ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA DISTRIBUIDO BASADO EN SOA (SERVICE-ORIENTED ARCHITECTURE) PARA GESTIÓN SEGURA DE CONTRASEÑAS

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

CRISTIAN PAÚL PÉREZ VALLE
thianpol@gmail.com

DIRECTOR: ING. RAÚL DAVID MEJÍA NAVARRETE, MSc.
david.mejia@epn.edu.ec

QUITO, JUNIO 2016

DECLARACIÓN

Yo, Cristian Paúl Pérez Valle, declaro bajo juramento que el trabajo descrito es de mi autoría; no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en el documento.

A través de la presente declaración cedo mi derecho de propiedad correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Cristian Paúl Pérez Valle

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Cristian Paúl Pérez Valle, bajo mi supervisión.

_____

Ing. David Mejía, MSc.
DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

Agradezco a cada profesor de la politécnica de quienes tuve la valiosa oportunidad de asimilar sus conocimientos compartidos, igualmente al personal administrativo de la Facultad por estar ahí siempre ayudándonos de manera positiva, y de manera muy especial al Ingeniero David Mejía por cada segundo dedicado a guiarme en este objetivo y por estar siempre ahí tendiéndome una mano. Muchas, muchas gracias.

DEDICATORIA

Dedicado con todo mi corazón a mi madre Enma Leonor Valle Moreno.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE SEGMENTOS DE CÓDIGO

ÍNDICE DE TABLAS

RESUMEN

El presente proyecto designado: “Diseño e implementación de un Prototipo de Sistema Distribuido basado en SOA (*Service-Oriented Architecture*) para Gestión Segura de Contraseñas” plantea una solución a la dificultad que enfrentan los administradores de red al tener que recordar numerosas contraseñas de acceso.

Entre los objetivos específicos planteados se pueden citar:

- ✓ Diseñar e implementar un prototipo de aplicación distribuida que permita administrar contraseñas de forma segura protegiéndolas mediante una sola contraseña maestra.
- ✓ Diseñar un sistema distribuido basado en una arquitectura SOA (*Service Oriented Architecture*) implementado mediante componentes.
- ✓ Implementar un servidor de base de datos para almacenar las contraseñas que el sistema administrará.
- ✓ Proporcionar los servicios de administración de contraseñas mediante Servicios Web.
- ✓ Desarrollar dos clientes ligeros: el primero de tipo Web y el segundo para la plataforma Android.
- ✓ Proporcionar comunicaciones seguras en base a los protocolos SSL y TLS.
- ✓ Realizar el *hardening* del sistema en base a las mejores prácticas de seguridad.
- ✓ Ejecutar pruebas de aceptación al prototipo y analizar los resultados.

El desarrollo de este proyecto se lo presenta en cuatro capítulos, como se explica a continuación:

En el **Capítulo I,** se describen brevemente las bases teóricas sobre las que se asienta el prototipo como son la Arquitectura Orientada a Servicios, *Windows Communication Foundation*, los protocolos de aseguramiento de canal SSL/TLS, prácticas *hardening*,

aplicaciones web, aplicaciones Android, así como conceptos de la metodología de desarrollo Kanban.

En el **Capítulo II,** se describe: el diseño propuesto para la solución en función del análisis de los antecedentes y requerimientos, se especifica la arquitectura a implementarse, como también la distribución de entregables y trabajo en curso para los componentes servicio y cliente.

En el **Capítulo III,** se trata el desarrollo e implementación del prototipo, además se presenta las pruebas realizadas sobre la implantación y el análisis obtenido de los resultados.

En el **Capítulo IV,** se plantean conclusiones y recomendaciones en base a la experiencia obtenida en la realización del presente proyecto.

En cada uno de los capítulos mencionados se incluyen figuras y segmentos de código numerados, para lo cual se agregó tablas de indexación. Como anexos se encuentran el detalle de las pruebas realizadas al sistema y el código completo de la aplicación, que complementan el documento.

PRESENTACIÓN

Hoy en día a toda organización le es imprescindible valerse de sistemas de comunicación que le permitan transportar su información a tiempo y con toda la seguridad posible, derivándose en un número de elementos en infraestructura creciente en hardware y software. El administrador de red debe manejar numerosa información, una de estas son las contraseñas de acceso a los dispositivos de red, servidores, terminales, servicios, programas, etc. El presente Proyecto de Titulación ofrece una solución a la dificultad que enfrentan tanto los administradores de red, como cualquier otro usuario, al tener que recordar numerosas contraseñas de acceso.

En general este tipo de información se documenta bajo técnicas incómodas y vulnerables, como son hojas de cálculo, archivos de texto, o incluso documentos físicos protegidos en una caja fuerte. El prototipo que se implementa en el presente Proyecto, ofrece al administrador de red un servidor de almacenamiento y gestión de contraseñas, al cual se accede previo a una autenticación de usuario, se protege así su información y se evita que deba recordar múltiples contraseñas. El sistema se presenta como una caja fuerte digital, que reduce la cantidad de información que el usuario debe recordar, su fin es convertirse en una herramienta útil en la administración de red.

La solución plantea un modelo distribuido, que puede ser accedido cómodamente mediante un cliente web, procurando que la comunicación sea segura debido al tipo de información que maneja. Otro eje importante resuelto es la modularidad del sistema, que se maneja bajo una Arquitectura Orientada a Servicios, en los que los componentes interactúan para completar un todo que solventa el objetivo trazado, uno de los componentes fundamentales del sistema es el Servicio, que se resuelve bajo un modelo de Transferencia de Estado Representacional (REST). Al usar REST como arquitectura técnica del Servidor, se consigue que el sistema sea lo suficientemente independiente de la plataforma del cliente.

Para plasmar lo dicho en el párrafo anterior se decidió usar WCF basado en el *framework* de .NET y desarrollar dos clientes, uno con la misma plataforma, y otro en la plataforma Java. Si se tiene en cuenta el problema a solucionar en la administración de contraseñas, se decide que no solo es importante el conseguir un cliente de otra plataforma totalmente funcional e igualmente facultado a consumir el servicio, sino además que corra sobre dispositivos móviles, y así facilitar al cliente en potencia la portabilidad de la solución. Para esto se desarrolló un aplicativo Android con las herramientas Android Development Tools y Android Studio. Finalmente hay que mencionar que el almacenamiento de los datos se implementó con SQL Server.

CAPÍTULO 1. MARCO TEÓRICO

1.1 INTRODUCCIÓN

En este capítulo se describe brevemente los conceptos relacionados con el sistema distribuido, como son: Arquitectura Orientada a Servicios (SOA), Software como Servicio (SaaS), Servicios Web, *Windows Comunication Foundation*, canales seguros con SSL y TLS, *hardening*, además de tratar detalles respecto al desarrollo de aplicaciones web, desarrollo en la plataforma Android y la metodología de desarrollo Kanban.

1.2 ARQUITECTURA ORIENTADA A SERVICIOS

La Arquitectura Orientada a Servicios (SOA) es un esquema empleado en el desarrollo de sistemas distribuidos, donde interactúan en conjunto algunos servicios completamente autónomos a través del paso de mensajes entre sus interfaces, siendo estos entre dispositivos o entre dos procesos en un mismo dispositivo [1].

SOA se ha diseñado para apoyar la implementación y la composición de servicios que automatizan procesos de negocio a través de componentes de aplicación que actúan como servicios que interoperan entre sí. Usándose de forma independiente y facilitando combinarse entre aplicaciones.

Estos componentes están disponibles como servicios externos que pueden residir en Internet o en una intranet. La arquitectura orientada a servicios es tanto un marco de trabajo para el desarrollo de software como un marco de trabajo de implementación.

Con SOA los servicios se coordinan para resolver actividades, de modo que al procesarse una solicitud el proveedor de servicios puede ser también el consumidor de servicios. Esta orquestación de servicios se da en la Capa de Negocio de la arquitectura.

SOA se construye en base a varias tecnologías estándar como son XML, SOAP, REST y WSDL, así garantiza la integración entre aplicaciones heterogéneas que interactúan cuando lo necesitan sin mantenerse conectadas, mediante una localización y acceso transparente a los servicios.

Esta visión puede parecer compleja, pero muchas de las bases en las que se establece son similares a las que se establecían en tecnologías con algunos años más de historia como son CORBA[1] y DCOM[2]. SOA no puede implementarse en base a la tecnología de un solo proveedor, organización o marca, el así serlo iría en contra de su filosofía.

SOA es conceptualizado a través de las experiencias de quienes lo implementan, de modo que cada quién puede describirlo según las ventajas que obtiene al utilizarlo, pero no existe un acuerdo común para definirlo.

1.2.1 PRINCIPIOS DE SOA

SOA proporciona un conjunto de principios que definen el diseño de los servicios:

I. Límites explícitos: La única forma de acceder a un servicio es a través de sus interfaces.
II. Servicios autónomos: Cada servicio debe ser independiente respecto a su versión, implementación y problemas de instalación.
III. Comunicación de servicios vía contrato: El cliente no conoce la implementación de los servicios, el solo accede a ellos a través de interfaces, las mismas que deben detallarse en un contrato que define las estructuras y tipos de datos a usar.
IV. Compatibilidad de servicio basada en políticas: SOA permite profundizar mediante políticas el nivel de detalle de un servicio, como trabaja y qué necesita

[1] **CORBA:** *Common Object Request Broker Architecture*, estándar de desarrollo de aplicaciones distribuidas para entornos heterogéneos.
[2] **DCOM:** *Distributed Component Object Model,* es un conjunto de conceptos y componentes de software propietario de Microsoft, mediante los cuales se consigue comunicar un programa servidor con otros programas servidores en una red de computadoras.

para ser invocado. De este modo se separa el nivel básico de interfaces a uno más detallado.

1.2.2 CAPAS

Las capas de SOA se presentan en la Figura 1.1 y se detallan a continuación:

- Sistemas Operacionales, son aplicativos construidos a través de cualquier tecnología, son esencialmente el núcleo con el que se quiere interactuar.

- Componentes, son servicios implantados que resuelven determinada funcionalidad.

- Servicios, contiene cada uno de los servicios de negocio expuestos.

- Procesos de Negocio – Orquestación, composición de servicios expuestos para lograr un objetivo de negocio final.

- Consumidores, capa que agrupa todos los potenciales consumidores de los servicios.

- Integración, infraestructura que media, enruta y transporta peticiones y respuestas.

1.2.3 SOFTWARE COMO SERVICIO SAAS

SaaS es básicamente un método de entrega de servicios en línea de forma pública o privada, remota y bajo demanda. El cliente no se preocupa por el soporte, mantenimiento y actualización del servicio.

Los sistemas bajo Arquitectura Orientada a Servicios usualmente usan esté método para llegar a los aplicativos clientes, entendiéndose que el objetivo de SOA es proveer de servicios a otros aplicativos. Dicho esto, se debe indicar que generalmente SaaS

tiene como objetivo servir a usuarios. El complemento resultante de ambos conceptos es servicios, no necesariamente centralizados, pero sí en línea que atienden usuarios a través de aplicaciones cliente.

Los clientes de Software como Servicio experimentan beneficios como la reducción de costos al evitar la carga de adquisición, mantenimiento, aseguramiento de servidores y software que son necesarios en la implementación de software convencional. Otro beneficio desde el punto de vista de infraestructura tecnológica, es el verse favorecidos de la inversión en seguridad y potenciamiento adquirida por el proveedor de SaaS.

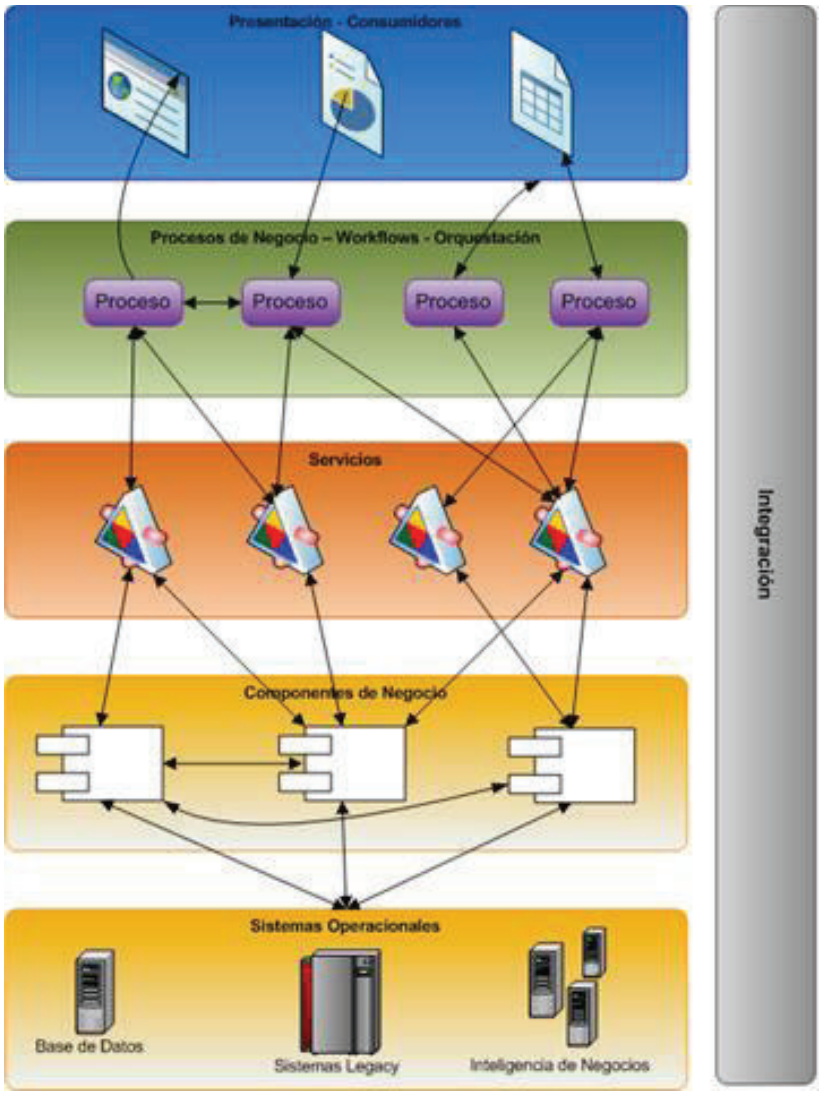


Figura 1.1. Capas de SOA [2]

El proveedor de Software como Servicio es el encargado de la compra, instalación, mantenimiento y actualización del hardware y software necesarios para ofrecer sus servicios. El proveedor administra el acceso a sus aplicativos, incluyendo la seguridad que se ofrece, la disponibilidad que se garantice y rendimiento que se requiera.

1.2.3.1 Características de SaaS

Algunas características que definen SaaS son:

- Acceso Web al software comercial.
- El software se gestiona desde una ubicación central.
- El software se entrega en un uno o muchos modelos.
- Los usuarios no necesitan manejar las actualizaciones de software y parches.
- Interfaces de programación de aplicaciones (API) que permiten la integración entre diferentes componentes de software.

Una barrera de entrada a enfrentar es el mantenimiento de la privacidad en los datos, ya que estos no estarán alojados en los equipos de la organización. Generalmente las empresas tienden a solo utilizar servicios en la nube con su información no crítica.

El acceso unificado, sin duplicación e independiente de la ubicación geográfica a la información, permite a los clientes cuantificar los beneficios que se puedan obtener, abriendo el campo de acción a introducirse con SaaS, esto es totalmente admisible gracias a la escalabilidad que ofrece este tipo de solución.

Otra barrera de entrada se basa en la integración con los sistemas ya existentes en las organizaciones, lo que aumenta la complejidad de ingresar con la solución en las empresas ya establecidas.

La forma lógica de distribución de SaaS es la renta, el usuario no necesita adquirir licencias, solo paga por lo que necesita, exigiendo el cumplimiento de que sea perfectamente adaptable a los cambios que puedan darse con el tiempo. De otro modo, para los nuevos requerimientos el cliente tendría que adquirir una nueva licencia que se adecúe a los cambios que la empresa vaya sufriendo.

1.2.4 SERVICIOS WEB

Es un componente que tiene la capacidad de interoperación en la Web a través del intercambio de datos que resuelven requisitos con servicios. Estos datos son transmitidos en mensajes definidos y son expuestos mediante interfaces entre servicios independientes entre sí de su estado. Para organizar este proceso se han establecido las siguientes definiciones:

1.2.4.1 Universal Description, Discover and Integration UDDI

Es un registro basado en XML donde las organizaciones alrededor del mundo listan por sí mismas los servicios y negocios que ofertan, agilizando las transacciones en línea entre empresas, haciendo sus sistemas interoperables para el comercio electrónico. "*Es a menudo comparado con las páginas blancas, amarillas y verdes de una guía telefónica en donde las empresas listan por su nombre, producto y ubicación los Servicios Web que ofrecen*" [3].

1.2.4.2 Web Service Description Language WSDL

Son archivos con formato XML que definen a los Servicios Web, los protocolos utilizados, el tipo de dato, qué información necesita y que información devuelve. Exponen el servicio para que pueda ser utilizado por el cliente, y al tener un formato establecido, facilitan el acceso al servicio por parte de la instancia cliente.

1.2.4.3 Simple Object Access Protocol SOAP

Es un protocolo que encapsula y transporta la data entre Servicios Web cuya principal característica es ser independiente del protocolo de red, pudiendo transferirse sobre

HTTP, SMTP[1], etc. Gracias a este comportamiento puede ser utilizado en una amplia variedad de ambientes, independientes de sistema operativo, entorno de trabajo o tecnología.

1.2.4.3.1 Sintaxis SOAP

- El mensaje tiene codificación XML
- El mensaje está contenido dentro del espacio de nombres SOAP
- El mensaje debe contener el espacio de nombres SOAP *Encoding*
- El mensaje no usa la definición de tipo de documento DTD
- El mensaje no tiene instrucciones de procesamiento XML

La estructura de un mensaje SOAP se define primero por estar contenida dentro del elemento SOAP, dentro está el *header* con la información de cabecera, *body* donde se contiene la información de las peticiones y respuestas, y el elemento *fault* donde se define el estado del mensaje conjunto con la información de errores si es el caso. Una muestra de un mensaje SOAP se indica en el Segmento de Código 1.1.

```
<SOAPenv:Envelope
     xmlns:SOAPenv="http://schemas.xmlSOAP.org/SOAP/envelope/"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAPenv:Body>
  <req:getNumberOfArticles
     xmlns:req="http://daily-moon.com/CMS/">
     <req:category<classifieds>/req:category>
  </req:getNumberOfArticles>
 </SOAPenv:Body>
</SOAPenv:Envelope>
```

Segmento de Código 1.1. Ejemplo básico de un mensaje SOAP

[1] **SMTP:** Protocolo de transferencia simple de intercambio de mensajes por correo electrónico.

1.2.5 HTTP

Es un protocolo de comunicaciones estandarizado basado en el esquema de petición (*request*) respuesta (*response*). El protocolo no maneja estados, la comunicación se cierra una vez se recibe el *response* resultante en función de la solicitud. Bajo este estándar se transfiere información entre clientes, servidores y *proxies* en la Web.

El mensaje en texto plano es la unidad fundamental de la comunicación HTTP. Su composición es una línea inicial, una cabecera y cuerpo. En la línea inicial se contiene el *request* (como el método de envío de datos y la URL[1] del recurso en solicitud) o *response* del mensaje. La cabecera contiene los atributos del mensaje. El cuerpo contiene información relacionada directamente con el objeto solicitado o resultante del mensaje, de modo que este último no siempre se envía.

1.2.5.1 HTTP Get

Es un método de solicitud al servidor que envía datos en la URL. La longitud de la URL es limitada, por ejemplo, en Internet Explorer no puede superar los 2083 caracteres, desventaja frente a POST donde esta longitud es ampliamente superada.

1.2.5.2 HTTP Post

En este método los datos se envían en el cuerpo del mensaje HTTP. De este modo los datos no pueden ser almacenados en la caché del navegador del cliente.

1.2.5.3 HTTP Put

Es similar al Post, sin embargo, tiene la característica de reemplazar completamente el recurso, para la cual es necesario pasar en el mensaje el objeto a modificar. Al conocer el recurso obtiene siempre el mismo resultado sobre él, estableciendo la propiedad de idempotencia en el método de envío.

---

[1] **URL:** Es una cadena de caracteres con la cual se asigna una dirección única a cada uno de los recursos de información disponibles en internet

1.2.5.4 HTTP Delete

Elimina el objeto definido en la URL.

1.2.5.5 Cookies

Es información que el navegador almacena en memoria dentro de ficheros. Incluyen datos generados por el servidor, o datos introducidos por el usuario, enviados al servidor y reenviados al cliente. HTTP es un protocolo sin estado de sesión entre peticiones sucesivas, por lo que las cookies pueden usarse para asociar el estado.

1.2.6 REST

Es una arquitectura de diseño de sistemas distribuidos que promueve simplificar mecanismos complejos como son por ejemplo CORBA y RPC[1] al usar simplemente HTTP. Básicamente no hay nada que se haga con Servicios Web que no se pueda hacer con la arquitectura REST.

REST es:

- Independiente de la plataforma.
- Independiente del lenguaje de programación.
- Basado en estándares tales como HTTP y puede ser usado fácilmente en presencia de firewalls.

Tal como los Servicios Web, REST no ofrece características de seguridad, cifrado, sesión, administración, calidad de servicio, etc.

Como los Servicios Web, estos pueden ser añadidos en la capa HTTP. Por ejemplo: para la seguridad se emplea frecuentemente la autenticación con credenciales

[1] **RPC:** *Remote Procedure Call*, es una técnica para la comunicación entre procesos en una o más computadoras conectadas a una red.

`username/password`, para el cifrado, REST puede ser implementado sobre HTTPS. Por más información sobre HTTPS referirse a la sección 1.4.4.

REST puede ser tan seguro como SOAP ayudado de HTTPS para proveer cifrado de canal con SSL/TLS, sin esto tanto SOAP como REST son inseguros; no obstante, con un cifrado apropiado, ambos son igualmente seguros.

Las cookies no forman propiamente parte de una buena implementación REST, sus siglas indican transferir el estado para mantenerse *stateless* en el cliente y el servidor, de tal forma que, con esta transferencia, el servidor tenga todo lo necesario para completar y responder a un *request*; no obstante, se permite mantener en caché datos que bien gestionados eliminan parcial o totalmente algunas interacciones entre el cliente y el servidor, mejorando la escalabilidad y rendimiento del servicio.

Las cookies pueden ayudar a mantener un contexto entre el cliente y el servidor, y así el servidor al tener en caché dicho contexto e identificarlo a través de la cookie, puede identificar al cliente que envía el *request*.

El cliente no necesita tener una arquitectura REST para consumir un Servicio REST. Es importante tratar que las consultas REST no retornen cantidades enormes de datos. Dicho esto, se concluye que REST promueve el mejor *performance* con menos tráfico en la red.

1.2.6.1 Componentes de la arquitectura REST

- Recursos, direccionados desde una URL.
- Una Web de recurso, implicando que cada recurso tenga un estado o funcionalidad simple.
- Clientes-Servidores.
- No debe existir estado de conexión (*stateless*).
- Posibilidad de almacenamiento de recursos en caché para lo que se utilizaría las cabeceras HTTP. El servidor especifica los detalles de caché.

- Posibilidad de *proxies* para rendimiento y escalabilidad. Un HTTP *Proxy* puede ser implementado.

Con el cumplimiento de estos componentes REST ofrece escalabilidad, simplicidad, modificabilidad, visibilidad, portabilidad, confiabilidad. Al usar métodos HTTP como son POST, GET, PUT y DELETE su ejecución es menos verbosa que SOAP, otra ventaja sobre este, es que su estructuración es más simple, se limita a una representación básica del objeto con formato JSON[1], XML, o un simple archivo de texto separado por comas. El cliente por su parte para ejecutar una petición solo necesita estructurar una URI[2], en lugar un complejo mensaje SOAP basado en XML.

La clave de la abstracción de información en REST es el recurso. Cualquier información que pueda ser descrita puede ser un recurso: un documento, una imagen, un servicio, o una colección de todos estos.

Un recurso se conceptualiza en un conjunto de entidades para ser mapeado en una representación. REST usa el identificador de recursos para identificar un recurso en particular.

1.3 WINDOWS COMUNICATION FOUNDATION (WCF)

Windows Communication Foundation, se puede describir rápidamente como un *framework* para construir aplicativos SOA basados en .NET. WCF está basado en los principios que establece SOA [1].

El equipo de desarrollo de WCF observó estos cuatro principios en el diseño de WCF, y aunque estos compendios son automáticamente respetados con solo el hecho de desarrollar una aplicación WCF, entender los cuatro puntos cardinales del diseño SOA (descritos en la sección 1.2.1) puede ayudar a entender mejor WCF.

---

[1] **JSON:** Es un formato ligero de notación e intercambio de datos basado en JavaScript.
[2] **URI:** Un identificador de recursos uniforme es una cadena de caracteres que identifica los recursos de una red de forma unívoca.

1.3.1 PRINCIPALES CARACTERÍSTICAS DE WCF

- Soporte para mensajes tanto fuertemente tipados, como no tipados. Este enfoque permite a las aplicaciones basadas en .NET compartir objetos personalizados eficientemente, además le permite ser consumido por software desarrollado en otras plataformas como Java, a través del intercambio de datos con simples flujos XML o JSON.

- Soporte de las últimas especificaciones de Servicios Web.

- Un modelo de seguridad totalmente integrado que abarca ambos protocolos de seguridad tanto los de Windows como los de .NET, y numerosas técnicas de seguridad implementadas en los estándares de Servicios Web.

- Soporte de técnicas de administración de estados de sesión, y soporte para mensajes *oneway* o *stateless.*

1.3.1.1 Composición WCF

La Figura 1.2 ilustra los componentes típicos de un aplicativo WCF. Un sistema distribuido WCF interrelaciona tres *Assembly*:

- WCF *Service Assembly*: Toda la funcionalidad para proveer el servicio

- WCF *Service Host*: Hospeda el WCF *Service Assembly*

- WCF *Client*: Accede al servicio a través de un *proxy*

WCF puede usar una librería de clases estándar o una librería de servicio WCF, esta última referencia automáticamente los WCF *Assemblies*, también crea un archivo `app.config` a pesar de no ser ejecutable, ya que esta herramienta es útil al momento de depurar o correr la librería de servicios WCF que ayuda a crear un cliente de

pruebas automático. En la Figura 1.2 se muestran los componentes típicos de un aplicativo WCF.

El archivo `app.config` presenta un esqueleto de las configuraciones WCF, este puede usarse como referencia para el archivo de configuración final de la aplicación. Además, el usar un archivo de configuración `app.config` de WCF, facilita los cambios y actualizaciones con solo reiniciar el aplicativo.

Las referencias indican que un WCF *Library Server* es más adecuado que un WCF *Service* para hospedar el servicio en varios aspectos; sin embargo, para el presente proyecto se utilizó un WCF *Service* que hospeda el servicio en un servidor IIS (*Internet Information Services*).

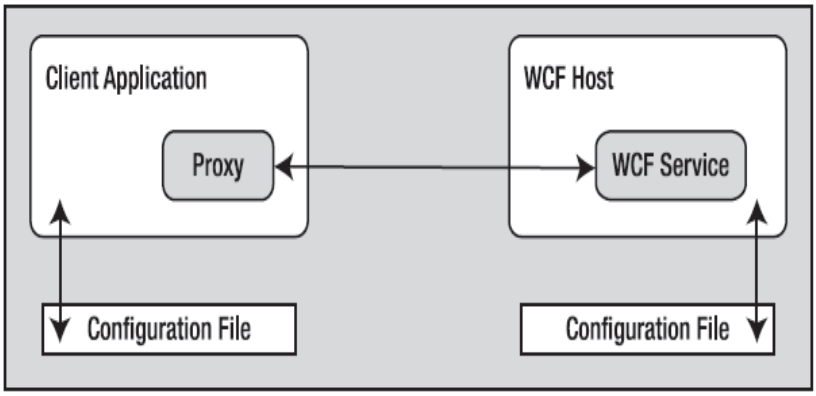


Figura 1.2. Componentes típicos de un aplicativo WCF [1]

Un WCF *Service Host* puede ser cualquier ejecutable .NET, el usar IIS como *host* facilita levantar el servicio ya que IIS configura el *Service Host* en segundo plano automáticamente.

El cliente WCF puede ser cualquier aplicativo .NET; no obstante, si se basa en las asociaciones de HTTP, este cliente puede ser implementado en cualquier otro *framework* como le es Java. Para este proyecto el cliente a desarrollar para la plataforma Android se basa en este protocolo para consumir el servicio.

1.3.1.2 ABC WCF

- *Address*: Describe la localización del código de acuerdo a un tipo de datos `system.uri` el cual típicamente está almacenado en un archivo de configuración.

- *Binding*: Especifica los protocolos de red, mecanismos de codificación, y capa de transporte del enlace o asociación del cliente al servicio.

- *Contract*: Describe cada método expuesto en el servicio WCF.

Usualmente la implementación de un servicio WCF empieza definiendo un contrato de servicio, para después establecer las direcciones y asociaciones.

1.3.2 ADDRESS

Especifica en donde encontrar el servicio de cada *endpoint*[1] representado por un URI. El *Address* se compone de cuatro partes; el esquema de transporte, el dominio que representa el *host* del servicio, el puerto donde se están escuchando solicitudes y la ubicación que identifica la petición. La URI se forma como se observa en la Figura 1.3:

```
ESQUEMA://dominio:puerto/ubicación
```

Figura 1.3. Formato del URI

1.3.3 BINDING

Se usan los *Binding* para especificar los protocolos de transporte y codificación requeridos para la comunicación entre clientes y servidores.

---

[1] ***ENDPOINT:*** Es el punto de entrada a un servicio, proceso, o destino de la Arquitectura Orientada a Servicios. En WCF consiste de una estructura *Address, Binding, Contract* que provee al cliente acceso a la funcionalidad ofertada por el servicio.

Entre los *bindings elements*[1] más utilizados están:

- *Basic* HTTP *Binding*: `WS-Basic`, usa HTTP para transferencia y XML como mecanismo de codificación.

- WS HTTP *Binding*: Añade más características al `basicHttpBinding`, como el soporte de mensajes confiables y MTOM[2].

- WS *Dual HTTP Binding*: Mensajes dúplex, soporta solo seguridad SOAP, requiere de mensajes confiables.

- WS *Federation* HTTP *Binding*: Soporte de HTTP basado en *Federation*[3], es un *Binding* seguro e interoperable. Soporta `WS-Trust`, `WS-Security`, `WS-SecureConversation` que son especificaciones de seguridad para Servicios Web.

1.3.4 CONTRACT

En el *Contract* básicamente se especifican las operaciones qué están siendo expuestas por el *endpoint* hacia el exterior. Para definir una operación se crea un método y se lo marca con el atributo `OperationContractAttribute`. El `OperationContractAttribute` es un atributo que indica que el método y firma del mismo está publicado, cualquier método que no tenga el atributo no definirá ninguna operación en el Servicio WCF. Al agrupar todas estas operaciones conteniéndolas dentro del atributo `ServiceContractAttribute` se obtiene una interfaz. Toda operación de la interfaz debe retornar algo, incluso *`void`*; sin embargo, a diferencia de los métodos locales los cuales pasan referencias de un objeto a otro, las operaciones

---

[1] ***BINDINGS ELEMENTS*:** Elementos del *Binding* que describen algunos aspectos de cómo el *endpoint* se comunica con los clientes.
[2] **MTOM:** *Message Transmission Optimization Mechanism* es un método para envío de datos binarios eficiente desde y hacia Servicios Web.
[3] **HTTP *BASED FEDERATION***: Soporte HTTP especificado por una identidad federada conformada por un grupo de compañías afines a las tecnologías de la información.

pasan modelos de objetos. Esto es significante ya que cada parámetro de entrada o valor de retorno debe ser serializable.

El precepto de definición de atributos serializables facilita su conversión de objetos en un *stream* de bytes y de un *stream* de bytes en objetos.

1.3.5 CONFIGURACION

Los términos a definir durante la configuración de un servicio WCF son *Address, Binding* y *Contract*, estos elementos se establecen en un archivo de configuración o también pueden ir definidos en las líneas de código de la implementación.

Si se usa un archivo de configuración la etiqueta raíz será `<system.serviceModel>`, seguido de la etiqueta `<service>` por cada servicio. En la etiqueta `<protocolMapping>` se establece los WCF *Bindings* a usar por defecto.

El archivo de configuración WCF tiene la estructura del Segmento de código 1.2.

- `<behaviors>`, aquí se configura el comportamiento del *endpoint* en todos los aspectos como son seguridad, concurrencia, *caching*, etc.
- `<client>`, aquí se listan los *endpoints* (ABC definidos) a los que puede acceder el *endpoint* en configuración. Un servicio puede en cualquier momento ser cliente de otros servicios.
- `<commonBehaviors>`, este argumento es necesario cuando el *endpoint* ofrece más de una funcionalidad.
- `<diagnostics>`, en este atributo se configura el diagnóstico de las características WCF.
- `<comContracts>`, usados para interoperabilidad COM-WCF, contiene elementos que permiten especificar varias propiedades de integración COM+.
- `<services>`, contiene una colección de WCF *Services*.

- `<bindings>`, aquí se definen los WCF *Bindings* que especifican los protocolos de transporte y encapsulamiento de los mensajes durante la interacción cliente servidor.

1.3.6 WCF REST

WebHttpBinging es el *Binding* para implementar REST sobre WCF, con este se implementa el esquema *request response* HTTP (PUT, DELETE, GET y POST). A continuación, se mencionan algunas consideraciones para definir un servicio WCF de tipo REST.

Primero, es necesario agregar la referencia `System.ServiceModel.Web.dll` en la interfaz `ServiceContract` para soportar REST, esta librería contiene los atributos `WebGetAttribute` y `WebInvokeAttribute` con los que se configura el método HTTP requerido en la operación, el formato del *response*, el `URITemplate`, etc.

```
<system.serviceModel>
     <behaviors>
     </behaviors>
     <client>
     </client>
     <commonBehaviors>
     </commonBehaviors>
     <diagnostics>
     </diagnostics>
     <comContracts>
     </comContracts>
     <services>
     </services>
     <bindings>
     </bindings>
</system.serviceModel>
```

Segmento de código 1.2. Esqueleto de definición de un `app.config` WCF

- Se aplica `WebGetAttribute` para operaciones de tipo solicitudes GET y se especifica la URI completa del `ServiceOperation`. Se usa llaves y números,

como `{0}` para los parámetros del *request*. El nombre del parámetro y el nombre de la variable en el servicio deben ser ambos iguales y de tipo `string`.

- `WebInvokeAttribute` permite especificar el método a través del cual se va a realizar el *request*. A diferencia de `WebGet`, `WebInvoke` declara un parámetro que identifica el método, que puede ser POST, PUT, DELETE, e incluso el mismo GET.

El método GET se aplica en solicitudes de solo lectura, que no cambian el estado del servicio y sus datos. Para crear, actualizar y borrar datos es recomendable usar POST.

Los servicios REST básicamente se comunican usando XML, pero existen otras formas de *parsing* de datos como JSON[1]. Para usar POST se necesita crear un XML con el esquema `DataMember`, el servicio REST analizará (*parsing)* el XML y construirá el objeto necesario.

Con XML se organizan la estructura de datos. En los casos en los que los parámetros del *request* sean simples es preferible no usar REST. Se recomienda verificar la validez del XML, ya que propiamente este es el encargado de transferir el estado.

De igual forma, los *response* son comúnmente XML. Los XML son de más fácil expansión, todo cliente entiende XML y es independiente de los tipos de dato. Se puede igual usar otros formatos. Por ejemplo, SOAP; sin embargo, SOAP está asociado a XML.

Otros formatos pueden ser CSV[2] que es más compacto, o JSON el cual es de fácil análisis en JavaScript y otros lenguajes. Si el servicio REST específicamente necesitan retornar algo entendible por el humano, como lo es un documento, se puede usar HTML como formato del *response*, solo para ese caso es aceptable esta estructura. Junto a la configuración de `WebHttpBinding` para servicios basados en REST que

[1] **JSON:** Es un formato ligero de notación e intercambio de datos basado en JavaScript.
[2] **CSV:** Formato abierto de representación de datos en tablas, en donde las columnas se separan por comas, y las filas por saltos de línea.

responden a peticiones HTTP en lugar de mensajes SOAP, se añade la propiedad `EndPointBehavior`. En conjunto permiten desplegar el modelo de programación web para servicios WCF.

1.3.7 SEGURIDAD DE WCF

WCF basa su seguridad en dos líneas de enfoque:

- Seguridad de transferencia: Autenticación, confidencialidad, integridad.
- Control de acceso: Permisos de lectura y escritura.

Existen dos modos para establecer seguridad en WCF, el primero es el Modo de Transporte, usando SSL sobre HTTP conocido como HTTPS, y el segundo es el Modo de Mensaje, para este se utilizan cabeceras SOAP. Para mayor información se recomienda revisar la referencia [4].

1.4 COMUNICACIONES SEGURAS CON SSL/TLS

Secure Sockets Layer y *Transport Layer Security* son protocolos de cifrado de transacciones en línea. El protocolo SSL/TLS se ejecuta en una capa entre los protocolos de aplicación como HTTP conocido como HTTPS. Las transacciones web se aseguran con certificados de clave pública que verifican la identidad de los extremos. Otros protocolos de aplicación que pueden operar con SSL/TLS son SSH[1], SMTP[2], NNTP[3], FTP[4], LDAP[5], POP3[6] e IMAP4[7].

