Extensión del subproceso de autenticación del Sistema de Administración de Identidades del CISED

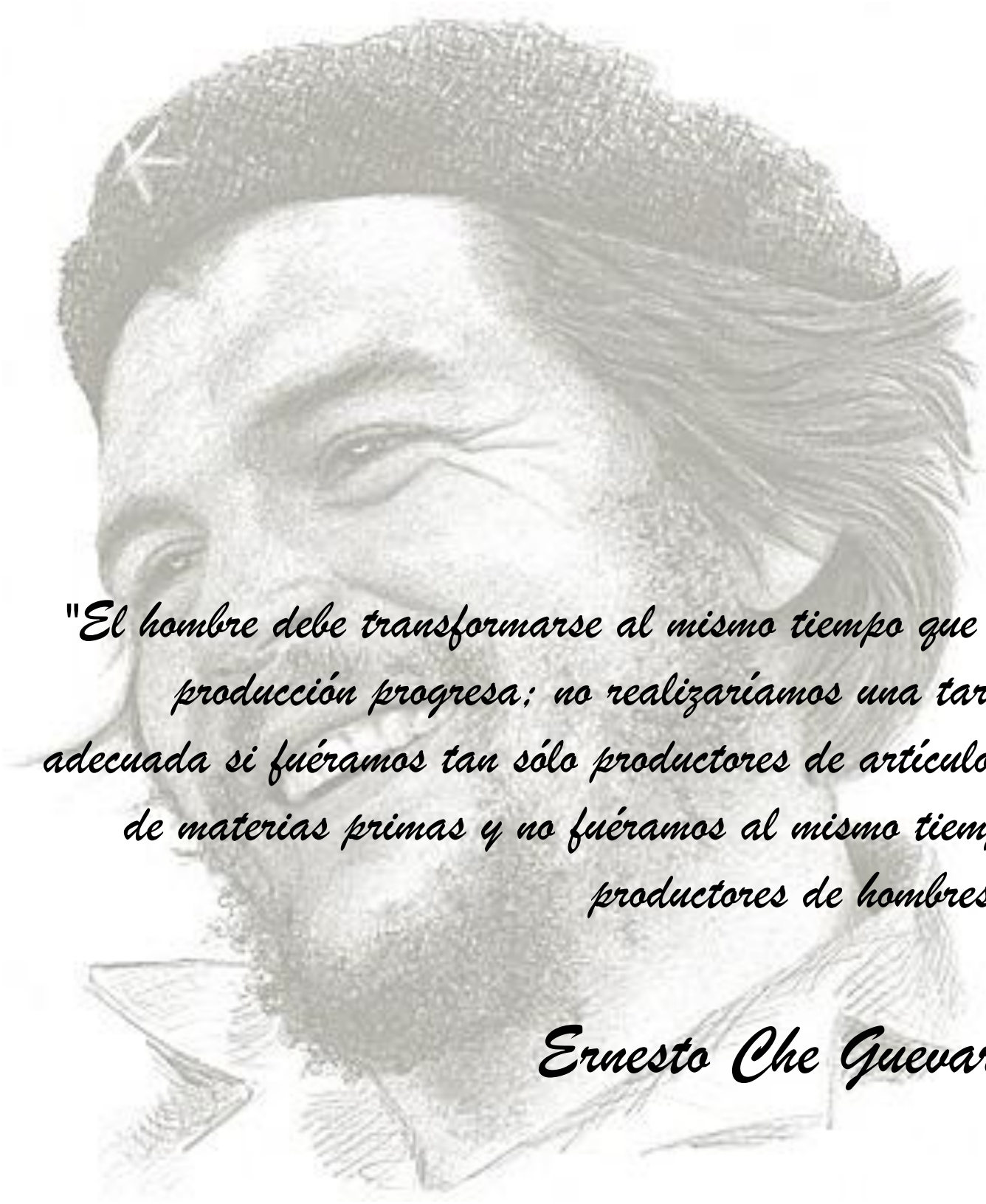
Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autores: Grettel Mola Oliveros
Reinier Ruíz Cuellar

Tutores: Ing. María Lourdes Morilla Faurés
Ing. José Carlos Correa Bautista

Consultante : Ing. Luis Fernández Leyva

La Habana, junio de 2013



"El hombre debe transformarse al mismo tiempo que la producción progresa; no realizaríamos una tarea adecuada si fuéramos tan sólo productores de artículos, de materias primas y no fuéramos al mismo tiempo productores de hombres."

Ernesto Che Guevara

Declaración de autoría

Declaramos ser autores del presente trabajo de diploma y reconocemos a la UCI los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste, firmamos el presente a los __ días del mes de _____ del año____.

_____	_____
Grettel Mola Oliveros	Reinier Ruíz Cuellar
_____	_____
Tutor	**Tutor**
Ing. María Lourdes Morilla Faurés	Ing. José Carlos Correa Bautista

Agradecimientos

Agradezco la familia en la que crecí, los padres tan maravillosos que la vida me dio, a los cuales les debo la vida y la persona que soy hoy. Gracias por sus consejos, su apoyo incondicional, por estar siempre ahí presentes en mi vida y sobre todo en esta etapa tan importante que nunca olvidaré.

A mi hermanito querido gracias por existir y por ser mi hermano del alma. Por apoyarme y también por ser parte de esta etapa, te quiero mucho, gracias por tu apoyo.

A mi familia que desde Camagüey aunque la distancia nos separaba sabía que estaban ahí, a mi Ame querida que tantos consejos me dio al enfrentarme en este nuevo viaje que muy útiles me fueron, a mi cuñada también por estar ahí presente, y a todos aquellos que de una forma u otra me ayudaron.

A mi compañero de tesis por soportarme día tras día, y por todo su trabajo y dedicación a este trabajo de diploma. A mis tutores que también dieron su apoyo a este trabajo, y sobre todo a Lourdiña que se convirtió en una amiga.

A mi amigo Haniel que tanto apoyo me dio cuando más lo necesitaba, en un momento tan importante y decisivo, nunca lo olvidaré.

A mis compañeros del grupo que fue genial conocerlos, y sobre todo a mi Mary, mi amiga del alma que cada día la extraño más y más, que ya no está pero desde donde esté hoy le quiero agradecer por su compañía y las cosas tan lindas y buenas que me aportó, a Yasiel por ser buen amigo, mis chicas del apto, a Daynelis y a Ronal por ser los amigos que nunca olvidaré.

A todos muchas gracias.

Grettel Mola Oliveros

Agradezco al entorno en el que crecí, que me dio las facultades para pensar en mi futuro y escoger esta carrera; sobre todo a mis padres, fieles amigos, acompañantes y consejeros que si no fueran por sus sacrificios no estaría en estos momentos aquí, a ellos que me han enseñado a encarar las adversidades sin perder nunca la dignidad, ni desfallecer en el intento.

A mi hermana y mi cuñado por creer y confiar siempre en mí, apoyándome en todas las decisiones que he tomado en la vida. A mis sobrinos por llenar de felicidad cada momento con su inocencia.

A toda mi familia que me ha dado todo lo que soy como persona, mis valores, mis principios, mi perseverancia y mi empeño, y todo ello con una gran dosis de amor y sin pedir nunca nada a cambio.

A mi compañera de tesis por su ardua labor en la realización de este trabajo.

A mis tutores, por ser guías y ejemplo, por tantas horas de apoyo sin temor al cansancio, o sencillamente por ser las personas que son.

A mis compañeros de clases y de residencia, por el apoyo, motivación y preocupación que de ellos he recibido. Al 1105 de hace 5 años atrás por ser el motor impulsor del comienzo de mi vida como universitario.

A mis profesores de la Universidad de Ciencias Informáticas, por su ejemplo de profesionalidad que nunca he de olvidar.

A todas aquellas personas que de una forma u otra colaboraron o participaron en la realización de esta investigación, hago extensivo mi más sincero agradecimiento.

Gracias a la vida que tengo y a mis amigos que más quiero. Si no fuera por ellos mi sueño no lo habría cumplido.

Gracias a la UCI, nuestra UCI, por permitirme terminar esta carrera en donde profesores y compañeros dejan parte de su vida, para dar vida a las ilusiones de niño y que hoy en día se hacen realidad. Solo espero que este camino sea solo el comienzo de una gran historia.

A todos muchas gracias.

Reinier Ruíz Cuellar

Dedicatoria

A mis padres y mi hermano que son lo más importante en mi vida. Los quiero mucho mucho.

Grettel Mola Oliveros

A mis abuelas, porque aunque no estén presentes sé que sus almas si lo están, ojalá y me hubiesen visto convertirme en la persona que soy, ojalá y estén orgullosas de mí. Les dedico mi tesis con todo mi corazón. Nunca las olvidaré.

Reinier Ruíz Cuellar

Resumen

El presente trabajo de diploma surge debido a la necesidad de fortalecer la seguridad en el subproceso de autenticación del Sistema de Administración de Identidades del Centro de Identificación y Seguridad Digital. Dicha premisa se refiere a la creación de un mecanismo para la inclusión de One-Time Password como adición de un token generado, agregando a su vez un factor en la clasificación de la fortaleza de la autenticación.

La actual solución está compuesta por 3 aplicaciones: una web, una desktop para clientes con sistema operativo Windows y una cliente para dispositivos móviles con sistema operativo Android, las cuales sin comunicación alguna, son capaces de generar el mismo OTP y proporcionarlo al usuario que realiza la autenticación; además un servicio web para la integración con el Sistema de Administración de Identidades. La misma fue realizada usando los estándares líderes de autenticación con One-Time Password.

Dicho mecanismo al no constituir una solución a la medida puede ser utilizado por diversos sistemas sin la necesidad de realizar grandes cambios en los mismos.

PALABRAS CLAVE: autenticación, One-Time Password, seguridad.

Índice

Índice de figuras

Índice de tablas

Introducción

En la actualidad la seguridad se vuelve un factor clave para evitar usos o acciones fraudulentas en aplicaciones informáticas. Por ello, se hace cada vez más necesario implementar soluciones más sólidas y seguras, que busquen prevenir este tipo de operaciones. De esta forma se evita la reutilización de datos estáticos con métodos efectivos pero sencillos, que no comprometan la experiencia del usuario, y que por otro lado, no representen un costo adicional para las empresas, ni para los clientes.

La autenticación de acuerdo con Borghello (2009) “es la verificación de que el usuario que trata de identificarse es válido”, casi siempre los programadores lo implementan con usuario y contraseña en el instante de iniciar una sesión. Hay cuatro tipos de técnicas que permiten realizar la autenticación de la identidad del usuario, estas pueden ser utilizadas individualmente o combinando varias técnicas al mismo tiempo.

Las técnicas de autenticación de identidad de los usuarios se basan según Borghello (2009) en:

1. Algo que solamente el individuo conoce: una contraseña.

2. Algo que la persona posee: una secuencia de caracteres generados por una aplicación.

3. Algo que el individuo es y que lo identifica unívocamente: las huellas dactilares.

4. Algo que solo el individuo es capaz de hacer: los patrones de escritura.

Las contraseñas constituyen una forma de autenticación que utiliza información que únicamente la persona conoce, o sea es secreta, para controlar el acceso hacia algún recurso protegido. Las mismas están compuesta por un código alfanumérico y en ocasiones solamente numérico (Personal Identification Number, PIN[1]). Mientras que las contraseñas crean una seguridad contra los usuarios no autorizados, el sistema de seguridad sólo puede verificar que la contraseña introducida es válida, y no si el usuario que está haciendo uso de la misma es el correcto para utilizarla.

[1]Valor numérico (contraseña) usado para identificarse y poder acceder a sistemas, como un celular, un cajero automático, etc.

La fortaleza de la autenticación es mayor cuantos más factores se adicionen, generalmente solo se utilizan hasta 3 factores en este tema Borghello (2009) planteó:

1 factor = contraseña

2 factores = contraseña + token[2]

3 factores = contraseña + token + biometría[3]

4 factores = contraseña + token + biometría+ localización geográfica (Global Positioning System, GPS)[4]

5 factores = contraseña + token + biometría + localización geográfica + perfil de usuario.

El conocimiento a priori de una prueba que solo pueda ser superada por el usuario se convierte en el modelo de autenticación más básico. Por lo general esta prueba consiste en una contraseña que en sus inicios solo es conocida por el propietario de la misma. Este esquema aunque es el más vulnerable a todo tipo de ataques, es también el más utilizado en aquellos entornos o entidades que no necesitan altos niveles de seguridad, debido a ser el más asequible y fácil de implementar.

La autenticación basada en contraseñas consiste en que las partes que participan en la autenticación acuerdan una clave común, que han de mantener en secreto si desean que la autenticación sea confiable. En el momento de realizar la autenticación una de las partes muestra el conocimiento de esta clave común ante la otra, de ser correcta la misma, será permitido el acceso hacia el recurso protegido.

En el funcionamiento de este tipo de esquema basado en contraseñas, la confidencialidad de la clave por parte de las entidades que participan en la autenticación, es quien brinda la seguridad del mismo. Basta con que una de las partes no mantenga la contraseña en secreto para que toda la seguridad del modelo se pierda. Por lo que se hace necesario aumentar el factor de autenticación para incrementar el nivel de seguridad en las aplicaciones. Siendo la adición de un token generado

[2]Secuencia de caracteres capaz de indicar la autoridad, prueba o autenticidad de un usuario.

[3]Métodos de identificación y autenticación de los seres humanos a través de características fisiológicas y de comportamiento.

[4]Sistema Global de Navegación por Satélite el cual permite determinar en todo el mundo la posición de un objeto.

una de las variantes posibles para dar mayor fortaleza a la autenticación de los usuarios.

En la Universidad de las Ciencias Informáticas (UCI) se encuentra el Centro de Identificación y Seguridad Digital (CISED), que es el encargado del desarrollo de aplicaciones destinadas a la gestión en diversas ramas de la identificación, biometría, tarjetas inteligentes y la seguridad de las mismas. De esta última se encarga el Departamento de Seguridad Digital, en el cual una línea de investigación, desarrolla el Sistema de Administración de Identidades, que consta del Módulo de Autenticación. El mismo se clasifica según su fortaleza en factor 1, por lo que no se considera una autenticación fuerte o segura.

El concepto de "autenticación fuerte" sugerido por PandaID Soluciones C.A (2012) "está íntimamente relacionado con los factores de autenticación". Así, cuántos más factores utilice un mecanismos de autenticación, mayor dificultad existirá para poder suplantar una identidad (la autenticación será "más fuerte"). De esta forma, un mecanismo que utilice los tres factores (autenticación de factor 3) será más seguro que uno que use dos (factor 2), que a su vez será más seguro que un mecanismo de factor 1. Así, la autenticación por usuario y contraseña sería un factor 1, mientras que la que requiriese la presentación de un token el que solo se conoce luego de la previa generación de una aplicación, sería una autenticación de factor 2.

El Módulo de Autenticación del Sistema de Administración de Identidades posee una baja seguridad debido a que solo cuenta con un sistema de autenticación basado en usuario/contraseña. Con la aplicación de la autenticación fuerte en el Módulo de Autenticación del Sistema de Administración de Identidades, se logrará aumentar la seguridad disminuyendo así la vulnerabilidad del mismo frente a numerosos tipos de ataques. Esta combinación reduce aún más el riesgo de que alguien no autorizado descubra o genere un token, y por lo tanto minimiza la vulnerabilidad ante muchos de los ataques que sufren las contraseñas estáticas.

Partiendo de la presente situación problemática se plantea el siguiente **problema a resolver:** ¿Cómo fortalecer la seguridad en el subproceso de autenticación del Sistema de Administración de Identidades del Centro de Identificación y Seguridad Digital?

Se define como **objeto de estudio** el proceso de autenticación en aplicaciones web de gestión de seguridad. Enmarcando como **campo de acción** el subproceso de autenticación a través de One-

Time Password (OTP) en el Sistema de Administración de Identidades.

Se define como **objetivo general**: Realizar el mecanismo de autenticación con One-Time Password para el subproceso de autenticación del Módulo de Autenticación del Sistema de Administración de Identidades.

Para darle solución al mismo se plantean los siguientes **objetivos específicos:**

- ✓ Conformar el marco teórico de la investigación que permita sentar las bases para darle solución al objetivo planteado.
- ✓ Realizar diseño del mecanismo de autenticación con One-Time Password para la implementación exitosa de la solución que se propone.
- ✓ Implementar el mecanismo de autenticación con One-Time Password para aumentar la seguridad del Módulo de Autenticación del Sistema de Administración de Identidades.
- ✓ Probar el correcto funcionamiento del mecanismo de autenticación con One-Time Password para validar la propuesta de solución.

Para darle solución a cada uno de ellos se plantean las siguientes **tareas de la investigación**:

- ✓ Identificación de los mecanismos existentes para la autenticación con OTP.
- ✓ Caracterización de los mecanismos existentes para la autenticación con OTP.
- ✓ Definición de la metodología, las tecnologías, herramientas y lenguajes necesarios para el desarrollo del mecanismo de autenticación con OTP para el Módulo de Autenticación del Sistema de Administración de Identidades del CISED.
- ✓ Definición de los escenarios y requerimientos de calidad de servicio.
- ✓ Elaboración del diseño del mecanismo.
- ✓ Definición de la arquitectura del mecanismo.
- ✓ Desarrollo de la funcionalidad de autenticación con OTP para el Módulo de Autenticación

del Sistema de Administración de Identidades del CISED.

- ✓ Desarrollo de la aplicación para el servidor de OTP, la cliente para Windows y Android.
- ✓ Realización de pruebas de caja negra para la aplicación de escritorio, la cliente de Android y la aplicación Web.
- ✓ Realización de pruebas de caja blanca para la aplicación de escritorio, la cliente de Android y la aplicación Web.
- ✓ Realización de pruebas de fiabilidad a todo el mecanismo de autenticación.

Métodos Científicos:

Métodos teóricos:

- ✓ Analítico – Sintético: Se emplea en el análisis de los diferentes estándares OTP que existen, así como en el análisis y resumen en las fuentes bibliográficas. Se logra conocer las características principales de los One-Time Password y la identificación e interpretación de sus ventajas y desventajas.
- ✓ Histórico – Lógico: Posibilitó el estudio de las diferentes formas de autenticación así como su seguridad y complejidad. También se obtuvo un conocimiento referente a la evolución de los One-Time Password y su impacto en la seguridad de sistemas en la actualidad.
- ✓ Inducción – Deducción: Es aplicado durante el estudio, análisis y revisión de los estándares OTP que existen. Así como en el estudio de las particularidades y comportamientos de los mismos ante diferentes eventos o determinadas situaciones.

Métodos empíricos:

- ✓ Observación: Es aplicado en el estudio descriptivo de los algoritmos analizados y en la ejecución de las distintas situaciones de pruebas haciendo una detallada observación de lo sucedido.

El trabajo se encuentra estructurado de la siguiente forma:

Capítulo 1. Fundamentación teórica: Este capítulo contiene una base teórica para entender el problema planteado, en él se describen los conceptos fundamentales relacionados con aplicaciones .Net y Android, además de hacer referencia a las tecnologías relacionadas con One-Time Password. Se presentan las fases de la metodología Microsoft Solution Framework (MSF) Ágil para dar solución al problema científico.

Capítulo 2. Visión y planeación: Se presentan los escenarios y requerimientos de calidad del servicio de las aplicaciones, y el plan de iteraciones.

Capítulo 3. Desarrollo y estabilización: Se muestra la arquitectura del mecanismo, el diagrama de clases, el diagrama de despliegue de la aplicación, así como el modelo de datos y las técnicas usadas durante la implementación. Se realizan pruebas al *software* usando métodos de pruebas de caja blanca, caja negra, así como el diseño de los casos de prueba y prueba de fiabilidad a todas las aplicaciones que componen el mecanismo.

