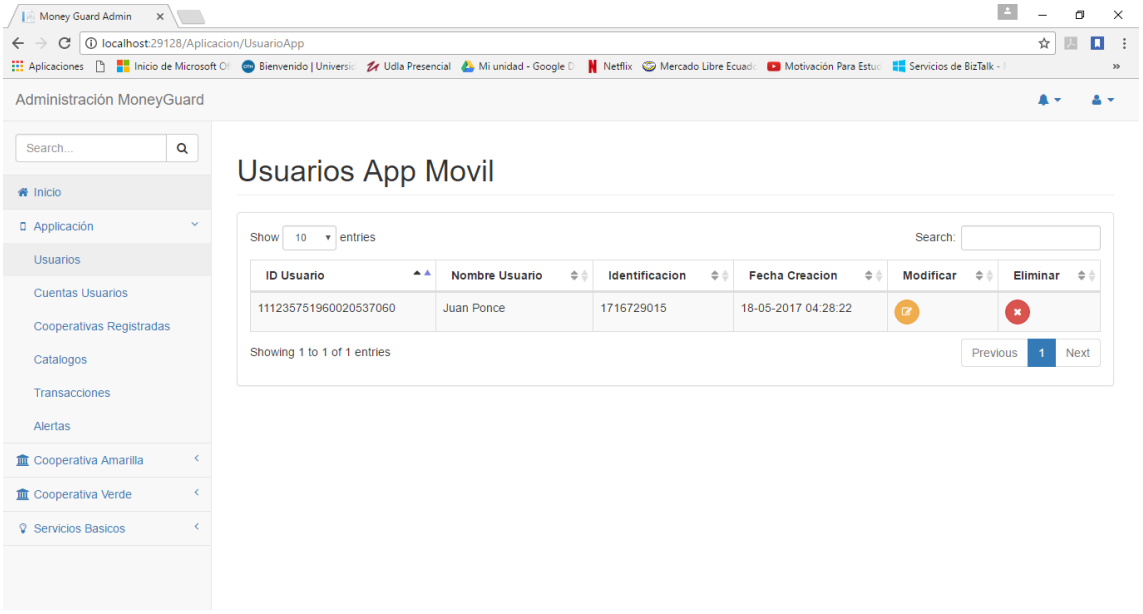


Figura 39. Consulta de usuarios registrados en la aplicación móvil.

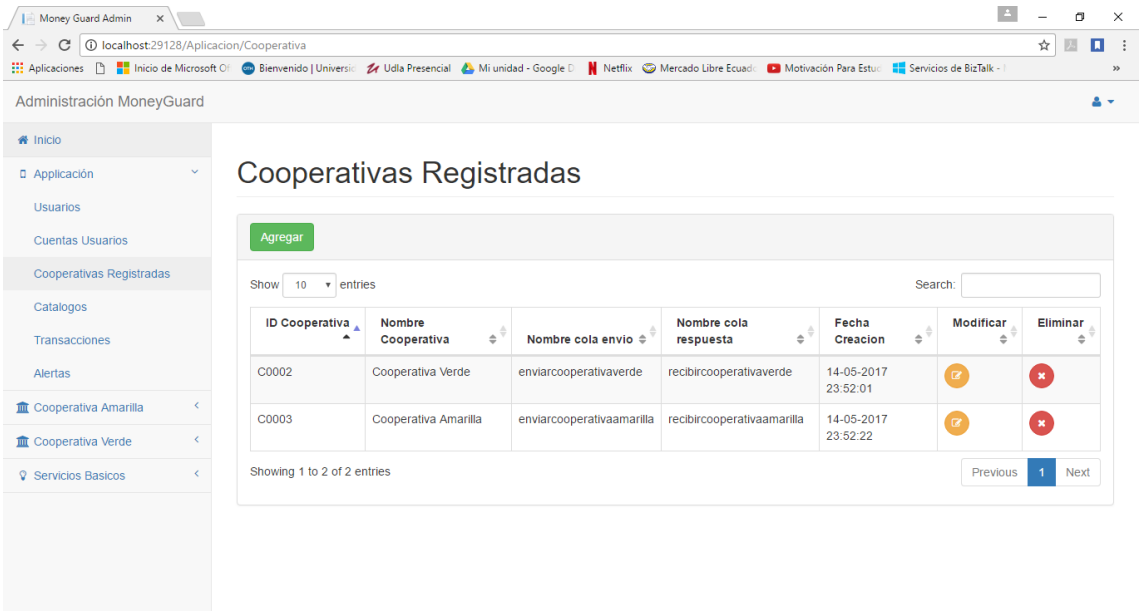


Figura 40. Cooperativas registradas en el esquema.

Tabla 23.

Diecisieteava historia de usuario

Historia de usuario

Número de historia de usuario: 17	**Usuario:** Administrador de la aplicación
Nombre de la historia: Creación de un sitio web para la administración de los simuladores.	
Prioridad de negocio: Media	**Riesgo de desarrollo:** Baja
Puntos Estimados: 8	**Iteración asignada:** 5
Programador Responsable: Juan José Ponce	
Descripción: • El administrador de la aplicación móvil requiere un sitio para poder administrar los datos de los simuladores de las cooperativas de ahorro y crédito creadas para probar el prototipo. • El sitio debe poder manejar las cuentas, los clientes, tarjetas de crédito y saldos pendientes por pagar, utilizando las mismas bases de datos a los que se conecta los simuladores que se encargan de responder las peticiones al aplicativo móvil. • El sitio tendrá las funciones básicas de crear, modificar, agregar y eliminar los datos del cliente, cuentas, tarjetas de crédito y saldos pendientes. Mientras que los movimientos solo se podrán consultar.	
Validación: • Para validar esta funcionalidad el administrador debe ingresar a la URL donde está publicado el sitio y validar cada una de las pantallas. • En la pantalla de cliente se debe mostrar información básica como por ejemplo nombres, correo electrónico, teléfono, número de identificación, etc. El botón de agregar usuario desplegará un formulario para ingresar un nuevo cliente. • En la pantalla de cuentas se mostrará todas las cuentas asociadas a los clientes, dando la posibilidad de que se pueda cambiar los datos como por ejemplo el saldo, o a que cliente se encuentra asociada dicha cuenta. También si se desea se puede eliminar una cuenta bancaria siempre y cuando no esté asociada a ningún cliente.	

