Universidad de Matanzas "Camilo Cienfuegos"

Facultad de Ciencias Técnicas

Tema: Procedimientos de seguridad para aplicaciones web de gestión para la Universidad de Matanzas

Autor: Jesús Martínez Ruiz

Tutores: Ing. Yeiniel Alfonso Martínez

MSc. Emma Regla Rizo Rizo

Matanzas, Cuba

Junio, 2018

FRASES

“El hombre necesita dificultades porque son necesarias para disfrutar el éxito”.

- A.P.J. Abdul Kalam

“La única forma de hacer un gran trabajo es amar lo que haces”.

- Steve Jobs.

DEDICATORIA

A mis padres, primeramente, ellos por ser los que me dieron la vida, a mi familia y mis buenos amigos.

AGRADECIMIENTOS

A la Universidad de Matanzas Camilo Cienfuegos por darme la oportunidad como estudiante y como diplomado.

Mis tutores Yeiniel y Emma, por el apoyo, dedicación, esfuerzo y fe en mí, como estudiante y persona.

Mis padres Jesús y Lilian, las cosas más importantes para mi vida, que sin ellos no hubiese sido posible nada, por su amor y fe incondicional, comprensión y paciencia durante todo este tiempo.

Mi familia, mis tíos en especial mi súper tía Zoila, mis dos hermanas Miry e Ismarys, mis primos Ernesto y Edel y mis dos abuelos, mis sobrinos y demás allegados.

A Siret, muchas gracias por la preocupación, esfuerzo y dedicación a pesar de todas las adversidades durante este tiempo.

Personas especiales, en particular, Frank, Adonys, Yaciel, Francis, Hookanis, Mayté, Lilian, y otros que siempre me apoyaron y me dieron fuerzas para superarme, también los buenos amigos inolvidables de Radio 26, de la Aduana en el Aeropuerto de Varadero, los de Feria de los Artesanos, y otros miles que siempre tuve presente.

A todos sinceramente… gracias.

DECLARACION DE AUDITORIA

Yo, Jesús Martínez Ruíz, declaro que soy el único autor de este trabajo y autorizo a la Universidad de Matanzas “Camilo Cienfuegos”, especialmente a la Facultad de Informática, a que hagan el uso que estimen pertinente de él.

Y para que así conste, firmo la presente a los 27 días del mes de noviembre del 2017.

_____ _____

Firma del Autor Firma del Tutor

OPINION DEL TUTOR

Datos personales del tutor

Nombre y apellidos:

Centro de trabajo: Facultad de Informática, Universidad de Matanzas “Camilo Cienfuegos”

Organismo al que pertenece: Ministerio de Educación Superior – MES.

Cargo que ocupa:

Especialidad de la que es graduado:

Categoría docente o investigativa: Instructor recién graduado.

Datos de la tesis y el diplomante:

Nombre y apellidos: Jesús Martínez Ruiz

Centro de estudio: Facultad de Informática, Universidad de Matanzas "Camilo Cienfuegos"

Título de la Tesis:

Opinión sobre el trabajo:

RESUMEN

La seguridad informática en la actualidad es un tema que afecta tanto al mundo en general como a Cuba. Este trabajo expone con datos estadísticos las brechas de seguridad en el desarrollo del software de los proyectos finales en la Universidad de Matanzas Camilo Cienfuegos. Para mejorar el desarrollo de la ciberseguridad en la creación de softwares de aplicaciones web de gestión en los futuros proyectos, se seleccionan patrones de seguridad, estrategias, configuraciones específicas y se agrupan en un modelo de procedimientos de seguridad para luego proponer la aplicación de los mismos paralelamente con el desarrollo del modelo del negocio independientemente de la tecnología que se elija para el desarrollo del software. Desde el despliegue físico, la gestión de la información en las bases de datos, la seguridad en la aplicación web y otros sectores específicos la seguridad y su funcionamiento es demostrado en este trabajo.

RESUME

In the current living, cyber security is a subject that affects the entire world in general, also in Cuba. This work exposes with statistic data security breaches of the security appliance in the software development in the Matanzas University Camilo Cienfuegos final thesis work. To improve the development of cybersecurity on future thesis works, some patterns, specific security configuration procedures are selected and grouped into a security procedure model to propose them to be applied at the same time with the business model development despite the framework or selected technology to develop. Since the physical deployment, information management on databases, security on the web application and some other specific areas security and functions on this work are tested.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Con el auge de la revolución tecnológica dentro del mundo industrial, social y empresarial, nacen nuevas tecnologías informáticas, desarrollándose Sistemas informáticos para el procesamiento electrónico de la información. En la actualidad no se concibe el desarrollo de ninguna esfera de la actividad humana sin la utilización de las Tecnologías de la Información y las Comunicaciones (TIC). Dicho progreso tecnológico, aumenta la demanda a soluciones de seguridad complejas en el desarrollo de las aplicaciones previamente a ser comercializadas. Por ende, todo sistema debe fortalecer aspectos de seguridad para defenderse de ataques maliciosos, proteger el acceso a la información y prevenir pérdidas o mala manipulación de la misma. Algunas estadísticas mundiales según Verizon Investigations Reports (Verizon) muestran cómo es afectada la seguridad en distintas áreas del mundo demostrando que hasta las grandes empresas son vulnerables.

En el mundo actual ningún sistema es seguro, por lo que las brechas a nivel mundial afectan a todo sector industrial, empresarial y social, ocasionando la pérdida de billones de dólares en información. En un escrito del sitio web cubano Cuba debates (Cubadebate, 2017) el alcance de las tecnologías de la información lleva hoy nuevos delitos como la usurpación de la identidad, fenómeno que ocasionó la pérdida de 100 mil millones de dólares en los últimos 6 años. Por otra parte el periódico Trabajadores (Trabajadores, 2017) expone que Cuba ha recibido más de 600 incidentes de este tipo. Esto demuestra que no estamos exentos como objetivo a penetrar y corromper nuestros sistemas, por eso es necesaria una adecuada educación en cuanto al desarrollo de la informatización en paralelo con la seguridad de la misma.

La seguridad dentro del desarrollo de aplicaciones web en redes tipo intranet tiene un peso esencial cuando diseñamos, programamos y compilamos un proyecto para luego su comercialización o uso, debemos tener en cuenta diversos aspectos. **¿Qué es la seguridad dentro de un software?** Según la comunidad de seguridad de software libre y abierta (OWASP, 2018) (The Open Web Application Security Project) son diferentes tipos de medidas que los programadores de aplicaciones tienen en cuenta y aplican, de forma independiente en toda área del desarrollo del software, para proteger el acceso a la información y prevenir pérdidas o mala manipulación de la misma. Desde el inicio del diseño de un software, la seguridad comienza a formar parte del desarrollo, por ende, es necesario, en dependencia del lenguaje, entorno, para quién o dónde será utilizado, las características de seguridad que puedan atentar con la estabilidad y longevidad de

un software. La seguridad en general debe ser diseñada y definida paralelamente con la del modelo del negocio. Elaborando una serie de procedimientos mediante objetivos específicos y generales al relacionarlos como requisitos no funcionales para cubrir así cada posible vulnerabilidad que atente contra la estabilidad de la aplicación y por ende la del sistema, modelo del negocio y la del usuario.

La universidad de Matanzas en el año 2000 crea la carrera de Ingeniería Informática, donde se forman los especialistas que trabajaran en las diferentes empresas de la provincia, se abre el curso regular diurno con un grupo de 30 estudiantes en ese primer año, creando una base tecnológica para garantizar la formación de los mismos, esta entidad realizó su primera graduación en el año 2005 mediante trabajos de diplomas como ejercicio de culminación de estudios de los Ingenieros Informáticos que son utilizados en la informatización del territorio, esto es válido hasta la fecha.

Como se explica anteriormente la seguridad es utilizada para proteger el acceso a la información y prevenir pérdidas o mala manipulación de la misma, lo que garantiza su credibilidad en el momento de toma de decisiones. Al realizar un estudio desde los años 2008 al 2017 a 142 trabajos de diplomas de desarrollo de software mediante la fórmula básica de muestreo (Domenech, 1999), de un total de 267, se detectó que no se garantizaba la protección de los datos, desde la programación, encontrándose déficits en distintas áreas como, bases de datos, aplicación web de manera interna y externa, y otras vinculadas al modelo del negocio. Del total de los trabajos analizados para esta investigación, se obtiene resultados como que de entre más de 20 aspectos de seguridad evaluados, menos del 50% de los trabajos hacen uso de los puntos de seguridad evaluados en esta investigación, es por este motivo que se realiza la misma, teniendo como p**roblema científico** a resolver: ¿Cómo mejorar la seguridad de las aplicaciones web para entornos de intranet en las tesis de la carrera de Ingeniería Informática durante el proceso de desarrollo?

Teniéndose como **objeto de estudio la** seguridad basados en patrones y estrategias de seguridad de las aplicaciones web.

Como **campo de acción:** la seguridad de las aplicaciones web desarrolladas en las tesis de la carrera de Ingeniería Informática para entornos de intranet en la Universidad de Matanzas.

Para dar respuesta al problema presentado se elabora la siguiente **hipótesis**:

El uso de procedimientos de seguridad basados en patrones y estrategias de seguridad durante el proceso de desarrollo de las aplicaciones web mejorará la seguridad del software presentados como parte de las tesis de la carrera de Ingeniería Informática de la Universidad de Matanzas.

Esta tesis tiene como **objetivo general**

Desarrollar procedimientos basados en patrones y estrategias de seguridad orientado a la mejora de la seguridad de las aplicaciones web de las tesis de la carrera de Ingeniería Informática para entornos de intranet durante su proceso de desarrollo en la UMCC

Objetivos específicos:

- Realizar un estudio de la situación actual de la implementación de la seguridad de las aplicaciones web de gestión desarrolladas en los trabajos de diploma publicadas en pasados años.
- Realizar un estudio de los distintos tipos de seguridad que pueden ser aplicados en las distintas áreas de las aplicaciones web, bases de datos y otros sectores, desde la fase de diseño.
- Proponer los pasos que integran los procedimientos partiendo de modelos, patrones y estrategias de seguridad orientado a la mejora de la seguridad, agnóstico a la tecnología que se implemente.
- Demostrar el uso de los procedimientos de seguridad creados en una aplicación web de gestión.

Se obtienen estadísticas sobre la situación actual de cómo se aplica la seguridad en los proyectos de tesis de los egresados obteniendo positivos resultados para el desarrollo de esta investigación.

La investigación se estructura de la siguiente forma:

- **Capítulo 1:** Fundamentación Teórica y Tendencias Tecnológicas: Se abordan aspectos teóricos, estadísticas previas y actuales de seguridad en los trabajos de tesis presentados, así como estadísticas mundiales. Se realiza un estudio de las herramientas, métodos, patrones, estrategias, etc. utilizadas para la seguridad.

- **Capítulo 2:** Propuesta de Modelo de Seguridad: se explican elementos aplicados para dar solución a la propuesta de utilización de procedimientos para

incrementar la seguridad en las aplicaciones web. Se abordan los métodos y estrategias.

- **Capítulo 3:** Análisis de los resultados: se realiza un análisis a trabajos presentados en años anteriores buscando cómo se aplica la seguridad y se sacan estadísticas. Se muestran resultados de la aplicación DEMO “SMS” al aplicarle los procedimientos de la seguridad.

CAPITULO 1: Fundamentación teórica y tendencias tecnológicas

1.1. Introducción

En el presente capítulo se hace un análisis de las características de la Universidad, al profundizar en los problemas de seguridad que presentan las tesis que se desarrollan para la culminación de estudio de los ingenieros informáticos, lo que hace necesario la creación de procedimientos que facilitan su aplicación. Se explican una serie de conceptos y elementos básicos sobre seguridad que permiten entender los principios de los mismos.

1.2. Seguridad Informática

La seguridad de tecnologías de información o ciberseguridad es el área de la informática que reúne la confiabilidad, integridad, disponibilidad y responsabilidad de la información en contra de acceso y manipulación sin autorización (Dangler, 2013) y mediante el diseño de normas, métodos y técnicas destinados a conseguir un sistema de información óptimo. Las funciones de la ciberseguridad son la protección de la infraestructura informática, los usuarios que utilizan y gestionan dicha información y la información que es manipulada por el sistema y los usuarios. Tradicionalmente se caracteriza como un RNF (Requisito No Funcional) que regula el funcionamiento del sistema y debe ser trabajar en paralelo con el modelo del negocio al desarrollar una aplicación informática. (Security vulnerabilities of the top ten programming languages, 2013).

La seguridad informática se ha convertido en el talón de Aquiles de las empresas, como se explica en la introducción de esta investigación constantemente se están recibiendo ataques a los equipos y software de las mismas, la compañía Verizon en el 2016 realizó una investigación donde se demuestra cuán vulnerable es. Ninguna localidad, industria u organización es a prueba de balas cuando se trata de brechas en la información. A continuación, se muestran gráficas de estadísticas de brechas de seguridad.

2015-2016 (Verizon, 2016)**:** En los reportes del 2016 presentan las zonas en el planeta donde han ocurrido los incidentes cibernéticos a lo largo de diversas e innumerables compañías llegando a una cifra de más de 80 países.

En la figura 1.8 se muestran las zonas del mundo más atacadas cibernéticamente, demostrando que es un problema global,

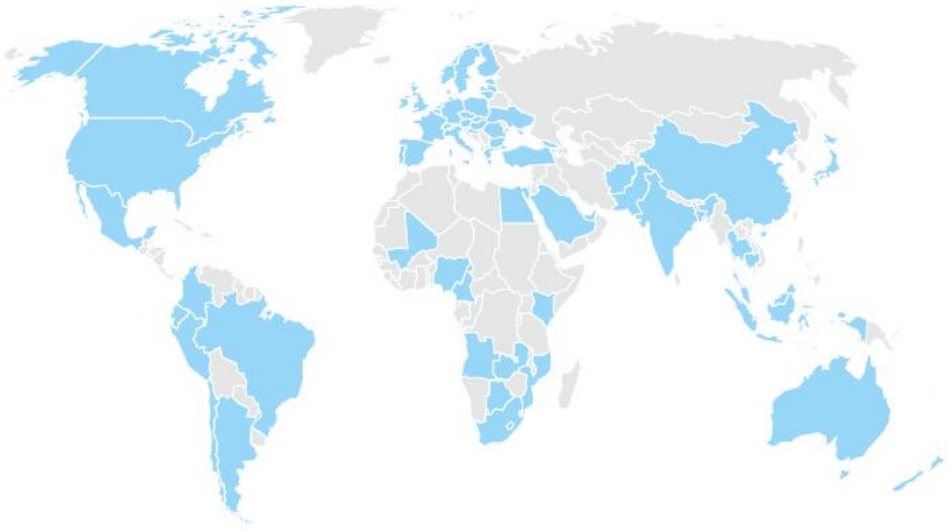

Figura 1.8 Regiones víctimas de ciberataques

Incidentes de seguridad por industria u organización con pérdidas de datos (2015)

Tabla 1-5 Cantidad de incidentes por industria u organización con pérdidas de datos

INDUSTRIA	TOTAL	PEQUEÑO	GRANDE	DESCONOCIDO
Alojamiento (72)	**282**	136	10	36
Administrativo (56)	**18**	6	2	10
Agricultura (11)	**1**	-	-	1
Construcción (23)	**4**	-	1	3
Educacional (61)	**29**	3	8	18
Entretenimiento (71)	**38**	18	1	19
Finanzas (52)	**795**	14	94	687
Cuidados médicos (62)	**115**	18	20	77
Información (51)	**194**	12	12	170
Dirección (55)	**-**	-	-	-
Manufacturado (31-33)	**37**	5	11	21
Minería (21)	**7**	-	6	1
Otros servicios (81)	**11**	5	2	4
Profesional (54)	**53**	10	4	39
Público (92)	**193**	4	122	67
Inmuebles (53)	**5**	3	-	2
Ventas (44-45)	**137**	96	12	29
Comercio (42)	**4**	2	2	-
Transportación (48-49)	**15**	1	3	11
Útiles (22)	**7**	-	-	7
Desconocido	**270**	109	-	161
TOTALES	**2,215**	**442**	**310**	**1,463**

- **Pequeño**: organizaciones con menos de 1000 empleados.
- **Grande**: organizaciones con más de 1001 empleados.

2017 (Verizon)

- **61%** de las víctimas de las brechas de información en el reporte del 2017 son empresas de negocios con más de mil empleados.
- **95%** de los ataques tipo Phishing (Suplantación de identidad) fueron producto a instalaciones de software.
- **88%** de las brechas informáticas caen dentro de los 9 patrones identificados en el 2014 por los que se rigen para la seguridad.

¿Qué tácticas son usadas?

- **62%** - Hacking.
- **51%** - Malware.
- **81%** - Contraseñas débiles e inseguras.
- **43%** - Ataques sociales.
- **14%** - Abuso de privilegios.

1.3. Antecedentes de la investigación

Se evaluaron tesis realizadas en la universidad de Matanzas para la culminación de estudios de los Ingenieros informáticos, búsquedas en Internet y no se encontró ninguna tesis que trate la seguridad con este enfoque. (Barnes, 2009) (Dangler, 2013).

Comportamiento de la seguridad en las aplicaciones realizadas en las tesis de la carrera de Ingeniería Informática en la Universidad Camilo Cienfuegos

Se realizó un estudio a un gran número de tesis de la carrera de ingeniería informática teniendo en cuenta como fue tratada la seguridad en las aplicaciones web realizadas en las mismas. Se analizaron seleccionaron aquellas tesis que en el cuerpo de la tesis apareciera la palabra seguridad, para el estudio estadístico realizado se eligieron un total de 267 tesis desde el año 2008-2017:

Tabla 1-6 Cantidad de trabajos elegidos. Propio del autor.

Año	**2008**	**2009**	**2010**	**2012**	**2014**	**2015**	**2016**	**2017**	**Total**
Cantidad	3	32	46	59	8	62	42	15	267

Cálculo mediante función básica de muestreo:

La función básica del muestreo (herramienta de la investigación científica) (Domenech, 1999) es determinar que parte de una realidad en estudio (población o universo) debe examinarse con el fin de hacer inferencias sobre dicha población. Obtener una muestra adecuada significa lograr una versión simplificada de la población, que reproduzca de algún modo sus rasgos básicos.

Por tanto, del total de 267 tesis se realiza un estudio mediante la fórmula:

$$n = \frac{N \cdot Z_{\propto}^2 \cdot p \cdot q}{d^2 \cdot (N-1) + Z_{\propto}^2 \cdot p \cdot q}$$

Figura 1.9 Fórmula para hallar la cantidad de tesis a revisar.

Donde:

- **n:** es el tamaño de la nuestra.
- **Z**: valor correspondiente a la distribución de Gauss (1,962 – nivel de confianza 95%).
- p: es prevalencia o proposición esperada del parámetro a evaluar, como se desconoce se aplica la opción más desfavorable (p = 0.5).
- q: 1-p (en este caso 1 – 0.5 = 0.5).
- i: es el error que se prevé cometer (precisión, en este caso se espera un 0.5% de error).

A continuación, se muestran los datos para utilizar la formula y obtener el número de tesis final a revisar.

Tabla 1-7 Datos para hallar los datos. Propio del autor.

Muestra		
Z	1.962	95%
p	0.5	
q	0.5	
d	0.05	
N	225	
n	**142.2338**	**142**
		63%

Aplicando la función básica de muestreo se seleccionaron **142** tesis a revisar de las **267** obteniendo como resultado la distribución por año que muestra la tabla 3, que representa el **63%** de todas las tesis.

Tabla 1-8 Datos de tesis a revisar por año. Propio del autor.

	Año	Cantidad	
Gestión	**2008**	3	100%
Gestión	**2009**	27	84%
Gestión	**2010**	32	78%
Calidad		4	
Gestión	**2012**	35	61%
Calidad		1	
Gestión	**2014**	8	100%
Gestión	**2015**	61	98%
Gestión	**2016**	39	93%
Gestión	**2017**	15	100%
		225	**84%**

	Promedio
	17.75
2008	0
2009	3
2010	0
	4
2012	11
	1
2014	8
2015	61
2016	39
2017	15
	142

Por lo tanto, por cada año se eligen un promedio de 17 tesis

Para seleccionar las tesis a evaluar se toman criterios con preferencias para aplicar un control más objetivo a la selección. A continuación, se muestran el orden y métodos de selección.

1. **Todas las tesis de calidad:** Se selecciona de forma objetiva las tesis de calidad, con el enfoque de que deben ser las de más rigor en niveles de diseño de todo tipo de factores en su desarrollo del software.

2. **Tesis más recientes:** se hace esta selección suponiendo que la seguridad o la tecnología es la mejor de años actuales hacia atrás años más antiguos.
3. Si queda alguna otra entonces entra dentro de la revisión.

De la investigación realizada a las tesis se logran algunos resultados satisfactorios para el desarrollo de este material. Se recopilan algunas estadísticas de cómo la seguridad fue evaluada. Se analizan todas las tesis principalmente las secciones de Etapa de Diseño del Software, las pruebas de aceptación y Herramientas utilizadas.

Puntos a evaluar en los Trabajos de Diploma: para la evaluación en los trabajos de tesis se tienen en cuenta métodos, patrones de seguridad, estrategias generales y especificas en distintas áreas dentro del desarrollo de aplicaciones web y bases de datos.

Bases de Datos:

- Gestión de la seguridad en las bases de datos.
- Utilización de cualquier tipo de Triggers.
- Utilización de scripts.
- Servicio de Correos.
- Creación Backups.

Aplicaciones:

- Patrones de seguridad, para este caso se eligieron cerca de 20 patrones explicados previamente, y se verificaron en los documentos si existían. También se admitieron la existencia de otros no evaluados.
- Gestión tipos de Logs.
- Cookies.
- Doble factor de autentificación como CAPTCHAS.
- Https.
- Despliegue Físico.
- Expresiones Regulares.

