

DESARROLLO DE UN PROTOTIPO DE SOFTWARE TIPO CASE SOPORTADO EN EL
MODELO SEUDOMATEMÁTICO PARA EL DISEÑO DE BASES DE DATOS
RELACIONALES

JHON ALEXIS MÉNDEZ LARA
JOSÉ EDWIN APONTE BLANCO

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
ESPECIALIZACIÓN EN INGENIERÍA DE SOFTWARE
BOGOTÁ D.C.

2016

DESARROLLO DE UN PROTOTIPO DE SOFTWARE TIPO CASE SOPORTADO EN EL
MODELO SEUDOMATEMÁTICO PARA EL DISEÑO DE BASES DE DATOS
RELACIONALES

JHON ALEXIS MÉNDEZ LARA
JOSÉ EDWIN APONTE BLANCO

Proyecto de grado para optar por el título de
Especialista en Ingeniería de Software

Director
Roberto Pava

Codirector
John Jairo Londoño

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
ESPECIALIZACIÓN EN INGENIERÍA DE SOFTWARE
BOGOTÁ D.C.

2016

DEDICATORIA

Dedico esta tesis a mis padres, hermanos y esposa, que con su incondicional apoyo permitieron
que pudiera llevar a cabo este proyecto.

Jhon Méndez

Al Anciano de Días Quien nos ha dotado de la capacidad necesaria, a la memoria de mi padre por
todo lo que pudo enseñarme, mi madre ejemplo de esfuerzo y dedicación, mi esposa en quien
encuentro alegría sincera por cada logro.

Edwin Aponte

INTRODUCCIÓN	9
PARTE I. CONTEXTUALIZACIÓN DE LA INVESTIGACIÓN.....	11
1. CAPÍTULO I. DESCRIPCIÓN DE LA INVESTIGACIÓN	12
1.1. Planteamiento/identificación del problema	12
1.2. Formulación del problema	13
1.3. Objetivos.....	13
1.4. Justificación del trabajo/investigación	14
1.5. Hipótesis.....	15
1.6. Marco referencial	15
1.7. Metodología de la investigación	32
1.8. Organización del trabajo de grado	34
PARTE II. DESARROLLO DE LA INVESTIGACIÓN	35
2. CAPÍTULO II. CARACTERIZACIÓN DEL MODELO SEUDOMATEMÁTICO PARA EL DISEÑO DE LAS BASES DE DATOS RELACIONALES	36
2.1. La cadena lógica del negocio	36
2.2. CLNS Monofuncional Simple con Dependencia Funcional Exclusiva (DFE).....	37
2.3. CLNS Monofuncional Compuesta con Dependencia Funcional Exclusiva (DFE)	41
2.4. CLNS Bifuncional Simple con Dependencia Funcional Exclusiva (DFE)	44
2.5. CLNS Monofuncional Compuesta con Dependencia Funcional No Exclusiva (DFNE)	49
2.6. CLNS Bifuncional Simple con Dependencia Funcional No Exclusiva (DFNE)	53
2.7. CLNS Híbrida “Genérico” (Simple y Compuesta).....	57
3. CAPÍTULO III. GESTIÓN DEL PROYECTO	62
3.1. Requerimientos funcionales.....	62
3.2. Requerimientos no funcionales.....	63
3.3. Fase1: planificación del proyecto	63
3.4. Fase2: diseño	66
3.5. Fase3: codificación	68
3.6. Fase4: pruebas.....	72
3.6.1. Pruebas de rendimiento con WAPT	73
4. Capítulo IV análisis y diseño del prototipo	75
4.1. Casos de uso	75

4.1.1.	Especificación de casos de uso prototipo CASE	76
4.2.	Diagrama de clases	77
4.3.	Diagramas de estado	78
4.4.	Diagramas de secuencia	80
4.4.1.	Diagrama de secuencia Administrar cadenas	80
4.5.	Diagrama de componentes	82
4.6.	Diagrama de despliegue	83
PARTE III CIERRE DE LA INVESTIGACIÓN		84
5.	Capítulo V RESULTADOS Y DISCUSIÓN	85
6.	Capítulo VI CONCLUSIONES.....	86
6.1.1.	Verificación	86
6.1.2.	Contraste	92
6.1.3.	Evaluación de los objetivos	95
7.	Capítulo VII Prospectivo Del Trabajo	96
7.1.1.	Líneas de investigación futuras	96
7.1.2.	Trabajos de investigación futuros	96
8.	BIBLIOGRAFÍA	97

