UNIVERSIDAD NACIONAL TECNOLÓGICA DE LIMA SUR

FACULTAD DE ADMINISTRACIÓN DE EMPRESAS E INGENIERÍA DE SISTEMAS

CARRERA PROFESIONAL INGENIERÍA DE SISTEMAS

“DESARROLLO DE UN SISTEMA GENERADOR DE CÓDIGO FUENTE JOMAKE-CODE PARA MEJORAR EL PROCESO DE DESARROLLO DE SOFTWARE EN EL ÁREA DE SISTEMAS DE LA EMPRESA PROCAMPO S.A.”

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de

INGENIERO DE SISTEMAS

PRESENTADO POR EL BACHILLER

RAMOS ZAPANA, KERVY JUNIOR

Villa El Salvador
2016

DEDICATORIA

A mis padres

John Walter Ramos Mosqueira y María Clelia Zapana Zapana

Por todo su apoyo incondicional en todo momento de mi vida, por siempre estar en los momentos felices y difíciles de mi vida, por aquellos consejos, palabras que me motivaron a seguir en pie de lucha cuando sentía que ya todo estaba perdido, por su comprensión, ayuda, amor que siempre me brindan y sobre todo por la confianza depositada en mí, porque hoy en día soy lo que soy gracias a ellos.

A mis hermanos

Carolina María Ramos Zapana y John Ramos Zapana

Por su incondicional apoyo y motivación a lo largo de toda mi vida, por esos buenos consejos y palabras de motivación para seguir adelante y sobre todo por siempre estar en los momentos claves de mi vida apoyándonos siempre en todo momento.

AGRADECIMIENTO

A papá Dios

Por darme la vida, por haberme acompañado, guiado y permitirme el haber llegado hasta este momento tan importante de mi formación profesional con salud, fortaleza para poder seguir luchando por mis sueños.

A mis padres

Por enseñarme que todo sacrificio es recompensado, que si uno se propone algo lucha por sus ideales y lo consigue sin importar que tan lejos esté de ello practicando la perseverancia.

Y sobre todo por enseñarme el verdadero significado de Familia

A mi Padrino

Cesar Raúl Ramos Mosqueira

Por haber apostado siempre por mí, por estar en las buenas y en las malas, por orientarme en lo largo de toda mi carrera profesional y formar parte de este proyecto como mi asesor.

A mis asesores

Doctor Frank Escobedo Bailón, Mg. Teodoro Diaz Leyva, Ing. Cesar Ramos Mosqueira; por todo su apoyo y asesoramiento en este proyecto de tesis y no solo asesores sino que también por formar parte de mi formación profesional como catedráticos y guías para ser un profesional competitivo.

Son muchas las personas que han formado parte de mi vida profesional a las cuales agradezco por brindarme su amistad, consejos, apoyo, ánimo y compañía en los momentos más difíciles de mi vida. Algunas están aquí conmigo y otras en mis recuerdos y en mi corazón, sin importar en donde estén quiero darles las gracias por formar parte de mí, por todo lo que me han brindado y por todas sus bendiciones.

ÍNDICE

LISTADO DE FIGURAS

LISTADO DE TABLAS

INTRODUCCIÓN

En la actualidad, la industria del software necesita un cambio de paradigmas, los desarrolladores de sistemas de nivel trainer's y junior's deben analizar diversas posibilidades que les puede brindar el uso de una plataforma que les permita generar código fuente. Este es el caso del área de Sistemas de la empresa PROCAMPO S.A., que preocupados en la mejora de la eficiencia en sus áreas de TI, concentran su atención en el desarrollo de un prototipo, el mismo que de ser exitoso se procederá a difundirlo a nivel provincial como nacional. Se entiende el cambio de paradigma por cambiar la forma de pensar y al enfoque que se tiene respecto a la programación de sistemas, llevando a cabo una manera más simplificada para poder generar código fuente inteligente.

Existe una problemática en la etapa de desarrollo de sistemas, la cual muchos de los programadores de software tienen un enfoque basado en que todo proyecto se debe realizar utilizando un Framework, ya que va a facilitar y agilizar la parte de la programación, pero de ese perspectiva nace otro problema, que hace que el desarrollador se adecue al Framework lo cual debería ser todo lo contrario que la plataforma se adecue a las necesidades del sistema que se está implementando.

Otras de las problemáticas que encontramos en el desarrollo es el tiempo de retraso que se lleva a cabo al momento de realizar las capas de negocio del sistema al momento de realizar un mantenimiento (listar, registrar, modificar,

eliminar y buscar) general de una entidad, es decir crear las Entidades de Negocio, Acceso a Datos y la Regla de negocio de las clases implementando los cinco métodos generales de un mantenimiento.

Para resolver dicha problemática se implementó una aplicación winforms, la misma que mediante un software permitirá resolver la problemática de los desarrolladores que usualmente se presenta cada vez que se empieza a realizar un proyecto de software. Y también permitirá cambiar la forma de pensar y el paradigma de programación que muchos desarrolladores tienen como un concepto mental al momento de desarrollar un sistema.

CAPÍTULO I: PLANTEAMIENTO DEL PROBLEMA

1.1 Descripción de la Realidad Problemática

En el desarrollo de software existen cinco etapas principales que todo proyecto debe cumplir, como realizar el requerimiento, análisis, desarrollo, pruebas y por último la implementación del sistema. Muchos de los desarrolladores de software dentro de todo el proceso del proyecto realizan tareas esenciales como el diseño de prototipos, codificación y documentación para el desarrollo de aplicaciones ya sea de tipo escritorio, web, móvil, etc. Lo cual para realizar los prototipos del sistema el desarrollador tarda aproximadamente entre una semana, dependiendo la magnitud del proyecto, ya sea utilizando un software para crear el diseño del mismo. Por otro lado, en el desarrollo dentro de la empresa se realizan las capas de datos de 3 a n niveles que están definidas como las Entidades de Negocio, Acceso a Datos y la Regla del Negocio y dentro de ello existen cinco funcionalidades básicas de mantenimiento que tiene todo sistema como Listar, Registrar, Modificar, Eliminar y Buscar. Desarrollar dichos métodos llevaría a cabo un tiempo aproximadamente de un día de trabajo para una sola entidad o tabla de base de datos que equivale a 8 horas de desarrollo por cada

entidad, dependiendo de la cantidad de entidades que tenga dicho sistema el tiempo de desarrollo sería más complejo; y por último la documentación del diccionario de datos de las tablas de la base de datos para poder visualizar el tipo de dato, descripción, tamaño y campo de un atributo de una entidad de negocio.

La problemática de los desarrolladores es el tiempo que tardan en desarrollar el diseño de prototipos, crear el mantenimiento de una entidad de negocio y sobre todo la inconsistencia de códigos como estándares de desarrollo al momento de asignar nombres a los campos, tablas, tipos de datos. Por otro lado, utilizan herramientas como un Framework para desarrollar parte de las capas de datos, pero al utilizarlo se genera otro nuevo problema el cual sobrecarga el sistema, que trae consigo código que a veces ni se utiliza en el sistema y eso hace que los sistemas sean un poco lento y pierdan la performance. Eso implica que el desarrollador tendría que aprender a utilizar el Framework el cual limita a que se aprenda a manejar dicha herramienta lo cual debería ser todo lo contrario que el Framework se adecue a tus necesidades.

1.2 Justificación de la Investigación

Visto la problemática mencionada anteriormente es importante implementar el sistema ya que tiene un beneficio bastante productivo para el desarrollador pues que, con el proyecto realizado se podrá optimizar el tiempo de desarrollo de un software ya que se va a minimizar tiempos en la etapa de desarrollo de código

fuente ya sea en la parte de programación como para la creación de los procedimientos almacenados de una base de datos y en la creación de prototipos. Así mismo el sistema permitirá estandarizar las nomenclaturas al momento de asignar los nombres de las clases, métodos, campos y procedimientos almacenados para poder llevar un estándar de desarrollo de software único para el área de sistemas. De esta manera se pretende solucionar la problemática, implementando el sistema en todos los proyectos realizados por el área de TI.

1.3 Delimitación del Proyecto

- **Delimitación Temporal**

  El trabajo ha sido efectuado desde enero 2016 hasta julio 2016

- **Delimitación Espacial**

  Se llevará a cabo en la empresa PROCAMPO S.A.

- **Delimitación Conceptual**

  Un generador de código permite agilizar la etapa de desarrollo de software, pues el ahorro de tiempo, la eficiencia en la programación y sobre todo la estandarización de código son los pilares fundamentales para la construcción de un proyecto.

1.4 Formulación del Problema

1.4.1 Problema General

¿De qué manera el desarrollo de un sistema generador de código fuente Jomake-Code permite mejorar el proceso de desarrollo de software en el área de Sistemas de la empresa Procampo S.A.?

1.4.2 Problemas Específicos

- ¿De qué manera el desarrollo de un sistema Jomake-Code permite minimizar el tiempo de desarrollo de software?

- ¿De qué manera el desarrollo de un sistema Jomake-Code permite generar prototipos de interfaz gráfica para las operaciones del mantenimiento?

- ¿De qué manera el desarrollo de un sistema Jomake-Code permite estandarizar el desarrollo de software?

1.5 Objetivos

1.5.1 Objetivo General

Desarrollar un sistema generador de código fuente Jomake-Code para mejorar el proceso de desarrollo de software en el área de Sistemas de la empresa Procampo S.A.

1.5.2 Objetivos Específicos

- Desarrollar un sistema Jomake-Code para minimizar el tiempo de desarrollo de software.

- Desarrollar un sistema Jomake-Code que permita generar prototipos de interfaz para las operaciones de mantenimiento.

- Desarrollar un sistema Jomake-Code para estandarizar el desarrollo de software.

CAPÍTULO II: MARCO TEÓRICO

2.1 Antecedentes de la Investigación

- Antecedente 01:

Título: Análisis, diseño y desarrollo de un generador de código fuente para gestión de información de MySQL, SQL Server y Access para los lenguajes Java, PHP y ASP.

Autor: Chávez Reina, Eduardo René

El presente proyecto de tesis propone realizar un generador de código fuente para varios lenguajes de programación, el cual controlará y gestionará distintas bases de datos, con el fin de ayudar al programador a realizar programas de una manera ágil y eficiente. Para realizar la tesis se ha optado por la metodología de desarrollo ¿Extremme Programming¿ (XP o Programación Extrema, en español) propuesta por Kent Beck, la cual es una alternativa ideal para desarrollo de software de pequeña y mediana complejidad, ya que omite diagramas y calendarios, pocas veces reales, que se deben realizar con otras metodologías de desarrollo de software. La programación extrema se basa en pequeños prototipos funcionales, los

cuales se van siguiendo a partir de historias de usuarios analizadas al inicio del proyecto y son entregados en períodos de tiempo relativamente cortos. Esta metodología es ideal para el desarrollo del presente tema ya que ayuda al programador a dividir en partes cada uno de los módulos que compondrán el sistema y las plantillas que se deban generar. Investigar uno a uno los lenguajes de programación y bases de datos propuestos y, para cada prueba, tener un producto totalmente terminado (Chávez Reina, Eduardo René, 2012).

- Antecedente 02:

Título: Generador de código fuente COBOL

Autor: Omar Isidro Pineda Mancilla

El Las grandes empresas siguen utilizando grandes equipos de cómputo (mainframes) por su gran capacidad de procesamiento. Pueden tener conectados a miles de usuarios al mismo tiempo, manejan una gran cantidad de transacciones instantáneamente, pueden administrar una gran capacidad de almacenamiento y hacerla expandible de acuerdo a las necesidades del negocio. En esta plataforma de procesamiento, el lenguaje COBOL es uno de los más favorecidos y muchas de las aplicaciones desarrolladas en dicho lenguaje son de naturaleza crítica o se trasforman en críticas para la operación del negocio, siendo una de sus principales características estar destinadas a operar las 24 horas del día. El volumen

de código existente en COBOL en el mundo entero se estima en millones de líneas y las aplicaciones siguen en mejora y en crecimiento continuos. Las aplicaciones desarrolladas en este lenguaje pueden ser categorizadas en: Procesos batch (ejecución en fecha y momento determinado). Transacciones on-line (de manera interactiva entre humanos u otras aplicaciones.).

La problemática que tienen las empresas son:

- Canibalización de código
- Construcción manual
- Poco tiempo para documentar.
- Código con baja calidad
- Nombre de sub-rutinas no descriptivas
- Falta de documentación

Para dar solución a la problemática se desarrolló un software que permita:

- Acceso a las tablas (CRUD) y cursores
- Incorporar reglas de negocio
- Vista previa
- Comentarios
- Conexión a Mainframe
- Documentación
- Muestra de selección de tablas y campos

Gracias a la implementación del software se pudo lograr mejorar el proceso de desarrollo de software en la empresa (Omar Pineda, 2013)

- **Antecedente 03:**

Título: Generador de código MyWay

Autor: José Carlos Temprado Morales

MyWay es un generador realizado en C#, capaz de facilitar tu tarea dándote el trabajo de toda una mañana o incluso días, en segundos, no solo para proyectos en C# sino que también Visual Basic .NET, e incluso la posibilidad de crear tus propias plantillas para así trabajan en otros lenguajes de programación. También existen distintas posibilidad de conexión a varias Bases de Datos (José Temprado, 2010).

- **Antecedente 04:**

Título: Generador de código para tu capa de datos, negocio y servicios Restful

Autor: Israel Romero Ponce

Casi todo sistema que requiere datos persistentes requiere de una base de datos, usando la tecnología de .Net tenemos varias formas de acceder y manipular esos datos.

Pero siempre es casi lo mismo cuando se empieza un sistema y muchos desarrolladores buscamos la manera más fácil de hacer la capa de acceso a datos con un mapeo tipo objeto relacional. El problema en ocasiones es que no tenemos el control total de algunas situaciones o nos cuesta mucho trabajo entender la API del que usemos.

Por eso les dejo el siguiente ORM que genera el código de nuestras capas de Domain, Data Access, Implementation, Business, Servicios Restful en C# y los Stored Procedures para realizar el CRUD. Lo mejor es que nos brinda el código para poder modificar la estructura que nos genera a nuestro gusto pero de un principio ya nos genera lo suficiente para empezar rápidamente a atacar el negocio y la interfaces de usuario (Israel Romero, 2014).

- **Antecedente 05:**

Título: CodeSmith Generator

Autor: Blake Niemyjski

CodeSmith Generator es una herramienta de desarrollo de software para ayudarle a hacer su trabajo más rápido. Técnicamente hablando, es un generador de código fuente basado en plantillas que automatiza la creación del código fuente de la aplicación común para cualquier idioma (C #, Java, VB, PHP, ASP.NET, SQL, etc.). CodeSmith generador incluye muchas plantillas útiles, así como conjuntos completos de plantillas para generar arquitecturas probadas. Usted puede modificar fácilmente cualquier plantilla o escribir su propia para generar el código exactamente la forma que desee (Blake Niemyjski, 2015).

2.2 Bases Teóricas

2.2.1. Framework

Un Framework es un entorno o ambiente de trabajo para desarrollo; dependiendo del lenguaje normalmente integra componentes que facilitan el desarrollo de aplicaciones como el soporte de programa, bibliotecas, plantillas y más.

En mi opinión como freelance lo más importante de todo este entorno de trabajo es que contiene máquinas virtuales, compiladores, bibliotecas de administración de recursos en tiempo de ejecución y especificaciones de lenguajes, haciendo nuestro trabajo más eficiente y recursivo.

La arquitectura más utilizada en casi todos los frameworks es conocida como MVC (Controlador, Modelo, Vista), esta arquitectura divide el desarrollo en tres grandes partes:

- Modelo: Son los datos de la aplicación y su reglamentación.
- Vista: Es la presentación de los datos.
- Controlador: Procesa las peticiones de los usuarios y controla el flujo de ejecución del sistema. (Cristina M., 2014).

2.2.2. NET Framework

Microsoft .NET Framework 4.0 es la última plataforma de desarrollo Microsoft que permite crear todo tipo de aplicaciones desde aplicaciones de consola para programas en lotes o batch, aplicaciones Windows para Cliente/Servidor, aplicaciones Web parta Internet o Intranet, Servicios Web para inter operar con otras plataformas, Servicios Windows para ejecutar tareas en segundo plano, y otros tipos de aplicaciones.

También nos permite crear Librerías o Bibliotecas de clases reusables, ya sea a nivel de código, de controles, de servicios, de flujos, etc., las cuales pueden utilizarse en diferentes tipos de aplicaciones.

A partir de la versión 3.0 se incluyó en el .NET Framework el poder crear aplicaciones de mejor presentación visual con WPF (Windows Presentations Foundation), también se unifico las aplicaciones distribuidas que antes se implementaban mediante Web Services, COM+ y NET Remoting en un solo conjunto de componentes llamado WCF (Windows Communication Foundation) y también se dio la posibilidad de crear aplicaciones de flujo de trabajo o colaboración con WWF (Windows Workflow Foundation).

Para realizar el desarrollo de aplicaciones o librerías en .NET el desarrollador puede elegir entre muchos Lenguajes .NET algunos de Microsoft como C#, J#, Visual Basic, Visual C++ y otros de diferentes proveedores: como ADA, APL, ASML, BETA, BF, C, Clarion#, COBOL, Cobra, CULE, E#, Eiffel, Flash, Forth, Fortran, G#, Jaskel, JavaScript, LISP, LOGO, Mercury, Modula 2, Oberon, Pascal, Perl, PHP, Prolog, Python, RPG, Ruby, Scala, Scheme, Smaltalk, etc.

Esta última característica de los lenguajes .NET es una diferencia fundamental con respecto a JAVA en donde el desarrollador solo programa usando un lenguaje (Java), en cambio con .NET Framework, podemos elegir entre muchos lenguajes, esta elección se hará de acuerdo a la experiencia que tenga la persona, en este libro vamos a usar el lenguaje Basic, es decir Visual Basic .NET.

Visual Basic .NET es recomendable para las personas que se inician en el desarrollo de software y no han visto ningún lenguaje, pero también para aquellos desarrolladores con experiencia que vienen de Visual Basic 3, 4, 5 o 6; de Foxpro, Visual Foxpro o Power Builder; ya que tiene características similares a estos, como el Modelo Conducido por Eventos (Event Driven Model o MDE), además de ser Orientado a Objetos (OOP), entre otras características modernas (Luis Dueñas, 2013).

2.2.3. Biblioteca de Clase de NET Framework

La BCL es una colección de más de 5000 tipos orientados a objetos distribuidos en más de 80000 miembros, los tipos se clasifican en:

- Tipos por Valor: Se copia el valor a la variable
- Tipos Simples del .Net Framework: char, boolean, byte, int, etc.
- Enumeraciones.
- Estructuras.
- Tipos por referencia
- Tipos complejos de .NET Framework: String, StringBuilder, DataSet, FileStream, StreamReader, StreamWriter, etc.
- Clases.

Cada tipo contiene elementos o miembros que pueden ser:

- Construtores y Destructores
- Propiedades, Metodos y Eventos
- Campos (variables públicas)
- Delegados (punteros a funciones)
- Enumeraciones
- Operadores

A su vez los tipos están organizados físicamente en Librerías (Assemblies) y lógicamente en Espacios de Nombres (Namespace). Por ejemplo los tipos simples se encuentran en la librería

System.Core.dll, los tipos principales en System.dll, el formulario y los controles Windows Forms en System. Windows.Forms.dll, etc.

La BCL tiene más de 255 espacios de nombres principales que agrupan a los tipos por categorías de uso, la mayoría inician con System y unos cuantos con Microsoft (Luis Dueñas, 2013).

2.2.4. Visual Studio NET

Visual Studio .NET es la herramienta de desarrollo Microsoft que utiliza la plataforma de desarrollo .NET Framework, para crear rápidamente aplicaciones .NET de todo tipo. Podría crear aplicaciones simples usando el bloc de notas y un compilador .NET, pero si la aplicación es compleja y tiene muchas pantallas demoraría demasiado, el Visual Studio le permite simplificar todo el desarrollo de desarrollo e inclusive las pruebas (Luis Dueñas, 2013).

2.2.5. Visual Basic y C#

Visual Basic y C# son lenguajes de programación diseñados para crear muy diversas aplicaciones que se ejecutan en .NET Framework. Se trata de lenguajes eficaces, que presentan seguridad de tipos y están orientados a objetos. Permiten a los desarrolladores crear aplicaciones Windows, web y móviles (Microsoft Developert, 2016).

2.2.6. SQL Server

Es un sistema de administración y análisis de bases de datos relacionales de Microsoft para soluciones de comercio electrónico, línea de negocio y almacenamiento de datos.

Microsoft SQL Server 2014 se basa en las funciones críticas ofrecidas en la versión anterior, proporcionando un rendimiento, una disponibilidad y una facilidad de uso innovadores para las aplicaciones más importantes. Microsoft SQL Server 2014 ofrece nuevas capacidades en memoria en la base de datos principal para el procesamiento de transacciones en línea (OLTP) y el almacenamiento de datos, que complementan nuestras capacidades de almacenamiento de datos en memoria y BI existentes para lograr la solución de base de datos en memoria más completa del mercado.

SQL Server 2014 también proporciona nuevas soluciones de copia de seguridad y de recuperación ante desastres, así como de arquitectura híbrida con Windows Azure, lo que permite a los clientes utilizar sus actuales conocimientos con características locales que aprovechan los centros de datos globales de Microsoft. Además, SQL Server 2014 aprovecha las nuevas capacidades de Windows Server 2012 y Windows Server 2012 R2 para ofrecer una escalabilidad sin

parangón a las aplicaciones de base de datos en un entorno físico o virtual (Microsoft Developert, 2014).

2.2.7. HyperText Markup Language (HTML)

Es el lenguaje básico de casi todo el contenido web. La mayor parte de lo que ves en la pantalla de tu navegador está escrita, fundamentalmente, usando HTML. Específicamente, HTML es el lenguaje con el que se escribe la estructura y la semántica del contenido de un documento web. El contenido dentro de una página web es etiquetado con elementos HTML como <img>, <title>, <p>, , y así sucesivamente.

HTML es un estándar internacional con especificaciones que son reguladas por el World Wide Web Consortium y el WHATWG. Es considerado un "estándar viviente" y está, técnicamente, siempre bajo construcción. La versión actual de la especificación HTML se conoce como HTML5.

HTML5 es la última versión de HTML y XHTML. El estándar HTML define un solo lenguaje que puede ser escrito usando la sintaxis de HTML, pero también usando la sintaxis más estricta de XML, y también se ocupa de las necesidades de las aplicaciones Web.

HTML5 no describe el estilo y formato del contenido, solo el propio contenido y su significado. El estilo y formato es definido y controlado usando Cascading Style Sheets (CSS) (Mozilla Developer, 2016).

2.2.8. Metodología de Software - Modelo Cascada

En esta fase se analizan las necesidades de los usuarios finales del software para determinar qué objetivos debe cubrir. De esta fase surge una memoria llamada SRD (documento de especificación de requisitos), que contiene la especificación completa de lo que debe hacer el sistema sin entrar en detalles internos.

Es importante señalar que en esta etapa se debe consensuar todo lo que se requiere del sistema y será aquello lo que seguirá en las siguientes etapas, no pudiéndose requerir nuevos resultados a mitad del proceso de elaboración del software.

- **Análisis de Requisitos**

  En esta fase se analizan las necesidades de los usuarios finales del software para determinar qué objetivos debe cubrir. De esta fase surge una memoria llamada SRD (documento de especificación de requisitos), que contiene la especificación

completa de lo que debe hacer el sistema sin entrar en detalles internos.

Es importante señalar que en esta etapa se debe consensuar todo lo que se requiere del sistema y será aquello lo que seguirá en las siguientes etapas, no pudiéndose requerir nuevos resultados a mitad del proceso de elaboración del software.

- **Diseño del Sistema**

Descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD (Documento de Diseño del Software), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras.

Es conveniente distinguir entre diseño de alto nivel o arquitectónico y diseño detallado. El primero de ellos tiene como objetivo definir la estructura de la solución (una vez que la fase de análisis ha descrito el problema) identificando grandes módulos (conjuntos de funciones que van a estar asociadas) y sus relaciones. Con ello se define la arquitectura de la solución elegida.

El segundo define los algoritmos empleados y la organización del código para comenzar la implementación.

- **Codificación**

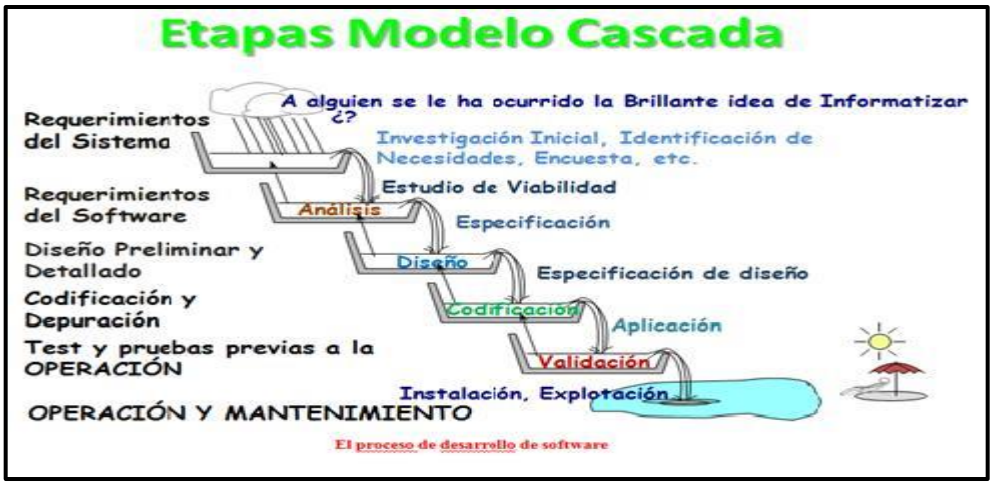
  Es la fase en donde se implementa el código fuente, haciendo uso de prototipos así como de pruebas y ensayos para corregir errores.

- **Verificación**

  Es la fase en donde el usuario final ejecuta el sistema, para ello el o los programadores ya realizaron exhaustivas pruebas para comprobar que el sistema no falle. (Isa Arteta, 2013).

Fases de la Metodología CASCADA

Figura 01: Estructura del modelo cascada



Fuente: Modelo Cascada (Isa Arteta, 2013)

2.2.9. Arquitectura 3 a N Niveles

Dividir un software en varias partes lógicas, ya sean módulos, paquetes o capas, ofrece la posibilidad de comprender fácilmente su filosofía y distribuir las tareas que ejecuta. Por ello la comunidad del software desarrolló la noción de una arquitectura de varios niveles y entre las más difundidas se encuentra la arquitectura de tres capas.

- **Capa de Presentación**

  Consiste en una interfaz gráfica que reúne los aspectos de software enfocados a la interacción con los diferentes tipos de usuarios. Es decir, incluye el manejo y aspecto de las ventanas, la autentificación, el formato de los reportes, menús, gráficos y demás elementos multimedia.

- **Capa de Negocio**

  Reúne los aspectos de software que automatizan los procesos de negocio. Conocida también como capa de la Lógica de la Aplicación. Recibe la entrada de la capa anterior, interactúa con los servicios de datos para ejecutar las operaciones y envía el resultado procesado a la capa de presentación.

- **Capa de Datos**

Contiene los datos necesarios para la aplicación. Es la encargada de almacenarlos, recuperarlos y mantener su integridad. Estos datos consisten en cualquier fuente de información, incluido una base de datos de empresa como Oracle o MySQL, SQLServer, etc. Un conjunto de documentos XML o incluso un servicio de directorio como LDAP. Además del tradicional mecanismo de almacenamiento relacional de base de datos, existen muchas fuentes diferentes de datos de empresa a las que pueden acceder las aplicaciones.

La separación entre la lógica de la aplicación y la interfaz de usuario ofrece mayor flexibilidad al diseño de la misma. De manera que los modelos de N capas están encaminados a maximizar aspectos importantes dentro de las aplicaciones, su autonomía, confiabilidad, disponibilidad, escalabilidad e interoperabilidad (Yanirys Montes, Yuliesky Brito, 2012).

2.2.10. Procedimiento Almacenado

Un procedimiento almacenado en SQL Server es un grupo de una o varias instrucciones Transact-SQL o una referencia a un método Framework Microsoft NET runtime lenguaje común (CLR).

Procedimientos asemejan construcciones en otros lenguajes de programación, ya que pueden:

- Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al programa de llamada.
- Contener instrucciones de programación que realicen operaciones en la base de datos. Estos incluyen llamar a otros procedimientos.
- Devolver un valor de estado a un programa de llamada para indicar el éxito o el fracaso (y el motivo de fallo) (Microsoft Developer, 2016).

2.3 Marco Conceptual

2.3.1. HTML

Lenguaje de Marcado de Hipertexto, se utiliza para crear las páginas web. Este lenguaje indica a los navegadores cómo deben mostrar el contenido de una página web.

2.3.2. Trainer's

Desarrolladores de sistema con conocimientos a nivel básico.

2.3.3. Junior’s

Desarrolladores de sistemas con conocimientos a nivel intermedio.

2.3.4. VB

Lenguaje de programación Visual Basic para .NET para crear sistemas.

2.3.5. C# o CSharp

Lenguaje de Programación para .NET para crear sistemas.

2.3.6. Entidad

Es cualquier tipo de objeto que recoge información, ejemplo: persona, animal, carro, etc.

2.3.7. Prototipo

Es la primera versión o modelo del producto, en que se han incorporado algunas características del producto final.

2.3.8. Software

Representa toda la parte inmaterial o intangible que hace funcionar a un ordenador para que realice una serie de tareas específicas.

2.3.9. Desarrollo

Es la parte de la creación del sistema.

2.3.10. Producción

Cuando el sistema se pone en marcha para que lo utilicen los usuarios.

2.3.11. Capas

Son módulos que contienen información del sistema que separan de la lógica de negocios de la lógica de diseño.

2.3.12. CS

Extensión de archivo para CSharp.

2.3.13. Cliente

Es el usuario que interactúa con el sistema.

2.3.14. Servidor

Es la parte interna del sistema el cual interactúa con la base de datos y los controles del mismo lenguaje.

2.3.15. Método

Es un proceso que efectúa una acción.

2.3.16. Evento

Son manejadores de controles al efectuar una acción estos se activan.