En el proceso de autenticación, el cliente SSL/TLS envía un mensaje a un servidor de SSL/TLS, y el servidor responde con la información que el servidor tiene que

---

[1] **SSH:** Protocolo de acceso remoto a dispositivos de red administrables.
[2] **SMTP:** Protocolo de transferencia simple de intercambio de mensajes por correo electrónico.
[3] **NNTP:** Protocolo para la transferencia de noticias en red.
[4] **FTP:** Protocolo para transferencia de archivos entre dispositivos conectados en una red TCP.
[5] **LDAP:** Protocolo ligero de acceso a servicio de directorios en un entorno de red.
[6] **POP3:** Protocolo cliente de descarga de correo electrónico desde servidores remotos.
[7] **IMAP4:** Protocolo cliente de acceso a correo electrónico desde servidores en Internet sin necesidad de que sean descargados.

autenticarse a sí mismo. El cliente y el servidor realizan un intercambio adicional de claves de sesión, y termina el diálogo de autenticación. Cuando se complete la autenticación, la comunicación segura SSL puede comenzar entre el servidor y el cliente utilizando las claves de cifrado simétrico que se establecen durante el proceso de autenticación.

Normalmente solo el servidor se autentica, ya que para que exista una autenticación cliente/servidor se necesita que el cliente maneje una infraestructura de claves públicas que comúnmente no sucede.

1.4.1 SECURE SOCKETS LAYER SSL

Es un protocolo de seguridad que se basa en el aseguramiento del canal a establecerse para comunicar al cliente con el servidor a través del cifrado. Para cifrar el canal se hace uso de algoritmos matemáticos e intercambio de claves, este proceso se lleva a cabo durante una fase de autenticación que precede al intercambio de mensajes.

SSL se preocupa por asegurar los datos impedientemente del protocolo al que estén respondiendo, es decir se estructura como una capa adicional a su carga, entre la capa aplicación y la capa transporte.

1.4.2 TRANSPORT LAYER SECURITY TLS

Transport Layer Security, es el sucesor de SSL. Está compuesto de dos capas TLS *Record Protocol* y TLS *Handshake Protocol*. La primera provee conexiones seguras, ayudado de algún método de cifrado como por ejemplo *Data Encryption Standard* DES[1].

La segunda capa permite la autenticación entre servidor y cliente, y la negociación del algoritmo de cifrado y llaves criptográficas antes del intercambio de datos. A pesar de

[1] **DES:** Algoritmo de cifrado basado en la transformación del texto claro con una clave criptográfica.

estar basado en SSL, TLS y SSL no son interoperables. No obstante, TLS provee un mecanismo que permite trabajar como SSL 3.0, de este modo si el mismo protocolo no está soportado por ambas partes, las partes deberán negociar un protocolo común para comunicarse con éxito.

1.4.3 MEJORAS DE TLS SOBRE SSL

- El algoritmo *Keyed-Hashing for Message Authentication Code* (HMAC[1]) reemplaza al algoritmo SSL *Message Authentication Code* (MAC[2]).
- HMAC produce un *hash* más seguro que el algoritmo MAC. HMAC produce un valor de comprobación de integridad como lo hace MAC, pero con una construcción de función *hash* que la hace mucho más difícil de romper.
- TLS está estandarizado en el RFC 2246[3].
- Se agregan nuevos mensajes de alerta.
- En TLS, no siempre es necesario incluir certificados de todo el camino de regreso a la raíz, se puede utilizar una autoridad intermediaria.
- TLS especifica valores de bloques de relleno que se utilizan con los algoritmos de cifrado de bloque.

1.4.4 HTTPS

Son mensajes estructurados bajo el Protocolo de Transferencia de Hipertexto que se transmiten en un canal seguro (cifrado). Para ello, el sitio donde se publica el servidor web, usa un Certificado SSL.

1.5 HARDENING

Son acciones que se realizan para reforzar al máximo las seguridades de un sistema, cuyo fin es frustrar cualquier ataque y ganar tiempo para minimizar las consecuencias

[1] **HMAC:** Es un valor MAC envuelto en una función Hash calculada en base una clave criptográfica.
[2] **MAC:** Es un valor calculado de longitud fija que se utiliza para autenticar un mensaje.
[3] **RFC 2246:** Publicación de la IETF que describe diversos aspectos del funcionamiento de la versión 1.0 de protocolo TLS.

de un incidente, como también evitarlo en su totalidad. Una política *hardening,* por ejemplo, es el vaciar los datos en forma de ficheros temporales, caché, etc. Otras actividades *hardening* se describen a continuación [5]:

- Proteger el entorno físico del sistema, actualizar el firmware de los componentes, controlar el acceso a los mismos mediante políticas de establecimientos de roles, proteger con contraseñas fuertes el arranque de los equipos y el ingreso al BIOS[1], inhabilitar el inicio del sistema para cualquier unidad que no sea el disco duro principal evitando el acceso a las otras unidades con discos de arranque no autorizados, controlar mediante autorizaciones y contraseñas la lectura de unidades ópticas y dispositivos USB o similares.

- Independizar la partición de instalación del sistema operativo de la partición as usarse para los archivos y carpetas de usuario. No instalar o habilitar componentes de software que ofrezca el sistema operativo, que no sean requeridos.

- Actualizaciones periódicas y controladas de los sistemas para mantener al día los parches de seguridad instalados. Contar con servidor de actualizaciones que proporcione la información del parche a ejecutarse. Los servidores de actualizaciones proporcionan información importante sobre la liberación, prerrequisito, vigencia y criticidad del segmento a ejecutarse. Si es posible mantener un laboratorio de pruebas de las actualizaciones a ejecutarse.

- Mantener actualizado el antivirus y demás herramientas de seguridad que garantizan la protección contra software mal intencionado y spam.

- Control de acceso a los ficheros y carpetas del sistema a través de la gestión de permisos. Bloqueo por defecto.

[1] **BIOS:** Es un estándar de facto que define la interfaz de *firmware*.

- Gestión de las opciones de seguridad de los aplicativos que hagan uso de la red. Entre estos se encuentras lo clientes de correo electrónico, clientes de redes virtuales privadas, navegadores web.

- Configuración adecuada del acceso remoto a los equipos, establecerlo sobre un canal seguro y con acceso solo a los usuarios necesarios. No habilitar el acceso remoto si no es necesario. Limitar el número de conexiones concurrentes. Mantener un historial de las conexiones remotas.

- Establecimiento de servicios de control y gestión de los usuarios de red, tales como Active Directory[1]. Deshabilitar cuentas locales de administración de los equipos. Manejar los permisos adecuados y pertenencia a grupos del dominio según la necesidad de cada usuario.

- Encriptar archivos y unidades de almacenamiento que puedan tener datos sensibles. Establecer políticas de aseguramiento de las claves de descifrado en estos casos. Encriptar los canales de comunicación de mensajería y correo electrónico.

- Tener políticas de restablecimiento de las comunicaciones y servicios en contingencia. Respaldar la data del sistema no solo localmente si no en sitios geográficamente distribuidos, con la finalidad de atenuar los efectos de un posible desastre.

1.6 APLICACIONES WEB

Son aplicaciones constituidas en servidores web, que son accedidas mediante el protocolo HTTP. El usuario accede a su funcionalidad a través del explorador web. En el lado cliente, el explorador presenta una interfaz web que generalmente solo tiene la

[1] **ACTIVE DIRECTORY:** Es una base de datos distribuida que permite almacenar información relativa a los recursos de una red con el fin de facilitar su localización y administración.

lógica para construir y hacer llegar al servidor web la petición del usuario y presentar la respuesta de este. Existen arquitecturas de desarrollo de aplicaciones web que mediante lenguajes en script dan mayor funcionalidad al lado del cliente para acciones sencillas como resolución de fechas y cálculos matemáticos, liberando al servidor web de algunos procesamientos y permitiendo enfocar sus recursos a resoluciones de acceso a datos y otra lógica de negocio del aplicativo.

Su arquitectura da la facilidad a una sencilla aplicación en distintos tipos de ambientes, lo único necesario es un explorador de Internet, o cualquier cliente en la capacidad de crear solicitudes HTTP. Su característica centralizada incluso de la interfaz cliente permite que sea de fácil actualización y transparente al usuario final. Entre las desventajas se encuentra la calidad de la interfaz de usuario respecto a los aplicativos de escritorio, los cuales al tener un enfoque específico resultan mejor adaptados al sistema operativo y hardware para el que son desarrollando; sin embargo, cada día los lenguajes de programación web presentan nuevas tecnologías adaptables a distintos entornos de hardware y software, obteniéndose interfaces más amigables e intuitivas a las necesidades del usuario final.

1.6.1 CARACTERÍSTICAS

- Personalización, actualización y soporte, es suficiente con realizar los cambios en el servidor web.

- Accesibilidad y cobertura, cualquier lugar con acceso a Internet o intranet que contenga el servidor de aplicaciones.

- Capacidad de usuarios concurrentes, alta debido a la arquitectura de clientes livianos que la pueden usar.

- Portabilidad, el sistema puede ser usado con cualquier navegador de Internet.

- Infraestructura y movilidad, solo se tiene que conectar a la red (intranet o Internet) computador local.

- Seguridad física y lógica, es responsabilidad del proveedor de servicio.

1.7 DESARROLLO EN LA PLATAFORMA ANDROID

Una aplicación Android consiste de una o más actividades. Una actividad es básicamente una pantalla, una actividad tiene una o más vistas, una vista está hecha con XML.

Tiene la gran ventaja de que al ser una aplicación nativa ofrece funcionalidades aún sin tener conexión a una red.

Las interfaces de usuario superior, son visibles en el escritorio del dispositivo y tienen mayor precisión en la adaptación de los Servicios Web.

1.7.1 ANDROID DEVELOPER TOOLS ADT

Es un *plugin* para el IDE Eclipse para extender su capacidad en la creación, construcción e implantación de proyectos basados en Android.

A continuación, se describen las características importantes de Eclipse y ADT:

- Creación de proyectos Android, construcción, embalaje, instalación y depuración.
- ADT integra muchas de las tareas de flujo de trabajo de desarrollo en Eclipse, por lo que es fácil desarrollar rápidamente y probar las aplicaciones de Android.
- Integración con las herramientas del SDK[1] .

[1] **SDK**: Es un conjunto de herramientas para desarrollo de software especializadas en un sistema en concreto.

- Muchas de las herramientas del SDK están integradas en los menús de Eclipse o como procesos en segundo plano.
- Programación con Java y editores XML. El editor de lenguaje de programación Java contiene características IDE comunes, como la comprobación de compilación tiempo sintaxis, auto-completado, y la documentación integrada para las API de Android. ADT también ofrece editores de XML personalizados que le permiten editar archivos XML específicos de Android en una interfaz de usuario basada en formularios. Un editor gráfico diseño le permite diseñar interfaces de usuario con una interfaz de arrastrar y soltar.
- Se puede acceder a la documentación solamente ubicando el cursor sobre los métodos, clases o variables.

1.7.2 SDK MANAGER

Permite adicionar complementos al ADT de Eclipse. El SDK de Android separa herramientas, plataformas y otros componentes en paquetes que se pueden descargar mediante el Administrador de SDK. Por ejemplo, cuando las herramientas de SDK se actualizan o una nueva versión de la plataforma Android se libera, puede utilizar el Administrador de SDK para descargar rápidamente a su entorno.

1.7.3 ANDROID MANIFEST

En el archivo `AndroidManifest.xml` que se encuentra en el directorio raíz de la aplicación se definen los requerimientos, configuraciones que necesita el sistema Android donde se ejecutará el aplicativo, así como el comportamiento que este tendrá en el mismo. A continuación, algunas funcionalidades para las que se usa el archivo `AndroidManifest.xml` [6]:

- Denominación del paquete Java.
- Detalle de los elementos del aplicativo tales como actividades, servicios y proveedores de contenido. Con estos datos el sistema Android sabe bajo qué condiciones se lanza un componente del aplicativo.

- Establecimiento de los permisos que requiere la aplicación para interactuar con el sistema Android y sus aplicativos.
- Muestra la versión mínima de Android en la cual corre la aplicación, como también la versión para la cual fue desarrollada.

1.7.4 ACTIVITY VIEWS

Una `ActivityView` es una actividad que en Android está enfocada a resolver algo que necesita el usuario. Interactúan con el usuario constantemente y cada actividad dibuja una interfaz de usuario. Las `ActivityView` se respaldan en el gestor de actividades para controlar su ciclo de vida.

1.7.5 INTENTS

Un `Intent` es básicamente una forma de empezar otra actividad y pasar información. La Figura 1.4 conceptualiza gráficamente la mediación del `Intent`.

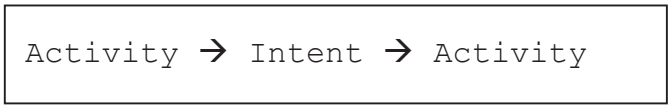


Figura 1.4. Conceptualización de `Intent` como intermediario entre `Activities`

1.8 METODOLOGÍA DE DESARROLLO KANBAN

Como indica Henrik Kniberg [7] "*La metodología es simplemente una herramienta y lo que realmente se necesita saber es cómo se usa, comprendiendo sus fortalezas y limitaciones*". Se usará Kanban como metodología del presente prototipo ya que propone que el desarrollo empiece lo antes posible con las herramientas y procesos que se tengan a la mano y estimula cambios continuos, incrementales y evolutivos del sistema.

A continuación, se detalla las directrices que defiende esta tendencia para desarrollo ágil de software.

1.8.1 WORK IN PROGRESS

El WIP (*work in progress*) es la medida de la carga de trabajo que se puede mantener en un determinado estado del proceso de desarrollo. Para cada etapa se establece un límite de tareas que se pueden atender, este límite es bloqueante si se llega a él. En este caso todo el equipo de desarrollo se enfoca en liberar la etapa. En la Figura 1.5 se observa un tablero en el cuál los elementos de trabajo pasan por diferentes etapas. El número que se identifica por debajo del nombre de la etapa es el WIP establecido para este proceso, se observa que en desarrollo se encuentran dos elementos trabajándose y que este número coincide con su límite WIP, por tanto, el recurso de la etapa anterior no intenta pasar a esta actividad la siguiente entrada. A su vez el equipo de desarrollo, aunque ha finalizado con sus tareas, no puede pasar el elemento al siguiente nivel donde se encuentra una actividad atascada y en su límite de WIP, que para este caso es uno. De modo que todo el equipo se enfoca en resolver la tarea en despliegue para continuar con el flujo de trabajo.

1.8.2 FLUJO PRODUCTIVO

Es el análisis, diseño, desarrollo y pruebas del trabajo, se vale del tablero Kanban para organizar cada paso del flujo.

1.8.3 LEAD TIME

Tiempo entre el proceso de desarrollo de software y la producción de un entregable. Para administrar, monitorizar y reportar el flujo de trabajo la metodología mide del tiempo de entrega, o *Lead Time*.

1.8.4 COMO EMPEZAR CON KANBAN

- Mapear el valor del flujo, visualizar las fases del flujo de trabajo
- Limitar el trabajo en curso, a cada paso del proceso se le asigna un WIP *Limit*
- Medir el tiempo de entrega (*Lead Time*), enfocándose en minimizarlo
- Ejecutar los respectivos cambios basados en las medidas tomadas

- Entender que Kanban es evolucionario y cíclico no continuo

1.8.5 KANBAN EN TEAM FOUNDATION SERVER

El *Team Foundation Server* da soporte sobre metodologías a través de plantillas de proceso que se definen en un proyecto de equipo. Basado en esto, el grupo ALM Rangers implementó una plantilla de proceso (*process template)* que permite fácilmente adaptar Kanban con TFS.

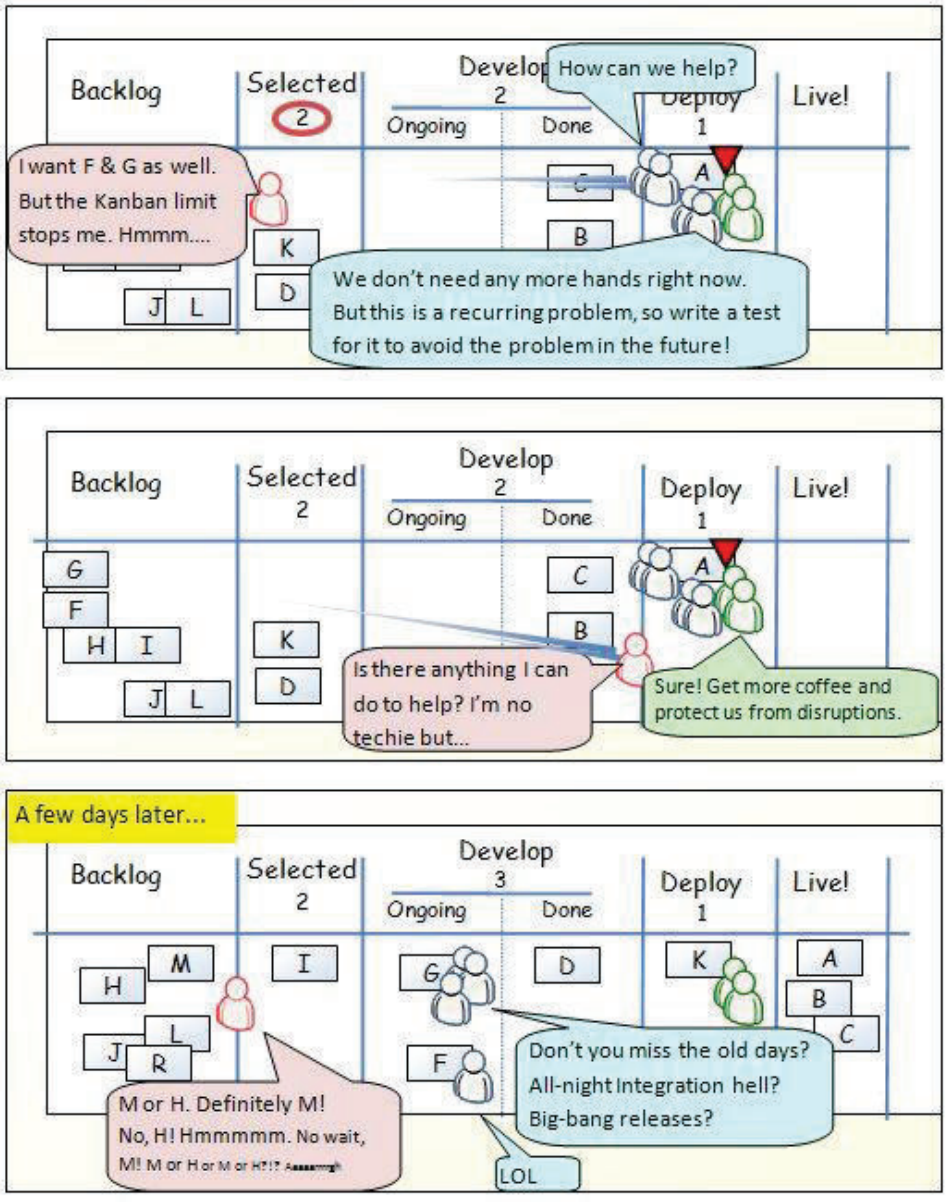


Figura 1.5. Ilustración práctica de Kanban

1.8.5.1 Microsoft KANBAN 1.0 Process Template

Es una plantilla basada en tipos *Work Item*[1], en donde las historias de usuario de las plantillas Scrum[2] son reemplazadas por tarjetas. Otro aspecto importante se refiere al flujo productivo; Cada proceso entra necesariamente en dos estados: En progreso y completo, como se observa en la Figura 1.6, cada proceso puede pasar a cualquier estado sin ningún límite impeditivo. Otros estados son en planificación y eliminado.

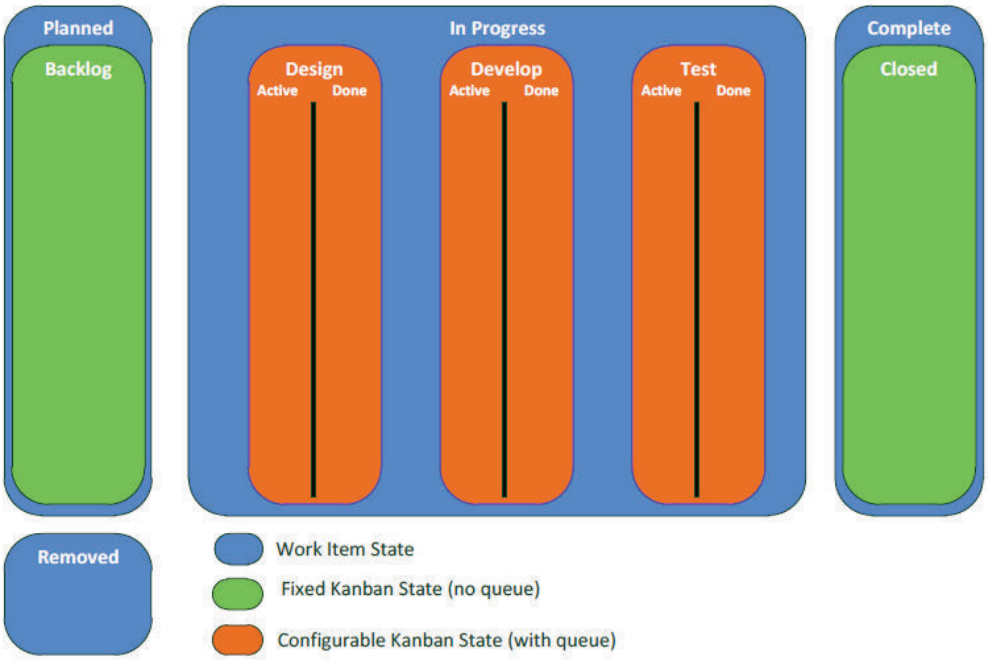


Figura 1.6. Modelo de tarjetas vs estados *Work Item* de Microsoft Kanban 1.0 Process Template [8]

1.8.5.1.1 Tarjetas

En el tablero de la plantilla se pueden ingresar de facto tres tipos de tarjetas:

- Tarjetas de trabajo: Valor de demanda del cliente.

---

[1] ***WORK ITEM***: Es una definición del TFS para asistir a los equipos de desarrollo proporcionándoles un *tracking* de los trabajos en proceso.
[2] ***SCRUM***: Metodología ágil de desarrollo que se diferencia básicamente de Kanban en la utilización de Iteraciones (*Springs*) e historias de usuario.

- Tarjetas de equipo: Valor intangible para el cliente, esencial para el grupo de desarrollo.
- Problemas: Demanda de fallo, necesidad del equipo de trabajo de rectificar un problema.

1.8.5.1.2 Paso de proceso

Información de cada paso en el proceso. En Kanban los procesos de desarrollo de software deben seguir una secuencia lineal de pasos. Cada tarjeta es descompuesta en tareas. Las tareas no se registran en el tablero Kanban, en el solo se registran tarjetas.

En un paso del proceso se define el trabajo en curso de cada proceso, es aquí donde se limitan el WIP, es decir el número de tareas que pueden estar en ese proceso al mismo tiempo. Se usan además pasos compartidos para definir un conjunto de pasos de prueba reusables.

CAPÍTULO 2. DISEÑO DEL SISTEMA

2.1 INTRODUCCIÓN

En este capítulo se describe el diseño del sistema distribuido basado en *Service Oriented Architecture* bajo la metodología Kanban. El sistema está conformado por los siguientes componentes: una base de datos; un servicio *Windows Communication Foundation* bajo el modelo de Transferencia de Estado Representacional; y dos aplicativos clientes, uno Web y otro para plataforma Android.

2.2 ANTECEDENTES

Los sistemas de información confidenciales siempre hacen el uso del método de autenticación bajo contraseñas.Según el portal FayerWayer en promedio los usuarios de internet en Estados Unidos y Reino Unido hacen uso de al menos 22 cuentas en las que deben manejar contraseñas [9]. Una práctica común es utilizar la misma contraseña en las 22 cuentas, y en la gran mayoría de casos jamás se les habrá solicitado cambiarlas.

Aunque el portal también indica que la media latinoamericana es de 5 sitios, este manejo se ve multiplicado para los profesionales encargados de la administración de red y recursos computacionales.

En estos casos, al verse obligados a manejar numerosas contraseñas diferentes, una práctica común es el manejo de archivos digitales protegidos con contraseña que contienen esta información, pero deben llevarlos consigo en un dispositivo de almacenamiento o tenerlos compartidos para acceder desde cualquier punto en su intranet, y en el peor de los casos, mantener una copia impresa.

El presente diseño tiene el objetivo de ofrecer una opción para cubrir la necesidad de contar con un sistema que almacene organizadamente numerosas contraseñas de modo que facilite encontrarlas o gestionarlas según el caso.

2.3 ANÁLISIS DE REQUERIMIENTOS

2.3.1 REQUERIMIENTOS FUNCIONALES

Se ha identificado las funcionalidades que el prototipo proveerá a los usuarios:

2.3.1.1 Requerimientos del Cliente

- Crear una cuenta: el usuario tendrá acceso a una interfaz que le permita registrar una cuenta, para ello ingresará necesariamente sus nombres, apellidos, dirección de correo electrónico, un alias de usuario y una única contraseña maestra utilizada para autenticarse en el sistema.

- Autenticarse en el sistema e iniciar sesión: el aplicativo cliente proveerá los controles necesarios para que un usuario genere una solicitud de autenticación en el servidor e inicie sesión.

- Cerrar Sesión: el aplicativo cliente proveerá los controles necesarios para que el usuario pueda cerrar sesión desde cualquier interfaz en la que esté trabajando.

- Recuperar una contraseña olvidada: el aplicativo cliente proveerá los controles necesarios para que el usuario pueda ingresar los datos requeridos con los que podrá recodar su contraseña de acceso y no su alias de usuario.

- Cerrar su cuenta: el aplicativo cliente proveerá los controles necesarios para que el usuario pueda usar el servicio de desactivación de cuentas.

- Reactivar una cuenta cerrada: el aplicativo cliente proveerá los controles necesarios para que el usuario pueda usar el servicio de reactivación de cuentas con la que recuperará todos los datos que la cuenta manejaba.

- Administración de cuenta: el aplicativo cliente proveerá los controles necesarios para que el usuario pueda administrar su información de usuario registrada con la cuenta.

- Cambio de contraseña maestra: el aplicativo cliente proveerá los controles necesarios para que el usuario pueda cambiar la contraseña maestra de acceso al sistema.

- Agrupar contraseñas: el aplicativo cliente proveerá los controles necesarios para que el usuario pueda.

- Registrar contraseñas: el aplicativo cliente proveerá los controles necesarios para que el usuario pueda registrar contraseñas, siendo estas entidades la razón del sistema.

- Administrar contraseñas registradas: el aplicativo cliente proveerá los controles necesarios para que el usuario pueda administrar la información registrada con cada contraseña ingresada al sistema.

- Recuperar contraseñas eliminadas o el último cambio realizado sobre una contraseña: el aplicativo cliente proveerá los controles necesarios para que el usuario pueda recuperar información relativa a una contraseña que haya sido modificada o eliminada respecto al último cambio que haya sufrido sea cual sea el caso.

2.3.1.2 Requerimientos del Servicio

- Crear una cuenta: se proporcionará el servicio de creación de cuentas.

- Autenticarse en el sistema e iniciar sesión: Un usuario ya registrado en el sistema podrá usar su alias de usuario y contraseña maestra para acceder al

sistema. Por cuestión de seguridad la sesión tendrá un tiempo de caducidad si el cliente deja de utilizar el sistema.

- Cerrar Sesión: El usuario en cualquier momento podrá optar por cerrar la sesión establecida.

- Recuperar una contraseña olvidada: El usuario que hubiera olvidado su contraseña maestra estará en la facultad de recuperarla y para ello utilizará el alias de usuario registrado durante la creación de su cuenta, ya que este representa su identificador único en el sistema. El usuario solo podrá recuperar su contraseña maestra y no su alias de usuario.

- Proporcionar un módulo de mensajería vía correo electrónico: El servicio de recuperación de la contraseña maestra enviará esta información por correo electrónico.

- Cerrar su cuenta: El sistema proveerá la funcionalidad de finalización de adhesión al sistema, es decir cerrar una cuenta bajo criterio del usuario.

- Reactivar una cuenta cerrada: El usuario en cualquier momento puede optar por reabrir una cuenta cerrada recuperando la totalidad de información de la misma, para ello proporcionará sus credenciales de acceso, es decir su alias de usuario y contraseña maestra.

- Administración de cuenta: El usuario podrá consultar la información registrada durante la creación de la cuenta y estará en la facultad de editarla si lo considera necesario.

- Cambio de contraseña maestra: El usuario podrá cambiar su contraseña maestra, para lo que debe ingresar además de la nueva contraseña, la contraseña actual con fines de verificación.

- Agrupar contraseñas: El usuario puede crear grupos de contraseñas para facilitar su administración y búsqueda. Para ello ingresará un nombre y una descripción del grupo. Cuando el usuario crea su cuenta en el sistema, se creará por defecto un grupo genérico automáticamente, ya que las contraseñas necesariamente deberán pertenecer a un grupo.

- Registrar contraseñas: para registrar una contraseña el usuario debe ingresar la contraseña, una palabra clave a funcionar como identificador de la contraseña, una descripción de la contraseña, y necesariamente el grupo al que va a pertenecer. Todos estos campos son obligatorios.

- Administrar contraseñas registradas: El usuario puede buscar información de las contraseñas que tiene registradas por su palabra clave o por grupo de contraseñas, entendiéndose que solo se devolverá en la consulta la palabra clave y la descripción. Cuando la búsqueda se realiza por la palabra clave no necesariamente deberá ingresar toda la palabra para ejecutar la búsqueda ya que la búsqueda retornará todas las coincidencias en todos sus grupos, incluso retornando todas las contraseñas si no se especifica la palabra clave.
  Una vez realizada la búsqueda se puede seleccionar la contraseña para editar la palabra clave y descripción. Junto con esto se habilitará la opción de descifrar la contraseña seleccionada. Una vez descifrada una contraseña en particular el usuario tiene la opción de editarla si lo necesitase, teniendo para ello que utilizar su contraseña maestra con fines de verificación.
  Otra opción administrativa posterior a la búsqueda de información de las contraseñas es la funcionalidad de eliminación de la contraseña encontrada previa a una confirmación de así proceder.

- Recuperar contraseñas eliminadas o el último cambio realizado sobre una contraseña: El sistema almacenará las contraseñas eliminadas o su última modificación con el fin de proveer la opción de recuperarlas mediante la búsqueda de estos registros de respaldo por la descripción de la contraseña.

2.3.2 REQUERIMIENTOS NO FUNCIONALES

- Almacenar la información que el sistema administra en un servidor de base de datos.
- Proporcionar los servicios de administración de contraseñas mediante WCF REST.
- Proporcionar servicios de administración de cuentas de usuario en el sistema mediante WCF REST.
- Generar códigos automáticos para identificar los registros en las tablas de la base de datos.
- Generar automáticamente un grupo genérico por defecto cuando un usuario se registra por primera vez.
- Mantener y controlar el tiempo de sesiones de usuario en el sistema.
- Desarrollar dos clientes ligeros: el primero un aplicativo web y el segundo aplicativo para la plataforma Android.
- Proporcionar comunicaciones seguras en base a los protocolos SSL y TLS.
- Cifrar la información sensible antes de almacenarla en el respectivo campo de su registro en la tabla de la base de datos con un esquema AES.
- Ejecutar prácticas de *hardening* en el servidor de aplicaciones implementado.

2.4 ARQUITECTURA DEL SISTEMA

Se desarrollará un sistema cliente/servidor. En el servidor se hospedará la base de datos y los Servicios Web para gestión de contraseñas. El usuario accederá a los servicios en el aplicativo cliente previa autenticación. Se dispone de dos tipos de clientes uno Web y otro instalable en dispositivos con sistema Android.

El sistema se explotará de la siguiente manera: el usuario accede al sistema con una clave única identificada como contraseña maestra, para ello usa el aplicativo cliente de su elección. El aplicativo cliente resuelve la solicitud del usuario a través de Servicios Web. Una vez iniciada la sesión, la comunicación entre el aplicativo cliente y

los Servicios Web se mantiene bajo dicha sesión, transportado la información en un canal seguro.

Los datos se manejarán desde el servidor de aplicaciones mediante Servicios Web. Por ningún motivo el cliente accederá al sistema de persistencia de datos por algún otro medio que no sea el servicio. La Figura 2.1 presenta un bosquejo de la arquitectura planteada.

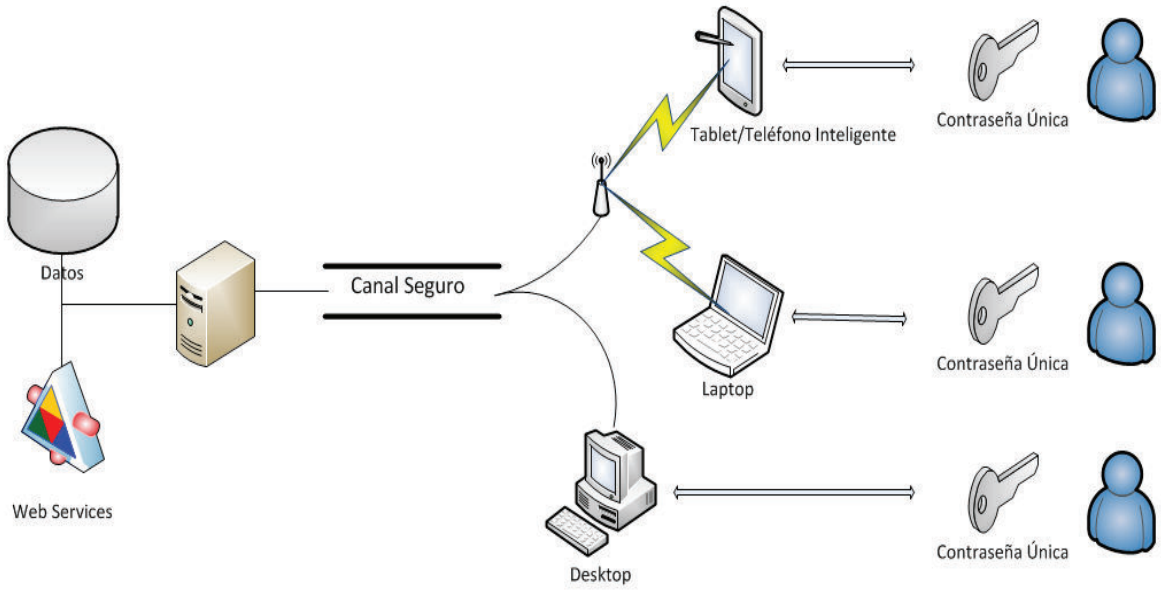


Figura 2.1. Arquitectura del sistema de administración de contraseñas

2.5 HERRAMIENTAS DE DESARROLLO A EMPLEAR

2.5.1 AMBIENTE DE DESARROLLO

El sistema se basa en la arquitectura cliente/servidor. El entorno de desarrollo en el que se construirá el servicio es el IDE Microsoft Visual Studio Ultimate 2012, y se empleará el *framework* .NET 4.5. El servicio es de tipo WCF, al ser la plataforma de construcción de SOA en .NET. Se desarrollarán dos aplicativos clientes: una aplicación

web, basada en ASP.NET, y construida usando el IDE Visual Studio; y un cliente Android, que será desarrollado con las herramientas del ADT *bundle* e IDE Eclipse Juno 4.2.2. Durante el desarrollo del prototipo se migró la aplicación Android del ADT *bundle* ya sin soporte a la herramienta Android Studio 1.3 [10].

2.5.2 HERRAMIENTA DE SEGUIMIENTO

Para realizar el control del proyecto mediante la metodología Kanban se utilizará la plantilla Microsoft Kanban 1.0 para *Team Foundation Server*, la cual puede obtenerse de [11], en donde, además, puede encontrarse información de cómo implementar Kanban ayudado de Visual Studio 2012.

La plantilla permite establecer el equipo de proyecto y asignar los recursos al mismo, para luego plantear flujos de trabajo organizados sobre el tablero Kanban, para ello se ingresan tarjetas por cada entrada del flujo y se asignan tareas con el respectivo límite de trabajo en curso (WIP *limit*), del que se habló en la sección 1.8. Debe haber al menos una tarea por tarjeta, según el peso del proceso. Si bien, el WIP es importante en la metodología Kanban, al existir solo un recurso trabajando en el prototipo, no se asignará demasiada importancia a este indicador.

En cada tarjeta del tablero, las tareas tendrán uno de los siguientes estados `new`, `active` y `closed`, lo cual se observa en la Figura 2.2.