Capítulo 1: Fundamentación teórica

El presente capítulo aborda el marco teórico de la investigación. Son conceptualizados los elementos claves de la autenticación y las principales aplicaciones y estándares de One-Time Password. Serán identificadas y caracterizadas las principales tecnologías, herramientas y lenguajes que serán utilizados en el desarrollo de las aplicaciones que darán solución al objetivo propuesto, así como la metodología que guiará el proceso de desarrollo del mecanismo de autenticación.

1. Introducción

La informatización de la sociedad ha proporcionado claras mejoras pero también nuevos problemas. Muchas entidades contienen en sus ficheros de datos información personal cuyo acceso o difusión a personas no autorizadas podría perjudicar gravemente a la persona involucrada.

A medida que la sociedad va creciendo, junto con ella también lo hacen las nuevas tecnologías de comunicación, las cuales se plantean la necesidad de mantener la confidencialidad y usabilidad de la información, por eso es necesario e importante establecer los sistemas y métodos eficaces que protejan sus redes y sistemas antes eventuales amenazas, ya sean presentes o futuras.

Las contraseñas producen más de un problema de seguridad para los administradores de tecnologías de la información, pues sucede que la mayoría de los usuarios crean contraseñas bastante sencillas o anotan éstas para asegurarse de que las recuerdan. Trágicamente hoy existen muy pocos procedimientos que puedan ser seguros y eficaces para poder restablecer las contraseñas.

1.1. Principales conceptos

Seguridad informática

"La seguridad informática es una disciplina que se encarga de proteger la integridad y la privacidad de la información almacenada en un sistema informático". (Rosa, 2013)

Sistema de Administración de Identidades

Sistema de investigación que se desarrolla en el Departamento de Seguridad Digital, del Centro de Identificación y Seguridad Digital (CISED).

Autenticación

La autenticación “es el proceso de detectar y comprobar la identidad de una entidad de seguridad mediante el examen de las credenciales del usuario y la validación de las mismas consultando a una autoridad determinada”. (Microsoft, 2013)

1.2. One-Time Password

Los One-Time Password “son utilizadas en procesos de autenticación, que tienen la peculiaridad de ser válidas en una única ocasión”. El objetivo de este tipo de contraseñas es dificultar el acceso a recursos protegidos. (Safelayer Labs, 2010)

1.2.1. OTP, estándares y aplicaciones

El estudio de varias de las aplicaciones que en la actualidad utilizan OTP como forma de autenticación para aumentar su seguridad, arrojó que no existe un estándar único para la programación de dichas aplicaciones. Cada empresa productora de dispositivos o aplicaciones generadoras de OTP, utiliza algoritmos de encriptación, los cuales nunca son develados en su totalidad para evitar que los hackers actúen en contra de sus sistemas. A continuación se enuncian algunos de los estándares y aplicaciones con mayor popularidad a nivel internacional.

Google Authenticator

Google Authenticator es un *software* basado en OTP desarrollado por Google. La aplicación proporciona un número de seis dígitos, además de su nombre de usuario y contraseña para acceder a los servicios de Google. El autenticador también puede generar el código para las aplicaciones de terceros.

Google Authenticator genera una clave secreta de 80-bit para cada usuario. Esta se proporciona como una cadena de 16 caracteres o como un QR *code*[5]. El cliente crea un HMAC-SHA1[6] utilizando esta clave secreta, con los números generados y el usuario tiene un período de 30 segundos para utilizarlo. Para generar la cadena de caracteres se toma una porción de la HMAC, se extrae y se convierte en un código de 6 dígitos. (Google Authenticator, s.f.)

HOTP

De acuerdo con M'Raihi (2005), HOTP es una HMAC basada One-Time Password. Es una piedra angular de la *Initiative For Open Authentication* (OATH)[7]. HOTP se publicó como un informativo en diciembre de 2005, desde entonces, el algoritmo fue adoptado por muchas empresas en todo el mundo y se convirtió en estándar líder para la autenticación OTP basado en eventos.

HOTP se puede utilizar para autenticar a un usuario en un sistema a través de un servidor de autenticación. Además, el servidor calcula el valor de OTP posterior y envía o muestra al usuario que lo comprueba con el valor OTP posterior calculado por su token.

Según plantea M'Raihi (2005) se pueden considerar dos vías diferentes para generar los OTP, la primera a través de una generación determinista o continua, en la cual los caracteres para las futuras generaciones se derivan de una contraseña maestra, la cual puede estar implícita en el OTP actual; y la segunda mediante la utilización de una generación aleatoria en la cual las contraseñas son generadas al azar en una fase inicial y deben ser almacenadas inmediatamente. Para esta última opción es aconsejable que las contraseñas sean resguardadas durante todo su ciclo de vida.

Es importante señalar que el algoritmo HOTP no es un sustituto para el cifrado y no facilita la privacidad de la transmisión de datos. Otros mecanismos deben ser dirigidos a evitar romper la confidencialidad y la privacidad de las transacciones. De esta forma se puede concluir que este

---

[5]*Quick Response (QR) code*: código de respuesta rápida es un módulo útil para almacenar información en una matriz de puntos o un código de barras bidimensional creado por la compañía japonesa Denso Wave, subsidiaria de Toyota, en 1994.

[6]HMAC-SHA1 es un tipo de algoritmo *hash* en clave que se crea desde la función *hash* SHA1 y se utiliza como HMAC o como código de autenticación de mensajes basado en *hash*.

[7]OATH es una industria de colaboración para desarrollar una arquitectura de referencia abierta utilizando estándares abiertos para promover la adopción de autenticación fuerte.

algoritmo es tan seguro como los protocolos de autenticación que lo implementen. Por lo tanto, se requiere del uso del protocolo seguro de comunicación para la autenticación de las aplicaciones como requerimiento mínimo.

OTPW

Al referirse a este tema Kuhn (2003) planteó que OTPW es un sistema OTP desarrollado para la autenticación en Unix, sistemas operativos de Markus Kuhn. En el mismo, la contraseña real de un usuario no se transmite directamente a través de la red. Más bien, la contraseña real se combina con un pequeño conjunto de caracteres para formar un OTP. Debido a que la contraseña se utiliza una vez no pueden ser interceptadas por un *sniffer*[8] o *keylogger*[9].

OTPW es compatible con Unix y Linux (a través de módulos de autenticación conectables), y una implementación genérica de código abierto puede ser utilizada para permitir su uso en otros sistemas.

OTPW, al igual que los otros sistemas OTP, es sensible a medios de ataque si se utilizan continuamente. Esto podría ser resuelto por ejemplo con poner protocolo SSL[10], la seguridad SPKM (Simple Public-Key Mechanism) o similares al servidor y dota de seguridad la conexión entre el cliente y el servidor.

La autenticación de los clientes Unix y Linux con OTPW se logra a través de un archivo el cual se encuentra en el directorio home de cada usuario. Al requerirse de una autenticación fuerte, cada usuario debe ejecutar el programa otpw-gen, este pedirá una contraseña prefijo, que al ser introducida, es generada una lista de contraseñas estándares y mostradas al usuario. La contraseña prefijo debe ser memorizada para la autenticación, mientras que las restantes pueden ser guardadas e impresas.

[8] *Sniffer* es un programa de captura de las tramas de una red de computadoras.

[9] *keylogger* o registrador de teclas es un tipo de *software* o un dispositivo *hardware* específico que se encarga de registrar las pulsaciones que se realizan en el teclado, para posteriormente memorizarlas en un fichero o enviarlas a través de internet.

[10] El protocolo SSL es un sistema de seguridad desarrollado por Netscape y utilizado actualmente por la mayoría de empresas que comercian a través de Internet. Es un sistema de seguridad ideado para acceder a un servidor garantizando la confidencialidad de los datos mediante técnicas de encriptación modernas.

TOTP

Como sugiere M'Raihi (2011) “TOTP es una variante de OTP basada en el tiempo, es una extensión de la HMAC[11] basada en una HOTP para apoyar un factor de tiempo basado en movimiento”. Un factor en movimiento no es más que un valor que debe ser cambiado cada vez que se genera una nueva contraseña con el fin de garantizar que una contraseña diferente siempre sea generada. TOTP es un estándar de la *Internet Engineering Task Force*[12] y una piedra angular de la OATH.

Según M'Raihi, es recomendado utilizar un espacio de tiempo de 30 segundos para la generación entre un OTP y su consecutivo. Debido a la posibilidad real de desincronización entre un cliente y un servidor de validación, se recomienda que el sistema validador tenga en cuenta un límite específico para el número de veces en que un cliente realice autenticaciones fallidas y sea clasificado y/o puesto "fuera de sincronización".

Se debe tener en cuenta que cuanto más tiempo transcurra entre las generaciones de los OTP, más engorroso será lograr la resincronización automática y en algunos casos, puede no funcionar si la desviación excede el umbral permitido. Por lo que se recomienda la utilización de medidas adicionales para autenticar de forma segura y sincronizar las aplicaciones, de manera específica se debe hacer énfasis en la desviación del tiempo entre el cliente y el sistema validador.

1.2.2. Selección del estándar OTP a utilizar en la solución

Por lo anteriormente expuesto y tras un estudio comparativo, los autores de esta investigación consideraron la no utilización del estándar TOTP debido a los problemas que pudieran generarse luego del despliegue al sincronizar las aplicaciones clientes, específicamente la aplicación móvil la cual en caso de ser apagado el teléfono o que la batería del mismo se agote, debería de resincronizarse nuevamente. Además de desechar la variante de estándar OTPW, debido a que fue concebida únicamente para clientes Unix y alguna de las variantes más conocidas de Linux.

[11] *Hash Message Authentication Code* (HMAC) o código de mensaje de autenticación basado en *hash* es una especificación para el cálculo de un código de autenticación de mensaje (MAC) que implica una función criptográfica *hash* en combinación con una clave secreta.

[12] *Internet Engineering Task Force (*IETF) o Fuerza de Tareas de Ingeniería de Internet es una organización internacional abierta de normalización, que tiene como objetivo contribuir a la ingeniería de Internet, actuando en diversas áreas, como transporte, encaminamiento, seguridad.

Por lo referido anteriormente, así como la resistencia que muestra ante diferentes tipos de ataque y por considerarse el estándar líder para la autenticación OTP basado en eventos fue elegido HOTP como base para el desarrollo de las aplicaciones de la actual investigación. Dicho estándar además cuenta con gran aceptación por las empresas que lo utilizan, siendo adaptable a disímiles medios y de fácil interacción con los usuarios finales.

1.3. Tecnologías y herramientas para clientes .Net

1.3.1. Tecnología .NET

Según establece José Antonio González Seco (2006) Microsoft .NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando durante los últimos años con el objetivo de obtener una plataforma sencilla y potente para distribuir el *software* en forma de servicios que puedan ser suministrados remotamente y que puedan comunicarse y combinarse unos con otros de manera totalmente independiente de la plataforma, lenguaje de programación y modelo de componentes con los que hayan sido desarrollados.

El CISED en la actualidad desarrolla sus productos sobre esta tecnología. Siendo la presente investigación un subproceso del Módulo de Autenticación del Sistema de Administración de Identidades perteneciente a dicho centro, se desarrollará la solución que se propone haciendo uso de las facilidades que brindan las tecnologías .NET.

1.3.2. Microsoft .NET Framework 4

Al referirse a este tema Microsoft (2013) afirma que Microsoft .NET Framework 4 proporciona varias mejoras y características nuevas, entre las que se destacan adelantos en Lenguaje Común de Rutina (*Common Language Runtime*, CLR) y la biblioteca de clases base (BCL), en el rendimiento, incluida una mayor compatibilidad con equipos multi-núcleo, recolección de elementos no utilizados en segundo plano y asociación del generador de perfiles en el servidor, además de incluir algunos avances en ASP.NET, por ejemplo: más control sobre HTML, identificadores de elemento y hojas CSS personalizadas que facilitan enormemente la creación de formularios Web que admiten optimización del motor de búsqueda y son conformes a los estándares.

Estas características y mejoras serán aprovechadas en el desarrollo del mecanismo de autenticación mediante One-Time Password, que dará solución al objetivo de la presente investigación.

1.3.3. Lenguaje de programación C Sharp

Sobre este tema Microsoft (2013) plantea que C Sharp o C#, como usualmente se le conoce, “es un lenguaje de programación orientado a objetos, desarrollado y estandarizado por Microsoft como parte de su plataforma .NET”. Utiliza el modelo de objetos de la plataforma .NET haciéndolo un lenguaje caracterizado por su sencillez, seguridad y alto nivel de productividad, siguiendo el mismo patrón de los lenguajes de programación modernos. Además incluye un amplio soporte de estructuras, componentes, programación orientada a objetos, manipulación de errores que permite mayor flexibilidad en lo que se desea desarrollar. Este potente lenguaje también provee soporte para interfaces, estructuras y características de componentes orientados, como propiedades, eventos y construcciones declaradas (también llamados atributos).

Además de las características que lo convierten en el lenguaje estandarizado por la plataforma .NET y contar con las características antes mencionadas, en el desarrollo de la solución propuesta se hace uso del mismo, por el dominio presentado por los autores del trabajo sobre este lenguaje.

1.3.4. Plataforma de programación ASP.Net

ASP.NET como explica Microsoft (2013) es un *framework* para aplicaciones web desarrollado y comercializado por Microsoft. Es un modelo de desarrollo Web unificado que incluye los servicios necesarios para crear aplicaciones Web con el código mínimo. ASP.NET forma parte de .NET Framework y al codificar las aplicaciones ASP.NET tiene acceso a las clases en .NET Framework. El código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el *Common Language Runtime* (CLR), entre ellos Microsoft Visual Basic y C#. Estos lenguajes permiten desarrollar aplicaciones ASP.NET que se benefician del *Common Language Runtime*, seguridad de tipos y herencia.

Siendo la plataforma utilizada para el desarrollo web en .NET y teniendo el Módulo de Autenticación del Sistema de Administración de Identidades características que lo ubican en este campo, se hace

necesario el uso de esta plataforma para el desarrollo de la aplicación web del mecanismo de autenticación mediante One-Time Password.

1.3.5. IDE de desarrollo Visual Studio 2010 Ultimate

Como ha publicado Microsoft (2013), Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Microsoft Visual Studio 2010 Ultimate es la completa suite de herramientas, incluye aplicaciones de gestión de ciclo de vida de los equipos para garantizar resultados de calidad, además de diseño y la implementación.

Este IDE proporciona grandes facilidades en el desarrollo de un *software* como la inclusión de un editor de código que permite resaltar los errores en el código escrito, así como accesos directos hasta ellos para corregirlos rápidamente, presenta excelentes características de navegación y búsqueda dentro del código, utiliza *IntelliSense* [13] para el auto-completamiento de código disminuyendo la complicidad del trabajo y la pérdida de tiempo.

Tanto por las facilidades que provee el entorno de desarrollo integrado Microsoft Visual Studio como el conocimiento de los autores sobre el trabajo con el mismo, fue seleccionado como el IDE de desarrollo para la implementación de las aplicaciones de escritorio y web de la solución propuesta.

1.3.6. Lenguaje de modelado

Lenguaje Unificado de Modelado (UML).

[13] *IntelliSense* es la aplicación de autocompletar, mejor conocido por su utilización en Microsoft Visual Studio entorno de desarrollo integrado. Además de completar el símbolo de los nombres que el programador está escribiendo, *IntelliSense* sirve como documentación y desambiguación de los nombres de variables, funciones y métodos de utilización de metadatos basados en la reflexión.

Joaquín García (2005) refiere que el Lenguaje Unificado de Modelado es un lenguaje gráfico que especifica, construye, visualiza y documenta las partes o artefactos que constituyen la información utilizada y originada mediante un proceso de *software*, además es un lenguaje orientado a objetos.

Los principales beneficios de UML son:

- ✓ Modelar sistemas utilizando conceptos orientados a objetos.
- ✓ Establecer conceptos y artefactos ejecutables.
- ✓ Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- ✓ Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- ✓ Mejor soporte a la planeación y al control de proyectos.
- ✓ Alta reutilización y minimización de costos.

Se puede emplear de diferentes formas para dar soporte a una metodología de desarrollo de *software* pero no especifica en sí, qué metodología o proceso utilizar. UML no se puede comparar con la programación estructurada, pues no es una programación, solo diagrama la realidad de un requerimiento. Mientras que, la programación estructurada es una forma de programar como lo es la orientada a objetos, sin embargo, la orientada a objetos es un complemento perfecto de UML, pero esto no quiere decir que UML se toma únicamente para lenguajes orientados a objetos.
Se hará uso de este lenguaje para la representación de todo el ciclo de vida de las aplicaciones desarrolladas para mecanismo de autenticación mediante One-Time Password.

Visual Paradigm

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El *software* de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación. (Visual Paradigm International Ltd., 2007)
Es una herramienta CASE [14] profesional que proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. También brinda un diseño centrado en

[14]Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software*.

casos de uso y enfocado al negocio lo que genera un *software* de mayor calidad. (Visual Paradigm International Ltd., 2007)
Además de las múltiples ventajas sobre otras herramientas de modelado, los autores de la presente investigación poseen conocimientos del uso de la misma por lo que fue la herramienta CASE seleccionada.

1.3.7. PostgreSQL

Es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD. (The PostgreSQL Global Development Group, 1996 - 2013)

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyada por organizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

Este potente gestor de base de datos implementa el estándar SQL92/SQL99 y soporta distintos tipos de datos, al mismo tiempo que brinda soporte para los tipos de datos base soporta además datos de tipo fecha, monetarios, elementos gráficos y cadenas de bits, permite la creación de tipos propios e incorpora una estructura de datos *arrays*.

Al reflexionar en este sentido Alex Vivas (2010), resalta que las características que presenta este gestor de base de datos son las siguientes:

- ✓ Alta concurrencia: Mediante un sistema denominado MVCC (Acceso concurrente multiversión) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo "*commit*". Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

- ✓ Amplia variedad de tipos nativos: PostgreSQL provee nativamente soporte para números de precisión arbitraria, texto de largo ilimitado, figuras geométricas, direcciones IP (IPv4 e IPv6), bloques de direcciones estilo CIDR[15], direcciones MAC[16] y *arrays*.
- ✓ Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL.
- ✓ Claves ajenas: también denominadas llaves ajenas o claves foráneas (*foreign keys*).
- ✓ Disparadores (*triggers*): Un disparador o *trigger* se define como una acción específica que se realiza de acuerdo a un evento, cuando este ocurra dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica.
- ✓ Integridad transaccional.
- ✓ Herencia de tablas.
- ✓ Tipos de datos y operaciones geométricas.
- ✓ Soporte para transacciones distribuidas. Permite a PostgreSQL integrase en un sistema distribuido formado por varios recursos (p.ej, una base de datos PostgreSQL, otra Oracle, una cola de mensajes IBM MQ JMS y un ERP SAP) gestionado por un servidor de aplicaciones donde el éxito ("*commit*") de la transacción global es el resultado del éxito de las transacciones locales.
- ✓ Soporta el uso de índices, vistas y procedimientos almacenados en múltiples lenguajes.
- ✓ Tiene interfaces de programación nativos para C/ C++, Java, Perl, Python, Ruby, Tcl, ODBC, entre otros.
- ✓ Es multiplataforma, funciona en todos los sistemas operativos importantes, incluyendo Linux, UNIX y Windows.

Por lo cual se decide utilizar este potente sistema de gestión de base de datos para soportar la información del sitio web dada la funcionalidades y características de este SGBD.

---

[15] *Classless Inter-Domain Routing* o CIDR (en español, enrutamiento entre dominios sin clases) se introdujo en 1993 por IETF y representa la última mejora en el modo de interpretar las direcciones IP. Su introducción permitió una mayor flexibilidad al dividir rangos de direcciones IP en redes separadas.