LISTA DE TABLAS

<i>Tabla 1 Descripción del Modelo Seudomatemático</i>	<i>36</i>
<i>Tabla 2 Requerimientos Funcionales.....</i>	<i>62</i>
<i>Tabla 3 Requerimientos No Funcionales</i>	<i>63</i>
<i>Tabla 4 Historias de Usuario.....</i>	<i>64</i>
<i>Tabla 5 Release Planning</i>	<i>64</i>
<i>Tabla 6 Iteration Planning.....</i>	<i>65</i>
<i>Tabla 7 Velocidad de Proyecto</i>	<i>66</i>
<i>Tabla 8 Tarjetas C.R.C.</i>	<i>66</i>
<i>Tabla 9 Caso de prueba Cadena Lógica del Negocio</i>	<i>72</i>
<i>Tabla 10 Especificación de casos de uso</i>	<i>76</i>
<i>Tabla 11 Diagramas de estado</i>	<i>79</i>
<i>Tabla 12 Lista de diagramas de secuencia</i>	<i>81</i>
<i>Tabla 13 Lista de Entidades.....</i>	<i>92</i>
<i>Tabla 14 Lista de Atributos por Entidad.....</i>	<i>93</i>

LISTA DE FIGURAS

<i>Figura 1. Información antes de convertir a Primera Forma Normal.....</i>	<i>19</i>
<i>Figura 2. Tablas en Primera Forma Normal.....</i>	<i>19</i>
<i>Figura 3. Tablas en Segunda Forma Normal</i>	<i>20</i>
<i>Figura 4. Tablas en Tercera Forma Normal</i>	<i>20</i>
<i>Figura 5. Tablas en Cuarta Forma Normal.....</i>	<i>21</i>
<i>Figura 6. Cadena Lógica de Negocio del Sistema.....</i>	<i>21</i>
<i>Figura 7. Ciclo de Vida XP tomada de http://www.extremeprogramming.org/map/project.html.26</i>	
<i>Figura 8. El paradigma de hacer prototipos, Pressman 2010.....</i>	<i>26</i>
<i>Figura 9. Estructura básica de un MVC.</i>	<i>28</i>
<i>Figura 10. Representación gráfica del sujeto y el grupo transaccional.....</i>	<i>37</i>
<i>Figura 11. Cadena Lógica del Negocio Monofuncional Simple con DFE</i>	<i>38</i>
<i>Figura 12. Modelo Relacional Monofuncional Simple con DFE</i>	<i>39</i>
<i>Figura 13. Modelo Entidad Relación Monofuncional Simple con DFE.....</i>	<i>39</i>
<i>Figura 14. CLNS Monofuncional Compuesta con Dependencia Funcional Exclusiva (DFE)</i>	<i>41</i>
<i>Figura 15. Modelo de Dependencias Funcionales Monofuncional Compuesta con DFE</i>	<i>42</i>
<i>Figura 16. Modelo Relacional Monofuncional Compuesta con DFE</i>	<i>42</i>
<i>Figura 17. Modelo Entidad Relación Monofuncional Compuesta con DFE.....</i>	<i>43</i>
<i>Figura 18. CLNS Bifuncional Simple con Dependencia Funcional Exclusiva (DFE)</i>	<i>45</i>
<i>Figura 19. Modelo Relacional Bifuncional Simple con Dependencia Funcional Exclusiva (DFE)</i> <i>.....</i>	<i>46</i>
<i>Figura 20. Modelo Entidad Relación Bifuncional Simple con Dependencia Funcional Exclusiva</i> <i>(DFE)</i>	<i>47</i>
<i>Figura 21. CLNS Monofuncional Compuesta con Dependencia Funcional No Exclusiva (DFNE)</i> <i>.....</i>	<i>49</i>
<i>Figura 22. Modelo de Dependencias Funcionales Monofuncional Compuesta con (DFNE)</i>	<i>50</i>
<i>Figura 23. Modelo Relacional CLNS Monofuncional Compuesta con (DFNE)</i>	<i>51</i>
<i>Figura 24. Modelo Entidad Relación Monofuncional Compuesta con (DFNE)</i>	<i>51</i>
<i>Figura 25. CLNS Bifuncional Simple con (DFNE).....</i>	<i>54</i>
<i>Figura 26. Modelo de Dependencias Funcionales Bifuncional Simple con (DFNE).....</i>	<i>55</i>

