UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

HERRAMIENTA PARA LA GENERACIÓN DE SCRIPT DE INSTALACIÓN EN EL MARCO DE TRABAJO SAUXE

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor:

Eidel Parada Guerra

Tutores:

Ing. René R. Bauta Camejo

Ing. Claudia Bravo Batista

La Habana, Junio de 2013

"Año 55 de la Revolución"

"Una baja probabilidad no significa cero."

Anónimo

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Firma del Autor

_____ _____

Firma del Tutor Firma del Tutor

DATOS DE CONTACTO

Síntesis de los tutores:

Tutor: Ing. René R. Bauta Camejo

Graduado de Ingeniería en Ciencias Informáticas en Julio del 2009

Dirección: Universidad de las Ciencias Informáticas (UCI), Edificio: 129, Apto: 105

Teléfono Oficina: +53 – 7 – 8358296. Teléfono Apto: +53 – 7 – 837 – 3156

E-mail: rrbauta@uci.cu

Categoría docente: Instructor

Tutor: Ing. Claudia Bravo Batista

Graduado de Ingeniería en Ciencias Informáticas en Julio del 2012

Dirección: Universidad de las Ciencias Informáticas (UCI), Edificio: 27, Apto: 201

Teléfono Oficina: +53 – 7 – 8358292. Teléfono Apto: +53 – 7 – 835 – 8767

E-mail: cbravo@uci.cu

DEDICATORIA

A mi familia, por la plena confianza que siempre han tenido en mí.

A Sandri, esto también es para ti, aunque ya hace mucho que eres parte imprescindible de mi familia.

A Rene, mi tutor, por todos los conocimientos transmitidos.

A Claudia, mi tutora, por la paciencia, que sé que conmigo hay que tener mucha.

A mis compañeros, por los mejores cinco años de mi vida.

A mi madre.

AGRADECIMIENTOS

Llega el difícil momento de escribir los agradecimientos, llevar casi un año pensando en lo que quisiera poner no me ha servido de nada. Son tantos los que de una forma u otra han contribuido en esta tesis y en mi formación como profesional y ser humano que creo injusto el tener que mencionarlos, ya que siempre alguno, no por ser menos importante, quedará olvidado.

Primero que todo quiero agradecer a mi madre, que, a pesar de todos los obstáculos y pruebas que le ha deparado la vida, siempre ha luchado por mí y por mi hermana entregando todo cuanto estuvo a su alcance, haciendo lo posible y lo imposible para apoyarnos siempre, con ese cariño infinito y esa confianza absoluta que siempre me ha entregado.

Al amor de mi vida, a Sandra (Sandrita, como cariñosamente me refiero a ella) la persona que siempre ha sabido apoyarme de todas las formas posibles. Cada vez que el mundo se me viene encima ahí está ella cargando una parte importante del peso que me toca. Conmigo ha hecho uso de sus conocimientos de psicología, e inclusive aportó a la programación de la solución de esta tesis. Gracias por el amor, la confianza, la paciencia… Gracias por hacer mi vida más viva.

A mi hermana, a pesar de la distancia sé que siempre está aquí, pensando constantemente en nosotros y soñando con volver a vernos. Gracias por el apoyo, la fe, el ejemplo.

A Ronald, mi cuñado, mi ejemplo a seguir como persona y profesional. Fue ese último empujón que me adentró en este mundo de la informática, a él le debo un gran porcentaje de mis conocimientos y gran parte de lo que soy. Gracias mi hermano por todas tus enseñanzas y ratos alegres que pasamos juntos. Gracias por ser tan amigo.

A mis tutores, Claudia y Rene, mil gracias por todo, la paciencia, los conocimientos, su amistad. Sin ustedes nada de esto sería posible. Les debo mucho a ustedes dos. Las palabras no alcanzan para expresar la infinita gratitud que siento hacia ustedes.

Agradecimientos

A mis abuelos, Francia y Octavio, por ese ejemplo de tenacidad y fortaleza. Mi abuelo siempre me enseñó el camino a seguir en la vida. "Hay que tratar de echar todo lo que se pueda en la mochila para que no nos falte en el viaje" decía, y tiene razón, en la sociedad actual aquel que sólo médico sea ni médico es.

A mis tíos, Sandra y Chachi, por ese ejemplo de fuerza y optimismo por la vida, por esa capacidad de no parar, de siempre estar buscando qué hacer y hacerlo. Gracias por apoyarme tanto, sin ustedes nada de lo que soy hoy sería posible.

A mi tía Cari, por ese cariño infinito, por esos abrazos que tanto me llenan, por el apoyo y la fe.

A mis amigos de cuarto, por estos cinco años de estudios, fiestas, juegos, sustentos y risas. Gracias por soportarme como soy y nunca pedirme que cambiara. Entré en esta universidad con una hermana y salgo con una hermana y cinco hermanos. Les aseguro que muy pronto nos estaremos encontrando de nuevo.

A mis amigos y profes del Departamento de Tecnologías, por todo lo aprendido en estos años y por dejarme ser parte de la gran familia que hemos formado.

A los amigos que han compartido conmigo los años que llevo en este mundo, compartiendo locuras, aventuras, alegrías y tristezas. A Carlos Ernesto, mi hermano de andadas y desandadas. Al Caco, el Buti y Rodolfo, la mejor pandilla que he tenido. A Patricia y Liz, las mejores amigas que se pueden desear.

A los que han compartido conmigo en algún momento de estos cinco años, mis inolvidables amigos del grupo de primer año. A los que me apoyaron en cada aventura en la que me lanzaba.

A mi prima Ale, espero que este logro le sirva de ejemplo para que siga con sus magníficos resultados como estudiante y como persona.

A mi primo Fabián. Estudia mucho, diviértete, disfruta la vida que en un abrir y cerrar de ojos estarás donde estoy ahora y te darás cuenta que todo pasa muy rápido.

A mis suegros, mejores no los quiero. Gracias por todo ese apoyo incondicional que siempre me han profesado y demostrado. Gracias por traer al mundo a esa maravillosa persona que hoy comparte mi vida.

Agradecimientos

A la UCI, sin duda la mejor universidad del mundo, por darme la oportunidad de conocer a tanta gente linda, por todos los sueños hechos realidad, por los buenos y malos momentos, por los recuerdos que me llevo.

Y por último, pero no menos importante, quiero agradecer a mi padre. A pesar de que la vida no me permitió seguir disfrutándolo y lo arrancó de mi lado tan temprano, me ha servido como ejemplo de hombre, persona y amigo. Sé que siempre me está observando y espero que con este resultado esté orgulloso de mí.

RESUMEN

El uso de marcos de trabajo ha facilitado el desarrollo de aplicaciones web. Para el correcto despliegue de este tipo de soluciones se elaboran instaladores los cuales se encargan de descomprimir el código fuente de la aplicación, a través de scripts para la creación y configuración de la base de datos. Para la creación de scripts se usan herramientas de generación proveídas por los desarrolladores de sistemas de base de datos. Con ellas se disminuye el tiempo en que se desarrolla un producto y las posibilidades de errores en la elaboración del mismo, además se incrementa la calidad del software creado y se facilita su mantenimiento. La Universidad de las Ciencias Informáticas (UCI) emplea en el despliegue de los productos elaborados en el marco de trabajo Sauxe, un instalador web que utiliza los scripts descritos con anterioridad, los mismos se elaboran manualmente provocando inconvenientes y vulnerabilidades para el sistema.

Luego del análisis del estado del arte de las principales herramientas existentes para la generación de scripts, y basado en la imposibilidad de cumplir las necesidades del instalador web con dichas herramientas, se realizó el diseño de clases del sistema, dando respuesta al modelo de arquitectura seleccionado. De esta forma se obtiene un sistema capaz de realizar la generación de scripts, proponiéndolo como herramienta de generación de scripts para instalación del marco de trabajo Sauxe. A raíz de esta propuesta, el mayor impacto radica en el logro de un sistema que garantiza al equipo de desarrollo la generación de los scripts para los instaladores web.

PALABRAS CLAVES

Generador, Instalador, Script.

ÍNDICE

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los Sistemas de Información han cambiado la forma en que operan las entidades actuales. A través de su uso se logran importantes mejoras, pues automatizan los procesos operativos y suministran una plataforma de información necesaria para la toma de decisiones, haciendo a la entidad más eficiente con mejores resultados en menos tiempo.

En el desarrollo de estos sistemas de información juegan un papel importante los marcos de trabajo (del inglés frameworks), los cuales ya traen definidos procesos comunes y necesarios en todos los sistemas enmarcados en este tipo de software. Los marcos de trabajo organizan mejor el proceso de desarrollo de cualquier solución que se desarrolle con ellos y reducen de forma notoria el tiempo empleado para el desarrollo.

En Cuba, en el marco de la informatización de la sociedad y en la búsqueda de la soberanía tecnológica, existen varias instituciones que se dedican al desarrollo de software entre los que se encuentra la Universidad de Ciencias Informáticas. Dentro de la UCI se encuentra el Centro de Informatización de la Gestión de Entidades (CEIGE), cuyo objetivo es crear sistemas que apoyen a la informatización de los procesos operativos de las entidades así como brindar soluciones que apoyen la toma de decisiones. Como base tecnológica para el desarrollo de sus productos creó el marco de trabajo Sauxe. Este marco de trabajo contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Siguiendo el paradigma de independencia tecnológica por el cual apuesta el país, este marco de trabajo es utilizado en diversos proyectos productivos de la UCI y en otras entidades productoras de software del país.

Para facilitar el proceso de despliegue de los productos desarrollados en el CEIGE sobre Sauxe, se desarrolló un instalador web que contiene paquetes de software con la información necesaria para automatizar este proceso, los cuales se encargan de crear y configurar la base de datos, establecer las configuraciones necesarias para la carga inicial y descomprimir el código fuente de la solución. Para la creación y configuración de la base de datos se necesitan scripts SQL[i].

El proceso de instalación de estas soluciones contiene tres subprocesos: la creación y configuración de la base de datos, la ejecución de la carga inicial del sistema y la personalización de los ficheros de configuración.

Actualmente los subprocesos de creación y configuración de la base de datos y el de ejecución de la carga inicial se realizan a través de la ejecución de scripts que contienen código SQL. Un script es un conjunto de instrucciones que se almacenan en un archivo de texto plano y permite la automatización de algunas tareas en interacción con algún sistema o base de datos.

La arquitectura definida para el instalador antes mencionado contempla tres tipos de script: los de estructura y permisos que contienen la información relacionada con la creación y configuración de la base de datos y los scripts de datos que contienen la carga inicial del sistema. Los scripts de estructuras crean toda la estructura de esquemas, secuencias, tablas, funciones, disparadores, índices, tipos de datos y vistas necesarias para lograr el correcto funcionamiento del sistema. Los scripts de permisos establecen los niveles de accesibilidad que tendrán los roles creados en los scripts de roles sobre los objetos de la base de datos. Por otro lado, los scripts de datos ejecutan la carga inicial del sistema.

El proceso de creación de los scripts antes descritos se realiza de forma manual mediante algún cliente de base de datos y cualquier editor de texto. Para ello se selecciona el elemento que se desee que configure el instalador y se copia el código SQL que nos provee el cliente hacia el editor de texto, estructurándolo en la organización requerida para los instaladores de las soluciones desarrolladas en Sauxe. Sin embargo, crear los script de esta forma implica varios inconvenientes. Para que este proceso se lleve a cabo satisfactoriamente se hace necesaria la preparación del personal que ejecutará este proceso. La planificación y ejecución de la capacitación de este personal implica dedicar tiempo y recursos que podrían apoyar otros procesos del desarrollo de la solución requerida.

Cuando la solución que se desplegará contiene un gran número de objetos a nivel de base de datos se dificulta aún más el proceso de creación de los script de estructura y datos afectando la fecha de entrega del producto. Como todo proceso que se ejecuta de forma manual no está exento de la introducción de errores, aumentado la ocurrencia de estos al introducir los datos. Esto provoca que el sistema sea más vulnerable a los ataques a la base de datos.

Todo lo anteriormente planteado provoca que el sistema no quede correctamente configurado, que no se realice la instalación total de todos sus subsistemas o que se presenten problemas derivados de la mala configuración y carga inicial de la base de datos que puede facilitar la ocurrencia de errores o posibles ataques a la seguridad del sistema.

La materialización de alguno de estos problemas implica que la solución no posea la efectividad y la eficacia para la que fue concebida y que al final dé pérdidas al cliente en vez de ganancias.

A partir de la situación descrita con anterioridad, se define el siguiente **problema a resolver:** ¿Cómo disminuir el tiempo y la introducción de errores en el proceso de creación de instaladores con el marco de trabajo Sauxe?

Con el fin de darle respuesta al problema se identifica el siguiente **objetivo general:** Desarrollar una herramienta que permita la generación de scripts de instalación para disminuir el tiempo y la introducción de errores en el proceso de creación de instaladores con el marco de trabajo Sauxe.

Para darle cumplimiento al objetivo general se propone el proceso de creación de instaladores para aplicaciones web como **objeto estudio**, enmarcando como **campo de acción** la generación de scripts para instaladores de aplicaciones web.

Con el propósito de darle cumplimiento al objetivo general se proponen los siguientes **objetivos específicos:**

1. Realización del Marco Teórico de la investigación que permita identificar posibles puntos de reutilización así como los conceptos principales asociados a la misma.
2. Realización del Análisis y Diseño de la Herramienta de Generación de scripts de instalación sirvan de base para la implementación de la misma.
3. Implementación de la Herramienta de Generación de scripts de instalación.
4. Validación de la Herramienta de Generación de scripts de instalación empleando técnicas para la validación de los requisitos, métricas para la validación del diseño y pruebas de caja blanca y caja negra para la validación de la aplicación.

A partir de la correcta realización de las tareas de la investigación se propone la siguiente **idea a defender:** Si se implementa correctamente una herramienta que automatice el proceso de generación de scripts se reducirá el tiempo y la introducción de errores en el proceso de creación de instaladores del marco de trabajo Sauxe.

Los métodos científicos de investigación son la forma de abordar la realidad con el propósito de descubrir su esencia y sus relaciones. Durante el desarrollo de la investigación se han utilizado un conjunto de métodos científicos para la obtención y procesamiento de la información y la elaboración de las conclusiones. Los cuales se mencionan a continuación:

Métodos Teóricos:

1. **Análisis Histórico – Lógico:** Se aplica con el objetivo de conocer la existencia de Sistemas de Información Geográfica, que realicen el proceso de gestión de las vistas de los mapas, para tener una

mayor visión de la forma de ejecución de estos procesos, en este tipo de sistema y utilizar estos como puntos de referencia.

2. **Analítico – Sintético:** Se aplica con el objetivo de analizar todos los documentos relacionados con la estructura de las bases de datos de Postgres, la extracción de información de las mismas y la generación de scripts con dicha información.
3. **Método Modelación:** Se aplica para generar los artefactos que se construyen durante el proceso de Ingeniería de Software, como modelo para comprender las propiedades, funcionalidades y relaciones que contiene la herramienta de generación de scripts.

Métodos Empíricos:

1. **Método entrevista:** Se aplica para obtener información sobre la estructura de las bases de datos de Postgres y las funcionalidades que debe tener la herramienta de generación de scripts.

Con el desarrollo de la investigación se espera alcanzar la construcción de una herramienta que permita la generación de scripts de instalación, así como la documentación técnica de la misma. Dicha herramienta contribuirá a disminuir el tiempo de creación de instaladores con el marco de trabajo Sauxe y la introducción de errores en este proceso.

El presente trabajo de diploma se encuentra estructurado en tres capítulos que estarán guiados por los siguientes temas a tratar:

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA: Está centrado en el análisis de los elementos y conceptos relacionados con la estructura de las bases de datos de Postgres y el proceso de generación de scripts. Se realiza una presentación del estado del arte en el mundo de las aplicaciones que proponen soluciones similares a la que se quiere brindar con la herramienta de generación de scripts de instalación del marco de trabajo Sauxe. Además se hace una descripción de las herramientas y tecnologías que se emplearán para la construcción de la solución.

CAPÍTULO II. PRESENTACIÓN DE LA SOLUCIÓN PROPUESTA: Está centrado en modelar el proceso de análisis y diseño de la herramienta de generación de scripts. Además se realizan los artefactos relacionados con la implementación.

CAPÍTULO III. VALIDACIÓN DE LA SOLUCIÓN: Se realiza la validación de los requisitos identificados aplicando técnicas de validación. Se aplican métricas para la validación del diseño de la solución propuesta. Se presentan los resultados de las pruebas de caja blanca y caja negra realizadas a la solución.

1. CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

El presente capítulo constituye el marco teórico de la investigación a realizar. En el mismo se describen los conceptos necesarios para tener una mayor comprensión del problema propuesto, profundizando en temas como los scripts y los diferentes tipos de scripts definidos para los instaladores de Sauxe. Además se realiza una reseña sobre el contexto en el que se encuentra en la actualidad el proceso de generación de scripts a nivel global. Se describen también las tecnologías utilizadas durante el proceso de desarrollo de software a partir de lo que propone el modelo de desarrollo empleado.

1.2. Conceptos asociados al dominio del problema

Script

Los scripts son archivos de texto, a menudo con la extensión .SQL. Cuando se ejecuta un archivo de script, se ejecutan las instrucciones de SQL almacenadas en el mismo [1]. Para el desarrollo de la presente investigación se definirá como script al conjunto de instrucciones que se almacenan en un archivo de texto plano y permite la automatización de algunas tareas en interacción con algún sistema o base de datos.

Tipos de scripts utilizados por los instaladores de Sauxe

Para los instaladores de las soluciones desarrolladas en el marco de trabajo Sauxe se han definido tres tipos de scripts: de estructura y permisos que contienen la información relacionada con la creación y configuración de la base de datos y los scripts de datos que contienen la carga inicial del sistema [2].

Scripts de estructura

Los scripts de estructuras crean toda la estructura de esquemas, secuencias, tablas, funciones, disparadores, índices, tipos de datos y vistas necesarias para lograr el correcto funcionamiento del sistema. Básicamente se encargan de crear la base de datos de la solución [2].

Scripts de permisos

Los scripts de permisos establecen los niveles de accesibilidad que tendrán los roles creados en los scripts de roles sobre los objetos de la base de datos [2].

Scripts de datos

Los scripts de datos ejecutan la carga inicial del sistema. Insertan en las estructuras creadas en los scripts de estructuras todos los datos necesarios y suficientes para el correcto funcionamiento de la solución [2].

Paquete de scripts

En la presente investigación se ha definido como paquete de scripts al conjunto de scripts generados. Generalmente se hará un paquete para cada sistema, pero, en el caso de que existan varias versiones de un mismo sistema o varios clientes que requieran un mismo sistema pero solicitando diferentes funcionalidades, se hará un paquete de scripts para cada versión y para cada cliente.