Generales:

- Búsqueda de las palabras “Seguridad”, “Patrones”, en todas las tesis para ver si realizaban mención de ellos.
- En caso de aparecer alguna mención o aplicación de seguridad, se buscaba en las historias de usuarios, así como en las pruebas de aceptación y resultados esperados.

1.4. Fundamentación Teórica

Según la definición de seguridad Informática expuesta anteriormente existen muchas tecnologías, patrones y estrategias que pueden ser aplicadas y desarrolladas dentro de nuestras aplicaciones web de gestión. A continuación, se explican todas las áreas dónde y cómo la seguridad puede ser tratada.

Como menciona Dangler (Dangler, 2013) para lograr los objetivos de la ciberseguridad se fundamenta en 4 principios que se explican a continuación con sus categorías:

Confiabilidad: es la protección de la información sensible de una compañía de un acceso sin autorización.

- ***Autentificación***: se refiere a la identidad del Usuario o una Entidad.
- ***Autorización***: es el acceso concedido mediante una contraseña segura a un Sistema de recursos como cuentas, archivos o procesos. Es otorgada después que los usuarios o entidades se autentifican y su comportamiento puede ser definido mediante permisos o roles dentro del sistema al que acceden.

Integridad: es la protección de la información de la empresa desde una modificación sin autorización.

- ***Completitud:*** es el almacenado de toda la información que el Sistema requiere para funcionar.
- ***Precisión***: es el grado al cual los modelos de información son “true” o valores correctos.
- ***Actualización (Timeliness)***: es el tiempo de actualización de la información del Sistema.
- ***Validación***: es la validación de la información, relativa a las entidades que abarcan el modelo al cual es regido.

Disponibilidad: es la protección de los datos de una empresa donde los mismos se vuelven indisponibles para usuarios autorizados.

- ***Usabilidad***: es la cantidad de tiempo y esfuerzo que utiliza el programa.
- ***Fiabilidad***: es la habilidad del Sistema para funcionar ante adversas circunstancias, incluyendo ataques hackers y fallas del Sistema.
- ***Compatibilidad***: es el grado de seguridad el cual el software opera en un entorno dado. Esto incluye la ocultación de información sensible de usuarios que no tienen autorización para manipular dicha información.

Responsabilidad: es la asociación de acciones que afectan los datos de la empresa con las personas que realizan esas acciones.

- ***Logging***: son las salvas de las acciones realizadas por el software del Sistema.
- ***Monitoreo***: es el estado de vigilancia del Sistema por acciones que pueden requerir notificaciones a una tercera persona.
- ***Reportes***: son las informaciones pertinentes hacia la tercera persona.

Para la elaborar algunos de los requisitos no funcionales enfocados a la ciberseguridad se puede, mediante algunas preguntas establecer ideas para el comportamiento en las distintas categorías y así encontrar distintas soluciones viables para el desarrollo de las mismas.

Autentificación:

- ¿Debemos autentificar al Usuario antes que acceda al Sistema?

Autorización:

- ¿El Sistema requiere que el Usuario sea autorizado para que pueda acceder a la información antes de permitirle al usuario manipular información sensible?

Completitud:

- Por cada tipo de datos que el Sistema almacena, ¿bajo qué circunstancias la información debe ser almacenada por completo?
- Por los tipos de datos que en ocasiones pueden ser almacenados en partes, ¿Cuándo es recomendado hacerlo?
- ¿Qué grado de completitud es aceptable?

Precisión:

- ¿Qué tan preciso, en dígitos significantes, deben ser los valores numéricos almacenados en el Sistema?
- ¿Los nombres deben incluir segundos nombres?

Actualización (Timeliness):

- Para cada tipo de datos que el Sistema presenta a sus usuarios, ¿bajo qué circunstancias debe el Sistema presentar la más reciente versión de esos tipos a sus usuarios?
- Para esos tipos de datos por los cuales el Sistema puede presentar una versión antigua de los datos, ¿qué tan antigua debe ser esa versión?

Validación:

- Para cada tipo de dato que el Sistema almacena, ¿bajo qué circunstancias puede esos datos pueden ser inválidos?

***Logging*:**

- ¿Qué eventos pueden ocurrir dentro del Sistema? Si es así, ¿Se debe llevar un registro?

Monitoreo:

- Para cada tipo de recurso que el Sistema soporta, ¿el sistema debe proveer monitoreo esos recursos?
- ¿Qué certera y oportuna la información puede ser?

Reportes:

- ¿A cuáles terceras personas, si existe alguna, el sistema debe reportar eventos u otra información sobre operaciones dentro del sistema?
- ¿Qué información debe reportar?

Usabilidad:

- ¿El Sistema debe permitir que los usuarios vean funcionalidades que ellos no están autorizados a usar?
- ¿Esos usuarios están autorizados a acceder a esta funcionalidad, entonces se informa cuando hagan cuando usen alguna funcionalidad sin autorización?

Fiabilidad:

- ¿Bajo qué circunstancias es aceptable para el Sistema que sea atacado de forma “offline” en un tiempo de duración de tiempo?

Compatibilidad:

- ¿En qué entornos (Linux, Windows, etc.) el Sistema debe correr?

1.5. Protocolos HTTP y HTTPS

Cuando pedimos un recurso en la internet o intranet mediante los navegadores enviamos una petición que tiene una sintaxis que se decodifica, nos localiza el recurso mediante la URL y envía al usuario la respuesta de la petición con información del recurso o del error si no procede. Las seguridades en las comunicaciones en las transmisiones de los datos también requieren nuestra atención para el desarrollo de nuestra investigación. A continuación, se da una explicación de cómo es su funcionamiento, características y las mejores prácticas basadas en las medidas de seguridad. (Universidad Nacional Experimental del Táchira (UNET))

HTTP: El protocolo de red HTTP (Hyper Text Transfer Protocol o Protocolo de Transferencia de hipertexto) que permite publicar páginas Web o HTML para luego los usuarios puedan visualizarla. Es la base con la cual está fundamentada la Internet o WWW y por ello es el protocolo más usado en la misma. Desarrollado por la World Wide Web Consortium y la Internet Engineering TaskForce publicando la versión 1.1 HTTP donde define la sintaxis y semántica utilizados por los elementos de software de la arquitectura web (clientes, servidores, proxies, etc.) para comunicarse.

Características:

- Su funcionamiento es a través de solicitudes y respuestas entre un cliente y un servidor creando así una sesión HTTP para cada secuencia de comunicación.
    - **Cliente**: efectúa la petición (un navegador web) se le denomina "useragent" (agente del usuario).
    - **Información**: denominada recurso y es identificada y transmitida mediante un localizador uniforme de recursos (URL).
- La información de las páginas mostradas en los navegadores se encuentra en la “Barra de Navegación” en nuestros navegadores de Internet que comienza con “http://” y se le conoce como URL (Uniform Resource Locator o Localizador de Recursos Uniforme).
- No guarda información sobre conexiones anteriores por lo que existen las cookies almacenadas en el PC del usuario.
- La comunicación se realiza en 2 etapas:
    - El navegador realiza una solicitud HTTP.
    - El servidor procesa la solicitud y después envía una respuesta HTTP.

Estructura de solicitudes (SSL, 2018):

- **Las cabeceras**: incluyen en la primera línea el método que queremos invocar (GET, POST, etc.)
- **Las cabeceras generales**: son las que se aplican tanto a peticiones como a respuestas, pero no al contenido que se transmite.
- **Las cabeceras de petición**: permiten al cliente pasar información al servidor sobre la petición y sobre el cliente.

Las cabeceras de entidad: permiten definir información adicional sobre el contenido que se transmite y en caso de que no haya contenido, sobre el recurso al que se quiere acceder con la petición.

Ejemplo de uso: (Mozilla, 2018)

Tabla 1-1 Ejemplos de solicitudes y respuestas HTTP. Propio del autor.

Petición de recurso	Respuesta del servidor:
GET /index.html HTTP/1.1 Host: www.example.com User-Agent: nombre-cliente [Línea en blanco]	HTTP/1.1 200 OK Date: Fri, 31 dec 2003 23:59:59 GMT Content-Type: text/html Content-Length: 1221 <html> ……. </html>

Seguridad en transmisión de datos: (Symantec, 2018)

- ¿Los datos viajan de forma insegura?
- ¿Puede alguien ver nuestros datos y manipularlos?

La respuesta a eso es **SI**, usando solamente HTTP estamos corriendo riesgos como:

- Intercepción y modificación de la información.
- Envío y recepción de virus informáticos.
- Suplantación de Identidad (Phishing) para robar información, dinero, etc.

HTTPS: Desde el 2008 hacia la actualidad se adoptó como un estándar web, el HTTPS (Hyper Text Transfer Protocol Secure) que es la versión segura de HTTP que dispone de protocolos criptográficos llamados SSL (Secure Socket Layer) y TSL (Transport Layer Security) siendo el ultimo la versión más reciente. El intercambio en la comunicación se realiza mediante el puerto 443. Cualquier tipo de servicio que requiera el envío de datos personales y/o contraseñas pues se proteger el tráfico de Internet con el fin de impedir que otros usuarios de la red puedan espiarlo o alterarlo.

Configuración del Servidor HTTPS: Para preparar un servidor web que acepte conexiones HTTPS, el administrador debe crear un certificado de clave pública para el servidor web. Este certificado debe estar firmado por una autoridad de certificación para que el navegador web lo acepte. La autoridad certifica que el titular del certificado es quien dice ser.

Ventajas:

- **Encriptación**: si algún atacante consigue interceptar esa información, no le servirá para nada ya que no sabrá descifrarla, solamente quien la implemente.
- **Integridad de datos**: los atacantes no podrán “modificar” el contenido del mensaje enviado.

- **Autenticación**: se evitan los ataques de suplantación de identidad o intermediarios (*"man in the middle"*) en el que tu usuario proporciona información a terceros cuando cree que te los está dando a ti.

1.6. Seguridad Física de Despliegue usando N-Tier

Teniendo en cuenta la seguridad física en la configuración de nuestros servidores y servicios físicos de red, se recomienda entre diversas configuraciones, el uso de niveles de seguridad o N-Tier. No confundamos N-Tier con N-Capas, que, aunque usan conjunto de nombres similares como Presentación, Servicios, Negocios y Datos, es importante no confundirlos. (Cherencio, 2009).

N-Tier o Niveles se refiere a la distribución física de componentes y funcionalidades en servidores separados, teniendo en cuenta topología de redes y localizaciones remotas. A continuación, un ejemplo gráfico de N-Tier.

Niveles físicos de despliegues (Tiers): (Naylor, 2016) (TrendMicro, 2007)
Representan separaciones físicas de las funcionalidades de representación, negocio y datos de diseño en diferentes maquinas usadas como servidores como servidores (para lógica de negocio y bases de datos) y otros sistemas (PC's para capas de presentación remotas, etc.) Los patrones de diseño comunes basados en niveles son "2-Tier", "3-Tier" y "NTier".

2-Tier: Este patrón representa una estructura básica con dos niveles principales, un nivel cliente y un servidor de bases de datos. En un escenario web típico, la capa de presentación cliente y la lógica de negocio co-existen normalmente en el mismo servidor, el cual accede a su vez al servidor de bases de datos. Así pues, en escenarios Web, el nivel cliente suele contener tanto la capa de presentación como la capa de lógica de negocio, siendo importante por mantenibilidad que se mantengan dichas capas lógicas internamente.

3-Tier: El usuario interacciona con una aplicación cliente desplegada físicamente en su máquina (PC, normalmente). Dicha aplicación cliente se comunica con un servidor de aplicaciones (Tier Web/App) que tendrá embebidas las capas lógicas de lógica de negocio y acceso a datos. Finalmente, dicho servidor de aplicaciones accede a un tercer nivel (Tier de datos) que es el servidor de bases de datos. Este patrón es muy común en todas las aplicaciones Rich-Client, RIA y OBA. También en escenarios Web, donde el cliente sería 'pasivo', es decir, un simple navegador.

N-Tier: En este escenario, el servidor Web (que contiene la capa de lógica de presentación) se separa físicamente del servidor de aplicaciones que implementa ya exclusivamente lógica de negocio y acceso a datos. Esta separación se suele hacer normalmente por razones de políticas de seguridad de redes, donde el servidor Web se despliega en una red perimetral y accede al servidor de aplicaciones que está localizado en una subred diferente, separados probablemente por un firewall. También es común que exista un segundo *firewall* entre el nivel cliente y el nivel Web.

Ventajas:

- La elección de niveles/tiers separando capas lógicas de nuestra aplicación en niveles físicos separados, impacta en el rendimiento de las aplicaciones (debido a la latencia de las comunicaciones remotas entre los diferentes niveles).
- También puede mejorar la seguridad el separar los componentes más sensibles de la aplicación a diferentes redes.

Desventajas:

- La adición de niveles/tiers incrementa la complejidad de los despliegues y en ocasiones impacta sobre el rendimiento, por lo que no se deben añadir más niveles de los necesarios.

1.7. Bases de Datos

Las bases de datos son las tecnologías dentro del mundo de la informática que nos permiten la utilización y re-utilización de infinitos recursos los cuales son almacenados en los servidores dentro de una entidad estatal o privada alrededor del planeta. Dichos recursos se convierten en toda la información que las redes muestran usuario interactuar con sus distintas interfaces. (Barnes, 2009) (Microsoft, 2017)

Es una tendencia del futuro que nuestras vidas dependan cada vez más de las tecnologías y la informatización de todo tipo de servicio usado por el hombre en la actualidad. Almacenadas éstas en las bases de datos estaremos poniendo en riesgo nuestro modo de vivir, así como nuestras cuentas bancarias, información personal, centros de trabajos, investigaciones, etc.

A continuación, se ponen de manifiesto medidas, vulnerabilidades y algunas de las más importantes sugerencias para mantener una base de datos segura.

Requerimientos de seguridad comunes en bases de datos: (Williams, et al.) (Database Security, 1999)

Tabla 1-1 Requerimientos de seguridad comunes en bases de datos. Propio del autor.

1	**Integridad de la Base de Datos Física y Lógica**

	• **Habilidad para manejar problemas físicos**:
	  ○ Fallas de electricidad.
	  ○ Destrucción física (fuego, agua).
	• **Habilidad para procesar transacciones correctamente.**
	  ○ Recuperación (Logs, Backups)

2	**Integridad**

	• **Asegurarse que la información contenida sea correcta mediante:**
	  ○ Chequeo de campos.
	  ○ Chequeos referenciales.
	  ○ Triggers.
	  ○ Procedimientos almacenados.
	• **Control de Multiusuario (Concurrencia / Consistencia):**
	  ○ ¿Qué sucede cuando múltiples usuarios acceden al mismo recurso?
	• **Seguros o Candados de la Base de Datos:**
	  ○ Compartido.
	  ○ Autorizado a la table, pagina o nivel de fila.

3	**Auditoría**

	• ¿Quién accedió al Sistema y qué hizo una vez logueado?
	• ¿Qué eventos necesitamos monitorear?
	• **Para ello se usan**:
	  ○ Triggers
	  ○ Shadow tables
	  ○ Monitorear campos (created_by, created_date, etc.)

4	**Control de Acceso**
	• **¿Quién puede ver y hacer en el Sistema?** • **Solución:** ○ Uso de roles y asignarle permisos a los usuarios.
5	**Autentificación de Usuarios**
	• Cada usuario debe ser identificado. • El uso de contraseñas con seguridad. • Chequeo la hora de autentificación del usuario.

6 Disponibilidad

- Los usuarios deben ser capaces de acceder a la Base de Datos lo requieran.
- ¿Cuáles son los requerimientos de tu Sistema?
    - Tiempo.
    - Rendimiento.

7 Principio de Menos Privilegios (Principle of Least Privilege)

- Otorgar a los usuarios los privilegios mínimos hasta donde amerite su rol.
- Si un servicio “X” no necesita acceso a todas las tablas en base de datos “Y”, no otorgar ese acceso a todas las tablas.
- No otorgar privilegios a cuentas que no los necesiten.

8 Contraseñas Fuertes y Seguras

- **CWE 521**: Requerimientos para contraseñas débiles.
- **Longitud**:
    - Cada carácter que se agregue incrementa la protección de la contraseña.
    - 8 o más caracteres son el requerimiento mínimo para una contraseña fuerte, 14 caracteres o más es ideal.
- **Complejidad:**
    - Combinar caracteres especiales conjunto con mayúsculas solidifica aún más la seguridad de la contraseña.
- Evitar contener el nombre de usuario, debe ser distinto a la contraseña.
- **Expiración:**
    - **CWE 262:** Se debe usar expiración por tiempo de contraseña.
- No reusar contraseñas.

9 Multi-nivel de Acceso

- Los usuarios deben garantizar acceso tipo “top secret” “secret” “confidential” o “unclassified” access (en acceso decreciente).
    - **Ejemplo**: los tipos de accesos u qué pueden ver.

MID	Nombre	Apellido	Clase
101	Jesus	Martinez	TS
102	Clara	Martinez	S
103	Frank	Formoso	U

10 Seguridad Interna de la Información

- Limitar campos que no necesiten ser consultados.
- Limpiar / Crear datos anónimos.
- Encriptar información.

1.8. Uso de Triggers

Se propone el uso de los triggers (Disparadores), que son un tipo especial de “Stored Procedure” o procedimiento almacenado que se ejecuta automáticamente cuando un evento ocurre en la base de datos del servidor. Existen diferentes tipos de triggers como son DML (Data Definition Language), DDL (Data Definition Language) y los triggers Logon. (Microsoft, 2011) (Lobel, et al., 2012).

DML triggers

Se ejecuta cuando el usuario trata de modificar datos a través de un evento DML. Los eventos DML son sintaxis INSERT, UPDATE, o DELETE en una tabla o vista. Estos triggers son activados cuando cualquier evento valido es activado, sin importar o no si las filas de la tabla son afectadas.

DDL triggers

Se ejecutan en variedad de la de los eventos DLL. Estos eventos corresponden a sintaxis CREATE, ALTER, y DROP y ciertos procedimientos almacenados de Sistema que realizan operaciones DDL.

Logon triggers

Ejecutan Stored Procedures en respuesta a un evento LOGON. Este evento es elevado cuando una sesión de usuario es establecida con una instancia de SQL Server.

Estructura básica:

- Una llamada de activación, la cual es una sentencia que permite la ejecución del código.
- Una condición necesaria para que se realice el código.
- La secuencia de instrucciones a ejecutar una vez que se han cumplido las condiciones iniciales.

Ventajas

- La entrada en vigor automática de restricciones de los datos, hace que los usuarios entren sólo valores válidos.
- El mantenimiento de la aplicación se reduce, el cambio a un disparador se refleja automáticamente en todas las aplicaciones que tienen que ver con la tabla sin la necesidad de recompilar.
- Logs automáticos de cambios a las tablas.

- La notificación automática de cambios a la Base de Datos con alertas de evento en los disparadores.
- Un trigger se puede ejecutar antes (**BEFORE**) o después (**AFTER**) de que sean modificados los datos usando dos palabras clave, **OLD** y **NEW** que se refieren a los valores que tienen las columnas antes y después de la modificación. Los INSERT permiten NEW, los DELETE sólo OLD y los UPDATE ambas.

1.9. Backups

Mantener las salvas de las bases de datos mediante copias de seguridad cada cierto tipo y de forma automática es muy importante, para ello SQL Server tiene funciones configurables para hacer respaldos de las bases de datos. En caso de perder la información producto a malfuncionamiento del sistema, rotura en el disco duro u otro inconveniente, si tenemos la información respaldada en otro lugar, podemos garantizar que la información no se pierda restaurándola con el archivo de salva deseado. Se recomiendo además que los lugares donde se realiza la salva sea en otro lugar fuera del sistema o PC que contiene las bases de datos originales para mayor seguridad. (Stack Overflow Comunity, 2016) (MSSQLTips, 2016)

Mediante las salvas y un historial de la actividad de las mismas podemos acceder a información perdida o editada de forma rápida y eficiente. Para ello también es importante crear logs de las copias de seguridad a las bases de datos.

Otro dato importante es que se recomienda realizar las salvas a solo datos modificados, para ahorrar espacio, no tener duplicada la información, etc.

1.10. Jobs en SQL Server

Son una serie de operaciones que puede hacer el servidor de SQL Server llevadas a cabo por el SQL Server Agent (Agente del Servidor SQL). Un trabajo puede correr scripts tipo Transact-SQL, comandos de la consola, etc. Para creación de un “Job” en SQL se elige el modo para ello son necesarios estos pasos. (Microsoft, 2011) (McMahon, 2012).

1- Crear “Job”.
2- Crear “Job Step”.
3- Crear “Schedule” (Agenda)
4- Adjuntar Job con Schedule.
5- Crear observador.

1.11. Sendmail en SQL Server

Algunos procedimientos generados dentro de la base de datos activan la cuenta y el perfil de correo activados dentro de SQL Server y hace uso del servidor de correos y envía notificaciones al administrador del sistema. Para hacer uso del servicio de correos mediante SQL Server es necesario, primeramente, activar opciones avanzadas, luego crear servicios de correo, crear perfil, añadir cuenta de correo, adjuntar cuenta y perfil y listo. (Microsoft, 2011).

1.12. Patrones de Seguridad

Según (Fernandez, y otros, 2001), los patrones de seguridad son elementos tomados en cuenta para el tratamiento de vulnerabilidades que puedan atentar contra la estabilidad de nuestro sistema. Tratando los patrones de seguridad como Requisitos No Funcionales podemos mejorar la protección de aplicaciones basándonos en principios, estrategias para formular una seguridad que soporte todo tipo de ataques o manipulaciones no deseadas.

La lista siguiente provee solo una descripción abreviada de cada patrón seleccionado para la creación del uso del modelo de seguridad sugerido, junto con la referencia de la fuente original del patrón de diseño. Se aclara dejar los nombres en ingles por su uso internacional. (Dangler, 2013). Algunos de los patrones son explicados para el conocer su funcionamiento.