- En la pantalla de tarjeta de crédito se podrá ver si un cliente dispone de una tarjeta de crédito, también se podrá ver los saldos disponibles, saldos pendientes, etc. Todos estos valores son modificables e incluso se puede crear más si es que el administrador del sitio web lo requiere.

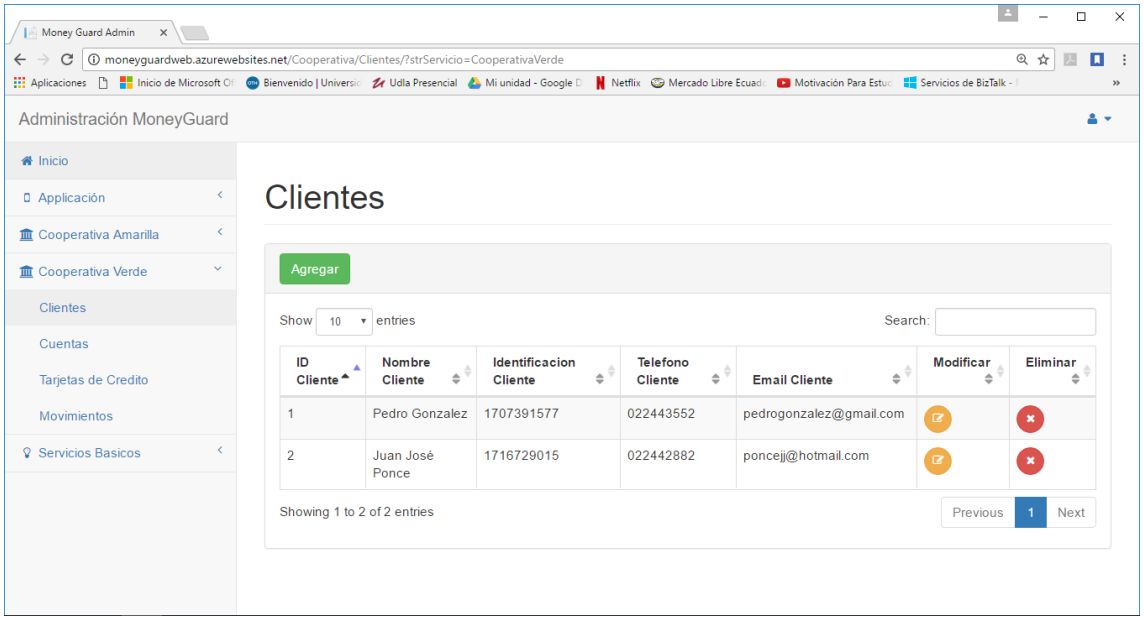


Figura 41. Datos de los clientes de la Cooperativa Verde

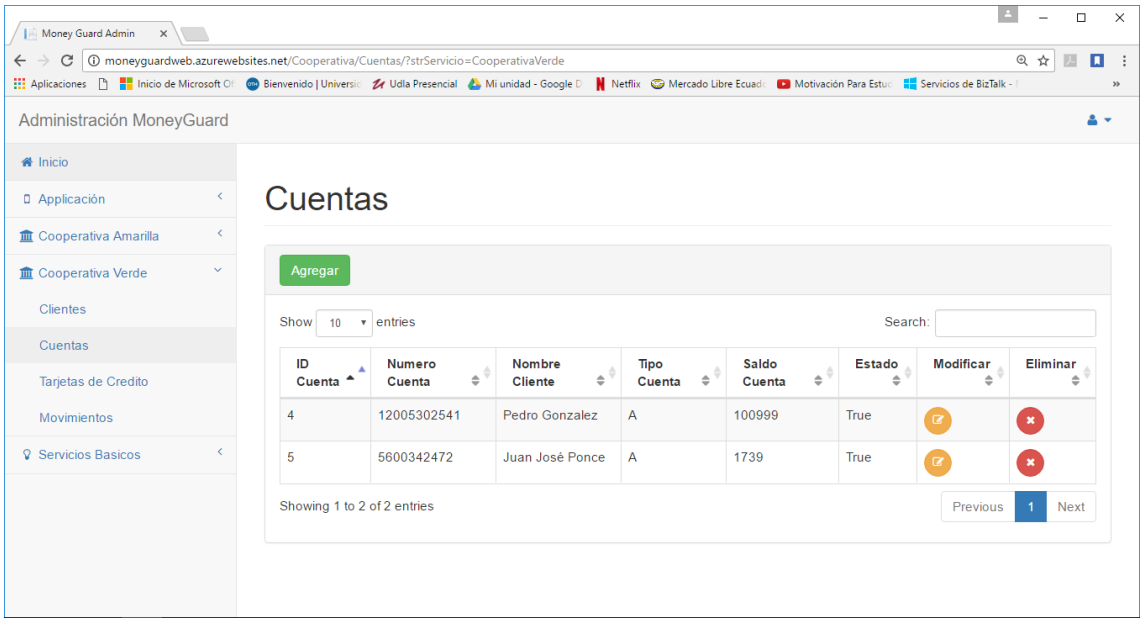


Figura 42. Datos de cuentas de Cooperativa Verde desde sitio.