Sistema

El término “sistema” tiene un sinfín de definiciones y conceptos. En este caso, se trata de un sistema en términos de software o sistema de información (SI). Según Laudon un SI puede definirse como un conjunto de componentes interrelacionados que reúne (u obtiene), procesa, almacena y distribuye información para apoyar la toma de decisiones y el control en una organización [3]. O´Brien define los SI como la combinación organizada de personas, mecanismos físicos (hardware), procedimientos e instrucciones de procesamiento de información (software), canales de comunicación (redes) y datos almacenados (recursos de datos) que reúne, transforma y disemina información en una organización [4].
Para el desarrollo de la presente investigación se asume como “sistema” al conjunto de componentes, o subsistemas, que relacionados entre sí procesa y/o almacena información siendo capaz de apoyar la toma de decisiones de una organización, definición elaborada a partir del análisis de las demás definiciones estudiadas.

Instalador

Se puede considerar como instalador a un paquete de software que se encarga de descomprimir y ejecutar el código fuente de una solución o programa y configurar toda la información necesaria para la primera ejecución de la misma [2].

1.3. Estado del Arte

El proceso de generación de scripts no es algo nuevo en el mundo. Ya varias compañías productoras de software han dedicado esfuerzos en brindar herramientas que suplan esta necesidad. Algunas de estas herramientas son:

Microsoft SQL Server Management Studio

SQL Server Management Studio es un entorno integrado para obtener acceso, configurar y administrar todos los componentes de SQL Server. Esta aplicación combina un amplio grupo de herramientas gráficas con cinco editores de script enriquecidos para ofrecer acceso a SQL Server a desarrolladores y administradores con cualquier grado de experiencia. Management Studio está diseñado para desarrollar y

administrar objetos de base de datos y para administrar y configurar los objetos existentes de Analysis Services[ii] [5].

En SQL Server Management Studio se pueden crear scripts de Transact-SQL[iii] para varios objetos utilizando el "Asistente Generar y Publicar Scripts" y se puede generar un script para objetos individuales o varios objetos utilizando el menú "Incluir como" del Explorador de objetos [6].

El asistente genera un script de todos los objetos de una base de datos o un subconjunto de los objetos que seleccione. Además dispone de muchas opciones para los scripts, como la posibilidad de incluir permisos, la intercalación, las restricciones, etc. [6].

De igual manera se puede utilizar el menú "Incluir como" del Explorador de objetos para generar un script de un solo objeto, de varios objetos o de varias instrucciones para un único objeto. Puede elegir uno de varios tipos de scripts; por ejemplo crear, modificar o quitar el objeto. Puede guardar un script en una ventana del Editor de consultas, en un archivo o en el Portapapeles. El script se crea en formato Unicode [6].

SQL-dump de PostgresSQL

La idea detrás del método SQL-dump[iv] es generar un archivo de texto con los comandos SQL que, cuando se alimenta de vuelta al servidor, recreará la base de datos en el mismo estado en que se encontraba en el momento de la descarga [7].

La aplicación Pg_dump es un utilitario para volcar una base de datos Postgres en un fichero de script conteniendo comandos de consulta. Los ficheros de script están en formato de texto y pueden ser usados para reconstruir la base de datos, incluso en otras máquinas y con otras arquitecturas. Pg_dump producirá las consultas necesarias para regenerar todos los tipos definidos por el usuario, funciones, tablas, índices, agregados y operadores. Adicionalmente, toda la información es copiada en formato de texto el cual puede ser nuevamente copiado, también puede ser importado a herramientas para su edición [8].

Pg_dump es útil para verter el contenido de una base de datos que se necesite trasladar de una instalación de Postgres a otra. Después de ejecutar Pg_dump, se debe examinar el script de salida a ver si contiene alguna advertencia [8].

Export de Oracle

Oracle es un sistema de gestión de base de datos objeto-relacional (o ORDBMS por el acrónimo en inglés de Object-Relational Data Base Management System), desarrollado por Oracle Corporation [9].

Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando entre sus características:

- Soporte de transacciones
- Estabilidad
- Escalabilidad
- Soporte multiplataforma [9].

Su dominio en el mercado de servidores empresariales ha sido casi mayoritario hasta hace poco, recientemente sufre la competencia de SQL Server de Microsoft y de la oferta de otros RDBMS (Relational Database Management System o Sistema de Gestión de Base de Datos Relacional) con licencia libre como PostgreSQL, MySQL o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux [9].

La utilidad Export de Oracle ofrece una manera sencilla para transferir objetos de datos entre bases de datos Oracle, incluso si residen en plataformas con diferente hardware y software. Cuando se ejecuta contra una base de datos de exportación de Oracle, los objetos (como tablas) se extraen, seguidos por sus objetos relacionados (tales como índices, comentarios y donaciones), si existen [10].

1.4. ANÁLISIS DE LAS HERRAMIENTAS EXISTENTES

A partir de la descripción de las herramientas presentadas, se puede destacar que Oracle y SQL Server tienen licencia propietaria y sus costos son elevados. Financiar la compra de las licencias requeridas para el uso de estas herramientas elevaría el presupuesto del proyecto, lo cual no es factible por las condiciones económicas a las que Cuba se ha visto sometida por más de cincuenta años.

Además ninguna de las soluciones estudiadas, incluyendo PostgreSQL, brinda la posibilidad de generar scripts con la estructura requerida por los instaladores de las soluciones desarrolladas sobre Sauxe. Las mismas generan automáticamente una estructura predefinida por el fabricante para la comprensión de las plataformas del propio fabricante.

Pero, a pesar de no cumplir con los requerimientos de los instaladores de Sauxe, estas herramientas brindan idas y elementos que pueden ser empleados en el desarrollo de la nueva solución. Ejemplo de lo anteriormente planteado es la idea de usar árboles para representar los objetos de las bases de datos organizándolos por esquemas y las funcionalidades y catálogos de PostgreSQL que permiten un mejor acceso a la información requerida para la generación de los scripts para los instaladores del marco de trabajo Sauxe.

Por lo anteriormente planteado, el Departamento de Tecnologías del CEIGE ha decidido desarrollar una solución propia y de código abierto para agilizar el proceso de creación de scripts para los instaladores.

1.5. TENDENCIAS Y TECNOLOGÍAS ACTUALES

1.5.1. Metodología de desarrollo

La producción del CEIGE se concentra en el desarrollo de proyectos generalmente de gran magnitud, por lo que se hace necesario contar con un modelo estandarizado, que establezca las distintas fases por las que se debe transitar y el conjunto de artefactos a generar en cada una de ellas. Teniendo como precedente dicha necesidad y en colaboración con las distintas líneas de desarrollo donde se ejecutan cada uno de estos proyectos, se ha decidido usar la metodología descrita en el documento "Modelo de desarrollo de software para el CEIGE". Dicho documento detalla el ciclo de vida de proyectos del centro con la incorporación de los distintos subprocesos definidos por el Nivel II de CMMI (Capability Maturity Model Integration), certificación obtenida por el centro en julio de 2011 y reconocida por el SEI (Software Engineering Institute) como aval de la calidad de su proceso de desarrollo de software [11].

Para el ciclo de vida de los proyectos del CEIGE se tienen en cuenta las fases y actividades por áreas de procesos que plantea el Nivel II de CMMI establecido en la UCI. En la figura 1 que a continuación se presenta se pueden apreciar el total de fases por las que pueden transitar los proyectos del centro, pudiendo realizar iteraciones a partir de la fase de Modelado del negocio. El conjunto de fases propuestas abarca el total de acciones que se realizan en las distintas líneas de desarrollo para la elaboración del servicio o producto final; sin embargo, se debe adaptar a las características particulares del proyecto que puede que no lleve a cabo alguna de las estas fases o la elaboración de algunos de sus artefactos.

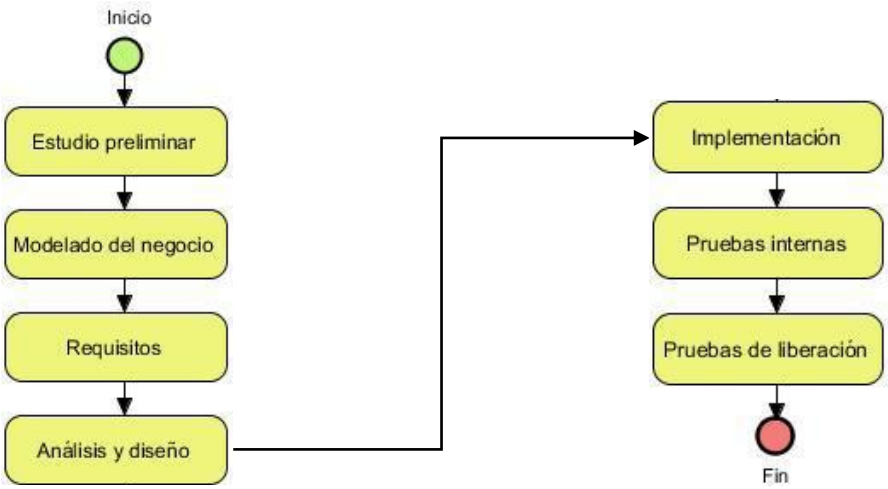


Figura 1: Ciclo de vida de proyectos del CEIGE.

1.5.2. Herramientas CASE

Visual Paradigm

Visual Paradigm es una herramienta CASE (Computer Aided Software Engineering o Ingeniería de Software Asistida por Computadora) profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Esta herramienta ayuda a construir aplicaciones con calidad y con un menor coste, además de disminuir el periodo de desarrollo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y documentación. La herramienta CASE también proporciona abundantes tutoriales, demostraciones interactivas y proyectos en UML (Unified Modeling Language o Lenguaje Unificado de Modelado). Para los ingenieros y arquitectos de software es excelente ya que permite centralizar y planificar las ideas, además de mejorar la productividad en el desarrollo y mantenimiento del software, aumentar la calidad del mismo, reducir el tiempo de desarrollo y mantenimiento, mejorar la planificación de un proyecto y realizar una gestión global en todas las fases de desarrollo de software con una misma herramienta. Esta herramienta puede ayudar en todos los aspectos del ciclo de vida del software. La mayor parte de los equipos de desarrollo del CEIGE cuentan con un dominio medio o alto de la herramienta, aspecto que le da un valor agregado a la decisión de adoptar esta solución para el modelado del sistema [12].

1.5.3. Herramientas para el desarrollo colaborativo

Subversion

Subversion es un sistema centralizado de control de versiones de código abierto. Se caracteriza por su fiabilidad como un refugio seguro para los datos valiosos, la simplicidad de su modelo, uso y capacidad para apoyar las necesidades de una amplia variedad de usuarios y proyectos, desde proyectos particulares hasta operaciones empresariales a gran escala [13].

RapidSVN

RapidSVN es una plataforma cruzada GUI[v] con interfaz de usuario para el sistema de revisión de Subversion. Está escrito en C++ utilizando el marco wxWidgets[vi]. RapidSVN está licenciado bajo la versión 3 de la Licencia Pública General GNU [14].

1.5.4. Herramientas para el desarrollo

PostgreSQL

Es un gestor de base de datos relacional y libre. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados y almacenamiento de grandes objetos. Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y JOINS[vii] de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Como toda herramienta de software libre PostgresSQL tiene, entre otras ventajas, las de contar con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y algo muy importante es que dicha herramienta es multiplataforma. Además de sus ofertas de soporte, cuenta con una importante comunidad de desarrollo de los que se pueden obtener beneficios y contribuir. Está disponible en casi cualquier Unix (34 plataformas en la última versión estable), además de una versión nativa de Windows [8].

NetBeans

NetBeans es una herramienta de desarrollo libre y de código abierto creada por la compañía Sun MicroSystems. La misma es capaz de generar el código necesario para la creación tanto de servicios web como de clientes de servicios web para Java. Todo esto es posible hacerlo mediante asistentes que permiten la configuración visual de los WS, de forma tal que le ahorre tiempo al desarrollador [15].

1.5.5. Librerías y marcos de trabajo

Sauxe

Sauxe es un marco de trabajo que contiene un conjunto de componentes reutilizables que provee la estructura genérica para el desarrollo de aplicaciones web de gestión, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo, siguiendo el paradigma de independencia tecnológica por el cual apuesta el país [16].

El marco de trabajo Sauxe da solución a un sin número de escenarios o aspectos arquitectónicos como: gestión y configuración dinámica de caché, integración de componentes de forma distribuida o no distribuida, acceso a bases de datos a través de una capa de abstracción, gestión de concurrencia de recursos, administración centralizada de transacciones, gestión dinámica de las trazas generadas por los sistemas, implementación de mecanismos de autenticación y autorización, implementación de mecanismos de mensajería y control de excepciones, gestión y configuración dinámica de precondiciones, postcondiciones y validaciones de variables, gestión y configuración de flujos de trabajo, visualización de

las funcionalidades de un sistema, entre otros escenarios de alta complejidad, garantizando los atributos de calidad de los sistemas que se desarrollen con el mismo. Cuenta con una arquitectura en capas que a su vez presenta en su capa superior un MVC (modelo-vista-controlador) [16].

ExtJS

Es una librería Java Script open-source (código abierto) de alto rendimiento para la creación y desarrollo de aplicaciones web dinámicas. Provee interfaces gráficas de usuario que brindan experiencias parecidas o iguales a las que se tienen con aplicaciones de escritorio. Permite la creación de aplicaciones complejas utilizando componentes predefinidos. Es extensible para la gran mayoría de los navegadores, evitando el tedioso problema de validar el código para cada uno de estos [17].

En el mercado actualmente existen múltiples librerías de JavaScript que permiten realizar todo tipo de maravillas en el navegador web. ExtJS permite que con pocas líneas de código sea posible realizar interfaces amigables para los usuarios. Es la librería más avanzada para el desarrollo rápido de aplicaciones con una apariencia totalmente novedosa y una arquitectura flexible. ExtJS permite construir aplicaciones complejas en Internet [17].

Esta librería incluye:

- Componentes de interfaces de usuarios de alto rendimiento y personalizables.
- Modelo de componentes extensibles.
- Un API (Application Programming Interface o Interfaz de programación de aplicaciones) fácil de usar.
- Licencias open-source y comerciales.

Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts (diseños por su traducción al inglés) similar al que provee Java Swing[viii]. Gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, Internet Explorer, Safari, etc.) [17].

Usar una librería con interfaces gráficas como ExtJS, permite tener además estos beneficios:

- Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo [16].
- Comunicación asíncrona. En este tipo de aplicación el motor de renderizado puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta [17].

- Eficiencia de la red. El tráfico de red puede disminuir al permitir que la aplicación elija qué información desea transmitir al servidor y viceversa; sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio debido al incremento del tráfico [17].

Zend Framework

Se trata de un framework para desarrollo de aplicaciones y servicios web con PHP que brinda soluciones para construir sitios web modernos, robustos y seguros. Además es de código abierto y trabaja con PHP5. A diferencia de CakePHP[ix] que trabaja con PHP 4 y PHP 5, este framework está formado por una serie de métodos estáticos y componentes que usarán estos métodos. Los componentes son varios y variados y aunque alguno es posible que no se use nunca, hay otros que pueden ser usados hasta la saciedad, por ejemplo el componente para la BD. Entre los componentes de vital importancia se encuentran: Zend_Config para temas de configuración de aplicaciones web, Zend_Db para tratar con bases de datos, Zend_Search o Zend_Feed, entre otros. La instalación es sencilla, tan solo se tendrá que añadir en el fichero de configuración php.ini, el path (camino) hasta la carpeta library (librería) del framework con la instrucción include_path [18].

Doctrine ORM

Doctrine es un potente y completo sistema ORM (object relational mapper o mapeador de objetos relacionales) para PHP 5.2+ con un DBAL (database abstraction layer o capa de abstracción de la base de datos) incorporado. Se está empezando a ver su potencial, pero de la documentación se puede decir que tiene todas las características necesarias para ser funcional en casi cualquier proyecto. Entre muchas otras características brinda la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. Por otro lado, como la librería es bastante grande, tiene un método para ser "compilada" al pasar a producción [19].

Cuando se trabaja con Doctrine, se necesita informar a su motor interno cuál es el modelo de la aplicación, para ello se puede hacer ingeniería inversa de la base de datos existente, o si se empieza la aplicación desde el principio, crear el modelo en la sintaxis específica que propone Doctrine y luego generar toda la base de datos [16].

Para crear el modelo, Doctrine permite hacer una clase por tabla indicando mediante PHP el tipo de datos que se almacenará en el mismo [16].

1.5.6. Lenguaje de Modelado

UML

El Lenguaje Unificado de Modelado (UML por sus siglas en inglés, Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un software orientado a objetos; es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90s. Se ha convertido en el estándar de la industria, ya que, a pesar de no tener un respaldo formal de una autoridad institucional o un organismo de normalización, es ampliamente usado. Incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. Una de las metas principales de UML es avanzar en el estado de la integración institucional proporcionando herramientas de interoperabilidad para el modelado visual de objetos [20].

El lenguaje está dotado de múltiples herramientas para lograr la especificación determinante del modelo, algunas de ellas son:

- Modelamiento de Clases
- Casos de Uso
- Diagrama de Interacción [20].

1.5.7. Lenguajes de programación

PHP

PHP, acrónimo de Hypertext Preprocessor, es un lenguaje open-source interpretado de alto nivel, especialmente pensado para desarrollos web que puede ser incrustado en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl. La meta de este lenguaje es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil, aunque se pueda hacer mucho más con PHP [21].

Lo mejor de usar PHP es que es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales [21].

JavaScript

JavaScript es un lenguaje de scripting basado en objetos, que se utiliza principalmente para crear páginas web dinámicas y permite el desarrollo de interfaces de usuario mejoradas. Una página web dinámica es aquella que permite la interacción entre el contenido de la misma y el usuario. JavaScript permite incorporar a dichas páginas efectos como texto que aparece y desaparece, animaciones, acciones que se

activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. A pesar de su nombre, no guarda relación directa con el lenguaje Java, sino que simplemente la compañía dueña del mismo lo adoptó por una cuestión de mercado [17].

1.6. PATRONES

¿Qué es un patrón?

Según Christopher Alexander cada patrón describe un problema que ocurre una y otra vez en un entorno, además describe el núcleo de la solución de ese problema de forma que se puede utilizar esta solución un millón de veces, sin hacerlo nunca de la misma manera [22].

1.6.1. Patrones de diseño

Los patrones de diseño son soluciones a problemas repetidos en la construcción de software, y en ocasiones pueden incluir sugerencias para aplicar estas soluciones en diversos entornos [23].

En el diseño que propone este trabajo, se utilizaron algunos patrones de diseño GoF (Gang of Four o Grupo de Cuatro) y GRASP (General Responsibility Assignation Software Pattern o Patrones Generales de Software de Asignación de Responsabilidades), para solucionar y/o evitar diferentes problemas que pudieran aparecer durante la implementación de la solución.

GRASP (Patrones Generales de Asignación de Responsabilidad)

Experto: Propone que la clase que contenga toda la información necesaria, será la responsable de la creación de un objeto o la implementación de un método. El comportamiento se distribuye entre las que contienen la información requerida, siendo más fáciles de entender, mantener y ampliar, aumentando sus posibilidades de reutilización [24].