Tabla 1-2 Listado de Patrones

No.	Nombre de Patrón de Diseño	Categoría del RNF	Nivel de Diseño
1	Authenticator	Confiabilidad	Diseño
2	Authorization	Confiabilidad	Diseño
3	Check Point	Confiabilidad	Diseño
4	Clear Sensitive Information	Confiabilidad	Implementación
5	Controlled Object Factory	Integridad	Diseño
6	Defer to Kernel	Confiabilidad	Arquitectura
7	Distrustful Decomposition	Integridad	Arquitectura
8	Full View with Errors	Disponibilidad	Diseño
9	Information Obscurity	Confiabilidad	Implementación
10	Input Validation	Integridad	Implementación
11	Limited View	Disponibilidad	Diseño
12	Multilevel Security	Confiabilidad	Arquitectura
13	Pathname Canonicalization	Integridad	Implementación
14	Privilege Separation	Integridad	Arquitectura
15	Resource Acquisition is Initialization (RAII)	Disponibilidad	Implementación
16	Role Rights Definition	Confiabilidad	Arquitectura
17	Role-Based Access Control	Confiabilidad	Arquitectura
18	Roles	Confiabilidad	Diseño
19	Secure Access layer	Integridad	Arquitectura
20	Secure Builder Factory	Integridad	Diseño
21	Secure Chain of Responsibility	Integridad	Diseño
22	Secure Channels	Confiabilidad	Implementación
23	Secure Directory	Integridad	Implementación
24	Secure Factory	Integridad	Diseño
25	Secure Logger	Responsabilidad	Implementación
26	Secure Session	Integridad	Diseño
27	Secure State Machine	Confiabilidad	Diseño
28	Secure Strategy Factory	Integridad	Diseño
29	Secure Visitor	Integridad	Diseño
30	Single Access Point	Confiabilidad	Diseño

A continuación, se explican algunos de los patrones de seguridad más importantes para su uso en la propuesta del modelo de procedimientos.

Authentificator: El patrón Authenticator verifica que el sujeto es quien dice ser. Este patrón suporta el uso de diferentes tipos de algoritmos de autentificación para diferentes usuarios. Este patrón guarda la información de autentificación por separado para incrementar la seguridad, pero el costo es complejo.

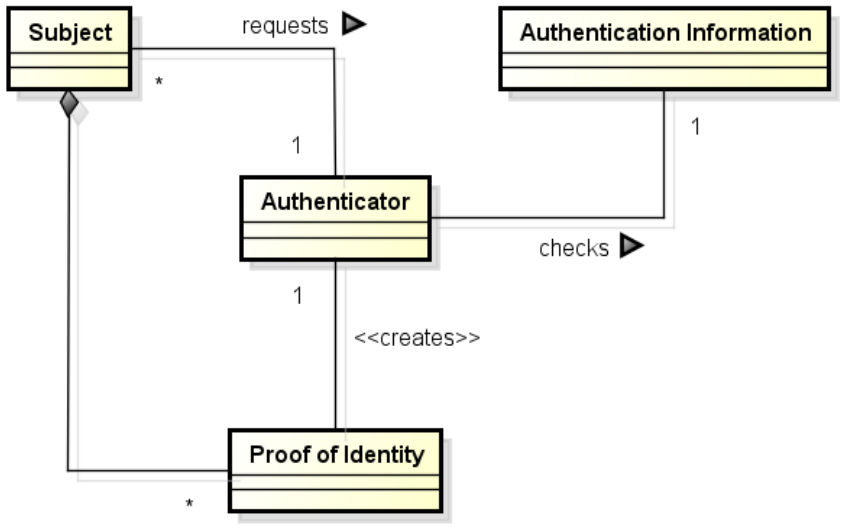

Figura 1.1 Representación gráfica del patrón Authenticator

Authorizator: Identifica los sujetos o usuarios y los recursos que pueden tener acceso de los recursos. Conjunto con el acceso se otorgan los privilegios de acceso para eses recurso. Este patrón separa permisos del sujeto y recursos.

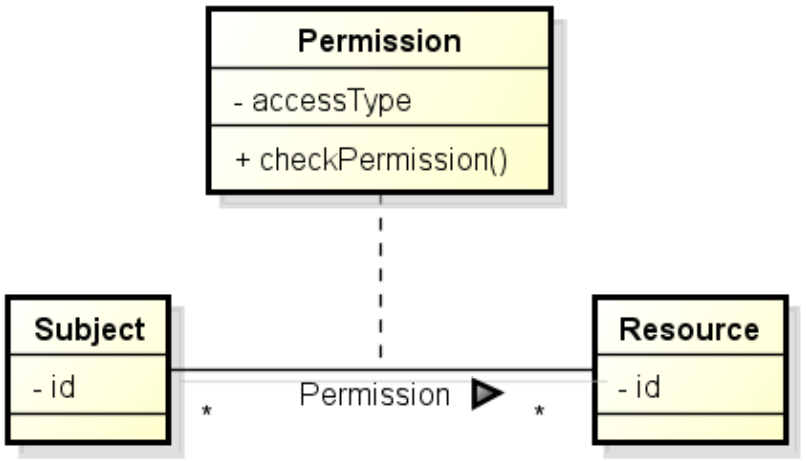

Figura 1.2 Representación gráfica del patrón Authorizator

Check Point: Este patrón previene que los usuarios tengan acceso a información clasificada y protege el sistema de actos mal intencionados. Puede ser aplicado en diferentes puntos dentro de la aplicación permitiendo medidas de seguridad variables para cada usuario o rol. Este patrón varía basado en como el usuario se logueó en el sistema, cuantos intentos de logins fallidos ocurrieron, o basado en la fecha, etc.

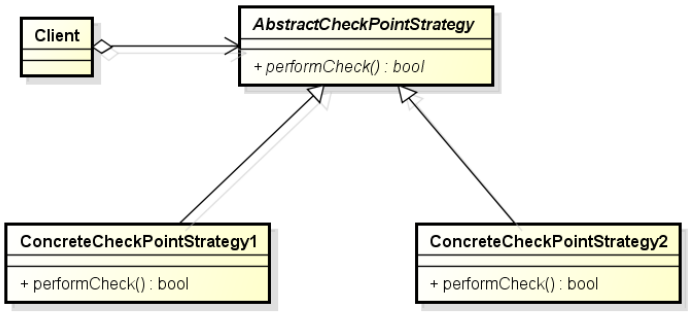

Figura 1.3 Representación gráfica del patrón Check Point

Clear Sensitive Information: Previene el acceso de información sensible de un recurso reusable que debe haber sido limpiado. Se utiliza cuando se mantiene información en la memoria, cache o discos. Cuando un recurso es liberado, puede contener información sensible para otros usuarios sin autorizar, por eso es importante limpiar los datos de este tipo generados.

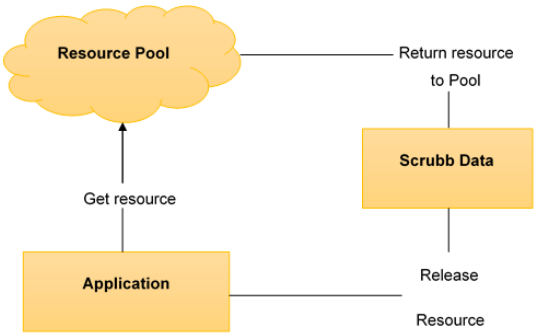

Figura 1.4 Representación gráfica del patrón Clear Sensitive Information.

Input Validation: Asegura que la información entrada por el usuario esté libre de texto malicioso. Este patrón se usa cuando la información entrada por el usuario no es confiable. Se protege así de ataques como SQL Injection y ataques “overflow”. Esto debe ser tratado para máxima seguridad por parte del servidor y por parte del cliente.

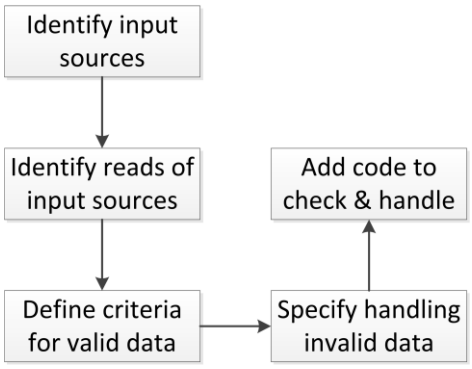

Figura 1.5 Representación gráfica del patrón Input Validation.

Limited View: Este patrón habilita solamente funciones que el usuario autentificado tiene el privilegio de acceder. Este patrón debe ser utilizado cuando la mayoría de los permisos deben ser limitadas a un número de operaciones. Además, usa la versión actual del usuario y construye una interfaz con un propósito especial para el usuario por lo que se debe configurar el material creado por el sistema para cada tipo de usuario.

Roles Based Control: Permite asociar permisos a los usuarios basado en un sistema de roles asignados por el sistema. Asociando usuarios con roles y roles con permisos eliminamos el trabajo de asociar usuarios individuales a un set de permisos. Este patrón es apropiado cuando un gran número de usuarios o un gran número de recursos comparten privilegios de accesos relacionados.

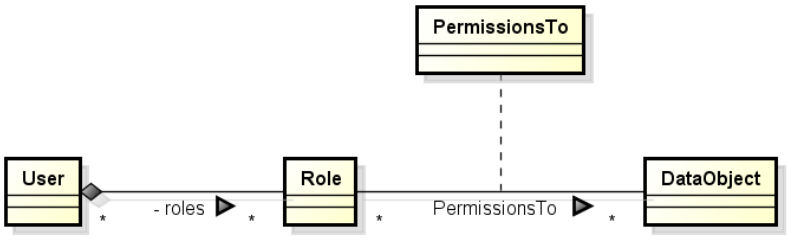

Figura 1.6 Representación gráfica del patrón Roles Based Control.

Pathname Cannonization: Este patrón asegura que los caminos a diferentes archivos y directorios sea válidos de enlaces libres a otras aplicaciones. Protegiéndose así de vulnerabilidades de directorios asegurando que el recurso ofrecido no es ningún enlace. Como siempre dependemos de ficheros y carpetas. Las direcciones físicas vienen de esta manera que conocemos: "`C:\WebSites\MyWebSite default.cshtml datafile.txt \images Logo.jpg \styles Styles.css`".

Las URL's tienen los siguientes detalles:

- Comienza con el nombre de dominio: "*http://www.example.com*", con el nombre de un servidor "*http://localhost*" o "*http://myserver*".
- Una URL corresponde a un camino físico de una computadora host: "*http://myserver*" corresponde a la carpeta "*C:\websites\mywebsite*" en el servidor.
- Una dirección virtual es utilizada primeramente para representar caminos en código sin tener que escribir el camino completo. Incluye la porción de la URL que sigue al nombre de dominio o servidor. Cuando se utilizan caminos virtuales, puedes mover el código a un dominio o servidor diferente sin tener que actualizar las direcciones.

Tabla 1-3 Representación gráfica del patrón Pathname Cannonization

URL Completa	`http://mycompanyserver/humanresources/CompanyPolicy/`
Nombre del server	*mycompanyserver*
Dirección Virtual	*/humanresources/CompanyPolicy/*
Dirección Físico	*C:\mywebsites\humanresources\CompanyPolicy.cshtm*

Secure Logger: Protege a los logs de sistema de acceso de potenciales atacantes y previene la alteración de los logs para esconder la actividad.

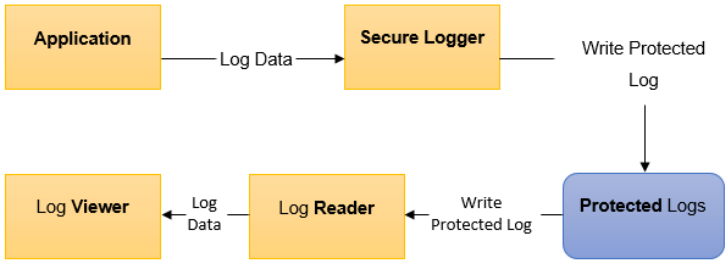

Figura 1.7 Representación gráfica del patrón Secure Logger.

1.13. CAPTCHA

Por sus siglas en ingles **CAPTCHA** (Completely Automated Public Turing test to tell Computers and Humans Apart) son las pruebas humanas interactivas más utilizadas, aparecen como un método de diferenciación entre usuarios humanos y máquinas para brindar seguridad a la información en internet y evitar el *spam*, especialmente. (e-Ciencias de la Información, 2014)

Tipos de CAPTCHA: OCR (Optical Character Recognition): son los más conocidos y utilizados en la actualidad, se presenta la imagen en una palabra en distorsión de diversos efectos, la cual debe ser escrita por los usuarios, y debido a los efectos pictóricos, no podrá ser reconocida por el equipo.

Existe otra extensión de CAPTCHA llamada reCAPTCHA adquirida por Google y protege a más de 100 000 sitios web de spam. Su lema es “Stop Spam, Read Books” y significa parar al spam y leer libros. A partir del 2012 se incluyen imágenes obtenidas por Google Street View como parte del proceso de reconocimiento para los usuarios. (Wikipedia, 2018).

Los CAPTCHAS son usados como doble factor de autentificación, existen otros como verificación vía correo y sms,

1.14. Expresiones regulares

También llamadas Regex o patrones, las expresiones regulares son una cadena de caracteres que define un conjunto de cadenas de caracteres sin enumerar sus elementos. Dicho de otra forma, es un patrón que genera un conjunto de palabras de un alfabeto (no necesariamente se compone de letras). (Alfonso, 2010)

Ejemplo:
Si queremos enviar un correo, cómo sabemos que los datos introducidos son válidos como dirección de correo. Para evaluar o restringir la entrada de datos de forma eficiente aplicamos una expresión regular con la siguiente estructura: texto1@texto2.texto3 donde:

texto1: es una cadena que admite caracteres [A-Z][a-z][0-9][.]

@**:** uso obligatorio posterior del texto1

texto2: coincide con el nombre de dominio y vuelve a coincidir con [A-Z][a-z][0-9][.]

.(el punto) : obligatorio.

texto3: [A-Z][a-z][0-9] y acotamos el tamaño a 4 máximo y mínimo 2.

Tabla 1-4 Ejemplo de Expresión Regular

English Rule	Regex Pattern
Buscar hasta encontrar la primera ocurrencia del símbolo $.	`.*\$`
Ahora necesitamos más números.	`\d+`
Puede que exista una coma.	`,?`
Y muchos números más.	`d+`

Ejemplo: Moneda Euro: este patrón busca primero el símbolo de euro, luego dos dígitos, un punto decimal y dos dígitos más. [\u20AC]\d\d\.[0-9][0-9]

Uso: (Booth, 2014)

- **Motor de búsqueda:** buscamos una palabra en un texto (típico Ctrl+F). Ejemplo: buscamos la palabra “byte” en la cadena “mil veinticuatro bytes son un kilobyte”. Esto nos daría dos coincidencias, la primera sería en “byte”, la segunda en “kilobyte”.
- **Lenguaje:** utilizamos los caracteres como meta-caracteres. Es decir, no por lo que son, sino por lo que representan. Por ejemplo, queremos encontrar todos los números de una cadena de texto. Si buscamos en “v1d4s_c0ncurr3nt3s”, iría encontrando todos los números: 1, 4, 0, 3, 3.

1.15. Sendmail en la aplicación Web

Según la página de ayuda de Microsoft (MSDN, 2018) también se necesita que la propia aplicación sea autónoma en el uso de servicios de correo para notificaciones a los administradores, usuarios de distintos roles de acciones que son ejecutadas por el sistema o por parte de los usuarios que deben ser controladas.

1.16. Conclusiones del Capítulo

En este capítulo hemos podido conocer las bases que sustentan esta investigación, estadísticamente la situación actual de la seguridad de los trabajos de tesis de nuestra universidad, así como también las bases teóricas y el funcionamiento de los elementos propuestos para erradicar las vulnerabilidades que atentan al desarrollo de las aplicaciones Web de gestión. Habiendo explicado las diferentes zonas y puntos de seguridad se hace énfasis en que es totalmente genérico el uso de estas tecnologías para cualquier plataforma y lenguaje de programación.

CAPITULO 2: Resultados de la investigación.

2.1. Introducción

Este capítulo trata de los resultados del estudio de la aplicación de la seguridad a los trabajos de diploma mencionados en el Capítulo 1. Se pueden apreciar los datos estadísticos en gráficas y tablas, así como los puntos evaluados para llevar a cabo la investigación. Además de la propuesta de un grupo de procedimientos seleccionados y agrupados para usarlos en el desarrollo de aplicaciones Web de gestión en entornos de intranet y elevar el cumplimiento de requisitos de seguridad para el despliegue de las mismas.

2.2. Análisis de la Investigación a trabajos de diplomas seleccionados anteriormente

Del estudio anterior a las 142 tesis evaluadas se logran extraer datos que exponen el comportamiento de la seguridad dando puntos críticos en cuanto al tratamiento de la misma. A continuación, se muestran tablas generales, específicas de los resultados del estudio de las tesis revisadas entre los años 2009 hasta el 2017 con los puntos de seguridad evaluados divididos por área de bases de datos y dentro de la aplicación con un total de 25 aspectos más utilizados en las diferentes áreas de la ciberseguridad en el desarrollo de aplicaciones web de gestión, bases de datos y despliegue. Durante la revisión de los proyectos, se hace notar la utilización de diferentes tecnologías para manejar la información de gestión en las bases de datos como Apache, SQL Server, MySQL, PostgreSQL, entre otros; así como lenguajes y ORM (Objet-Relational Mapping) que es modelo de objeto relacional en la programación de sus vistas como ASP.NET MVC y Entity Framework, php, Doctrine, Symphony, Phyton, HTML5, CSS3, Javascript, JQuery, Bootstrap siendo ASP.NET MVC y el Framework de desarrollo Symphony los más populares.

Tabla de Panorama General: A continuación, se demuestran 25 puntos evaluados y su comportamiento en los trabajos de tesis desde los años 2009 hasta el 2017, obteniendo los datos de la cantidad de aspectos evaluado por año en las áreas de bases de datos y la aplicación como estrategias específicas.

Tabla 2-1 Estadística General. Propio del autor.

TOTAL = 142		**Tesis evaluadas por año**							
		15	39	61	8	12	4	3	
		2017	**2016**	**2015**	**2014**	**2012**	**2010**	**2009**	**TOTAL**
BD	**Autorización/Contraseñas**	-	-	1	-	-	-	-	**1**
	Trigger DML	-	1	-	-	-	-	-	**1**
	Triger DDL	-	-	-	-	-	-	-	**0**
	Triger Logon	-	-	-	-	-	-	-	**0**
	Logs	-	-	-	-	-	-	-	**0**
	Scripts	-	-	-	-	-	-	-	**0**
	Correo	-	-	-	-	-	-	-	**0**
	Backups	-	2	1	-	-	-	-	**3**
PATRONES	**Authenticator & Authorization**	10	15	56	8	11	-	3	**103**
	Input Validation	2	-	38	2	5	1	3	**51**
	Logs Database View	-	-	-	-	-	-	-	**-**
	Logs Logon View	-	-	-	-	-	-	-	**-**
	Logs System View	7	7	5	-	-	-	1	**20**
	Password Security	2	1	3	-	-	-	-	**6**
	Pathname Canonicalization	-	-	-	-	-	-	-	**-**
	Role-Based Access Control	10	15	56	8	6	-	3	**98**
	Https	-	-	-	-	-	-	-	**-**
	Despliegue Físico	1	3	-	-	-	-	-	**4**
	Cookies	2	-	-	-	-	-	-	**2**
	Cache	-	-	-	-	-	-	-	**-**
	Otros patrones reforzar la programación	1	9	4	-	2	-	-	**16**
	Desactivación de usuarios en caliente	-	-	-	-	-	-	-	**-**
	CAPTCHA	-	-	-	-	-	-	-	**-**
	Regex	-	-	-	-	-	-	-	**-**
	Pruebas con Softwarres (VEGA)	1	4	1	-	-	-	-	**6**

Tabla de % de tratamiento por puntos válidos: Se seleccionan 25 aspectos a evaluar y se determinan los resultados en porcientos generales del uso por aspecto de entre todos los trabajos revisados. Solamente los patrones Roles Based Control, Authentication y Authorization superaron el 50%. En las bases de datos el índice de uso de seguridad es muy bajo, así como en logs, servicios de correos y otros.

Tabla 2-2 Datos en % de tratamiento de puntos válidos. Propio del autor.

		Cant	%
BD	Autorización/Contraseñas	1	1%
	Trigger DML	1	1%
	Triger DDL	0	0%
	Triger Logon	0	0%
	Logs	0	0%
	Scripts	0	0%
	Correo	0	0%
	Backups	3	2%
PATRONES	Authenticator & Authorization	103	73%
	Input Validation	2	1%
	Logs Database View	0	0%
	Logs Logon View	0	0%
	Logs System View	2	1%
	Password Security	1	1%
	Pathname Canonicalization	0	0%
	Role-Based Access Control	98	69%
	Https	0	0%
	Despliegue Físico	4	3%
	Cookies	0	0%
	Cache	0	0%
	Otros patrones reforzar la programación	0	0%
	Desactivación de usuarios en caliente	0	0%
	Regex	0	0%
	CAPTCHA	0	0%
	Pruebas con Softwares (VEGA)	4	3%

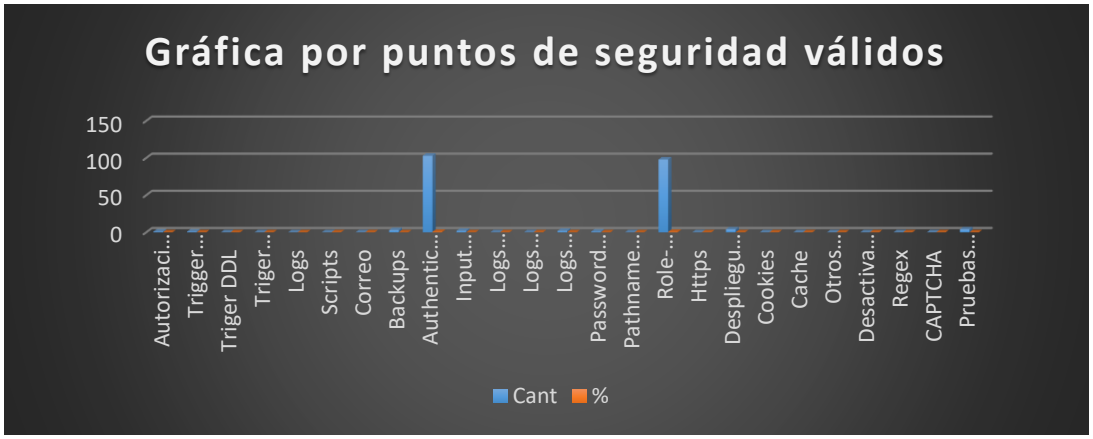

Figura 2.1 Datos en % de tratamiento de puntos válidos. Propio del autor.