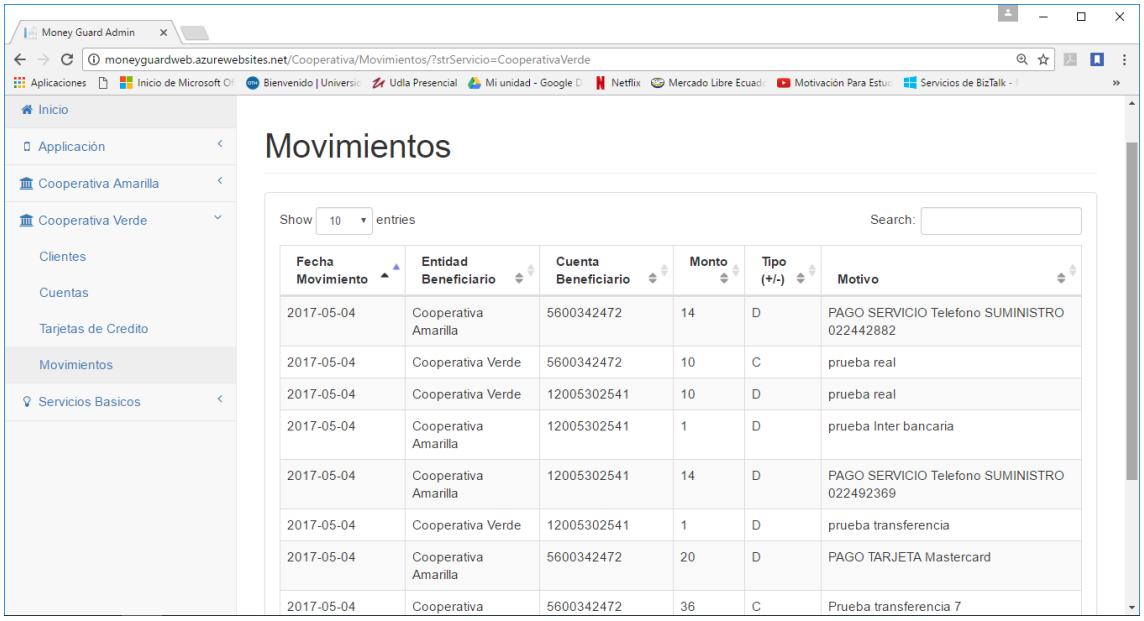


Figura 43. Pantalla de movimientos bancarios en la Cooperativa Verde.

3.2.6. *Sprint* 6

En esta última iteración se va a realizar las pruebas correspondientes para poder entregar el prototipo terminado, verificando que los objetivos estén cumplidos, que el funcionamiento de la aplicación sea correcto y que sea simple de probar. Se realizarán pruebas de carga, para saber exactamente el número de usuarios simultáneos pueden realizar transacciones por segundo. Adicionales pruebas unitarias para tener la certeza de que cada funcionalidad sirve adecuadamente. Se desea concluir con la documentación del proyecto. Realizando los dos últimos capítulos, y agregando anexos al documento.

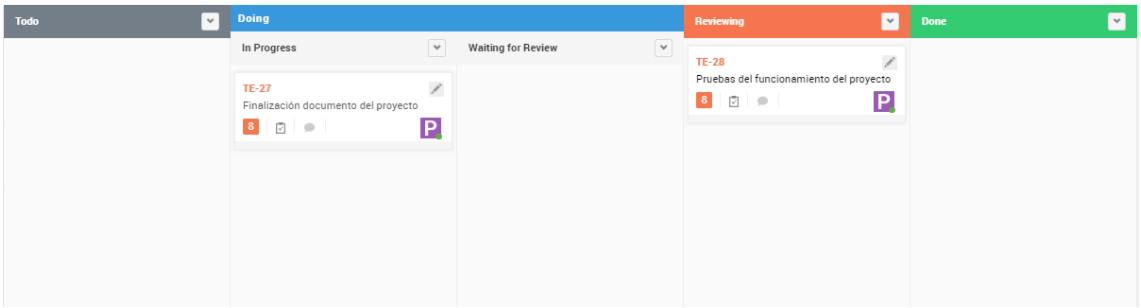


Figura 44. Tablero *Scrum* del sexto *Sprint*

Tabla 24.

Sexto Sprint Backlog

Tipo de Ítem	**Número de Sprint**	**Nombre**	**Trabajo Actual**	**Inicio**	**Fin**
Ítem de Trabajo	6	Realización del capítulo IV y V del documento del proyecto	20 horas	Lunes 17/4/17	Viernes 21/4/17
Ítem de Trabajo	6	Pruebas del funcionamiento del proyecto	20 horas	Lunes 24/4/17	Viernes 28/4/17

Las pruebas de funcionamiento se realizaron acorde esta descrito en la parte de validación de cada historia de usuario, con esto se aseguró que se hizo un control de calidad sobre el desarrollo de software. Además, se cumplió con lo que se propone en la metodología de *Scrum*, que se realiza pruebas antes de entregar el software funcionando.

3.3. Descripción de Procedimientos

A continuación, se explicarán todos los procedimientos que se realizaron durante el desarrollo del proyecto. También se detalla cierta funcionalidad de los componentes externos utilizados en seguridad, o en el despliegue de la infraestructura.

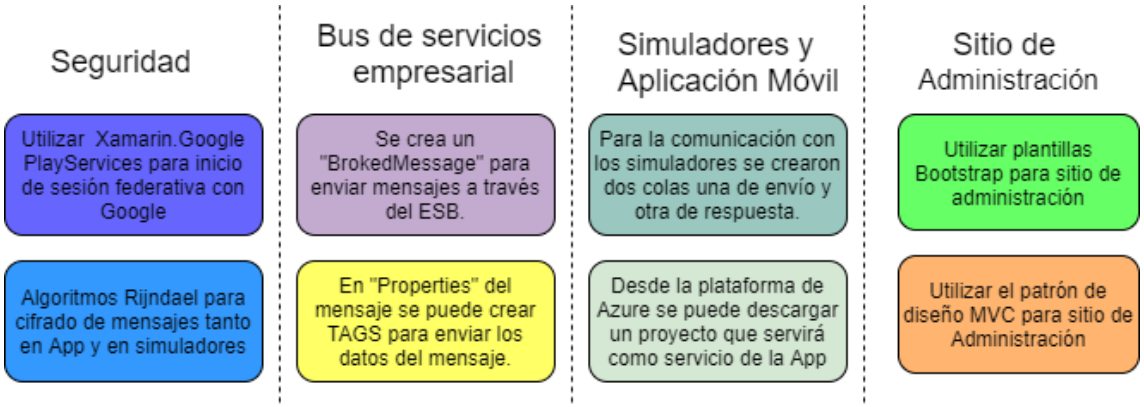


Figura 45. Imagen que muestra procedimientos organizados por área

Para lograr el inicio de sesión con Google, se debe instalar en el proyecto de la aplicación móvil *Xamarin.Google.PlayServices*. El cual permite agregar mediante código un botón para que realice el enlace con los servicios de *Google Play* en los dispositivos móviles. Se debe dar acceso al dispositivo para el acceso a las cuentas de Google que tenga registradas el usuario, y también dar acceso a los servicios de internet, para que cuando el cliente inicie sesión en el dispositivo este se pueda conectar a los servicios para la autenticación del usuario.

