# GENERACIÓN AUTOMÁTICA DE CÓDIGO DE INTERFACES CRUD, EN ENTORNOS WEB A PARTIR DE UNA BASE DE DATOS, PARA AMBIENTES DE SOFTWARE LIBRE

**JUAN FRANCISCO MENDOZA MORENO**

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA – UNAB
MAESTRÍA EN SOFTWARE LIBRE
CONVENIO UNAB – UOC
BUCARAMANGA
2013

# GENERACIÓN AUTOMÁTICA DE CÓDIGO DE INTERFACES CRUD, EN ENTORNOS WEB A PARTIR DE UNA BASE DE DATOS, PARA AMBIENTES DE SOFTWARE LIBRE

**JUAN FRANCISCO MENDOZA MORENO**

Trabajo de grado para optar al título de

**MAGÍSTER EN SOFTWARE LIBRE**

DIRECTOR:

**DANIEL ARENAS SELEEY**

MsC. Ciencias de la Computación

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA – UNAB
MAESTRÍA EN SOFTWARE LIBRE
CONVENIO UNAB – UOC
BUCARAMANGA
2013

**TABLA DE CONTENIDO**

AGRADECIMIENTOS 1

DEDICATORIA 2

INTRODUCCIÓN 3

1. 6

1.1. 6

1.2. 6

1.3. 7

1.4. 8

1.4.1. 8

1.4.2. 8

1.5. 8

1.6. 9

1.7. 10

2. 11

2.1. 11

2.2. 12

2.2.1. 13

2.2.2. 14

2.2.3. 14

2.2.4. 14

2.2.5. 14

2.2.6. 15

2.3. 16

2.3.1. 16

2.3.2. 16

2.3.3. 17

2.3.4. 18

2.3.5. 20

2.3.6. 21

2.3.7. 22

2.3.8. 22

2.3.9. 23

2.3.10. 23

2.4. 25

2.4.1. 25

2.4.2. 26

2.5. 26

2.6. 26

3. 27

3.1. 35

3.1.1. 35

3.1.2. 42

3.1.3. 61

4. 79

5. 81

5.1. 81

5.2. 82

6. 84

6.1. 85

6.2. 93

6.3. 107

6.3.1. 107

6.3.2. 142

**TABLA DE ILUSTRACIONES**


Ilustración 1. Taxonomía Temática (CDU - Clasificación Decimal Universal) 5

Ilustración 2. El proceso SCRUM (tomado de http://en.wikipedia.org/wiki/File:Scrum_process.svg) 16

Ilustración 3. Componentes de una herramienta CASE (Tomado de (PIATTINI, CALVO-MANZANO, CERVERA, & FERNÁNDEZ, 2004)) 18

Ilustración 4. Arquitectura ISO / IEC 14102 23

Ilustración 5. Bosquejo metodológico 26

Ilustración 6. Secuencia y duración de las tareas del proyecto 30

Ilustración 7. Diagrama de Gantt Cronograma Proyecto 33

Ilustración 8. Encabezado Página Web Software Libre de Boyacá (Fuente: http://softwarelibreboyaca.org) 34

Ilustración 9. Actividades Grupo Software Libre Boyacá (Fuente: ídem) 35

Ilustración 10. Proceso de Validación y Selección Fuente: (ROBERTO IMENES, 2006) 38

Ilustración 11. Opciones de DBMS de phpMaker 42

Ilustración 12. Utilidades generadas y lenguaje generado (php) 42

Ilustración 13. Lectura del diccionario de datos por parte de phpMaker 43

Ilustración 14. Manejo de temas de phpMaker 43

Ilustración 15. Esquema de seguridad y de roles de la aplicación generada en phpMaker 44

Ilustración 16. Posibilidades de exportación de phpMaker 44

Ilustración 17. Generador de menús de phpMaker 45

Ilustración 18. Ingeniería Inversa en Sybase para DBMS 46

Ilustración 19. Algunos motores de DBMS soportados por Sybase 47

Ilustración 20. Lenguajes de programación del código generado en Sybase 47

Ilustración 21. Grandes agrupaciones de modelos 48

Ilustración 22. Soporte de DBMS de Enterprise Architect 49

Ilustración 23. Modelo Físico de Datos en Enterprise Architect 50

Ilustración 24. Diagramas que se pueden generar en Enterprise Architect 50

Ilustración 25. Ingeniería Inversa a la Base de Datos (Fuente VisualParadigm) 52

Ilustración 26. Lenguajes de programación soportados por Visual Paradigm 53

Ilustración 27. Importar Modelo de Datos en Oracle SQL Developer 54

Ilustración 28. Diagramas Oracle SQL Developer 55

Ilustración 29. Diseño manual de una interfaz en VisualWade 56

Ilustración 30. Creación de una clase con Hibernate en Eclipse IDE 58

Ilustración 31. Modelamiento con Hibernate 58

Ilustración 32. Interfaz generada con Hibernate 59

Ilustración 33. Proyecto FredyCRUD bajo metodología SCRUM en plataforma Trello 61

Ilustración 34. Fases de Desarrollo Herramienta CASE 63

Ilustración 35. Modelamientos soportados por PowerDesigner (Fuente Ayuda en línea PowerDesigner) 64

Ilustración 36. Modelamiento general del proyecto en su fase de análisis y diseño 65

Ilustración 37. Modelo de Procesos del Negocio (Fuente PowerDesigner OnLine Help) 66

Ilustración 38. Correspondencia de los diagramas UML con fases de un proyecto 70

Ilustración 39. Programación del Módulo Listar Tablas del proyecto FredyCRUD en lenguaje JAVA 71

Ilustración 40. Definición de la clase Conexión del proyecto FredyCRUD como paradigma de POO 72

Ilustración 41. Diseño de la forma de ingreso para el proyecto FredyCRUD, mediante NetBeans IDE 73

Ilustración 42. Documentación de la Clase Conexión mediante JavaDoc 74

Ilustración 43. Clase de prueba a la clase Conexión mediante JUnit 74

Ilustración 44. Versionamiento de la Aplicación mediante Subversión en los servidores de Google 75

Ilustración 45. Encuesta en Línea para Evaluar la Percepción de la Herramienta a Desarrollar 83

Ilustración 46. Conformación Grupo Software Libre Boyacá por municipios (Fuente GSL-Boyacá) 90

Ilustración 47. Muestra poblacional GSL Boyacá (Fuente El Autor) 91

Ilustración 48. Resultados a la pregunta: ¿alguna vez ha desarrollado software? 91

Ilustración 49. Herramientas (y lenguajes) de desarrollo utilizadas 92

Ilustración 50. Respuestas a la pregunta ¿ha desarrollado software libre? 92

Ilustración 51. Herramientas de desarrollo de software libre utilizadas 93

Ilustración 52. Respuestas a la pregunta: ¿ha utilizado alguna herramienta para generar código de forma automática? 93

Ilustración 53. Herramientas utilizadas de generación automática de software 94

Ilustración 54. Respuestas a la pregunta ¿qué opinión le merece las herramientas de generación automática de software? 95

Ilustración 55. Ventajas de utilizar herramientas que generan código de forma automática 96

Ilustración 56. Desventajas al utilizar herramientas que generan código de forma automática 97

Ilustración 57. Etapa propicia a partir de la cual se genera automáticamente el código 97

Ilustración 58. Funciones que debe proporcionar el código generado 98

Ilustración 59. Tipos de plataformas para donde se debe generar automáticamente código 98

Ilustración 60. Utilidad del código generado de forma automática 99

Ilustración 61. Características exigidas por el código generado, en la etapa del mantenimiento 99

Ilustración 62. Funcionalidades de la herramienta propuesta 100

Ilustración 63. Usuarios de la herramienta 101

Ilustración 64. Criterios de calidad del software generado 102

Ilustración 65. Importancia de este tipo de herramientas para la comunidad 102

Ilustración 66. Razones de la importancia de este tipo de herramientas 103

Ilustración 67. Análisis de Requisitos proyecto FredyCRUD 104

Ilustración 68. Diagrama Jerárquico de Procesos FredyCRUD 105

Ilustración 69. Diagrama de Procesos del Negocio 106

Ilustración 70. Análisis de Servicios de Proceso 107

Ilustración 71. Matriz CRUD de FredyCRUD 107

Ilustración 72. Organigrama de un esquema de adquisición de un nuevo Sistema de Información 112

Ilustración 73. Diagrama de Comunicación del Negocio 113

Ilustración 74. Mapa de Procesos 114

Ilustración 75. Diagrama de Planeación de la Ciudad 114

Ilustración 76. Diagrama Arquitectura de la Aplicación 115

Ilustración 77. Diagrama Orientado al Servicio 116

Ilustración 78. Diagrama de Infraestructura Tecnológica 117

Ilustración 79. Diagrama General de Casos de Uso 118

Ilustración 80. Diagrama de Clases aplicación FredyCRUD: Conexión, Ingreso y ListaBD 119

Ilustración 81. Diagrama de Clases aplicación FredyCRUD: ServicioBD 119

Ilustración 82. Diagrama de Clases aplicación FredyCRUD: DatoTabla 120

Ilustración 83. Diagrama de Clases aplicación FredyCRUD: ListaTablas – Atributos 120

Ilustración 84. Diagrama de Clases aplicación FredyCRUD: ListaTablas – Métodos 121

Ilustración 85. Diagrama general de Clases para el diccionario de datos mySQL 121

Ilustración 86. Diagrama de clases diccionario datos mySQL (Vista detallada izquierda-superior) 122

Ilustración 87. Diagrama de clases diccionario datos mySQL (Vista detallada derecha-superior) 122

Ilustración 88. Diagrama de clases diccionario datos mySQL (Vista detallada izquierda-inferior) 123

Ilustración 89. Diagrama de clases diccionario datos mySQL (Vista detallada derecha-inferior) 123

Ilustración 90. Diagrama Estructura Compuesta FredyCRUD 124

Ilustración 91. Diagrama de Objetos FredyCRUD 125

Ilustración 92. Diagrama de Paquetes de FredyCRUD 125

Ilustración 93. Diagrama de Comunicaciones 126

Ilustración 94. Diagrama General de Secuencias 127

Ilustración 95. Diagrama de Secuencias Seleccionar BD 127

Ilustración 96. Diagrama de Secuencia Parametrizar Aplicación 128

Ilustración 97. Diagrama Secuencias Generador Automático de Código 128

Ilustración 98. Diagrama de Actividades 129

Ilustración 99. Diagrama de Estados 130

Ilustración 100. Diagrama de Interacción 131

Ilustración 101. Diagrama de Componentes FredyCRUD 132

Ilustración 102. Diagrama de Despliegue FredyCRUD 132

Ilustración 103. Diagrama Físico del Diccionario de Datos de mySQL 140

Ilustración 104. Modelo Lógico del Diccionario de Datos de mySQL 141

Ilustración 105. Modelo Conceptual del diccionario de datos de mySQL 142

Ilustración 106. Diagrama XML General 143

Ilustración 107. Diseño Interfaz Conexión BD 143

Ilustración 108. Interfaz de Login 144

Ilustración 109. Interfaz Escoger BD 144

Ilustración 110. Interfaz Personalizar y Generar Aplicación 145

**INDICE DE TABLAS**


Tabla 1. Principales actividades del objetivo 1 28

Tabla 2. Principales actividades objetivo 2 28

Tabla 3. Principales Actividades Objetivo 3 29

Tabla 4. Presupuesto del proyecto 33

Tabla 5. Requerimientos que debe reunir una herramienta de generación automática de código considerados por los desarrolladores de software libre de la región 40

Tabla 6. Diagnóstico comparativo de phpMaker 41

Tabla 7. Diagnóstico comparativo de PowerDesigner 46

Tabla 8. Diagnóstico Comparativo de Enterprise Architect 49

Tabla 9. Diagnóstico comparativo de Visual Paradigm for UML 51

Tabla 10. Diagnóstico Comparativo Oracle SQL Developer 54

Tabla 11. Diagnóstico comparativo Visualwade 56

Tabla 12. Diagnóstico Comparativo Hibernate 57

Tabla 13. Parámetros de simulación en SIMUL8 para los proceso de negocio de la aplicación 112

Tabla 14. Tabla de Ingeniería de Requerimientos 139

## AGRADECIMIENTOS

A Dios Todopoderoso fuente de vida y de amor.

A Luz permanente impulsadora de mi vida, a Juana, mi más bella motivación y a mi familia cuna desbordante de amor hacia a mí.

Al Ingeniero Daniel Arenas, director del proyecto, quien se convirtió en apoyo incondicional para cumplir con los objetivos de este trabajo.

**DEDICATORIA**

"*Nunca digas, nunca jamás…*" Ian Flemming.

Trabajo dedicado a Dios Todopoderoso, a mi familia y a la comunidad de software libre.

**INTRODUCCIÓN**

# E

l cambio genera oportunidades, la Ingeniería de Software no es la excepción, como resultado de la llamada "Crisis del Software", se generaron profundas reflexiones sobre la actividad de la ingeniería, que a su vez propusieron fundamentos teóricos, técnicas y buenas prácticas para poder crear software. En este orden de ideas, cabe mencionar a los lenguajes de cuarta generación, cuando se analizó la posibilidad de utilizar el "software para crear más software" de forma automática (o semiautomática). Es una gran oportunidad, por cuanto se empieza a ofrecer al desarrollador de software herramientas que minimizan su esfuerzo de programación, que permiten disminuir la posibilidad de cometer errores de programación y que le ofrecen un panorama de interés sobre el propósito del software a crear, en lugar de preocuparse por las especificidades de la herramienta de programación.

De otra parte, el conocimiento es un derecho, un deber y un bien de la humanidad, gracias a él, el ser humano ha podido concretar el don de la creación, el mundo actual es muy diferente debido a la intervención humana y en el futuro será aún más diferente. El aspecto diferenciador con respecto a otros seres vivos, precisamente

radica en el uso del cerebro humano, como repositorio de inteligencia (memoria) y como generador de nuevos conocimientos. Y más interesante aún, es la creación y el uso colectivo de la inteligencia. Cualquier ser humano puede trascender, incluso más allá de la muerte, por el conocimiento heredado que deja a sus congéneres y a las futuras generaciones. En la actualidad, el software se consolidó como herramienta propicia para generar y usar el conocimiento, para el apoyo en la toma de decisiones del ser humano. Corrientes socio-económicas parecen que ven de forma distorsionada este principio humano, el interés económico prima y propende para que el conocimiento sea restringido solo a una porción de la humanidad, aquella que tenga la solvencia económica para usufructuarlo. La protección de los derechos de autor se ha visto confundida con una irracional apropiación del conocimiento malversando el principio de patentar la autoría sobre el conocimiento. El movimiento del software libre tiene como objetivo dar a conocer las libertades de uso, creación y adaptación del software para que el conocimiento sea un derecho, un deber y un bien de la humanidad.

Sin embargo, la labor de desarrollar software es una tarea difícil que requiere esfuerzos por parte del desarrollador, en cuanto a tiempo e innovación intelectual; y para el caso de la comunidad de software libre, caracterizada, en su mayoría, por tener a entusiastas como miembros, la labor de creación de software es más difícil y a menudo se encuentran proyectos de software que no pudieron ser culminados.

El desarrollo de aplicaciones basadas en la recuperación de información de una base de datos, mediante un administrador de base de datos (DBMS), requiere de una serie de interfaces dirigidas al usuario, para que éste a su vez pueda realizar las cuatro operaciones más comunes: Crear, Recuperar, Actualizar y Borrar un registro de información, su acepción en el idioma inglés es más conocida como CRUD (*Create, Retrieve, Update, Delete*). Este tipo de aplicaciones, conocidas por algunos como sistemas de información, en realidad están conformadas en su mayoría por estas interfaces, para permitir al usuario manipular la información de la base de datos, de acuerdo a perfiles y permisos de acceso a los datos. El desarrollo de estas interfaces es muy repetitivo y puede ser extenuante, además existe la probabilidad de cometer errores de programación.

Precisamente, para generar de forma automática o semiautomática, este tipo de sistemas de información con interfaces CRUD, existe un gran repertorio de herramientas. Algunas de ellas generan el software a partir de diagramas UML, otras de una forma más fácil, permiten generar el software a partir de una base de datos sin necesidad de utilizar este tipo de diagramas. Este tipo de herramientas son poco comunes para ambientes de software libre.

El caso de investigación de este proyecto, precisamente radica en la anterior situación: Proponer a la comunidad de software libre una herramienta que permita generar de forma automática aplicaciones con interfaces CRUD, a partir de una base de datos, con el motor mySQL.

Con este software, la comunidad de software libre tendrá la oportunidad de generar aplicaciones de sistemas de información y a su vez poder modificar manualmente el código fuente resultante, para poder empoderar y personalizar la aplicación, de acuerdo con los resultados de las etapas iniciales del ciclo de vida de la ingeniería de software.

La aplicación propuesta en este proyecto, consta de cuatro grandes módulos: parametrización de la aplicación generada, conexión a la base de datos, interpretación del diccionario de datos y generación del código fuente en el lenguaje de programación escogido.

Además el proyecto hace un análisis de las herramientas de similar función, ya sean de software libre o de licencia privativa, para tener en cuenta aspectos técnicos que deben ser considerados en el diseño de este tipo de herramientas.

El estudio de investigación se aplicó a una muestra de población de la comunidad de software libre de Boyacá, para tener en cuenta sus intereses y expectativas con este tipo de proyectos de software y para su vez satisfacer primordialmente las necesidades de esta comunidad.

Para el desarrollo de software se siguieron metodologías para el desarrollo de herramientas CASE y metodologías de desarrollo ágil de software, concretamente SCRUM.

El documento está estructurado en tres grandes capítulos: El primer de ellos contempla los aspectos generales del proyecto, su formulación, objetivos, la justificación y la hipótesis. El segundo capítulo enfatiza el marco referencial, teniendo en cuenta el marco histórico, el estado de arte, el marco teórico y los instrumentos de recolección de información. Todos ellos enfocados a áreas como la Ingeniería de Software, las herramientas CASE y metodologías para el desarrollo de este tipo de software. El tercer capítulo trata sobre la metodología y su desarrollo propio del proyecto.

# 1. CONTEXTUALIZACIÓN DEL PROYECTO

## 1.1. Título

Generación automática de código de interfaces CRUD, en entornos Web a partir de una base de datos, para ambientes de software libre.

## 1.2. Tema

Generación automática de código de interfaces CRUD, en entornos Web a partir de una base de datos, para ambientes de software libre

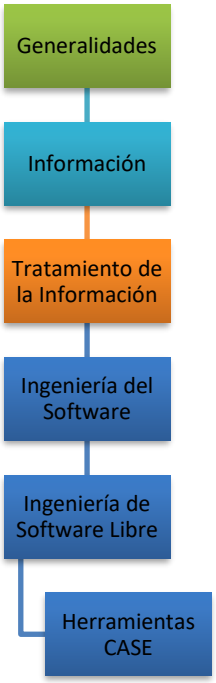



Ilustración 1. Taxonomía Temática (CDU - Clasificación Decimal Universal)

## 1.3. Formulación del Problema

Las herramientas que permiten el rápido desarrollo de código, que además ayuden al programador a minimizar errores de codificación, a centrar su atención en el objetivo del sistema de información más que en especificidades del lenguaje de programación y a concentrarse en la lógica del negocio, han facilitado el surgimiento de software para automatizar la generación de código a partir de una base de datos. Es decir, son herramientas utilizadas para la creación de sitios web o sistemas de información web, con facilidades de ver, editar, buscar, agregar y eliminar registros de las tablas de las bases de datos.

En el campo del software propietario existen muchas herramientas, la mayoría de ellas generan código en lenguajes de programación como PHP[1], para realizar tareas básicas y repetitivas, como la creación de código para mostrar al usuario un menú de opciones, proponer formularios para ingresar, borrar, consultar o actualizar datos de una tabla o conjunto de tablas, esquemas de seguridad o incluso la generación de reportes simples; código que una vez generado de manera automática, el desarrollador puede reutilizar para iniciar la codificación de su aplicación.

Pero este tipo de soluciones de software libre son escasas para el desarrollador, aunque existen editores o IDE (Ambientes de Desarrollo Integrados) que facilitan codificar las aplicaciones y ofrecen la posibilidad de evitar errores de codificación, no contemplan la generación automática de código para las operaciones básicas de un sistema de información.

Por lo tanto, la comunidad de software libre requiere de una herramienta bajo licencia GPL (o compatible) que sea flexible y que ofrezca opciones para generar el código fuente de aplicaciones en un lenguaje igualmente libre, que se adapte a las necesidades básicas de programación de aplicaciones, a partir de sistemas de gestión de bases de datos que utilice la comunidad (por ejemplo *PostgreSQL*[2] o la versión GPL de *mySQL*[3]). Además, el código generado debe ser limpio, fácil de entender, documentado, estandarizado y con facilidades de personalizar y reutilizar, de tal forma que permita continuar con el desarrollo de la aplicación. De esta forma, el desarrollador experimentado puede ahorrar tiempo y el novato se motiva en crear este tipo de aplicaciones, reduciendo la complejidad en las etapas del desarrollo del software.

Los proyectos de software libre, como *phpMyAdmin*[4] y *pgAdmin*[5], proponen herramientas que facilitan el trabajo del desarrollador, pero se ven limitadas a la gestión de una base de datos, sin apoyar la generación de código para las interfaces básicas de la aplicación, como el ingreso, la modificación, la eliminación o la consulta de los datos a través de formularios.

Otros proyectos de software libre, como *ArgoUML*[6], son capaces de generar código de forma automática, a partir de un modelo de clases previamente construido bajo estándares UML, para diseños orientados a

[1] PHP. Preprocesador de Hipertexto. http://www.php.net/manual/es/intro-whatis.php. Recuperado Noviembre 2011
[2] PostgreSQL. http://www.postgresql.org/about/. Recuperado Noviembre 2011
[3] MySQL Community Edition. http://www.mysql.com/products/community/. Recuperado Noviembre 2011
[4] phpMyAdmin. http://www.phpmyadmin.net/home_page/index.php. Recuperado Noviembre 2011
[5] pgAdmin. http://www.pgadmin.org/. Recuperado Noviembre 2011
[6] argoUML. http://argouml.tigris.org/. Recuperado Noviembre 2011

objetos. Sin embargo, para algunos proyectos de desarrollo web, su punto de partida puede darse a partir de los repositorios de datos.

El proyecto se centra en dotar a la comunidad de software libre de una herramienta (acorde a su licenciamiento) que genere código de forma automática para aplicaciones en entornos Web, a partir de una base de datos, de tal forma que reduzca la complejidad del desarrollo de software.

## 1.4. Objetivos

### 1.4.1. Objetivo General

Desarrollar una herramienta que permita la generación automática de código de interfaces CRUD, en entornos Web a partir de una base de datos, para ambientes de software libre.

### 1.4.2. Objetivos Específicos

- Identificar los requerimientos que debe reunir una herramienta de generación automática de código considerados por los desarrolladores de software libre de la región
- Realizar un diagnóstico comparativo del uso de herramientas actuales, de software libre o propietario, para la generación automática de código, a partir de una base de datos
- Crear una herramienta de automatización para la generación de código de interfaces CRUD, a partir de una base de datos, en ambientes de desarrollo de software libre

## 1.5. Resultados Esperados

- **Primer objetivo**: Documento de requerimientos funcionales y no funcionales para desarrollar una herramienta de generación automática de código. Estos requerimientos son obtenidos aplicando instrumentos de recolección de información a desarrolladores de software libre de la región.
- **Segundo objetivo**: Documento de diagnóstico comparativo del uso de herramientas actuales, de software libre o propietario, para la generación automática de código, a partir de una base de datos.
- **Tercer objetivo**: Herramienta de software para la generación automática de código de interfaces CRUD, a partir de una base de datos, en ambientes de desarrollo de software libre
- **Socialización**: Publicar la herramienta en un sitio web de colaboración para proyectos de software libre.

## 1.6. Justificación

Para algunos desarrolladores de software libre, les resulta paradójico no poder generar código de forma automática, en especial en lo referente al desarrollo de código de formularios, menús y reportes básicos a partir de una base de datos, que a su vez harán parte de la aplicación final del usuario; porque no existe la suficiente oferta de software libre que permita realizar esta labor. De otra parte, el software propietario ofrece gran variedad de herramientas de este tipo que facilitan la labor al programador.

La Ley de Parkinson aplicada a la arquitectura de computadores, dice que existe un recurso valioso y que nunca es suficiente, hace referencia a la memoria principal del computador[7]. De forma similar, en la Ingeniería del Software se podría aplicar esta misma ley con un recurso muy valioso y que nunca es suficiente, es decir el tiempo. En este orden de ideas, cada uno de los colaboradores de un proyecto de software busca formas, métodos y mecanismos que permitan ahorrar tiempo en su labor.

Por ejemplo, herramientas utilizadas en la etapa del análisis de requerimientos de información han avanzado mucho en el campo del software libre. Se pueden "dibujar" diagramas de clases, de usos, de estados, entre otros, que dan la opción de generar código de forma automática de tal forma que ahorra tiempo, esfuerzo y minimiza errores de codificación del programador. La gestión de la base de datos no es la excepción, muchas herramientas permiten la generación automática de código para la creación de bases de datos, de tablas, de índices, de relaciones, entre otros.

Pero al pensar en el desarrollo de interfaces del usuario, en especial tareas básicas que el mismo usuario puede hacer al interactuar con la base de datos, las posibilidades que ofrece el software libre están muy limitadas, caso diferente al software propietario que ofrece variadas herramientas, que inclusive puede generar código de lenguajes de programación libre.

Son tareas rutinarias como crear formularios o interfaces para que el usuario pueda adicionar, eliminar, actualizar o consultar la información de una tabla o un conjunto de tablas de una base de datos. Así como la configuración de seguridad para determinar el acceso a la información de acuerdo con niveles y permisos del usuario. También, la generación de reportes simples o incluso la generación de menús que permitan acceder de forma ordenada y lógica a las diferentes opciones del sistema de información.

Con el uso de este tipo de herramientas, el desarrollador ahorrará esfuerzo de programación, tiempo, recursos, minimizará errores y centrará su atención en la lógica del negocio, sin distraerse en especificidades del lenguaje de programación utilizado.

Este tipo de herramientas no generará la totalidad del código fuente de una aplicación, se limitará a la construcción de formularios para las tareas básicas y rutinarias, para que posteriormente el desarrollador pueda reutilizar este código en la culminación del proyecto de desarrollo.

[7] TANENBAUM, Andrew S. *Sistemas Operativos Modernos*. Segunda Edición. Prentice Hall. Pearson Educación, México 2003. ISBN 970-26-0315-3. Página 189

La experiencia de muchos grupos y proyectos de desarrollo de sistemas de información sobre entornos web en el ámbito de software libre, demuestra la necesidad de disponer de este tipo de herramientas.

## 1.7. Hipótesis

El uso de una herramienta de generación automática de software para interfaces CRUD, reducirá el tiempo de desarrollo de aplicaciones libres en entornos Web que requieren de una base de datos.

# 2. MARCO REFERENCIAL

## 2.1. Marco Histórico

Entre muchos logros e inventos, la historia informática reconoce a John Von Neumann[8], como el pionero de la computación moderna, porque incluía un nuevo concepto: el programa almacenado, es decir, un programa que reside dentro de la memoria del computador, para que éste a su vez ejecute sus instrucciones, sin necesidad de volverlo a escribir. A partir de este momento, el foco de interés no se centra únicamente en el hardware de la máquina, sino que se concibe al computador como una dualidad de hardware y software. Sin embargo, no hay que desconocer el trabajo que mucho tiempo atrás realizó Ada Augusta Byron King[9], a quien se le reconoce el haber descrito la máquina analítica de Charles Babbage[10] y a continuación escribir una serie de pasos para calcular los números de Bernoulli[11] y operaciones trigonométricas. La historia reconocería tiempo después, como el primer programador a Ada Byron.

En los años sucesivos a las investigaciones de Von Neumann, el software del computador se convirtió en la tecnología individual más importante del mundo (PRESSMAN, 2009), pero a su vez, como lo expresa Pressman[12], el software "es uno de los ejemplos principales de la ley de consecuencias imprevistas".

Aparecieron entonces muchos entusiastas (hackers[13]) desarrolladores de software, personas que creaban programas informáticos por diversión o con el objetivo de adquirir más conocimiento y de esta forma compartirlo con sus homólogos. Dentro de este grupo, cabe mencionar a Dennis Ritchie[14] y Ken Thompson[15], creadores del sistema operativo UNIX y del lenguaje de programación C; generación de científicos que concebían el desarrollo de software como un ejercicio investigativo y no como industria económica, porque el hardware era el bastión económico de las grandes compañías que se conformaron, el software era un aditamento más del hardware.

Poco tiempo después, algunos entusiastas de software, como Bill Gates[16], vieron en el software una industria muy prometedora, incluso mayor que la industria del hardware. El software ya no era considerado parte del

---

[8] VON NEUMANN, John. 28 de diciembre de 1903 – 8 febrero de 1957. Austro-húngaro. Matemático

[9] BYRON KING, Ada Augusta. 10 de diciembre de 1815 - 27 de noviembre de 1852. Londinense. Primera programadora.

[10] BABBAGE, Charles. 26 de diciembre de 1791 - 18 de octubre de 1871. Británico. Matemático: Padre de la Computación

[11] Número de Bernoulli. Denominado por Abraham de Moivre, en honor de Jakob Bernoulli. Sucesión de números racionales basada en la teoría de números.

[12] PRESSMAN, Roger S. *Ingeniería del Software. Un Enfoque Práctico*. Página 1.

[13] Entiéndase el término hacker como experto en computación

[14] RITCHIE, Dennis MacAlistair. 9 de septiembre de 1941. Estadounidense. Científico computacional.

[15] THOMPSON, Kenneth Lane. 4 de febrero de 1943. Estadounidense. Científico computacional.

[16] GATES, William Henry III. 28 de octubre de 1955. Estadounidense. Empresario cofundador de Microsoft.

conocimiento mundial, que se podía construir entre todos para el beneficio de todos; incluso la propiedad del propio Unix era disputada por varias compañías informáticas.

En 1983 comenzó el movimiento de software libre, liderado por Richard Stallman[17], mediante el proyecto GNU[18], cuyo objetivo es buscar la libertad a los usuarios informáticos para reemplazar el software propietario (licencias restrictivas) por el software de licencias libres, es decir el software basado en las cuatro libertades: ejecutarlo, copiarlo, modificarlo y distribuirlo libremente.

La liberación de la propiedad intelectual, Internet, redes P2P (*Peer to Peer*) como Napster para compartir contenidos como música, actividades humanas libres (cine, literatura, música, entre otras), la importancia de la información en la sociedad del conocimiento, el servicio sobre el software como posibilidad económica, la democratización de la economía (ley antimonopolios), la imparcialidad de los estados, la globalización, la calidad del software libre, la gran comunidad de software libre, la educación y el conocimiento libre, la ciudadanía electrónica, las enciclopedias libres, el conocimiento construido de forma colaborativa y cooperativa, Linux, entre muchos otros factores (Grupo de Sistemas y Comunicaciones, 2004), catapultaron al movimiento de software libre como un fenómeno social.

Cerca de los años 80, el término Ingeniería de Software comenzó a ser utilizado para referirse al área de conocimiento de la problemática que ofrecía el software en esa época (HERNÁNDEZ & al., 2007).La informática era inmadura, muy ligada al área electrónica, de tal forma que existió una era de "Crisis del Software" entre 1965 y 1985. Los proyectos de software se caracterizaban por su incumplimiento y podían causar pérdidas económicas. A partir de 1985, aparecieron metodologías de planificación, de aprovechamiento de recursos y de aseguramiento de calidad en el desarrollo del software.

Entre estas herramientas interesantes, se destacan las herramientas CASE[19], cuyo objetivo es el de aumentar la productividad en una de las etapas del ciclo de desarrollo de software más exigente y que requiere de muchos recursos: la codificación de software. Estas herramientas proporcionan ayuda automatizada a muchas actividades del proceso de Ingeniería del Software (VILLAPECELLÍN CID, 2004).

## 2.2. Estado del Arte

Según Fritz Bauer[20], la Ingeniería de Software es "…*el establecimiento y el uso de principios sólidos de la Ingeniería para obtener económicamente un software confiable y que funcione de modo eficiente en máquinas reales*…", pretende deducir que la Ingeniería del Software es un proceso que ha evolucionado históricamente

[17] STALLMAN, Richard Matthew. 16 de marzo de 1953. Estadounidense. Fundador del movimiento por el software libre

[18] GNU. GNU is Not Unix.

[19] CASE: *Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora

[20] NAUR, P. y RANDALL, B. *Software Engineering: A Report on a Conference Sponsored by the NATO Science Comitte*. NATO, 1999

a la par de la historia de la computación, preocupado por la eficiencia y eficacia del producto para donde está destinado y para el aprovechamiento más óptimo del hardware. Sin embargo, este proceso es costoso, consume recursos, especialmente el tiempo y esfuerzo de programación, requiere de minuciosa revisión para propender por su calidad, su producto sufre de un corto periodo de producción (tendencia a la obsolescencia), depende de terceros (especialmente de las plataformas operativas) y es difícil calcular métricas que determinen precisamente su valor.

El ciclo de vida del software (PRESSMAN, 2009) se caracteriza por la consecución de una serie de fases o etapas, como son:

- La toma de requisitos
- El análisis
- El diseño
- La implementación
- Las pruebas
- La implantación
- El mantenimiento

Sobre este ciclo de vida del software pueden aplicar diversos modelos: Ciclo típico, modelo en cascada, modelo en espiral, modelo iterativo, entre otros. Todas las etapas son exigentes, pero la que requiere mayor esfuerzo es la construcción del sistema de información (implementación), porque aquí se genera el código de los componentes del sistema de información, se desarrollan todos los procedimientos de operación y seguridad, se documentan, y se elaboran los manuales de usuario y de explotación. Como se puede apreciar la labor del desarrollador de software (en general, el equipo de desarrollo) es muy exigente y ante el factor tiempo (que siempre está en contra, en cualquiera de estos proyectos), es necesario recurrir a metodologías y a herramientas ágiles de desarrollo de software.

La mayoría de sistemas de información requiere de una base de datos que sirven de bodegaje o repositorio de los datos de la aplicación y además suministran un administrador de estos datos que permiten su recuperación o manipulación. Debido a estas bases de datos, muchos desarrolladores parten de ellas para generar el código de la aplicación. Y si estas aplicaciones se van a ejecutar sobre entornos web, se encuentran tareas básicas comunes a todas las aplicaciones, como es el caso de generar formularios de manipulación de datos (ingreso, consulta, modificación o eliminación), definición de perfiles y niveles de seguridad, formularios de ingreso (*login*) y desconexión de usuarios, trabajo de sesiones, y reportes sencillos. De acuerdo con esto, han surgido herramientas, principalmente software propietario, que generan de forma automática, el código de estas formas, reportes y perfiles, para que pueda ser reusado y complementado con el código restante de la aplicación.

Entre las herramientas más destacadas en la actualidad, se pueden mencionar a:

### 2.2.1. PHPMaker[21]

[21] PHPMaker. http://www.hkvstore.com/phpmaker/. Recuperado 9 de mayo de 2011.

Es un generador de guiones (*scripts*) en PHP que permite manipular datos a partir de una base de datos MySQL, de tal forma que los usuarios puedan ver, editar, buscar, agregar o eliminar registros de la base de datos desde la Web, incluso cuenta con unos asistentes que simplifican la tarea.

Es una herramienta de software propietario muy poderosa y condecorada, porque ofrece características avanzadas de seguridad, un sistema de registro del usuario, exportar datos en formatos muy conocidos, vistas personalizadas, reportes, plantillas personalizadas y sincronización con la base de datos.

### 2.2.2. VisualWade[22]

Es un generador de código gratuito, produce aplicaciones web desde una interface de modelos. Se conecta a base de datos como MySQL, PostgreSQL u Oracle. Solo funciona en plataformas Windows. Sin embargo, funciona a partir de un diagrama de clases, no desde la base de datos.

### 2.2.3. AppGini[23]

Herramienta que acelera el desarrollo de aplicaciones web de bases de datos. Genera una aplicación PHP a partir de una base de datos MySQL. Permite exportar a diferentes formatos. Los scripts generados son personalizados. Es software propietario.

### 2.2.4. PHPScaffold[24]

Es una aplicación on-line, genera código de programación a partir de la información de la base de datos MySQL. Genera formas para introducir, leer, modificar y eliminar (CRUD) registros de las bases de datos. Es necesario ingresar las sentencias SQL.

### 2.2.5. POG: PHP Object Generator[25]

[22] VisualWade. http://www.visualwade.com/. Recuperado 9 de mayo de 2011.
[23] AppGini. http://www.bigprof.com/appgini/. Recuperado 9 de mayo de 2011.
[24] PHPScaffold. http://www.phpscaffold.com/. Recuperado 9 de mayo de 2011.
[25] POG: PHP Object Generator. http://www.phpobjectgenerator.com/. Recuperado 9 de mayo de 2011.

Es un generador de código PHP de código abierto que genera automáticamente código limpio y probado para aplicaciones PHP, contemplando los métodos integrados CRUD. Trabaja con bases de datos MySQL. Necesita de un editor, como Eclipse o ZendIDE. Es necesario interactuar directamente con el código. Las formas no son personalizables, a no ser al nivel de código.

### 2.2.6. PHP Generator for MySQL[26]

Es un *frontend* que permite generar guiones de código PHP de alta calidad para las tablas seleccionadas, vistas y búsquedas, para poder después trabajar con estos objetos sobre la web. Administra los datos (CRUD), HTML personalizable, con capacidades de filtraje, protección de datos, presentaciones maestro detalle, administración de eventos y exportación a formatos comunes. Es una solución propietaria.

La exploración de metodologías, estrategias y herramientas para la generación automática de software no es reciente, de hecho es una promesa incumplida en el área del desarrollo de software (MUÑETÓN, ZAPATA, & ARANGO, 2007). Muñetón, et al., consideran que las herramientas CASE se han convertido en software generador parcial de código, muy alejadas de ofrecer una solución de generación completa de código que permita su ejecución en una plataforma específica. Estos autores proponen un conjunto de reglas de transformación para la obtención automática de código que utilizan como punto de partida a modelos de diagramas de clases, secuencias y máquinas de estados, reglas como las descritas en la Lógica de Predicados de Primer Orden (MORGAN, 1998). Es necesario establecer los elementos de transformación de un lenguaje a otro, procedimiento descrito por Kepple (KEPPLE, WARMER, & BAST, 2003), teniendo en cuenta la relación entre el conjunto fuente y el conjunto destino. Ejemplos de herramientas CASE que siguen este procedimiento son *Together*[27], *Rational Rose*[28] y *Fujaba*[29], que permiten la generación de código a partir de diagramas de clase o diagramas de comportamiento.

Con respecto a *frameworks*, grupos de desarrollo como *sitepoint®*, con su producto *Rails*[30] optando por la arquitectura Modelo-Vista-Controlador, separa los niveles de datos de la aplicación, la interfaz del usuario y la lógica de control, en un nuevo nivel., a partir de una configuración de base de datos. Plataformas IDE, como *NetBeans*[31] , han desarrollado componentes (*plugins*) para generar código CRUD, aprovechando la integración con bases de datos, desde los cuales se pueden crear clases de entidades, para generar pequeñas

[26] PHP Generator for MySQL. http://www.sqlmaestro.com/products/mysql/phpgenerator/. Recuperado 9 de Mayo de 2011

[27] Borland Software Corporation. Together ™. http://www.borland.com/us/products/together/index.aspx. Recuperado Noviembre de 2011
[28] IBM Corporation. Rational Rose Architect™. http://www-01.ibm.com/software/rational/. Recuperado Noviembre 2011
[29] University of Paderborn. Fujaba Tool Suite. http://www.fujaba.de/. Recuperado Noviembre 2011.
[30] Sitepoint. Rails. http://www.sitepoint.com/rails-for-beginners/. Recuperado Noviembre 2011.
[31] Oracle. NetBeans Plataform. http://platform.netbeans.org/tutorials/nbm-crud.html. Recuperado Noviembre 2011

aplicaciones. Una vez que se tiene el código de acceso a la base de datos, se generan las clases de entidad en un módulo, junto con los módulos para la relación JPA JARS[32]. A continuación surge un nuevo módulo que ofrece la interfaz del usuario en la aplicación, que muestra una jerarquía de los datos, y luego otro módulo que permite editar los datos al usuario, donde se puede implementar la funcionalidad de CRUD, en su orden: lectura, actualización, creación y eliminación.

## 2.3. Marco Teórico

### 2.3.1. Ingeniería de Software

La Ingeniería de software fue concebida por entusiastas de la computación que descubrieron un maravilloso mundo de posibilidades, que podría considerarse infinito, que ofrece el hardware para el procesamiento de la información. El avance vertiginoso de la tecnología, se constituyó en otro factor que catapultó al software y lo posicionó en elemento primordial para el procesamiento de la información. A su vez, la sociedad vio el poseer y utilizar de forma inteligente a la información como el recurso más valioso, de tal forma que sucedió una revolución que promulgó los nuevos paradigmas de la Sociedad de la Información y el Conocimiento, con la consecuente necesidad de mejorar y optimizar el procesamiento de la información, coincidente con la consolidación de Internet como elemento importante de comunicación global, el desarrollo de software se perfeccionó como una disciplina: la Ingeniería de Software. Una disciplina preocupada por facilitar metodologías y técnicas para permitir ahorrar tiempo para obtener el software terminado, también para disminuir los altos costos de desarrollo, para minimizar la presencia de errores antes de entregar el software al cliente, para reducir el esfuerzo de programación y para tratar de medir el progreso de desarrollo[33]. La Ingeniería de Software es una tecnología estratificada enfocada en la calidad, el proceso, los métodos y las herramientas del software[34].

Los diferentes modelos del proceso de software, ya sea en cascada, incremental, el desarrollo rápido de aplicaciones, los modelos evolutivos, en espiral, modelo de desarrollo concurrente, el de componentes, métodos formales, desarrollo de software orientado a aspectos, proceso unificado, entre otros, todos sin excepción tienen en cuenta como una fase de su modelo a la construcción, fase neurálgica por cuanto requiere de grandes recursos humanos y materiales, con un gran esfuerzo de sus desarrolladores.

### 2.3.2. Ingeniería Ágil de Software

[32] JPA (Java Persistence API). http://docs.oracle.com/javaee/5/tutorial/doc/bnbpz.html. Recuperado Noviembre 2011
[33] PRESSMAN, Roger S. *Ingeniería del Software. Un Enfoque Práctico*. Página 4
[34] Ídem, página 24

La Ingeniería Ágil de Software combina la filosofía y las directrices de desarrollo, fue propuesta por la "Alianza Ágil". Se busca minimizar el uso de recursos y maximizar la satisfacción del cliente. Contempla equipos motivados, pequeños y con tareas muy simples. La comunicación entre cliente y desarrollador es muy estrecha. Se destacan modelos como la programación extrema (XP), el desarrollo adaptativo de software, método de desarrollo de sistemas dinámicos, melé, cristal, desarrollo conducido por características, modelado ágil, entre otros.

### 2.3.3. El Método SCRUM (SCRUM METHODOLOGY ORG)

Su nombre proviene del deporte Rugby, para indicar a la jugada permite reiniciar el juego después de una falta accidental. Incentiva el trabajo en equipo para el logro de un objetivo en común. Su creador es Jeff Sutherland[35], quien planteó una metodología de desarrollo ágil para cumplir con postulados comunes a las metodologías de desarrollo ágil:

- Individuos e interacciones, sobre procesos y herramientas
- Software operativo, sobre documentación extensiva
- Colaboración con el cliente, sobre negociación de contratos
- Respuesta a los cambios, sobre cumplimiento estricto del plan

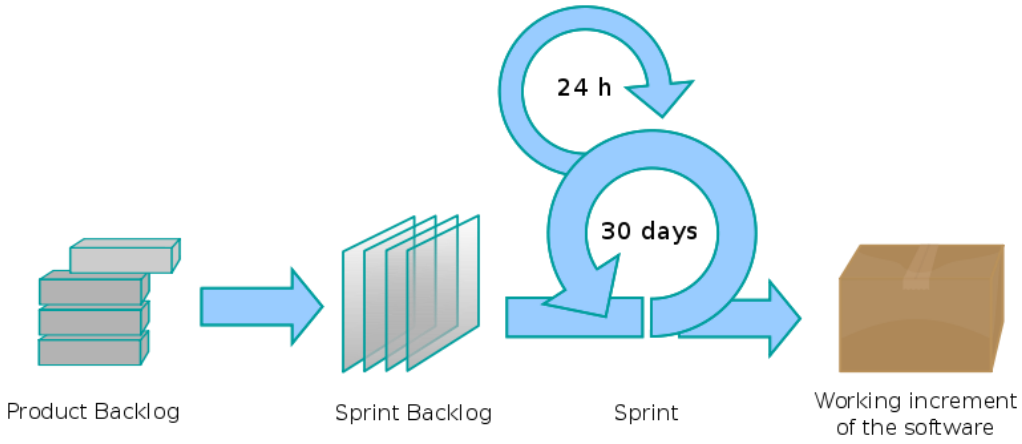



**Ilustración 2. El proceso SCRUM (tomado de http://en.wikipedia.org/wiki/File:Scrum_process.svg)**

La metodología *Scrum* asume como una caja negra al proceso de desarrollo de software, que se puede iniciar en cualquier etapa y puede cambiar de una a otra en cualquier momento. Sus iteraciones se denominan

[35] SUTHERLAND, Jeff. Jeff Sutherland's Blog. http://jeffsutherland.com/. Recuperado 9 de Enero 2012.

*sprints*, que tienen varias semanas de duración. Al principio de cada *sprint* se establece una lista de requerimientos llamada *backlog*. A diario se realizan reuniones breves del equipo de desarrollo, donde se exponen los avances y problemas encontrados y se determinan los caminos para resolverlos. Los interesados en el proyecto se denominan *stakeholders*, el líder del grupo se denomina *Scrum Master*.

Las fases del desarrollo *Scrum* son:

- Revisión de los planes *reléase*
- Distribución
- Revisión y ajuste de estándares de producto
- *Sprint*
- Revisión de *sprint*
- Cierre

### 2.3.4. CASE

Las herramientas CASE (*Computer Aided Software Engineering*) son software que suministra la asistencia a analistas, desarrolladores e Ingenieros de Software en cada fase del ciclo de vida del proceso de software (SOMMERVILLE, 2005). Se trata que estos procesos sean automatizados y que faciliten la colaboración entre los miembros del equipo de trabajo, con el objetivo de producir software con calidad, aumentar la productividad de los equipos de desarrollo, automatizar muchas tareas comunes, generación automática de la documentación de la aplicación y generar algunas estructuras de código[36].

Los generadores de código deben reunir algunas de estas características:

- Lenguaje generado puede ser estándar o propietario
- Portabilidad del lenguaje generado
- Generación parcial o total del código de la aplicación
- Posibilidad de modificar el código generado
- Generación de código de tareas simples y rutinarias dentro de la aplicación
- Generación de la interface del usuario

Las Herramientas CASE surgieron en la época denominada “Crisis del Software”, denominada así porque la industria del software no alcanzaba a cubrir la demanda con las metodologías que se estaban usando, fue así que se pensó en metodologías para crear estándares de desarrollo. Las herramientas CASE hacen parte de las herramientas de cuarta generación (4G), que busca especificar el software a un nivel muy próximo al lenguaje natural, de tal forma que se pueda generar automáticamente el código fuente con base en las especificaciones técnicas de la aplicación. Dentro de este amplio abanico se tienen a lenguajes no procedimentales para consulta a base de datos, generación de informes, manipulación de datos, interacción y definición de pantallas y generación de códigos, capacidades gráficas de alto nivel y capacidad de hojas de cálculo.

[36] RUIZ LUQUE, José Carlos, et al. Herramientas CASE. Universidad de las Palmas de Gran Canarias.

La mayoría de soluciones de herramientas CASE son propietarias, como es el caso de *CA ERwin Data Modeler*[37], *Oracle Designer*[38], *PowerDesigner*[39], *Rational System Architect*[40], *Visual Paradigm*[41], entre otros. Dentro de las soluciones para software libre se encuentra a *ArgoUML*[42], *Fujaba*[43], *monoUML*[44], *starUML*[45], *UMLet*[46], *Omondo*[47], entre otros, que se pueden apreciar, están orientados a UML. JUDE[48] era una solución de software libre, pero ahora es propietaria con el nombre *astah*.

La tecnología CASE supone "la informatización de la informática" (PIATTINI, CALVO-MANZANO, CERVERA, & FERNÁNDEZ, 2004), un enfoque que busca calidad y productividad en el desarrollo de la Ingeniería de Sistemas. Los componentes de una herramienta CASE son el repositorio, el metamodelo, el generador de informes, carga/descarga de datos, comprobación de errores y la interfaz de usuario.

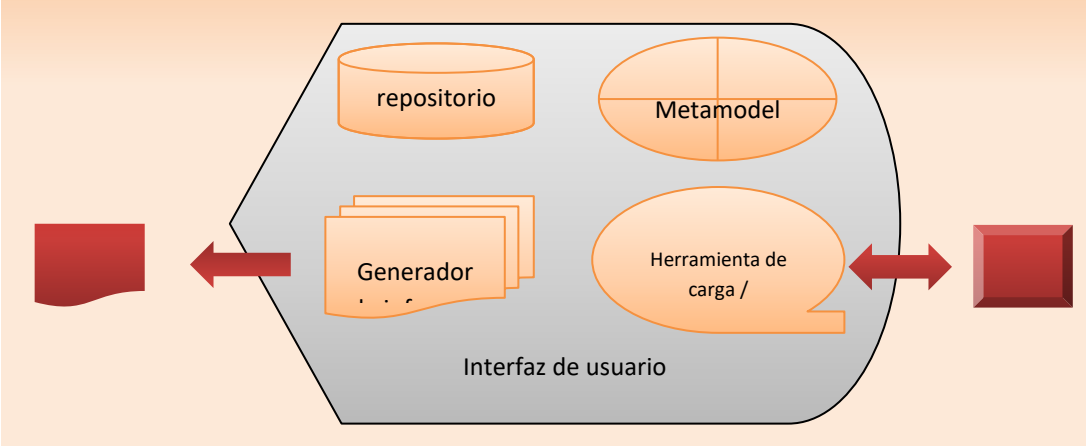



**Ilustración 3. Componentes de una herramienta CASE (Tomado de (PIATTINI, CALVO-MANZANO, CERVERA, & FERNÁNDEZ, 2004))**

[37] CA Erwin Data Modeler. http://erwin.com/. Recuperado 26 de mayo de 2011

[38] Oracle. Designer. http://www.oracle.com/technetwork/developer-tools/designer/overview/index.html. Recuperado 26 de mayo de 2011
[39] Sybase. Power Designer. http://www.sybase.com/products/modelingdevelopment/powerdesigner. Recuperado 26 de mayo de 2011
[40] IBM. Rational System Architect. http://www-01.ibm.com/software/awdtools/systemarchitect/#. Recuperado 26 de mayo de 2011
[41] Visual Paradigm. http://www.visual-paradigm.com/. Recuperado 26 de mayo de 2011
[42] Tigris. ArgoUML. http://argouml.tigris.org/. Recuperado 26 de mayo de 2011
[43] Fujaba. Fujaba Tool Suite. http://www.fujaba.de/. Recuperado 26 de mayo de 2011
[44] monoUML. monoUML CASE tool. http://www.monouml.org/. Recuperado 26 de mayo de 2011
[45] starUML. http://staruml.sourceforge.net/en/. Recuperado 26 de mayo de 2011
[46] UMLet. http://www.umlet.com/. Recuperado 26 de mayo de 2011
[47] OMG Omondo. Omondo. http://www.ejb3.org/. Recuperado 26 de mayo de 2011
[48] Change Vision. Astah. http://jude.change-vision.com/jude-web/index.html. Recuperado 26 de mayo de 2011

Se pueden considerar tres categorías (PIATTINI, CALVO-MANZANO, CERVERA, & FERNÁNDEZ, 2004) de herramientas CASE:

- Herramientas de gestión. Permiten estimar, planificar y hacer seguimiento al proyecto
- Herramientas técnicas. Que se subdividen en:
    - CASE Frontales o superiores (*Front-end o Upper CASE*). Abarca la fase de análisis y diseño
    - CASE Dorsales o inferiores (*Back-end o Lower CASE*). Diseño detallado y generación de código
- Herramientas de soporte. Sistema de repositorio o diccionario, control y configuración, seguridad

Las herramientas ICASE (*Integrated Case*) integran a los CASE frontales y dorsales, e IPSE (*Integrated Programming Support Environment*), aquellas herramientas CASE que además incluyen componentes para la gestión de proyectos y la gestión de configuración.

La adopción de CASE se gestiona como un proyecto con un plan detallado de implantación (PIATTINI, CALVO-MANZANO, CERVERA, & FERNÁNDEZ, 2004):

a. **Planificación**: Un equipo de trabajo que establezca los estándares y procedimientos del proyecto, el calendario y el personal que participa, al igual que los sistemas que se desarrollarán. Es necesario valorar económicamente el proyecto e incluir un plan de riesgos
b. **Adquisición**: Infraestructura (hardware, software y personal), proceso de desarrollo (ciclo de vida, técnicas y metodologías, y productos resultantes) y herramientas
c. **Introducción**: Organización del personal, instalación y adaptación de la herramienta, formación, proyecto piloto y evaluación
d. **Utilización**: Formación de personal, instalación de herramientas, migración de aplicaciones
e. **Retirada**: Cuando la tecnología que se ha implantado queda obsoleta

Algunos de los indicadores de costo en la adopción de herramientas CASE son:

- Alcance CASE, complejidad y riesgo
- Complejidad del entorno
- Organización
- Cultura de cambio
- Tecnología actual
- Práctica actual
- Nivel actual
- Capacidad del personal de desarrollo
- Velocidad del avance tecnológico

### 2.3.5. Software Libre

"El software libre es una cuestión de la libertad de los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. Más precisamente, significa que los usuarios de programas tienen las cuatro libertades esenciales.

- La libertad de ejecutar el programa, para cualquier propósito (libertad 0).
- La libertad de estudiar cómo trabaja el programa, y cambiarlo para que haga lo que usted quiera (libertad 1). El acceso al código fuente es una condición necesaria para ello.
- La libertad de redistribuir copias para que pueda ayudar al prójimo (libertad 2).
- La libertad de distribuir copias de sus versiones modificadas a terceros (la 3ª libertad). Si lo hace, puede dar a toda la comunidad una oportunidad de beneficiarse de sus cambios. El acceso al código fuente es una condición necesaria para ello."[49]

El código abierto ofrece (JORBA ESTEVE):

- Acceso y/o modificación del código fuente
- Gratuidad
- No depender de una única opción o único fabricante de nuestro software
- Compartir del conocimiento para lograr progresos de forma más rápida, con mejor calidad, ya que las elecciones tomadas están basadas en el consenso de la comunidad.

Existen dos comunidades que promueven el movimiento de software libre: La Fundación de Software Libre (FSF) liderada por *Richard Stallman* y el código abierto (*open source*). La diferencia entre ambos movimientos radica en postulados filosóficos.

El software libre enfatiza los aspectos morales y éticos para producir software con excelencia técnica, gracias al poder compartir el código fuente para lograr este fin. La mayoría de software libre se produce por equipos a nivel mundial mediante esquemas de fuerte cooperación. Su trabajo es expuesto al escrutinio público, de tal forma que pueden depurarse errores que un principio no es detectado.

### 2.3.6. Sistemas de Gestión de Bases de Datos

Un Sistema de Gestión de Base de Datos (DBMS) es software que suministra una interfaz entre la base de datos, el usuario y la aplicación, para administrar de forma sencilla y ordenada el conjunto de datos.

Una base de datos es un conjunto de datos almacenados para su posterior uso. Existen modelos de bases de datos como: Bases de datos jerárquicas, en red, transaccionales, relacionales, multidimensionales, orientadas a objetos, documentales, distribuidas y deductivas. De los cuales el modelo más popular ha sido a través de

[49] GNU Operating System. La Definición de Software Libre. http://www.gnu.org/philosophy/free-sw.es.html. Recuperado 26 de mayo de 2011

la historia, el modelo relacional, cuyo lenguaje más común para construir consultas a las bases de datos es SQL (*Structured Query Language*), que está implementado por la mayoría de gestores de bases de datos.

EL SQL es un lenguaje declarativo de alto nivel orientado al manejo de conjunto de registros, suministrando alta productividad en la codificación y la orientación a objetos. La gestión de la base de datos proporciona dos tipos de lenguajes: el lenguaje de definición de datos (DDL) que se encarga de la modificación de la estructura de los objetos de la base de datos y el Lenguaje de Manipulación de Datos (DML) que permite ejecutar tareas de consulta o manipulación de datos.

### 2.3.7. MySQL

Es un sistema de gestión de bases de datos relacional, multihilo y multiusuario muy popular. Desde 2009 es desarrollado como software libre en un esquema de licenciamiento dual. MySQL comúnmente es muy utilizado en aplicaciones web, en plataformas con sistemas operativos Linux/Windows, servidor web Apache y lenguajes como PHP/Perl/Python; y además por herramientas de seguimiento de errores. MySQL es muy rápido con el motor MyISAM, pero no es muy fuerte con este motor en situaciones concurrentes.

MySQL en su versión 5 o posterior, basa su administrador del diccionario de datos en las tablas *INFORMATION_SCHEMA* (Oracle Corp.), cuya función es proporcionar acceso a los metadatos de la base de datos., por ejemplo, el nombre de la base de datos, nombres de las tablas, el tipo de datos de los campos, permisos de acceso, entre otros, este conjunto de información se conoce como el diccionario de datos o catálogo del sistema. Este diccionario de datos sigue el estándar ANSI/ISO SQL:2003 Parte 11 (*schemata*), característica básica F021 (*Basic information schema*).