Tabla: % de seguridad aplicada a los trabajos de tesis: En la gráfica se muestra el porciento de la seguridad en cuanto a los puntos revisados por cada año. Como se aprecian los resultados no llegan ni al 50% del uso general de los aspectos evaluados a pesar de haberse elevado desde 2009 hasta la actualidad.

Tabla 2-3 % de puntos revisados por año. Propio del autor.

	2009	2010	2012	2014	2015	2016	2017
Objetivos chequeados	4	1	4	3	9	9	9
Porciento efectivo	16%	4%	16%	12%	36%	36%	36%

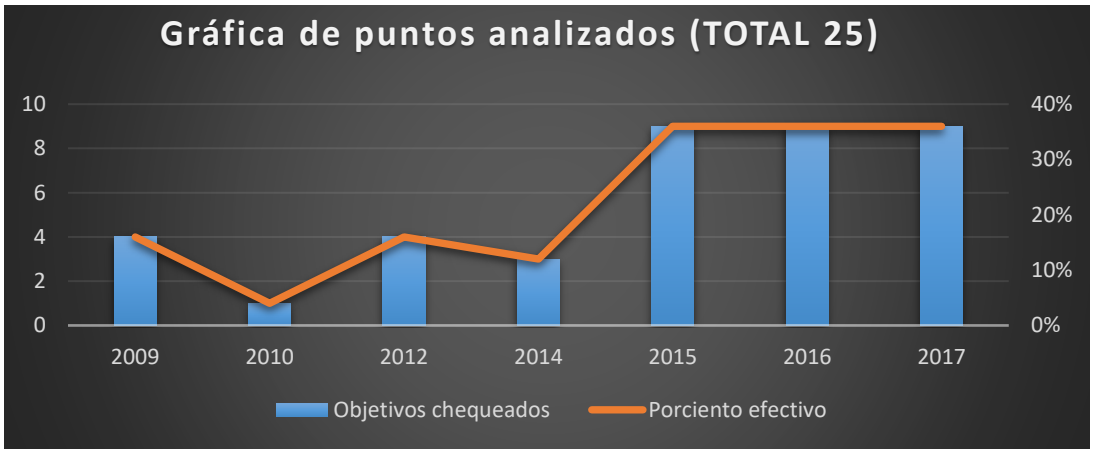

Figura 2.2 Gráfica % de tratamiento seguridad a los trabajos por año. Propio del autor.

2.2. Resultados de la investigación de los trabajos de tesis

Como se aprecia, el tratamiento de la seguridad en cualquiera de las gráficas no alcanza ni al 40% de efectividad. Por lo que da validez a la investigación y propuesta para la aplicación de los procedimientos. Algunos aspectos generales se mencionan posteriormente:

- La palabra seguridad se nombra en muchas tesis, pero nunca es tratada.
- El tratamiento de seguridad en historias de usuarios es escaso, válido solo en algunas ocasiones para Autentificación y Autorización.
- Se utilizan bases de datos, pero se limitan en la gran mayoría, a la gestión de la información mediante la aplicación por ORM.
- Los reportes generados no cumplen con casi ninguno de los parámetros de la seguridad evaluados.
- No se demuestra ningún código de implementación de la seguridad.
- La seguridad no se evalúa como riesgo alto en el desarrollo de las aplicaciones.

- Como punto de interés, se utiliza en un gran porciento la seguridad con el trabajo de autentificación, autorización y trabajo con roles.
- Se utiliza el software VEGA para realizarles testeos de seguridad, pero solo son evaluados los puntos específicos del mismo.
- No hay mención de las trazas de la aplicación donde se guardan ni cómo se verifican.

2.2. Procedimientos Propuestos

Para tener un mejor entendimiento y control de los procedimientos que se proponen, estos se dividen en grupos, donde a cada procedimiento le es asignado un número dentro de los grupos. A continuación, se muestra una tabla con los grupos de procedimientos seleccionados y aplicados en la aplicación web. Para identificar un procedimiento se hacen distinguir por “PROC- # - #”, donde el primer número es el número el grupo, y el segundo es el número del procedimiento.

Tabla 2-4 Areas separadas para la seguridad. Propio del autor.

GRUPO	No.	NOMBRE DEL PROCEDIMIENTO
1	01	HTTP y HTTPS.
	02	Despliegue Físico N-Tier.
2	01	Triggers.
	02	Sendmail.
	04	Backups.
3	01	Authenticator
	02	Authentificator
	03	Input Validation.
	04	Role Based Control.
	05	Limited View.
	06	Pathname Cannonization.
4	01	System’s Logs
	02	Busines’ Logs
	03	Logon’s Logs
	04	TXT’s Logs
5	01	Controller Monitoring
	02	CAPTCHAs.
	03	Regular Expressions.

Modelo de identificación de procedimiento:

A continuación, se muestra el modelo propuesto para la agrupación de los procedimientos para ser utilizados en las aplicaciones web de gestión.

PROC - # - #
Nombre del Procedimiento
Descripción: describe el área de seguridad que cubre cuando es aplicado el procedimiento y cuáles son sus funciones.
Riesgo: explica cuáles son las vulnerabilidades si no se aplica el procedimiento.

2.1. PROCEDIMIENTOS GRUPO No. 1: Despliegue Físico y HTTPS

2.1.1. Despliegue Físico: HTTPS

PROC - 1 - 01
Despliegue Físico: HTTPS
Uso de HTTPS: se propone como procedimiento el uso de un certificado de cifrado de datos SSL para el servidor web montado para la encriptación de la información.
Riesgo: la no aplicación de este procedimiento produce que la información viaje de forma no cifrada ocasionando altos riesgos de que sea manipulada indebidamente.

2.1.2. Despliegue Físico: Niveles físicos de 2 capas (2-Tier)

PROC - 1 - 02
Despliegue Físico: Utilización de los niveles físicos de despliegue 2-Tier
Aplicación Cliente-Servidor Se quiere desarrollar una aplicación cliente-servidor que acceda directamente a un servidor de bases de datos. Este escenario es muy diferente, pues todas las capas lógicas estarían situadas en un nivel cliente que en este caso sería el PC cliente. Esta arquitectura es útil cuando se requiere un rendimiento muy alto y accesos rápidos a la base de datos.
Riesgo: el desacoplamiento de la información es vital para su protección, se elige el 2-Tier para la velocidad en el acceso debido al entorno de despliegue.

2.2. PROCEDIMIENTOS GRUPO No. 2: Bases de Datos

2.2.1. Bases de Datos: Triggers

PROC - 2 - 01

Bases de Datos: Triggers

De forma automática disparan acciones dentro de las bases de datos en dependencia de la interacción que tenga el usuario y la intención con que se crea. Cada trigger debe ir situado en la base de datos que le corresponda.

Nombre Trigger	Tipo	Descripción
trgAuditUpdateSalaryField	DML	Actualiza dinámicamente el campo Salary.
trgDDLforCreateDatabase	DDL	Evita que se creen bases de datos dentro del servidor.
trgDDLforDropDatabase	DDL	Evita que se eliminen bases de datos dentro del servidor.
trgDDLforCreateTable	DDL	Previene que se creen tablas en el servidor.
trgDDLforDropTable	DDL	Previene que se eliminen tablas en el servidor.
trgLogon	Logon	Registra los accesos externos al servidor.
trgPayrollAuditDelete	DML	Guarda en la tabla de auditoría los registros borrados.
trgPayrollAuditInsert	DML	Guarda en la tabla de auditoría los registros nuevos insertados.
trgPayrollAuditUpdate	DML	Guarda en la tabla de auditoría los registros antes y después de modificarlos.
trgPayrollDBLogUpdate	DML	Previene que se actualice la tabla.

Riesgo: de no utilizarlos no existiría nada de las funciones que realizan los triggers.

2.2.2. Bases de Datos: Servicio Sendmail

PROC - 2 - 02

Bases de Datos: Servicio Sendmail

Utilizar los servicios de correo dentro de SQL Server nos permite de manera automática realizar notificaciones de alerta a administradores, etc. Debemos utilizar Transact-SQL o Microsoft SQL Server Management para habilitar y configurar el servicio de correos.

Riesgo: si no tenemos configurado el correo dentro de SQL Server no podremos tener alertas de acceso o uso indebido dentro del servidor de Bases de Datos.

2.2.3. Bases de Datos: Backups

PROC - 2 - 03
Bases de Datos: Backups
Estos scripts permiten que las informaciones de las bases de datos sean duplicadas para prevenir pérdidas o manipulaciones indebidas. Ya sean de forma automática o manual, se recomienda que las salvas de las bases de datos deben quedar almacenadas en otra localización con medidas de seguridad diferentes.

Scripts en SQL	Descripción
sp_Backup_DB_Audit	Crea stored procedures para la creación y ejecución de los backups manuales por parte del usuario.
sp_Backup_DB_Identity	
sp_Backup_DB_Payroll	
sp_Backup_DB_Audit_AUTO	Crea stored procedures para creación y ejecución backups automáticos que que se activan mediante el Server Agent.
sp_Backup_DB_Identity_AUTO	
sp_Backup_DB_Payroll_AUTO	

Riesgo: para mantener un respaldo de la información de las bases de datos es recomendable esta opción. De lo contrario si ocurre una manipulación, pérdida o restauración no sería posible.

2.3. PROCEDIMIENTOS GRUPO No. 3: Patrones de Seguridad

2.3.1. Patrones de Seguridad: Authenticator

PROC - 3 - 01
Patrones de Seguridad: Authenticator
Pide una credencial para el usuario antes de entrar al sistema y utilizar los recursos. Dentro del controlador *AccountController* en el método de acción *Login*.

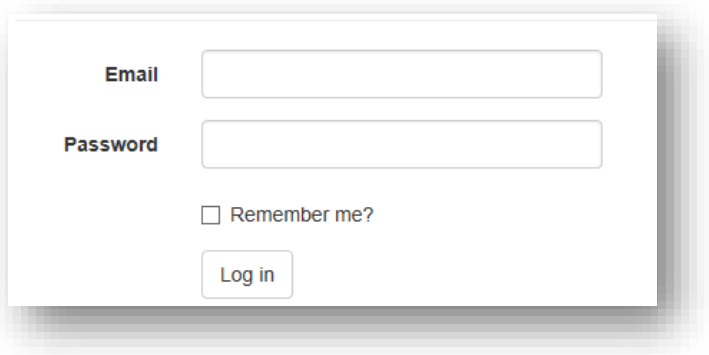

Figura 2.8 Formulario de entrada de datos. Propio del autor.

Riesgo: Si no disponemos de una función para acceder al sistema corremos el riesgo que cualquier usuario pueda manipular o extraer la información del modelo del negocio.

2.3.2. Patrones de Seguridad: Authorizator

PROC - 3 - 02
Patrones de Seguridad: Authorizator
El usuario al encontrarse dentro del sistema, se sabe quién es y el nivel de autorización que tiene, lo cual permite acceder solamente a las áreas que les son asignadas.
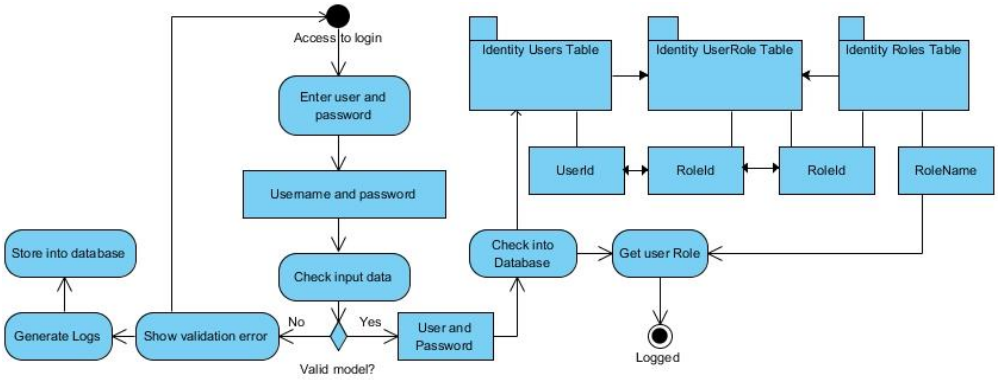
Figura 2.9 Patrón Authorization. Propio del autor.
Riesgo: Si el sistema de seguridad no tiene como definición quién accede a qué recurso cualquier nivel de acceso puede hacer uso a cualquier recurso del modelo del negocio.

2.3.3. Patrones de Seguridad: Input Validation

PROC - 3 - 03
Patrones de Seguridad: Input Validation
Es importante prevenir que los tipos de datos entrados por los usuarios sean correctos. Se aplican para ello atributos especiales en los campos de las clases del dominio.
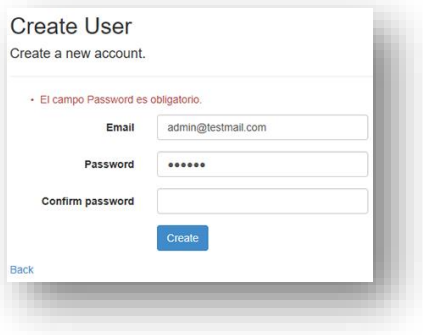
Figura 2.11 Formulario crear nuevo usuario. Propio del autor.
Ejemplo: Si en el campo email se introduce algo distinto al patrón x@y.z conocido como “email” produce un error y lo notifica al usuario. Esta es una forma de utilización de expresiones regulares.
Riesgo: de no utilizarse corremos el riesgo que la información que sea almacenada de manera incorrecta.

2.3.4. Patrones de Seguridad: Roles Based Control

PROC - 3- 04
Patrones de Seguridad: Roles Based Control
Se crea un sistema basado en roles, “Administrator” que tiene acceso total a todo el sistema, “Moderator”, que permite crear registros de la tabla Payroll, así como editarlos o eliminarlos y por último el rol “User” que solamente puede ver su registro de pago.

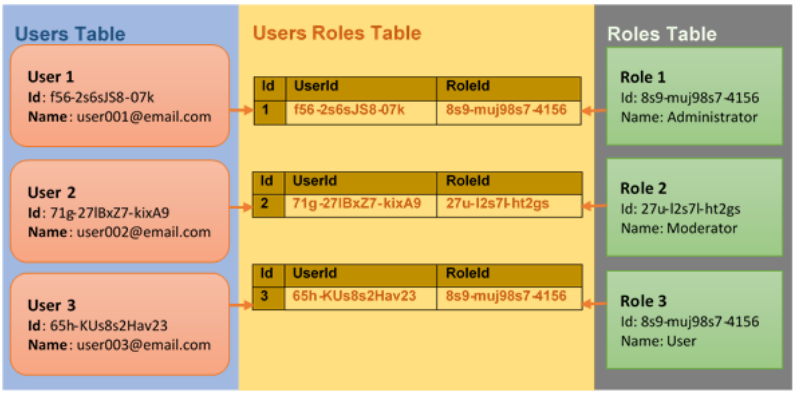

Figura 2.10 Patrón Roles Based Control. Propio del autor.

Riesgo: de no hacer uso de este patrón no existirá un nivel jerárquico de acceso a los recursos por parte de los usuarios.

2.3.5. Patrones de Seguridad: Limited View

PROC - 3- 05
Patrones de Seguridad: Limited View
Las vistas limitadas son los recursos (en este caso las paginas dentro de la aplicación) a los que el usuario tiene acceso. Mediante Identity podremos garantizar diferentes niveles de acceso por roles o por usuarios, o los dos combinados. En nuestro dominio. Este patrón viene vinculado con el patrón Role Based Control.

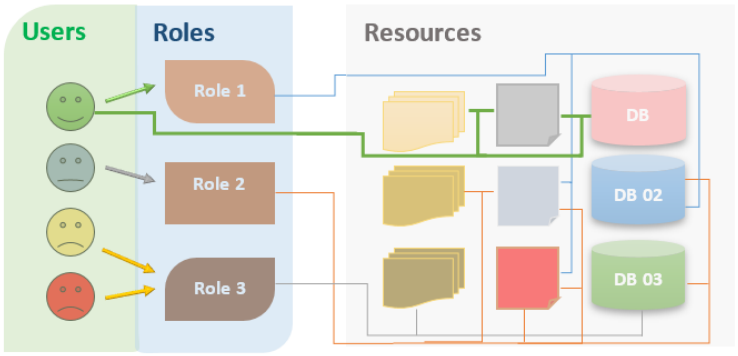

Figura 2.24 Representación de acceso a recursos basados en usuarios y roles. . Propio del autor.

Riesgo: los recursos deben ser agrupados en niveles de seguridad para protegerlos de acceso indebido.

2.3.6. Patrones de Seguridad: Pathname Cannonization

PROC - 3- 06
Patrones de Seguridad: Pathname Cannonization
Ocultar la visibilidad de los recursos a los usuarios es otra de las medidas de seguridad que garantiza que la información mostrada se mantenga oculta. **Ejemplo:** • http://localhost/
Riesgo: riesgo de mostrar las direcciones físicas de los recursos.

2.4. PROCEDIMIENTOS GRUPO No. 4: Logs

2.3.1. Logs: Logs del Sistema

PROC - 4 - 01
Patrones de Seguridad: Secure Logger (System's Log)
Son los logs, o trazas generadas las acciones del sistema, que son almacenadas las para un posterior monitoreo. Independientemente de quién o qué genere los errores o trazas, se crean vistas para su auditoria por parte de los administradores del sistema. **Algunos ejemplos de trazas del sistema:** • Información de acceso séase Login o Logout. • Creación de usuarios. • Envío de correos. • Acceso a recursos. • Etc.
Riesgo: no se puede auditar las acciones provocadas por los usuarios sin un control de las trazas generadas por las acciones de los usuarios.

2.3.2. Logs: Logs del modelo del negocio

PROC - 4 - 02
Patrones de Seguridad: Secure Logger (Business Model's Log)
Todas las acciones generadas por parte del modelo del negocio son registradas. Asimismo, salva automática en otro servidor de bases de datos, los registros insertados, modificados y eliminados.
Riesgo: no se puede tener un control acciones del área del modelo del negocio.

2.3.3. Logs: Logs de acceso al servidor de base de datos

PROC - 4 - 03
Patrones de Seguridad: Secure Logger (Logon's Log)
Muestran los registros de accesos de logins independientes por parte de SQL Server con la información del evento. Se genera a partir del trigger *trgLogon* que está activado en todo el servidor.
Riesgo: no es posible ver que usuario accedió por independiente al servidor de bases de datos.

2.3.4. Logs: Logs de Salvas de las Bases de Datos

PROC - 4 - 054
Patrones de Seguridad: Secure Logger (Backup's Log)
Podemos mendiante .
Riesgo: no es posible ver que usuario accedió por independiente al servidor de bases de datos.

2.3.5. Logs: Log en un sistema de archivos externos txt

PROC - 4 - 05
Patrones de Seguridad: Secure Logger (Txt's Log)
Son los logs creados en ficheros txt, que almacenan trazas configurables para posteriormente hacer un estudio en minería de datos para auditorías. Estos archivos vienen con tamaño limitado y niveles de errores que diferencian el tipo de trazas.
Riesgo: no se pueden realizar posteriores auditorias

2.4. PROCEDIMIENTOS GRUPO No. 5: Externos

2.4.1. Procedimientos externos: Desactivación en caliente

PROC - 5 - 01
Procedimientos Externos: Hot Activation/Deactivation
Sucede cuando un usuario es logueado en el sistema y se encuentra utilizando un recurso cualquiera, y entonces que el administrador lo desactiva, pero al estar en el sistema el usuario sigue conectado producto a cookies, temporales, etc, se hace entonces la comprobación en cada Método de Acción ("ActionResult") del Controlador verificando si el usuario está activo o no.
Riesgo: de no aplicarse este método especial los usuarios aunque sean desactivados siguen utilizando los recursos en caso de estar activos en el momento de la desactivación.

2.4.2. Procedimientos externos: Doble factor de autentificación CAPTCHA

PROC - 5 - 02
Procedimientos Externos: CAPTCHAs
Mediante la clase auxiliar reCAPTCHA.cs generar otro factor de identificación. Este Captcha genera un código de 5 dígitos combinados entre números desde el 0-9 y todas las letras del abecedario en ingles de forma aleatoria. En el caso de las letras elige también si son mayúsculas o minúsculas de forma aleatoria también, luego teniendo el código almacenado en una variable de tipo texto, genera una imagen Bitmap con dicho código. Véase Anexo 2.

Figura 2.27 Ejemplo de códigos CAPTCHA generados

Riesgo: al no tener un doble factor de autentificación no podemos garantizar que no sea un bot quien accede al sistema. Este método aumenta la seguridad de acceso al sistema.

2.4.3. Procedimientos externos: Expresiones Regulares

PROC - 5 - 03
Procedimientos Externos: Expresiones regulares
La aplicación de aplicaciones regulares en algunos de las áreas de la aplicación. En este caso en varios modelos de clase dentro de la aplicación donde se necesite el campo email, este patrón es aplicado.
Riesgo: El uso varia de lo que se requiera obtener, puede ser usado como seguridad como en input validation u otro parámetro de seguridad.