Creador: La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. Este patrón proporciona asistencia al identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. Mediante el uso de este patrón se logra un bajo acoplamiento, facilitando la posibilidad de mantenimiento y reutilización [24].

Controlador: El patrón controlador funciona como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la interfaz quien recibe los datos del usuario y los envía a las distintas clases según el método invocado [24].

Alta cohesión: Propone que la información que almacena una clase debe ser coherente y estar, en la medida de lo posible, relacionada con la clase. Al realizar un cambio en una clase con alta cohesión, todos los métodos que pueden verse afectados, estarán a la vista, en el mismo archivo. Incrementa la claridad, la reutilización y la facilidad de comprensión del diseño [24].

Bajo acoplamiento: Este patrón expresa que entre las clases deberán existir pocas ataduras, es decir, estarán lo menos relacionadas posible, de forma tal que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, incrementando la reutilización, y disminuyendo la dependencia entre ellas [24].

GoF (Gang of Four)

Proxy: Cuando no se desea el acceso directo a un objeto sobre el que se va a aplicar determinada acción, este patrón propone la adición de un nivel que permita solamente el acceso al objeto a través de un objeto sustituto, que será el responsable de controlar o mejorar el acceso al objeto real [24].

Facade (Fachada): Propone la definición de un único punto de conexión con un componente. Este objeto fachada presenta una única interfaz unificada y es responsable de colaborar con los componentes del subsistema [24].

Singleton: El patrón Singleton garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ésta instancia [25].

1.6.2. Patrones Arquitectónicos

En la solución propuesta se utilizó el patrón arquitectónico Modelo-Vista-Controlador (MVC), el cual se emplea en la capa de funcionamiento y es el encargado de separar los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos [26]:

- Modelo: está compuesto por datos, reglas de negocio y las funcionalidades correspondientes para la comunicación con el marco de persistencia de datos Doctrine.
- Controlador: gestiona las entradas del usuario.
- Vista: muestra la información del modelo usuario.

1.7. CONCLUSIONES PARCIALES

Se ha realizado un estudio del estado del arte de la generación de scripts a nivel mundial permitiendo identificar y especificar los conceptos que son imprescindibles para un mejor entendimiento de la investigación. Además, este estudio permitió identificar los posibles puntos de reutilización de las herramientas que actualmente brindan la generación de scripts como una de sus funcionalidades.

2. CAPÍTULO II. PROPUESTA DE SOLUCIÓN

2.1. Introducción

En el presente capítulo, se realiza una propuesta de solución que va a estar conformada por la identificación y descripción de los requisitos funcionales y no funcionales que debe cumplir el sistema a implementar, además de los artefactos generados durante las fases de Análisis y Diseño de la herramienta.

2.2. Modelo Conceptual

El Modelo Conceptual es la representación visual de los conceptos u objetos del mundo real que son significativos para el problema o área de interés que se analiza, representando las clases conceptuales, no los componentes de software. El mismo, captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema, además de incluir las responsabilidades de las personas que ejecutan las actividades [27].

2.2.1. Descripción del modelo conceptual

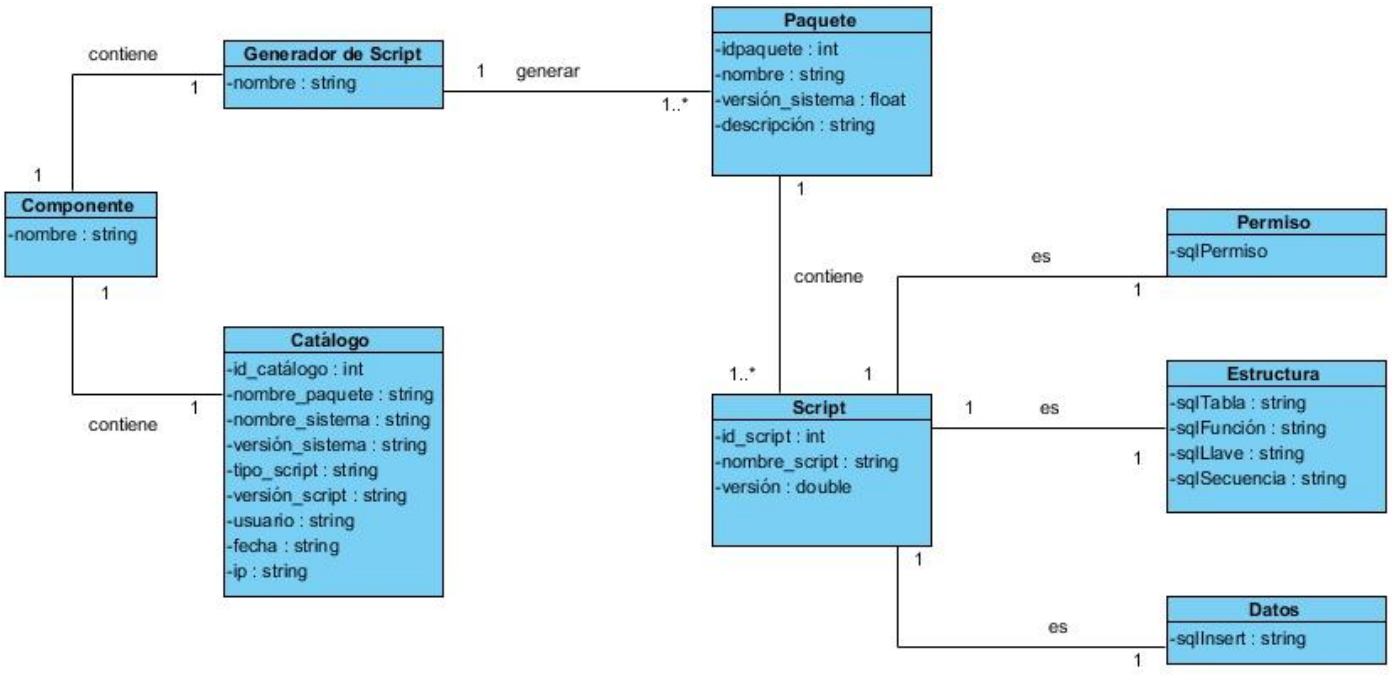


Figura 2: Modelo Conceptual.

El modelo conceptual de la solución propuesta (ver Figura 2) se encuentra representado a través de un Componente que contiene un Generador de script que va a escribir en Catálogo los script generados y va

a permitir generar un conjunto de Paquetes que contienen los Script de instalación que pueden ser de tres tipos: de Permiso, identificándose por las consultas SQL sobre los permisos a las bases de datos; de Estructura, refiriéndose a las consultas SQL de tablas, funciones, llaves y secuencias en la base de datos y por último de Datos que contiene las consultas SQL de los INSERT[x].

2.3. Requisitos

El análisis de las soluciones existentes en el capítulo anterior, la representación del modelo conceptual para la herramienta Generador de scripts y las entrevistas realizadas a los especialistas del proyecto permitió efectuar una correcta captura de requisitos, quedando reflejado en los epígrafes que se relacionan a continuación.

2.3.1. Técnicas para la captura de requisitos

La captura de requisitos permite descubrir o averiguar lo que se debe construir. Este proceso mejora la capacidad de predecir cronogramas de proyecto proporcionando un punto de partida para controlar actividades específicas. De igual manera mejora la calidad del software pues se tiene una guía de lo que el usuario desea de modo que si se cumplen estos requerimientos se satisfacen las necesidades del cliente [28].

Para la captura de requisitos de la solución propuesta se emplearon principalmente dos técnicas: la encuesta y la tormenta de ideas.

Entrevista:

- Entrevistas de Cuestionarios: Este tipo de entrevistas recomienda que se genere un cuestionario de preguntas , el cual será aplicado al cliente para comenzar la captura de requisitos [29].
- Entrevistas de final abierto: Este tipo de entrevistas son del tipo que realizan los psicólogos. La idea es que el ingeniero de requisitos permita que el Cliente le vaya narrando su problemática y el ingeniero de Software lo guíe a través de la narración para ir determinando los requisitos del sistema [29].
- Entrevistas en grupos de desarrollo: Este tipo de entrevistas recomienda formar grupos específicos con el personal del cliente. Estos grupos tendrán en común algún área de trabajo o especialidad. El objetivo es poder contar con los expertos en cierta área de la empresa para poder llegar en conjunto a la especificación de requisitos [29].
- Discusiones: Este tipo de entrevistas pretende que el Ingeniero de Requisitos sostenga una discusión con el Cliente sobre su problemática para tratar de determinar en conjunto los requisitos del sistema [29].

Para la captura de requisitos de la solución propuesta se empleó la entrevista de final abierto, entrevistando a especialistas en la generación de scripts, arquitectos y administradores de base de datos del CEIGE.

Tormenta de Ideas:

Es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios. Puede ayudar a generar una gran variedad de vistas del problema y a formularlo de diferentes formas, sobre todo al comienzo del proceso de captura [30].

2.3.2. Requisitos funcionales

Los requisitos funcionales son las capacidades o condiciones que el sistema debe cumplir, para que el usuario resuelva un problema o cumpla sus objetivos. Deben estar claros y libres de ambigüedades, asegurando que los involucrados comprendan claramente el significado de cada uno [31]. A continuación se definen los requisitos funcionales para la herramienta a desarrollar.

RF1. <Conectar a la base de datos> Permitirá conectarse a la base de datos, introduciendo el nombre del servidor, el nombre de la base de datos, el puerto, el usuario y la contraseña del mismo.

El sistema debe permitir generar un conjunto de paquetes que van a contener los tres tipos de script de permiso, de estructura y de datos.

RF2. <Adicionar paquete> Permitirá al usuario adicionar un nuevo paquete, introduciendo el nombre del paquete, la versión del sistema que refiere y una descripción del mismo. Los paquetes deben contener los tres tipos de script: de permiso, de estructura y de datos.

RF3. <Modificar paquete> Permitirá al usuario modificar un paquete que ha sido insertado, introduciendo los datos a modificar.

RF4. <Eliminar paquete> Permitirá al usuario seleccionar un paquete de los que han sido insertados y eliminarlo.

El sistema debe permitir configurar los tres tipos de script: de estructura, de permiso y de datos.

FR5. <Generar script de estructura> Permitirá que el usuario seleccione un script de estructura, que se encuentra contenido en el paquete, configure la información seleccionando de los esquemas de la base de datos las tablas, funciones y secuencia y pueda generar el script con la información configurada.

FR6. <Generar script de permiso> Permitirá que el usuario seleccione un script de permiso, que se encuentra contenido en el paquete, configure la información seleccionando los objetos de la base de datos y pueda generar el script con la información configurada.

FR7. <Generar script de datos> Permitirá que el usuario seleccione un script de datos, que se encuentra contenido en el paquete, configure la información seleccionando las tuplas de las tablas deseadas y pueda generar el script con la información configurada.

FR8. <Generar Historial de script> Permitirá persistir los datos de los script generados.

FR9. <Eliminar un script del historial> Permitirá eliminar un script de los que se muestran en el historial.

2.3.3. Descripción de Requisitos

Descripción del requisito Conectar a la base de datos

Precondiciones	El usuario debe haber sido autenticado en el sistema.
Flujo de eventos	
Flujo básico Conectarse a la base de datos	
1	El usuario inserta los datos de la conexión a la bases de datos: Servidor Puerto Usuario Contraseña Bases de datos
2	El sistema valida (ver validación 1) los datos introducidos.
3	Si los datos son correctos el sistema los registra.
4	El sistema confirma el registro de los datos.
5	Concluye el requisito.
Post-condiciones	
1	Se conecta el sistema a la base de datos.
Flujos alternativos	
Flujo alternativo 3.a Datos incorrectos	
1	El sistema señala los datos erróneos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 2 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo 3.b Campos vacíos	
1	El sistema señala los datos vacíos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 2 del flujo básico.
Post-condiciones	
1	N/A
Flujo alternativo *.a El usuario cancela la acción	
1	Concluye el requisito.
Post-condiciones	
1	El sistema no se conecta a la base de datos.
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual Generador de scripts.

Conceptos	**Conexión**	Visibles en la interfaz: Servidor Puerto Usuario Contraseña Base de datos
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	

Prototipo de interfaz

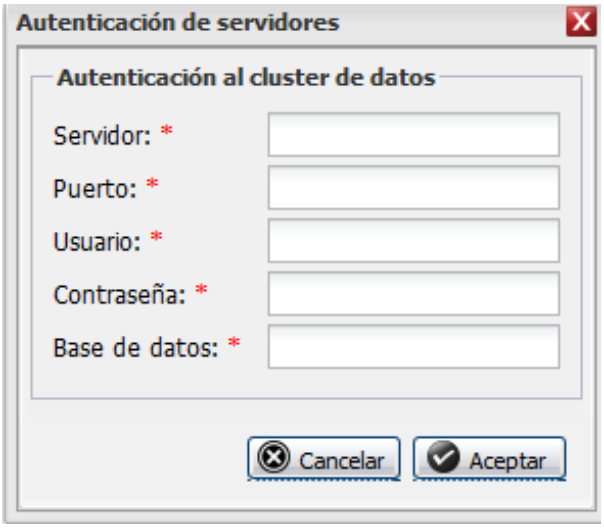


Tabla 1: Descripción del requisito Conectar a la base de datos.

Descripción del requisito Adicionar paquete

Precondiciones	El usuario debe haber sido autenticado en el sistema.
Flujo de eventos	
Flujo básico Adicionar paquete	
1	El usuario selecciona la opción **Adicionar Paquete**.
2	El usuario inserta los datos del paquete: Nombre Versión Descripción
3	El sistema valida (ver validación 1) los datos introducidos.
4	Si los datos son correctos el sistema los registra.
5	El sistema confirma el registro de los datos.
6	Concluye el requisito.
Post-condiciones	
1	Se actualiza la lista de paquetes.
Flujos alternativos	

Flujo alternativo 3.a Datos incorrectos	
1	El sistema señala los datos erróneos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 3 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo 3.b Campos vacíos	
1	El sistema señala los datos vacíos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 3 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo *.a El usuario cancela la acción	
1	Concluye el requisito.
Post-condiciones	
1	El sistema no adiciona el paquete.
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual Generador de scripts.

Conceptos	**Paquete**	Visibles en la interfaz: Nombre del paquete Versión Descripción
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	

Prototipo de interfaz

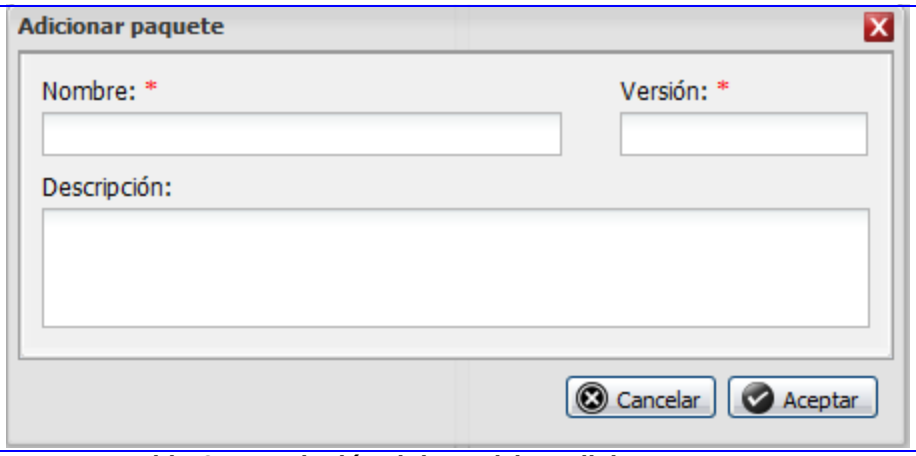


Tabla 2: Descripción del requisito Adicionar paquete.

Descripción del requisito Modificar paquete

Precondiciones	El usuario debe haber sido autenticado en el sistema. El usuario debe haberse conectado a la base de datos. El usuario debe haber creado un paquete.	
Flujo de eventos		
Flujo básico Modificar paquete		
1	El usuario selecciona el paquete de los que se muestran en el sistema.	
2	El usuario selecciona la opción **Modificar Paquete**.	
3	El sistema muestra y permite editar los datos del paquete.	
4	El usuario modifica los datos del paquete: Nombre Versión Descripción	
5	El sistema valida (ver validación 1) los datos introducidos.	
6	Si los datos son correctos el sistema los registra.	
7	El sistema confirma el registro de los datos.	
8	Concluye el requisito.	
Post-condiciones		
1	Se modificaron los datos del paquete.	
Flujos alternativos		
Flujo alternativo 6.a Coincidencia en los datos insertados		
1	El sistema señala los datos erróneos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 5 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo *.a El usuario cancela la acción		
1	Concluye el requisito.	
Post-condiciones		
1	El sistema no modifica el paquete.	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual Generar Script.	
Conceptos	**Tipo de dato**	Visibles en la interfaz: Nombre del paquete Versión Descripción
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	
Prototipo de interfaz		

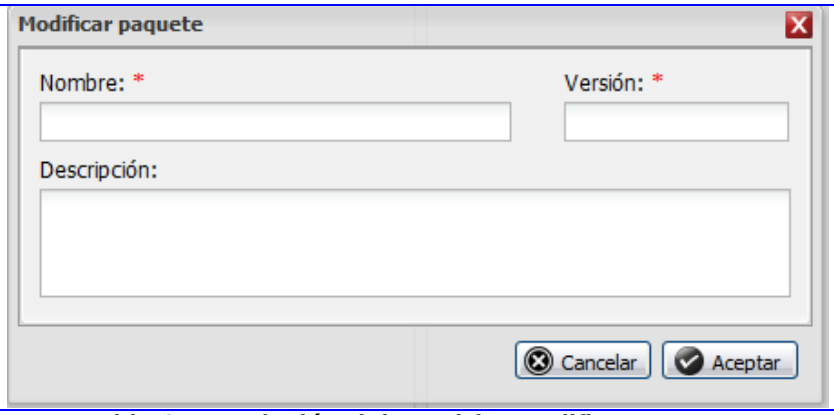
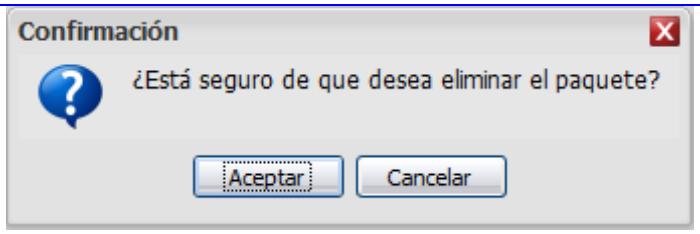


Tabla 3: Descripción del requisito Modificar paquete.