2.3.17. Performance

Proceso de determinación de la velocidad o la eficacia de un software.

CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA

Historial de Revisión

Tabla 01: Historial de versiones

Fecha	Versión	Descripción	Autor
07/01/2016	1.0	Desarrollo del alcance del proyecto	Kervy Ramos Z.
07/03/2016	1.1	Desarrollo de la primera versión	Kervy Ramos Z.
07/06/2016	1.2	Etapa de pruebas finales	Kervy Ramos Z.
07/07/2016	1.3	Culminación del proyecto	Kervy Ramos Z.

Fuente: Elaboración propia

3.1 Análisis del Sistema

3.1.1. Lista de Interesado – Stakeholders

Tabla 02: Lista de personas involucradas en el proyecto

Cargo	Rol	Nombres	Correo	Teléfono
Gerente de Administración y Finanzas	Sponsor del Cliente	Gissella Minaya Mendoza	gminaya@procampo.com.pe	975774363
Jefe de Sistemas	Jefe de Proyectos	Victor Carbajal Alcarraz	vcarbajal@procampo.com.pe	975774363
Asistente de Sistemas	Analista Desarrollador de Software	Kervy Ramos Zapana	kramos@procampo.com.pe	949189420

Fuente: Elaboración propia

3.1.2. Modelado de Requerimiento

Se refiere a todas aquellas tareas o actividades que se requiere que el sistema llegue hacer. Identificar atributos y relaciones entre requerimientos.

3.1.2.1. Requerimientos Funcionales

Tabla 03: Listado de requerimientos funcionales

NRO	REQUISITOS	CALIFICACIONES		
	FUNCIONALES	DIFICULTAD	PRIORIDAD	EXIGIBLE
1	El sistema deberá tener un menú de opciones para el manejo de prototipos.	3	1	SI
2	El sistema deberá contener dos sub menú de opciones generar HTML y crear prototipos dentro del menú prototipos.	3	1	SI
3	El sistema deberá tener un menú de opciones para el manejo de desarrollo.	3	1	SI
4	El sistema deberá contener dos sub menú de opciones librerías y base de datos dentro del menú desarrollo.	3	1	SI
5	El sistema deberá tener un menú de opciones para el manejo de documentación	3	1	SI
6	El sistema deberá contener un sub menú de opciones crear diccionario de datos dentro del menú documentación.	3	1	SI
7	El sistema debe permitir visualizar el detalle de los campos de cada tabla.	3	1	SI
8	El sistema debe mostrar los 100 primeros registros de cada tabla.	3	2	SI
9	El sistema debe permitir visualizar el detalle de los procedimientos almacenados.	3	1	SI
10	El sistema debe crear una carpeta de almacenamiento por cada base de datos consultada.	1	1	SI
11	El sistema debe contemplar los dos tipos de inicio de sesión para SqlServer, tanto como seguridad de Sql Server como para seguridad Integrada y/o de Windows.	2	1	SI
12	El sistema debe generar un reporte en formato Word donde muestre el diccionario de datos de las tablas de la base de datos.	4	1	SI
13	El sistema debe crear imágenes de prototipos en formato jpg	5	1	SI

Fuente: Elaboración propia

3.1.2.2. Requerimientos No Funcionales

Tabla 04: Listado de requerimientos no funcionales

NRO	DESCRIPCIÓN	TIPO
1	Se debe contar con una servidor de base de datos Sql Server	Hardware
2	Se debe contar con las licencias respectivas de la base de datos Sql Server	Software
3	El sistema deberá contar con accesos remotos (OpenVPN, FortiClient)	Software
4	El sistema debe permitir el acceso a multi usuario.	Eficiencia
5	El sistema deberá funcionar sobre la base de datos SqlServer.	Software
6	Toda funcionalidad del desarrollo del sistema debe responder al usuario en menos de 1 minuto de espera.	Eficiencia
7	El tiempo de aprendizaje del sistema por un usuario deberá ser menor a 30 minutos.	Usabilidad
8	El sistema debe estar en la capacidad de permitir en el futuro el desarrollo de nuevas funcionalidades, modificar o eliminar funcionalidades.	Escalabilidad
9	El sistema debe proporcionar mensajes de error que sean informativos y orientados a usuario final.	Usabilidad
10	La aplicación debe ser desarrollada para la platamorfa .NET CSharp (C#)	Software

Fuente: Elaboración Propia

3.1.3. Criterio de Calificación

Tabla 05: Nivel de dificultad

NIVEL DE DIFICULTAD	RANGO
Muy baja	1
Baja	2
Media	3
Alta	4
Muy alta	5

Fuente: Elaboración propia

Tabla 06: Nivel de prioridad

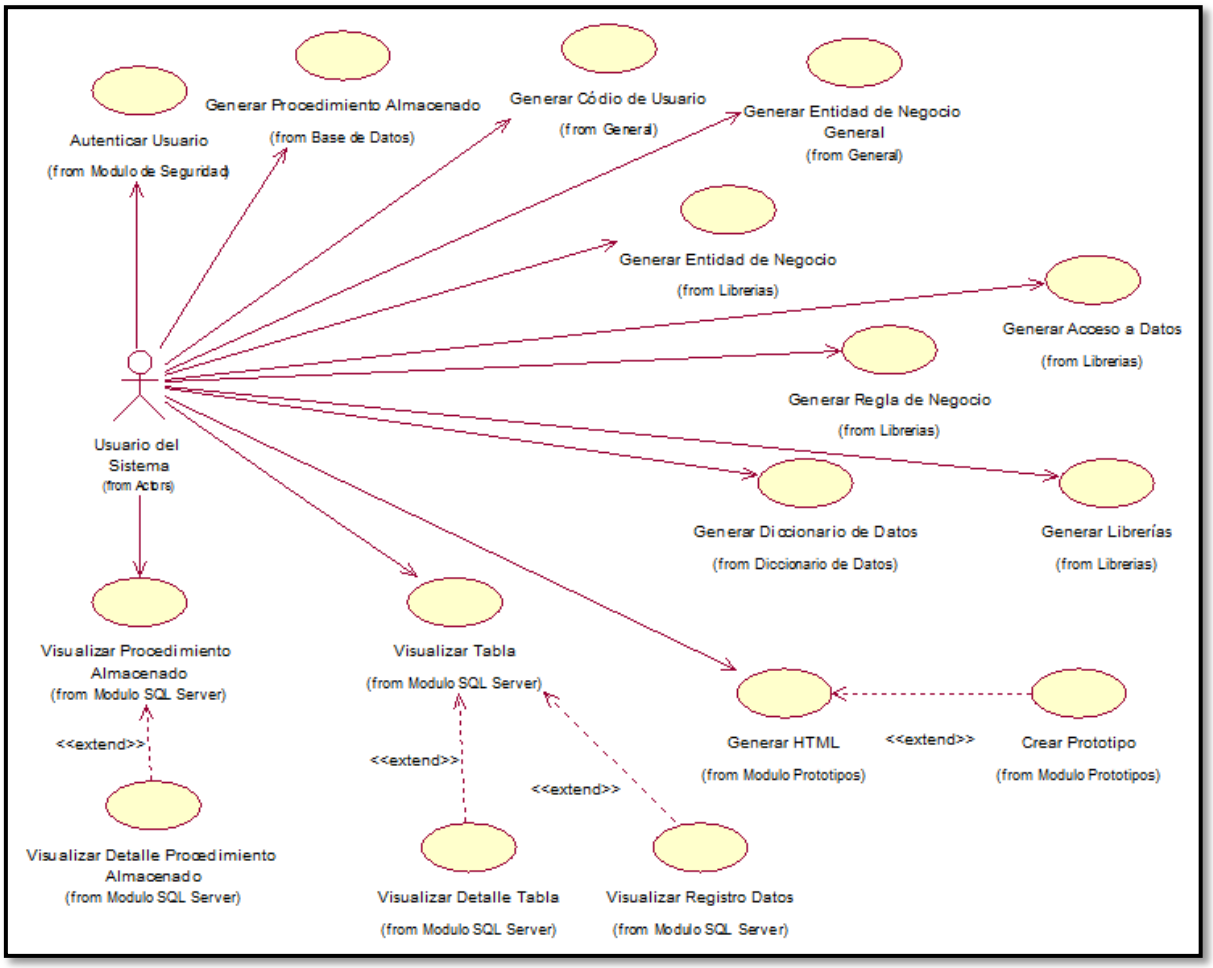
NIVEL DE PRIORIDAD	RANGO
Muy baja	5
Baja	4
Media	3
Alta	2
Muy alta	1

Fuente: Elaboración propia

3.2 Diseño del Sistema

3.2.1. Diagrama General de Caso de Uso del Sistema

Figura 02: Caso de uso general del sistema



Fuente: Elaboración propia

3.2.2. Especificación de Casos de Usos

Tabla 07: Descripción de casos de usos

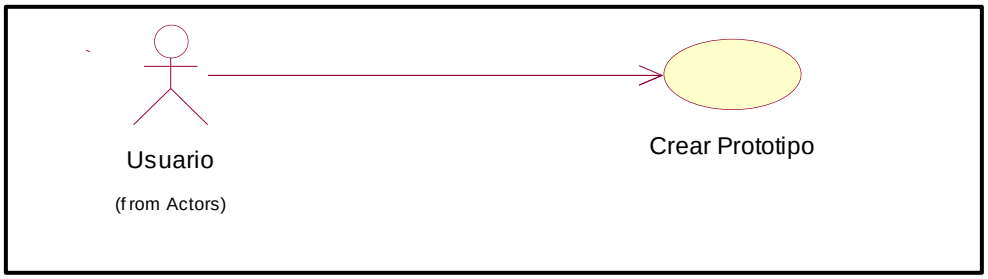
CASO DE USO	DESCRIPCIÓN
 Autenticar Usuario (from Modulo de Seguridad)	Este caso de uso permite a los usuarios iniciar sesión. Proceso que permite validar los datos ingresados por el usuario para poder acceder a la pantalla principal del sistema.
 Generar Procedimiento Almacenado (from Base de Datos)	Este caso de uso permite a los usuarios generar procedimientos almacenados de mantenimiento como Listar, Insertar, Actualizar, Eliminar y Buscar creando un archivo .Sql con el nombre de la tabla seleccionada.
 Generar Códio de Usuario (from General)	Este caso de uso permite a los usuarios generar la clase uc_Objeto la cual se almacenará en una librería que contendrá código fuente general ya sea para utilizar en el desarrollo de escritorio como para entorno web.
 Generar Entidad de Negocio General (from General)	Este caso de uso permite generar entidades de negocio generales ya que se pueden utilizar en proyectos con funcionalidades distintas ya sea para escritorio, web o móvil.
 Generar Entidad de Negocio (from Librerias)	Este caso de uso permite generar los atributos y propiedades de las clases de entidad de negocio para las tablas de la base de datos a trabajar.
 Generar Acceso a Datos (from Librerias)	Este caso de uso permite generar los métodos principales de mantenimiento a su vez sirve de puente de comunicación para acceder a la base de datos.
 Generar Regla de Negocio (from Librerias)	Permite generar los métodos principales de mantenimiento a su vez sirve de enlace para enviar datos a la capa de presentación.

 Generar Librerías (from Librerias)	Este proceso permite generar las tres capas de datos (Entidad de Negocio, Acceso a Datos y Regla de Negocio) de forma automática.
 Generar Diccionario de Datos (from Diccionario de Datos)	Este proceso permite crear un diccionario de datos de las tablas seleccionas de la base de datos.
 Generar HTML (from Modulo Prototipos)	Este proceso permite generar código HTML para la parte de diseño de prototipos.
 Crear Prototipo (from Modulo Prototipos)	Este proceso permite crear los prototipos de imágenes en formato .JPG
 Visualizar Tabla (from Modulo SQL Server)	Este proceso permite visualizar todas las tablas que se encuentran creadas en la base de datos.
 Visualizar Detalle Tabla (from Modulo SQL Server)	Este proceso permite mostrar al usuario el detalle de cada tabla (Nombre, campo, tamaño, descripción, datos nulos y campo con PK)
 Visualizar Registro Datos (from Modulo SQL Server)	Este proceso permite mostrar al usuario el detalle de los registros ingresados a por el sistema a la tabla en consulta.
 Visualizar Procedimiento Almacenado (from Modulo SQL Server)	Este proceso permite visualizar todos los procedimientos almacenados que se encuentran creados en la base de datos.
 Visualizar Detalle Procedimiento Almacenado (from Modulo SQL Server)	Este proceso permite al usuario visualizar el detalle del procedimiento almacenado que ya se encuentra creado en base de datos.

Fuente: Elaboración propia

3.2.2.1. Caso de Uso Autenticar Usuario

Figura 03: Diagrama caso de uso autenticar usuario



Fuente: Elaboración propia

3.2.2.2. Descripción Caso de Uso: Autenticar Usuario

Tabla 08: Detalle del caso de uso autenticar usuario

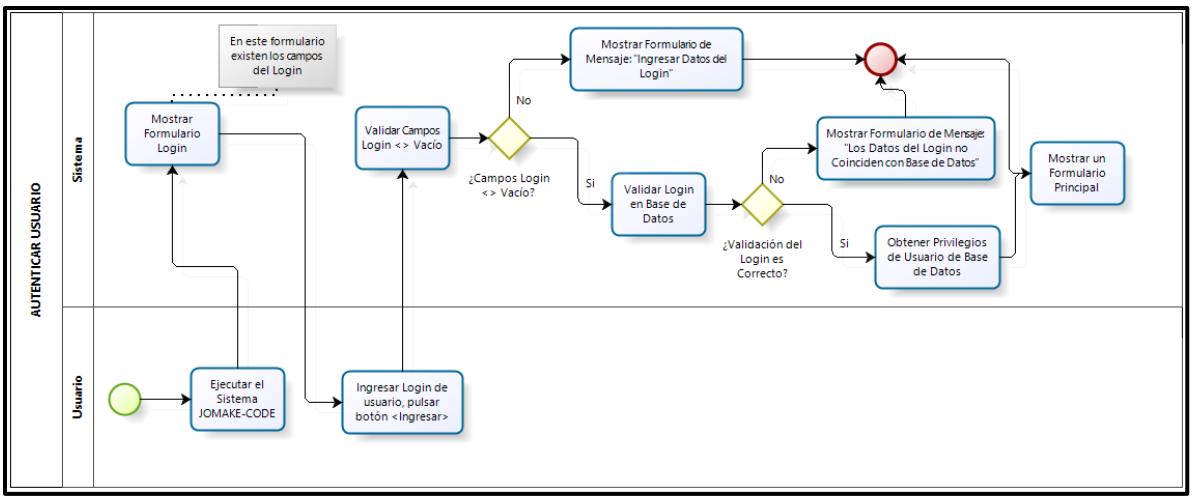
ID:	CU01
Caso de Uso:	Autenticar Usuario
Actor:	Usuario de Sistemas
Descripción:	Este caso de uso permite a los usuarios iniciar sesión. Proceso que permite validar los datos ingresados por el usuario para poder acceder a la pantalla principal del sistema.
Precondición:	El usuario debe tener una cuenta SqlServer para autenticar de forma seguridad de SqlServer y/o estar en el dominio de la empresa para una autenticación Integrada o de windows
Flujo de Eventos	
Flujo Principal	
1. El usuario ejecuta el sistema Jomake-Code. 2. El sistema muestra un formulario de Login con sus respectivos campos. 3. El usuario ingresa todos los datos del Login y pulsa el botón <Ingresar> 4. El sistema valida los campos que no existan datos en blanco. 5. Si no existen datos en blanco, el sistema valida los datos del usuario. Caso contrario el sistema muestra un formulario con el mensaje indicando que debe ingresar datos del Login. 6. Si es un usuario válido, obtiene los privilegios de usuario de base de datos. Caso contrario el sistema muestra un formulario con el mensaje indicando que los datos del Login no coinciden con la base de datos. 7. Finalmente el sistema muestra un formulario principal.	
Flujo Alternativo de Eventos	
Flujo Secundario	
No aplica	

Post-condición :	1. El usuario ingresa al menú principal del sistema. 2. Se asignan los privilegios pertinentes. 3. Se crea una carpeta principal con el nombre de la base de datos donde se va almacenar por defecto todos los archivos obtenidos en el sistema.

Fuente: Elaboración propia

3.2.2.3. Diagrama de actividades: Autenticar Usuario

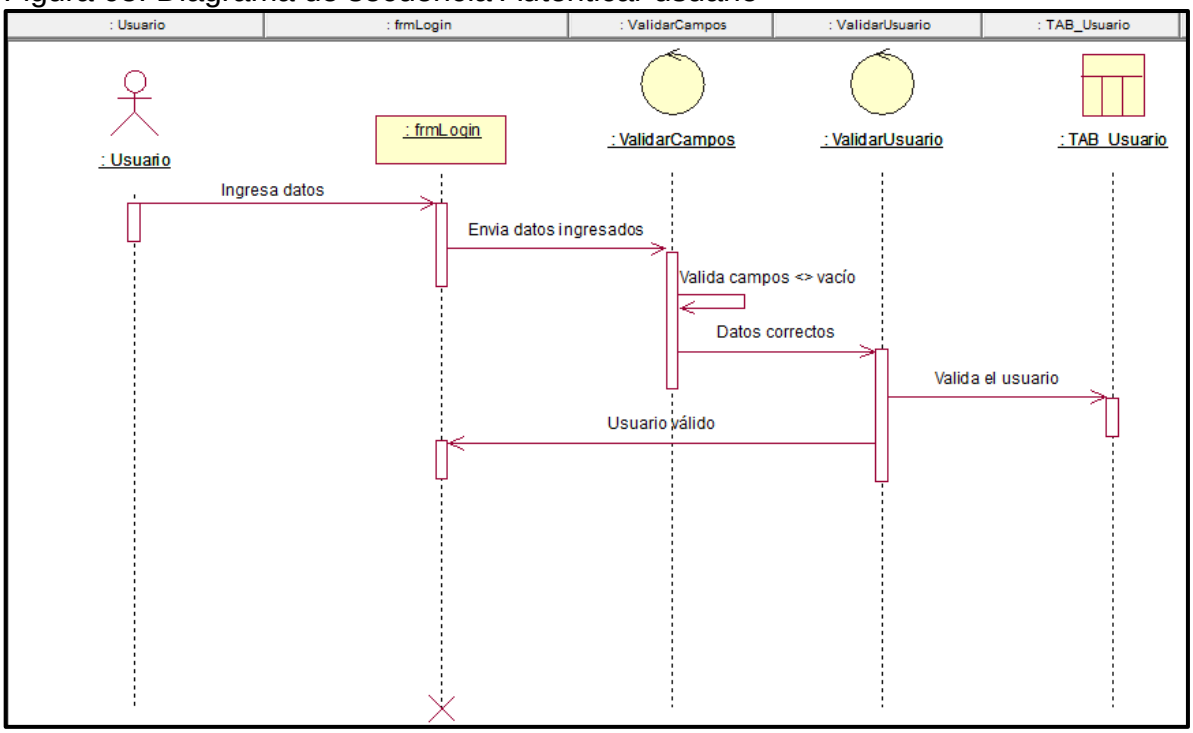
Figura 04: Detalla el diagrama de actividades autenticar usuario



Fuentes: Elaboración propia

3.2.2.4. Diagrama de Secuencias: Autenticar Usuario

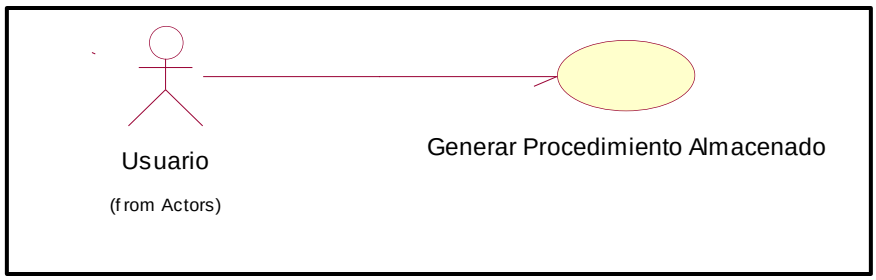
Figura 05: Diagrama de secuencia Autenticar usuario



Fuente: Elaboración propia

3.2.2.5. Caso de Uso Generar Procedimiento Almacenado

Figura 06: Diagrama caso de uso generar procedimiento almacenado



Fuente: Elaboración propia

3.2.2.6. Descripción Caso de Uso: Generar Procedimiento Almacenado

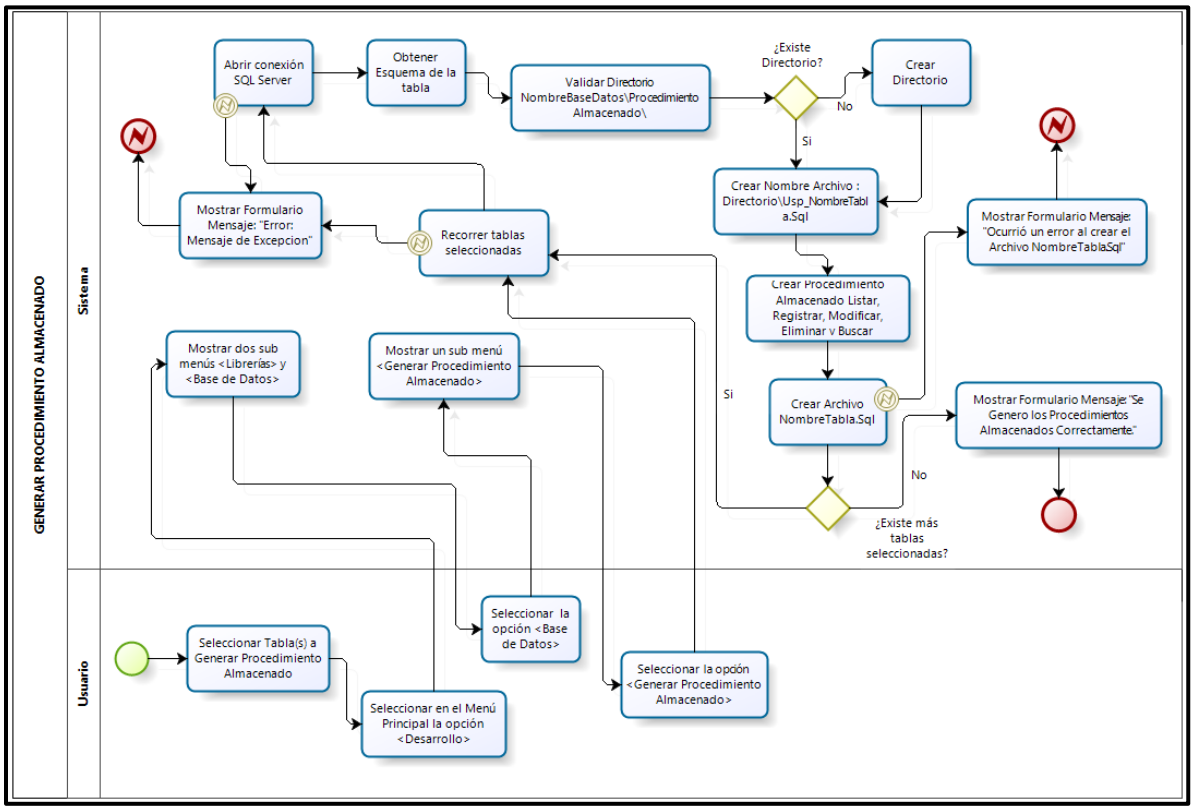
Tabla 09: Detalle del caso de uso generar procedimiento almacenado

ID:	CU02
Caso de Uso:	Generar Procedimiento Almacenado
Actor:	Usuario de Sistemas
Descripción:	Este caso de uso permite a los usuarios generar procedimientos almacenados de mantenimiento como Listar, Registrar, Modificar, Eliminar y Buscar creando un archivo .Sql con el nombre de la tabla seleccionada.
Precondición:	El usuario debe de estar autenticado al sistema.
Flujo de Eventos	
Flujo Principal	
1. El usuario selecciona las tablas a generar procedimiento almacenado. 2. Selecciona en el menú principal la opción <Desarrollo> 3. El sistema muestra dos sub menús <Librerías> y <Base de Datos> 4. El usuario selecciona la opción <Base de Datos> 5. El sistema muestra un sub menú <Generar Procedimiento Almacenado> 6. El usuario selecciona la opción <Generar Procedimiento Almacenado> 7. El sistema recorre las tablas seleccionadas. 8. Abre la conexión de SqlServer y obtiene el esquema de la tabla. 9. El sistema valida que exista el directorio NombreBaseDatos\Procedimiento Almacenado\ 10. Si el directorio no existe, el sistema crea el directorio. 11. Crear el nombre del archivo Directorio\Usp_NombreTabla.Sql 12. El sistema crea el procedimiento almacenado de mantenimiento Listar, Insertar, Actualizar, Eliminar y Buscar con el archivo NobreTabla.Sql 13. Si existen más tablas seleccionadas, se repiten los pasos 7 – 11 caso contrario el sistema muestra un formulario con el mensaje “Se generó los Procedimientos Almacenados Correctamente.”	
Flujo Alternativo de Eventos	
Flujo Secundario	
No aplica	
Post-condición :	1. Se crea una carpeta con el nombre de procedimiento almacenado dentro de la carpeta principal con el nombre de la base de datos donde se almacena todos los procedimientos almacenados.

Fuente: Elaboración propia

3.2.2.7. Diagrama de Actividades: Generar Procedimiento Almacenado

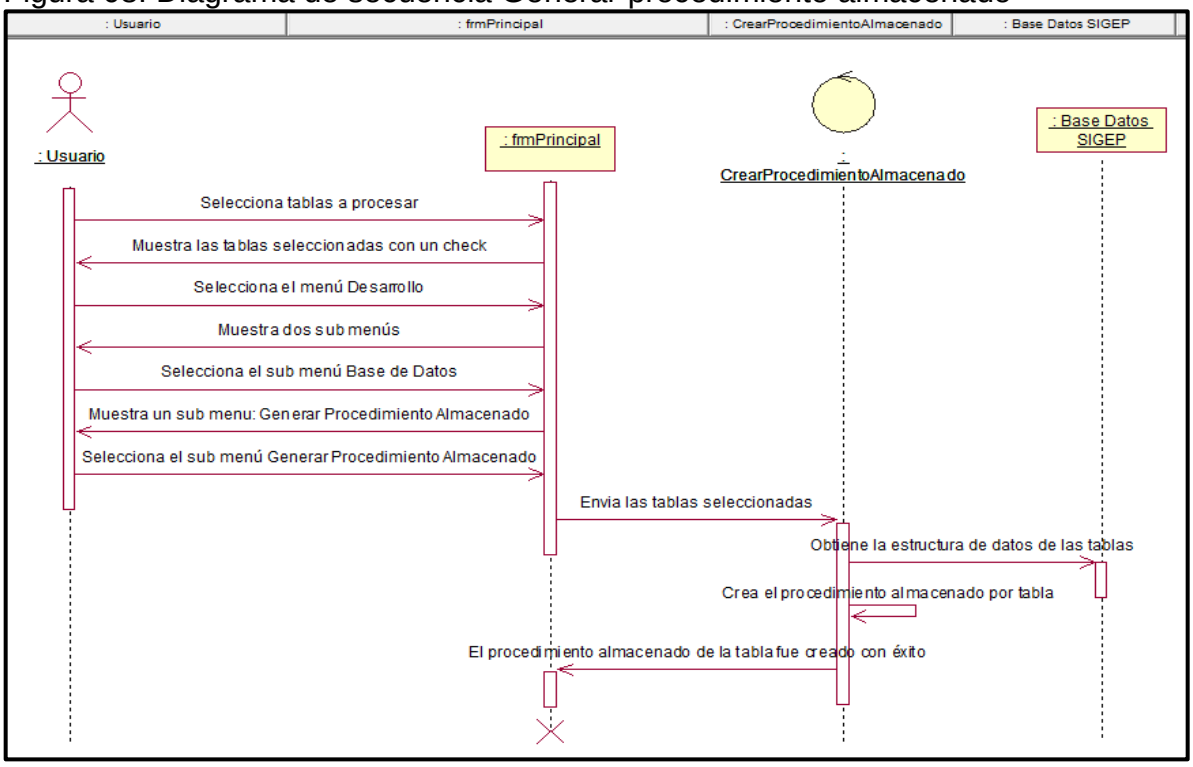
Figura 07: Detalle de diagrama de actividades generar procedimiento almacenado.