2.4.4. Procedimientos externos: Servicio Sendmail en la aplicación web

PROC - 5 - 04
Procedimientos Externos: Sendmail en la aplicación
Para este caso se crea una clase auxiliar que permita utilizar los servicios de mensajería de correo para notificar al Administrador de acciones mal intencionadas o no producto de la interacción de los usuarios y el sistema, también si los usuarios necesitan ser notificados por el servidor o por el propio Administrador.
Riesgo: Los administradores no son notificados de las acciones realizadas por los usuarios, y los mismos no tienen notificaciones de informaciones enviadas por el servidor o por el Administrador.

2.5. Conclusiones del Capítulo

En este capítulo se exponen los resultados del análisis de las tesis evaluadas mediante graficas de estadísticas obteniendo que la seguridad es pobremente aplicada en los trabajos de tesis presentados. Además, se explica el uso de un sistema de procedimientos organizados y agrupados para la propuesta de un modelo con diversos procedimientos de seguridad divididos en áreas para implementarlos dentro de cualquier aplicación web de gestión sin tener en cuenta la tecnología o marco de trabajo en el que se desarrolle para mejorar la seguridad en diferentes áreas dentro del desarrollo de la aplicación.

CAPITULO 3: Validación utilizando Caso de estudio SMS y Herramienta de validación Vega.

3.1. Introducción

Introducimos en este capítulo la aplicación de versión DEMO desarrollada en plataforma ASP.NET MVC 5 utilizando Microsoft Visual Studio 2015. Tiene como nombre “SMS”, por sus siglas Security Model Sample (Ejemplo de Modelo de Seguridad). La aplicación fue creada para demostrar la aplicación de la seguridad utilizando diversos patrones mencionados en el Capítulos anteriores en conjunto con estrategias que se descomponen de clases auxiliares que ayudan al funcionamiento y demostración de la aplicación.

3.2. Modelo del Negocio

El modelo del negocio se basa en gestionar nóminas de pago para usuarios registrados y activados en la base de datos. Los usuarios con roles de “*Administrator*” y “*Moderator*” tienen acceso a crear y modificar los registros de nómina. Los usuarios con roles “*User*” solo pueden acceder a los registros que pertenezcan a él. El usuario “*admin@testmail.com*” tiene acceso a todo tipo de acciones de control dentro del sistema como ver logs, crear o borrar usuarios, así como activarlos o desactivarlos, etc. También crear, editar nóminas de pagos y ver todas las nóminas registradas por los moderadores.

3.3. Herramientas y lenguajes utilizados

Visual Studio 2015 Ultimate: Desarrollado por Microsoft, es un entorno de desarrollo (IDE) en el cual pueden ser desarrollados diversos tipos de proyectos como Web, Consola, Servicios, Escritorio, etc., en distintos lenguajes de programación tales como C#, C++, Javascript, Python, etc.

Lenguajes de Programación:

- Se utilizó el C# que es el lenguaje desarrollado por Microsoft, este ha ganado terreno durante años y consta de una gran librería de clases que pueden ser integradas a cualquier tipo de proyecto en plataforma .NET. **(Michaelis, y otros, 2016)**

- Para el trabajo con consultas a las bases de datos se utilizó Transact-SQL, Lenguaje integrado de Microsoft SQL Server por la cual se crearon los queries o consultas para los triggers, procedures, Jobs, email SQL Services, etc.

Microsoft Identity: Es una librería de clases e interfaces desacopladas del modelo del negocio que permiten mediante Entity Framework la gestión de usuarios, roles, protección de accesos a áreas protegidas, etc. También es configurable en todos sus aspectos. (Chanda, 2012)

log4Net: Es una clase integrada que ayuda al programador a generar distintos tipos de logs, dependiendo del tipo de salida (consola, ficheros txt, etc.), para llevar un control de las trazas (hora, nombre de la clase, debug, etc.) de todo tipo de eventos generados por la aplicación. Es un paquete muy configurable para el desarrollador con múltiples opciones para la salida de los datos. (Apache Community, 2018)

SQL Server 2017: La última versión del SQL Server para el procesamiento de la información a las bases de datos, así como servicios como Server Agent y Management para la configuración de SQL Server Mail Services entre otros.

SQL Server Management Studio 2017: Herramientas para las conexiones y servicios dentro del servidor SQL Server 2017. Mediante este software se crea dentro de la instancia de SQL Server 2017 el servicio de Server Agent Database Mail, Automatic Backups entre otros.

Clases auxiliares utilizadas en C#: Muchas clases de Microsoft Identity fueron modificadas para lograr objetivos específicos dentro de la gestión y control de usuarios incorporándoles métodos, campos, propiedades etc. Otras clases fueron creadas para manejar correos, logs, doble factor de autentificación, etc. A continuación, se muestran algunas de las clases utilizadas por parte de ASP.NET y propias del autor.

System.Web.Helpers.Webmail: clase integradora que permite hacer uso de servicios de correos dentro de la aplicación. Para su uso debe tener el servidor SMTP configurado para la gestión de correos.

System.Drawing y System.Drawing.Drawing2D: mediante esta clase es posible crear las imágenes del factor de autentificación CAPTCHA. Se crea un texto y mediante esta clase se impregna en un archivo tipo Bitmap para luego ser renderizado en la web como una imagen.

3.4. Modelo de clases auxiliares empleadas

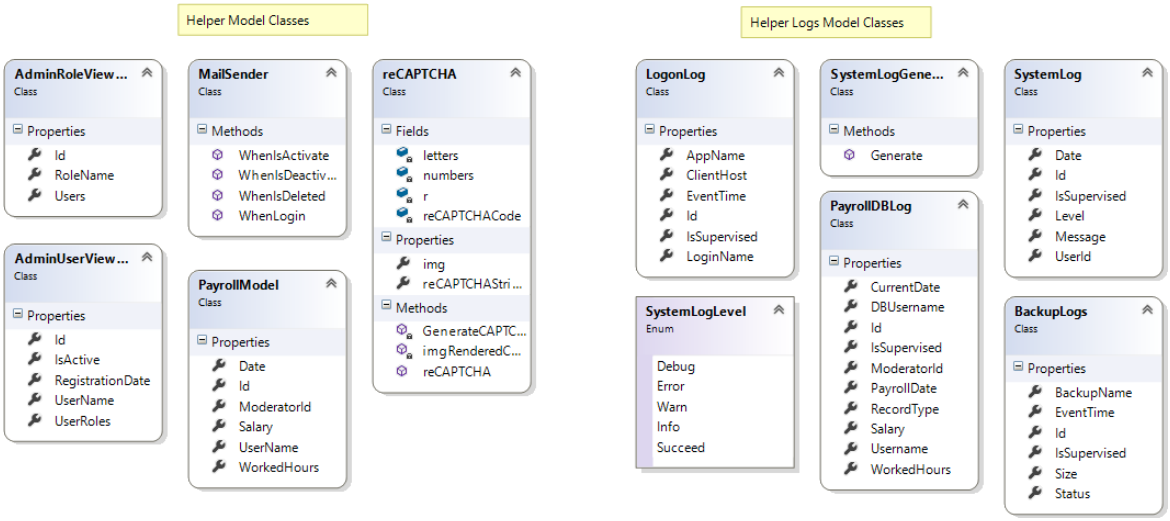

Figura 3.1 Diagrama de Clases Auxiliares. Propio del autor.

Modelo de clases controladoras empleadas:

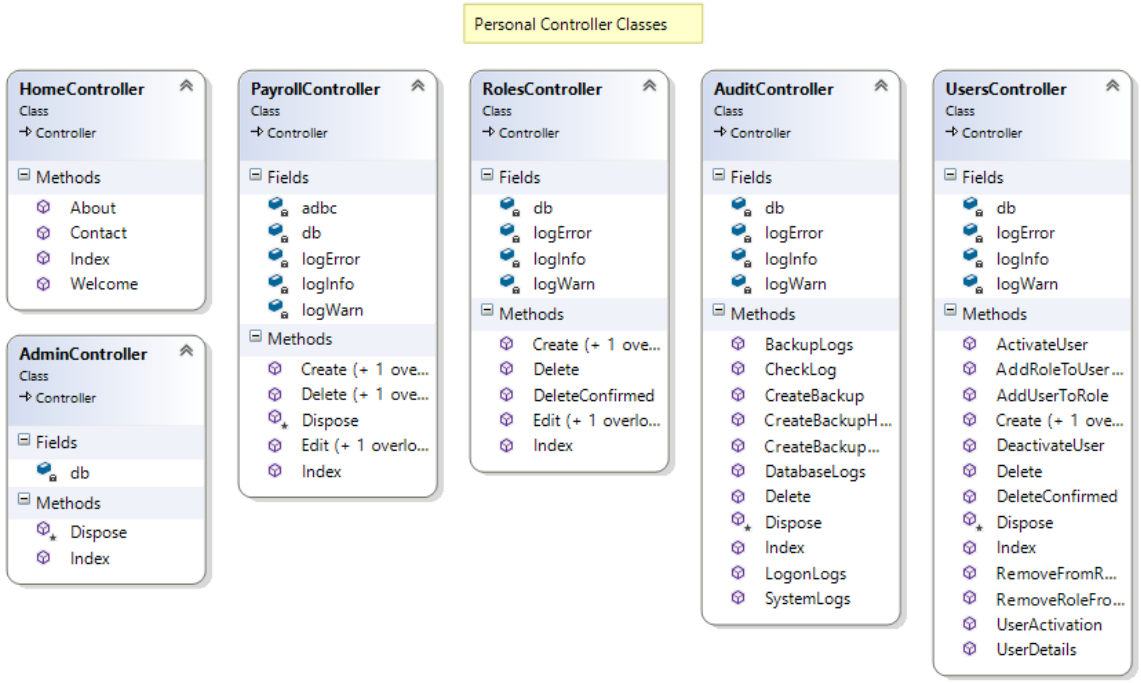

Figura 3.2 Diagrama de Clases de los Controladores. Propio del autor.

3.5. Uso de los Procedimientos del Grupo No. 1: Despliegue Físico

3.5.1. PROC-1-01. Despliegue Físico: HTTPS

Para implementar **PROC-1-01** es necesario tener las condiciones físicas para llevarla a cabo al adquirir un certificado SSL. Por lo que se deja como recomendación en este caso.

3.5.2. PROC-1-02. Despliegue Físico: Arquitectura 2-Tier

Se utilizó una máquina virtual con Windows Server 2016 para el uso de la arquitectura 2-Tier. Tanto el cliente como el servidor se conectaron satisfactoriamente. Además, se instaló un Servidor de Bases de Datos SQL Server 2017 y la aplicación Web desplegada mediante IIS (Internet Information Services).

3.6. Uso de los Procedimientos del Grupo No. 2: Bases de Datos

Para demostrar la importancia de la seguridad en las bases de datos, ponemos de ejemplo dos bases de datos con que almacenamos información en nuestro proyecto mostrado de la siguiente estructura de bases de datos y sus tablas.

Base de Datos DB_Audit: contiene las tablas con registros basados en la auditoria de otras bases de datos y de la información de acciones dentro del proyecto Web MVC:

Figura 3.3 Tablas de la base de datos DB_Audit. Propio del autor.

- **LogonLogs**: registra la actividad de los usuarios al acceder al servidor SQL.
- **PayrollDBLogs**: registra los datos de nóminas insertados, borrados y actualizados de la tabla PayrollModels de la base de datos DB_Payroll.
- **SystemLogs**: registra la actividad de las acciones de la aplicación Web MVC, así como, errores, informaciones, acciones y mensajes de prevención.
- **BackupLogs**: registra la actividad de los backups realizados**.**

Para los backups manuales el administrador del sistema elige que base de datos desea guardar, y dependiendo de la base de datos seleccionada, y dependiendo de la base de datos seleccionada. La opción automática se realiza una vez al día, 1:00am por ser horario de que menos transacciones existentes en los servidores.

Base de Datos DB_Payroll: Tabla de información de las nóminas de pago usadas por el modelo del negocio.

Figura 3.4 Imagen de la tabla de la base de datos DB_Bayroll. Propio del autor.

- **PayrollModels**: registra la información pertinente a las nóminas relacionadas con los usuarios registrados que pertenecen al modelo del negocio.

3.6.1. PROC-2-01. Triggers

Con el uso de los triggers en las tablas utilizadas para gestionar la información se demuestra lo siguiente:

- Actualización de campos dinámicamente, al recibir datos de otra tabla esa toma esos datos, los modifica luego y los inserta o modifica en la tabla actual.
- Elimina la posibilidad de eliminar bases de datos del servidor.
- Elimina la posibilidad de crear bases de datos del servidor.
- Elimina la posibilidad que la tabla pueda ser eliminada del servidor.
- Elimina la posibilidad de crear tablas nuevas en el servidor.
- Controla el usuario, recurso, así como fecha y otros parámetros del uso del servidor de bases de datos.
- Audita los cambios para eliminación, inserción y modificación de registros pertinentes al modelo del negocio.
- Las modificaciones en los campos de las tablas quedan totalmente prohibidas por parte del servidor de SQL.
- Gestiona la creación de backups automáticamente y manualmente.

A continuación, se muestran un total de 3 ejemplos de los diferentes triggers; DML, DDL y Logon. Para los demás ejemplos de triggers, véase Anexo 1.

Tabla 3-1 Código de trigger DML. Propio del autor.

```
USE DB_Audit;
GO

CREATE TRIGGER trgAuditUpdateSalaryField
       ON DB_Audit.dbo.PayrollDBLogs
       FOR INSERT
       AS
       BEGIN
              UPDATE DB_Audit.dbo.PayrollDBLogs
              SET Salary = WorkedHours * 16

       END
```

Tabla 3-2 Código de trigger DDL. Propio del autor.

```
CREATE TRIGGER [trgDDLforDropTable]
ON ALL SERVER
FOR DROP_TABLE
AS
BEGIN
       DECLARE @LoginName NVARCHAR(MAX)
       SET @LoginName = ORIGINAL_LOGIN()

       IF @LoginName != 'sa'
       BEGIN
              ROLLBACK

              DECLARE @Datetime NVARCHAR(MAX)
              DECLARE @User NVARCHAR(MAX)
              DECLARE @HostName NVARCHAR(MAX)
              DECLARE @ProgramName NVARCHAR(MAX)
              DECLARE @FullBodyMsg NVARCHAR(MAX)

              SET @Datetime = GETDATE()
              SET @User = ORIGINAL_LOGIN()
              SET @HostName = HOST_NAME()
              SET @ProgramName = PROGRAM_NAME()
              SET @FullBodyMsg =  'Datetime: ' + @Datetime + '\r\n' +
                                  'User: ' + @User + '\r\n' +
                                  'Host Name: ' + @HostName + '\r\n' +
                                  'Program Name: ' + @ProgramName

              EXEC msdb.dbo.sp_send_dbmail
                     @profile_name = 'SQL Mail Services',
                     @recipients = 'admin@testmail.com',
                     @body = @FullBodyMsg,
                     @subject = 'DDL Manipulation Alert - Unauthorize DROP TABLE
statement attempt.';

              RAISERROR ('Server blocked up. Contact to the administrator.', 16, 10);
       END
END
```

Tabla 3-3 Código de trigger tipo Logon. Propio del autor.

```
CREATE TRIGGER [trgLogon]
	ON ALL SERVER
	FOR LOGON
	AS
	BEGIN
		INSERT INTO DB_Audit.dbo.LogonLogs
		(
			EventTime,
			LoginName,
			ClientHost,
			AppName,
			IsSupervised

		)
		VALUES
		(
			GETDATE(),
			ORIGINAL_LOGIN(),
			PROGRAM_NAME(),
			HOST_NAME(),
			0
		)
	END
```

3.6.2. PROC-2-02. Servicio de correos mediante SQL Server

Con el servicio de correos perteneciente al servidor de SQL podemos de manera independiente enviar correos si existe alguna manipulación de la base de datos de alguna fuente externa. Este servicio se instala mediante los scripts Jobs (ver Anexo 1).

3.6.3. PROC-2-03. Backups de las bases de datos

Para prevenir pérdida, mal manipulación y recuperación de registros de las bases de datos es necesario tener un sistema de respaldo de la información. Éste a su vez crea un registro con la información de creación de los backups que son visualizadas por parte de los administradores en la aplicación web, donde mediante la misma se crean los backups para las distintas bases de datos. Véase Anexo 1.

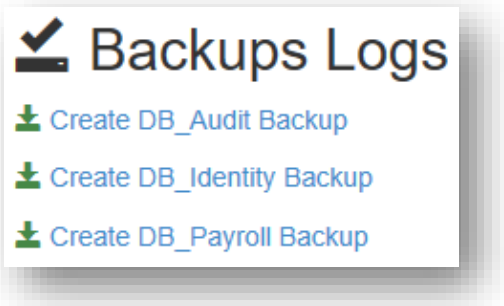

Figura 3.5 Opciones para crear Backups de Bases de Datos

Stored Procedures: Para la configuración del servicio de correo “Database Mail” por parte del Servidor de SQL Server 2017 se necesitaron los scripts mencionados en el capítulo anterior en el Modelo PROC-2-03. A continuación, se muestra el código del Stored Procedure de “***sp_Backup_DB_Payroll.sql***”. Véase Anexo 1.

Tabla 3-3 Código de Store Procedure de Log. Propio del autor.

```
USE DB_Payroll;
GO

CREATE PROCEDURE sp_Backup_DB_Payroll
AS
	BEGIN TRANSACTION
	DECLARE @path NVARCHAR(MAX)
	DECLARE @fullPath NVARCHAR(MAX)

	DECLARE @currentDate DATETIME
	DECLARE @fullDate NVARCHAR(MAX)

	DECLARE @timeHour NVARCHAR(MAX)
	DECLARE @timeMinute NVARCHAR(MAX)
	DECLARE @timeSecond NVARCHAR(MAX)
	DECLARE @fullTime NVARCHAR(MAX)

	DECLARE @name NVARCHAR(MAX)

	SET @path = N'C:\Users\kami\Documents\SQL Backups\'

	SET @currentDate = GETDATE()
	SET @fullDate = CONVERT(date, @currentDate)
	SET @timeHour = DATEPART(HOUR, @currentDate)
	SET @timeMinute = DATEPART(MINUTE, @currentDate)
	SET @timeSecond = DATEPART(SECOND, @currentDate)
	SET @fullTime = @timeHour + @timeMinute + @timeSecond

	SET @name = '_MANUAL_Backup_DB_Payroll.bak'
	SET @fullTime = @timeHour + @timeMinute + @timeSecond
	SET @fullPath = @path + @fullDate +  '_' + @fullTime + @name

	COMMIT
	BACKUP DATABASE DB_Audit
		TO  DISK = @fullPath

		WITH CHECKSUM;
```

3.7. Uso de los Procedimientos del Grupo No. 3: Patrones de Seguridad

3.7.1. PROC-3-01 y PROC-3-02. Patrones: Authentificator y Authorizator

Se pone en ejemplo el método de “Login” de la clase controladora “Account” así como:

- Las modificaciones si el usuario no se encuentra.
- Si no existen ningún usuario registrado.
- Si el usuario está activo en el sistema o no.
- La opción habilitada dado el caso que el usuario falle el intento 5 veces donde automáticamente se bloquea hasta que el administrador lo active nuevamente.
- La verificación de que el modelo de entrada sea válido.
- Las opciones de autentificación y autorización en cualquiera de los casos como, fallido, bloqueado, garantizado, así como las vistas que muestran las informaciones para cada caso.

A continuación, se muestra el método de acción asíncrono “Login” modificado para lograr los objetivos propuestos. Este método de acción chequea si el no existen usuarios en la base de datos, si el usuario entrado no existe y el más importante, es si el usuario existe, pero esté activado. Para observar el comportamiento de todas las clases véase Anexo 2.

Tabla 3-3 Código C# del método acción Login de la clase controladora AccountController. Propio del autor.

```
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
{
    ...
    //check if user exists
    var qUser = from n in appdbc.Users
                where n.Email == model.Email
                select n;
    if (qUser.Count() == 0)
    {
        ...
    }
    //check if exists 0 users in the database
    if (appdbc.Users.Count() == 0) {...}

    else
    {
        var qId = from n in appdbc.Users
                  where n.Email == model.Email
                  select n.Id;

        if (!appdbc.Users.Find(qId.Single()).IsActive) {...}
        ...
    }

}
```

3.7.2. PROC-3-03. Input Validation

ASP.NET contiene la clase `System.ComponentModel.DataAnnotations`, que ayudan a la implementación y uso de este patrón. Se modifica la clase PayrollModel que guarda relación con el modelo del negocio.

Tabla 3-4 Código C# de la clase PayrollModel. Propio del autor.

```
public class PayrollModel
{
        ...
        [Required]
        [Display(Name = "User")]
        [EmailAddress]
        public string UserName { get; set; }

        [Display(Name = "Worked Hours")]
        [Range(1, 8)]
        [Required]
        public int WorkedHours { get; set; }

        [DataType(DataType.Currency)]
        public double Salary { get; set; }
}
```

Para cada campo se selecciona una validación de entrada dependiendo del tipo de dato del campo o la intención a la hora de entrada de datos.

- **Display:** para poner el nombre deseado en vez del nombre del campo de la base de datos.
- **DataType:** los datos de este campo son mostrados en formato fecha sin hora.
- **Requiered:** es el más importante pues obliga a que el usuario no pase por alto llenar los campos antes de proseguir.
- **EmailAddress:** obliga a que el usuario obligatoriamente entre los datos en forma de email. Es un ejemplo de expresión regular.
- **Range:** limita a que la entrada del número entero en este caso sea de 1 a 8 para agregarle lógica al modelo del negocio al poder trabajar solamente un valor entre ese rango.

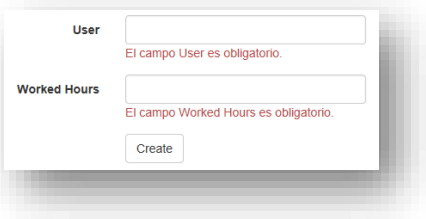

Figura 2.17 Opciones para crear Backups de Bases de Datos

3.7.3. PROC-3-04. Role Based Control

La base de datos DB_Identity alberga las tablas y relaciones de los usuarios, roles, etc. Esta base de datos es creada por Microsoft Identity y contiene la información necesaria para la gestión de usuarios, roles y relaciones entre ellos.