Descripción del requisito Eliminar paquete

Precondiciones	El usuario debe haber sido autenticado en el sistema. El usuario debe haberse conectado a la base de datos. El usuario debe haber creado un paquete.	
Flujo de eventos		
Flujo básico Eliminar paquete		
1	El usuario selecciona un paquete a eliminar.	
2	El usuario selecciona la opción **Eliminar Paquete**.	
3	Se solicita confirmación para eliminar el paquete.	
4	Si el usuario confirma se elimina el paquete.	
5	El sistema confirma la eliminación.	
6	Concluye el requisito.	
Post-condiciones		
1	Se elimina el paquete seleccionado.	
Flujos alternativos		
Flujo alternativo *.a El usuario cancela la acción		
1	Concluye el requisito.	
Post-condiciones		
1	El sistema no elimina el paquete.	
Validaciones		
1	N/A.	
Concepto		Visibles en la interfaz
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	

Prototipo de interfaz



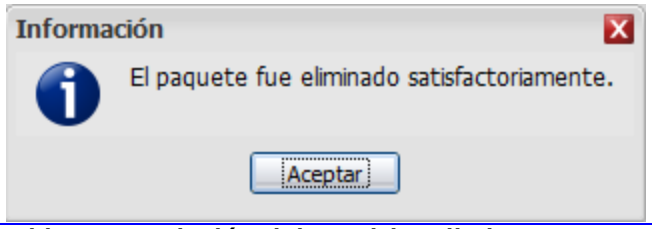
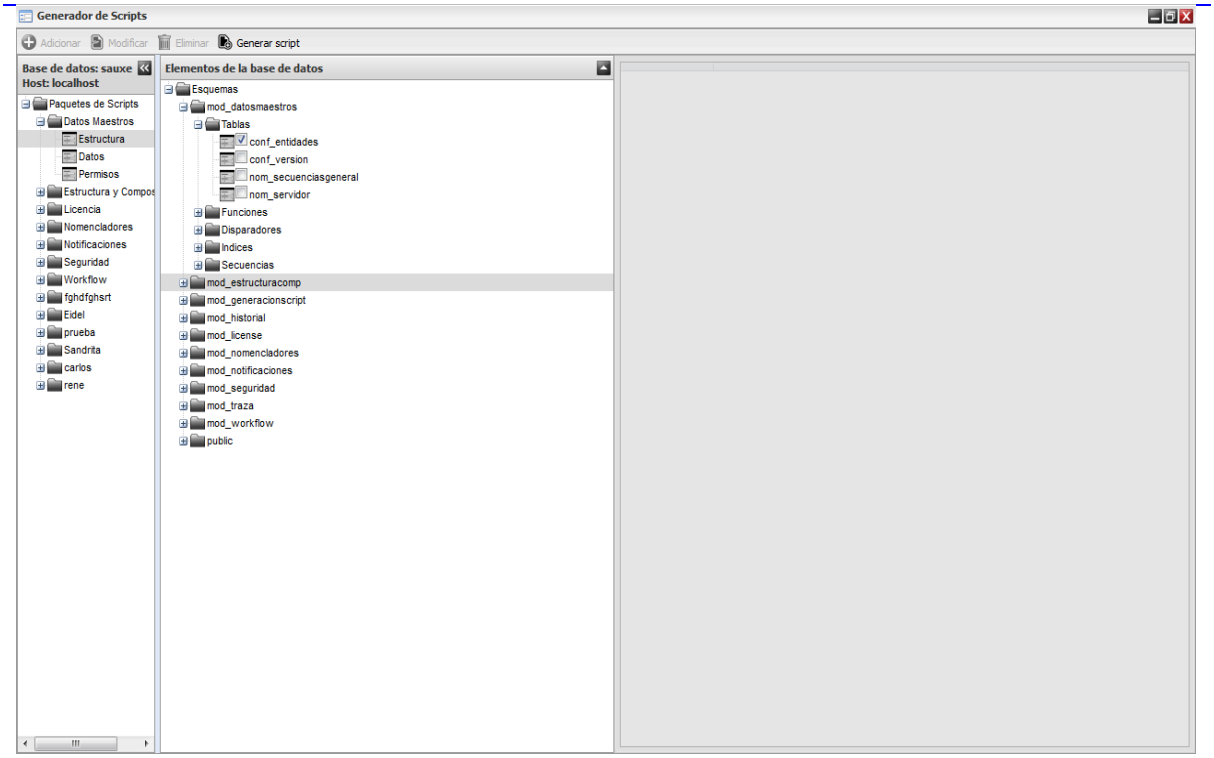


Tabla 4: Descripción del requisito Eliminar paquete.

Descripción del requisito Generar script de estructura

Precondiciones		El usuario debe haber sido autenticado en el sistema. El usuario debe haberse conectado a la base de datos. El usuario debe haber creado un paquete.
Flujo de eventos		
Flujo básico Generar script de estructura		
1	El usuario selecciona el paquete de los que se muestran en el sistema.	
2	El usuario selecciona el tipo de script de Estructura.	
3	El usuario selecciona el tipo o los tipos de objetos a generar por el script.	
4	El usuario selecciona los objetos a generar por el script.	
5	El usuario selecciona la opción **Generar script**.	
6	El sistema confirma que el script fue generado.	
7	Concluye el requisito.	
Post-condiciones		
1	Se genera el script de estructura con la nueva configuración realizada.	
Validaciones		
1	N/A	
Conceptos		Visibles en la interfaz:
Requisitos especiales	N/A	
Asuntos pendientes	N/A	
Prototipo de interfaz		



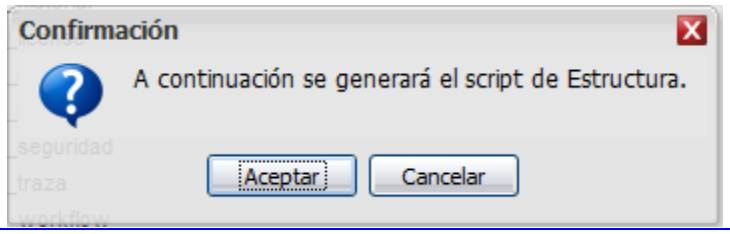
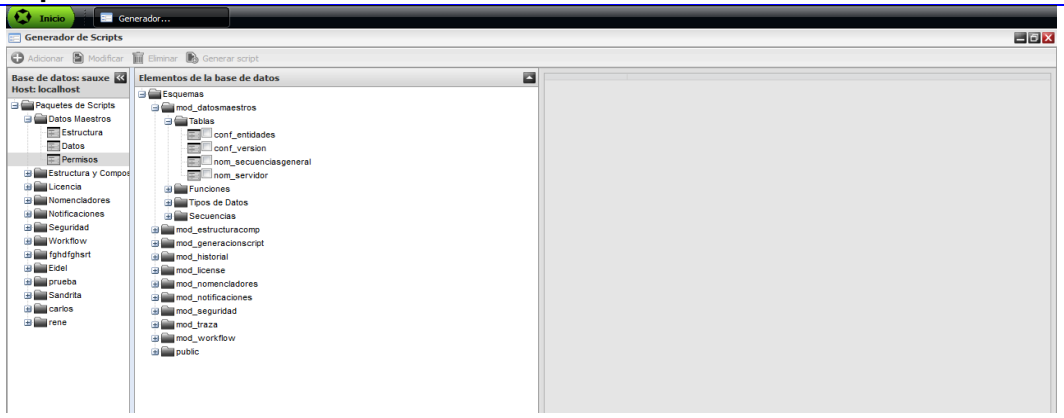


Tabla 5: Descripción del requisito Configurar script de estructura.

Descripción del requisito Generar script de permiso

Precondiciones	El usuario debe haber sido autenticado en el sistema. El usuario debe haberse conectado a la base de datos. El usuario debe haber creado un paquete.
Flujo de eventos	
Flujo básico Generar script de permiso	
1	El usuario selecciona el paquete de los que se muestran en el sistema.
2	El usuario selecciona el tipo de script de Permiso.
3	El usuario selecciona el tipo o los tipos de objetos a generar por el script.
4	El usuario selecciona los objetos a generar por el script.
5	El usuario selecciona la opción **Generar script**.
6	El sistema confirma que el script fue generado.
7	Concluye el requisito.
Post-condiciones	
1	Se genera el script de permiso con la nueva configuración realizada

Validaciones		
1	N/A.	
Conceptos		Visibles en la interfaz:
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	
Prototipo de interfaz		



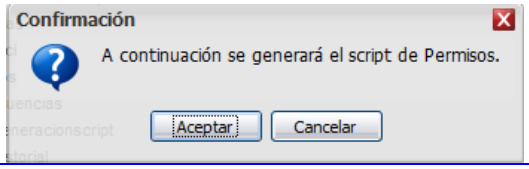
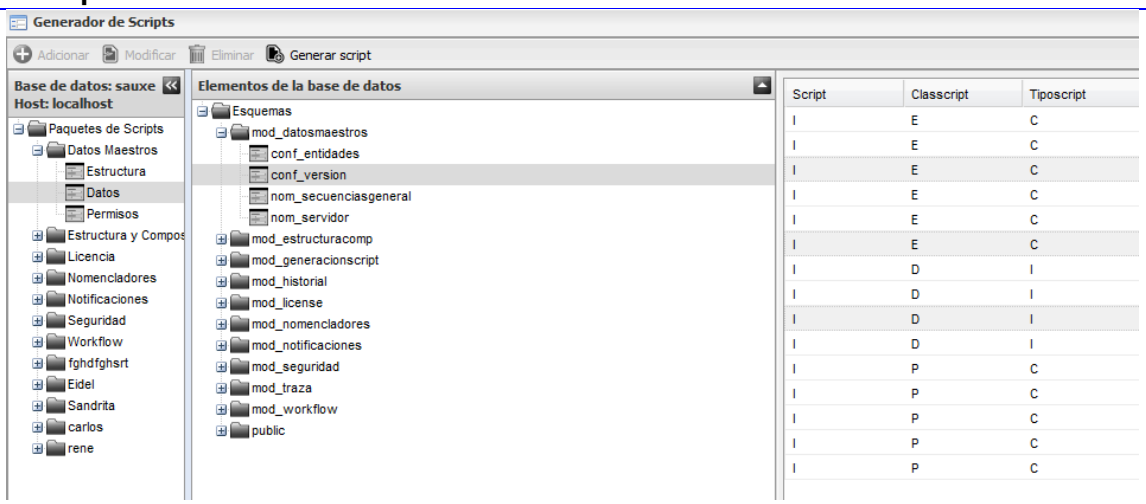


Tabla 6: Descripción del requisito Configurar script de permiso.

Descripción del requisito Generar script de datos

Precondiciones	El usuario debe haber sido autenticado en el sistema. El usuario debe haberse conectado a la base de datos. El usuario debe haber creado un paquete.
Flujo de eventos	
Flujo básico Generar script de datos	
1	El usuario selecciona el paquete de los que se muestran en el sistema.
2	El usuario selecciona el tipo de script de Datos.
3	El usuario selecciona la tabla o las tablas para insertar los datos.
4	El usuario selecciona los elementos de la tabla que se van a insertar en el script.
5	El usuario selecciona la opción **Generar script**.
6	El sistema confirma que el script fue generado.
7	Concluye el requisito.
Post-condiciones	
1	Se genera el script de datos con la nueva configuración realizada.
Validaciones	
1	N/A.

Conceptos		Visibles en la interfaz:
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	
Prototipo de interfaz		



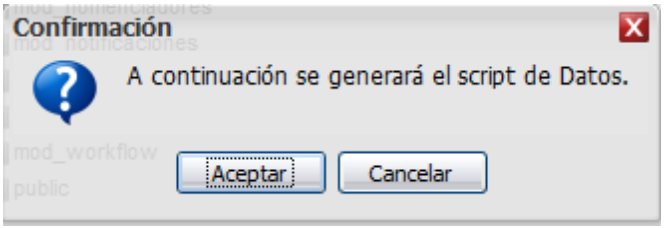
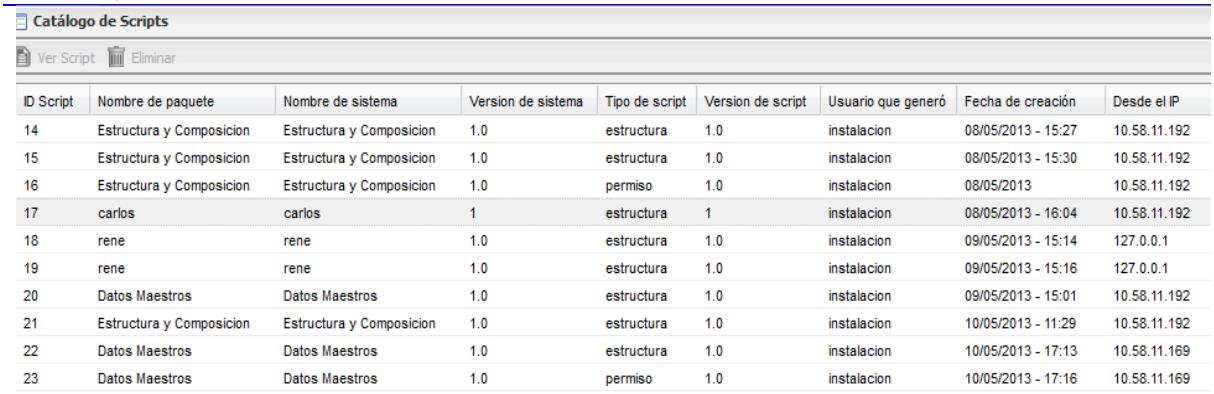


Tabla 7: Descripción del requisito Configurar script de datos.

Descripción del requisito Generar Historial de script

Precondiciones	El usuario debe haber sido autenticado en el sistema. El usuario debe haberse conectado a la base de datos. El usuario debe haber generado al menos un script.
Flujo de eventos	
Flujo básico Generar historial de script	
1	El usuario selecciona un script de los que se muestran en el sistema.
2	El usuario selecciona la opción **Ver Script**.
3	El sistema muestra las instrucciones SQL contenidas en el script seleccionado.
4	Concluye el requisito.
Post-condiciones	
1	Se muestra el código del script seleccionado.
Validaciones	
1.	N/A.

Conceptos		Visibles en la interfaz:
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	
Prototipo de interfaz		



ID Script	Nombre de paquete	Nombre de sistema	Version de sistema	Tipo de script	Version de script	Usuario que generó	Fecha de creación	Desde el IP
14	Estructura y Composicion	Estructura y Composicion	1.0	estructura	1.0	instalacion	08/05/2013 - 15:27	10.58.11.192
15	Estructura y Composicion	Estructura y Composicion	1.0	estructura	1.0	instalacion	08/05/2013 - 15:30	10.58.11.192
16	Estructura y Composicion	Estructura y Composicion	1.0	permiso	1.0	instalacion	08/05/2013	10.58.11.192
17	carlos	carlos	1	estructura	1	instalacion	08/05/2013 - 16:04	10.58.11.192
18	rene	rene	1.0	estructura	1.0	instalacion	09/05/2013 - 15:14	127.0.0.1
19	rene	rene	1.0	estructura	1.0	instalacion	09/05/2013 - 15:16	127.0.0.1
20	Datos Maestros	Datos Maestros	1.0	estructura	1.0	instalacion	09/05/2013 - 15:01	10.58.11.192
21	Estructura y Composicion	Estructura y Composicion	1.0	estructura	1.0	instalacion	10/05/2013 - 11:29	10.58.11.192
22	Datos Maestros	Datos Maestros	1.0	estructura	1.0	instalacion	10/05/2013 - 17:13	10.58.11.169
23	Datos Maestros	Datos Maestros	1.0	permiso	1.0	instalacion	10/05/2013 - 17:16	10.58.11.169

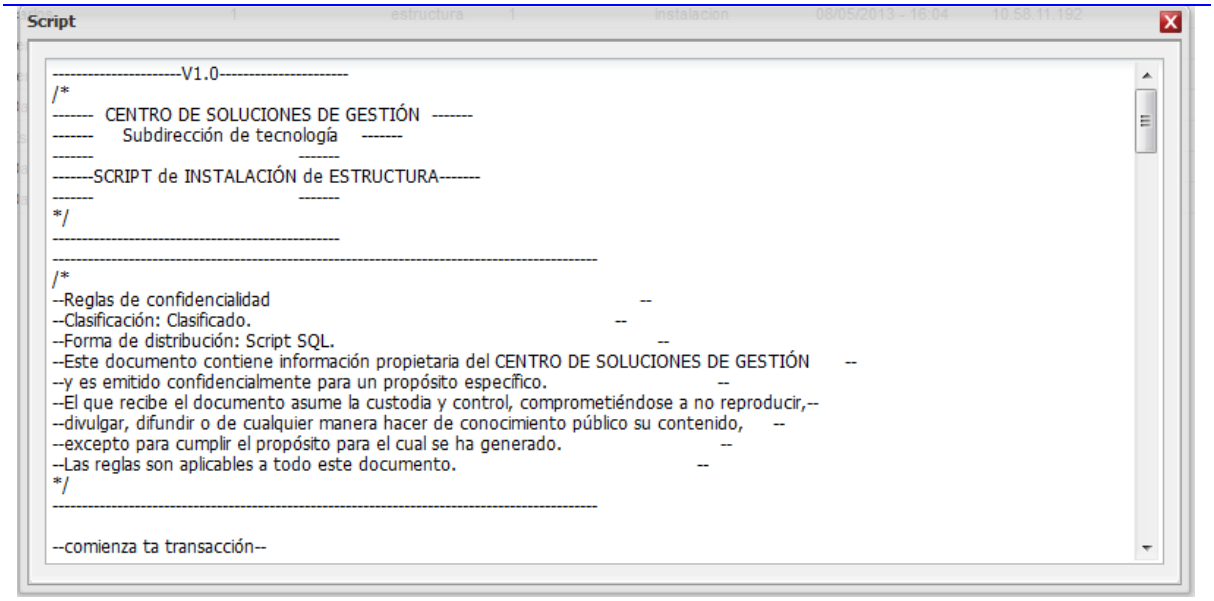
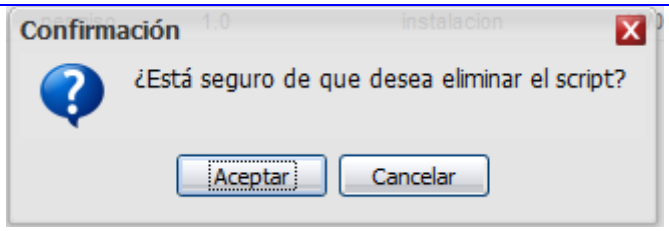


Tabla 8: Descripción del requisito Generar Historial de script.