Fuente: Elaboración propia

3.2.2.8. Diagrama de Secuencias: Generar Procedimiento Almacenado

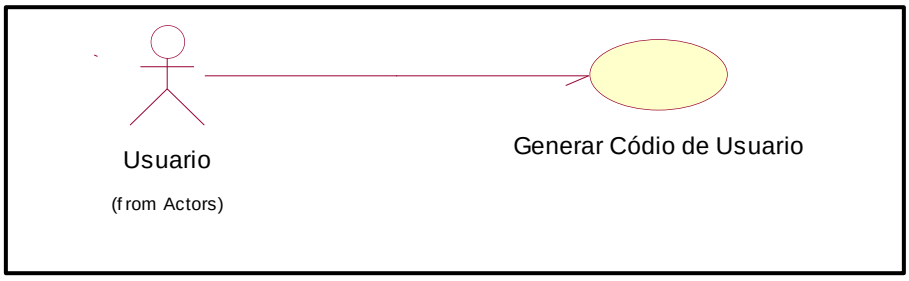
Figura 08: Diagrama de secuencia Generar procedimiento almacenado



Fuente: Elaboración propia

3.2.2.9. Caso de Uso Generar Código de Usuario

Figura 09: Diagrama caso de uso generar código de usuario



Fuente: Elaboración propia

3.2.2.10. Descripción Caso de Uso: Generar Código de Usuario

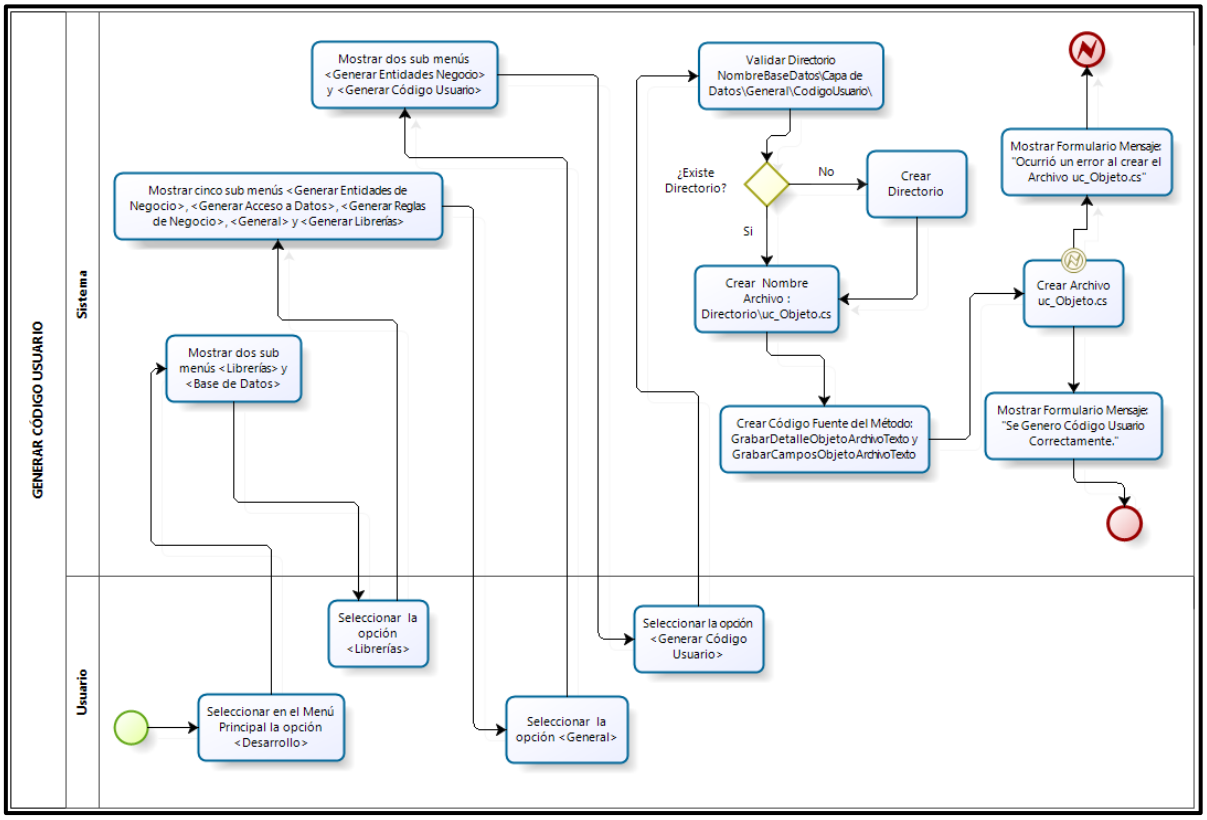
Tabla 10: Detalle de caso de uso generar código de usuario

ID:	CU03
Caso de Uso:	Generar Código de Usuario
Actor:	Usuario de Sistemas
Descripción:	Este caso de uso permite a los usuarios generar la clase uc_Objeto la cual se almacenará en una librería que contendrá código fuente general ya sea para utilizar en el desarrollo de escritorio como para entorno web.
Precondición:	El usuario debe de estar autenticado al sistema.
Flujo de Eventos	
Flujo Principal	
1. El usuario selecciona en el menú principal la opción <Desarrollo> 2. El sistema muestra dos sub menús <Librerías> y <Base de Datos> 3. El usuario selecciona la opción <Librerías> 4. El sistema muestra cinco sub menús <Generar Entidad de Negocio>, <Generar Acceso a Datos>, <Generar Reglas de Negocio>, <General> y <Generar Librerías> 5. El usuario selecciona la opción <General> 6. El sistema muestra dos sub menús <Generar Entidades Negocio> y <Generar Código Usuario> 7. El usuario selecciona la opción <Generar Código Usuario> 8. El sistema valida que exista el directorio NombreBaseDatos\Capa de Datos\General\CodigoUsuario\ 9. Si el directorio no existe, el sistema crea el directorio y luego crear el nombre del archivo Directorio\uc_Objeto.cs 10. El sistema crea el código fuente de los métodos: GrabarDetalleObjetoArchivoTexto y GrabarCamposObjetoArchivoTexto con el archivo uc_Objeto.cs 11. El sistema muestra un formulario con un mensaje de “Se Generó Código Usuario Correctamente.”	
Flujo Alternativo de Eventos	
Flujo Secundario: Error al crear Archivo	
1. Si ocurre un error al momento de crear el archivo uc_Objeto.cs, el sistema muestra un formulario con el mensaje “Ocurrió un error al crear el Archivo uc_Objeto.cs”	
Post-condición :	1. Se crea una carpeta con el nombre de CodigoUsuario donde se almacenará todas las clases genéricas para el desarrollo de .NET ya sea en el entorno escritorio como para web.

Fuente: Elaboración propia

3.2.2.11. Diagrama de Actividades: Generar Código de Usuario

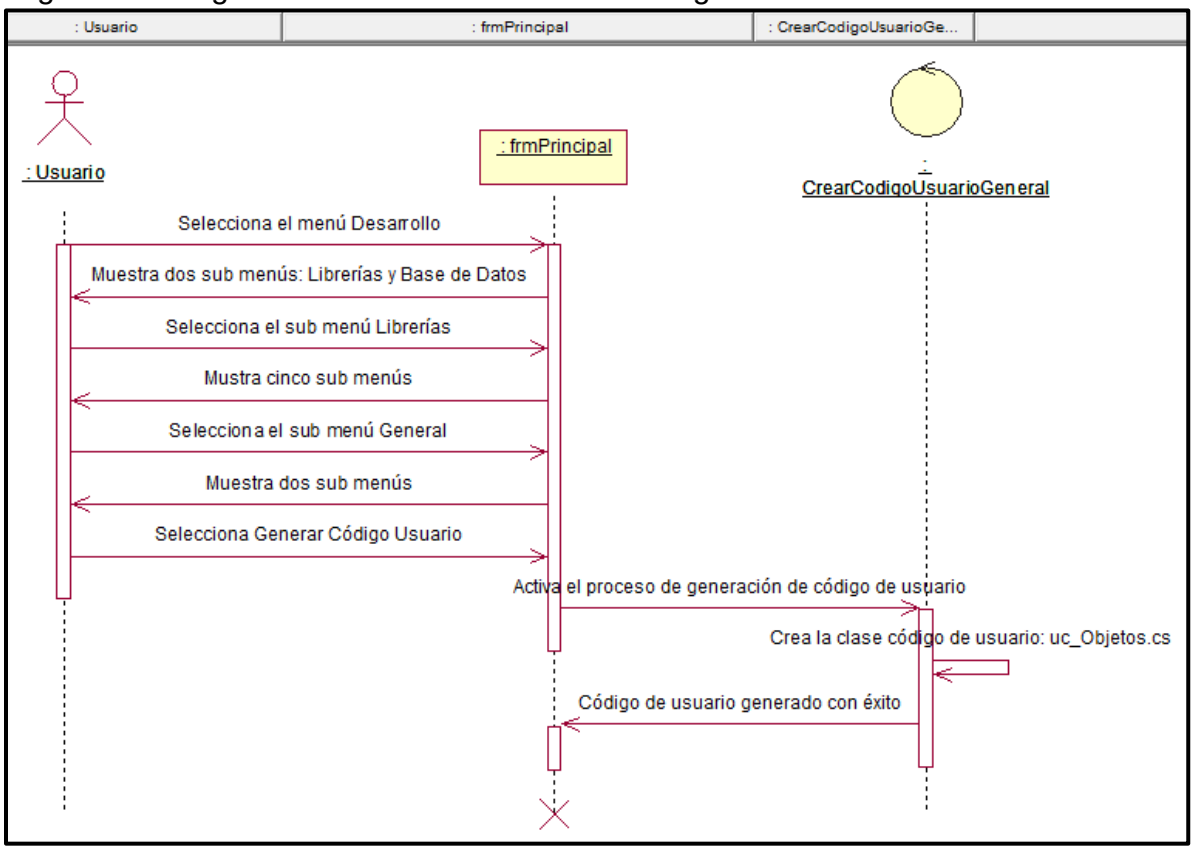
Figura 10: Detalle de diagrama de actividades generar código de usuario



Fuente: Elaboración propia

3.2.2.12. Diagrama de Secuencias: Generar Código de Usuario

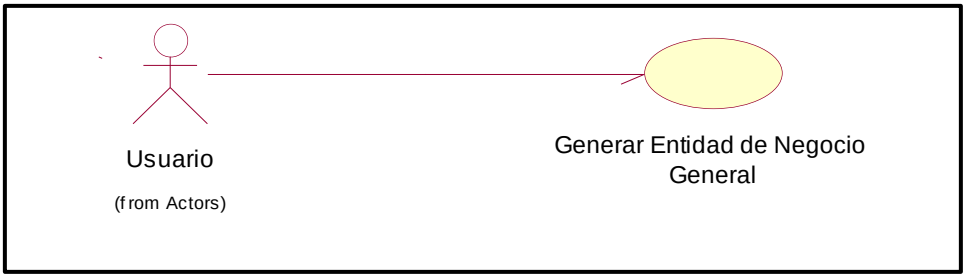
Figura 11: Diagrama de secuencia Generar código de usuario



Fuente: Elaboración propia

3.2.2.13. Caso de Uso Generar Entidad de Negocio General

Figura 12: Diagrama caso de uso generar entidad de negocio general



Fuente: Elaboración propia

3.2.2.14. Descripción Caso de Uso: Generar Entidad de Negocio General

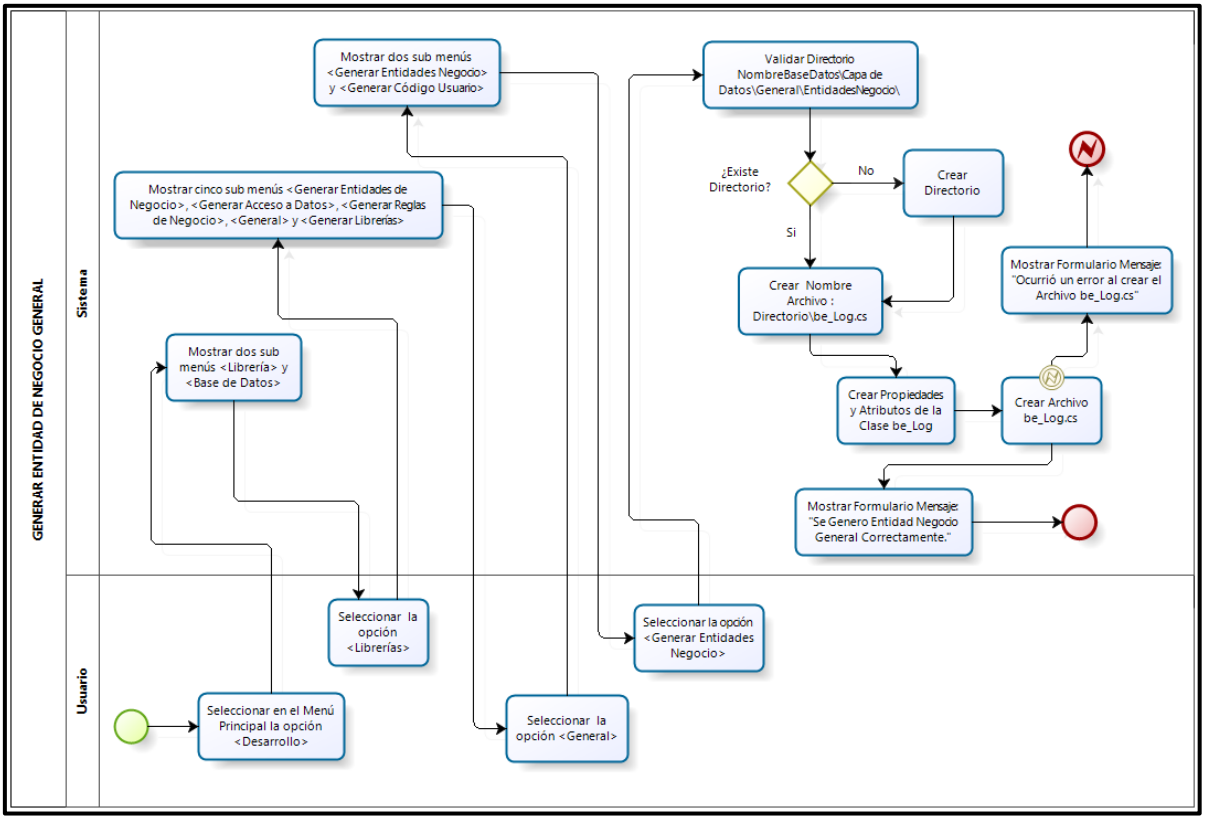
Tabla 11: Detalle de caso de uso generar entidad de negocio general

ID:	CU04
Caso de Uso:	Generar Entidad de Negocio General
Actor:	Usuario de Sistemas
Descripción:	Este caso de uso permite generar entidades de negocio generales ya que se pueden utilizar en proyectos con funcionalidades distintas ya sea para escritorio, web o móvil.
Precondición:	El usuario debe de estar autenticado al sistema.
Flujo de Eventos	
Flujo Principal	
1. El usuario selecciona en el Menú Principal la opción <Desarrollo>. 2. El sistema muestra dos sub menús <Librería> y <Base de Datos>. 3. El usuario selecciona la opción <Librerías>. 4. El sistema muestra cinco sub menús <Generar Entidades de Negocio>, <Generar Acceso a Datos>, <Generar Reglas de Negocio>, <General> y <Generar Librerías>. 5. El usuario selecciona la opción <General>. 6. El sistema muestrar dos sub menús <Generar Entidades Negocio> y <Generar Código Usuario>. 7. El usuario seleccionar la opción <Generar Entidades Negocio>. 8. El sistema valida si existe el Directorio NombreBaseDatos\Capa de Datos\General\EntidadesNegocio\. 9. Si el directorio no existe, entonces el sistema crea el directorio. 10. El sistema crea Nombre Archivo: Directorio\be_Log.cs. 11. Crea propiedades y atributos de la Clase be_Log. 12. Crea el archivo be_Log.cs 13. Si al crear el archivo existe un error, el sistema muestra un formulario con el mensaje “Ocurrió un error al crear el Archivo be_Log.cs”. Caso contrario el sistema muestra un formulario con el mensaje "Se Generó Entidad Negocio General Correctamente.".	
Flujo Alternativo de Eventos	
Flujo Secundario	
No aplica	
Post-condición :	1. Se crea una carpeta con el nombre de General/EntidadesNegocio donde se almacenará todas las entidades genéricas para el desarrollo de .NET ya sea en el entorno escritorio como para web.

Fuente: Elaboración propia

3.2.2.15. Diagrama de Actividades: Generar Entidad de Negocio General

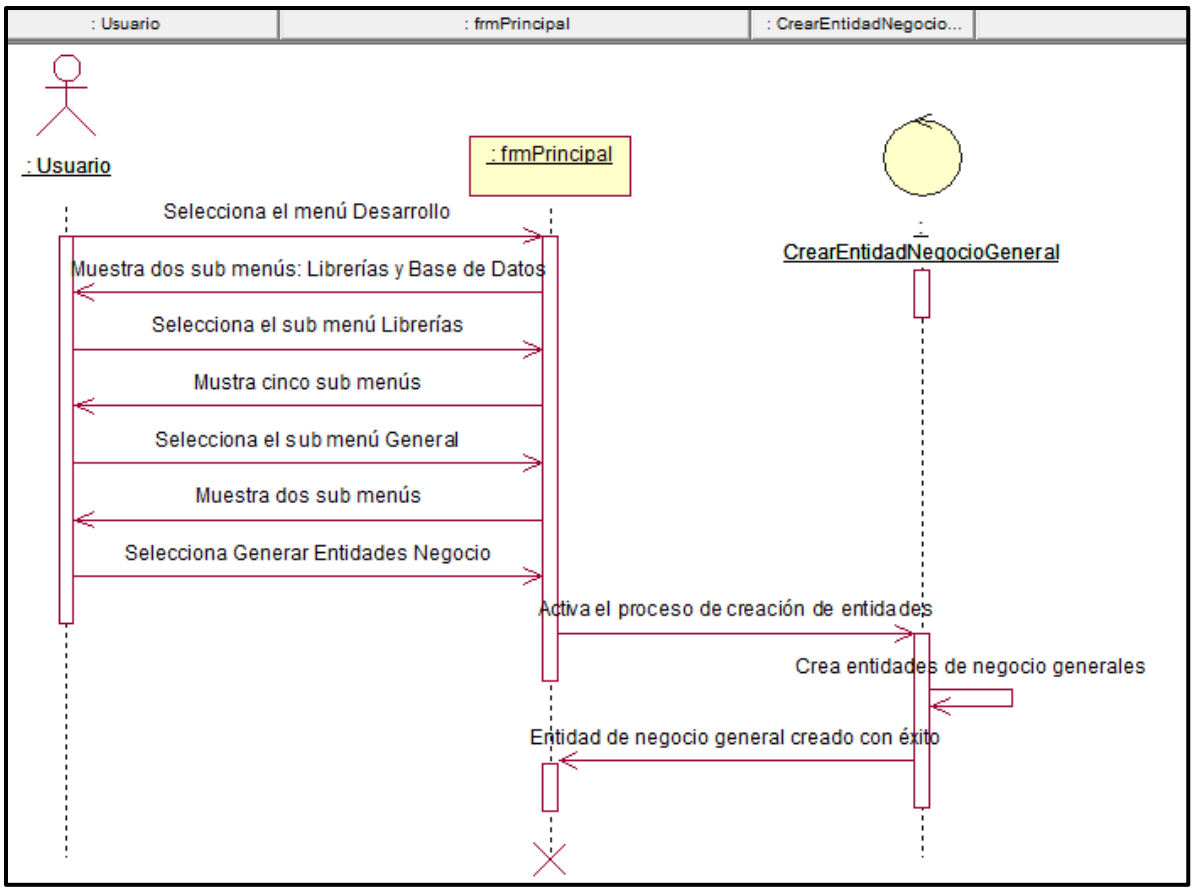
Figura 13: Detalle de diagrama de actividades generar entidad de negocio general



Fuente: Elaboración propia

3.2.2.16. Diagrama de Secuencias: Generar Entidad de Negocio General

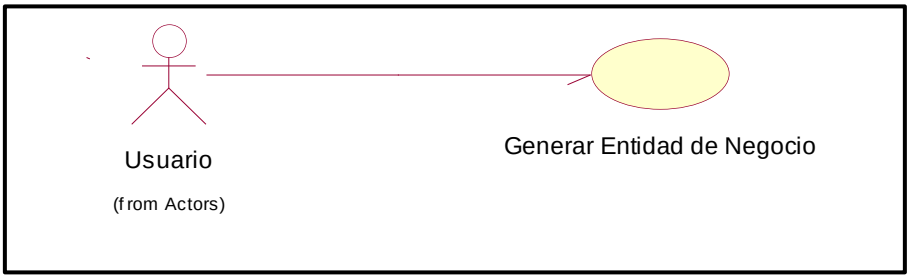
Figura 14: Diagrama de secuencia Generar entidad de negocio general



Fuente: Elaboración propia

3.2.2.17. Caso de Uso Generar Entidad de Negocio

Figura 15: Diagrama caso de uso generar entidad de negocio



Fuente: Elaboración propia

3.2.2.18. Detalle Caso de uso: Generar Entidad de Negocio

Tabla 12: Detalle de caso de uso generar entidad de negocio

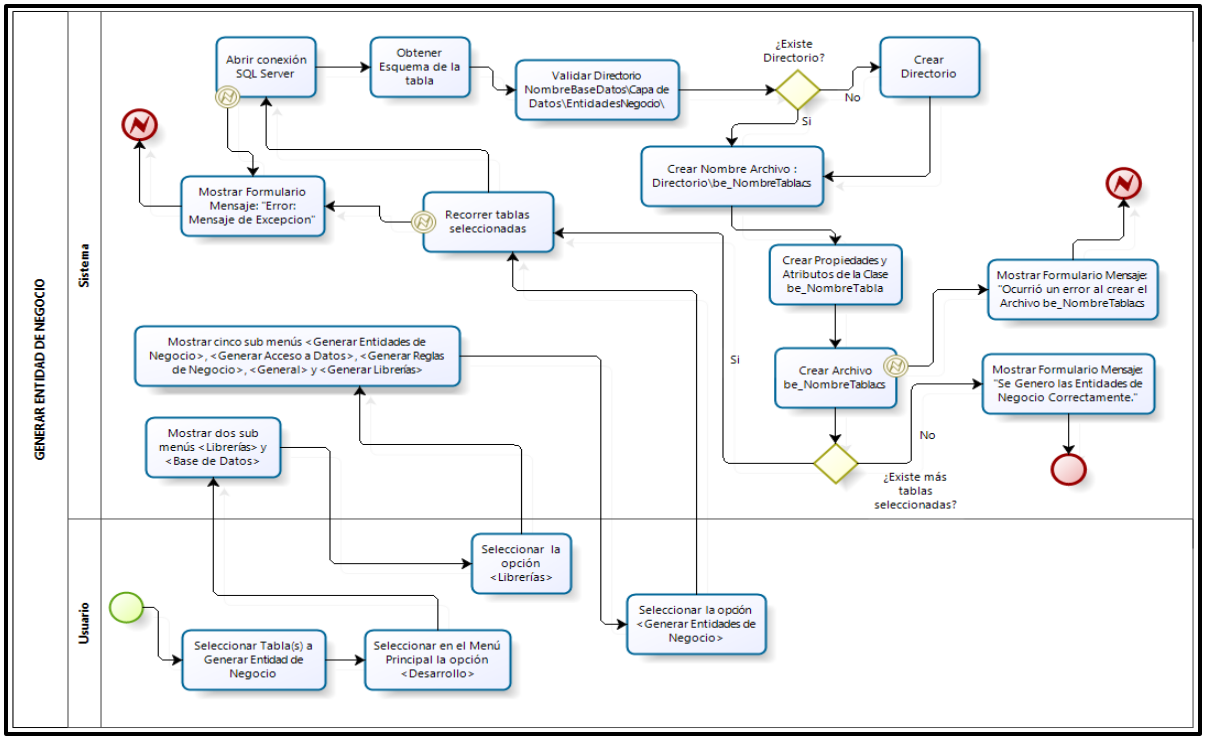
ID:	CU05
Caso de Uso:	Generar Entidad de Negocio
Actor:	Usuario de Sistemas
Descripción:	Este caso de uso permite generar los atributos y propiedades de las clases de entidad de negocio para las tablas de la base de datos a trabajar.
Precondición:	El usuario debe de estar autenticado al sistema.
Flujo de Eventos	
Flujo Principal	
1. El usuario selecciona tabla(s) a generar entidad de negocio. 2. Selecciona en el Menú Principal la opción <Desarrollo>. 3. El sistema muestra dos sub menús <Librerías> y <Base de Datos>. 4. El usuario selecciona la opción <Librerías>. 5. El sistema muestra cinco sub menús <Generar Entidades de Negocio>, <Generar Acceso a Datos>, <Generar Reglas de Negocio>, <General> y <Generar Librerías>. 6. El usuario selecciona la opción <Generar Entidades de Negocio>. 7. El sistema recorre las tablas seleccionadas. 8. Si existe error al recorrer las tablas, el sistema muestra un formulario con el mensaje: "Error: Mensaje de excepción". Caso contrario abre la conexión a Sql Server. 9. Si existe error al abrir la conexión a Sql Server, el sistema muestra un formulario con el mensaje: "Error: Mensaje de excepción". Caso contrario obtiene el esquema de la tabla. 10. El sistema valida si existe el directorio NombreBaseDatos\Capa de Datos\EntidadesNegocio\. 11. Si el directorio no existe, entonces el sistema crea el directorio.	

12. El sistema crear Nombre Archivo : Directorio\be_NombreTabla.cs 13. Crea propiedades y atributos de la clase be_NombreTabla 14. Crea un archivo be_NombreTabla.cs 15. Si existe un error al crear el archivo, el sistema muestra un formulario con el mensaje: "Ocurrió un error al crear el Archivo be_NombreTabla.cs. Caso contrario valida si existen más tablas seleccionadas por recorrer. 16. Si existen más tablas seleccionadas, se repiten los pasos 7 – 15 caso contrario el sistema muestra un formulario con el mensaje “Se generó las Entidades de Negocio Correctamente.”	
Flujo Alternativo de Eventos	
Flujo Secundario	
No aplica	
Post-condición :	1. Se crea una carpeta principal con el nombre EntidadesNegocio donde se va almacenar todas las clases generadas por proceso generar entidad de negocio.

Fuente: Elaboración propia

3.2.2.19. Diagrama de Actividades: Generar Entidad de Negocio

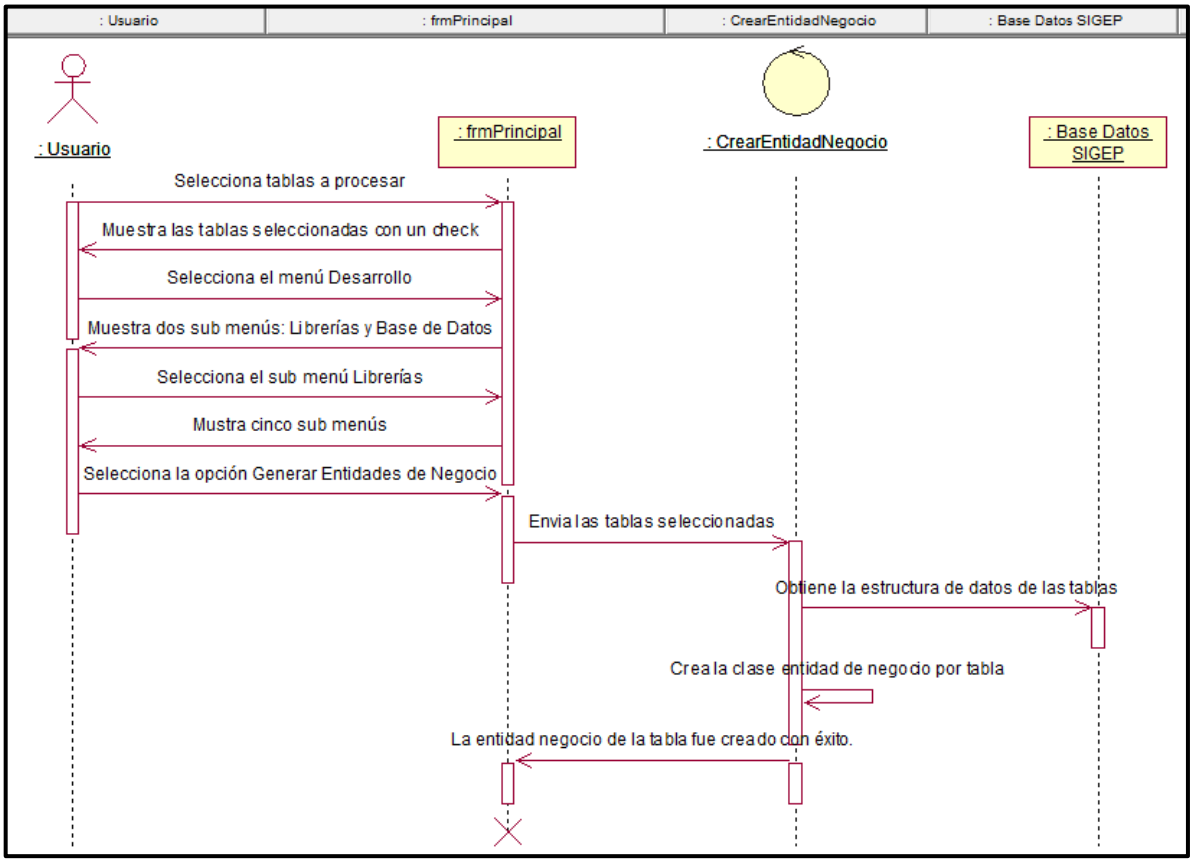
Figura 16: Detalle de diagrama de actividades generar entidad de negocio



Fuente: Elaboración propia

3.2.2.20. Diagrama de Secuencias: Generar Entidad de Negocio

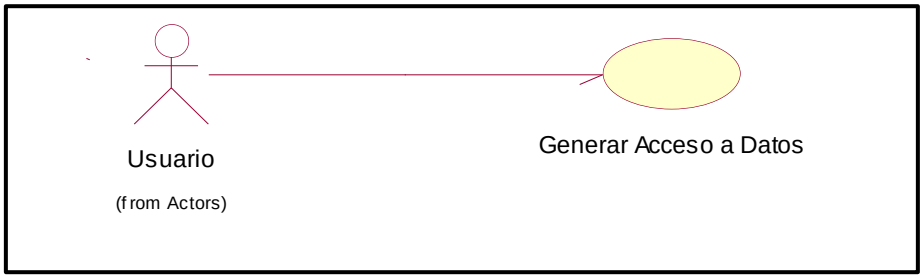
Figura 17: Diagrama de secuencia Generar entidad de negocio



Fuente: Elaboración propia

3.2.2.21. Caso de Uso Generar Acceso a Datos

Figura 18: Diagrama caso de uso generar acceso a datos



Fuente: Elaboración propia

3.2.2.22. Descripción Caso de Uso: Generar Acceso a Datos

Tabla 13: Detalle de caso de uso generar acceso a datos

ID:	CU06
Caso de Uso:	Generar Acceso a Datos
Actor:	Usuario de Sistemas
Descripción:	Este caso de uso permite generar los métodos principales de mantenimiento a su vez sirve de puente de comunicación para acceder a la base de datos.
Precondición:	El usuario debe de estar autenticado al sistema.
Flujo de Eventos	
Flujo Principal	
1. El usuario selecciona tabla(s) a generar acceso a datos. 2. Selecciona en el Menú Principal la opción <Desarrollo>. 3. El sistema muestra dos sub menús <Librerías> y <Base de Datos>. 4. El usuario selecciona la opción <Librerías>. 5. El sistema muestra cinco sub menús <Generar Entidades de Negocio>, <Generar Acceso a Datos>, <Generar Reglas de Negocio>, <General> y <Generar Librerías>. 6. El usuario selecciona la opción <Generar Acceso a Datos >. 7. El sistema recorre las tablas seleccionadas. 8. Si existe error al recorrer las tablas, el sistema muestra un formulario con el mensaje: "Error: Mensaje de excepción". Caso contrario abre la conexión a Sql Server. 9. Si existe error al abrir la conexión a Sql Server, el sistema muestra un formulario con el mensaje: "Error: Mensaje de excepción". Caso contrario obtiene el esquema de la tabla. 10. El sistema valida si existe el directorio NombreBaseDatos\Capa de Datos\AccesoDatos\.	