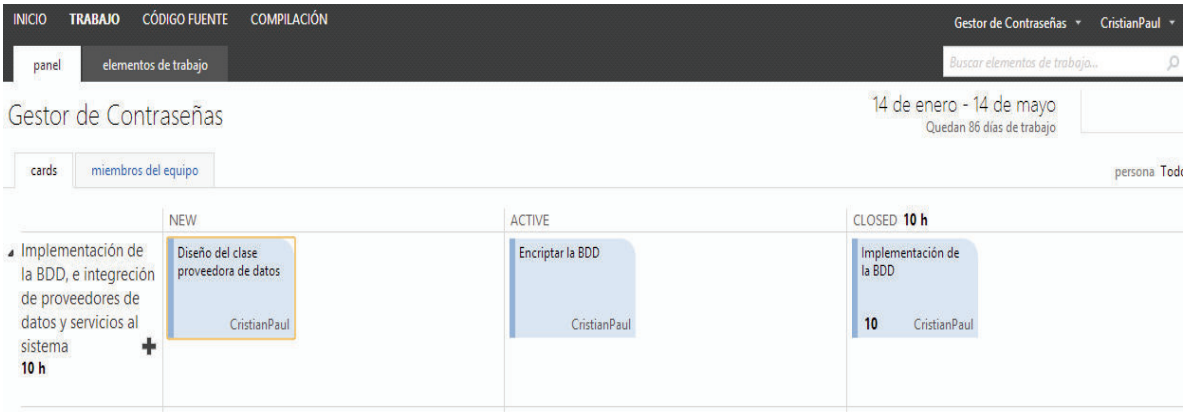
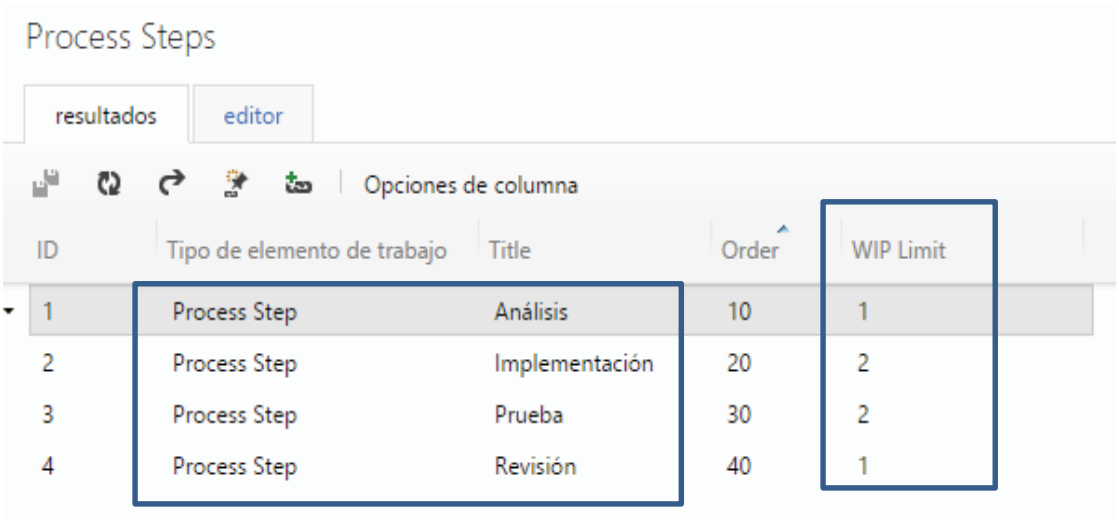


Figura 2.2. Tarjetas del tablero Kanban

2.5.2.1 ESPECIFICACIÓN DEL TABLERO KANBAN

Se decidió implementar el tablero en base a entregables. Cada entregable debe pasar por cuatro pasos: análisis, implementación, pruebas y revisión. En la Figura 2.3 se observa los límites en el trabajo en curso WIP establecidos para cada paso del proceso.



ID	Tipo de elemento de trabajo	Title	Order	WIP Limit
1	Process Step	Análisis	10	1
2	Process Step	Implementación	20	2
3	Process Step	Prueba	30	2
4	Process Step	Revisión	40	1

Figura 2.3. Trabajo en curso (WIP) acorde a la metodología Kanban

En la Figura 2.4 se detallan las tarjetas de tablero Kanban, cada tarjeta identifica un entregable y define el responsable del mismo.

Los entregables definidos son la base de datos a usarse, el modelo *Entity Framework*[1] con el que el servicio interactuará con la base de datos, los *data contracts* para el manejo de los objetos del servicio, la interfaz de contrato que será utilizada para la exposición de los servicios, la configuración del Servicio WCF, los mecanismos de construcción de solicitudes y consultas, todos los servicios que resolverán los requerimientos establecidos. Con respecto a los clientes los entregables tanto para el aplicativo web como para el aplicativo Android serán: La generación de formularios o *layouts* respectivamente que respondan a la funcionalidad establecida, métodos de

[1] ***ENTITY FRAMEWORK*:** Es un *framework* de .NET para conversión de datos de un modelo orientado a objetos (empleado en ciertos lenguajes de programación) a un modelo relacional (empleado en las bases de datos).

generación de las estructuras XML que son argumentos de entrada de los servicios en base a los *data contracts*, generación de mecanismos de comunicación encargados de la construcción de solicitudes HTTP al servicio y desglose de los datos recibidos en la respuesta para pasarlos a los formularios, establecimientos de mecanismos de control de sesión en base a la información proporcionada por el servicio.

Casos

Nuevo | Crear consulta | Opciones de columna

Ord...	Título	Estado De Tarea	Área De Valor	Ruta De Acceso De La Iteración
1	Base de datos	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables levatamiento servicio
2	Modelo Entity Framework	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables levatamiento servicio
3	Data contracts	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables levatamiento servicio
4	Interfaz de contrato	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables levatamiento servicio
5	Configuración del Adress, Binding, Contract del Servicio WCF	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables levatamiento servicio
6	Implementación del mecanismo de construcción de solicitudes de administración de usuario	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables levatamiento servicio
7	Implementación del mecanismo de construcción de solicitudes de administración de grupo	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables levatamiento servicio
8	Implementación del mecanismo de construcción de consultas de grupo	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables levatamiento servicio
9	Implementación del mecanismo de construcción de administración de contraseñas	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables levatamiento servicio
10	Implementación del mecanismo de construcción de consultas de contraseña	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables levatamiento servicio
11	Servicio de autenticación y establecimiento de sesiones por contexto HTTP	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo servicio
12	Servicio de registro de usuarios	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo servicio
13	Servicio de consulta de información del usuario	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo servicio
14	Servicio de modificación de la información de un registro en la tabla Usuarios	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo servicio
15	Servicios para cambiar la contraseña maestra	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo servicio
16	Servicio para registrar usuarios	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo servicio
17	Servicios para consultar la información de los registros de la tabla Grupos	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo servicio
18	Servicio para registrar contraseñas	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo servicio
19	Servicio para eliminar contraseñas	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo servicio
20	Servicio para modificar contraseñas	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo servicio
21	Servicio de consulta y búsqueda de contraseñas	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo servicio
22	Servicio de respaldo de contraseñas	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo servicio
23	Formulario login	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo cliente web
24	Implementación de un mecanismo de generación de XMLs cliente web	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables desarrollo cliente web
25	Implementación de un mecanismo de manejo de sesiones cliente web	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables desarrollo cliente web
26	Implementación de un mecanismo de comunicación con el servicio para el cliente web	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables desarrollo cliente web
27	Formularios y funcionalidad para administración de cuentas	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo cliente web
28	Formularios y funcionalidad para administración de contraseñas	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo cliente web
29	Implantación de estilos y plantilla CSS	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo cliente web
30	Layout de login	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo cliente Android
31	Implementación de un mecanismo de generación de XMLs cliente Android	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables desarrollo cliente Android
32	Implementación de un mecanismo de manejo de sesiones cliente Android	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables desarrollo cliente Android
33	Implementación de un mecanismo de comunicación cliente Android	Nuevo	De arquitectura	GestorContrasenasKanban\Entregables desarrollo cliente Android
34	Layout y funcionalidad de ingreso de contraseñas	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo cliente Android
35	Layout y funcionalidad de consulta de contraseñas	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo cliente Android
36	Layout y funcionalidad de edición de contraseñas	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo cliente Android
37	Layout y funcionalidad de de eliminación de contraseñas	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo cliente Android
38	Layout y funcionalidad de recuperación de contraseñas	Nuevo	Negocios	GestorContrasenasKanban\Entregables desarrollo cliente Android

Figura 2.4. Tarjetas definidas en el tablero Kanban

2.6 BASE DE DATOS

Los datos del sistema se guardarán en cinco tablas que son: `Usuarios`, `Contrasenas`, `Grupos`, `ContrasenaGrupo` y `ContrasenasBackup`. La Figura 2.5 presenta el modelo relacional de la base de datos implementada.

A un usuario le pertenecerán varios grupos, y a través de estos grupos todas las contraseñas que desee almacenar. Los registros de la tabla `Contrasenas` se relacionan con los registros de la tabla `Usuarios` por medio de los registros de la tabla `Grupos`. Es decir, las contraseñas necesariamente se registrarán en un grupo. Nótese la relación de muchos a muchos entre los grupos y contraseñas establecida con la tabla `ContresenaGrupo`.

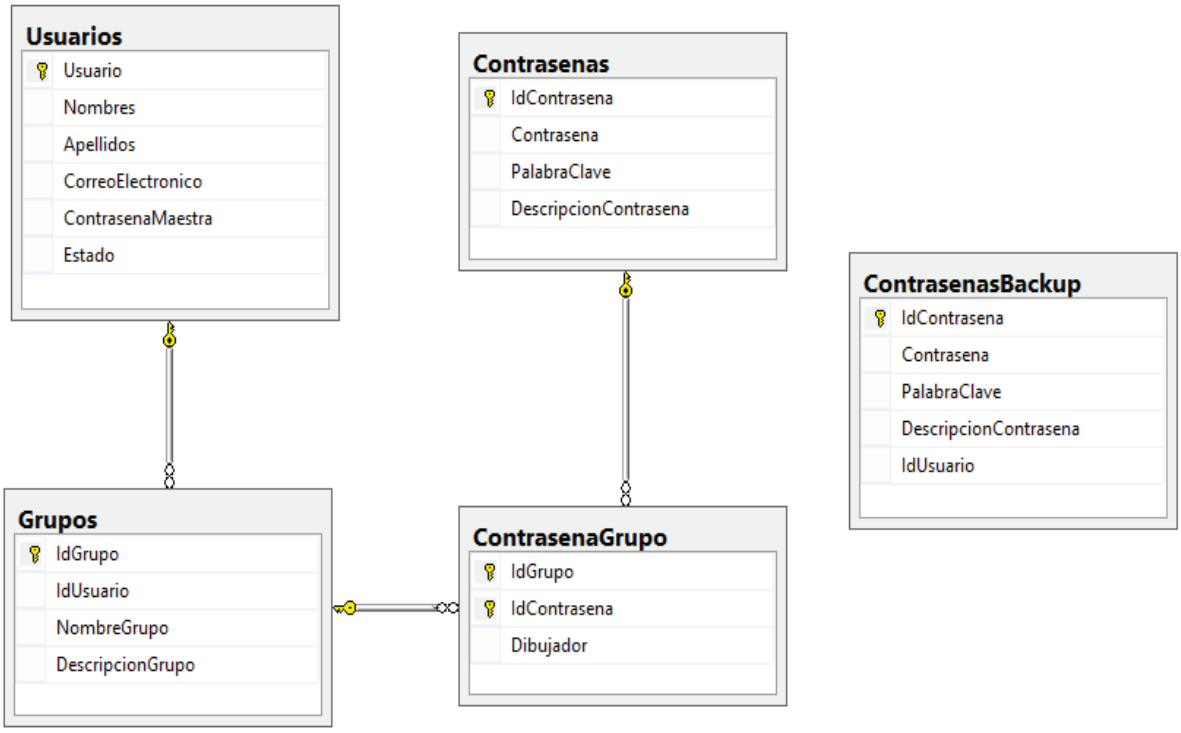


Figura 2.5. Modelo relacional de la base de datos

Se establecerá una copia de la tabla de contraseñas que guarde el estado anterior de un objeto `Contrasena`, con el fin de facilitar su recuperación. Para esto se manejará

la tabla `ContrasenasBackup`, que replica los mismos campos de un registro `Contrasena` junto a un campo que identifica al usuario que provocó el respaldo.

Un registro de la tabla `Usuarios` se compondrá de los siguientes campos:

- `Usuario`: Contiene el nombre de usuario o *login* que junto a la contraseña maestra conforman las credenciales de acceso al sistema, este campo representa la clave primaria del registro, su longitud máxima es 50 caracteres y no acepta nulos.
- `Nombres`: Almacena los nombres del usuario, su longitud máxima es 50 caracteres y no acepta nulos.
- `Apellidos`: Almacena los apellidos del usuario, su longitud máxima es 50 caracteres y no acepta nulos.
- `CorreoElectronico`: Almacena el correo electrónico del usuario, su longitud máxima es 100 caracteres y no acepta nulos.
- `ContrasenaMaestra`: Contiene la contraseña maestra de acceso al sistema, su longitud máxima es 50 caracteres y no acepta nulos.
- `Estado`: Almacena un booleano que indica si el usuario está habilitado o no para usar el sistema y no acepta nulos.

Un registro de la tabla `Grupos` se compondrá de los siguientes campos:

- `IdGrupo`: Contiene el identificador del grupo que representa la clave primaria del registro su longitud máxima es 20 caracteres y no acepta nulos.
- `IdUsuario`: Clave foránea de la tabla `Usuarios`, mantiene la relación un usuario varias contraseñas su longitud máxima es de 50 caracteres y no acepta nulos.
- `NombreGrupo`: Almacena el nombre del grupo su longitud máxima es 50 caracteres y no acepta nulos.
- `DescripcionGrupo`: Almacena una descripción del grupo su longitud máxima es 1000 caracteres y no acepta nulos.

Un registro de la tabla `Contrasenas` se compondrá de los siguientes campos:

- `IdContrasena`: Contiene el identificador de la contraseña que representa la clave primaria del registro su longitud máxima es 20 caracteres y no acepta nulos.
- `Contrasena`: Almacena la contraseña a preservar en el sistema su longitud máxima es 500 caracteres y no acepta nulos.
- `PalabraClave`: Contiene una palabra que tiene como objetivo representar en forma breve la contraseña almacenada; longitud máxima 100 caracteres; no se aceptan nulos.
- `DescripcionContrasena`: Almacena una descripción de la contraseña su longitud máxima es 1000 caracteres y no se acepta nulos.

Un registro de la tabla `ContrasenaGrupo` se compondrá de los siguientes campos:

- `IdGrupo`: Clave foránea de la tabla `Grupos`, junto a `IdContrasena` conforman la clave primaria compuesta del registro su longitud máxima es 20 caracteres y no se acepta nulos.
- `IdContrasena`: Clave foránea de la tabla `Contrasenas`, junto a `IdGrupo` conforman la clave primaria compuesta del registro su longitud máxima es 20 caracteres y no se acepta nulos.
- `Dibujador`: Este campo se creó únicamente para que la tabla se dibuje en el modelo *Entity Framework*, su longitud es un carácter y acepta nulos.

Un registro de la tabla `ContrasenaBackup` se compondrá de los siguientes campos:

- `IdContrasena`: Contiene el identificador de la contraseña que representa la clave primaria del registro su longitud máxima es 20 caracteres y no acepta nulos.
- `Contrasena`: Almacena la contraseña a respaldar en el sistema; longitud máxima es 500 caracteres y no acepta nulos.

- `PalabraClave`: Contiene una palabra que tiene como objetivo representar en forma breve la contraseña respaldada su longitud máxima es 100 caracteres y no acepta nulos.
- `DescripcionContrasena`: Almacena una descripción idealmente detallada de la contraseña respaldada su longitud máxima es 1000 caracteres; no acepta nulos.
- `IdUsuario`: Almacena el identificador del usuario que registró el cambio y motivó el respaldo de la contraseña su longitud máxima es 50 caracteres y no acepta nulos.

2.7 SERVICIO WCF

El servicio es el único componente que tiene comunicación con la base de datos del sistema. Cualquier información a almacenar, leer, actualizar o respaldar, deberá pasar a través del servicio. El gestor de base de datos seleccionado es Microsoft SQL Server 2012.

El servidor de aplicaciones contiene el motor de base de datos SQL e Internet Information Services. Este último consta de un sitio web protegido con un certificado digital local de este modo los aplicativos webs en él publicados, se acceden con el protocolo HTTPS. Al ser un certificado local, los aplicativos clientes manejan una interfaz de confiabilidad del certificado, reconociéndolo y aceptándolo. Este procedimiento se justifica para proveer al prototipo transacciones por un canal seguro. El sistema siempre va a poder ser implementado con un certificado validado, otorgado por una Entidad de Certificación acreditada nacional como lo es el Banco Central del Ecuador, o internacional como lo es Verisign.

2.7.1 INTERFAZ DE CONTRATO

La forma recomendada para crear un contrato en WCF son las interfaces. `IServiciosGestorContrasenas` es la única interfaz diseñada del proyecto que

especifica la colección y estructura de los mensajes necesarios para acceder a las operaciones de la oferta de servicios.

Cada operación definida en la interfaz direcciona un método específico encargado de llevar a cabo la funcionalidad requerida (ver Figura 2.6). De este modo se hace más fácil para el cliente procesar el catálogo de servicios.

De la Figura 2.6 se identifican las siguientes operaciones:

- `CerrarSesion`
- `ConstruirConsultaContrasenas`
- `ConstruirConsultaGrupos`
- `ConstruirSolicitudContrasena`
- `ConstruirSolicitudGrupo`
- `ConstruirSolicitudUsuario`
- `ObtenerInformacionUsuario`

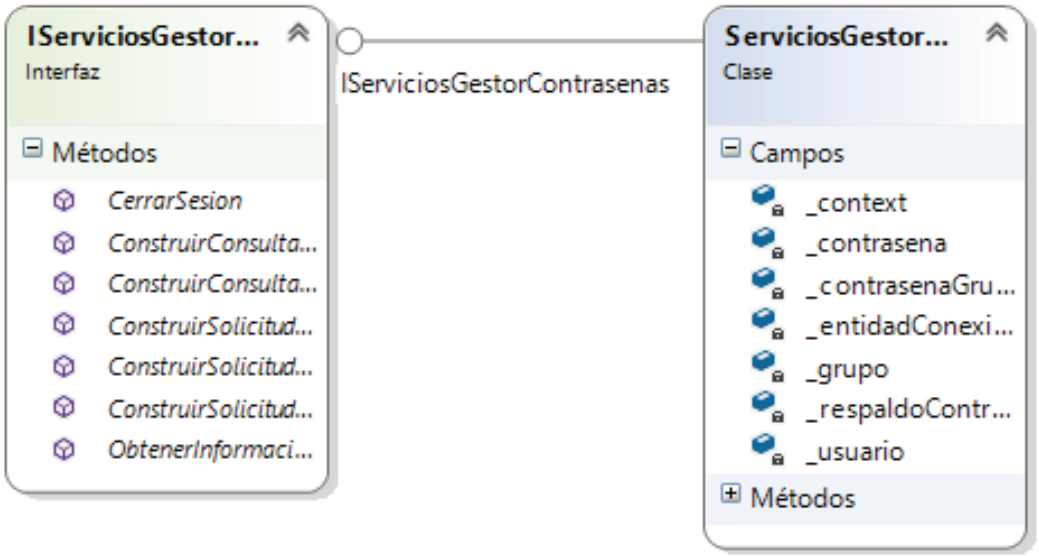


Figura 2.6. Clases de la interfaz de contrato WCF del Gestor de Contraseñas

Cada operación interactuará prácticamente con la misma secuencia de mensajes hasta retornar un dato.

Cada operación identifica un objeto de entrada y salida propio en conjunto y se reutiliza para acceder a distintos métodos del servicio. La interacción entre operación y método de solución se ejemplifica en el diagrama de secuencia de la Figura 2.7.

Los servicios ofertados incluyen el almacenamiento, edición, actualización y eliminación de objetos basados en las entidades de la base de datos. Además se añaden clases que tienen el fin de proveer funcionalidades de cifrado de campos (`EncriptorDescriptor`) y mensajería a través de correo electrónico (`Correo`), estas clases están diagramadas en la Figura 2.8.

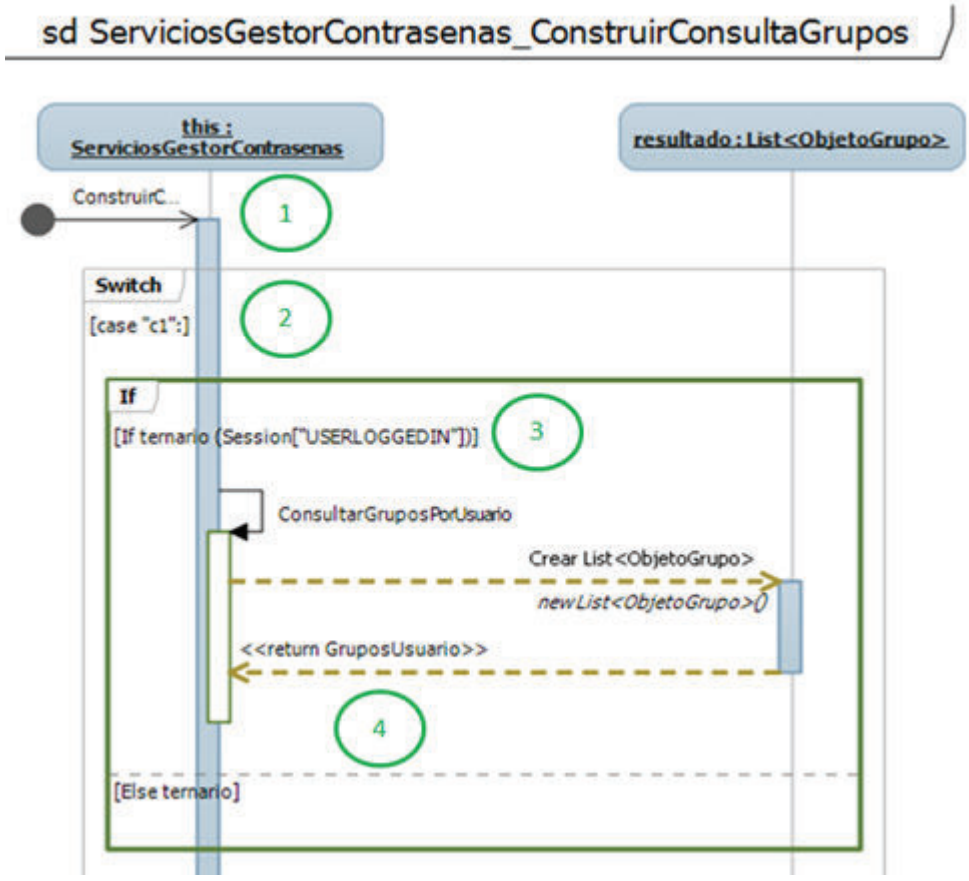


Figura 2.7. Diagrama de secuencia resumido de la interacción entre la operación y el método del servicio WCF

1. Operación definida en el contrato o servicio `[OperationContract]`. El diagrama describe la operación `ConstruirConsultaGrupo`.

2. Evaluación de la propiedad agregada a la operación que identifica la solución deseada, de esta forma se maneja que, si en la operación está involucrado el mismo objeto de entrada y salida, se la reutilice con distintos métodos, resumiendo significativamente la cantidad de contratos de operación y *address* a manejar desde los clientes. En el caso del diagrama `c1` resulta en el método `ConsultarGruposPorUsuario`.
3. Antes de dar paso a la ejecución del proceso identificado, se evalúa la sesión manejada a través del contexto HTTP. Si no existiere se devuelve un valor asignado por defecto que indica que no existe una sesión y no se ejecuta el proceso.
4. Si la sesión existe se ejecuta el proceso identificado y se devuelve el valor obtenido acorde a los valores de objeto definido en la operación [`DataContract`].

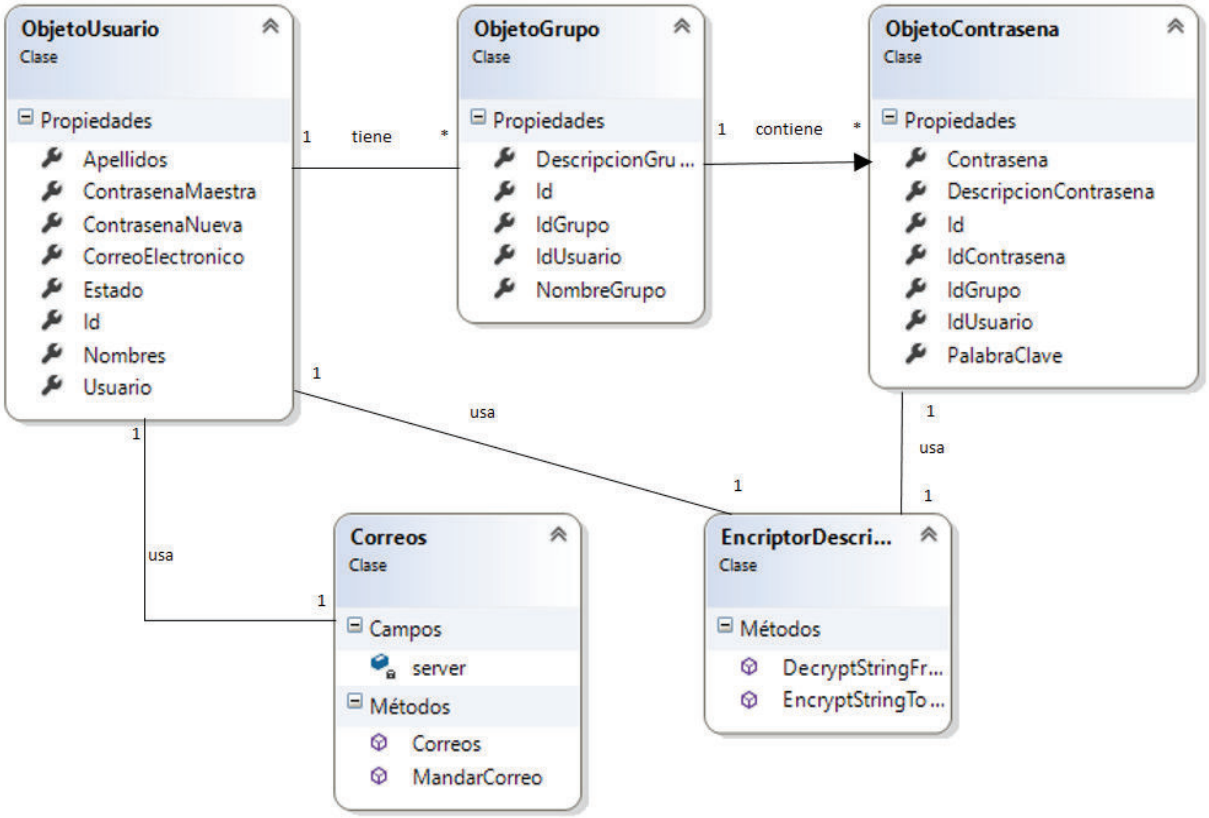


Figura 2.8. Diagrama de clases del servicio WCF

Los métodos que resuelven las operaciones definidas en el contrato tienen como tipos compuestos de entrada las clases Objeto de la Figura 2.8 que usan para este fin los atributos `[DataContract]` y `[DataMember]`. En resumen, solo 3 objetos son necesariamente serializados en formato XML, estos son: `ObjetoGrupo` para la operación `ConstruccionSolicitudGrupo`; `ObjetoContrasena` para la operación `ConstruccionSolicitudContrasena`; y `ObjetoUsuario` para la operación `ConstruccionSolicitudUsuario`.

2.7.2 COMUNICACIÓN CON LA BASE DE DATOS

Se utilizará la herramienta *Entity Framework* EF para consumir la base de datos desde el servicio. Este *framework* de acceso a datos basado en modelos reduce la brecha entre los lenguajes SQL y los lenguajes orientados a objetos al representar tablas en objetos.

Entity Framework, construye un modelo de la base de datos en base a `entitity classes` como se puede observar en la Figura 2.9. Una `entitity class` es una clase con los mismos campos de la tabla en la base, pero se la podrá cambiar sin ningún inconveniente según el negocio de la aplicación. En este proyecto se decidió mantener las `entity class` intactas de tal forma que solo contienen los campos de un registro en la tabla, de este modo se facilitan los cambios en el modelo de base de datos sin afectar a la aplicación servidor. Para ayudar a serializar la información a transmitir se creará un objeto idéntico a la `entity class` que además contendrá información adicional para el flujo del negocio. Estos objetos se nombrarán igual a la `entity class` precedidos de la palabra “Objeto”, ejemplo `ObjetoEntidad`.

En la se observa en un diagrama de secuencia el proceso de comunicación entre el servicio de administración de contraseñas y la base de datos representados bajo un esquema de objetos o `entity model` que se constituye de *entity classes*. Se puede observar en particular que la `entity class` es la única entidad que el servicio ve, ya que el servicio no conoce la base de datos, solo conoce un modelo de *Entity*

Framework donde existen objetos que se utilizan como fuentes para obtención y actualización de información relacionada a las tablas de la base de datos.

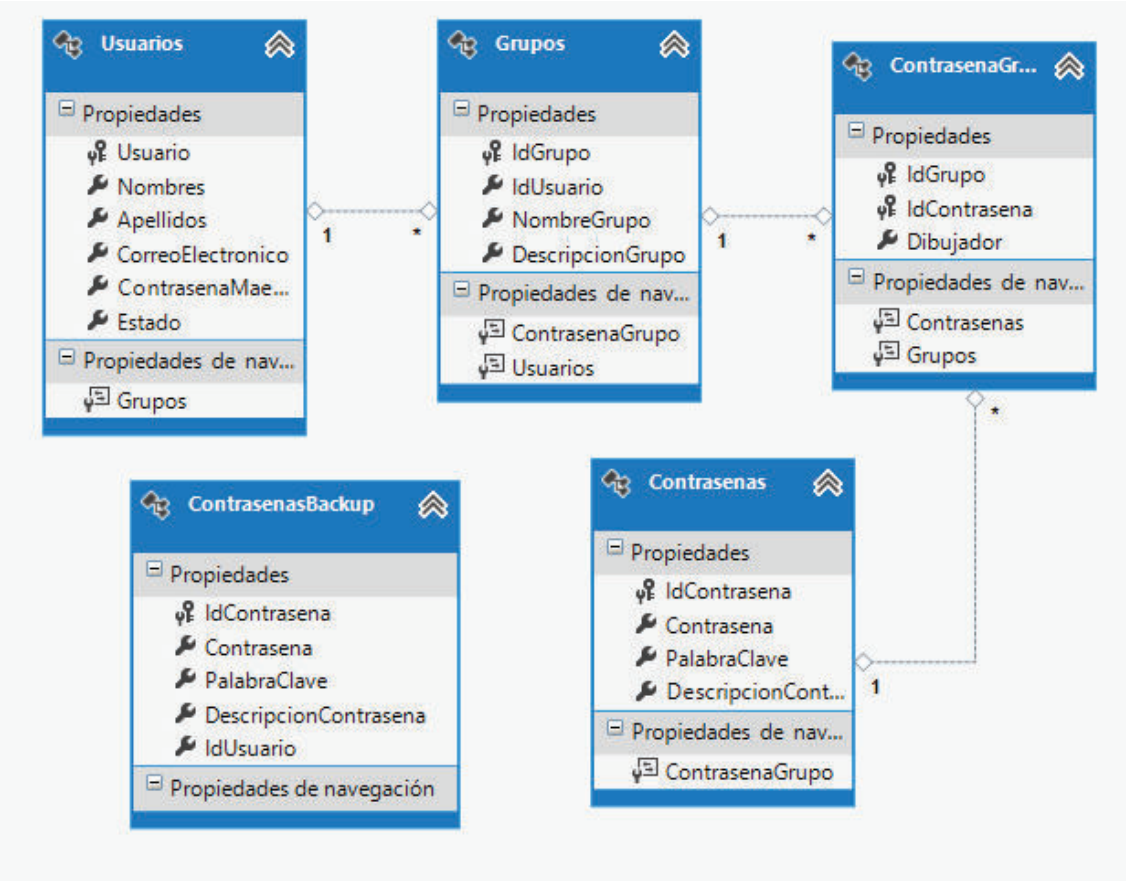


Figura 2.9. Modelamiento EF de la base de datos del Gestor de Contraseñas

Las transacciones con las *entity classes* se manejan con comandos LINQ[1] y operadores lambda[2].

Durante el proceso se identifican las siguientes etapas ():

1. El Servicio `ServiciosGestorContrasenas` recibe una solicitud de edición de contraseña.

[1] **LINQ:** *Language-Integrated Query* es un conjunto de características que agrega capacidades de consulta y actualización de almacén de datos eficaz a la sintaxis de los lenguajes C# y Visual Basic.
[2] **EXPRESIONES LAMBDA:** Son funciones anónimas que se pueden usar para crear tipos delegados o de árbol de expresión. Las expresiones lambda son especialmente útiles para escribir expresiones de consulta LINQ.

2. Se crea una instancia de la clase `ObjetoUsuario` con los argumentos de entrada recibidos en la solicitud.

3. Una vez construido el `ObjetoUsuario` se lanza el método `Autenticar()`.

4. Mediante una instancia de la *entity class* `Usuarios` se hace la consulta en la entidad de la tabla `Usuarios` para traer los datos del usuario que tiene el mismo identificador del `ObjetoUsuario`.

5. Se ejecuta el proceso de cifrado de contraseña, con el fin de concatenar su valor con información obtenida de la tabla `Usuarios` para lo que se utiliza un método de la clase `EncriptorDescriptor`.

6. Si la autenticación es exitosa se instancia una *entity class* `Contrasenas` con la que se realizará el proceso de identificación y actualización de datos.

7. Se instancia la *entity class* `ContrasenasBackup`, entidad que se encargara de la identificación y actualización de la información de la tabla de respaldos.

8. Si se ha identificado que existe un registro de respaldo asociado a la contraseña por editar, se actualiza el campo a editar cifrando la contraseña y registrándose en la tabla.

9. Cuando no existe un respaldo asociado a la contraseña por editar, se da paso a la creación del objeto.

10. Mediante una instancia de la *entity class* `ContrasenasBackup`, se realiza el proceso de ingreso del nuevo registro.

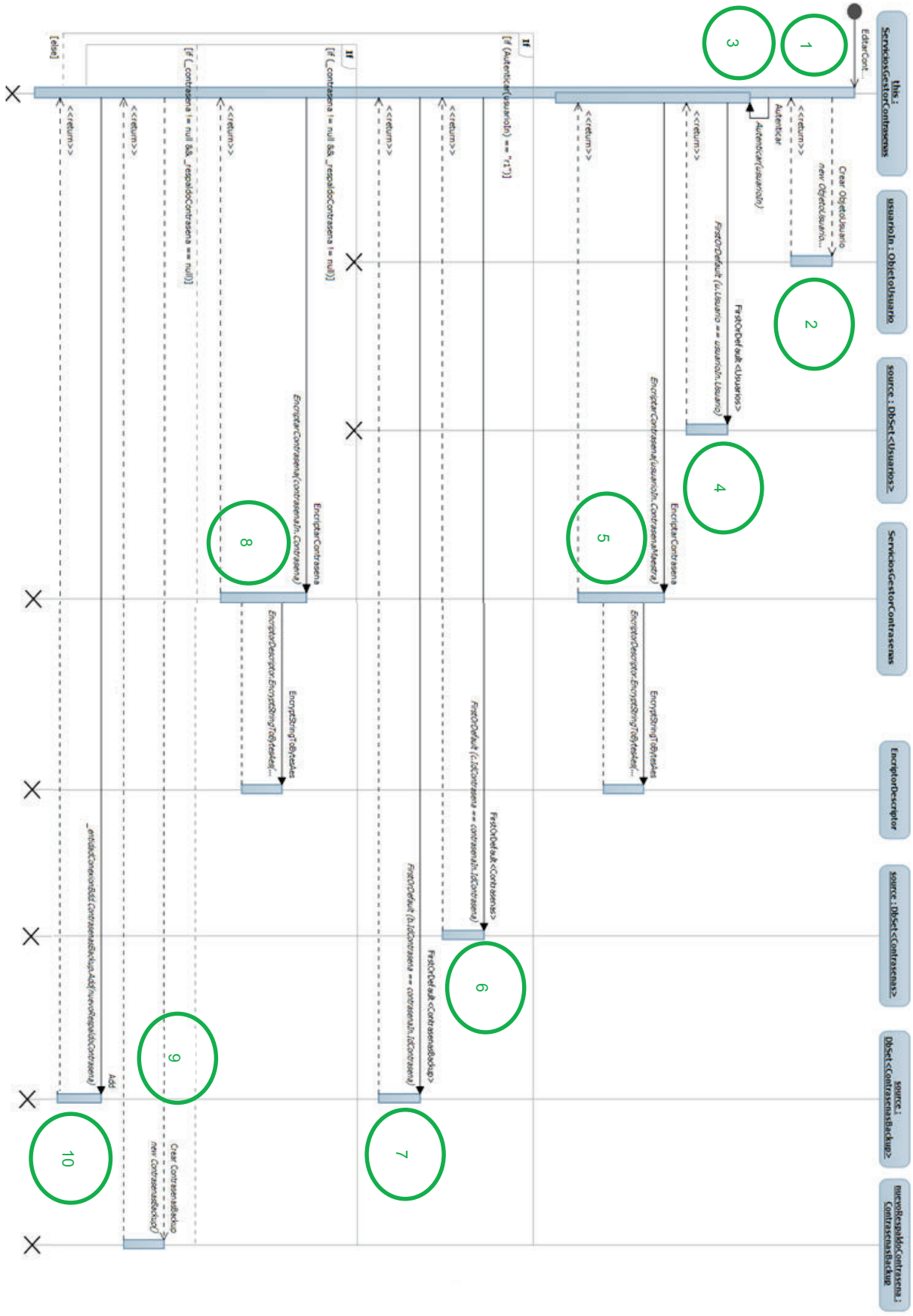


Figura 2.10. Diagrama de secuencia del acceso a datos del servicio WCF con *entity classes*

2.7.3 ARQUITECTURA DEL SERVICIO

Los aplicativos clientes consumirán los servicios WCF a través de un módulo de comunicaciones o *proxy* desarrollado en la misma plataforma que el cliente. Un *proxy* para el sistema es un objeto que contiene implementaciones de funciones que construyen solicitudes HTTP. Un *proxy* serializa la información a enviar y deserializa el *response* recibido del servicio. La seguridad en el canal de comunicaciones se mantendrá con *Hypertext Transfer Protocol Secure* HTTPS.