Descripción del requisito Eliminar script del historial

Precondiciones		El usuario debe haber sido autenticado en el sistema El usuario debe haberse conectado a la base de datos El usuario debe haber generado al menos un script
Flujo de eventos		
Flujo básico Eliminar script del historial		
1	El usuario selecciona un script de los que se muestran.	
2	El usuario selecciona la opción **Eliminar**.	
3	Se solicita confirmación para eliminar el paquete.	
4	Si el usuario confirma se elimina el paquete.	
5	El sistema confirma la eliminación.	
6	Concluye el requisito.	
Post-condiciones		
1	Se elimina el script seleccionado.	
Validaciones		
1	N/A.	
Conceptos		Visibles en la interfaz:
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	
Prototipo de interfaz		



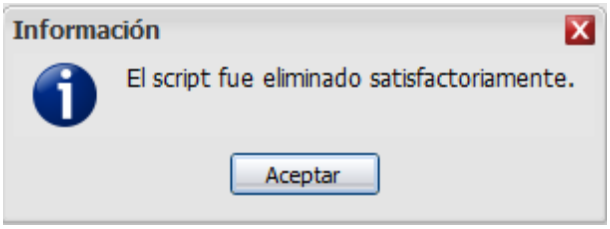


Tabla 9: Descripción del requisito Eliminar script del historial.

2.3.4. Requisitos no funcionales

Los requisitos no funcionales son las propiedades o cualidades que el producto debe tener, especificando criterios que pueden usarse para juzgar la operación de un sistema [31]. El sistema a desarrollar se va a integrar al Marco de Trabajo Sauxe; los requisitos no funcionales fueron definidos teniendo en cuenta este elemento, los cuales se mencionan a continuación:

Software

Se requiere para las PC clientes:

- Navegador Mozilla Firefox 3.6 o superior.
- Sistemas operativos GNU/Linux, Windows XP o superior.

Se requiere para las PC servidoras:

- Sistema operativo Linux en cualquiera de sus distribuciones.
- Servidor web Apache en su versión 2.0 o superior, con los módulos de PHP 5.0 disponibles.
- PostgreSQL en su versión 8.3 o superior, como Sistema Gestor de Base de Datos.

Hardware

En el caso de las PC clientes donde se ejecutará la aplicación, se requiere de las siguientes condiciones:

Procesador Pentium IV 2x2 caché, con velocidad de procesamiento a 1 GHz como mínimo y 256Mb de RAM.

En dependencia de la funcionalidad concebida para cada uno de los servidores, son las condiciones que se requiere que deban cumplir:

Servidor web: Requiere un procesador Pentium IV 2X2 caché, velocidad del procesador de 1GHz como mínimo y memoria de 1GB de RAM.

Servidor de datos: Requiere un procesador Pentium IV 2x2 caché, velocidad del procesador de 1 GHz como mínimo, memoria de 1 GB de RAM, almacenamiento en disco con una capacidad igual o superior a los 10GB (como mínimo).

Rendimiento: Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 5 segundos para las actualizaciones y 20 para las recuperaciones.

Usabilidad: El sistema es fácil de administrar por lo que los usuarios que accedan a la aplicación pueden, aunque no tengan vastos conocimientos sobre la rama de la informática, hacer uso de la misma.

2.4. Modelo de diseño

El modelo de diseño es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es utilizado como entrada esencial en las actividades relacionadas con la implementación. El modelo de diseño puede contener: diagramas, clases, paquetes, subsistemas, interfaces, relaciones y los atributos [32].

2.4.1. Diagrama de clases

Un Diagrama de Clases representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas [33]. A continuación (Figuras 2 y 3) se muestran los Diagramas de Clases con estereotipos web modelados para el desarrollo de la herramienta “Generador de Scripts de Instalación”.

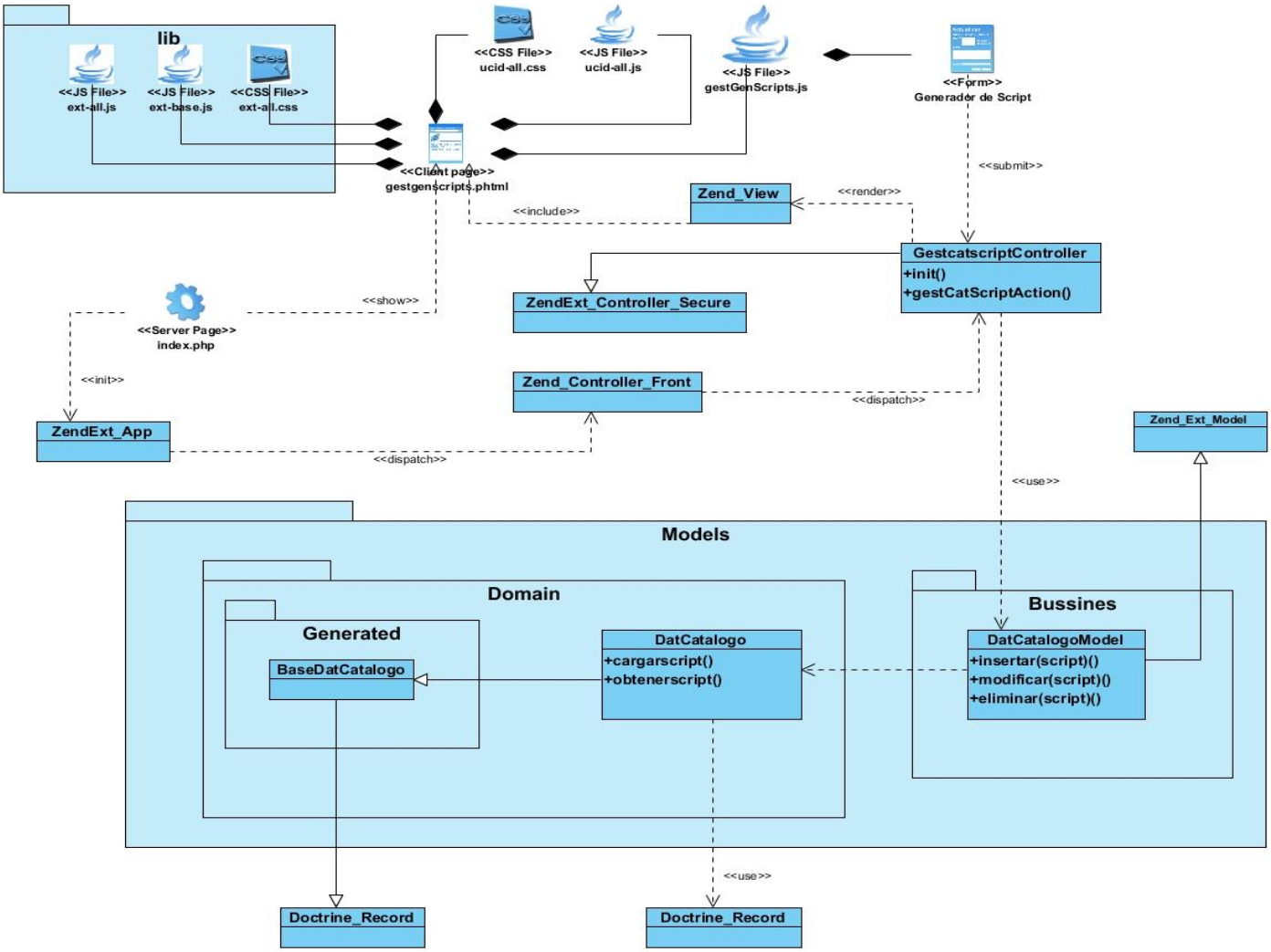


Figura 3: Diagrama de clases del Catálogo de scripts.

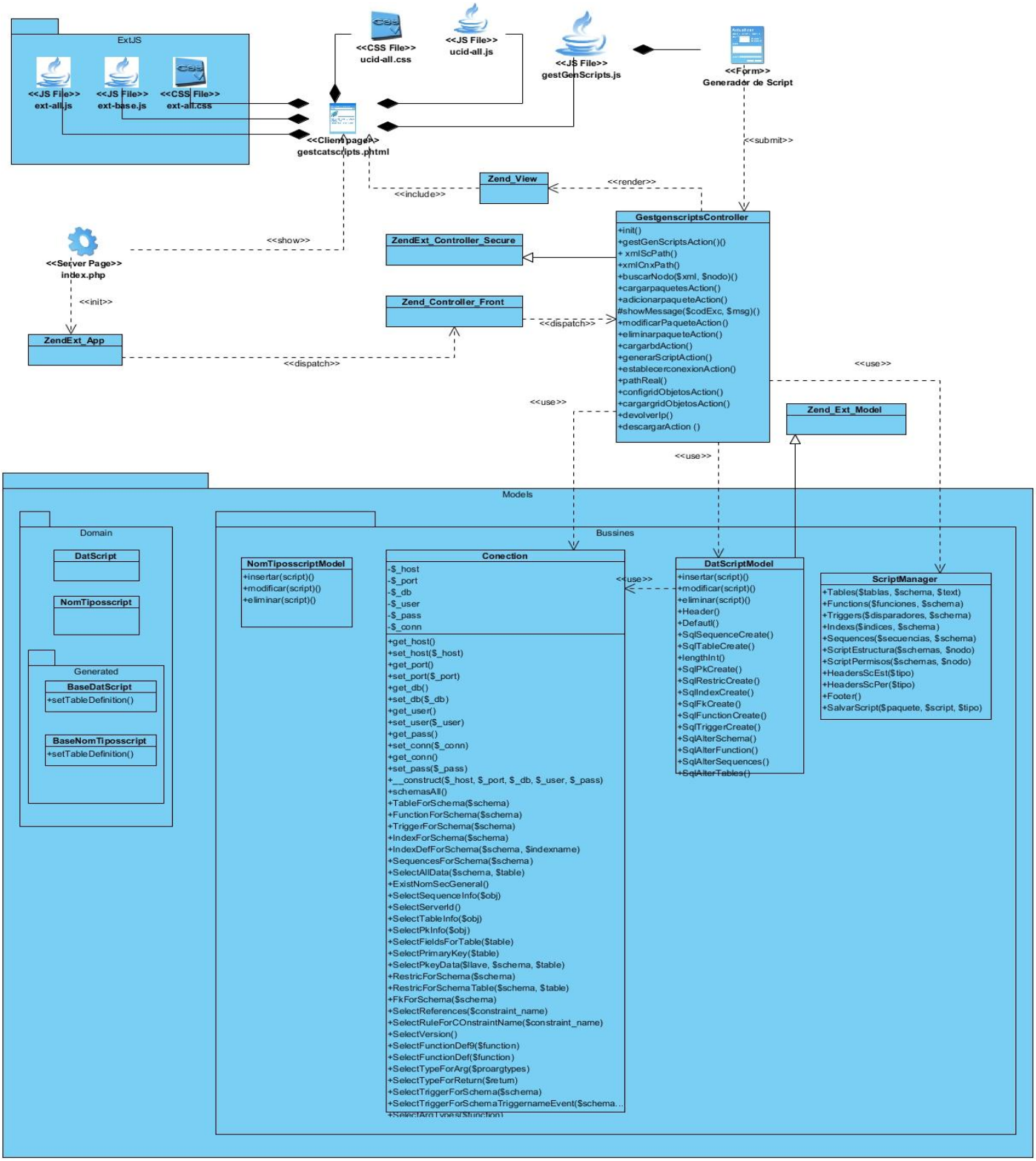


Figura 4: Diagrama de clases del Generador de scripts.

2.5. Modelo de Datos

El Modelo de Datos es usado para describir la representación lógica y física de la información persistente administrada por el sistema. Puede ser inicialmente creado a través de ingeniería inversa de un almacenamiento de datos persistentes que existan en la base de datos o puede ser inicialmente creado a partir de un conjunto de clases del diseño persistente en el modelo de diseño [34].

El Modelo de Datos definido en la solución propuesta contiene cinco clases persistentes. A continuación se muestra el diagrama entidad-relación que lo describe.

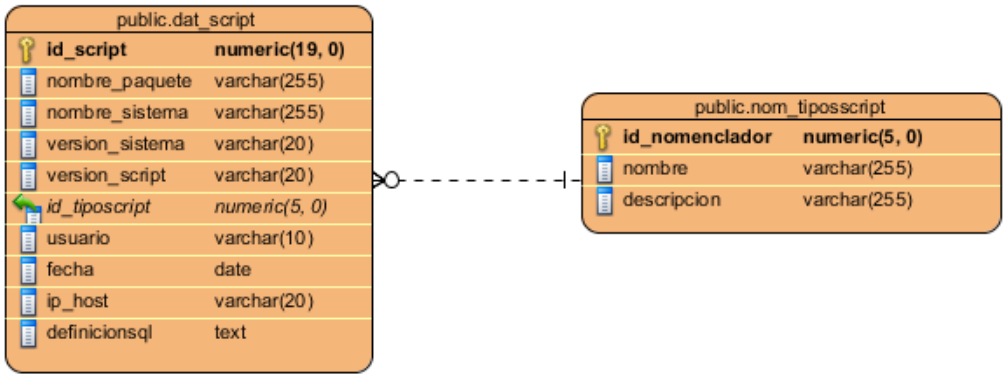


Figura 5: Modelo de Datos de la herramienta Generador de scripts.

En el Modelo de Datos que se muestra con anterioridad se representan las dos tablas necesarias en la base de datos para el funcionamiento de la Herramienta para la Generación de Scripts de Instalación en el marco de trabajo Sauxe. Se tiene la tabla dat_script, que contiene la información referente a los scripts generados, y la tabla nom_tiposscript que contiene el nombre y la descripción de los posibles tipos de scripts a generar.

2.6. Conclusiones del capítulo

En este capítulo se realizó el Análisis y Diseño de la Herramienta de Generación de scripts de instalación que sirvieron de base para llevar a cabo correctamente el proceso de implementación de la misma. Se elaboró el modelo conceptual de la solución, donde se identificó la relación que debe existir entre cada uno de los conceptos identificados. Se realizó la descripción de los requisitos funcionales y no funcionales identificados durante el proceso de Ingeniería de Requisitos. Como parte del modelo de diseño se identificaron los patrones de diseño a utilizar en el desarrollo de la solución. Por último, se elaboró y describió el Diagrama Clases y el Modelo de Datos de la solución. Este capítulo sirve de base para la implementación de la solución propuesta.

3. CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

3.1. Introducción

En el presente capítulo se describirán los estándares de codificación a utilizar para garantizar un mejor entendimiento y legibilidad del código, así como los componentes necesarios para realizar la implementación y despliegue de la solución. Además se mostrarán los resultados de la evaluación mediante métricas de diseño, pruebas de caja negra y caja blanca, métodos escogidos para validar la solución.

3.2. Validación de requisitos

Para la validación de los requisitos definidos se emplearon básicamente tres técnicas. La primera es el empleo del documento CIG-SXE-I-i6301.xls del expediente del proyecto Sauxe que contiene criterios para la validación de requisitos del cliente a modo de cuestionario. También se emplearon los prototipos de interfaz de usuario y las revisiones técnicas formales.

Prototipos de interfaz de usuario:

El prototipo de interfaz de usuario es una técnica de representación aproximada de la interfaz de usuario de un sistema software que permite a clientes y usuarios entender más fácilmente la propuesta de los ingenieros de requisitos para resolver sus problemas de negocio [35]. Los dos tipos principales de prototipos de interfaz de usuario son:

- Desechables: se utilizan sólo para la validación de los requisitos y posteriormente se desechan. Pueden ser prototipos en papel o en software [35].
- Evolutivos: una vez utilizados para la validación de los requisitos, se mejora su calidad y se convierten progresivamente en el producto final [35].

En la validación de los requisitos de la solución propuesta se emplearon prototipos de interfaz de usuario evolutivos.

Revisiones técnicas formales:

Es un proceso manual que involucra a ambas partes, tanto cliente como equipo de desarrollo, en la revisión formal el equipo de desarrollo conduce al cliente a través de los requerimientos, explicándole las implicaciones de cada uno. Los conflictos, contradicciones, errores y omisiones deben señalarse durante la revisión y registrarse formalmente. Se revisan aspectos como consistencia, integridad, verificabilidad, comprensibilidad, rastreabilidad y adaptabilidad [31].

3.2.1. Valoración de los resultados de la validación de requisitos

La validación de los requisitos demostró que estos no son ambiguos, que están completos, que existe congruencia entre los requisitos relacionados y que pueden ser implementados y probados. Esto permitió garantizar que el proceso de captura de requisitos se realizó correctamente facilitando así el proceso de implementación de la solución.

3.3. Modelo de Implementación

El Modelo de Implementación describe cómo los elementos del Modelo de Diseño, se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modulación disponibles en el entorno de implementación y en el/los lenguajes de programación utilizados y cómo dependen los componentes unos de otros, además de los recursos necesarios para poder ejecutar el sistema desarrollado [36].

3.3.1. Diagrama de Despliegue

El Diagrama de Despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes (nodos). Los nodos son objetos físicos que existen en tiempo de ejecución y que representan algún tipo de recurso computacional, también pueden ser dispositivos del sistema [37].

El usuario accede, desde su puesto de trabajo al sistema que se encuentra instalado en el servidor de aplicaciones y dicho servidor se conecta a los servidores de base de datos, para realizar las consultas y obtener los resultados deseados; en éste caso obtener los permisos otorgados a usuarios y/o roles sobre los recursos.

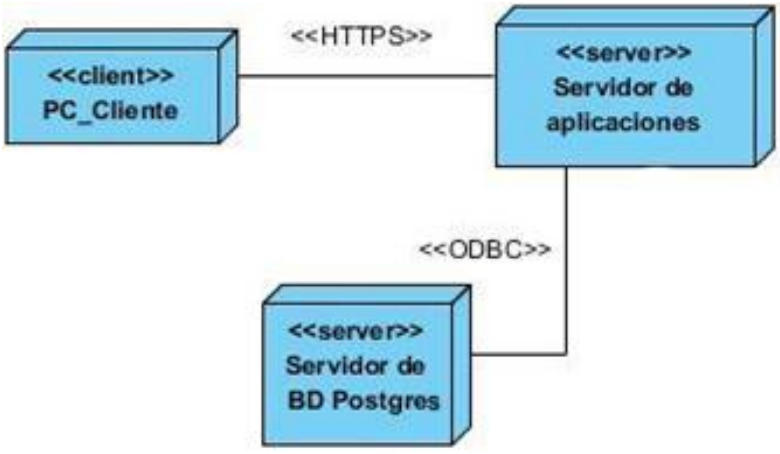


Figura 6: Diagrama de Despliegue.

3.3.2. Diagrama de Componentes

Los Diagramas de Componentes describen los elementos físicos (o componentes) del sistema y sus relaciones, muestran las organizaciones y dependencias lógicas entre componentes de software, sean éstos componentes de código fuente, binarios o ejecutables [38]. Los diagramas de componentes ilustran las piezas del software, controladores embebidos, etc., que conformarán un sistema. Un diagrama de componentes tiene un nivel más alto de abstracción que un diagrama de clases, ya que usualmente un componente se implementa por una o más clases en tiempo de ejecución [39].