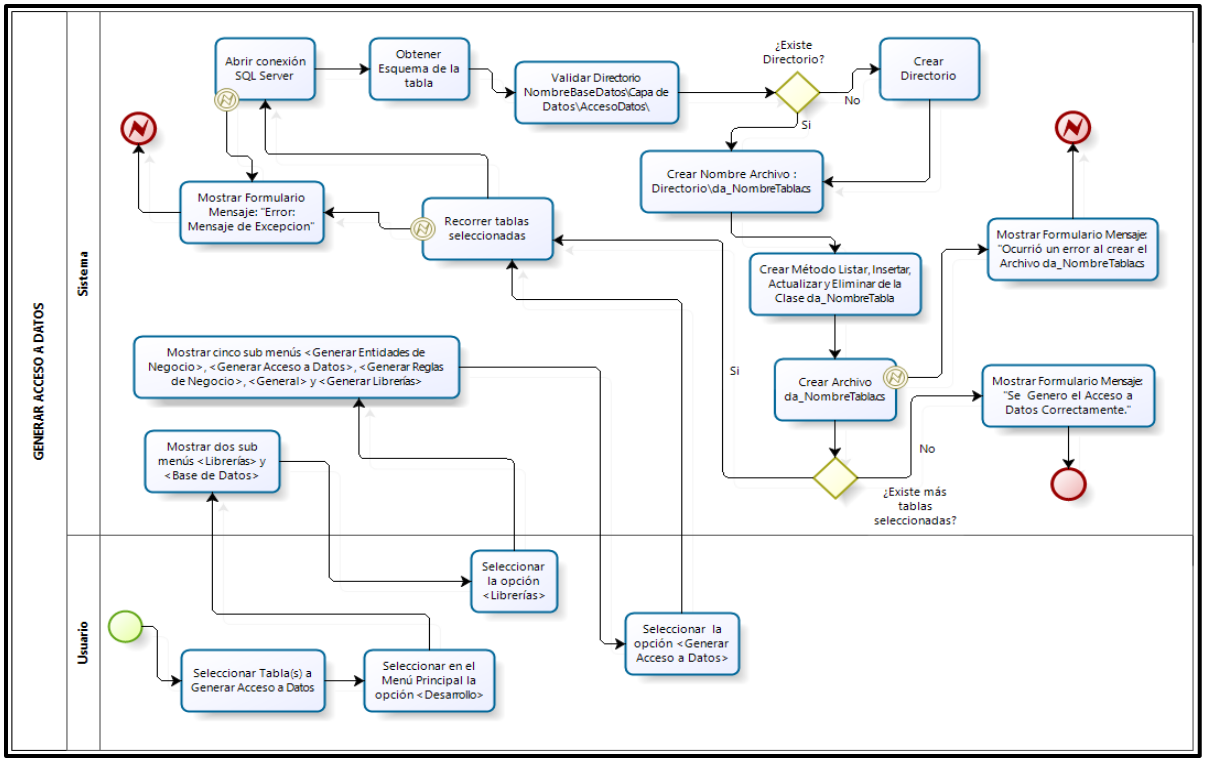
11. Si el directorio no existe, entonces el sistema crea el directorio. 12. El sistema crear Nombre Archivo : Directorio\da_NombreTabla.cs 13. Crea los métodos Listar, Registrar, Modificar, Eliminar y Buscar de la Clase da_NombreTabla. 14. Crea un archivo da_NombreTabla.cs 15. Si existe un error al crear el archivo, el sistema muestra un formulario con el mensaje: "Ocurrió un error al crear el Archivo da_NombreTabla.cs. Caso contrario valida si existen más tablas seleccionadas por recorrer. Si existen más tablas seleccionadas, se repiten los pasos 7 – 15 caso contrario el sistema muestra un formulario con el mensaje "Se generó el Acceso a Datos Correctamente."
Flujo Alternativo de Eventos
Flujo Secundario
No aplica

Post-condición :	1. Se crea una carpeta principal con el nombre AccesoDatos donde se va almacenar todas las clases generadas por el proceso generar Acceso a Datos.

Fuente: Elaboración propia

3.2.2.23. Diagrama de Actividades: Generar Acceso a Datos

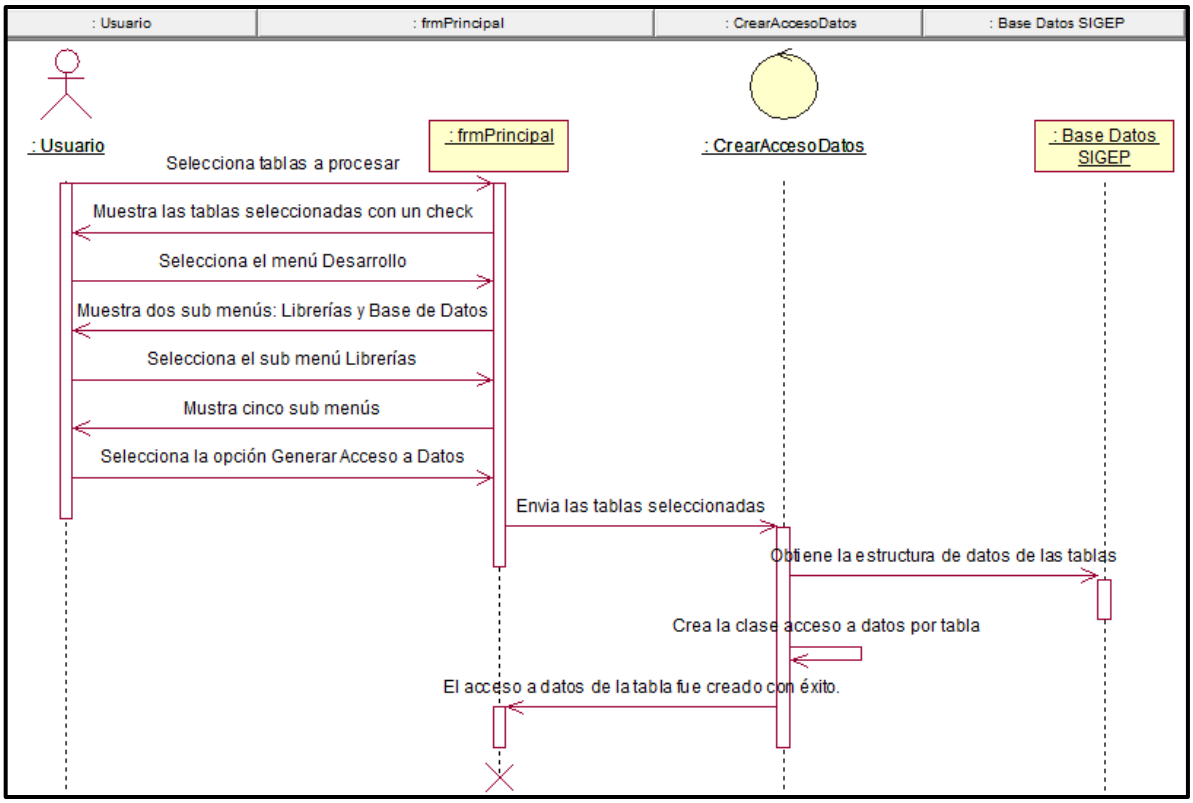
Figura 19: Detalle de diagrama de actividades generar acceso a datos



Fuente: Elaboración propia

3.2.2.24. Diagrama de Secuencias: Generar Acceso a Datos

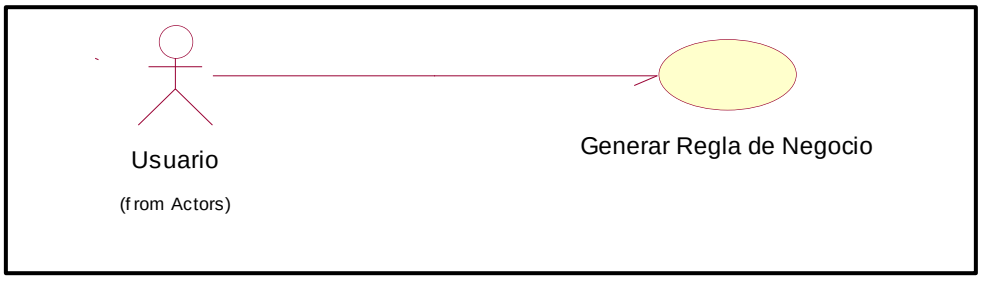
Figura 20: Diagrama de secuencia Generar acceso a datos



Fuente: Elaboración propia

3.2.2.25. Caso de Uso Generar Regla de Negocio

Figura 21: Diagrama caso de uso generar regla de negocio



Fuente: Elaboración propia

3.2.2.26. Descripción Caso de uso: Generar Regla de Negocio

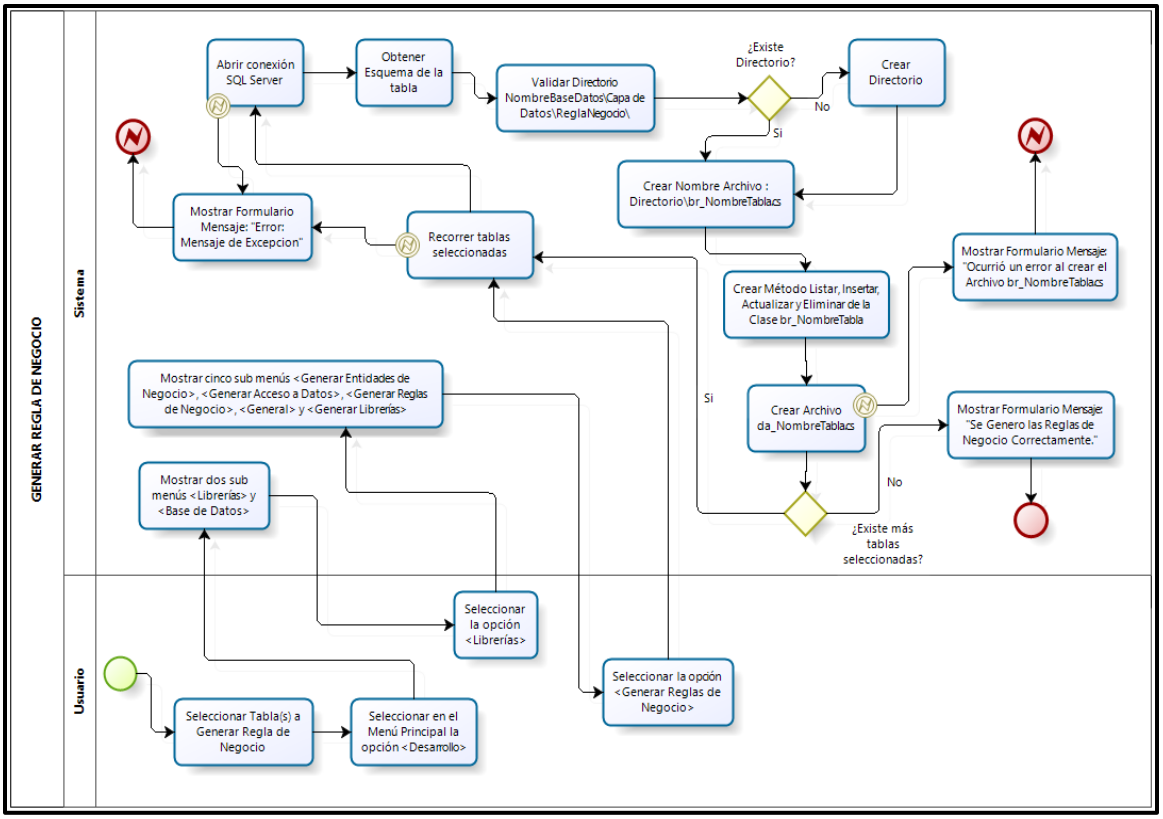
Tabla 14: Detalle de caso de uso generar regla de negocio

ID:	CU07
Caso de Uso:	Generar Regla de Negocio
Actor:	Usuario de Sistemas
Descripción:	Permite generar los métodos principales de mantenimiento a su vez sirve de enlace para enviar datos a la capa de presentación.
Precondición:	El usuario debe de estar autenticado al sistema.
Flujo de Eventos	
Flujo Principal	
1. El usuario selecciona tabla(s) a generar regla de negocio. 2. Selecciona en el Menú Principal la opción <Desarrollo>. 3. El sistema muestra dos sub menús <Librerías> y <Base de Datos>. 4. El usuario selecciona la opción <Librerías>. 5. El sistema muestra cinco sub menús <Generar Entidades de Negocio>, <Generar Acceso a Datos>, <Generar Reglas de Negocio>, <General> y <Generar Librerías>. 6. El usuario selecciona la opción <Generar Reglas de Negocio>. 7. El sistema recorre las tablas seleccionadas. 8. Si existe error al recorrer las tablas, el sistema muestra un formulario con el mensaje: "Error: Mensaje de excepción". Caso contrario abre la conexión a Sql Server. 9. Si existe error al abrir la conexión a Sql Server, el sistema muestra un formulario con el mensaje: "Error: Mensaje de excepción". Caso contrario obtiene el esquema de la tabla. 10. El sistema valida si existe el directorio NombreBaseDatos\Capa de Datos\ReglaNegocio\. 11. Si el directorio no existe, entonces el sistema crea el directorio. 12. El sistema crear Nombre Archivo : Directorio\br_NombreTabla.cs 13. Crea los métodos Listar, Registrar, Modificar, Eliminar y Buscar de la Clase br_NombreTabla. 14. Crea un archivo br_NombreTabla.cs 15. Si existe un error al crear el archivo, el sistema muestra un formulario con el mensaje: "Ocurrió un error al crear el Archivo br_NombreTabla.cs. Caso contrario valida si existen más tablas seleccionadas por recorrer. Si existen más tablas seleccionadas, se repiten los pasos 7 – 15 caso contrario el sistema muestra un formulario con el mensaje “Se generó las Reglas de Negocio Correctamente.”	
Flujo Alternativo de Eventos	
Flujo Secundario	
No aplica	
Post-condición :	1. Se crea una carpeta principal con el nombre ReglaNegocio donde se va almacenar todas las clases generadas por el proceso generar Regla de Negocio.

Fuente: Elaboración propia

3.2.2.27. Diagrama de Actividades: Generar Regla de Negocio

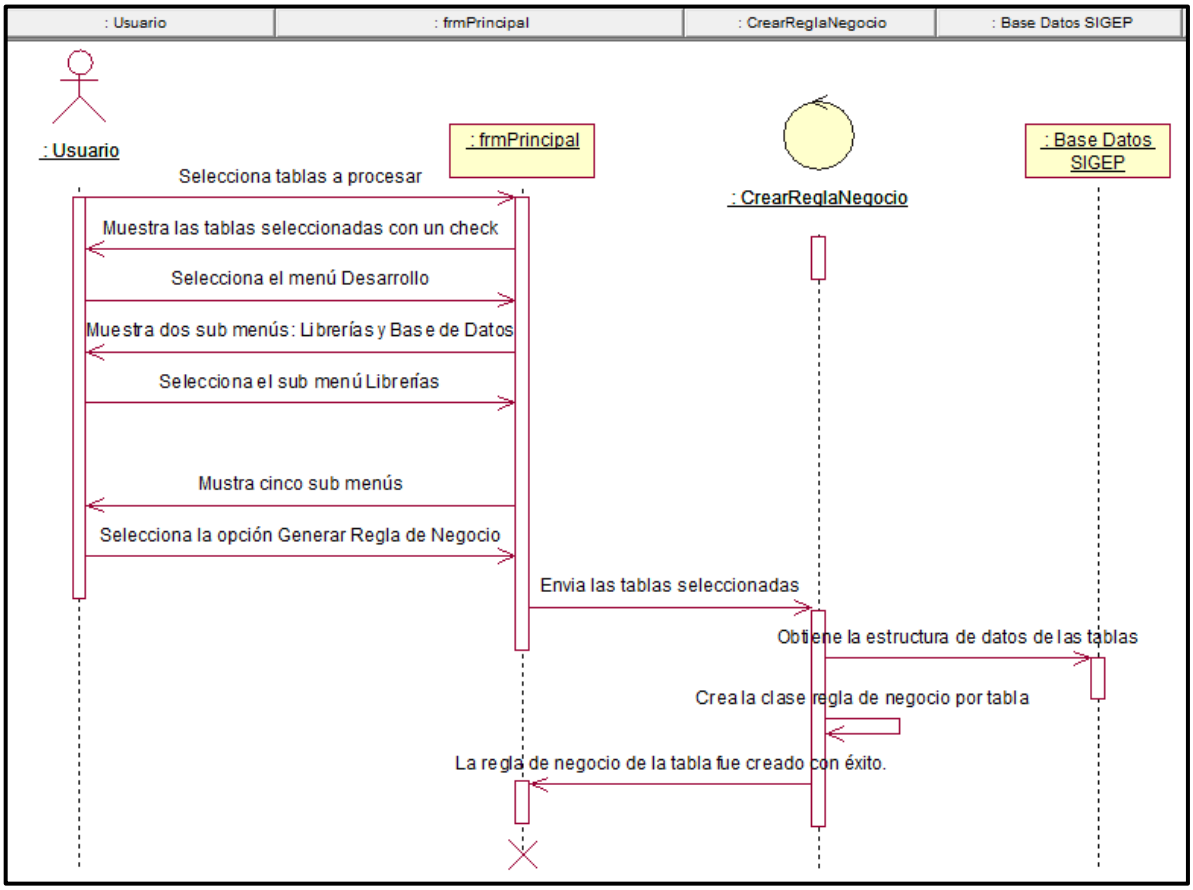
Figura 22: Detalle de diagrama de actividades generar regla de negocio



Fuente: Elaboración propia

3.2.2.28. Diagrama de Secuencias: Generar Regla de Negocio

Figura 23: Diagrama de secuencia Generar regla de negocio



Fuente: Elaboración Propia

3.2.2.29. Caso de Uso Generar Librerías

Figura 24: Diagrama caso de uso generar librerías

Fuente: Elaboración propia

3.2.2.30. Diagrama de Caso de Uso: Generar Librerías

Tabla 15: Detalle de caso de uso generar librerías

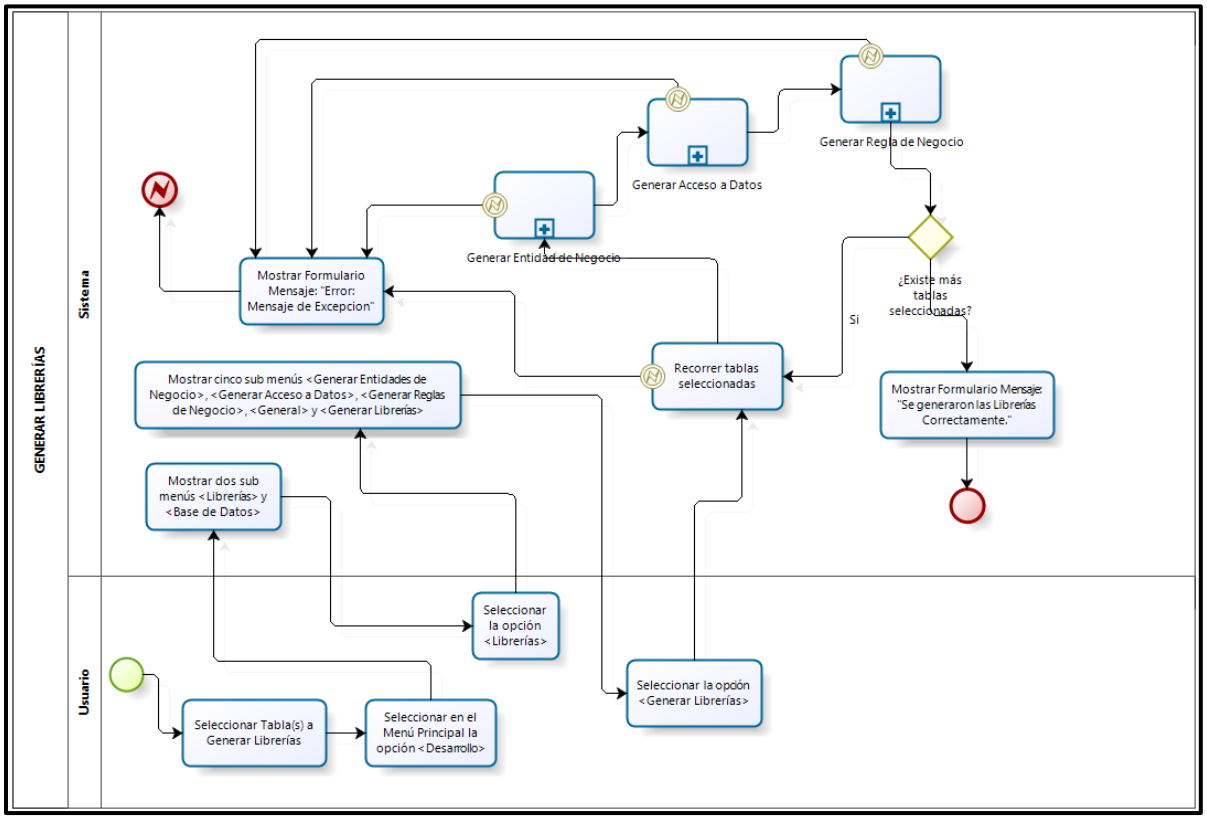
ID:	CU08
Caso de Uso:	Generar Librerías
Actor:	Usuario de Sistemas
Descripción:	Este proceso permite generar las tres capas de datos (Entidad de Negocio, Acceso a Datos y Regla de Negocio) de forma automática.
Precondición:	El usuario debe de estar autenticado al sistema.
Flujo de Eventos	
Flujo Principal	
1. El usuario selecciona tabla(s) a generar librerías. 2. Selecciona en el Menú Principal la opción <Desarrollo>. 3. El sistema muestra dos sub menús <Librerías> y <Base de Datos>. 4. El usuario selecciona la opción <Librerías>. 5. El sistema muestra cinco sub menús <Generar Entidades de Negocio>, <Generar Acceso a Datos>, <Generar Reglas de Negocio>, <General> y <Generar Librerías>. 6. El usuario selecciona la opción <Generar Librerías>. 7. El sistema recorre las tablas seleccionadas. 8. Si existe error al recorrer las tablas, el sistema muestra un formulario con el mensaje: "Error: Mensaje de excepción". Caso contrario genera el proceso Entidad de Negocio, genera el proceso Acceso a Datos y finalmente el proceso Regla de Negocio. 9. El sistema valida si existen más tablas seleccionadas por recorrer. 10. Si existen más tablas seleccionadas, se repiten los pasos 7 – 9 caso contrario el sistema muestra un formulario con el mensaje “Se generó las Librerías Correctamente.”	

Flujo Alternativo de Eventos	
Flujo Secundario	
No aplica	
Post-condición :	1. Se crea tres carpetas principales con el nombre EntidadNegocio, AccesoDatos y ReglaNegocio donde se va almacenar todas las clases generadas por el proceso generar Librerías.

Fuente: Elaboración propia

3.2.2.31. Diagrama de Actividades: Generar Librerías

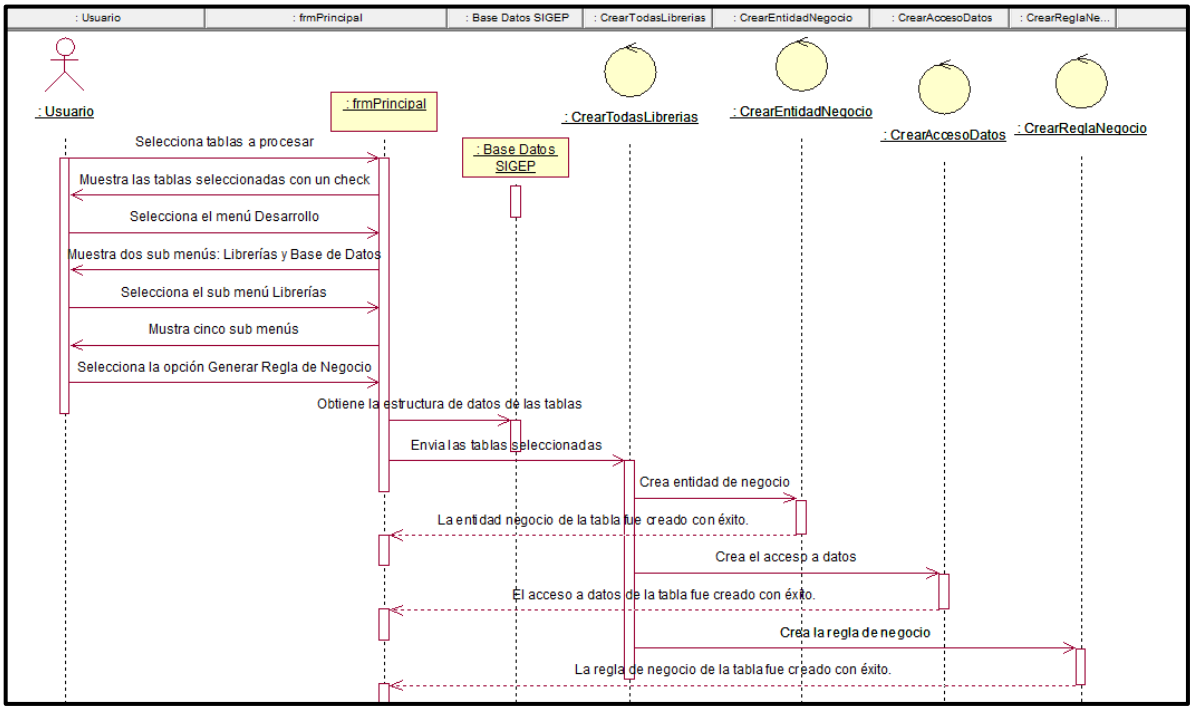
Figura 25: Detalle de diagrama de actividades generar librerías



Fuente: Elaboración propia

3.2.2.32. Diagrama de Secuencias: Generar Librerías

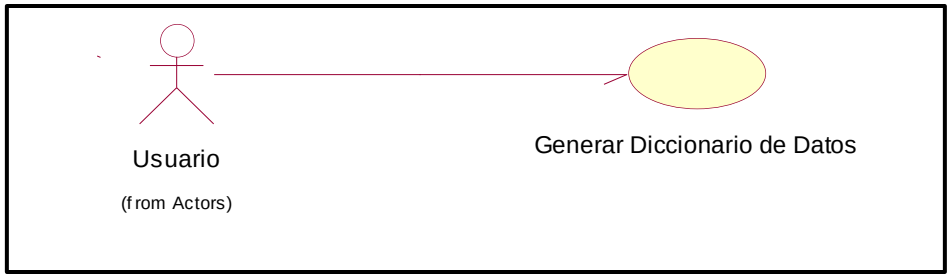
Figura 26: Diagrama de secuencia Generar librerías



Fuente: Elaboración propia

3.2.2.33. Caso de Uso Crear Diccionario de Datos

Figura 27: Diagrama caso de uso crear diccionario de datos



Fuente: Elaboración propia

3.2.2.34. Descripción Caso de uso: Crear Diccionario de Datos

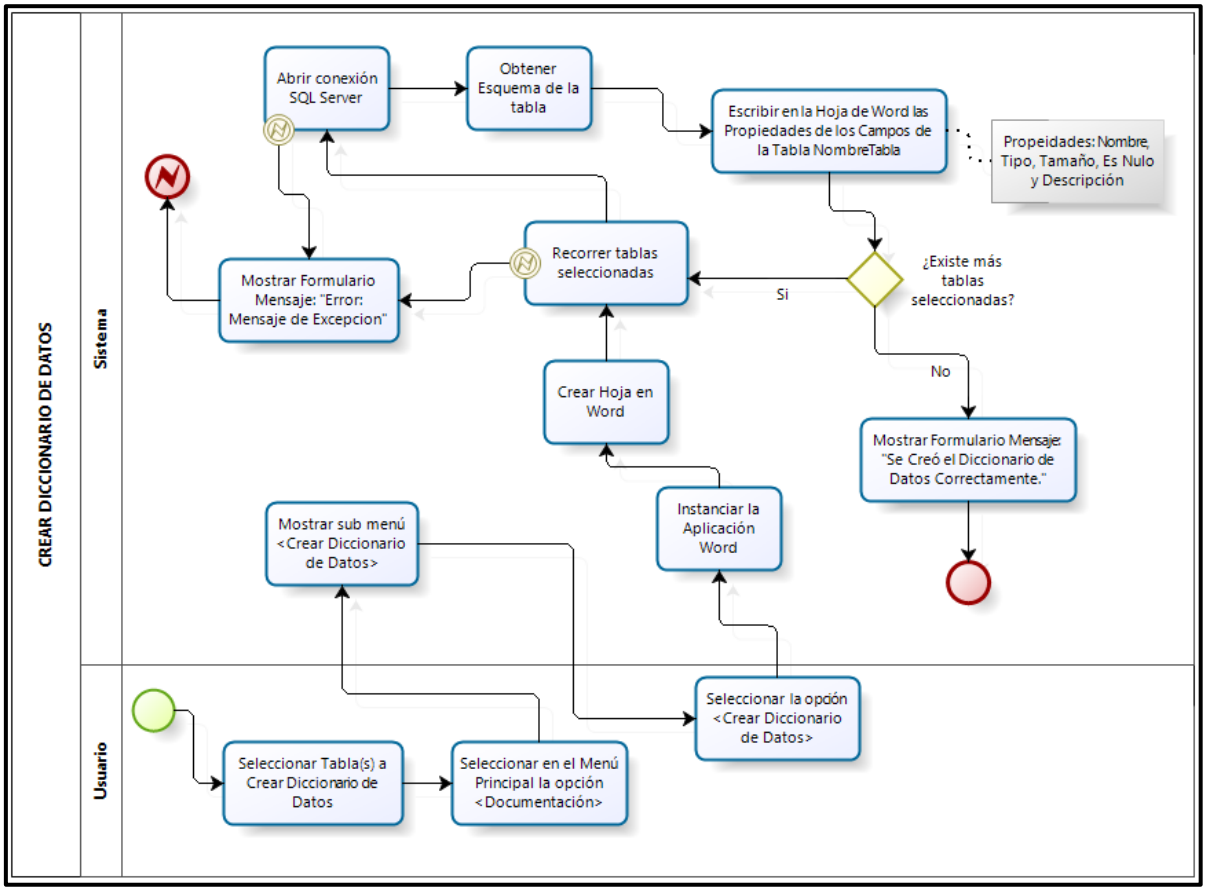
Tabla 16: Detalle de caso de uso crear diccionario de datos