### 2.3.8. PHP

PHP (*Hypertext Preprocessor*) (PHP Group) es un lenguaje de *scripting* de propósito general de amplio uso que especialmente está enfocado en el desarrollo de aplicaciones Web y puede ser incrustado en páginas HTML. El código es ejecutado en el lado del servidor, generando HTML y enviándolo al cliente. El cliente recibirá los resultados y ejecutará el script, sin posibilidad de determinar qué código ha producido el resultado recibido. Es un lenguaje muy simple para el principiante, pero a su vez, ofrece características avanzadas para desarrolladores profesionales. PHP puede ser utilizado en la mayoría de sistemas operativos, es soportado por la mayoría de servidores web. Tiene la posibilidad de utilizar programación por procedimientos, programación orientada a objetos, o su mezcla. Soporta gran cantidad de bases de datos y puede comunicarse con otros servicios usando protocolos como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM, entre otros. Además puede utilizar objetos de JAVA de forma transparente.

### 2.3.9. CRUD

Es el acrónimo de Crear, Obtener (*read o retrieve*), actualizar (*update*) y borrar (*delete*), usado para referirse a las funciones básicas en bases de datos o a la capa de persistencia del software. También se utiliza para describir las convenciones de la interface del usuario que facilitan la vista, búsqueda y el cambio de la información, típicamente utilizado en formas y reportes basados en el computador. El término fue popularizado por *James Martin*[50] en 1983, en un artículo titulado "*From semantic to object-oriented data modeling*".

CRUD puede ser implementado con una base de datos de objetos, una base de datos XML, archivos de texto plano, formatos personalizables de archivo, cintas o tarjetas.

El ciclo CRUD describe las funciones más elementales de una base de datos persistente. Estas funciones también están descritas en el ciclo de vida de datos[51]. Diferentes usuarios pueden tener diferentes ciclos CRUD basados en los requerimientos del sistema[52]. El ciclo CRUD es el corazón de la mayoría de sitios web dinámicos, especialmente aquellos creados en la era del Web 2.0.

### 2.3.10. Estándares de Calidad Aplicados al Software

**ISO 9000-3:1993[53]**

Enfocada a compañías desarrolladoras de software. Permite incursionar en la competencia, busca satisfacer las expectativas del cliente, busca calidad y ventajas competitivas, estrategia de mercado y de reducción de costos de producción

[50] MARTIN, James. *Managing the Data-base Environment*. Prentice-Hall. 1983

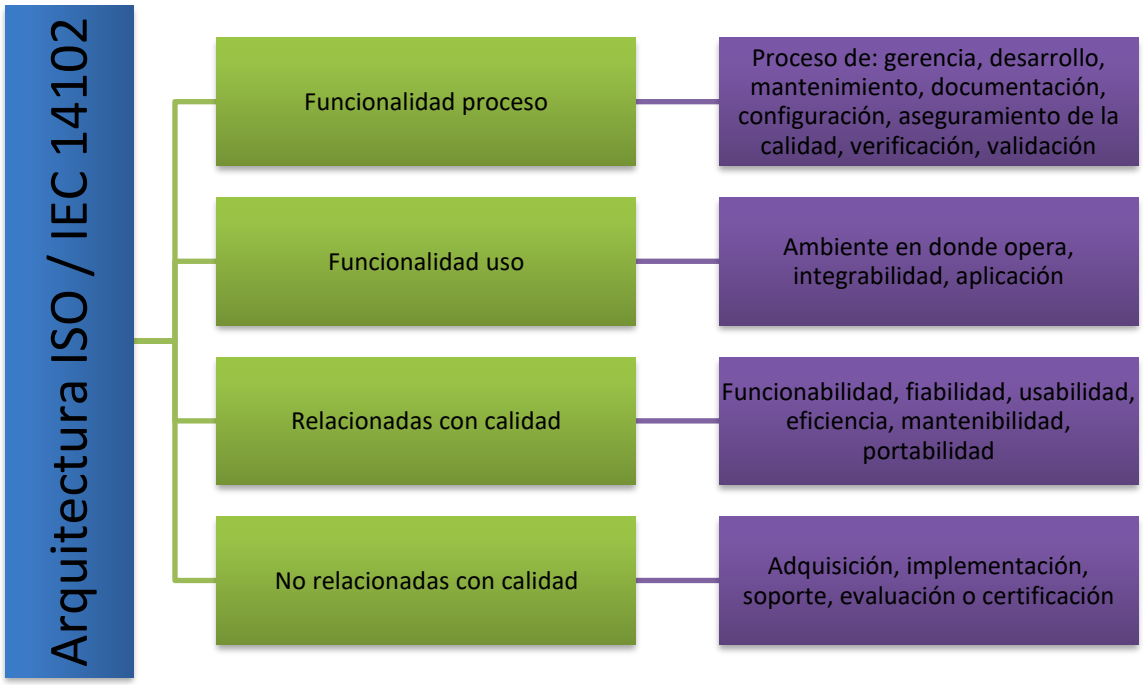
[51] DLM. Ciclo de Vida de Datos. Es una ventaja basada en políticas para administrar el flujo de los datos de un sistema de información a través de su ciclo de vida: desde la creación y el almacenamiento inicial al momento cuando comienza a ser obsoleto y es borrado.
[52] TechTarget. SearchDataManagment. http://searchdatamanagement.techtarget.com/definition/CRUD-cycle. Recuperado Noviembre 2011
[53] ISO. ISO 9000-3. http://www.docquality.info/es_f-33~d-8182~n-calidad+iso+9000+ISO+9000+3.PDF~. Recuperado Noviembre 2011

**ISO 12207:1995**[54]

Actividades en el ciclo de vida de software: Especificaciones, diseño, programación, integración y prueba. También incluye a la documentación, administración, aseguramiento de la calidad, verificación, evaluación, revisión conjunta.

**ISO/IEC 14102:1995**[55]

Guía para la evaluación y selección de herramientas CASE




[54] ISO. ISO 12207. http://www.12207.com/. Recuperado Noviembre 2011
[55] ISO. ISO 14102. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43189. Recuperado Noviembre 2011

Ilustración 4. Arquitectura ISO / IEC 14102

## 2.4. Instrumentos de Recolección de Información

La formulación de la hipótesis de este proyecto requiere la validación de variables para corroborar o negar la hipótesis planteada, entre estas variables cabe mencionar:

- V1: Rapidez de Desarrollo
- V2: Calidad de Desarrollo de Código
- V3: Costo de Desarrollo de Código
- V4: Usabilidad de Herramientas de Generación Automática de Código
- V5: Nivel de compatibilidad con entornos de desarrollo

Para obtener los datos que permitan ser procesados para validar estas variables, puede hacerse uso de alguno de estos instrumentos de recolección de datos, que pueden utilizar técnicas para la investigación como la observación y recolección de datos de fuentes confiables (primarias y secundarias).

### 2.4.1. Primer Instrumento: Tabla para la Identificación de Elementos de Calidad de Desarrollo de Código

**Descripción del instrumento**: Este instrumento se constituye como un registro que busca documentar los diferentes elementos que se deben tener en cuenta para determinar la calidad de desarrollo de código, que recolecta el indicador, su descripción, el elemento primario de calidad, la cantidad de componentes externos, entre otros elementos.

**Justificación**: Este instrumento se requiere para iniciar la identificación y selección de los elementos de mayor influencia en la determinación de la calidad del proceso de desarrollo de código de software así como la identificación de los métodos que demuestran mayor eficiencia desde diversos ángulos evaluativos.

### 2.4.2. Segundo Instrumento: Tabla Identificación de la Velocidad de Desarrollo de Código Con y Sin Herramientas de Generación Automática de Software

**Descripción del instrumento**: Este instrumento se constituye como un registro para comparar el grado de rapidez de desarrollo al usar o no una herramienta de generación automática de código, junto con los elementos que los califican, para definir de forma coherente un punto de partida en la investigación.

**Justificación**: Este instrumento se requiere puesto es necesario identificar cuál es el aporte en la velocidad de desarrollo de código de software al utilizar una herramienta de generación automática de código. Contando con estos datos será posible iniciar nuevas propuestas o mejoras a los elementos de desarrollo de software.

## 2.5. Población Objetivo

La indagación por la hipótesis del proyecto se centra en confrontar las variables con dos tipos de personas: Un grupo de desarrolladores, preferiblemente de la región, que hayan o estén trabajando en algún proyecto de software libre sin importar su magnitud, buscando conocer su experiencia en el desarrollo rápido de sistemas de información a partir de una base de datos, así como también sus expectativas de disponer de herramientas que faciliten su labor. Un segundo grupo de personas conformado por entusiastas de la ingeniería del software en entornos del software libre, con el propósito de conocer sus opiniones acerca del desarrollo automático de aplicaciones para facilitar la labor del desarrollador de software.

## 2.6. Formas de Recolectar la Información

La encuesta y la observación directa van a ser los instrumentos primordiales que van a proveer al investigador de los datos necesarios para verificar las variables de la hipótesis propuesta. Dependiendo de la muestra de la población objetivo, si existe contacto directo con ella, los dos instrumentos pueden ser aplicados; pero en caso de que la muestra de la población se encuentre remotamente al sitio de la investigación se aplicará la encuesta mediante medios electrónicos de comunicación. Sin embargo, el entorno geográfico de la investigación se centrará en la región de la investigación.

## 3. METODOLOGÍA

La metodología de la investigación es deductiva, donde se aceptan datos generales como válidos, para llegar a una conclusión de tipo particular, de esta forma se puede cumplir cada uno de los objetivos propuestos mediante una metodología de investigación coherente para el éxito de la investigación. El interés de la investigación es probar que si el uso de una herramienta de generación automática de software de tareas básicas reducirá el tiempo de desarrollo de aplicaciones libres en entornos Web que requieren de una base de datos.

La generación automática de código sugiere el uso de una metodología CASE, que para este caso, el plan detallado de implantación de herramientas CASE (PIATTINI, CALVO-MANZANO, CERVERA, & FERNÁNDEZ, 2004) proporciona métodos adecuados para el desarrollo y uso de este tipo de herramientas.

Pero a su vez, es necesario el desarrollo de software, por la naturaleza de software libre y por la necesidad del desarrollo rápido, la metodología Agile Scrum, proporciona los elementos necesarios para el desarrollo de esta aplicación.

El uso de estas dos últimas metodologías es requerido porque la investigación exige probar una herramienta con la muestra de la población, que en la actualidad no existe cumpliendo con la totalidad de esas características (que genere código de forma automática, que cree interfaces CRUD, que su punto de partida sea una base de datos ya construida, y que sea en ambientes de software libre).

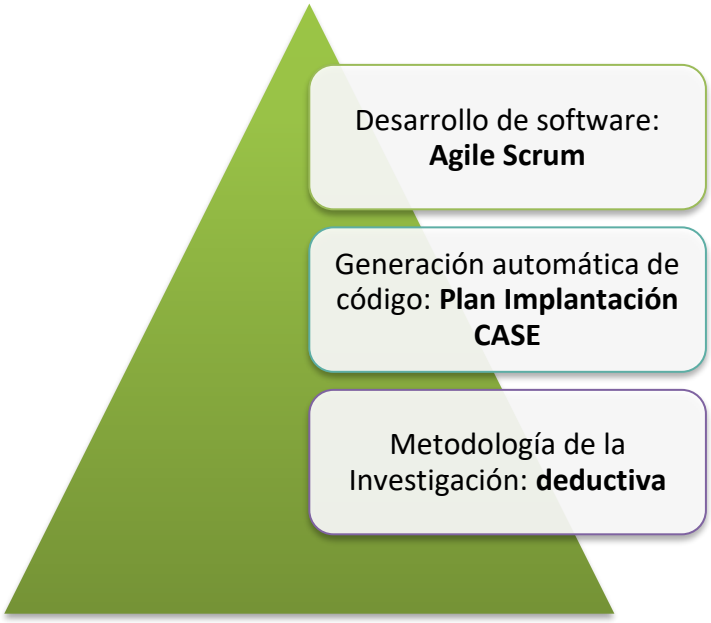



**Ilustración 5. Bosquejo metodológico**

Se empezará desarrollando las etapas de la investigación cuantitativa que empieza con la formulación del problema de investigación, basándose en preguntas referentes a la trascendencia de esta investigación y su

pertinencia, de acuerdo con Oppenheimer[56] "*Investigar significa pagar la entrada por adelantado y entrar sin saber lo que se va a ver*". De lo anterior se dedujo el problema de una necesidad de tipo disciplinar, que permitió la definición del título y sus objetivos.

En la fase de exploración se extrae la revisión de la literatura y se construye el marco teórico de la investigación, se busca un esquema de software libre para apoyar a la misma comunidad, como algunas soluciones propietarias lo hacen y el por qué es una de las estrategias de implantación a seguir, teniendo en cuenta fundamentación teórico - práctica.

Dentro del diseño de la investigación se obtiene el tipo de estudio que se realizará, en este caso descriptivo, analiza cómo es y cómo se manifiesta un fenómeno y sus componentes. Permite detallar el fenómeno estudiado básicamente a través de la medición de uno o más de sus atributos. Se escogió el diseño apropiado de la investigación, en el diseño experimental donde se construye una realidad.

A continuación se procede con la extracción de la muestras, teniendo en cuenta que sea representativa o que se refleje el conjunto o universo que se va a estudiar, reproduciendo de la manera más exacta posible las características de éste. Su tamaño sea estadísticamente proporcional al tamaño de la población. El error muestral se mantenga dentro de límites aceptables. Se utilizará muestras probabilísticas donde los resultados son generalizables a la población.

Para la recopilación de datos se debe efectuar los tres procedimientos o técnicas más usadas en este proceso, la observación, la entrevista y encuestas. En la observación se utilizará en cada empresa en la observación de los procesos habituales que interviene en la investigación. Se harán entrevistas a equipos de desarrollo de software libre, técnicos y usuarios finales para tener la información de primera mano sobre los problemas principales. Las entrevistas se aplicarán a los usuarios finales.

Se efectuará el análisis de la información de la investigación, donde primero se escriben los datos estadísticos recopilados (representación escrita), tabulación de la información ordenada con los datos numéricos en filas y columnas, con las especificaciones correspondientes acerca de su naturaleza, es necesaria la representación gráfica de los datos. Se diseñará el diseño del informe final, donde se mostrarán las conclusiones de la investigación, obtenidas en un análisis detallado de los datos.

De otra parte, para el desarrollo de la herramienta informática se propone metodologías de desarrollo rápido, como es el caso de *Agile Scrum*, que permiten el desarrollo "ágil" del proyecto de software. Se basan en los principios de la simplicidad, la comunicación, la retroalimentación y el coraje para implicar a todo el equipo (y a los usuarios o clientes) en la gestión del proyecto (HERNÁNDEZ & al., 2007).

La lista de actividades se determina con el propósito de cumplir los objetivos:

[56] OPPENHEIMER, Julius Robert. 22 de abril de 1904 – 18 de febrero de 1967. Estadounidense. Físico. Padre de la Bomba Atómica

| Objetivo | Actividades |
|---|---|
| Identificar los requerimientos que debe reunir una herramienta de generación automática de código considerados por los desarrolladores de software libre de la región | Documentar los diferentes elementos que se deben tener en cuenta para determinar la calidad de desarrollo de código |
| | Extracción de muestras de la población objetivo |
| | Aplicar los instrumentos de recolección de información a la población objetivo |
| | Análisis de la información de la investigación: representación escrita, tabulación y diseño del informe final |
| | Comparar el grado de rapidez de desarrollo, al usar o no una herramienta de generación automática de código |

**Tabla 1. Principales actividades del objetivo 1**

| Objetivo | Actividades |
|---|---|
| Realizar un diagnóstico comparativo del uso de herramientas actuales, de software libre o propietario, para la generación automática de código, a partir de una base de datos | Generar un listado de herramientas de generación de código en bases de datos en línea, repositorios de software y en sitios de proyectos de software. |
| | Revisión de la literatura y construcción del marco teórico de la investigación |
| | Generar un diagnóstico comparativo sobre las herramientas seleccionadas |

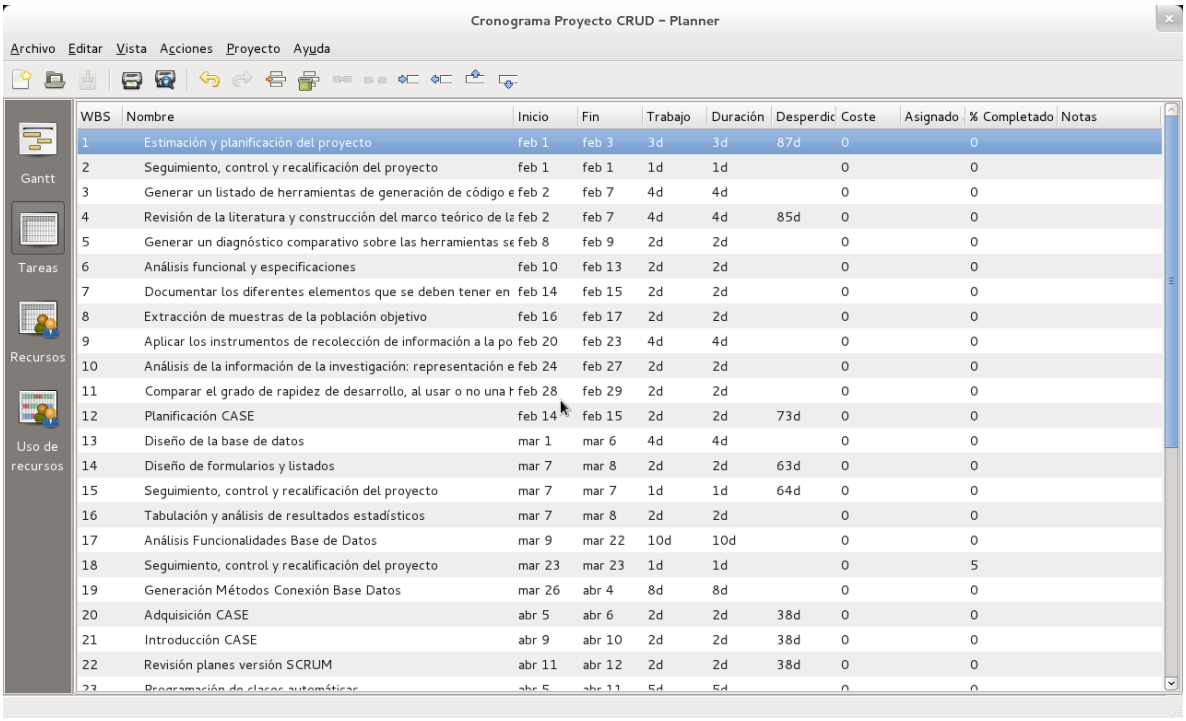
**Tabla 2. Principales actividades objetivo 2**

| Objetivo | Actividades |
|---|---|
| Crear una herramienta de automatización para la generación de código de interfaces CRUD, a partir de una base de datos, en ambientes de desarrollo de software libre | Etapa planificación herramienta CASE |
| | Etapa adquisición herramienta CASE |
| | Etapa introducción herramienta CASE |
| | Etapa utilización herramienta CASE |
| | Etapa formulación retirada herramienta CASE |
| | Fase revisión de planes de versión (SCRUM) |
| | Fase Distribución, revisión y ajuste de estándares de producto (SCRUM) |

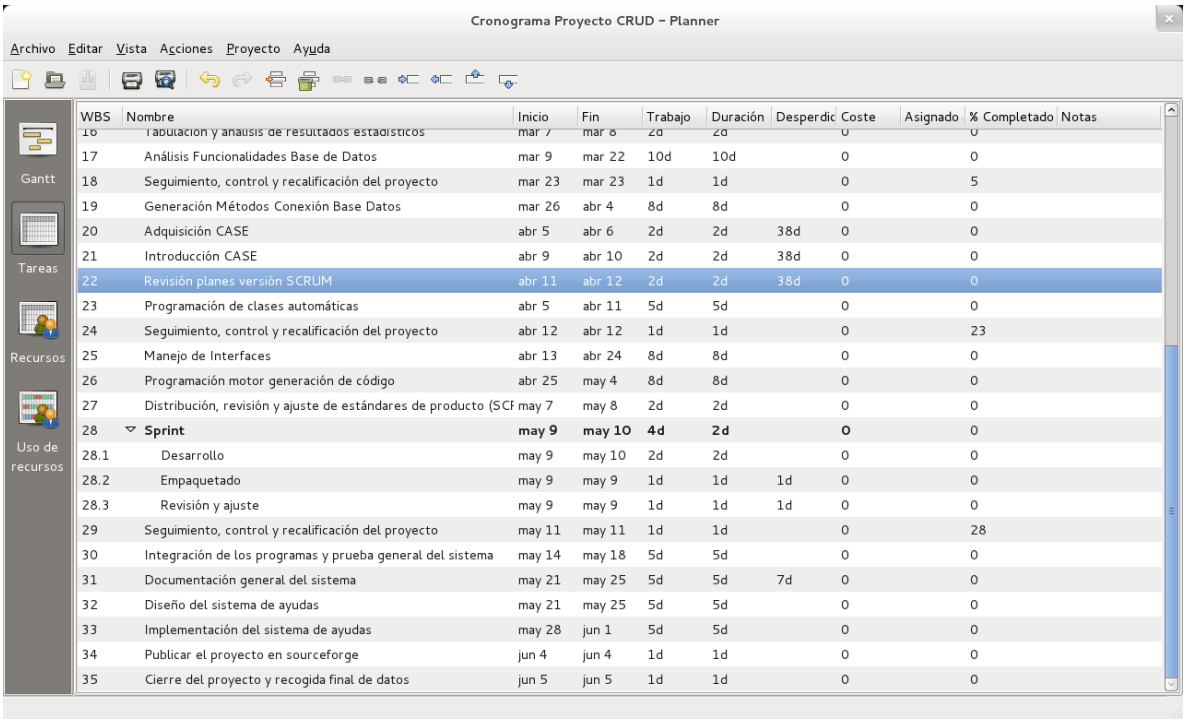
| | |
|---|---|
| | Fase Sprint (SCRUM: Desarrollo, empaquetado, revisión y ajuste) |
| | Fase Revisión Sprint SCRUM |
| | Fase Análisis de variables SCRUM |
| | Fase Cierre SCRUM |

**Tabla 3. Principales Actividades Objetivo 3**

La secuencia y duración de estas actividades está proyectada así:



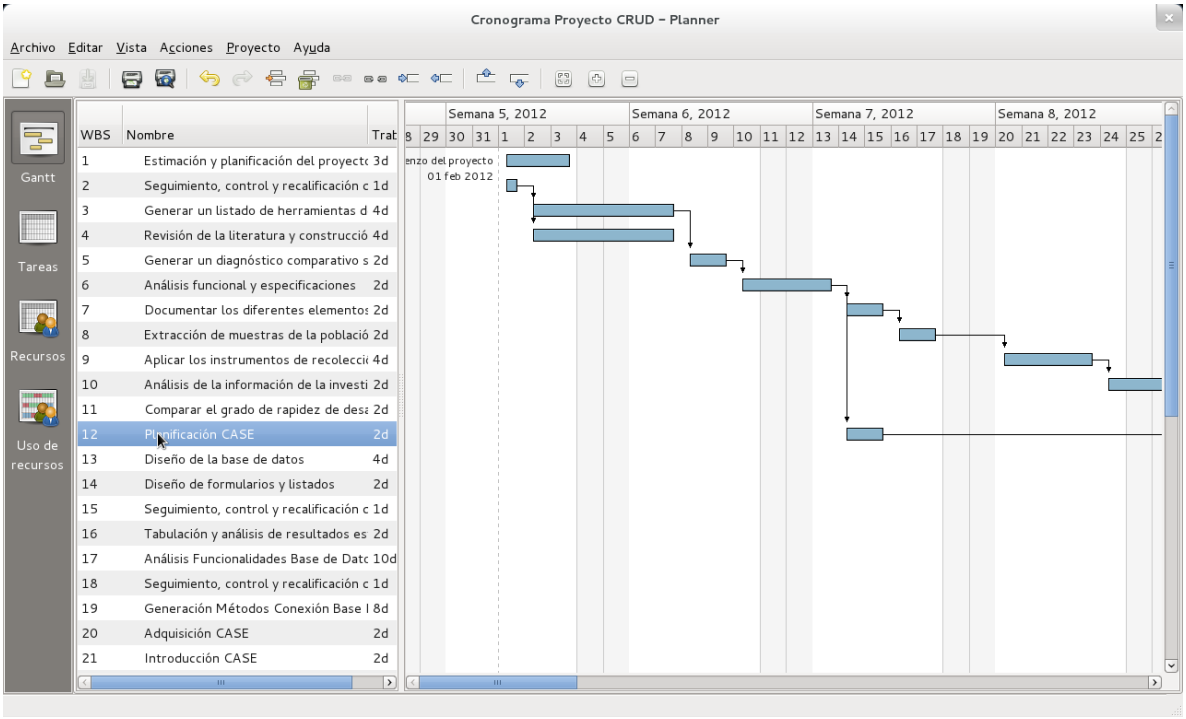

| WBS | Nombre | Inicio | Fin | Trabajo | Duración | Desperdic | Coste | Asignado | % Completado | Notas |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Estimación y planificación del proyecto | feb 1 | feb 3 | 3d | 3d | 87d | 0 | | 0 | |
| 2 | Seguimiento, control y recalificación del proyecto | feb 1 | feb 1 | 1d | 1d | | 0 | | 0 | |
| 3 | Generar un listado de herramientas de generación de código e | feb 2 | feb 7 | 4d | 4d | | 0 | | 0 | |
| 4 | Revisión de la literatura y construcción del marco teórico de la | feb 2 | feb 7 | 4d | 4d | 85d | 0 | | 0 | |
| 5 | Generar un diagnóstico comparativo sobre las herramientas se | feb 8 | feb 9 | 2d | 2d | | 0 | | 0 | |
| 6 | Análisis funcional y especificaciones | feb 10 | feb 13 | 2d | 2d | | 0 | | 0 | |
| 7 | Documentar los diferentes elementos que se deben tener en | feb 14 | feb 15 | 2d | 2d | | 0 | | 0 | |
| 8 | Extracción de muestras de la población objetivo | feb 16 | feb 17 | 2d | 2d | | 0 | | 0 | |
| 9 | Aplicar los instrumentos de recolección de información a la po | feb 20 | feb 23 | 4d | 4d | | 0 | | 0 | |
| 10 | Análisis de la información de la investigación: representación e | feb 24 | feb 27 | 2d | 2d | | 0 | | 0 | |
| 11 | Comparar el grado de rapidez de desarrollo, al usar o no una h | feb 28 | feb 29 | 2d | 2d | | 0 | | 0 | |
| 12 | Planificación CASE | feb 14 | feb 15 | 2d | 2d | 73d | 0 | | 0 | |
| 13 | Diseño de la base de datos | mar 1 | mar 6 | 4d | 4d | | 0 | | 0 | |
| 14 | Diseño de formularios y listados | mar 7 | mar 8 | 2d | 2d | 63d | 0 | | 0 | |
| 15 | Seguimiento, control y recalificación del proyecto | mar 7 | mar 7 | 1d | 1d | 64d | 0 | | 0 | |
| 16 | Tabulación y análisis de resultados estadísticos | mar 7 | mar 8 | 2d | 2d | | 0 | | 0 | |
| 17 | Análisis Funcionalidades Base de Datos | mar 9 | mar 22 | 10d | 10d | | 0 | | 0 | |
| 18 | Seguimiento, control y recalificación del proyecto | mar 23 | mar 23 | 1d | 1d | | 0 | | 5 | |
| 19 | Generación Métodos Conexión Base Datos | mar 26 | abr 4 | 8d | 8d | | 0 | | 0 | |
| 20 | Adquisición CASE | abr 5 | abr 6 | 2d | 2d | 38d | 0 | | 0 | |
| 21 | Introducción CASE | abr 9 | abr 10 | 2d | 2d | 38d | 0 | | 0 | |
| 22 | Revisión planes versión SCRUM | abr 11 | abr 12 | 2d | 2d | 38d | 0 | | 0 | |
| 23 | Programación de clases automáticas | abr 5 | abr 11 | 5d | 5d | | 0 | | 0 | |



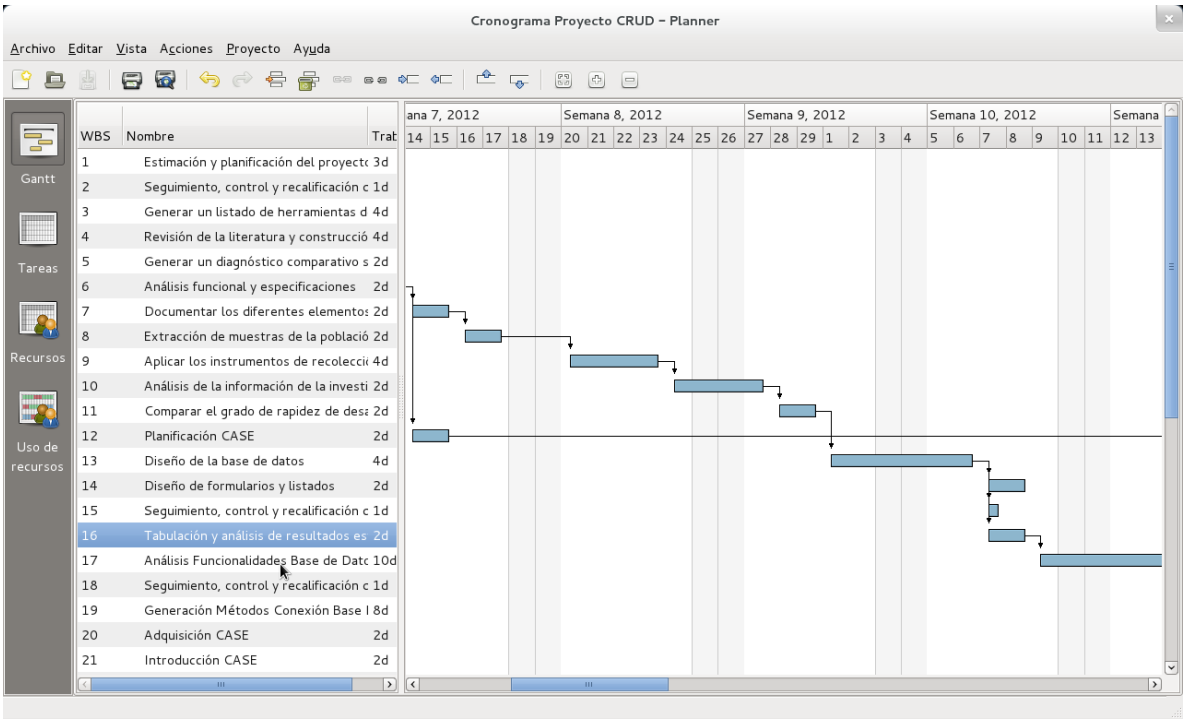

| WBS | Nombre | Inicio | Fin | Trabajo | Duración | Desperdic | Coste | Asignado | % Completado | Notas |
|---|---|---|---|---|---|---|---|---|---|---|
| 16 | Tabulación y análisis de resultados estadísticos | mar 7 | mar 8 | 2d | 2d | | 0 | | 0 | |
| 17 | Análisis Funcionalidades Base de Datos | mar 9 | mar 22 | 10d | 10d | | 0 | | 0 | |
| 18 | Seguimiento, control y recalificación del proyecto | mar 23 | mar 23 | 1d | 1d | | 0 | | 5 | |
| 19 | Generación Métodos Conexión Base Datos | mar 26 | abr 4 | 8d | 8d | | 0 | | 0 | |
| 20 | Adquisición CASE | abr 5 | abr 6 | 2d | 2d | 38d | 0 | | 0 | |
| 21 | Introducción CASE | abr 9 | abr 10 | 2d | 2d | 38d | 0 | | 0 | |
| 22 | Revisión planes versión SCRUM | abr 11 | abr 12 | 2d | 2d | 38d | 0 | | 0 | |
| 23 | Programación de clases automáticas | abr 5 | abr 11 | 5d | 5d | | 0 | | 0 | |
| 24 | Seguimiento, control y recalificación del proyecto | abr 12 | abr 12 | 1d | 1d | | 0 | | 23 | |
| 25 | Manejo de Interfaces | abr 13 | abr 24 | 8d | 8d | | 0 | | 0 | |
| 26 | Programación motor generación de código | abr 25 | may 4 | 8d | 8d | | 0 | | 0 | |
| 27 | Distribución, revisión y ajuste de estándares de producto (SCF | may 7 | may 8 | 2d | 2d | | 0 | | 0 | |
| 28 | **Sprint** | **may 9** | **may 10** | **4d** | **2d** | | **0** | | 0 | |
| 28.1 | Desarrollo | may 9 | may 10 | 2d | 2d | | 0 | | 0 | |
| 28.2 | Empaquetado | may 9 | may 9 | 1d | 1d | 1d | 0 | | 0 | |
| 28.3 | Revisión y ajuste | may 9 | may 9 | 1d | 1d | 1d | 0 | | 0 | |
| 29 | Seguimiento, control y recalificación del proyecto | may 11 | may 11 | 1d | 1d | | 0 | | 28 | |
| 30 | Integración de los programas y prueba general del sistema | may 14 | may 18 | 5d | 5d | | 0 | | 0 | |
| 31 | Documentación general del sistema | may 21 | may 25 | 5d | 5d | 7d | 0 | | 0 | |
| 32 | Diseño del sistema de ayudas | may 21 | may 25 | 5d | 5d | | 0 | | 0 | |
| 33 | Implementación del sistema de ayudas | may 28 | jun 1 | 5d | 5d | | 0 | | 0 | |
| 34 | Publicar el proyecto en sourceforge | jun 4 | jun 4 | 1d | 1d | | 0 | | 0 | |
| 35 | Cierre del proyecto y recogida final de datos | jun 5 | jun 5 | 1d | 1d | | 0 | | 0 | |

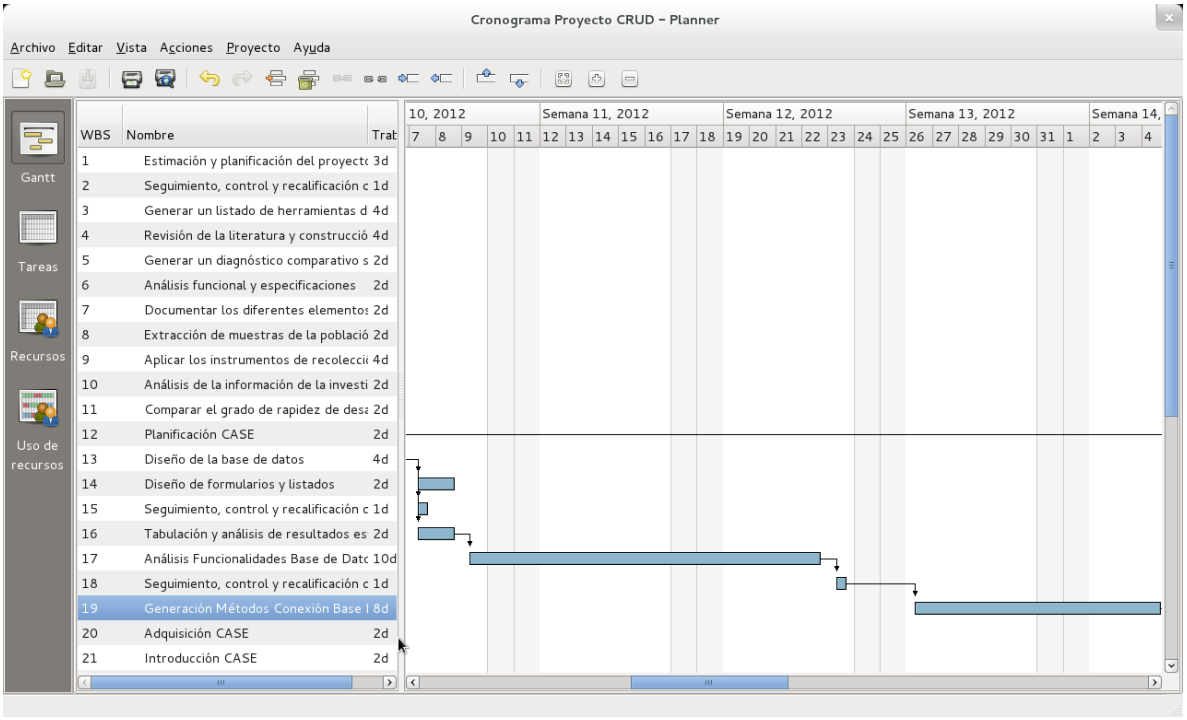
**Ilustración 6. Secuencia y duración de las tareas del proyecto**

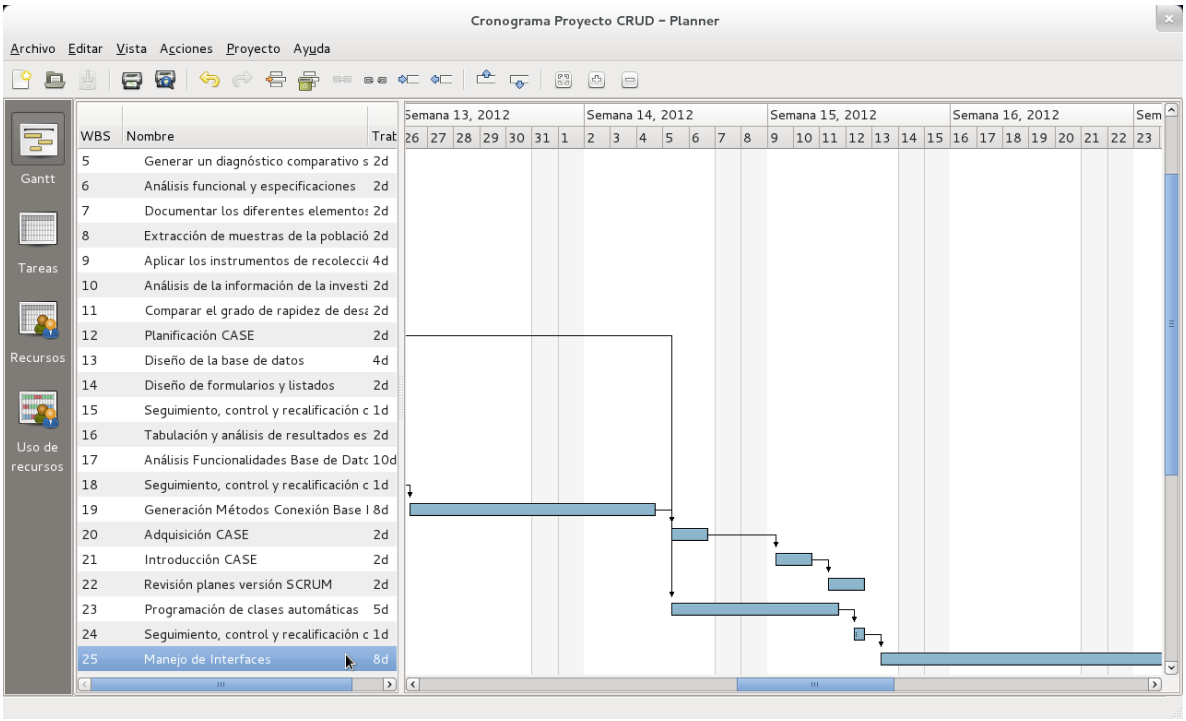
Estas tareas se distribuyen en el tiempo por un periodo cercano a los 11 meses, de acuerdo con los recursos planteados inicialmente en el proyecto:

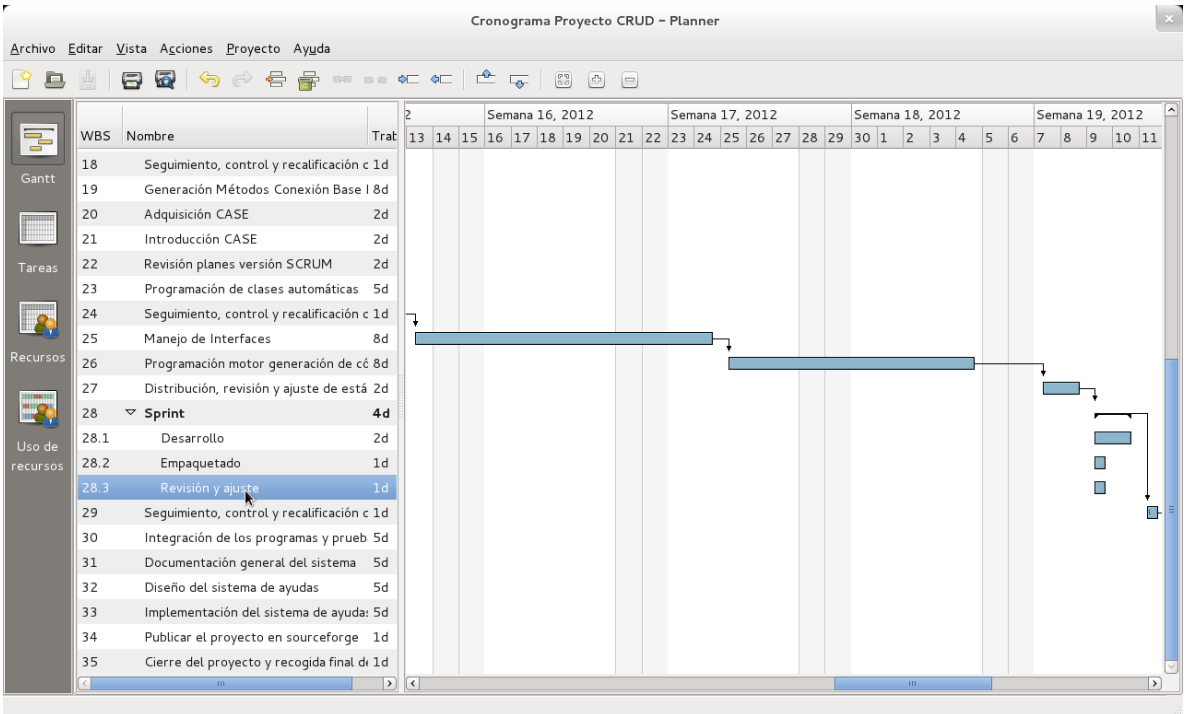



| WBS | Nombre | Trab |
|---|---|---|
| 1 | Estimación y planificación del proyecto | 3d |
| 2 | Seguimiento, control y recalificación c | 1d |
| 3 | Generar un listado de herramientas d | 4d |
| 4 | Revisión de la literatura y construcció | 4d |
| 5 | Generar un diagnóstico comparativo s | 2d |
| 6 | Análisis funcional y especificaciones | 2d |
| 7 | Documentar los diferentes elementos | 2d |
| 8 | Extracción de muestras de la población | 2d |
| 9 | Aplicar los instrumentos de recolecci | 4d |
| 10 | Análisis de la información de la investi | 2d |
| 11 | Comparar el grado de rapidez de desa | 2d |
| 12 | Planificación CASE | 2d |
| 13 | Diseño de la base de datos | 4d |
| 14 | Diseño de formularios y listados | 2d |
| 15 | Seguimiento, control y recalificación c | 1d |
| 16 | Tabulación y análisis de resultados es | 2d |
| 17 | Análisis Funcionalidades Base de Datc | 10d |
| 18 | Seguimiento, control y recalificación c | 1d |
| 19 | Generación Métodos Conexión Base I | 8d |
| 20 | Adquisición CASE | 2d |
| 21 | Introducción CASE | 2d |


Cronograma Proyecto CRUD – Planner
Archivo Editar Vista Acciones Proyecto Ayuda
Gantt
Tareas
Recursos
Uso de recursos
WBS Nombre Trab
1 Estimación y planificación del proyecto 3d
2 Seguimiento, control y recalificación c 1d
3 Generar un listado de herramientas d 4d
4 Revisión de la literatura y construcció 4d
5 Generar un diagnóstico comparativo s 2d
6 Análisis funcional y especificaciones 2d
7 Documentar los diferentes elementos 2d
8 Extracción de muestras de la població 2d
9 Aplicar los instrumentos de recolecci 4d
10 Análisis de la información de la investi 2d
11 Comparar el grado de rapidez de desa 2d
12 Planificación CASE 2d
13 Diseño de la base de datos 4d
14 Diseño de formularios y listados 2d
15 Seguimiento, control y recalificación c 1d
16 Tabulación y análisis de resultados es 2d
17 Análisis Funcionalidades Base de Datc 10d
18 Seguimiento, control y recalificación c 1d
19 Generación Métodos Conexión Base I 8d
20 Adquisición CASE 2d
21 Introducción CASE 2d
ana 7, 2012
Semana 8, 2012
Semana 9, 2012
Semana 10, 2012
Semana


Cronograma Proyecto CRUD – Planner
Archivo Editar Vista Acciones Proyecto Ayuda
Gantt
Tareas
Recursos
Uso de recursos
WBS Nombre Trab
1 Estimación y planificación del proyecto 3d
2 Seguimiento, control y recalificación c 1d
3 Generar un listado de herramientas d 4d
4 Revisión de la literatura y construcció 4d
5 Generar un diagnóstico comparativo s 2d
6 Análisis funcional y especificaciones 2d
7 Documentar los diferentes elementos 2d
8 Extracción de muestras de la població 2d
9 Aplicar los instrumentos de recolecci 4d
10 Análisis de la información de la investi 2d
11 Comparar el grado de rapidez de desa 2d
12 Planificación CASE 2d
13 Diseño de la base de datos 4d
14 Diseño de formularios y listados 2d
15 Seguimiento, control y recalificación c 1d
16 Tabulación y análisis de resultados es 2d
17 Análisis Funcionalidades Base de Datc 10d
18 Seguimiento, control y recalificación c 1d
19 Generación Métodos Conexión Base I 8d
20 Adquisición CASE 2d
21 Introducción CASE 2d
10, 2012
Semana 11, 2012
Semana 12, 2012
Semana 13, 2012
Semana 14,


Cronograma Proyecto CRUD – Planner
Archivo Editar Vista Acciones Proyecto Ayuda
Gantt
Tareas
Recursos
Uso de recursos
WBS Nombre Trab
5 Generar un diagnóstico comparativo s 2d
6 Análisis funcional y especificaciones 2d
7 Documentar los diferentes elementos 2d
8 Extracción de muestras de la población 2d
9 Aplicar los instrumentos de recolecci 4d
10 Análisis de la información de la investi 2d
11 Comparar el grado de rapidez de desa 2d
12 Planificación CASE 2d
13 Diseño de la base de datos 4d
14 Diseño de formularios y listados 2d
15 Seguimiento, control y recalificación c 1d
16 Tabulación y análisis de resultados es 2d
17 Análisis Funcionalidades Base de Datc 10d
18 Seguimiento, control y recalificación c 1d
19 Generación Métodos Conexión Base I 8d
20 Adquisición CASE 2d
21 Introducción CASE 2d
22 Revisión planes versión SCRUM 2d
23 Programación de clases automáticas 5d
24 Seguimiento, control y recalificación c 1d
25 Manejo de Interfaces 8d
Semana 13, 2012
Semana 14, 2012
Semana 15, 2012
Semana 16, 2012
26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23


Cronograma Proyecto CRUD – Planner
Archivo Editar Vista Acciones Proyecto Ayuda
Gantt
Tareas
Recursos
Uso de recursos
WBS Nombre Trab
18 Seguimiento, control y recalificación c 1d
19 Generación Métodos Conexión Base I 8d
20 Adquisición CASE 2d
21 Introducción CASE 2d
22 Revisión planes versión SCRUM 2d
23 Programación de clases automáticas 5d
24 Seguimiento, control y recalificación c 1d
25 Manejo de Interfaces 8d
26 Programación motor generación de cć 8d
27 Distribución, revisión y ajuste de está 2d
28 Sprint 4d
28.1 Desarrollo 2d
28.2 Empaquetado 1d
28.3 Revisión y ajuste 1d
29 Seguimiento, control y recalificación c 1d
30 Integración de los programas y prueb 5d
31 Documentación general del sistema 5d
32 Diseño del sistema de ayudas 5d
33 Implementación del sistema de ayuda: 5d
34 Publicar el proyecto en sourceforge 1d
35 Cierre del proyecto y recogida final d 1d
Semana 16, 2012
Semana 17, 2012
Semana 18, 2012
Semana 19, 2012
13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10 11

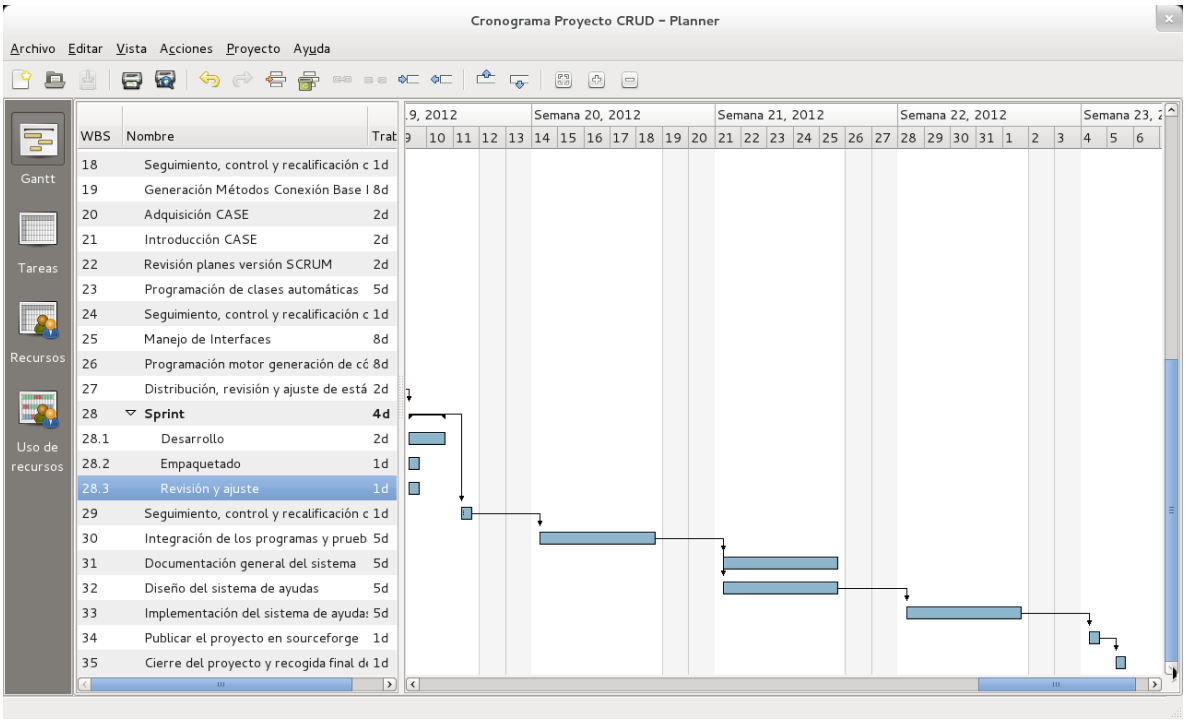



**Ilustración 7. Diagrama de Gantt Cronograma Proyecto**

Los recursos que requiere este proyecto pueden ser cuantificados de la siguiente manera:

| Recurso | Cantidad | Valor |
|---|---|---|
| Computadores | 2 | 4’000.000 |
| Papelería | | 1’000.000 |
| Jefe Proyecto | 1 * 5 meses | 15’000.000 |
| Analista | 2 * 5 meses | 20’000.000 |
| Desarrollador | 4 * 5 meses | 30’000.000 |
| Asesorías | | 10’000.000 |
| Otros gastos | | 2’000.000 |
| **TOTAL** | | **82’000.000** |

**Tabla 4. Presupuesto del proyecto**

## 3.1. Desarrollo Metodológico

### 3.1.1. Requerimientos de una Herramienta de Generación Automática de Código Considerados por los Desarrolladores de Software Libre de la Región

#### 3.1.1.1. Software Libre de Boyacá

El grupo de "Software Libre de Boyacá", se define a sí mismo como: "*El grupo de Software Libre Boyacá es un punto de encuentro de saberes, ideas, iniciativas y actividades en torno al Software Libre ubicado en el contexto de Boyacá pero sin desconocer la necesidad de interactuar con el exterior*"[57].


**Ilustración 8. Encabezado Página Web Software Libre de Boyacá (Fuente: http://softwarelibreboyaca.org)**

Es un grupo de personas voluntarias que pretende difundir y adaptar el software libre en el Departamento de Boyacá[58] y sus regiones circunvecinas, conformado por entusiastas y expertos en el software libre. El grupo está conformado por personas de diversas profesiones y actividades, como profesores de universidades y de

[57] Software Libre de Boyacá. Definición. Recuperado de http://softwarelibreboyaca.org/index.php/es/quienes-somos, en Junio de 2012

[58] Departamento de Boyacá Colombia. Mapa del Departamento de Boyacá. Recuperado de http://maps.google.com/maps?hl=es&bav=on.2,or.r_gc.r_pw.r_qf.&bpcl=39650382&biw=1280&bih=671&q=boyaca&um=1&ie=UTF-8&hq=&hnear=0x8e69a475f890a283:0xffe1255b64bc9ff7,Boyac%C3%A1&gl=co&sa=X&ei=OCDHULK9I4eq8ASIj4CgCw&ved=0CKIBELYD en Junio de 2012

instituciones de educación media, estudiantes, líderes comunales, entusiastas, entre otras, quienes se dedican principalmente a seis actividades, que pueden ser apreciadas en la Ilustración 9:

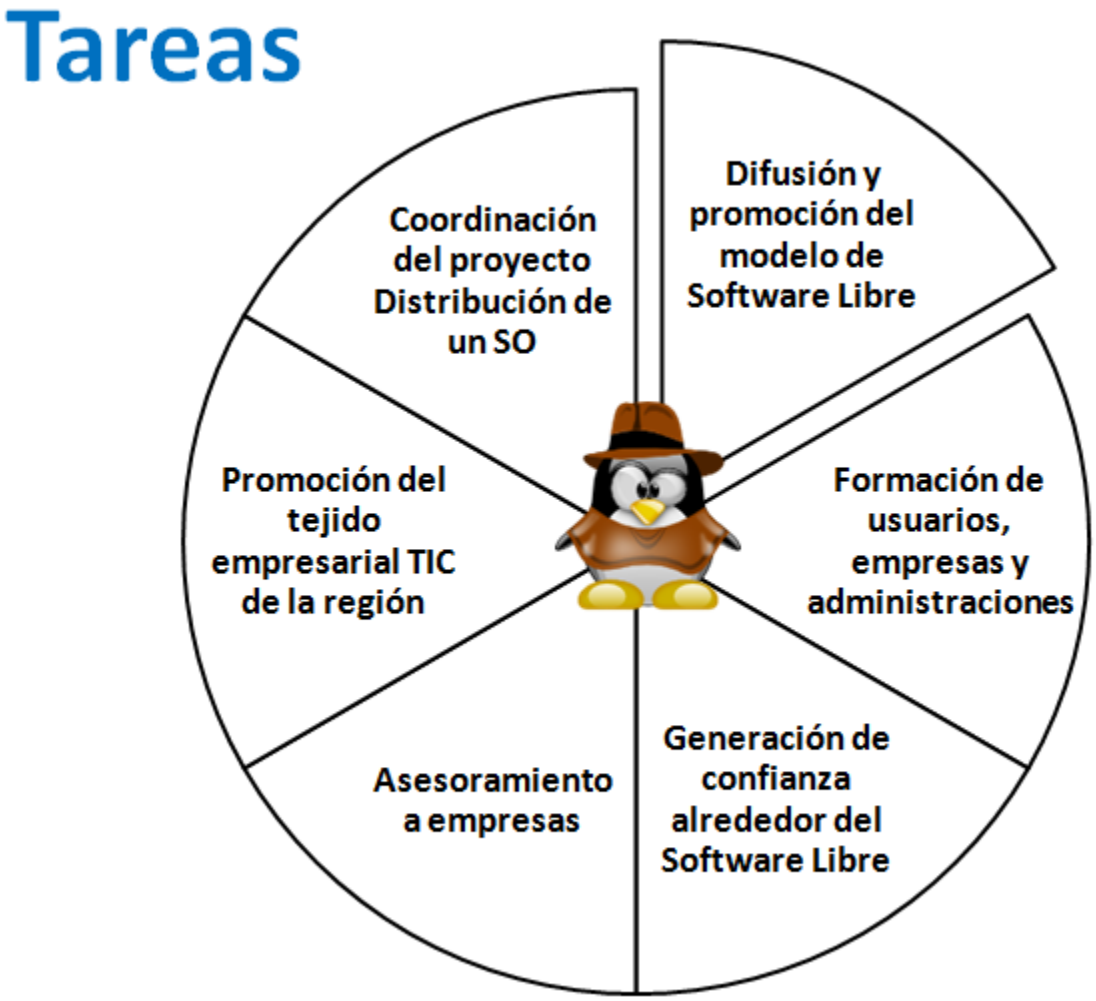



**Ilustración 9. Actividades Grupo Software Libre Boyacá (Fuente: ídem)**

Como trabajos destacados del grupo, en su corta edad (se conformó a inicios del año 2011), caben mencionar: Organización de la visita de Richard Stallman a Colombia y en especial a Boyacá, organización de eventos como FLISOL 2012 (Festival Latinoamericano de Instalación de Software Libre), el JSL (Jornadas de Software Libre), SFD (Software Freedom Day), muestras de software libre, instalación de software libre para mipymes y desarrollo de software libre para veeduría ciudadana en la ciudad de Duitama Boyacá.

### 3.1.1.2. Muestra de la Población Objeto de Estudio

Para los intereses de este trabajo de grado, especialmente en su primer objetivo y de acuerdo con la propuesta metodológica, encuentra en el grupo de Software Libre de Boyacá, como la muestra de la población objetivo, necesaria para indagar por las características contextualizadas en la región, en especial en el desarrollo de software, grupo que apreciaría, en un principio, por disponer de herramientas de generación automática de software libre que beneficien a la propia comunidad. Aunque inicialmente el instrumento estadístico se aplicaría con los miembros en general del grupo, su enfoque de interés son los miembros desarrolladores o involucrados en el desarrollo de software del mismo grupo (**Ver Anexo 1. Encuesta de Percepción**).