<i>Figura 27. Modelo Entidad Relación Bifuncional Simple con (DFNE)</i>	55
<i>Figura 28. Modelo Relacional Bifuncional Simple con (DFNE)</i>	56
<i>Figura 29. CLNS Híbrida “Genérico” (Simple y Compuesta)</i>	58
<i>Figura 30. Modelo de Dependencias Funcionales Híbrida “Genérico” (Simple y Compuesta)</i> ..	59
<i>Figura 31. Modelo Relacional Híbrida “Genérico” (Simple y Compuesta)</i>	59
<i>Figura 32. Modelo Entidad Relación Híbrida “Genérico” (Simple y Compuesta)</i>	60
<i>Figura 33. Directorios del proyecto Archibase</i>	69
<i>Figura 34. Directorio database del proyecto archibase</i>	70
<i>Figura 35. Carpeta public del framework laravel para el proyecto archibase</i>	71
<i>Figura 36. Carpeta resources del proyecto archibase utilizando Laravel</i>	71
<i>Figura 37. Resultados de prueba de carga al prototipo Archibase</i>	74
<i>Figura 38. Caso de uso general</i>	75
<i>Figura 39. Diagrama de clases</i>	77
<i>Figura 40. Diagrama de estados administrar cadenas</i>	78
<i>Figura 41. Diagrama de secuencia administrar cadenas</i>	81
<i>Figura 42. Diagrama de Componentes</i>	82
<i>Figura 43. Diagrama de Despliegue</i>	83
<i>Figura 44. CLNS usando el prototipo CASE</i>	87
<i>Figura 45. Adición Atributos a la Entidad SUCURSALES usando el prototipo CASE</i>	88
<i>Figura 46. Creación de Tabla Auxiliar CIUDADES usando el prototipo CASE</i>	88
<i>Figura 47. Adición Atributos a la Entidad VEHICULOS usando el prototipo CASE</i>	89
<i>Figura 48. Creación de Tabla Auxiliar MARCAS usando el prototipo CASE</i>	89
<i>Figura 49. Creación de Tabla Auxiliar TIPOVEHICULO usando el prototipo CASE</i>	89
<i>Figura 50. Creación de Tabla Auxiliar TIPOCOMB usando el prototipo CASE</i>	90
<i>Figura 51. Creación de Tabla que resuelve DNFE usando el prototipo CASE</i>	91
<i>Figura 52. Modelo de Dependencias Funcionales usando el prototipo CASE</i>	92
<i>Figura 53. Vista parcial del script SQL obtenido del prototipo CASE</i>	92
<i>Figura 54. Caso de Estudio Modelo Entidad Relación - Método Tradicional</i>	94

INTRODUCCIÓN

Bajo el entendimiento de los sistemas de información automáticos como piezas de un diseño que busca representar en objetos intangibles una realidad que describir o controlar, se encuentra el modelo de datos como una parte fundamental de este entramado; por cuanto su contenido se manifiesta en la información misma de la realidad que se ha plasmado a través de su desarrollo. A partir del concepto absoluto como se concibe el sistema, rico en lógica y procedimientos, se hace posible diseñar una solución automática que vincula software y datos. Esta información almacenada debe contar con el suficiente grado de precisión como para considerarse confiable y usable dentro del marco del sistema que representa.

Dicho esto se entiende que el modelo de datos supone parte de la plataforma sobre la que se construye un sistema de información que independiente de su implementación o de la forma como se persistan los datos obtenidos de la realidad modelada, siempre depende del juicioso y coherente análisis y diseño de la estructura de datos para que el sistema de información sea considerado como útil y de calidad a los usuarios finales.