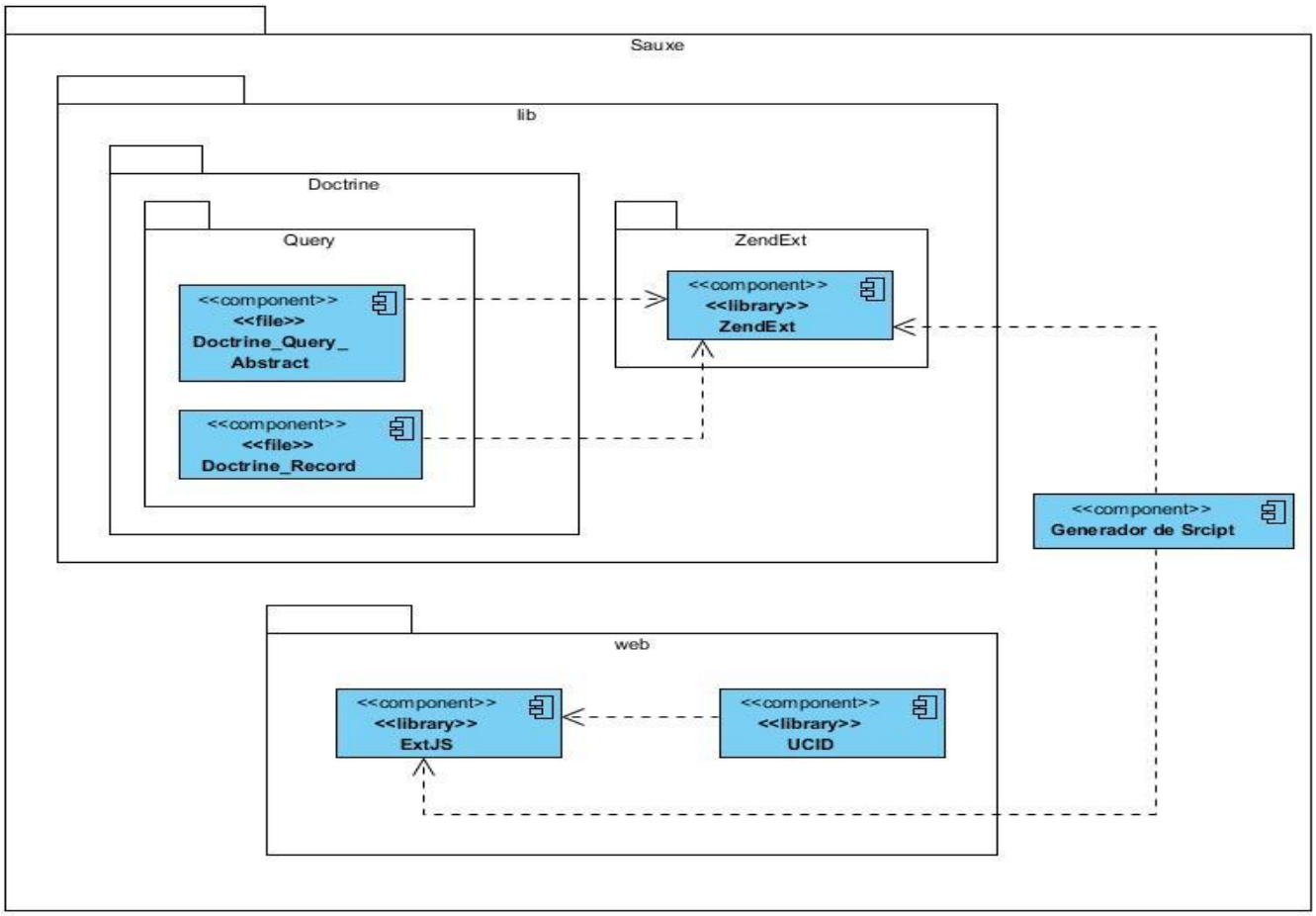


Figura 7: Diagrama de componentes.

3.3.3. Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código [40].

Para el desarrollo de la herramienta “Generador de Scripts de instalación” se utilizarán algunos de los estándares de codificación y normas propuestos como parte de la Línea de Arquitectura determinada para el desarrollo del ERP[xi] Cuba [40].

3.3.3.1. Estándares de nomenclatura

Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing*. Con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: GestionarUsuario.

Nomenclatura según el tipo de clases

Clase controladora.

Las clases controladoras después del nombre llevan la palabra: “Controller”.

Ejemplo: GestionarUsuarioController

Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing* y con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: insertarMoneda.

Nomenclatura de las variables

El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing*, comenzando con un prefijo según el tipo de datos.

Ejemplo: arrMoneda.

Prefijos para los tipos de datos

Los prefijos a utilizar en la creación de variables serán los siguientes:

Tipos de datos	Prefijos
Arreglos	arr
Objetos	obj

Tabla 10: Prefijos para tipos de datos.

Normas de comentariado

Es una necesidad comentar todo lo que se haga dentro del desarrollo, es decir, establecer las pautas que conlleven a lograr un código más legible y reutilizable, de manera que se pueda aumentar su mantenibilidad a lo largo del tiempo.

Los comentarios deben ser lo bastante claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando.

- **En las clases**

Antes de la declaración de una clase se escribe una breve descripción donde se explica el propósito de la misma. Dicha descripción puede contener el nombre de la clase, el autor, el paquete al que pertenece, la versión, entre otros datos. Se puede estructurar de la forma que se muestra a continuación:

*/**

** Nombre de la clase **

** Descripción **

** @author **

** @package *(módulo)*

** @subpackage *(sub módulo)*

** @copyright **

** @version (versión - parche)*

**/*

- **En las funciones**

Antes de la declaración de la función se escribe una breve descripción donde se explica el propósito de la misma, además del autor, los parámetros y, de ser necesarios, algunos señalamientos. Se escribe de la siguiente forma:

*/**

** Nombre de la función **

** Descripción **

** @author * (en caso de que no sea el autor de la clase)*

** @param *(los parámetros que se le pasan a la función con su descripción)*

** @throws *(en caso de que dispare una excepción)*

***/**

3.3.3.2. Estilo del código

Sangría o indexado

La política de sangría a utilizar en la implementación es por tabulación.

```php
<?php
/**
 * Indentation
 */
class Example {
    var $theInt = 1;
    function foo($a, $b) {
        switch ( $a) {
            case 0 :
                $Other->doFoo ();
            break;
            default :
                $Other->doBaz ();
        }
    }
    function bar($v) {
        for($i = 0; $i < 10; $i ++) {
            $v->add ( $i );
        }
    }
}
?>
```

Figura 8: Estilo del código: Sangría o indexado.

Brazas o llaves

En la declaración de clases o interfaces, métodos, bloques y switch[xii], la apertura de llaves se hace en la misma línea.

```php
<?php
/**
 * Braces
 */
interface EmptyInterface {
}

class Example {
    function bar($p) {
        for($i = 0; $i < 10; $i ++) {
        }
        switch ( $p) {
            case 0 :
                $fField->set ( 0 );
            break;
            case 1 :
                {
                    break;
                }
            default :
                $fField->reset ();
        }
    }
}
?>
```

Figura 9: Estilo del código: brazas o llaves.

Espacios en blanco

Arreglos.

La declaración de los espacios en blanco en los arreglos se hace tal y como se muestra a continuación.

Ejemplo:

```php
<?php
list ( $a, $b ) = array (1, 2, 3 );
$array = array (1 => 2, 2 => 3 );
$array [$i]->foo ();
$array [] = 'first cell';
?>
```

Figura 10: Estilo de código: Espacio en blanco_Arreglos.

3.4. Métricas de Diseño

La evaluación de un producto, mediante métricas, es un aspecto fundamental a tener en cuenta ya que, aunque las métricas del producto no suelen ser absolutas, brindan la posibilidad de evaluar la calidad a partir de varias reglas definidas claramente. Permiten detectar y corregir los posibles problemas que se puedan presentar durante el proceso de desarrollo y no después de terminado el producto.

IEEE (Institute of Electrical and Electronics Engineers o Instituto de Ingenieros Eléctricos y Electrónicos) define las métricas como: “una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado” [41].

Según Pressman [42], la calidad del diseño se puede evaluar aplicando métricas básicas para la calidad del diseño orientado a objetos. Los atributos de calidad involucrados son:

- ***Responsabilidad***: responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta [42].
- ***Complejidad de implementación***: grado de complejidad que posee la implementación de un diseño de clases específico [42].
- ***Reutilización***: grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software [42].
- ***Acoplamiento***: grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización [42].
- ***Complejidad del mantenimiento***: grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto [42].
- ***Cantidad de pruebas***: número o grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, módulo, clase, conjunto de clases, entre otros) diseñado [42].

Las métricas seleccionadas para evaluar la calidad del diseño de la Herramienta para la Generación de Script de Instalación en el marco de trabajo Sauxe son:

3.4.1. Tamaño Operacional de Clases (TOC)

Está dado por el número de métodos u operaciones (de instancia privada y heredada) que están encapsulados dentro o por una clase. Evalúa los siguientes atributos de calidad [43][44]:

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Aumento del TOC provoca aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Aumento del TOC provoca aumento de la complejidad de implementación de la clase.
Reutilización	Aumento del TOC provoca disminución del grado de reutilización de la clase.

Tabla 11: Atributos de calidad que evalúa TOC.

Para la evaluación de dichos atributos de calidad, se definieron los siguientes criterios y categorías de evaluación:

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad de implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Tabla 12: Criterios de evaluación de la métrica TOC.

Resultados obtenidos:

Evaluando la cantidad de operaciones de cada una de las clases con que cuenta la herramienta "Generador de Scripts de Instalación", según los criterios establecidos en la Tabla 12, se obtuvo que un 60% de las clases cuentan con Responsabilidad y Complejidad: Baja y Reutilización: Alta.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
GestcatscriptController	8	Baja	Baja	Alta
GestgenscriptsController	18	Media	Media	Media
Connection	48	Alta	Alta	Baja
DatScriptModel	22	Media	Media	Media
NomTiposscriptModel	4	Baja	Baja	Alta
ScriptManager	12	Media	Media	Media
DatScript	1	Baja	Baja	Alta
NomTiposscript	1	Baja	Baja	Alta
BaseDatScript	1	Baja	Baja	Alta
BaseNomTiposscript	1	Baja	Baja	Alta

Tabla 13: Instrumento de evaluación de la métrica TOC.

Cantidad de clases por intervalos de procedimientos

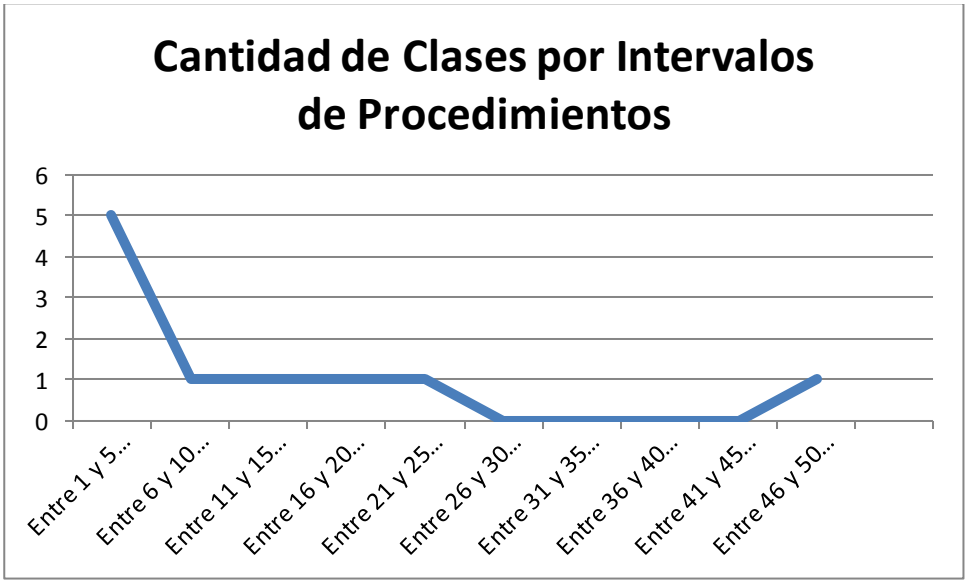


Figura 11: Representación de las clases según la cantidad de operaciones.

Teniendo en cuenta el umbral de cantidad de operaciones con que cuenta cada clase, está establecido que un tamaño de clase pequeño es aquel que tiene un valor menor o igual que 3, un tamaño medio para aquellas clases que tengan entre 4 y 12 operaciones y un tamaño de clase grande es aquel que es mayor o igual que 16. Se concluye que la herramienta cuenta con 6 clases fundamentales cuyo promedio de cantidad de operaciones equivale a 18. Los valores de tamaño quedan distribuidos de la siguiente manera:

Umbral	Tamaño	Cantidad de Clases
Pequeño	<=3	0
Medio	>=4 y <=12	3
Grande	>=16	3

Tabla 14: Tamaño de clases.

Como se puede observar en la Tabla 14 el 50% de las clases analizadas están clasificadas de umbral Grande y el otro 50% de Medio, lo cual se considera un resultado positivo según los parámetros de calidad Responsabilidad, Complejidad de Implementación y Reutilización propuestos para esta métrica.

A continuación se grafican los resultados obtenidos de la aplicación de la métrica TOC.

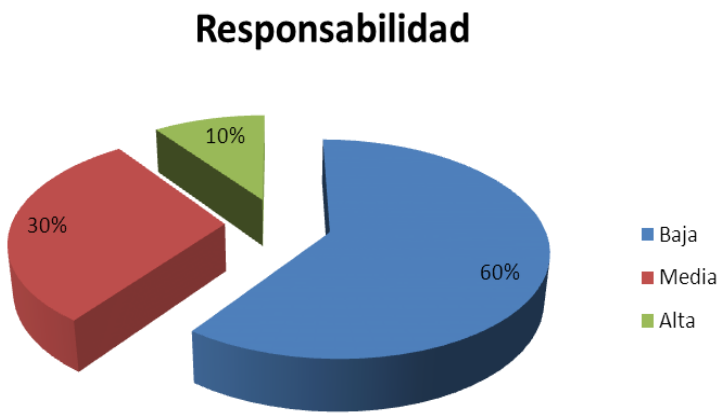


Figura 12: Resultados de la evaluación de la métrica TOC para el atributo de calidad Responsabilidad.

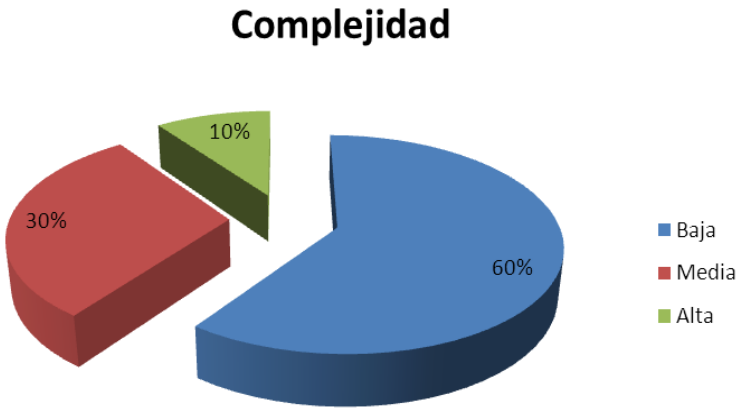


Figura 13: Resultados de la evaluación de la métrica TOC para el atributo de calidad Complejidad.

Figura 14: Resultados de la evaluación de la métrica TOC para el atributo de calidad Reutilización.

3.4.2. Relación entre Clases (RC)

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad [43][44]:

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Aumento del RC provoca aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Aumento del RC provoca aumento de la complejidad del mantenimiento de la clase.
Reutilización	Aumento del RC provoca disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Aumento del RC provoca aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 15: Atributos de calidad que evalúa RC.

Para la evaluación de dichos atributos de calidad, se definieron los siguientes criterios y categorías de evaluación:

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

Tabla 16: Criterios de evaluación de la métrica RC.

Resultados obtenidos:

Evaluando la cantidad de relaciones entre clases con que cuenta la herramienta “Generador de Scripts de Instalación”, según los criterios establecidos en la Tabla 16, se obtuvo que un 90% de las clases cuentan con ningún Acoplamiento y baja Complejidad de Mantenimiento y Cantidad de Pruebas, además de una alta Reutilización.

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mantenimiento	Reutilización	Cantidad de Pruebas
GestcatscriptController	0	Ninguno	Baja	Alta	Baja
GestgenscriptsController	4	Alta	Alta	Baja	Alta
Connection	0	Ninguno	Baja	Alta	Baja
DatScriptModel	0	Ninguno	Baja	Alta	Baja

NomTiposscriptModel	0	Ninguno	Baja	Alta	Baja
ScriptManager	0	Ninguno	Baja	Alta	Baja
DatScript	0	Ninguno	Baja	Alta	Baja
NomTiposscript	0	Ninguno	Baja	Alta	Baja
BaseDatScript	0	Ninguno	Baja	Alta	Baja
BaseNomTiposscript	0	Ninguno	Baja	Alta	Baja

Tabla 17: Instrumento de evaluación de la métrica RC.

Teniendo en cuenta la cantidad de dependencias entre las clases mostradas, está establecido que una cantidad de relaciones pequeña será igual a 1, una cantidad media será entre 1 y 3 relaciones y grande será más de 3 relaciones.

Intervalos	Cantidad de Clases
0 dependencias	9
1 dependencia	0
> 1 dependencia	1

Tabla 18: Cantidad de dependencias por clases.

Umbral	Relaciones	Cantidad de Clases
Pocas	<=1	9
Medias	>1 y <= 3	0
Muchas	>3	1

Tabla 19: Evaluación según cantidad de relaciones.

Como se puede observar en la tabla anterior el 90% de las clases están clasificadas de Pocas según sus relaciones, lo cual representa un resultado positivo según los atributos de Acoplamiento, Complejidad del Mantenimiento, Reutilización y Cantidad de Pruebas.

A continuación se grafican los resultados obtenidos al aplicar la métrica RC.

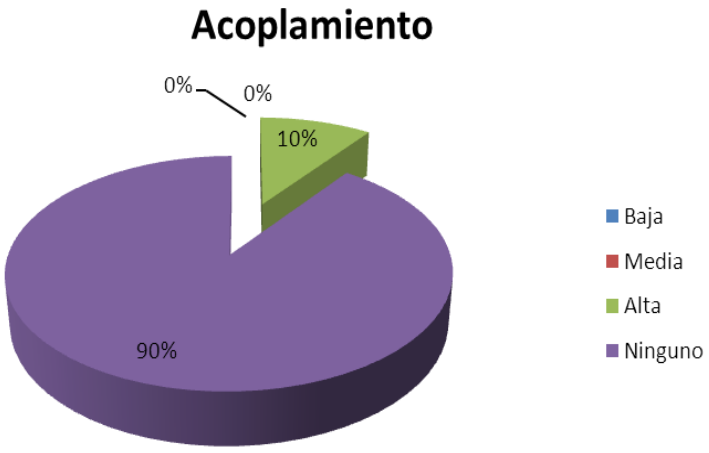


Figura 15: Resultados de la evaluación de la métrica RC para el atributo de calidad Acoplamiento.