El servicio define una interfaz de acceso a cada contrato, su comunicación con la base de datos se establecerá con asociaciones con *Entity Framework.* Entre el servicio y la base de datos intervienen los siguientes procesos que interrelacionan los componentes: estructuración de tipos de datos (`Data Members`) que especifican objetos compuestos (`Data Contracts`) para WCF, y entidades (`Entity Classes`) que componen el modelo de datos (`Entity Model`) que representa la base en *Entity Framework.* Otros procesos relacionados con la administración de datos en el servicio son el de cifrado de información a almacenar en la base de datos y el correo electrónico para la notificación de contraseña olvidada (ver Figura 2.11).

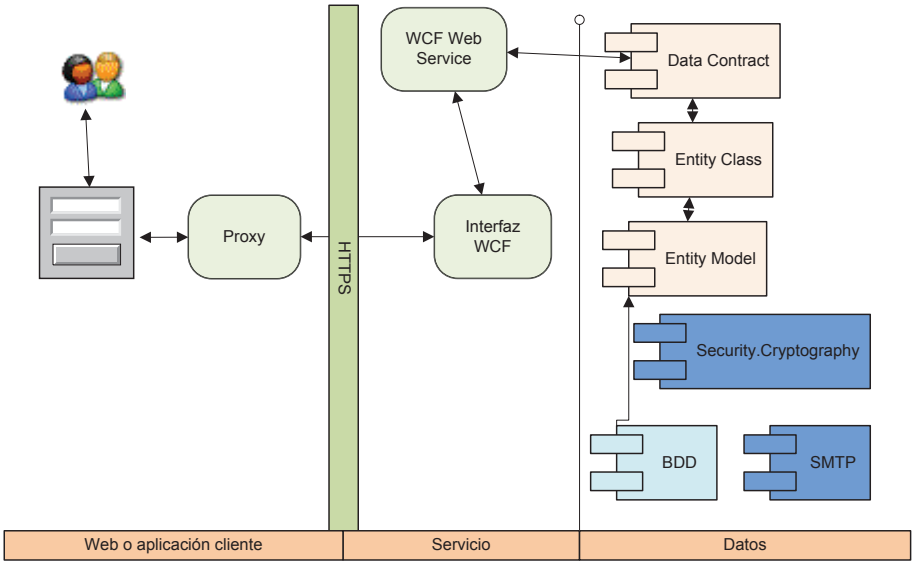


Figura 2.11. Arquitectura Orientada a Servicios en base a WCF a implementar en el prototipo

2.8 CLIENTE WEB

Se definen dos capas, una para la interfaz de usuario y otra para la generación de solicitudes al servicio. En esta última se programa toda la comunicación con el catálogo del servicio, resolviendo y entregando al aplicativo web la información necesaria para presentar soluciones al usuario. La Figura 2.12 representa en un diagrama de secuencia la interacción entre estas capas.

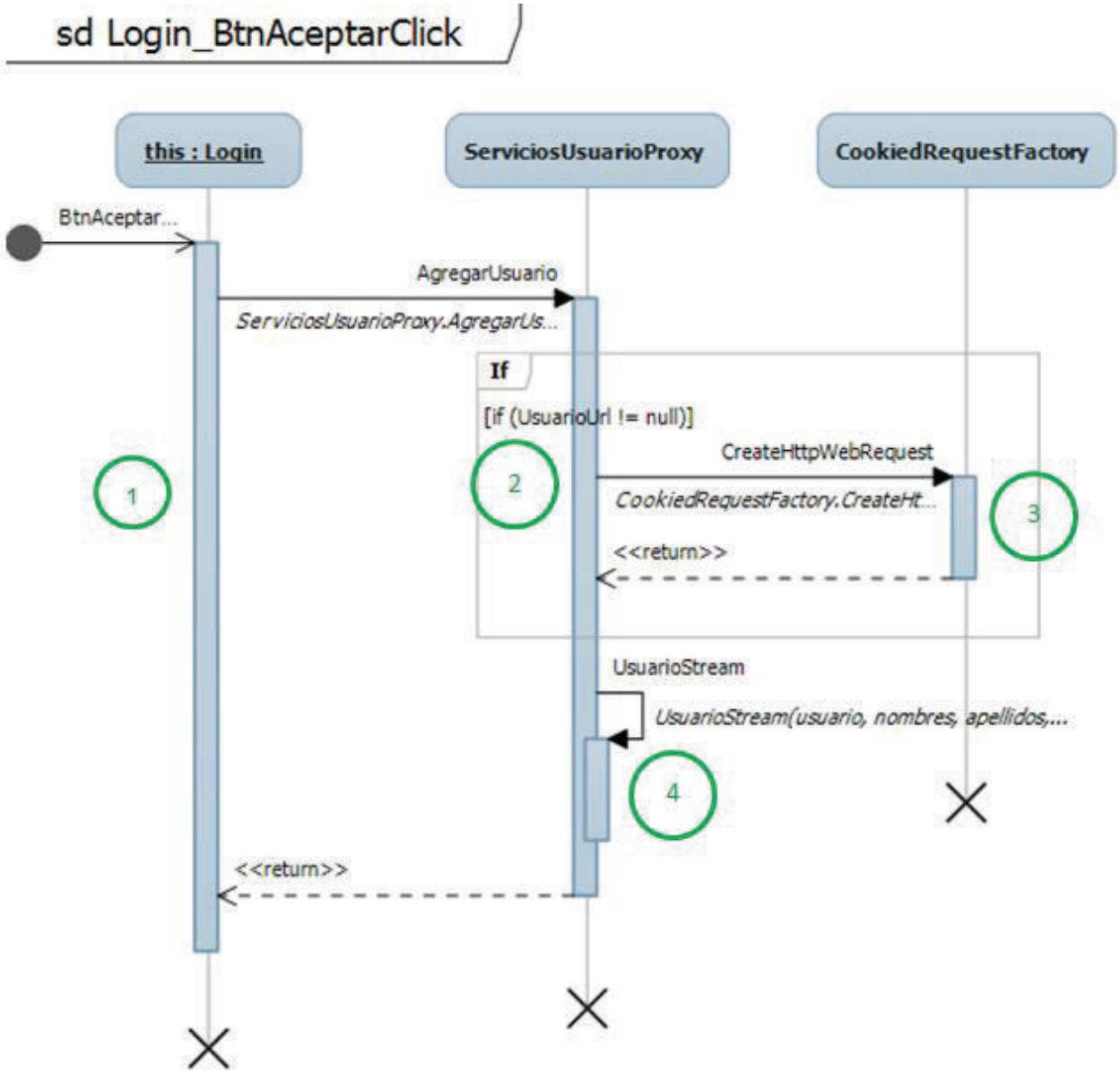


Figura 2.12. Diagrama de secuencia de la interacción entre el formulario web y la capa de generación de solicitudes

1. El usuario hace clic en un control del formulario, el formulario lanza un método de la capa de generación de solicitudes.
2. Se lee la URL del servicio y se da paso a la generación de la solicitud con la librería `CookiedRequestFactory`.
3. Se añade la cookie HTTP de sesión almacenada durante el establecimiento de la comunicación.
4. Cuando se recibe una respuesta del servidor se lanza un proceso encargado de la deserialización de la trama.

Se crearán las clases de la Figura 2.13 para la capa de generación de solicitudes. El cliente solo verá los métodos que necesita en la capa de comunicaciones. Estos métodos se encargan de des serializar el `HTTP Response` y enviar simplemente la información necesaria que solicita el aplicativo web con el fin de presentarla al usuario.

2.8.1 PROCESO

Los siguientes diagramas describen el proceso de interacción entre el aplicativo web y el usuario del sistema, y de este, a través del aplicativo, con el servicio WCF.

El usuario accede al portal del aplicativo web, si es nuevo, llena el formulario de registro y el aplicativo envía la información; dicha información se procesa en el servidor, si todo está bien, se permite el acceso a la interfaz principal.

El portal le da la opción de recordar su contraseña ingresando su nombre de usuario, si está registrado, se le envía su contraseña maestra al correo electrónico.

El camino positivo del proceso es ingresar al portal de autenticación, ingresar credenciales y acceder al sistema.

Por cualquier resultado contrario al proceso positivo, y contrario a los caminos definidos en el párrafo anterior, el usuario no podrá acceder.

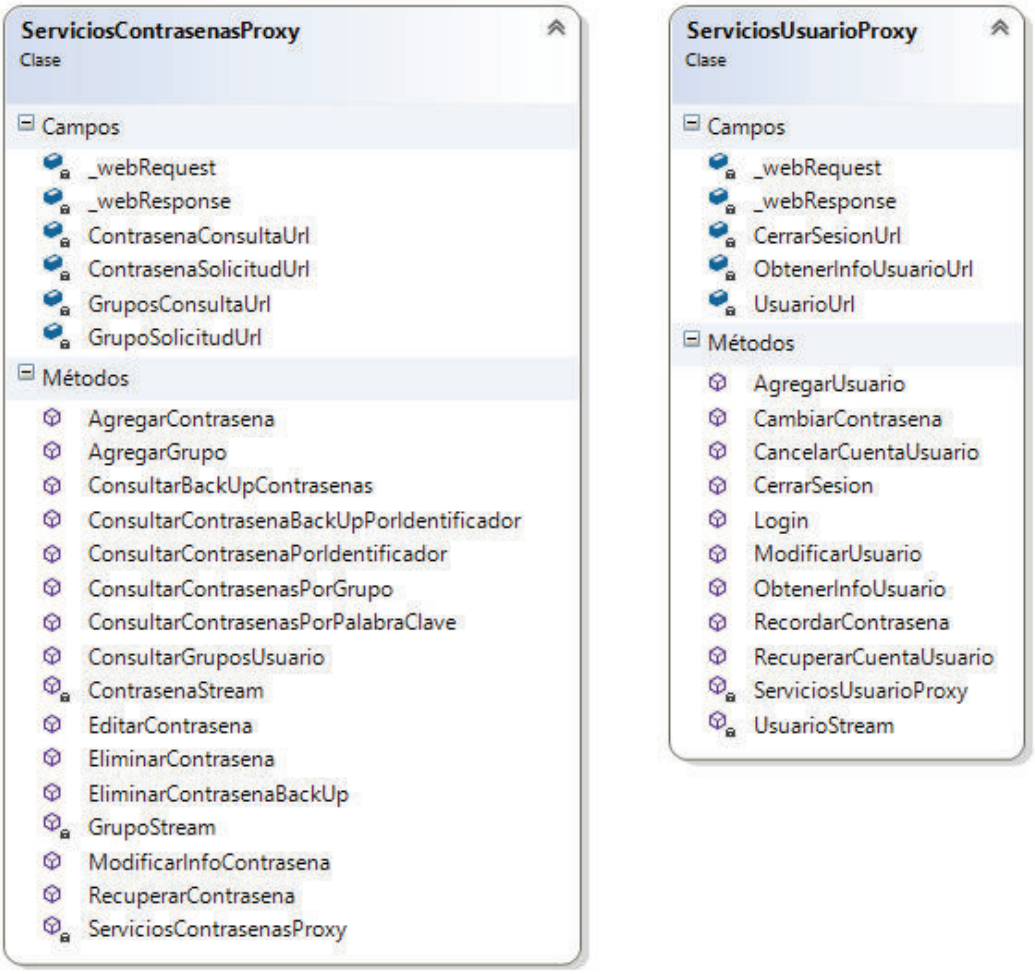


Figura 2.13. Clases de interacción del aplicativo web con el servicio WCF

Una vez en la página principal, el aplicativo web presenta un menú de gestión de usuario, ingreso de grupo y gestión de contraseñas. La gestión de usuario se resume en tres opciones: Modificar la información de su cuenta de usuario, cambiar su contraseña maestra y abandonar el sistema. Modificar una cuenta se resume en recibir su información en un formulario web, editar la información y enviarla al servicio para que se guarde en la base. Cambiar la contraseña maestra se resume en llenar el formulario con los datos necesarios para la gestión requerida, por último, abandonar el sistema es confirmar en un formulario que su cuenta se borre lógicamente del sistema. Un cliente nunca es borrado físicamente ya que siempre tendrá la opción de recuperar su cuenta a través del portal de autenticación del aplicativo web.

- Proceso de autenticación, Figura 2.14 y Figura 2.15:

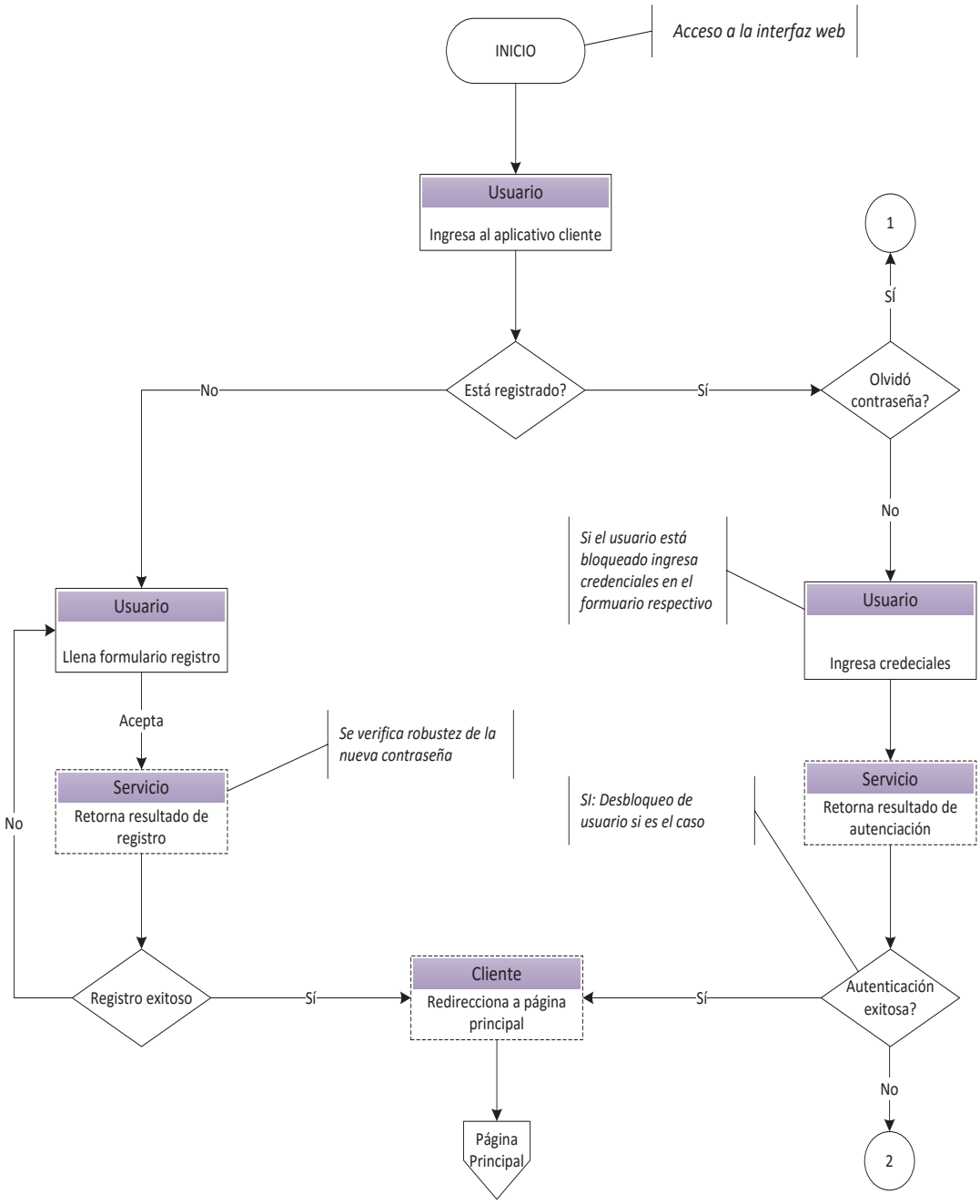


Figura 2.14. Diagrama de procesos del portal de autenticación web parte A

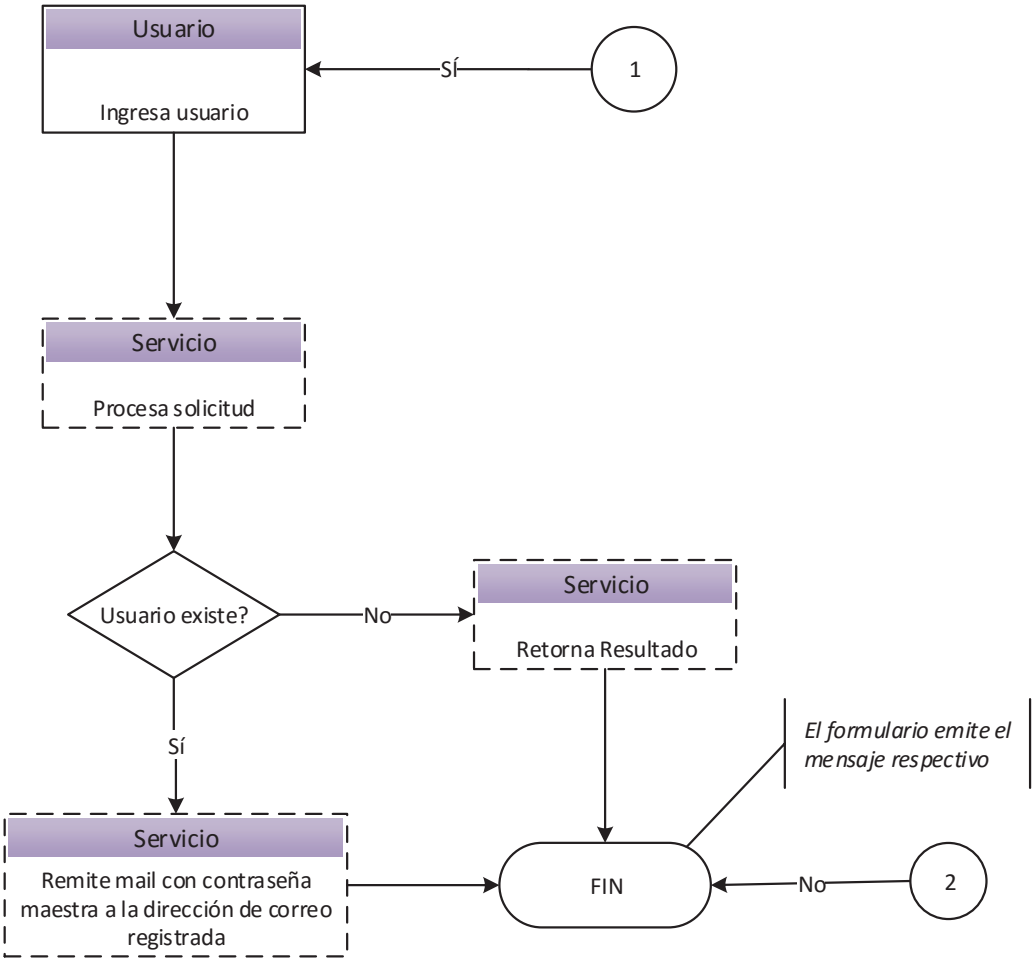


Figura 2.15. Diagrama de procesos del portal de autenticación web parte B

Ingresar un grupo es llenar un formulario con los datos requeridos. Es importante en este punto aclarar que todo registro nuevo en la base de datos, con excepción de la tabla `Usuarios`, cifra campos si es el caso, y secuencia automáticamente las claves primarias desde el servidor, el usuario solo llena la información que es obligatoria para el registro. Como se indica en la sección 0 la tabla `Usuarios` tiene como clave primaria el nombre de usuario o campo `Usuario` de la tabla. La gestión de contraseñas se define en el siguiente proceso.

- Página principal, Figura 2.16:

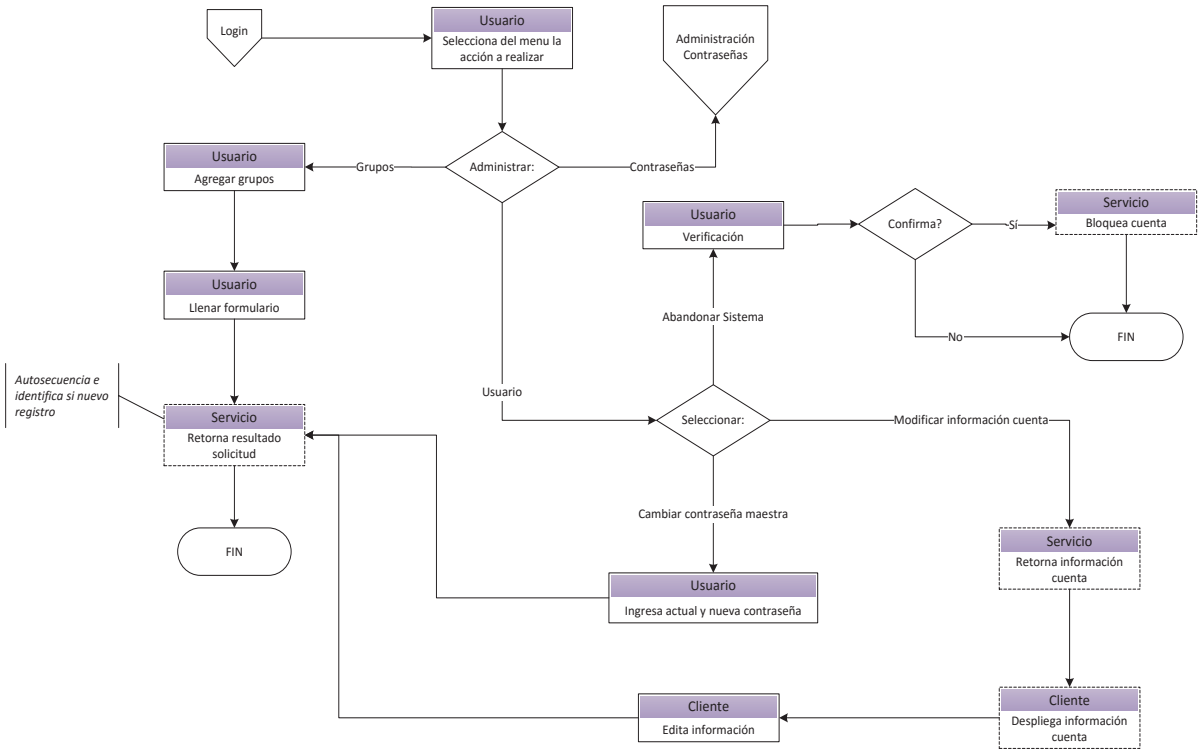


Figura 2.16. Diagrama de procesos de la interfaz principal de la aplicación web

Las opciones para gestionar contraseñas son ingresar, editar, descifrar y eliminar contraseñas, como también recuperar el último cambio editado o la última contraseña eliminada.

Para ingresar una contraseña el usuario selecciona un grupo previamente ingresado o el grupo creado por defecto al instanciar una nueva cuenta en el sistema. Este grupo más la información necesaria se envía al servidor y se registra la contraseña, caso contrario se notifica al usuario.

Para editar una contraseña se selecciona la opción de administración de contraseñas, una vez seleccionada esta opción se despliega una grilla con todas las contraseñas ingresadas en el sistema, el formulario además presenta controles de búsqueda por

palabra clave o por grupo seleccionado en un combo. La palabra clave como se define en la sección 0, en la tabla `Contraseñas`, es un identificador de búsqueda que facilita al usuario recuperar la información deseada sin necesidad de especificar o describir la contraseña en administración. Una vez encontrada la contraseña se procede a seleccionar la opción de edición, y se presenta una pantalla modal en la que se podrá editar la información obtenida a través del servicio y enviarla a grabar en la base mediante otra opción del servicio.

- Interfaz de gestión de contraseñas, Figura 2.17 y Figura 2.18:

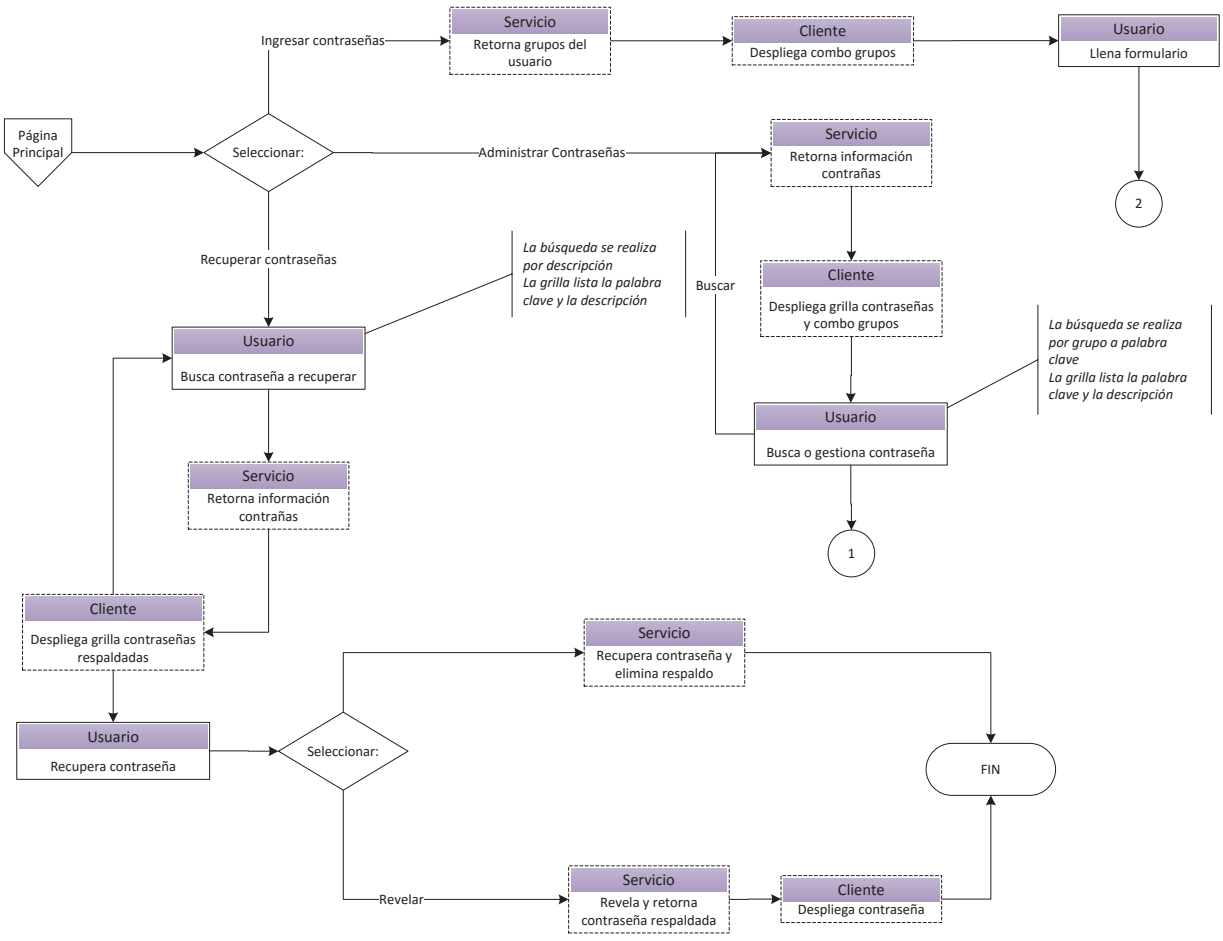


Figura 2.17. Diagrama de procesos de la interfaz de administración de contraseñas del aplicativo web parte A

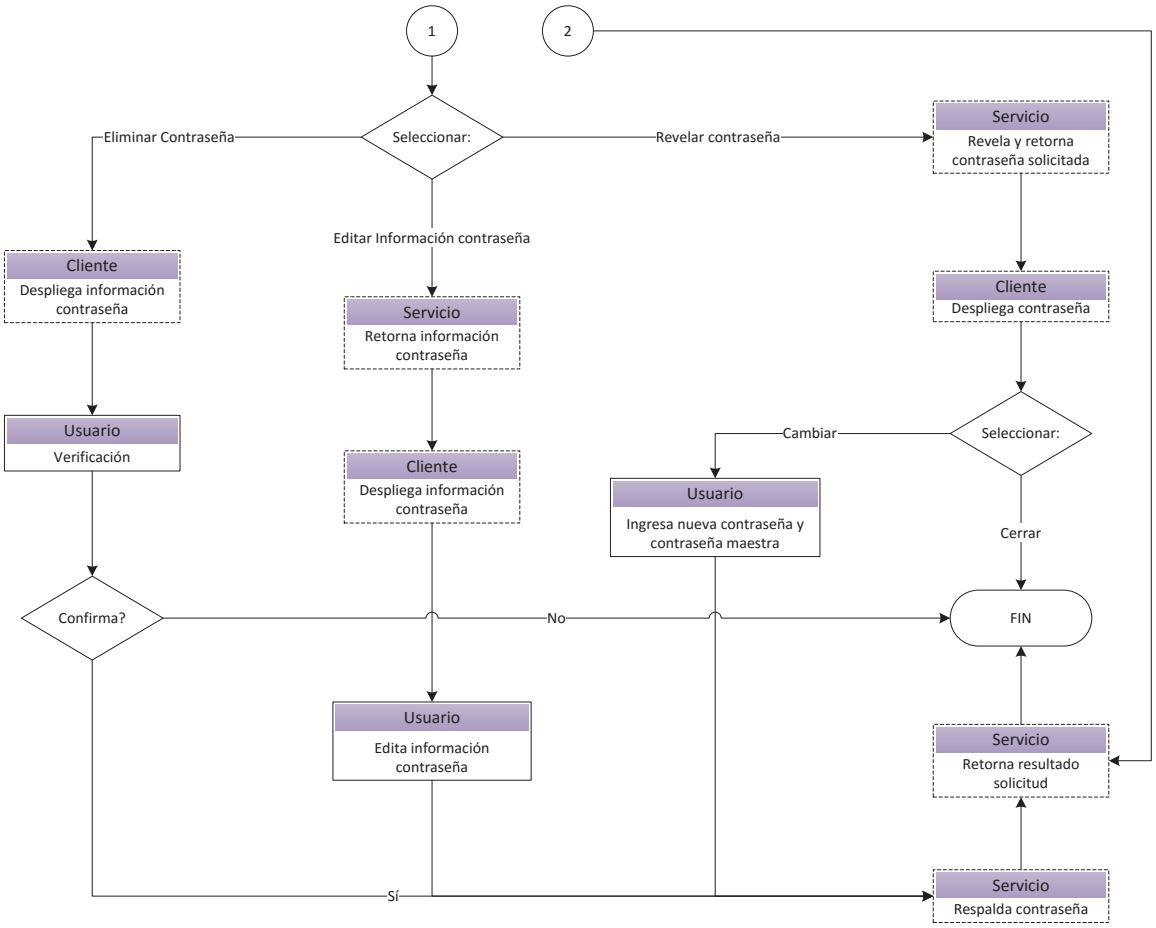


Figura 2.18. Diagrama de procesos de la interfaz de administración de contraseñas del aplicativo web parte B

Otras opciones que se despliegan en la grilla del formulario de administración de contraseñas son el descifrado o revelación de la contraseña guardada y la eliminación del registro.

A través de la opción de descifrado el usuario tiene la posibilidad de editar la contraseña, sin necesidad de borrar o ingresar una nueva entrada en la base. Cuando se elimina o edita una contraseña ya sea su información o la contraseña en sí, el servicio almacena este cambio en la tabla de *backups*, ver sección 0. El usuario puede acceder a una interfaz de recuperación del último cambio en el registro contraseñas o recuperar una contraseña eliminada.

Parar recuperar una contraseña se selecciona la opción de recuperación, se busca la contraseña por su descripción y se la selecciona en la grilla desplegada. Otra opción que se presenta en la grilla es la de mostrar la contraseña sin recuperarla. La búsqueda se realiza por descripción y no por palabra clave, porque esta entrada es más fácil de recordar en el caso de contraseñas eliminadas hace un largo periodo.

Por último, cabe resaltar que las grillas a presentarse en esta interfaz, no traen la contraseña almacenada en la base de datos, simplemente traen información de la misma y la revelación se produce individualmente previa solicitud del usuario mediante la opción publicada para ese fin.

2.9 CLIENTE ANDROID

La aplicación se construiría inicialmente en lenguaje Java en el IDE Eclipse Juno de 4.2.2 del ADT *bundle*, pero durante el desarrollo de aplicativo se migro al IDE Android Studio, ya que el primero dejó de tener soporte oficial [10].

Junto a las capas propias de la plataforma Android, se añadirán dos paquetes: `epn.redes.droidpassmanager` y `epn.redes.proxies` el primero contiene toda la lógica de administración de `ActivityViews` (ver sección 1.7.4) y el segundo se encarga de construir solicitudes HTTP para comunicar el aplicativo con el servicio WCF manteniendo la sesión que se establece durante la negociación de la comunicación.

Con la ayuda del diagrama de clases de la Figura 2.19 se puede resumir el proceso de comunicación con el servicio:

1. La actividad cliente accede al método deseado implementado en la clase *proxy*.
2. La clase *proxy* genera una cadena XML del objeto en cuestión ayudado del objeto `XMlParser`.
3. Una vez generado el XML utiliza un método de la clase `HttpRequest` a la que accede a través de un proceso asíncrono (hilo propio) implementado por objeto `EjecutorAsincrono`. El proceso asíncrono permite que la ejecución de la

aplicación no se interrumpa mientras se tramita la solicitud y se obtiene respuesta en la transacción. Los métodos de la clase `HttpRequest` tienen como parámetro de entrada la cadena XML generada.

4. `HttpRequest` utiliza la clase `CookieContainer` para guardar datos de la sesión con propiedades desde su establecimiento hasta que exista.
5. La clase `MySSLSocketFactory` es utilizada con fines de administración de certificados digitales durante la comunicación con HTTPS.

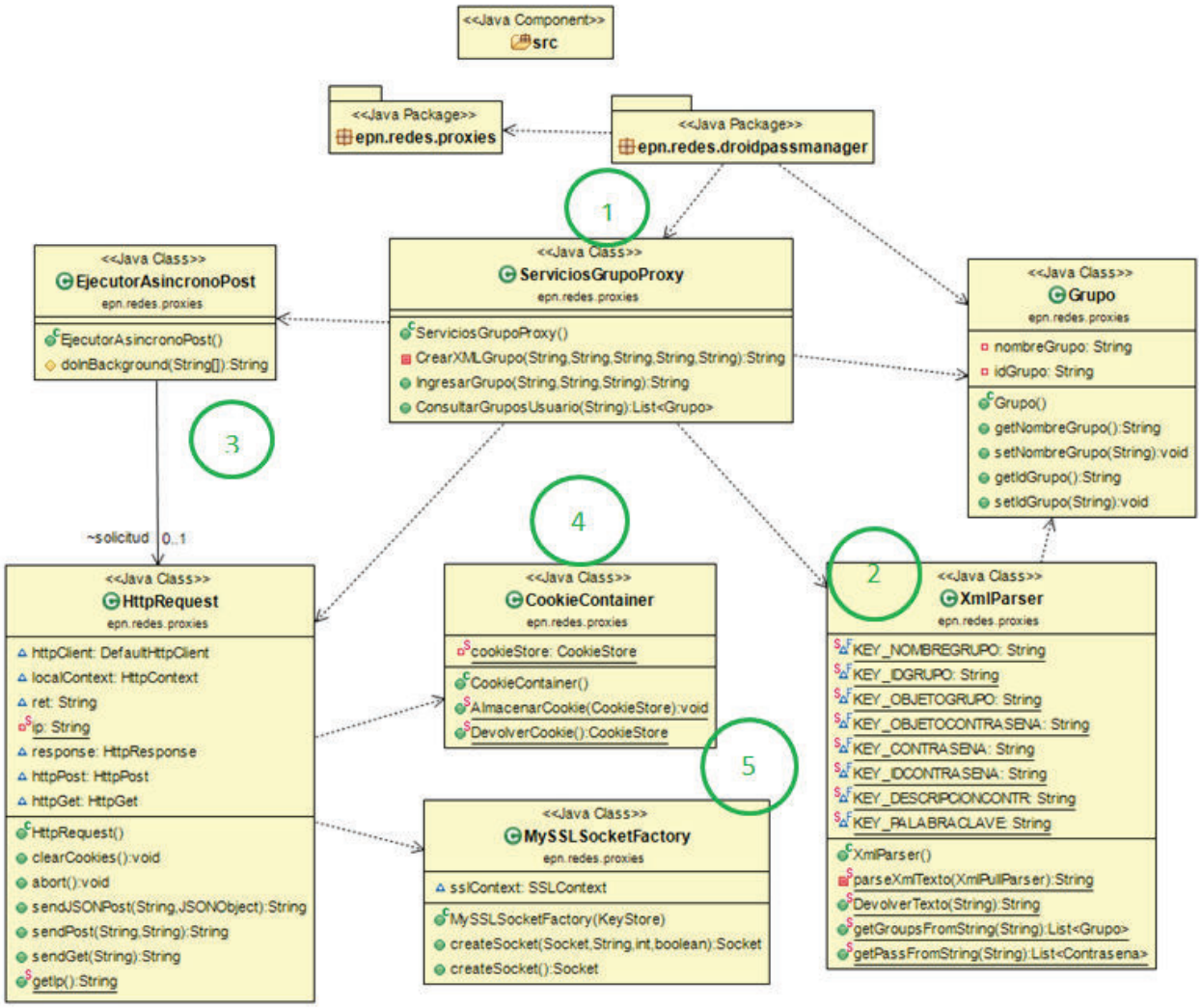


Figura 2.19. Diagrama de clases de comunicación con el servicio WCF del aplicativo Android

2.9.1 PROCESO

Los siguientes diagramas describen la interacción entre el aplicativo Android y el usuario del sistema, y de este, a través del aplicativo, con el servicio WCF.