Desde esta perspectiva, el denominado Modelo Seudomatemático para el Diseño de Bases de Datos Relacionales autoría del ingeniero John Jairo Londoño como propuesta de una metodología que, mediante la comprensión del entorno real que se busca modelar, cobra un enorme valor en la medida en que permite identificar los componentes principales de la realidad y el contexto en la que se modela. El mencionado modelo permite obtener una vista general de la creación de un diseño de bases de datos a partir del contexto que da un sistema; esto se logra con la contemplación de dos preguntas: “¿para quién se quiere controlar?” y “¿qué se quiere controlar?”. Respondiendo la primera se tiene lo que la metodología denomina “el sujeto” y la segunda, “el grupo transaccional” ambos componentes forman lo que se denomina la cadena lógica de negocio del sistema, que no es más que la abstracción completa del sistema a analizar. El modelo permite construir la base de datos a partir de la misma normalización, involucrando desde el principio los fundamentos de las dependencias funcionales.

En ese orden ideas ha surgido el interés por llevar a cabo el desarrollo del prototipo CASE que contemple la creación de diseños de bases de datos para soluciones que se basen en esta metodología, se busca caracterizar toda la teoría del ingeniero Londoño y mostrar cómo a través del uso de una metodología de desarrollo de prototipos se puede estructurar la aplicación propuesta.

En este documento, por tanto, se describe el trabajo investigativo dividido en tres partes y en siete capítulos donde se exponen las etapas por la que se ha atravesado para completar la labor inicialmente propuesta

La primera parte corresponde a la descripción de la investigación y se compone de un capítulo. En la segunda parte, se describe el desarrollo de la investigación y en la tercera se hace el cierre de la misma.

En el primer capítulo se contextualiza el trabajo de investigación, se plantea el problema, se delinean los objetivos, se manifiesta la justificación del trabajo de investigación, así como la

presentación de la hipótesis, el marco referencial, la metodología de investigación y la organización del trabajo.

En el segundo capítulo se lleva a cabo la caracterización del modelo seudomatemático para el diseño de bases de datos relacionales propuesto por el ingeniero John Jairo Londoño.

El tercer capítulo corresponde a la gestión del proyecto, es donde se describen los requerimientos funcionales y no funcionales, de igual forma, se describen la fases de planificación, diseño, codificación y pruebas del proyecto.

En el cuarto capítulo se aborda el tema de análisis y diseño del prototipo, donde se presentan los casos de uso, los diagramas de clases, estado, secuencia, componentes y despliegue.

En el quinto capítulo se presentan los resultados y discusión donde se presenta un artículo científico, producto del trabajo de investigación.

El sexto capítulo corresponde a las conclusiones donde, a través de la aplicación de un caso de estudio, se busca verificar la solución obtenida mediante el uso del prototipo desarrollado es equivalente a la solución de diseño de bases de datos que se lleva a cabo mediante el enfoque tradicional.

El séptimo capítulo presenta el enfoque prospectivo del trabajo en cuanto a las líneas de investigación futuras, así como los trabajos derivados del mismo que puedan llevarse a cabo en el futuro.

PARTE I. CONTEXTUALIZACIÓN DE LA INVESTIGACIÓN

1. CAPÍTULO I. DESCRIPCIÓN DE LA INVESTIGACIÓN

1.1. Planteamiento/identificación del problema

En la actualidad existen muchos modelos para el diseño de las bases de datos, cada uno de ellos basados en modelos matemáticos, teoría de conjuntos, lógica, etc. los más usados de todos ellos son el Modelo Entidad-Relación y el Modelo Relacional. El Ingeniero y profesor de la Universidad Distrital Francisco José De Caldas John Jairo Londoño Pérez ha creado una metodología denominada: “Modelo seudomatemático para el diseño de las bases de datos relacionales”; implementada con un lenguaje que simplifica, de alguna manera, el proceso de diseño de las bases de datos. Dicha metodología parte del análisis del contexto del sistema, en donde se espera que opere el diseño, y la contemplación de dos preguntas únicas “¿Qué se quiere controlar?” y “¿Para quién se quiere controlar?”.