ID:	CU09
Caso de Uso:	Crear Diccionario de Datos
Actor:	Usuario de Sistemas
Descripción:	Este proceso permite crear un diccionario de datos de las tablas seleccionas de la base de datos.
Precondición:	El usuario debe de estar autenticado al sistema.
Flujo de Eventos	
Flujo Principal	
1. El usuario selecciona Tabla(s) a Crear Diccionario de Datos. 2. Selecciona en el Menú Principal la opción <Documentación>. 3. El sistema muestra un sub menú <Crear Diccionario de Datos>. 4. El usuario selecciona la opción <Crear Diccionario de Datos>. 5. El sistema instanciar la Aplicación Word. 6. Crea una Hoja en Word. 7. Recorre las tablas seleccionadas. 8. Si existe error al recorrer las tablas, el sistema muestra un formulario con el mensaje: "Error: Mensaje de excepción". Caso contrario abre la conexión a Sql Server. 9. Si existe error al abrir la conexión a Sql Server, el sistema muestra un formulario con el mensaje: "Error: Mensaje de excepción". Caso contrario obtiene el esquema de la tabla. 10. El sistema escribe en la Hoja de Word las Propiedades de los Campos de la Tabla NombreTabla 11. El sistema valida si existen más tablas seleccionadas por recorrer. 12. Si existen más tablas seleccionadas, se repiten los pasos 7 – 11 caso contrario el sistema muestra un formulario con el mensaje “Se creó el Diccionario de Datos Correctamente.”	
Flujo Alternativo de Eventos	
Flujo Secundario	
No aplica	
Post-condición :	No aplica

Fuente: Elaboración propia

3.2.2.35. Diagrama de actividades: Crear Diccionario de Datos

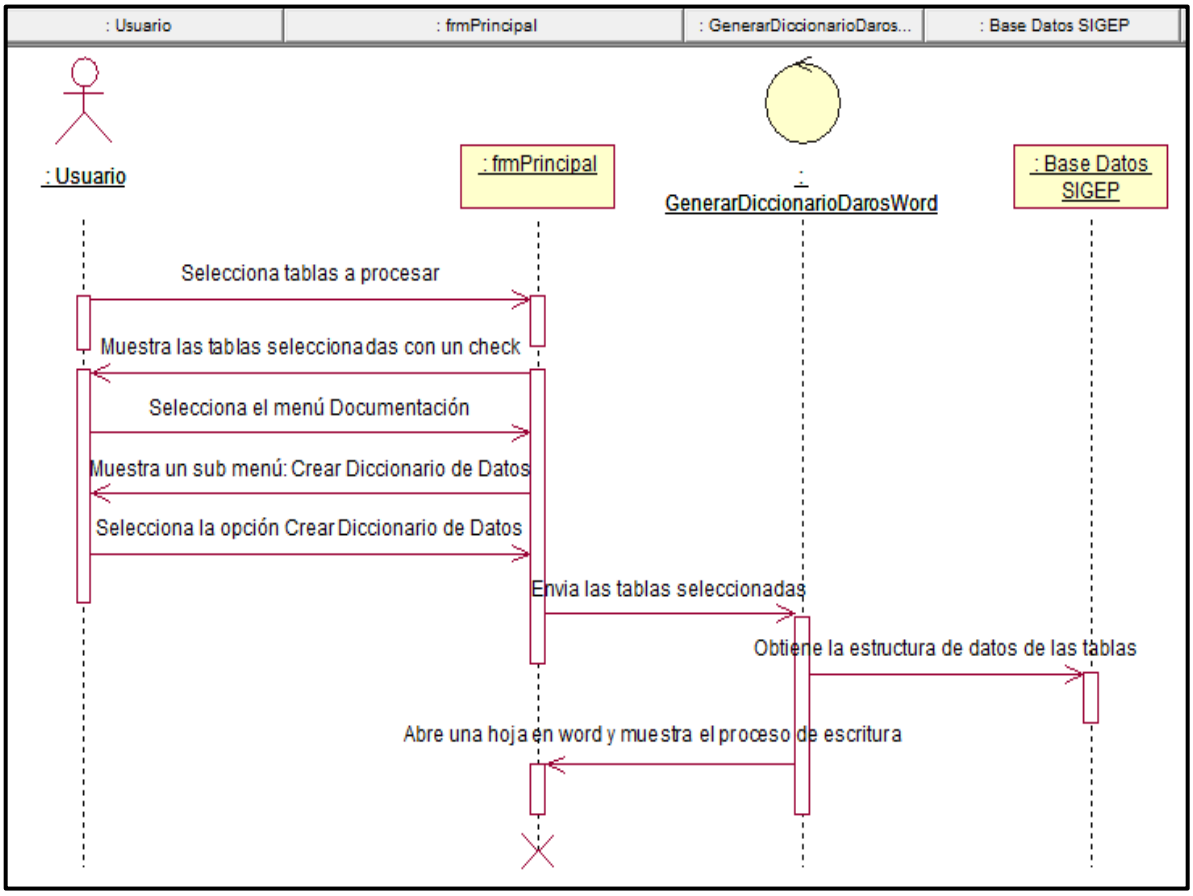
Figura 28: Detalle de diagrama de actividades crear diccionario de datos



Fuente: Elaboración propia

3.2.2.36. Diagrama de Secuencias: Crear Diccionario de Datos

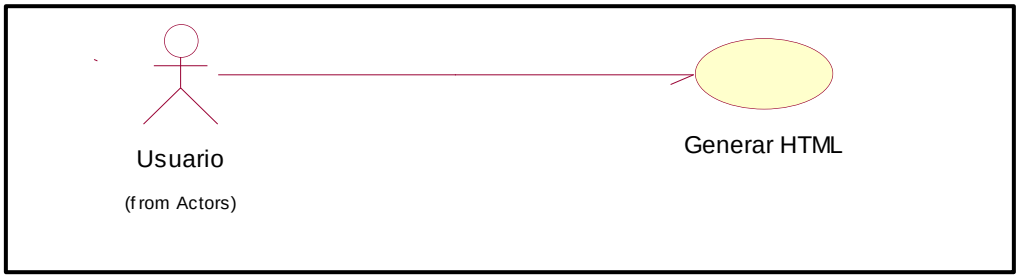
Figura 29: Diagrama de secuencia Crear diccionario de datos



Fuente: Elaboración propia

3.2.2.37. Caso de Uso Generar HTML

Figura 30: Diagrama caso de uso Generar HTML



Fuente: Elaboración propia

3.2.2.38. Descripción Caso de Uso: Generar HTML

Tabla 17: Detalle de caso de uso generar HTML

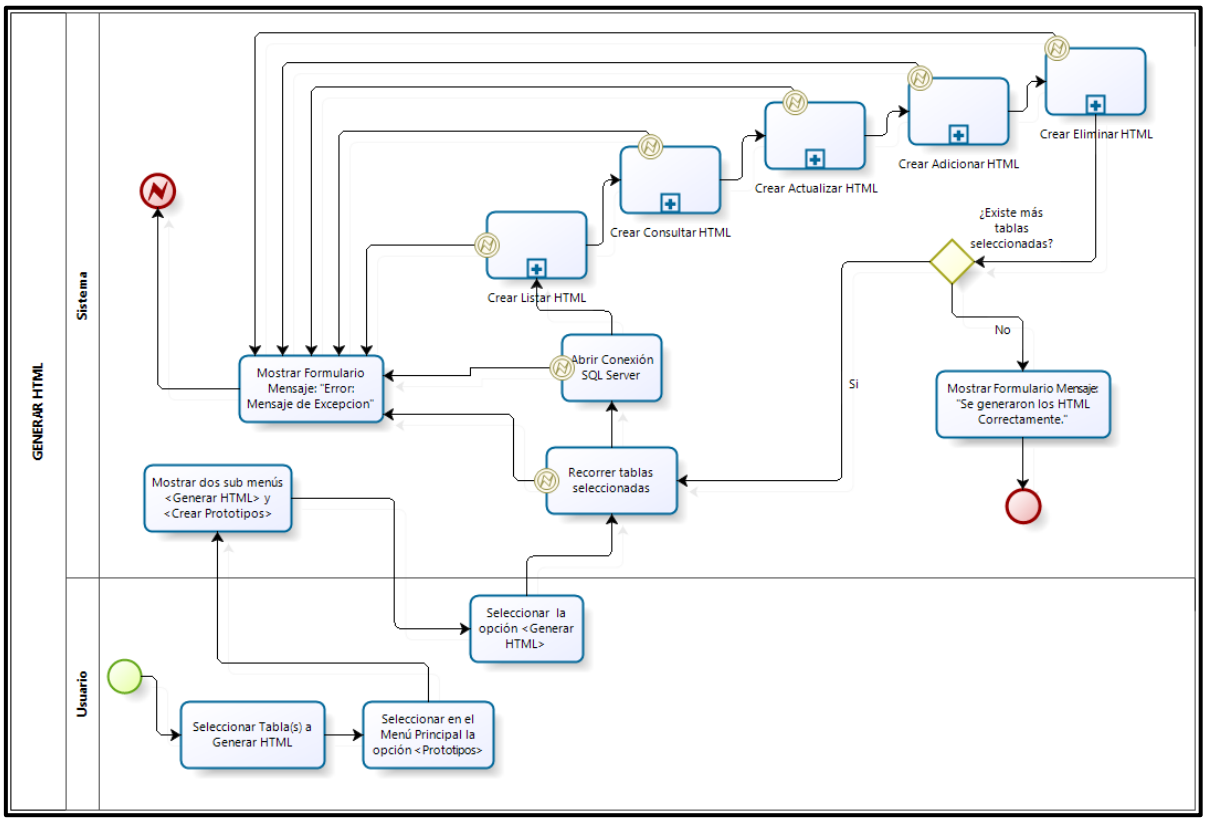
ID:	CU10
Caso de Uso:	Generar HTML
Actor:	Usuario de Sistemas
Descripción:	Este proceso permite generar código HTML para la parte de diseño de prototipos.
Precondición:	El usuario debe de estar autenticado al sistema.
Flujo de Eventos	
Flujo Principal	
1. El usuario selecciona tabla(s) a generar HTML. 2. Selecciona en el Menú Principal la opción <Prototipos>. 3. El sistema muestra dos sub menús <Generar HTML> y <Crear Prototipos> 4. El usuario selecciona la opción < Generar HTML>. 5. El sistema recorre las tablas seleccionadas. 6. Si existe error al recorrer las tablas, el sistema muestra un formulario con el mensaje: "Error: Mensaje de excepción". Caso contrario Abre la conexión a Sql Server. 7. Si existe error al abrir la conexión a Sql Server, el sistema muestra un formulario con el mensaje: "Error: Mensaje de excepción". Caso contrario genera el proceso Crear Listar HTML, Crear Consultar HTML, Crear Actualizar HTML, Crear Adicionar HTML y finalmente Crear Eliminar HTML. 8. El sistema valida si existen más tablas seleccionadas por recorrer. 9. Si existen más tablas seleccionadas, se repiten los pasos 5 – 8 caso contrario el sistema muestra un formulario con el mensaje “Se generaron los HTML	

Correctamente.”	
Flujo Alternativo de Eventos	
Flujo Secundario	
No aplica	
Post-condición :	1. Se crea una carpetas principal con el nombre HTM donde se va almacenar todos los archivos .HTML generados por el proceso generar HTML.

Fuente: Elaboración propia

3.2.2.39. Diagrama de Actividades: Generar HTML

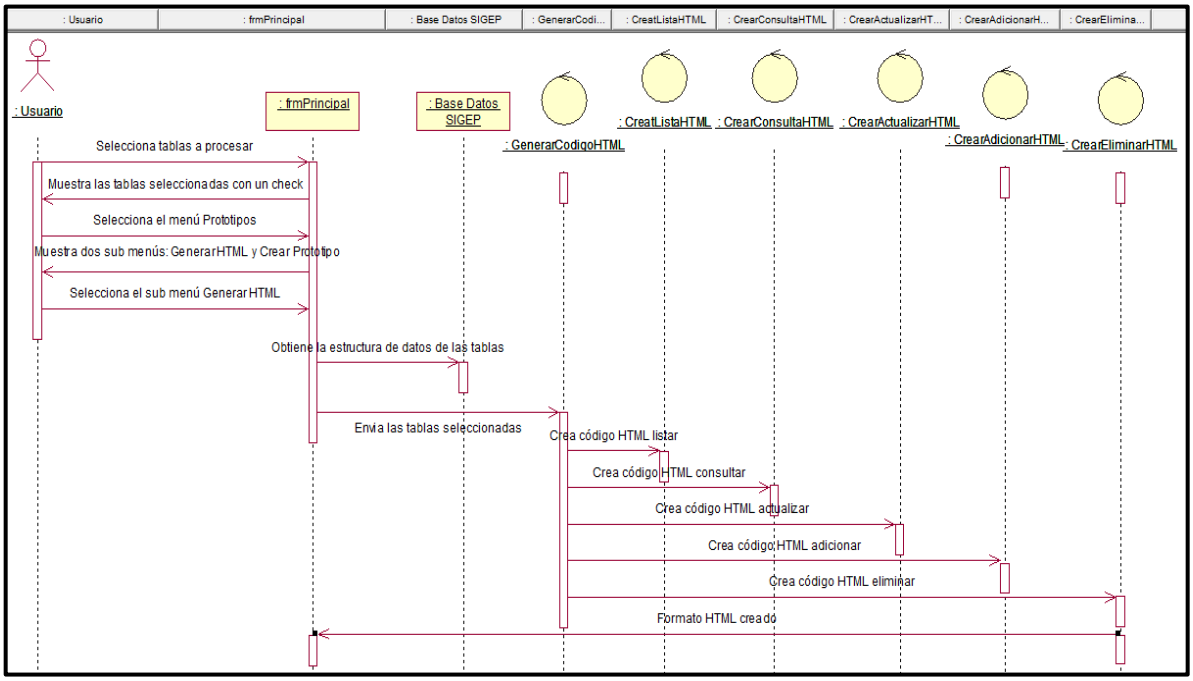
Figura 31: Detalle de diagrama de actividades generar HTML



Fuente: Elaboración propia

3.2.2.40. Diagrama de Secuencias: Generar HTML

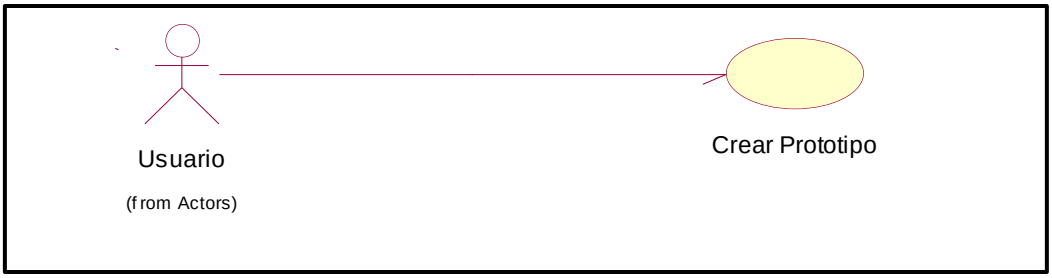
Figura 32: Diagrama de secuencia Generar HTML



Fuente: Elaboración propia

3.2.2.41. Caso de Uso Crear Prototipo

Figura 33: Diagrama caso de uso Crear Prototipo



Fuente: Elaboración propia

3.2.2.42. Descripción Caso de Uso: Crear Prototipo

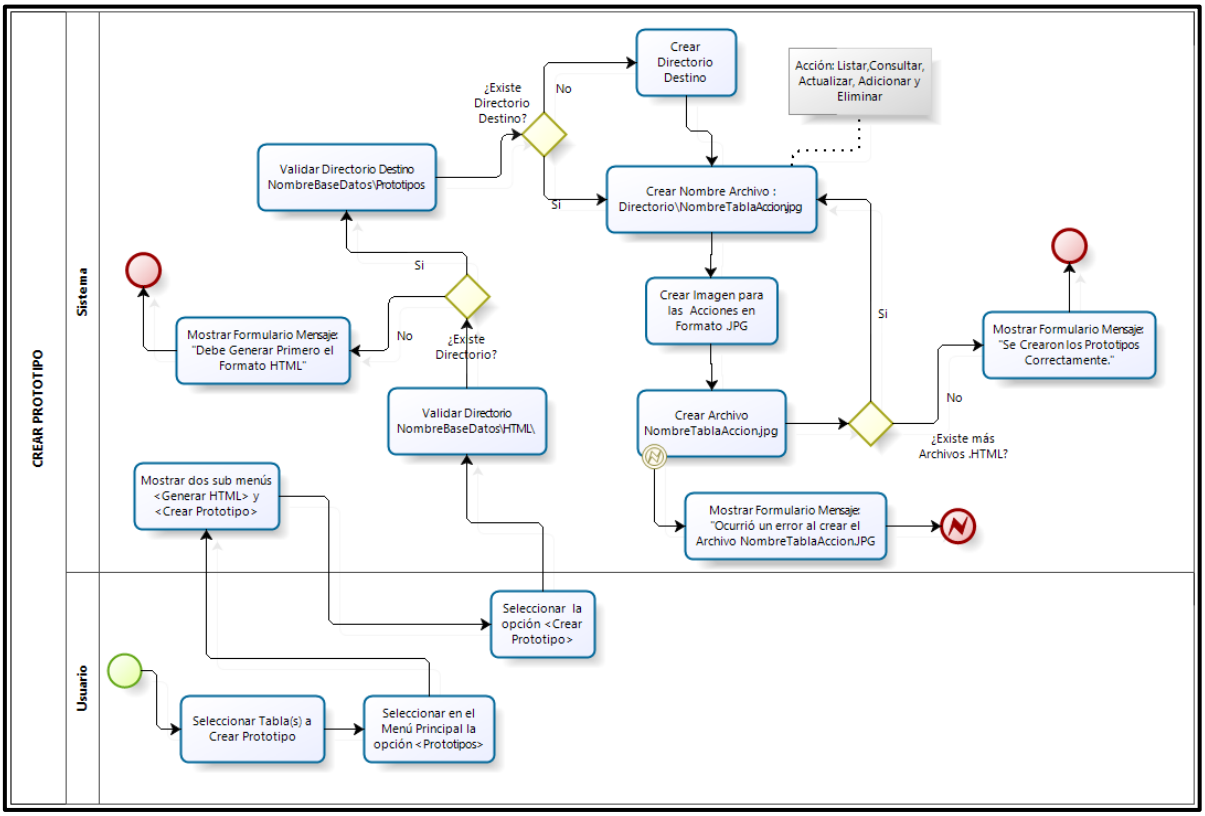
Tabla 18: Detalle de caso de uso crear prototipo

ID:	CU11
Caso de Uso:	Crear Prototipo
Actor:	Usuario de Sistemas
Descripción:	Este proceso permite crear los prototipos de imágenes en formato .JPG
Precondición:	El usuario debe de haber realizado el proceso de Generar HTML
Flujo de Eventos	
Flujo Principal	
1. El usuario selecciona Tabla(s) a Crear Prototipo. 2. Selecciona en el Menú Principal la opción <Prototipos>. 3. El sistema muestra dos sub menús <Generar HTML> y <Crear Prototipo>. 4. El usuario seleccionar la opción <Crear Prototipo>. 5. El sistema validar Directorio NombreBaseDatos\HTML\. 6. Si el directorio existe, validar directorio destino NombreBaseDatos\Prototipos. Caso contrario muestra un formulario con el mensaje: "Debe Generar Primero el Formato HTML." 7. Valida si existe el directorio destino. 8. Si existe el directorio destino, Crear Nombre Archivo : Directorio\NombreTablaAccion.jpg. Caso contrario crea directorio destino. 9. El sistema crea las imágenes para las acciones en formato .JPG 10. Crea el archivo NombreTablaAccion.jpg 11. Si existe un error al crear el archivo, el sistema muestra un formulario con el mensaje: "Ocurrió un error al crear el Archivo NombreTablaAccion.JPG". Caso contrario verifica si existen más archivos HTML. 12. Si existen más archivos HTML por recorrer, entonces repite los pasos 8 – 11. Caso contrario el sistema muestra un formulario con el mensaje: "Se Crearon los Prototipos Correctamente."	
Flujo Alternativo de Eventos	
Flujo Secundario	
No aplica	
Post-condición :	1. Se crea una carpetas principal con el nombre Prototipos donde se va almacenar todos los archivos .JPG generados por el proceso generar Prototipos.

Fuente: Elaboración propia

3.2.2.43. Diagrama de Actividades: Crear Prototipo

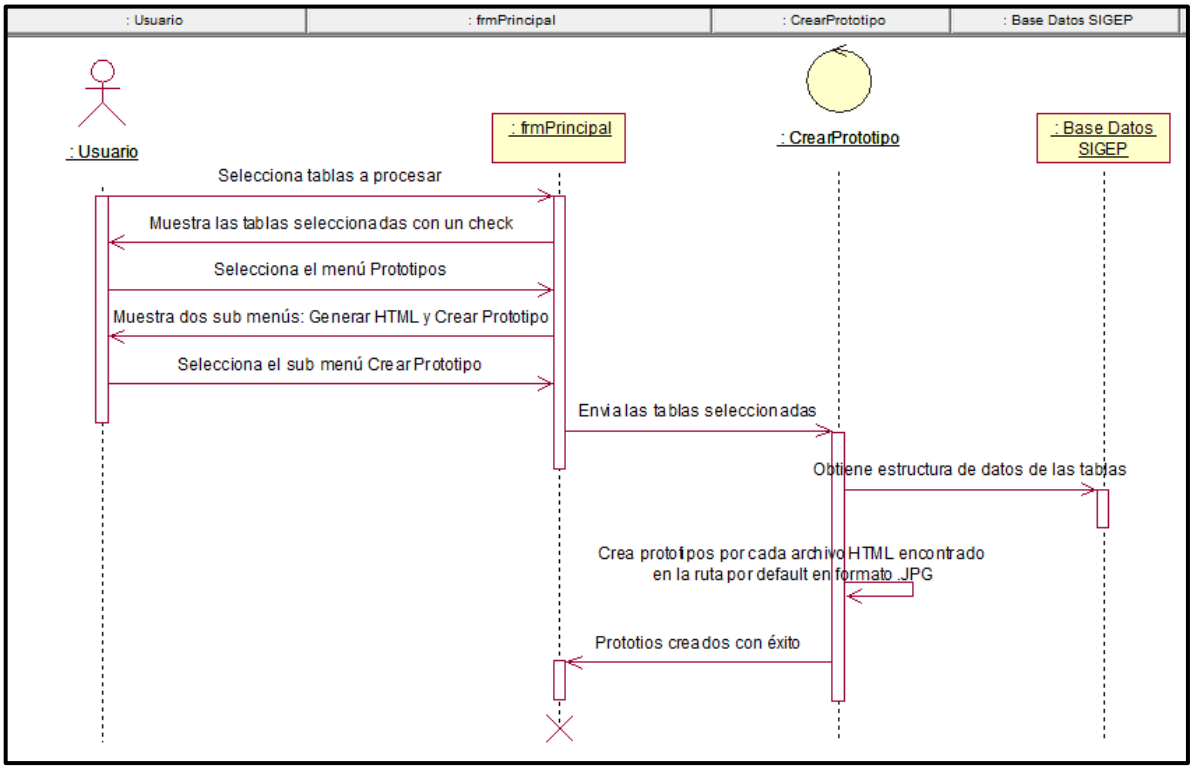
Figura 34: Detalle de diagrama de actividades crear prototipo



Fuente: Elaboración propia

3.2.2.44. Diagrama de Secuencias: Crear Prototipo

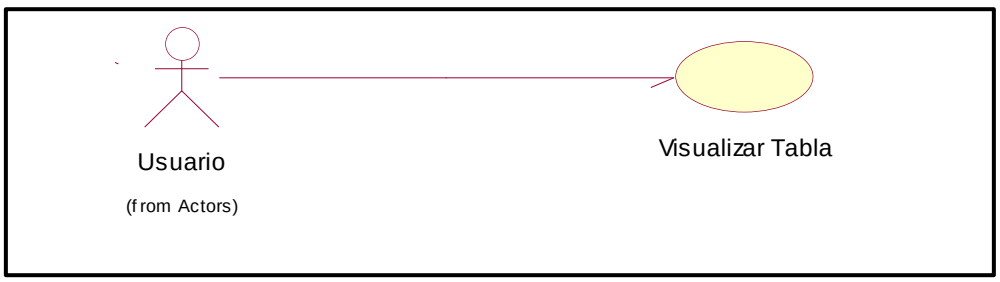
Figura 35: Diagrama de secuencia Crear prototipo



Fuente: Elaboración propia

3.2.2.45. Caso de Uso Visualizar Tabla

Figura 36: Diagrama caso de uso visualizar tabla



Fuente: Elaboración propia

3.2.2.46. Descripción Caso de Uso: Visualizar Tabla

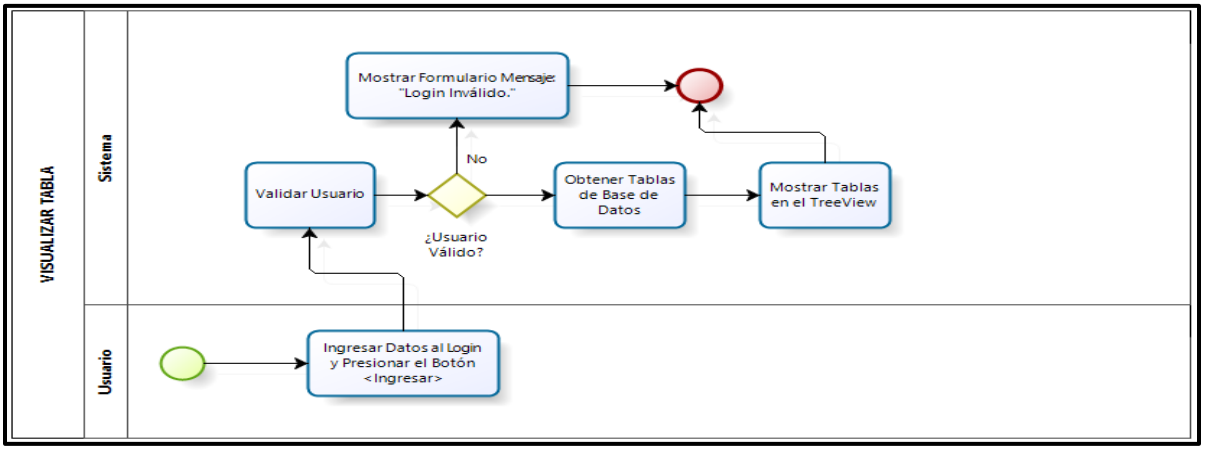
Tabla 19: Detalle de caso de uso visualizar tabla

ID:	CU12
Caso de Uso:	Visualizar Tabla
Actor:	Usuario de Sistemas
Descripción:	Este proceso permite visualizar todas las tablas que se encuentran creadas en la base de datos.
Precondición:	No aplica
Flujo de Eventos	
Flujo Principal	
1. El usuario debe ingresar datos al login y presionar el botón <Ingresar>. 2. El sistema valida al Usuario. 3. Si el usuario es válido, entonces el sistema obtiene las tablas de Base de Datos. Caso contrario el sistema muestra un formulario con el mensaje: "Login Inválido." Y sale del sistema. 4. El sistema muestra las tablas en un TreeView del formulario principal.	
Flujo Alternativo de Eventos	
Flujo Secundario	
No aplica	
Post-condición :	No aplica

Fuente: Elaboración propia

3.2.2.47. Diagrama de Actividades: Visualizar Tabla

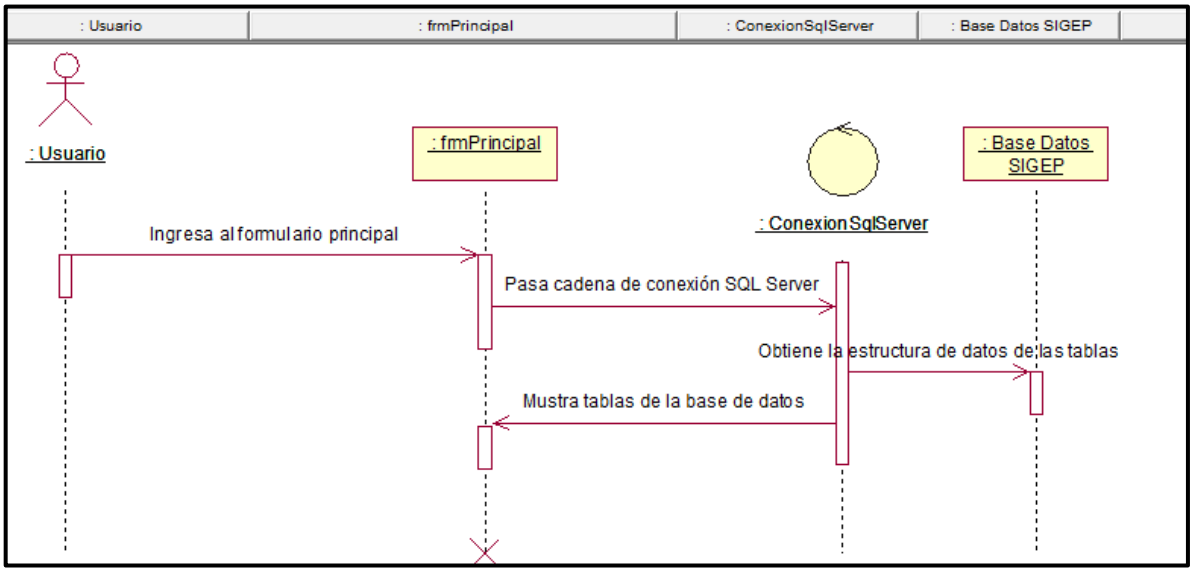
Figura 37: Detalle de diagrama de actividades visualizar tabla