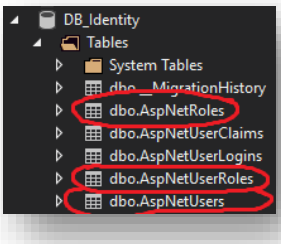

Figura 3.6 Opciones para crear Backups de Bases de Datos

Para el uso en aplicaciones Web de entornos de intranet, usaremos solamente dbo.AspNetRoles, dbo.AspNetUserRoles y dbo.AspNetUsers.

dbo.AspNetUserRoles: guarda la relación entre la tabla de los usuarios y la tabla de los roles.

	UserId	RoleId
	684d23f6-79ee-4b8a-b4c5-ef46a4be8cc9	1e96d384-f47b-4f79-ad25-3884412ee70b
	6d641ba2-ceb5-4a2f-a969-ba87f1cdb6ae	1e96d384-f47b-4f79-ad25-3884412ee70b
	c64eace7-43f3-4589-b812-78f9df638c63	1e96d384-f47b-4f79-ad25-3884412ee70b
	d485d5c5-8c44-4e5b-a4a0-12bf9e40e8ae	6e208f51-ca6a-4d02-b218-49fc46d3dd76
▷	NULL	NULL

Figura 3.7 Opciones para crear Backups de Bases de Datos

dbo.AspNetRoles: guarda el id del rol y el nombre del mismo. Así Microsoft Identity puede hacer uso del control de usuarios, roles, acceso a vistas, etc. en la aplicación web.

dbo.AspNetUsers: guarda toda la información correspondiente al usuario así como id, Fecha de Registro, si el usuario está activo, Nombre, Correo, contaseñas que se guardan en formato HASH CODE para protegerlos hasta de los propios administradores, número de teléfono y muchos otros que son necesarios para otras funciones.

3.7.4. PROC-3-05. Limited View

Para restringir las vistas, recursos o acceso a información por parte de los usuarios se controlan las vistas y los controladores y mediante los roles, usuarios y objetos se define para quién y cómo y la información es mostrada.

- **Limitación de vistas limitadas mediante control de autentificación:** Limita el uso del recurso si el usuario no está autentificado.

Tabla 3-4 Código C# Vistas limitadas. Propio del autor.

```
@if (!User.Identity.IsAuthenticated)
{
    <hgroup>
        <h1 class="text-danger">Welcome</h1>
        <h3 class="text-info">Please sign in or register to continue</h3>
    </hgroup>

}
```

- **Limitación de vistas limitadas mediante usuarios:** el acceso es garantizado si determinado usuario con permisos accede al recurso.

Tabla 3-5 Código C# Vistas limitadas. Propio del autor.

```
[Authorize(Users = "admin@testmail.com")]
public class AdminController : Controller
{
        ...

}
```

- **Limitación de vistas limitadas mediante roles:** en este caso el acceso al recurso está sujeto solamente a los roles elegidos.

Tabla 3-6 Código C# Vistas limitadas. Propio del autor.

```
[Authorize(Roles = "Administrator, Moderator")]
public class PayrollController : Controller
{
     ...

}
```

3.7.5. PROC-3-06. Pathname Cannonization

Esta opción el Framework ASP.NET seleccionado para el desarrollo del proyecto lo aplica de forma automática. Se trata del ocultamiento físico de los recursos a los usuarios que acceden a ellos. Estos muestran que dirección virtual es mostrada.

- http://localhost:1995/
- http://localhost:1995/Audit/SystemLogs

3.8. Uso de los Procedimientos del Grupo No. 4: Logs

A continuación, se muestra una tabla con los distintos tipos de logs generados en todas las áreas.

Tabla 3-7 Casos generados por Logs. Propio del autor.

	Caso
LogonLogs	Cuando se accede a la base de datos.
PayrollDBLogs	Registro de todo tipo de actividad en la tabla de Payroll.
SystemLogs	Cualquier intento de inicio de sesión con sus diversos resultados. Cierre de sesión. Cuando usuario se registra. Correos enviados por parte del sistema, si se envía o si ocurre error. Cuando se desactiva, desactiva usuarios. Cualquier tipo de error de creación de nómina. Registra actividad con backups de bases de datos.
BackupLogs	Registra actividad de creación de backups de las bases de datos.

3.8.1. PROC-4-01. Logs: Logs de Sistema

SystemDBLogs: Crean registros en la tabla DB_Audit\SystemLogs con los niveles Debug, Error, Warn, Info, Succeed para los distintos sucesos que ocurran. El sistema automáticamente está configurado para enviar correos si el usuario con rol administrador activa o desactiva, agrega, elimina otros usuarios, etc. Se crean distintos colores en representación del nivel de los logs. El siguiente ejemplo muestra cómo se ejecuta el método static ***Generate(…)*** para generar este tipo de log.

Tabla 3-9 Ejemplo de la clase SystemLog. Propio del autor.

```
if (qUser.Count() == 0)
            {
                SystemLogGenerator.Generate(DateTime.Now, "Model View User",
SystemLogLevel.Warn, "User " + model.Email + " failed to log in.");
                logWarn.Warn("User " + model.Email + " failed to log in.");
                return RedirectToAction("UserNotFound", "Error");

            }
```

3.8.2. PROC-4-02. Logs: Logs del modelo del negocio

PayrollDBLog: visualiza los elementos manipulados de la tabla *DB_Audit\SystemLogs*. Estos registros los generan los triggers en las tablas del SQL Server *trgPayrollAuditDelete*, *trgPayrollAuditDelete* y *trgPayrollAuditDelete*. Son insertados en base si se insertan, eliminan o actualizan registros en la tabla PayrollModels de la base de datos DB_Payroll. Véase Anexo 1 para consultar los triggers utilizados para ese tipo e logs.

3.8.3. PROC-4-03. Logs: Logs de acceso al servidor de base de datos

LogonLogs: es la tabla de la base de datos *DB_Audit\SystemLogs* que muestra los registros de accesos de logins independientes por parte de SQL Server con la información del evento. Se genera a partir del trigger *trgLogon* que está activado en todo el servidor. Para probarlo se hizo uso de la aplicación Navicat Premium que es un gestor de bases de datos. Al acceder el servidor el trigger se activó y generó un registro que podemos visualizarlo por medio de la aplicación. Véase Anexo 1 para consutar los triggers.

3.8.4. PROC-4-04. Logs: Logs de Salvas de las Bases de Datos.

BackupLogs: es una tabla de la base de datos *DB_Audit\BackupLogs* que crea registros si el administrador genera un backup de alguna base de datos o mediante el servicio automático generador de backups configurado en el servidor SQL. El procedimiento es crear un registro y este dispara el trigger *trgCreateBackup* que activan la creación de la copia del Backup.

3.8.5. PROC-4-05. Logs: Logs de un sistema de archivos externos txt

Logs de texto: mediante el uso de la librería *log4net*, se generan logs por independiente en forma de archivo externo con extensión .txt configurado para que almacene de forma distinta las trazas generadas por la aplicación. Está compuesto por 3 archivos, *logs_info.txt*, *logs_warn.txt* y *logs_error.txt*. con un máximo de peso 100kb.

Tabla 3-10 Instancias de la clase log4net. Propio del autor.

```
ILog logInfo = LogManager.GetLogger("logInfo");
ILog logWarn = LogManager.GetLogger("logWarn");
ILog logError = LogManager.GetLogger("logError");
```

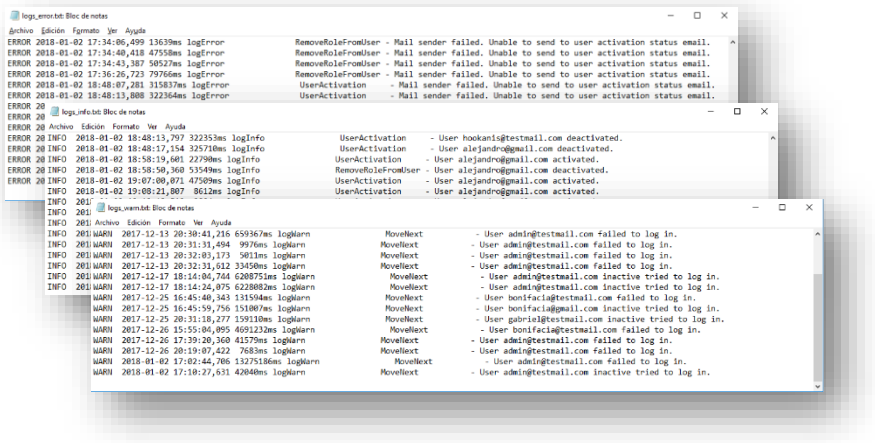

Figura 3.8 Representación de los Logs txt. Propio del autor.

3.9. Uso de los Procedimientos del Grupo No. 5: Externos

3.9.1. PROC-5-01. Logs: Activar/Desactivar usuario en caliente

Siempre que el usuario esté desactivado y el Administrador lo activa en el sistema, el usuario recibe un correo de notificación, así como asignación automática del rol de “User”. De forma contraria, si se desactiva un usuario por el Administrador, aunque las cookies estén activas el usuario debe verificarse antes de usar cualquier recurso de la clase controladora “*PayrollController*” en todos los métodos de acción. El siguiente ejemplo muestra uno de ellos y cómo verifica si el usuario autentificado está activo o no, si no lo está entonces lo redirecciona a la vista “Login”.

Tabla 3-11 Activación y desactivación en caliente de usuario. Propio del autor.

```
public ActionResult Index()
{
        if (adbc.Users.Find(User.Identity.GetUserId()).IsActive)
        {
                return View(db.Payrolls.ToList());
        }
        else
        {
                return RedirectToAction("Login", "Account");
        }

}
```

3.9.2. PROC-5-02. Logs: Uso de CAPTCHA

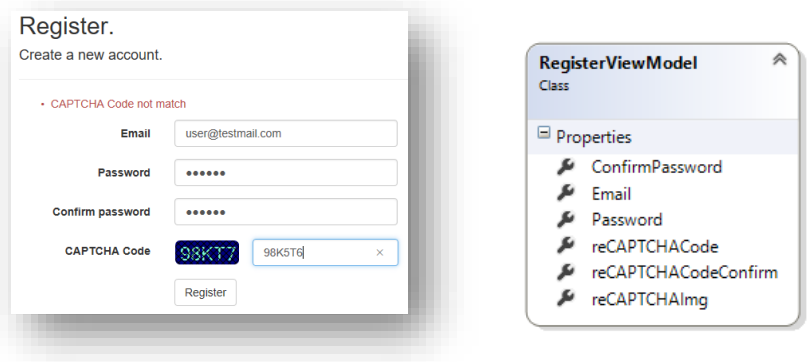

Figura 3.9 Diagrama de la clase modificada RegisterViewModel y vista en el navegador. Propio del autor.

La clase auxiliar Captcha se crea para generar el factor doble de autentificación. Esta clase es modificable y su funcionamiento consta de generar de forma independiente números del 0-9 y letras Aa-Zz del idioma ingles y mezclarlos con una variable Random. Una vez obtenido el código este es adjuntado en un objeto tipo imagen que es mostrado en la vista del modelo RegisterViewModel, donde es tratado como Input Validation, es decir, sin él no es válido el acceso. Ver a continuación el código de la clase auxiliar reCAPTCHA.

Tabla 3-12 Clase CAPTCHA. Propio del autor.

```
public class reCAPTCHA
    {
        string[] letters = new string[] { "a", "b", "c", "d", "e", "f", "g", "h", "i", "j",
"k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z" };
        int[] numbers = new int[] { 0, 1, 3, 4, 5, 6, 7, 8, 9 };
        Random r = new Random();
        string reCAPTCHACode = "";
        public Bitmap img { get; set ; }
        public string reCAPTCHAStringCode { get { return reCAPTCHACode; } }

        public reCAPTCHA()
        {
            GenerateCAPTCHACode();
            img = imgRenderedCAPTCHA();
        }
        void GenerateCAPTCHACode()
        {
            if (reCAPTCHACode.Count() < 5)
            {
                //if r.Next(1, 3) == 1 generate letter
                if (r.Next(1, 3) == 1)
                {
                    //if r.Next(1, 3) == 1 generate uppercase
                    if (r.Next(1, 3) == 1)
                    {
                        reCAPTCHACode += letters[r.Next(0, 26)].ToUpper();
                        GenerateCAPTCHACode();
                    }
                    else
                    {
                        reCAPTCHACode += letters[r.Next(0, 26)];
                        GenerateCAPTCHACode();
                    }
                }
                //generate generate number
                else
                {
                    reCAPTCHACode += numbers[r.Next(0, 9)];
                    GenerateCAPTCHACode();
                }
            }
        }
        Bitmap imgRenderedCAPTCHA()
        {
            Bitmap img = new Bitmap(80, 30);
            Graphics g = Graphics.FromImage(img);
            g.Clear(Color.Navy);
            g.DrawString(reCAPTCHACode, new Font("Courier", 18), new
SolidBrush(Color.Aquamarine), 2, 2);
            g.FillRectangle(new HatchBrush(HatchStyle.BackwardDiagonal, Color.FromArgb(255, 0,
0, 0), Color.Transparent), g.ClipBounds);
            g.FillRectangle(new HatchBrush(HatchStyle.ForwardDiagonal, Color.FromArgb(255, 0,
0, 0), Color.Transparent), g.ClipBounds);

            return img;
        }

    }
```

3.9.3. PROC-5-03. Logs: Expresiones Regulares

Para demostrar el uso de expresiones regulares se utiliza la clase del Modelo PayrollModel y se usa como ***Data Type Attribute*** que solamente acepte entradas del tipo expresión regular tipo correo como está configurado. Así en la vista correspondiente si el usuario entra un dato que no corresponda con el patrón devuelve un error de validación.

Tabla 3-13 Uso de expresión regular. Propio del autor.

```
[Required]
[Display(Name = "User")]
[RegularExpressionAttribute(@"\b([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.[0-
9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{2,4}|[0-
9]{1,3})(\]?)\b")]
public string UserName { get; set; }
```

3.9.4. PROC-5-04. Logs: Sendmail en la aplicación

Se crea la clase auxiliar MailSender y su función es el envío automático de notificaciones a los usuarios, administrador por parte del mismo o alguna cuenta de servicios creada en el servidor de correos. Para el uso del servicio de correos en la aplicación, debe estar instalado en el Servidor un cliente de correo y así Framework.NET hace uso de la clase **System.Web.Helpers.Webmail** mencionada previamente. La clase MailSender tiene cuatro métodos que hacen uso del servicio de correo dependiendo de cuando sea llamado. A continuación, se muestra el código de uno de los servicios implementados para el uso automático del servicio de correos por parte de la aplicación web.

Tabla 3-14 Ejemplo de método estático de la clase SendMail. Propio del autor.

```
public static void WhenLogin(string user)
        {
            WebMail.SmtpServer = "localhost";
            WebMail.SmtpPort = 25;
            WebMail.EnableSsl = false;
            WebMail.UserName = "serverservices@testmail.com";
            WebMail.Password = "kamisama";
            WebMail.From = "serverservices@testmail.com";
            WebMail.Send("admin@testmail.com", "New User Confirmation", "Please check your
admin manager page and activate the new user. User: " + user + ", " + "Date " +
DateTime.Now.ToLongDateString() + " - " + DateTime.Now.ToLongTimeString());

        }
```

3.10. Validación de la aplicación DEMO SMS con Vega

Se utiliza la Herramienta de Prueba de Software Vega en su última versión mostrando 3 errores.

- **Alto riesgo: ClearText Password over HTTP.** Este tipo de vulnerabilidad se trata con un adecuado certificado SSL para encriptar la información cuando viaja en una petición. Este certificado se adquiere mediante el pago debido.
- **Medio Riesgo: Local Filesystem Path Found.** Este tipo de vulnerabilidad atenta con mostrar las rutas fisicas de estructura de ficheros y directorios.
- **Medio Bajo: Form Password Field with autocomplete Enabled:** Esta vulnerabilidad mantiene en cookies los campos de usuarios y contraseñas.

Se buscan soluciones a 2 de los problemas encontrados apartando solamente al de ClearTextPassword over HTTP por no disponer de un certificado SSL.

Form Password Field with autocomplete Enabled fue corregido gracias a la opción deshabilitar del atributo autocomplete=”off” dentro de la etiqueta ***<input>.*** (Community, 2016)**.**

Local System Path Found fue corregido configurando el IIS habilitando “Basic Authentication”, deshabilitando el “Anonymous Authentication” o editando el archivo web.config del proyecto ASP.NET MVC y agregándole el siguiente código en xml: (ServerFault.com, 2011)

Tabla 3-15 Configuración de Basic Authentication mediante el archivo web.config. Propio del autor.

```
<system.webServer>
        <security>
            <authentication>
                <basicAuthentication enabled="true" />
            </authentication>
        </security>
    </system.webServer>
```

Utilizando .htacces (hypertext access) para servidores Apache permitiendo a los administradores aplicar distintas políticas de acceso a directorios o a archivos para mejorar la seguridad. Se pueden controlar varias configuraciones como redireccionamiento, mensajes personalizados. (Apache, 2018)

3.11. Conclusiones del Capítulo

En este capítulo se analizan los resultados del estudio a los trabajos de tesis presentados en años anteriores con datos estadísticos y gráficas. Se introduce la aplicación web DEMO “SMS” donde se demuestra el uso de los patrones y estrategias propuestas para el mejoramiento de la seguridad en el desarrollo del software, así como los resultados de estos.

CONCLUSIONES GENERALES

El resultado de la investigación desarrollada evidenció el cumplimiento de los objetivos propuestos, permitiendo así, llegar a las siguientes conclusiones:

- El estudio a los trabajos de tesis presentados en años anteriores demostró con datos estadísticos el estado actual de la seguridad dentro del desarrollo del software y aportó elementos para proporcionar solución a la problemática planteada.
- Partiendo del estudio de los distintos tipos de seguridad existentes, se seleccionó un gran número de los elementos dentro de las diferentes áreas del desarrollo del software de aplicaciones web de gestión y se agruparon para proponerlos como un modelo de procedimientos para el desarrollo de este tipo de aplicaciones.
- Se creó un modelo de procedimientos agnóstico a la tecnología para insertarlos dentro del desarrollo de aplicaciones web de gestión en entornos de intranet para mejorar la seguridad del producto final.
- Se demostró con ejemplos de código en algunas áreas dentro de la programación y mediante algunas pruebas resultados positivos al usar el modelo de procedimientos propuestos y mediante la utilización de la Herramienta Vega.

En sentido general el uso del modelo de procedimientos propuesto demuestra que mejora el desarrollo de aplicaciones web de gestión en entornos de intranet sin tener en cuenta la tecnología y avala la solución a la problemática planteada anteriormente.

RECOMENDACIONES

Luego de analizar el alcance de esta investigación y los resultados obtenidos, se recomienda que:

- Utilizar el modelo de procedimientos de seguridad agnóstico a la tecnología para mejorar el desarrollo de las aplicaciones web de gestión en entornos de intranet.
- Continuar investigando sobre la seguridad en las distintas áreas del desarrollo de aplicaciones de este tipo.

REFERENCIAS BIBLIOGRAFICAS

1. **Alfonso, Yeniel. 2010.** *Optimización de expresiones regulares a partir del autómata finito equivalente.* s.l. : Universidad de Matanzas Camilo Cienfuegos, 2010.

2. **Apache. 2018.** Apache. *Apache.* [En línea] 2018. http://httpd.apache.org/docs/.

3. **Apache Community. 2018.** log4net Library. [En línea] 2018. http://logging.apache.org/log4net/.

4. **Barnes, Rob. 2009.** Database Security and Auditing: Leading Practices. s.l. : Application Security, Inc., 2009.

5. **Booth, Joe. 2014.** *Regular Expressions.* 2014.

6. **Chanda, Sandeep. 2012.** *Microsoft Windows Identity Foundation Cookbook.* s.l. : PACKT Publishing, 2012.

7. **Cherencio, Guillermo. 2009.** *Desarrollo de Aplicaciones N-Tier (Versión 1.0).* 2009.

8. **Community, Stack Overflow. 2016.** Stack Overflow Community. *Stack Overflow Community.* [En línea] Stack Overflow Community, 2016. https://www.codeproject.com/Questions/1071390/How-to-stop-Auto-complete-option-for-Password.

9. **Cubadebate. 2017.** Cubadebate. *Cubadebate.* [En línea] 2017. http://www.cubadebate.cu/etiqueta/ciberdelitos/.

10. **Dangler, Jeremiah Y. 2013.** *Categorization of Security Design Patterns.* s.l. : East Tennessee State University, 2013.

11. *Database Security.* **Samarati, Pierangela, De Capitani di Vimercati, Sabrina y Jajodia, Sushil. 1999.** 1999.

12. **Domenech, JM. 1999.** *Métodos Estadísticos en Ciencias de Salud.* Barcelona : s.n., 1999.

13. *e-Ciencias de la Información.* **Semestral, Revista Electrónica. 2014.** 1, s.l. : Revista Electrónica Semestral, ISSN-1659-4142, 2014, Vol. 4. 2.

14. **Fernandez, Eduardo B. y Pan, Rouyi. 2001.** s.l. : Florida Atlantic University, 2001.

15. **Lobel, Leonard G. and Brust, Andrew J. 2012.** *Programming Microsoft SQL Server 2012.* s.l. : Microsoft, 2012.

16. **McMahon, Russ. 2012.** *Database Security SQL Server 2012.* s.l. : Associate Prof of Information Technology - CECH - UC University of Cincinaty, 2012.

17. **Michaelis, Mark y Lippert, Eric. 2016.** *Essential C# 6.0.* s.l. : The Addison-Wesley Microsoft Technology Series, 2016.

18. **Microsoft. 2017.** *Security Center for SQL Server Database Engine and Azure SQL Server.* s.l. : Microsoft, 2017.

19. **—. 2011.** Visual Studio Documentation. [Online] 2011. http://msdn.microsoft.com.