- Proceso de autenticación, Figura 2.20:

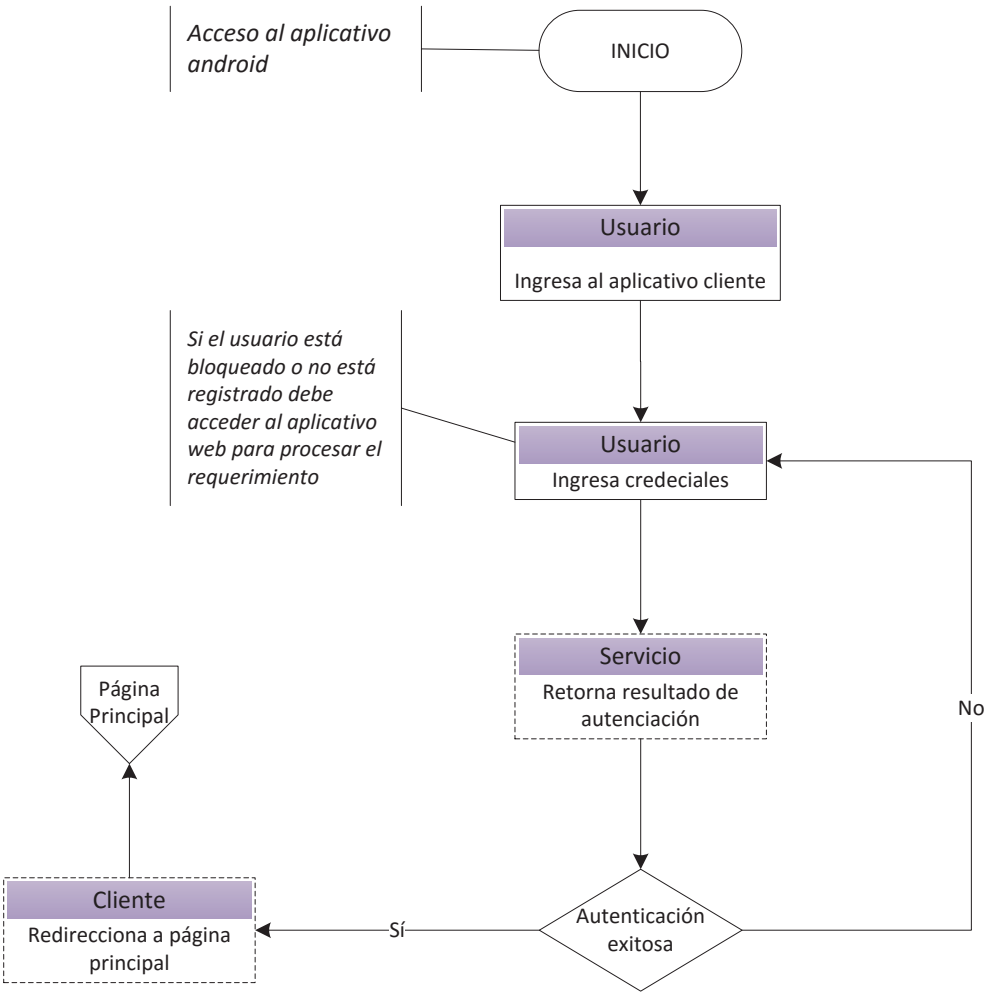


Figura 2.20. Diagrama de procesos del portal de autenticación del aplicativo Android

En el portal de autenticación ingresar las credenciales y acceder al sistema. Por cualquier resultado contrario al proceso positivo el usuario no podrá acceder.

- Gestión de contraseñas, Figura 2.21 y Figura 2.22:

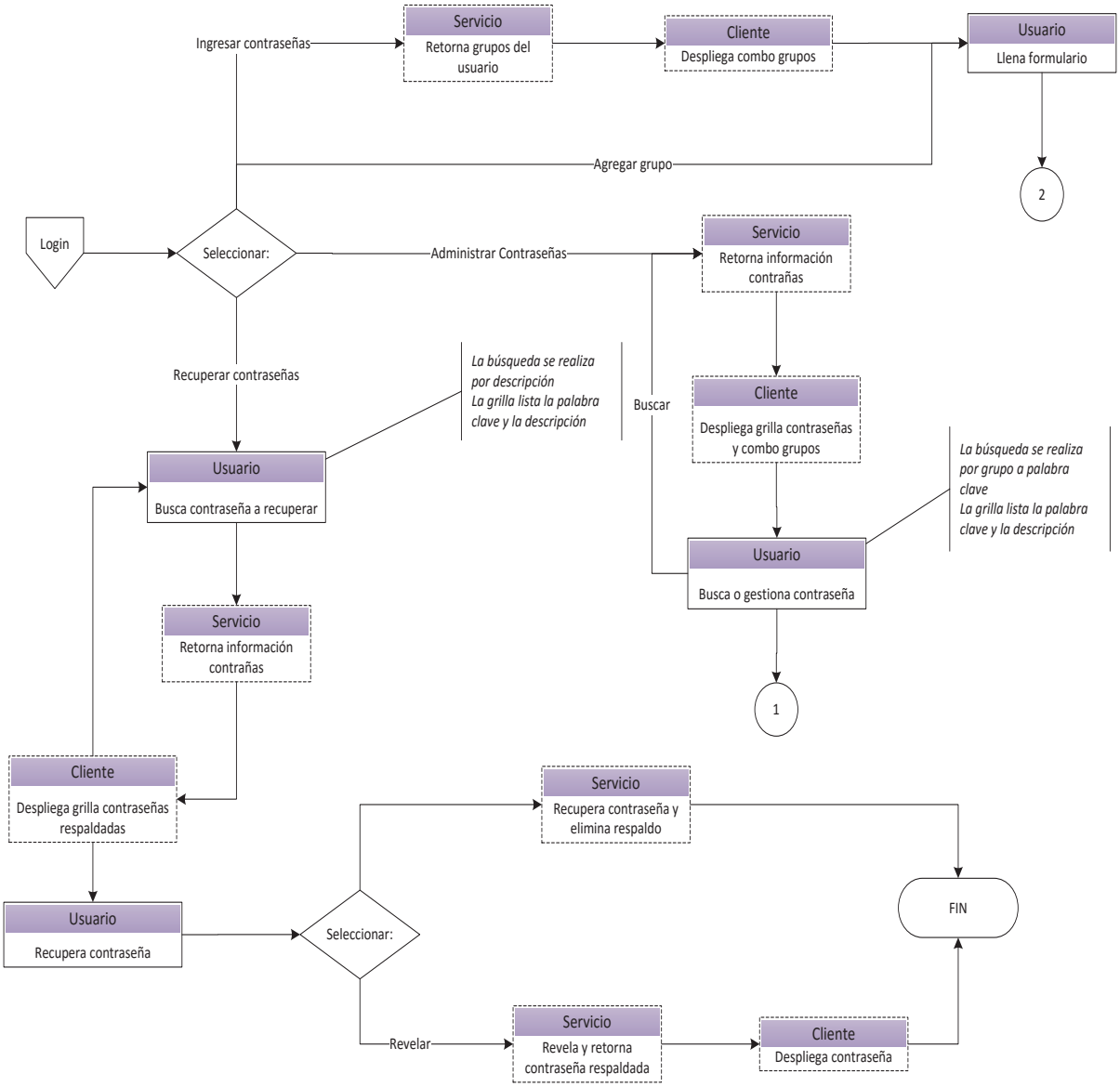


Figura 2.21. Proceso de la interfaz de administración de contraseñas del aplicativo Android parte A

La descripción del proceso es la misma que la de “Interfaz de gestión de contraseñas Web” de la sección 2.8.1, también se incluye el ingreso de grupos, para esto se llena el formulario respectivo y se envía a procesar en el servidor.

El aplicativo Android no procesará información relativa a la gestión de la cuenta de usuario. Las cuentas se procesarán solo a través del aplicativo web, el resto de gestiones se replican.

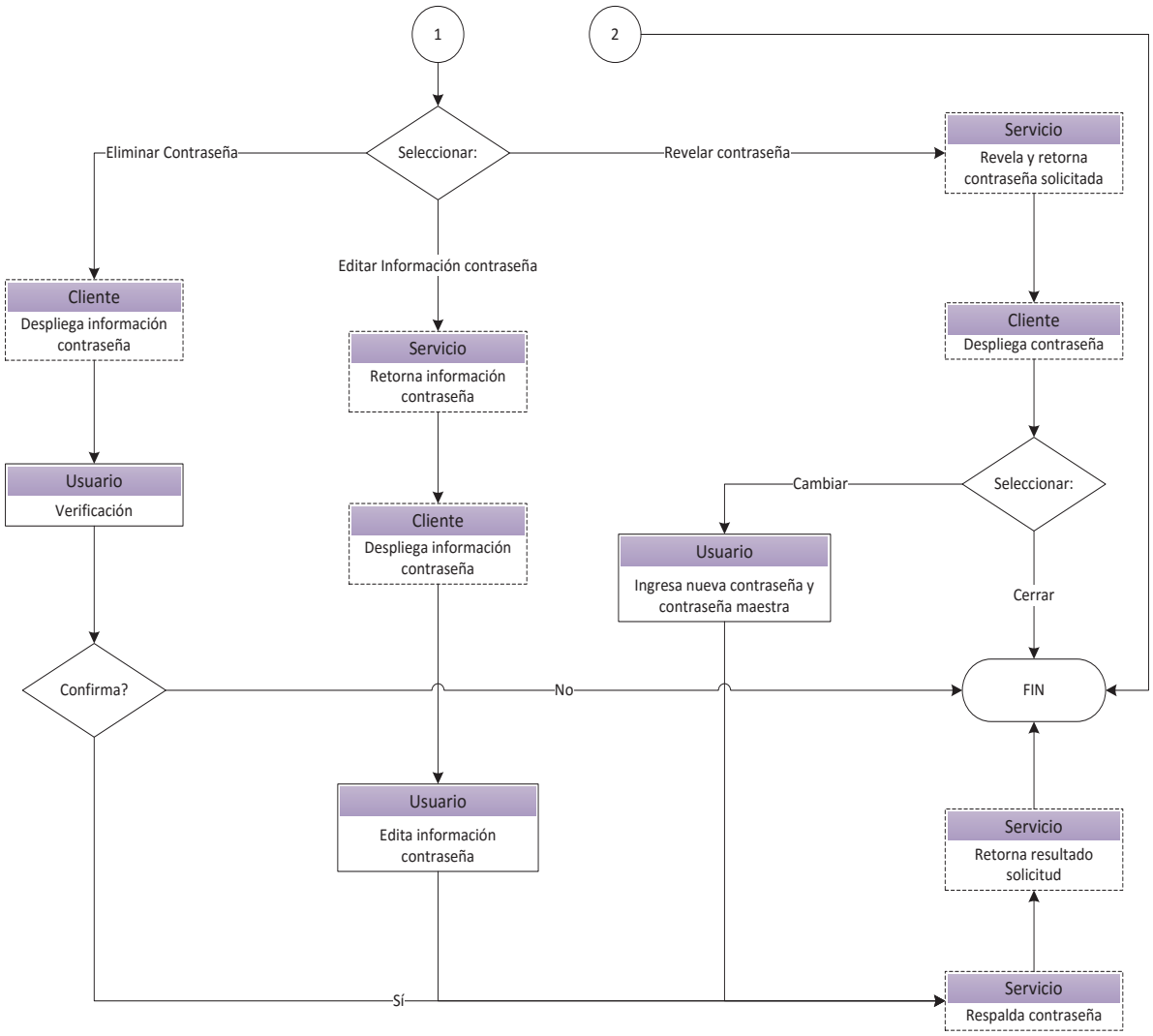


Figura 2.22. Diagrama de procesos de la interfaz de administración de contraseñas del aplicativo Android parte B

CAPÍTULO 3. IMPLEMANTACIÓN, PRUEBAS Y ANÁLISIS DE RESULTADOS

3.1 INTRODUCCIÓN

En este capítulo se describe la implementación de los componentes que conforman el prototipo de sistema distribuido para gestión de contraseñas, se incluye el análisis de los resultados obtenidos a través de las pruebas realizadas al sistema y se presentan las especificaciones mínimas recomendadas para la implantación del prototipo.

Adicionalmente se describen las prácticas, los métodos y las configuraciones empleadas para realizar el *hardening* del prototipo.

3.2 ORGANIZACIÓN DEL SISTEMA

El sistema cliente/servidor estará organizado de la siguiente manera: servicio de base de datos, servicio de Gestión de Contraseñas, aplicativo web cliente, aplicativo Android cliente. Los aplicativos clientes consumirán los Servicios Web del Servicio de Gestión de Contraseñas y este a su vez utilizará el servicio de base de datos para brindar persistencia en la información. La comunicación entre el aplicativo cliente y los Servicios Web mantienen una sesión y la información se transportará en un canal seguro. El cliente solo puede acceder a la base de datos a través del servicio.

3.3 IMPLEMENTACIÓN DE LOS COMPONENTES

3.3.1 BASE DE DATOS

La primera tarjeta del grupo de entregables de levantamiento del servicio corresponde a la implementación de la base de datos, se encuentra en el paso de proceso “Implementación” como se indica en la Figura 3.1. Se tomará la tarjeta y trabajará en la tarea de generación del *script* para la creación de tablas que se encuentra con estado “Activo” como se observa en la Figura 3.2.

En el Segmento de código 3.1 se muestra el script generado. Este script se ejecutará en una base de SQL Server con nombre `GesConDesa`. El script pasó por rediseños para incluir características como el soporte de respaldo de contraseñas por lo que se generó el problema “Rediseño de la base de datos necesario” que para la generación de del script final se encuentra en el estado “Resuelto” como se observa en la Figura 3.2.

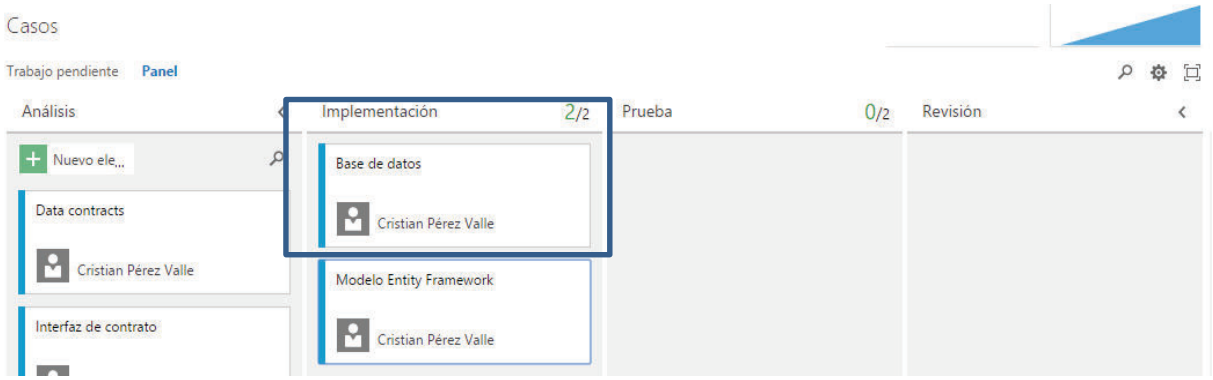


Figura 3.1. Tarjeta “Base de datos” en el paso de proceso “Implementación”

Figura 3.2. Tarea “Generación del script” de creación de tablas y problema “Rediseño de base de datos necesario”

3.3.2 SERVICIO

3.3.2.1 Interfaz

Para la definición de la Interfaz de contrato servicio se tomará la tarjeta del entregable “Interfaz de contrato” que se muestra en la Figura 3.3.

```
CREATE TABLE Usuarios (
Usuario VARCHAR (50) PRIMARY KEY NOT NULL,
Nombres VARCHAR (50) NOT NULL,
Apellidos VARCHAR (50) NOT NULL,
CorreoElectronico VARCHAR (100) NOT NULL,
ContrasenaMaestra VARCHAR (50) NOT NULL,
Estado BIT NOT NULL
)
GO

CREATE TABLE Grupos (
IdGrupo VARCHAR (20) PRIMARY KEY NOT NULL,
IdUsuario VARCHAR (50) FOREIGN KEY REFERENCES Usuarios(Usuario)  NOT NUll,
NombreGrupo VARCHAR (50) NOT NULL,
DescripcionGrupo VARCHAR (1000) NOT NULL
)
GO

CREATE TABLE Contrasenas (
IdContrasena VARCHAR (20) PRIMARY KEY NOT NULL,
Contrasena VARCHAR (500) NOT NULL,
PalabraClave VARCHAR (100) NOT NULL,
DescripcionContrasena VARCHAR (1000) NOT NULL
)
GO

CREATE TABLE ContrasenaGrupo (
IdGrupo VARCHAR (20) FOREIGN KEY REFERENCES Grupos(IdGrupo),
IdContrasena VARCHAR (20) FOREIGN KEY REFERENCES  Contrasenas(IdContrasena),
Dibujador VARCHAR (1) NULL,
PRIMARY KEY (IdGrupo, IdContrasena)
)
GO

CREATE TABLE ContrasenasBackup (
IdContrasena VARCHAR (20) PRIMARY KEY NOT NULL,
Contrasena VARCHAR (500) NOT NULL,
PalabraClave VARCHAR (100) NOT NULL,
DescripcionContrasena VARCHAR (1000) NOT NULL,
IdUsuario VARCHAR (50) NOT NULL
)
GO
```

Segmento de código 3.1. *Script* de creación de las tablas de la base de datos del prototipo

En el Segmento de código 3.2 se describe un ejemplo de cómo se implementará la interfaz de contrato (la interfaz completa se encuentra en el Anexo B.1), se atiende las tareas de la tarjeta que se muestran en la Figura 3.4. De este segmento de código se pueden definirá sus atributos:

```
[ServiceContract]
    public interface IServiciosGestorContrasenas
    {
        /// <summary>
        /// Contrato que redirecciona por id la solicitud del usuario
        /// </summary>
        /// <param name="usuarioIn">objeto usuario solicitante</param>
        /// <returns>devuelve el resultado exitoso o no de la autenticación</returns>
        [OperationContract]
        [WebInvoke(Method = "POST", ResponseFormat = WebMessageFormat.Xml,
            BodyStyle = WebMessageBodyStyle.Bare, UriTemplate = "solicitudUsuario")]
        string ConstruirSolicitudUsuario(ObjetoUsuario usuarioIn);


        /// <summary>
        /// Verifica si existe sesión y la cierra
        /// </summary>
        /// <returns>true: si la sesión no existe o se la finalizó</returns>
        [OperationContract]
        [WebGet(UriTemplate = "/cerrarSesion")]
        string CerrarSesion();
    }
```

1
2
3

Segmento de código 3.2. Interfaz de contrato del Servicio WCF

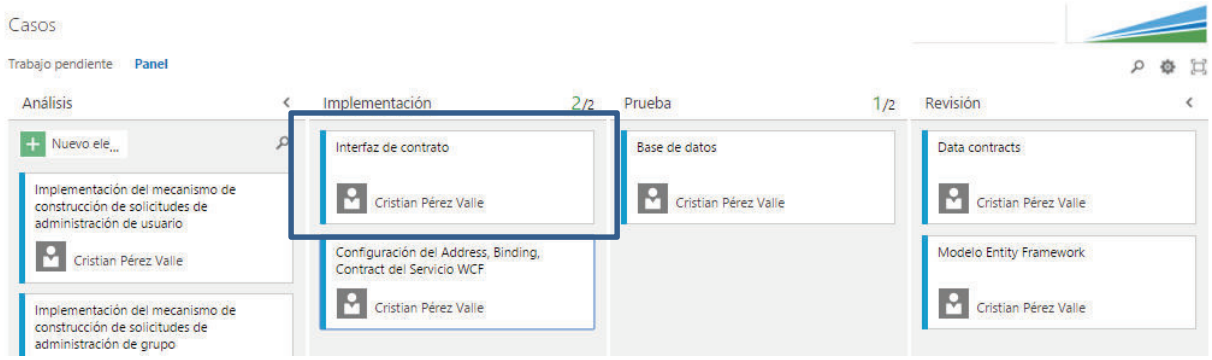


Figura 3.3. Tarjeta “Interfaz de contrato” en el paso de proceso “Implementación”

1. El atributo `ServiceContractAttribute` le da la a la interfaz la propiedad de contrato de servicio.

2. El `OperationContractAttribute` en este caso indica que a continuación hay una operación del Servicio WCF invocada a través de HTTP POST.

- `WebInvokeAttribute`, este atributo indica que la operación va a ser invocada bajo el modelo WCF REST.

    - `Method`, con esta propiedad se indica el método HTTP al que responde la operación, aquí se define que este servicio va a ser invocado a través de HTTP POST.

    - `ResponseFormat`, con esta propiedad se indica el formato que va a encapsular al tipo de dato. Para WCF REST se puede usar JSON y XML.

    - `BodyStile`, con esta propiedad se indica como los parámetros son recibidos y enviados en la operación, la selección `WebMessageBodyStyle.Bare` define que no se encapsule en ningún objeto o colección tanto a las solicitudes como a las respuestas.

    - `UriTemplate`, con esta propiedad se establece el identificador único de recursos de la operación.

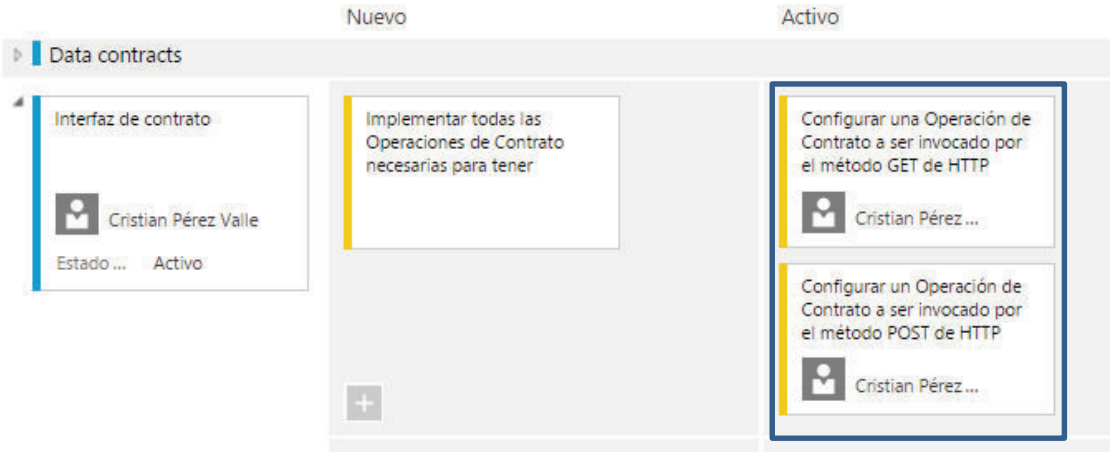


Figura 3.4. Tareas de la tarjeta “Interfaz de contrato”

3. El `OperationContractAttribute` en este caso indica que a continuación hay una operación del Servicio WCF invocada a través de HTTP GET.

    - `WebGetAttribute`, con este atributo se define a la operación como un servicio que va a ser invocado bajo un modelo WCF REST, el método HTTP GET va implícito en el atributo.

        - `UriTemplate`, con esta propiedad se establece el identificador único de recursos de la operación.

3.3.2.2 Configuración del Servicio

Como se observa en la Figura 3.5, dos entregables se encuentran en la fase de implantación, señalado en el círculo verde muestra el límite de trabajo en curso para esta etapa.

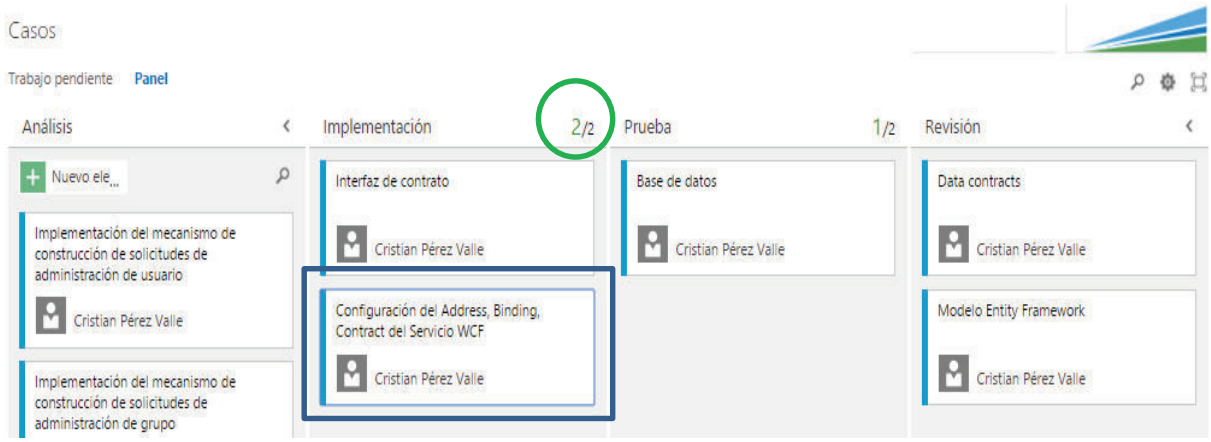


Figura 3.5. Tarjeta "Configuración del *Address*, *Binding*, *Contract* del Servicio WCF" en el paso de proceso "Implementación"

A continuación, se describen las etiquetas y atributos del archivo de configuración a implementar del Segmento de código 3.3:

1. En la etiqueta `<system.serviceModel>` están todos los elementos de configuración de un Servicio WCF. Entre los elementos fundamentales están `<bindings>`, `<services>` y `<behaviors>`.

2. Con la etiqueta `<bindings>` se especifican las características del protocolo de transporte y codificación con las que se enviará y recibirá la información.

   En la etiqueta `<webHttpBinding>` se define un elemento de enlace que se utiliza para configurar los *endpoints* que responden a solicitudes HTTP en lugar de mensajes SOAP, es decir que este *binding* en particular define al Servicio WCF como un servicio basado en REST.

   Con `<binding>` se personaliza el enlace, con `<security>` se especifica que en enlace usa un protocolo de seguridad a nivel de transporte, al ser solicitudes HTTP, la seguridad a nivel de transporte es establecida con HTTPS.

3. Mediante la etiqueta `<services>` se define el servicio, con la etiqueta `<service>` se configura el Servicio WCF, en la etiqueta `<endpoint>` se configura el ABC (*address, binding, contract*). En el Segmento de código 3.3 se observa que estas configuraciones tienen propiedades que definen al servicio como WCF REST con el *behavior* web y el `webHttpBinding`.

4. En la etiqueta `<behaviors>` se configura el comportamiento tanto del *endpoint* usando la etiqueta `<endpointBehaviors> y` el comportamiento del servicio usando la etiqueta `<serviceBehaviors>`. Estas etiquetas serán utilizadas en la definición de servicios en la etiqueta `<services>`.

5. En la etiqueta `<protocolMapping>` se asignan esquemas del protocolo de transporte a soportarse por el servicio. En el Segmento de código 3.3 se puede observar las configuraciones utilizadas por el prototipo.

```
<system.serviceModel>
  <bindings>                                                    1
    <webHttpBinding>                                            2
      <binding name="LargeWeb" maxBufferPoolSize="2147483647"
               maxReceivedMessageSize="2147483647" maxBufferSize="2147483647"
               transferMode="Streamed">
        <security mode="Transport" />
      </binding>
    </webHttpBinding>
  </bindings>
  <services>
    <service behaviorConfiguration="ServiceBehaviour"           3
             name="ServiciosWcf.ServiciosGestorContrasenas">
      <endpoint address="" behaviorConfiguration="Web" binding="webHttpBinding"
                bindingConfiguration="LargeWeb"
                contract="ServiciosWcf.IServiciosGestorContrasenas">
        <identity>
          <dns value="localhost" />
        </identity>
      </endpoint>
    </service>
  </services>
  <behaviors>
    <endpointBehaviors>                                         4
      <behavior name="Web">
        <webHttp />
      </behavior>
    </endpointBehaviors>
    <serviceBehaviors>
      <behavior name="ServiceBehaviour">
        <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true" />
        <serviceDebug includeExceptionDetailInFaults="false" />
      </behavior>
    </serviceBehaviors>
  </behaviors>
  <protocolMapping>                                             5
    <add binding="basicHttpsBinding" scheme="https" />
  </protocolMapping>

</system.serviceModel>
```

Segmento de código 3.3. Elementos de configuración del Servicio WCF

3.3.2.3 Servicios del Gestor de Contraseñas

En la interfaz de contrato y en las configuraciones del servicio se establece cómo el Servicio WCF va ser publicado; sin embargo, la funcionalidad que presta está definida en una clase con extensión `.svc` que representa a el Servicio WCF hospedado en un servidor web con IIS.

El archivo `ServiciosGestorContrasenas.svc` tendrá 3 regiones que son `Usuarios`, `Grupos` y `Contraseñas`, en el Segmento de código 3.4 se muestra las regiones del archivo y las variables globales utilizadas.

Como variables globales de la clase `ServiciosGestorContrasenas.svc` tenemos el modelamiento *Entity Framework*, con nombre `GesConDesaEntities()`, el cuál es un entregable que se encuentra ya en el paso de proceso “Revisión” como se observa en la Figura 3.6.

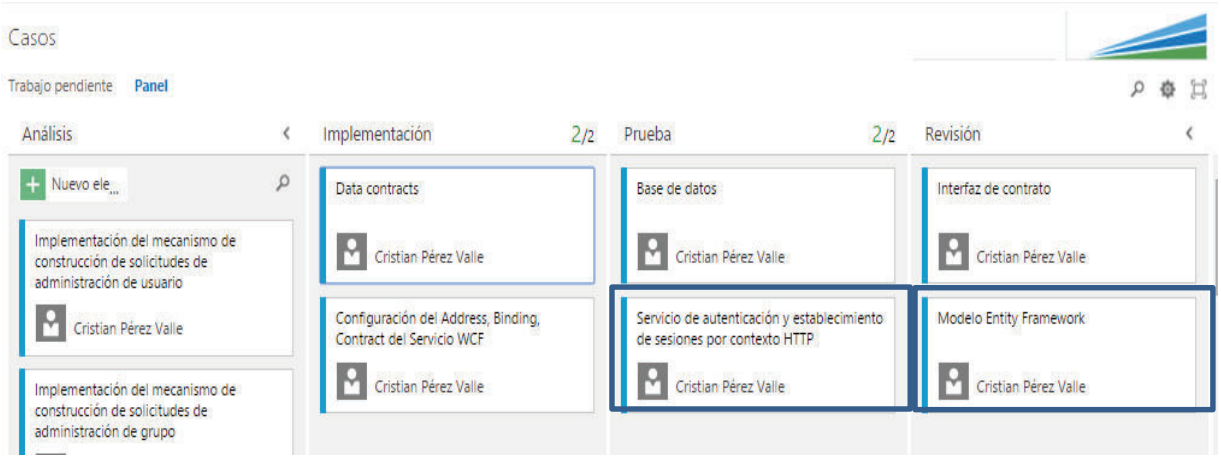


Figura 3.6. Tarjetas “Model Entity Framework” y “Servicio de autenticación y establecimiento de sesiones por contexto HTTP”

También hay una variable global por cada `Entity Class`. Estas variables permitirán la administración de la información intercambiada con la base de datos. Otra variable global que se observa en Segmento de código 3.4, es el contexto HTTP[1], usado para gestionar sesiones de usuario en el servidor que se describirá en la sección 3.4.2.1. En la Figura 3.6 además se observa el entregable del “Servicio de autenticación y establecimiento de sesiones por contexto HTTP”.

[1] **Contexto HTTP**: Encapsula toda la información HTTP específica acerca de una petición HTTP individual.

```
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Net.Mail;
using System.Security.Cryptography;
using System.Text;
using System.Web;
using ServiciosWcf.Modelo;

namespace ServiciosWcf
{
    public class ServiciosGestorContrasenas : IServiciosGestorContrasenas
    {
        /// <summary>
        /// Variables globales privadas para el desempeño de los servicios generados
        /// </summary>
        readonly GesConDesaEntities _entidadConexionBdd = new GesConDesaEntities();
        Usuarios _usuario = new Usuarios();
        Grupos _grupo = new Grupos();
        Contrasenas _contrasena = new Contrasenas();
        ContrasenaGrupo _contrasenaGrupo = new ContrasenaGrupo();
        ContrasenasBackup _respaldoContrasena = new ContrasenasBackup();
        readonly HttpContext _context;

        public ServiciosGestorContrasenas()
        {
            _context = HttpContext.Current;
        }

        Servicios de administración de usuarios

        Servicios de administración de grupos de contraseñas

        Servicios de administración de contraseñas
    }
}
```

Segmento de código 3.4. Estructura de la clase ServiciosGestorContrasenas.svc

Los servicios se prestarán como un catálogo de funciones a los que se accederá mediante un identificador. El servicio utiliza tres objetos que son `ObjetoUsuario`, `ObjetoGrupo` y `ObjetoContrasena`, cada objeto tiene los atributos de una entidad de la base de datos más otros atributos para ser utilizados por el servicio. En el Segmento de código 3.5 se muestra un ejemplo de la estructura de estos objetos, que se implementa para el entregable “Data Contracts” y la tarea “Implementar el `DataContract` para interactuar con el `Entity Class` Contraseña”, como se puede observar en la Figura 3.7, para ver el resto de `DataContracts` dirigirse al Anexo B.2.

La etiqueta `[Serializable]` indica que el objeto puede serializarse, la etiqueta `[DataContract]` indica que el objeto puede ser implementado como contrato de datos, y la etiqueta `[DataMember]` indica que forma parte de un contrato de datos. Se define un tipo de dato compuesto que se serializa para conducir la información que llevan las peticiones y respuestas hacia y desde el servicio.

El dato a transmitir se define en el contrato de servicios (`Contract`) y se estructura en el contrato de datos (`DataContract)` y resulta ser el parámetro de entrada que el servicio aceptará.

El Segmento de código 3.6 muestra el servicio `ConstruirSolicitudContrasena` que se accede con la Interfaz de Contrato. El argumento de entrada es un tipo de dato compuesto definido en la clase `ObjetoContrasena.cs`. Una vez definido este servicio se cierra la tarea “Generación de un método que permite realizar solicitudes de administración de contraseñas” del entregable “Implementación de un mecanismo de construcción de solicitudes de administración de contraseñas”, como se observa en la Figura 3.8.

El servicio utiliza un identificador para encontrar el método que lo lee desde el argumento de entrada y así llega a la función que requiere; por ejemplo, `AgregarContrasena(contrasena)` permite ingresar un registro de contraseña en la base de datos.

Figura 3.7. Tarea “Implementar el `DataContract` para interactuar con el `Entity Class Contraseña`” de la tarjeta “Data Contracts”

```
using System;
using System.Runtime.Serialization;

namespace ServiciosWcf.Modelo
{
    [Serializable]
    [DataContract(Namespace = "http//localhost:5310/gestorContrasenas")]
    public class ObjetoContrasena
    {
        [DataMember]
        public string Contrasena { get; set; }
        [DataMember]
        public string DescripcionContrasena { get; set; }
        [DataMember]
        public string Id { get; set; }
        [DataMember]
        public string IdContrasena { get; set; }
        [DataMember]
        public string IdUsuario { get; set; }
        [DataMember]
        public string PalabraClave { get; set; }
        [DataMember]
        public string IdGrupo { get; set; }
    }
}
```

Segmento de código 3.5. Clase `ObjetoContrasena.cs`

El servicio usa una variable de sesión denominada `HTTPContext`, enlazada al objeto de servicio creado para atender a un usuario para manejo de sesiones que se detalla en la sección 3.4.2.1.

Los objetos que gestiona el servidor (`grupos`, `contraseñas` y `usuarios`) tienen dos tipos de servicios:

1. Servicios que ejecutan acciones sobre los datos y retornan el éxito o no.

2. Servicios que ejecutan consultas sobre la base y retornan objetos o colecciones de objetos.

```
public string ConstruirSolicitudContrasena(ObjetoContrasena contrasena)
     {
         string retorno;
         try
         {
             switch (contrasena.Id)
             {
                 case "s1":
                     retorno = (string)_context.Session["USERLOGGEDIN"] == "YES" ?
                        AgregarContrasena(contrasena) : "r0";
                     break;
                 case "s2":
                     retorno = (string)_context.Session["USERLOGGEDIN"] == "YES" ?
                        ModificarInfoContrasena(contrasena) : "r0";
                     break;
                 case "s3":
                     retorno = (string)_context.Session["USERLOGGEDIN"] == "YES" ?
                        ConstularContrasenaPorIdentificador(contrasena.IdContrasen
                        ) : "r0";
                     break;
                 case "s4":
                     retorno = (string)_context.Session["USERLOGGEDIN"] == "YES" ?
                        EliminarContrasena(contrasena) : "r0";
                     break;
                 case "s5":
                     retorno = (string)_context.Session["USERLOGGEDIN"] == "YES" ?
                        EditarContrasena(contrasena) : "r0";
                     break;
                 case "s6":
                     retorno = (string)_context.Session["USERLOGGEDIN"] == "YES" ?
                        RecuperarContrasena(contrasena) : "r0";
                     break;
                 case "s7":
                     retorno = (string)_context.Session["USERLOGGEDIN"] == "YES" ?
                        EliminarContrasenaBackUp(contrasena.IdContrasena) : "r0";
                     break;
                 case "s8":
                     retorno = (string)_context.Session["USERLOGGEDIN"] == "YES" ?
                        ConsultarContrasenaBackUpPorIdentificador(contrasena.
                               IdContrasena : "r0";
                     break;
                 default:
                     retorno = "r5";
                     break;
             }
         }
         catch (Exception)
         {
             retorno = "r7";
         }
         return retorno;}
```

Segmento de código 3.6. Definición del Servicio `ConstruirSolicitudContrasena`

Figura 3.8. Tarea “Generación de un método que permite realizar solicitudes de administración de contraseñas” del entregable “Implementación de un mecanismo de construcción de solicitudes de administración de contraseñas” cerrada.