Fuente: Elaboración propia

3.2.2.48. Diagrama de Secuencias: Visualizar Tabla

Figura 38: Diagrama de secuencia Visualizar tabla



Fuente: Elaboración propia

3.2.2.49. Caso de Uso Visualizar Detalle Tabla

Figura 39: Diagrama caso de uso visualizar detalle tabla

Fuente: Elaboración propia

3.2.2.50. Descripción Caso de Uso: Visualizar Detalle Tabla

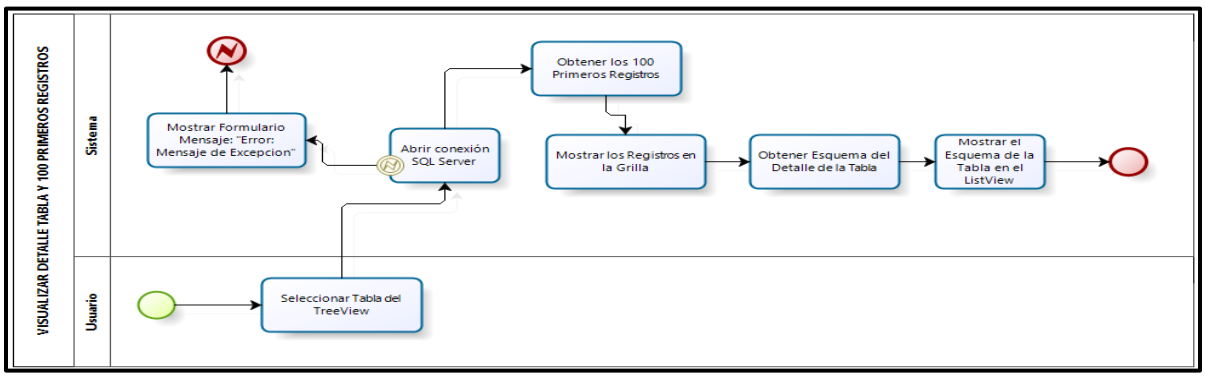
Tabla 20: Detalle de caso de uso visualizar detalle tabla

ID:	CU13
Caso de Uso:	Visualizar Detalle Tabla
Actor:	Usuario de Sistemas
Descripción:	Este proceso permite mostrar al usuario el detalle de cada tabla (Nombre, campo, tamaño, descripción, datos nulos y campo con PK)
Precondición:	El usuario debe de estar autenticado al sistema.
Flujo de Eventos	
Flujo Principal	
1. El usuario selecciona la tabla del árbol de opciones. 2. El sistema abre la conexión SQL Server. 3. Si existe un error en la conexión, el sistema muestra un formulario con el mensaje: "Error: Mensaje de Excepción". Caso contrario obtiene los 100 primeros registros. 4. El sistema muestra los registros en la grilla. 5. Obtiene el esquema del detalle de la tabla. 6. El sistema muestra el esquema de la tabla en el ListView.	
Flujo Alternativo de Eventos	
Flujo Secundario	
No aplica	
Post-condición :	No aplica

Fuente: Elaboración propia

3.2.2.51. Diagrama de Actividades: Visualizar Detalle Tabla

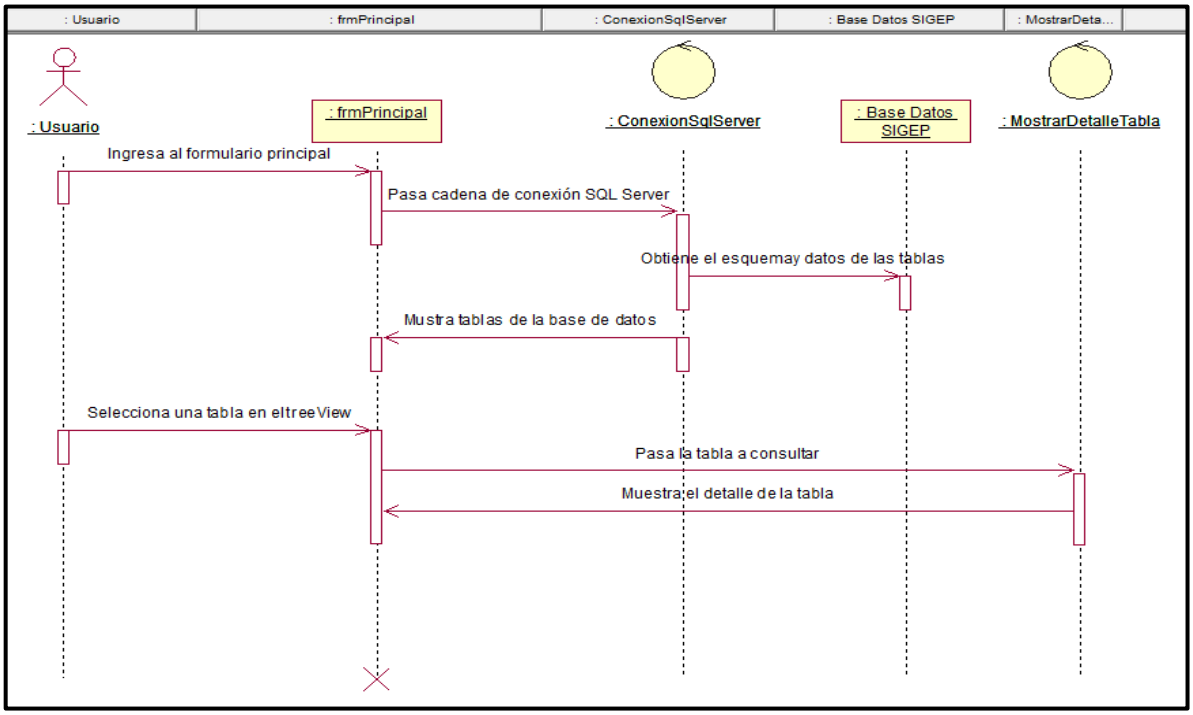
Figura 40: Detalle de diagrama de actividades visualizar detalle tabla



Fuente: Elaboración propia

3.2.2.52. Diagrama de Secuencias: Visualizar Detalle Tabla

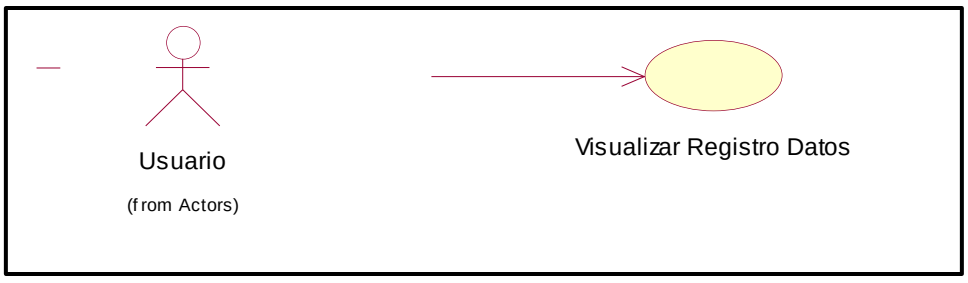
Figura 41: Diagrama de secuencia Visualizar detalle tabla



Fuente: Elaboración propia

3.2.2.53. Caso de Uso Visualizar Registro Datos

Figura 42: Diagrama caso de uso visualizar registro datos



Fuente: Elaboración propia

3.2.2.54. Descripción Caso de Uso: Visualizar Registro Datos

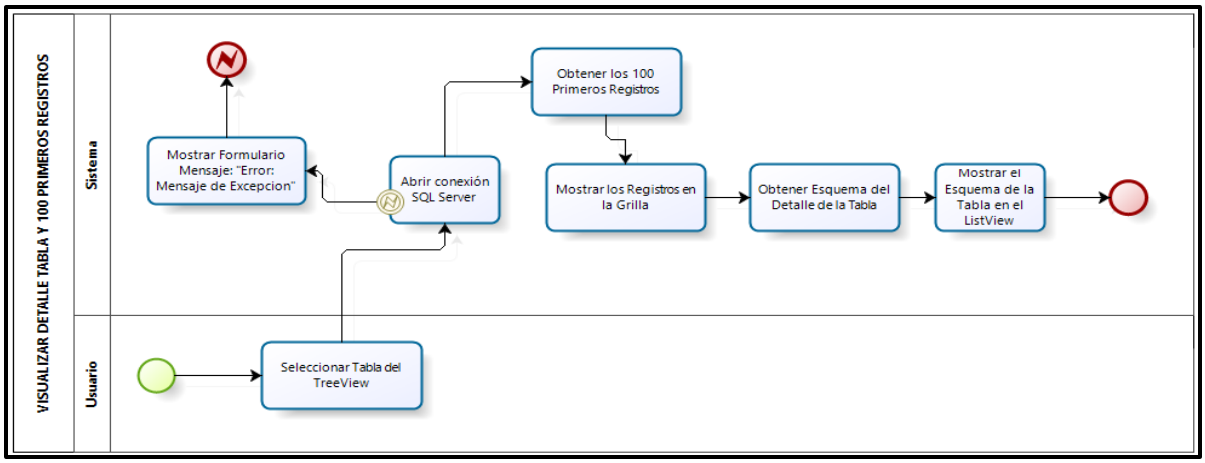
Tabla 21: Detalle de caso de uso visualizar registro datos

ID:	CU14
Caso de Uso:	Visualizar Registro Datos
Actor:	Usuario de Sistemas
Descripción:	Este proceso permite mostrar al usuario el detalle de los registros ingresados a por el sistema a la tabla en consulta.
Precondición:	El usuario debe de estar autenticado al sistema.
Flujo de Eventos	
Flujo Principal	
1. El usuario selecciona la tabla del árbol de opciones. 2. El sistema abre la conexión SQL Server. 3. Si existe un error en la conexión, el sistema muestra un formulario con el mensaje: "Error: Mensaje de Excepción". Caso contrario obtiene los 100 primeros registros. 4. El sistema muestra los registros en la grilla.	
Flujo Alternativo de Eventos	
Flujo Secundario	
No aplica	
Post-condición :	No aplica

Fuente: Elaboración propia

3.2.2.55. Diagrama de Actividades: Visualizar Registro Datos

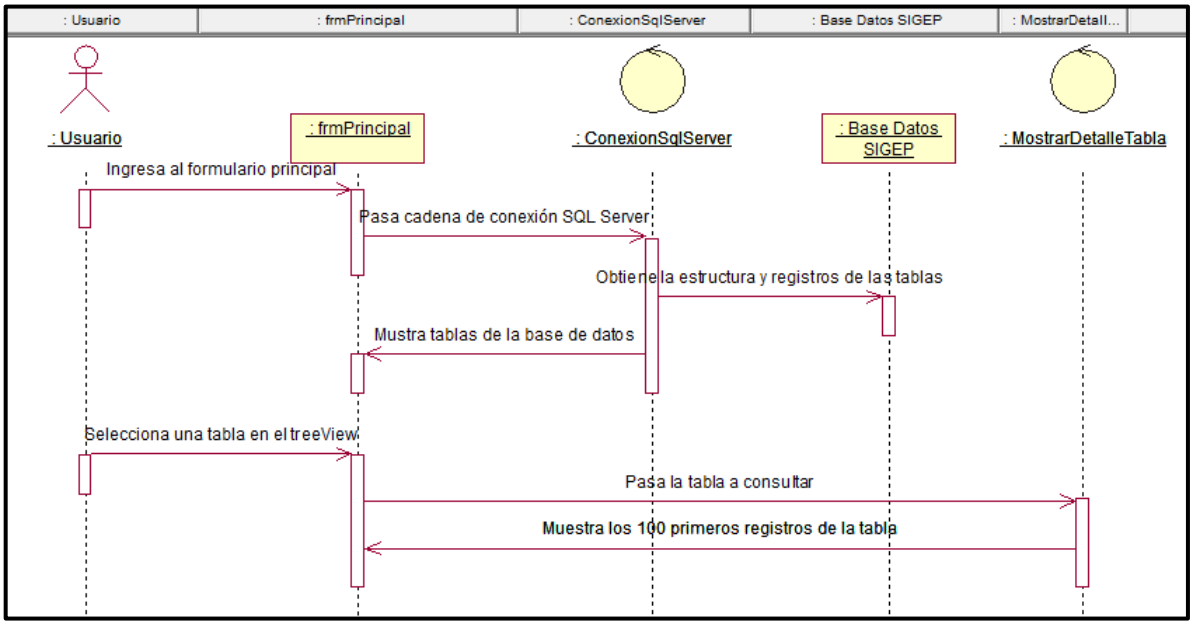
Figura 43: Detalle de diagrama de actividades visualizar registro datos



Fuente: Elaboración propia

3.2.2.56. Diagrama de Secuencias: Visualizar Registro Datos

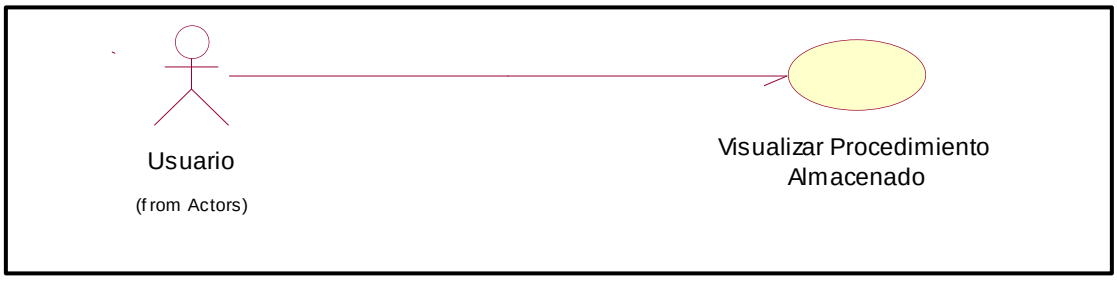
Figura 44: Diagrama de secuencia Visualizar registro datos



Fuente: Elaboración propia

3.2.2.57. Caso de Uso Visualizar Procedimiento Almacenado

Figura 45: Diagrama caso de uso visualizar procedimiento almacenado



Fuente: Elaboración propia

3.2.2.58. Descripción Caso de Uso: Visualizar Procedimiento Almacenado

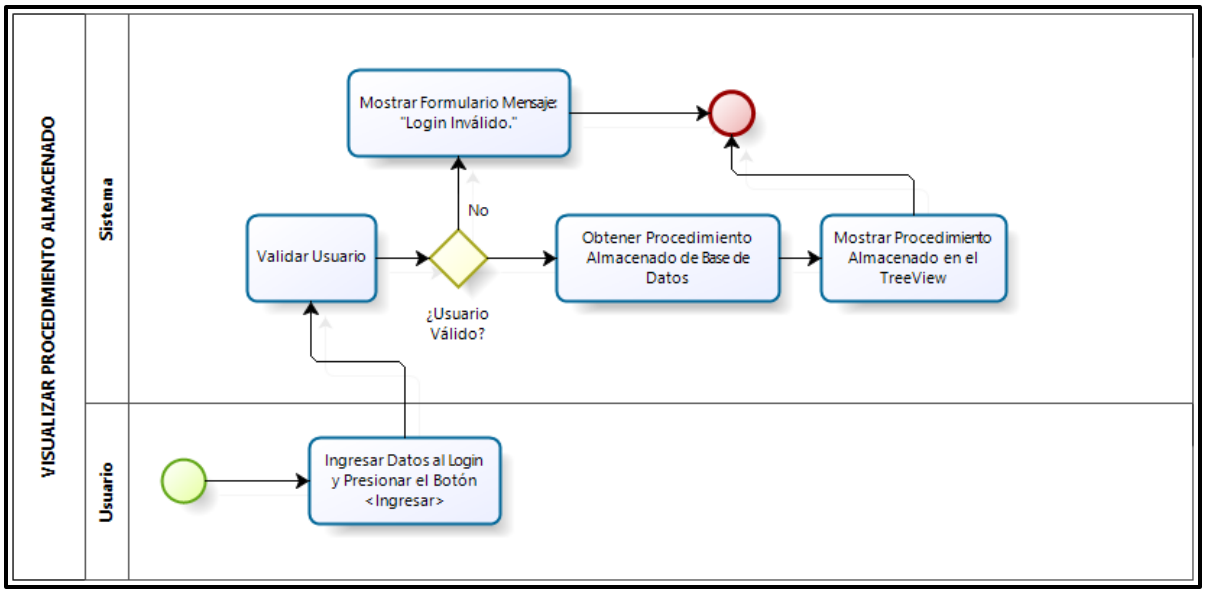
Tabla 22: Detalle de caso de uso visualizar procedimiento almacenado

ID:	CU15
Caso de Uso:	Visualizar Procedimiento Almacenado
Actor:	Usuario de Sistemas
Descripción:	Este proceso permite visualizar todos los procedimientos almacenados que se encuentran creados en la base de datos.
Precondición:	No aplica
Flujo de Eventos	
Flujo Principal	
1. El usuario debe ingresar datos al login y presionar el botón <Ingresar>. 2. El sistema valida al Usuario. 3. Si el usuario es válido, entonces el sistema obtiene los procedimientos almacenados. Caso contrario el sistema muestra un formulario con el mensaje: "Login Inválido." Y sale del sistema. 4. El sistema muestra los procedimientos almacenados en un TreeView del formulario principal.	
Flujo Alternativo de Eventos	
Flujo Secundario	
No aplica	
Post-condición :	No aplica

Fuente: Elaboración propia

3.2.2.59. Diagrama de Actividades: Visualizar Procedimiento Almacenado

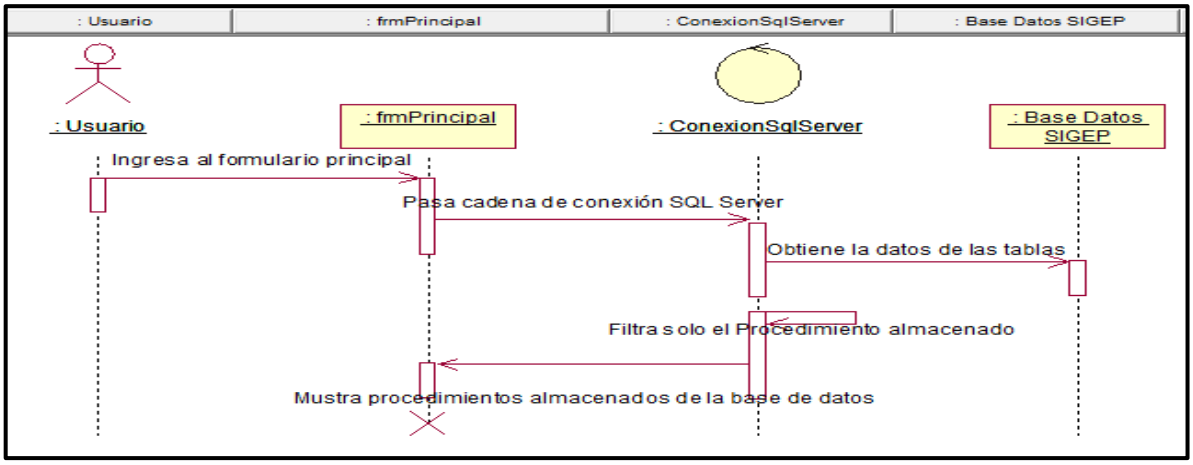
Figura 46: Detalle de diagrama de actividades visualizar procedimiento almacenado



Fuente: Elaboración propia

3.2.2.60. Diagrama de Secuencias: Visualizar Procedimiento Almacenado

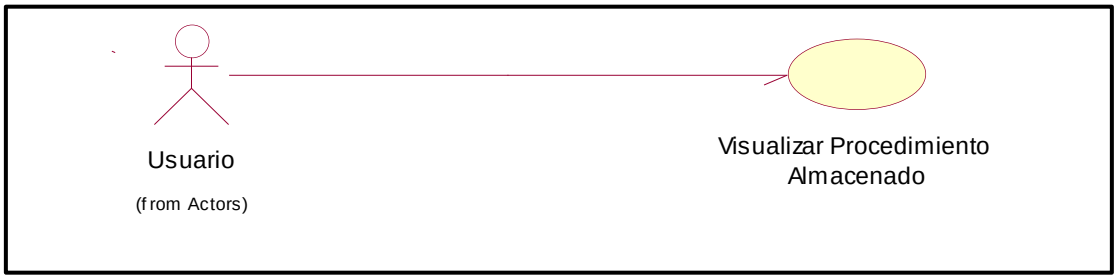
Figura 47: Diagrama de secuencia Visualizar procedimiento



Fuente: Elaboración propia

3.2.2.61. Caso de Uso Visualizar Detalle Procedimiento Almacenado

Figura 48: Diagrama caso de uso visualizar detalle procedimiento almacenado.



Fuente: Elaboración propia

3.2.2.62. Descripción Caso de Uso: Visualizar Detalle Procedimiento Almacenado

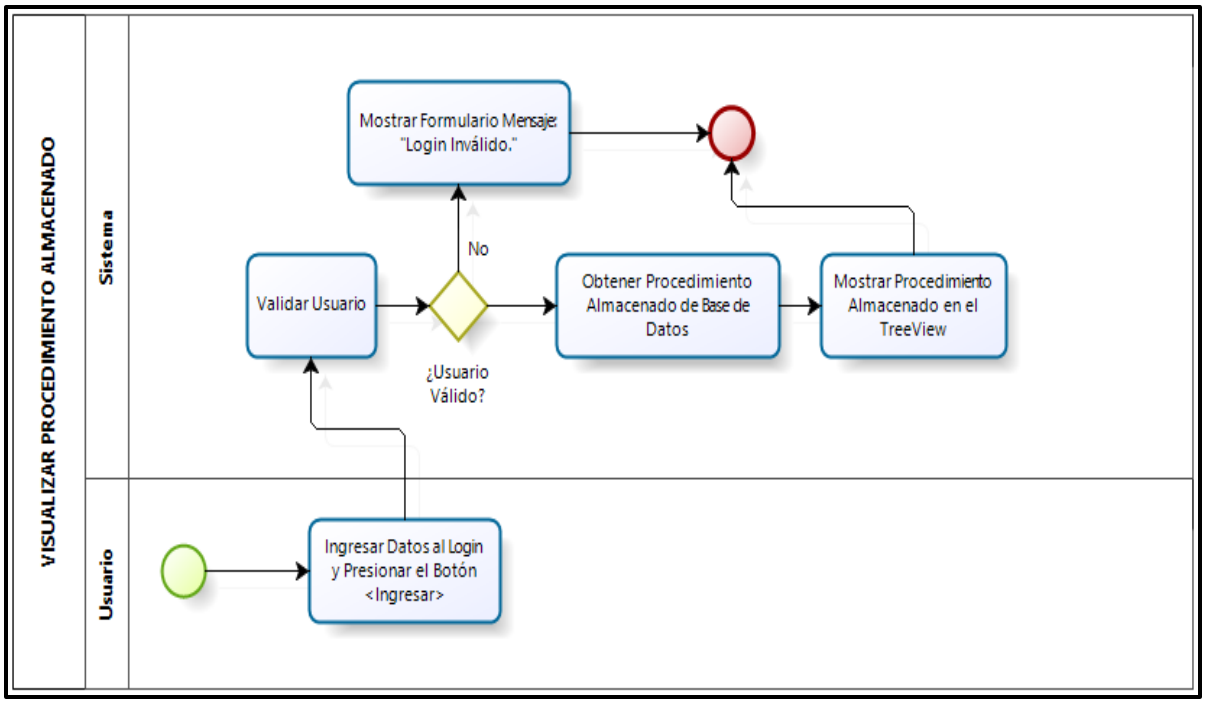
Tabla 23: Detalle de caso de uso visualizar detalle procedimiento almacenado

ID:	CU16
Caso de Uso:	Visualizar Detalle Procedimiento Almacenado
Actor:	Usuario de Sistemas
Descripción:	Este proceso permite al usuario visualizar el detalle del procedimiento almacenado que ya se encuentra creado en base de datos.
Precondición:	El usuario debe estar autenticado en el sistema.
Flujo de Eventos	
Flujo Principal	
1. El usuario selecciona un procedimiento almacenado del treeview. 2. El sistema abre la conexión a Sql Server. 3. Si existe un error de conexión, el sistema muestra un formulario con el mensaje: "Error: Mensaje de Excepción". Caso contrario obtiene el esquema del procedimiento almacenado. 4. Muestra el detalle del procedimiento almacenado en el texto del panel1	
Flujo Alternativo de Eventos	
Flujo Secundario	
No aplica	
Post-condición :	No aplica

Fuente: Elaboración propia

3.2.2.63. Diagrama de Actividades: Visualizar Detalle Procedimiento Almacenado

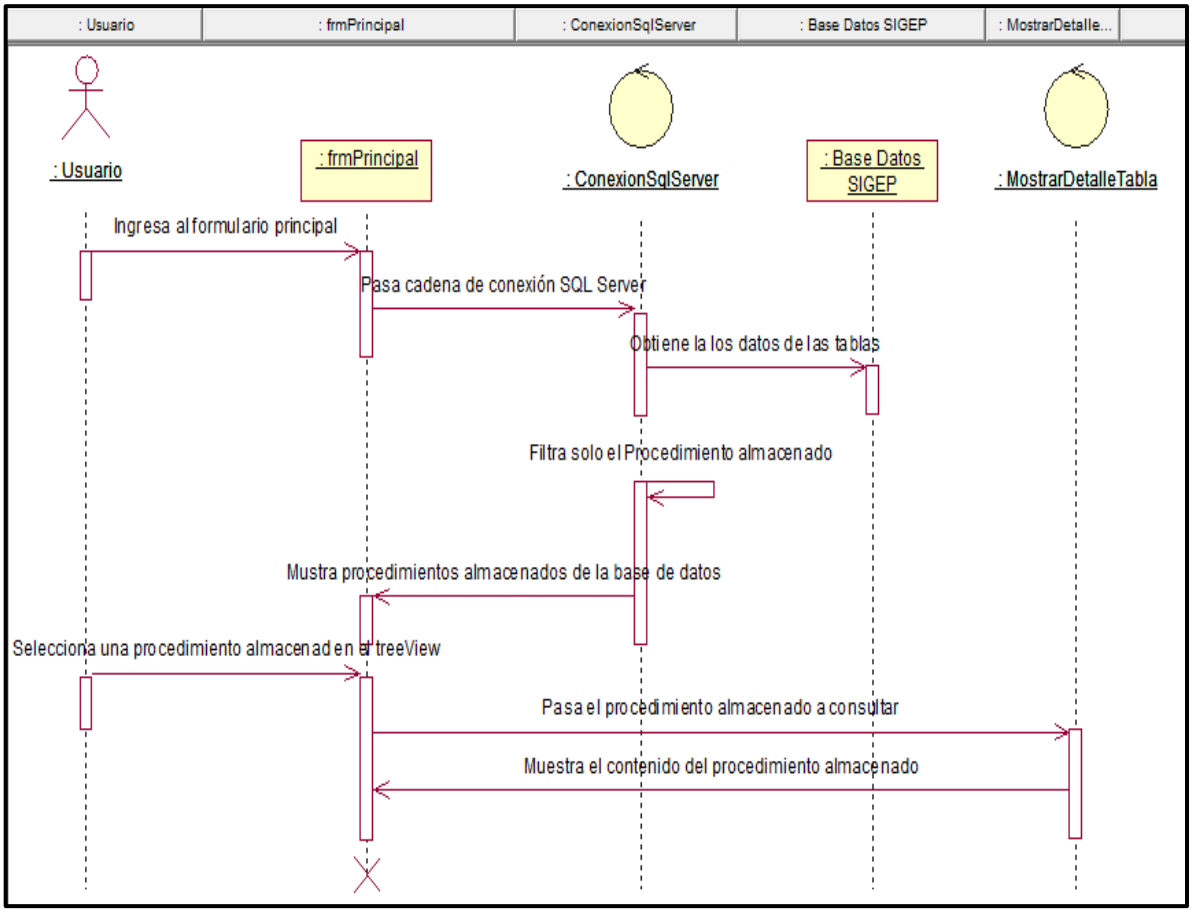
Figura 49: Detalle de diagrama de actividades visualizar detalle procedimiento almacenado