Para determinar los requerimientos que debe reunir una herramienta de generación automática de código, considerados por los desarrolladores de software libre de la región, se parte de los resultados de este instrumento estadístico que se pueden apreciar en el **Anexo 2. Resultados de la aplicación del instrumento estadístico**. Los resultados de la aplicación del instrumento demuestran, entre otros, factores importantes a tener en cuenta:

- La muestra de población demuestra que conocen de desarrollo de software
- La mayoría ha trabajado con lenguajes de programación con licencia de software libre
- Muchos de ellos conocen herramientas CASE o de generación automática de software
- Existe una gran aceptación para que se desarrolle una herramienta CASE contextualizada en la región
- Proponen unos requerimientos importantes para desarrollar este proyecto como son la reutilización del código generado y que la aplicación resultante sea portable. Además que sea desarrollada para ambientes web y móviles principalmente, sin descuidar ambientes de escritorio.
- Son más las ventajas que desventajas presentes en este tipo de herramientas, de las cuales cabe destacar la posibilidad de minimizar el esfuerzo de programación, desarrollar un código en cumplimiento de estándares
- Los encuestados consideran que con este tipo de herramientas pueden ser más efectivos en el aporte a la comunidad de software libre, pueden pasar de ser dependientes tecnológicos a proponentes tecnológicos.

### 3.1.1.3. Fuentes de la Identificación de Requerimientos

La fuente primaria de la investigación está constituida por la aplicación del instrumento estadístico aplicado a la muestra de la población objeto de estudio. Con los resultados de esta encuesta se perfila la identificación de los requerimientos que debe reunir la herramienta de generación automática de código contextualizada en las necesidades de la comunidad de software libre en la región. Sin embargo, otras fuentes de información, especialmente las relacionadas con el estándar de herramientas CASE, propuesta en la norma ISO / IEC 14102,

como el trabajo monográfico de Elison Roberto Imenes[59], el Método GQM[60], el *paper* de "I Gusti Made Aditya"[61], el artículo de indicadores organizacionales para la selección de Herramientas CASE[62], Criterios para la Selección de Herramientas de Ingeniería de Software en PYMEs[63] y el estado de arte de este proyecto, ayudaron a identificar estos requisitos.

#### 3.1.1.4. Identificación de Requerimientos

Es variada la clasificación de herramientas CASE de acuerdo con la plataforma que atienten, las fases del ciclo de vida del desarrollo de software que cubren, la arquitectura de las aplicaciones y la función para que están destinadas[64]. Por consiguiente, las herramientas CASE se pueden agrupar de la siguiente forma:

- I-CASE: CASE Integrado (CASE Workbench). Abarcan todas las fases del ciclo de vida del software.
- U-CASE: CASE Superior (Front-end). Primeras fases del desarrollo: análisis y diseño
- L-CASE: CASE Inferior (Back-end). Últimas fases del desarrollo: construcción e implantación
- Tools-CASE: Juego de Herramientas. Herramientas CASE simples, automatizan una fase del ciclo de vida.

La herramienta propuesta en este proyecto de investigación se clasificaría dentro de la última categoría, porque parte del diccionario de una base de datos previamente creada, concebida como una herramienta de programación para software de sistemas, es decir solamente es una herramienta generadora de código, no modela, ni tampoco controla el proyecto de ingeniería.

El proceso de validación o selección de una herramienta CASE, debe pasar por los siguientes procesos:

[59] ROBERTO IMENES, Elison. Seleção de Ferramentas CASE. Jaguariúna. 2006
[60] SOLIGEN, Rini van y BERGHOUT, Egon. The Goal / Question / Metric Method: A Practical Guide for Quality Improvement of Software Development. The McGraw-Hill Companies, London, England, 1999
[61] I Gusti Made Aditya. CASE Tools Comparison. 2009
[62] MENDOZA, Luis Eduardo, et al. Organizational Indicators for CASE Tools Selection: A Case Study. Revista Colombiana de Computación – RCC. Editorial UNAB.
[63] RIVAS, Lornel, et al. Criterios para la Selección de Herramientas de Ingeniería de Software en PYMEs. Universidad Simón Bolívar. Revista de la Facultad de Ingeniería UCV. Vol. 25, No 1, pp. 89-104. 2010
[64] INEI. Herramientas CASE. Perú. 1999

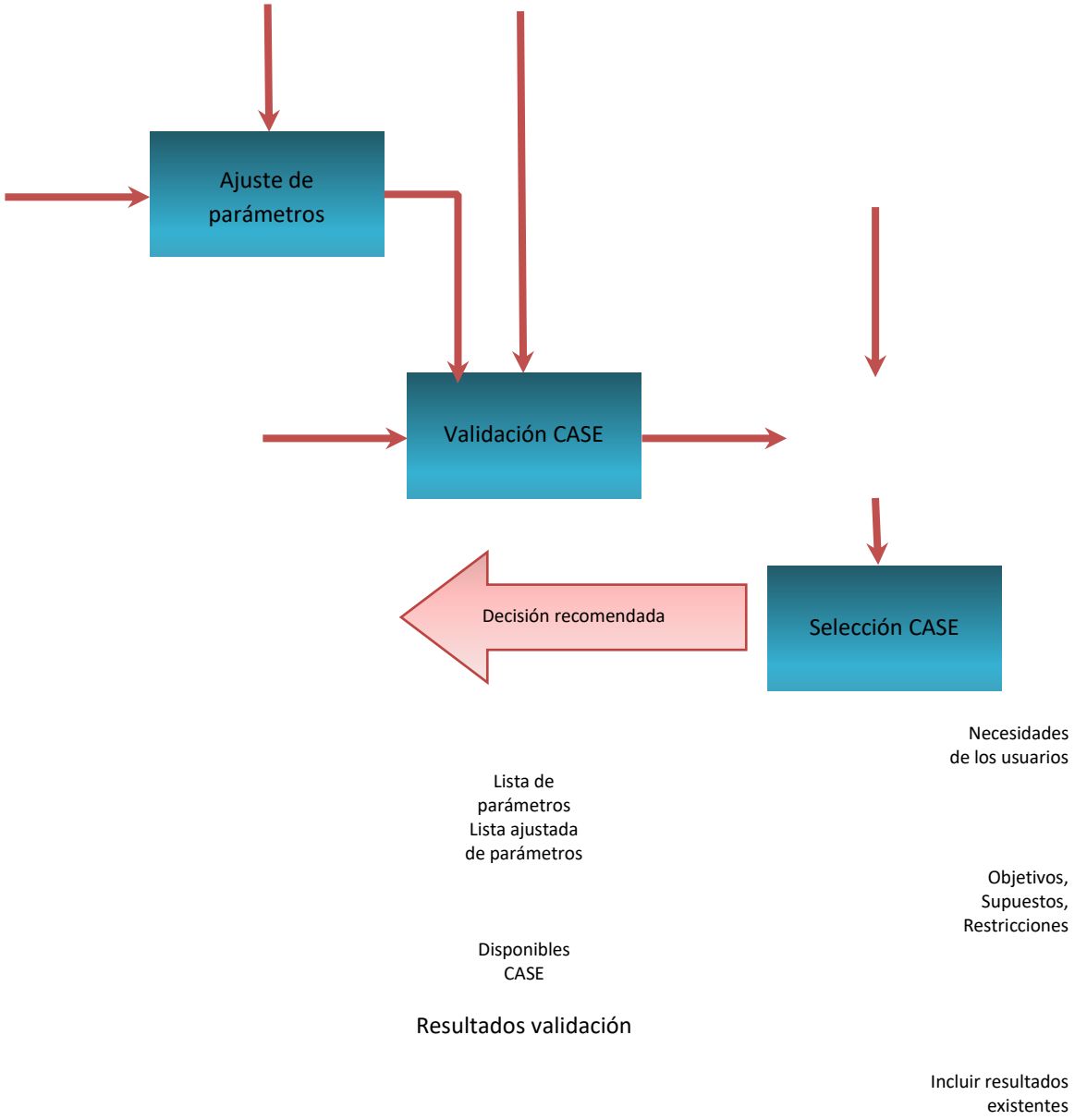



**Ilustración 10. Proceso de Validación y Selección Fuente: (ROBERTO IMENES, 2006)**

De acuerdo con la Ilustración 10. Proceso de Validación y Selección Fuente: (ROBERTO IMENES, 2006)(ROBETO IMENES, 2006), el proceso de evaluación y selección de una herramienta puede servir para diversas aplicaciones, como:

- Validación de herramientas CASE y selección de una de ellas,
- Mantener los resultados para futuros usos,
- Seleccionar una determinada herramienta usando los datos de validaciones anteriores.

| Ambiente | Requerimiento | Descripción |
|---|---|---|

| | | |
|---|---|---|
| **Componentes** | Repositorio | El repositorio amplia el concepto al diccionario de datos, para incluir la información de la base de datos seleccionada |
| | Módulo de Diagramación y Modelización | No aplica al proyecto |
| | Herramienta de prototipado | No aplica al proyecto |
| | Generador de código | Generalmente en una estación de trabajo<br>**Lenguaje generado**: Lenguaje estándar libre<br>**Portabilidad código generado**: Ejecución en principales preferencialmente en plataformas de entorno libre<br>**Generación del framework o del programa completo**: Genera el código completo de la interfaz del usuario CRUD, aunque será necesario completar manualmente las funciones adicionales de la aplicación<br>**Posibilidad de modificar el código generado**: Si, accediendo de forma directa a éste<br>**Generación de código asociado a las pantallas e informes de la aplicación**: Sí, se obtendrá la interfaz del usuario de la aplicación |
| | Módulo generador de la documentación | **Generación automática a partir de los datos del repositorio**: Si<br>**Combinación de información textual y gráfica**: No aplica<br>**Generación de referencias cruzadas**: No aplica<br>**Ayuda de tratamiento de textos**: No aplica<br>**Interfaz con otras herramientas**: Editores de texto |
| **Comunidad Software Libre** | Impacto | El grado de impacto del Sistema Información creado por el área de desarrollo, que tenga una misión, objetivos y operaciones de la comunidad. |
| | Posicionamiento | La posición jerárquica del desarrollo dentro de la comunidad, con respecto a si es considerada como una unidad de servicio o si es considerada una unidad estratégica. |
| | Dependencia | Grado de dependencia de la comunidad con el Sistema de Información desarrollado para permitir alcanzar sus objetivos |
| | Apoyo directivo | Soporte dado por la administración de la comunidad a los planes de inversión tecnológica asignados al desarrollo del sistema de información |
| | Resistencia a la innovación | Grado de resistencia de la organización contra la incorporación de novedades |
| | Visión estratégica | Grado de relevancia garantizado por la organización para el uso de la aplicación |
| **Implementación** | Sintaxis | Sintaxis relacionada con el lenguaje de programación del código generado |
| | Generación de código | Paradigma del código generado |
| | Generación de esquemas de banco de datos | No aplica. El banco de datos es de entrada. |
| | Compilación | Dependiente del lenguaje del código generado |
| | Conversión de código fuente | Dependiente del lenguaje del código generado |
| | Análisis de confiabilidad | A cargo del equipo de *testing* |

| | | |
|---|---|---|
| | Ingeniería inversa | Del diccionario de datos al lenguaje de programación del código generado |
| | Reestructuración del código | Manual, a cargo del equipo de desarrollo |
| | Análisis de código fuente | Generación de código documentado |
| | Depuración | Generación de archivos log del proceso, para posterior análisis del equipo de mantenimiento y desarrollo |
| **Pruebas** | Definición de pruebas | A partir del archivo de log |
| | Ejecución automática de pruebas | No aplica |
| | Análisis de resultados de pruebas | No aplica |
| | Análisis de cobertura de pruebas | No aplica |
| | Análisis de rendimiento | No aplica |
| | Capacidades de simulación | No aplica |
| **Usabilidad** | Consistencia de la interfaz del usuario | De acuerdo con interfaces CRUD |
| | Internacionalización | No aplica |
| | Facilidad de aprendizaje durante la operación | Aplicación tipo asistente (Wizard) |
| | Facilidad de adaptación | Aplicación tipo asistente (Wizard) |
| | Calidad de la documentación | No aplica |
| | Disponibilidad para entrenamiento | No aplica |
| | Conocimiento previo requerido | Desarrollo de software en el lenguaje de programación del código generado |
| | Uniformidad de la interface con el usuario | Interfaces CRUD |
| | Ayuda en Línea | Desarrollo manual posterior |
| | Tiempo de respuesta | Tiempos promedio de respuesta a solicitudes Web |
| | Facilidad de instalación | Aplicación tipo asistente (Wizard) |
| **Eficiencia** | Requisitos de almacenamiento | Almacenamiento del código fuente generado |
| | Requisitos de memoria | Estación de trabajo |
| | Requisitos de procesador | Estación de trabajo |
| | Workload | Exclusiva en tiempo de generación, no existe manejo multi-thread |
| | Rendimiento | Aplicación Web |
| **Mantenibilidad** | Atención de solicitudes | Repositorio de software libre |
| | Atención de actualizaciones | Repositorio de software libre, por la comunidad |
| | Atención de solicitudes de compatibilidad | Repositorio de software libre |
| **Portabilidad** | Compatibilidad de versiones de sistemas operativos | Desarrollado en Java. Compatibilidad de esta plataforma |
| | Escalabilidad | Comunidad software libre |
| | Patrones | Licencia software libre |
| **Criterios generales** | Costo de implantación | Licencia software libre |
| | Influencia de la comunidad | Altamente dependiente de las políticas de la comunidad de software libre |

| | | |
|---|---|---|
| | Entrega | Repositorio de software libre |

**Tabla 5. Requerimientos que debe reunir una herramienta de generación automática de código considerados por los desarrolladores de software libre de la región**

### 3.1.2. Realizar un Diagnóstico Comparativo del Uso de Algunas Herramientas Actuales, para la Generación Automática de Código, a partir de una Base de Datos

A partir del estado de arte realizado en esta investigación y con el apoyo de material bibliográfico, principalmente de (ROBERTO IMENES, 2006), (RIVAS & al., 2010) y (INEI, 1999), se pudieron analizar algunas herramientas CASE que generaban el código a partir de una base de datos, en la mayoría de los casos usando versiones de prueba, porque son herramientas con licencia privativa, y en otros casos, basándose en el análisis previo que algunos autores hayan hecho.

#### 3.1.2.1. E.World PHPMaker

| Indicador | Descripción |
|---|---|
| **Usa un SGBD** | Sí, es la principal función de la herramienta, soporta motores de MySQL, PostreSQL, Microsoft Access y Microsoft SQL Server |
| **Licencia** | Privativo |
| **¿Filosofía de Arquitectura abierta?** | No |
| **Genera utilidades de software, procedimientos de lectura, biblioteca de fuentes y crea especificaciones** | Si |
| **Lenguaje de programación del código generado** | Php, aunque existen otros productos similares de la misma empresa que generan el código fuente en lenguajes de programación distintos |
| **Tiene interfaz con otras herramientas** | No |
| **Dispone de utilidades gráficas, capaces de generar diagramas de proyecto y especificaciones del diccionario** | No, basa su manejo en tablas |
| **Provee capacidades de prototipado** | No, genera el código resultante directamente |
| **¿Genera automáticamente un ámbito de aplicación de las especificaciones de diseño físico de las especificaciones del proyecto?** | Si, lee el diccionario de datos, del modelo físico de datos y genera las especificaciones del proyecto |
| **Apoya el análisis y documenta el diseño** | No, se basa únicamente sobre el diccionario de la base de datos |
| **¿Genera automáticamente los informes sobre las especificaciones del proyecto?** | No, aunque maneja documentación en el código fuente |

| | |
|---|---|
| **¿Genera documentación para el usuario final?** | No |
| **¿Ofrece diferentes opciones de estilo o temas para el usuario final?** | Si, basados en CSS |
| **¿Maneja roles de usuarios y esquemas de seguridad para ellos?** | Si, basados en una tabla adicional (o existente) que se crea en la base de datos |
| **¿Posibilidad de exportar los reportes a otros formatos, o de enviar vía correo electrónico?** | Si, tiene opción de imprimir, enviar al correo electrónico, o exportar en formatos csv, html, xls, doc, xml y pdf. |
| **¿Permite formas de navegación en la aplicación resultante?** | Si, mediante un generador de menús |
| **¿Ofrece ingeniería inversa?** | No, se limita a leer el diccionario de datos |

Tabla 6. Diagnóstico comparativo de phpMaker

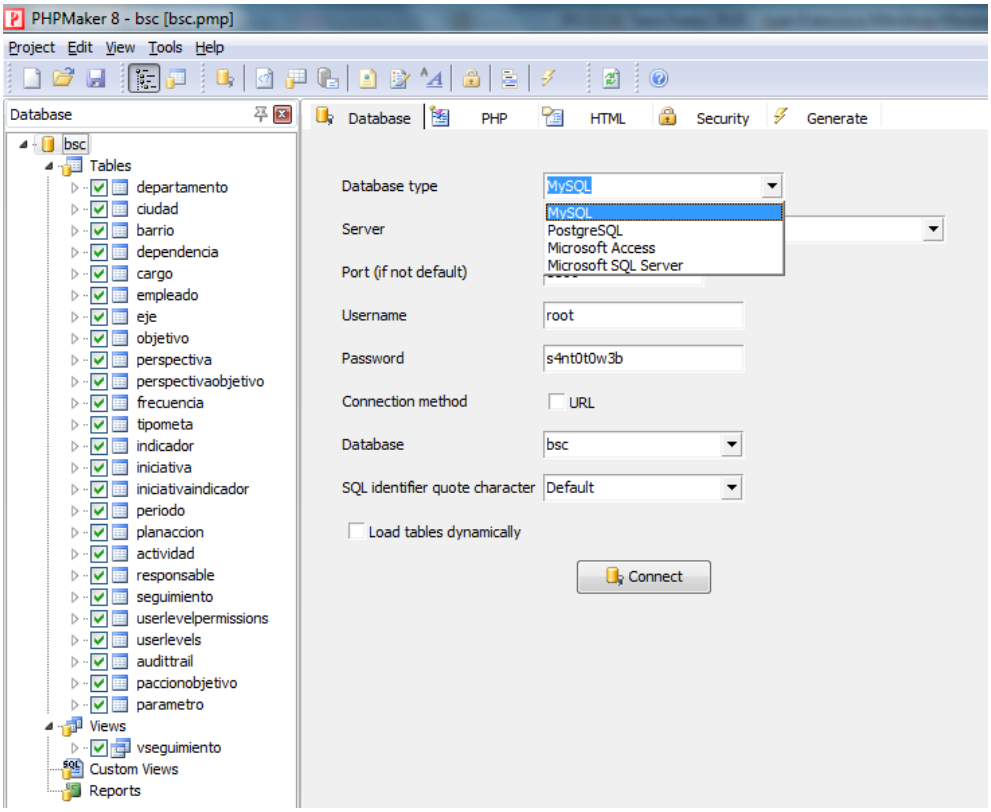



Ilustración 11. Opciones de DBMS de phpMaker

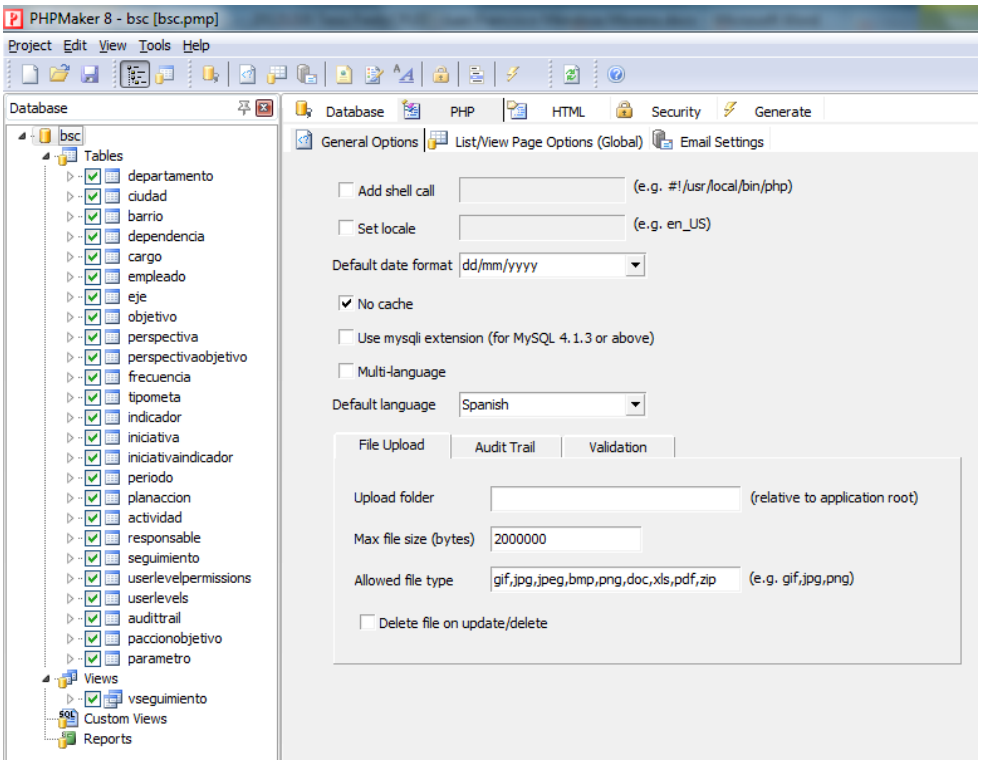



**Ilustración 12. Utilidades generadas y lenguaje generado (php)**

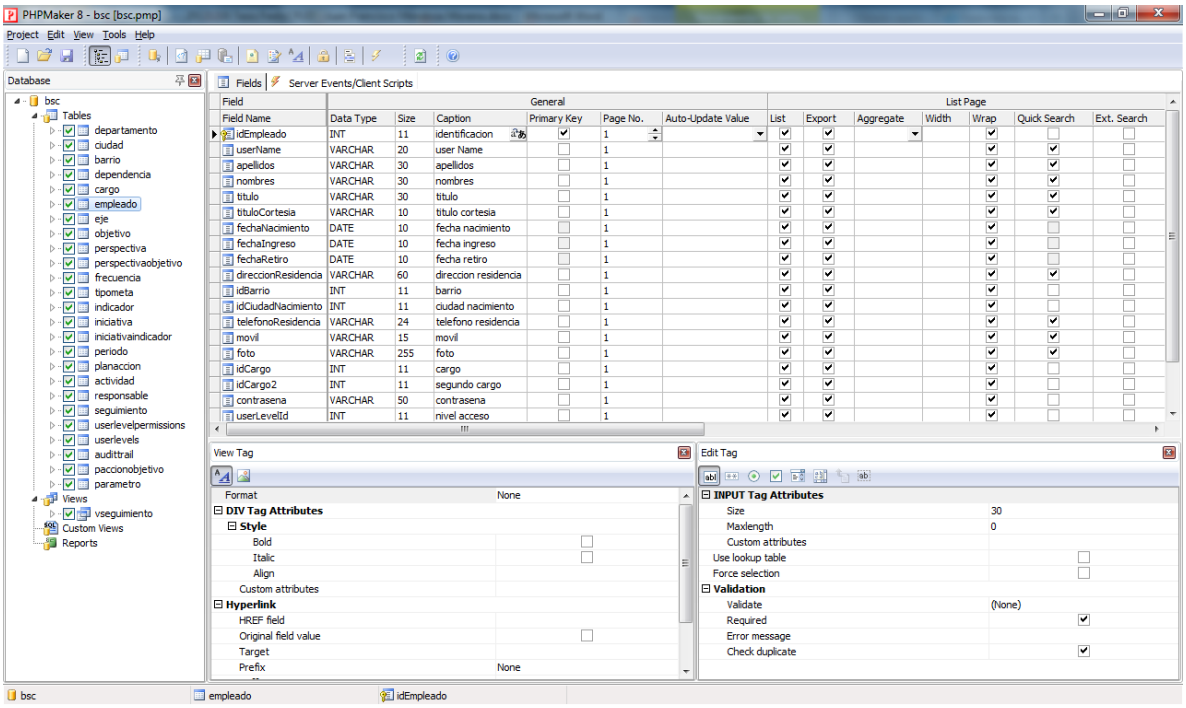



**Ilustración 13. Lectura del diccionario de datos por parte de phpMaker**

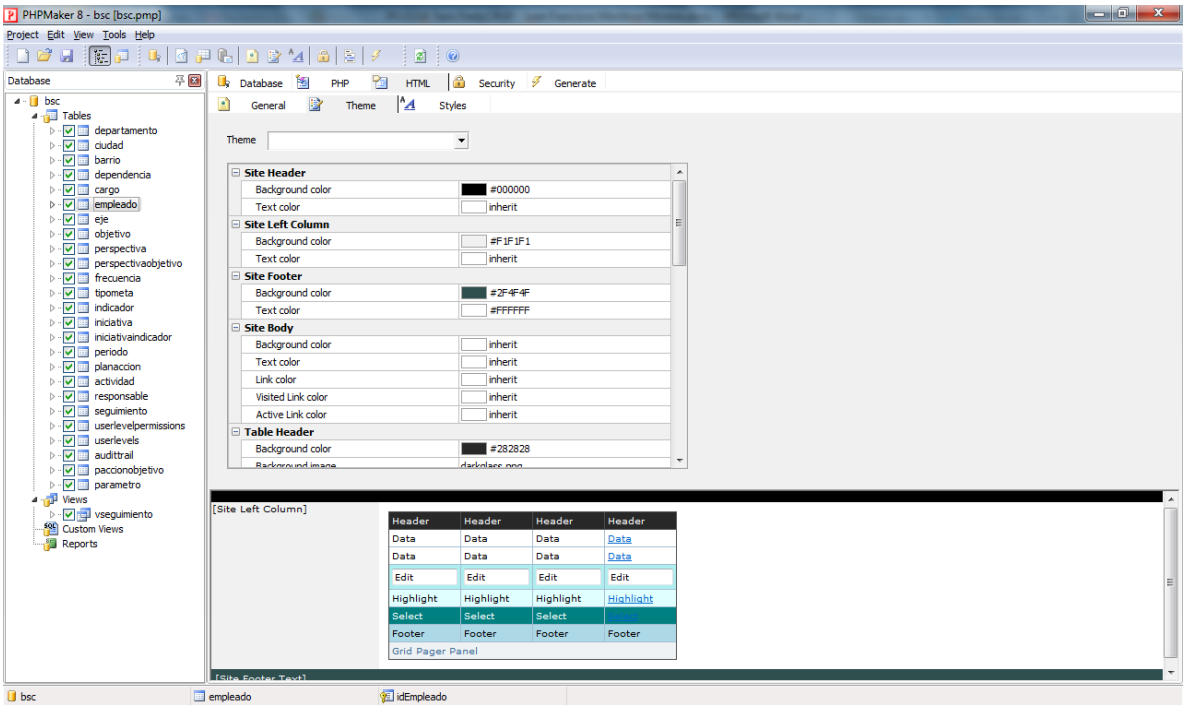



**Ilustración 14. Manejo de temas de phpMaker**

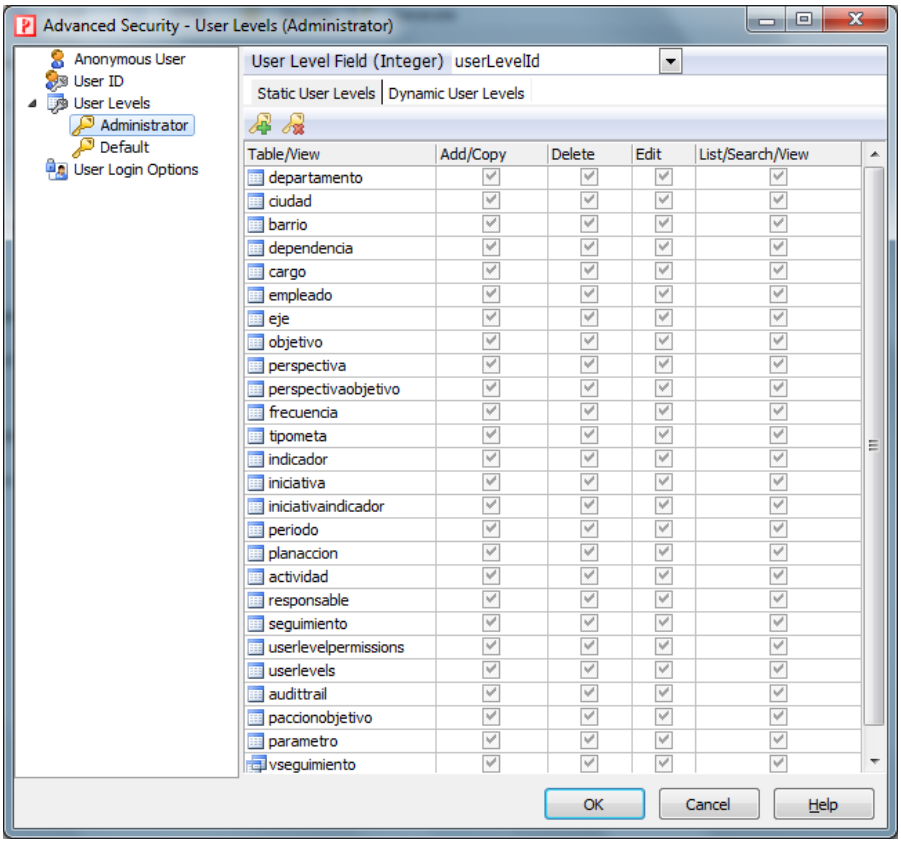



**Ilustración 15. Esquema de seguridad y de roles de la aplicación generada en phpMaker**

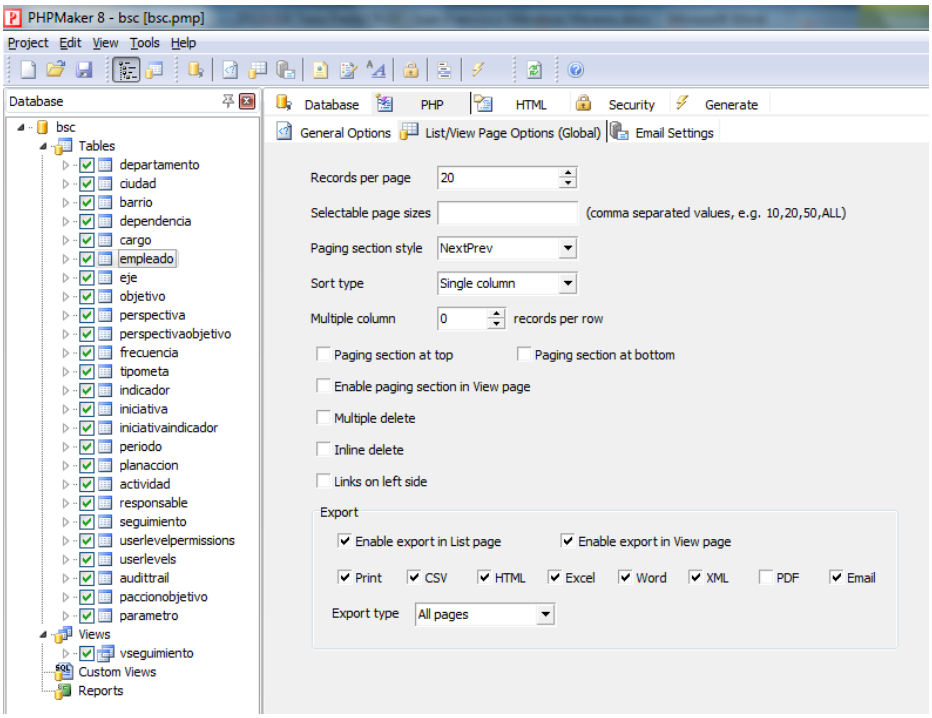



**Ilustración 16. Posibilidades de exportación de phpMaker**

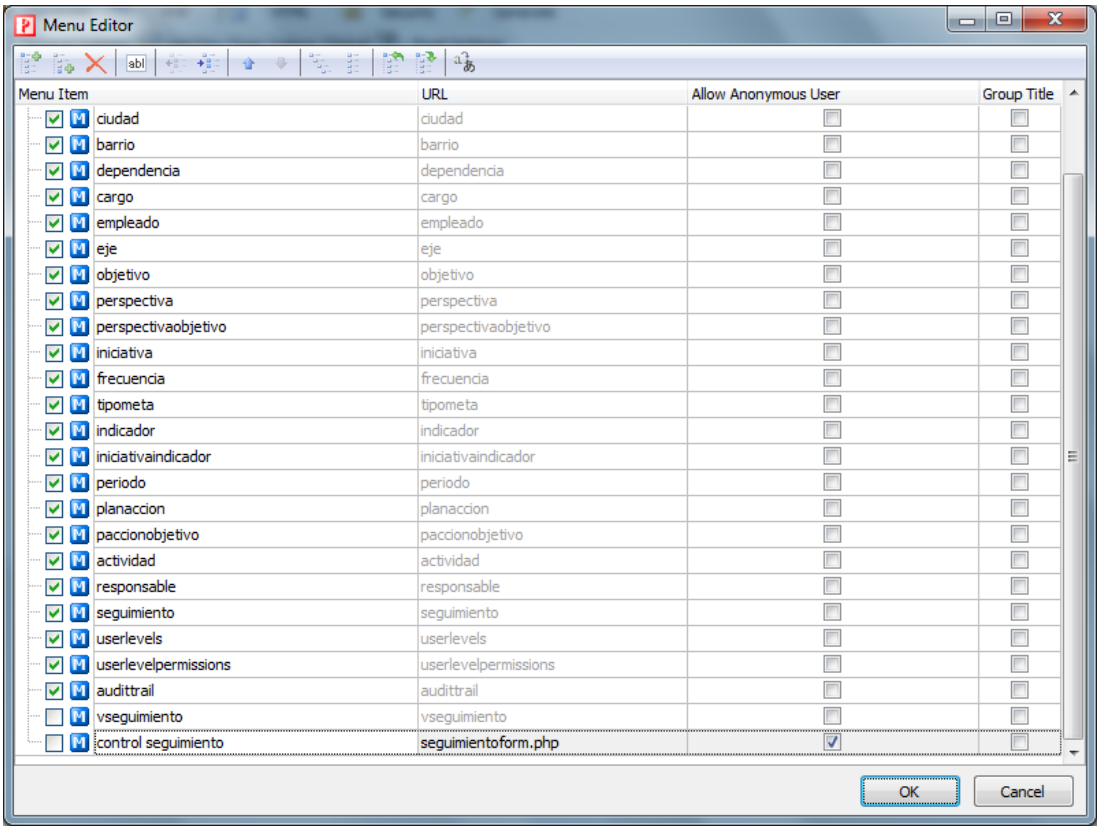



**Ilustración 17. Generador de menús de phpMaker**

### 3.1.2.2. Sybase PowerDesigner

| Indicador | Descripción |
|---|---|
| **Usa un SGBD** | Sí, mediante ingeniería inversa. Soporta 44 motores de bases de datos, además ofrece soporte a ODBC o archivos script SQL. Además soporta lenguajes de procesos, lenguajes de objetos e importación por formato XML |
| **Licencia** | Privativo |
| **¿Filosofía de Arquitectura abierta?** | Si, UML, Modelo XML, BPMN, Modelo de Requerimientos |
| **Genera utilidades de software, procedimientos de lectura, biblioteca de fuentes y crea especificaciones** | Si |
| **Lenguaje de programación del código generado** | A partir del modelo de objetos puede generar código en quince lenguajes de programación |
| **Tiene interfaz con otras herramientas** | Si, con los productos Sybase, mediante archivos XMI |
| **Dispone de utilidades gráficas, capaces de generar diagramas de proyecto y especificaciones del diccionario** | Si, agrupados en 12 grandes modelos |
| **Provee capacidades de prototipado** | No |
| **¿Genera automáticamente un ámbito de aplicación de las especificaciones de diseño físico de las especificaciones del proyecto?** | Si |
| **Apoya el análisis y documenta el diseño** | Si |
| **¿Genera automáticamente los informes sobre las especificaciones del proyecto?** | Si |
| **¿Genera documentación para el usuario final?** | Si, por medio de metamodelo de ayudas |
| **¿Ofrece diferentes opciones de estilo o temas para el usuario final?** | No |
| **¿Maneja roles de usuarios y esquemas de seguridad para ellos?** | Si, en el momento del modelamiento |
| **¿Posibilidad de exportar los reportes a otros formatos, o de enviar vía correo electrónico?** | Si |
| **¿Permite formas de navegación en la aplicación resultante?** | Si, mediante un generador de menús |
| **¿Ofrece ingeniería inversa?** | Si |

Tabla 7. Diagnóstico comparativo de PowerDesigner

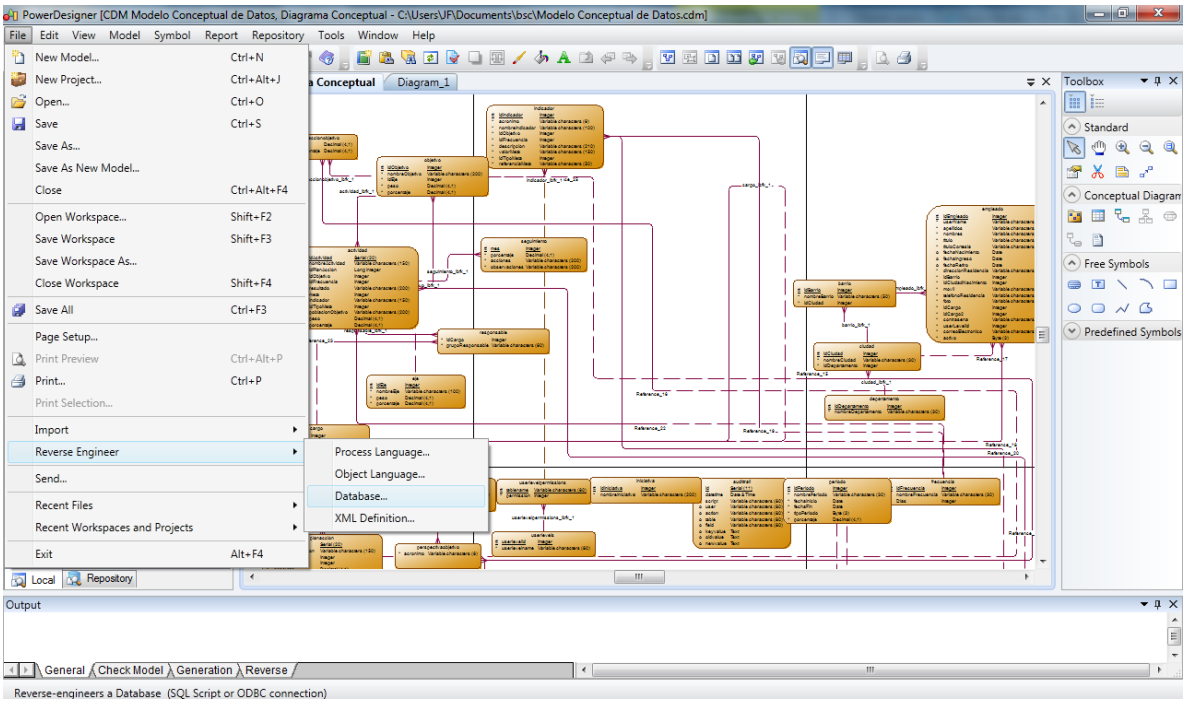



**Ilustración 18. Ingeniería Inversa en Sybase para DBMS**

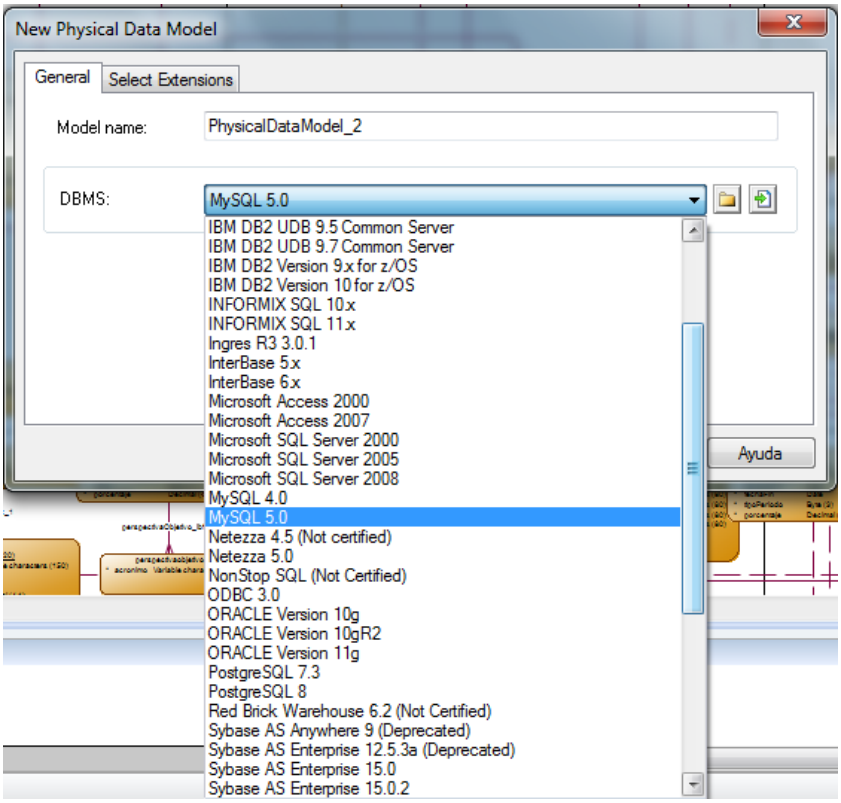



**Ilustración 19. Algunos motores de DBMS soportados por Sybase**

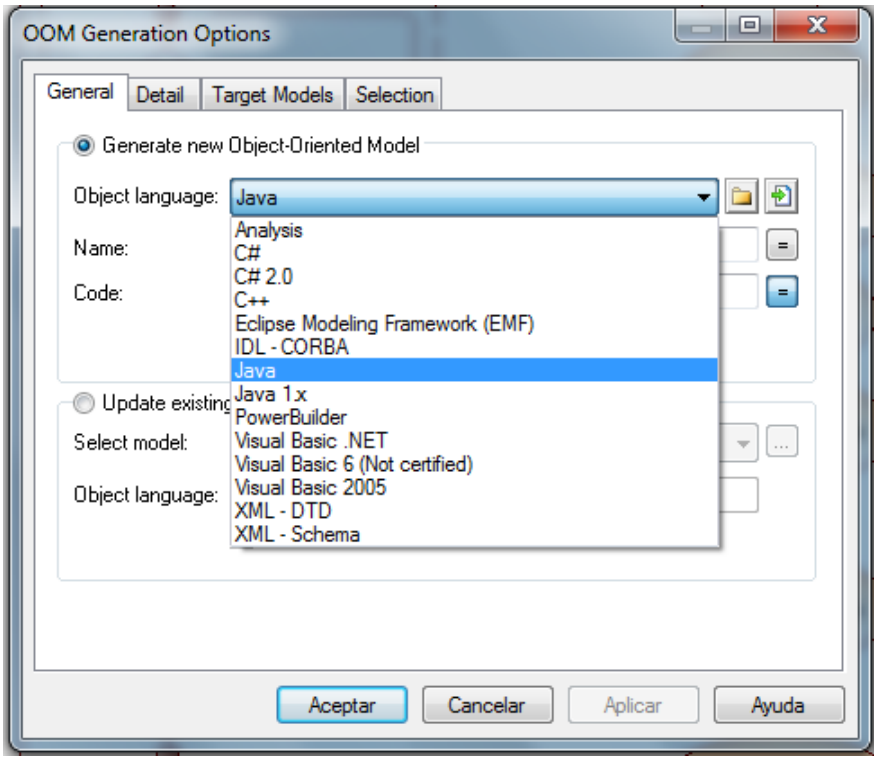



**Ilustración 20. Lenguajes de programación del código generado en Sybase**

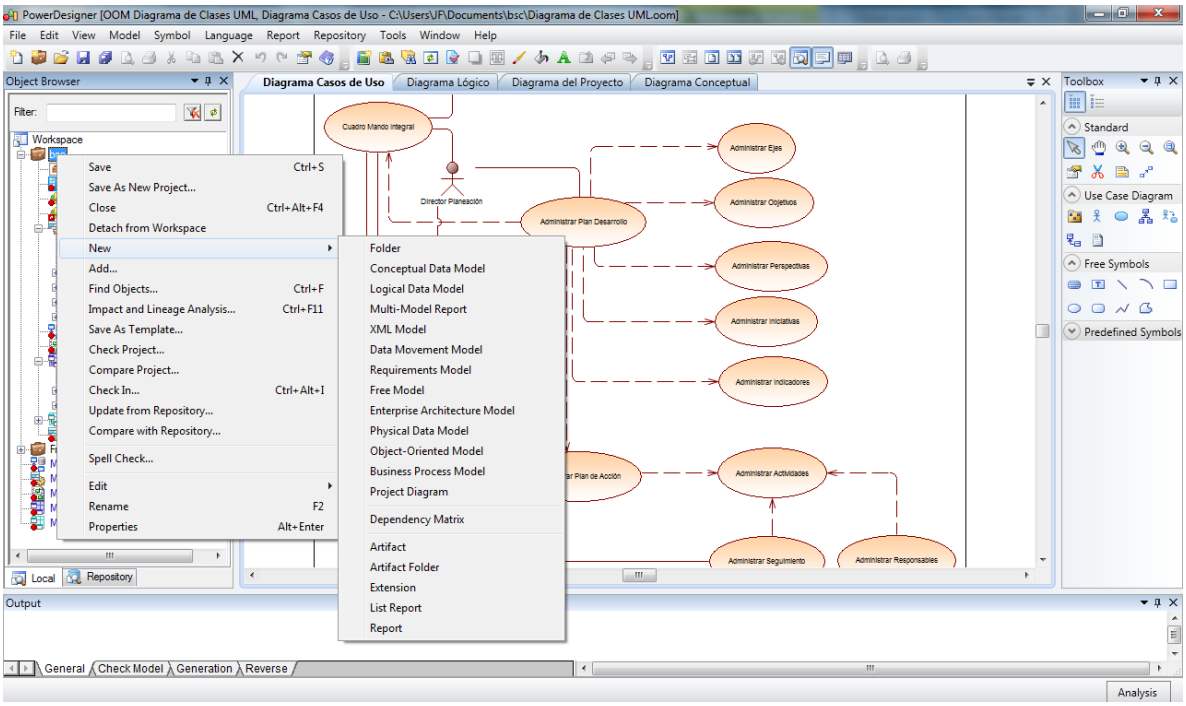



**Ilustración 21. Grandes agrupaciones de modelos**

### 3.1.2.3. Sparx Enterprise Architect

| Indicador | Descripción |
|---|---|
| **Usa un SGBD** | Sí, mediante ingeniería inversa. Mediante conexión ODBC |
| **Licencia** | Privativo |
| **¿Filosofía de Arquitectura abierta?** | Si, UML 2.1 |
| **Genera utilidades de software, procedimientos de lectura, biblioteca de fuentes y crea especificaciones** | Si |
| **Lenguaje de programación del código generado** | Si, para 9 lenguajes de programación |
| **Tiene interfaz con otras herramientas** | No |
| **Dispone de utilidades gráficas, capaces de generar diagramas de proyecto y especificaciones del diccionario** | Si, 13 diagramas de UML agrupados en 3 áreas: Diagramas estructurales, diagramas de comportamiento y extendidos |
| **Provee capacidades de prototipado** | Si, modelamiento UML |
| **¿Genera automáticamente un ámbito de aplicación de las especificaciones de diseño físico de las especificaciones del proyecto?** | Si |
| **Apoya el análisis y documenta el diseño** | Si |
| **¿Genera automáticamente los informes sobre las especificaciones del proyecto?** | Si |
| **¿Genera documentación para el usuario final?** | No |
| **¿Ofrece diferentes opciones de estilo o temas para el usuario final?** | No |
| **¿Maneja roles de usuarios y esquemas de seguridad para ellos?** | Si, en UML |
| **¿Posibilidad de exportar los reportes a otros formatos, o de enviar vía correo electrónico?** | No |
| **¿Permite formas de navegación en la aplicación resultante?** | No |
| **¿Ofrece ingeniería inversa?** | Si |

Tabla 8. Diagnóstico Comparativo de Enterprise Architect

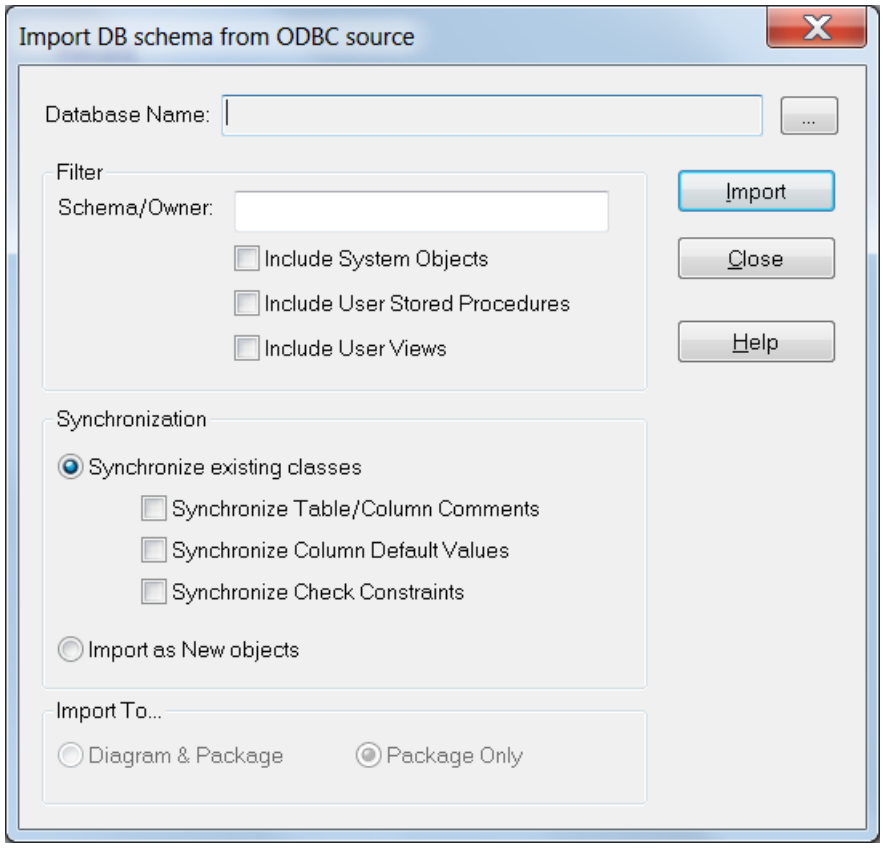



**Ilustración 22. Soporte de DBMS de Enterprise Architect**

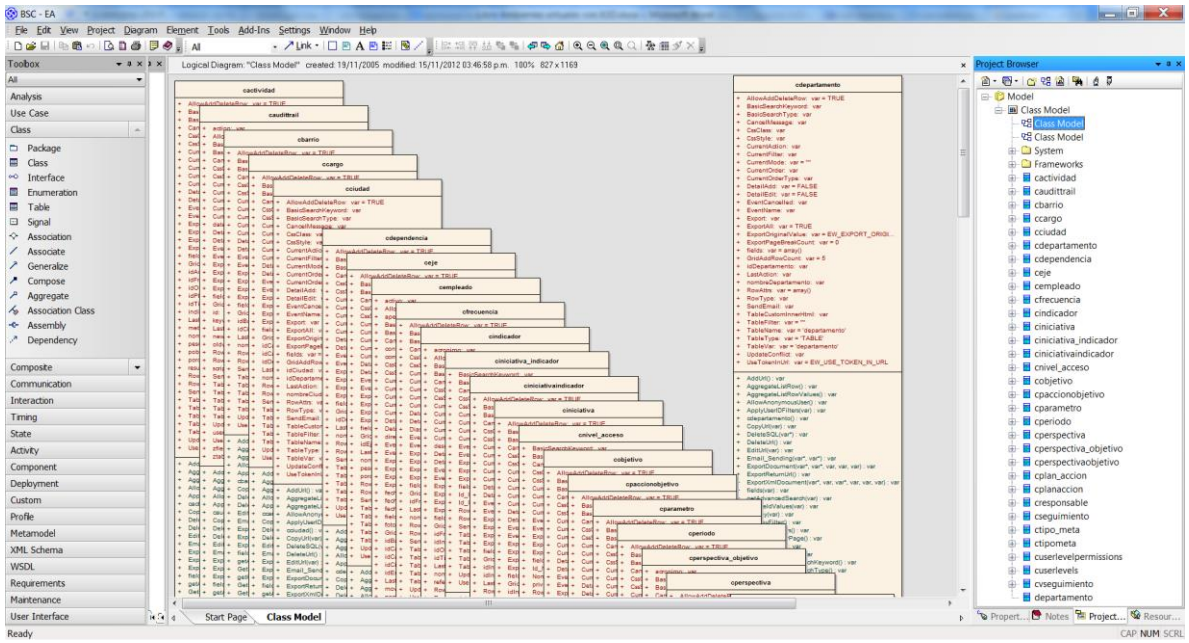



**Ilustración 23. Modelo Físico de Datos en Enterprise Architect**

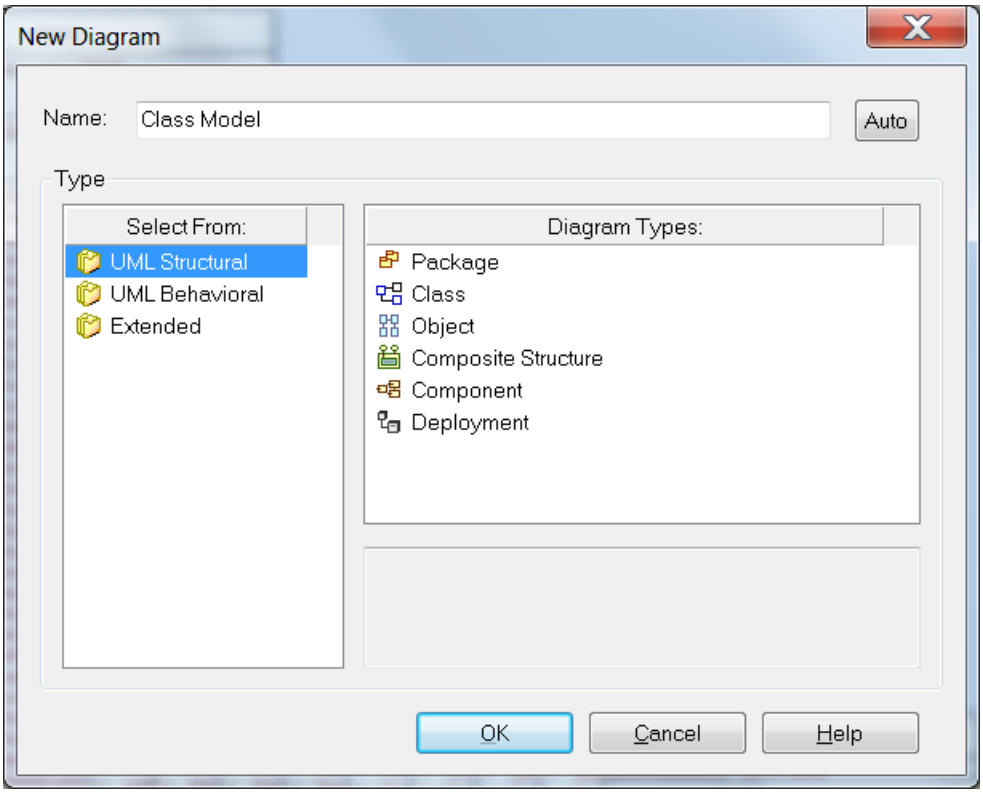



Ilustración 24. Diagramas que se pueden generar en Enterprise Architect

### 3.1.2.4. Visual Paradigm for UML

| Indicador | Descripción |
|---|---|
| **Usa un SGBD** | Sí, mediante ingeniería inversa, ERD (*Entity Relationship Diagram*) |
| **Licencia** | Privativo |
| **¿Filosofía de Arquitectura abierta?** | Si, UML |
| **Genera utilidades de software, procedimientos de lectura, biblioteca de fuentes y crea especificaciones** | Si |
| **Lenguaje de programación del código generado** | Si, dos lenguajes Java y .NET |
| **Tiene interfaz con otras herramientas** | Si mediante plugins en Eclipse y NetBeans IDE |
| **Dispone de utilidades gráficas, capaces de generar diagramas de proyecto y especificaciones del diccionario** | Si, 5 diagramas de UML con asistentes Wizard |
| **Provee capacidades de prototipado** | No |

| | |
|---|---|
| **¿Genera automáticamente un ámbito de aplicación de las especificaciones de diseño físico de las especificaciones del proyecto?** | Si |
| **Apoya el análisis y documenta el diseño** | Si |
| **¿Genera automáticamente los informes sobre las especificaciones del proyecto?** | Si |
| **¿Genera documentación para el usuario final?** | No |
| **¿Ofrece diferentes opciones de estilo o temas para el usuario final?** | No |
| **¿Maneja roles de usuarios y esquemas de seguridad para ellos?** | Si, en UML |
| **¿Posibilidad de exportar los reportes a otros formatos, o de enviar vía correo electrónico?** | No |
| **¿Permite formas de navegación en la aplicación resultante?** | No |
| **¿Ofrece ingeniería inversa?** | Si |