En caso de que no tenga registrada una cuenta en el teléfono, automáticamente se direccionará a la pantalla de inicio de sesión con *Google* para que pueda iniciar con una cuenta en el dispositivo.

Para la creación del sitio Web se utilizó una plantilla de sitios de administración basada en la tecnología de *Bootstrap*. Este sitio fue modificado para adaptarse a las necesidades de este prototipo.

Se utilizó el diseño de MVC para su construcción, la primera capa es Modelos, contiene los contenedores de datos encargados del transporte y manipulación de los datos que circulan por el sitio. La segunda capa es la Vista, que contiene todo el código HTML mezclado con C# para mostrar la interfaz de usuario. Y por último la capa de controladores en la que están todas las acciones que realiza el sitio cuando se ejecuta un botón en la interfaz.

Para la implementación de seguridad en la aplicación móvil se utilizó el algoritmo *Rijndael*, el cual requiere una contraseña para el cifrado de los datos, por lo que se determinó que dicha llave de encriptación y des encriptación fuese el id del dispositivo desde el que se está realizando la petición a los simuladores. Con esto se podría evitar el ataque de "*Man in the Middle*" el cual consiste en que el atacante simulara ser el servicio de la aplicación y recibirá los datos, ya que estos se encuentran cifrados y no podría conseguir información valiosa.

En los simuladores de cooperativas y pago de servicios se agregó el archivo que contiene los algoritmos de encriptación y des encriptación del cifrado de datos.

Otro mecanismo de seguridad implementado en el prototipo es que los canales por los que se comunica el dispositivo hacia el servicio en la nube, y la comunicación con el Bus de Servicios Empresarial es de tipo HTTPS, esto quiere decir que es un canal seguro y es casi imposible para un atacante informático vulnerar este tipo de seguridad.

Para conectar el bus de servicios empresarial con el servicio aplicación móvil primero se debe agregar la referencia a *Microsoft.ServiceBus* que proviene del *NugetPackage*. Después se debe copiar la cadena de conexión de los roles en el Azure que puedan escuchar y enviar mensajes a través de las colas.

Se debe crear un *BrokedMessage* para poder enviar un mensaje a las cooperativas y "*Properties*" para agregar propiedades especificas al mensaje que se desea enviar. Se desea enviar el tipo de operación que va a realizar el simulador una vez que procese el mensaje. La identificación del cliente para realizar la consulta de existencia de dicho cliente en la cooperativa, el cual se

debe enviar cifrado al ser un dato crítico. Finalmente, la contraseña con la que se desencriptará los datos enviados en el mensaje.

Para la conexión del bus de servicios empresarial con los simuladores y el servicio de la aplicación móvil, se requiere crear colas que serán las encargadas del transporte de la mensajería de un servicio hacia otro.

Las colas son de tipo asincrónico, es decir solo pueden transportar datos desde un sitio al otro y no en ambos sentidos a la vez, por lo que se decidió crear dos colas por cada simulador, uno que se encargue de enviar la mensajería de respuesta y otro en recibir las peticiones desde el servicio móvil.

Para la creación del servicio de la aplicación móvil en *Azure* fue necesario primero crear la base de datos donde se almacenarían todos los datos propios de la *App*.

Para la creación del servicio existe una opción llamada “Inicio Rápido” en el que se puede descargar un proyecto por defecto en C# para realizar las conexiones con el bus de servicios empresariales y los simuladores.

Conclusiones y Recomendaciones

4.1. Conclusiones

En conclusión, la utilización del bus de servicios empresarial como arquitectura para el desarrollo del prototipo, permitió que se integren los simuladores y la aplicación móvil sin la necesidad de programar un canal de comunicación para envió y recepción de los mensajes que contienen la información de consulta y respuesta. Además, al no existir acoplamiento entre ambos, permitió que el desarrollo sea por partes y al final se pueda enlazar todo con el bus de servicios empresarial.

Esta arquitectura también permite que las cooperativas de ahorro crédito pequeñas se puedan ir uniendo al esquema sin la necesidad de desarrollos adicionales, porque ofrece a las cooperativas la facilidad de exponer servicios *REST* para que se puedan conectar con el aplicativo móvil, garantiza que sea compatible con cualquier *CORE* Bancario, que esté desarrollado en un lenguaje de programación diferente, debido a que *REST* tiene una sintaxis universal que puede ser interpretado por muchos lenguajes.

Me sirvió, utilizar en su mayoría herramientas tecnológicas de un mismo proveedor, debido a que fue más fácil enlazar las diferentes partes del sistema, además conté con documentación que me permitió ahorrar tiempo en investigación sobre el funcionamiento y acoplamiento de sistemas externos, como fue el caso de la seguridad federativa con Google que tomó más tiempo de lo estimado.

Se podría utilizar distintas herramientas para realizar la orquestación de las transacciones, con el fin de mejorar los tiempos de respuesta de las peticiones de los clientes, ya que con otras herramientas se puede aprovechar mejor los recursos que tiene *Azure*. Se puede también quitar la dependencia que existe en el servicio de la aplicación móvil que actualmente es la encargada de realizar la orquestación.

La arquitectura propuesta en este proyecto puede acoplarse a otro tipo de industrias ya que el bus de servicios empresarial permite que se realicen canales de comunicación sin programación adicional, lo que les puede ser útil a industrias como: comercio, especialmente en aquellas tiendas que cuentan con varias sucursales, y desean enlazar todas sus oficinas por un solo canal. La aplicación, podría servir para cobro y facturación de los bienes o servicios entregados a los clientes. Asimismo, se puede utilizar para los sistemas médicos, en donde se pueda enlazar los datos de los pacientes en línea.