Fuente: Elaboración propia

3.2.2.64. Diagrama de Secuencias: Visualizar Detalle Procedimiento Almacenado

Figura 50: Diagrama de secuencia Visualizar detalle procedimiento almacenado



Fuente: Elaboración propia

3.2.3. Prototipos de las Pantallas de Caso de Uso del Sistema

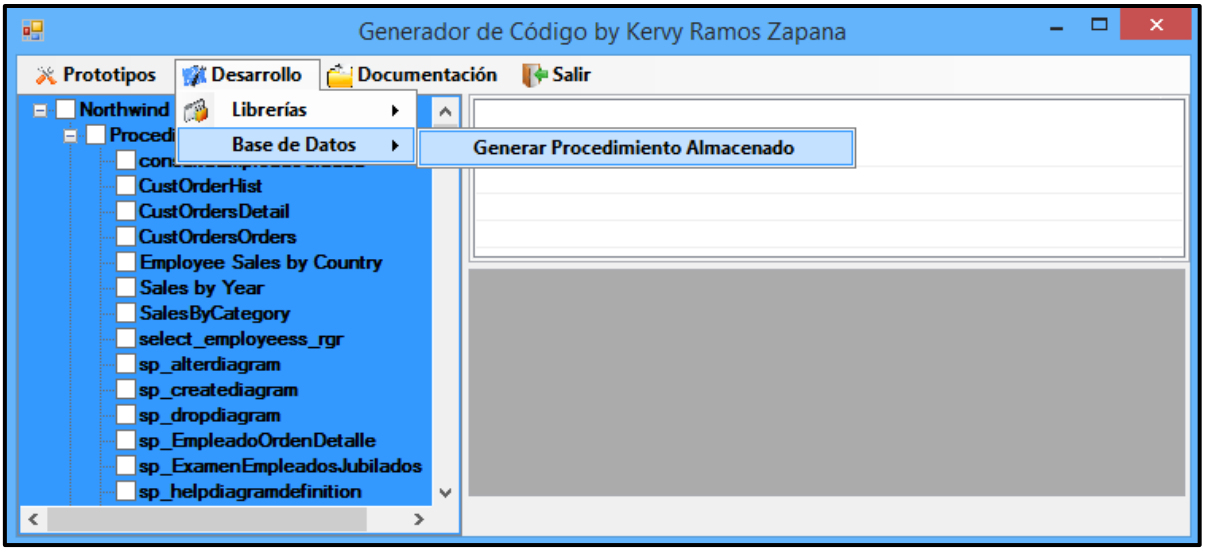
3.2.3.1. Autenticar Usuario

Figura 51: Prototipo autenticar usuario

Fuente: Elaboración propia

3.2.3.2. Generar Procedimiento Almacenado

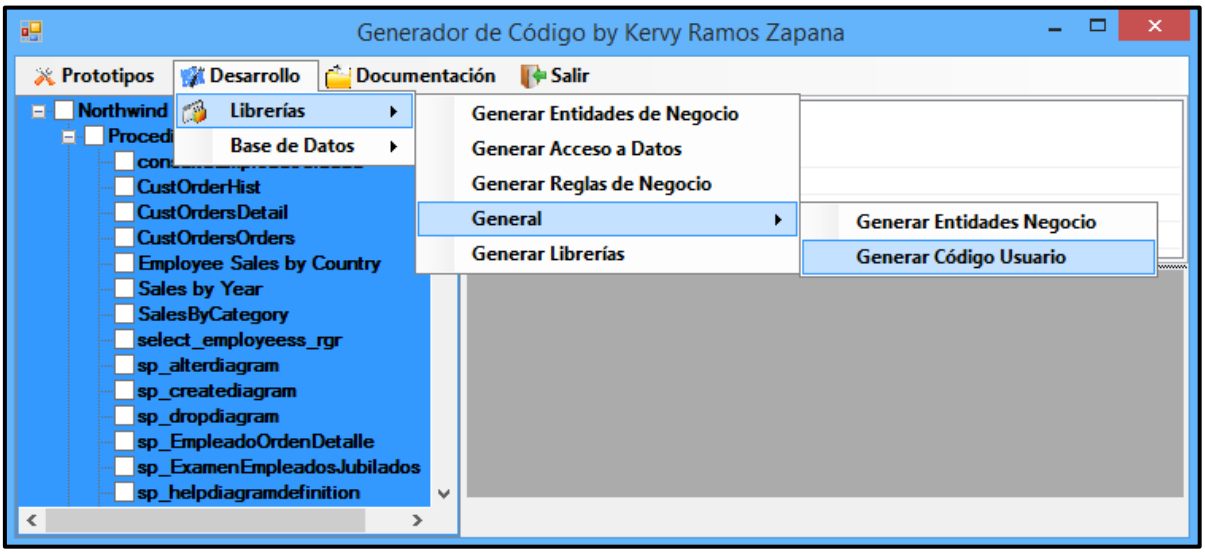
Figura 52: Prototipo generar procedimiento almacenado



Fuente: Elaboración propia

3.2.3.3. Generar Código de Usuario

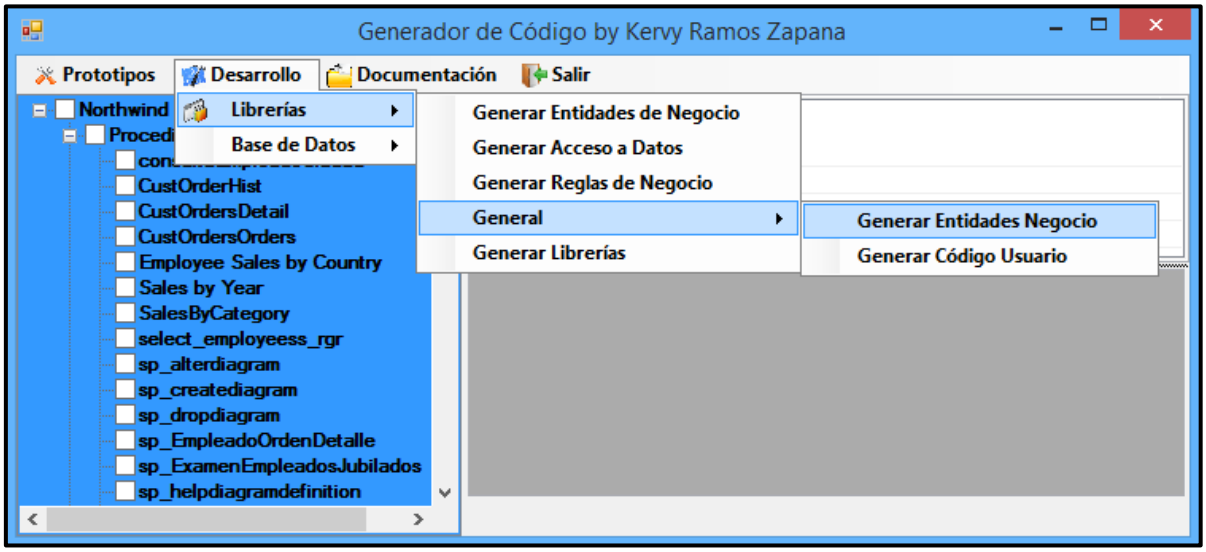
Figura 53: Prototipo generar código de usuario



Fuente: Elaboración propia

3.2.3.4. Generar Entidad de Negocio General

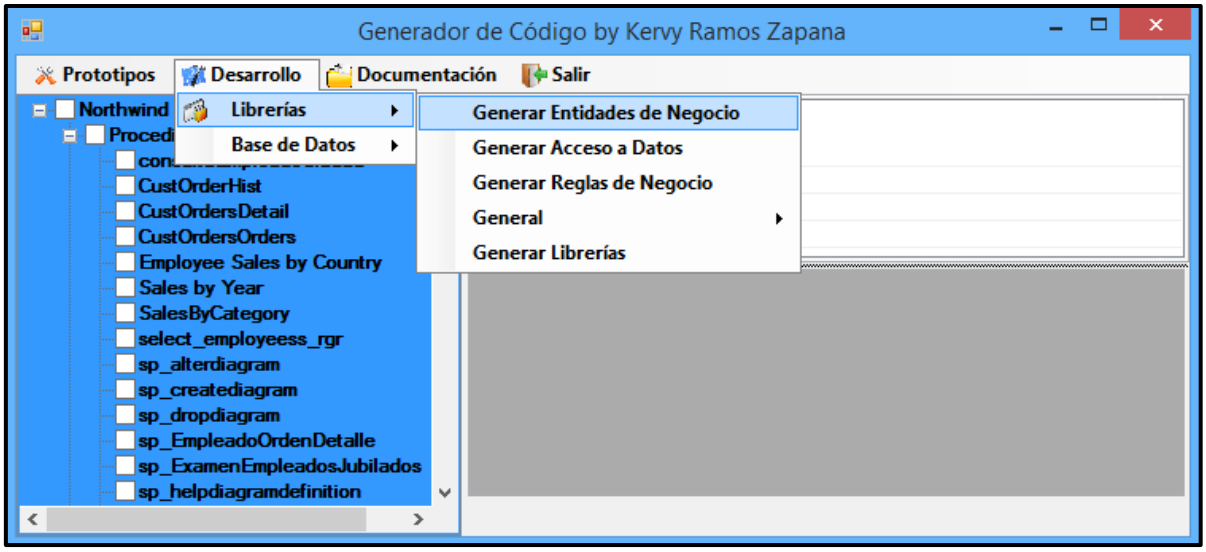
Figura 54: Prototipo generar entidad de negocio general



Fuente: Elaboración propia

3.2.3.5. Generar Entidad de Negocio

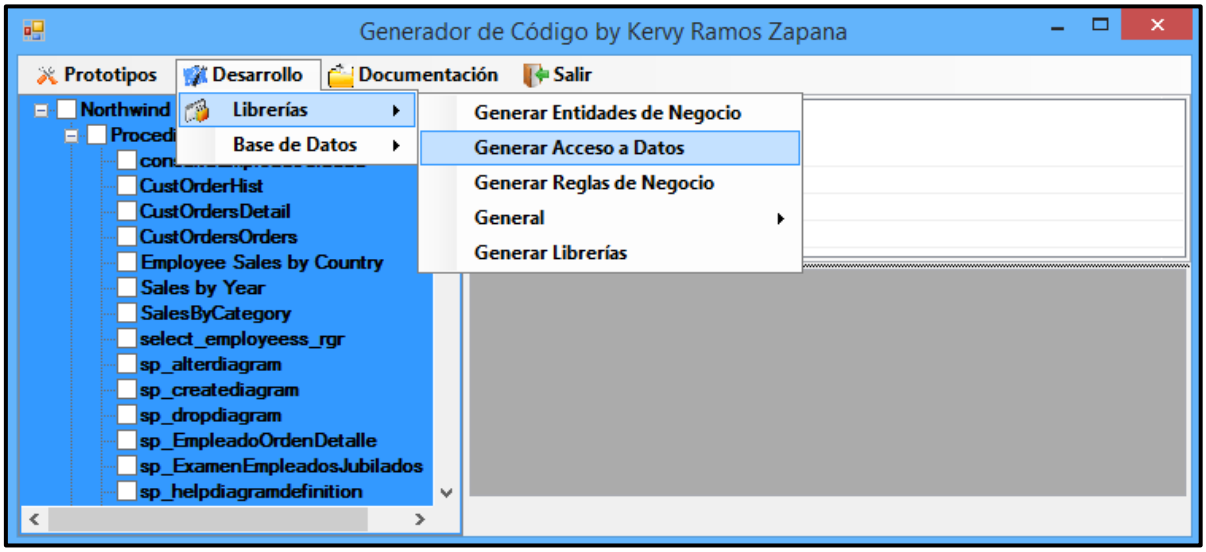
Figura 55: Prototipo generar entidad de negocio



Fuente: Elaboración propia

3.2.3.6. Generar Acceso a Datos

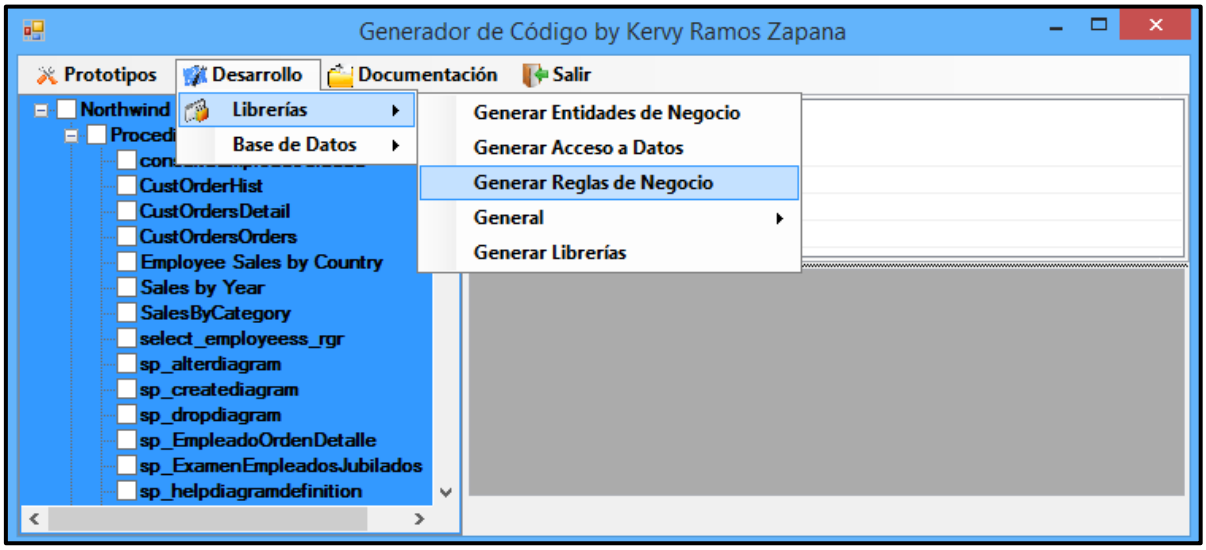
Figura 56: Prototipo generar acceso a datos



Fuente: Elaboración propia

3.2.3.7. Generar Regla de Negocio

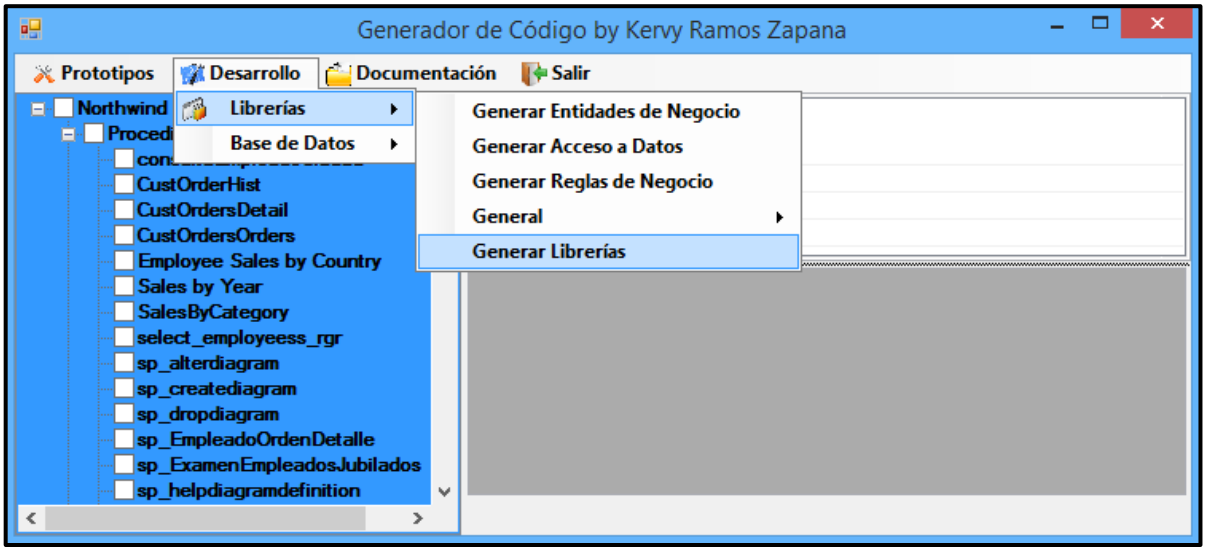
Figura 57: Prototipo generar regla de negocio



Fuente: Elaboración propia

3.2.3.8. Generar Librerías

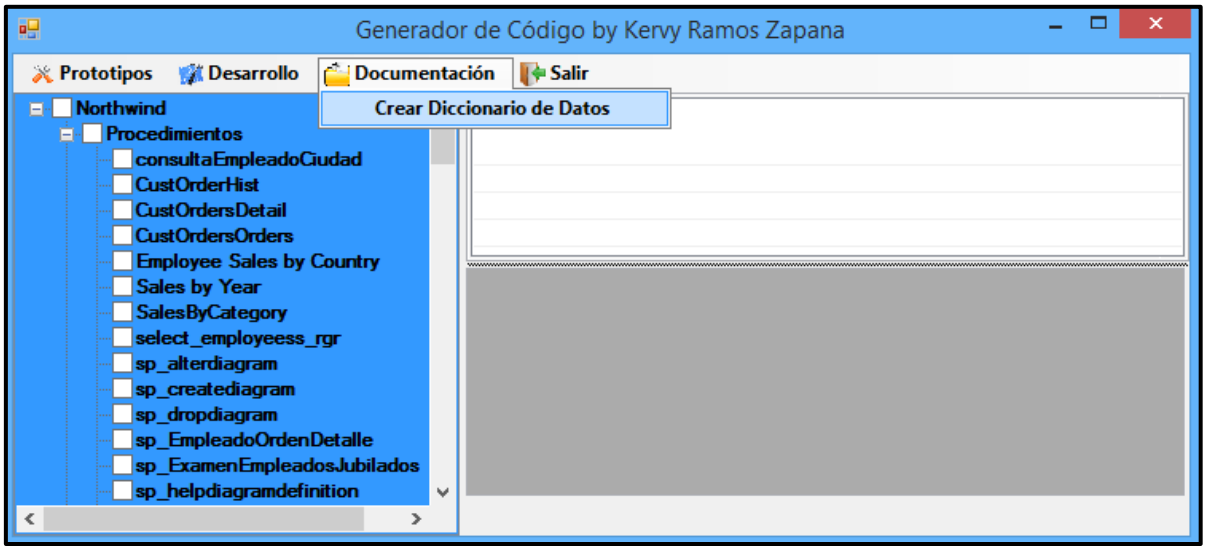
Figura 58: Prototipo generar librerías



Fuente: Elaboración propia

3.2.3.9. Generar Diccionario de Datos

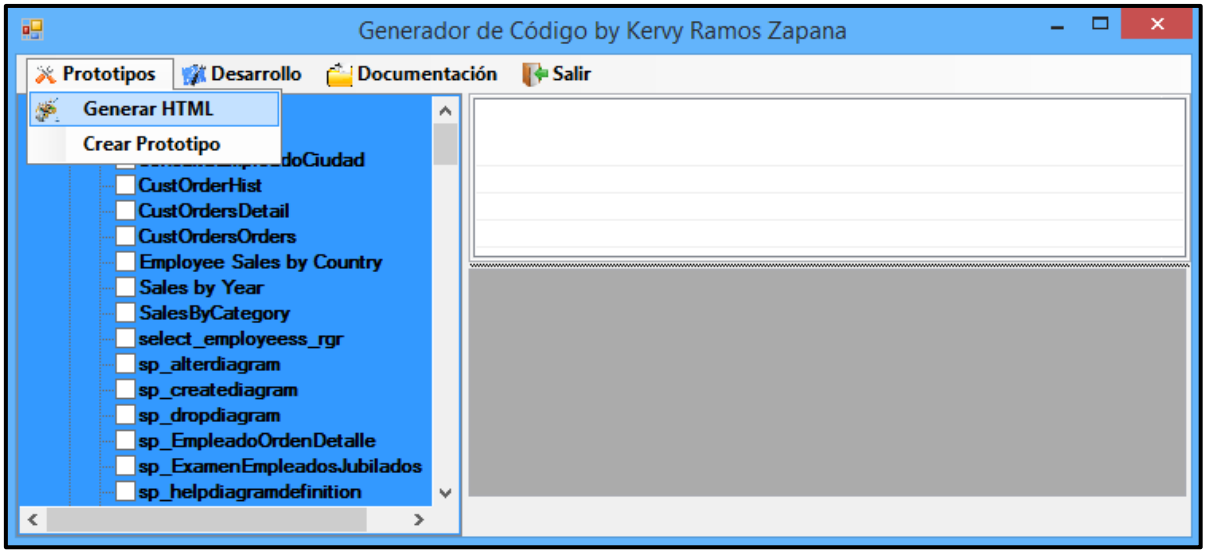
Figura 59: Prototipo generar diccionario de datos



Fuente: Elaboración propia

3.2.3.10. Generar HTML

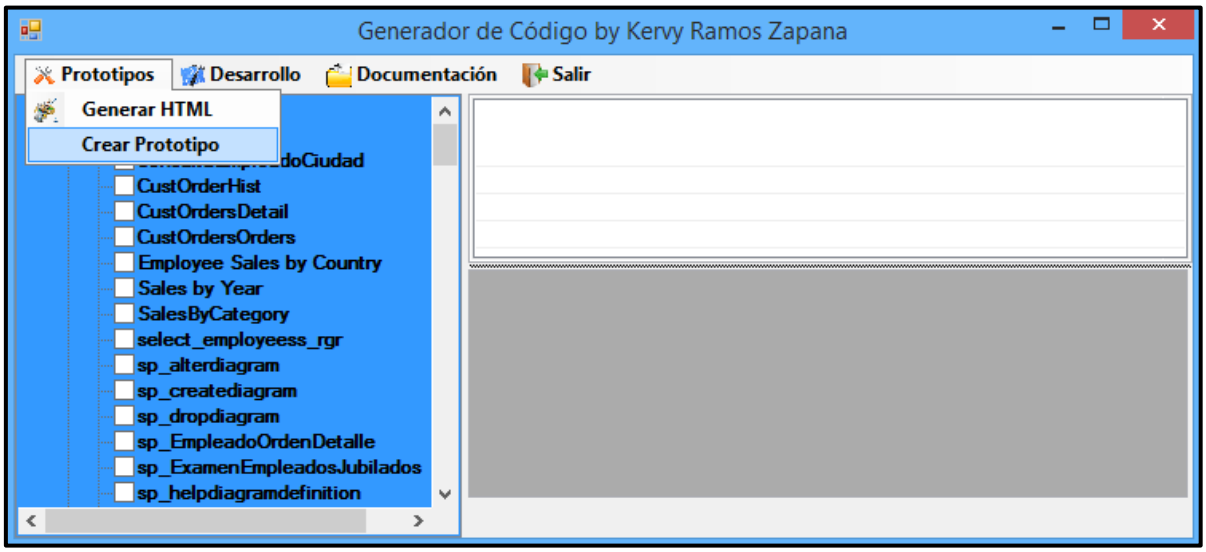
Figura 60: Prototipo generar HTML



Fuente: Elaboración propia

3.2.3.11. Crear Prototipo

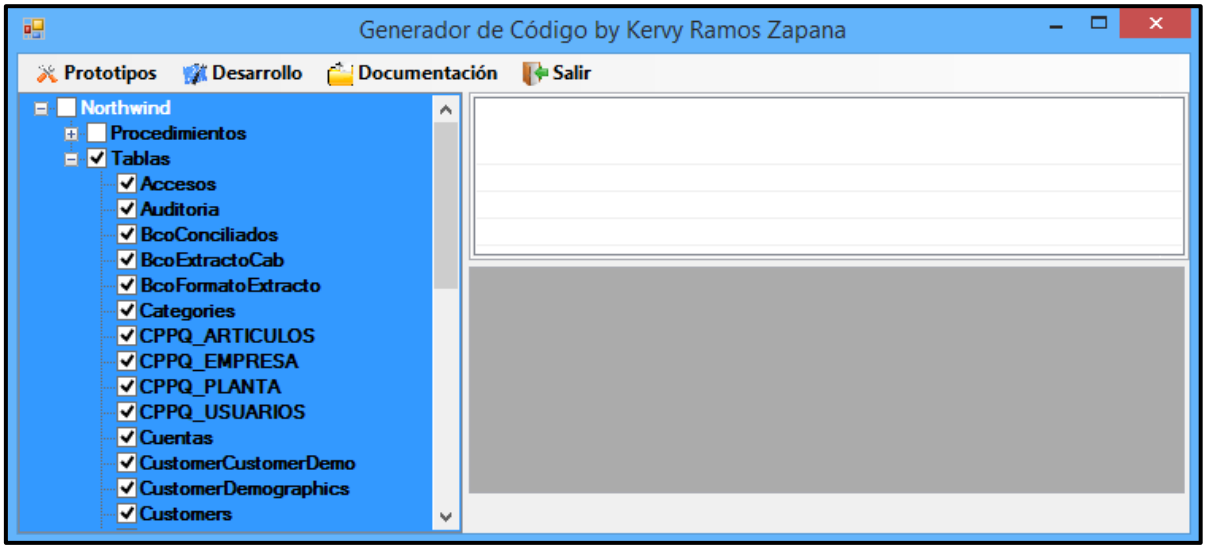
Figura 61: Prototipo crear prototipo



Fuente: Elaboración propia

3.2.3.12. Visualizar Tabla

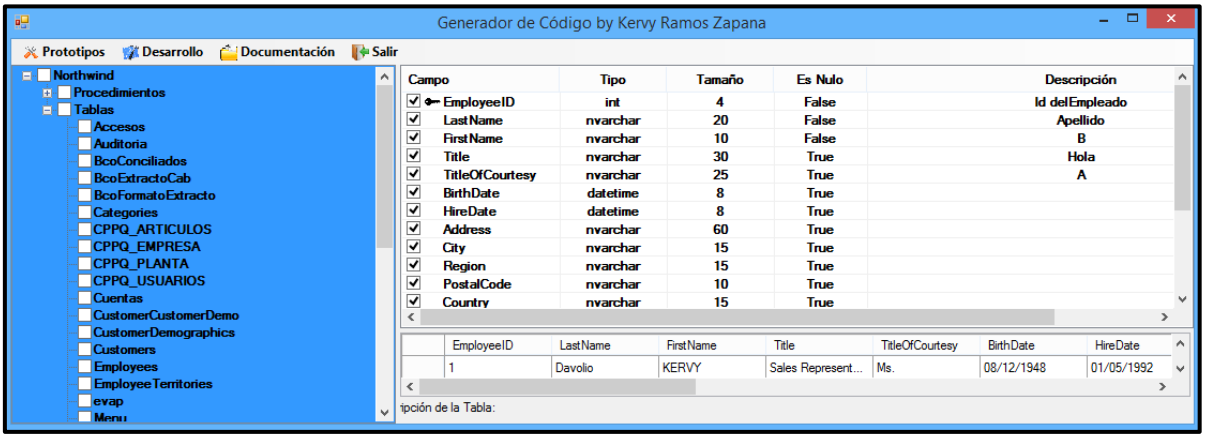
Figura 62: Prototipo visualizar tabla



Fuente: Elaboración propia

3.2.3.13. Visualizar Detalle Tabla

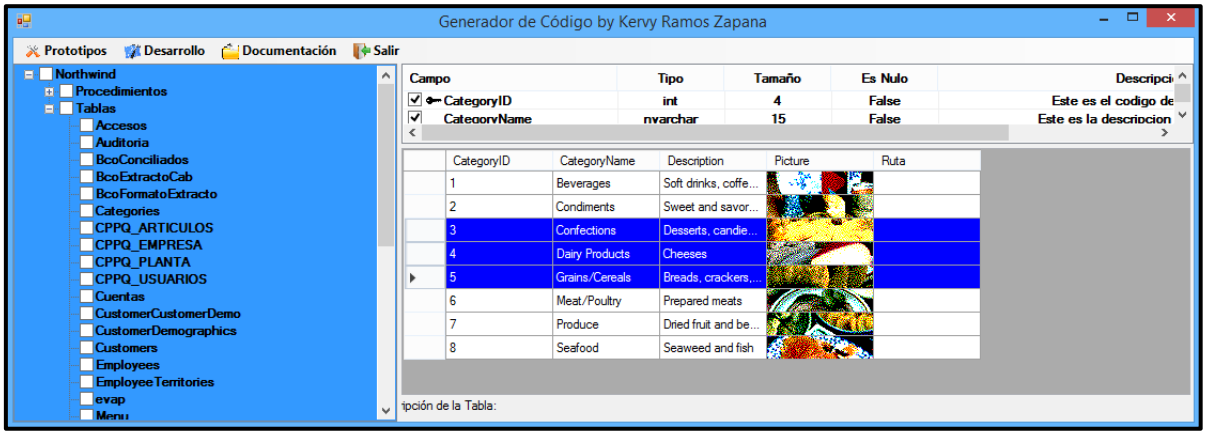
Figura 63: Prototipo visualizar detalle tabla



Fuente: Elaboración propia

3.2.3.14. Visualizar Registro Datos

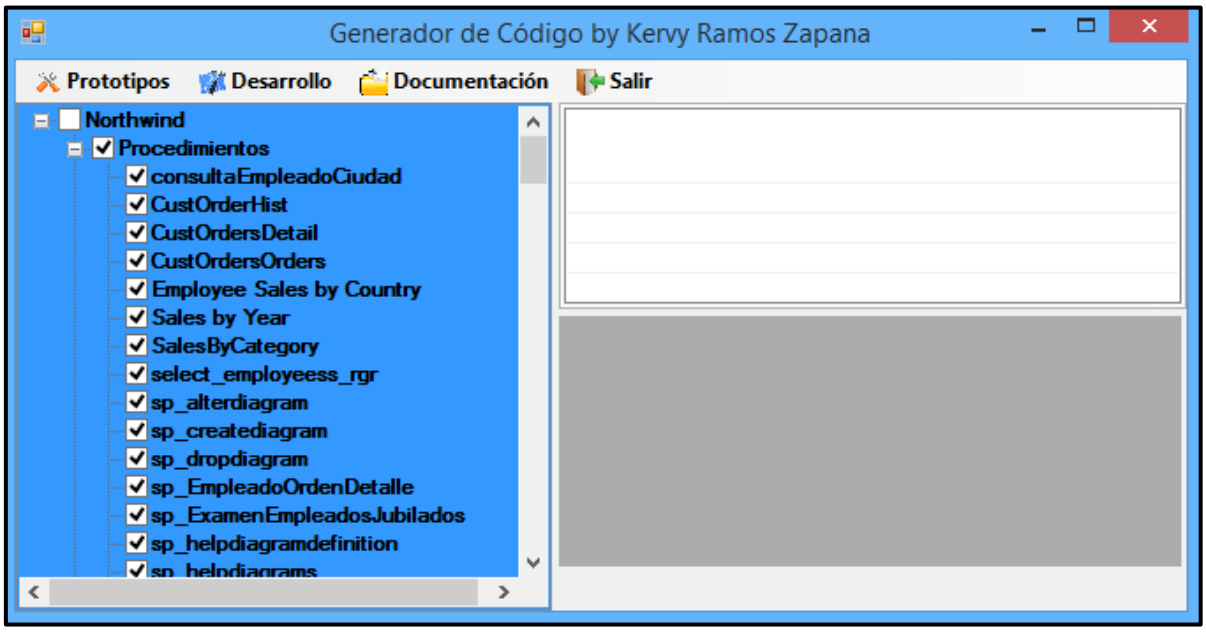
Figura 64: Prototipo visualizar registro datos