El servicio `ConstruirConsultaContrasenas` servirá para ejecutar consultas y se lo puede observar en el Segmento de código 3.7, este tipo de servicio retornará una colección de datos encapsulados en XML.

Una vez definido el servicio se cierra la tarea “Generación de un método que permita realizar consultas en la base relacionadas al `DataContract` Contraseñas” del entregable “Implementación del mecanismo de construcción de consultas de contraseña”, como se observa en la Figura 3.9., además en la misma figura que la tarea “Implementar todas las Operaciones de Contrato necesarias para tener completo el Servicio de Contrato” de la tarjeta “Interfaz de contrato” se encuentra “Activa” ya que se cerrará una vez se definan todos los servicios, ya que aquí se programan las interfaces para llegar a ellos. El resto de servicios a publicar se encuentran en el Anexo B.3.

```
public List<ObjetoContrasena> ConstruirConsultaContrasenas(ObjetoContrasena contrasena)
        {
            List<ObjetoContrasena> retorno;

            try
            {
                switch (contrasena.Id)
                {
                    case "c1":
                        retorno = (string)_context.Session["USERLOGGEDIN"] == "YES" ?
                           ConstularContrasenasPorGrupo(contrasena.IdGrupo) : null;
                        break;
                    case "c2":
                        retorno = (string)_context.Session["USERLOGGEDIN"] == "YES" ?
                           ConsultarContrasenasPorPalabraClave(contrasena) : null;
                        break;
                    case "c3":
                        retorno = (string)_context.Session["USERLOGGEDIN"] == "YES" ?
                           ConsultarContrasenaBackUpPorDescripcion(contrasena) : null;
                        break;
                    default:
                        retorno = null;
                        break;
                }
            }
            catch (Exception)
            {
                retorno = null;
            }
            return retorno;

        }
```

Segmento de código 3.7. Servicio `ConstruirConsultaContrasenas`

Se estudiará como ejemplo el método `EditarContrasena()` del entregable “Servicio para modificar contraseñas” de la Figura 3.10.

El método se puede observar en el Segmento de código 3.8 y su código tiene casi la totalidad de ejecuciones que se llevan a cabo en el servidor para crear, leer, actualizar y eliminar registros sobre la base de datos con LINQ y *Entity Framework*.

El resto de métodos del servicio `ConstruirSolicitudContrasena` y de los otros servicios a publicar se encuentran en el Anexo B.4.

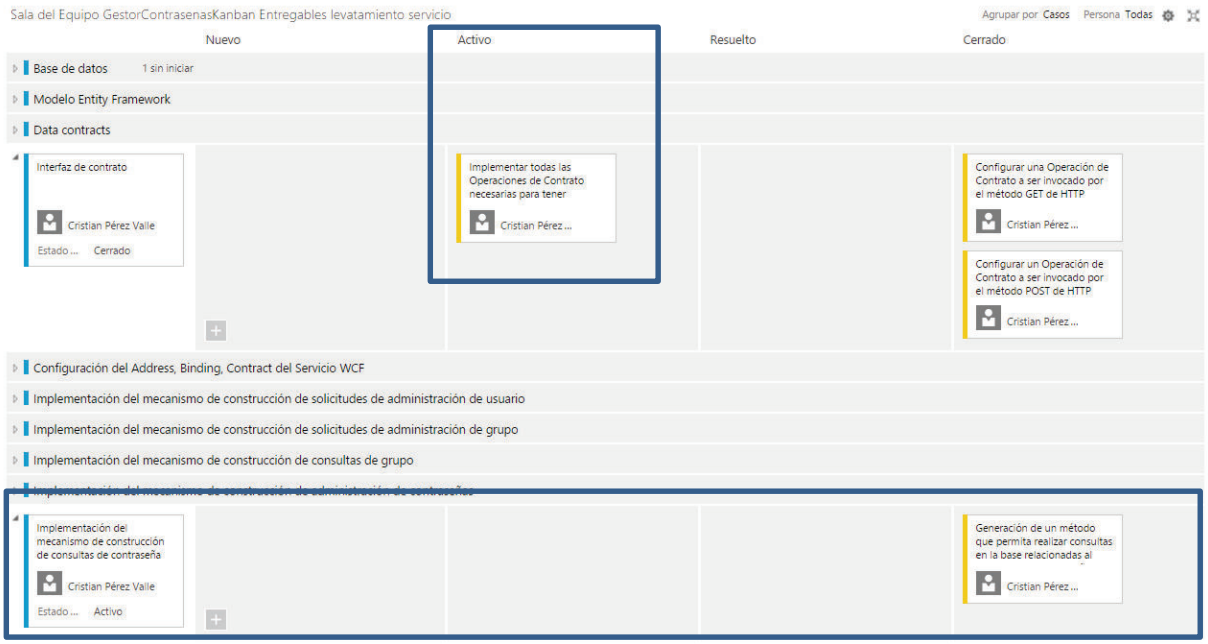


Figura 3.9. Tarea “Generación de un método que permita realizar consultas en la base relacionadas al `DataContract` Contraseñas” cerrada y tarea “Implementar todas las Operaciones de Contrato Necesarias”

1. Con información de la instancia `ObjetoContrasena`, se hace una consulta sobre la base para crear una instancia `ObjetoUsuario`.
2. Si la resultando del método `Autenticar()` es `r1`, es decir, hubo éxito en la autenticación, se continua procesando la respuesta, para cualquier otro resultado, el método retorna la respuesta `r3` que indica que no hubo éxito durante la operación.
3. Si existe la contraseña y el respaldo, se reemplaza la información del último respaldo por el contenido de la contraseña. Luego se edita el registro de la contraseña.
4. Se busca la contraseña en la tabla de contraseñas como en la de respaldos.
5. Si existe la contraseña, pero no se ha creado aún un respaldo, se crea un nuevo registro en la tabla de respaldo, luego se editar el registro en la base. Finalmente se guardan los cambios en la base ayudados del modelo *Entity Framework*, y se envía el resultado de éxito `r1`; a cualquier otra excepción se retornará error `r2`.

```
public string EditarContrasena(ObjetoContrasena contrasenaIn)
{
    ObjetoUsuario usuarioIn = new ObjetoUsuario
                                  {
                                      Usuario = contrasenaIn.IdUsuario,
                                      ContrasenaMaestra =
                                          contrasenaIn.PalabraClave
                                  };
    string resultado;

    if (Autenticar(usuarioIn) == "r1")
    {
        try
        {
            _contrasena = _entidadConexionBdd.Contrasenas.
                FirstOrDefault(c => c.IdContrasena ==
                   contrasenaIn.IdContrasena);

            _respaldoContrasena = _entidadConexionBdd.ContrasenasBackup.
           FirstOrDefault(b => b.IdContrasena == contrasenaIn.IdContrasena);

            if (_contrasena != null && _respaldoContrasena != null)
            {
                _respaldoContrasena.DescripcionContrasena =
                   _contrasena.DescripcionContrasena;
                _respaldoContrasena.PalabraClave = _contrasena.PalabraClave;
                _respaldoContrasena.Contrasena = _contrasena.Contrasena;
                _respaldoContrasena.IdUsuario = contrasenaIn.IdUsuario;

                _contrasena.Contrasena =
                   EncriptarContrasena(contrasenaIn.Contrasena);
            }
            else if (_contrasena != null && _respaldoContrasena == null)
            {
                ContrasenasBackup nuevoRespaldoContrasena =
                    new ContrasenasBackup();
                nuevoRespaldoContrasena.IdContrasena = _contrasena.IdContrasena;
                nuevoRespaldoContrasena.Contrasena = _contrasena.Contrasena;
                nuevoRespaldoContrasena.DescripcionContrasena =
                    _contrasena.DescripcionContrasena;
                nuevoRespaldoContrasena.PalabraClave = _contrasena.PalabraClave;
                nuevoRespaldoContrasena.IdUsuario = contrasenaIn.IdUsuario;
                _entidadConexionBdd.ContrasenasBackup.
                   Add(nuevoRespaldoContrasena);
            }
            _entidadConexionBdd.SaveChanges();
            resultado = "r1";
        }
        catch (Exception)
        {
            resultado = "r2";
        }
    }
    else
    {
        resultado = "r3";
    }
    return resultado;

}
```


Segmento de código 3.8. Método `EditarContrasena`

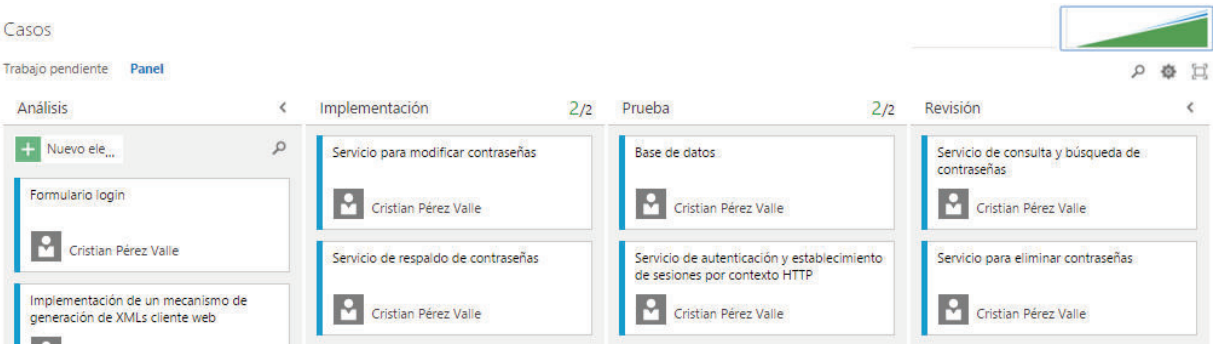


Figura 3.10. Entregable "Servicio para modificar contraseñas" en el paso de proceso "Implementación"

3.3.2.4 Instalación del servicio de gestión de contraseñas

Pasos para instalar el aplicativo:

1. Activar las características de *Internet Information Services* en Windows, verificar que estén seleccionadas la consola de administración IIS y ASP.NET 4.5 o 4.6. como se indica en la Figura 3.11. Verificar que en los servicios avanzados del *framework* esté marcada la Activación HTTP, para que el equipo funcione como Servidor WFC y entienda archivos con extensión `.svc`, como se indica en Figura 3.12.

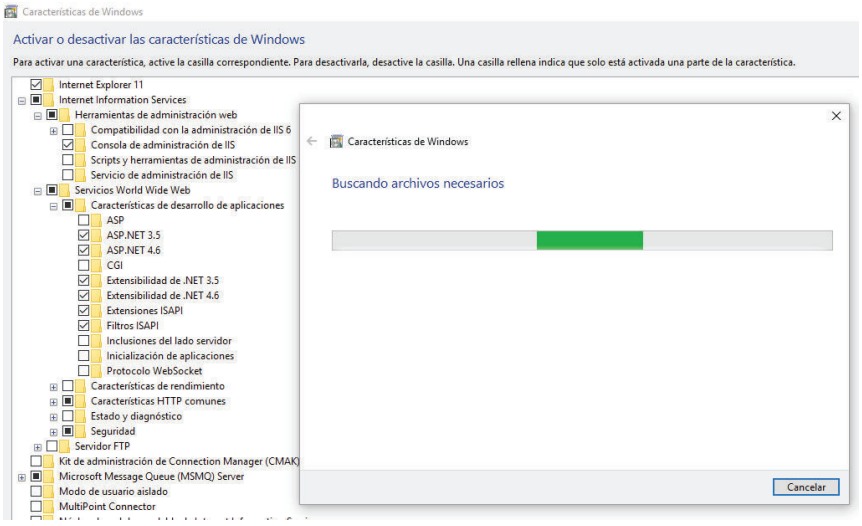


Figura 3.11. Activación del IIS

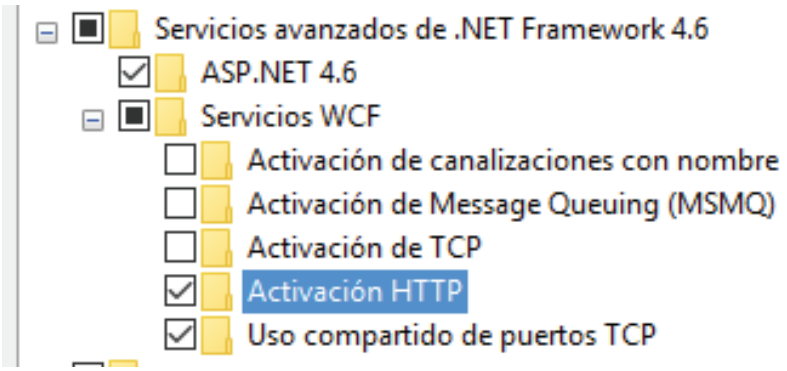


Figura 3.12. Activación del servidor WCF

2. Una vez en la consola de administración crear un sitio web cifrado con el certificado para ambiente de pruebas como se indica en Figura 3.13.

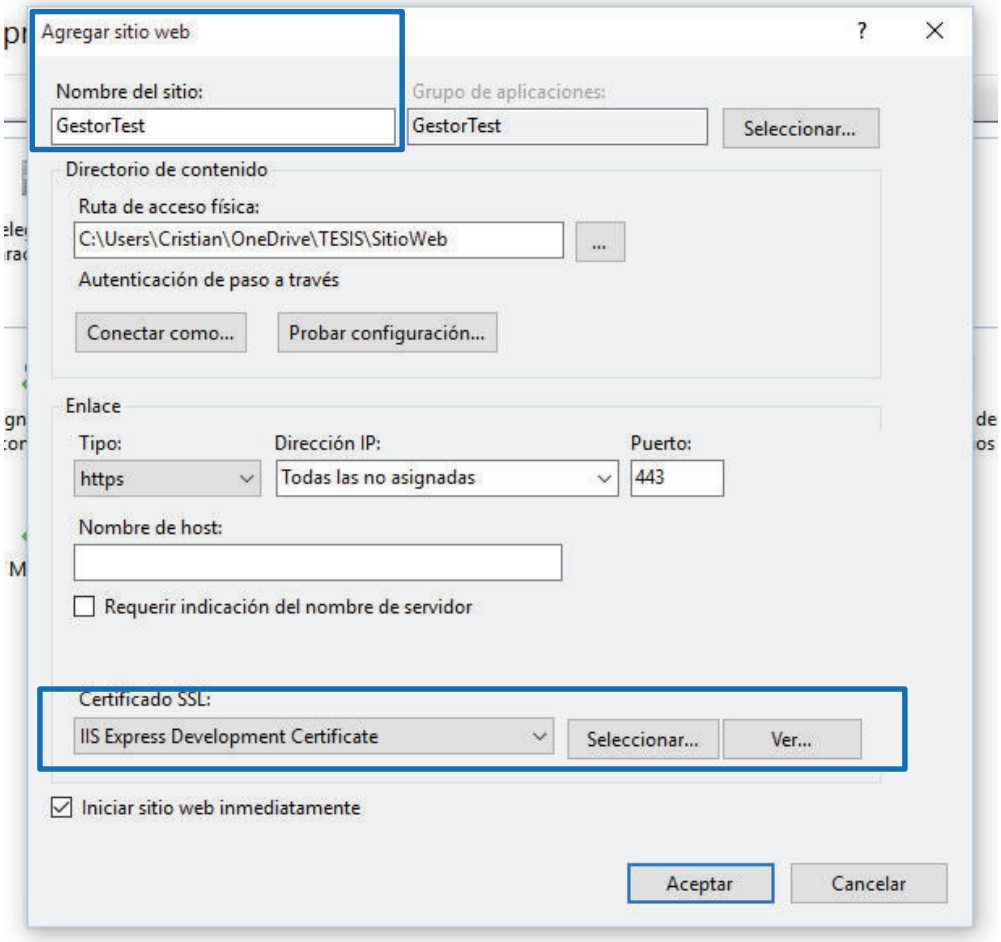


Figura 3.13. Creación de un sitio web con HTTPS en IIS

3. Ubicar en la ruta física del sitio el sistema de archivos publicado desde Visual Studio (carpeta ServiciosGestorContrasenas). Convertir el sistema de archivos en aplicación web con las opciones del Administrador IIS y con grupo de aplicaciones con versión de .NET CLR 4.0 y modo de canalización integrada, como se indica en la Figura 3.14.

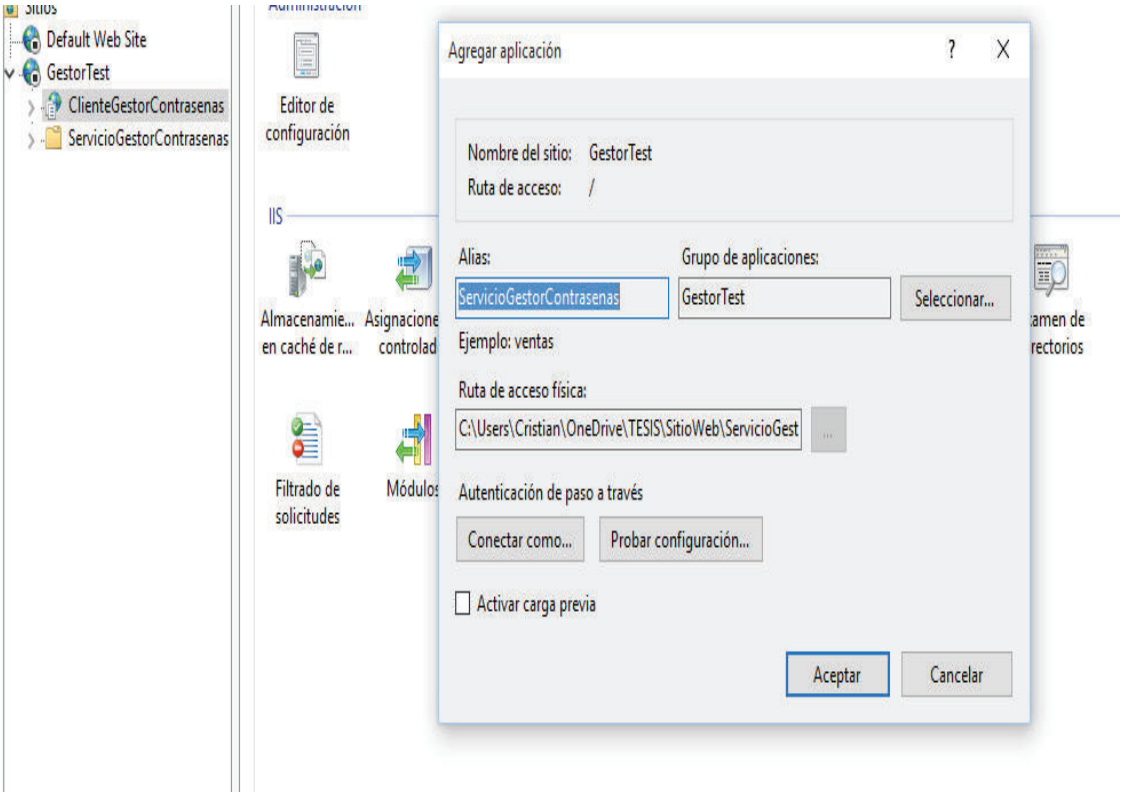


Figura 3.14. Conversión del sistema de archivos publicado en aplicación

4. Proceder con la encriptación de la cadena de conexión a la base datos de desarrollo, pruebas o producción. El proceso de cifrado de la cadena de conexión se presentará en la sección 3.4.1.1.

5. Desde un explorador web se accede a la aplicación publicada en el recurso SVC, si se obtiene el resultado de la Figura 3.15 la instalación del servicio ha sido realizada con éxito.

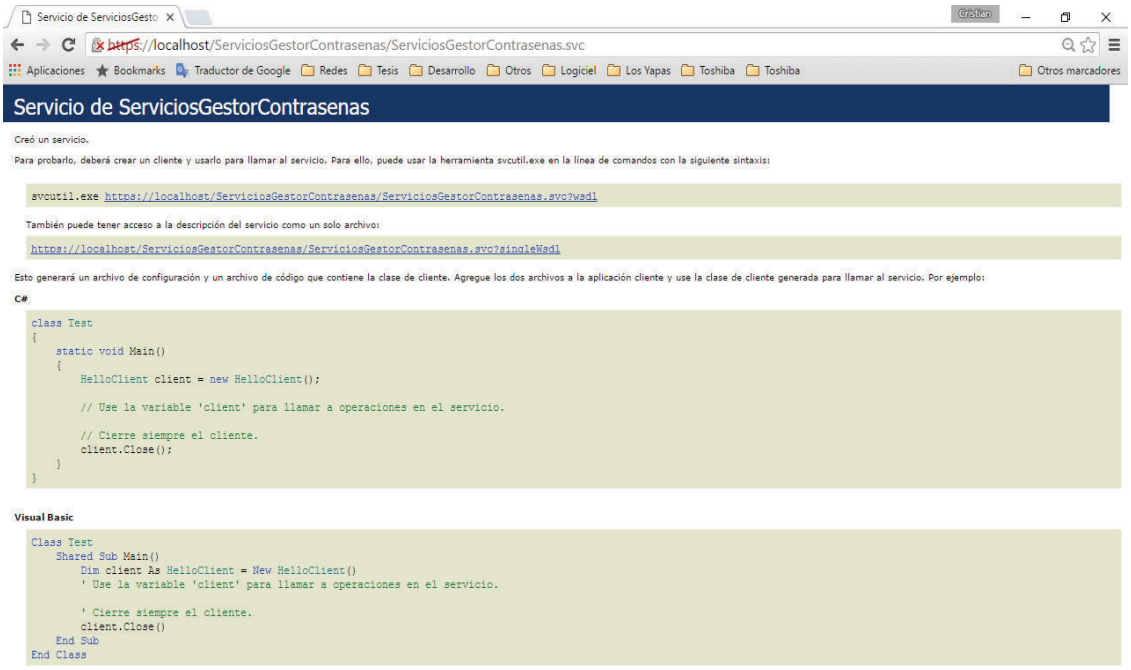


Figura 3.15. Servicio Gestor de Contraseñas publicado exitosamente

3.3.3 CLIENTE WEB

El cliente web es un aplicativo ASP.NET. Las peticiones al servidor se harán de la siguiente manera:

1. Un evento en el formulario accederá al `proxie` para que construya la petición al servicio. El formulario espera que el `proxie` le retorne la respuesta que envía el servidor. Ver el Segmento de código 3.9.

2. El `proxie` configurará la solicitud como se indica en el Segmento de código 3.10.
   2.1. Con la URL del servicio, se configurará un objeto `WebRequest` con información necesaria. Se hará uso del objeto `CookiedRequestFactory`, a través de él se usará un contenedor de cookies en el servidor de aplicaciones cliente como se observa en el

Segmento de código 3.11. El almacén de cookies relaciona al usuario con la cookie guardada, para construir las solicitudes usando la cookie de cada usuario. Si no existe una cookie, se construye y almacena una nueva.

```
if (txtContrasena.Text == txtRepetirContrasena.Text && txtCorreoElectronico.Text ==
txtRepetirCorreoElectronico.Text)
            {
                try
                {
                    String status =
                           ServiciosUsuarioProxy.AgregarUsuario(txtNombreUsuaro.Text
                    ,
                                txtNombres.Text, txtApellidos.Text,
                                       txtCorreoElectronico.Text,
                                txtContrasena.Text);
                    if (status == "r1")
                    {
                        Session["s_usuario"] = txtNombreUsuario.Text;
                        FormsAuthentication.SetAuthCookie(txtUsuario.Text, false);
                        Response.Redirect("~/Interfaz/wfmPrincipal.aspx");
                    }
                    else
                    {
                        lblMensajeRegistro.Text = status;


                    }
                }
                catch (Exception ex)
                {
                    lblMensajeRegistro.Text = ex.Message;

                 }
             }
```

Segmento de código 3.9. Petición `proxie` para interactuar con el servicio

2.2. Se estructurará una cadena de bytes que contienen el tipo de dato que espera el servicio, con los argumentos necesarios para que el servidor ejecute la solicitud. Se implementará en cada `proxie` un método que estructurará con formato XML la información que se le ingrese. Este método se encuentra en el Segmento de código 3.12.

```
public static String AgregarUsuario(string usuario, string nombres...)
        {
            try
            {
                //Configuración de la petición
                if (UsuarioUrl != null)
                {
                    _webRequest =
                           CookiedRequestFactory.CreateHttpWebRequest(UsuarioUrl,
                                  usuario);
                    _webRequest.Method = "POST";
                    _webRequest.ContentType = "application/xml; charset=utf-8";
                    _webRequest.Timeout = 30000;
                }

                //Carga de la estructura del requisito a solicitar
                var streamReaderRequest = new StreamReader(UsuarioStream(usuario,
                    nombres, apellidos correoElectronico, contrasena, null, "s2"));
                string requestString = streamReaderRequest.ReadToEnd();
                var sw = new StreamWriter(_webRequest.GetRequestStream());
                sw.Write(requestString);
                sw.Close();

                //Obtención de una respuesta del servidor
                _webResponse = _webRequest.GetResponse();
                Stream responseStream = _webResponse.GetResponseStream();
                if (responseStream != null)
                {
                    var streamReaderResponse = new StreamReader(responseStream);
                    string responseString = streamReaderResponse.ReadToEnd();

                    //Chequeo de respuesta para control de acceso al aplicativo web
                    switch (responseString)
                    {
                        case "<string>r1</string>":
                          //Programación de lógica a devolver con la variable estado
                            break;
                        case "<string>r2</string>":
                           //Programación de lógica a devolver con la variable estado
                            break;
                        case "<string>r3</string>":
                          //Programación de lógica a devolver con la variable estado
                            break;
                        default:
                          //Programación de lógica a devolver con la variable estado
                            break;
                    }
                }
            }
            catch (Exception ex)
            {
                estado = ex.Message;
            }

            return estado;
        }
```

Segmento de código 3.10. Ejecución de una petición al servicio

2.3. Cuando el objeto `WebRequest` contenga las configuraciones y los datos estructurados en XML, se instanciará un objeto `WebResponce` con el `WebRequest`. Se ejecutará la petición y el *stream* de bytes que se retornan serán evaluados para determinar el resultado a devolver.

```
using System.Net;
using System.Collections.Generic;

namespace GestorServiciosLibrary.Utilidades
{
    public class CookiedRequestFactory
    {
        // This dictionary keeps all the cookie containers for
        // each domain.
        private static readonly Dictionary<string, CookieContainer> Containers
            = new Dictionary<string, CookieContainer>();

        /// <param name="url">uri de la petición a crear</param>
        /// <param name="validador">dato a contener en el contexto de la petición
http
        /// </param>
        /// <returns></returns>
        public static HttpWebRequest CreateHttpWebRequest(string url,
             string validador)
        {
            // Create a HttpWebRequest object
            var request = (HttpWebRequest)WebRequest.Create(url);

            // try to get a container from the dictionary, if it is in the
            // dictionary, use it. Otherwise, create a new one and put it
            // into the dictionary and use it.
            CookieContainer container;
            if (!Containers.TryGetValue(validador, out container))
            {
                container = new CookieContainer();
                Containers[validador] = container;
            }

            // Assign the cookie container to the HttpWebRequest object
            request.CookieContainer = container;

            return request;
        }
    }
}
```

Segmento de código 3.11. Clase `CookiedRequestFactory` que implementa una solicitud HTTP que incluye en su cabecera una cookie almacenada en memoria

```
        /// <summary>
        /// Generador de la estructura para solicitudes de entidades usuario
        /// </summary>
        /// <param name="nombresIn">Nombres del usuario</param>
        /// <param name="apellidosIn">Apellidos del usuario</param>
        /// <param name="correoElectronicoIn">Correo Electronico</param>
        /// <param name="usuarioIn">Usuario</param>
        /// <param name="contrasenaIn">Contraseña</param>
        /// <param name="contrasenaNuevaIn">Variable cambio de contraseña </param>
        /// <param name="idIn">Identificador de solicitud a realizar</param>
        /// <returns>Bytes con la estructura para solicitudes a la entidad
             usuario</returns>
        private static Stream UsuarioStream(string usuarioIn, string nombresIn...)
        {
            // Buffer que almacenará los bytes con la estructura deseada
            var buffer = new MemoryStream();

            //Variable que ayudará a escribir una estructura XML
            var escribirXml = new XmlTextWriter(buffer, Encoding.UTF8);

            escribirXml.WriteStartDocument(false);
            escribirXml.WriteStartElement("ObjetoUsuario");
            escribirXml.WriteAttributeString("xmlns",
                "http//localhost:5310/gestorContrasenas");
            escribirXml.WriteElementString("Apellidos", apellidosIn);
            escribirXml.WriteElementString("ContrasenaMaestra", contrasenaIn);
            escribirXml.WriteElementString("ContrasenaNueva", contrasenaNuevaIn);
            escribirXml.WriteElementString("CorreoElectronico",
                correoElectronicoIn);
            escribirXml.WriteElementString("Estado", "true");
            escribirXml.WriteElementString("Id", idIn);
            escribirXml.WriteElementString("Nombres", nombresIn);
            escribirXml.WriteElementString("Usuario", usuarioIn);

            escribirXml.WriteEndDocument();
            escribirXml.Flush();

            var sw = new StreamWriter(buffer);
            sw.Flush();
            buffer.Position = 0;

            return buffer;
        }
```

Segmento de código 3.12. Construcción de la estructura XML a serializar

Las URL de los servicios publicados son leídas en la creación de la solicitud HTTP en el `web.config` del aplicativo cliente. Las variables a leer en el archivo de configuración son las que se muestran en el Segmento de código 3.13.

```
<appSettings>
    <add key="UsuarioUrl"
      value="https://localhost:44300/ServiciosGestorContrasenas.svc/solicitudUsuari
o"/>
    <add key="GruposSolicitudUrl"
      value="https://localhost:44300/ServiciosGestorContrasenas.svc/solicitudGrupo"
/>
    <add key="GruposConsultaUrl"
      value="https://localhost:44300/ServiciosGestorContrasenas.svc/consultaGrupos"
/>
    <add key="ContrasenaSolicitudUrl"
      value="https://localhost:44300/ServiciosGestorContrasenas.svc/solicitudContra
sena"/>
    <add key="ContrasenaConsultaUrl"
      value="https://localhost:44300/ServiciosGestorContrasenas.svc/consultaContras
enas"/>
    <add key="ObtenerInfoUsuarioUrl"
      value="https://localhost:44300/ServiciosGestorContrasenas.svc/infoUser?userna
me="/>
    <add key="CerrarSesion"
      value="https://localhost:44300/ServiciosGestorContrasenas.svc/cerrarSesion"/>
  </appSettings>
```

Segmento de código 3.13. Claves con las URLs sin cifrar apuntando a un servicio

Una vez ejecutadas las pruebas se procede a cerrar la totalidad de entregables del desarrollo del cliente web como indica la metodología seguida. Estos se pueden observar en la Figura 3.16.

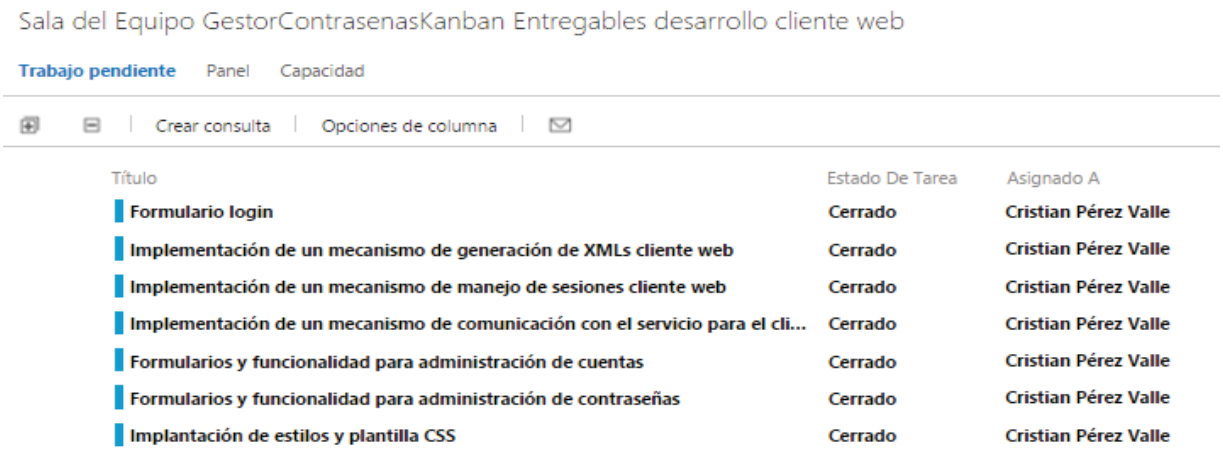


Figura 3.16. Tarjetas de los entregables del desarrollo del cliente web cerradas

3.3.4 CLIENTE ANDROID

El ADT Bundle de Windows para desarrollo con Android disponía del Android Developer Tools Pakage, el Android SDK Manager, el Android Virtual Device Manager, todos en versión 23.0.2, y el IDE Eclipse Juno 4.2.2. Otros componentes del ADT eran la interfaz LogCat y la consola Dalvik Debug Monitor Server (DDMS) que durante la depuración del aplicativo registraba la información de lo que sucedía en el dispositivo de simulación y el desempeño de la máquina virtual para la plataforma Android, que se denominaba Dalvik. Ahora este paquete de programas tiene un IDE propio conocido como Android Estudio que está en la versión 1.5.1. El soporte para las Android Development Tools ha finalizado. Para recibir ayuda sobre el cambio de IDE, se puede consultar el sitio oficial de Andriod Studio, donde existen publicaciones dedicadas a la migración.

Dalvik es la máquina virtual del entorno Android, los paquetes del *plugin* para Android Studio se encargan de la generación del Aplication Package File (APK) transformando programación Java en una compilación Android. Un APK es un instalable Android. Durante la instalación el sistema operativo lee el *manifest* o `AndroidManifest`, que es un archivo XML que le indica al sistema operativo los permisos que necesita para un correcto despliegue. También incluye información del sobre la versión del API (Interfaz de programación de aplicaciones) de desarrollo y la versión del API de compatibilidad mínima, y otra información necesaria para la ejecución de la aplicación. En el prototipo la versión del API de desarrollo Android será la 21 y la versión 8 será mínima compatible.

Si al ejecutarse un aplicativo migrado a Android Studio, se lanza el error "*duplicate files during packaging of apk"*, que se refiere a problemas con las librerías de terceros, y que para este aplicativo serán las librerías que facilitan la generación de solicitudes HTTP; se deberá ingresar las exclusiones que se indican en el Segmento de código 3.14 acerca de la generación de metadatos en el archivo `grandel.build`, dentro del de parámetro Andriod.

Con el software Android Virtual Device Manager se configurará un dispositivo una imagen de Android 4.1.2 Jelly Bean API 16.

```
packagingOptions {
    exclude 'META-INF/LICENSE'
    exclude 'META-INF/DEPENDENCIES'
    exclude 'META-INF/NOTICE'
}
```

Segmento de código 3.14. Exclusiones en la generación de metadatos

3.3.4.1 Consumo del servicio

Existirán dos paquetes, uno para la programación de negocio del aplicativo `epn.redes.droidpassmanager` y otro para gestionar la interacción con el servicio `epn.redes.proxies`. Un nuevo proyecto vacío sin programaciones preinstaladas, se presenta como en la Figura 3.17.

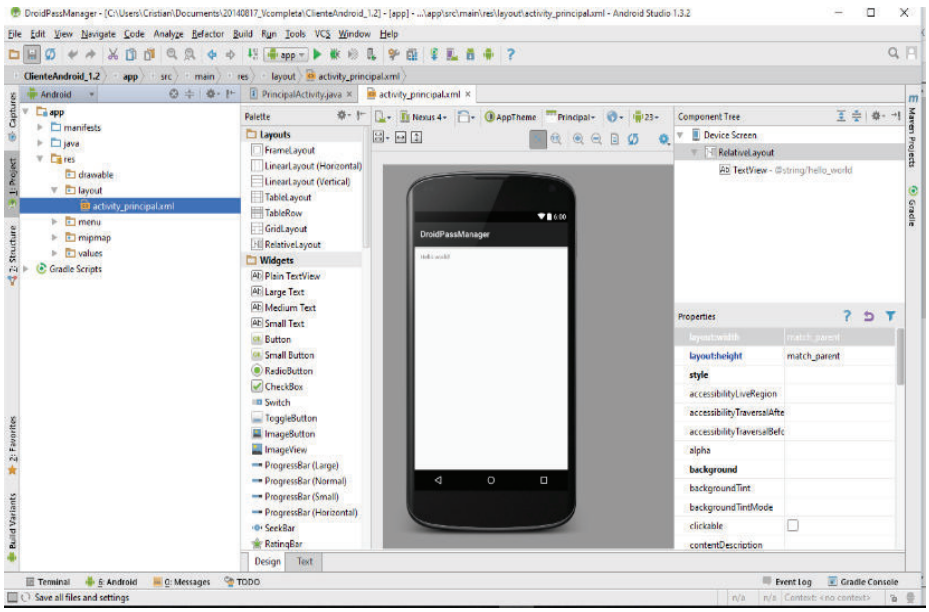


Figura 3.17. Layout de la actividad principal por defecto para un proyecto vacio

Cada pantalla está definida por un `layout` que es un archivo XML donde se van definiendo los controles de cada vista a presentarse. Cada layout es un

ActivityView, que se definieron en la sección 1.7.4. Los ActivityViews y los menús se estructuran en archivos XML.

```
btnAceptar.setOnClickListener(new OnClickListener(){
   @Override
   public void onClick(View v){
      String contrasena = inputContrasena.getText().toString();
      String descripcion =
               inputDescContrasena.getText().toString();
      String palabra = inputPalabraClave.getText().toString();
      Grupo selectedItem = (Grupo) sprGrupos.getSelectedItem();
      String idGrupo = selectedItem.getIdGrupo();

      String resultado =
               proxyContrasena.IngresarContrasena(idGrupo,
                        contrasena, palabra, descripcion);

      if(resultado != ""){
         switch(resultado)
         {
         case "r1":
            display.setText("El grupo ha sido ingresado con
                              éxito.");
            break;
         case "r0":
            pasarCaducidad();
            break;
         default:
            display.setText("Lo sentimos la contraseña no fue
                        registrada, por favor vuelva a intentar.");
            break;
         }
      }
   }
});
```

Segmento de código 3.15. Llamada a un método del proxie en función del evento

Se seguirá una llamada al servicio desde el cliente Andriod para el ingreso de una contraseña al sistema.