[16] La dirección MAC (siglas en inglés de *media access control*; en español "control de acceso al medio") es un identificador de 48 bits (6 bloques hexadecimales) que corresponde de forma única a una tarjeta o dispositivo de red. Se conoce también como dirección física, y es única para cada dispositivo.

1.3.8. ER/Studio

El ER/Studio es una herramienta para el modelado de datos, que ayuda a las empresas o instituciones que la utilicen a documentar y re-utilizar los activos de datos. Con soporte completo a las bases de datos, los arquitectos de las mismas tienen en poder de fácilmente realizar ingeniería a la inversa. Analiza y optimiza las bases de datos existentes.

Como es citado en (Embarcadero Technologies, 2009) esta herramienta:

- ✓ Documenta y mejora las bases de datos.
- ✓ Mejora la consistencia de los datos.
- ✓ Comunica eficientemente los modelos.
- ✓ Traza los orígenes de los datos y mejora la integración y exactitud.

Además también tiene la posibilidad de creación de diagramas de una forma rápida y clara. Igualmente es posible cambiar el estilo de las líneas, los colores, tipos de letra, niveles de acercamiento, y modelos de despliegue. Por estas características y por ser la herramienta de modelado de datos conocida por los autores se utilizará la misma en la solución que se propone.

1.4. Tecnologías y herramientas para cliente Android

1.4.1. Definición de Android

Al referirse a este aspecto Charles (1999) puntualiza que es una plataforma de *software* para dispositivos móviles que incluye un sistema operativo, *middleware*[17] y aplicaciones base. Los desarrolladores pueden crear aplicaciones para la plataforma usando el SDK[18] de Android. Estas aplicaciones se escriben utilizando el lenguaje de programación Java y se ejecutan en Dalvik[19], una máquina virtual personalizada que se ejecuta en la parte superior de un núcleo de Linux.

[17] *Middleware* es un *software* que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, *software*, redes, *hardware* y/o sistemas operativos. Éste simplifica el trabajo de los programadores en la compleja tarea de generar las conexiones que son necesarias en los sistemas distribuidos.

[18] SDK (siglas en inglés de *Software Development Kit*): un conjunto de herramientas de desarrollo de *software* que le permite al programador crear aplicaciones para un sistema concreto.

[19] Dalvik: es la máquina de proceso virtual (VM) en el sistema operativo Android de Google. Es el *software* que ejecuta las aplicaciones en los dispositivos Android.

Esta será la plataforma donde los autores del trabajo desarrollarán la aplicación móvil del mecanismo de autenticación con One-Time Password.

1.4.2. Arquitectura de Android

Google adquirió la compañía Android Inc. y desarrolló la plataforma Android, posteriormente en acuerdo con el consorcio Open Handset Alliance decide promocionar los estándares de códigos abiertos para dispositivos móviles. (Sama, Androiddeity, 2011)

Entre las características notables se tiene que la plataforma es adaptable a pantallas más grandes, VGA, biblioteca de gráficos 2D y 3D basada en las especificaciones de OpenGL; utiliza SQLite para el almacenamiento de datos; emplea SMS y MMS como vías de mensajería; soporta múltiples tecnologías de conectividad (GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE y WiMAX); emplea varios formatos multimedia (AMR, MP3, MIDI , WAV, JPEG, PNG, GIF, BMP…); incluye además soporte para *hardware* adicional (cámaras de fotos, de vídeo, pantallas táctiles, GPS (*Global Positioning System* o Sistema de Posicionamiento Global), acelerómetros, giroscopios, magnetómetros, termómetros, sensores de proximidad y de presión). La Figura 1 muestra los componentes principales de la arquitectura Android.

Para realizar aplicaciones sobre la plataforma Android se instala el plugin ADT y el SDK en el IDE (Integrated Development Environment o Entorno de Desarrollo Integrado) de desarrollo. Un SDK de Android contiene las librerías, el emulador, documentación, ejemplos de código, tutoriales y una versión de la plataforma. Se usa el lenguaje de programación Java.

Figura 1- Arquitectura de Sistemas Operativos Android. Fuente: *(Sama, Androiddeity, 2011)*

Tecnología utilizada:

- ✓ **SQLite**: SQLite es una librería de *software* en lenguaje C, que implementa un sistema autónomo, sin servidor, sin necesidad de configuración y con el motor transaccional de base de datos SQL. El código fuente de SQLite es de dominio público.

1.4.3. Lenguaje de programación Java

Exes al profundizar sobre este particular, explica que Java es un lenguaje orientado a objetos. También posee otras características muy importantes:

Es un lenguaje que es compilado, generando ficheros de clases compilados, pero estas clases compiladas, son en realidad interpretadas por la máquina virtual de java. Siendo la máquina virtual de java la que mantiene el control sobre las clases que se estén ejecutando. (Exes, 2004)

Es un lenguaje multiplataforma: El mismo código java que funciona en un sistema operativo, funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual java.

Es un lenguaje seguro: La máquina virtual, al ejecutar el código java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.

Es el lenguaje utilizado en la implementación de aplicaciones para Android, por lo que fue necesario la utilización del mismo en el desarrollo de la aplicación móvil para Android del mecanismo de autenticación mediante One-Time Password.

1.4.4. Herramientas

Para desarrollar la aplicación para el dispositivo móvil se utilizaron diversas herramientas de *software* libre, ya sean IDEs, SDKs, simuladores o emuladores.

- ✓ **Eclipse IDE:** Entorno de desarrollo multiplataforma. Permite utilizar varios lenguajes de programación como C++, Python, Java. Consume algunos recursos del sistema en tiempo de ejecución.
- ✓ **Android SDK:** provee las herramientas y APIs necesarias para desarrollar aplicaciones sobre la plataforma de Android utilizando el lenguaje de programación Java.
- ✓ **Android Development Tools (ADT):** es un *plugin* de desarrollo para el IDE de Eclipse. Extiende las capacidades de Eclipse para permitirte configurar rápidamente nuevos proyectos de Android, crear una interfaz de usuario de la aplicación, añadir componentes en función de la API de Android Framework, depurar las aplicaciones mediante las herramientas de Android SDK e incluso exportar APKs con (o sin) firma.
- ✓ **Máquina Virtual de Java:** es un intérprete que convierte el *bytecode* compilado de Java en el código de máquina nativo en que debería ejecutarse, aunque es más correcto decir que es un emulador de la ejecución de un código nativo. La tecnología de máquinas virtuales se ha distinguido de la tecnología de interpretación básicamente por el nivel en que se realiza la interpretación. Los intérpretes trasladan un "código de bit" nativo directamente a llamadas del sistema o instrucciones de máquina, pero la emulación consiste en permitir una arquitectura de máquina intermedia.

1.5. Servicios web

En este tema Hjalmar Ruiz Tückler (2012), puntualiza que un servicio web es una aplicación que puede ser descripta, publicada, localizada e invocada a través de una red, generalmente Internet. Combinan los mejores aspectos del desarrollo basado en componentes y la Web.

Al igual que los componentes, los servicios web son funcionalidades que pueden ser reutilizados sin preocuparse de cómo fueron implementados. A diferencia de la actual tecnología de componentes, no son accedidos por medio de protocolos específicos del modelo de objetos; sino que son accedidos utilizando protocolos web como ser HTTP y XML[20].

Las interfaces de los servicios web está definida en términos de los mensajes que el mismo acepta y retorna, por lo cual los consumidores de los servicios web pueden ser implementados en cualquier plataforma y en cualquier lenguaje de programación, solo tiene que poder crear y consumir los mensajes definidos por las interfaces de los servicios web.

Tecnologías asociadas

El modelo de servicios web está basado en ciertas tecnologías emergentes que son el resultado del trabajo de varias compañías y organizaciones entre las cuales se destacan IBM y Microsoft. Estas tecnologías son SOAP, WSDL y UUDI.

SOAP (Simple Object Access Protocol)

Los argumentos expuestos por Hjalmar Ruiz Tückler (2012), muestran que SOAP es un protocolo para el intercambio de información en un ambiente descentralizado y distribuido. Es el protocolo más utilizado para realizar el intercambio de información en el modelo de servicios web. Está basado en XML y potencialmente puede ser utilizado en combinación con una variedad de protocolos de comunicación, siendo el más utilizado HTTP. Por lo tanto se utiliza HTTP para transportar la información, y XML para representar la misma.

[20] *eXtensible Markup Language* (XML), surgió como un lenguaje de marcado para sustituir a HTML, es un sistema para definir, validar y compartir formatos de documentos en la Web.

WSDL (Web Services Description Language)

Hjalmar Ruiz Tückler (2012) al investigar sobre este tema, explica que WSDL es un lenguaje basado en XML que se utiliza para describir un servicio web.

Un archivo con formato WSDL provee información de los distintos métodos (operaciones) que el servicio web brinda, muestra cómo accederlos y que formatos deben de tener los mensajes que se envían y se reciben. Es un contrato entre el proveedor del servicio y el cliente, en el cual el proveedor se compromete a brindar ciertos servicios solo si el cliente envía un requerimiento con determinado formato. Es el documento principal a la hora de documentar un servicio web, pero puede no ser el único. En la mayoría de los casos es conveniente que esté acompañado por un documento escrito en lenguaje natural que brinde información de qué es lo que hace cada uno de los métodos brindados por el servicio web, un ejemplo de ello son los mensajes SOAP que espera y responde el servicio.

En forma resumida Hjalmar Ruiz Tückler (2012) plantea que un archivo WSDL describe lo siguiente:

- ✓ Mensajes que el servicio espera y mensajes que el servicio responde.
- ✓ Protocolos que el servicio soporta.
- ✓ A dónde mandar los mensajes.

UDDI (Universal Description, Discovery and Integration)

Hjalmar Ruiz Tückler (2012) al profundizar sobre este particular, explica que UDDI es un proyecto inicialmente propuesto por Ariba, Microsoft e IBM; es un estándar para registrar y descubrir servicios web. La idea es que las distintas empresas registran su información acerca de los servicios web que proveen para que puedan ser descubiertas y utilizadas por potenciales usuarios.

La información es ingresada al registro de empresas UDDI, un servicio lógicamente centralizado, y físicamente distribuido a través de múltiples nodos los cuales replican su información en forma regular. Una vez que una empresa se registra en un determinado nodo del registro de empresas UDDI la información es replicada a los otros nodos y queda disponible para ser descubierta por otras empresas.

El estudio de estas tecnologías fue muy útil para el posterior desarrollo del servicio web que brindará el mecanismo de autenticación que se propone para la solución.

1.6. Metodología de desarrollo

Microsoft Solution Framework Agile

Se seleccionó esta metodología de desarrollo de *software* porque se caracteriza por ser de planificación adaptable a los cambios y orientada a las personas. Su proceso introduce ideas importantes del *software* ágil, junto con los principios y prácticas de MSF (Microsoft Solution Framework), además admite una estrategia que utiliza múltiples iteraciones y un enfoque para la construcción de aplicaciones que se basa en escenarios.

Esta metodología también incorpora prácticas para el manejo de la calidad del servicio (el rendimiento y la seguridad) y facilita la automatización y la orientación que se necesita para apoyar el equipo de trabajo, incluyendo la gestión de configuración y de proyectos.

Al profundizar sobre este tema Jonnathan De La Barra (2010) explica que el desarrollo ágil de *software* es un marco de trabajo conceptual de la ingeniería de *software* que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto. El proceso de desarrollo en MSF consta de 5 fases, las cuales complementan el ciclo de desarrollo de *software* en esta metodología.

Figura 2- Modelo de proceso de desarrollo de aplicaciones MSF. Fuente: Elaboración propia.

Fase 1: Visión. En esta fase el equipo y el cliente definen los requerimientos del negocio y los objetivos generales del proyecto. La fase culmina con el hito Visión y Alcance aprobados.

Fase 2: Planeación. Durante la fase de planeación el equipo crea un borrador del plan maestro del proyecto, además de un cronograma y de la especificación funcional del proyecto. Esta fase culmina con el hito Plan del proyecto aprobado.

Fase 3: Desarrollo. Esta fase involucra una serie de liberaciones internas del producto, desarrollados por partes para medir su progreso y para asegurarse que todos sus módulos o partes están sincronizados y pueden integrarse. La fase culmina con el hito Alcance completo.

Fase 4: Estabilización. Esta fase se centra en probar el producto. El proceso de prueba hace énfasis en el uso y el funcionamiento del producto en las condiciones del ambiente real. La fase culmina con la liberación terminada aprobada.

Fase 5: Implantación: En esta fase el equipo implanta la tecnología y los componentes utilizados por la solución, estabiliza la implantación, apoya el funcionamiento y la transición del proyecto, y obtiene la aprobación final del cliente. La fase termina con el hito Implantación completa.

1.7. Conclusiones del capítulo

Una vez concluido el presente capítulo, el análisis realizado sobre los temas relacionados con la autenticación, la caracterización de aplicaciones y de estándares de One-Time Password, brindaron un mayor conocimiento y dominio sobre el campo de acción en que se desarrollará la solución que se propone.

Por otra parte fue realizada la selección de las tecnologías, herramientas y lenguajes que serán utilizados en el desarrollo de las aplicaciones que tendrá el mecanismo de autenticación que se propone implementar. De esta manera fueron sentadas las bases teóricas para darle solución al objetivo planteado en la presente investigación.

Capítulo 2: Visión y planeación

En el presente capítulo se hace una propuesta del mecanismo de autenticación con One-Time Password, para lo cual se estudiaron los términos relacionados con el objeto de estudio y el campo de acción. Para una mayor compresión de cómo estará estructurado el mecanismo se realiza una descripción del subproceso de autenticación con OTP. Además se especifican y describen los escenarios. Consecutivamente se identifican los requerimientos de calidad de servicio.

2.1. Introducción

En el proceso de desarrollo de un *software* es necesario explicar algunos conceptos y requerimientos de una forma sencilla y clara para cualquier persona que esté implicada en el desarrollo. La metodología del marco de trabajo de solución de Microsoft (Microsoft Solution Framework) provee una serie de actividades y artefactos para llevar una idea del proceso de desarrollo del *software* en su ciclo de vida.

Esta metodología plantea obtener una visión precisa de lo que se desea desarrollar y propone realizar una planificación que guíe al grupo de trabajo hacia la construcción exitosa del mecanismo de autenticación.

2.2. Descripción del subproceso de autenticación con OTP

El mecanismo que se prevé desarrollar será el encargado de la administración de los OTP para la autenticación a través del Sistema de Administración de Identidades. Para el funcionamiento del mismo se hace necesaria primeramente una gestión de los usuarios con acceso a dicha aplicación y automáticamente serán generadas las semillas y las muestras asociadas a cada usuario. Una semilla es una combinación de caracteres aleatorios (números y letras mayúsculas) que se utilizan como enlace entre el usuario y su OTP, constituyendo la base para la generación de los correspondientes OTP. Una muestra de OTP, es un OTP en cuestión y se diferencia de estos en su forma de generación.

En el momento en que cada cliente tengan las aplicaciones desktop de Windows y/o móvil de Android generarán, luego de haber introducido sus semillas, sus propios OTP. Al proceder a la autenticación a través del Sistema de Administración de Identidades, este último realizará la gestión vía web de los OTP, mediante la consulta de un servicio web. Este servicio web dará una

respuesta verdadera en 3 casos, el primero de ellos cuando el OTP introducido coincida con el OTP que posee el usuario, en el segundo, cuando coincida con una de las muestras que posee cada usuario y por último, cuando el OTP introducido sea uno de los próximos 10 OTP generados a partir del actual, dando la posibilidad de desechar 9 OTP intermedios a partir del actual.

Para finalizar, el Sistema de Administración de Identidades se encargará del bloqueo de los usuarios en dependencia de las especificaciones introducidas por los administradores, sugiriéndose sean 3 intentos fallidos. La principal funcionalidad en la interacción de este mecanismo corresponde a la generación de los OTP, ya que todas las aplicaciones deben generar los mismos OTP sin comunicación entre ellas, basándose únicamente en su semilla y algoritmos criptográficos o funciones resumen. Es recomendado que los usuarios finales utilicen de forma indistinta cada una de las aplicaciones para evitar desincronizaciones de estas.

Figura 3- Descripción del subproceso de autenticación con OTP. Fuente: Elaboración propia.

2.2.1. Definición de personas

Los argumentos expuestos en Microsoft (2013), describen en una persona las habilidades típicas, necesidades, deseos, hábitos de trabajo, las tareas y los antecedentes de un conjunto determinado de usuarios, recogiendo así los datos reales que describen las características importantes de un grupo de usuarios en particular en un personaje de ficción. De esta manera, es más fácil hablar y razonar sobre el grupo de usuarios, ya que podemos relacionar, comprender que son las personas con más facilidad de lo que puede hacer un grupo.

Cada vez que nos referimos a una persona, nos sentimos como si estuviéramos frente a un individuo, pero estamos en efecto frente al grupo que la persona representa.

Tabla 1- Definición de personas. Fuente: Elaboración propia.

Administrador	Es el encargado de realizar todas las actividades relacionadas con la autenticación y el que administra la base de datos del mecanismo de autenticación.
Usuario	Usa su sesión para autenticarse y acceder a los recursos protegidos.

2.3. Fase de planeación

En esta fase del proyecto se determina la planificación. Los principales artefactos de esta fase son los escenarios y los requerimientos de calidad de servicios que sirven de guía para todo el proceso de desarrollo.

2.3.1. Listados de escenarios y requerimientos de calidad del servicio

A continuación se listan los escenarios que para el desarrollo de la aplicación fueron enumerados. Los mismos estarán divididos en escenarios para el Servidor así como escenarios para las aplicaciones Cliente (Windows y Android).

Servidor

- **Administrar usuario.**

ES01 – Crear usuario.

ES02 – Modificar usuario.

ES03 – Eliminar usuario.

ES04 – Mostrar usuario.

ES05 – Buscar usuario.

- **Administrar semilla.**

ES06 – Generar semilla.

ES07 – Actualizar semilla.

ES08 – Eliminar semilla.

ES09 – Mostrar semilla.

- **Administrar muestra OTP.**

ES10 – Generar muestras de OTP.

ES11 – Actualizar muestras de OTP.

ES12 – Eliminar muestras de OTP.

ES13 – Mostrar muestras de OTP.

ES14 – Guardar muestra de OTP.

- **Escenarios adicionales.**

ES15 – Comprobar OTP.

ES16 – Asociar semilla a usuario.

ES17 – Integrar mecanismo de autenticación al Sistema de Administración de Identidades.

Clientes

ESC18 – Actualizar semilla.

ESC19 – Guardar semilla.

ESC20 – Generar OTP (*Hash*).

Requerimientos de calidad del servicio.

- **De *software*.**

RCS01 – Utilizar sistema operativo Windows Server 2003 o superior.

RCS02 –Trabajar con la Máquina Virtual de Java versión 1.3 o superior.

- **De diseño e implementación.**

RCS03 –Se utilizará lenguaje de programación C# para la aplicación Desktop de clientes Windows.

RCS04 – Se usará el lenguaje de programación Java para la aplicación de clientes Android.

RCS05 – Se hará uso del lenguaje de programación ASP.Net para aplicación web.

RCS06 – Se empleará el IDE de desarrollo Visual Studio 2010 para el desarrollo de aplicaciones web y desktop de clientes Windows.

RCS07 – Se utilizará el IDE de desarrollo Eclipse para el desarrollo de la aplicación de clientes Android.

RCS08 – La plataforma utilizada para la aplicación web será Internet Information Service 7.0 o superior.

- **De apariencia e interfaz externa.**

RCS09 – Diseño ajustable a la resolución de pantalla.

RCS10 – Diseño que facilita la interacción con el usuario, acorde a los patrones de diseño del Sistema de Administración de Identidades.

- **De seguridad.**

RCS11 – Accesible desde cualquier computadora del CISED.