Tabla 9. Diagnóstico comparativo de Visual Paradigm for UML

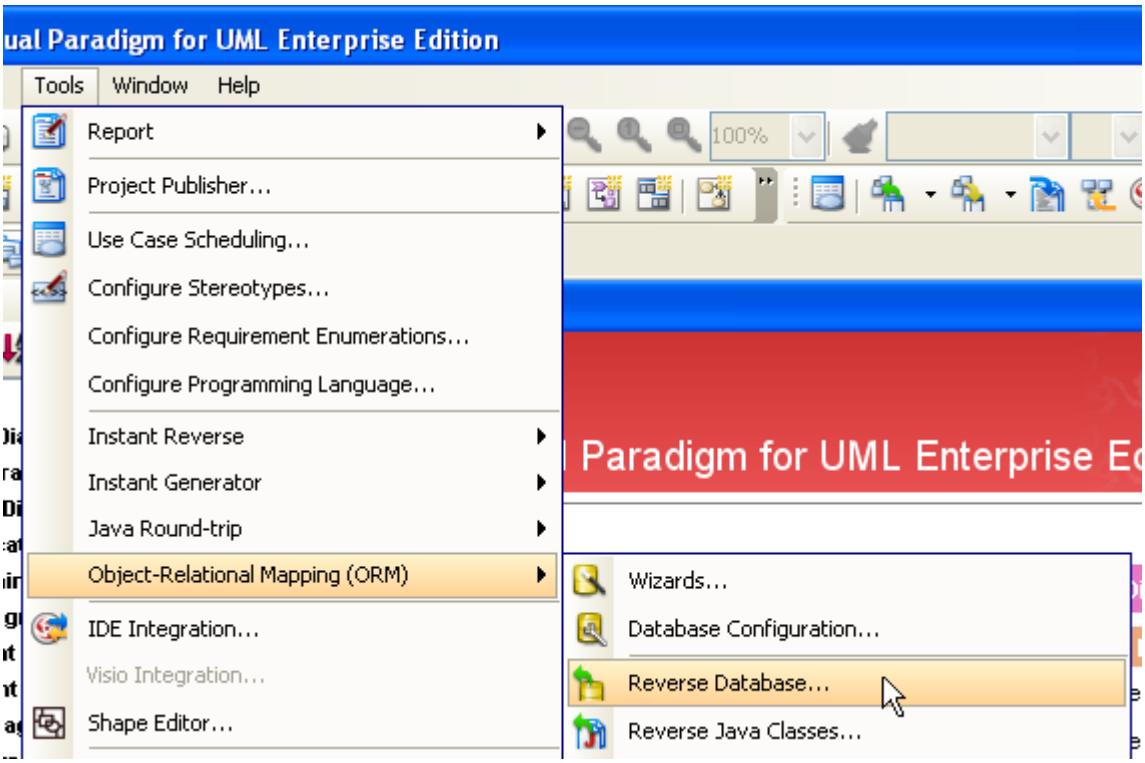


Ilustración 25. Ingeniería Inversa a la Base de Datos (Fuente VisualParadigm)[65]

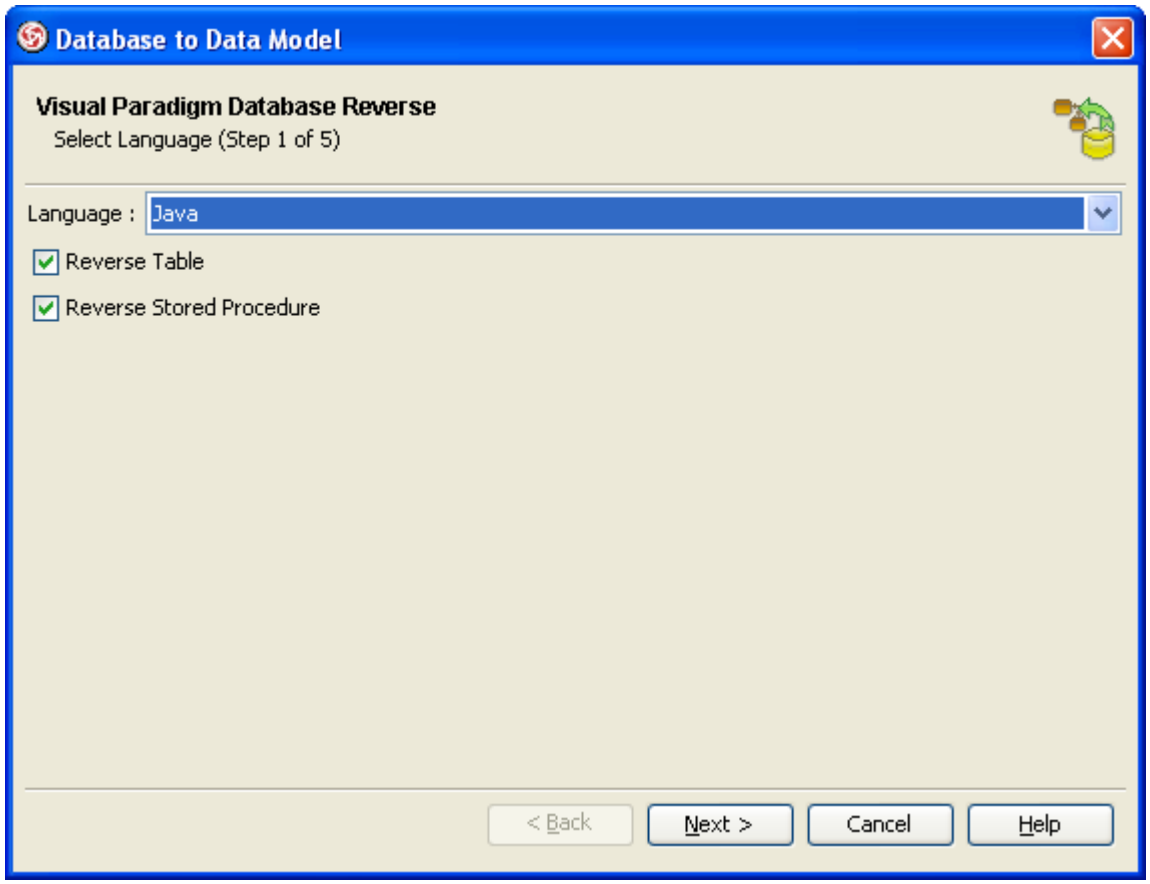



Ilustración 26. Lenguajes de programación soportados por Visual Paradigm

### 3.1.2.5. Oracle SQL Developer Data Modeler

| Indicador | Descripción |
|---|---|
| **Usa un SGBD** | Sí, al motor Oracle |
| **Licencia** | Privativo |
| **¿Filosofía de Arquitectura abierta?** | No |
| **Genera utilidades de software, procedimientos de lectura, biblioteca de fuentes y crea especificaciones** | No |
| **Lenguaje de programación del código generado** | SQL |

[65] VisualParadigm. VisualParadigm for UML. Recuperado de http://oldresources.visual-paradigm.com/vpsuite3.0sp1/reverse_db_enhancement.html Noviembre 2012

| | |
|---|---|
| **Tiene interfaz con otras herramientas** | Solamente con base de datos Oracle |
| **Dispone de utilidades gráficas, capaces de generar diagramas de proyecto y especificaciones del diccionario** | Si, diagramas BD |
| **Provee capacidades de prototipado** | No |
| **¿Genera automáticamente un ámbito de aplicación de las especificaciones de diseño físico de las especificaciones del proyecto?** | No |
| **Apoya el análisis y documenta el diseño** | No |
| **¿Genera automáticamente los informes sobre las especificaciones del proyecto?** | No |
| **¿Genera documentación para el usuario final?** | No |
| **¿Ofrece diferentes opciones de estilo o temas para el usuario final?** | No |
| **¿Maneja roles de usuarios y esquemas de seguridad para ellos?** | No |
| **¿Posibilidad de exportar los reportes a otros formatos, o de enviar vía correo electrónico?** | No |
| **¿Permite formas de navegación en la aplicación resultante?** | No |
| **¿Ofrece ingeniería inversa?** | No |

**Tabla 10. Diagnóstico Comparativo Oracle SQL Developer**

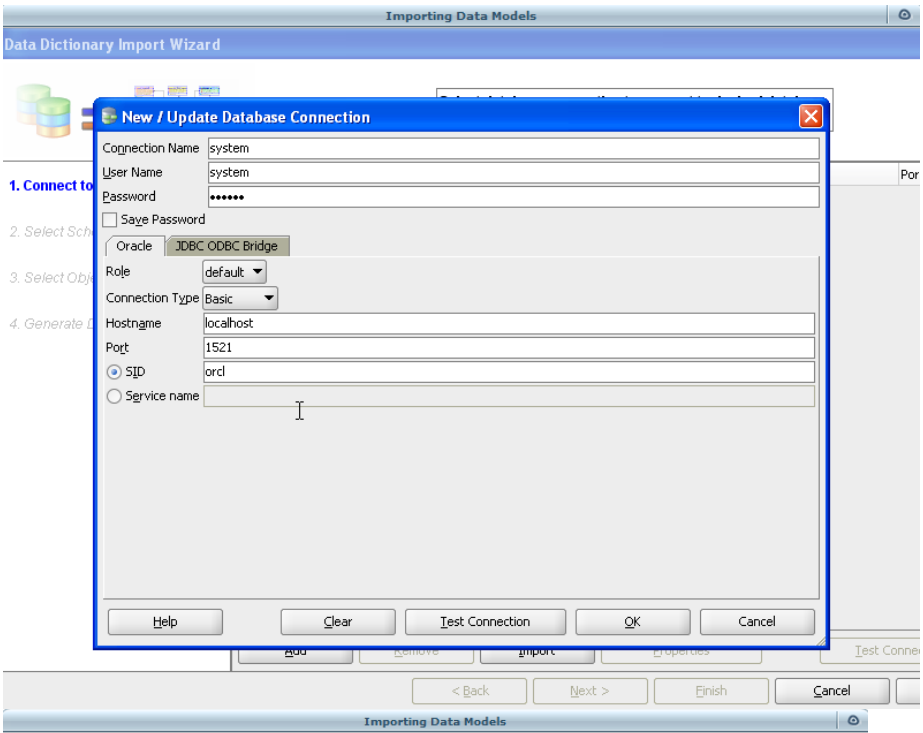



Ilustración 27. Importar Modelo de Datos en Oracle SQL Developer

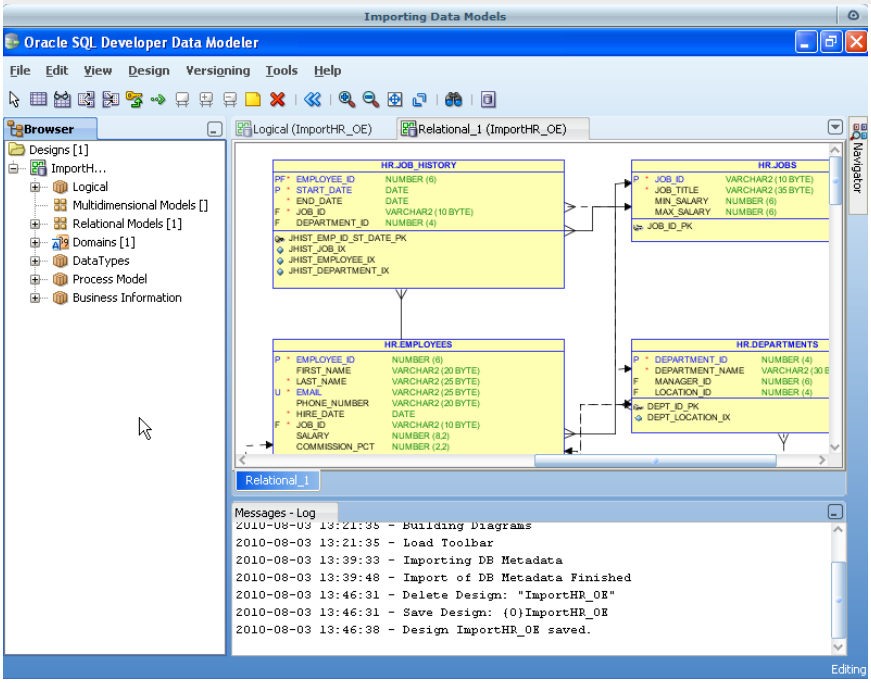



Ilustración 28. Diagramas Oracle SQL Developer

### 3.1.2.6. VisualWade

| Indicador | Descripción |
|---|---|
| **Usa un SGBD** | Sí, al motor Oracle y mySQL |
| **Licencia** | Libre |
| **¿Filosofía de Arquitectura abierta?** | Si, modelo vista controlador MVC |
| **Genera utilidades de software, procedimientos de lectura, biblioteca de fuentes y crea especificaciones** | No |
| **Lenguaje de programación del código generado** | Php |
| **Tiene interfaz con otras herramientas** | No |
| **Dispone de utilidades gráficas, capaces de generar diagramas de proyecto y especificaciones del diccionario** | No |
| **Provee capacidades de prototipado** | No |
| **¿Genera automáticamente un ámbito de aplicación de las especificaciones de diseño físico de las especificaciones del proyecto?** | No |
| **Apoya el análisis y documenta el diseño** | No |
| **¿Genera automáticamente los informes sobre las especificaciones del proyecto?** | No |
| **¿Genera documentación para el usuario final?** | No |
| **¿Ofrece diferentes opciones de estilo o temas para el usuario final?** | No |
| **¿Maneja roles de usuarios y esquemas de seguridad para ellos?** | No |
| **¿Posibilidad de exportar los reportes a otros formatos, o de enviar vía correo electrónico?** | No |
| **¿Permite formas de navegación en la aplicación resultante?** | No |
| **¿Ofrece ingeniería inversa?** | No |

**Tabla 11. Diagnóstico comparativo Visualwade**

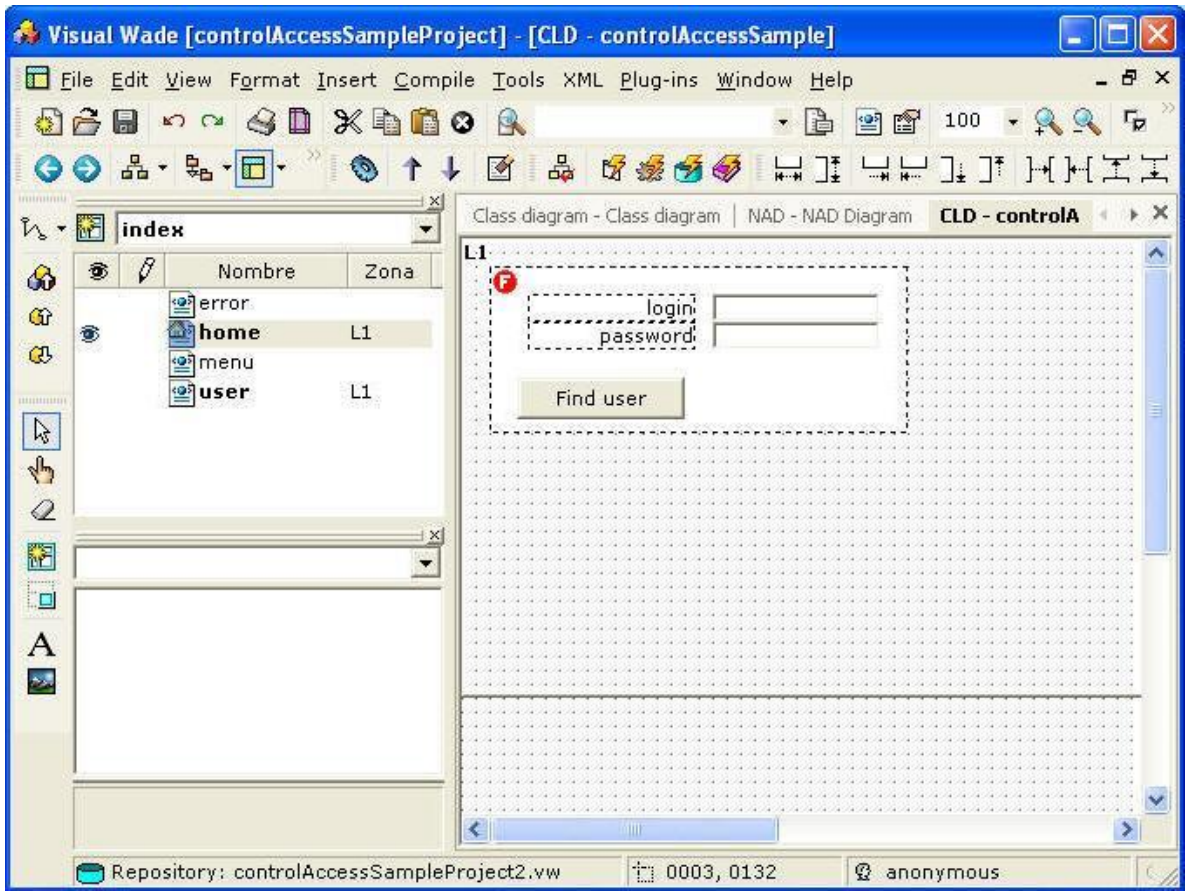



Ilustración 29. Diseño manual de una interfaz en VisualWade

### 3.1.2.7. Jboss Company Hibernate

| Indicador | Descripción |
|---|---|
| **Usa un SGBD** | Sí, 17 tipos de bases de datos |
| **Licencia** | Libre |
| **¿Filosofía de Arquitectura abierta?** | Si, mediante plugins en IDEs como Netbeans y Eclipse |
| **Genera utilidades de software, procedimientos de lectura, biblioteca de fuentes y crea especificaciones** | Si, mediante ORM (Object / Relational Modeling) |
| **Lenguaje de programación del código generado** | Java y .NET, modelo POJO |
| **Tiene interfaz con otras herramientas** | Si, como plugins con IDEs en JAVA y .NET |
| **Dispone de utilidades gráficas, capaces de generar diagramas de** | No |

| | |
|---|---|
| **proyecto y especificaciones del diccionario** | |
| **Provee capacidades de prototipado** | No |
| **¿Genera automáticamente un ámbito de aplicación de las especificaciones de diseño físico de las especificaciones del proyecto?** | No |
| **Apoya el análisis y documenta el diseño** | No |
| **¿Genera automáticamente los informes sobre las especificaciones del proyecto?** | No |
| **¿Genera documentación para el usuario final?** | No |
| **¿Ofrece diferentes opciones de estilo o temas para el usuario final?** | No |
| **¿Maneja roles de usuarios y esquemas de seguridad para ellos?** | No |
| **¿Posibilidad de exportar los reportes a otros formatos, o de enviar vía correo electrónico?** | No |
| **¿Permite formas de navegación en la aplicación resultante?** | Si, manualmente |
| **¿Ofrece ingeniería inversa?** | No |

Tabla 12. Diagnóstico Comparativo Hibernate

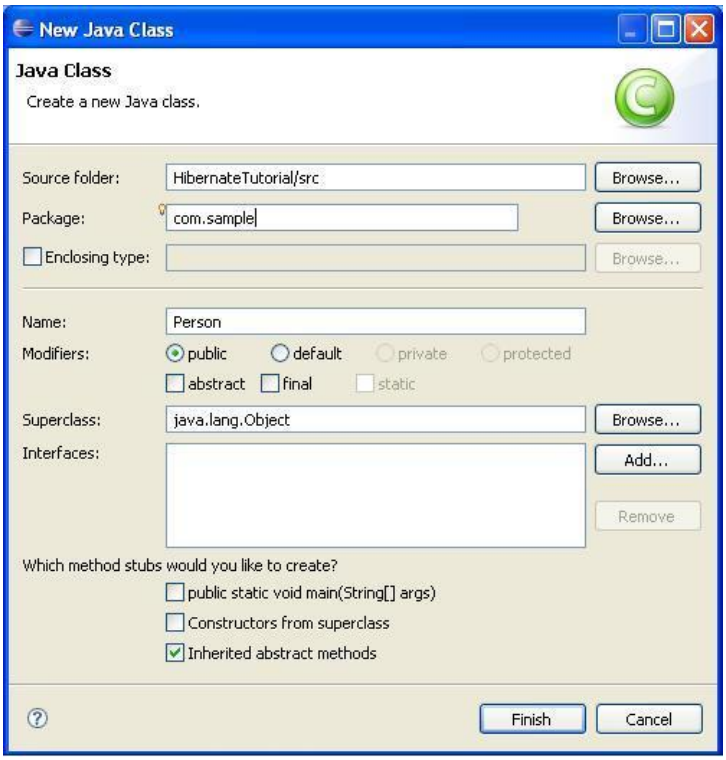


**Ilustración 30. Creación de una clase con Hibernate en Eclipse IDE**

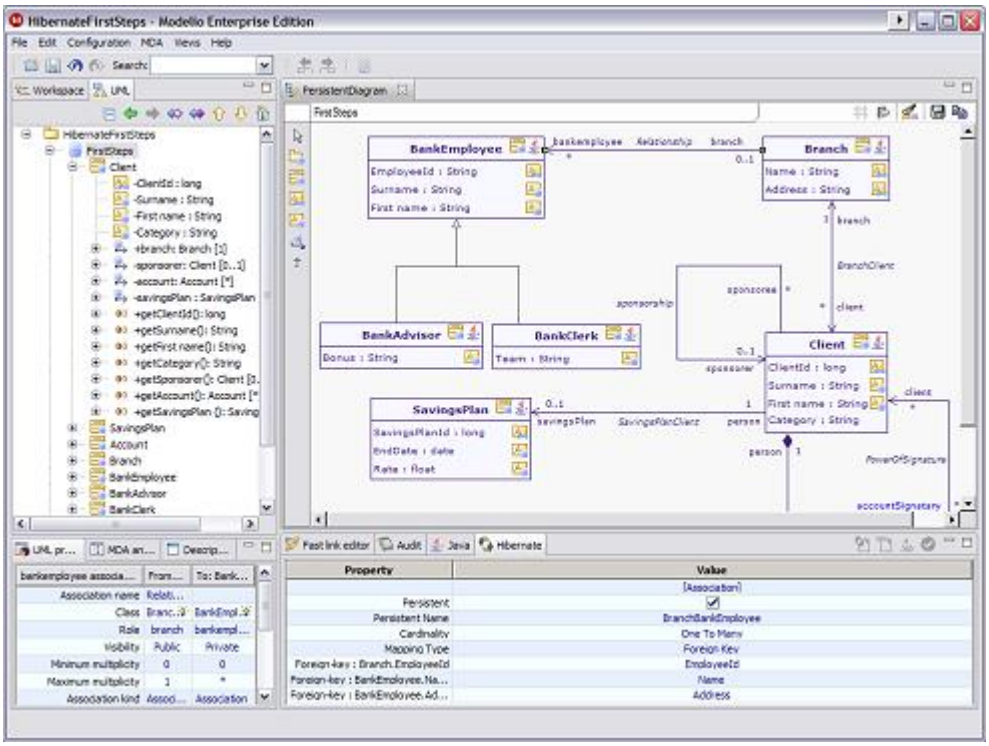



**Ilustración 31. Modelamiento con Hibernate**


**Ilustración 32. Interfaz generada con Hibernate**

#### 3.1.2.8. Resultados de la Comparativa de Herramientas

Aunque la totalidad de las herramientas analizadas anteriormente pueden generar el código de la nueva aplicación a partir de una base de datos existente, la mayoría de ellas requieren invocar manualmente un proceso de ingeniería inversa para la lectura del diccionario de datos del motor. Las herramientas que no requieren de invocar al proceso de ingeniería inversa y que se destacan por la funcionalidad de conexión directa, están a PHPMaker y a Hibernate. PHPMaker tiene una licencia privativa, mientras que Hibernate aunque maneja una licencia de código abierto, trabaja mediante *plugins* en ambientes integrados de desarrollo (IDE) como es el caso de NetBeans y Eclipse, restringe el desarrollo del software a ese tipo de IDE, y requiere de actividades manuales.

Por facilidad de uso y potencia al generar código, se destacan herramientas con licencia privativa, como es el caso de PHPMaker y PowerDesigner. PHPMaker se preocupa por brindar más opciones de diseño de interfaz al usuario, con un amplio sistema de navegación por menús, multiplicidad de opciones de conversión de formatos y envío, así como la seguridad de la información, mediante el perfilamiento de usuarios.

Con respecto a la documentación, son muy destacables las posibilidades que brinda PHPMaker con respecto a las demás herramientas analizadas. Brinda información dentro del código, como documentación externa de acuerdo con estándares de la ANSI.

PHPMaker se muestra como la herramienta más completa y poderosa para generar código de forma automática a partir de una base de datos, lo puede hacer en diferentes lenguajes de programación, pero la herramienta es diferente, aunque es de la misma casa, por ejemplo, a parte de PHPMaker, existen ASPMaker, JSPMaker y CFMMaker, por citar algunos ejemplos. Pero la herramienta no tiene licencia apropiada para la comunidad de software libre.

### 3.1.3. Crear una Herramienta de Automatización para la Generación de Código de Interfaces CRUD, a partir de una Base de Datos, en Ambientes de Desarrollo de Software Libre

#### 3.1.3.1. SCRUM

Para permitir el desarrollo ágil de la aplicación se hizo uso de la metodología Scrum, cuyas actividades fueron seguidas mediante la herramienta Trello[66], el tablero principal (board - dashboard) se denominó con el nombre del proyecto “FredyCRUD”, este tablero se divide verticalmente en tres columnas, la primera “To Do”, que contiene las actividades “por hacer”, la columna intermedia “Doing” indica las actividades que se están haciendo a la fecha, y la tercer columna “Done”, contiene las actividades finalizadas (hechas). Y

[66] Trello. Trello. Recuperado de http://trello.com Junio 2012.

horizontalmente se encuentran los “Sprints” o hitos del proyecto (determinados por una pestaña de color verde). El primer sprint se definió de acuerdo al primer objetivo del proyecto, es decir identificar los requerimientos que debe reunir una herramienta CASE de este tipo. El segundo sprint tiene que ver con el segundo objetivo del proyecto, es decir la generación de un diagnóstico comparativo de herramientas CASE actuales que sean similares a la herramienta desarrollada en este proyecto. El tercer objetivo, por su amplitud y exigencia, requiere de varios sprints que están determinados por cada una de las etapas de desarrollo de una herramienta CASE y por cada una de las fases del ciclo de vida de Ingeniería de Software para el desarrollo de la aplicación.

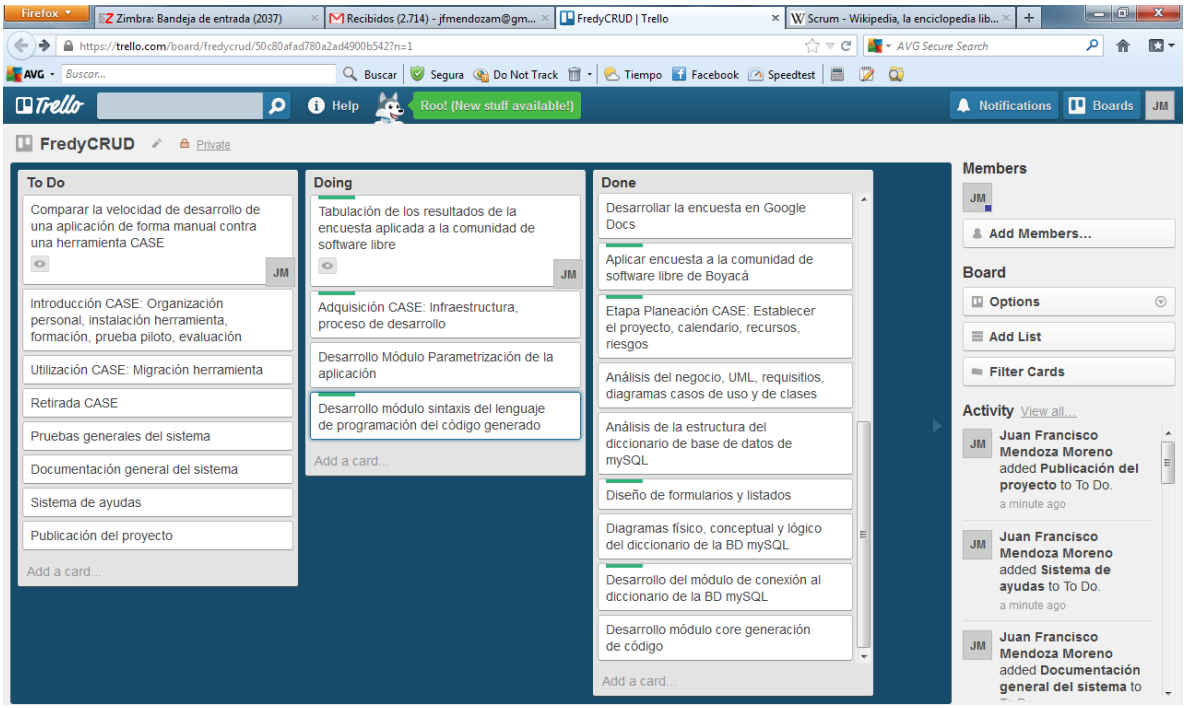



**Ilustración 33. Proyecto FredyCRUD bajo metodología SCRUM en plataforma Trello**

El autor del proyecto, asumió de forma paralela los roles de ProductOwner (interlocutor del cliente), ScrumMaster (facilitador), Desarrollador y en forma conjunta el rol de Manager. Los roles de stakeholders (clientes) fueron asumidos por algunos miembros de la comunidad de software libre de Boyacá, compartiendo el rol de Manager. Aunque suene paradójico, se realizaban reuniones por cada sprint (Daily Scrum), que como los todos roles eran asumidos por el autor del proyecto, estas reuniones se convertían en un punto de interrupción (breakpoint), para evaluar de forma muy rápida el avance en el cumplimiento de las actividades. Algunas actividades presentan retrasos significativos, debido especialmente, por la falta de personal para asumir cada rol y era necesario realizar un punto de interrupción de varios sprints (Scrum de Scrum). Entre cada sprint era necesario planear el siguiente sprint (Spring Planning Meeting) donde se replanteaban estrategias para poder llevar a cabo ese sprint. En el cumplimiento de un sprint se aplicaba el Sprint Review Meeting, para revisar el trabajo realizado y para aprovechar mostrar el avance del proyecto a los skateholders.

En la segunda fase del desarrollo de una herramienta CASE, la Adquisición, existe una gran actividad que consiste en el desarrollo de la aplicación, donde se aplican todas las etapas del ciclo de vida del desarrollo de software. La documentación de la Ingeniería de Software se encuentra como documento anexo a este trabajo (Ver Anexo 3. Documento Técnico Ingeniería de Software Proyecto FredyCRUD).

### 3.1.3.2. Fases del Desarrollo de Herramientas CASE

Son cinco las fases que se tienen en cuenta a la hora de desarrollar una herramienta CASE.

La primera de ellas es la Planificación, donde se establecieron los estándares y procedimientos necesarios para la consecución del proyecto. También en esta fase se determinó el calendario del proyecto (Ver Ilustración 7. Diagrama de Gantt Cronograma Proyecto), el personal requerido, que de acuerdo con SCRUM, se pudieron identificar los siguientes roles:

- Product Owner: Conocido como el cliente del proyecto, en este caso fueron identificados miembros de la comunidad de software libre de Boyacá, quienes son los interesados en usar el software resultado del proyecto, por lo tanto fueron tenidas en cuenta sus sugerencias al respecto.
- Scrum Master: Es el facilitador preocupado para que todos los miembros del equipo SCRUM sigan las reglas y den continuidad al proceso SCRUM. Es la persona que asegura que la lista de requisitos se cumpla de forma priorizada, además él organiza las reuniones de iteración, de sincronización, de demostración y de retrospectiva. Por último, está atento a eliminar los impedimentos para que el trabajo se pueda realizar y protege su equipo de interrupciones externas.
- Team: Es el equipo de trabajo, cuyo tamaño se sugiere entre 5 y 9 personas, pero para el caso del proyecto en realidad fue realizado por tan solo una persona que cumplía varios perfiles. Es necesaria la auto-organización para seleccionar y cumplir con los requisitos por cada iteración, identificando las tareas necesarias y estimando el esfuerzo. Por último, demuestra al cliente el cumplimiento de los requisitos por cada iteración.

Además, en esta primera fase se identificó el costo del proyecto (Ver Tabla 4. Presupuesto del proyecto), el sistema que se desarrollará (en este caso, la herramienta generadora de software para interfaces CRUD) y el análisis del riesgo (Ver Tabla 14. Tabla de Ingeniería de Requerimientos).

En la segunda fase, la de adquisición, se busca la herramienta CASE que satisfaga las necesidades estipuladas, en caso contrario, como en este proyecto, se crea la herramienta. Es necesario determinar la infraestructura (Ilustración 63. Diagrama de Infraestructura Tecnológica), el proceso de desarrollo, que en este caso es el ciclo de vida del software (Ampliado en el siguiente numeral), las técnicas y metodologías de desarrollo de software.

Una vez desarrollada la herramienta, viene la tercera fase de Introducción, donde es importante la organización del personal, la instalación y adaptación de la herramienta, la formación y las pruebas y

evaluación. Para este caso, se utilizó la estrategia de introducción propia de la comunidad de software libre, en este caso, la publicación de la herramienta en un repositorio de software libre que brinde los servicios necesarios para dar a conocer la aplicación y poder asistir al usuario de la misma.

Aunque no está desarrollada en este proyecto, la cuarta fase la de Utilización, es compromiso tácito dentro de la comunidad de software libre la asistencia al usuario y las futuras migraciones o actualizaciones de las aplicaciones, no solo por parte del autor del software, sino por parte de la comunidad entera.

Otra fase no desarrollada en este proyecto, la quinta fase la de Retirada, es una fase inevitable en cualquier ciclo de vida del desarrollo del software, cuando este quede obsoleto ya sea por su atraso tecnológico o porque quedó relegado del uso e interés por parte del usuario.

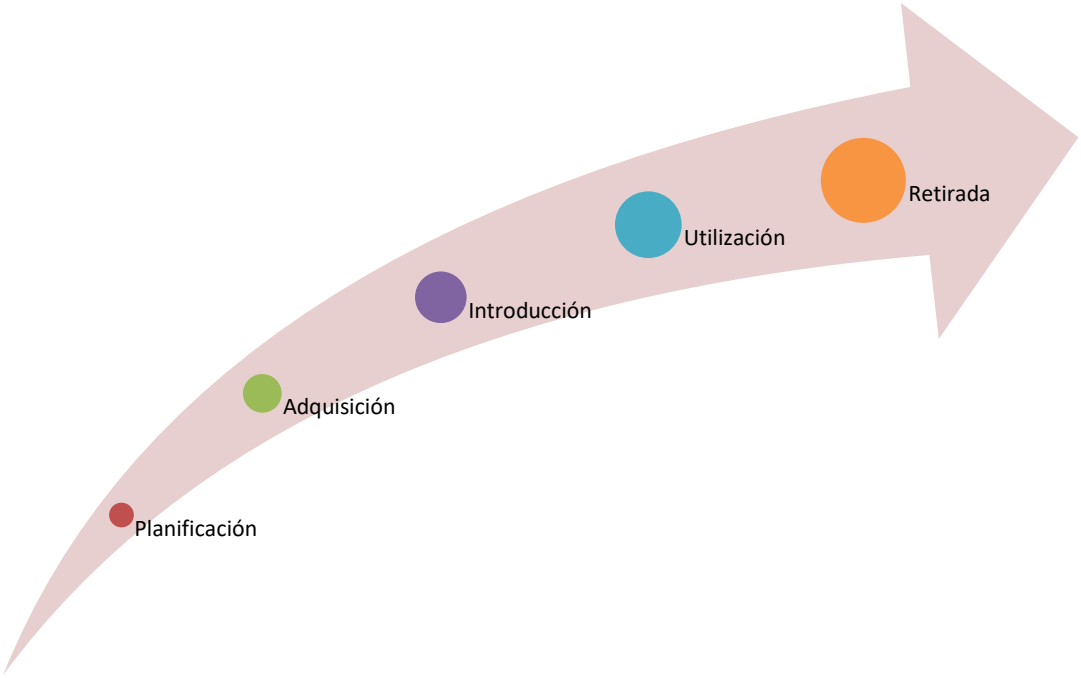



**Ilustración 34. Fases de Desarrollo Herramienta CASE**

### 3.1.3.3. Ciclo de Vida del Software

Comprendido dentro de la segunda fase de desarrollo de herramientas CASE, concebido por la ISO[67] como "El Marco de Referencia" que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso".

[67] ISO. ISO 12207-1. Ciclo de Vida del Software

Dentro de los procesos del ciclo de vida del software, se pueden mencionar tres grandes grupos:

- Procesos principales: Comprenden la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento
- Procesos de soporte: Comprenden la documentación, la gestión de la configuración, el aseguramiento de la calidad, la verificación, la validación, la revisión en equipo, la auditoría y la resolución de problemas
- Procesos de la organización: Comprenden la gestión, la mejora, la infraestructura y la formación

El modelo a seguir este ciclo de vida fue el iterativo incremental, donde al final de cada ciclo se entrega una versión del software mejorada. Estos ciclos se van repitiendo hasta obtener un producto satisfactorio. La evaluación del producto por parte del usuario se da en cada iteración. El desarrollo de software en este proyecto es un proceso empírico, no definido, por lo tanto, se necesita de mayor control en la aplicación. Cada iteración se conoce como *Sprint* y dentro de cada *Sprint* existen reuniones diarias muy rápidas (*Scrum*). Esta metodología fue escogida por cuanto los requisitos son cambiantes y emergentes, el equipo de trabajo se auto-organiza y el proceso debe ser ágil y escalable.

En las fases iniciales del proyecto para facilitar la comprensión, visibilidad y documentación, se hizo necesario el uso de software de apoyo para las fases de análisis y diseño principalmente, como herramientas UML y herramientas como *PowerDesigner*, que es una solución gráfica de modelamiento empresarial que soporta metodologías y notaciones estándar. El modelamiento puede ser apreciado en detalle en el **Anexo 3. Documento Técnico Ingeniería de Software Proyecto FredyCRUD**.

#### 3.1.3.3.1. Análisis de Requisitos y Diseño de Software

El trabajo inició con el **modelamiento del Proceso del Negocio (BPMN)**, luego el modelo de requisitos, el modelo orientado a objetos, el modelo XML, el modelo de movimiento de datos y los modelos físicos, conceptuales y lógicos, soportado con la herramienta PowerDesigner.

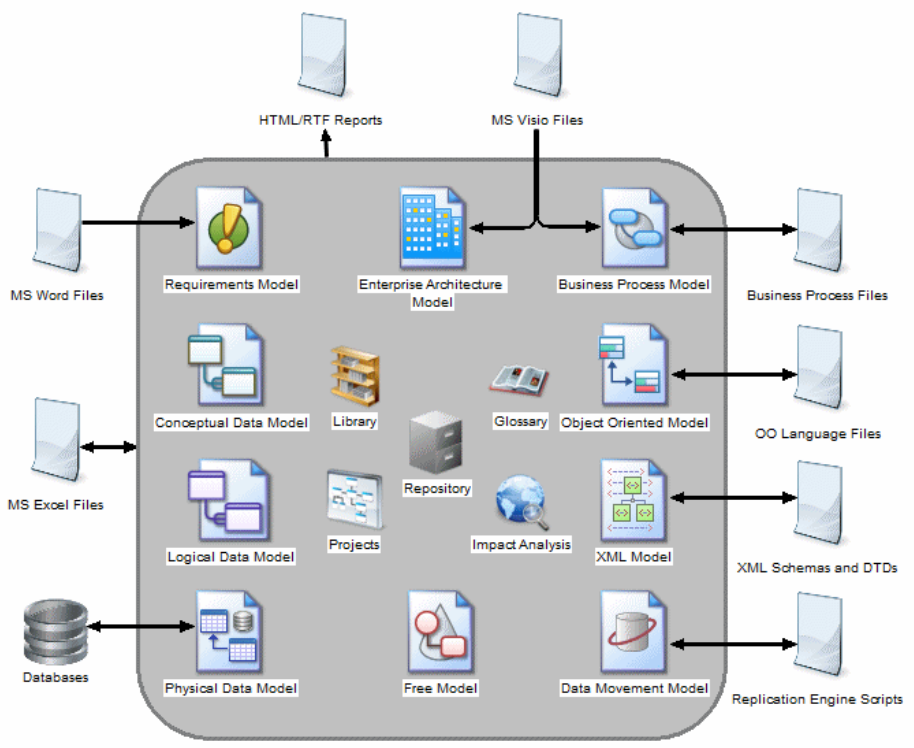



**Ilustración 35. Modelamientos soportados por PowerDesigner (Fuente Ayuda en línea PowerDesigner)**

El modelamiento para el proyecto se puede ver resumido en la Ilustración 36. Modelamiento general del proyecto en su fase de análisis y diseño.

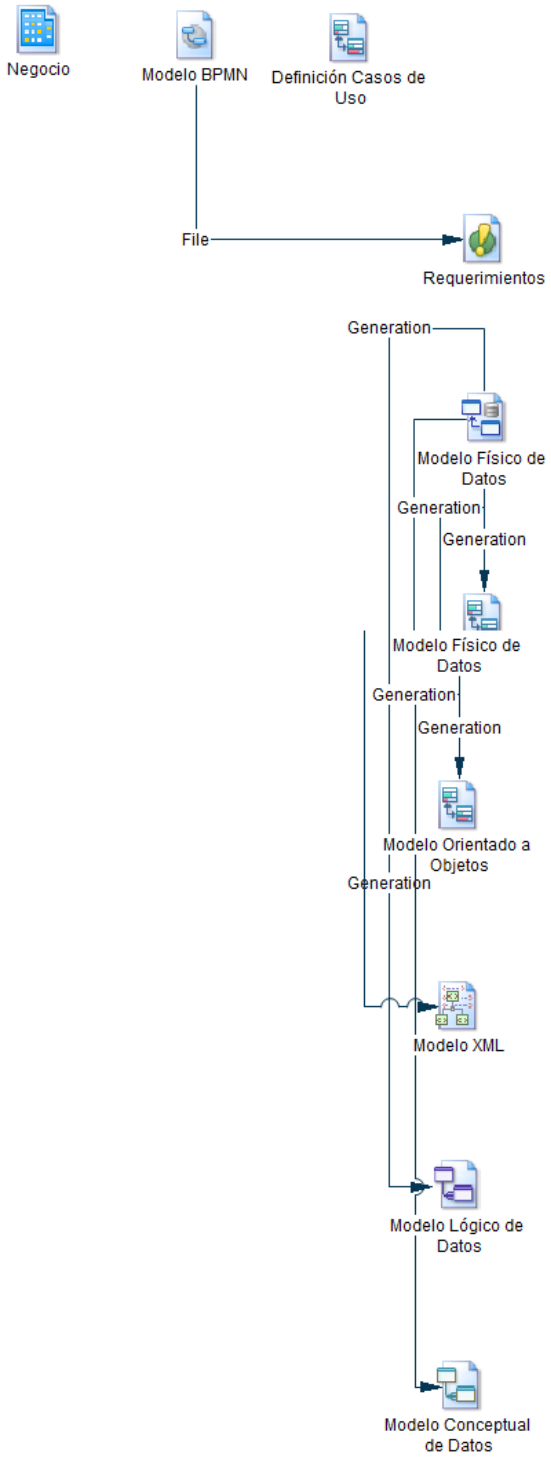



**Ilustración 36. Modelamiento general del proyecto en su fase de análisis y diseño**

En el BPM se contemplan las fases hito del proyecto y su interacción con cada uno de los perfiles de personas involucradas en el proyecto. Posteriormente, estas fases se jerarquizan en el Diagrama Jerárquico del Proceso, para que finalmente se determinen los servicios del proceso (Ver Ilustración 56. Diagrama de Procesos del

Negocio, Ilustración 55. Diagrama Jerárquico de Procesos FredyCRUD e Ilustración 57. Análisis de Servicios de Proceso).

Con el BPM se puede identificar, describir y analizar los procesos del negocio. El sistema se puede analizar en varios niveles de detalle y enfocarse alternativamente en el flujo de control (la secuencia de ejecución) o el flujo de datos (el intercambio de datos). Para el Modelo de Procesos del Negocio se puede utilizar lenguajes como BPMN (como es el caso de este proyecto), BPEL, entre otros.

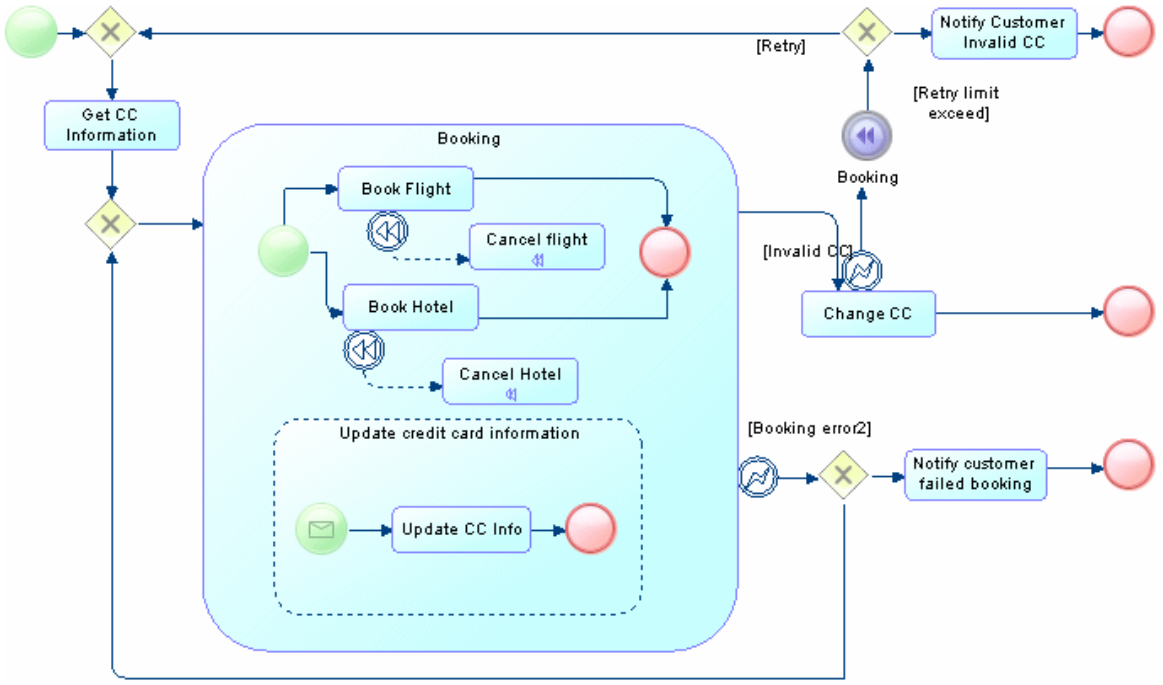



**Ilustración 37. Modelo de Procesos del Negocio (Fuente PowerDesigner OnLine Help)**

Las etapas “hito” del proyecto son:

- Crear nueva aplicación. Donde se invoca el software para la generación del nuevo sistema de información
- Parametrizar aplicación. Donde se configura y personaliza la nueva aplicación
- Conexión DB. Se establece la comunicación con el motor de la Base de Datos (mySQL)
- Generación de la Aplicación. Se obtiene el código fuente de la nueva aplicación
- Modificación de la aplicación. Posibilidad para que el desarrollador modifique y empodere el código fuente de la nueva aplicación generada
- Uso de la aplicación. Utilización de los *runtimes* de la nueva aplicación generada.

Con respecto al modelamiento para conocer el **Negocio**, se desarrollaron varios modelos:

- Organigrama (Ver Ilustración 60. Organigrama de un esquema de adquisición de un nuevo Sistema de Información), con doce perfiles de integrantes identificados
- El Diagrama de Comunicación del Negocio (Ver Ilustración 61. Diagrama de Comunicación del Negocio), donde se analizan las interacciones de los perfiles identificados con las etapas del negocio.
- Diagrama de Planeación de la ciudad (Ver Ilustración 63. Diagrama de Planeación de la Ciudad). Se establecen tres grandes áreas: Fuente de datos, Aplicación y Relación con el cliente
- Diagrama orientado al servicio (Ver Ilustración 65. Diagrama Orientado al Servicio), con cuatro capas: Negocio, Servicio, Aplicación y Tecnología
- Diagrama arquitectura de la aplicación (Ver Ilustración 64. Diagrama Arquitectura de la Aplicación). Tiene en cuenta la arquitectura de tres componentes: Sistema de Información, el Servidor de Aplicaciones y el Servidor de la Base de Datos.
- Diagrama de Infraestructura Tecnológica (Ver Ilustración 66. Diagrama de Infraestructura Tecnológica), analiza los dos extremos (*sides*) el del cliente y el del servidor
- Mapa de procesos (Ver Ilustración 62. Mapa de Procesos). Agrupados en dos macro-procesos: los de administración y los de operación. Son nueve procesos en total.

Un **modelo de requerimientos (RQM)** permite analizar cualquier clase de requerimientos por escrito y asociarlos con los usuarios y grupos quienes los implementarán y con los objetos de diseño de otros modelos. Se puede usar los RQM para representar cualquier documento estructurado (por ejemplo, las especificaciones funcionales, el plan de prueba, los objetivos del negocio, entre otros) y se pueden importar o exportar las jerarquías de los requerimientos (Ver Ilustración 55. Análisis de Requisitos proyecto FredyCRUD).

Las acciones contempladas en el RQM de este proyecto son:

- Descripción del proyecto
- Descripción de los escenarios, donde se identificaron cuatro escenarios
- Requerimientos funcionales, agrupados en cinco grandes requerimientos, de acuerdo con las etapas "hito" del proyecto. Para un total de siete grandes requerimientos funcionales
- Requerimientos no funcionales, donde se identificaron tres requisitos principales
- Plataformas de desarrollo
- Análisis del Riesgo, donde se tienen en cuenta tres riesgos principalmente

Cada requerimiento tiene su identificador numérico, su descripción completa, su código interno, el nivel de prioridad, la carga de trabajo estimada (*Workload*), el nivel de riesgo y el estado actual del requerimiento.

El **Modelamiento Orientado a Objetos** permite analizar un sistema de información a través de casos de uso, análisis estructurales y de comportamiento, y en términos de desarrollo, se usa el Lenguaje Unificado de Modelamiento (UML). Se puede modelar, aplicar ingeniería inversa y generar código en distintos lenguajes de programación. Se desarrollaron los siguientes diagramas UML:

- Diagramas de Casos de uso (Ilustración 67. Diagrama General de Casos de Uso). Se pueden apreciar los actores del sistema interactuando con los módulos identificados en el modelamiento de negocios. Este diagrama provee una vista gráfica de los requerimientos del sistema y permite identificar cómo

los usuarios interactúan con éste. De inmediato se obtiene una panorámica de la funcionalidad del sistema.

- Diagramas Estructurales:
    - Diagrama de Clases (Ver Ilustración 68. Diagrama general de Clases para el diccionario de datos mySQL). Permite apreciar las clases, las interfaces y los paquetes que componen el sistema y la relación entre ellos.
    - Diagramas de Estructura Compuesta. Permite ver de forma gráfica las clases, interfaces y paquetes que componen el sistema, incluyendo los puertos y las partes que describen las estructuras internas.
    - Diagrama de objetos. Permite apreciar la estructura del sistema a través de instancias en concreto de las clases (objetos), sus asociaciones (enlaces de instancia) y dependencias.
    - Diagrama de paquetes. Se aprecia la organización de la aplicación y permite identificar la generalización y enlaces de dependencia entre los paquetes. Para este proyecto, se determinó administrar un solo paquete denominado “FredyCRUD”.
- Diagramas Dinámicos:
    - Diagrama de Comunicación. Permite conocer las interacciones entre los objetos para un escenario de caso de uso, la ejecución de una operación, o una interacción entre clases, con énfasis en la estructura del sistema.
    - Diagrama de Secuencias (Ver Ilustración 73. Diagrama General de Secuencias). Vista de la cronología de intercambio de mensajes entre objetos y actores para un caso de uso, la ejecución de una operación, o una interacción entre clases. En este diagrama se puede apreciar cronológicamente la interacción entre actores y operaciones del sistema, con sus respectivos mensajes de ida y vuelta. A su vez se analizan de forma detallada tres modelos de secuencia para cada uno de los casos de uso relevantes para el desarrollo del sistema: Parametrización de la aplicación, Selección de la BD y Generación de la nueva aplicación.
    - Diagrama de Actividades (Ver Ilustración 77. Diagrama de Actividades). Muestra el comportamiento del sistema y permite descomponerlo funcionalmente para analizar cómo será implementado. De este diagrama se deduce que la implementación de este sistema sucede en tres grandes módulos: Parametrización de la aplicación, Conexión a la Base de Datos y Generación de la Nueva aplicación.
    - Diagrama de Estados (Ver Ilustración 78. Diagrama de Estados). Permite conocer la Máquina de Estados, el comportamiento público de un clasificador como componente o clase, en la forma de cambios en el momento del estado o en los eventos que permiten la transición de un estado a otro. El diagrama permite identificar cinco estados: Apertura de la Base de Datos, Conexión a la Base de Datos, Configuración de la Aplicación, Generación de la Aplicación y Aplicación Generada.
    - Diagrama de Interacción (Ver Ilustración 79. Diagrama de Interacción). Permite apreciar el flujo de control del sistema que se puede descomponer en otros diagramas de interacción. De este diagrama se determinan tres grandes diagramas de secuencia: Selección de Base de Datos, Generar la Aplicación y Parametrizar aplicación.

- Diagramas de Implementación:

- Diagrama de Componentes: Permite observar las dependencias y generalizaciones entre los componentes de software, incluyendo los componentes de código fuente, los componentes de código binario y los componentes ejecutables.
- Diagrama de Despliegue: Permite observar la configuración física de los elementos al momento de ejecución del sistema.

Con respecto al **Modelamiento de Datos**, se diseñaron tres modelos:

- Modelo Físico de Datos (Ver Ilustración 80. Diagrama Físico del Diccionario de Datos de mySQL) o PDM. Permite analizar las tablas, vistas y otros objetos en la base de datos, incluyendo objetos multidimensionales necesarios para el *datawarehousing* de los datos. El PDM es más concreto que el modelo de datos del Diagrama Conceptual o del Diagrama Lógico. Concretamente este proyecto analiza el modelo físico de datos del metamodelo del diccionario de datos de mySQL. Para obtener este modelo, se realizó ingeniería inversa sobre el diccionario de datos de mySQL, mediante PowerDesigner. Son 37 tablas con sus respectivas llaves primarias y foráneas y con los demás campos.
- Modelo de Datos Conceptual y Lógico (Ver Ilustración 82. Modelo Conceptual del diccionario de datos de mySQL e Ilustración 81. Modelo Lógico del Diccionario de Datos de mySQL). El modelo conceptual permite analizar la estructura conceptual de un sistema de información, para identificar las entidades principales que serán representadas y las relaciones entre ellas. Un modelo conceptual es más abstracto que el modelo lógico, o que el modelo físico. El modelo lógico permite analizar la estructura de un sistema de información, independiente de cualquier implementación de una base de datos en específico.

Los principales diagramas aplicados en las dos primeras fases del ciclo de vida del software se pueden ver resumidos en la Ilustración 38. Correspondencia de los diagramas UML con fases de un proyecto.

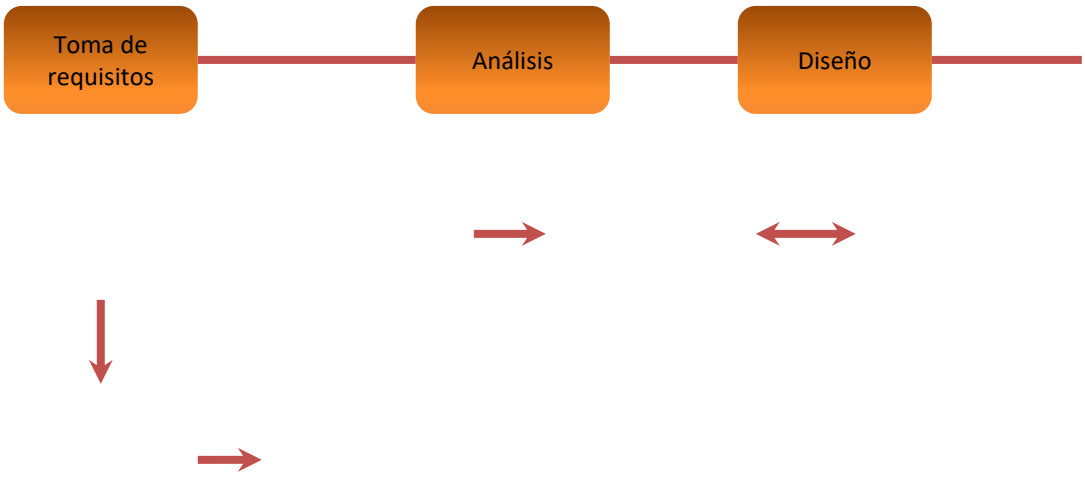



Ilustración 38. Correspondencia de los diagramas UML con fases de un proyecto[68]

### 3.1.3.3.2. Construcción de Software (Codificación)

[68] AYCART PÉREZ, David et al. Ingeniería del Software en entornos de SL. UOC. Catalunya 2007

**El Lenguaje de programación** escogido para la codificación fue Java, por cuanto se ha constituido como la base de cualquier aplicación en red y es un estándar mundial de desarrollo, desde aplicaciones, pasando por juegos, hasta aplicaciones empresariales. Tiene más de 9 millones de desarrolladores en el mundo. Su ecosistema es maduro y su rendimiento muy sólido y ofrece portabilidad en diversos entornos[69].