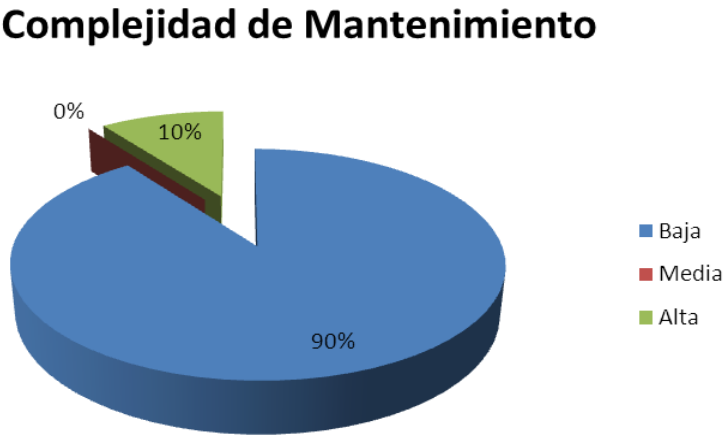


Figura 16: Resultados de la evaluación de la métrica RC para el atributo de calidad Complejidad de Mantenimiento.

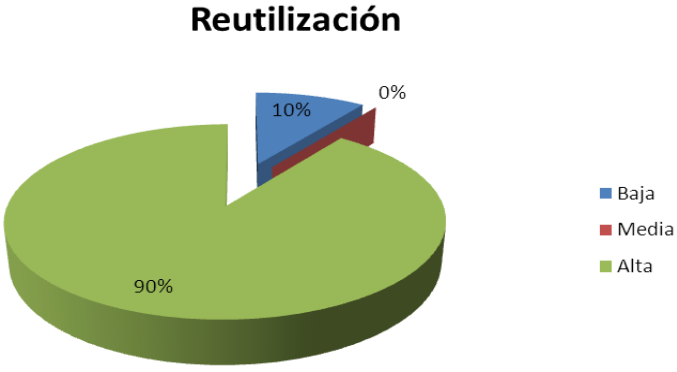


Figura 17: Resultados de la evaluación de la métrica RC para el atributo de calidad Reutilización.

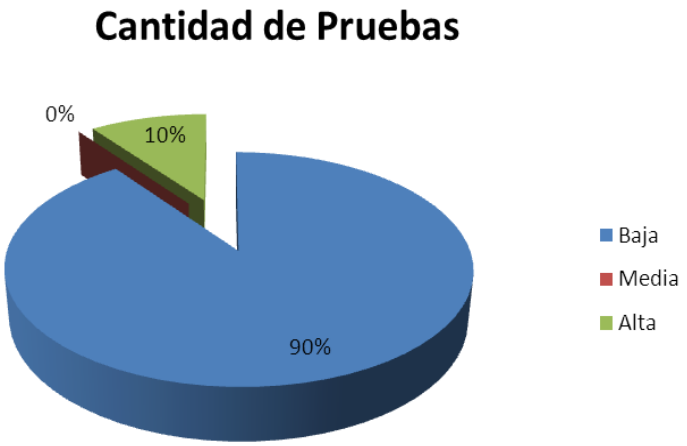


Figura 18: Resultados de la evaluación de la métrica RC para el atributo de calidad Cantidad de Pruebas.

3.4.3. Matriz interferencia de indicadores de calidad

En la investigación se define que más del 60% de las clases con Responsabilidad, Complejidad y Acoplamiento clasificados de Baja; más del 60% de las clases con Complejidad de mantenimiento y Cantidad de pruebas clasificadas de Media y más del 60% de las clases con Reutilización Alta o Media, serán considerados resultados Favorables y siendo No Favorables por debajo de 60%.

La matriz de interferencia de indicadores de calidad es una representación de los atributos de calidad y las métricas utilizadas para evaluar la calidad del diseño propuesto para el componente. Con ella se puede conocer si los resultados obtenidos de las relaciones atributo/métrica son positivos o no, llevando estos resultados a una escalabilidad numérica donde, si los resultados son positivos se le asigna el valor "1", si son negativos toma valor "0" y si no existe relación es considerada como nula y es representada con un guión simple (-). Luego se puede obtener un resultado general para cada atributo promediando todas sus relaciones no nulas [45].

Atributo/Métrica	TOC	RC	Promedio
Responsabilidad	1	-	1
Complejidad de Implementación	1	-	1
Reutilización	1	1	1
Acoplamiento	-	1	1
Complejidad de Mantenimiento	-	1	1
Cantidad de pruebas	-	1	1

Tabla 20: Resultados de la evaluación de la relación Atributo/Métrica.

Teniendo en cuenta los resultados obtenidos al aplicar las métricas de diseño TOC y RC, que se muestran en las Tabla 13 y 17, se construye el siguiente gráfico (Figura 19), donde se muestran los resultados obtenidos al evaluar los atributos de calidad involucrados en cada una de las métricas evaluadas.

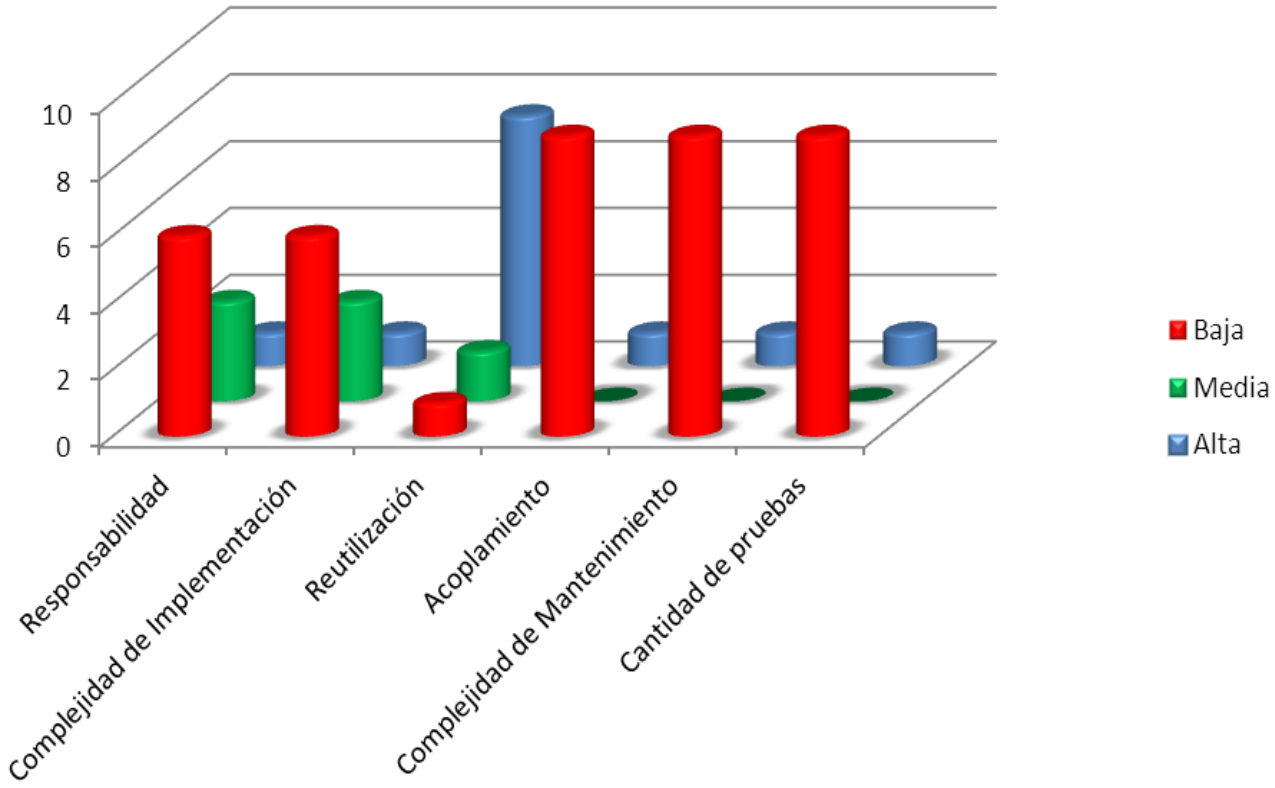


Figura 19: Resultados de la evaluación.

3.4.4. Valoración de los resultados de las métricas

Luego de aplicar de las métricas TOC y RC se han obtenido resultados positivos, esto evidencia que la solución está correctamente diseñada garantizando un diseño robusto con una alta reutilización. De igual manera se garantiza que la solución será fácil de mantener y/o modificar en caso de ser necesario. También se garantiza que la cantidad de pruebas necesarias para probar las clases de la solución.

3.5. Pruebas de Software

Las pruebas de software son un instrumento para determinar el status de la calidad de un software. Tienen como objetivo, además de descubrir errores, medir el grado en que el software cumple con los requerimientos definidos [46].

3.5.1. Pruebas de Caja Blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal, es un método de diseño de casos de prueba que usa la estructura del control de diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que (1) garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; (2) ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; (3) ejecuten todos los bucles en sus límites y con sus límites operacionales; y (4) ejerciten las estructuras internas de datos para asegurar su validez [47].

La prueba del camino básico es una técnica de prueba de caja blanca que permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de pruebas obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos unas vez cada sentencia del programa [47]. A continuación se aplica esta técnica a la solución propuesta, por lo que se procede a enumerar las sentencias de código y a partir del mismo se construye el grafo de flujo asociado.

```
function eliminarpaqueteAction() {
    $nombre_paquete = $this->_request->getPost('nombre_paquete'); (1)
    $id_paquete = $this->_request->getPost('id_paquete'); (1)

    $path = DIRECTORY_SEPARATOR . "herramientas" . DIRECTORY_SEPARATOR . "genscripts"; (1)
    $dir = $this->pathReal($path) . $this->xmlScPath(); (1)
    $paquetes = simplexml_load_file($dir);(1)
    $paquete = $paquetes->children(); (1)
    $cont = 0;(1)
    foreach ($paquete as $var) { (2)
        if (count($paquete) > 0){ (3)
            if ($var->nombre == $nombre_paquete && $var->version == ($id_paquete - $nombre_paquete)) { (4)
                unset($paquete[$cont]); (5)
            }
            $cont++; (6)
        }
    }(7)
    $paquetes->asXML($dir); (8)
    $xml = new DOMDocument(); (8)
    $xml->preserveWhiteSpace = false; (8)
    $xml->formatOutput = true; (8)
    $xml->load($dir); (8)
    $xml->save($dir); (8)
    $this->showMessage(1, 'El paquete fue eliminado satisfactoriamente.'); (8)
}
```

Figura 20: Código fuente de la funcionalidad Eliminar paquete.

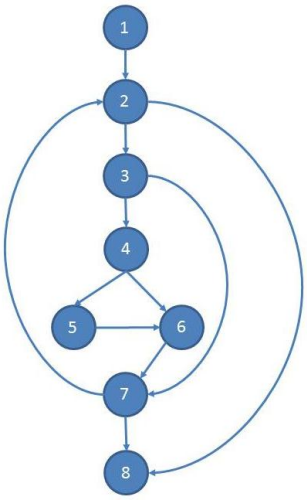


Figura 21: Grafo de flujo asociado a la funcionalidad Eliminar paquete.

Luego de haber construido el grafo se realiza el cálculo de la complejidad ciclomática mediante las tres fórmulas descritas a continuación, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto [47].

1. V(G) = (A-N)+2 // Siendo "A" la cantidad total de aristas y "N" la cantidad total de nodos.
   V(G) = (11-8)+2 = 5
2. V(G) = P+1 // Siendo "P" la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).
   V(G) = 4+1 = 5
3. V(G)=R // Siendo "R" la cantidad total de regiones, para cada fórmula "V(G)" representa el valor del cálculo.
   V(G) = 5

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, dando como resultado 5, lo que indica que existen 5 posibles caminos por donde el flujo puede circular y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo:

- Camino básico #1: 1-2-3-4-5-6-7-2-8
- Camino básico #2: 1-2-8
- Camino básico #3: 1-2-3-4-6-7-2-8
- Camino básico #4: 1-2-3-4-6-7-8
- Camino básico #5: 1-2-3-7-2-8

Caso de prueba para el camino básico # 1:

Descripción: Sin datos de entrada. Representa el cumplimiento de todas las condiciones definidas en la implementación del método.

Condición de ejecución: El arreglo de paquetes tiene varios elementos y se cumple que:

$var->nombre == $nombre_paquete && $var->version == ($id_paquete - $nombre_paquete)

Es decir, que se encuentra el nombre y la versión del paquete especificado.

Entrada: Nada.

Resultados esperados: Se elimina correctamente el paquete especificado.

Caso de prueba para el camino básico # 2:

Descripción: Sin datos de entrada. Representa el incumplimiento de todas las condiciones definidas en la implementación del método y sin elementos en el arreglo de paquetes.

Condición de ejecución: El arreglo de paquetes no tiene elementos y por ende no se cumple la condición:

$var->nombre == $nombre_paquete && $var->version == ($id_paquete - $nombre_paquete)

Entrada: Nada.

Resultados esperados: No se encuentra el paquete especificado y por ende no se elimina.

Caso de prueba para el camino básico # 3:

Descripción: Sin datos de entrada. Representa el cumplimiento parcial de las condiciones definidas en la implementación del método.

Condición de ejecución: El arreglo de paquetes tiene elementos pero no se cumple la condición:

$var->nombre == $nombre_paquete && $var->version == ($id_paquete - $nombre_paquete)

Es decir, que no se encuentra el paquete especificado.

Entrada: Nada.

Resultados esperados: No se encuentra el paquete especificado y por ende no se elimina.

Caso de prueba para el camino básico # 4:

Descripción: Sin datos de entrada. Representa el cumplimiento parcial de las condiciones definidas en la implementación del método.

Condición de ejecución: El arreglo de paquetes tiene un solo elemento y no cumple con la condición:

$var->nombre == $nombre_paquete && $var->version == ($id_paquete - $nombre_paquete)

Es decir, el único paquete que se encuentra en el arreglo no coincide con el paquete especificado.

Entrada: Nada.

Resultados esperados: No se encuentra el paquete especificado y por ende no se elimina.

Caso de prueba para el camino básico # 5:

Descripción: Sin datos de entrada. Representa el incumplimiento de todas las condiciones definidas en la implementación del método pero el arreglo de paquetes contiene más de un elemento.

Condición de ejecución: El arreglo de paquetes tiene más de un elemento pero ninguno cumple con la condición:

$var->nombre == $nombre_paquete && $var->version == ($id_paquete - $nombre_paquete)

Entrada: Nada.

Resultados esperados: No se encuentra el paquete especificado y por ende no se elimina.

3.5.2. Pruebas de Caja Negra

Las pruebas de caja negra son las que se aplican a la interfaz del software y se centran en los requisitos funcionales del sistema, permitiendo derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa [42].

Para la aplicación de este tipo de pruebas se diseñaron un conjunto de escenarios de pruebas con el objetivo de evaluar el cumplimiento de los requisitos del sistema. A continuación (Tablas 21, 22 y 23) se muestra el caso de prueba del requisito “Generar script de estructura”, los restantes casos de prueba se muestran en los anexos (Anexo 1) y en el expediente del proyecto Sauxe, específicamente en la carpeta Diseños Casos de Prueba de la herramienta Generador de scripts de instalación.

Condiciones de ejecución:

- El usuario debe haber sido autenticado en el sistema.
- El usuario debe seleccionar el menú Inicio, Herramientas, Generador de scripts.
- El usuario debe haber conectado la base de datos.
- El usuario debe de haber creado un paquete.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Generar script de estructura	Permite seleccionar un script de estructura contenido en un paquete y configurar la información que se desee generar.	EP 1.1: Generar script	– Seleccionar un paquete de los que se muestran en el sistema. – Desplegar el árbol del paquete seleccionado. – Seleccionar el tipo de script de Estructura. – Desplegar el esquema o los esquemas deseados. – Desplegar el tipo o los tipos de objetos a generar por el script.

				– Dar clic sobre los **checkbox** de los objetos a generar por el script. – Dar clic en el botón **Generar script**.

Tabla 21: Escenario de prueba del requisito "Generar script de estructura".

No	Nombre de campo	Tipo	Válido	Inválido
1	Objeto	Campo de texto	Representa los objetos tablas, funciones, disparadores, índices y secuencias.	N/A

Tabla 22: Descripción de variables del escenario de prueba del requisito "Generar script de estructura".

Id del escenario	Escenario	Variable 1 Objeto	Respuesta del sistema	Resultado de la prueba
EP 1.2	Elementos de la base de datos	Tabla (conf_entidades)	Es posible configurar un script de estructura generando el mismo.	Satisfactorio

Tabla 23: Juegos de datos a probar en escenario de prueba del requisito "Generar script de estructura".

3.5.3. Valoración de los resultados de las pruebas de software

Luego de realizar las pruebas de software de caja blanca y caja negra se obtuvo como resultado que la solución implementada cumple con los requerimientos definidos y que la ocurrencia de errores en los escenarios identificados es nula, garantizando así un software correctamente funcional.

3.6. Conclusiones del capítulo

Se implementó la Herramienta para la Generación de Scripts de Instalación en el marco de trabajo Sauxe logrando reducir el tiempo empleado en la generación de estos scripts de forma manual así como la ocurrencia de errores en este proceso. Se validó la solución desarrollada empleando técnicas para la validación de los requisitos, métricas para la validación del diseño y pruebas de caja blanca y caja negra para la validación de la aplicación. Todo esto permitió garantizar que se ha obtenido un software que responde a las necesidades del usuario y correctamente funcional.

CONCLUSIONES GENERALES

Al concluir la investigación para el desarrollo de la Herramienta para la Generación de Script de Instalación en el marco de trabajo Sauxe, se pudo arribar a las siguientes conclusiones:

- Se ha realizado un estudio del estado del arte de la generación de scripts a nivel mundial permitiendo identificar y especificar los conceptos que son imprescindibles para un mejor entendimiento de la investigación. Además, este estudio permitió identificar los posibles puntos de reutilización de las herramientas que actualmente brindan la generación de scripts como una de sus funcionalidades.
- Se realizó el Análisis y Diseño de la Herramienta de Generación de scripts de instalación que sirvieron de base para llevar a cabo correctamente el proceso de implementación de la misma. Se elaboró el modelo conceptual de la solución, donde se identificó la relación que debe existir entre cada uno de los conceptos identificados. Se realizó la descripción de los requisitos funcionales y no funcionales identificados durante el proceso de Ingeniería de Requisitos. Como parte del modelo de diseño se identificaron los patrones de diseño a utilizar en el desarrollo de la solución. Por último, se elaboró y describió el Diagrama Clases y el Modelo de Datos de la solución.
- Se implementó la Herramienta para la Generación de Scripts de Instalación en el marco de trabajo Sauxe logrando reducir el tiempo empleado en la generación de estos scripts de forma manual así como la ocurrencia de errores en este proceso. Se validó la solución desarrollada empleando técnicas para la validación de los requisitos, métricas para la validación del diseño y pruebas de caja blanca y caja negra para la validación de la aplicación. Todo esto permitió garantizar que se ha obtenido un software que responde a las necesidades del usuario y correctamente funcional.