RCS12 – El acceso al sistema se realizará con autenticación contra el LDAP[21] de la UCI.

- **De soporte.**

RCS13 – Servidor de 1 GB de RAM como mínimo con sistema operativo Windows Server 2003 o superior.

RCS14 – Clientes móviles con sistema operativo Android.

RCS15 – Tiempo máximo de respuesta 3 segundos para todas las aplicaciones.

2.3.2. Plan de iteración

Para realizar una adecuada planificación, es de gran importancia hacer una estimación del tiempo de cada uno de los escenarios. En relación con la prioridad que tengan, se decide cuáles de ellos se desarrollarán en las primeras iteraciones, pues las funcionalidades críticas del mecanismo deben ser codificadas en las iteraciones más tempranas.

MSF define las iteraciones como un período fijo de tiempo para programar tareas. Una iteración consta de un período aproximadamente de entre dos y seis semanas. Dichas iteraciones generalmente son numeradas consecutivamente y siguen una a otra de manera continua.

Dada la fecha de entrega orientada se ha tomado en consideración la realización de dos iteraciones, las cuales tomarían un tiempo de doce semanas aproximadamente para su total desarrollo.

Iteración #1: Se implementarán los escenarios de mayor complejidad, siendo necesario un total de diez semanas.

Iteración #2: Se implementarán los escenarios de complejidad media, necesitando un total de dos semanas.

Tabla 2- Plan de iteración. Fuente: Elaboración propia.

Iteración	No	Nombre del escenario	Complejidad	Duración

[21] *Lightweight Directory Access Protocol* o Protocolo Ligero de Acceso a Directorios.

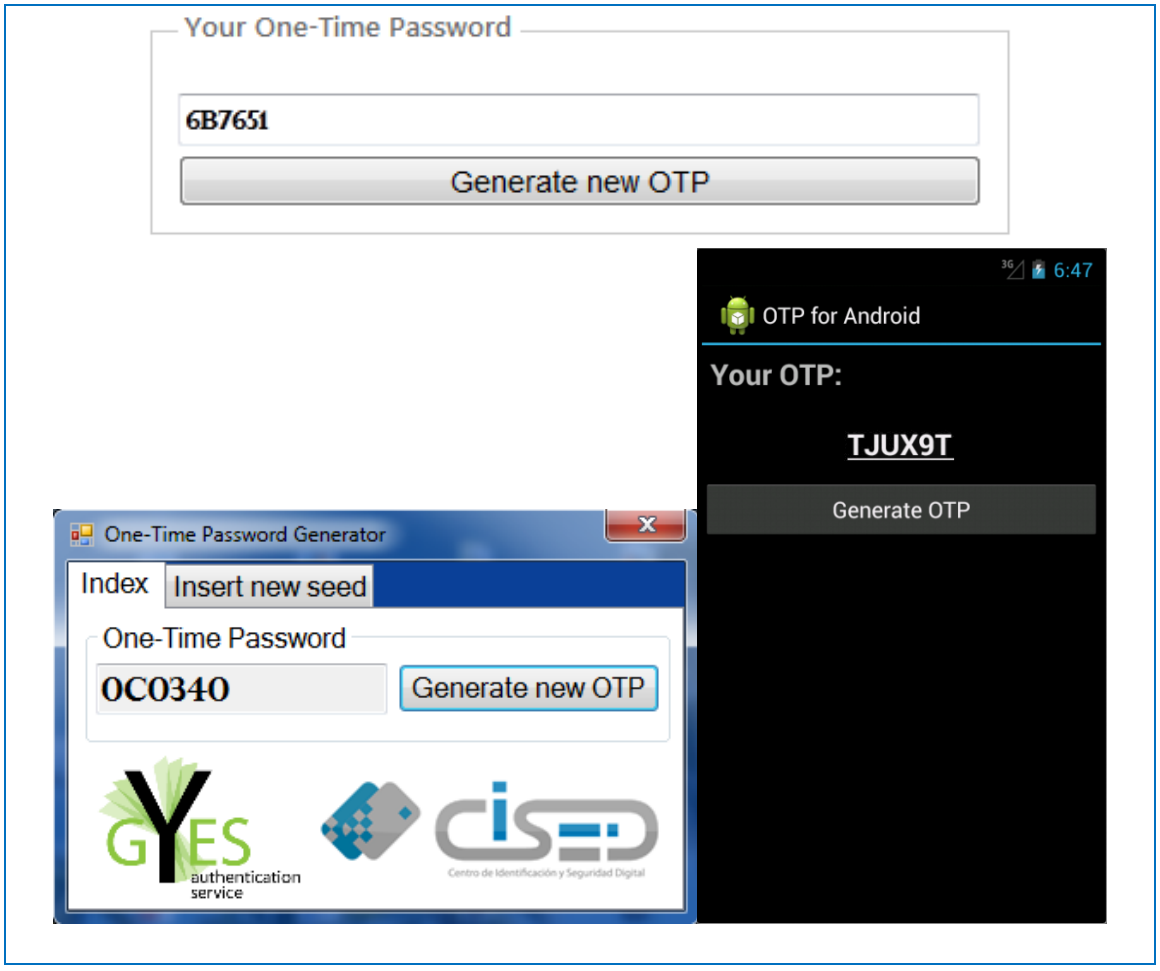
Iteración #1	1	Administrar usuario.	3	10 semanas.
	2	Administrar semilla.		
	3	Administrar muestra de OTP.		
	4	Actualizar semilla.		
	5	Guardar semilla.		
	6	Generar OTP (*Hash*).		
Iteración #2	7	Comprobar OTP.	2	2 semanas.
	8	Asociar semilla a usuario.		
	9	Integrar mecanismo de autenticación al Sistema de Administración de Identidades.		

2.3.3. Descripción de los escenarios

Un escenario es una sola trayectoria de interacción con el usuario a través del sistema. A medida que la persona trata de llegar a una meta, el escenario registra los pasos específicos en el intento de alcanzar ese objetivo (Microsoft, 2013). A continuación se muestra un ejemplo de unos de los escenarios del mecanismo de autenticación.

Tabla 3- Generar OTP (*Hash*). Fuente: Elaboración propia.

Nombre del escenario: Generar OTP (*Hash*).		**Identificador:** 3
Objetivo del escenario: Generar el OTP.		
Persona: Usuario.		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: Debe existir el usuario con su semilla en el sistema, y además debe estar autenticado.		
Descripción: Con la semilla asignada a un usuario, se genera el OTP a partir de un algoritmo *Hash* y la introducción de caracteres consecutivos.		
Validaciones: Verificar la existencia del usuario con su semilla correspondiente.		
Prototipo de interfaz de usuario:		



Ver Anexo 1.

2.4. Conclusiones del capítulo

En el desarrollo del presente capítulo fue realizada la descripción del subproceso de autenticación proporcionando una visión clara de los conceptos que interactúan en el mecanismo de autenticación con OTP.

La especificación de los escenarios realizada permitió una mejor descripción de las funcionalidades brindadas por las aplicaciones que forman parte del mecanismo de autenticación con OTP. Fueron planificadas las iteraciones para la realización de cada escenario facilitando una mejor estimación de su prioridad y del tiempo a dedicar a los mismos. Se realizó la descripción de los requisitos de calidad del servicio esclareciendo de esta forma del ambiente de desarrollo y el buen funcionamiento de las aplicaciones.

Una vez realizado el diseño del mecanismo de autenticación con One-Time Password es posible realizar la implementación exitosa de las aplicaciones que se proponen para darle cumplimiento al objetivo de la presente investigación.

Capítulo 3: Desarrollo y estabilización

El presente capítulo constituye la parte más extensa e importante de la investigación. En el mismo se les da seguimiento a las fases de desarrollo y de estabilización del producto. La primera de ellas contiene los diagramas necesarios para la correcta comprensión del lector sobre el funcionamiento del mecanismo de autenticación con OTP. En la fase de estabilización se realizan las pruebas de calidad de las aplicaciones para demostrar su correcto funcionamiento.

3.1. Introducción

Las fases de desarrollo y estabilización de la metodología MSF generan un conjunto de artefactos que facilitan el desarrollo del *software*. En el presente capítulo será definida la arquitectura a utilizar, se especificarán los patrones de diseño para la implementación de las aplicaciones. Además se modelarán y describirán las clases principales; se construirá el modelo de datos, y para verificar el correcto funcionamiento de la solución se realizarán un conjunto de pruebas tanto a la interfaz como a las funcionalidades de las aplicaciones.

3.2. Fase de desarrollo

3.2.1. Especificación de la arquitectura a utilizar

Sobre este particular Etcheverry (2010) considera que el primer paso importante en la fase de desarrollo es la especificación de la Arquitectura de Software, la cual a manera de concepto es: la organización fundamental de un sistema representada en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución. La misma involucra un conjunto de decisiones significativas acerca de la organización del sistema, selecciona sus elementos estructurales y sus interfaces, así como su comportamiento. También involucra funcionalidad, usabilidad, tolerancia a cambios, rendimiento, reutilización y aspectos estéticos.

Los Patrones Arquitectónicos son los que definen la estructura de un sistema, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema.

Arquitectura 3 capas

Para el desarrollo de todas las aplicaciones se utilizará la arquitectura 3 capas. El objetivo principal de la misma es que las distintas lógicas presentes en la aplicación se separen y posean estructuras bien definidas.

Algunas de las ventajas de utilizar este patrón indicado por Rodríguez (2010) son:

- ✓ **Mejoras en las posibilidades de mantenimiento,** debido a que cada capa es independiente de la otra, los cambios o actualizaciones pueden ser realizados sin afectar la aplicación como un todo.
- ✓ **Escalabilidad,** ya que las capas están basadas en diferentes etapas, el escalamiento de la aplicación hacia afuera es razonablemente sencillo.
- ✓ **Flexibilidad,** como cada capa puede ser manejada y escalada de forma independiente, la flexibilidad se incrementa.
- ✓ **Disponibilidad,** las aplicaciones pueden aprovechar la arquitectura modular de los sistemas habilitados usando componentes que escalan fácilmente lo que incrementa la disponibilidad.

Capa de Presentación: es la única que interactúa directamente con el usuario presentándole el sistema, permitiendo el intercambio de información entre ambos. Contiene una interfaz gráfica que posibilita capturar los datos insertados por el cliente, además de mostrarle los resultados de sus peticiones y los estados de la aplicación. Esta capa solo interactúa con la capa de negocio, que es su inferior inmediata. (Rodríguez, 2010)

Capa de Negocio: Esta es la que contiene los procesos a realizar con la información recibida desde la capa de presentación, las peticiones que el usuario ha realizado, y responsabilizándose de que se le envíen las respuestas adecuadas a la capa de presentación.

También conocida como lógica de negocio, es la que recibe las peticiones de la capa de presentación y le envía las respuestas tras el proceso. Aquí se realiza la mayor parte del procesamiento de la información del dominio de la aplicación, se ejecutan cálculos sobre los datos de entrada o almacenados, se validan los contenidos provenientes de la capa superior y se ejecutan los algoritmos específicos del programa. (Rodríguez, 2010)

Capa de Datos: Esta es la capa donde se almacenan los datos. Mediante la capa de negocio se puede ofrecer, modificar, almacenar, borrar y recuperar datos, mediante el gestor de bases de datos que la aplicación requiera.

En consonancia con los razonamientos de Ferrás (2012) podemos afirmar que es la encargada de intercambiar con otros sistemas, solo mediante ella se puede acceder a los recursos obtenidos de terceras aplicaciones. Contiene uno o más gestores de base de datos. Esta capa interactúa con la capa de negocio, recibe sus solicitudes de almacenamiento o recuperación de información, y envía la respuesta a las peticiones.

Con el objetivo de exponer claramente el funcionamiento de la propuesta arquitectónica para el desarrollo de las aplicaciones, se presentan el siguiente diagrama que muestra la estructura de la arquitectura:

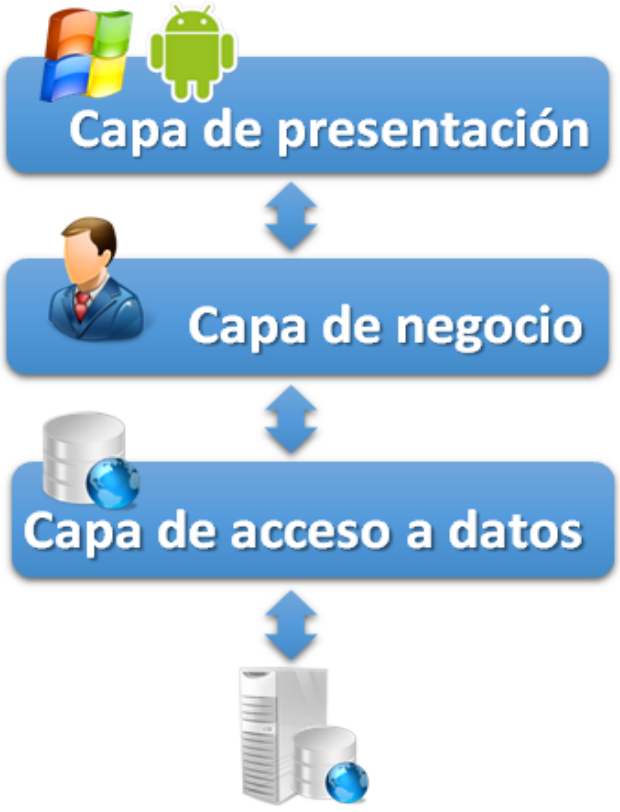


Figura 4- Diagrama general de la estructura de la arquitectura. Fuente: Elaboración propia.

A continuación se describen los elementos de cada capa:

1. **Capa de Presentación.**

a) Componentes de interfaz de usuario.

- ✓ Los elementos de formularios tienen diseños nativos de los sistemas operativos utilizados (Windows y Android), agregando botones, etiquetas, selectores.
- ✓ En los contenedores de texto se posibilitará la captura de cadenas de caracteres insertadas por el usuario, por lo general en forma de texto plano.
- ✓ Los elementos de multimedia permitirán la visualización de imágenes acreditativas al sistema en desarrollo.

b) Componentes de proceso de interfaz de usuario.

- ✓ Las actividades van a representar la capa de presentación de toda aplicación, por ejemplo, una pantalla que el usuario ve. Una aplicación puede tener varias actividades y se puede cambiar entre ellas en tiempo de ejecución de la aplicación.

2. Capa de Negocio.

a) Componentes lógicos de flujos de trabajo.

- ✓ Las clases para los flujos de trabajo se encargarán de ejecutar las tareas propias de la aplicación, en ellas se van a realizar los cálculos, que se validarán y se convertirán al formato correcto los datos provenientes de la capa de presentación.
- ✓ En los servicios se realizarán tareas propias del sistema operativo en un segundo plano, sin ofrecer una interfaz de usuario.
- ✓ El receptor de mensajes va a recibir los mensajes del sistema y las solicitudes implícitas, este se puede utilizar para responder a condiciones cambiantes en el sistema.

3. Capa de Acceso a Datos.

a) Componentes de gestión de datos.

- ✓ Las clases para el acceso a contenidos encapsularán los datos que se compartirán entre aplicaciones para tener control sobre cómo se accede a la información.
- ✓ Las consultas a la base de datos permitirá insertar, eliminar y modificar datos contenidos en las tablas de las bases de datos a las que accede la aplicación.

3.2.2. Patrones de diseño

Un patrón de diseño es una solución general a un problema común en el diseño de *software*, es una descripción o patrón de cómo resolver un problema y que puede ser usado en diferentes situaciones. Muchas veces son más útiles con algunos lenguajes de programación y en otros casos son aplicables a cualquier lenguaje. Ayudan a los desarrolladores a la reutilización de código haciendo de manera más práctica describir ciertos aspectos de la organización de un programa. Estos indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO). (Wesley, 2003)

Según Wesley (2003) los patrones se clasifican según su propósito en:

Patrones de Creación: Tratan la creación de instancias o sobre qué objetos un objeto delegará responsabilidades.

Patrones Estructurales: Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad, describiendo así la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros.

Patrones de Comportamiento: Tratan la interacción y cooperación entre clases. Organizan, manejan y combinan comportamientos.

Para hacer un diseño eficaz se tomaron en cuenta un conjunto de patrones, que permiten dar soluciones eficientes, facilitando notablemente el trabajo posterior. A continuación se muestra una breve descripción de cada uno de ellos.

Modelo Provider

Al profundizar sobre este aspecto Schackow (2006) considera que el modelo Provider es un patrón de diseño formulado por Microsoft para usarlo en ASP.NET Starter Kits y formalizarlo en .NET versión 2.0. Este patrón puede incluirse dentro de los estructurales. Es usado para permitir a una aplicación escoger una de muchas implementaciones en la configuración de la aplicación, por ejemplo, proporciona acceso a diferentes orígenes de datos para recuperar la información del *login*, o para usar diferentes metodologías de almacenamiento como puede ser una base de datos, binaria a disco, XML, etc.

Microsoft (2005) al profundizar sobre este particular, explica que con la ayuda del modelo Provider, ASP.NET 2.0 puede ser configurado para almacenar un estado prácticamente en cualquier lugar. Algunas compañías prefieren adquirir proveedores personalizados de terceros. Otros, sin embargo, prefieren escribir el suyo propio, ya sea porque ningún proveedor adecuado está disponible fuera de la plataforma, o porque desean adaptar ASP.NET para medios de almacenamiento tradicionales.

Con la utilización de este patrón en la actual investigación, la aplicación web desarrollada es capaz de obtener los datos del usuario autenticado, la configuración y conectarse a la base de datos mediante la utilización de varias librerías. Dando la posibilidad de realizar cambios en los servidores físicos y lógicos de base de datos, y con solo cambiar dicha configuración, la aplicación está apta para su uso.

Patrón Singleton

Al referirse a este particular Schackow (2006) refiere que el patrón Singleton se utiliza cuando un desarrollador quiere que exista una única instancia de una clase dentro de una aplicación, esto generalmente ocurre cuando la instanciación de un objeto y la destrucción de una clase pueden llegar a ser complejos, y por lo tanto es posible que sólo se quiera una instancia para incurrir jamás en la sobrecarga de la construcción de objetos.

El patrón Singleton se utiliza cuando se quiere mediar en el acceso a un recurso específico con una simple instancia de objeto que bloquea el acceso al recurso, es posible implementar la sincronización lógica dentro de la instancia de objeto de modo que sólo un subproceso activo puede tener acceso al recurso a la vez.

```
public partial class Home : System.Web.UI.Page
{
    private ManagerImpl initialSolution;
    protected void Page_Load(object sender, EventArgs e)
    {
        initialSolution = new ManagerImpl();
    }
}
```

Figura 5- Ejemplo de utilización del Patrón Singleton. Fuente: Elaboración propia.

3.2.3. Diagramas de clases

En el diagrama de clases representa una agrupación lógica o física de las clases presentes en el mecanismo y sus relaciones.

La utilización de la arquitectura 3 capas permitió la división y mejor organización de las clases agrupando las mismas en dependencia de sus funcionalidades en diferentes paquetes de clases. La misma ofrece la posibilidad de interacción de una capa con la siguiente inferior y/o superior siempre y cuando la posea, encapsulando métodos y demás componentes que no son de interés del resto de las capas.

La clase _**Default.aspx.cs** será visualizada por todos los usuarios sin la necesidad de estar autenticados en el sistema. La misma proporciona un vínculo para realizar dicha operación a través del Mecanismo de Autenticación dando paso a las clases **Home.aspx.cs**, siendo esta de acceso permitido a los usuarios estándares y administradores; no siendo así con las clases **AddUser.aspx.cs, EditUser.aspx.cs, DeleteUser.aspx.cs, ListUser.aspx.cs** de acceso exclusivo a usuarios con privilegios administrativos.