La aplicación móvil ofrece funcionalidades como: transacciones en línea, pago de servicios, consulta de movimientos y saldos, que son los servicios que actualmente ofrece la banca móvil de los bancos. La ventaja de mi prototipo con respecto a aplicaciones que se encuentran disponibles en el mercado está aplicación permite manejar varias cuentas en distintas cooperativas asociadas a un solo usuario. Mientras que con las existentes solo se puede manejar las cuentas de un solo usuario perteneciente a un banco.

Según la Gestión del Riesgo Operativo encontrada en las Normas Generales para las Instituciones del sistema financiero emitido por la Superintendencia de Bancos, se cumple con criterios de seguridad propuestos en el documento como: los datos que viajan a través de los canales están cifrados y enmascarados, se cuenta con usuario diferente a la cedula de identidad, los servidores de Azure están constantemente auditados y monitoreados, cada vez que el cliente cierra la aplicación se solicita que ingrese sus credenciales y se creó un sitio en donde se lleva un registro de todas las transacciones que se realizan a través del prototipo y cuenta con sistemas de alerta en caso de que se esté produciendo errores en el sistema.

Cuando se realizó la conexión del bus de servicios empresarial con los simuladores, pude percatarme que las colas que utiliza Azure son asincrónicas por lo que se utilizó una cola para envío de los mensajes y otra para la respuesta, esto genera un cuello de botella ya que los simuladores deben ir procesando cada mensaje y enviar la respuesta al emisor para poder continuar con el siguiente mensaje que se encuentre en la cola. Debido a que los simuladores

pudieron responder las peticiones en milisegundos, no se notó la falencia ya que el cliente recibe rápidamente la respuesta

La metodología Scrum sirvió para organizar el proyecto de forma que se pueda ir desarrollando y enlazando los componentes del sistema hasta concluir con el prototipo. Además, las tareas se dividieron en unidades pequeñas para que se puedan terminar tomando en cuenta el tiempo disponible. Pese a no cumplir con todos los roles que propone Scrum, porque no existía segregación de funciones, debido a que fue un trabajo personal de un solo desarrollador se logró cumplir con los objetivos propuestos al inicio del proyecto.

Debido a la sensibilidad de la información que manejan las cooperativas de ahorro y crédito, y por temas de sigilo bancario de sus clientes los simuladores fueron la mejor alternativa para poder probar el funcionamiento del prototipo sin la necesidad de realizar convenios con cooperativas. y que pueda pasar por procesos más rigurosos.

4.2. Recomendaciones

Se recomienda utilizar otro tipo de canal de comunicación en el Bus de servicios empresarial, como *Relays* ya que las colas como se encuentran implementadas actualmente generan un cuello de botella en las transacciones. También se debe realizar pruebas de carga para validar la capacidad máxima de usuarios concurrentes que soporta la arquitectura.

Se recomienda utilizar BizTalk como herramienta tecnológica para orquestar las transacciones en línea, y poder realizar flujos de movimientos en los paquetes, de esta manera se puede quitar la lógica que existe en el servicio móvil, y dar apertura a mejores flujos y mejor manejo de la concurrencia.

Se recomienda para una siguiente etapa del desarrollo de este proyecto se tome en consideración que la aplicación debe funcionar en teléfonos inteligentes de gama baja porque se desea que sea compatible con la mayoría de dispositivos

que poseen los clientes de las cooperativas. Además, debe ser compatible con más sistemas operativos, no solo con Android.

Se recomienda agregar nuevas funcionalidades que podría incluir transacciones con entidades de servicio como: recarga de saldo para celular, pago de colegiaturas, pago de servicios de transporte público.

De acuerdo con las Normas Generales del sistema Financiero emitida por la superintendencia de bancos se recomienda incluir mecanismos adicionales de autenticación para una siguiente fase del proyecto como: escaneo de rostro del usuario en vez de utilizar el PIN, Preguntas de desafío similar a la que la Banca utiliza, donde se selecciona una imagen en el registro de usuario, y cada vez que se realice una transferencia o un pago se solicite seleccionar dicha imagen. Además, se debe enviar un mensaje informando al cliente que inició sesión con su usuario.

Se recomienda utilizar una metodología distinta a Scrum cuando se realice proyectos personales, debido a que principalmente se enfoca a proyectos de desarrollo de software en grupo, o también se puede omitir ciertos artefactos o ciertos roles como se hizo en este proyecto.

Se recomienda, realizar pruebas de todos los desarrollos, debido a que esto garantiza la calidad del software antes de que sea desplegado en producción. Se puede crear simuladores en caso de que se complique tener acceso a datos de empresas reales.

REFERENCIAS

Álvarez, M. (2014). ¿Qué es MVC? Recuperado el 15 de febrero del 2017 de https://desarrolloweb.com/articulos/que-es-mvc.html

Collier, M. y Shahan R. (2016). *Fundamentals of Azure*. Redmond, Estados Unidos: Microsoft Press.

Gironés, J. T. (2012). *El gran libro de Android*. Barcelona, España: Marcombo.

Guerrero, N. (2016). ¿Conoces en que consiste la Metodología Ágil en Scrum? Recuperado el 28 de mayo del 2017 de http://programaenlinea.net/conoces-en-que-consiste-la-arquitectura-de-software-en-scrum/

Lambert, M. (2015). *Bootstrap Site Blueprints Volume II* (Vol. 2). Birmingham, Reino Unido: Packt Publishing Ltd.

Li, C. (2016). Questions to ask about your Sprint Backlog. Recuperado el 28 de mayo del 2017 de https://static.wixstatic.com/media/fa773b_4439109d184241ce831dc802c45bb317~mv2.jpg/v1/fill/w_364,h_192,al_c,q_80,usm_0.66_1.00_0.01/fa773b_4439109d184241ce831dc802c45bb317~mv2.webp

Microsoft (2017). Sprint burndown. Recuperado el 7 de Julio del 2017 de https://www.visualstudio.com/en-us/docs/work/scrum/_img/alm_sb_introhealthychart.png

Microsoft (2017). Qué son los temas y las suscripciones del Bus de servicio. Recuperado el 28 de mayo del 2017 de: https://docs.microsoft.com/es-es/azure/includes/media/howto-service-bus-topics/sb-topics-01.png.