Primero, el control del ActivityView estará escuchando por una acción específica del usuario, en general un clic. En el evento se capturará el estado de las variables del ActivityView con el fin de pasar los valores que tenga al método del proxie para

construir la solicitud al servicio. Según la respuesta que obtenga, devolverá el resultado al usuario.

En el Segmento de código 3.15 el `proxie` a utilizar es el de gestión de contraseñas, el método a acceder es `IngresarContrasena()` que se muestra en el Segmento de código 3.16.

`IngresarContrasena()` utilizará el método que estructura los argumentos recibidos en una cadena con formato XML.

El método que genera el XML es `CrearXMLContrasena()` que se lo puede observar en el Segmento de código 3.17.

```
public String IngresarContrasena(String idGrupo,
                String contrasena, String palabraClave,
                        String descripcionContrasena) {
   String resultado = "";
   String xml = CrearXMLContrasena(idGrupo, "", contrasena,
         palabraClave, descripcionContrasena, "s1", "");

   try {
      String result = new EjecutorAsincronoPost()
            .execute(
                  "https://"+ HttpRequest.getIp()+
"/ServiciosGestorContrasenas/ServiciosGestorContrasenas.svc/sol
icitudContrasena",xml).get();

      resultado = XmlParser.DevolverTexto(result);

   } catch (InterruptedException e) {
      // TODO Auto-generated catch block
      resultado = e.getMessage();
   } catch (ExecutionException e) {
      // TODO Auto-generated catch block/
      resultado = e.getMessage();
   }

   return resultado;
}
```

Segmento de código 3.16. Método `IngresarContrasena`

```
private String CrearXMLContrasena(String idGrupoIn...) {
   StringWriter writer = new StringWriter();
   String xmlRetorno = "";
   XmlSerializer serializer = Xml.newSerializer();

   try {
      serializer.setOutput(writer);
      serializer.startDocument("UTF-8", false);
      serializer.startTag(null, "ObjetoContrasena");
      serializer.attribute(null, "xmlns",
            "http//localhost:5310/gestorContrasenas");

      serializer.startTag(null, "Contrasena");
      serializer.text(contrasenaIn);
      serializer.endTag(null, "Contrasena");

      serializer.startTag(null, "DescripcionContrasena");
      serializer.text(descripcionContrasenaIn);
      serializer.endTag(null, "DescripcionContrasena");

      serializer.startTag(null, "Id");
      serializer.text(idIn);
      serializer.endTag(null, "Id");

      serializer.startTag(null, "IdContrasena");
      serializer.text(idContrasenaIn);
      serializer.endTag(null, "IdContrasena");

      serializer.startTag(null, "IdGrupo");
      serializer.text(idGrupoIn);
      serializer.endTag(null, "IdGrupo");

      serializer.startTag(null, "IdUsuario");
      serializer.text(idUsuarioIn);
      serializer.endTag(null, "IdUsuario");

      serializer.startTag(null, "PalabraClave");
      serializer.text(palabraClaveIn);
      serializer.endTag(null, "PalabraClave");

      serializer.endTag(null, "ObjetoContrasena");
      serializer.endDocument();
      serializer.flush();

      xmlRetorno = writer.toString();
   } catch (IllegalArgumentException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
   } catch (IllegalStateException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
   } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
   }
   return xmlRetorno;
}
```

Segmento de código 3.17. Método que devuelve una cadena con estructura XML

Se instancia un objeto del tipo `EjecutorAsincronoPost` como se indica en Segmento de código 3.18, esta clase es una extensión del tipo `AsyncTask`[1]. La función de ese objeto es que, durante la llamada al servicio y espera por la respuesta, el dispositivo se quede inoperable y pueda sincronizar otras solicitudes. Cada solicitud es manejada por una instancia del `EjecutorAsincronoPost` que se encarga de que la respuesta se envíe a la `Activity` de manera fluida, sin interrupciones, y sin la necesidad de programarse hilos al pasar los datos. La información que se pasará, son la URL de servicio y la cadena XML a armar. En segundo plano se contactará al servicio y se devolverá al `proxie` el resultado.

```
package epn.redes.proxies;

import android.os.AsyncTask;

public class EjecutorAsincronoPost extends AsyncTask<String, Void,
String> {

   HttpRequest solicitud = new HttpRequest();

   @Override
   protected String doInBackground(String... params) {
      String resultado;
      resultado = solicitud.sendPost(params[0], params[1]);
      return resultado;
   }
}
```

Segmento de código 3.18. Clase que hereda del tipo AsyncTask la funcionalidad de manipular el hilo principal

Para construir la solicitud se usa el objeto `HTTPRequest` del paquete `org.apache.http` (librería de terceros) que es similar al `HTTPRequest` utilizado en el cliente ASPX. De este objeto se usa el método `SendPost()` que se puede observar en el Segmento de código 3.19. Este método arma la solicitud HTTP utilizando POST

[1] ***ASYNCTASK***: Es una clase del API Android que facilita la manipulación del hilo principal de la aplicación o `UI Thread`, que permite el uso de operaciones en segundo plano sin la necesidad de manejar directamente otros hilos.

y verificando si se tiene almacenada localmente una cookie con el identificador del `HTTPContext`, para manejar la sesión de usuario en el servidor.

```
public String sendPost(String url, String data) {
   ret = null;

   httpPost = new HttpPost(url);
   response = null;

   StringEntity tmp = null;

   try {
      tmp = new StringEntity(data, HTTP.UTF_8);
      tmp.setContentType("text/xml");
   } catch (UnsupportedEncodingException e) {
      Log.e("GesConClient", "HttpUtils :
                UnsupportedEncodingException : "
            + e);
   }

   httpPost.setEntity(tmp);

    localContext  = new BasicHttpContext();
   CookieStore cookieStore = new BasicCookieStore();
   cookieStore = CookieContainer.DevolverCookie();
    localContext.setAttribute(ClientContext.COOKIE_STORE,
cookieStore);

   Log.d("GesConClient", url + "--" + data);

   try {
      response = httpClient.execute(httpPost, localContext);

      if(cookieStore == null)

CookieContainer.AlmacenarCookie(httpClient.getCookieStore());

      if (response != null) {
         ret = EntityUtils.toString(response.getEntity());
      }
   } catch (Exception e) {
      Log.e("GesConClient", "HttpUtils: " + e.getMessage());
      ret = e.getMessage();
   }

   Log.d("GesConClient", "Returning value:" + ret);

   return ret;
}
```

Segmento de código 3.19. Método que se encarga de armar la solitud HTTP y recibir la respuesta del servidor

El aplicativo cliente debe almacenar en memoria la cookie de sesión, ya que el `EjecutorAsincronoPost` se destruye una vez se obtenga la respuesta del servidor, y para otra solicitud se la debe cargar nuevamente.

En el prototipo se usa una variable estática de la clase `CookieContainer` como se indica en el Segmento de código 3.20. Cuando se recibe la primera respuesta del servidor que contiene la cookie, se usa el método `AlmacenarCookie()`. Luego las siguientes peticiones hacen uso del método `DevolverCookie()` para leer la cookie de este contenedor y pasarla al servicio en la cabecera de la solicitud HTTP.

```
package epn.redes.proxies;

import org.apache.http.client.CookieStore;

public class CookieContainer {

   private static CookieStore cookieStore;

   public static void AlmacenarCookie(CookieStore
cookieStoreIn)
   {
      cookieStore = cookieStoreIn;
   }

   public static CookieStore DevolverCookie()
   {
      return cookieStore;
   }
}
```

Segmento de código 3.20. Clase que define un objeto encarga de almacenar el valor de una variable estática del tipo `CookieStore`

Durante la ejecución del método `SendPost()` se recibe la respuesta, este la entrega al `EjecutorAsincronoPost` para pasarlo al `proxie`.

Con el método estático `DevolverTexto()` de la clase `XMLParser` se extrae el resultado como cadena de texto, como se indica en el Segmento de código 3.21.

```
public static String DevolverTexto(String entrada) {

   String resultado;
   // Parse XML
   XmlPullParserFactory pullParserFactory;

   try {
      // convert String into InputStream
      InputStream stream = new
ByteArrayInputStream(entrada.getBytes());

      pullParserFactory = XmlPullParserFactory.newInstance();
      XmlPullParser parser = pullParserFactory.newPullParser();


parser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES,
false);
      parser.setInput(stream, null);
      resultado = parseXmlTexto(parser);
   } catch (XmlPullParserException e) {
      resultado = e.getMessage();
   } catch (IOException e) {
      resultado = e.getMessage();
   }

   return resultado;
}
```

Segmento de código 3.21. Método que se encarga de extraer el texto de la etiqueta XML String

Una vez ejecutadas las pruebas se procede a cerrar la totalidad de entregables del desarrollo del cliente Android como indica la metodología seguida. Estos se pueden observar en la Figura 3.18.

3.4 HARDENING DEL PROTOTIPO

3.4.1 ESPECIFICACIONES DE SEGURIDAD A NIVEL DE SERVIDOR

3.4.1.1 Cifrado de la Cadena de Conexión

Una vez publicado el servicio, se identifica el ID del sitio publicado, para lo que se accede al Administrador IIS y sobre la carpeta sitios se obtiene la información de la Figura 3.19.

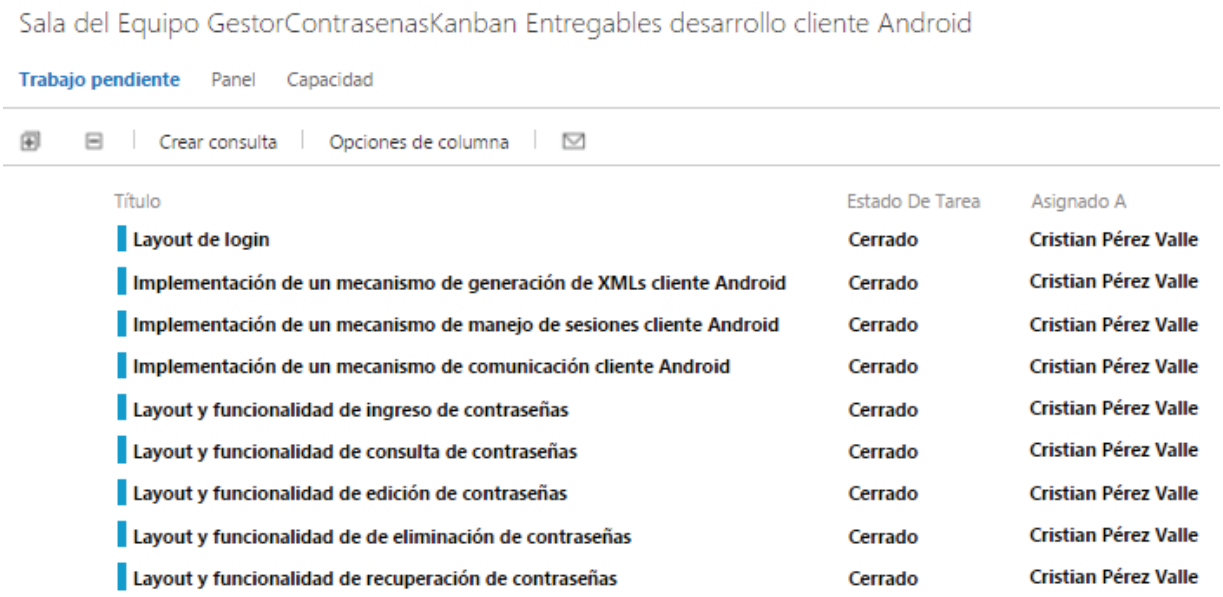


Figura 3.18. Tarjetas cerradas de los entregables del desarrollo del cliente Android

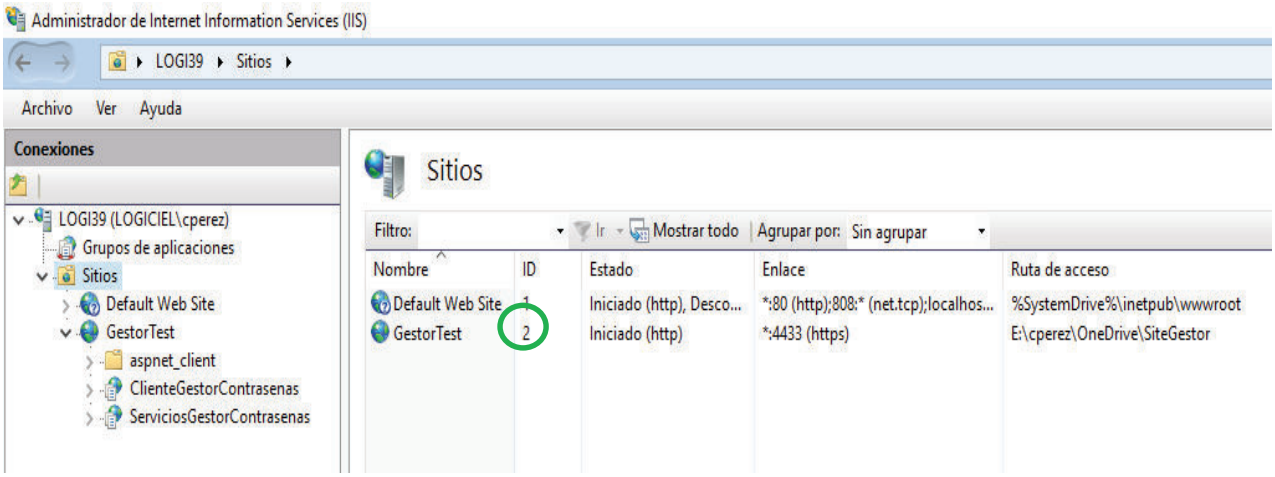


Figura 3.19. Identificador del sitio en visto desde el administrador IIS

En la Figura 3.20 se muestra la cadena a cifrar en el archivo de configuración. El objetivo es proteger las credenciales de acceso a la base de datos.

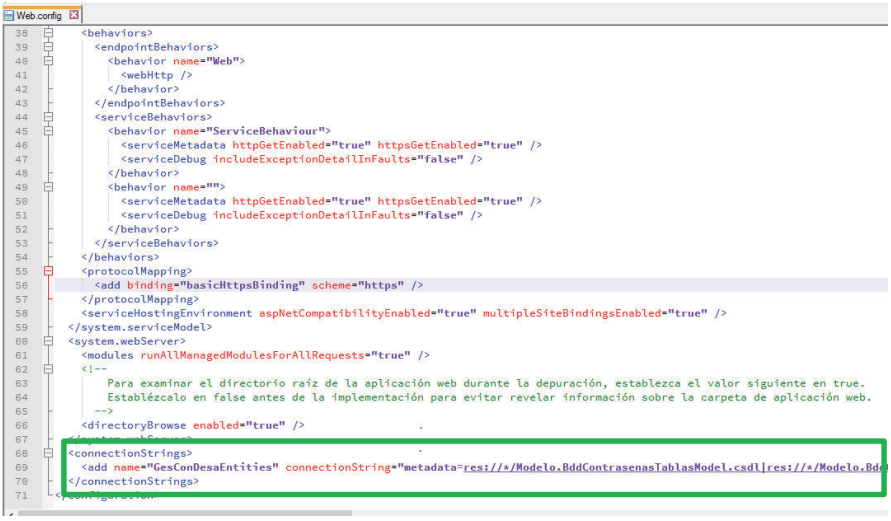


Figura 3.20. Cadena de conexión sin cifrar en el `web.config`

Con el ID de sitio se hace uso de la instrucción *`aspnet_regis`* para lo que se accede al directorio "`C:\Windows\Microsoft.NET\Framework64\v4.0.30319\`" que es la ruta donde está instalado el *framework*.

Se ejecuta el comando de la Figura 3.21:

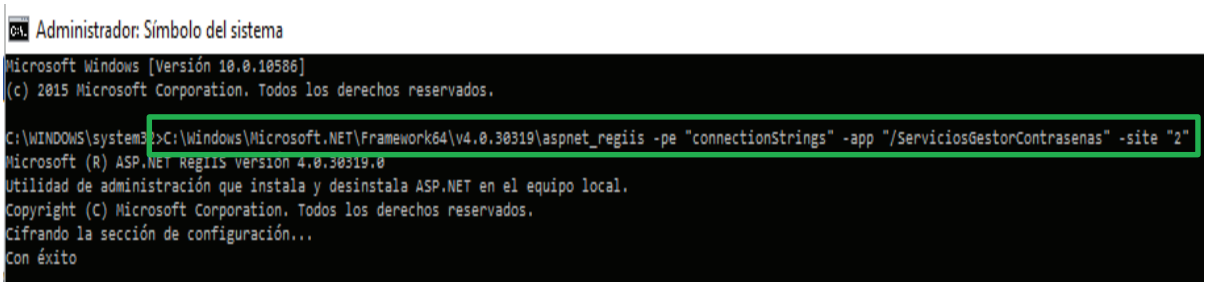


Figura 3.21. Ejecución del comando para cifrar configuraciones web con `aspnet_regiis`

La opción `-pe` le indica a la instrucción que a continuación se le pasará la etiqueta del archivo `web.config` que se requiere cifrar que para este caso es la cadena de conexión a la base de datos `connectionStrings`.

La opción `-app` le indica a la instrucción de qué aplicativo web se accederá a su archivo `web.config` para cifrar la etiqueta, que para este caso es la aplicación web `ServiciosGestorContrasenas`.

La opción `-site` le indica a la instrucción el sitio web donde está publicada la aplicación por medio de su identificador que en este caso `2`.

Se puede mandar niveles en la etiqueta para cifrar cualquier configuración sensible como son claves (*keys*), valores de variables de sesión, y URLs del servicio. Como ejemplo se tiene el Segmento de código 3.22, donde en la etiqueta `system` se quiere cifrar la etiqueta `machineKey`. Más información en [12].

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\aspnet_regiis -pe
"system.web/machineKey" -app "/ServiciosGestorContrasenas" -site "2"
```

Segmento de código 3.22. Comando para cifrar configuraciones web

3.4.1.1.1 Prueba de cifrado

En la Figura 3.22 se observa que la etiqueta `connectionStrings` dejó de ser solo una línea de configuraciones como se indicó en la Figura 3.20, si no que ahora tiene información relacionada con su proceso de cifrado. La cadena cifrada sería la marcada en el cuadro azul.

3.4.1.2 Establecimiento de canal seguro

Se instalará un certificado digital en el servidor web para crear un sitio web que escuche usando HTTPS. En este sitio se publicará el servicio y el aplicativo web. El certificado digital será firmado localmente, no se usará un certificado emitido por una entidad certificadora. Con el certificado se obtiene SSL sobre HTTP (HTTPS) para asegurar el transporte de datos.

HTTPS lee un *token*[1] sobre un canal no seguro con el objetivo de protegerlo. Para probar la funcionalidad del servicio con HTTPS en fase de desarrollo, en las propiedades del proyecto hay que marcar el atributo `SSL Habilitado` en *true* y modificar la URL del proyecto para IIS Express, como se muestra en la Figura 3.23.

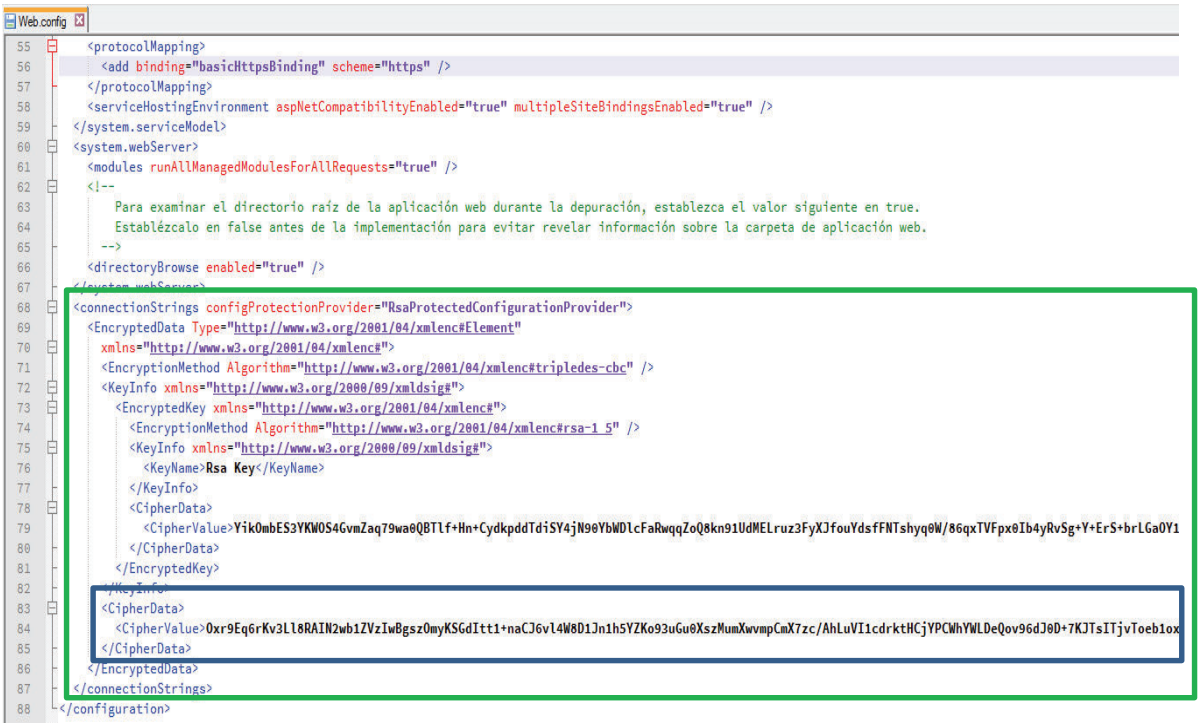


Figura 3.22. Cadena de conexión cifrada en el `web.config`

3.4.1.2.1 Prueba de la publicación en canal seguro

En la Figura 3.24 y la Figura 3.25 se observa la publicación utilizando un canal seguro en IIS Express mediante la ejecución de la solución en Visual Studio.

En Visual Studio 2012 se necesita correr la actualización KB3002339 para corregir un problema que impide crear una aplicación web vacía o una aplicación de formularios web de ASP.NET tras instalar Microsoft .NET Framework 4.5.3 o superiores.

[1] ***TOKEN***: Es una cadena de caracteres con un significado especifico según su programación.

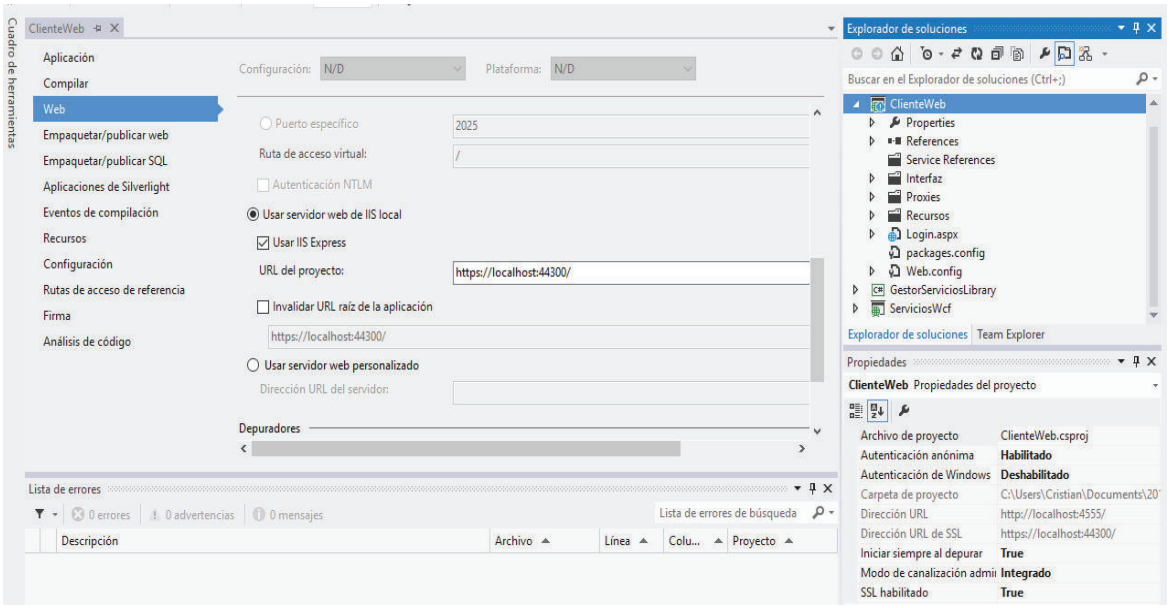


Figura 3.23. Publicación en desarrollo con SSL habilitado

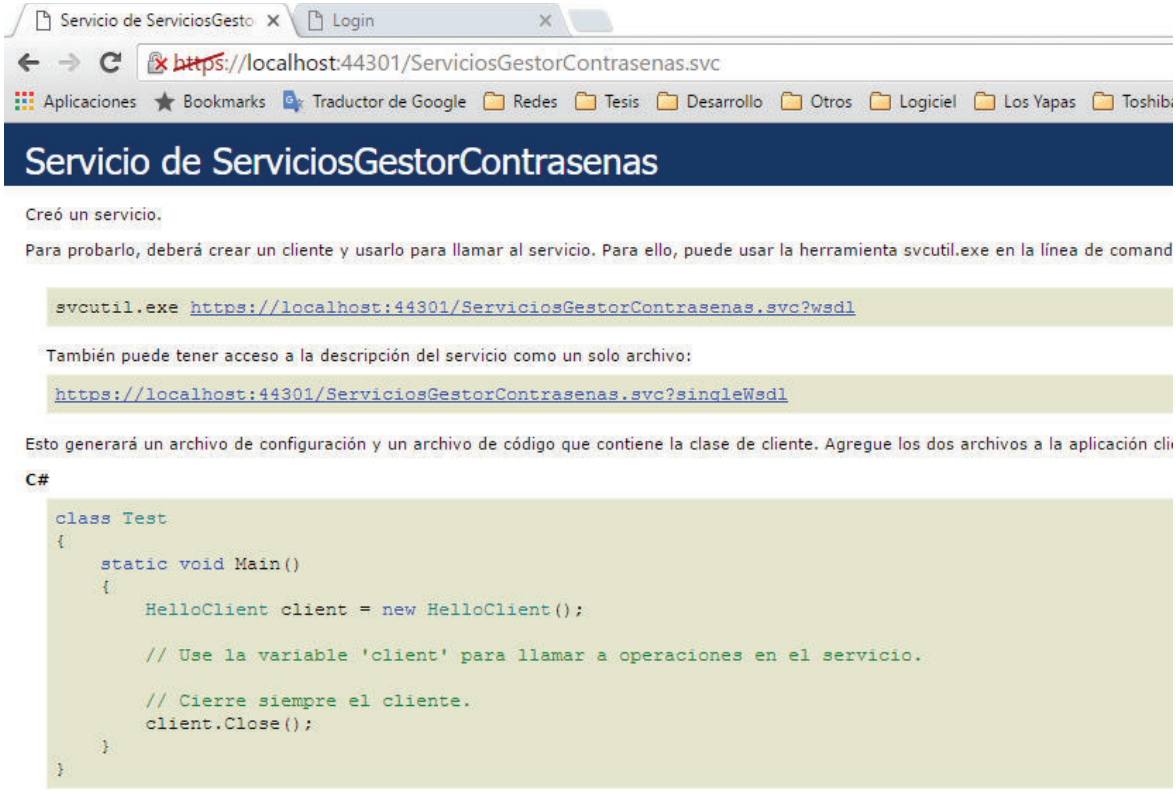


Figura 3.24. Publicación con SSL en IIS Express del servicio

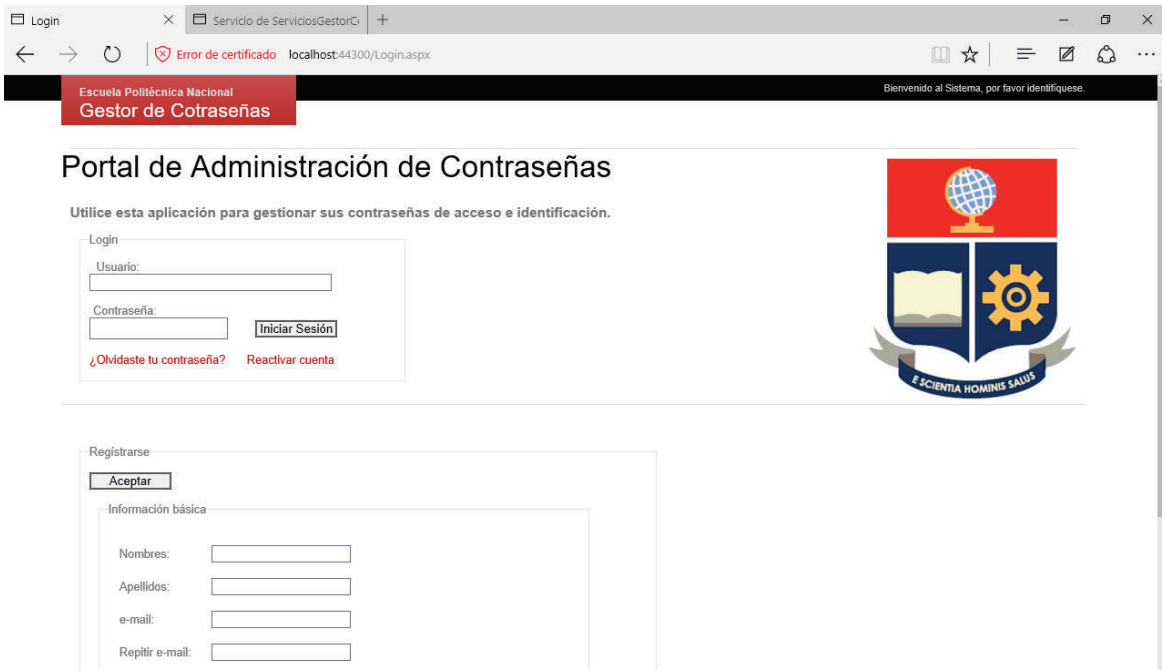


Figura 3.25. Publicación con SSL en IIS Express del cliente

3.4.1.3 Otros procedimientos de seguridad a nivel del servidor

- Actualización de parches de seguridad del sistema operativo del servidor.
- Seguridades físicas del centro de datos donde se ubica el servidor de base de datos y el servidor de aplicaciones.
- Controlar que el equipo tenga abajo los servicios que no son necesarios para la ejecución del prototipo.

3.4.2 ESPECIFICACIONES DE SEGURIDAD A NIVEL DE SERVICIO

3.4.2.1 Establecimiento de sesión y *timeout*

Los métodos relacionados a la gestión de usuarios están publicados mediante el servicio `ContruirSolicitudUsuario`. El método `Autenticar`() que se muestra en el Segmento de código 3.23, establece una sesión de usuario mediante el mantenimiento del contexto HTTP, al cual se le da un *timeout* que es leído desde el archivo de configuración del servicio. Las solicitudes que lleguen con el identificador

del contexto HTTP para ser legitimar el usuario solo podrán hacerlo mientras el servidor guarde esta información que está determinada por el *timeout*.

```
        /// <summary>
        /// Autentica un usuario en el sistema
        /// </summary>
        /// <param name="usuarioIn">usuario a autenticar</param>
        /// <returns>resultado de la auntenticación</returns>
        private String Autenticar(ObjetoUsuario usuarioIn)
        {
            string resultado;

            try
            {
                _usuario = _entidadConexionBdd.Usuarios.
                    FirstOrDefault(u => u.Usuario == usuarioIn.Usuario);
                if (_usuario != null && _usuario.ContrasenaMaestra ==
                    EncriptarContrasena(usuarioIn.ContrasenaMaestra) &&
                           _usuario.Estado)
                {
                    resultado = "r1";
                    _context.Session["USERLOGGEDIN"] = "YES";
                    _context.Session.Timeout =
                           Convert.ToInt32(ConfigurationManager.
                                  AppSettings["timeOut"]);
                }
                else if (_usuario != null && _usuario.Estado)
                    resultado = "r2";
                else
                    resultado = "r3";
            }
            catch (Exception)
            {
                resultado = "r7";
            }

            return resultado;

        }
```

Segmento de código 3.23. Método `Auntenticar()`

Como el servicio es sin estado, se construye una nueva instancia del objeto en por cada interacción HTTP con el tiempo de vida determinado por el valor del *timeout*, como se indica en el Segmento de código 3.24, de modo que, si el cliente registra en la cabecera de la solicitud HTTP siempre el mismo contexto con su respectivo identificador, esta solicitud pertenecerá siempre al mismo contexto HTTP y por tanto siempre a la misma instancia del servicio. Pero además es necesario grabar un valor

en este contexto que durará mientras no se destruya la instancia de servicio ya que con este valor se validará que el usuario ha pasado por la autenticación en el servicio, es decir en cualquier otra instancia no autenticada del servicio que solicite otro método que no sea autenticar, no podrán utilizar el servicio. Lo métodos utilizan la siguiente cadena: `context.Session["USERLOGGEDIN"]=="YES"`, para permitir ser accedidos.

```
public ServiciosGestorContrasenas()
        {
            _context = HttpContext.Current;
        }
```

Segmento de código 3.24. Obtención del contexto HTTP

El método `CerrarSesion()` se encarga de destruir los valores que relacionan las solicitudes y respuestas finalizando la sesión de usuario, como se observa en el Segmento de código 3.25. `CerrarSesion()` utiliza el método GET de HTTP.

```
public string CerrarSesion()
{
    string retorno;
    try
    {
        // Check if client is logged in, if fail, return the status
        if ((string)_context.Session["USERLOGGEDIN"] != "YES")
        {
            retorno = "r0";
        }
        else
        {
            _context.Session.RemoveAll();
            retorno = "r0";
        }
    }
    catch (Exception)
    {
        retorno = "r7";
    }
    return retorno;
}
```

Segmento de código 3.25. Método `CerrarSesion`

3.4.2.1.1 Prueba del establecimiento de sesión

Para empezar, se pasa el entregable que corresponde al servicio de establecimiento de sesión al paso de proceso de pruebas como se observa en la Figura 3.26.

Primero se procede a probar desde el cliente web, el caso de autenticación exitosa para lo que se ingresa un usuario registrado a los controles de *login* como se observa en la Figura 3.27, el resultado inmediato es acceso a la interfaz principal del aplicativo que se muestra en la Figura 3.28. En segundo lugar, se procede a hacer la misma prueba ingresando una contraseña maestra falsa, el resultante es el mostrado en la Figura 3.29. Finalmente se prueba el servicio con credenciales no registradas, la respuesta es la de la Figura 3.30.

Figura 3.26. Tarjeta “Servicio de autenticación y establecimiento de sesiones por contexto HTTP” en el paso de procceso “Prueba”

Figura 3.27. Controles de *login* del aplicativo web

Figura 3.28. Pantalla principal del aplicativo web

Figura 3.29. Mensaje en el que el aplicativo web indica que la contraseña ingresada es incorrecta

Figura 3.30. Mensaje en el que el aplicativo web indica que la contraseña ingresada es incorrecta

3.4.2.2 Cifrado de campos sensibles en la base de datos

Previo a guardar en la base un campo sensible, ayudados del modelo *Entity Framework* y operadores lambda[1], el campo se enviará al método `EncriptarContrasena()` que se lo puede observar en el Segmento de código 3.26 que establece los parámetros con los que se desea cifrar la cadena de texto y que a su vez hace uso de funciones de cifrado y descifrado de la clase EncriptorDescriptor.cs cuyo código se encuentra compartido en el Anexo B.5.

Cuando se desea acceder al campo cifrado se hace uso del método inverso `DesencriptarContrasena()` del Segmento de código 3.27.

```
        private static string EncriptarContrasena(string contrasenaIn)
        {
            string retorno;
            try
            {
                // Create a new instance of the Aes
                // class.  This generates a new key and initialization
                // vector (IV).
                using (var myAes = Aes.Create())
                {
                    // Encrypt the string to an array of bytes.
                    if (myAes != null)
                    {
                        myAes.Key = Encoding.UTF32.GetBytes("Ll@v3G3s");
                        myAes.IV = Encoding.ASCII.GetBytes("GestorContrasena");
                        var encrypted = EncriptorDescriptor.EncryptStringToBytesAes(
                            contrasenaIn, myAes.Key, myAes.IV);
                        retorno = Convert.ToBase64String(encrypted);
                    }
                    else
                    {
                        retorno = string.Empty;
                    }
                }
            }
            catch (Exception)
            {
                retorno = string.Empty;
            }
            return retorno;
        }
```

Segmento de código 3.26. Método que realiza el cifrado previo al registro en la BDD

[1] **EXPRESIONES LAMBDA:** Son funciones anónimas que se pueden usar para crear tipos delegados o de árbol de expresión. Las expresiones lambda son especialmente útiles para escribir expresiones de consulta LINQ.