De Java se destacan los siguientes beneficios:

- Multiplataforma
- Entorno de programación rápido
- Soporte de multihilos
- Gestión de memoria y de excepciones
- Tiene el respaldo del Java Community Process
- Es muy popular, con poderosas herramientas, libros, bibliotecas y ejemplos de código
- Diferentes versiones para cada una de las necesidades: escritorio, móviles, dispositivos, servidores

Java tiene como funciones a:

- Independencia de la plataforma
- Alto rendimiento
- Fácil de aprender
- Basado en estándares
- Predominio internacional
- Entornos de tiempo de ejecución constantes
- Optimizado para lo incrustado
- Modelo de seguridad

[69] Oracle. Java. http://www.oracle.com/lad/technologies/java/overview/index.html Recuperado el 12 de Diciembre de 2012

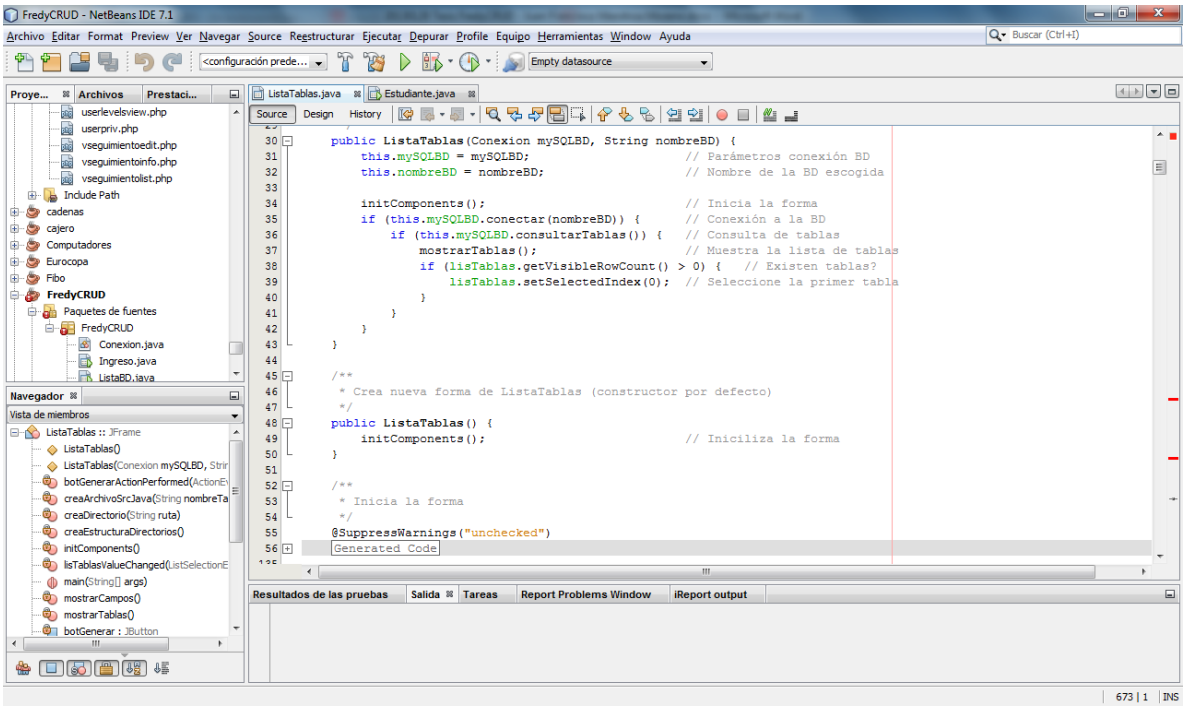



**Ilustración 39. Programación del Módulo Listar Tablas del proyecto FredyCRUD en lenguaje JAVA**

Por lo tanto, el **paradigma de programación** es la orientada a objetos, donde impera que los programas deben ser diseñados, precisamente la disciplina de la Ingeniería de Software se preocupa por la construcción de programas correctos, que trabajen y que estén bien escritos. El ingeniero de software intenta utilizar métodos aceptados y suministrados para analizar el problema a ser resuelto y para diseñar el programa que resolverá el problema. El concepto central radica en el objeto, que es una clase de módulo que contiene datos y subrutinas[70]. En la programación orientada a objetos, ese objeto es una clase de entidad autosuficiente que tiene un estado interno (datos que contiene) y que puede responder a mensajes (llamadas a sus subrutinas). La programación orientada a objetos aprovecha la ingeniería de software al inicio con la identificación de objetos involucrados en un problema y los mensajes que esos objetos deberían responder. El programa resultante es una colección de objetos cada uno con sus propios datos y su propio conjunto de responsabilidades. Los objetos que tienen los mismos tipos de datos y responden a los mismos mensajes de la misma forma pertenecen a la misma clase, aunque pueden existir objetos similares de clases diferentes. En resumen la programación orientada a objetos puede ser una herramienta superior de desarrollo de programas y una solución parcial para el problema del reuso de software.

[70] ECK, David J. et al. Introducción a la Programación utilizando Java. Nueva York, 2012

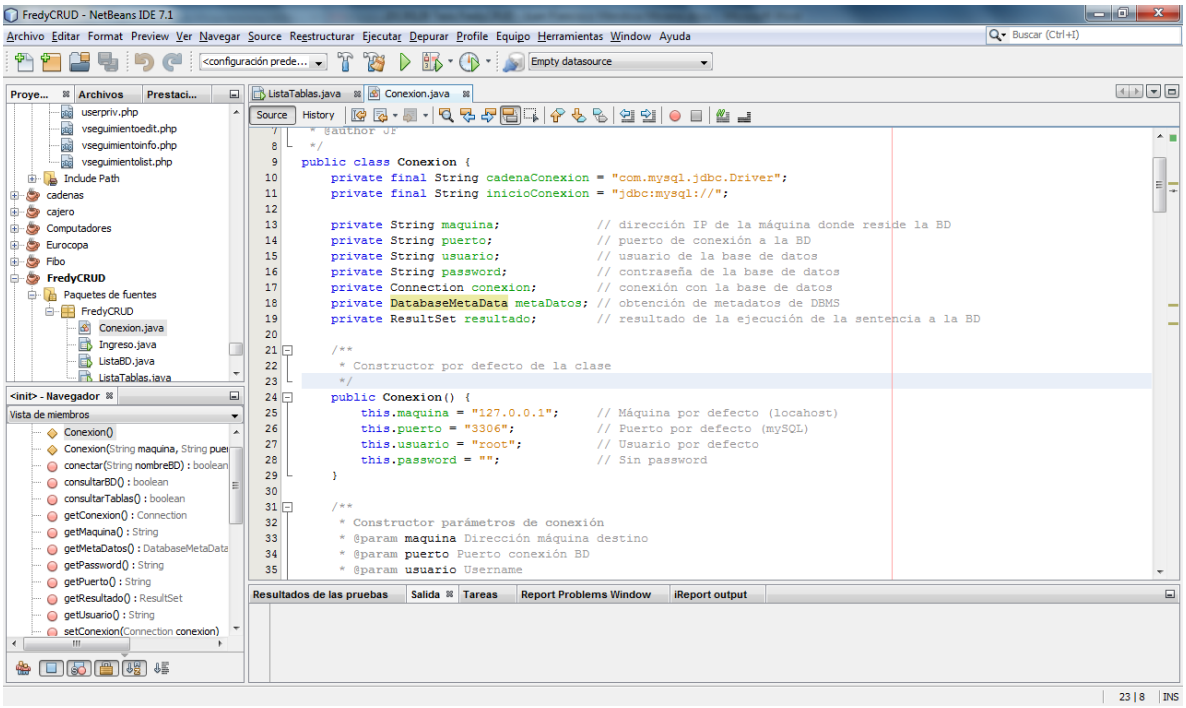



**Ilustración 40. Definición de la clase Conexión del proyecto FredyCRUD como paradigma de POO**

Como **entorno de desarrollo** (IDE) fue escogido a NetBeans, producto de Oracle y con más de 100 socios, es un entorno de desarrollo integrado libre, hecho principalmente para lenguaje Java, aunque soporta otros lenguajes de programación. Es un entorno de desarrollo -una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas-. Está escrito en Java. Existe además un número importante de módulos para extender el NetBeans IDE[71].

Entre sus características cabe destacar:

- Desarrollo rápido mediante una interface al usuario
- Soporte a las tecnologías de JAVA
- Independiente de la plataforma
- Soporta múltiples lenguajes de programación
- Conjunto enriquecido de complementos suministrado por la comunidad
- Soporte a HTML 5
- Edición de código rápido e inteligente
- Administración de proyectos fácil e independiente
- Código libre de errores de escritura

[71] Oracle. NetBeans. Recuperado de http://netbeans.org/index_es.html el 12 de Diciembre de 2012

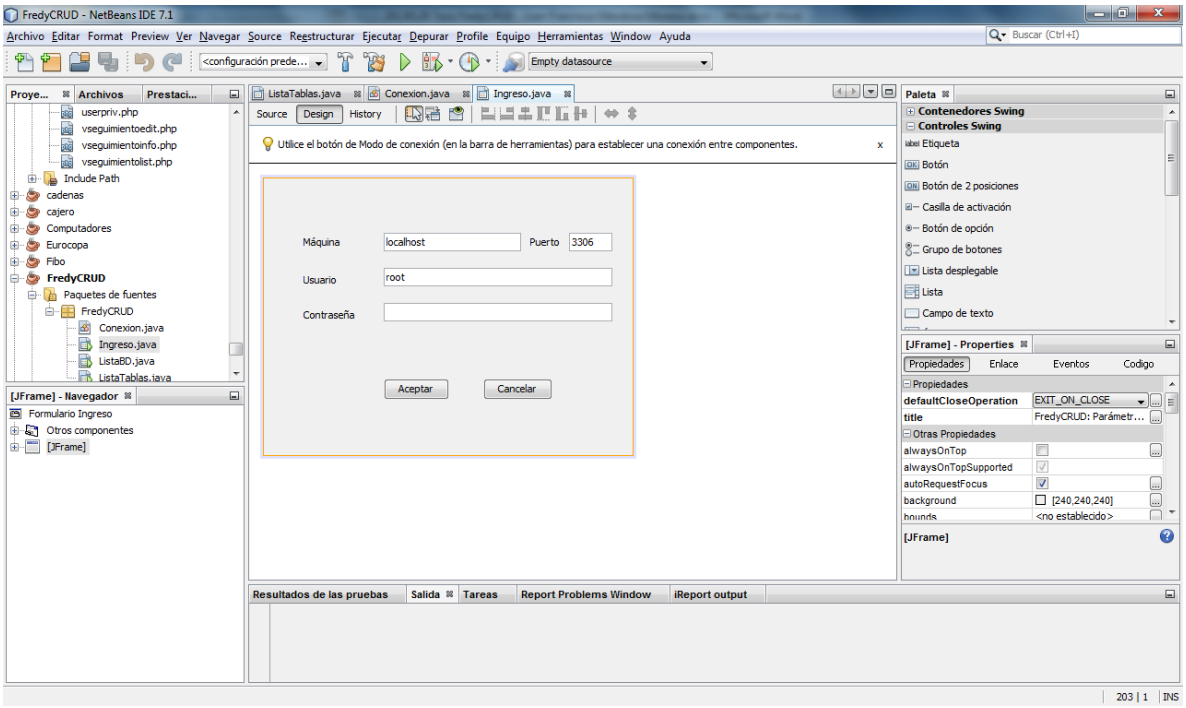



**Ilustración 41. Diseño de la forma de ingreso para el proyecto FredyCRUD, mediante NetBeans IDE**

La **documentación del código** está apoyada por Javadoc[72]. Javadoc es una herramienta para generar documentación API en formato HTML desde comentarios de documentación en el código fuente. Javadoc se descarga como parte del Java 2 SDK. El Doclet genera el documento HTML y se construye en la herramienta JavaDoc. Los principales Doclet son:

- Doclet API
- Taglet API
- MIF Doclet
- DocCheck Doclet
- Exclude Doclet
- Doclet Toolkit

[72] Oracle. Javadoc. Recuperado de http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html el 12 de Diciembre de 2012

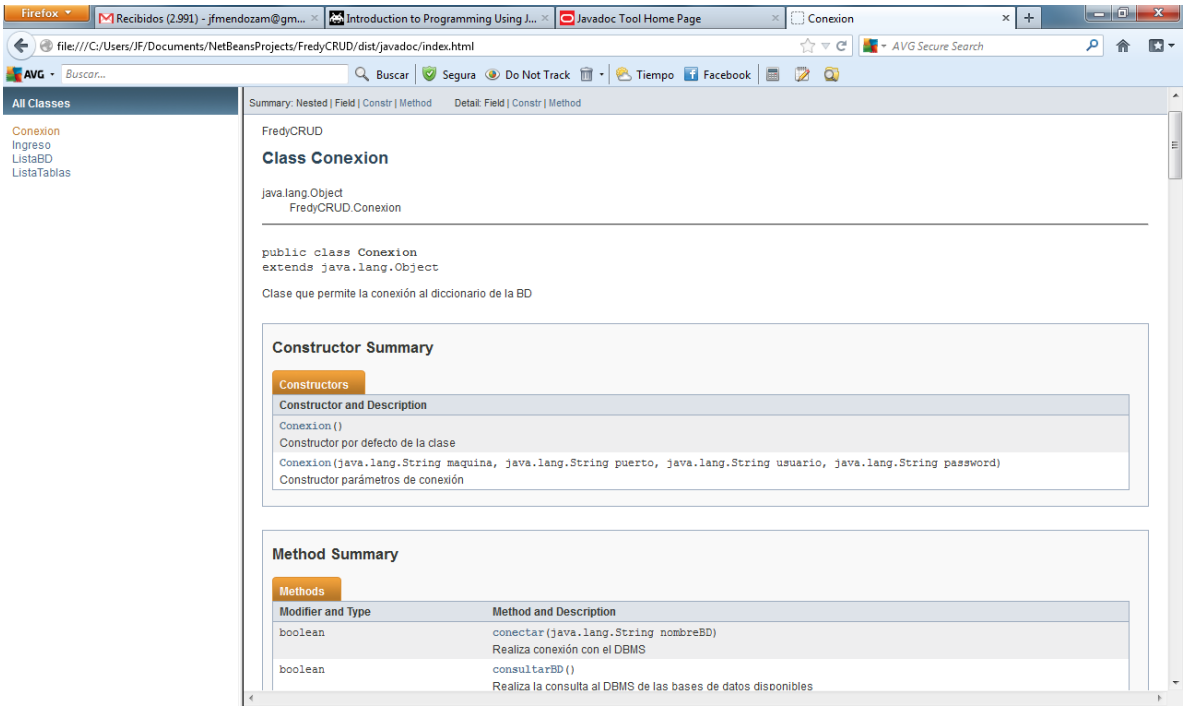
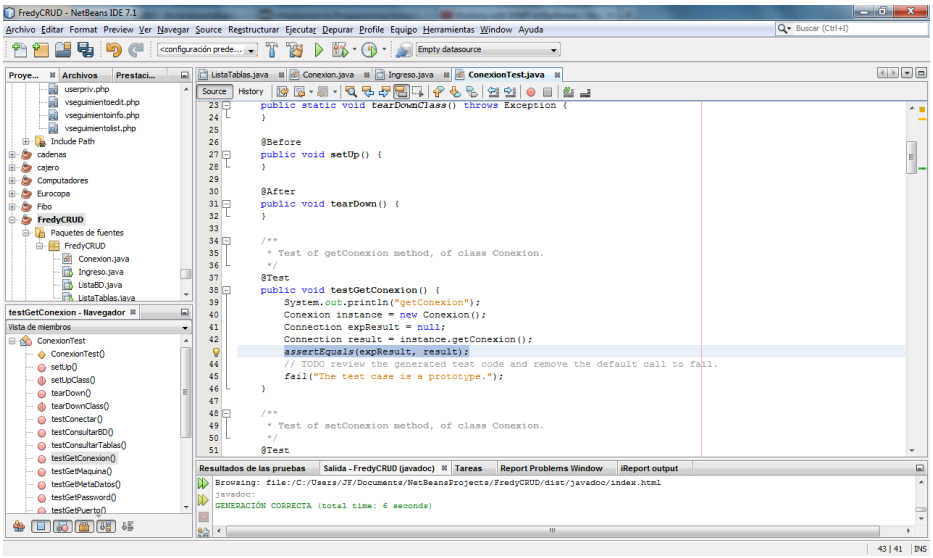



**Ilustración 42. Documentación de la Clase Conexión mediante JavaDoc**

Las **pruebas de software** se hicieron mediante JUnit[73]. JUnit es un sencillo *framework* para escribir pruebas repetibles. Es una instancia de la arquitectura xUnit para *frameworks* de prueba unitarias.




[73] JUnit.org. Junit. Recuperado de http://junit.sourceforge.net/ el 12 de Diciembre de 2012

Ilustración 43. Clase de prueba a la clase Conexión mediante JUnit

La **gestión del versionamiento** se hace mediante Subversión en un servidor Google.

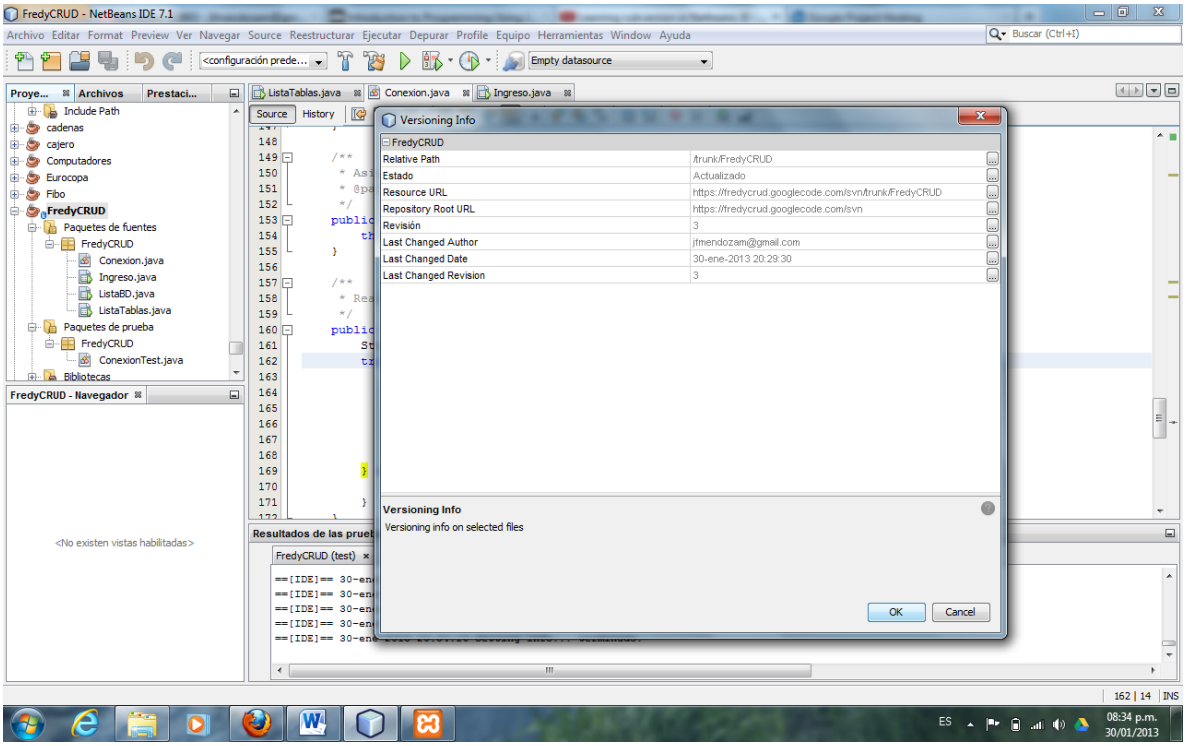



Ilustración 44. Versionamiento de la Aplicación mediante Subversión en los servidores de Google

# 4. CONCLUSIONES Y RECOMENDACIONES

- Los requerimientos de una herramienta de generación automática de código considerados por los desarrolladores de software libre de la región están agrupadas en varios ambientes de acuerdo con: los componentes, la comunidad de software libre, implementación de la herramienta, pruebas sobre la misma, usabilidad, eficiencia, mantenibilidad, portabilidad, entre otros requerimientos.
- Al aplicar el instrumento de recolección de información a la muestra de la comunidad de Software Libre de Boyacá, esta comunidad manifiesta que un tipo de herramientas de generación automática de código son muy necesarias para su labor porque les permite contribuir más con la comunidad con soluciones efectivas y adecuadas con la necesidad de la región. A los entusiastas les permite adentrarse y colaborar más efectivamente.
- Con respecto al diagnóstico comparativo del uso de algunas herramientas actuales, para la generación automática de código, a partir de una base de datos, fueron muchas las herramientas encontradas, pero la mayoría de ellas tienen licencias privativas. De las pocas herramientas con licencia adecuadas a la comunidad de software libre, algunas de ellas generan código de forma automática a partir de una base de datos, porque principalmente trabajan a partir de diagramas UML. Y las únicas herramientas con licencia libre, que generan código a partir de una base de datos, son aquellas que funcionan como componentes de un IDE (Ambiente Integrado de Desarrollo) lo que implica mayor trabajo manual de codificación y pocas posibilidades de personalización.
- Este diagnóstico comparativo realizado denota la necesidad que tiene la comunidad de software libre de disponer este tipo de herramientas, que permiten satisfacer las necesidades de la región. A su vez se presenta como un sustento teórico que justifica la realización de estos tipos de proyectos de investigación
- Para la creación de la herramienta de automatización para la generación de código de interfaces CRUD, a partir de una base de datos, en ambientes de desarrollo de software libre, se encontraron muchos problemas, de los cuales el que más impacto generó fue la falta de personal para conformar el equipo de desarrolladores, porque el software fue desarrollado por una sola persona (el autor). Como estrategia, se hizo uso de métodos de desarrollo ágil de software como SCRUM, que aunque exige un equipo entre 5 y 9 personas, el autor del proyecto tomó como estrategia manejar varios perfiles de usuario para seguir los procesos SCRUM. Aunque el proyecto de desarrollo tomó más tiempo de lo estimado, fue factible su realización.
- El software fue analizado y diseñado de una forma muy sencilla, tan solo en tres grandes módulos: Parametrización de la nueva aplicación, conexión al diccionario de datos y generación del código de la nueva aplicación. Sin embargo, en la etapa de codificación la complejidad de cada uno de estos módulos fue mayor, verificable por la cantidad de código creado.
- El modelamiento del software en las fases de análisis y diseño requirió de apoyo de software para facilitar la labor y disminuir la complejidad del proyecto. Sin embargo, hay que reconocer que herramientas con licencias privativas fueron muy prácticas para realizar el trabajo. Queda como recomendación a la comunidad de software libre tener en cuenta esta necesidad para futuros proyectos de investigación.
- Efectivamente la hipótesis que se había planteado en el proyecto con respecto a que el uso de una herramienta de generación automática de software de tareas básicas reduciría el tiempo de desarrollo de aplicaciones libres en entornos Web que requieren de una base de datos se cumplió.

Era de suponerse por cuanto codificar de forma manual interfaces CRUD es una tarea que requiere mucho tiempo, es rutinaria y propensa a cometer errores de codificación.

# 5. REFERENTES BIBLIOGRÁFICOS

## 5.1. BIBLIOGRAFÍA


Grupo de Sistemas y Comunicaciones. (2004). *Compilación de Ensayos sobre Software Libre.* Madrid, España: Universidad Rey Juan Carlos.

HERNÁNDEZ, J. M., & al., e. (2007). *Ingeniería del Software en Entornos del Software Libre.* Catalunya: Fundació per a la Universitat Oberta de Catalunya.

INEI. (1999). *Herramientas CASE.* Perú: Editorial INEI.

JORBA ESTEVE, J. (s.f.). *Introducción al Sistema Operativo GNU/Linux.* Catalunya: uoc.

KEPPLE, A., WARMER, J., & BAST, W. (2003). *MDA Explained, The Model Driven Architecture: Practice and Promise.* Indianapolis: Addison-Wesley.

MORGAN, C. (1998). *Programming from Specifications.* Prentice Hall international.

MUÑETÓN, A., ZAPATA, C. M., & ARANGO, F. (2007). Reglas para la Generación Automática de Código Defiidas sobre Metamodelos Simplificados de los Diagramas de Clases, Secuencias y Máquinas de Estados de UML 2.0. *Dyna*, 18.

Oracle Corp. (s.f.). *La Base de Datos de Información*. Recuperado el 27 de 11 de 2011, de MySQL: http://dev.mysql.com/doc/refman/5.0/es/information-schema.html

PHP Group. (s.f.). *php*. Recuperado el 27 de 11 de 2011, de php: http://www.php.net/

PIATTINI, M. G., CALVO-MANZANO, J. A., CERVERA, J., & FERNÁNDEZ, L. (2004). *Análisis y Diseño de Aplicaciones Informáticas de Gestión.* México: Alfaomega Rama.

PRESSMAN, R. S. (2009). *Ingeniería del Software "Un Enfoque practico". 7a Edición.* España: McGraw-Hill.

RIVAS, L., & al., e. (2010). Criterios para la Selección de Herramientas de Ingeniería de Software en PYMEs. *Revista de la Facultad de Ingeniería UCV*, 89-104.

ROBERTO IMENES, E. (2006). SELEÇÃO DE FERRAMENTAS CASE. *Jaguariúna*, 1-42.

SCRUM Methodology Org. (s.f.). *SCRUM Methodology Org*. Recuperado el 9 de 01 de 2012, de http://www.scrummethodology.org/

SOMMERVILLE, I. (2005). *Ingeniería del Software.* Madrid: Pearson Educación, S.A.

VILLAPECELLÍN CID, M. M. (2004). *Desarrollo de Aplicaciones en Entornos de 4a Generación y con Herramientas CASE.* Ra-Ma, Librería y Editorial Microinformática.

### 5.2. CITAS BIBLIOGRÁFICAS

## Casos


AppGini. http://www.bigprof.com/appgini/. Recuperado 9 de mayo de 2011 13

BABBAGE, Charles. 26 de diciembre de 1791 - 18 de octubre de 1871. Británico. Matemático: Padre de la Computación 10

BYRON KING, Ada Augusta. 10 de diciembre de 1815 - 27 de noviembre de 1852. Londinense. Primera programadora. 10

CA Erwin Data Modeler. http://erwin.com/. Recuperado 26 de mayo de 2011 17

CASE: *Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora 11

Change Vision. Astah. http://jude.change-vision.com/jude-web/index.html 18

Fujaba. Fujaba Tool Suite. http://www.fujaba.de/ 18

GATES, William Henry III. 28 de octubre de 1955. Estadounidense. Empresario cofundador de Microsoft 10

GNU Operating System. La Definición de Software Libre. http://www.gnu.org/philosophy/free-sw.es.html 20

IBM. Rational System Architect. http://www-01.ibm.com/software/awdtools/systemarchitect/# 18

MARTIN, James. *Managing the Data-base Environment*. Prentice-Hall. 1983 22

monoUML. monoUML CASE tool. http://www.monouml.org/ 18

NAUR, P. y RANDALL, B. *Software Engineering: A Report on a Conference Sponsored by the NATO Science Comitte*. NATO, 1999 11

Número de Bernoulli. Denominado por Abraham de Moivre, en honor de Jakob Bernoulli. Sucesión de números racionales basada en la teoría de números 10

OMG Omondo. Omondo. http://www.ejb3.org/ 18

OPPENHEIMER, Julius Robert. 22 de abril de 1904 – 18 de febrero de 1967. Estadounidense. Físico. Padre de la Bomba Atómica 27

Oracle. Designer. http://www.oracle.com/technetwork/developer-tools/designer/overview/index.html 17

PHP Generator for MySQL. http://www.sqlmaestro.com/products/mysql/phpgenerator/. Recuperado 9 de Mayo de 2011 14

PHPMaker. http://www.hkvstore.com/phpmaker/. Recuperado 9 de mayo de 2011 12

PHPScaffold. http://www.phpscaffold.com/. Recuperado 9 de mayo de 2011 13

POG: PHP Object Generator. http://www.phpobjectgenerator.com/. Recuperado 9 de mayo de 2011 13

PRESSMAN, Roger S. *Ingeniería del Software. Un Enfoque Práctico* 15

PRESSMAN, Roger S. *Ingeniería del Software. Un Enfoque Práctico*. Página 1 10

RITCHIE, Dennis MacAlistair. 9 de septiembre de 1941. Estadounidense. Científico computacional 10

RUIZ LUQUE, José Carlos, et al. Herramientas CASE. Universidad de las Palmas de Gran Canarias 17

STALLMAN, Richard Matthew. 16 de marzo de 1953. Estadounidense. Fundador del movimiento por el software libre 11

starUML. http://staruml.sourceforge.net/en/ 18

Sybase. Power Designer. http://www.sybase.com/products/modelingdevelopment/powerdesigner 18

TANENBAUM, Andrew S. *Sistemas Operativos Modernos*. Segunda Edición. Prentice Hall. Pearson Educación, México 2003. ISBN 970-26-0315-3. Página 189 8

THOMPSON, Kenneth Lane. 4 de febrero de 1943. Estadounidense. Científico computacional 10

Tigris. ArgoUML. http://argouml.tigris.org/ 18

UMLet. http://www.umlet.com/ 18

Visual Paradigm. http://www.visual-paradigm.com/ 18

VisualWade. http://www.visualwade.com/. Recuperado 9 de mayo de 2011 13

VON NEUMANN, John. 28 de diciembre de 1903 – 8 febrero de 1957. Austro-húngaro. Matemático 10

## 6. ANEXOS

## 6.1. ANEXO 1. ENCUESTA DE PERCEPCIÓN

https://docs.google.com/spreadsheet/viewform?fromEmail=true&formkey=dE0xcXNralNOT0doT0VwU3pGUW9tU2c6MA

AVG Secure Search

AVG · Buscar... | Buscar | Segura | Do Not Track | Tiempo | Facebook | Speedtest

# Encuesta de Percepción para el Desarrollo de una Herramienta de Software libre para generar código fuente de forma automática

Como proyecto de tesis de grado de Maestría, se pretende desarrollar una herramienta de software libre que permita generar de forma automática el código fuente para desarrollar un nuevo sistema de información, a partir de una base de datos mySQL. Su opinión, como miembro del grupo de Software Libre de Boyacá, es muy valiosa para nosotros, ya que nos permite perfilar dicha herramienta en beneficio de nuestra comunidad boyacense de software libre.

*Obligatorio

**Nombre completo** *

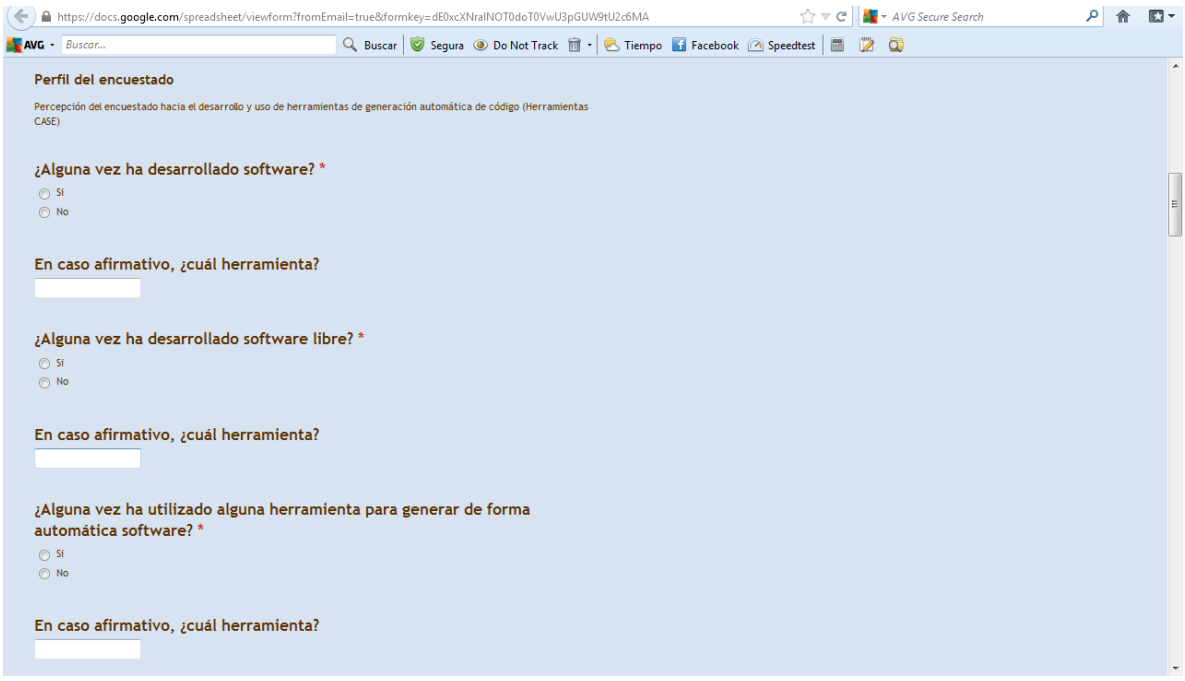
Digite sus nombres y apellidos

**Dirección**

Si así lo desea, digite la dirección de residencia

**Ciudad, Departamento** *

Ciudad y Departamento de Residencia



https://docs.google.com/spreadsheet/viewform?fromEmail=true&formkey=dE0xcXNralNOT0doT0VwU3pGUW9tU2c6MA

AVG Secure Search

AVG · Buscar... | Buscar | Segura | Do Not Track | Tiempo | Facebook | Speedtest

**Perfil del encuestado**

Percepción del encuestado hacia el desarrollo y uso de herramientas de generación automática de código (Herramientas CASE)

**¿Alguna vez ha desarrollado software?** *

- Si
- No

**En caso afirmativo, ¿cuál herramienta?**

**¿Alguna vez ha desarrollado software libre?** *

- Si
- No

**En caso afirmativo, ¿cuál herramienta?**

**¿Alguna vez ha utilizado alguna herramienta para generar de forma automática software?** *

- Si
- No

**En caso afirmativo, ¿cuál herramienta?**

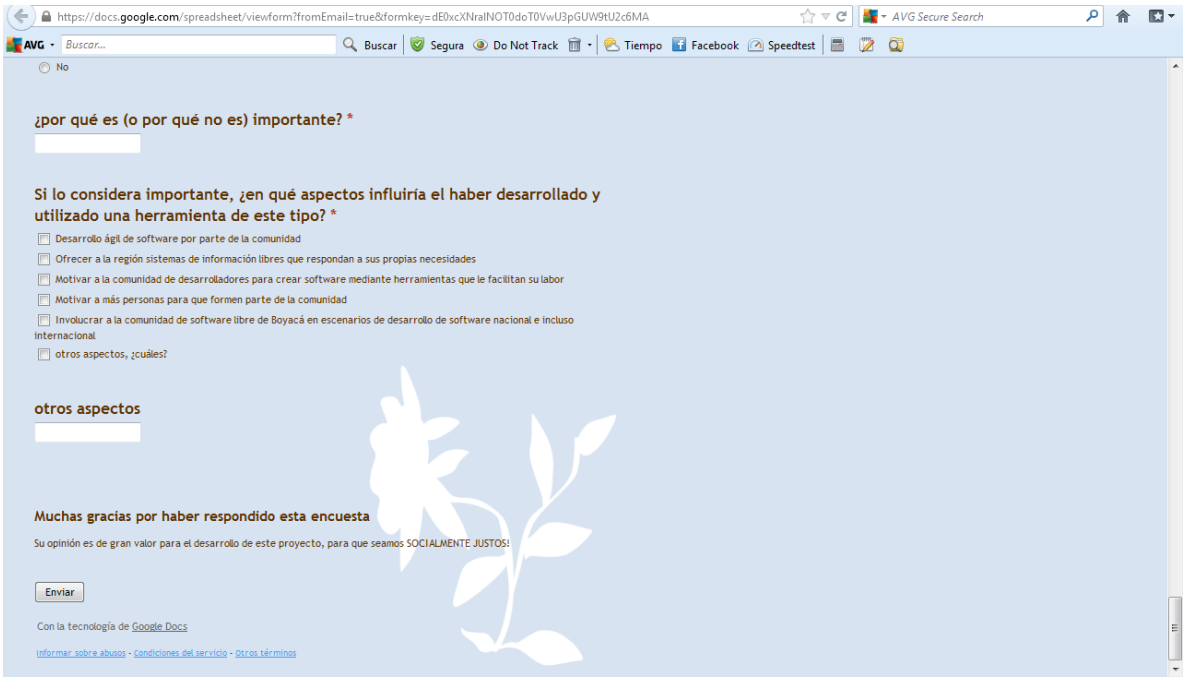



**Ilustración 45. Encuesta en Línea para Evaluar la Percepción de la Herramienta a Desarrollar**[74]

# Encuesta de Percepción para el Desarrollo de una Herramienta de Software libre para generar código fuente de forma automática

Como proyecto de tesis de grado de Maestría, se pretende desarrollar una herramienta de software libre que permita generar de forma automática el código fuente para desarrollar un nuevo sistema de información, a partir de una base de datos mySQL. Su opinión, como miembro del grupo de Software Libre de Boyacá, es muy valiosa para nosotros, ya que nos permite perfilar dicha herramienta en beneficio de nuestra comunidad boyacense de software libre.

Nombre completo * Digite sus nombres y apellidos

Dirección Si así lo desea, digite la dirección de residencia

Ciudad, Departamento * Ciudad y Departamento de Residencia

[74] Fuente: El Autor. Encuesta de Percepción. Recuperado de https://docs.google.com/spreadsheet/viewform?fromEmail=true&formkey=dE0xcXNralNOT0doT0VwU3pGUW9tU2c6MA Octubre 2012

Número móvil Número del teléfono móvil

nombre Twitter nombre del usuario en twitter

## Perfil del encuestado

Percepción del encuestado hacia el desarrollo y uso de herramientas de generación automática de código (Herramientas CASE)

¿Alguna vez ha desarrollado software? *

- Si No En caso afirmativo, ¿cuál herramienta?

¿Alguna vez ha desarrollado software libre? *

- Si No En caso afirmativo, ¿cuál herramienta?

¿Alguna vez ha utilizado alguna herramienta para generar de forma automática software? *

- Si No En caso afirmativo, ¿cuál herramienta?

## Opinión sobre herramientas de generación automática de código

Herramientas de generación automática de código son aquellas que permiten crear nuevo software sin la necesidad de digitar su código de forma manual, comúnmente son conocidas como herramientas CASE

¿Qué opinión le merece las herramientas que permiten generar código? *

- Me gustan porque son muy útiles y ayudan al desarrollador de software a hacer su trabajo
- Me son indiferentes, me da igual generar código de forma manual

- No me gustan, me gusta desarrollar mi propio código fuente, sin ningún tipo de ayuda

¿Qué ventajas ofrecen las herramientas de generación automática de software? *

- Reducción en tiempo de desarrollo
- Minimizar la posibilidad de cometer errores de codificación
- Reducir el esfuerzo de programación por parte del desarrollador
- Estandarización de codificación de software
- Evitar el desarrollo de código de módulos repetitivos
- Otras ventajas, ¿cuáles?

otras ventajas

¿Qué desventajas ofrecen las herramientas de generación automática de software? *

- Código ineficiente
- Entender la forma de codificación de un tercero
- Mantenimiento de software complicado
- Otras desventajas, ¿cuáles?

otras desventajas

## Requisitos que debe reunir una herramienta de generación automática de código

Me gustaría que el código generado sea a partir de la etapa de: *

- El análisis de software
- El diseño de software
- El análisis y diseño de software

- ☐ Los diagramas UML (Lenguaje Unificado de Modelado)
- ☐ Ninguna de las etapas anteriores, se me facilita que sea solamente a partir de una base de datos

Al momento de generar las interfaces de Consulta, Ingreso, Modificación o Borrado de Registros de la Base de Datos, la herramienta de generación automática de código debe tener en cuenta: *

- ☐ El diseño de la interfaz al usuario
- ☐ La seguridad de acceso a los datos de acuerdo al perfil del usuario
- ☐ La fácil navegabilidad entre registros
- ☐ otras características, ¿cuáles?

otras características

Este tipo de herramientas, debe generar código para generar aplicaciones tipo: *

- ☐ Escritorio
- ☐ Web
- ☐ Móviles
- ☐ otras plataformas, ¿cuáles?

otras plataformas

El código generado por este tipo de herramientas debe ser para: *

- ☐ La Interfaz del usuario
- ☐ El Acceso a la Base de Datos
- ☐ Los Reportes
- ☐ Otro tipo de generación, ¿cuál?

otro tipo de generación

Con respecto al mantenimiento de la aplicación generada, Usted exigiría de una herramienta de generación automática de código:

- ☐ Código bien estructurado
- ☐ Código bien documentado
- ☐ Código estandarizado
- ☐ Otra exigencia, ¿cuál?

otra exigencia

La herramienta desarrollada debe permitir *

- ☐ La reutilización del código
- ☐ La portabilidad del software
- ☐ La estandarización de la documentación
- ☐ otra posibilidad, ¿cuál?

otra posibilidad

Qué roles de la Ingeniería de software deben hacer uso de este tipo de herramientas: *

- ☐ Gerente del proyecto
- ☐ Analista
- ☐ Diseñador
- ☐ Desarrollador de software (programador)
- ☐ Tester
- ☐ Documentador
- ☐ Administrador de la aplicación
- ☐ Administrador de la base de datos (dbA)
- ☐ Usuario
- ☐ otro rol, ¿cuál?

otro rol

## Calidad del software

¿Cuáles criterios son importantes tener en cuenta al momento de evaluar la calidad del software del código generado? *

- Funcionalidad
- Fiablidad
- Usabilidad
- Eficiencia
- Mantenibilidad
- Portabilidad
- otro criterio, ¿cuál?

otro criterio

## Contextualización del proyecto de software a desarrollar

Pertinencia de este tipo de herramientas para la comunidad de software libre de Boyacá

¿Cree importante este tipo de herramientas de software libre para la comunidad en Boyacá? *

- Si
- No

¿por qué es (o por qué no es) importante? *

Si lo considera importante, ¿en qué aspectos influiría el haber desarrollado y utilizado una herramienta de este tipo? *

- Desarrollo ágil de software por parte de la comunidad
- Ofrecer a la región sistemas de información libres que respondan a sus propias necesidades
- Motivar a la comunidad de desarrolladores para crear software mediante herramientas que le facilitan su labor
- Motivar a más personas para que formen parte de la comunidad
- Involucrar a la comunidad de software libre de Boyacá en escenarios de desarrollo de software nacional e incluso internacional
- otros aspectos, ¿cuáles?

otros aspectos

## Muchas gracias por haber respondido esta encuesta

Su opinión es de gran valor para el desarrollo de este proyecto, para que seamos SOCIALMENTE JUSTOS!

---

Final del formulario

## 6.2. ANEXO 2. RESULTADOS DE LA APLICACIÓN DEL INSTRUMENTO ESTADÍSTICO

El grupo de Software Libre de Boyacá está conformado actualmente por 66 miembros voluntarios, son personas de universidades, colegios, de empresas o ciudadanos. Su distribución por municipios se puede apreciar en la "Ilustración 36. Conformación Grupo Software Libre Boyacá por municipios (Fuente GSL-Boyacá)".

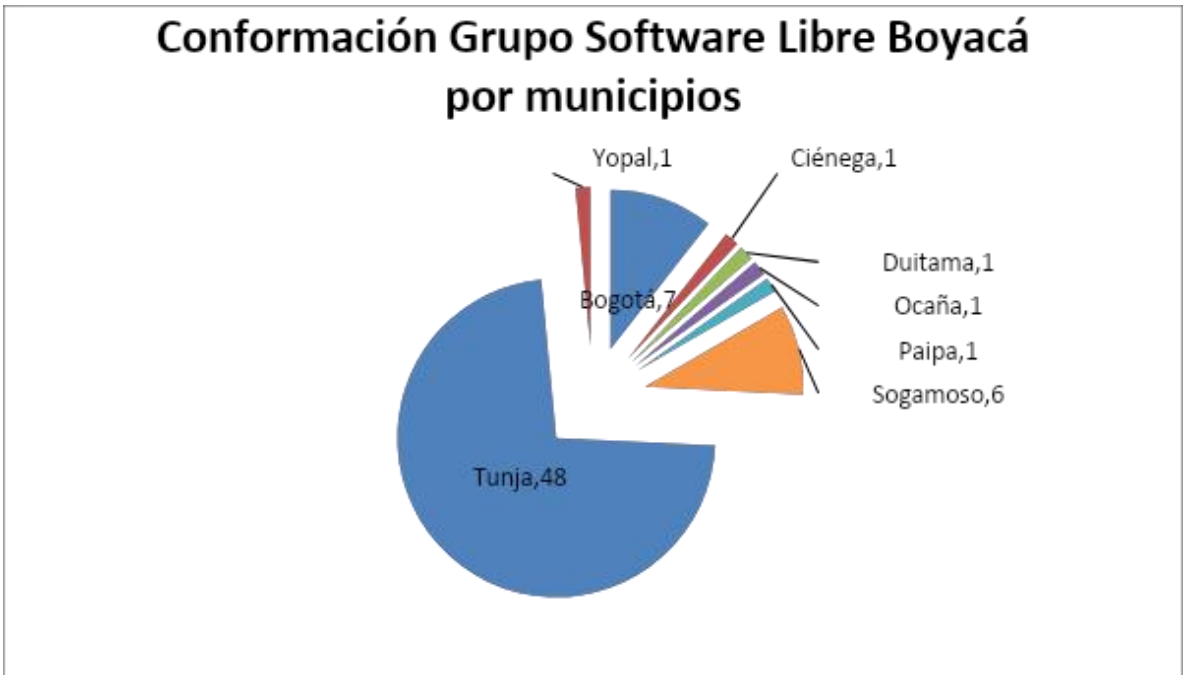



**Ilustración 46. Conformación Grupo Software Libre Boyacá por municipios (Fuente GSL-Boyacá)**

Gracias a la colaboración de los coordinadores del Grupo de Software Libre de Boyacá, se envió el formulario de la encuesta de percepción de este proyecto al correo electrónico de cada uno de los miembros registrados en este grupo. A esta invitación contestaron 10 personas, con ellas se conformó el grupo de muestra de la población, con una representatividad del 15% sobre la población total del grupo.

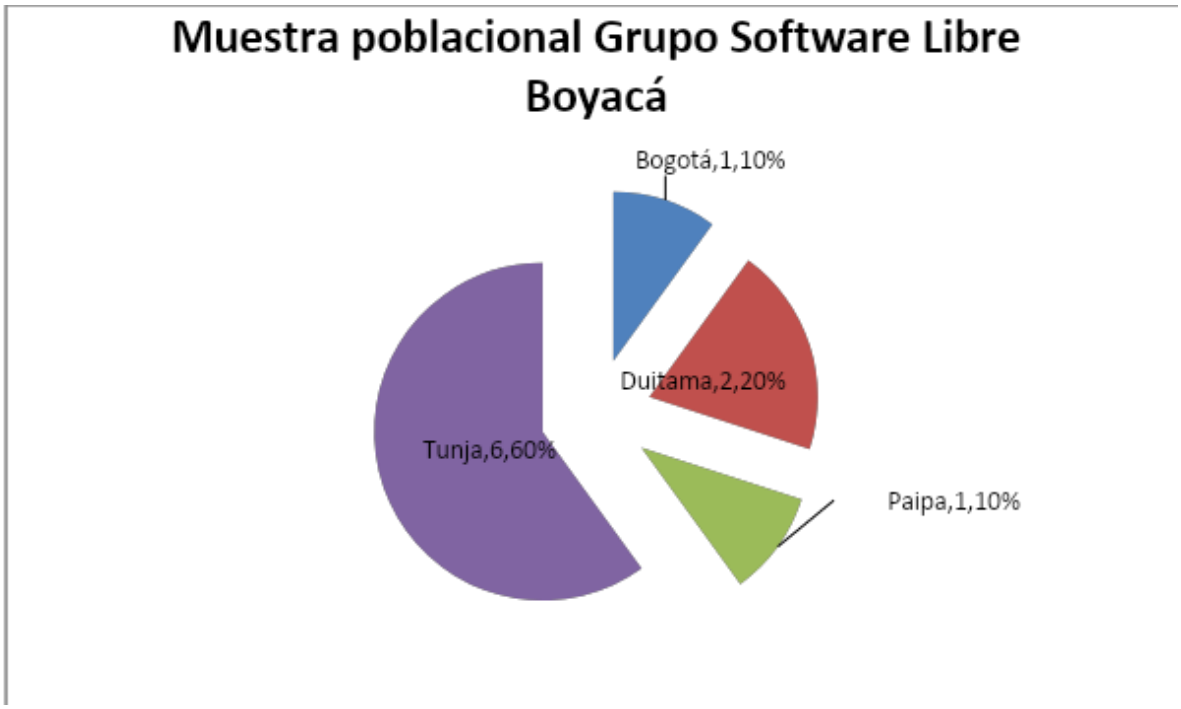



Ilustración 47. Muestra poblacional GSL Boyacá (Fuente El Autor)

A continuación se presentan los resultados por cada una de las respuestas:

Todos los miembros encuestados han tenido experiencia en el desarrollo de software:

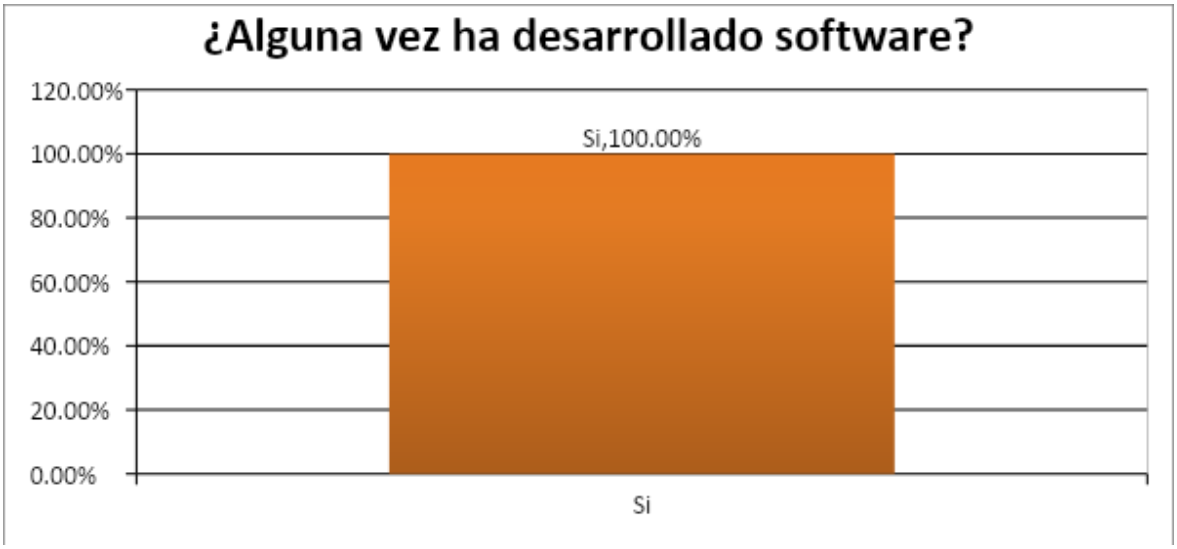



Ilustración 48. Resultados a la pregunta: ¿alguna vez ha desarrollado software?

El lenguaje de programación más utilizado por los encuestados es JAVA, usado por 60% de los encuestados. Le siguen los productos de Microsoft (ASP, Visual Basic o .NET Visual Studio) con un 30%. Cada respuesta se muestra en un porcentaje con respecto a la totalidad de personas de la muestra. Es decir existen casos de encuestados que utilicen más de un lenguaje de programación.

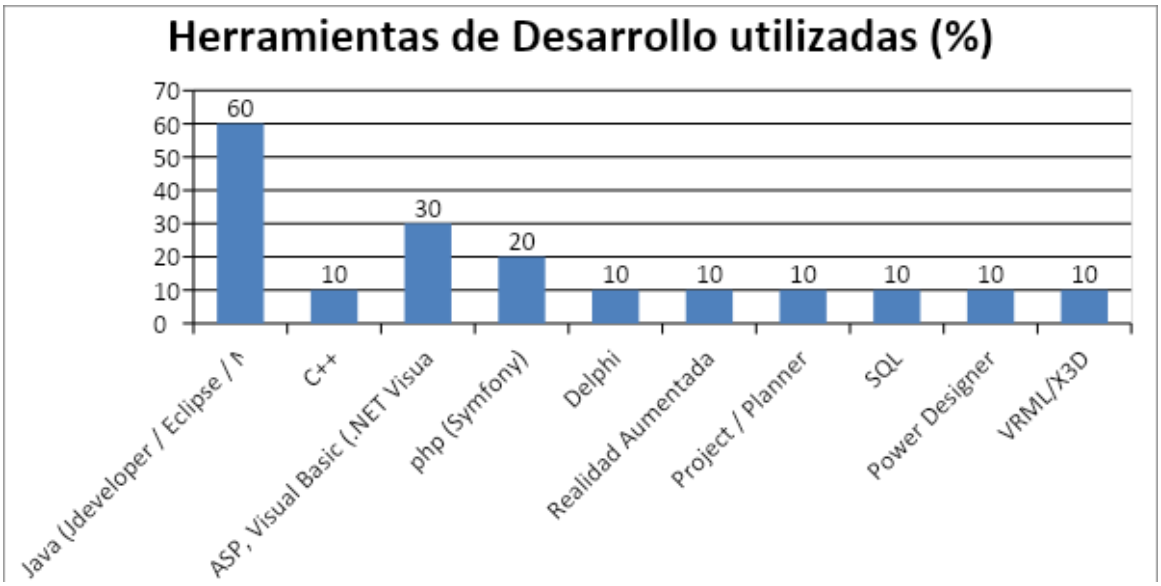



**Ilustración 49. Herramientas (y lenguajes) de desarrollo utilizadas**

El 60% de los encuestados ha desarrollado con alguna herramienta con licencia de software libre:

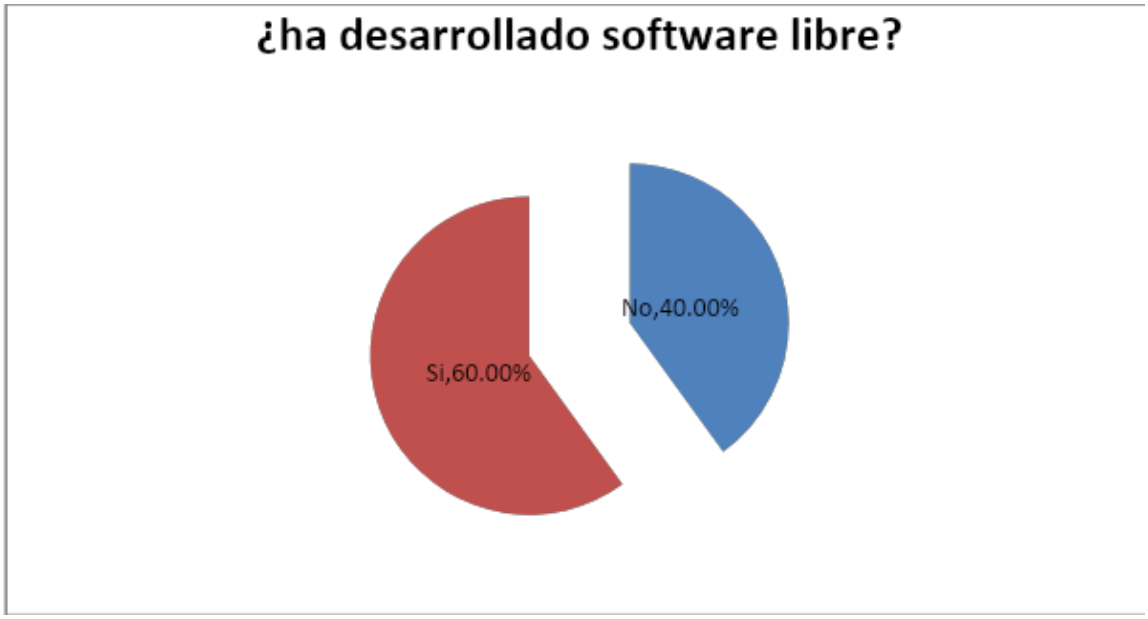



**Ilustración 50. Respuestas a la pregunta ¿ha desarrollado software libre?**

Dentro de las herramientas de licencia de software libre más utilizadas por parte de los encuestados, están JAVA (20%), y php (incluyendo al framework Symfony) con igual porcentaje:

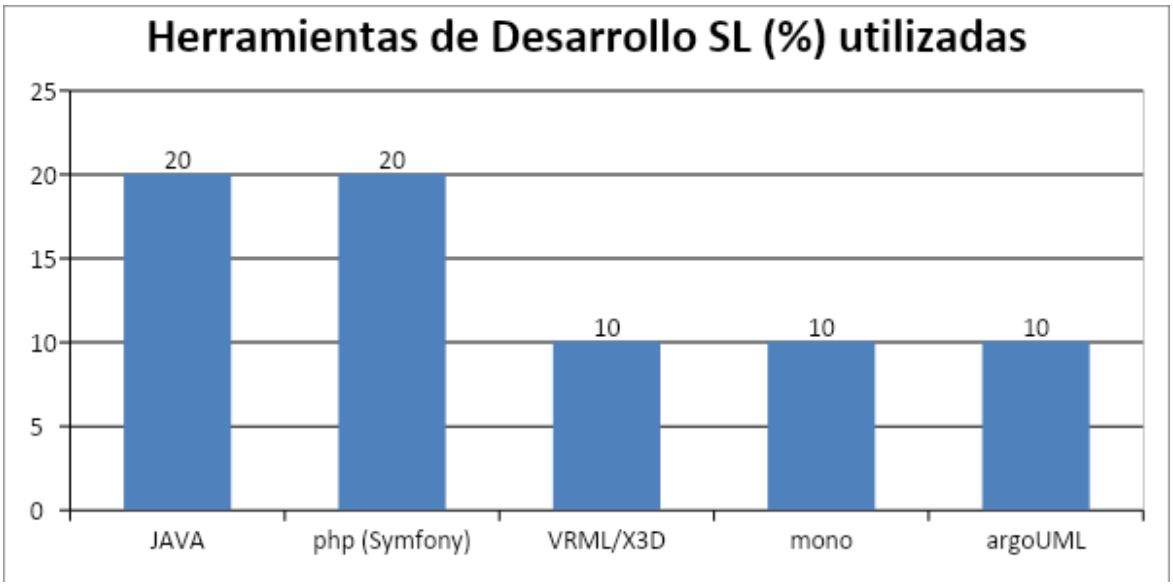



**Ilustración 51. Herramientas de desarrollo de software libre utilizadas**

El 80% de los encuestados alguna vez ha utilizado alguna herramienta generadora de código fuente:

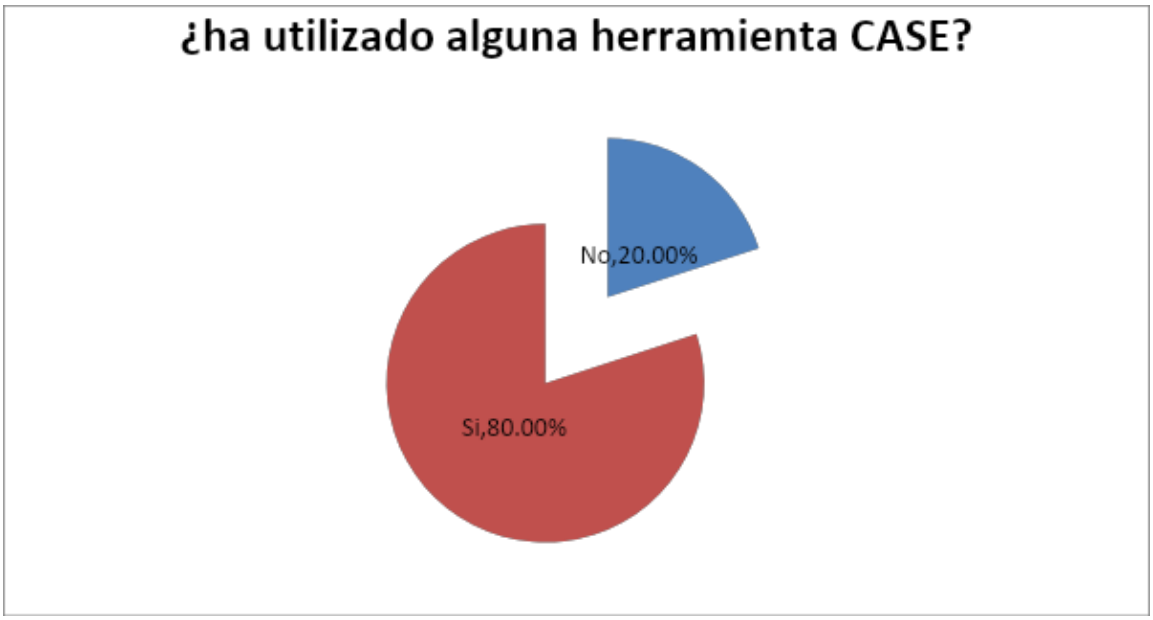



**Ilustración 52. Respuestas a la pregunta: ¿ha utilizado alguna herramienta para generar código de forma automática?**

Dentro de las herramientas CASE más utilizadas (cada una con un 20%) están: *frameworks* o *plug-ins* que hacen parte de un IDE como NetBeans o Visual Studio, herramientas de modelamiento como Power Designer y Enterprise Architect.