De la capa de negocio cabe resaltar el método *generate_OTP* de la clase **ManagerImpl,** el cual es el encargado de la generación de los OTP y las muestras de cada usuario por diferentes vías. A continuación se muestra en la Figura 6 el diagrama de clases de la aplicación web.

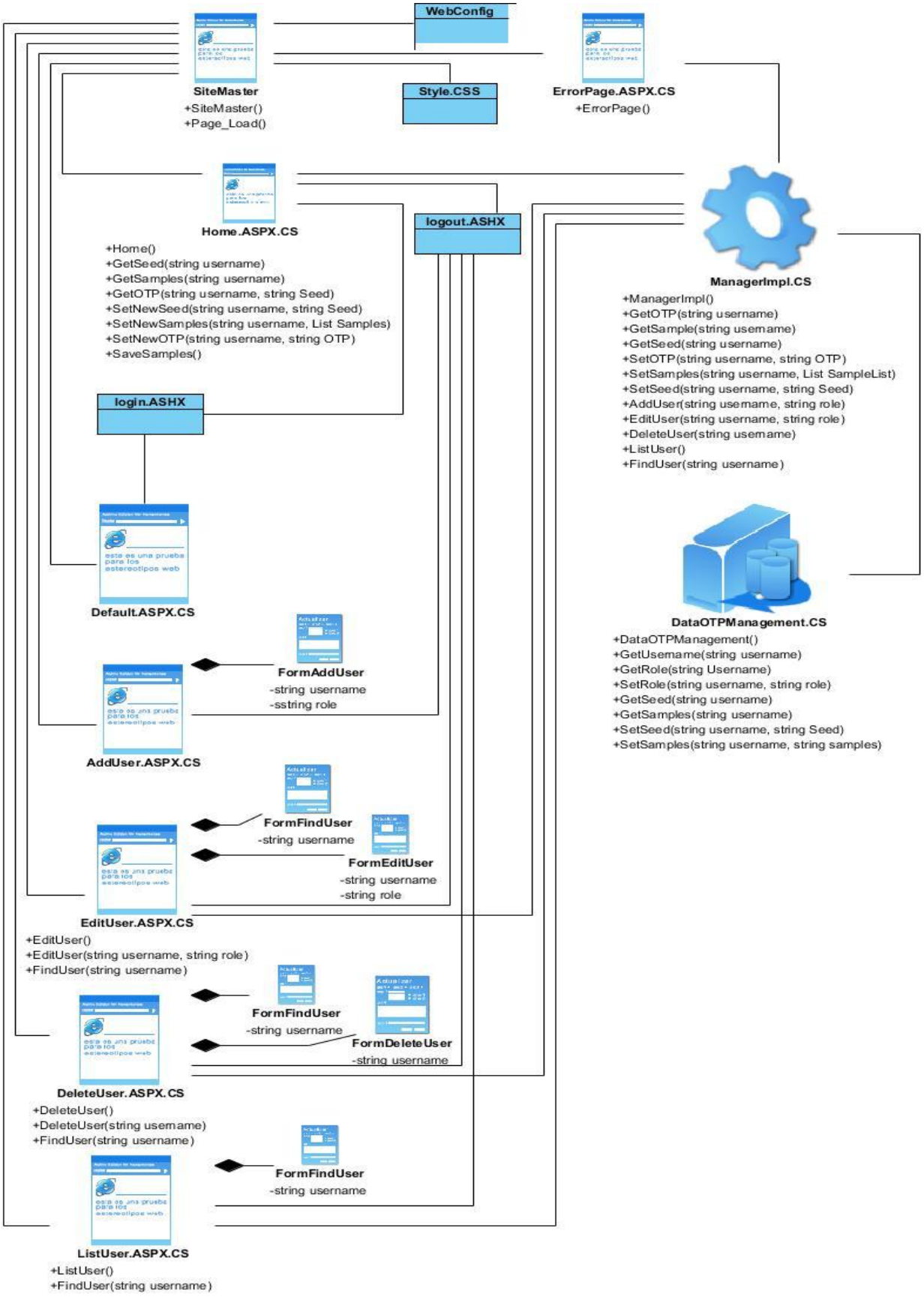

WebConfig
SiteMaster
+SiteMaster()
+Page_Load()
Style.CSS
ErrorPage.ASPX.CS
+ErrorPage()
Home.ASPX.CS
+Home()
+GetSeed(string username)
+GetSamples(string username)
+GetOTP(string username, string Seed)
+SetNewSeed(string username, string Seed)
+SetNewSamples(string username, List Samples)
+SetNewOTP(string username, string OTP)
+SaveSamples()
logout.ASHX
ManagerImpl.CS
+ManagerImpl()
+GetOTP(string username)
+GetSample(string username)
+GetSeed(string username)
+SetOTP(string username, string OTP)
+SetSamples(string username, List SampleList)
+SetSeed(string username, string Seed)
+AddUser(string username, string role)
+EditUser(string username, string role)
+DeleteUser(string username)
+ListUser()
+FindUser(string username)
login.ASHX
Default.ASPX.CS
DataOTPManagement.CS
+DataOTPManagement()
+GetUsername(string username)
+GetRole(string Username)
+SetRole(string username, string role)
+GetSeed(string username)
+GetSamples(string username)
+SetSeed(string username, string Seed)
+SetSamples(string username, string samples)
FormAddUser
-string username
-sstring role
AddUser.ASPX.CS
FormFindUser
-string username
FormEditUser
-string username
-string role
EditUser.ASPX.CS
+EditUser()
+EditUser(string username, string role)
+FindUser(string username)
FormFindUser
-string username
FormDeleteUser
-string username
DeleteUser.ASPX.CS
+DeleteUser()
+DeleteUser(string username)
+FindUser(string username)
FormFindUser
-string username
ListUser.ASPX.CS
+ListUser()
+FindUser(string username)

Figura 6- Diagrama de clases de la aplicación web. Fuente: Elaboración propia.

En la aplicación para clientes Windows se presenta de forma sencilla la generación de los correspondientes OTP a la semilla introducida por el usuario. En la misma son utilizados solamente dos capas del modelo N-Capas, fusionándose las capas modelo de negocio y acceso a datos, para evitar que los usuarios finales debieran instalar aplicaciones auxiliares para el almacenamiento de los datos necesarios en dicha generación. Al igual que en el diagrama anterior la mayor importancia recae sobre el método *generate_OTP,* los cuales deben ser capaces de mostrar al usuario el mismo OTP sin ningún tipo de conexión posible, basándose solamente en la similitud de su semilla. A continuación en la Figura 7 se muestra el diagrama de clases de la aplicación desktop.

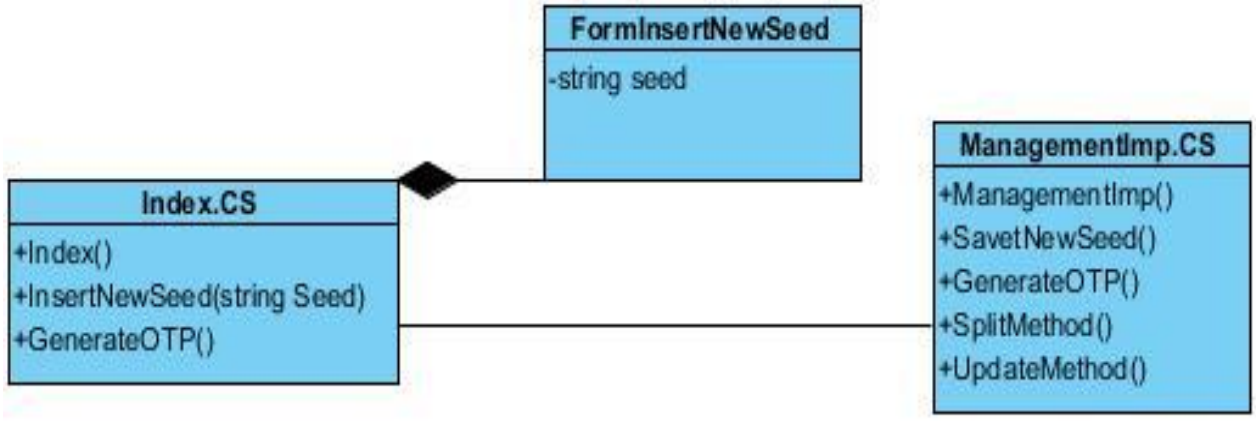


Figura 7- Diagrama de clases de la aplicación desktop. Fuente: Elaboración propia.

La aplicación móvil a desarrollar a pesar de utilizar la misma arquitectura, no está compuesta por clases sino que Android utiliza XML para visualizar sus aplicaciones a los usuarios finales y los métodos correspondientes a cada una de ellas son tratados como actividades, que pueden interactuar entre sí e intercambiar los resultados de sus ejecuciones. La capa de acceso a datos estará constituida por actividades de tipo ***SQLiteOpenHelper*** que serán capaces de la gestión de datos de dicha aplicación.

3.2.4. Diagrama de base de datos

Un modelo de datos es un conjunto de conceptos que permiten describir los datos, las relaciones que existen entre ellos, la semántica y las restricciones de consistencia. Además es una representación lógica y física de los datos persistentes usados por la aplicación, permite describir:

- **Las estructuras de datos:** los tipos de datos y la forma en que se relacionan.
- **Las restricciones de integridad:** condiciones que deben cumplir los datos para reflejar correctamente la realidad deseada.
- **Operaciones de manipulación de los datos:** operaciones sobre la información en la base de datos.

El modelo de datos del mecanismo está formado por dos tablas: *Users* y *Sample*. Los usuarios están compuestos por un identificador, una semilla, los caracteres consecutivos y el rol del usuario. Por otra parte, la tabla *Sample* contiene un identificador para cada ejemplo, el identificador del titular de la relación y el ejemplo en cuestión. A continuación en la Figura 8 se muestra el diagrama de Clases del Modelo de Datos.

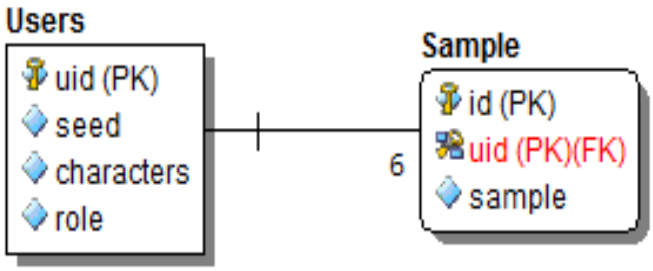


Figura 8- Diagrama de Clases del Modelo de Datos. Fuente: Elaboración propia.

3.2.5. Diagrama de despliegue

En el diagrama de despliegue se muestran elementos desplegables como los servicios Web y aplicaciones Web, además las bases de datos externas. El diagrama muestra las conexiones entre estas aplicaciones que reflejan su configuración actual en la solución.

Se destaca la presencia de un sistema externo (Mecanismo de Autenticación) al cual se le está desarrollando la actual extensión, pudiendo ser desplegada en el mismo servidor físico, debido a la compatibilidad de las tecnologías utilizadas. A continuación en la Figura 9 se muestra el diagrama de despliegue.

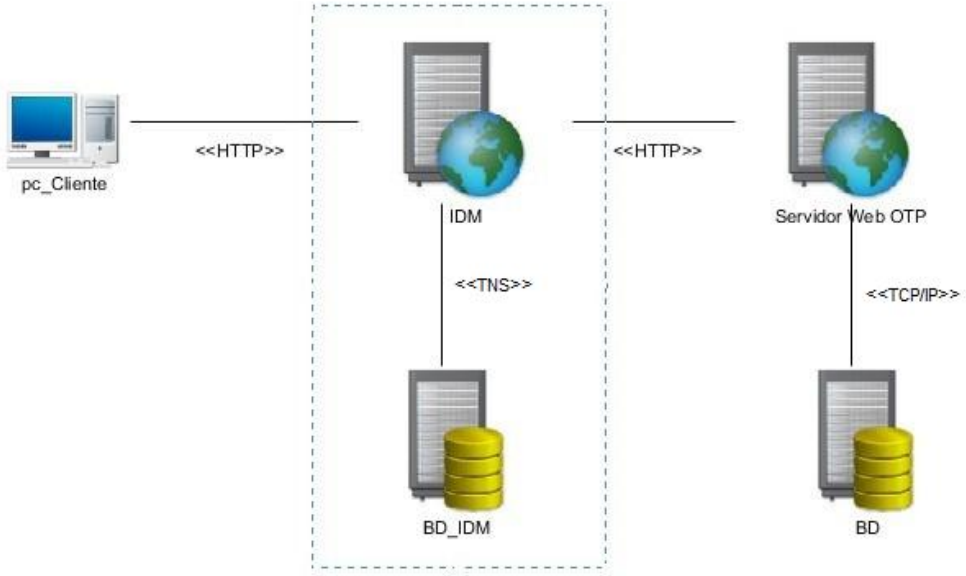


Figura 9- Diagrama de despliegue. Fuente: Elaboración propia.

3.3. Fase de estabilización

3.3.1. Pruebas

Los argumentos expuestos por Preesman (2002) demuestran que las pruebas son el proceso de ejecución de un programa con la intención de descubrir un error. Las mismas son un elemento crítico para la garantía de calidad del *software* y representan una visión final de las especificaciones, del diseño y de la codificación.

Las pruebas de *software* involucran las operaciones del sistema bajo condiciones controladas y evaluando los resultados. Las condiciones controladas pueden ser normales o anormales. La prueba puede intencionalmente esforzar al programa y producir errores en las respuestas para determinar si los sucesos ocurren cuando no tendrían que ocurrir o cuando los hechos no suceden cuando deberían suceder.

Se puede ir realizando pruebas desde la fase de visión del *software* hasta la fase de desarrollo y estabilización, siendo esta última fase donde tiene mayor volumen el flujo de trabajo de prueba.

3.3.2. Técnicas de prueba

Las pruebas se pueden clasificar de acuerdo a la parte visible o no visible de la aplicación, es decir, si se prueba el código o la funcionalidad a través de la interfaz.

Prueba de caja blanca

Según Preesman (2002) la prueba de caja blanca del *software* se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del *software* proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles.

En este sentido el propio autor señala que las pruebas de caja blanca, también llamadas pruebas estructurales, técnicas de caja transparente o de cristal, analizan de manera minuciosa el funcionamiento y estructura interna del programa. Su objetivo es elaborar casos de pruebas que permitan la ejecución de todas las sentencias del programa, incluidas las condiciones, al menos una vez cada una.

Debido a que en un programa sería muy complejo probar todos los caminos posibles, existen criterios que permiten decidir cuáles de ellos se deben examinar. Ver Anexo 2.

Prueba de caja negra

Al referirse a este aspecto Preesman (2002) puntualiza que las pruebas de caja negra, también denominadas Pruebas de Comportamiento, se centran en los requisitos funcionales del *software*. La prueba de caja negra intenta encontrar errores de las siguientes categorías:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a bases de datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y de terminación.

Estas pruebas se refieren a las pruebas que se llevan a cabo sobre la interfaz del *software*. Los casos de prueba pretenden demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto. Una prueba de caja negra

examina algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del *software*. (Preesman, 2002)

Para diseñar un caso de prueba de caja negra existen varios criterios, los cuales se encuentran detallados en el Anexo 2.

Estas técnicas de pruebas se llevaron a la práctica para probar el funcionamiento de las aplicaciones desarrolladas. Se usaron dos estrategias de pruebas, las pruebas unitarias hacia las funcionalidades de mayor complejidad, y la estrategia de prueba del sistema, usando la técnica partición de equivalencia donde se validaron las entradas válidas e inválidas de datos utilizando los casos de pruebas.

Pruebas unitarias

La prueba de unidad es la primera fase de las pruebas dinámicas y se realizan sobre cada módulo del *software* de manera independiente. El objetivo es comprobar que el módulo, entendido como una unidad funcional de un programa independiente, está correctamente codificado. Las pruebas unitarias se le aplicaron a las funcionalidades de más peso, o sea a los métodos más complejos del sistema desarrollado. Para ello se utilizó la herramienta Microsoft Visual Studio 2010. Se muestra el resultado del método Generar OTP (*Hash*), las demás se encuentran en el Anexo 3.

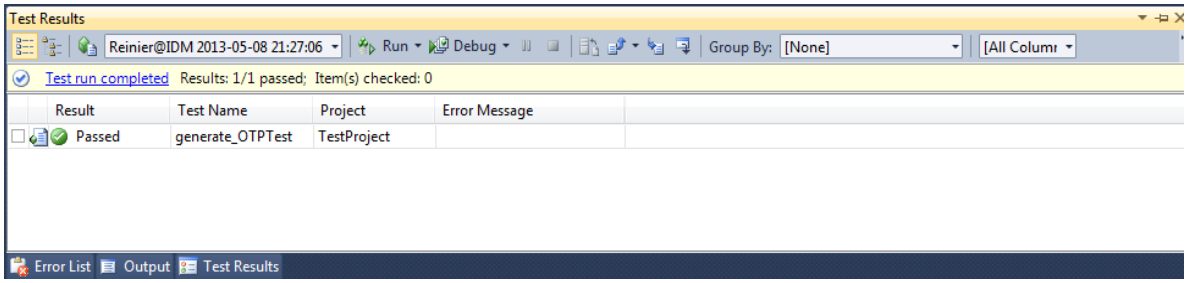
```
public string[] generate_OTP(string menssage)
        {
            byte[] mensajeBytesArray = Encoding.Default.GetBytes(menssage);

            SHA512Managed algoritmoHash = new SHA512Managed();

            byte[] codigoHashBytesArray = algoritmoHash.ComputeHash(mensajeBytesArray);

            return BitConverter.ToString(codigoHashBytesArray, 0).Split('-');
        }
```

Tabla 4 - Descripción de la prueba unitaria del método Generar OTP (*Hash*). Fuente: Elaboración propia.

Prueba Unitaria	
Nombre de la Prueba	Generar OTP (*Hash*)

Estado	Tipo	Ultima Ejecución
Satisfactoria	Caja Blanca	08/05/2012
Ejecutado por	**Verificado por**	
Grettel Mola Oliveros	Reinier Ruíz Cuellar	
Descripción	Para la ejecución de la prueba se le pasa como parámetro una cadena de caracteres de tipo string, la cual está constituida por la semilla correspondiente a un usuario y sus respectivos caracteres, los cuales deben devolver una cadena de 6 caracteres alfanuméricos de tipo arreglo de string que constituyen el próximo OTP.	
Entrada	semilla, caracteres	
Criterio de aceptación	Retorna un arreglo de tipo string de 6 caracteres alfanuméricos que van a constituir el OTP generado.	

Resultado:



Resultados de las pruebas unitarias

Se efectuaron dos iteraciones de pruebas unitarias a los métodos de mayor complejidad de la Extensión del Módulo de Autenticación del Sistema de Administración de Identidades. Fueron realizadas pruebas de unidad a 6 de las funcionalidades principales de las cuales ninguna resultó fallida. Los resultados se muestran en el Anexo 3.

3.3.3. Prueba de fiabilidad

En consonancia con los razonamientos de Pressman (2002) la fiabilidad del *software*, a diferencia de otros factores de calidad, puede ser medida o estimada mediante datos históricos o de desarrollo. La fiabilidad del *software* se define en términos estadísticos como: la probabilidad de operación libre de fallos de un programa de computadora en un entorno determinado y durante un tiempo específico.

El fallo es cualquier falta de concordancia con los requisitos del *software*. Los fallos pueden ser simplemente extraordinarios o ser catastróficos. Puede que un fallo sea corregido en segundos mientras que otro lleve semanas o incluso meses; la corrección de un fallo puede llevar a la introducción de otros errores que, finalmente lleven a más fallos.

Con la prueba de fiabilidad se comprueba que un programa realice su objetivo satisfactoriamente (sin fallos) en un determinado período de tiempo y en un entorno concreto.