```
        private static string DesencriptarContrasena(string contrasenaIn)
        {
            string retorno;
            try
            {
                var inputBytes = Convert.FromBase64String(contrasenaIn);

                // Create a new instance of the Aes
                // class.  This generates a new key and initialization
                                    // vector (IV).
                using (var myAes = Aes.Create())
                {
                    // Decrypt the bytes to a string.
                    if (myAes != null)
                    {
                        myAes.Key = Encoding.UTF32.GetBytes("Ll@v3G3s");
                        myAes.IV = Encoding.ASCII.GetBytes("GestorContrasena");
                        retorno = EncriptorDescriptor.DecryptStringFromBytesAes(inputBytes,
                            myAes.Key, myAes.IV);
                    }
                    else
                    {
                        retorno = string.Empty;
                    }
                }
            }
            catch (Exception)
            {
                retorno = string.Empty;
            }
            return retorno;
        }
```

Segmento de código 3.27. Método que descifra la información almacenada en la BDD

3.4.2.2.1 Prueba del cifrado de campos sensibles

Se realiza una consulta a la tabla usuarios de la base de datos, como se muestra en la Figura 3.31. Se verifica que el campo `ContrasenaMaestra` de la tabla `Usuarios` tiene sus registros cifrados.

El resultado es el mismo para los campos a cifrar de la tabla `Contrasenas` y `ContrasenasBakcup`.

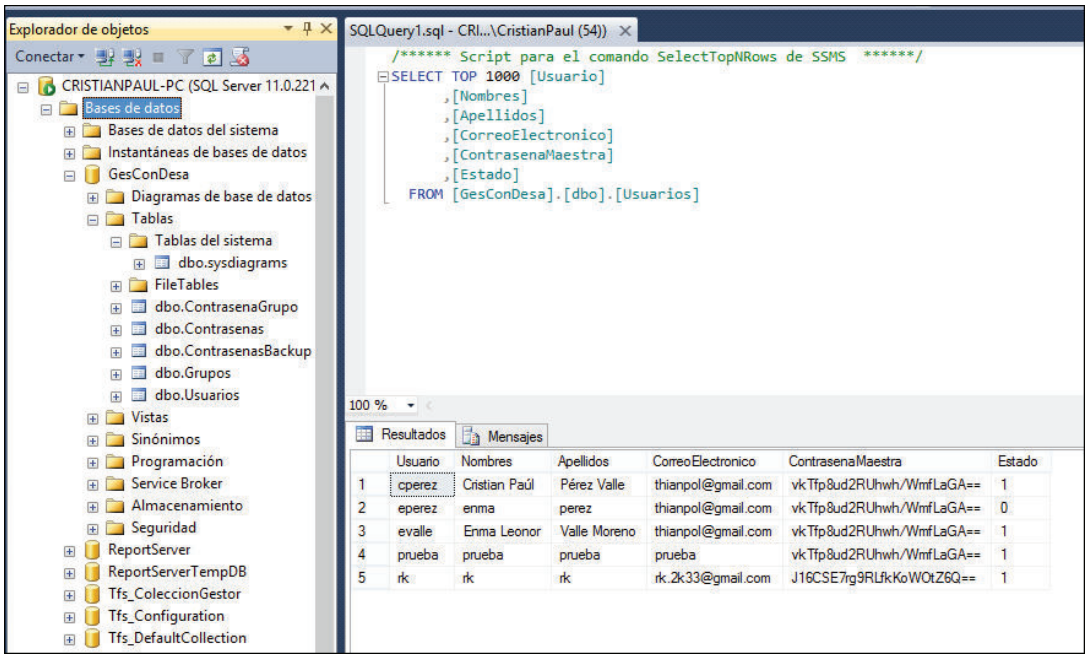


Figura 3.31. Tabla `Usuarios` de la base de datos en la que se observan los registros cifrados

3.4.2.3 Validación en el servicio de que se registre una contraseña maestra fuerte

Durante la ejecución del servicio de registro de usuarios o la ejecución del servicio de cambio de contraseña maestra, se ejecutará el método `ValidarContrasena()` que se observa en el Segmento de código 3.28. Se exige que la contraseña tenga al menos una letra mayúscula, una letra minúscula, un número y un carácter especial. Se valida además que la contraseña tenga al menos 8 caracteres.

3.4.2.3.1 Prueba de la validación en el servicio para que se registre una contraseña maestra fuerte

Al ingresar en el formulario de registro que solo se implementa en el cliente web, un valor que no cumpla con los criterios de registro de una contraseña fuerte, el cliente web obtiene esta respuesta del servidor y presenta el mensaje que se observa en la Figura 3.32.

Figura 3.32. Mensaje en el que el aplicativo web indica las caracterísiticas que debe cumplir la contraseña maestra a registrar

```
private static bool ValidarContrasena(string contrasena)
        {
            var mayus = false;
            var minus = false;
            var especial = false;
            var retorno = false;

            if (contrasena.Length >= 8)
            {
                foreach (var caracter in contrasena)
                {
                    var alphaMayus =
                           "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ".ToCharArray();

                    if (alphaMayus.Any(letra => caracter == letra))
                    {
                        mayus = true;
                    }
                    var numeros = "1234567890".ToCharArray();

                    if (numeros.Any(letra => caracter == letra))
                    {
                        minus = true;
                    }
                    var caracteresEspeciales = "!@#$%&*".ToCharArray();

                    if (caracteresEspeciales.Any(letra => caracter == letra))
                    {
                        especial = true;
                    }
                }

                if (mayus && minus && especial)
                    retorno = true;

            }
```

Segmento de código 3.28. Método que valida la fuerza de la contraseña que exige el sistema

3.4.2.4 Servicio de recuperación de contraseña maestra por correo electrónico

Se seguirá el proceso para recordar la contraseña maestra de un usuario. Mediante la `UriTemplate = "solicitudUsuario"` el usuario accede al servicio `ContruirSolicitudUsuario`. La solicitud lleva encapsulado con XML un `ObjetoUsuario` con el identificador `s6` que indica se usará el método `RecordarContrasenaMaestra()` del Segmento de código 3.29, como se puede observar en el Segmento de código 3.30.

```
private String RecordarContrasenaMaestra(string nombreUsuario)
        {
            string resultado;
            try
            {
                _usuario = _entidadConexionBdd.Usuarios.
                    FirstOrDefault(u => u.Usuario == nombreUsuario);

                if (_usuario != null)
                {
                    var cr = new Correos();
                    var mnsj = new MailMessage();
                    var contrasena =
                           DesencriptarContrasena(_usuario.ContrasenaMaestra);

                    mnsj.Subject = "Recuperación contraseña sistema de
                           administración";
                    mnsj.To.Add(new MailAddress(_usuario.CorreoElectronico));
                    mnsj.From = new MailAddress("gestorcontrasenas@gmail.com",
                        "Servicios de administración de contraseñas");
                    mnsj.Body = "Su contraseña es: " + contrasena;
                    /* Enviar */
                    cr.MandarCorreo(mnsj);

                    resultado = "r1";
                }
                else
                {
                    resultado = "r2";
                }
            }
            catch (Exception)
            {
                resultado = "r7";
            }
            return resultado;

        }
```

Segmento de código 3.29. Método `RecordarContrasenaMaestra`

```
case "s6":
    retorno = RecordarContrasenaMaestra(user.Usuario);
    break;
```

Segmento de código 3.30. Acceso desde el servicio al método `RecordarContrasenaMaestra`

En el método `RecordarContrasenaMaestra()` del Segmento de código 3.29 hace uso del método `DesencriptarContrasena()` que realiza el proceso opuesto al utilizado para registrar la contraseña maestra, descifrando la cadena de texto que se argumente, y se lo puede observar en el Segmento de código 3.27. Luego se construye un correo electrónico y se lo envía ayudado del método `MandarCorreo()` del objeto `Correos` cuyo código se puede ver en el Anexo B.5

3.4.2.4.1 Prueba de recuperación de contraseña maestra por correo electrónico

Al hacer clic en el enlace "¿Olvidaste tu contraseña?" del formulario de *login* se presentan los controles que permiten enviar el usuario que desea recuperar la contraseña como se muestra en la Figura 3.33.

Figura 3.33. Controles del cliente web para recuperacion de contraseñas

Si el usuario está registrado se procede a enviar el correo electrónico de la Figura 3.34 y a imprimir en el formulario del aplicativo web el mensaje de la Figura 3.35.

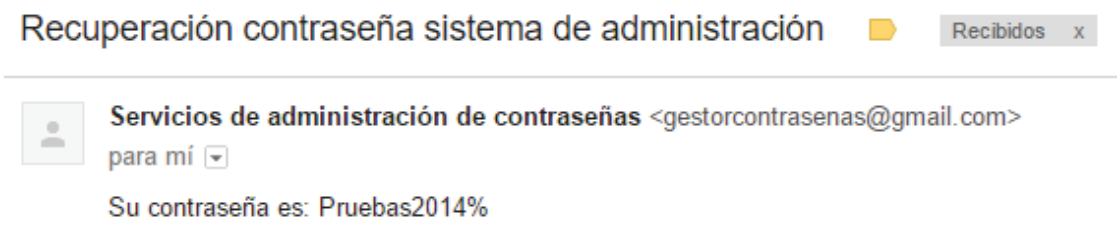


Figura 3.34. Correo electrónico enviado por el servicio de recuperación de contraseñas

Si el usuario no está registrado en el sistema, el formulario imprime el mensaje de la Figura 3.36.

Figura 3.35. Mensaje en el que el aplicativo web indica que la contraseña a sido enviado al correo electrónico registrado en el sistema

Figura 3.36. Mensaje en el que el aplicativo web indica que la contraseña a sido enviado al correo electrónico registrado en el sistema

Otros procedimientos de seguridad a nivel del servidor

- Servicio de cancelación lógica de cuenta
- Servicios de *backup* de contraseñas

La programación de estos procedimientos se la puede encontrar en el Anexo B.

Una vez implantadas estas funcionalidades del servicio, y otras derivadas o similares que se comparten en el Anexo B, se procede a entregar y cerrar todas las tarjetas de los entregables de levantamiento de servicio y entregables del desarrollo del servicio, esto se muestra en la Figura 3.37 y la Figura 3.38.

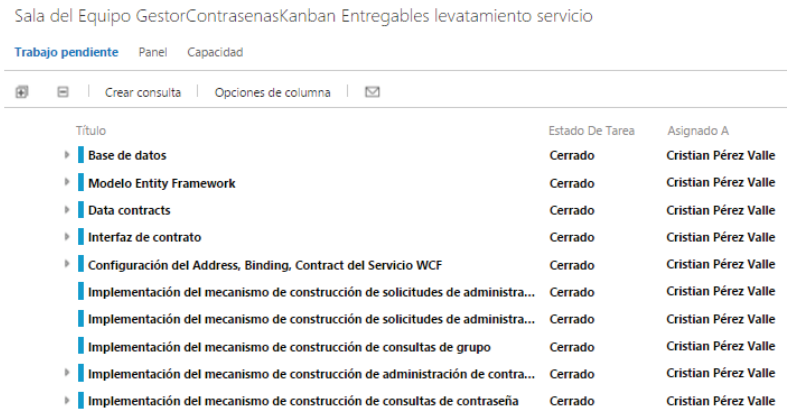


Figura 3.37. Tarjetas cerradas de los entregables del levantamiento del servicio

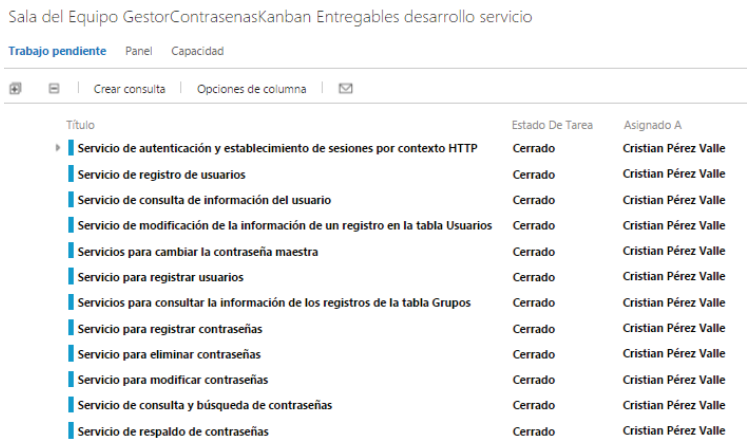


Figura 3.38. Tarjetas cerradas de los entregables del desarrollo del servicio

3.5 ANÁLISIS DEL PROTOCOLO

Se analizará el tráfico a nivel de protocolo con la herramienta WireShark en el lado del servidor, el servicio está publicado sin asegurar el canal.

El cliente está publicado en otro equipo con el objetivo de capturar el tráfico en la interfaz de red durante las solicitudes cliente servidor.

Si el cliente y el servicio estuviesen publicados en el mismo servidor web, el tráfico capturado no sería de mucha ayuda. Lo único que se puede obtener con esa configuración son las solicitudes entre el explorador de Internet y el servidor web.

Como el tráfico en el canal no está cifrado se puede usar el siguiente filtro en WireShark para capturar las solicitudes generadas desde la publicación cliente: `ip.addr == 192.168.0.9 && http`, donde 192.168.0.9 es la dirección IP de la publicación cliente.

Se captura una solicitud al servicio publicado localmente con la URI:

http://192.168.0.6/ServiciosGestorContrasenas/ServiciosGestorContrasenas.svc/solicitudUsuario

Por cada llamada del aplicativo cliente al Servicio WCF se generan tres tramas HTTP que se muestran en la Figura 3.39, la primera es la confirmación de que el servidor está escuchando, la segunda es la solicitud generada por el aplicativo cliente, y la tercera es la respuesta del servicio al cliente. En la a trama generada en la solicitud maneja los protocolos HTTP/XML.

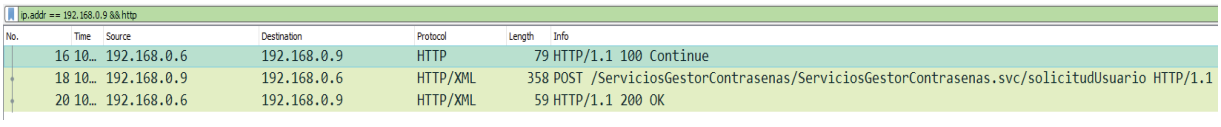


Figura 3.39. Tramas HTTP capturadas

La trama generada en la solicitud maneja los protocolos HTTP/XML. El método HTTP utilizado es POST, la carga útil de la petición va embebida en la solicitud, pero al tener capturada la trama en un canal sin cifrar, se procede a analizar su contenido, el cual se observa en la Figura 3.40.

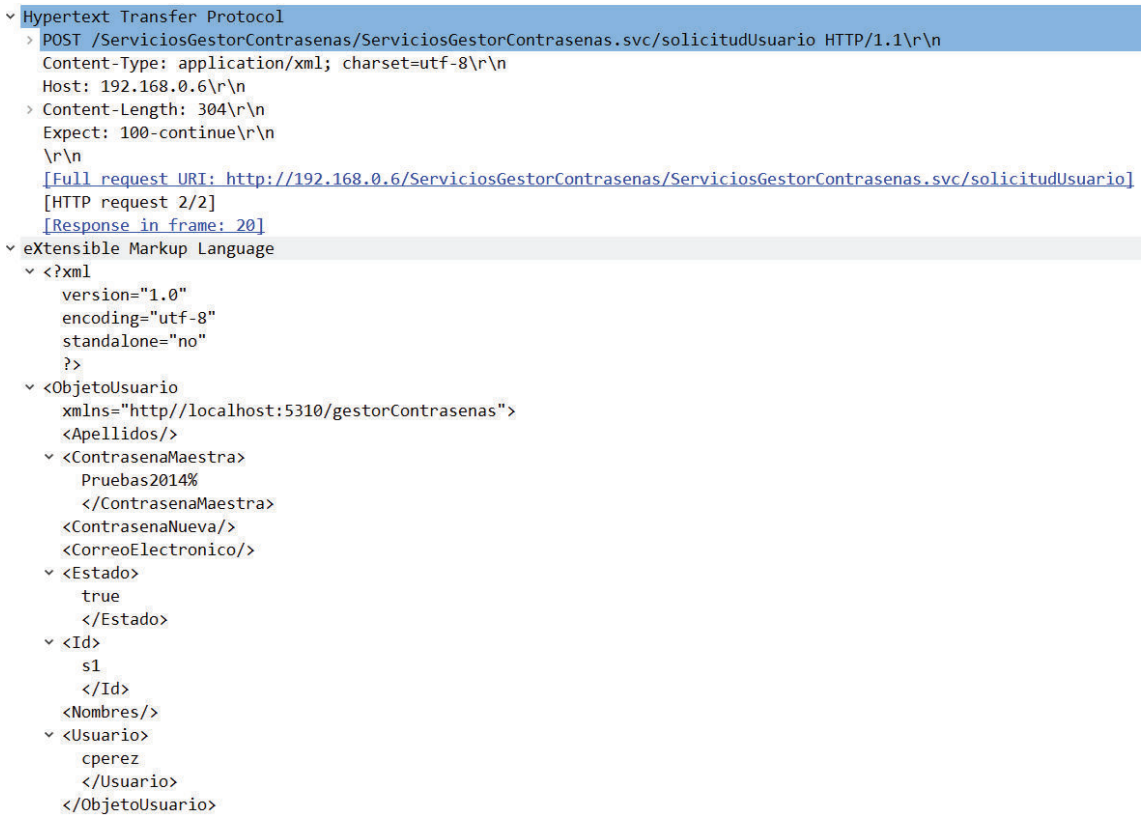


Figura 3.40. Trama de solicitud con los valores necesarios para ejecutar el servicio de autenticación de usuario

En el cuerpo de la trama POST viaja con estructura XML la figura `ObjetoUsuario`, la etiqueta `<id>` es el identificador del método al que se direccionará la solicitud una vez sea procesada en el lado del servidor, los otros datos que se contienen en el XML servirán como argumentos de entrada del método de autenticación. No todas las etiquetas de la estructura llevan información, ya que esta misma figura XML servirá para llevar datos a otros métodos según se requiera haciendo uso del identificador `<id>`.

El servicio responde con otra trama HTTP/XML, como se identifica en la Figura 3.41. Con la particularidad de que esta respuesta en su estructura HTTP lleva un identificador de la sesión a mantenerse con una cookie con el valor `ASP.NET_SessionId=v2yq5kvczpjer0t3cwgiqtoq`, particularidad solo de respuestas a solicitudes de autenticación o de solicitudes de creación de usuarios. Como se observa en la Figura 3.44 que también constituye otra repuesta del servidor, esta cookie ya no es transferida. Por el contrario, ahora la cookie debe ser embebida en la solicitud HTTP del cliente como se puede observar el la Figura 3.42 y la Figura 3.43 que son otras peticiones al servidor. Mediante este proceso se maneja el establecimiento de sesiones cliente servidor.

En la Figura 3.41 se observa la estructura XML donde se encuetra la respuesta del servicio, que en este caso es un *string* con el valor “r1”. Cuando el cliente recibe esta respuesta por parte del servidor se le redirecciona a la pantalla administrativa del aplicativo.

```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Cache-Control: private\r\n
  Transfer-Encoding: chunked\r\n
  Content-Type: application/xml; charset=utf-8\r\n
  Server: Microsoft-IIS/10.0\r\n
  Set-Cookie: ASP.NET_SessionId=v2yq5kvczpjer0t3cwgiqtoq; path=/; HttpOnly\r\n
  X-AspNet-Version: 4.0.30319\r\n
  X-Powered-By: ASP.NET\r\n
  Date: Sun, 29 May 2016 16:09:19 GMT\r\n
  \r\n
  [HTTP response 2/2]
  [Time since request: 0.002449000 seconds]
  [Prev response in frame: 16]
  [Request in frame: 18]
  HTTP chunked response
eXtensible Markup Language
  <string
     xmlns="http://schemas.microsoft.com/2003/10/Serialization/">
     r1
     </string>
```

Figura 3.41. Respuesta del servidor indicando autenticación exitosa

La solicitud POST de la Figura 3.42, que se ejecuta sobre la URI que apunta al servicio `consultaGrupos`, además de contener la cookie de identificación de sesión, lleva

una estructura XML definida para la entidad `ObjetoGrupo`. El Id del método a direccionarse una vez llegue al servidor, es “c1”, su funcionalidad es traer todos los grupos que le pertenecen al usuario identificado en la etiqueta `<IdUsuario>`. Por tanto, esta vez la respuesta que se puede observar en la Figura 3.44, no solo traerá en formato XML un *string* que identifique el éxito o no de la solicitud, sino que, esta vez traerá una colección de objetos grupo en formato XML, la respuesta trae 7 objetos grupo, la figura muestra el contenido del último grupo.

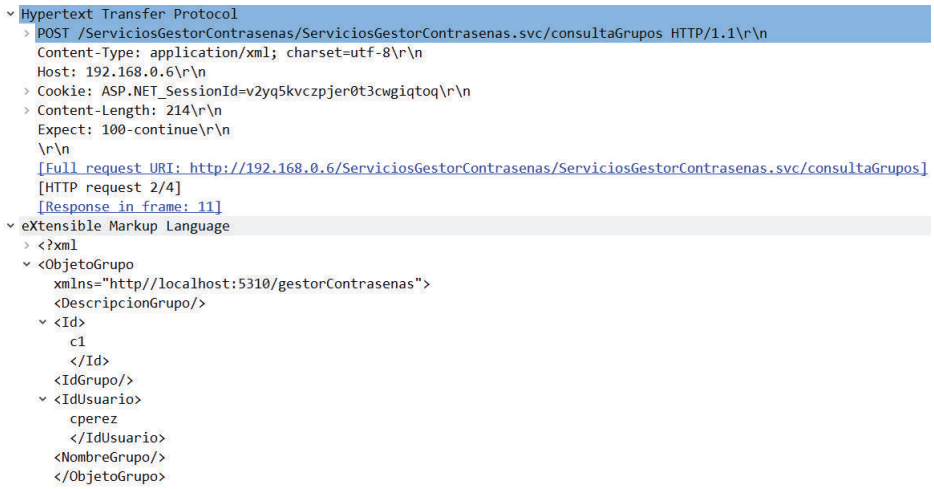


Figura 3.42. POST que consulta los grupos que le pertenecen a un usuario

En la pantalla de administración de contraseñas, donde se ejecuta la consulta de los grupos que le pertenecen a un usuario, se carga en una grilla las contraseñas del grupo seleccionado. Para esto se efectúa la consulta que se indica en la Figura 3.43 la lógica de esta consulta es la misma que la descrita para la Figura 3.42 y el parámetro para la consulta es el identificador de grupo que es “grup-1”. En la Figura 3.45 observa que este grupo solo contiene una contraseña que está contenida en la colección de objetos contraseña. En la misma figura se puede observar que la respuesta del servidor carga ninguna cookie de sesión. La cookie de sesión solo se envía durante el establecimiento de sesión a través de las funcionalidades de creación de cuenta o autenticación. La misma debe ser guardada y enviada en cada solicitud del cliente.

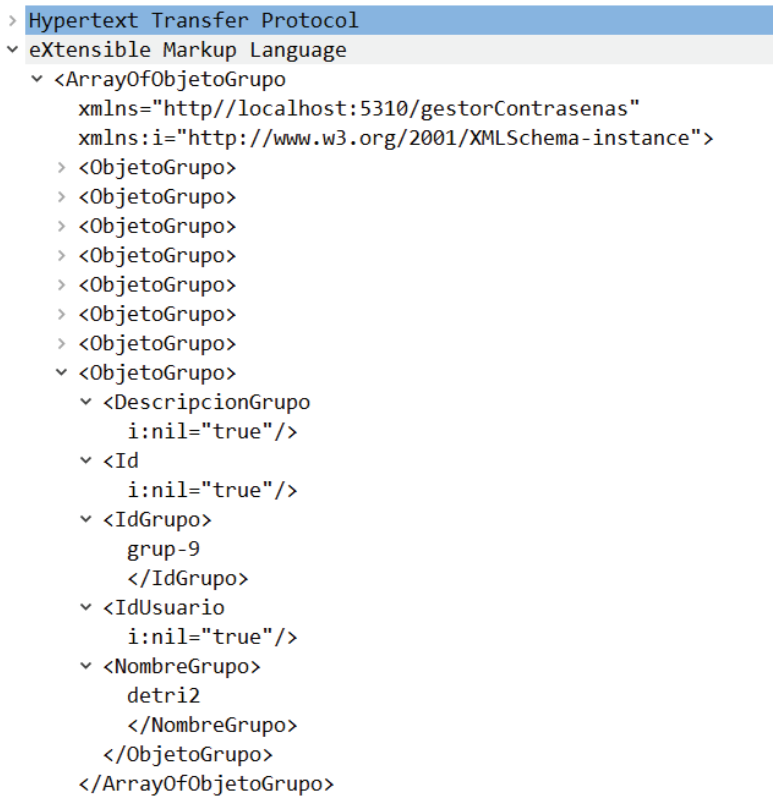


Figura 3.44. Respuesta del servidor con una colección de grupos en XML

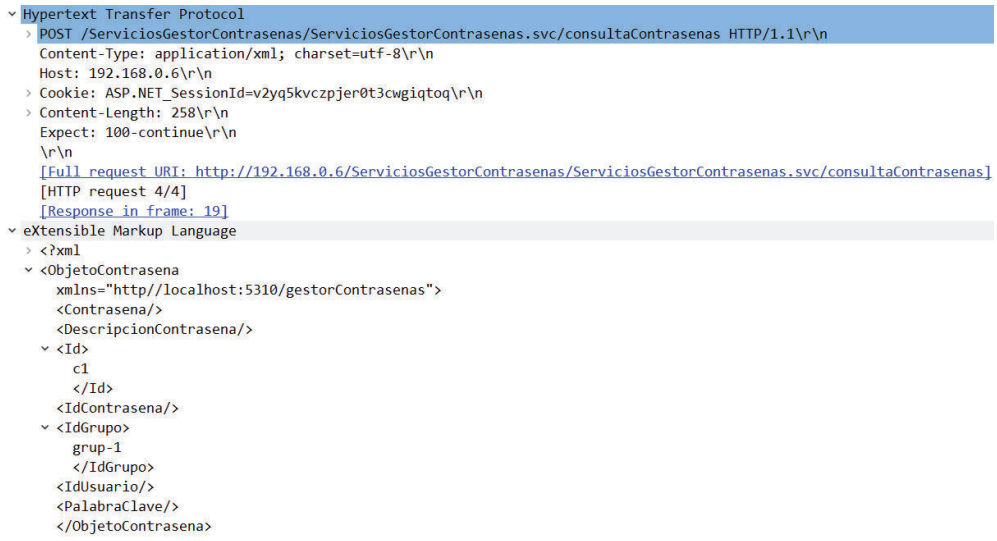


Figura 3.43. POST que consulta las contraseñas que pertenecen a un grupo

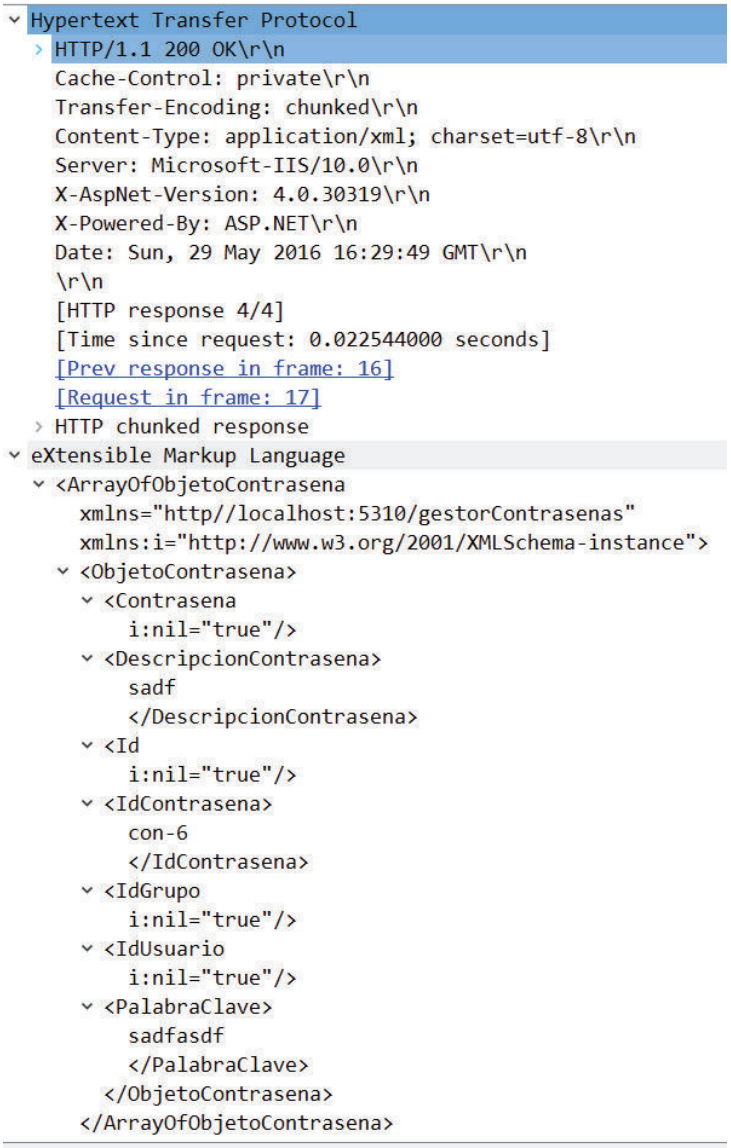


Figura 3.45. Respuesta del servidor con una colección de contraseñas en XML

3.6 REQUERIMIENTOS DE LOS COMPONENTES DEL PROTOTIPO

Con base en las características de los dispositivos utilizados para implementar el prototipo, los requisitos necesarios para la implantación del sistema se listan en la Tabla 3.1 y la Tabla 3.2.

Se garantiza que si se manejan estos requisitos el sistema trabajará con normalidad.

Tabla 3.1. Requisitos de Software

Software	
Nombre	Versión
Sistema Operativo Microsoft Windows	7 Professional o superior
.NET Framework	4.5
Internet Information Services	8.0
Microsoft SQL Server	2012 Express o superior
Emulador Android Nexus S con 480p de resolución e imagen Android 4.1.2	API 16 – *Jelly Bean*, 343MB en RAM, sin aceleramiento por *hardware,* Pantalla HVGA de 3.5 pulgadas, 128MB de memoria ROM o memoria interna

Tabla 3.2. Requisitos de Hardware

Hardware	
Dispositivo	Características
Servidor de aplicaciones	Procesador Intel(R) Core(TM)2 Duo o similar AMD; CPU con una frecuencia de reloj de 2.1 GHz y 4GB RAM
Servidor de base de datos	Procesador Intel(R) Core(TM)2 Duo o similar AMD; CPU con una frecuencia de reloj de 2.1 GHz y 4GB RAM

CAPÍTULO 4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- Se diseñó un sistema distribuido basado en una arquitectura SOA (*Service-Oriented Architecture*) implementado mediante un servidor de base de datos, servicios de administración de contraseñas mediante WCF REST, dos clientes ligeros, el primero de tipo Web y el segundo para la plataforma Android. Se demostró que todos los componentes interactuaron según los requerimientos.

- Con los dos clientes se demostró que el servicio al ser WCF REST es adaptable a cualquier tipo de plataforma y que el cliente solo necesita el manejar protocolo HTTP ampliamente utilizado. Con las librerías adecuadas se puede obtener el mismo resultado al acceder a Servicios Web basados en REST.

- Se aplicó seguridad en las comunicaciones con cifrado de información, comunicaciones sobre un canal seguro, establecimiento de sesión, control de tiempo de sesión, recuperación de contraseña, control de contraseña fuerte, respaldo de cambios sobre la contraseña, etc.

- En el sistema se demostró que existen mecanismos de transferencia de estado que pueden ser definidos sobre protocolos sin estado como es el caso de HTTP, ya que en la solicitud se encapsuló toda la información necesaria para manejar un mecanismo de sesión, con instancias de servicio independientes por cliente. En el servicio se aplicaron mecanismos para manejar sesiones independientes a través de la transferencia del estado a una instancia de escucha del servidor.

- Kanban es una buena metodología para arrojar entregables de un producto software en tiempos cortos.

- Se aplicó, para el cliente Android, solo la funcionalidad de administración de contraseñas. Para trabajar con la administración de cuentas de usuario y el registro de usuarios, es necesario usar la interfaz web.

- Con *Entity Framework* se accedió desde el Servicio WCF a la base de datos, el beneficio que se obtuvo mediante esta tecnología es liberar al sistema de procesos almacenados y disparadores para leer, actualizar, ingresar, eliminar los datos de la base.

- Se demostró mediante el análisis del protocolo que el cliente enviaba una solicitud POST en una trama HTTP/XML. El XML embebido en la trama tenía los datos necesarios para que el servidor atienda la solicitud.

- Se demostró en el análisis del protocolo que el servidor puede encapsular colecciones de objetos con el protocolo XML y transportarlos sobre HTTP.

4.2 RECOMENDACIONES

- Al trabajar con *Entity Framework* se recomienda crear objetos por cada tabla de la base de datos ya que un cambio en el modelo de base de datos durante las pruebas unitarias, se traduce en la generación de un nuevo `Entity Model` en el prototipo y con ello nuevas `Entity Class`.

- Es importante asegurarse que las aplicaciones puedan soportar futuros cambios de los algoritmos criptográficos, por lo se sugiere mantener el módulo sencillo de encriptación del prototipo para facilitar el cambio.

- Durante el desarrollo es beneficioso utilizar un sistema de control de código fuente que mantenga un historial de versiones de las clases y demás ficheros de la solución, o en su lugar guardar versiones del proyecto independientes del código

en desarrollo, con lo que se obtiene líneas de base tanto para posibles situaciones de reversión de código, como también para tener recursos de consulta durante el desarrollo del mismo.

- Durante el desarrollo e implantación del presente proyecto de titulación se usó un certificado local, no verificado; sin embargo, se sugiere usar un certificado de prueba otorgado por alguna entidad certificadora para proyectos a implantarse en ambientes reales en especial durante las pruebas de preproducción, con estos certificados la interacción con los Servicios Web suele presentar *bugs* que no se presentan con los certificados no verificados.

- Se recomienda que el equipo de trabajo tenga al menos una persona enfocada en cada etapa del desarrollo y que esto incluya la documentación del software. En el peor de los casos que un programador pruebe el módulo implementado por otro programador y viceversa.

- Se recomienda trabajar con el entorno de ejecución *Android Runtime* para futuras implementaciones enfocadas a sistemas Android 5.0 y superiores.

- Durante el desarrollo del aplicativo Android los *debug* necesarios para realizar pruebas unitarias eran continuos. Esos corren sobre un dispositivo emulado que consume bastantes recursos. Por esta razón se sugiere contar con un equipo de buenas capacidades y que pueda simular sistemas por hardware para el entorno de desarrollo de estos aplicativos.

- Se recomienda al menos dos desarrolladores para trabajar en paralelo con las tarjetas según el límite de trabajo en curso establecido por recurso. Así se puede aprovechar de una mejor manera las ventajas de la metodología Kanban. Se necesitaría de igual manera al menos un recurso adicional solo enfocado en las pruebas.

REFERENCIAS BILIOGRÁFICAS

[1] T. Andrew, Pro C# 2010 and the .NET 4 Platform, Fifth Edition, New York: Apress, 2010.

[2] R. Diego, 2009. [Online]. Available: http://icomparable.blogspot.com/2009/01/capas-del-modelo-conceptual-de-soa.html.

[3] R. Margaret, 2005. [Online]. Available: http://searchsoa.techtarget.com/definition/UDDI.

[4] Microsoft Corporation, 2014. [Online]. Available: http://msdn.microsoft.com/es-ec/library/ms734769.aspx.

[5] C. Paúl, 2012. [Online]. Available: http://blog.smartekh.com/%C2%BFque-es-hardening/.

[6] Android Developers, "developer.android.com," 2013. [Online]. Available: http://developer.android.com/guide/topics/manifest/manifest-intro.html.

[7] K. Henrik and S. Mattias, Kanban y Scrum - obteniendo lo mejor de ambos, USA: C4Media, editores de InfoQ.com., 2010.

[8] Visual Studio ALM RANGERS, Team Foundation Server Practival Kanban Guidance, Microsoft Corporation., 2012.

[9] G. Rafael, 2012. [Online]. Available: http://www.fayerwayer.com/2012/12/los-usuarios-de-internet-necesitan-22-paswords/.

[10] Android Developers, "developer.android.com," 2015. [Online]. Available: http://developer.android.com/intl/es/tools/sdk/eclipse-adt.html.

[11] Visual Studio ALM RANGERS, 2012. [Online]. Available: http://vsarkanbanguide.codeplex.com/releases/view/88534.

[12] Microsoft Corporation, "MSDN," 2015. [Online]. Available: https://msdn.microsoft.com/en-us/library/zhhddkxy(v=vs.100).aspx.

[13] S. Jesús, 2014. [Online]. Available: http://pensandoensoa.com/2014/06/29/tenemos-hacer-preguntas-para-empezar-con-soa/.

[14] A. Luis, "Programania.net," 2009. [Online]. Available: http://www.programania.net/desarrollo-agil/desarrollo-agil-con-kanban/. [Accessed 18 Noviembre 2012].

[15] Android Developers, 2014. [Online]. Available: http://developer.android.com/about/dashboards/index.html#platform.

ANEXOS

Los anexos se incluyen en el CD adjunto al presente documento

ANEXO A: Pruebas

A.1 CASOS DE PRUEBA

A.2 REPORTE DETALLADO DE PRUEBAS

ANEXO B: Segmentos de código complementarios

B.1 INTERFAZ DE CONTRATO

B.2 DATA CONTRACS

B.3 DEFINICIÓN DE SERVICIOS

B.4 DEFINICIÓN DE METODOS DEL SERVICIO

B.5 CLASES ENCRIPTOR-DESCRIPTOR Y CORREOS