20. **Mozilla. 2018.** Mozilla. [Online] 2018. https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction.

21. **MSDN, Microsoft. 2018.** MSDN. *MSDN.* [En línea] Microsoft, 2018. https://msdn.microsoft.com/en-us/library/system.net.mail(v=vs.110).aspx.

22. **MSSQLTips. 2016.** MSSQLTips. *MSSQLTips.* [En línea] 2016. https://www.mssqltips.com/sqlservertutorial/21/sql-server-backup-log-command/.

23. **Naylor, Lee. 2016.** *ASP.NET. MVC with Entity Framework and CSS.* s.l. : APress, 2016.

24. **OWASP. 2018.** Project, The Open Web Application Security Project. [En línea] 2018. https://www.owasp.org/index.php/Main_Page.

25. *Security vulnerabilities of the top ten programming languages.* **Turner, Stephen. 2013.** s.l. : Known-Quantity.com, part of Turner & Associates, Inc., 2013.

26. **ServerFault.com. 2011.** ServerFault.com. *ServerFault.com.* [En línea] 2011. https://serverfault.com/questions/272290/iis-basic-authorization-ala-htaccess-htpasswd-in-apache.

27. **SSL. 2018.** SSL Site. [Online] 2018. https://https.cio.gov.

28. **Stack Overflow Comunity. 2016.** Stack Overflow. [Online] 2016. https://stackoverflow.com/.

29. **Symantec. 2018.** Symantec Security. [En línea] 2018. https://www.websecurity.symantec.com/.

30. **Trabajadores. 2017.** Periódico Trabajadores. *Periódico Trabajadores.* [En línea] 2017. http://www.trabajadores.cu.

31. **TrendMicro. 2007.** *Client Server Security 3.* s.l. : TrendMicro, 2007.

32. **Universidad Nacional Experimental del Táchira (UNET).** *Protocolos HTTP y HTTPS.*

33. **Verizon. 2016.** *2016 Breach Investigations Reports.* s.l. : Verizon, 2016.

34. **—.** *2017 Breach Investigations Reports.* s.l. : Verizon.

35. **—.** *2017 Breach Investigations Reports.* s.l. : Verizon.

36. **Wikipedia. 2018.** Wikipedia. [En línea] 2018. http://en.wikipedia.org.

37. **Williams, Laurie and Slankas, John.** *Database Security.* s.l. : NC State University.

Anexo 1: Triggers y procedures utilizados en SQL Server

Este Anexo contiene los scripts utilizados dentro de las bases de datos de SQL Server y tablas utilizadas en el proyecto MVC DEMO SMS.

Trigger `trgAuditUpdateSalaryField`

```
USE DB_Audit;
GO

CREATE TRIGGER trgAuditUpdateSalaryField
       ON DB_Audit.dbo.PayrollDBLogs
       FOR INSERT
       AS
       BEGIN
              UPDATE DB_Audit.dbo.PayrollDBLogs
              SET Salary = WorkedHours * 16

       END
```

Trigger `trgLogon`

```
CREATE TRIGGER [trgLogon]
       ON ALL SERVER
       FOR LOGON
       AS
       BEGIN
              INSERT INTO DB_Audit.dbo.LogonLogs
              (
                     EventTime,
                     LoginName,
                     ClientHost,
                     AppName,
                     IsSupervised

              )
              VALUES
              (
                     GETDATE(),
                     ORIGINAL_LOGIN(),
                     PROGRAM_NAME(),
                     HOST_NAME(),
                     0
              )
       END
```

Trigger `trgDDLforCreateDatabase`

```
CREATE TRIGGER [trgDDLforCreateDatabase]
ON ALL SERVER
FOR CREATE_DATABASE
AS
BEGIN
	DECLARE @LoginName NVARCHAR(MAX)
	SET @LoginName = ORIGINAL_LOGIN()

	IF @LoginName != 'sa'
	BEGIN
		ROLLBACK

		DECLARE @Datetime NVARCHAR(MAX)
		DECLARE @User NVARCHAR(MAX)
		DECLARE @HostName NVARCHAR(MAX)
		DECLARE @ProgramName NVARCHAR(MAX)
		DECLARE @FullBodyMsg NVARCHAR(MAX)

		SET @Datetime = GETDATE()
		SET @User = ORIGINAL_LOGIN()
		SET @HostName = HOST_NAME()
		SET @ProgramName = PROGRAM_NAME()
		SET @FullBodyMsg =  'Datetime: ' + @Datetime + '\r\n' +
                                    'User: ' + @User + '\r\n' +
                                    'Host Name: ' + @HostName + '\r\n' +
                                    'Program Name: ' + @ProgramName

		EXEC msdb.dbo.sp_send_dbmail
			@profile_name = 'SQL Mail Services',
			@recipients = 'admin@testmail.com',
			@body = @FullBodyMsg,
			@subject = 'DDL Manipulation Alert - Unauthorize CREATE DATABASE
statement attempt.';

		RAISERROR ('Server blocked up. Contact to the administrator.', 16, 10)
	END
END
```

Trigger `trgDDLforCreateTable`

```
CREATE TRIGGER [trgDDLforCreateTable]
ON ALL SERVER
FOR CREATE_TABLE
AS
BEGIN
	DECLARE @LoginName NVARCHAR(MAX)
	SET @LoginName = ORIGINAL_LOGIN()

	IF @LoginName != 'sa'
	BEGIN
		ROLLBACK

		DECLARE @Datetime NVARCHAR(MAX)
		DECLARE @User NVARCHAR(MAX)
		DECLARE @HostName NVARCHAR(MAX)
		DECLARE @ProgramName NVARCHAR(MAX)
		DECLARE @FullBodyMsg NVARCHAR(MAX)

		SET @Datetime = GETDATE()
		SET @User = ORIGINAL_LOGIN()
		SET @HostName = HOST_NAME()
		SET @ProgramName = PROGRAM_NAME()
		SET @FullBodyMsg =  'Datetime: ' + @Datetime + '\r\n' +
                                    'User: ' + @User + '\r\n' +
                                    'Host Name: ' + @HostName + '\r\n' +
                                    'Program Name: ' + @ProgramName

		EXEC msdb.dbo.sp_send_dbmail
			@profile_name = 'SQL Mail Services',
			@recipients = 'admin@testmail.com',
			@body = @FullBodyMsg,
			@subject = 'DDL Manipulation Alert - Unauthorize CREATE TABLE
statement attempt.';

		RAISERROR ('Server blocked up. Contact to the administrator.', 16, 10);
	END
END
```

Trigger `trgDDLforDropDatabase`

```
CREATE TRIGGER [trgDDLforDropDatabase]
ON ALL SERVER
FOR DROP_DATABASE
AS
BEGIN
	DECLARE @LoginName NVARCHAR(MAX)
	SET @LoginName = ORIGINAL_LOGIN()

	IF @LoginName != 'sa'
	BEGIN
		ROLLBACK

		DECLARE @Datetime NVARCHAR(MAX)
		DECLARE @User NVARCHAR(MAX)
		DECLARE @HostName NVARCHAR(MAX)
		DECLARE @ProgramName NVARCHAR(MAX)
		DECLARE @FullBodyMsg NVARCHAR(MAX)

		SET @Datetime = GETDATE()
		SET @User = ORIGINAL_LOGIN()
		SET @HostName = HOST_NAME()
		SET @ProgramName = PROGRAM_NAME()
		SET @FullBodyMsg =  'Datetime: ' + @Datetime + '\r\n' +
                                    'User: ' + @User + '\r\n' +
                                    'Host Name: ' + @HostName + '\r\n' +
                                    'Program Name: ' + @ProgramName

		EXEC msdb.dbo.sp_send_dbmail
			@profile_name = 'SQL Mail Services',
			@recipients = 'admin@testmail.com',
			@body = @FullBodyMsg,
			@subject = 'DDL Manipulation Alert - Unauthorize DROP DATABASE
statement attempt.';

		RAISERROR ('Server blocked up. Contact to the administrator.', 16, 10);
	END
END
```

Trigger `trgDDLforDropTable`

```
CREATE TRIGGER [trgDDLforDropTable]
ON ALL SERVER
FOR DROP_TABLE
AS
BEGIN
	DECLARE @LoginName NVARCHAR(MAX)
	SET @LoginName = ORIGINAL_LOGIN()

	IF @LoginName != 'sa'
	BEGIN
		ROLLBACK

		DECLARE @Datetime NVARCHAR(MAX)
		DECLARE @User NVARCHAR(MAX)
		DECLARE @HostName NVARCHAR(MAX)
		DECLARE @ProgramName NVARCHAR(MAX)
		DECLARE @FullBodyMsg NVARCHAR(MAX)

		SET @Datetime = GETDATE()
		SET @User = ORIGINAL_LOGIN()
		SET @HostName = HOST_NAME()
		SET @ProgramName = PROGRAM_NAME()
		SET @FullBodyMsg =  'Datetime: ' + @Datetime + '\r\n' +
                                    'User: ' + @User + '\r\n' +
                                    'Host Name: ' + @HostName + '\r\n' +
                                    'Program Name: ' + @ProgramName

		EXEC msdb.dbo.sp_send_dbmail
			@profile_name = 'SQL Mail Services',
			@recipients = 'admin@testmail.com',
			@body = @FullBodyMsg,
			@subject = 'DDL Manipulation Alert - Unauthorize DROP TABLE
statement attempt.';

		RAISERROR ('Server blocked up. Contact to the administrator.', 16, 10);
	END
END
```

Trigger `trgPayrollAuditDelete`

```
USE DB_Payroll;
GO

CREATE TRIGGER dbo.trgPayrollAuditDelete
ON dbo.PayrollModels
AFTER DELETE
AS
BEGIN
	INSERT INTO DB_Audit.dbo.PayrollDBLogs
	(
		RecordType,
		CurrentDate,
		DBUsername,
		PayrollDate,
		ModeratorId,
		Username,
		WorkedHours,
		Salary,
		IsSupervised
	)
	SELECT
		'DELETED',
		GETDATE(),
		SUSER_SNAME(),
		deleted.Date,
		deleted.ModeratorId,
		deleted.UserName,
		deleted.WorkedHours,
		deleted.Salary,
		0
	FROM deleted

END
```

Trigger `trgPayrollAuditInsert`

```
USE DB_Payroll;
GO

CREATE TRIGGER dbo.trgPayrollAuditInsert
ON dbo.PayrollModels
AFTER INSERT
AS
BEGIN
	-- insert from inserted
	INSERT INTO DB_Audit.dbo.PayrollDBLogs
	(
		RecordType,
		CurrentDate,
		DBUsername,
		PayrollDate,
		ModeratorId,
		Username,
		WorkedHours,
		Salary,
		IsSupervised
	)
	SELECT
		'INSERTED',
		GETDATE(),
		SUSER_SNAME(),
		inserted.Date,
		inserted.ModeratorId,
		inserted.UserName,
		inserted.WorkedHours,
		0,
		0
	FROM inserted

END
```

Trigger `trgPayrollAuditUpdate`

```
USE DB_Payroll;
GO
CREATE TRIGGER dbo.trgPayrollAuditUpdate
ON dbo.PayrollModels
AFTER UPDATE AS
IF (COLUMNS_UPDATED() & 32) > 0
BEGIN
	-- insert from inserted
	INSERT INTO DB_Audit.dbo.PayrollDBLogs
	(
		RecordType,
		CurrentDate,
		DBUsername,
		PayrollDate,
		ModeratorId,
		Username,
		WorkedHours,
		Salary,
		IsSupervised
	)
	SELECT
		'UPDATED NEW',
		GETDATE(),
		SUSER_SNAME(),
		inserted.Date,
		inserted.ModeratorId,
		inserted.UserName,
		inserted.WorkedHours,
		inserted.Salary,
		0
	FROM inserted

	-- insert from deleted
	INSERT INTO DB_Audit.dbo.PayrollDBLogs
	(
		RecordType,
		CurrentDate,
		DBUsername,
		PayrollDate,
		ModeratorId,
		Username,
		WorkedHours,
		Salary,
		IsSupervised
	)
	SELECT
		'UPDATED OLD',
		GETDATE(),
		SUSER_SNAME(),
		deleted.Date,
		deleted.ModeratorId,
		deleted.UserName,
		deleted.WorkedHours,
		deleted.Salary,
		0
	FROM deleted

END
```

Trigger `trgPayrollDBLogUpdateRaiseError`

```
USE DB_Audit;
GO

CREATE TRIGGER trgPayrollDBLogUpdateRaiseError
       ON [dbo].PayrollDBLogs
       FOR UPDATE
       AS IF (COLUMNS_UPDATED() & 1020) > 0
       BEGIN
              RAISERROR ('Cannot change this item.', 16, 10);

              EXEC msdb.dbo.sp_send_dbmail
                     @profile_name = 'SQL Mail Services',
                     @recipients = 'admin@testmail.com',
                     @query = 'SELECT login_time, login_name, host_name,
program_name
                                          FROM sys.dm_exec_sessions
                                          WHERE status = ''running''',
                     @subject = 'DML Manipulation Alert',
                     @attach_query_result_as_file = 1 ;

       END
```

Script `configureSQLServerEmailService`

```
USE msdb;
GO

sp_configure 'show advanced options',1
RECONFIGURE

--Enabling Database Mail
sp_configure 'Database Mail XPs',1
RECONFIGURE

--Creating a Profile
EXECUTE msdb.dbo.sysmail_add_profile_sp
       @profile_name = 'localhost',
       @description = 'Mail Service for SQL Server' ;

-- Create a Mail account for gmail. We have to use our company mail account.
EXECUTE msdb.dbo.sysmail_add_account_sp
       @account_name = 'SQL_Email_Account',
       @email_address = 'sqlemailservices@testmail.com',
       @mailserver_name = 'localhost',
       @port=25,
       @enable_ssl=0,
       @username='sqlemailservices@testmail.com',
       @password='kamisama'

-- Adding the account to the profile
EXECUTE msdb.dbo.sysmail_add_profileaccount_sp
       @profile_name = 'localhost',
       @account_name = 'SQL_Email_Account',
       @sequence_number =1 ;

-- Granting access to the profile to the DatabaseMailUserRole of MSDB
EXECUTE msdb.dbo.sysmail_add_principalprofile_sp
       @profile_name = 'localhost',
```

```
       @principal_id = 0,
       @is_default = 1 ;

--Sending Test Mail
EXEC msdb.dbo.sp_send_dbmail
       @profile_name = 'localhost',
       @recipients = 'admin@testmail.com',
       @body = 'Database Mail Testing...',
       @subject = 'Databas Mail from SQL Server';

--Verifying, check status column
SELECT * FROM sysmail_profile
SELECT * FROM sysmail_account
SELECT * FROM sysmail_profileaccount
SELECT * FROM sysmail_principalprofile
```

Script `createAutomaticBackup`

```
USE msdb ;
GO

EXEC dbo.sp_add_job
    @job_name = N'Backup_Database' ;
GO

EXEC sp_add_jobstep
    @job_name = N'Backup_Database',
    @step_name = N'Backup DB_Audit Database',
    @command = N'BACKUP DATABASE DB_Audit TO  DISK = ''C:\Temp\DB_Audit.bak''
WITH CHECKSUM; INSERT INTO DB_Audit.BackupLogs (EventTime, BackupName, Size,
Status) VALUES (''GETDATE(), ''AUTO_Backup_DB_Audit'', 0,''NOT COMMITED'')',
    @retry_attempts = 5,
    @retry_interval = 5 ;
GO

EXEC sp_add_jobstep
    @job_name = N'Backup_Database',
    @step_name = N'Backup DB_Identity Database',
    @command = N'BACKUP DATABASE DB_Identity TO  DISK =
''C:\Temp\DB_Identity.bak'' WITH CHECKSUM; INSERT INTO DB_Audit.BackupLogs
(EventTime, BackupName, Size, Status) VALUES (''GETDATE(),
''AUTO_Backup_DB_Identity'', 0,''NOT COMMITED'')',
    @retry_attempts = 5,
    @retry_interval = 5 ;
GO

EXEC sp_add_jobstep
    @job_name = N'Backup_Database',
    @step_name = N'Backup DB_Payroll Database',
    @command = N'BACKUP DATABASE DB_Payroll TO  DISK =
''C:\Temp\DB_Payroll.bak'' WITH CHECKSUM; INSERT INTO DB_Audit.BackupLogs
(EventTime, BackupName, Size, Status) VALUES (''GETDATE(),
''AUTO_Backup_DB_Payroll'', 0,''NOT COMMITED'')',
    @retry_attempts = 5,
    @retry_interval = 5 ;
GO

EXEC dbo.sp_add_schedule
    @schedule_name = N'RunDaily',
    @freq_type = 1,
    @active_start_time = 173800 ;
GO
```

```
USE msdb ;
GO

EXEC sp_attach_schedule
   @job_name = N'Backup_Database',
   @schedule_id = N'RunDaily'
GO

select * from sysschedules

EXEC dbo.sp_add_jobserver
    @job_name = N'Backup_Database';
GO

select * from sysschedules

select * from sysjobs
```

Stored Procedure `sp_Backup_DB_Audit`

```
USE DB_Audit;
GO

CREATE PROCEDURE sp_Backup_DB_Audit
AS
       BEGIN TRANSACTION
       DECLARE @path NVARCHAR(MAX)
       DECLARE @fullPath NVARCHAR(MAX)

       DECLARE @currentDate DATETIME
       DECLARE @fullDate NVARCHAR(MAX)

       DECLARE @timeHour NVARCHAR(MAX)
       DECLARE @timeMinute NVARCHAR(MAX)
       DECLARE @timeSecond NVARCHAR(MAX)
       DECLARE @fullTime NVARCHAR(MAX)

       DECLARE @name NVARCHAR(MAX)

       SET @path = N'C:\Users\kami\Documents\SQL Backups\'

       SET @currentDate = GETDATE()
       SET @fullDate = CONVERT(date, @currentDate)
       SET @timeHour = DATEPART(HOUR, @currentDate)
       SET @timeMinute = DATEPART(MINUTE, @currentDate)
       SET @timeSecond = DATEPART(SECOND, @currentDate)
       SET @fullTime = @timeHour + @timeMinute + @timeSecond

       SET @name = '_MANUAL_Backup_DB_Audit.bak'
       SET @fullTime = @timeHour + @timeMinute + @timeSecond
       SET @fullPath = @path + @fullDate +  '_' + @fullTime + @name

       COMMIT
       BACKUP DATABASE DB_Audit
              TO  DISK = @fullPath

              WITH CHECKSUM;
```

Stored Procedure `sp_Backup_DB_Audit_AUTO`

```
USE DB_Audit;
GO

CREATE PROCEDURE sp_Backup_DB_Audit_AUTO
AS
	BEGIN TRANSACTION
	DECLARE @path NVARCHAR(MAX)
	DECLARE @fullPath NVARCHAR(MAX)

	DECLARE @currentDate DATETIME
	DECLARE @fullDate NVARCHAR(MAX)

	DECLARE @timeHour NVARCHAR(MAX)
	DECLARE @timeMinute NVARCHAR(MAX)
	DECLARE @timeSecond NVARCHAR(MAX)
	DECLARE @fullTime NVARCHAR(MAX)

	DECLARE @name NVARCHAR(MAX)

	SET @path = N'C:\Users\kami\Documents\SQL Backups\'

	SET @currentDate = GETDATE()
	SET @fullDate = CONVERT(date, @currentDate)
	SET @timeHour = DATEPART(HOUR, @currentDate)
	SET @timeMinute = DATEPART(MINUTE, @currentDate)
	SET @timeSecond = DATEPART(SECOND, @currentDate)
	SET @fullTime = @timeHour + @timeMinute + @timeSecond

	SET @name = '_AUTO_Backup_DB_Audit.bak'
	SET @fullTime = @timeHour + @timeMinute + @timeSecond
	SET @fullPath = @path + @fullDate +  '_' + @fullTime + @name

	COMMIT
	BACKUP DATABASE DB_Audit
		TO  DISK = @fullPath

		WITH CHECKSUM;
```

Stored Procedure `sp_Backup_DB_Identity`

```
USE DB_Identity;
GO

CREATE PROCEDURE sp_Backup_DB_Identity
AS
	BEGIN TRANSACTION
	DECLARE @path NVARCHAR(MAX)
	DECLARE @fullPath NVARCHAR(MAX)

	DECLARE @currentDate DATETIME
	DECLARE @fullDate NVARCHAR(MAX)

	DECLARE @timeHour NVARCHAR(MAX)
	DECLARE @timeMinute NVARCHAR(MAX)
	DECLARE @timeSecond NVARCHAR(MAX)
	DECLARE @fullTime NVARCHAR(MAX)

	DECLARE @name NVARCHAR(MAX)
```

```
        SET @path = N'C:\Users\kami\Documents\SQL Backups\'

        SET @currentDate = GETDATE()
        SET @fullDate = CONVERT(date, @currentDate)
        SET @timeHour = DATEPART(HOUR, @currentDate)
        SET @timeMinute = DATEPART(MINUTE, @currentDate)
        SET @timeSecond = DATEPART(SECOND, @currentDate)
        SET @fullTime = @timeHour + @timeMinute + @timeSecond

        SET @name = '_MANUAL_Backup_DB_Identity.bak'
        SET @fullTime = @timeHour + @timeMinute + @timeSecond
        SET @fullPath = @path + @fullDate +  '_' + @fullTime + @name

        COMMIT
        BACKUP DATABASE DB_Audit
               TO  DISK = @fullPath

               WITH CHECKSUM;
```