Una sencilla medida de la fiabilidad es el tiempo medio entre fallos (TMEF), donde;

$$TMEF = TMDF + TMDR$$

Las siglas TMDF y TMDR corresponden a tiempo medio de fallo y tiempo medio de reparación, respectivamente. (Preesman, 2002)

Utilizando esta simple ecuación matemática para probar la fiabilidad de nuestras aplicaciones se concluyó que para las aplicaciones desarrolladas dentro del mecanismo de autenticación propuesto el TMEF fue tan bajo que no tendrá alto impacto en la fiabilidad del *software*.

Aplicaciones	**Desktop**	**Móvil**	**Web**
TMEF	2 días	3 días	1 días

3.3.4. Descripción de casos de pruebas

De acuerdo con las valoraciones de Pressman (2002) las pruebas de caja negra se refieren a las pruebas que se llevan a cabo sobre la interfaz del *software*. O sea, los casos de prueba pretenden demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma

adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. Una prueba de caja negra examina algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del *software*.

La siguiente tabla muestra la descripción del caso de prueba realizado al escenario Generar OTP (*Hash*). Las restantes tablas se encuentran en el Anexo 4.

Tabla 5- Descripción del caso de prueba Generar OTP (*Hash*). Fuente: Elaboración propia.

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central
ES: Generar OTP (*Hash*).	Con la semilla asignada a un usuario, se genera el OTP a partir de un algoritmo *Hash* y la introducción de caracteres consecutivos.	-	El sistema muestra una cadena de 6 caracteres alfanuméricos.	Luego de que el usuario esté autenticado es redireccionado a la página principal del sistema en la que se muestra el actual OTP. Al dar clic en el botón Generate new OTP, el sistema muestra el OTP generado.

Resultados de las pruebas de caja negra

En las pruebas de caja negras realizadas, de los 20 escenarios se identificaron 20 casos de pruebas. En la primera iteración se efectuaron 12 casos de pruebas detectándose 4 no conformidades a las cuales se les dio solución. En la segunda y última iteración se detectaron 3 no conformidades de los 8 casos de prueba restantes, las mismas fueron resueltas en su totalidad. En las dos iteraciones efectuadas se detectaron un total de 7 no conformidades, las cuales en su mayoría respondían a errores de bajo impacto en el correcto funcionamiento del sistema y todas tuvieron solución un tiempo máximo de 3 días, lo que indica que el mecanismo de autenticación con One-Time Password para el Módulo de Autenticación del Sistema de Administración de Identidades del Centro de Identificación y Seguridad Digital desarrollada presenta buena calidad. A continuación se muestra gráficamente lo anteriormente expuesto.

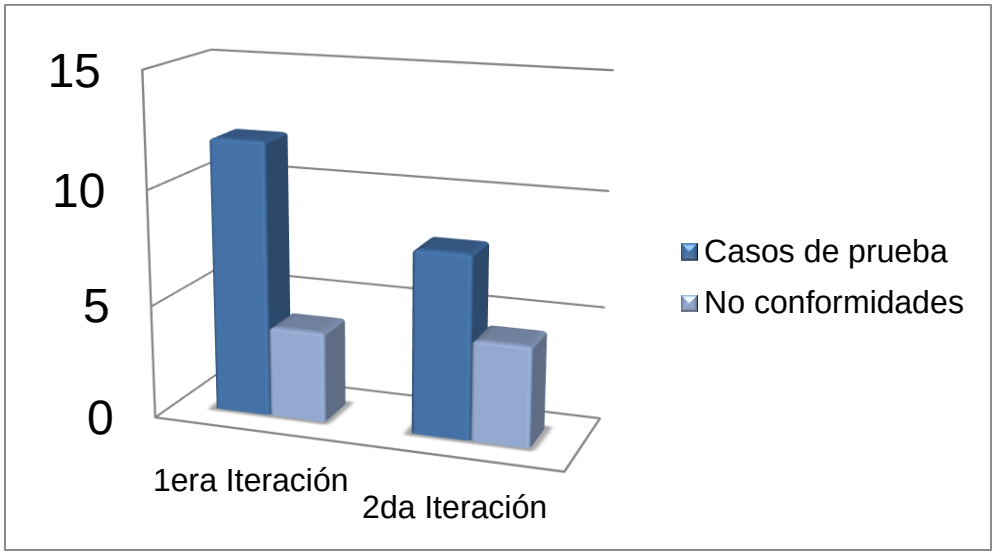


Figura 10- Resultados de pruebas de caja negra. Fuente: Elaboración propia.

3.4. Conclusiones del capítulo

Luego de concluidas las fases de desarrollo y estabilización documentadas en este capítulo se obtuvo el mecanismo de autenticación con OTP para el Módulo de Autenticación del Sistema de Administración de Identidades de CISED.

La arquitectura definida, los artefactos generados, así como los patrones de diseño seleccionados, facilitaron y favorecieron la implementación de la solución propuesta. Las pruebas realizadas validaron el correcto funcionamiento de las aplicaciones del mecanismo.

Una vez concluido y validado el desarrollo de las aplicaciones desarrolladas, el sistema cuenta con una herramienta que aumenta la seguridad del mismo en las aplicaciones que así lo requieran.

Conclusiones generales

- ✓ Durante el desarrollo del trabajo se realizó un estudio de las características de las aplicaciones y estándares existentes de One-Time Password que permitieron la realización de un análisis previo y con él un mayor entendimiento del problema a resolver.
- ✓ El análisis y estudio, de las herramientas y tecnologías necesarias para el desarrollo del mecanismo de autenticación con One-Time Password del Módulo de Autenticación del Sistema de Administración de Identidades del CISED, permitió definir los lenguajes y herramientas para su implementación; así como la metodología que guió el proceso de desarrollo.
- ✓ El diseño realizado del mecanismo de autenticación con One-Time Password permitieron la implementación exitosa del mismo.
- ✓ Con la implementación mecanismo de autenticación con One-Time Password el subproceso de autenticación del Sistema de Administración de Identidades del CISED, consta con mayor fortaleza, aumentando así considerablemente la seguridad del mismo.
- ✓ La ejecución de las pruebas realizadas validaron el correcto funcionamiento del mecanismo.

Recomendación

- ✓ Extender la aplicación cliente de OTP para Windows a otras plataformas.

Bibliografía

Barra, J. D. (18 de Septiembre de 2010). Recuperado el 16 de Enero de 2013, de http://justindeveloper.wordpress.com/2010/09/18/microsoft-solutions-framework/

Borghello, C. (2009). *Seguridad de la Información*. Recuperado el 2 de Diciembre de 2012, de Seguridad de la Información: http://www.segu-info.com.ar/logica/identificacion

Charles, J. (Mayo de 1999). *IEEExplore Digital Library.* Recuperado el 20 de Enero de 2013, de http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=762786&url=http%3A%2F%2Fieeexplore.ieee.org%2Fielx5%2F2%2F16523%2F00762786

Dofactory. (2001). *Dofactory*. Recuperado el 1 de Marzo de 2013, de http://www.dofactory.com/Patterns/PatternCommand.aspx

Embarcadero Technologies. (30 de enero de 2009). *Embarcadero Technologies.* Obtenido de Embarcadero Technologies: http://www.soluciones-ag.com/pdf_productos/Spanish_ER-Studio_Datasheet_2009.pdf

Etcheverry, L. (Marzo de 2010). *Pedeciba.* Recuperado el 1 de Abril de 2013, de www.pedeciba.edu.uy/bioinformatica/sibdyw/Clase_3.pdf

Exes. (23 de abril de 2004). *Mailxmail.* Recuperado el 18 de mayo de 2013, de Mailxmail: http://www.emagister.com/curso-java/caracteristicas-lenguaje-java-1

Ferrás, C. B. (2012). Propuesta de arquitectura para desarrollo de aplicaciones compuestas para dispositivos móviles con Android. *Propuesta de arquitectura para desarrollo de aplicaciones compuestas para dispositivos móviles con Android.* La Habana.

García., J. (2005). *Desarrollo de Software Orientado a Objetos.* Recuperado el 12 de noviembre de 2012, de Desarrollo de Software Orientado a Objetos.: http://www.ingenierosoftware.com/analisisydiseno/uml.php

Google Authenticator. (s.f.). *Apps Documentation & Support*. (Google Team) Recuperado el 12 de Enero de 2013, de http://support.google.com/a/bin/answer.py?hl=en&answer=1037451

Huanca, D. (2009). *HerrorSoft*. Recuperado el 29 de enero de 2013, de HerrorSoft: http://herrorsoft.zxq.net/pruebacajanegra.html

Kuhn, M. (2003). *OTPW*. Obtenido de http://www.cl.cam.ac.uk/~mgk25/otpw.html

Lasater, C. (2007). *Codeproject*. Recuperado el 1 de Marzo de 2013, de http://www.codeproject.com/KB/books/DesignPatterns.aspx

Microsoft. (Octubre de 2005). *Microsoft*. Recuperado el 14 de Febrero de 2013, de http://msdn.microsoft.com/en-us/library/aa479030.aspx.

Microsoft. (2013). *Autenticación*. Recuperado el 29 de Enero de 2013, de http://msdn.microsoft.com/es-es/library/syf5yeat.aspx

Microsoft. (2013). *Microsoft*. Recuperado el 19 de mayo de 2013, de Microsoft: http://msdn.microsoft.com/es-es/library/4w3ex9c2%28v=vs.100%29.aspx

Microsoft. (2013). *MSDN*. Recuperado el 20 de nov de 2012, de MSDN: http://msdn.microsoft.com/es-es/library/dd566231(v=vs.100).aspx

Microsoft. (2013). *MSDN*. Recuperado el 20 de marzo de 2012, de MSDN: http://msdn.microsoft.com/es-es/library/w0x726c2(v=vs.100).aspx

Microsoft. (2013). *MSDN*. Recuperado el 21 de noviembre de 2012, de MSDN: http://msdn.microsoft.com/es-es/library/52f3sw5c(v=vs.100).aspx

Microsoft. (2013). *MSDN*. Recuperado el 21 de noviembre de 2011, de MSDn: http://msdn.microsoft.com/es-es/library/kx37x362(v=vs.100).aspx

Microsoft. (2013). *Share Point. Tech Center*. Recuperado el 16 de enero de 2013, de Share Point. Tech Center: http://technet.microsoft.com/en-us/sharepoint/bb267375.aspx

M'Raihi. (Diciembre de 2005). Recuperado el 12 de Enero de 2013, de http://tools.ietf.org/html/rfc4226

M'Raihi, D. (Mayo de 2011). *Time-Based One-Time Password Algorithm.* Recuperado el 13 de Diciembre de 2012, de http://tools.ietf.org/html/rfc6238

Oracle. (s.f.). *Java.Acerca de la tecnología Java.* (Oracle) Recuperado el 05 de enero de 2013, de Acerca de la tecnología Java: http://www.java.com/es/download/faq/whatis_java.xml

PandaID Soluciones, C.A. (11 de Octubre de 2012). *¿Qué es la Autenticación Fuerte?* Recuperado el 2 de Diciembre de 2012, de http://www.pandaid.com/que-es-la-autenticacion-fuerte/

Preesman, R. S. (2002). Ingeniería de Software. Un enfoque práctico.5ta Edición. En D. Ince, *Ingeniería de Software. Un enfoque práctico.5ta Edición.*

Rodríguez, D. (08 de febrero de 2010). *Tecnologías, software libres y otras inquietudes.* Recuperado el 23 de enero de 2013, de Tecnologías, software libres y otras inquietudes.: http://davidjguru.com/2010/02/08/la-arquitectura-de-tres-capas-introduccion/

Rosa. (28 de Enero de 2013). *Seguridad Informática*. (Wordpress) Recuperado el 20 de Enero de 2013, de http://enboliviacom.wordpress.com/2013/01/28/seguridad-informatica/

Safelayer Labs. (2010). *Investigación y desarrollo: Seguridad, confianza y privacidad*. Recuperado el 13 de Diciembre de 2012, de Contraseñas de un solo uso (One Time Passwords): http://labs.safelayer.com/es/investigacion-y-desarrollo/areas/seguridad-confianza-y-privacidad/446-contrasenas-de-un-solo-uso

Sama, C. (04 de junio de 2011). *Androiddeity*. Recuperado el 21 de noviembre de 2012, de Arquitectura de android: http://androideity.com/2011/07/04/arquitectura-de-android/

Sama, C. (4 de Julio de 2011). *Androiddeity.* Recuperado el 20 de Enero de 2013, de Arquitectura de Android: http://androideity.com/2011/07/04/arquitectura-de-android/

Schackow, S. (2006). *ASP.NET 2.0 Security,Membership, and Role Management.* Wiley Publishing: Indianapolis, Indiana.

Seco, J. A. (29 de Septiembre de 2006). *Microsoft .NET.* Recuperado el 12 de Enero de 2013, de http://www.devjoker.com/contenidos/articulos/88/Introduccion-a-NET.aspx

The PostgreSQL Global Development Group. (1996 - 2013). *PostgresSQL* . Recuperado el 03 de diciembre de 2012, de PostgresSQL : http://www.postgresql.org/

Tückler, H. R. (2012). *GXtechnical.com.* Recuperado el 3 de Diciembre de 2012, de http://www.gxtechnical.com/gxdlsp/pub/genexus/internet/technicalpapers/web_services.htm

Vegas, S., Moreno, A., & Juristo, N. (2006). *Técnicas de Evaluación de Software.*

Visual Paradigm International Ltd. (05 de marzo de 2007). *Free Donwload Manager.* (Visual Paradigm International Ltd.) Recuperado el 12 de nov de 2012, de Free Donwload Manager.: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

Vivas, A. (26 de agosto de 2010). *KUMACHE.TICKS.Centro de Programación*. Recuperado el 15 de 11 de 2012, de KUMACHE.TICKS.Centro de Programación: https://sites.google.com/site/centrodeprogramacion/planeta-sql/postgresql

Wesley, A. (2003). *Design Patterns. Elements of Reusable Object-Oriented.*

Glosario de términos

B

Biometría: Métodos de identificación y autenticación de los seres humanos a través de características fisiológicas y de comportamiento.

C

CASE (*Computer Aided Software Engineering*): son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software*.

CIDR (*Classless Inter-Domain Routing*): representa la última mejora en el modo de interpretar las direcciones IP. Su introducción permitió una mayor flexibilidad al dividir rangos de direcciones IP en redes separadas.

D

Dalvik: es la máquina de proceso virtual (VM) en el sistema operativo Android de Google. Es el *software* que ejecuta las aplicaciones en los dispositivos Android.

G

GPS (*Global Positioning System*): Sistema Global de Navegación por Satélite el cual permite determinar en todo el mundo la posición de un objeto.

H

HMAC: código de mensaje de autenticación basado en *hash*, es una especificación para el cálculo de un código de autenticación de mensaje (MAC) que implica una función criptográfica *hash* en combinación con una clave secreta.

HMAC-SHA1: algoritmo *hash* en clave que se crea desde la función *hash* SHA1 y se utiliza como HMAC.

I

Initiative For Open Authentication: industria de colaboración para desarrollar una arquitectura de referencia abierta utilizando estándares abiertos para promover la adopción de autenticación fuerte.

IntelliSense: aplicación de autocompletar, mejor conocido por su utilización en Microsoft Visual Studio entorno de desarrollo integrado. Además de completar el símbolo de los nombres que el programador está escribiendo, *IntelliSense* sirve como documentación y desambiguación de los nombres de variables, funciones y métodos de utilización de metadatos basados en la reflexión.

IETF (Internet Engineering Task Force): organización internacional abierta de normalización, que tiene como objetivos el contribuir a la ingeniería de Internet, actuando en diversas áreas, como transporte, encaminamiento, seguridad.

K

Keylogger: *software* o un dispositivo *hardware* específico que se encarga de registrar las pulsaciones que se realizan en el teclado, para posteriormente memorizarlas en un fichero o enviarlas a través de internet.

L

LDAP (*Lightweight Directory Access Protocol*): Protocolo Ligero de Acceso a Directorios, es un servidor que provee la autenticación a los usuarios de una subred o un dominio.

M

MAC (*Media Access Control*): es un identificador de 48 bits (6 bloques hexadecimales) que corresponde de forma única a una tarjeta o dispositivo de red. Se conoce también como dirección física, y es única para cada dispositivo.

Middleware: es un *software* que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, *software*, redes, *hardware* y/o sistemas operativos. Éste simplifica el trabajo de los programadores en la compleja tarea de generar las conexiones que son necesarias en los sistemas distribuidos.

P

PIN (*Personal Identification Number*): Valor numérico (contraseña) usado para identificarse y poder acceder a sistemas, como un celular, un cajero automático.

Q

Quick Response (*QR*) code: módulo útil para almacenar información en una matriz de puntos o un código de barras bidimensional.

S

SDK (*Software Development Kit*): conjunto de herramientas de desarrollo de *software* que le permiten al programador crear aplicaciones para un sistema concreto.

Sniffer: programa de captura de las tramas de una red de computadoras.

SSL (*Secure Sockets Layer*): Es un sistema de seguridad ideado para acceder a un servidor garantizando la confidencialidad de los datos mediante técnicas de encriptación modernas.

T

Token: Secuencia de caracteres capaz de indicar la autoridad, prueba o autenticidad de un usuario.

X

XML (*eXtensible Markup Language*): surgió como un lenguaje de marcado para sustituir a HTML, es un sistema para definir, validar y compartir formatos de documentos en la Web.

Anexo 1: Descripción de los escenarios.

Tabla 6- Crear usuario. Fuente: Elaboración propia.

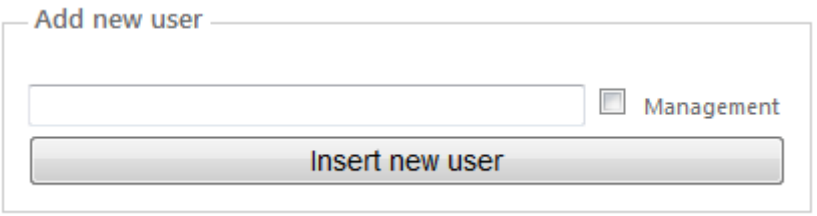
Nombre del escenario: Crear usuario.		**Identificador:**1
Objetivo del escenario: Crear el usuario con el cual se realizarán las operaciones en la aplicación.		
Persona: Administrador		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: El Administrador debe estar autenticado.		
Descripción: El Administrador crea el usuario comprobando que no exista otro usuario con las mismas credenciales, añadiéndole además los permisos que requiere el mismo dentro de la aplicación.		
Validaciones: Verificar que el campo no esté vacío. El usuario no debe estar creado, debe utilizarse solamente caracteres alfanuméricos, *underscore* (_), punto (.), no debe poseer espacios.		
Prototipo de interfaz de usuario: 		

Tabla 7- Modificar usuario. Fuente: Elaboración propia.

Nombre del escenario: Modificar usuario.		**Identificador:**2
Objetivo del escenario: Modifica al usuario.		
Persona: Administrador		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: Debe estar creado el usuario, el administrador debe estar autenticado para hacer las modificaciones.		
Descripción: El administrador realiza las modificaciones necesarias en los permisos que tiene el usuario a modificar, en caso de necesitarlas.		
Validaciones: Verificar que el campo no esté vacío. Comprobar que el usuario esté creado, debe utilizarse solamente caracteres alfanuméricos, *underscore* (_), punto (.), no debe poseer espacios.		
Prototipo de interfaz de usuario:		

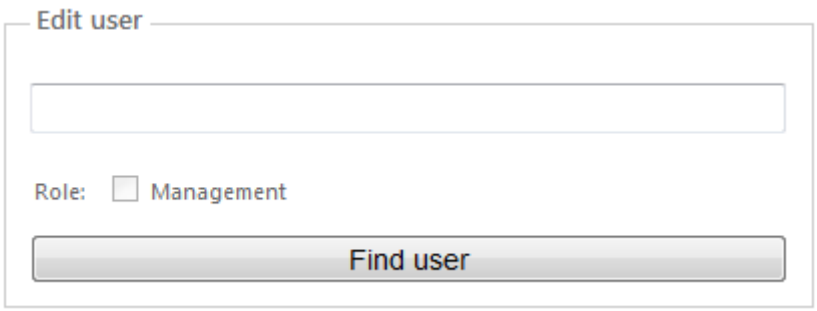


Tabla 8- Eliminar usuario. Fuente: Elaboración propia.