Fuente: Elaboración propia

3.2.3.15. Visualizar Procedimiento Almacenado

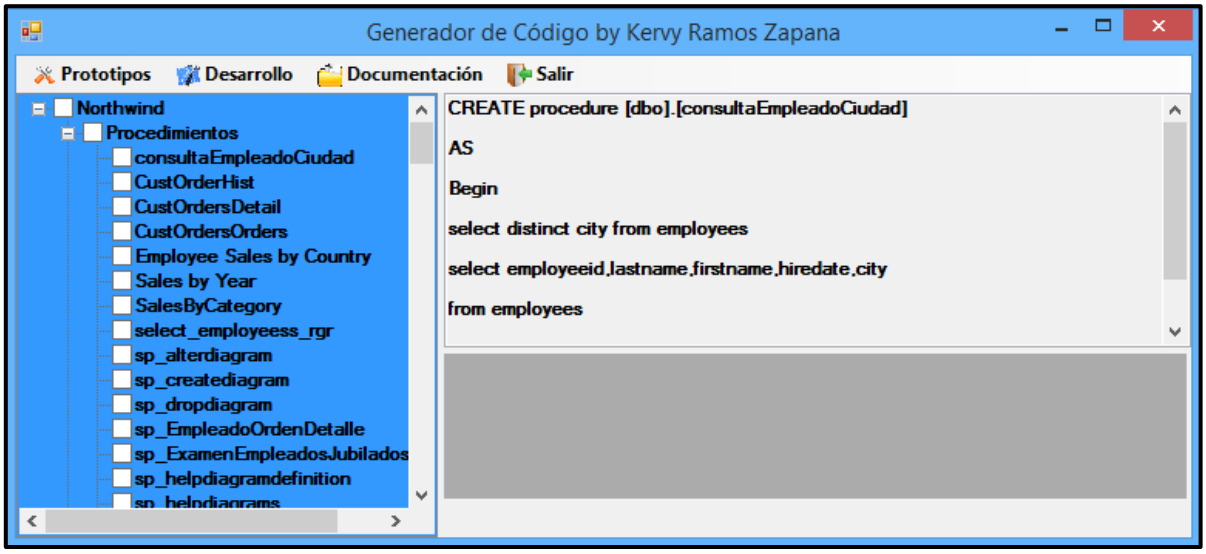
Figura 65: Prototipo visualizar procedimiento almacenado



Fuente: Elaboración propia

3.2.3.16. Visualizar Detalle Procedimiento Almacenado

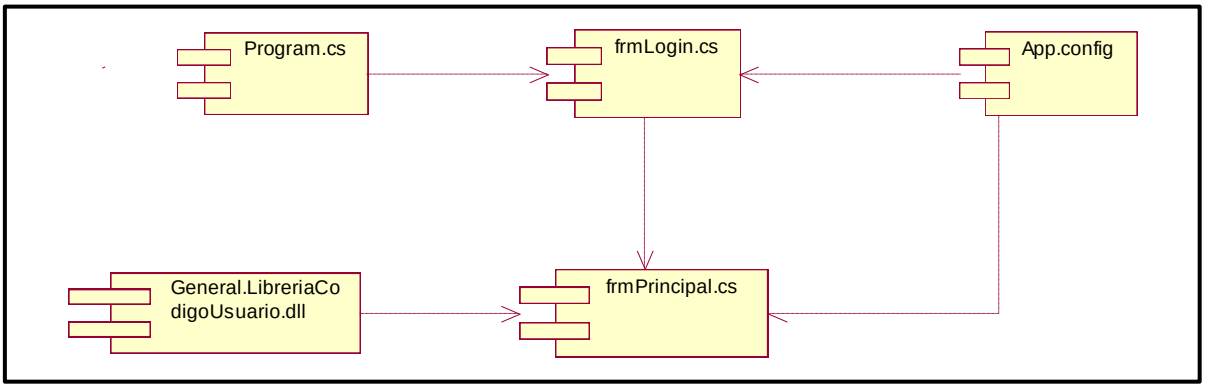
Figura 66: Prototipo visualizar detalle procedimiento almacenado



Fuente: Elaboración propia

3.2.4. Diagrama de Componentes

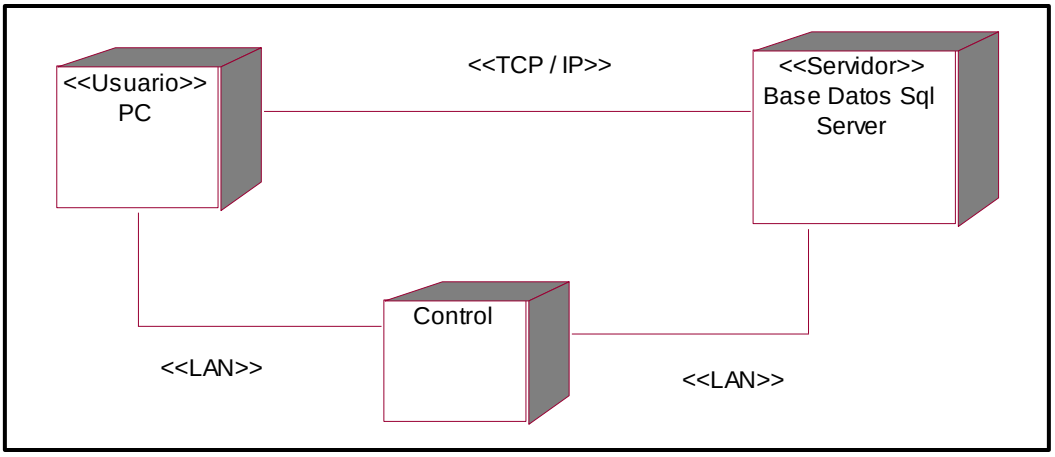
Figura 67: Diagrama de componentes



Fuente: Elaboración propia

3.2.5. Diagrama de Despliegue

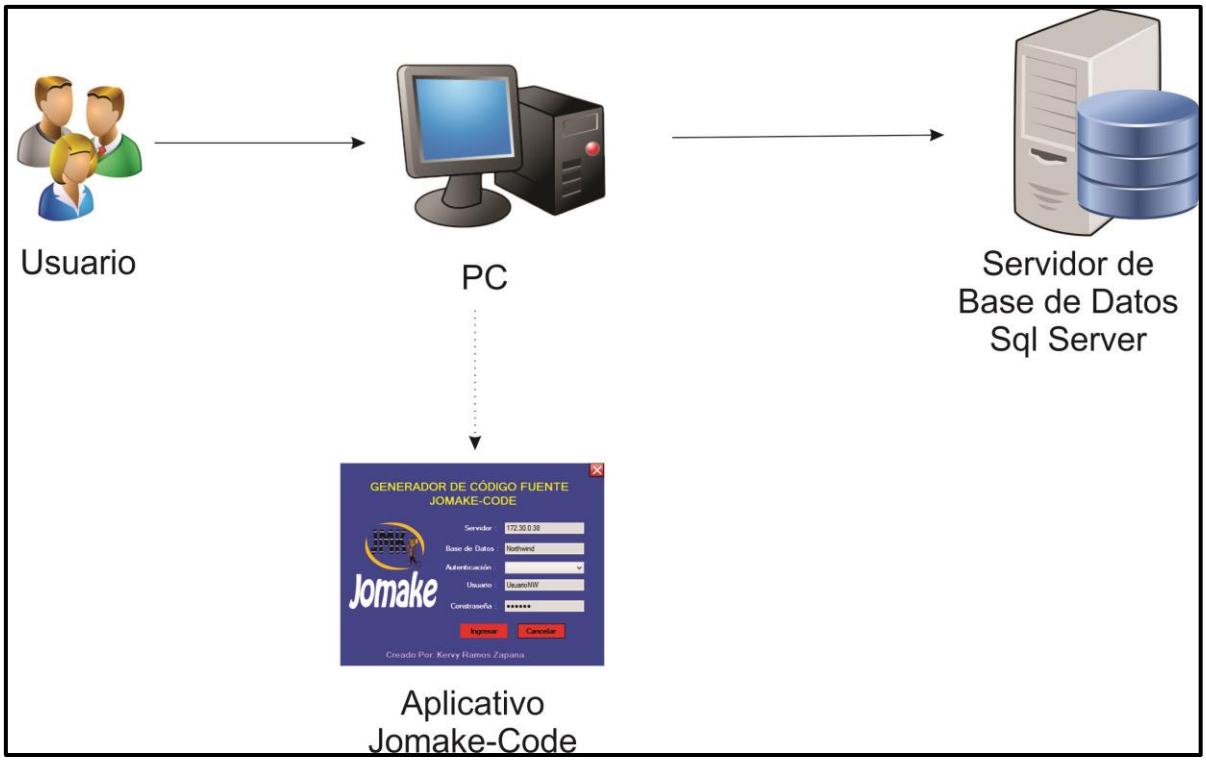
Figura 68: Diagrama de despliegue



Fuente: Elaboración propia

3.2.6. Arquitectura de Solución

Figura 69: Arquitectura de solución



Fuente: Elaboración propia

3.3 Revisión y Consolidación de Resultados

3.3.1 Casos de prueba

Después de haber desarrollador los casos de pruebas de integración como se detalla a continuación en la **Tabla 24.** Se da por aceptado los casos de pruebas para el sistema Jomake-Code generador de código fuente. Realizando con éxito el set de pruebas de las siguientes casuísticas:

Tabla 24: Casos de prueba de integración

CASOS DE PRUEBA		
EMPRESA:	**PROCAMPO**	
LLENADO POR:	**VICTOR CARBAJAL ALCARRAS**	
CARGO EN LA EMPRESA:	**JEFE DE SISTEMAS**	
FECHA:	**02/07/2016**	
TIPO DE OPERACIÓN	**CASO**	**ESTADO**
Autenticar Usuario	Autenticación de windows y/o integrada	APROBADO
	Autenticación de Sql Server	APROBADO
Generar Procedimiento Almacenado	1 tabla	APROBADO
	5 tablas	APROBADO

	todas las tablas	APROBADO
Generar Código de Usuario	Generar código de usuario	APROBADO
Generar Entidad de Negocio General	Generar entidad de negocio general	APROBADO
Generar Entidad de Negocio	1 tabla	APROBADO
	5 tablas	APROBADO
	todas las tablas	APROBADO
Generar Acceso a Datos	1 tabla	APROBADO
	5 tablas	APROBADO
	todas las tablas	APROBADO
Generar Regla de Negocio	1 tabla	APROBADO
	5 tablas	APROBADO
	todas las tablas	APROBADO
Generar Librerías	Generar librerías	APROBADO
Generar Diccionario de Datos	1 tabla	APROBADO
	5 tablas	APROBADO
	todas las tablas	APROBADO
Generar HTML	1 tabla	APROBADO
	5 tablas	APROBADO
	todas las tablas	APROBADO
Crear Prototipo	1 tabla	APROBADO
	5 tablas	APROBADO
	todas las tablas	APROBADO
Visualizar Tabla	Visualizar todas las tablas de la base de datos	APROBADO
Visualizar Detalle Tabla	Visualizar detalle de una tabla	APROBADO

Visualizar Registro Datos	Visualizar registro datos de una tabla	APROBADO
Visualizar Procedimiento Almacenado	Visualizar todos los procedimiento almacenado de una base de datos	APROBADO
Visualizar Detalle Procedimiento Almacenado	Visualizar e detalle de un procedimiento almacenado	APROBADO

Fuente: Elaboración propia

3.4.1 Ventajas y Beneficios

El presente cuadro comparativo muestra las ventajas y beneficios que se obtuvo después de la implementación del Sistema Jomake-Code para la generación del código fuente. Así mismo deja demostrado la eficiencia operativa del proceso automático frente al proceso actual que se venía desarrollando antes de la implementación del nuevo sistema.

Los datos referenciales que se utilizaron en la muestra para las pruebas de ejecución y performance de los procesos del sistema se detallan a continuación:

- Nombre de base de datos: SIGEP_FEB16
- Número total de procedimientos almacenados: 1718
- Número total de tablas: 503

Para las pruebas de mantenimiento (Listar, Registrar, Modificar, Eliminar y Filtrar) para las tablas maestro de Empleado, Producto, Proveedor y Cliente afectan las tablas:

- Empleado: Se almacena los datos de los empleados.
- Invmae: Se almacena los datos de los productos.
- Promae: Se almacena los datos de los proveedores.
- Climae: Se almacena los datos de los clientes.

Definido ya los datos para las pruebas comparativas llegamos al siguiente cuadro que resumen el detalle de los procesos.

Tabla 25: Cuadro comparativo de procesos

Nombre Proceso	Proceso Anterior	Descripción Proceso Anterior	Duración del Proceso Anterior	Proceso Actual	Descripción Proceso Actual	Duración del Proceso Actual	Ventajas y Beneficios
Generación de procedimientos almacenados	Proceso manual	Se desarrollaba creando por cada función de mantenimiento un procedimiento almacenado	3600,0000 s	Proceso automático	Permite crear las funciones principales de mantenimiento de forma automática	48,3305 s	Reducción de tiempo Reducción de errores Tiempo de respuesta mínima
Generación de la estructura de capa de datos	Proceso manual	Se creaban las capa de datos de forma manual de todas las tablas involucradas en el mantenimiento	18000,0000 s	Proceso automático	Generador de código fuente de capa de datos	159,3731 s	Reducción de tiempo Reducción de errores Alto rendimiento en la performance Estandarización de código fuente
Generación de prototipos de pantalla	Proceso manual	Se diseñaba personalmente a través de un software de prototipos	4800 s	Proceso automático	Diseñador de prototipos en formato JPG	55,6847 s	Reducción de tiempo Tiempo de respuesta mínima
Generación de diccionario de datos	No existe	No existe	No existe	Proceso automático	Crea un documento sobre el detalle de las tablas de base de datos	108,9463 s	Reducción de tiempo

Fuente: Elaboración propia

CONCLUSIÓN

- Después de analizar la revisión y consolidación de resultados se pudo llegar a la conclusión que en el Proceso Anterior el tiempo de desarrollo que tomaba para poder realizar cada proceso tiene un tiempo muy elevado a comparación con el Nuevo Proceso. Con la implementación del Nuevo Sistema Jomake-Code podemos ver que existe una mejora de solución ya que los procesos ahora son de forma automática y con un tiempo de respuesta inmediata. Lo cual indica que se ha optimizado el proceso minimizando los tiempos y maximizando la productividad.

- Con el desarrollo del sistema Jomake-code se ha podido lograr reducir el tiempo de diseño de prototipos a su vez se ha logrado generar de forma automática los prototipos para la etapa de diseño y también la implementación del proceso generar código HTML para el entorno de desarrollo Web.

- Al realizar el proceso de generación de las capas de datos (entidad de negocio, acceso a datos y regla de negocio) el sistema ha logrado permitir estandarizar el desarrollo de software creando así un estándar de programación basado en las buenas prácticas para el desarrollo de software, reduciendo errores en codificación, minimizando tiempos y generar gran nivel de performance, ya que el desarrollo está orientado a trabajar con objetos y listas de objetos.

- Con este proyecto se logró el objetivo de mejorar el proceso de la etapa de desarrollo de software permitiendo reducir tiempos, reducción de errores al momento de codificar, generar alto rendimiento en la performance y poder estandarizar todos los proyectos en general utilizando las buenas prácticas de la programación.

RECOMENDACIÓN

- El sistema Jomake-Code está orientado a la programación nativa, es decir desarrollar sistemas con código netamente del lenguaje de programación .NET como por ejemplo utilizar ADO .NET y no un framework que haga la conexión a base de datos.

- El sistema en la actualidad trabaja con el motor de base de datos de Sql Server, pero se puede adaptar al motor de base de datos de Oracle como MySql ya que el desarrollo está orientado a la escalabilidad.

- En la etapa de desarrollo de software se recomienda a los programadores utilizar la programación nativa, ya que es muy eficiente para poder aplicar la performance del sistema. Por otro lado, invito a crear sus propios métodos reusables para los sistemas y no usar códigos de terceros ya creados. Eso solo te facilita el trabajo, pero crearlo te va ayudar bastante en conocimiento.

- Como trabajos a futuro en esta etapa de desarrollo, se recomienda incorporar nuevos procesos automatizados como realizar una reversa o un proceso de contingencia que permite reversar procesos anteriores.

BIBLIOGRAFÍA

- José Temprado. (2010). Generador de codigo MyWay. Sitio web: http://www.thempra.net/2010/09/generador-de-codigo-myway/

- Blake Niemyjski. (2015). CodeSmith Generator. Sitio web: https://visualstudiogallery.msdn.microsoft.com/4fbfd05a-f3e8-4f79-b912-def3e9cb28fc.

- Israel Romero. (2014). Generador de código para tu capa de datos, negocio y servicios Restful. Sitio web: http://vidaencodigo.blogspot.pe/2014/09/generador-de-codigo-para-tu-capa-de.html.

- Chávez Reina, Eduardo René. (2012). Análisis, diseño y desarrollo de un generador de código fuente para gestión de información de MySQL, SQL Server y Access para los lenguajes Java, PHP y ASP. Sitio web: http://repositorio.espe.edu.ec/handle/21000/5906.

- Omar Pineda. (2013). Generador de código fuente COBOL. Sitio web: http://slideplayer.es/slide/5824832/

- Cristina M. (2014). ¿Qué son los Frameworks?. Sitio web: http://blog.nubelo.com/que-son-los-frameworks/

- Luis Dueñas. (2013). Libro la Biblia parte I. Lima, Perú: Coredise.

- Microsoft Developert Network. (2016). Microsoft SQL Server. Sitio web: https://msdn.microsoft.com/es-es/library/bb545450.aspx

- Mozilla Developer Network. (2016). HTML. Sitio web: https://developer.mozilla.org/es/docs/Web/HTML

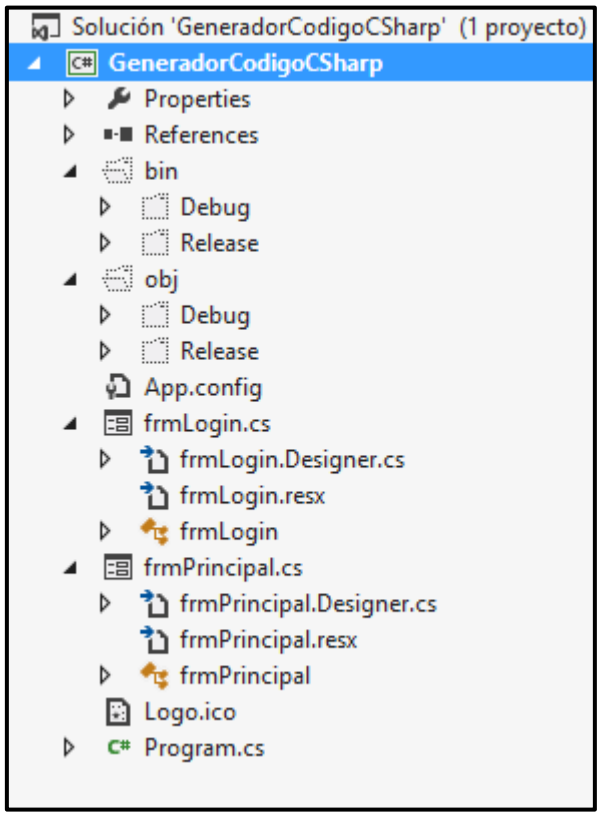
- Isa Arteta. (2013). Metodología de software en cascada. Sitio web: http:// http://modelo-cascada.blogspot.pe

- Yanirys Montes, Yuliesky Brito. (2012). Arquitectura de N Capas. Sitio web: http://www.eumed.net/libros-gratis/2012b/1232/arquitectura-N-capas.html

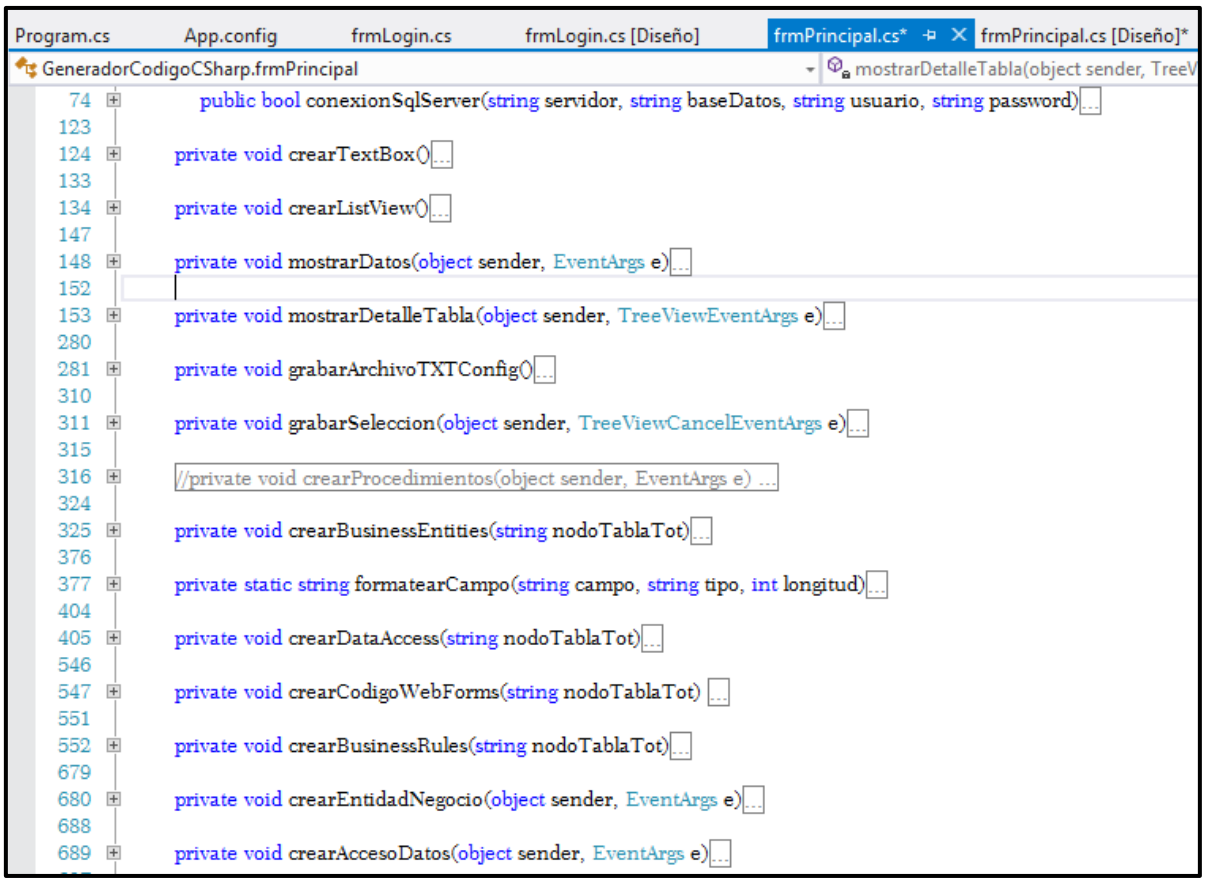
- Microsoft Developer Network. (2016). Stored Procedures. 1, de 1 Sitio web: https://msdn.microsoft.com/en-us/library/ms190782.aspx

ANEXOS

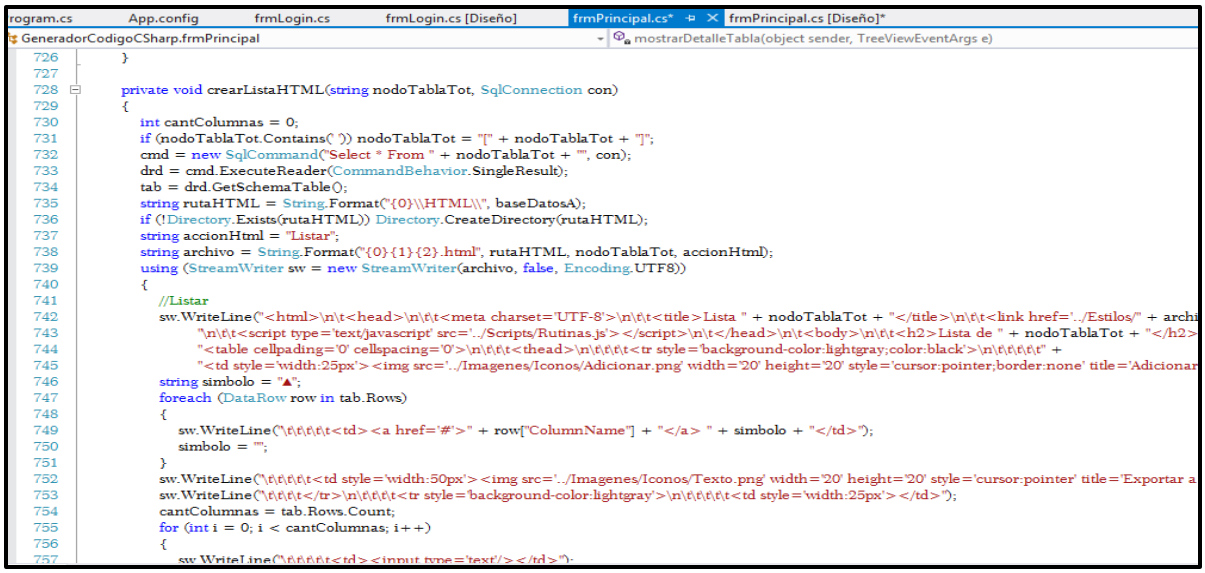
Solución del proyecto Jomake-Code



Métodos del proyecto



Método crear Lista HTML



```
rogram.cs    App.config    frmLogin.cs    frmLogin.cs [Diseño]    frmPrincipal.cs*    frmPrincipal.cs [Diseño]*
GeneradorCodigoCSharp.frmPrincipal    mostrarDetalleTabla(object sender, TreeViewEventArgs e)
726         }
727
728         private void crearListaHTML(string nodoTablaTot, SqlConnection con)
729         {
730             int cantColumnas = 0;
731             if (nodoTablaTot.Contains(' ')) nodoTablaTot = "[" + nodoTablaTot + "]";
732             cmd = new SqlCommand("Select * From " + nodoTablaTot + "", con);
733             drd = cmd.ExecuteReader(CommandBehavior.SingleResult);
734             tab = drd.GetSchemaTable();
735             string rutaHTML = String.Format("{0}\\HTML\\", baseDatosA);
736             if (!Directory.Exists(rutaHTML)) Directory.CreateDirectory(rutaHTML);
737             string accionHtml = "Listar";
738             string archivo = String.Format("{0}{1}{2}.html", rutaHTML, nodoTablaTot, accionHtml);
739             using (StreamWriter sw = new StreamWriter(archivo, false, Encoding.UTF8))
740             {
741                 //Listar
742                 sw.WriteLine("<html>\n\t<head>\n\t\t<meta charset='UTF-8'>\n\t\t<title>Lista " + nodoTablaTot + "</title>\n\t\t<link href='../Estilos/" + archi
743                     "\n\t\t<script type='text/javascript' src='../Scripts/Rutinas.js'></script>\n\t</head>\n\t<body>\n\t\t<h2>Lista de " + nodoTablaTot + "</h2>
744                     "<table cellpading='0' cellspacing='0'>\n\t\t\t<thead>\n\t\t\t\t<tr style='background-color:lightgray;color:black'>\n\t\t\t\t\t" +
745                     "<td style='width:25px'><img src='../Imagenes/Iconos/Adicionar.png' width='20' height='20' style='cursor:pointer;border:none' title='Adicionar
746                 string simbolo = "▲";
747                 foreach (DataRow row in tab.Rows)
748                 {
749                     sw.WriteLine("\t\t\t\t\t<td><a href='#'>" + row["ColumnName"] + "</a> " + simbolo + "</td>");
750                     simbolo = "";
751                 }
752                 sw.WriteLine("\t\t\t\t\t<td style='width:50px'><img src='../Imagenes/Iconos/Texto.png' width='20' height='20' style='cursor:pointer' title='Exportar a
753                 sw.WriteLine("\t\t\t\t</tr>\n\t\t\t\t<tr style='background-color:lightgray'>\n\t\t\t\t\t<td style='width:25px'></td>");
754                 cantColumnas = tab.Rows.Count;
755                 for (int i = 0; i < cantColumnas; i++)
756                 {
757                     sw.WriteLine("\t\t\t\t\t<td><input type='text'/></td>");
```

Método crear entidad de negocio general

```
private void crearEntidadNegocioGeneral(object sender, EventArgs e)
{
    try
    {
        string rutaDA = String.Format("{0}\\Capa de Datos\\General\\BusinessEntities\\", baseDatosA);
        if (!Directory.Exists(rutaDA)) Directory.CreateDirectory(rutaDA);
        string archivo = String.Format("{0}be_Log.cs", rutaDA);
        using (StreamWriter sw = new StreamWriter(archivo, false, Encoding.Default))
        {
            sw.WriteLine("using System;");
            sw.WriteLine("using System.Collections.Generic;");
            sw.WriteLine("using System.Linq;");
            sw.WriteLine("using System.Text;");
            sw.WriteLine("using System.Threading.Tasks;");
            sw.WriteLine("");
            sw.WriteLine("namespace General.Libreria.EntidadesNegocio");
            sw.WriteLine("{");
            sw.WriteLine(tab1 + "public class be_Log");
            sw.WriteLine(tab1 + "{");
            sw.WriteLine(tab2 + "public DateTime FechHora { set; get; }");
            sw.WriteLine(tab2 + "public string Aplicacion { set; get; }");
            sw.WriteLine(tab2 + "public string IP { set; get; }");
            sw.WriteLine(tab2 + "public string Usuario { set; get; }");
            sw.WriteLine(tab2 + "public string NombrePC { set; get; }");
            sw.WriteLine(tab2 + "public string Clase { set; get; }");
            sw.WriteLine(tab2 + "public string Metodo { set; get; }");
            sw.WriteLine(tab2 + "public string MensajeError { set; get; }");
            sw.WriteLine(tab2 + "public string DetalleError { set; get; }");
            sw.WriteLine("");
            sw.WriteLine(tab2 + "public be_Log()");
            sw.WriteLine(tab2 + "{");
            sw.WriteLine(tab3 + "FechHora = DateTime.Now;");
```

Método grabar archivo de texto (.txt)

```
283     private void grabarArchivoTXTConfig()
284     {
285         if (lvwPrincipal.Items.Count > 0)
286         {
287             //Datos a grabar en .txt como registro
288             StringBuilder sb1 = new StringBuilder();
289             string valor;
290             for (int i = 0; i < lvwPrincipal.Items.Count; i++)
291             {
292                 if (i == lvwPrincipal.Items.Count - 1)
293                 {
294                     valor = lvwPrincipal.Items[i].Checked ? "1" : "0";
295                     sb1.Append(String.Format("{0},{1}", lvwPrincipal.Items[i].SubItems[0].Text, valor));
296                 }
297                 else
298                 {
299                     valor = lvwPrincipal.Items[i].Checked ? "1" : "0";
300                     sb1.AppendLine(String.Format("{0},{1}", lvwPrincipal.Items[i].SubItems[0].Text, valor));
301                 }
302             }
303 
304             if (File.Exists(archivoTablas)) File.Delete(archivoTablas);
305             using (StreamWriter sw = new StreamWriter(archivoTablas, false, Encoding.UTF8))
306             {
307                 sw.WriteLine(sb1.ToString());
308             }
309         }
310 
311     }
```