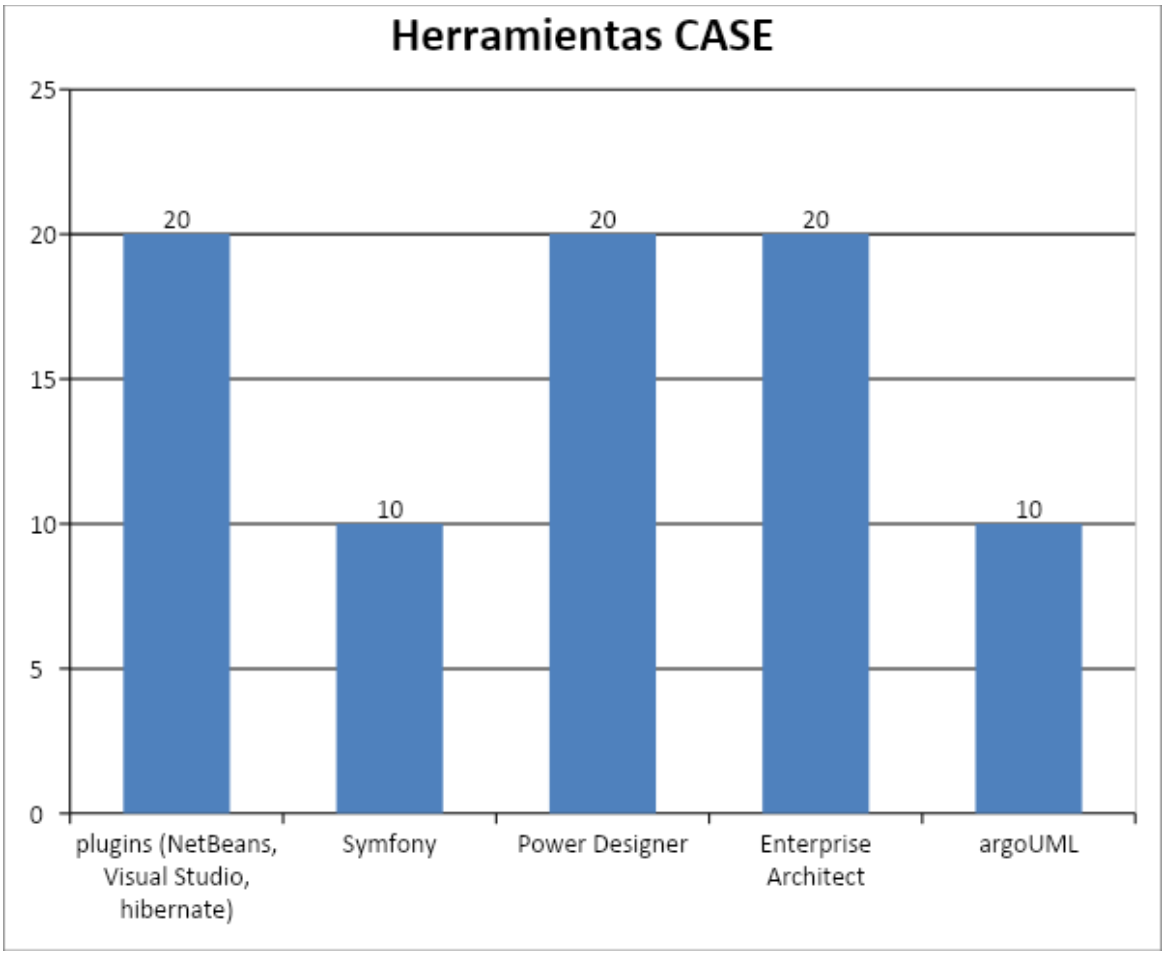



**Ilustración 53. Herramientas utilizadas de generación automática de software**

En un 60% de los encuestados demuestra su entusiasmo por utilizar herramientas CASE debido a su utilidad y apoyo al desarrollador de software. A un 10% de los encuestados le es indiferente el uso de herramientas CASE, mientras que el 30% manifiesta no gustarle este tipo de herramientas por cuanto prefieren desarrollar de la manera tradicional, escribiendo por ellos mismos la totalidad del código de la aplicación.

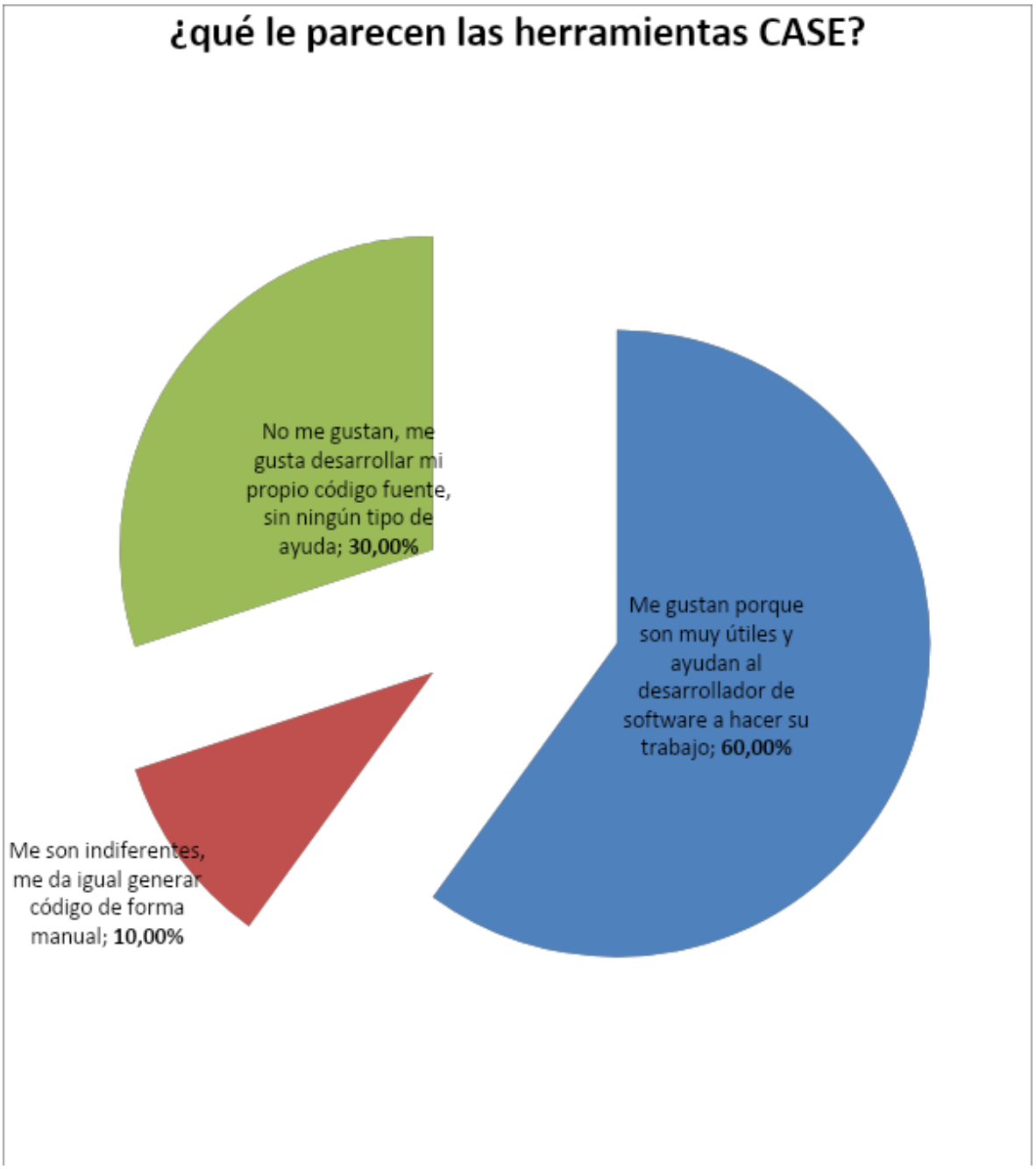



**Ilustración 54. Respuestas a la pregunta ¿qué opinión le merece las herramientas de generación automática de software?**

El 80% de los encuestados notan como ventajas del uso de herramientas CASE la posibilidad de reducir el esfuerzo de programación por parte del desarrollador. También con el 80%, valoran de las herramientas CASE la posibilidad de evitar el desarrollo de código de módulos repetitivos. El 70% considera que se puede reducir el tiempo de desarrollo y el 60% valora que se facilita la estandarización de codificación de software.

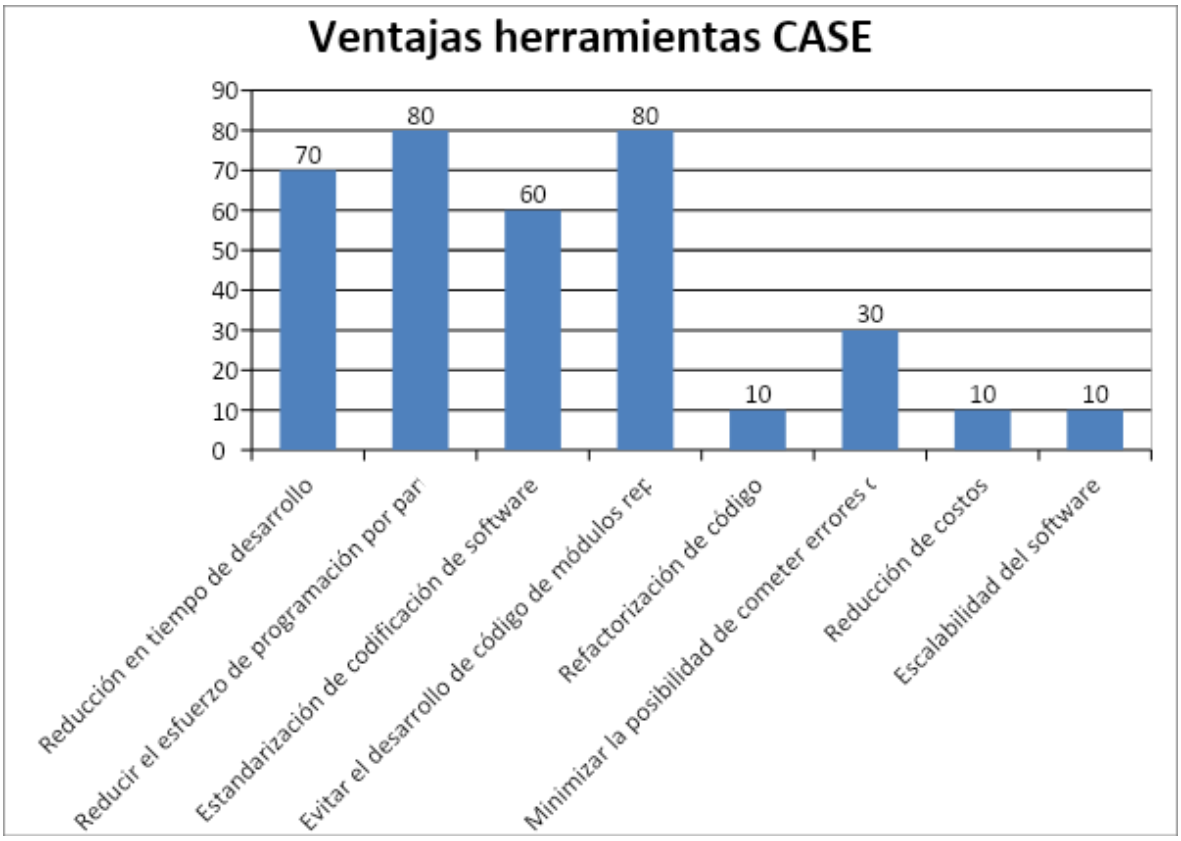



**Ilustración 55. Ventajas de utilizar herramientas que generan código de forma automática**

Como una notoria desventaja del uso de herramientas CASE, el 80% de los encuestados cree que es difícil entender la forma de codificación de un tercero, le sigue un 60%, que considera que estas herramientas pueden generar código ineficiente y un 50% cree que el mantenimiento de software se complica al utilizar las herramientas CASE.

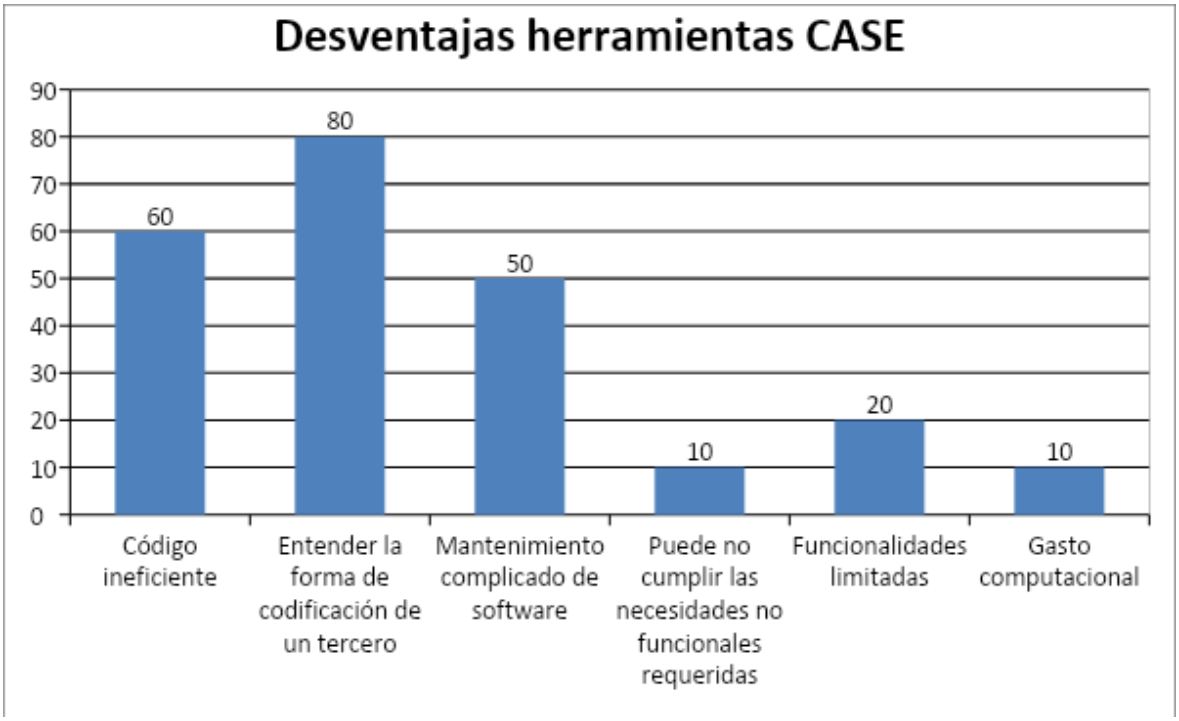



**Ilustración 56. Desventajas al utilizar herramientas que generan código de forma automática**

A la totalidad de encuestados les gustaría que el código fuente generado sea partir de una base de datos, resultante del trabajo en el Diseño de Software (80%), de Diagramas UML (80%), del Análisis de Software (40%). El 10 % contempla generar código sin tener en cuenta ninguna etapa anterior, es decir a partir de la base de datos independientemente de la forma como se generó.

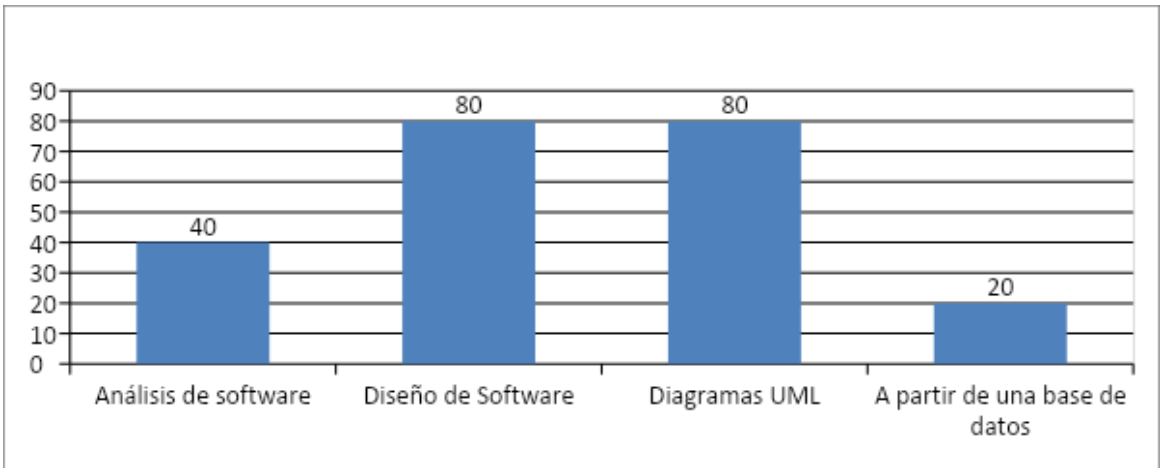



**Ilustración 57. Etapa propicia a partir de la cual se genera automáticamente el código**

El 70% considera que una función primordial que deben ofrecer las herramientas CASE es la seguridad al momento de manipular los datos. También son importantes las funcionalidades de navegación dentro de la aplicación resultante (60%) y que la interfaz presente un diseño agradable al usuario (50%).

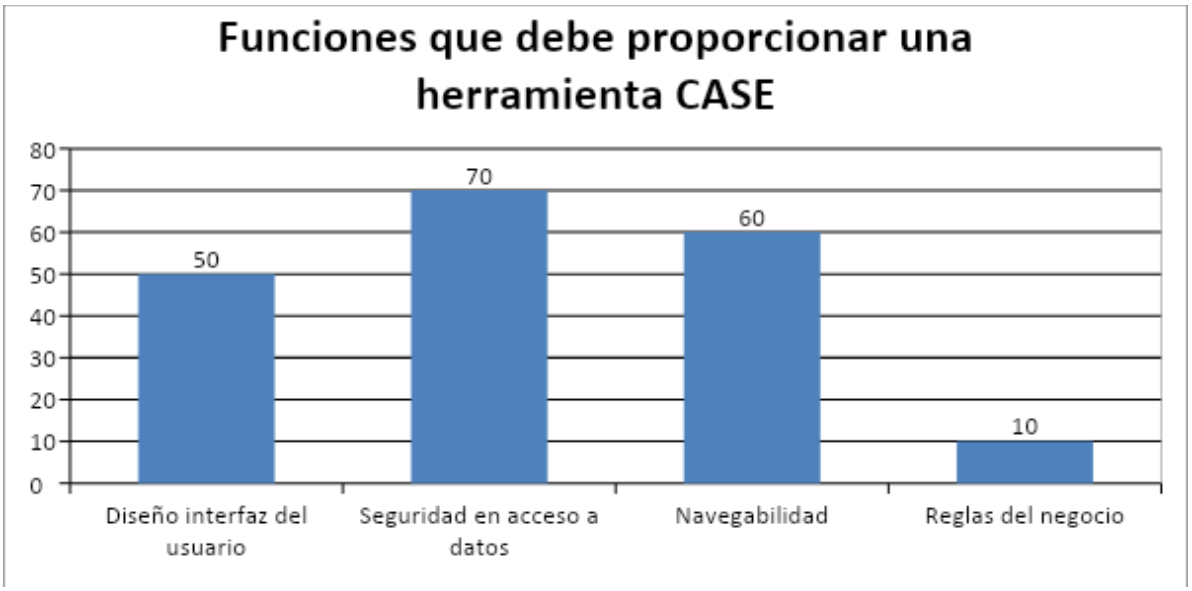



**Ilustración 58. Funciones que debe proporcionar el código generado**

Estas aplicaciones generadas deberían estar desarrolladas de tal forma que funcionen en ambientes Web (100%), también la muestra denota su interés para que también se generen aplicaciones en entornos móviles (90%) y los ambientes de escritorio aún cuentan con interés por parte de los encuestados (30%).

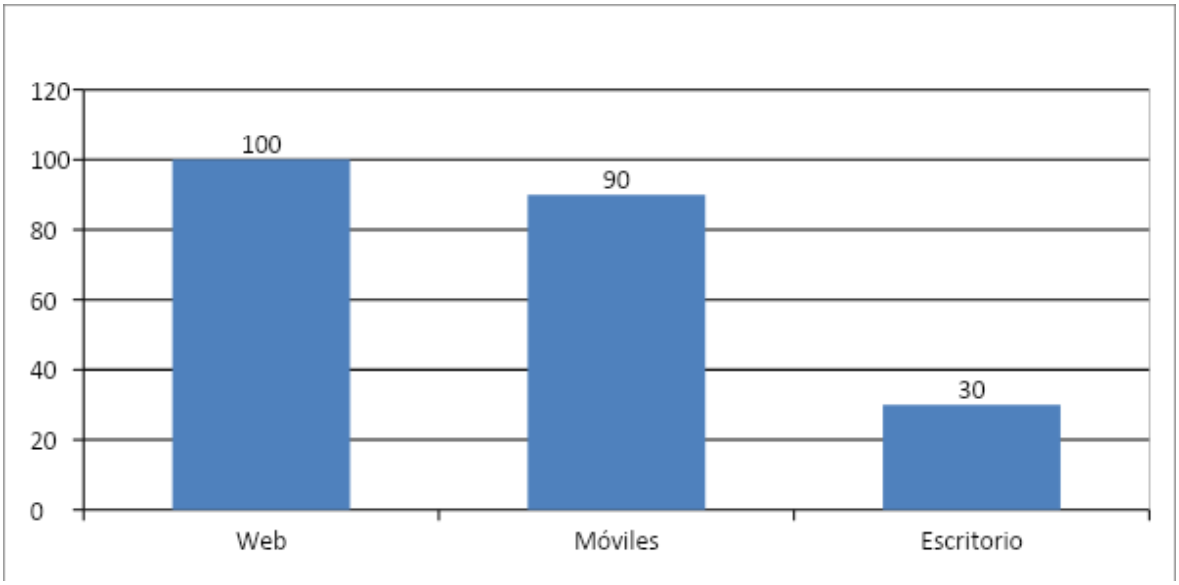



**Ilustración 59. Tipos de plataformas para donde se debe generar automáticamente código**

Las funcionalidades que deben estar presentes en estas herramientas generadas automáticamente deben ser la posibilidad de crear la interfaz al usuario (80%), que permitan manipular la información de una base de datos (70%) y la generación de reportes (60%).

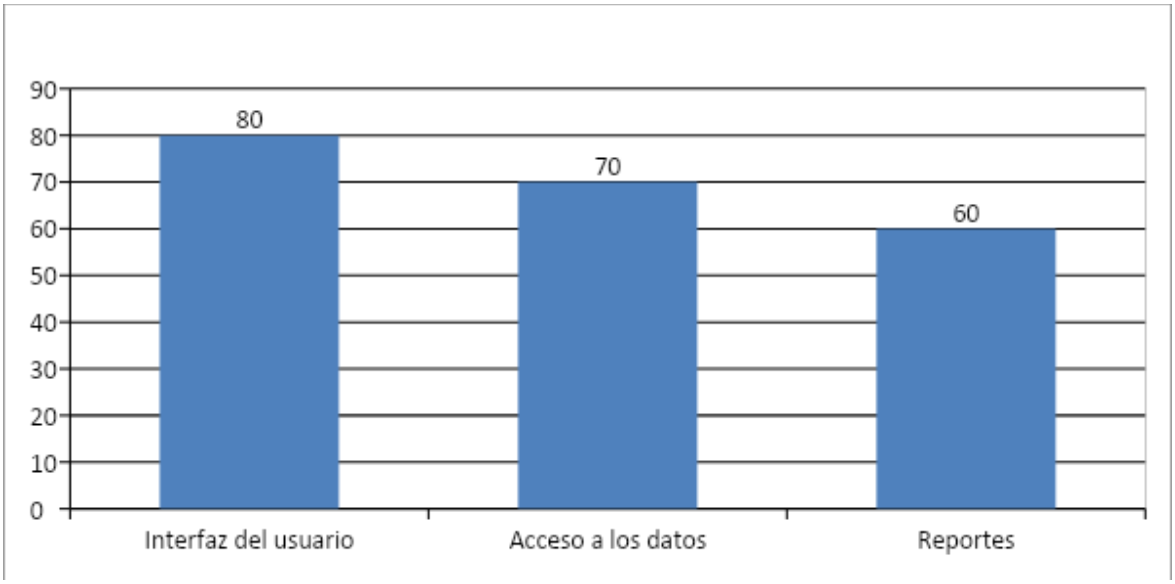



**Ilustración 60. Utilidad del código generado de forma automática**

La muestra en estudio, espera que el software generado presente características ideales como que el código esté bien documentado (80%), que se haya creado siguiendo un estándar (70%) y que esté bien estructurado para su fácil comprensión (50%).

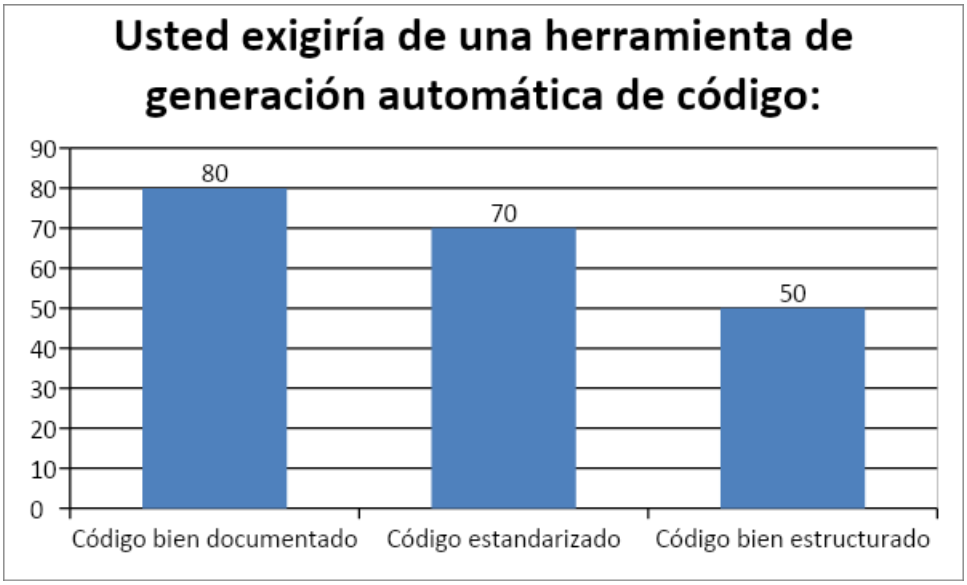



**Ilustración 61. Características exigidas por el código generado, en la etapa del mantenimiento**

Los principales objetivos de la herramienta generada son: Reutilizar el código fuente, es decir que se pueda seguir desarrollando a partir del código fuente generado (90%), que la herramienta sea portable, es decir que se pueda ejecutar en diferentes plataformas operativas (80%), que el código fuente generado permita posterior mantenimiento (10%) y que esté bien documentado, de acuerdo con estándares (60%).

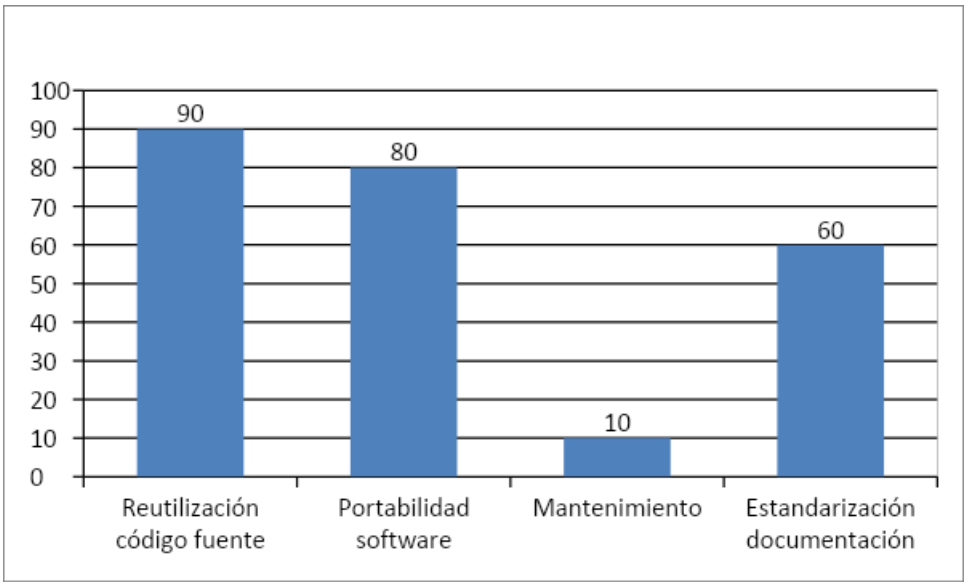



**Ilustración 62. Funcionalidades de la herramienta propuesta**

Dentro de los roles de la Ingeniería de Software que pueden utilizar de mayor forma este aplicativo, caben mencionar al Desarrollador (100%), seguido muy de lejos por el Diseñador (40%).

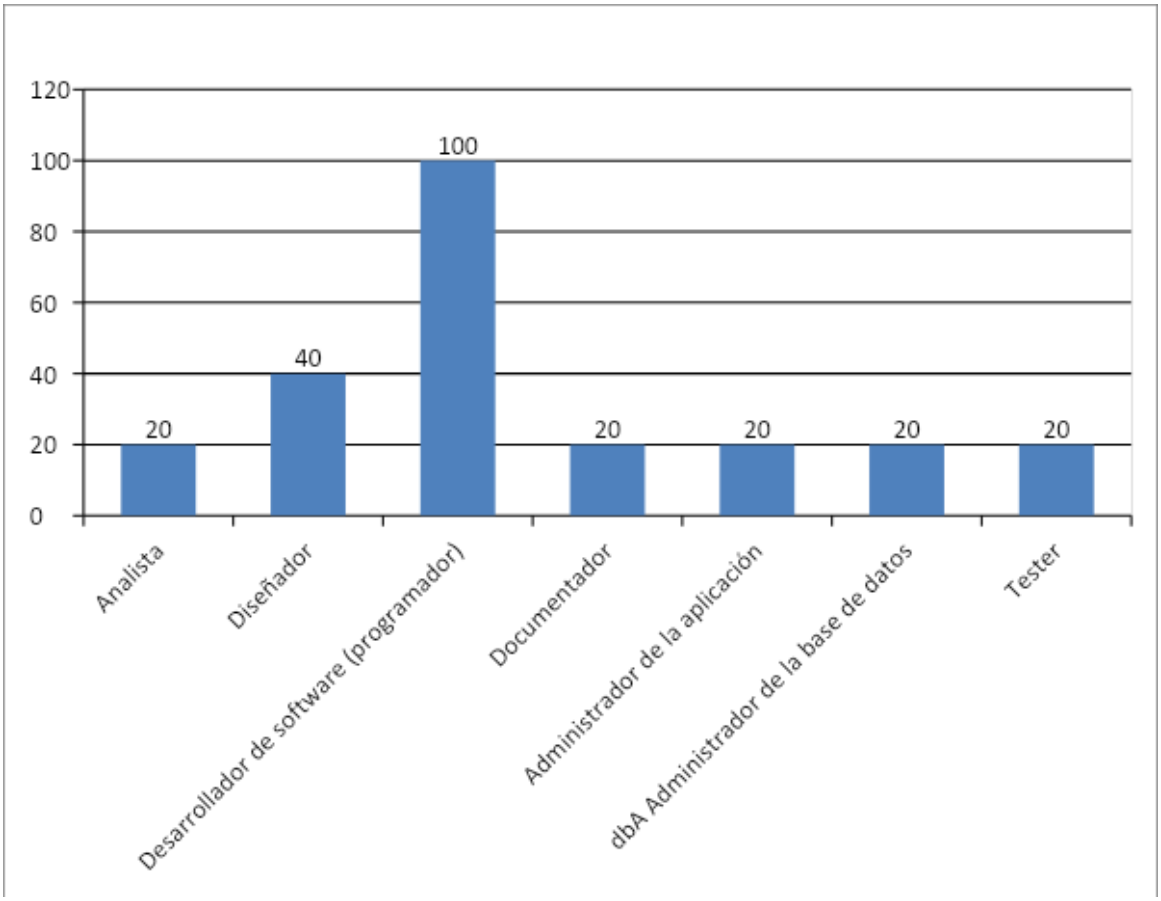



**Ilustración 63. Usuarios de la herramienta**

Como criterios a considerar para evaluar la calidad del software generado, están la Funcionalidad y Eficiencia de la aplicación generada (cada una con 90%), la Mantenibilidad (80%), la Fiabilidad (70%) y la usabilidad y portabilidad (cada una con un 70%).

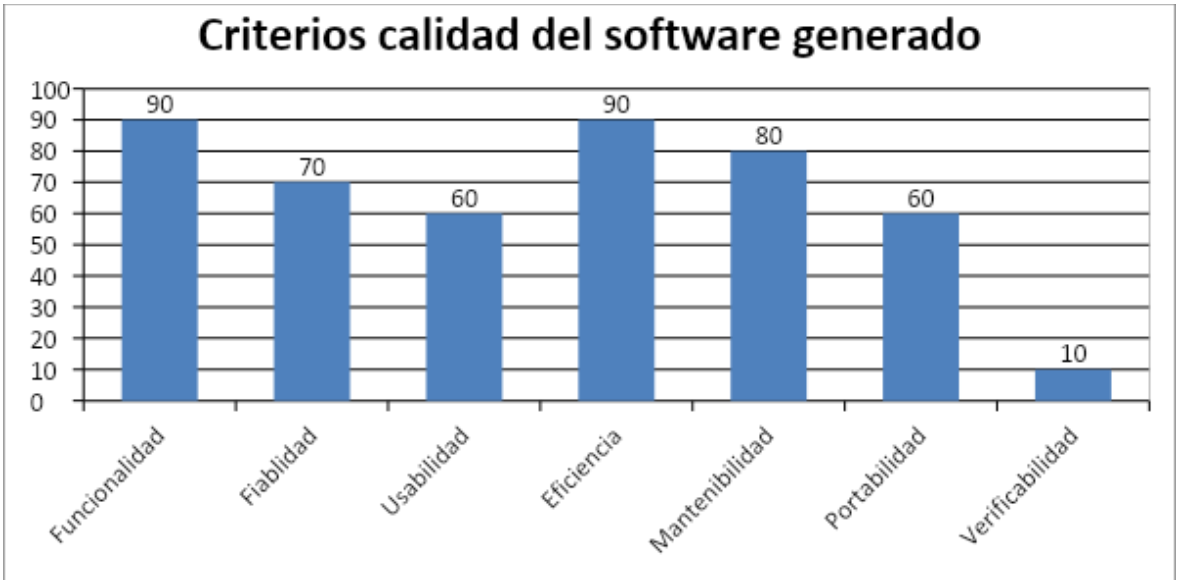



Ilustración 64. Criterios de calidad del software generado

Contextualizando en la región de donde se extrajo la muestra, los encuestados consideran en un 90% que las herramientas que generan software son muy importantes para la región. Este elemento de la encuesta es muy importante para este proyecto de investigación porque podría satisfacer una necesidad sentida.


Ilustración 65. Importancia de este tipo de herramientas para la comunidad

Las razones que aducen los encuestados para justificar el desarrollo y uso de este tipo de aplicaciones son:

- Desarrollo ágil de software por parte de la comunidad (80%)
- Motivar a la comunidad de desarrolladores para crear software mediante herramientas que le facilitan su labor (80%)
- Involucrar a la comunidad de software libre de Boyacá en escenarios de desarrollo de software nacional e incluso internacional (60%)
- Motivar a más personas para que formen parte de la comunidad (60%)
- Ofrecer a la región sistemas de información libres que respondan a sus propias necesidades (40%)

De otra parte, un 10% de los encuestados no prefiere este tipo de herramientas por cuanto prefieren desarrollar el código de una forma tradicional

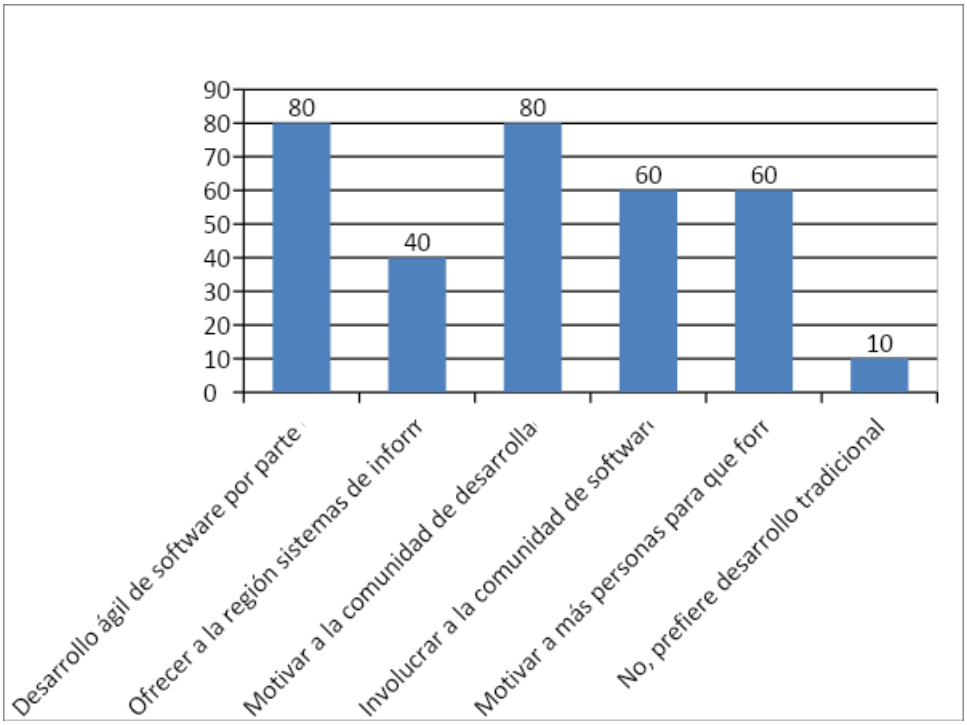



**Ilustración 66. Razones de la importancia de este tipo de herramientas**

## 6.3. ANEXO 3. DOCUMENTO TÉCNICO INGENIERÍA DE SOFTWARE PROYECTO FREDYCRUD

El proceso de Ingeniería de Software para el proyecto FredyCRUD fue apoyado con herramientas como SyBase PowerDesigner, Sparx Enterprise Architect, WireFraming, UMbreLlo, ArgoUML y phpmyAdmin.

### 6.3.1. ANÁLISIS DE REQUISITOS

El análisis de requerimientos o requisitos se dividen en cuatro grandes áreas de trabajo:

- El Negocio
- El Modelo BPMN
- La Definición de los Casos de Uso
- La Ingeniería de Requisitos

#### 6.3.1.1. EL NEGOCIO

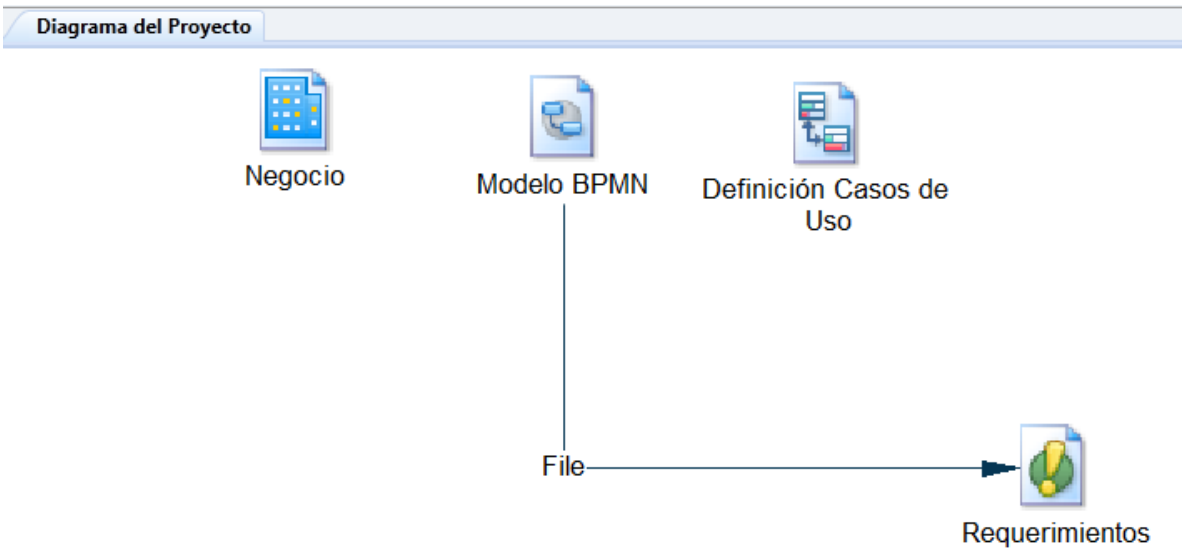



Ilustración 67. Análisis de Requisitos proyecto FredyCRUD

Se inicia con el Modelamiento del Proceso del Negocio (BPM) que permite identificar, describir y descomponer los procesos de negocio. Se puede analizar el sistema en varios niveles de detalle y alternativamente enfocarse en el control del flujo (la secuencia de ejecución) o el flujo de datos (intercambio de datos). Puede usarse BPEL

(*Business Process Execution Language*), BPMN (*Business Process Modeling Notation*) y muchos otros lenguajes de proceso.

Los **Diagramas Jerárquicos del Proceso** (o diagrama de descomposición funcional) suministra una vista gráfica de las funciones de un sistema y ayuda a descomponerlos en un árbol de subprocesos.

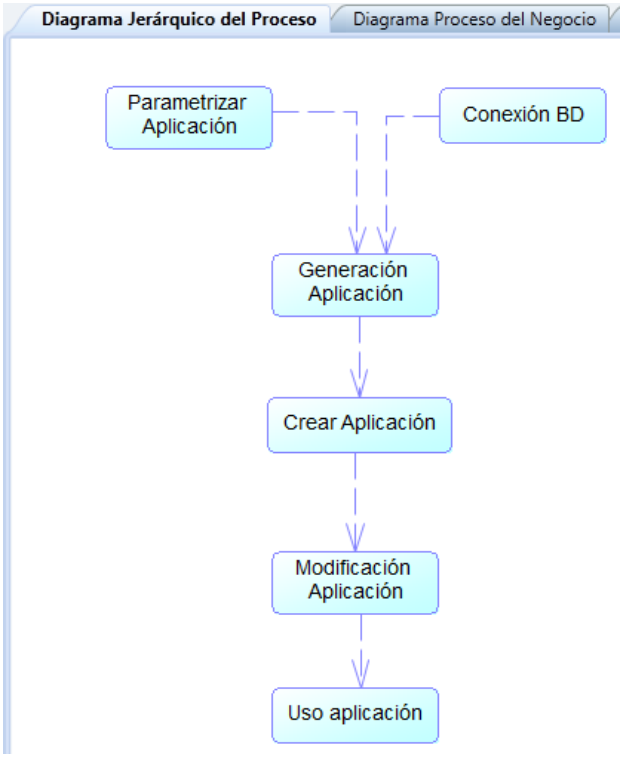



**Ilustración 68. Diagrama Jerárquico de Procesos FredyCRUD**

Se inicia con un gran proceso de "Parametrizar la Aplicación Generada", donde se configura la nueva aplicación, como el estilo de interfaz para el usuario, el tipo de aplicación web o de escritorio, el esquema de seguridad de la nueva aplicación y los roles de sus usuarios, el sistema de navegación de la aplicación, el lenguaje de programación del código generado, entre otros aspectos. Para poder iniciar con el proceso de generación de código, con anterioridad es necesario establecer el ingreso y conexión al diccionario de datos de mySQL, para obtener el listado de todas las bases de datos del sistema, para así poder escoger la base de datos a trabajar, esta programación se encuentra en el macro-proceso de "Conexión a la Base de Datos". Es así que el usuario puede invocar el "core" de la aplicación, es decir el motor de generación de código fuente de la nueva aplicación. Es así que el próximo macroproceso es el de "Crear la Aplicación" de acuerdo con el lenguaje de programación destino escogido previamente por el usuario. La creación de la aplicación debe tener en cuenta aspectos para procesos posteriores, es decir que el código generado, sea susceptible de modificar manualmente por parte del equipo de desarrollo y por último, lógicamente que se pueda utilizar la aplicación generada, previo proceso de compilación, interpretación o traducción a lenguaje ejecutable. Cabe

resaltar, que los dos últimos procesos del diagrama jerárquico no son propios de la aplicación desarrollada por este proyecto, son procesos manuales, aunque sí los tiene en cuenta.

Los **Diagramas de Procesos del Negocio** (o diagrama de flujo del proceso) suministra una vista gráfica del flujo de control (la secuencia de ejecución) o flujo de datos (intercambio de datos) entre procesos en cualquier nivel del sistema.

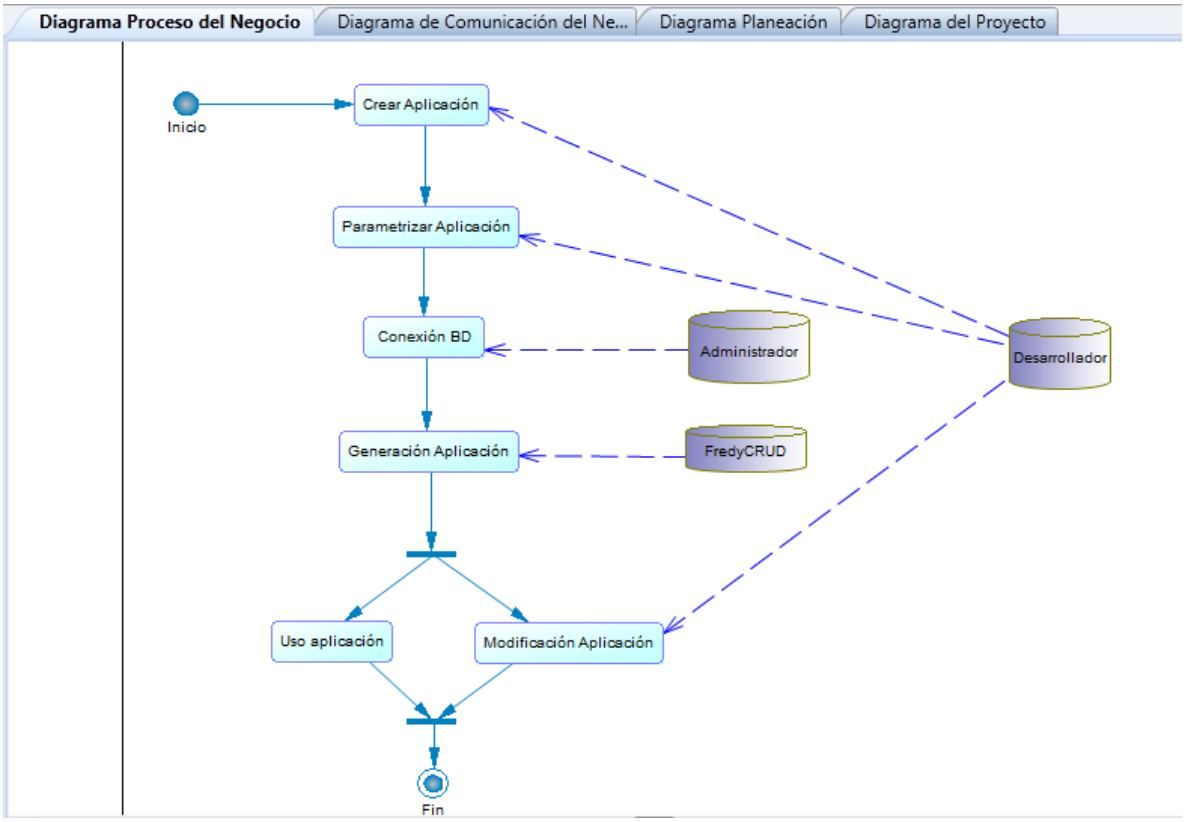



Ilustración 69. Diagrama de Procesos del Negocio

La ejecución del software inicia con la creación de la nueva aplicación a cargo del desarrollador, a continuación el mismo desarrollador configura la nueva aplicación. Los accesos a la base de datos son proporcionados por el Administrador de la Base de Datos (dbA) ya que se necesita privilegios de administrador para poder obtener el diccionario de datos mySQL. De forma automática, el software (FredyCRUD) invoca su motor de generación para obtener el código fuente de la nueva aplicación. Finalmente, el desarrollador puede modificar el código generado para preparar el uso final de la aplicación generada.

El **Diagrama de Servicio del Proceso** suministra una visión gráfica de los servicios, operaciones e interfaces disponibles en el sistema.

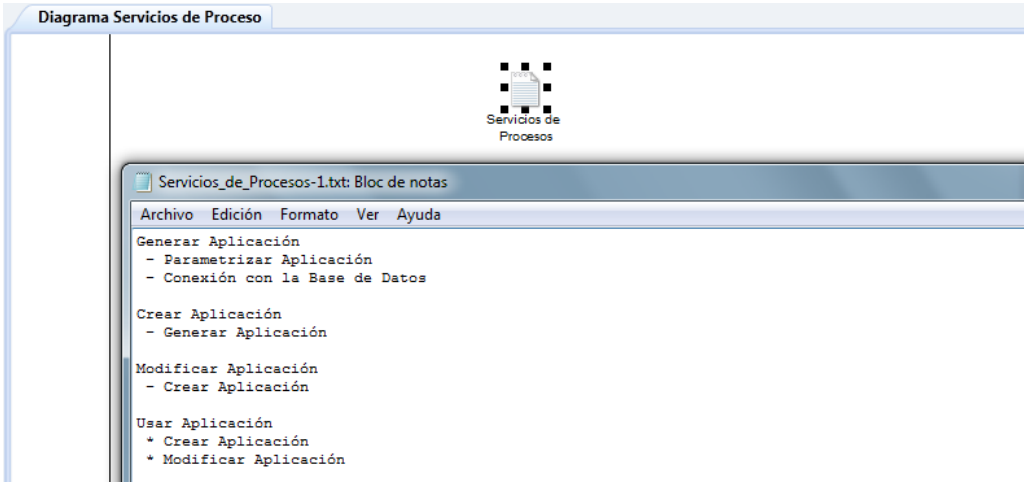



Ilustración 70. Análisis de Servicios de Proceso

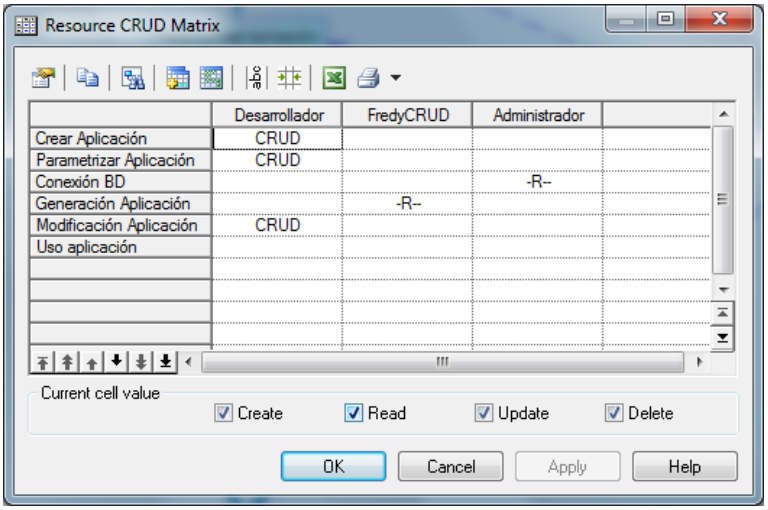



Ilustración 71. Matriz CRUD de FredyCRUD

**Simular un Modelo de Proceso de Negocios** con SIMUL8, ayuda a entender el rendimiento esperado de los procesos de negocio antes de su implementación, suministrando métricas de análisis útiles y asistencia en la optimización del proceso de negocios.

```
<SIMUL8XML>
   <SolutionXML>
      <ObjectID>72BDF2AC-7520-411A-88A3-2F85E98B663E</ObjectID>
      <FolderPath>Modelo BPMN\DIAG_SEP\Diagrama Proceso del Negocio</FolderPath>
      <RightShift>0</RightShift>
      <DownShift>0</DownShift>
      <DiagramScale>100</DiagramScale>
      <PageScale>100</PageScale>
   </SolutionXML>
   <SimulationParameters>
      <Units>
         <Time>0</Time>
```

```
    </Units>
    <Finance>
       <OverheadCost>0</OverheadCost>
       <OverheadRevenue>0</OverheadRevenue>
    </Finance>
    <ResultsCollectionPeriod>2400</ResultsCollectionPeriod>
    <ResultsSummary>
       <ResultItem>
          <ObjectID>1</ObjectID>
          <Code1>1</Code1>
          <Description>Number Entered</Description>
       </ResultItem>
       <ResultItem>
          <ObjectID>1</ObjectID>
          <Code1>2</Code1>
          <Description>Number Lost</Description>
       </ResultItem>
       <ResultItem>
          <ObjectID>2</ObjectID>
          <Code1>1</Code1>
          <Description>Number Completed Jobs</Description>
       </ResultItem>
       <ResultItem>
          <ObjectID>2</ObjectID>
          <Description>Waiting %</Description>
          <Maxlimit>100</Maxlimit>
       </ResultItem>
       <ResultItem>
          <ObjectID>2</ObjectID>
          <Code2>1</Code2>
          <Description>Working %</Description>
          <Maxlimit>100</Maxlimit>
       </ResultItem>
       <ResultItem>
          <ObjectID>2</ObjectID>
          <Code2>2</Code2>
          <Description>Blocked %</Description>
          <Maxlimit>100</Maxlimit>
       </ResultItem>
       <ResultItem>
          <ObjectID>3</ObjectID>
          <Code1>1</Code1>
          <Description>Utilization %</Description>
          <Maxlimit>100</Maxlimit>
       </ResultItem>
       <ResultItem>
          <ObjectID>4</ObjectID>
          <Code1>1</Code1>
          <Description>Number Completed Jobs</Description>
       </ResultItem>
       <ResultItem>
          <ObjectID>4</ObjectID>
          <Description>Waiting %</Description>
          <Maxlimit>100</Maxlimit>
       </ResultItem>
       <ResultItem>
          <ObjectID>4</ObjectID>
          <Code2>1</Code2>
          <Description>Working %</Description>
          <Maxlimit>100</Maxlimit>
       </ResultItem>
       <ResultItem>
          <ObjectID>4</ObjectID>
          <Code2>2</Code2>
          <Description>Blocked %</Description>
          <Maxlimit>100</Maxlimit>
       </ResultItem>
       <ResultItem>
          <ObjectID>5</ObjectID>
          <Code1>1</Code1>
          <Description>Number Completed Jobs</Description>
       </ResultItem>
       <ResultItem>
          <ObjectID>5</ObjectID>
          <Description>Waiting %</Description>
          <Maxlimit>100</Maxlimit>
       </ResultItem>
       <ResultItem>
          <ObjectID>5</ObjectID>
          <Code2>1</Code2>
          <Description>Working %</Description>
          <Maxlimit>100</Maxlimit>
       </ResultItem>
       <ResultItem>
          <ObjectID>5</ObjectID>
          <Code2>2</Code2>
          <Description>Blocked %</Description>
          <Maxlimit>100</Maxlimit>
       </ResultItem>
       <ResultItem>
          <ObjectID>6</ObjectID>
          <Code1>1</Code1>
          <Description>Utilization %</Description>
          <Maxlimit>100</Maxlimit>
```

```
        </ResultItem>
        <ResultItem>
           <ObjectID>7</ObjectID>
           <Code1>1</Code1>
           <Description>Number Completed Jobs</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>7</ObjectID>
           <Description>Waiting %</Description>
           <Maxlimit>100</Maxlimit>
        </ResultItem>
        <ResultItem>
           <ObjectID>7</ObjectID>
           <Code2>1</Code2>
           <Description>Working %</Description>
           <Maxlimit>100</Maxlimit>
        </ResultItem>
        <ResultItem>
           <ObjectID>7</ObjectID>
           <Code2>2</Code2>
           <Description>Blocked %</Description>
           <Maxlimit>100</Maxlimit>
        </ResultItem>
        <ResultItem>
           <ObjectID>8</ObjectID>
           <Code1>1</Code1>
           <Description>Utilization %</Description>
           <Maxlimit>100</Maxlimit>
        </ResultItem>
        <ResultItem>
           <ObjectID>9</ObjectID>
           <Code1>1</Code1>
           <Description>Number Completed Jobs</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>9</ObjectID>
           <Description>Waiting %</Description>
           <Maxlimit>100</Maxlimit>
        </ResultItem>
        <ResultItem>
           <ObjectID>9</ObjectID>
           <Code2>1</Code2>
           <Description>Working %</Description>
           <Maxlimit>100</Maxlimit>
        </ResultItem>
        <ResultItem>
           <ObjectID>9</ObjectID>
           <Code2>2</Code2>
           <Description>Blocked %</Description>
           <Maxlimit>100</Maxlimit>
        </ResultItem>
        <ResultItem>
           <ObjectID>10</ObjectID>
           <Code1>1</Code1>
           <Description>Number Completed Jobs</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>10</ObjectID>
           <Description>Waiting %</Description>
           <Maxlimit>100</Maxlimit>
        </ResultItem>
        <ResultItem>
           <ObjectID>10</ObjectID>
           <Code2>1</Code2>
           <Description>Working %</Description>
           <Maxlimit>100</Maxlimit>
        </ResultItem>
        <ResultItem>
           <ObjectID>10</ObjectID>
           <Code2>2</Code2>
           <Description>Blocked %</Description>
           <Maxlimit>100</Maxlimit>
        </ResultItem>


        <ResultItem>
           <ObjectID>13</ObjectID>
           <Code1>1</Code1>
           <Description>Number Completed</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>13</ObjectID>
           <Code1>3</Code1>
           <Description>Minimum Time in System</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>13</ObjectID>
           <Description>Average Time in System</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>13</ObjectID>
           <Code1>7</Code1>
           <Description>Maximum Time in System</Description>
        </ResultItem>
```

```
        <ResultItem>
           <ObjectID>13</ObjectID>
           <Code1>9</Code1>
           <Description>St Dev of </Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>16</ObjectID>
           <Code1>22</Code1>
           <Description>Current Contents</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>16</ObjectID>
           <Code1>1</Code1>
           <Description>Average queue size</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>16</ObjectID>
           <Code1>2</Code1>
           <Description>Maximum queue size</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>16</ObjectID>
           <Code1>23</Code1>
           <Description>Items Entered</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>16</ObjectID>
           <Code1>5</Code1>
           <Description>Average Queuing Time</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>16</ObjectID>
           <Code1>8</Code1>
           <Description>Number of non zero queuing times</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>17</ObjectID>
           <Code1>22</Code1>
           <Description>Current Contents</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>17</ObjectID>
           <Code1>1</Code1>
           <Description>Average queue size</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>17</ObjectID>
           <Code1>2</Code1>
           <Description>Maximum queue size</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>17</ObjectID>
           <Code1>23</Code1>
           <Description>Items Entered</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>17</ObjectID>
           <Code1>5</Code1>
           <Description>Average Queuing Time</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>17</ObjectID>
           <Code1>8</Code1>
           <Description>Number of non zero queuing times</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>18</ObjectID>
           <Code1>22</Code1>
           <Description>Current Contents</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>18</ObjectID>
           <Code1>1</Code1>
           <Description>Average queue size</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>18</ObjectID>
           <Code1>2</Code1>
           <Description>Maximum queue size</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>18</ObjectID>
           <Code1>23</Code1>
           <Description>Items Entered</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>18</ObjectID>
           <Code1>5</Code1>
           <Description>Average Queuing Time</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>18</ObjectID>
           <Code1>8</Code1>
           <Description>Number of non zero queuing times</Description>
```

```
        </ResultItem>
        <ResultItem>
           <ObjectID>19</ObjectID>
           <Code1>22</Code1>
           <Description>Current Contents</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>19</ObjectID>
           <Code1>1</Code1>
           <Description>Average queue size</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>19</ObjectID>
           <Code1>2</Code1>
           <Description>Maximum queue size</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>19</ObjectID>
           <Code1>23</Code1>
           <Description>Items Entered</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>19</ObjectID>
           <Code1>5</Code1>
           <Description>Average Queuing Time</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>19</ObjectID>
           <Code1>8</Code1>
           <Description>Number of non zero queuing times</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>21</ObjectID>
           <Code1>22</Code1>
           <Description>Current Contents</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>21</ObjectID>
           <Code1>1</Code1>
           <Description>Average queue size</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>21</ObjectID>
           <Code1>2</Code1>
           <Description>Maximum queue size</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>21</ObjectID>
           <Code1>23</Code1>
           <Description>Items Entered</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>21</ObjectID>
           <Code1>5</Code1>
           <Description>Average Queuing Time</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>21</ObjectID>
           <Code1>8</Code1>
           <Description>Number of non zero queuing times</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>20</ObjectID>
           <Code1>22</Code1>
           <Description>Current Contents</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>20</ObjectID>
           <Code1>1</Code1>
           <Description>Average queue size</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>20</ObjectID>
           <Code1>2</Code1>
           <Description>Maximum queue size</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>20</ObjectID>
           <Code1>23</Code1>
           <Description>Items Entered</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>20</ObjectID>
           <Code1>5</Code1>
           <Description>Average Queuing Time</Description>
        </ResultItem>
        <ResultItem>
           <ObjectID>20</ObjectID>
           <Code1>8</Code1>
           <Description>Number of non zero queuing times</Description>
        </ResultItem>
     </ResultsSummary>
   </SimulationParameters>
</SIMUL8XML>
```

Tabla 13. Parámetros de simulación en SIMUL8 para los proceso de negocio de la aplicación

### 6.3.1.2. DIAGRAMAS DE LA CAPA DEL NEGOCIO

Los diagramas en la capa del negocio del modelo de arquitectura empresarial permiten analizar la organización en términos de unidades de negocio, procesos y funciones.