Nombre del escenario: Eliminar usuario.		**Identificador:** 3
Objetivo del escenario: Elimina al usuario en caso necesario.		
Persona: Administrador		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: Debe estar creado el usuario, el administrador debe estar autenticado para proceder a la eliminación del usuario.		
Descripción: El mecanismo realiza una comprobación previa y cuestiona al administrador sobre la seguridad de eliminar o no el usuario introducido; en caso de que así sea se procede a eliminar el usuario.		
Validaciones: Verificar que el campo no esté vacío. Comprobar que el usuario esté creado.		
Prototipo de interfaz de usuario:		

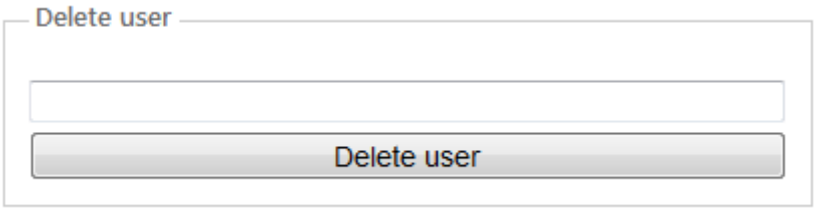


Tabla 9- Mostrar usuarios. Fuente: Elaboración propia.

Nombre del escenario: Mostrar usuarios.		**Identificador:** 4
Objetivo del escenario: Muestra los usuarios.		
Persona: Administrador.		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: Debe estar creado el usuario, el administrador debe estar autenticado para proceder a mostrar el usuario.		
Descripción: Se muestra una lista de los usuarios ya creados.		
Validaciones: -		
Prototipo de interfaz de usuario:		

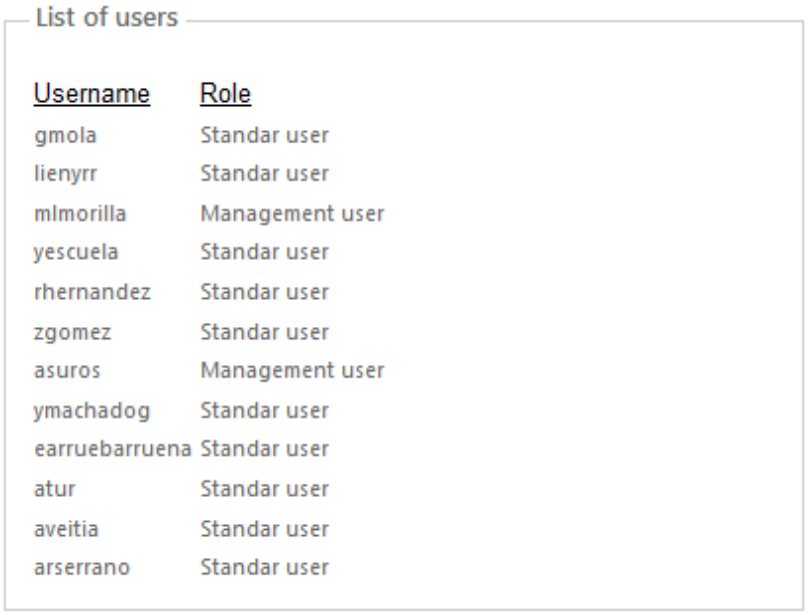


Tabla 10- Buscar usuario. Fuente: Elaboración propia.

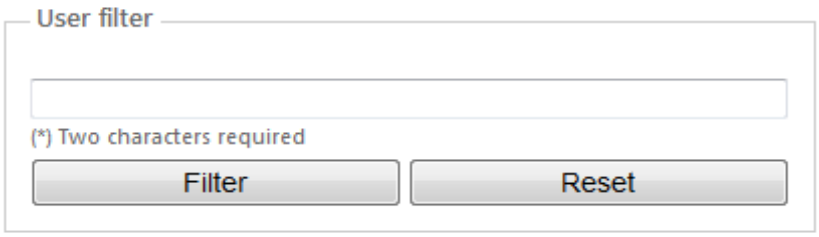
Nombre del escenario: Buscar usuario.		**Identificador:** 5
Objetivo del escenario: Busca al usuario.		
Persona: Administrador o Usuario.		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: Debe estar creado el usuario, el administrador debe estar autenticado para proceder a buscar el usuario o el usuario en sí está procediendo a autenticarse.		
Descripción: El mecanismo brinda la posibilidad de buscar los usuarios cuando van a ser ejecutados los escenarios crear usuario, modificar usuario, eliminar usuario y mostrar usuario; así como cuando el usuario procede a autenticarse comprobándose que dicho usuario esté registrado en el mismo.		
Validaciones: -		
Prototipo de interfaz de usuario: 		

Tabla 11- Generar semilla. Fuente: Elaboración propia.

Nombre del escenario: Generar semilla.	**Identificador:** 6
Objetivo del escenario: Genera la semilla.	

Persona: Administrador.		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: El usuario debe estar autenticado y poseer permisos de administración.		
Descripción: Genera una cadena con una longitud de 16 caracteres, que contiene aleatoriamente letras en mayúsculas y números, la cual es asignada a un nuevo usuario creado por el administrador.		
Validaciones: -		
Prototipo de interfaz de usuario: -		

Tabla 12- Actualizar semilla. Fuente: Elaboración propia.

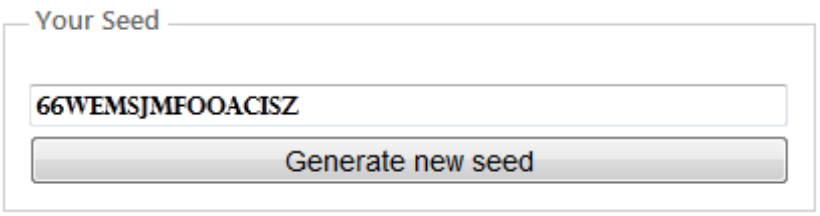
Nombre del escenario: Actualizar semilla.		**Identificador:** 7
Objetivo del escenario: Actualiza la semilla.		
Persona: Usuario.		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: El usuario debe estar autenticado.		
Descripción: Se genera una nueva cadena de caracteres aleatorios que contiene letras mayúsculas y números, con una longitud de 16 caracteres, la cual es asignada al usuario autenticado.		
Validaciones: -		
Prototipo de interfaz de usuario: 		

Tabla 13-Eliminar semilla. Fuente: Elaboración propia.

Nombre del escenario: Eliminar semilla.		**Identificador:** 8
Objetivo del escenario: Elimina la semilla.		
Persona: Administrador		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: Debe estar creado el usuario con su semilla correspondiente y el usuario que procede a la eliminación debe poseer permisos administrativos.		
Descripción: Al eliminarse un usuario, seguidamente se eliminará su semilla correspondiente.		
Validaciones: -		
Prototipo de interfaz de usuario: -		

Tabla 14- Mostrar semilla. Fuente: Elaboración propia.

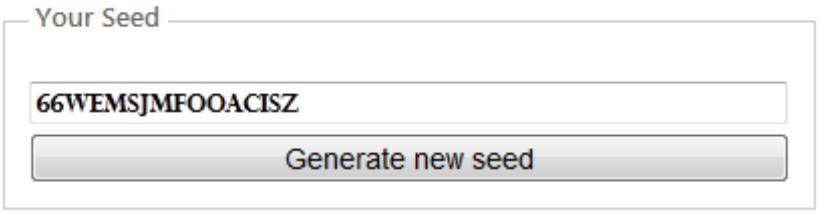
Nombre del escenario: Mostrar semilla.		**Identificador:** 9
Objetivo del escenario: Muestra la semilla.		
Persona: Usuario		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: Debe estar creado el usuario con su semilla correspondiente, además el mismo debe estar autenticado.		
Descripción: Muestra la semilla asignada al usuario en cuestión.		
Validaciones: -		
Prototipo de interfaz de usuario: 		

Tabla 15- Generar muestras de OTP. Fuente: Elaboración propia.

Nombre del escenario: Generar muestras de OTP		**Identificador:** 10
Objetivo del escenario: Genera las muestras de OTP.		
Persona: Administrador.		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: Debe existir el usuario con su semilla, y el administrador debe estar autenticado en la aplicación.		
Descripción: Con la semilla asignada a un usuario, se genera mediante un algoritmo *Hash* y la introducción de caracteres aleatorios, las 6 muestras de OTP, las cuales son asignadas a dicho usuario.		
Validaciones: -		
Prototipo de interfaz de usuario: -		

Tabla 16- Actualizar muestras de OTP. Fuente: Elaboración propia.

Nombre del escenario: Actualizar muestras de OTP.		***Identificador:** 11*
Objetivo del escenario: Actualiza las muestras de OTP.		
Persona: Usuario.		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: Debe existir el usuario con su semilla, y además debe estar autenticado.		

Descripción: Se genera nuevamente las muestras de OTP, a partir de la semilla asignada al usuario y la generación se realiza mediante un algoritmo *Hash* y la introducción de caracteres aleatorios.
Validaciones: Verificar que esté autenticado el usuario con su semilla.
Prototipo de interfaz de usuario: 

Tabla 17- Eliminar muestras de OTP. Fuente: Elaboración propia.

Nombre del escenario: Eliminar muestras de OTP		**Identificador:** 12
Objetivo del escenario: Elimina las muestras de OTP.		
Persona: Administrador.		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: Debe existir el usuario, y además el administrador debe estar autenticado.		
Descripción: Elimina las muestras de OTP.		
Validaciones: -		
Prototipo de interfaz de usuario: -		

Tabla 18- Mostrar muestras de OTP. Fuente: Elaboración propia.

Nombre del escenario: Mostrar muestras de OTP.		**Identificador:** 13
Objetivo del escenario: Muestra las muestras de OTP.		
Persona: Usuario.		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: Debe existir el usuario con su semilla, y además debe estar autenticado.		
Descripción: El usuario visualiza las muestras de OTP generadas.		
Validaciones: -		
Prototipo de interfaz de usuario: 		

Tabla 19- Guardar muestras de OTP. Elaboración propia.

Nombre del escenario: Guardar muestras de OTP		**Identificador:** 14
Objetivo del escenario: Guardar las muestras de OTP.		
Persona: Usuario.		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: Debe existir el usuario con su semilla, y además debe estar autenticado.		
Descripción: Las muestras de OTP son guardadas en un archivo de texto generado.		
Validaciones: -		
Prototipo de interfaz de usuario: 		

Tabla 20- Actualizar semilla en el cliente. Fuente: Elaboración propia.

Nombre del escenario: Actualizar semilla en el cliente		**Identificador:** 1
Objetivo del escenario: Actualiza las semillas en el cliente.		
Persona: Usuario.		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: -		
Descripción: El usuario obtiene su semilla a través de la aplicación web y la misma es introducida en la aplicación cliente.		
Validaciones: La semilla debe contener 16 caracteres alfanuméricos sin espacios intermedios u otros caracteres especiales.		
Prototipo de interfaz de usuario:		

Tabla 21- Guardar semilla en el cliente. Fuente: Elaboración propia.

Nombre del escenario: Guardar semilla en el cliente		**Identificador:** 2
Objetivo del escenario: Guarda las semillas en el cliente.		
Persona: Usuario.		
Iteración: 1	**Prioridad:** Crítico.	**Complejidad:** 3
Precondiciones: Debe haber sido actualizada la semilla.		
Descripción: Después de actualizada la semilla en el cliente, se guarda esta semilla en un archivo predeterminado por los administradores.		
Validaciones: -		
Prototipo de interfaz de usuario: -		

Tabla 22- Comprobar OTP. Fuente: Elaboración propia.

Nombre del escenario: Comprobar OTP		**Identificador:** 15
Objetivo del escenario: Comprobar el OTP generado.		
Persona: Usuario.		
Iteración: 2	**Prioridad:** Crítico.	**Complejidad:** 2
Precondiciones: Debe existir el usuario con su semilla.		
Descripción: Se realiza la comprobación de que el OTP introducido se corresponda con el OTP generado por la aplicación web; en caso de no coincidir, se comprueba que dicho OTP se corresponda con alguna		

de las muestras de OTP que brinda también la aplicación web; si no se calculan los próximos 10 OTP de forma secuencial y se comprueba en cada uno de los casos si se corresponden con el OTP introducido.
Validaciones: Deben ser introducidos un usuario válido y una cadena alfanumérica de 6 caracteres.
Prototipo de interfaz de usuario: -

Tabla 23- Asociar semilla a usuario. Fuente: Elaboración propia.

Nombre del escenario: Asociar semilla a usuario.		**Identificador:** 16
Objetivo del escenario: Asociar semilla al usuario.		
Persona: Administrador.		
Iteración: 2	**Prioridad:** Crítico.	**Complejidad:** 2
Precondiciones: Debe existir el usuario.		
Descripción: Se realiza cuando se crea un usuario, seguidamente se le asocia una semilla o cuando el usuario decide por voluntad propia generar una nueva semilla.		
Validaciones: -		
Prototipo de interfaz de usuario: -		

Tabla 24- Integrar mecanismo de autenticación al Sistema de Administración de Identidades. Fuente: Elaboración propia.

Nombre del escenario: Integrar mecanismo de autenticación al Sistema de Administración de Identidades.		**Identificador:** 17
Objetivo del escenario: La integración del mecanismo de autenticación al Sistema de Administración de Identidades.		
Persona: Administrador.		
Iteración: 2	**Prioridad:** Crítico.	**Complejidad:** 2
Precondiciones: Debe estar listo el mecanismo de autenticación para integrarlo.		
Descripción: Se realiza a través de la implementación de librerías y el uso del servicio web, para la integración del mecanismo de autenticación al Sistema de Administración de Identidades.		
Validaciones: -		
Prototipo de interfaz de usuario: -		

Anexo 2: Criterios de prueba para diseñar casos de pruebas de caja blanca y caja negra.

Para pruebas de caja blanca:

Cobertura de sentencias: se escriben casos de prueba suficientes para que cada sentencia en el programa se ejecute, al menos, una vez.

Cobertura de decisión: se escriben casos de prueba suficientes para que cada decisión en el programa se ejecute una vez con resultado verdadero y otra con resultado falso.

Cobertura de condiciones: se escriben casos de prueba suficientes para que cada condición en una decisión tenga una vez resultado verdadero y otra resultado falso.

Cobertura decisión/condición: se escriben casos de prueba suficientes para que cada condición en una decisión tome todas las posibles salidas, al menos una vez, y cada decisión tome todas las posibles salidas, al menos una vez.

Cobertura de condición múltiple: se escriben casos de prueba suficientes para que todas las combinaciones posibles de resultados de cada condición se invoquen, al menos una vez.

Cobertura de caminos: se escriben casos de prueba suficientes para que se ejecuten todos los caminos de un programa, entendiendo camino como una secuencia de sentencias encadenadas desde la entrada del programa hasta su salida. (Vegas, Moreno, & Juristo, 2006)

Para pruebas de caja negra:

Particiones de equivalencia: este método busca crear clases de datos de cada campo de entrada de un programa. Su objetivo es descubrir tipos de errores que permitan reducir la cantidad de casos de pruebas a ejecutar. Esto es importante, porque con un caso de prueba que se corresponde con una clase de dato, sería suficiente para demostrar que con otro valor similar, también se obtendría el mismo resultado, tanto errores como respuestas válidas.

Análisis de valores límites: esta técnica se centra en diseñar los casos de pruebas con los valores límites que puedan ser admitidos por los campos, debido a que proporcionan una idea más precisa de la efectividad de la evaluación. El análisis también se aplica a los datos de salida.

Métodos basados en grafos: este método plantea que se debe crear un grafo donde los nodos son objetos, tales como datos, módulos o grupo de sentencias, en dependencia de cómo se

considere diseñar los casos de prueba. Las conexiones entre los nodos serían las relaciones entre esos objetos.

Luego de haber creado el grafo, los casos de pruebas que se diseñen quedarán de forma tal que se visiten todos los nodos y sus relaciones. El ingeniero americano Boris Bézier definió un conjunto de modelos a tener en cuenta a la hora de aplicar este método:

- **Modelado del flujo de transacción:** los nodos representan los pasos de alguna transacción y los enlaces representan las conexiones lógicas entre los pasos.
- **Modelado de estado finito:** los nodos representan diferentes estados del *software* observables por el usuario (por ejemplo, cada una de las pantallas que aparecen cuando un telefonista coge una petición por teléfono), y los enlaces representan las transiciones que ocurren para moverse de estado a estado (por ejemplo, petición-información).
- **Modelado de flujo de datos:** los nodos objetos de datos y los enlaces son las transformaciones que ocurren para convertir un objeto de datos en otro.
- **Modelado de planificación:** los nodos son objetos de programa y los enlaces son las conexiones secuenciales entre esos objetos. Los pesos de enlace se usan para especificar los tiempos de ejecución requeridos al ejecutarse el programa. (Huanca, 2009)

Pruebas de comparación: también son llamadas pruebas de mano a mano; se le realizan a distintas versiones de una misma aplicación; si las versiones se desarrollan a la vez, lo lógico es que después determinadas arrojen los mismos resultados. También pueden servir para determinar semejanzas y diferencias entre una versión superior y otra inferior.

Prueba de la tabla ortogonal: se realiza cuando los campos de entrada son pocos y los posibles valores también lo son. El objetivo es probar exhaustivamente con cada posible valor del dominio, debido a que aplicar otros métodos podría ser más costoso.

Prueba de conjetura de errores: para llevar a cabo esta prueba se realiza una lista de errores o situaciones comunes que suelen provocar fallos o resultados inesperados en las aplicaciones. No existe un procedimiento estándar a seguir, solo que la lista debe ser confeccionada por una persona con experiencia realizando evaluaciones de *software*.

Análisis causa-efecto: este modelo pretende que se cree una gráfica donde se planteen la causa y el efecto de determinados problemas con el objetivo de tener una visión clara de los errores

presentes. Algunos autores incluyen este método como un modelo basado en grafos. Es importante aclarar que está muy relacionado con el anterior.

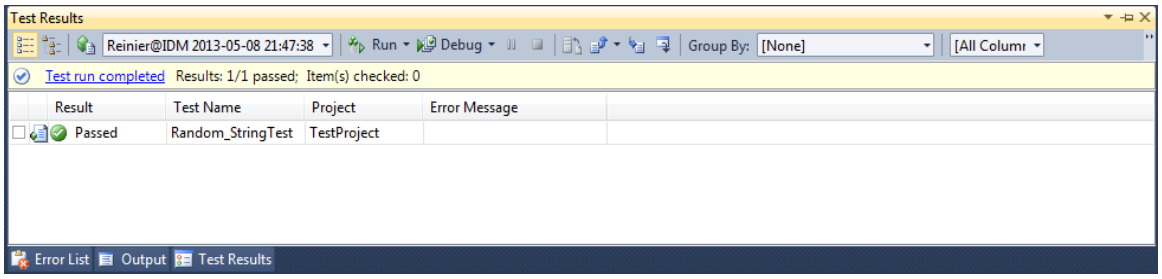
Anexo 3: Pruebas unitarias.

```
public string Random_String(int size)
        {
            char[] buffer = new char[size];

            for (int i = 0; i < size; i++)
            {
                buffer[i] = _chars[_rng.Next(_chars.Length)];
            }
            return new string(buffer);
        }
```

Figura 11- Método Random_String(). Fuente: Elaboración propia.