Microsoft. (2012). *Introducing Microsoft SQL Server 2012*. Washington, Estados Unidos: Microsoft Press.

Microsoft. (2017). Microsoft Office Project. Recuperado el 18 de febrero del 2017 de https://products.office.com/es/project/project-and-portfolio-management-software?tab=tabs-1

MSDN Library. (2016). Herramientas y lenguajes de desarrollo. Visual Studio 2015: MSDN Library. Recuperado el 18 de febrero del 2017 de https://msdn.microsoft.com/es-es/library/dd831853.aspx

MSDN Library. (2016). Visual Studio y Xamarin. Microsoft Developer Network. Recuperado el 18 de febrero del 2017 de https://msdn.microsoft.com/es-es/library/mt488768.aspx

Pitchaiah, M. y Daniel, P. (2012). *Implementation of advanced encryption standard algorithm*. *International Journal of Scientific & Engineering Research* (Volume 3).

Process OnLine. (2016). Bus de Servicio Empresarial. Recuperado el 5 de marzo del 2017 de http://www.processonline.com.co/blog/bus-de-servicios-empresariales-esb-que-es-y-cual-es-su-utilidad/

Schwaber K.y Sutherland J (2013). Scrum Guides. Recuperado el 30 de mayo del 2017 de: http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-es.pdf

Scrum Manager (2013). Historia usuario ejemplo 1. Recuperado el 28 de mayo del 2017 de http://www.scrummanager.net/bok/index.php?title =File:Historia_usuario_ejemplo_1.jpg

Windows Azure Team. (2013). *Introducing Windows Azure for TI Professional*. Redmond, Estados Unidos: Microsoft Press.

Y. Jadeja, K. Modi (2012). Cloud computing - concepts, architecture and challenges. Recuperado el 6 de junio del 2017 de http://ieeexplore.ieee.org/abstract/document/620387

ANEXOS

Anexo 1:

LIBRO I.- NORMAS GENERALES PARA LAS INSTITUCIONES DEL SISTEMA FINANCIERO

CAPÍTULO V.- DE LA GESTIÓN DEL RIESGO OPERATIVO

4.3.5 Medidas de seguridad en canales electrónicos.- Con el objeto de garantizar que las transacciones realizadas a través de canales electrónicos cuenten con los controles, medidas y elementos de seguridad para evitar el cometimiento de eventos fraudulentos y garantizar la seguridad y calidad de la información de los usuarios así como los bienes de los clientes a cargo de las instituciones controladas, éstas deberán cumplir como mínimo con lo siguiente: (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.5.1. Las instituciones del sistema financiero deberán adoptar e implementar los estándares y buenas prácticas internacionales de seguridad vigentes a nivel mundial para el uso y manejo de canales electrónicos y consumos con tarjetas, los cuales deben ser permanentemente monitoreados para asegurar su cumplimiento; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.5.2. Establecer procedimientos y mecanismos para monitorear de manera periódica la efectividad de los niveles de seguridad implementados en hardware, software, redes y comunicaciones, así como en cualquier otro elemento electrónico o tecnológico utilizado en los canales electrónicos, de tal manera que se garantice permanentemente la seguridad y calidad de la información; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.5.3. Canales de comunicación seguros mediante la utilización de técnicas de encriptación acorde con los estándares internacionales vigentes; (incluido con resolución No. JB-2014-3066 de 2 de septiembre del 2014)

4.3.5.4. El envío de información de sus clientes relacionada con al menos números de cuentas y tarjetas debe ser realizado bajo condiciones de seguridad de la información, considerando que cuando dicha información se envíe mediante correo electrónico o utilizando algún otro medio vía Internet, ésta deberá ser enmascarada; (incluido con resolución No. JB-2012-2148 de 26 de

abril del 2012 y reformado con resolución No. JB-2014-3066 de 2 de septiembre del 2014)

4.3.5.5. La información confidencial que se transmita entre el canal electrónico y el sitio principal de procesamiento de la entidad deberá estar en todo momento protegida mediante el uso de técnicas de encriptación acordes con los estándares internacionales vigentes y deberá evaluarse con regularidad la efectividad del mecanismo utilizado; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012 y reformado con resolución No. JB-2014-3066 de 2 de septiembre del 2014)

4.3.5.6. Las instituciones del sistema financiero deberán contar en todos sus canales electrónicos con software antimalware que esté permanentemente actualizado, el cual permita proteger el software instalado, detectar oportunamente cualquier intento o alteración en su código, configuración y/o funcionalidad, y emitir las alarmas correspondientes para el bloqueo del canal electrónico, su inactivación y revisión oportuna por parte de personal técnico autorizado de la institución; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012 JB-2012-2148 de 26 de abril del 2012)

4.3.5.7. Las instituciones del sistema financiero deberán utilizar tecnología de propósito específico para la generación y validación de claves para ejecutar transacciones en los diferentes canales electrónicos y dicha información en todo momento debe estar encriptada; (incluido con resolución No. JB- 2012-2148 de 26 de abril del 2012 y reformado con resolución No. JB- 2014-3066 de 2 de septiembre del 2014)