El **Organigrama** permite una visión gráfica de la organización en estructura de árbol y ayuda a analizar y mostrar la relación entre las unidades de la organización (divisiones, grupos, equipos, etc.), individuos y roles.

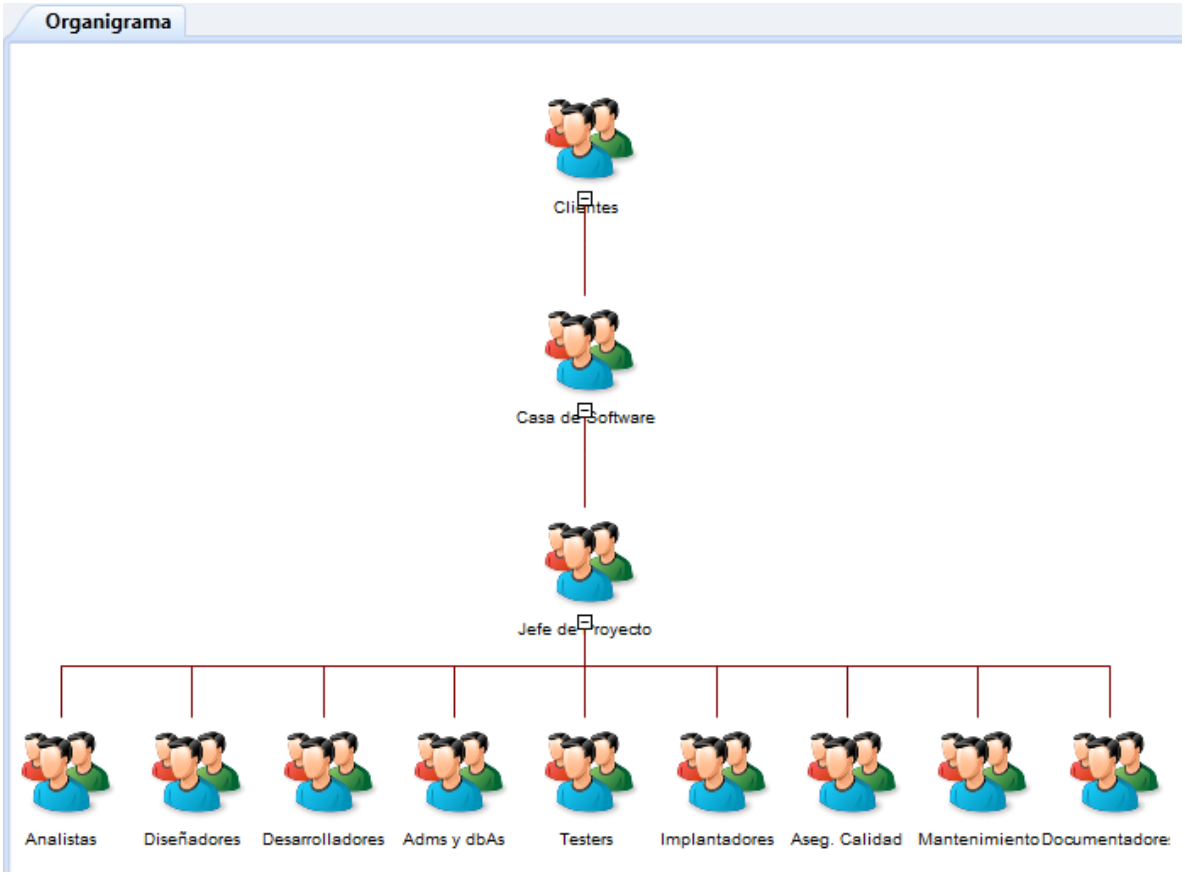



Ilustración 72. Organigrama de un esquema de adquisición de un nuevo Sistema de Información

El **Diagrama de Comunicación del Negocio** suministra una vista gráfica de la organización y permite analizar las relaciones, los flujos, y otras conexiones entre las funciones del negocio, las unidades organizacionales, los roles y los sitios.

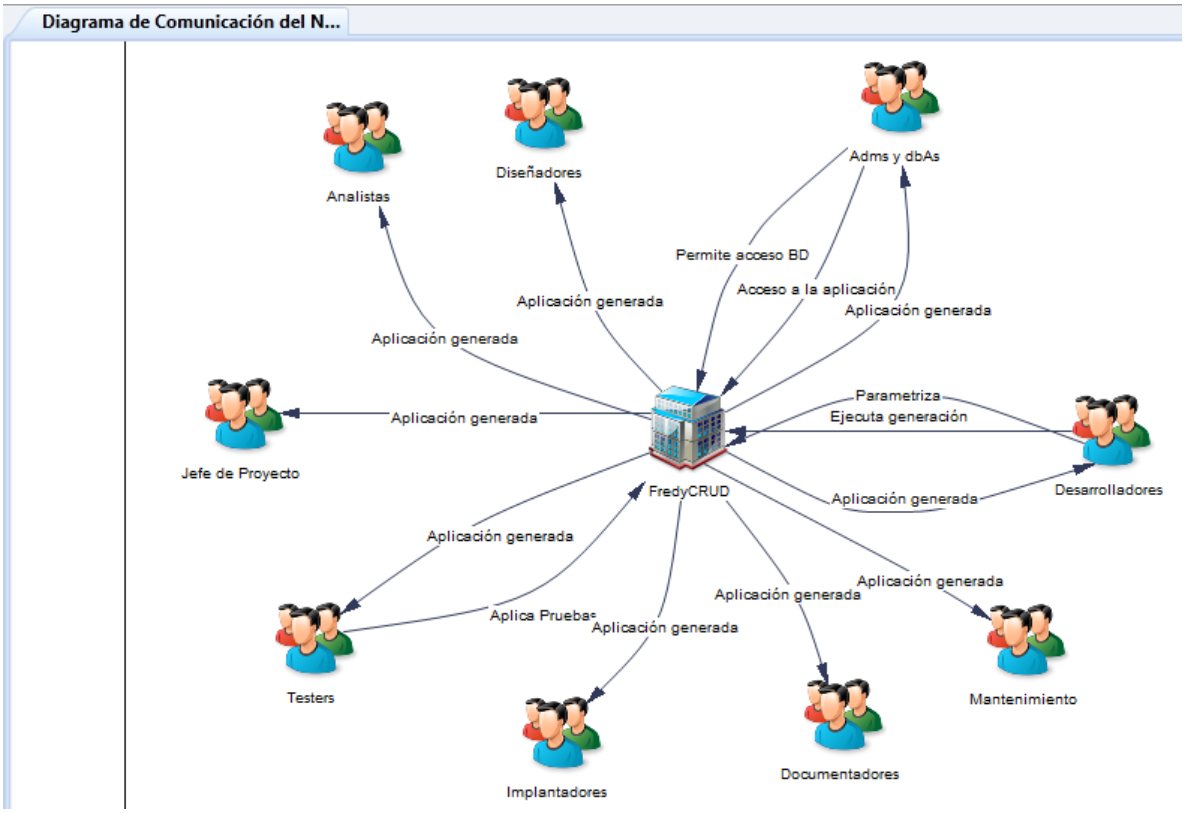



Ilustración 73. Diagrama de Comunicación del Negocio

El **Mapa de Procesos** suministra una vista gráfica de la arquitectura del negocio y permite identificar las funciones del negocio y los procesos de alto nivel, independientes de las personas y unidades de negocio quienes los cumplen.

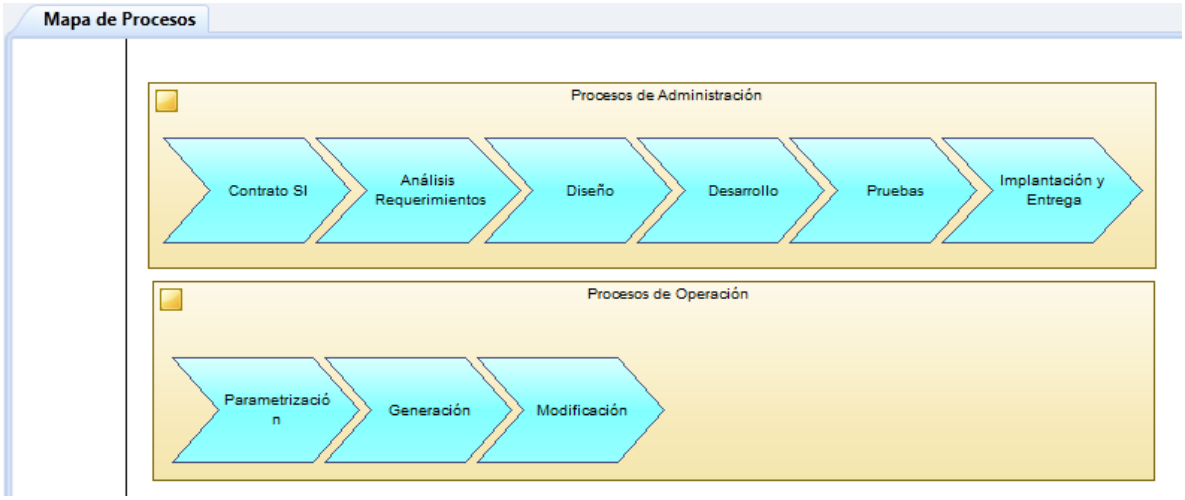



Ilustración 74. Mapa de Procesos

Un **Diagrama de Planeación de la Ciudad** suministra una vista gráfica de la gran visión de la arquitectura empresarial, usando la metáfora de la planeación de la infraestructura de una ciudad que representa la organización de sistemas, aplicaciones, etc. en áreas arquitectónicas.

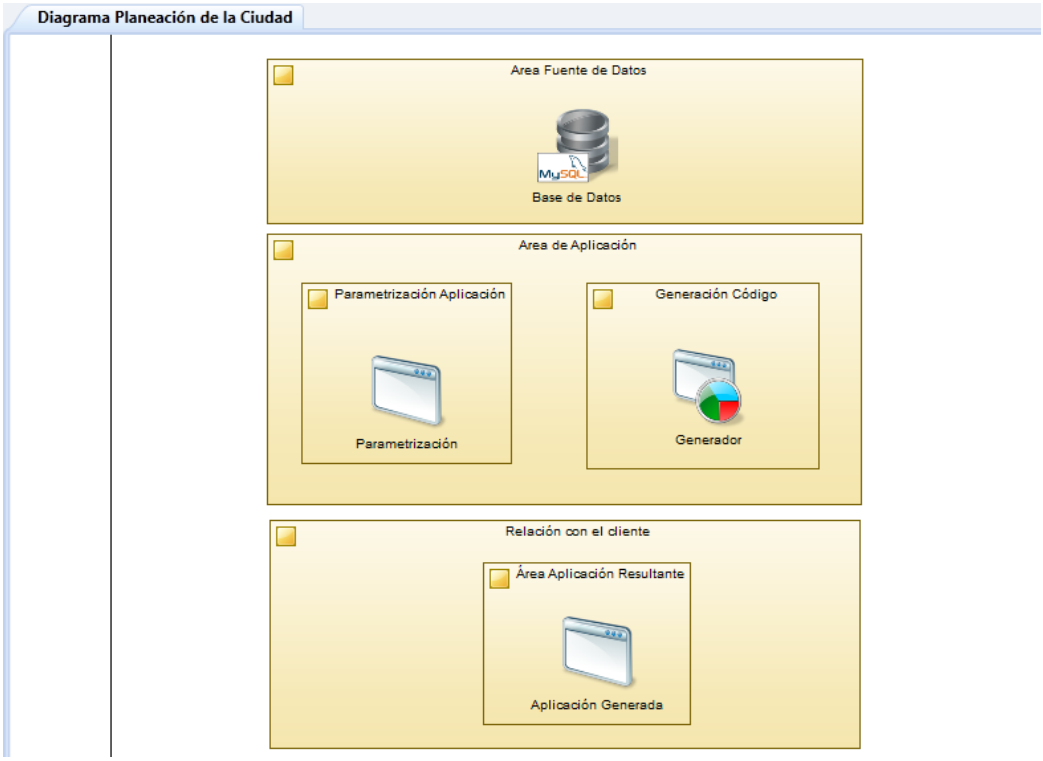



Ilustración 75. Diagrama de Planeación de la Ciudad

Las unidades de Administración de la Arquitectura Empresarial (EAM), son:

- Las Unidades de la Organización (EAM) representan un grupo de personas o de otras unidades organizacionales
- Una persona (EAM) representa un individuo quien mantiene una posición dentro de una organización
- Un rol (EAM) es un conjunto de responsabilidades que son ejercidas por una persona o una unidad de la organización
- Un área arquitectónica (EAM) es un objeto abstracto que puede contener y agrupar a otros objetos
- Un sitio (EAM) es una ubicación física que puede contener y agrupar gente, organizaciones y otros objetos
- Una función del negocio (EAM) es una agregación de procesos y sub-funciones relacionadas
- Un proceso (EAM) es una orden de tareas o actividades que puede incluir operaciones manuales y/o automáticas.
- Un flujo de negocios (EAM) es un enlace orientado que es usado para conectar objetos y puede contener documentos

#### 6.3.1.3. DIAGRAMAS DE LA CAPA DE APLICACIÓN

Los diagramas de la capa de aplicación permiten modelar la arquitectura de los sistemas para identificar las aplicaciones y sus principales componentes, y para analizar sus interacciones y cómo ellas implementan procesos y funciones del negocio.

Un **Diagrama de Arquitectura de la Aplicación** suministra una vista gráfica de alto nivel de la arquitectura de la aplicación y permite identificar las aplicaciones, sub-aplicaciones, componentes, bases de datos, servicios, etc. y sus interacciones.

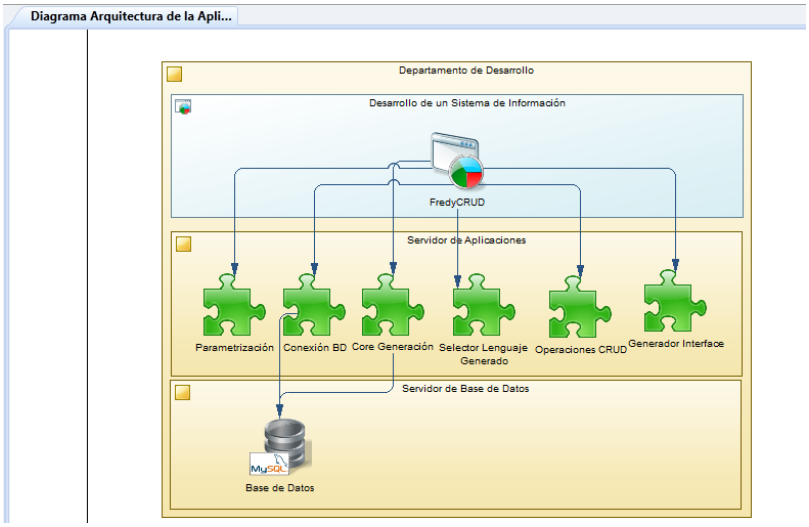



Ilustración 76. Diagrama Arquitectura de la Aplicación

Un **Diagrama Orientado al Servicio** suministra una vista gráfica del negocio y de los servicios de la aplicación y las relaciones entre ellos, y permite asociar aplicaciones y otros objetos de la capa de aplicación con los servicios del negocio y los proceso para asistir con el diseño SOA.

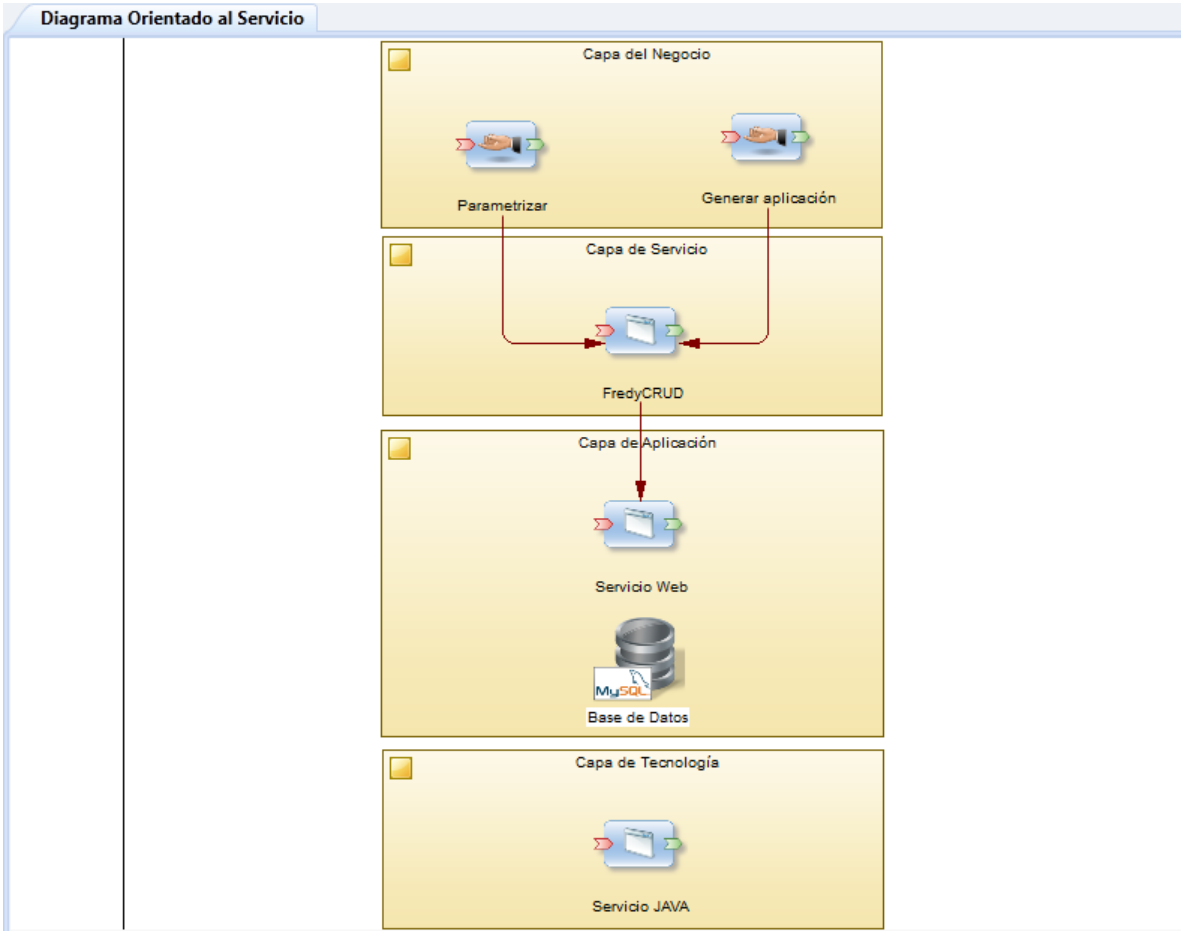



Ilustración 77. Diagrama Orientado al Servicio

### 6.3.1.4. DIAGRAMAS DE LA CAPA TECNOLÓGICA

Un diagrama de la infraestructura tecnológica suministra una vista gráfica de alto nivel de la arquitectura física requerida para soportar la arquitectura de la aplicación.

Un diagrama de infraestructura tecnológica es el único diagrama en la capa tecnológica.

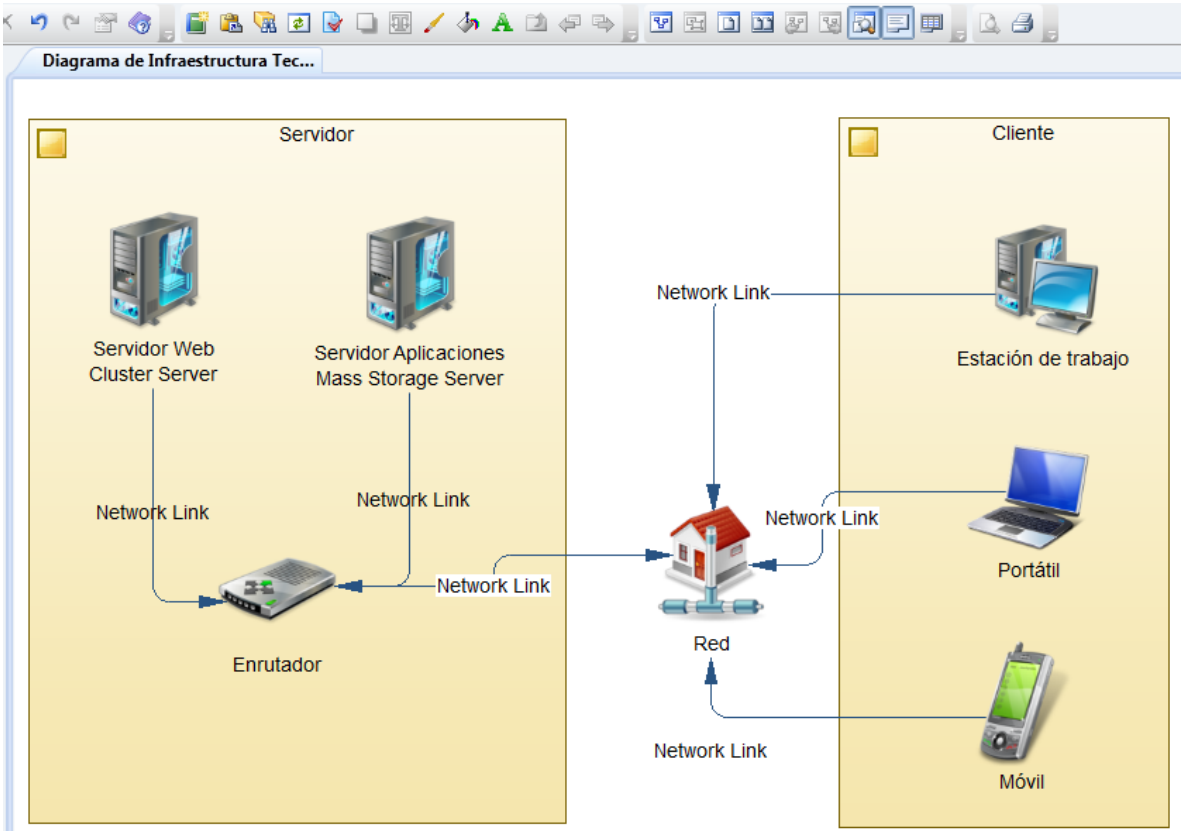



Ilustración 78. Diagrama de Infraestructura Tecnológica

### 6.3.1.5. MODELAMIENTO ORIENTADO A OBJETOS (OOM)

Un modelo orientado a objetos (OOM) permite analizar un sistema de información a través de los casos de uso, los análisis estructurales y de comportamiento, y en términos de desarrollo, usar el Lenguaje Unificado de Modelamiento (UML). Se puede modelar, aplicar ingeniería inversa y generar código para Java, .NET y otros lenguajes de programación.

Un **Diagrama de Casos de Uso** es un diagrama UML que permite una vista gráfica de los requerimientos del sistema y permite identificar cómo los usuarios interactúan con éste.

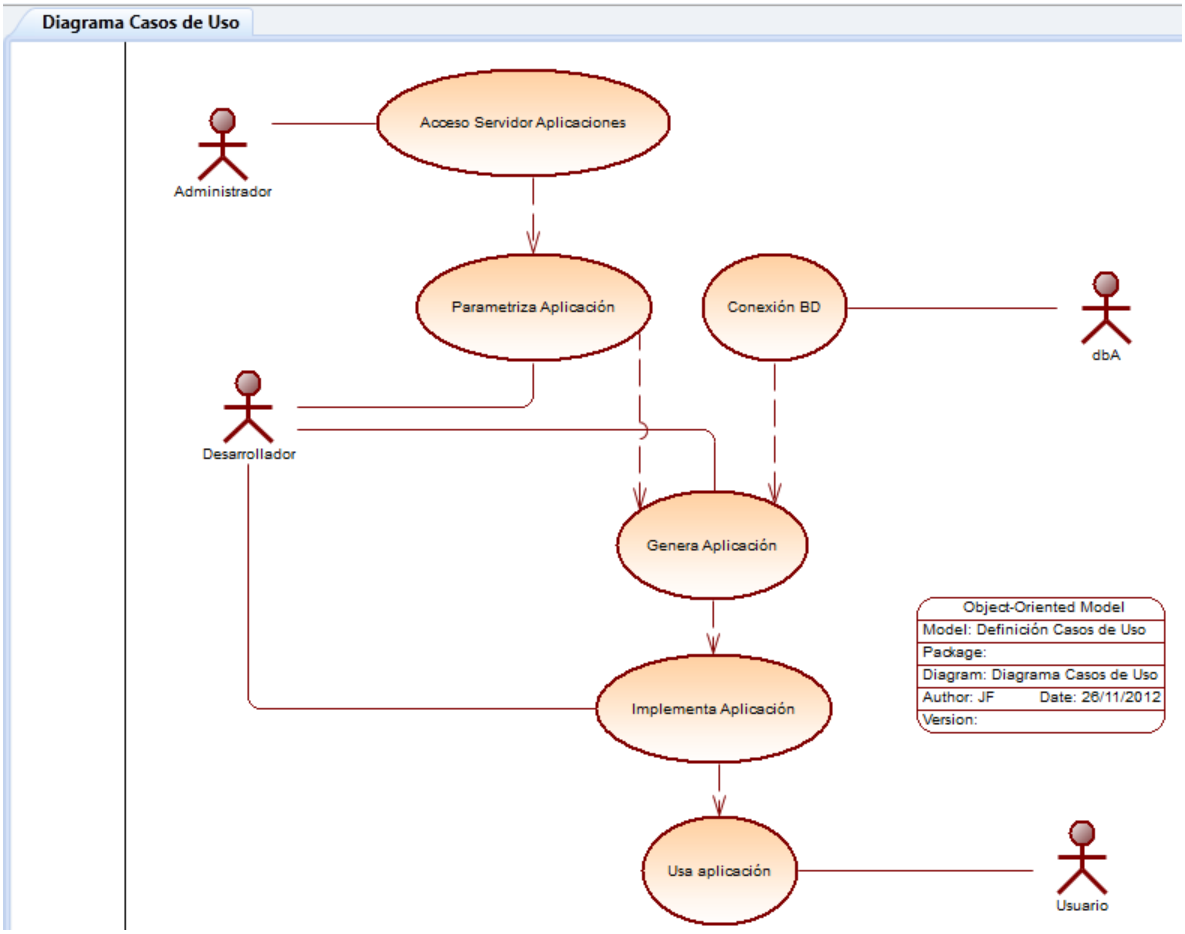



Ilustración 79. Diagrama General de Casos de Uso

Los **Diagramas Estructurales** permiten modelar la estructura estática del sistema. Existen tres tipos de diagramas para modelar el sistema de esta forma, cada uno de los cuales ofrece una vista diferente de los objetos y sus relaciones:

- **Diagramas de Clases**: Un diagrama de clases es un diagrama UML que suministra una vista gráfica de las clases, interfaces y paquetes que componen un sistema y la relación entre ellos. El diagrama de clases simplifica la interacción de objetos en el sistema que se está modelando. Los diagramas de clases expresan la estructura estática de un sistema en términos de clases y relaciones entre esas clases. Una clase describe un conjunto de objetos y una asociación describe un conjunto de enlaces; los objetos son instancias de las clases y los enlaces son sus instancias de asociación. Un diagrama de clases no expresan nada en específico acerca de los enlaces de un objeto dado, pero describen, en forma abstracta, el enlace potencial de un objeto a otro.

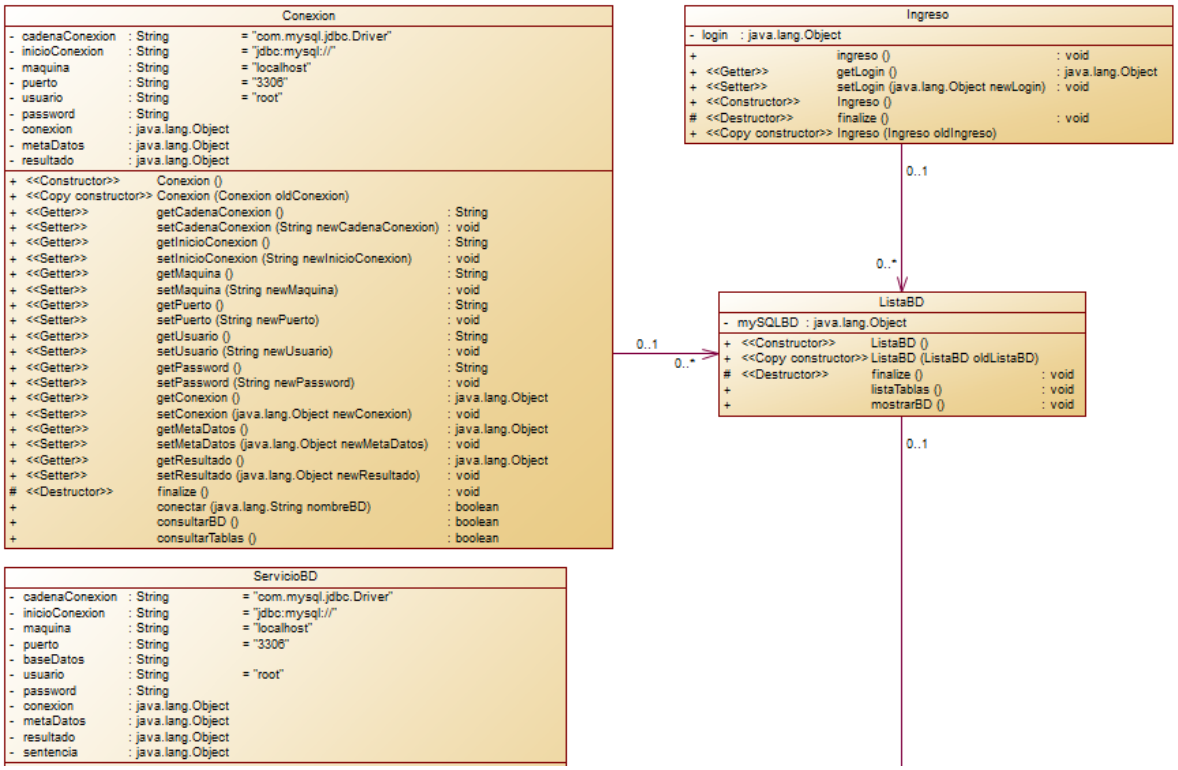



**Ilustración 80. Diagrama de Clases aplicación FredyCRUD: Conexión, Ingreso y ListaBD**

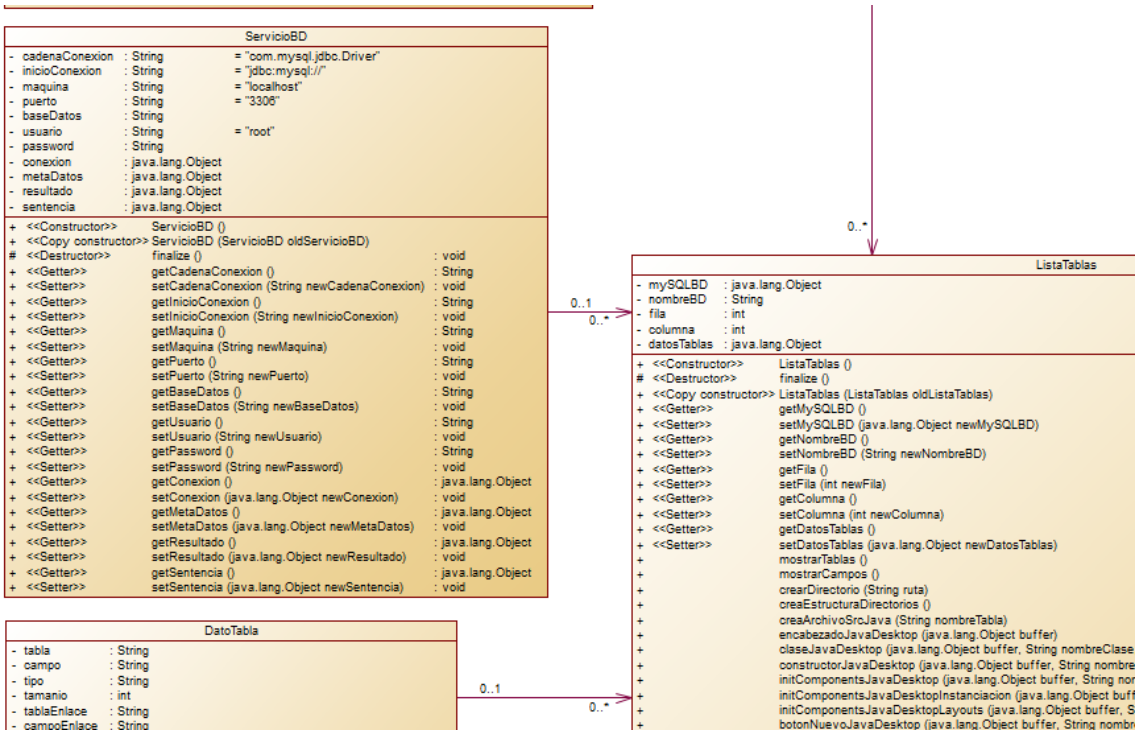



**Ilustración 81. Diagrama de Clases aplicación FredyCRUD: ServicioBD**

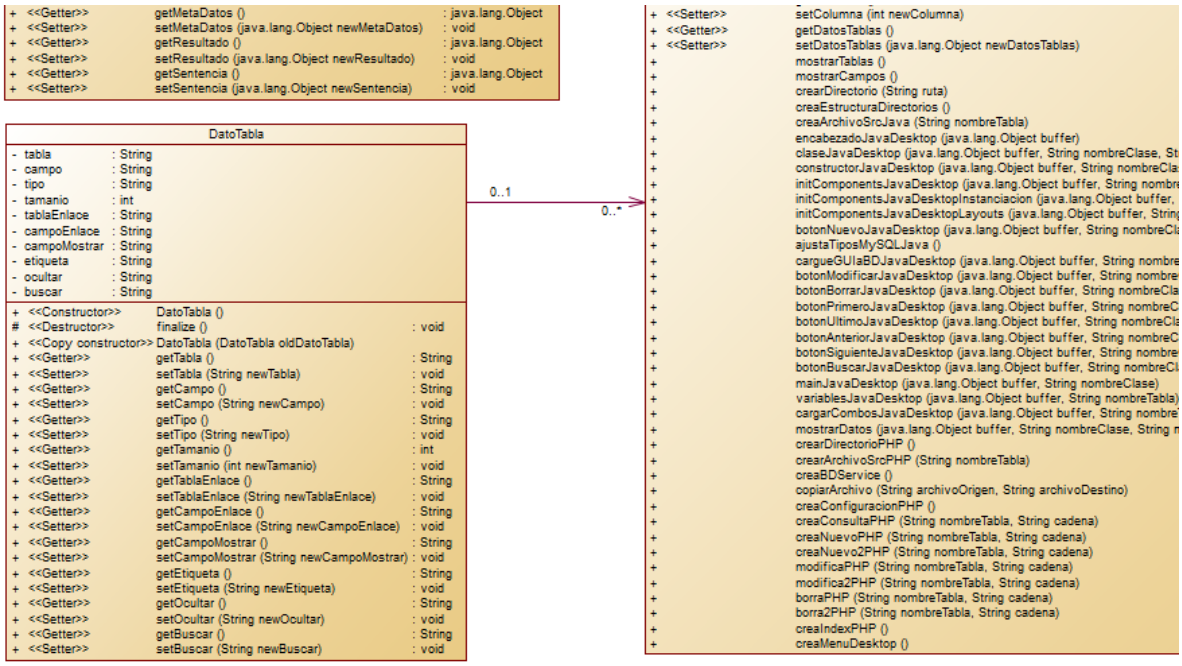



**Ilustración 82. Diagrama de Clases aplicación FredyCRUD: DatoTabla**

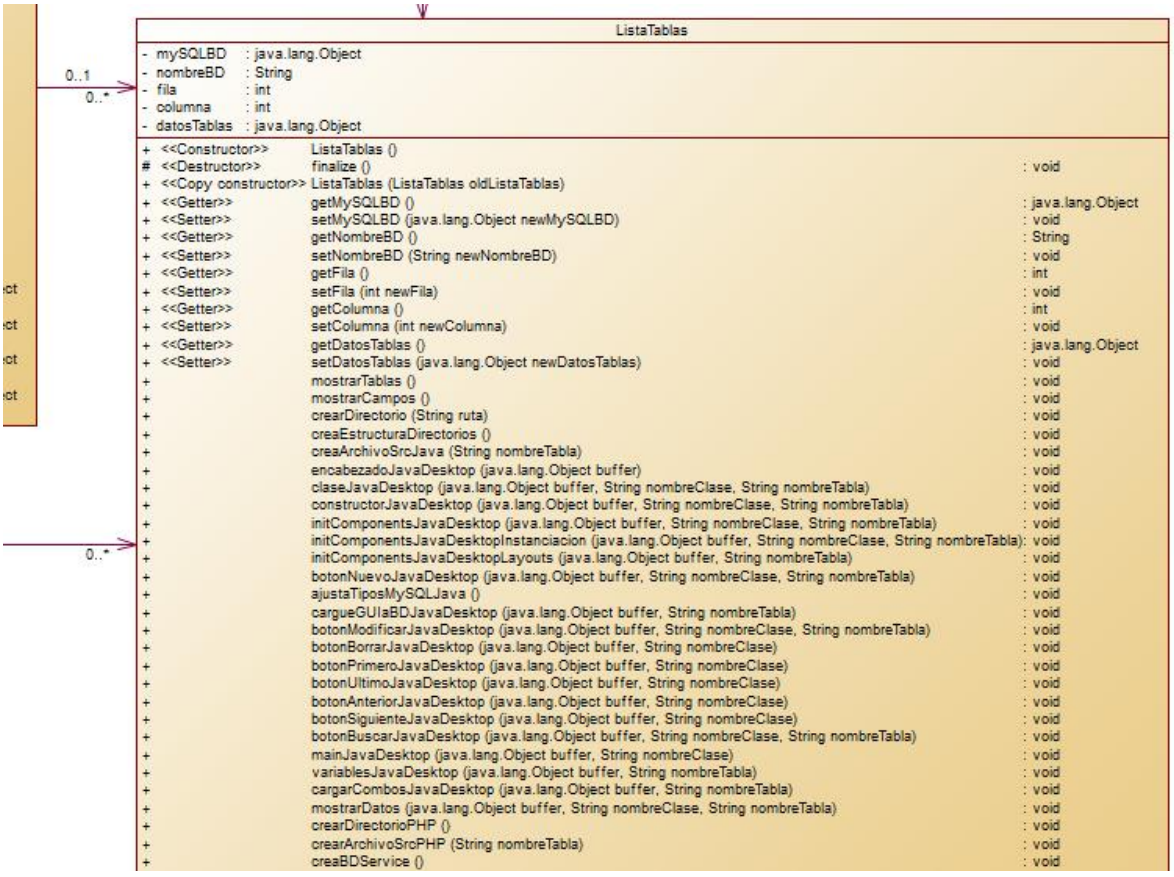



**Ilustración 83. Diagrama de Clases aplicación FredyCRUD: ListaTablas – Atributos**


**Ilustración 84. Diagrama de Clases aplicación FredyCRUD: ListaTablas – Métodos**

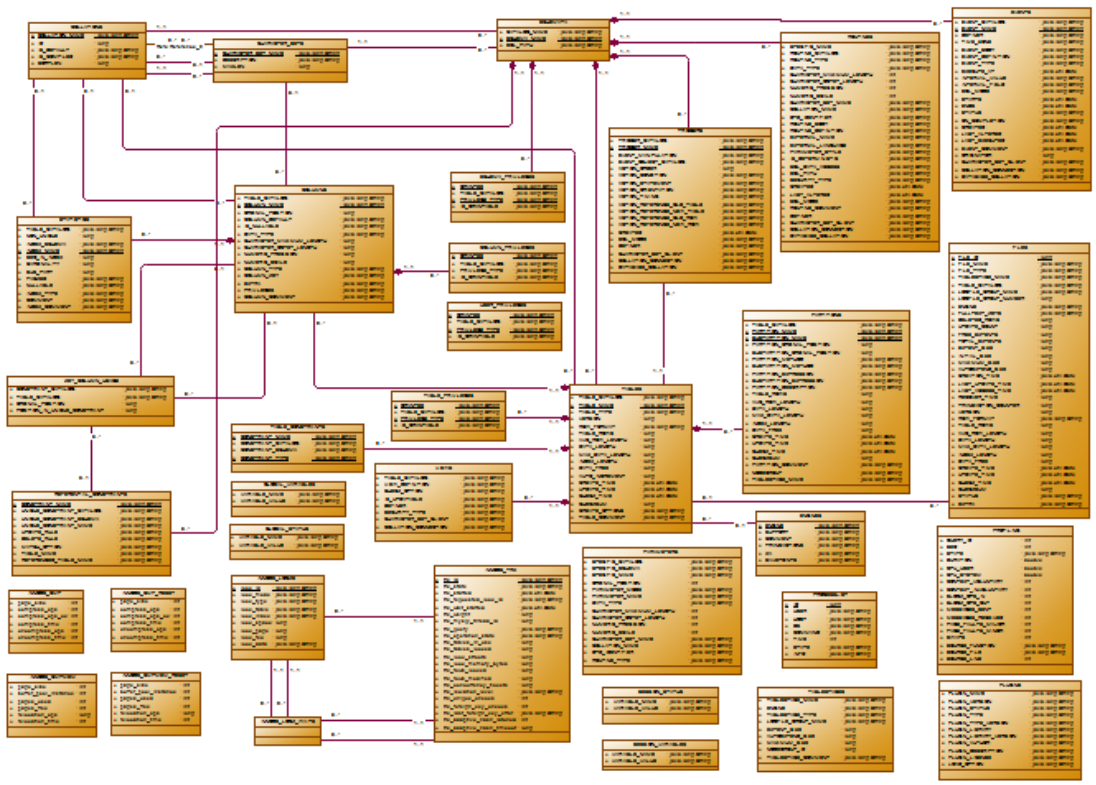


**Ilustración 85. Diagrama general de Clases para el diccionario de datos mySQL**

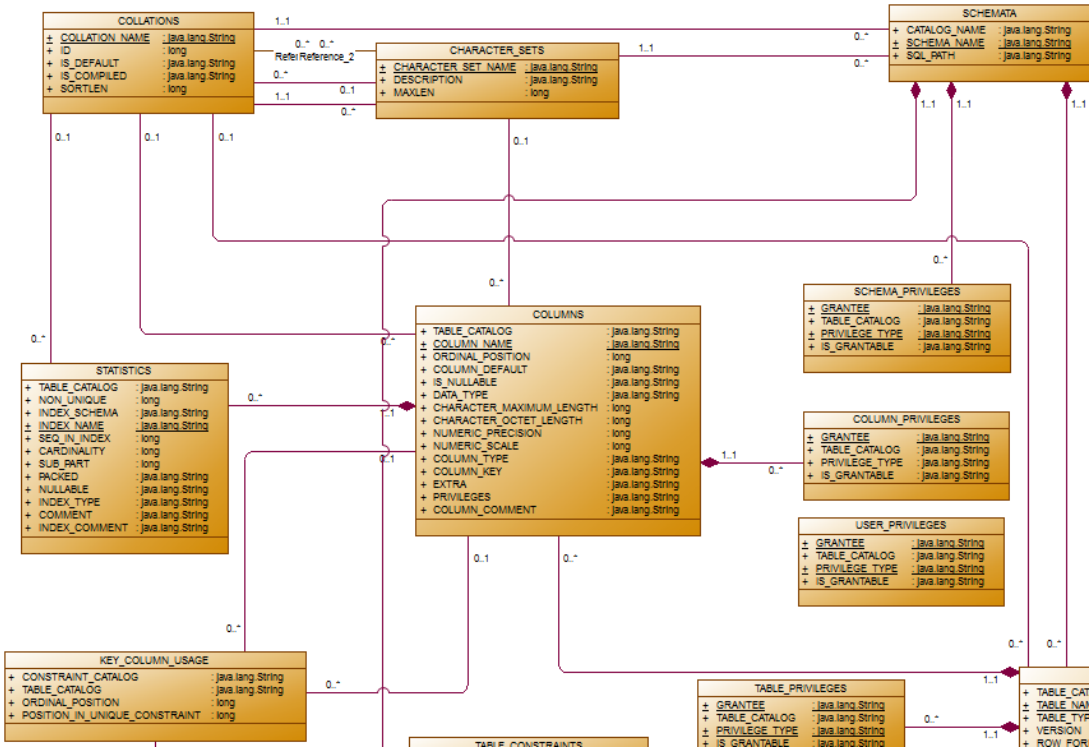



**Ilustración 86. Diagrama de clases diccionario datos mySQL (Vista detallada izquierda-superior)**

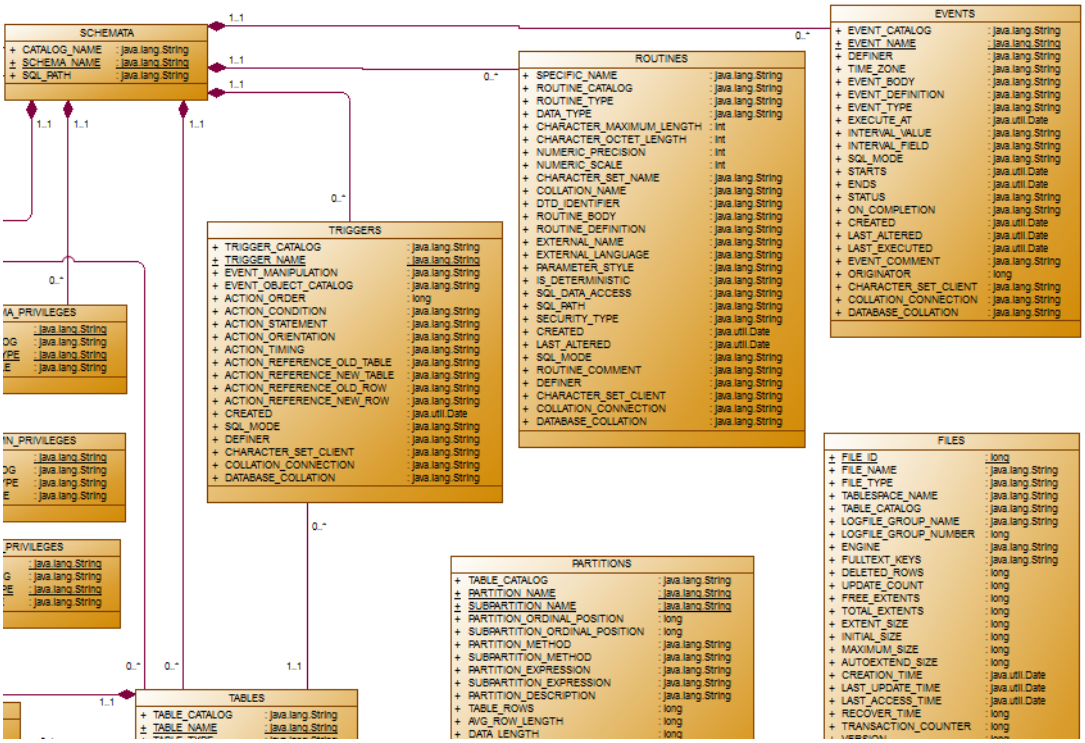



**Ilustración 87. Diagrama de clases diccionario datos mySQL (Vista detallada derecha-superior)**

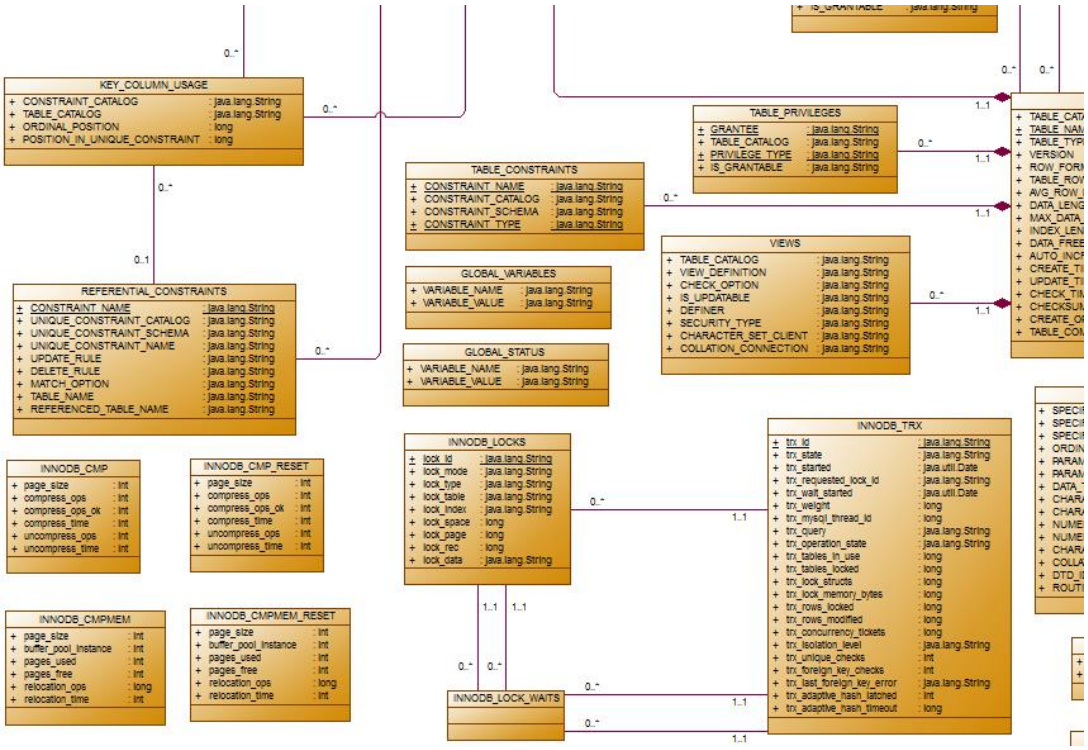



**Ilustración 88. Diagrama de clases diccionario datos mySQL (Vista detallada izquierda-inferior)**

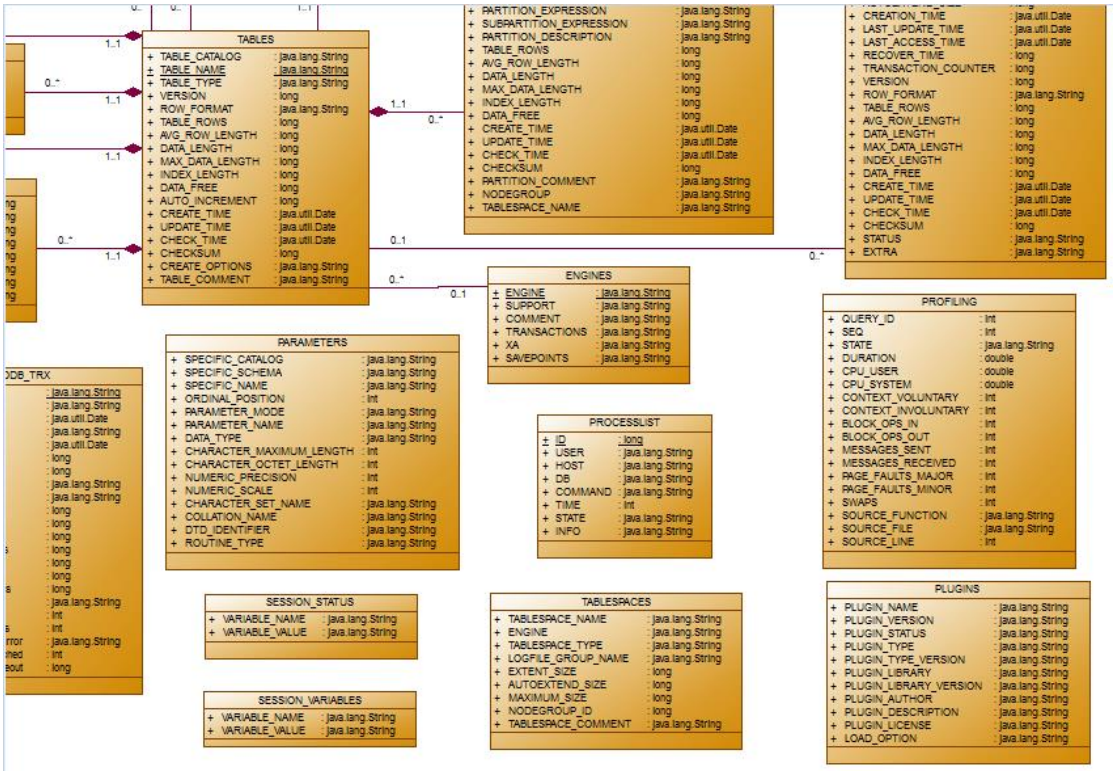



**Ilustración 89. Diagrama de clases diccionario datos mySQL (Vista detallada derecha-inferior)**

- **Diagrama de Estructura Compuesta**: Un diagrama de estructura compuesta es un diagrama UML que suministra una vista gráfica de las clases, interfaces y paquetes que componen un sistema, incluyendo los puertos y las partes que describen las estructuras internas. Un diagrama de estructura compuesta ejecuta un rol similar a un diagrama de clases, pero permite ir más al detalle para describir la estructura interna de múltiples clases y mostrar las interacciones entre ellas. Se puede representar gráficamente clases y partes internas, y mostrar asociaciones entre y dentro de las clases.