Tabla 25- Descripción de la prueba unitaria del método Random_string. Fuente: Elaboración propia.

Prueba Unitaria		
Nombre de la Prueba	Random_String	
Estado	**Tipo**	**Ultima Ejecución**
Satisfactoria	Caja Blanca	08/05/2012
Ejecutado por	**Verificado por**	
Grettel Mola Oliveros	Reinier Ruíz Cuellar	
Descripción	Para la ejecución de la prueba se le pasa como parámetro un número de tipo int el cual representa la longitud que debe tener la cadena de caracteres alfanuméricos que debe devolver la funcionalidad.	

Entrada	longitud
Criterio de aceptación	Retorna un string de n caracteres alfanuméricos definidos en la entrada de los datos el cual va a ser utilizado como semilla o caracteres aleatorios.
Resultado: 	


```
private void GetOTP(string username)
        {
            string seed = "";
            string characters = "";
            StringBuilder sb = new StringBuilder("SELECT ").Append('"').Append("Users").Append('"').Append(".seed, ")
                  .Append('"').Append("Users").Append('"').Append(".characters FROM public.").Append('"').Append("Users")
                  .Append('"').Append(" WHERE ").Append('"').Append("Users").Append('"').Append(".uid='")
                  .Append(username).Append("';");
            IDbConnection dbcon = new NpgsqlConnection(connectionString);
            dbcon.Open();
            IDbCommand dbcmd = dbcon.CreateCommand();
            dbcmd.CommandText = sb.ToString();
            IDataReader reader = dbcmd.ExecuteReader();
            while (reader.Read())
            {
                Label1.Text = "";
                seed = reader.GetString(reader.GetOrdinal("seed"));
                characters = reader.GetString(reader.GetOrdinal("characters"));
                textBox1.Text = seed;
                textBox2.Text = characters;
                for (int i = 0; i < 2; i++)
                {
                    textBox2.Text += initialSolution.generate_OTP(seed + characters)[i];
                };
            }
            reader.Close();
            reader = null;
            dbcmd.Dispose();
            dbcmd = null;
            dbcon.Close();
            dbcon = null;
        }
```

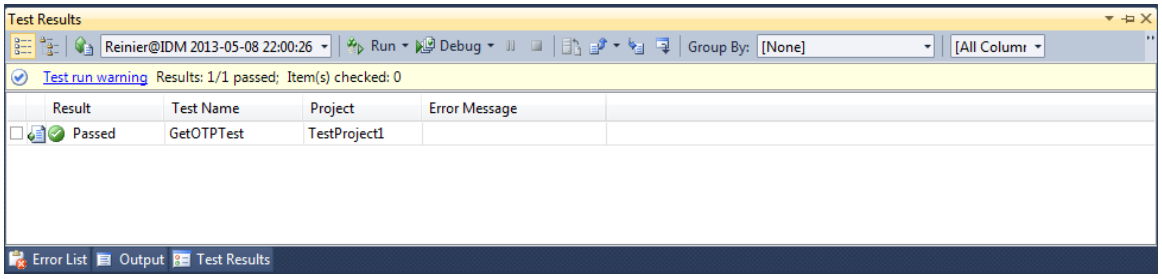
Figura 12- Método GetOTP(). Fuente: Elaboración propia.

Tabla 26- Descripción de la prueba unitaria del método GetOTP. Fuente: Elaboración propia.

Prueba Unitaria

Nombre de la Prueba	GetOTP	
Estado	**Tipo**	**Ultima Ejecución**
Satisfactoria	Caja Blanca	08/05/2012
Ejecutado por	**Verificado por**	
Grettel Mola Oliveros	Reinier Ruíz Cuellar	
Descripción	Para la ejecución de la prueba se le pasa como parámetro una cadena de caracteres de tipo string que representa un usuario, del mismo se obtiene su correspondiente OTP.	
Entrada	usuario	
Criterio de aceptación	Asigna un string de 6 caracteres alfanuméricos el cual corresponde al actual OTP del usuario introducido.	

Resultado:



```
private void GetSamples(string username)
        {
            string sample = "";
            string aux = "";
            StringBuilder sb = new StringBuilder("SELECT ").Append('"').Append("Sample").Append('"')
                 .Append(".sample FROM public.").Append('"').Append("Sample").Append('"').Append(" WHERE ")
                 .Append('"').Append("Sample").Append('"').Append(".uid='").Append(username).Append("';");
            IDbConnection dbcon = new NpgsqlConnection(connectionString);
            dbcon.Open();
            IDbCommand dbcmd = dbcon.CreateCommand();
            dbcmd.CommandText = sb.ToString();
            IDataReader reader = dbcmd.ExecuteReader();
            while (reader.Read())
            {
                sample = reader.GetString(reader.GetOrdinal("sample"));
                aux += sample + " ";
            }
            string[] auxSplit = aux.Split(' ');
            SetSamples(auxSplit);
            reader.Close();
            reader = null;
            dbcmd.Dispose();
            dbcmd = null;
            dbcon.Close();
            dbcon = null;
        }

        private void SetSamples(string[] aux)
        {
            textBox3.Text = aux[0];
            textBox4.Text = aux[1];
            textBox5.Text = aux[2];
            textBox6.Text = aux[3];
            textBox7.Text = aux[4];
            textBox8.Text = aux[5];
        }
```

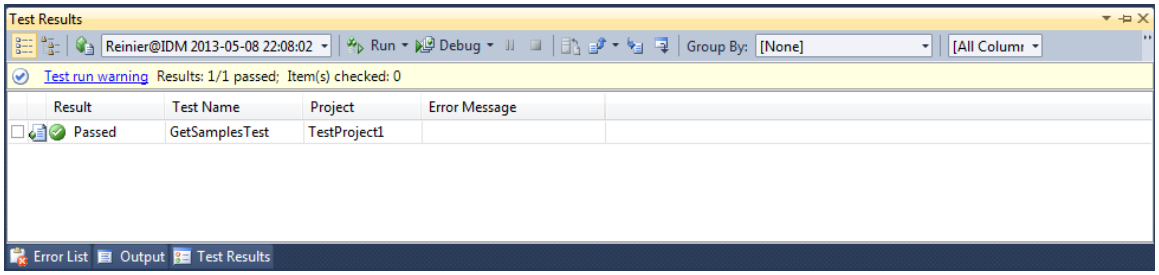
Figura 13- Método GetSamples(). Fuente: Elaboración propia.

Tabla 27- Descripción de la prueba unitaria del método GetSamples. Fuente: Elaboración propia.

Prueba Unitaria		
Nombre de la Prueba	GetSamples	
Estado	**Tipo**	**Ultima Ejecución**
Satisfactoria	Caja Blanca	08/05/2012
Ejecutado por	**Verificado por**	
Grettel Mola Oliveros	Reinier Ruíz Cuellar	

Descripción	Para la ejecución de la prueba se le pasa como parámetro una cadena de caracteres de tipo string que representa un usuario, del mismo se obtiene sus correspondientes muestras de OTP.
Entrada	usuario
Criterio de aceptación	Asigna cada muestra obtenida a su correspondiente textbox en la página principal de la aplicación.

Resultado:



Anexo 4: Descripción de casos de pruebas.

Tabla 28- Descripción del caso de prueba Administrar usuario. Fuente: Elaboración propia.

Escenario	Descripción	Nombre de usuario	Rol	Respuesta del sistema	Flujo central
ES01: Crear usuario.	El Administrador crea el usuario comprobando que no exista otro usuario con las mismas credenciales, añadiéndole además los permisos que requiere el mismo dentro de la aplicación.	V pepe_cuba	N/A	La aplicación web muestra un mensaje informando que el usuario ha sido creado.	Luego de estar autenticado se comprueba que el usuario sea administrador, accediendo a través del menú principal se da clic en el icono que representa la creación de un usuario, se introduce el nombre del usuario y se selecciona el rol. Luego se da clic en el botón Add user.
		I _borja	N/A	La aplicación web muestra un mensaje de error.	
		V guido.mola	N/A	La aplicación web muestra un mensaje informando que el usuario ha sido creado.	
ES02: Modificar usuario.	El administrador realiza las modificaciones necesarias en los permisos que tiene el usuario a modificar, en caso de necesitarlas.	V fuente2.ramos	N/A	La aplicación web muestra una nueva interfaz en la que se permite modificar el rol del usuario.	Luego de estar autenticado se comprueba que el usuario sea administrador, accediendo a través del menú principal se da clic en el icono que representa la modificación del usuario, se introduce el nombre
		I .cuellar_bad	N/A	La aplicación web muestra un mensaje de error.	

					de usuario, se da clic en el botón Find user, se comprueba que los datos introducidos pertenezcan a un usuario del sistema, se muestra una nueva interfaz en la que el sistema permite modificar los permisos que posee dicho usuario, se da clic en el botón Edit user.
ES02: Eliminar usuario.	El sistema realiza una comprobación previa y cuestiona al administrador del sistema sobre la seguridad de eliminar o no el usuario introducido; en caso de que así sea se procede a eliminar el usuario.	V gutierrez.dos	-	La aplicación web muestra una nueva interfaz en la que se confirma la eliminación del usuario introducido.	Luego de estar autenticado se comprueba que el usuario sea administrador, accediendo a través del menú principal se da clic en el icono que representa la eliminación del usuario, se introduce el nombre de usuario, se da clic en el botón Delete user, se comprueba que los datos introducidos pertenezcan a un
		I ._jackwas	-	La aplicación web muestra un mensaje de error.	

					usuario del sistema, se muestra una nueva interfaz en la que el sistema confirma la eliminación de dicho usuario dando clic en el botón Delete user, en caso de que no se confirme se da clic en el botón Cancel.
ES02: Mostrar usuario.	Se muestra una lista de los usuarios ya creados.	-	-	La aplicación web muestra una lista con los 12 primeros usuarios y su correspondiente rol.	Luego de estar autenticado se comprueba que el usuario sea administrador, accediendo a través del menú principal se da clic en el icono utilizado para listar los usuarios.
ES02: Buscar usuario.	Brinda la posibilidad de buscar los usuarios cuando van a ser ejecutados los escenarios crear usuario, modificar usuario, eliminar usuario y mostrar usuario; así como	V yurqui_gil	-	La aplicación web comprueba que exista un usuario que comprenda la cadena introducida.	Luego de estar autenticado se comprueba que el usuario sea administrador, accediendo a través del menú principal se da clic en el icono utilizado para listar los usuarios, se introduce el nombre
		I jade 2013	-	La aplicación web muestra un mensaje de	

	cuando el usuario procede a autenticarse comprobándose que dicho usuario esté registrado.			error.	de usuario se da clic en el botón Filter user y en caso de que algún usuario el sistema comprenda la cadena introducida, son mostrados los 12 primeros usuarios.

Tabla 29- Descripción de las variables del caso de prueba Administrar usuario. Fuente: Elaboración propia.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre de usuario	Campo de texto	No	Admite letras y de forma no consecutiva *underscore* y puntos.
2	Rol	Campo booleano	Si	Admite la selección o no del checkbox.

Tabla 30- Descripción del caso de prueba Administrar semilla. Elaboración propia.

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central

ES06: Generar semilla.	Genera una cadena con una longitud de 16 caracteres, que contiene aleatoriamente letras en mayúsculas y números, la cual es asignada a un nuevo usuario creado por el administrador.	-	Se muestra una cadena de 16 caracteres de longitud, la cual está compuesta por letras y números.	Luego de estar autenticado se comprueba que el usuario sea administrador, accediendo a través del menú principal se da clic en el icono que representa la creación de un usuario, se introduce el nombre del usuario y se selecciona el rol. Luego se da clic en el botón Add user y es allí cuando se genera una nueva semilla la cual es asignada a dicho usuario.
ES07: Actualizar semilla.	Se genera una nueva cadena de caracteres aleatorios que contiene letras mayúsculas y números, con una longitud de 16 caracteres, la cual es asignada al usuario autenticado.	-	Se actualiza la semilla que posee el usuario autenticado, mostrando una nueva cadena de caracteres.	Luego de estar autenticado un usuario es redireccionado a la página principal del sistema en el cual, al dar clic en el botón Generate new seed se le muestra la nueva semilla.

ES08: Eliminar semilla.	Al eliminarse un usuario del sistema, seguidamente se eliminará su semilla correspondiente.	-	Al eliminar un usuario elimina su semilla y muestra un mensaje de que ha sido eliminado el usuario.	Luego de estar autenticado se comprueba que el usuario sea administrador, accediendo a través del menú principal se da clic en el icono que representa la eliminación del usuario, se introduce el nombre de usuario, se da clic en el botón Delete user, se comprueba que los datos introducidos pertenezcan a un usuario del sistema, se muestra una nueva interfaz en la que el sistema confirma la eliminación de dicho usuario dando clic en el botón Delete user, procediéndose a la eliminación de su semilla.
ES09: Mostrar semilla.	Muestra la semilla asignada al usuario en cuestión.	-	Se muestra la semilla correspondiente al usuario autenticado.	Luego de estar autenticado el usuario es redireccionado a la página principal, mostrando así su semilla.

Tabla 31- Descripción del caso de prueba Administrar muestras de OTP. Fuente: Elaboración propia.

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central
ES10: Generar muestras de OTP.	Con la semilla asignada a un usuario, se genera mediante	-	Se proporciona al usuario las 6 muestras de OTP que le	Luego de estar autenticado un usuario es redireccionado a la página principal, el cual proporciona las 6 muestras de

	un algoritmo *Hash* y la introducción de caracteres aleatorios, las 6 muestras de OTP, las cuales son asignadas ha dicho usuario.		corresponden.	OTP.
ES11: Actualizar muestras de OTP.	Se genera nuevamente las muestras de OTP, a partir de la semilla asignada al usuario y la generación se realiza mediante un algoritmo *Hash* y la introducción de caracteres aleatorios.	-	Se actualiza las muestras que posee el usuario autenticado, proporcionando las nuevas muestras generadas.	Luego de estar autenticado un usuario es redireccionado a la página principal del sistema en el cual, al dar clic en el botón Generate new samples se le porporciona las nuevas muestras de OTP.
ES12: Eliminar muestras de OTP.	Elimina las muestras de OTP.	-	Al eliminar un usuario se elimina su semilla y sus correspondientes muestras, mostrando un mensaje de que ha sido eliminado el usuario.	Luego de estar autenticado se comprueba que el usuario sea administrador, accediendo a través del menú principal se da clic en el icono que representa la eliminación del usuario, se introduce el nombre de usuario, se da clic en el botón Delete user, se comprueba que los datos introducidos pertenezcan a un

				usuario, se muestra una nueva interfaz en la que el sistema confirma la eliminación de dicho usuario dando clic en el botón Delete user, procediéndose a la eliminación de su semilla y a continuación las muestras asociadas a esta.
ES13: Mostrar muestras de OTP.	El usuario visualiza las muestras de OTP generadas.	-	Se proporcionan las muestras de OTP correspondiente al usuario autenticado.	Luego de estar autenticado el usuario es redireccionado a la página principal, proporcionándole las muestras de OTP.
ES14: Guardar muestras de OTP.	Las muestras de OTP son guardadas en un archivo de texto generado.	-	Se ofrece al usuario la posibilidad de guardar las muestras de OTP en un archivo de texto.	Luego de estar autenticado el usuario es redireccionado a la página principal, proporcionándole las muestras de OTP, al dar clic en el botón Save samples, las muestras son guardadas en un archivo de texto generado con el nombre “Temporal.txt” el cual se guarda en “D:/”.

Tabla 32- Descripción del caso de prueba Comprobar OTP. Fuente: Elaboración propia.

Escenario	Descripción	Nombre de usuario	OTP	Respuesta del sistema	Flujo central
ES15: Comprobar OTP.	Se realiza la comprobación de que el OTP introducido se	V pepe_cuba	I 89*POU	Devuelve falso.	Al realizarse la autenticación en una aplicación a través del Sistema
		I	V	Devuelve falso.	

	corresponda con el OTP generado por la aplicación web; en caso de no coincidir, se comprueba que dicho OTP se corresponda con alguna de las muestras de OTP que brinda también la aplicación web; sino se calculan los próximos 10 OTP de forma secuencial y se comprueba en cada uno de los casos si se corresponden con el OTP introducido.	_borja	45SDF6		de Administración de Identidades, este último comprueba que el OTP introducido se corresponda con el usuario autenticado, pudiendo ser esta cadena el OTP actual, una de las 6 muestras o uno de los próximos 10 OTP que le suceden al actual.
		V guido.mola	V W86EDT	Devuelve verdadero.	

Tabla 33- Descripción de las variables del caso de prueba Comprobar OTP. Fuente: Elaboración propia.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre de usuario	Campo de texto	No	Admite letras y de forma no consecutiva underscore y puntos.

2	OTP	Campo de texto	No	Admite letras mayúsculas y números.

Tabla 34- Descripción del caso de prueba Asociar semilla a usuario. Fuente: Elaboración propia.

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central
ES16: Asociar semilla a usuario.	Se realiza cuando se crea un usuario en el sistema, seguidamente se le asocia una semilla o cuando el usuario decide por voluntad propia generar una nueva semilla.	-	-	Luego de estar autenticado se comprueba que el usuario sea administrador, accediendo a través del menú principal se da clic en el icono que representa la creación de un usuario, se introduce el nombre del usuario y se selecciona el rol. Luego se da clic en el botón Add user y es allí cuando se genera una nueva semilla la cual es asignada a dicho usuario.

Tabla 35- Descripción del caso de prueba Integrar mecanismo de autenticación al Sistema de Administración de Identidades. Fuente: Elaboración propia.

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central
ES17: Integrar mecanismo de autenticación al Sistema de Administración de	Se realiza a través de la implementación de librerías y el uso del servicio web, para la	-	-	-

Identidades	integración del mecanismo de autenticación al Sistema de Administración de Identidades.			

Tabla 36- Descripción del caso de prueba Actualizar semilla en el cliente. Fuente: Elaboración propia.

Escenario	Descripción	Semilla	Respuesta del sistema	Flujo central
ES18: Actualizar semilla en el cliente.	El usuario obtiene su semilla a través de la aplicación web y la misma es introducida en la aplicación cliente.	V QWE1R4TY7UI8O9P6	Realiza un cambio de interfaz.	Al seleccionarse la opción Insert new seed en el *tabcontrol* de la aplicación desktop de Windows, y luego de haberse introducido la cadena el sistema comprueba que la longitud de la misma sea de 16 caracteres y además solo contenga letras mayúsculas y números.
		I 789/*-ert	Muestra un mensaje de error.	

Tabla 37- Descripción de las variables del caso de prueba Actualizar semilla en el cliente. Fuente: Elaboración propia.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Semilla	Campo de texto	No	Admite letras mayúsculas y números.

Tabla 38- Descripción del caso de prueba Guardar semilla en el cliente. Fuente: Elaboración propia.

Escenario	Descripción	Semilla	Respuesta del sistema	Flujo central
ES19: Guardar semilla en el cliente.	Después de actualizada la semilla en el cliente, se guarda esta semilla en un archivo predeterminado por los administradores.	V 98TYGH32SD65RT98	Realiza un cambio de interfaz.	Al seleccionarse la opción Insert new seed en el tabcontrol de la aplicación desktop de Windows, y luego de haberse introducido la cadena el sistema comprueba que la longitud de la misma sea de 16 caracteres y además solo contenga letras mayúsculas y números, se guarda la semilla en un archivo de texto.
		I QW)(YU/&+--	Muestra un mensaje de error.	

Tabla 39- Descripción de las variables del caso de prueba Guardar semilla en el cliente. Fuente: Elaboración propia.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Semilla	Campo de texto	No	Admite letras mayúsculas y números.