4.3.5.8. Establecer procedimientos para monitorear, controlar y emitir alarmas en línea que informen oportunamente sobre el estado de los canales electrónicos, con el fin de identificar eventos inusuales, fraudulentos o corregir las fallas; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.5.9. Ofrecer a los clientes los mecanismos necesarios para que personalicen las condiciones bajo las cuales desean realizar sus transacciones que impliquen movimiento de dinero a través de los diferentes canales electrónicos y tarjetas,

dentro de las condiciones o límites máximos que deberá establecer cada entidad. (reformado con resolución No. JB-2014-3066 de 2 de septiembre del 2014) Entre las principales condiciones de personalización por cada tipo de canal electrónico, deberá constar: el registro de las cuentas a las cuales desea realizar transacciones monetarias, números de suministros de servicios básicos, números de telefonía fija y móvil, montos máximos por transacción diaria, semanal y mensual, entre otros. (sustituido con resolución No. JB- 2014-3066 de 2 de septiembre del 2014) Para el caso de consumos con tarjetas, se deberán personalizar los cupos máximos, principalmente para los siguientes servicios: consumos nacionales, consumos en el exterior, compras por internet, entre otros; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.5.10. Requerir a los clientes que el registro y modificación de la información referente a su número de telefonía móvil y correo electrónico, se realicen por canales presenciales, además no se debe mostrar esta información por ningún canal electrónico; (incluido con resolución No. JB-2014-3066 de 2 de septiembre del 2014) 4.3.5.11. Las instituciones del sistema financiero deben registrar las direcciones IP y números de telefonía móvil desde las que se realizan las transacciones. Para permitir transacciones desde direcciones IP y telefonía móvil de otros países se debe tener la autorización expresa del cliente; (incluido con resolución No. JB- 2014-3066 de 2 de septiembre del 2014)

4.3.5.12. Incorporar en los procedimientos de administración de seguridad de la información la renovación de por lo menos una vez (1) al año de las claves de acceso a los canales electrónicos, la clave de banca electrónica debe ser diferente de aquella por la cual se accede a otros canales electrónicos; (incluido con resolución No.

4.3.5.13. Las instituciones deberán establecer procedimientos de control y mecanismos que permitan registrar el perfil de cada cliente sobre sus comportamientos transacciones que impliquen movimiento de dinero en el uso de canales electrónicos y tarjetas y definir procedimientos para monitorear en línea y permitir o rechazar de manera oportuna la ejecución de transacciones que impliquen movimiento de dinero que no correspondan a sus hábitos, lo cual

deberá ser inmediatamente notificado al cliente mediante mensajería móvil, correo electrónico, u otro mecanismo; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012 y reformado con resolución No. JB-2014- 3066 de 2 de septiembre del 2014)

4.3.5.14. Incorporar en los procedimientos de administración de la seguridad de la información, el bloqueo de los canales electrónicos o de las tarjetas cuando se presenten eventos inusuales que adviertan situaciones fraudulentas o después de un número máximo de tres (3) intentos de acceso fallido. Además, se deberán establecer procedimientos que permitan la notificación en línea al cliente a través de mensajería móvil, correo electrónico u otro mecanismo, así como su reactivación de manera segura; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.5.15. Asegurar que exista una adecuada segregación de funciones entre el personal que administra, opera, mantiene y en general accede a los dispositivos y sistemas usados en los diferentes canales electrónicos y tarjetas; (incluido con resolución No. JB-2012- 2148 de 26 de abril del 2012)

4.3.5.16. Las entidades deberán establecer procedimientos y controles para la administración, transporte, instalación y mantenimiento de los elementos y dispositivos que permiten el uso de los canales electrónicos y de tarjetas; (incluido con resolución No. JB- 2012-2148 de 26 de abril del 2012)

4.3.5.17. Las instituciones del sistema financiero deben mantener sincronizados todos los relojes de sus sistemas de información que estén involucrados con el uso de canales electrónicos; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.5.18. Mantener como mínimo durante doce (12) meses el registro histórico de todas las transacciones que se realicen a través de los canales electrónicos, el cual deberá contener como mínimo: fecha, hora, monto, números de cuenta (origen y destino en caso de aplicarse), código de la institución del sistema financiero de origen y de destino, número de transacción, código del dispositivo: para operaciones por cajero automático: código del ATM, para transacciones por

internet: la dirección IP, para transacciones a través de sistemas de audio respuesta - IVR y para transacciones de banca electrónica mediante dispositivos móviles: el número de teléfono con el que se hizo la conexión.

En caso de presentarse reclamos, la información deberá conservarse hasta que se agoten las instancias legales. Si dicha información constituye respaldo contable se aplicará lo previsto en el tercer inciso del artículo 80 de la Ley General de Instituciones del Sistema Financiero; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012 y reformado con resolución No. JB-2014-3066 de 2 de septiembre del 2014)

4.3.5.19. Incorporar en los procedimientos de administración de la seguridad de la información, controles para impedir que funcionarios de la entidad que no estén debidamente autorizados tengan acceso a consultar información confidencial de los clientes en ambiente de producción. En el caso de información contenida en ambientes de desarrollo y pruebas, ésta deberá ser enmascarada o codificada. Todos estos procedimientos deberán estar debidamente documentados en los manuales respectivos. Además, la entidad deberá mantener y monitorear un log de auditoría sobre las consultas realizadas por los funcionarios a la información confidencial de los clientes, la cual debe contener como mínimo: identificación del funcionario, sistema utilizado, identificación del equipo (IP), fecha, hora, e información consultada. Esta información deberá conservarse por lo menos por doce (12) meses; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.5.20. Las instituciones del sistema financiero deberán poner a disposición de sus clientes un acceso directo como parte de su centro de atención telefónica (*call center*) para el reporte de emergencias bancarias, el cual deberá funcionar las veinticuatro (24) horas al día, los siete (7) días de la semana; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.5.21. Mantener por lo menos durante seis (6) meses la grabación de las llamadas telefónicas realizadas por los clientes a los centros de atención telefónica (call center), específicamente cuando se consulten saldos, consumos o cupos disponibles; se realicen reclamos; se reporten emergencias bancarias;

o, cuando se actualice su información. De presentarse reclamos, esa información deberá conservarse hasta que se agoten las instancias legales; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.5.22. Las entidades deberán implementar los controles necesarios para que la información de claves ingresadas por los clientes mediante sistemas de audio respuesta IVR), estén sometidas a técnicas de encriptación acordes con los estándares internacionales vigentes; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.5.23. Las instituciones del sistema financiero deberán enviar a sus clientes mensajes en línea a través de mensajería móvil, correo electrónico u otro mecanismo, notificando el acceso y la ejecución de transacciones realizadas mediante cualquiera de los canales electrónicos disponibles, o por medio de tarjetas; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012 y sustituido con resolución No. JB-2014-3021 de 30 de julio del 2014)