Stored Procedure `sp_Backup_DB_Identity_AUTO`

```
USE DB_Identity;
GO

CREATE PROCEDURE sp_Backup_DB_Identity_AUTO
AS
        BEGIN TRANSACTION
        DECLARE @path NVARCHAR(MAX)
        DECLARE @fullPath NVARCHAR(MAX)

        DECLARE @currentDate DATETIME
        DECLARE @fullDate NVARCHAR(MAX)

        DECLARE @timeHour NVARCHAR(MAX)
        DECLARE @timeMinute NVARCHAR(MAX)
        DECLARE @timeSecond NVARCHAR(MAX)
        DECLARE @fullTime NVARCHAR(MAX)

        DECLARE @name NVARCHAR(MAX)

        SET @path = N'C:\Users\kami\Documents\SQL Backups\'

        SET @currentDate = GETDATE()
        SET @fullDate = CONVERT(date, @currentDate)
        SET @timeHour = DATEPART(HOUR, @currentDate)
        SET @timeMinute = DATEPART(MINUTE, @currentDate)
        SET @timeSecond = DATEPART(SECOND, @currentDate)
        SET @fullTime = @timeHour + @timeMinute + @timeSecond

        SET @name = '_AUTO_Backup_DB_Identity.bak'
        SET @fullTime = @timeHour + @timeMinute + @timeSecond
        SET @fullPath = @path + @fullDate +  '_' + @fullTime + @name

        COMMIT
        BACKUP DATABASE DB_Audit
               TO  DISK = @fullPath

               WITH CHECKSUM;
```

Stored Procedure `sp_Backup_DB_Payroll`

```
USE DB_Payroll;
GO

CREATE PROCEDURE sp_Backup_DB_Payroll
AS
	BEGIN TRANSACTION
	DECLARE @path NVARCHAR(MAX)
	DECLARE @fullPath NVARCHAR(MAX)

	DECLARE @currentDate DATETIME
	DECLARE @fullDate NVARCHAR(MAX)

	DECLARE @timeHour NVARCHAR(MAX)
	DECLARE @timeMinute NVARCHAR(MAX)
	DECLARE @timeSecond NVARCHAR(MAX)
	DECLARE @fullTime NVARCHAR(MAX)

	DECLARE @name NVARCHAR(MAX)

	SET @path = N'C:\Users\kami\Documents\SQL Backups\'

	SET @currentDate = GETDATE()
	SET @fullDate = CONVERT(date, @currentDate)
	SET @timeHour = DATEPART(HOUR, @currentDate)
	SET @timeMinute = DATEPART(MINUTE, @currentDate)
	SET @timeSecond = DATEPART(SECOND, @currentDate)
	SET @fullTime = @timeHour + @timeMinute + @timeSecond

	SET @name = '_MANUAL_Backup_DB_Payroll.bak'
	SET @fullTime = @timeHour + @timeMinute + @timeSecond
	SET @fullPath = @path + @fullDate +  '_' + @fullTime + @name

	COMMIT
	BACKUP DATABASE DB_Audit
		TO  DISK = @fullPath

		WITH CHECKSUM;
```

Stored Procedure `sp_Backup_DB_Payroll_AUTO`

```
USE DB_Payroll;
GO

CREATE PROCEDURE sp_Backup_DB_Payroll_AUTO
AS
	BEGIN TRANSACTION
	DECLARE @path NVARCHAR(MAX)
	DECLARE @fullPath NVARCHAR(MAX)

	DECLARE @currentDate DATETIME
	DECLARE @fullDate NVARCHAR(MAX)

	DECLARE @timeHour NVARCHAR(MAX)
	DECLARE @timeMinute NVARCHAR(MAX)
	DECLARE @timeSecond NVARCHAR(MAX)
	DECLARE @fullTime NVARCHAR(MAX)

	DECLARE @name NVARCHAR(MAX)
```

```
SET @path = N'C:\Users\kami\Documents\SQL Backups\'

SET @currentDate = GETDATE()
SET @fullDate = CONVERT(date, @currentDate)
SET @timeHour = DATEPART(HOUR, @currentDate)
SET @timeMinute = DATEPART(MINUTE, @currentDate)
SET @timeSecond = DATEPART(SECOND, @currentDate)
SET @fullTime = @timeHour + @timeMinute + @timeSecond

SET @name = '_AUTO_Backup_DB_Payroll.bak'
SET @fullTime = @timeHour + @timeMinute + @timeSecond
SET @fullPath = @path + @fullDate +  '_' + @fullTime + @name

COMMIT
BACKUP DATABASE DB_Audit
       TO  DISK = @fullPath
       WITH CHECKSUM;
```

Anexo 2: Clases utilizadas y modificadas en C#.

Este anexo contiene clases creadas y modificadas ya existentes dentro del proyectó MVC DEMO SMS.

Clase `LoginViewModel`

```
public class LoginViewModel
    {
        [Required]
        [Display(Name = "Email")]
        [EmailAddress]
        public string Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [Display(Name = "Remember me?")]
        public bool RememberMe { get; set; }

        public string reCAPTCHACodeConfirm { get; set; }
        public Bitmap reCAPTCHAImg { get; set; }

        [Required]
        [Display(Name = "CAPTCHA Code")]
        [Compare("reCAPTCHACodeConfirm", ErrorMessage = "CAPTCHA Code not
match")]
        public string reCAPTCHACode { get; set; }

    }
```

Clase `RegisterViewModel`

```
public class RegisterViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Email")]
        public string Email { get; set; }
        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2}
characters long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }
        [DataType(DataType.Password)]
        [Display(Name = "Confirm password")]
        [Compare("Password", ErrorMessage = "The password and confirmation
password do not match.")]
        public string ConfirmPassword { get; set; }
        public string reCAPTCHACodeConfirm { get; set; }
        public Bitmap reCAPTCHAImg { get; set; }
        [Required]
        [Display(Name = "CAPTCHA Code")]
```

```
        [Compare("reCAPTCHACodeConfirm", ErrorMessage = "CAPTCHA Code not
match")]
        public string reCAPTCHACode { get; set; }

    }
```

Clase AdminRoleViewModel

```
public class AdminRoleViewModel
    {
        public string Id { get; set; }

        [Display(Name = "Role Name")]
        public string RoleName { get; set; }

        [Display(Name = "Users")]
        public IEnumerable<string> Users { get; set; }

    }
```

Clase AdminUserViewModel

```
public class AdminUserViewModel
    {
        public string Id { get; set; }

        [Display(Name = "Registration Date")]
        public string RegistrationDate { get; set; }

        [Required]
        [EmailAddress]
        [Display(Name ="User Name")]
        public string UserName { get; set; }
        [Display(Name = "Roles")]
        public IEnumerable<string> UserRoles { get; set; }

        [Display(Name = "Is Active")]
        public bool IsActive { get; set; }

    }
```

Clase ApplicationUser

```
public class ApplicationUser : IdentityUser
    {
        [Display(Name ="Date")]
        public string RegistrationDate { get; set; }

        [Display(Name ="Is Active")]
        public bool IsActive { get; set; }

        ...
        }

    }
```

Clase `MailSender`

```
public class MailSender
    {
        public static void WhenLogin(string user)
        {
            WebMail.SmtpServer = "localhost";
            WebMail.SmtpPort = 25;
            WebMail.EnableSsl = false;
            WebMail.UserName = "serverservices@testmail.com";
            WebMail.Password = "kamisama";
            WebMail.From = "serverservices@testmail.com";
            WebMail.Send("admin@testmail.com", "New User Confirmation", "Please
check your admin manager page and activate the new user. User: " + user + ", "
+ "Date " + DateTime.Now.ToLongDateString() + " - " +
DateTime.Now.ToLongTimeString());
        }
        public static void WhenIsActivate(string user)
        {
            WebMail.SmtpServer = "localhost";
            WebMail.SmtpPort = 25;
            WebMail.EnableSsl = false;
            WebMail.UserName = "serverservices@testmail.com";
            WebMail.Password = "kamisama";
            WebMail.From = "serverservices@testmail.com";
            WebMail.Send(user, "Account inactive", "Hello " + user + ", " +
"your account is inactive now due to security reasons.");
        }
        public static void WhenIsDeactivated(string user)
        {
            WebMail.SmtpServer = "localhost";
            WebMail.SmtpPort = 25;
            WebMail.EnableSsl = false;
            WebMail.UserName = "serverservices@testmail.com";
            WebMail.Password = "kamisama";
            WebMail.From = "serverservices@testmail.com";
            WebMail.Send(user, "Account active", "Hello " + user + ", " + "your
account is active now. Try to login in. If you have problems please reply this
email. Thanks!!!");
        }
        public static void WhenIsDeleted(string user)
        {
            WebMail.SmtpServer = "localhost";
            WebMail.SmtpPort = 25;
            WebMail.EnableSsl = false;
            WebMail.UserName = "serverservices@testmail.com";
            WebMail.Password = "kamisama";
            WebMail.From = "serverservices@testmail.com";
            WebMail.Send(user, "Account deleted", "Hello ex-user " + user + ",
" + "your account hass been deleted due to security reasons.");

        }
```

Clase `PayrollModel`

```
public class PayrollModel
    {
        [Display(Name = "No.")]
        public int Id { get; set; }

        [DataType(DataType.Date)]
        public DateTime Date { get; set; }

        [Display(Name = "Moderator Id")]
        public string ModeratorId { get; set; }

        [Required]
        [Display(Name = "User")]
        [RegularExpressionAttribute(@"\b([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.[0-
9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{2,4}|[0-
9]{1,3})(\]?)\b")]
        public string UserName { get; set; }
        [Display(Name = "Worked Hours")]
        [Range(1, 8)]
        [Required]
        public int WorkedHours { get; set; }

        [DataType(DataType.Currency)]
        public double Salary { get; set; }

    }
```

Clase `UserRoleViewModel`

```
public class UserRoleViewModel
    {
        public string userId { get; set; }

        [Required]
        [Display(Name = "Role name")]
        public string RoleName { get; set; }

    }
```

Clase reCAPTCHA

```
public class reCAPTCHA
    {
        string[] letters = new string[] { "a", "b", "c", "d", "e", "f", "g",
"h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w",
"x", "y", "z" };
        int[] numbers = new int[] { 0, 1, 3, 4, 5, 6, 7, 8, 9 };
        Random r = new Random();
        string reCAPTCHACode = "";
        public Bitmap img { get; set ; }
        public string reCAPTCHAStringCode { get { return reCAPTCHACode; } }
        public reCAPTCHA()
        {
            GenerateCAPTCHACode();
            img = imgRenderedCAPTCHA();
        }
        void GenerateCAPTCHACode()
        {
            if (reCAPTCHACode.Count() < 5)
            {
                //if r.Next(1, 3) == 1 generate letter
                if (r.Next(1, 3) == 1)
                {
                    //if r.Next(1, 3) == 1 generate uppercase
                    if (r.Next(1, 3) == 1)
                    {
                        reCAPTCHACode += letters[r.Next(0, 26)].ToUpper();
                        GenerateCAPTCHACode();
                    }
                    else
                    {
                        reCAPTCHACode += letters[r.Next(0, 26)];
                        GenerateCAPTCHACode();
                    }
                }
                //generate generate number
                else
                {
                    reCAPTCHACode += numbers[r.Next(0, 9)];
                    GenerateCAPTCHACode();
                }
            }
        }
        Bitmap imgRenderedCAPTCHA()
        {
            Bitmap img = new Bitmap(80, 30);
            Graphics g = Graphics.FromImage(img);
            g.Clear(Color.Navy);
            g.DrawString(reCAPTCHACode, new Font("Courier", 18), new
SolidBrush(Color.Aquamarine), 2, 2);
            g.FillRectangle(new HatchBrush(HatchStyle.BackwardDiagonal,
Color.FromArgb(255, 0, 0, 0), Color.Transparent), g.ClipBounds);
            g.FillRectangle(new HatchBrush(HatchStyle.ForwardDiagonal,
Color.FromArgb(255, 0, 0, 0), Color.Transparent), g.ClipBounds);

            return img;
        }

    }
```

Clase `SystemLogGenerator`

```
public class SystemLogGenerator
    {
        public static void Generate(DateTime date, string userId,
SystemLogLevel level, string message)
        {
            var db = new DbContext_Audit();
            var sysLog = new SystemLog();
            sysLog.Date = date;
            sysLog.UserId = userId;
            sysLog.Level = level;
            sysLog.Message = message;

            db.SystemLogs.Add(sysLog);
            db.SaveChanges();
        }

    }
```

Clase `BackupLogs`

```
public class BackupLogs
    {
        public int Id { get; set; }
        [Display(Name = "Event Time")]
        public DateTime EventTime { get; set; }
        [Display(Name = "Backup Name")]
        public string BackupName { get; set; }
        [Display(Name ="Size (KBytes)")]
        public double Size { get; set; }
        public string Status { get; set; }
        public bool IsSupervised { get; set; }


    }
```

Clase `LogonLog`

```
public class LogonLog
    {
        public int Id { get; set; }
        [Display(Name = "Event Time")]
        public DateTime EventTime { get; set; }
        [Display(Name = "Login Name")]
        public string LoginName { get; set; }
        [Display(Name = "Client Host")]
        public string ClientHost { get; set; }
        [Display(Name = "App Name")]
        public string AppName { get; set; }
        public bool IsSupervised { get; set; }

    }
```

Clase `PayrollDBLog`

```
public class PayrollDBLog
    {
        public int Id { get; set; }
        public string RecordType { get; set; }
        [Display(Name ="Current Date")]
        public DateTime CurrentDate { get; set; }
        [Display(Name = "Data Base User")]
        public string DBUsername { get; set; }
        [Display(Name = "Payroll Date")]
        public DateTime PayrollDate { get; set; }
        [Display(Name = "Moderator Id")]
        public string ModeratorId { get; set; }
        public string Username { get; set; }
        [Display(Name = "Worked hours")]
        public int WorkedHours { get; set; }
        [DataType(DataType.Currency)]
        public double Salary { get; set; }
        public bool IsSupervised { get; set; }

    }
```

Clase `SystemLog`

```
public class SystemLog
    {
        public int Id { get; set; }
        [Display(Name ="Date - Time")]
        //[DisplayFormat(DataFormatString = "{0:yyyy/MM/dd - HH:MM:ss}")]
        public DateTime Date { get; set; }
        [Display(Name ="User Id")]
        public string UserId { get; set; }
        public SystemLogLevel Level { get; set; }
        //public string Level { get; set; }
        public string Message { get; set; }
        public bool IsSupervised { get; set; }


    }
```

Clase `SystemLogLevel`

```
public enum SystemLogLevel
    {
        Debug,
        Error,
        Warn,
        Info,
        Succeed

    }
```

Clase controladora AccountController Método async Login

```
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
        {
            var appdbc = new DbContext_Identity();
            var result = SignInStatus.LockedOut;

            //check if user exists
            var qUser = from n in appdbc.Users
                        where n.Email == model.Email
                        select n;

            if (qUser.Count() == 0)
            {
                SystemLogGenerator.Generate(DateTime.Now, "Model View User",
SystemLogLevel.Warn, "User " + model.Email + " failed to log in.");
                logWarn.Warn("User " + model.Email + " failed to log in.");
                return RedirectToAction("UserNotFound", "Error");
            }

            //check if exists 0 users in the database
            if (appdbc.Users.Count() == 0)
            {
                return RedirectToAction("UserNotFound", "Error");
            }

            else
            {
                var qId = from n in appdbc.Users
                          where n.Email == model.Email
                          select n.Id;

                if (!appdbc.Users.Find(qId.Single()).IsActive)
                {
                    SystemLogGenerator.Generate(DateTime.Now, qId.Single(),
SystemLogLevel.Warn, "User " + model.Email + " inactive tried to log in.");
                    logWarn.Warn("User " + model.Email + " inactive tried to
log in.");

                    return View("Lockout");
                }

                if (!ModelState.IsValid)
                {
                    return View(model);
                }

                else
                {
                    result = await
SignInManager.PasswordSignInAsync(model.Email, model.Password,
model.RememberMe, shouldLockout: true);

                    switch (result)
                    {
                        case SignInStatus.Success:
                            {
                                SystemLogGenerator.Generate(DateTime.Now,
qId.Single(), SystemLogLevel.Succeed, "User logon succeeded.");
                                logInfo.Info("User " + qId.Single() + " logon
succeeded.");
```

```
                            return RedirectToLocal(returnUrl);
                        }

                    case SignInStatus.LockedOut:
                        {
                            SystemLogGenerator.Generate(DateTime.Now,
qId.Single(), SystemLogLevel.Warn, "User account locked out, login attempt
failed 5 times.");

                            try
                            {
                                MailSender.WhenIsDeactivated(model.Email);

MailSender.WhenIsDeactivated("admin@testmail.com");
                                SystemLogGenerator.Generate(DateTime.Now,
qId.Single(), SystemLogLevel.Info, "Email sender services OK. Email sent to
user " + qId.Single());
                            }
                            catch (Exception)
                            {
                                SystemLogGenerator.Generate(DateTime.Now,
User.Identity.GetUserId(), SystemLogLevel.Error, "Automatic email sender
services failed to send user lockout's notification to " + qId.Single() + ".");
                                logError.Error("Automatic email sender
services failed.");
                                throw;
                            }

                            return View("Lockout");
                        }
                    case SignInStatus.RequiresVerification:
                        return RedirectToAction("SendCode", new { ReturnUrl
= returnUrl, RememberMe = model.RememberMe });
                    case SignInStatus.Failure:
                    default:
                        ModelState.AddModelError("", "Invalid login
attempt.");
                        SystemLogGenerator.Generate(DateTime.Now,
User.Identity.GetUserId(), SystemLogLevel.Warn, "User " + model.Email + " login
attempt failed.");
                        logInfo.Info("User " + model.Email + " login
attempt failed.");

                        return View(model);
                }
            }
        }

    }
```

Clase controladora AccountControler Método LogOff

```
public ActionResult LogOff()
        {
            SystemLogGenerator.Generate(DateTime.Now,
User.Identity.GetUserId(), SystemLogLevel.Succeed, "User logout succeeded.");

AuthenticationManager.SignOut(DefaultAuthenticationTypes.ApplicationCookie);

            return RedirectToAction("Index", "Home");        }
```

Clase Controladora HomeController Metodo Index

```
public async Task<ActionResult> Index()
        {
            DbContext_Identity db = new DbContext_Identity();

            //check if there's any user created. this runs only once, first
time DB is empty
            if (!db.Database.Exists())
            {
                db.Database.Create();
                var userManager = new ApplicationUserManager(new
UserStore<ApplicationUser>(db));

                //crete role administrator
                IdentityRole roleAdmin = new IdentityRole("Administrator");
                IdentityRole roleModerator = new IdentityRole("Moderator");
                IdentityRole roleUser = new IdentityRole("User");

                db.Roles.Add(roleAdmin);
                db.Roles.Add(roleModerator);
                db.Roles.Add(roleUser);
                db.SaveChanges();

                //create admin user
                ApplicationUser userAdmin = new ApplicationUser { UserName =
"admin@testmail.com", Email = "admin@testmail.com", IsActive = true,
RegistrationDate = DateTime.Now.Date.ToString(), LockoutEnabled = false };
                var resultCreateUser = await userManager.CreateAsync(userAdmin,
"K@mi87");

                //bind role with user
                var resultAssignRole = await
userManager.AddToRoleAsync(userAdmin.Id, roleAdmin.Name);

                if ((resultCreateUser.Succeeded) &&
(resultAssignRole.Succeeded))
                {

                    return View();
                }

                else
                {
                    return View("SeedDatabaseError");
                }
            }

            return View();
                                        }
```

Clase Controladora PayrollControler Metodo de acción Create

```
[HttpPost]
        public ActionResult Create(PayrollModel payrollModel)
        {
            if (adbc.Users.Find(User.Identity.GetUserId()).IsActive)
            {
                if (ModelState.IsValid)
                {
                    //payroll user don't exists
                    var qUsers = from n in adbc.Users
                                 where n.UserName == payrollModel.UserName
                                 select n;

                    //compare if user not found or put money to itself or user
not active
                    if ((qUsers.Count() == 0) || (payrollModel.UserName ==
User.Identity.Name) || qUsers.Single().IsActive == false)
                    {
                        SystemLogGenerator.Generate(DateTime.Now,
User.Identity.GetUserId(), SystemLogLevel.Warn, "Failed to create a payroll.
Unable to select user.");
                        logWarn.Warn(User.Identity.GetUserId() + " Failed to
create a payroll. Unable to select user.");
                        return RedirectToAction("UserNotFound", "Error");
                    }

                    payrollModel.Date = DateTime.Now;
                    payrollModel.ModeratorId = User.Identity.GetUserId();
                    payrollModel.Salary = payrollModel.WorkedHours * 16;

                    db.Payrolls.Add(payrollModel);
                    db.SaveChanges();

                    SystemLogGenerator.Generate(DateTime.Now,
User.Identity.GetUserId(), SystemLogLevel.Succeed, "Payroll " + payrollModel.Id
+ " created.");
                    logInfo.Info("Payroll created.");

                    return RedirectToAction("Index");
                }

                return View(payrollModel);
            }

            else
            {
                return RedirectToAction("Login", "Account");

                                        }
```

Clase controladora RolesController

```
public class RolesController : Controller
    {
        ILog logInfo = LogManager.GetLogger("logInfo");
        ILog logWarn = LogManager.GetLogger("logWarn");
        ILog logError = LogManager.GetLogger("logError");

        private DbContext_Identity db = new DbContext_Identity();

        public ActionResult Index()
        {
            var am = from n in db.Roles
                     orderby n.Name
                     select new AdminRoleViewModel()
                     {
                         Id = n.Id,
                         RoleName = n.Name,
                         Users = from x in n.Users
                                 join t in db.Users on x.UserId equals t.Id
                                 orderby t.UserName
                                 select t.UserName
                     };

            return View(am);
        }
                                        }
```

Véase Clase Controladora UsersController.cs dentro del proyecto para más información sobre la clase y su funcionamiento.

Clase IdentityConfig.cs: Metodo Create

```
// Configure validation logic for passwords
            manager.PasswordValidator = new PasswordValidator
            {
                RequiredLength = 6,
                RequireNonLetterOrDigit = true,
                RequireDigit = true,
                RequireLowercase = true,
                RequireUppercase = true,

            };


// Configure user lockout defaults
            manager.UserLockoutEnabledByDefault = true;
            manager.DefaultAccountLockoutTimeSpan =
TimeSpan.FromMinutes(5);

            manager.MaxFailedAccessAttemptsBeforeLockout = 5;
```