RECOMENDACIONES

Se recomienda que en próximas versiones de la Herramienta para la Generación de Script de Instalación en el marco de trabajo Sauxe, se generen los scripts de permisos teniendo en cuenta los permisos asignados en la base de datos.

BIBLIOGRAFÍA

[1] «Conceptos sobre los procedimientos almacenados del sistema de replicación». [En línea]. Disponible: http://msdn.microsoft.com/es-es/library/ms147302.aspx. [Accedido: 26-ene-2013].

[2] Ricardo Pérez Velázquez, «Instalador Web para el Sistema Integral de Gestión Cedrux», Universidad de las Ciencias Informáticas, 2010.

[3] L. K. C. J. P. Laudon, *Sistemas De Información Gerencial: Administración de la empresa digital*. 2008.

[4] O´b. J. A., *Sistemas de Información Gerencial*. 2001.

[5] «Introducción a las características y herramientas (SQL Server 2008)». [En línea]. Disponible: http://msdn.microsoft.com/es-es/library/bb500397(v=sql.105).aspx. [Accedido: 28-ene-2013].

[6] «Generar un script (SQL Server Management Studio)». [En línea]. Disponible: http://msdn.microsoft.com/es-es/library/ms178078(v=sql.105).aspx. [Accedido: 26-ene-2013].

[7] «PostgreSQL: Documentation: 8.1: Backup and Restore». [En línea]. Disponible: http://www.postgresql.org/docs/8.1/static/backup.html. [Accedido: 28-ene-2013].

[8] E. e. d. d. d. PostgreSQL, *Manual del usuario de PostgreSQL*. 2008.

[9] «Database 11g | Oracle Database 11g | Oracle». [En línea]. Disponible: http://www.oracle.com/us/products/database/overview/index.html. [Accedido: 28-ene-2013].

[10] «Export». [En línea]. Disponible: http://docs.oracle.com/cd/B10501_01/server.920/a96652/ch01.htm#1004671. [Accedido: 28-ene-2013].

[11] Centro de Informatización de Gestión de Entidades, Subdirección de Producción, «Modelo de Desarrollo de Software». Universidad de las Ciencias Informáticas, 2012.

[12] K. Prada Nicot, H.y.S.G., «Desarrollo de los componentes Puesto de Trabajo y Pagos Adicionales del subsistema Capital Humano integrado al sistema integral de gestión CEDRUX», 2009.

[13] «Apache Subversion». [En línea]. Disponible: http://subversion.apache.org/. [Accedido: 28-ene-2013].

[14] «RapidSVN». [En línea]. Disponible: http://www.rapidsvn.org/. [Accedido: 28-ene-2013].

[15] «NetBeans Docs & Support». [En línea]. Disponible: http://netbeans.org/kb/index.html. [Accedido: 23-ene-2010].

[16] Dr. Oiner Gómez Baryolo, «Solución informática de autorización en entornos multientidad y multisistema», Universidad de las Ciencias Informáticas, 2010.

[17] R. Frederick, S. «Cutter» Colin y Blades, y Shay, *Learning ExtJS*. 2008.

BIBLIOGRAFÍA

[18] Esser, S., *Secure Programming with the Zend-Framework*. 2009.

[19] Y. Aquino, A. y L., «Implementación del módulo de Contabilidad General del Sistema Integral de Gestión Cedrux», 2009.

[20] José Enrique González Cornejo, «¿Qué es UML?» [En línea]. Disponible: http://www.docirs.cl/uml.htm. [Accedido: 25-may-2012].

[21] M. A. Friedhelm Betz, N. L. Antony Dovgal, G. R. Hannes Magnusson, J. V. Damien Seguy, y otros, *Manual de PHP*. 2012.

[22] Martínez, J.S., «Una Arquitectura para una Herramienta de Patrones de Diseño», 1999.

[23] Kaisler, *Software Paradigms*. 2005.

[24] Pérez Mariñán, P., *Patrones de Diseño*. 2007.

[25] «El Patrón Singleton». [En línea]. Disponible: http://msdn.microsoft.com/es-es/library/bb972272.aspx. [Accedido: 04-jun-2013].

[26] Steve Burbeck, «Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)», *st-www.cs.illinois.edu*. [En línea]. Disponible: http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html. [Accedido: 31-may-2013].

[27] «Modelo Conceptual». Universidad de Buenos Aires, 2009.

[28] Ing. Ritzaida Tomé, «Captura de Requisitos». Instituto Universitario Politécnico «Santiago Mariño», oct-2011.

[29] Ing. Juan Carlos González Moreno, «Análisis de Requisitos». Universidad de Vigo, Pontevedra, España, 2008.

[30] Carlos Mario Zapata, Carolina Palacio, y Natalí Olaya, «UNC-Analista: Hacia la captura de un corpus de requisitos». Escuela de Ingeniería de Antioquia, Medellín, Colombia, jun-2007.

[31] Ian Sommerville, *Software Engineering*, 9th ed. Pearson Education, Inc., 2006.

[32] Roger Pressman, *Software Engineering, a practical approach*, 5th ed. Mc-Graw Hill, 2002.

[33] «Osmosislatina.com», *Osmosislatina.com*. [En línea]. Disponible: http://www.osmosislatina.com/lenguajes/uml/clasesob.htm. [Accedido: 08-jun-2012].

[34] «Métodos de desarrollo de software». Massachusetts Institute of Technology, 2002.

[35] «Técnicas de validación de requisitos | Marco de Desarrollo de la Junta de Andalucía». [En línea]. Disponible: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/419. [Accedido: 04-jun-2013].

BIBLIOGRAFÍA

[36] Karenny Brito Acuña, «RUP Diseño e implementación», *eumed.net*. [En línea]. Disponible: http://www.eumed.net/libros/2009c/584/RUP%20Diseno%20e%20implementacion%20del%20sistema.htm. [Accedido: 09-abr-2012].

[37] SPARX Sistem, «UML Demployment Diagram», *SPARX Sistem*. [En línea]. Disponible: http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html. [Accedido: 23-feb-2012].

[38] Scott W. Ambler, «UML 2 Component Diagramming Guidelines», *Agile Modeling*, 2012. [En línea]. Disponible: http://www.agilemodeling.com/style/componentDiagram.htm.

[39] «Sparx Systems - Tutorial UML 2 - Diagrama de Componentes», *Sparx Systems*, 2007. [En línea]. Disponible: http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html. [Accedido: 17-may-2013].

[40] Damián Pérez Alfonso, *Normas y estándares de codificación*. Universidad de las Ciencias Informáticas, 2012.

[41] IEEE. 2007, «Introducción a la Ingeniería de Requisitos. Ingeniería de software». 2007.

[42] Roger Pressman, *Software Engineering. A practitioner´s Approach*, 7th Edition. 2009.

[43] «Métrica de diseño - EcuRed». [En línea]. Disponible: http://www.ecured.cu/index.php/M%C3%A9trica_de_dise%C3%B1o. [Accedido: 01-jun-2013].

[44] «Unidad 5 | IngSoftwareII-CUFM». [En línea]. Disponible: http://cufmingsoftware.wordpress.com/estandares-de-diseno/. [Accedido: 01-jun-2013].

[45] T. F. G. Miguel La Llave Iglesias, «Solución informática para la Compartimentación de la información en los sistemas que utilizan el Sistema de Gestión Integral de Seguridad ACAXIA.», Universidad de las Ciencias Informáticas, 2012.

[46] «Gestión de Calidad y Pruebas de Software», *pruebasdesoftware.com*, 30-may-2012. [En línea]. Disponible: http://pruebasdesoftware.com/laspruebasdesoftware.htm. [Accedido: 26-ene-2013].

[47] Laurie Williams, *Software Engineering*. North Carolina State University.

ANEXOS

Anexo 1

Diseño del caso de prueba Adicionar paquete

Condiciones de ejecución:

- El usuario debe haber sido autenticado en el sistema.
- El usuario debe seleccionar el menú Inicio, Herramientas, Generador de scripts.
- El usuario debe haber conectado la base de datos.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Adicionar paquete	Permite adicionar un paquete en el sistema.	EP 1.1: Adicionar un paquete correctamente	– Dar clic en el botón **Adicionar** para adicionar un nuevo paquete. – El sistema muestra la Interfaz **Adicionar paquete**. – Insertar el nombre, la versión y una descripción del paquete. – Dar clic en el botón **Aceptar.**
		EP 1.2: Adicionar un paquete con datos inválidos	– Dar clic en el botón **Adicionar** para adicionar un nuevo paquete. – El sistema muestra la Interfaz **Adicionar paquete**. – Insertar datos inválidos – Dar clic en el botón **Aceptar.** – Se muestra un mensaje de error.

Tabla 24: Escenario del caso de prueba Adicionar paquete.

No	Nombre de campo	Tipo	Válido	Inválido
1	Nombre	campo de texto	Cadena de caracteres que contengan letras, números y caracteres especiales.	

2	Versión	campo de texto	Cadena de caracteres que contengan números y el carácter especial punto.	Letras y caracteres especiales.
3	Descripción	campo de texto	Cadena de caracteres que contengan letras, números y caracteres especiales.	

Tabla 25: Descripción de variables del caso de prueba Adicionar paquete.

Id del escenario	Escenario	Variable 1 Nombre	Variable 2 Versión	Variable 3 Descripción	Respuesta del sistema	Resultado de la prueba
EP 1.2	Adicionar paquete	Prueba	2.2	Es una prueba al sistema	Es posible adicionar el paquete.	Satisfactorio
		N/A	N/A	Es una prueba de campos vacíos	El sistema emite un mensaje que los campos nombre y versión son obligatorios y señala de color rojo los mismos.	Satisfactorio
		Prueba2(	2.29s	Es una prueba de datos incorrectos.	El sistema no deja que en el campo versión se pueda introducir letras.	Satisfactorio
		Prueba	2.2	Es una prueba de coincidencia de datos insertados	El sistema muestra un mensaje que existe un paquete con el mismo nombre y la versión.	Satisfactorio

Tabla 26: Juego de datos a probar del caso de prueba Adicionar paquete.

Diseño del caso de prueba Conectar a la base de datos

Condiciones de ejecución:

- El usuario debe haber sido autenticado en el sistema.
- El usuario debe seleccionar el menú Inicio, Herramientas, Generador de scripts.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Conectarse a la base de datos	Permite conectarse a la base de datos donde serán generados los script de instalación.	EP 1.1: Conectarse a la base de datos correctamente.	– Insertar el servidor, el puerto, el usuario, la contraseña y el nombre de la base de datos. – Dar clic sobre el botón **Aceptar.** – El sistema muestra un mensaje de confirmación.
		EP 1.2: Interfaz Autenticación de servidores.	– Insertar datos incorrectos. – Dar clic sobre el botón **Aceptar.** – El sistema muestra un mensaje de error.

Tabla 27: Escenario del caso de prueba Conectar a la base de datos.

No	Nombre de campo	Tipo	Válido	Inválido
1	Servidor	campo de texto	Cadena de caracteres que contengan letras, números y el carácter especial punto.	Caracteres especiales menos el punto.
2	Puerto	campo de texto	Cadena de caracteres que contengan números.	Letras y caracteres especiales.
3	Usuario	Campo de texto	Cadena de caracteres que contengan letras, números y el carácter especial guión bajo.	Caracteres especiales menos el guión bajo.

4	Contraseña	Campo de texto	Cadena de caracteres que contengan letras, números y caracteres especiales.	N/A
5	Base de datos	Campo de texto	Cadena de caracteres que contengan letras, números y caracteres especiales.	N/A

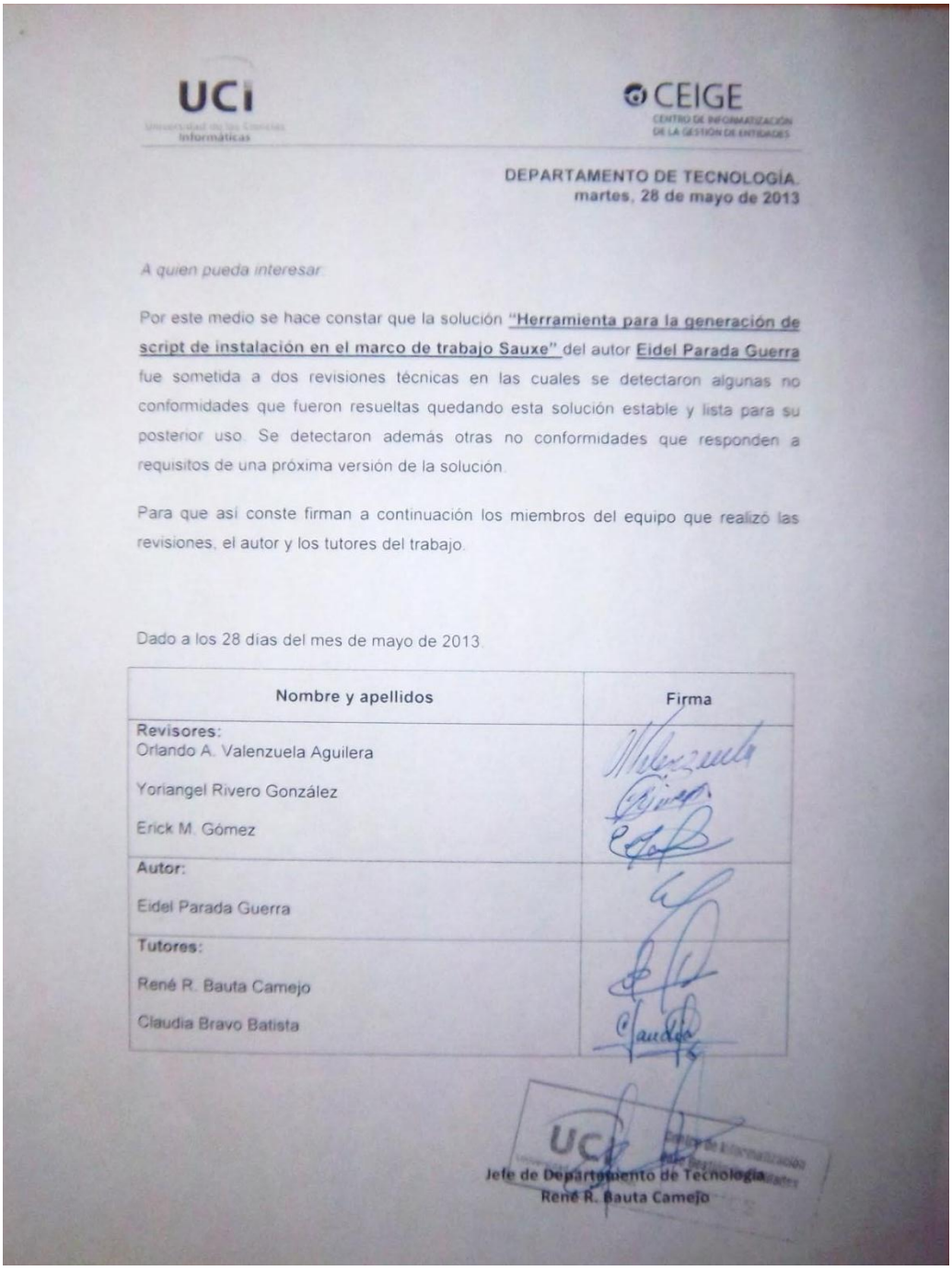
Tabla 28: Descripción de variables del caso de prueba Conectar a la base de datos.

Id del escenario	Escenario	Variable 1 Servidor	Variable 2 Puerto	Variable 3 Usuario	Variable 4 Contraseña	Variable 5 Base de datos	Respuesta del sistema	Resultado de la prueba
EP 1.2	Autenticación de servidores.	localhost	5432	postgres	postgres	sauxe	Es posible conectarse a la base de datos insertando la cadena de caracteres.	Satisfactorio
		N/A	N/A	N/A	N/A	N/A	El sistema emite un mensaje que los campos servidor, puerto, usuario y contraseña son obligatorios y señala de color rojo los mismos.	Satisfactorio
		localhost	5432a	Postgres*	postgres	sauxe	El sistema no deja que en el campo servidor se introducir caracteres especiales solamente el punto, que en el puerto se pueda introducir letras o caracteres especiales y en el usuario se pueda introducir caracteres especiales solamente el guión bajo.	Satisfactorio

Tabla 29: Juego de datos a probar del caso de prueba Conectar a la base de datos.

Anexo 2

Acta de liberación de la Herramienta para la Generación de Scripts de Instalación en el marco de trabajo Sauxe.



UCI
Universidad de las Ciencias Informáticas

CEIGE
CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE ENTIDADES

DEPARTAMENTO DE TECNOLOGÍA.
martes, 28 de mayo de 2013

A quien pueda interesar:

Por este medio se hace constar que la solución **"Herramienta para la generación de script de instalación en el marco de trabajo Sauxe"** del autor **Eidel Parada Guerra** fue sometida a dos revisiones técnicas en las cuales se detectaron algunas no conformidades que fueron resueltas quedando esta solución estable y lista para su posterior uso. Se detectaron además otras no conformidades que responden a requisitos de una próxima versión de la solución.

Para que así conste firman a continuación los miembros del equipo que realizó las revisiones, el autor y los tutores del trabajo.

Dado a los 28 días del mes de mayo de 2013.

Nombre y apellidos	Firma
Revisores: Orlando A. Valenzuela Aguilera Yoriangel Rivero González Erick M. Gómez	
Autor: Eidel Parada Guerra	
Tutores: René R. Bauta Camejo Claudia Bravo Batista	

UCI
Jefe de Departamento de Tecnología
René R. Bauta Camejo

Figura 22: Acta de liberación.

GLOSARIO DE TÉRMINOS

[i] Siglas en inglés de structured query language (lenguaje de consulta estructurado).

[ii] Analysis Services: Herramienta de Microsoft SQL Server que ofrece funciones de procesamiento analítico en línea y minería de datos para aplicaciones de negocio inteligente.

[iii] Transact-SQL: Funciones SQL integradas de SQL Server.

[iv] SQL-dump: Utilidad de copia de seguridad de una base de datos PostgreSQL.

[v] GUI: Siglas de Grafic User Interface (Interfaz Gráfica de Usuario).

[vi] wxWidgets: Bibliotecas multiplataforma y libres, para el desarrollo de interfaces gráficas programadas en lenguaje C++.

[vii] JOIN: En lenguaje SQL permite combinar registros de dos o más tablas en una base de datos relacional.

[viii] Java Swing: Biblioteca gráfica para Java.

[ix] CakePHP: Marco de trabajo para desarrollo rápido en PHP.

[x] INSERT: En lenguaje SQL permite adicionar información a las tablas de una base de datos.

[xi] ERP: Enterprise Resource Planning o sistemas de Planificación de Recursos Empresariales.

[xii] Switch: Tipo de estructura de selección empleada en la programación de algoritmos.