4.3.5.24. Las tarjetas emitidas por las instituciones del sistema financiero que las ofrezcan deben ser tarjetas inteligentes, es decir, deben contar con microprocesador o chip; y, las entidades controladas deberán adoptar los estándares internacionales de seguridad y las mejores prácticas vigentes sobre su uso y manejo; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.5.25. Mantener permanentemente informados y capacitar a los clientes sobre los riesgos derivados del uso de canales electrónicos y de tarjetas; y, sobre las medidas de seguridad que se deben tener en cuenta al momento de efectuar transacciones a través de éstos; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.5.26. Informar y capacitar permanentemente a los clientes sobre los procedimientos para el bloqueo, inactivación, reactivación y cancelación de los canales electrónicos ofrecidos por la entidad; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012 y reformado con resolución No. JB-2014-3066 de 2 de septiembre del 2014)

4.3.5.27. Es función de auditoría interna verificar oportunamente la efectividad de las medidas de seguridad que las instituciones del sistema financiero deben implementar en sus canales electrónicos; así también deberán informar sobre las medidas correctivas establecidas en los casos de reclamos de los usuarios financieros que involucren debilidades o violación de los niveles de seguridad; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.5.28. Implementar técnicas de seguridad de la información en los procesos de desarrollo de las aplicaciones que soportan los canales electrónicos, con base en directrices de codificación segura a fin de que en estos procesos se contemple la prevención de vulnerabilidades; (incluido con resolución No. JB- 2012-2148 de 26 de abril del 2012)

4.3.5.29. En todo momento en donde se solicite el ingreso de una clave, ésta debe aparecer enmascarada; (incluido con resolución No. JB- 2014-3066 de 2 de septiembre del 2014)

4.3.8. Banca electrónica. - Con el objeto de garantizar la seguridad en las transacciones realizadas mediante la banca electrónica, las instituciones del sistema financiero que ofrezcan servicios por medio de este canal electrónico deberán cumplir como mínimo con lo siguiente: (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.8.1 Implementar los algoritmos y protocolos seguros, así como certificados digitales, que ofrezcan las máximas seguridades en vigor dentro de las páginas web de las entidades controladas, a fin de garantizar una comunicación segura, la cual debe incluir el uso de técnicas de encriptación de los datos transmitidos acordes con los estándares internacionales vigentes; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.8.2 Realizar como mínimo una vez (1) al año una prueba de vulnerabilidad y penetración a los equipos, dispositivos y medios de comunicación utilizados en la ejecución de transacciones por banca electrónica; y, en caso de que se realicen cambios en la plataforma que podrían afectar a la seguridad de este canal, se deberá efectuar una prueba adicional. Las pruebas de vulnerabilidad y

penetración deberán ser efectuadas por personal independiente a la entidad, de comprobada competencia y aplicando estándares vigentes y reconocidos a nivel internacional. Las instituciones deberán definir y ejecutar planes de acción sobre las vulnerabilidades detectadas; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.8.3 Los informes de las pruebas de vulnerabilidad deberán estar a disposición de la Superintendencia de Bancos y Seguros, incluyendo un análisis comparativo del informe actual respecto del inmediatamente anterior; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.8.4 Implementar mecanismos de control, autenticación mutua y monitoreo, que reduzcan la posibilidad de que los clientes accedan a páginas web falsas similares a las propias de las instituciones del sistema financiero.

4.3.8.5 Implementar mecanismos de seguridad incluyendo dispositivos tales como IDS, IPS, firewalls, entre otros, que reduzcan la posibilidad de que la información de las transacciones de los clientes sea capturada por terceros no autorizados durante la sesión; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.8.6 Establecer un tiempo máximo de inactividad, después del cual deberá ser cancelada la sesión y exigir un nuevo proceso de autenticación al cliente para realizar otras transacciones; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.8.7 Se deberá informar al cliente al inicio de cada sesión, la fecha y hora del último ingreso al canal de banca electrónica; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.8.8 La institución del sistema financiero deberá implementar mecanismos para detectar la copia de los diferentes componentes de su sitio web, verificar constantemente que no sean modificados sus enlaces (links), suplantados sus certificados digitales, ni modificada indebidamente la resolución de su sistema de nombres de dominio (DNS); (incluido con resolución No. JB-2012-2148 de 26

de abril del 2012 y reformado con resolución No. JB-2014-3066 de 2 de septiembre del 2014)

4.3.8.9 La institución del sistema financiero debe implementar mecanismos de autenticación al inicio de sesión de los clientes, en donde el nombre de usuario debe ser distinto al número de cédula de identidad y éste así como su clave de acceso deben combinar caracteres numéricos y alfanuméricos con una longitud mínima de seis (6) caracteres; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012)

4.3.8.10 Para la ejecución de transacciones de clientes, se deberán implementar mecanismos de autenticación que contemplen por lo menos dos de tres factores: "algo que se sabe, algo que se tiene, o algo que se es", considerando que uno de ellos debe: ser dinámico por cada vez que se efectúa una transacción, ser una clave de una sola vez OTP (one time password), tener controles biométricos, entre otros; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012 y reformado con resolución No. JB-2014-3066 de 2 de septiembre del 2014)
4.3.8.11 Para establecer las condiciones personales bajo las cuales los clientes realizarán sus transacciones por internet, tales como: matriculación de cuentas, definición de montos máximos, registro de números de teléfono celular, entre otros, que han sido definidos por la institución del sistema financiero, se debe validar o verificar la autenticidad del cliente a través de un canal diferente al de internet; (incluido con resolución No. JB-2014-3021 de 30 de julio del 2014 y reformado con resolución No. JB-2014-3066 de 2 de septiembre del 2014)

4.3.9. Banca móvil. - Las instituciones del sistema financiero que presten servicios a través de banca móvil deberán sujetarse en lo que corresponda a las medidas de seguridad dispuestas en los subnumerales 4.3.5 y 4.3.8; (incluido con resolución No. JB-2012-2148 de 26 de abril del 2012 y reformado con resolución No. JB-2014-3066 de 2 de septiembre del 2014)