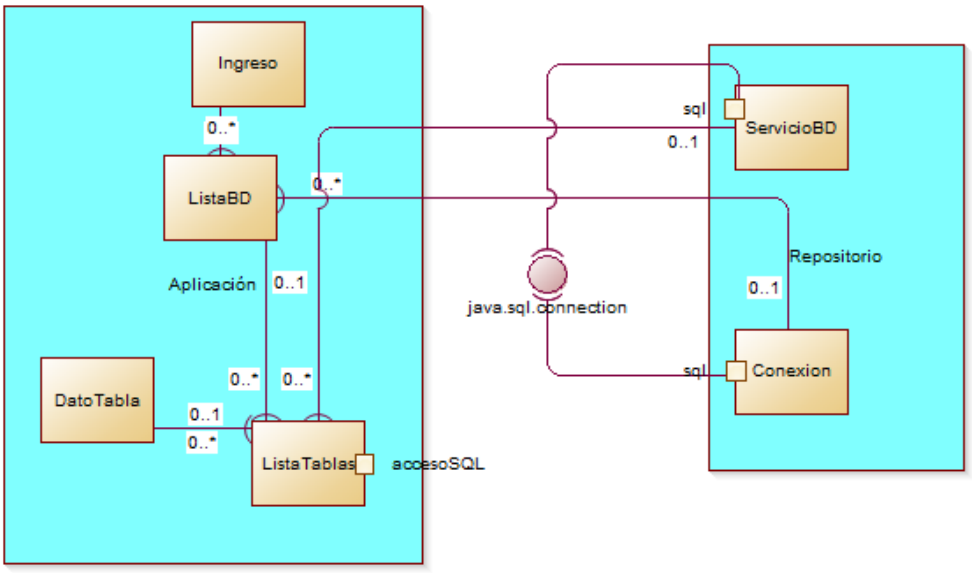



Ilustración 90. Diagrama Estructura Compuesta FredyCRUD

- **Diagrama de objetos**: Un diagrama de objetos es un diagrama UML que suministra una vista gráfica de la estructura de un sistema a través de instancias concretas de clases (objetos), asociaciones (enlaces de instancias) y dependencias. Como un diagrama de instancias, el diagrama de objetos muestra un ejemplo de estructuras de datos con valores de datos que corresponden a una situación detallada del sistema en un punto particular en el tiempo. El diagrama de objetos pueden ser usados para propósitos de análisis: restricciones entre clases que no están representadas.

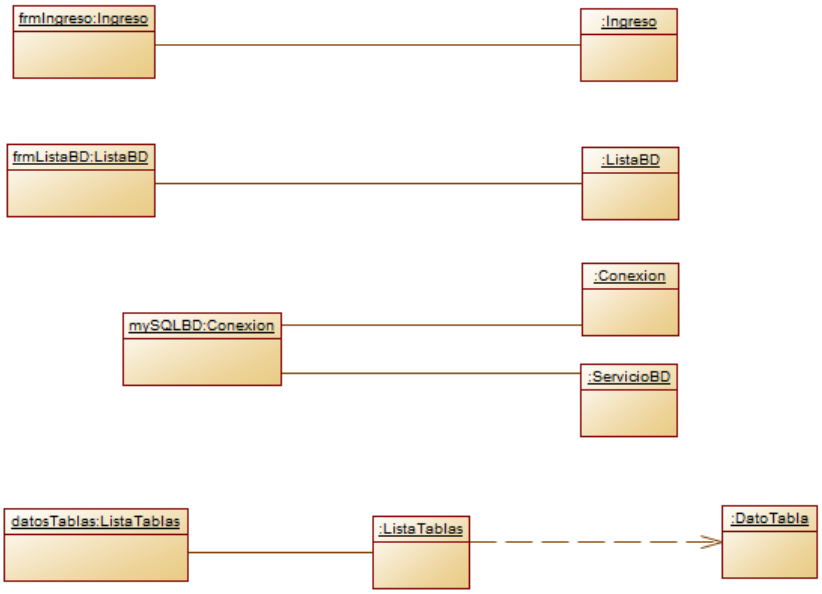



Ilustración 91. Diagrama de Objetos FredyCRUD

- Un **diagrama de paquetes** es un diagrama que suministra una vista gráfica de alto nivel de la organización de la aplicación y permite identificar la generalización y los enlaces de dependencia entre los paquetes.


Ilustración 92. Diagrama de Paquetes de FredyCRUD

Los **Diagramas Dinámicos** permiten modelar el comportamiento dinámico de un sistema, cómo los objetos interactúan en el momento de ejecución. Son cinco los diagramas dinámicos:

- **Diagramas de Comunicaciones**. Un diagrama de comunicaciones es un diagrama UML que suministra una vista gráfica de las interacciones entre objetos para un escenario de casos de uso, la ejecución de una operación o una interacción entre clases, con un énfasis en la estructura del sistema.

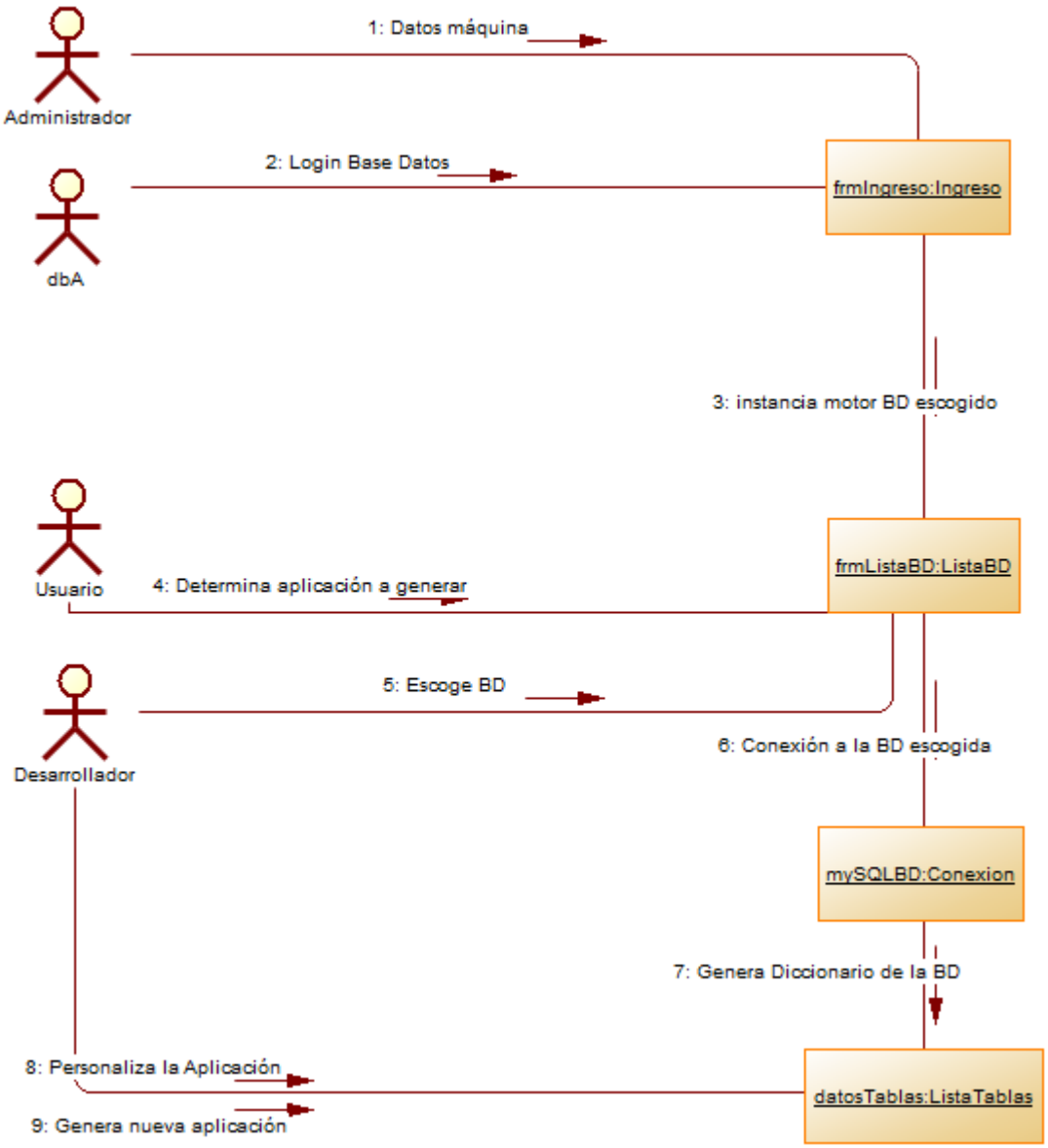



Ilustración 93. Diagrama de Comunicaciones

- **Diagramas de Secuencia**. Un diagrama de secuencia es un diagrama UML que suministra una vista gráfica de la cronología del intercambio de mensajes entre objetos y actores para un caso de uso, la ejecución de una operación o una interacción entre clases con énfasis en su cronología.

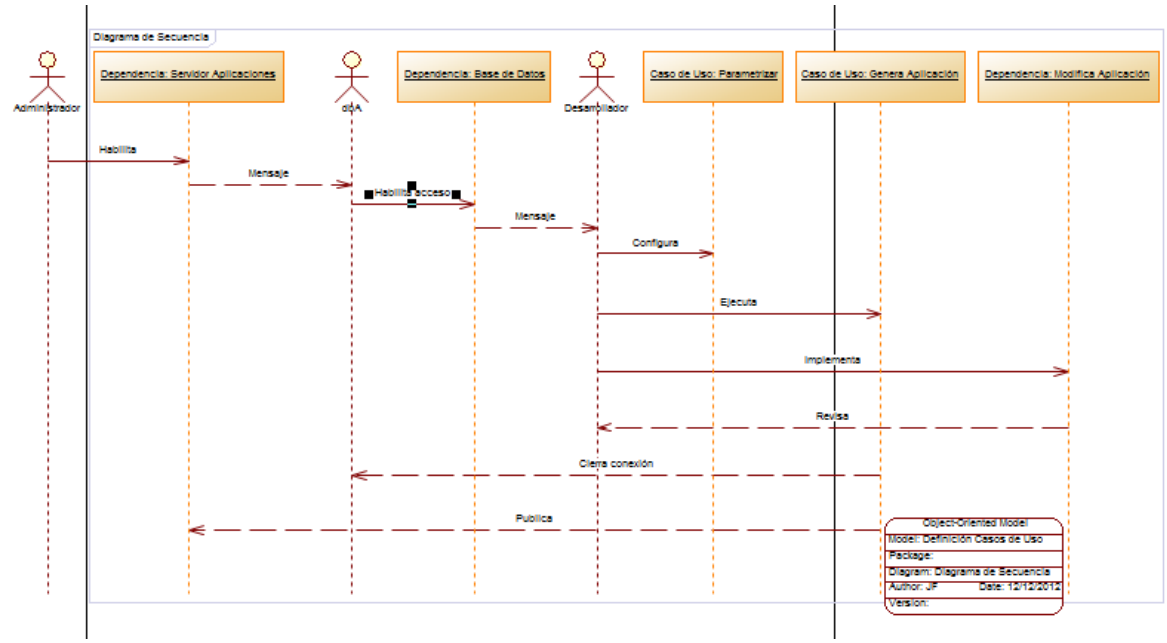



Ilustración 94. Diagrama General de Secuencias

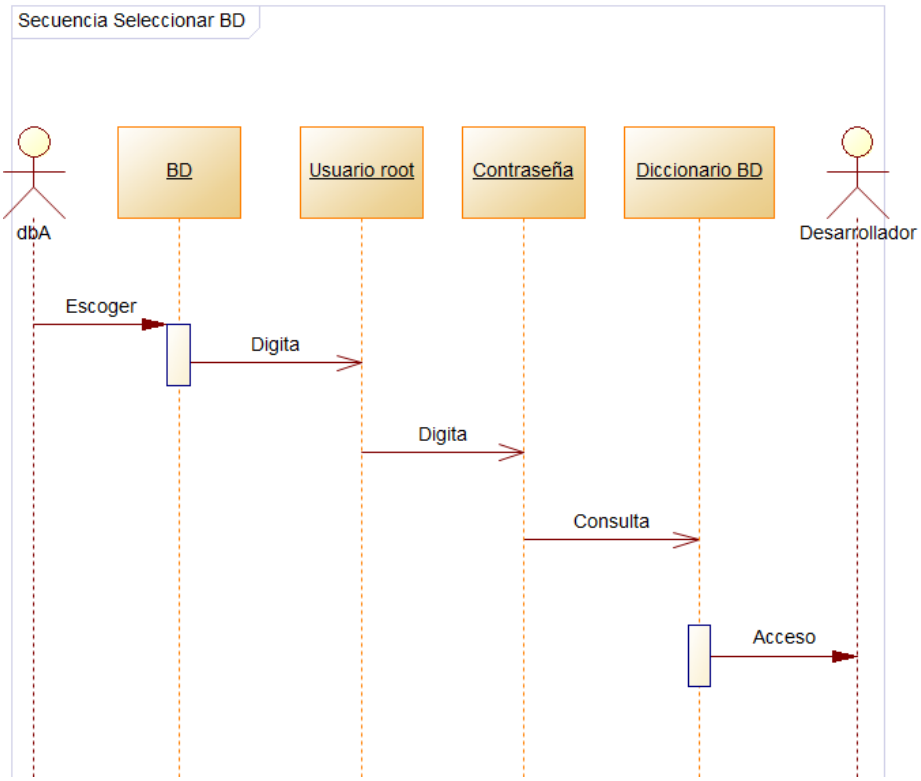



Ilustración 95. Diagrama de Secuencias Seleccionar BD

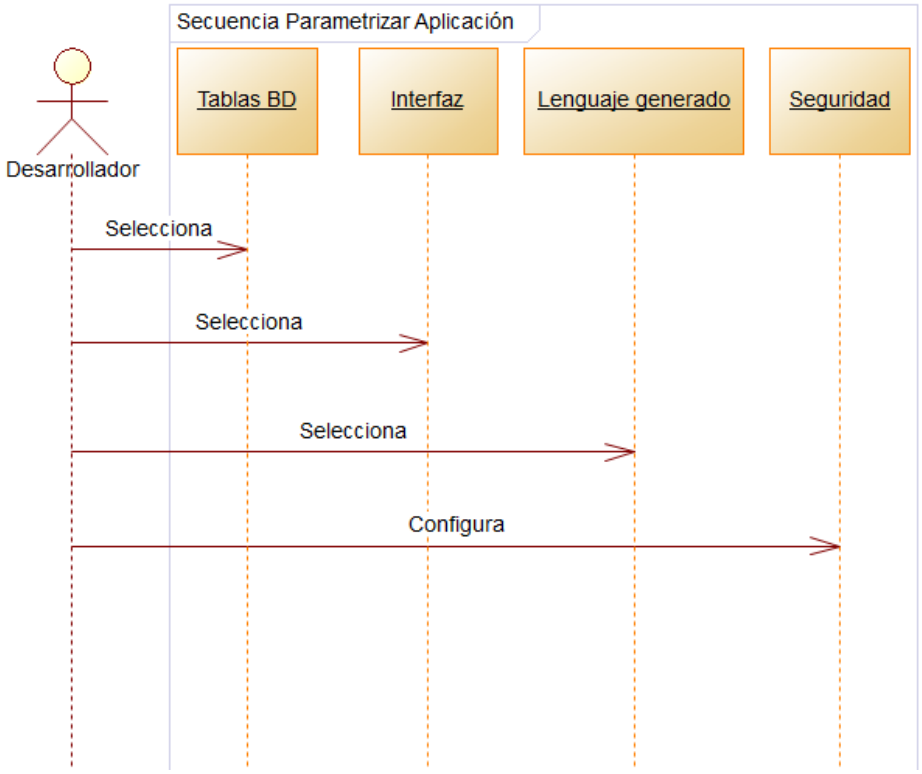



Ilustración 96. Diagrama de Secuencia Parametrizar Aplicación

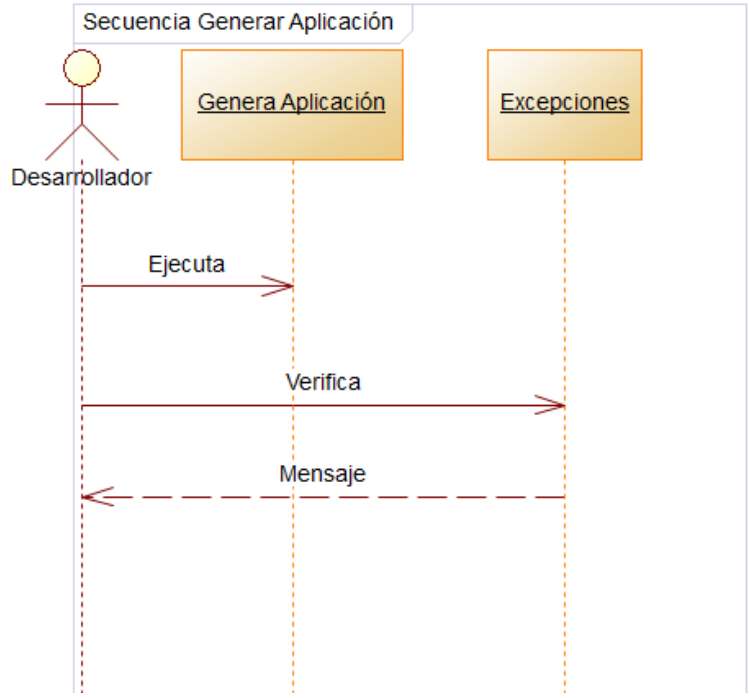



Ilustración 97. Diagrama Secuencias Generador Automático de Código

- Un **Diagrama de Actividades** es un diagrama UML que suministra una vista gráfica de un comportamiento del sistema y lo permite descomponer funcionalmente para analizar cómo será implementado.

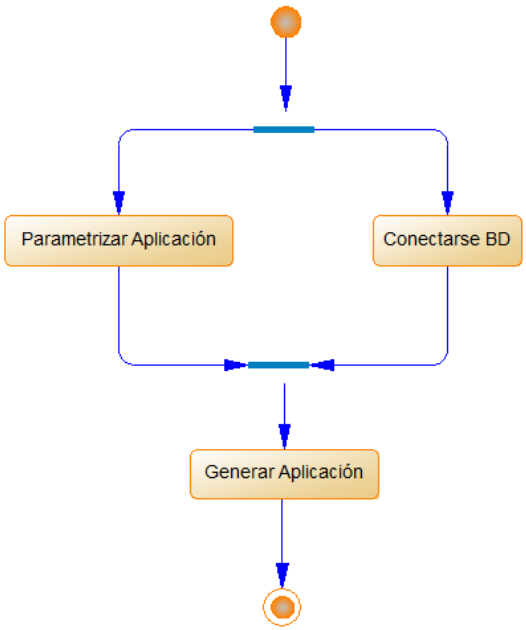



Ilustración 98. Diagrama de Actividades

- Un **Diagrama de Estados** es un diagrama UML que suministra una vista gráfica de una Máquina de Estados, el comportamiento público de un clasificador (componente o clase), en la forma de cambios sobre el momento del estado del clasificador y los eventos que permiten la transición de un estado a otro.

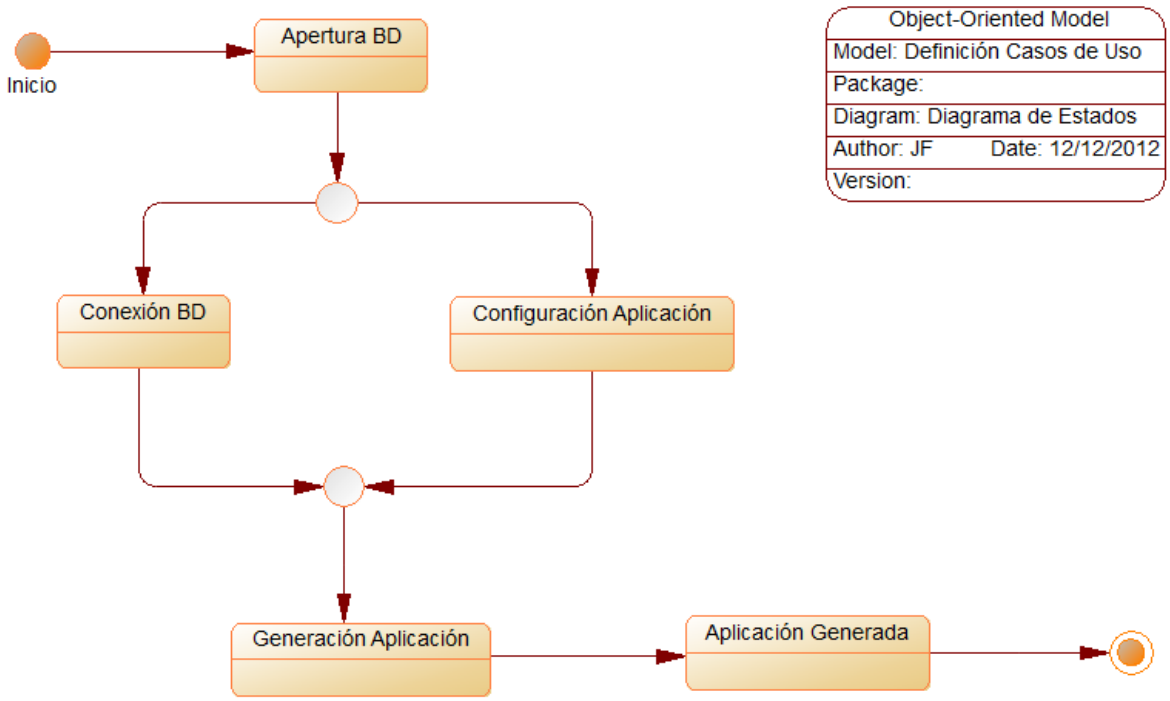



Ilustración 99. Diagrama de Estados

- Un **diagrama de Interacción** es un diagrama UML que suministra una vista gráfica de alto nivel del flujo de control del sistema como se descompone en secuencia y otros diagramas de interacción.

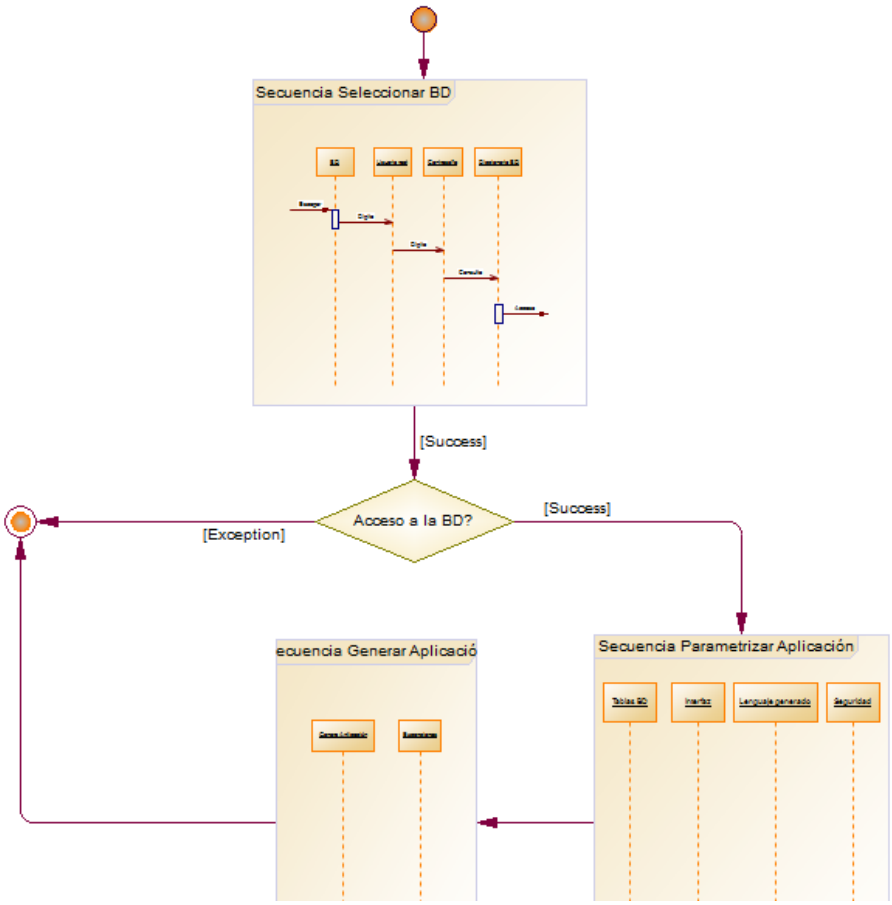



Ilustración 100. Diagrama de Interacción

### 6.3.1.6. DIAGRAMAS DE IMPLEMENTACIÓN

Permiten modelar el ambiente físico del sistema y cómo sus componentes serán desplegados. Son dos diagramas:

- Un **diagrama de componentes** representa el sistema descompuesto en componentes o sub-sistemas que se contienen a sí mismos. Puede mostrar los clasificadores que mantienen juntos a estos sistemas con los artefactos que lo implementan y exponen las interfaces ofrecidos o requeridos por cada componente y las dependencias entre ellos.

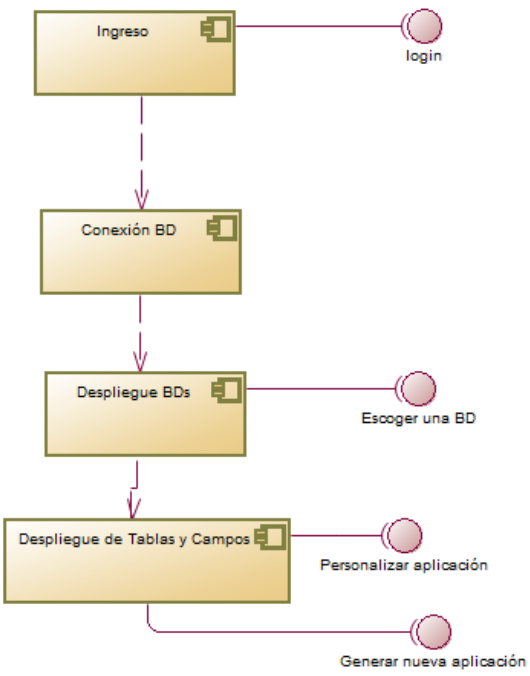



Ilustración 101. Diagrama de Componentes FredyCRUD

- Un **diagrama de despliegue** permite representar el ambiente de ejecución para un proyecto. Describe el hardware en cada uno de los componentes que se ejecutarán y cómo el hardware se conecta.

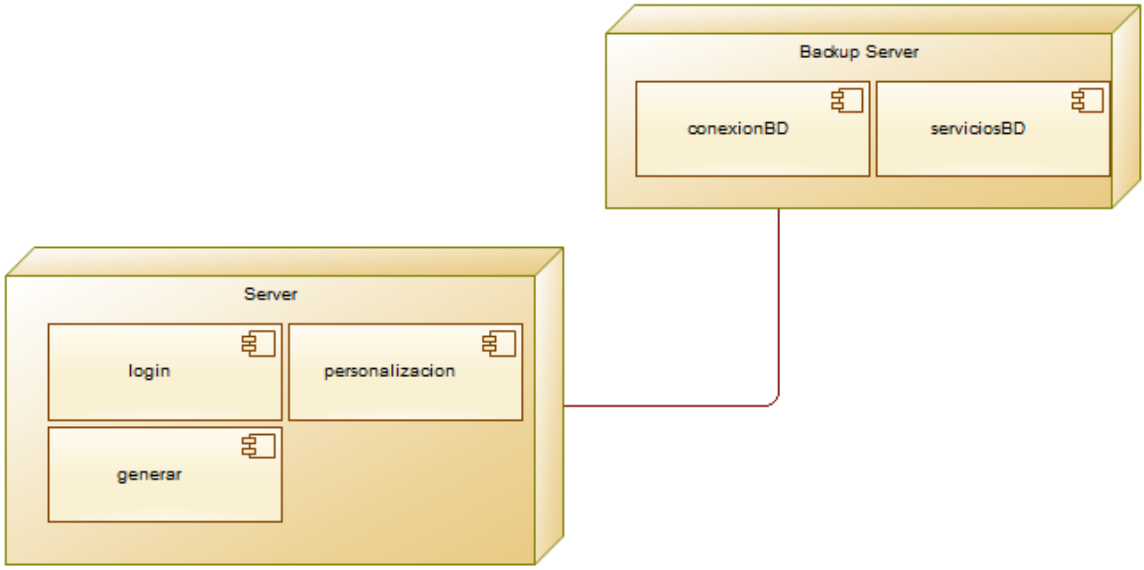



Ilustración 102. Diagrama de Despliegue FredyCRUD

#### 6.3.1.7. SERVICIOS WEB

Un servicio Web se ofrece vía web. El principio en el cual un servicio web trabaja es el siguiente: Una aplicación de negocios envía una solicitud a un servicio a una dirección URL dada. La solicitud puede usar el protocolo SOAP sobre HTTP. El servicio recibe la solicitud, procesarla, y retornar un respuesta. Un ejemplo comúnmente usado por un servicio Web es un servicio de cotización de inventarios en el cual la solicitud pregunta por el precio de un inventario en específico y la respuesta da el precio del inventario.

En un OOM, se diseña un servicio Web como un componente (EJB, servlet o un componente estándar) que incluye una clase de implementación de un servicio Web.

Cuando se trabaja con servicios Web en un OOM, se usan los diagramas de clases, componentes y despliegue, los cuales permiten:

- Crear un nuevo componente de Servicio Web
- Ingeniería inversa WSDL para crear el componente de Servicio Web
- Navegar UDDI para buscar WSDL
- Generar WSDL desde la definición del componente de Servicio Web
- Genera el código de Servicios Web del lado del servidor para Java (AXIS, JAXM, JAX-RPC, Web Services para J2EE) o .NET (C# y VB.NET)
- Genera el proxy del cliente para Java o .NET
- Ingeniería inversa para Java y .NET

Para trabajar con servicios Web, se necesita de un compilador de Java, C# o Visual Basic .NET

Para Java, también se necesita una herramienta WSDL a Java y Java a WSDL que genere el código del proxy de Java y el código del lado del servidor compatible con JAX-RPC. Las herramientas WSDL a Java y Java a WSDL son usadas por el WSDL para el archivo con extensión Java. Por ejemplo, el WSDP (Web Service Developer Pack) suministra una herramienta XRPCC, Apache ACIS suministra una herramienta wsdl2java y una java2wsdl (la cual puede ser descargada del Sitio de Desarrollo de Java: http://java.sun.com/index.jsp). Apache AXIS puede ser descargado de: http://ws.apache.org/axis.

#### 6.3.1.8. MODELO DE REQUERIMIENTOS

Un modelo de requerimientos (RQM) permite analizar cualquier clase de requerimientos por escrito y enlazarlas con los usuarios y grupos quienes los implementarán y con el diseño de objetos en otros modelos. Se puede usar un RQM para representar cualquier document estructurado (por ejemplo, especificaciones funcionales, plan de pruebas, objetivos del negocio, etc.) e importar y exportar jerarquías de requerimientos.

| Title ID | Full Description | Code | Priority | Workload | Risk | Status |
|---|---|---|---|---|---|---|
| **1.** | Descripción del Proyecto<br><br>En la Ingeniería de Software, una de las fases que requiere mayor esfuerzo se trata de la codificación del software, por cuanto se intenta hacer realidad en un programa informático, el trabajo realizado en etapas predecesoras, como es el caso del Análisis de Requerimientos y el Diseño del Software. Aprovechando el paradigma de generación de software, el proyecto consiste en generar de forma automática el código de la aplicación a desarrollar, a partir de la base de datos de esa aplicación, MySQL. El código generado permite el acceso en sus cuatro operaciones (consultar, insertar, modificar o borrar, conocido como CRUD)de la información contenidad en las tablas de esa base de datos. El proyecto se desarrolla bajo entornos de software libre. | REQ_0001 | 1 | | Low | Approved |
| **2.** | Descripción de los Escenarios | REQ_0002 | 1 | 0 | Low | Verified |
| **2.1** | Escenario 1<br><br>Aunque el desarrollo de software libre busca contribuir al progreso de la comunidad de software libre y en general de toda la sociedad, para permitir la apropiación del conocimiento, actualmente los proyectos de ingeniería de software son muy serios y exigentes en el cumplimiento de tiempos, a pesar que muchos de sus desarrolladores trabajan de forma desinteresada. Este apremio de tiempo busca | REQ_0003 | 1 | | Low | Verified |

| | | | | | | |
|---|---|---|---|---|---|---|
| | requerir de herramientas que permitan el desarrollo ágil de software. Existen muchas de estas herramientas en entornos del software privativo, pero no es el caso del software libre. | | | | | |
| **2.2** | Escenario 2<br><br>Algunas herramientas de software libre, permiten la generación automática de código para el desarrollo de aplicaciones, en especial, sistemas de información. Pero estas herramientas exigen como insumo, el diagrama de clases o cualquier otro tipo de diagrama, tipo UML. Se necesita de una herramienta que genere el código a partir de una base de datos previamente desarrollada, preferiblemente MySQL, por ser un DBMS ampliamente utilizado en los entornos de software libre. | REQ_0004 | 1 | | Low | Verified |
| **2.3** | Escenario 3<br><br>El software que genere código debe permitir realizar las cuatro operaciones básicas sobre los datos de las tablas de la Base de Datos, es decir, Consultar, Insertar, Actualizar y Borrar, conocido como formularios CRUD. | REQ_0005 | 1 | | Low | Verified |
| **2.4** | Escenario 4<br><br>El código generado por el software debe ser también entornos de software libre, que le permita al desarrollador de este entorno, posteriormente modificar (o empoderar) el código generado para adecuar la aplicación resultante, de acuerdo con los intereses del usuario. | REQ_0006 | 1 | | Low | Verified |

| | | | | | | |
|---|---|---|---|---|---|---|
| **3.** | Requerimientos Funcionales | REQ_0007 | 1 | 120 | Low | Draft |
| **3.1** | Parametrizar la nueva aplicación | REQ_0008 | 3 | 25 | Medium | Draft |
| **3.1.1** | Selección de tablas de las bases de datos<br><br>De las bases de datos, el desarrollador de la nueva aplicación, puede seleccionar las tablas, de las cuales se quiere que el software genere código | REQ_0009 | 1 | 5 | Medium | Verified |
| **3.1.2** | Configuración de las opciones de vista<br><br>Permite seleccionar opciones´ como: cuántos registros a ser vistos por página, tipo de ordenamiento de datos y formatos para exportar datos | REQ_0010 | 3 | 5 | Medium | Verified |
| **3.1.3** | Opciones de la organización<br><br>Permite configurar el nombre de la organización, su logo y el tema de visualización | REQ_0011 | 3 | 5 | Medium | Draft |
| **3.1.4** | Seguridad<br><br>Permite configurar los accesos al administrador de la nueva aplicación y usar una tabla de configuración de accesos a los demás usuarios | REQ_0012 | 3 | 10 | High | Draft |
| **3.2** | Conexión a la base de datos | REQ_0013 | 3 | 25 | High | Approved |
| **3.2.1** | Conexión al diccionario de datos<br><br>Permite la conexión al diccionario de la base de datos de Mysql, para mostrar las bases de datos que el usuario posteriormente escogerá | REQ_0014 | 4 | 10 | High | Approved |
| **3.2.2** | Ingreso a la base de datos<br><br>Lee el usuario (username) y la contraseña (password) de administración para poder | REQ_0015 | 5 | 15 | High | Approved |

| | | | | | | |
|---|---|---|---|---|---|---|
| | acceder a la base de datos escogida por el usuario | | | | | |
| **3.3** | Generación de código | REQ_0016 | 3 | 55 | High | Approved |
| **3.3.1** | Selección del código destino<br><br>Permite seleccionar el código destino, con preferencia php, para servidor web. | REQ_0017 | 3 | 30 | High | Approved |
| **3.3.2** | Generar código<br><br>Una vez revisado los parámetros de generación, el usuario mediante un clic sobre un botón, procede a generar el cógido de la aplicación destino | REQ_0018 | 5 | 20 | High | Approved |
| **3.3.3** | Log de errores<br><br>La aplicación después del proceso de generación automática de código, procede a crear un archivo del registro de errores presentados en este proceso. | REQ_0019 | 2 | 5 | Low | Approved |
| **3.4** | Posterior modificación a la aplicación generada<br><br>El código generado debe ser limpio, documentado y con posibilidad de modificación, para que de esta forma se pueda empoderar la aplicación obtenida | REQ_0020 | 4 | 10 | High | Approved |
| **3.5** | Acceso web<br><br>El software de generación automática de software debe poderse ejecutar en entornos web | REQ_0021 | 1 | 5 | Low | Approved |
| **4.** | Requerimientos no funcionales | REQ_0022 | 1 | 15 | Low | Draft |
| **4.1** | Interface amigable<br><br>Interface al usuario limpia, sencilla y muy amigable, que permita en una serie de pasos, | REQ_0023 | 2 | 5 | Low | Draft |

| | | | | | | |
|---|---|---|---|---|---|---|
| | poder generar el código de la aplicación resultante | | | | | |
| **4.2** | Velocidad de acceso<br><br>Debe esperar hasta un máximo de 2 minutos, la conexión a la base de datos. Los demás tiempos deben ser acordes a las aplicaciones web en general. La generación de código, se ve supeditada en tiempo, por la misma complejidad de la base de datos. | REQ_0024 | 1 | 5 | Low | Draft |
| **4.3** | Seguridad<br><br>La aplicación debe presentar un esquema de seguridad para la información de quien la use | REQ_0025 | 1 | 5 | Low | Draft |
| **5.** | Plataformas de Desarrollo<br><br>Debe ser hecha con herramientas de software libre y el código generado igualmente deber ser hecho con este tipo de licenciamiento | REQ_0026 | 1 | 5 | Low | Draft |
| **6.** | Análisis del Riesgo<br><br>Pérdida de Personal. Debido a que se trata de un proyecto de investigación de grado, la contratación de personal es limitada y el interés es netamente académico. Como medida de contingencia, el personal asignado puede o debe asumir varios roles. Además, se debe hacer uso de una metodología de desarrollo ágil de software, que permita lidiar con estos contratiempos.<br><br>Restricciones de seguridad. Para acceder a las bases de datos se | REQ_0027 | 1 | | Medium | Draft |

| | | |
|---|---|---|
| | necesitan plenos permisos de acceso a su información. Al igual permisos para hacer uso del espacio, donde se generará la aplicación. | |
| | Desinterés por la aplicación. Debido a que existen aplicaciones similares, pero con licenciamiento privativo, se puede perder interés en el uso de esta aplicación. Como contingencia, usar un repositorio de software libre para publicar la aplicación. | |

Tabla 14. Tabla de Ingeniería de Requerimientos

### 6.3.2. DISEÑO DE SOFTWARE

#### 6.3.2.1. MODELO DE DATOS

Un modelo de datos es una representación de la información consumida y producida por un sistema. El modelamiento de datos involucra el análisis de los objetos de datos presentes en un sistema y la relación entre ellos. Los diagramas son los modelos de datos físico, lógico y conceptual que le permiten analizar y modelar el sistema en todos los niveles de abstracción.

- El **Diagrama Físico de Datos** suministra una vista gráfica de la estructura de la base de datos y permite analizar sus tablas (incluyendo sus columnas, índices y detonadores), vistas y procedimientos y las referencias entre ellos.

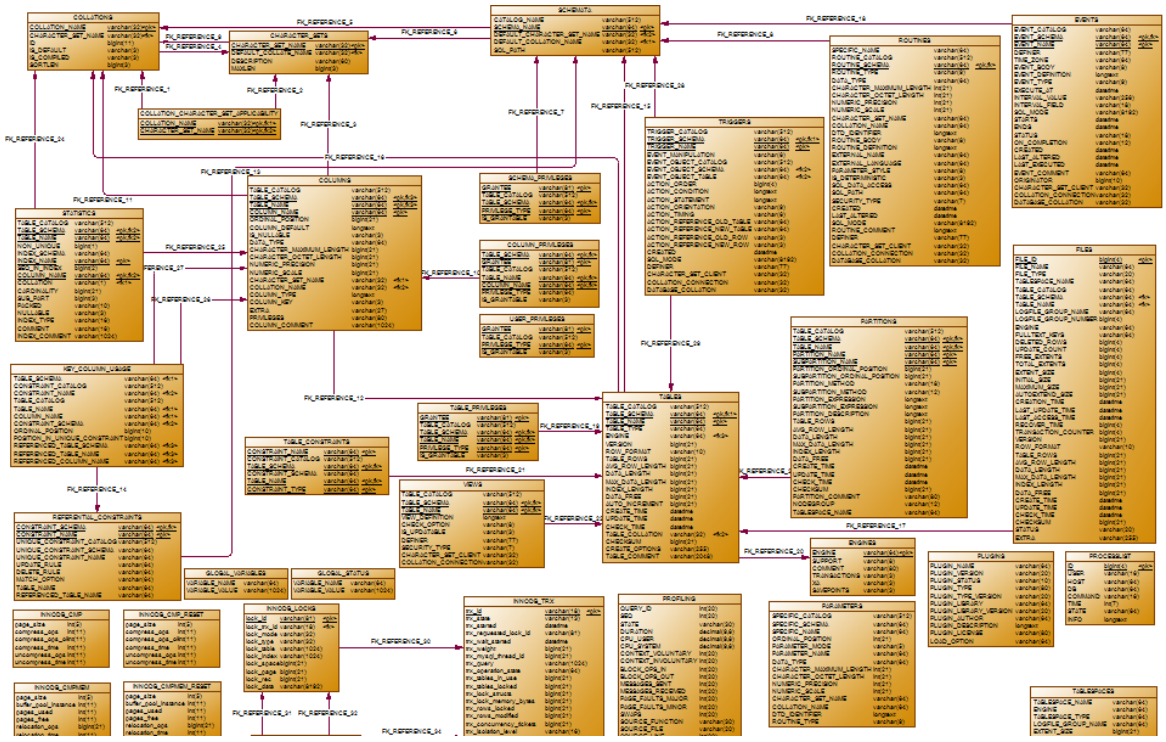


Ilustración 103. Diagrama Físico del Diccionario de Datos de mySQL

- **Diagramas Lógicos y Conceptuales** son modelos de datos que permiten modelar la estructura lógica y semántica del sistema.

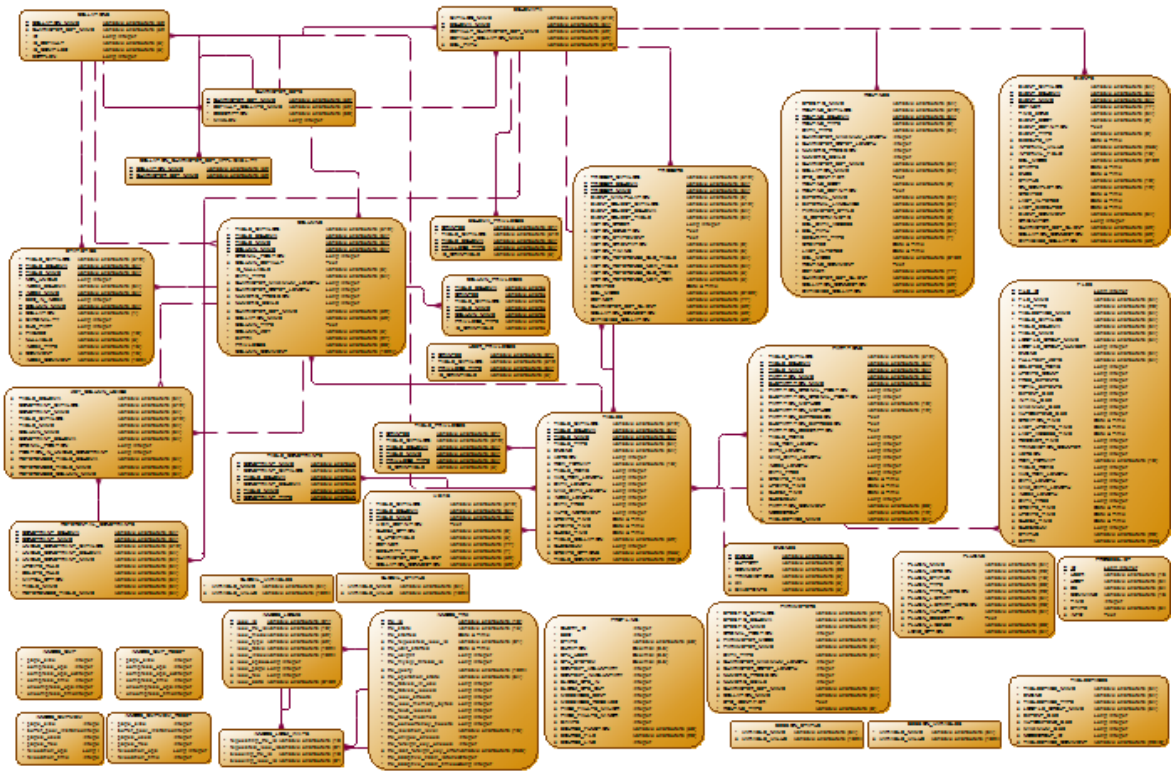


Ilustración 104. Modelo Lógico del Diccionario de Datos de mySQL

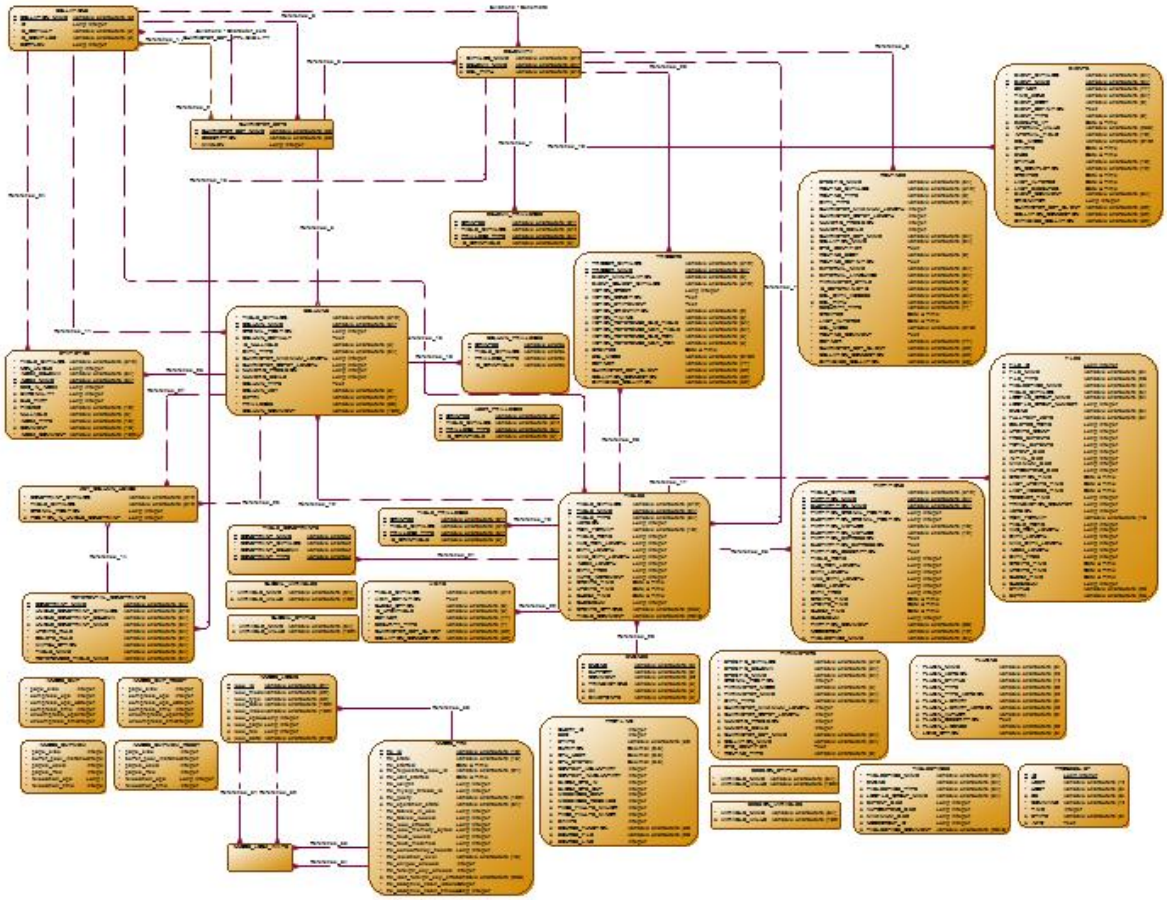


**Ilustración 105. Modelo Conceptual del diccionario de datos de mySQL**

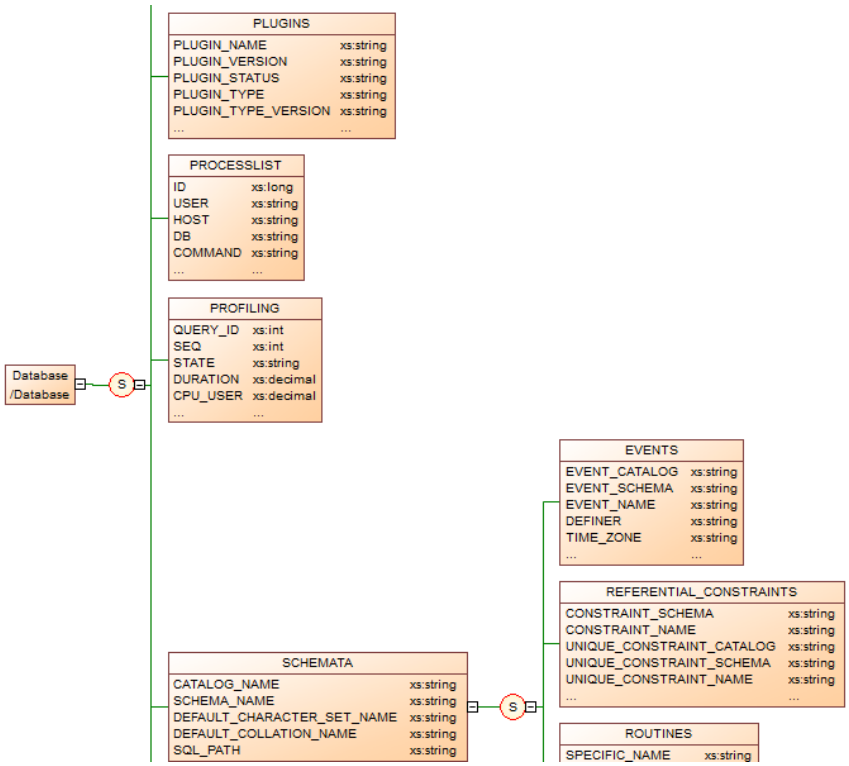



Ilustración 106. Diagrama XML General

### 6.3.2.2. DISEÑO DE INTERFAZ

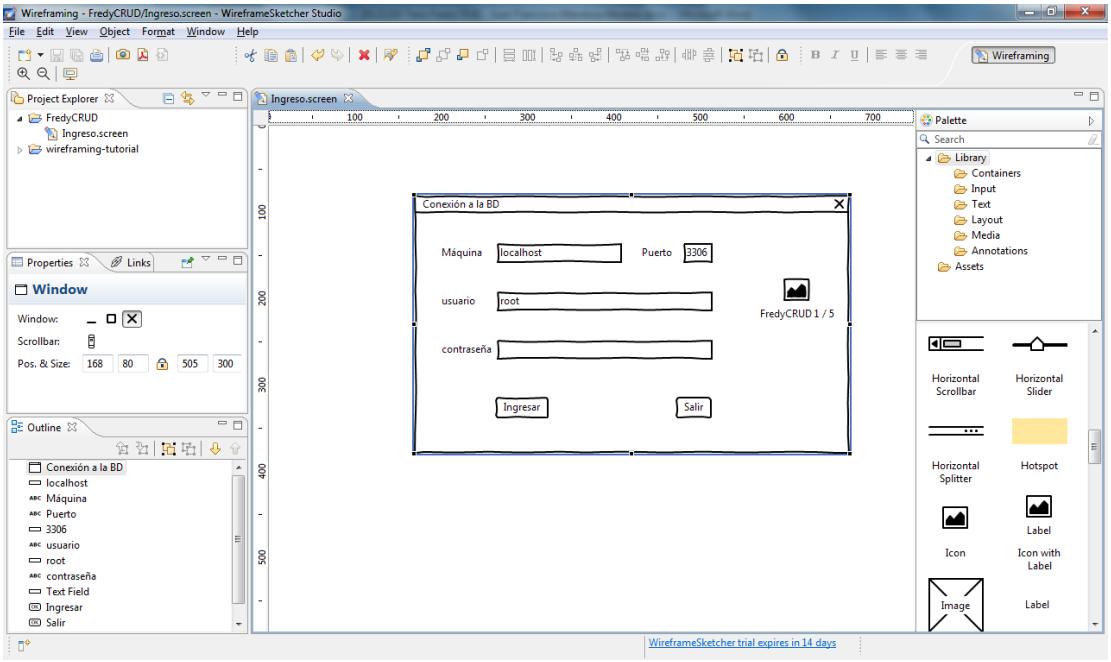



Ilustración 107. Diseño Interfaz Conexión BD


Ilustración 108. Interfaz de Login

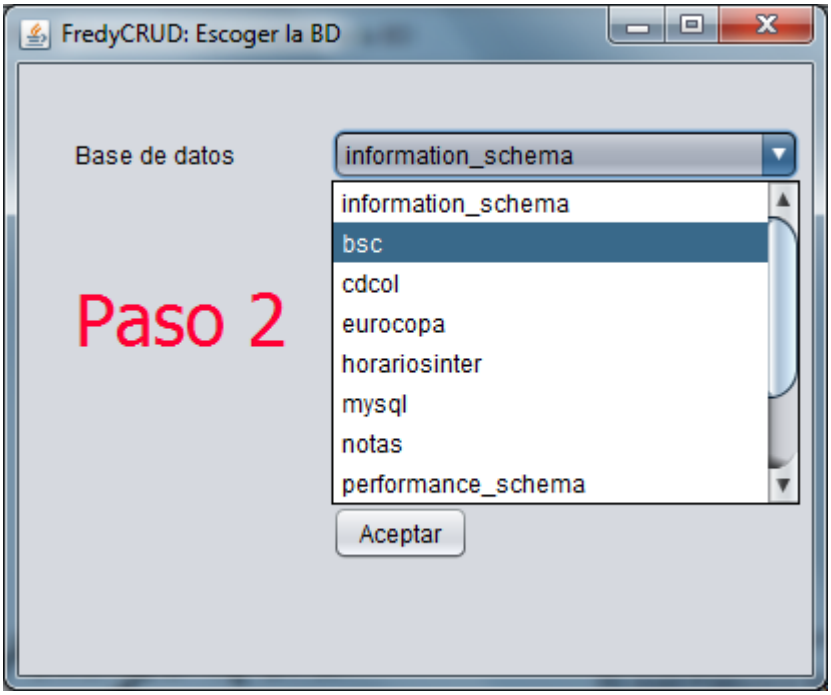



Ilustración 109. Interfaz Escoger BD

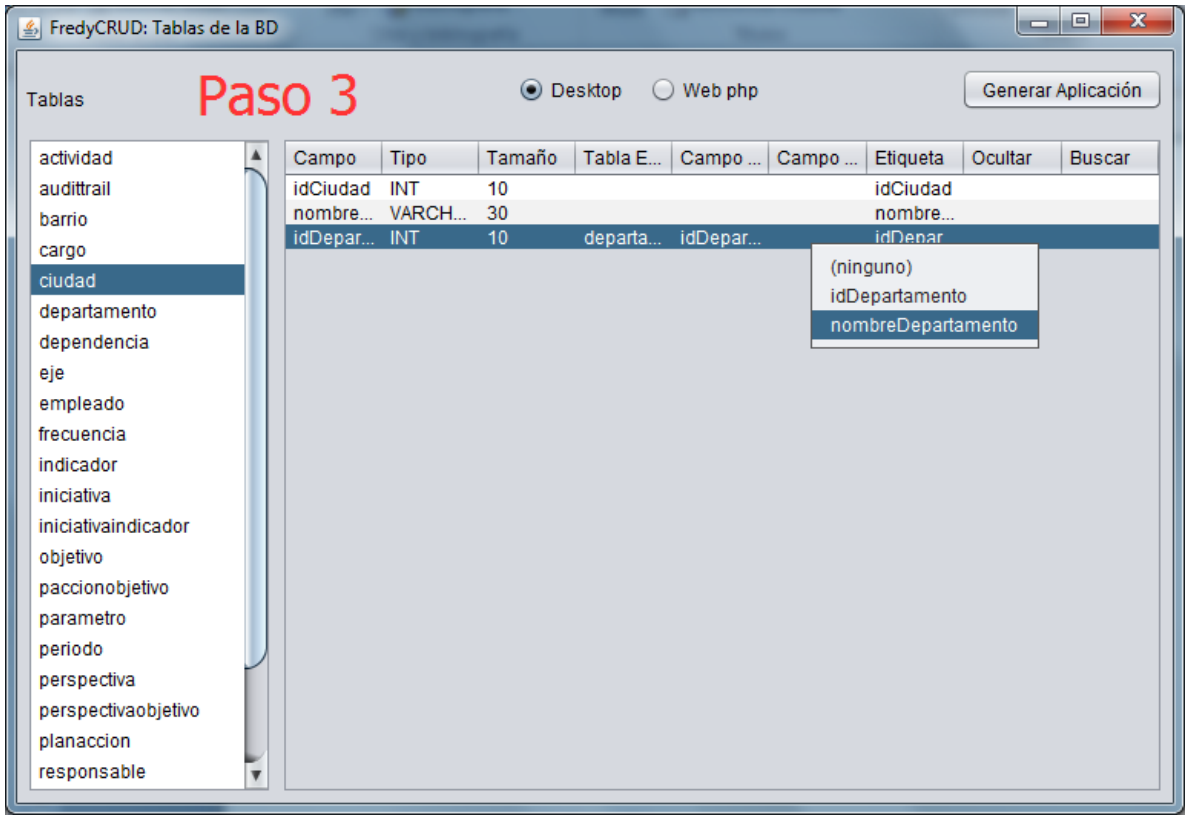



**Ilustración 110. Interfaz Personalizar y Generar Aplicación**