Hoy día existen una gran variedad de herramientas CASE para el diseño de las bases de datos, entre las más populares están: Oracle Designer, Microsoft Visio, SQL Server, Power Designer entre otras; todas ellas desarrolladas bajo la premisa del modelo entidad-relación. El adquirir estas herramientas hace que el proceso de diseño de bases de datos sea un poco más rápido y ágil al momento de implementarlas en un proyecto de software. Sin embargo estas herramientas no parten directamente del contexto del sistema o negocio sino que deja todo en manos de la subjetividad y criterio del ingeniero que diseña la solución. Bajo esta perspectiva se hace evidente que no existe una herramienta CASE que opere bajo una metodología que contemple el contexto del sistema como primordial eje en el diseño de una base de datos y tampoco ninguna que aplique la metodología creada por el profesor Londoño Pérez.

En consecuencia con lo anterior sin una herramienta CASE que contemple el diseño de bases de datos partiendo del análisis de contexto del sistema, los ingenieros se ven obligados a continuar diseñando bases de datos a partir de supuestos y especulaciones que con dificultad se pueden juzgar en torno al grado de congruencia con el sistema analizado.

Al desarrollar una aplicación CASE para el diseño de las bases de datos que utilicen la metodología del modelo seudomatemático se tiene una herramienta que permite el uso de buenas prácticas en el diseño de las bases de datos, redundando en que éstas vayan de acuerdo al contexto del sistema y evitando subjetividades en su construcción.

1.2. Formulación del problema

Para hacer más comprensible la problemática descrita anteriormente es necesario conocer, ¿Cómo se puede desarrollar un prototipo de aplicación CASE que sirva como herramienta en el diseño de las bases de datos realizadas bajo el “modelo seudomatemático para el diseño de bases de datos relacionales”?

¿Qué aspectos hay que tener en cuenta para caracterizar el “modelo seudomatemático para el diseño de bases de datos relacionales”, que sirvan como base para el desarrollo del software tipo CASE?

¿Cómo se puede mostrar el cumplimiento de los axiomas de Armstrong desde una herramienta CASE aplicados en el modelo seudomatemático y así verificar la validez del modelo?

¿De qué manera el uso del modelo de prototipos de software permite el desarrollo de un aplicativo tipo CASE?

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar un software tipo CASE, utilizando el modelo de desarrollo orientado a prototipos, como una alternativa para el diseño de bases de datos que apliquen el “modelo seudomatemático para el diseño de bases de datos relacionales”.

1.3.2. Objetivos específicos

Caracterizar el modelo seudomatemático para el diseño de las bases de datos relacionales, haciendo uso de los fundamentos teóricos planteados por el ingeniero John Jairo Londoño Pérez, que permitan definir las funcionalidades principales de la herramienta CASE a desarrollar.

Mostrar el cumplimiento de los axiomas de Armstrong aplicados en el modelo seudomatemático por medio del prototipo de software CASE con el fin de verificar la validez del modelo.

Construir el prototipo de software CASE utilizando el modelo de desarrollo de prototipos, que permita diseñar una base de datos.

1.4. Justificación del trabajo/investigación

En el diseño de bases de datos el modelo de Entidad-Relación es uno de los más extendidos e implementados, el ingeniero parte de un supuesto de tablas para crear un esquema que sirva de soporte de persistencia de datos para interactuar con el software a crear, esto según el profesor de la Universidad Distrital Ingeniero John Jairo Londoño genera confusión al momento de interpretar o abstraer un sistema, por lo cual el diseño del modelo de base de datos se torna subjetivo y se pierde la noción del sistema analizado. El profesor Londoño ha creado una metodología que permite tener una concepción general del sistema para así poder diseñar una base de datos; la ha definido como el "modelo seudomatemático para el diseño de las bases de datos relacionales". Esta metodología se basa principalmente en crear un conjunto de componentes que represente el funcionamiento del contexto a modelar, esto es el sistema, que ya existe, sólo que se representa, a este conjunto de componentes el ingeniero John Jairo Londoño Pérez le ha denominado "Cadena Lógica de Negocio del Sistema" (CLNS) que en conjunto con la definición de las Dependencias Funcionales entre los componentes de la cadena se logra representar el sistema desde la base de datos.

Este modelo planteado por el Ingeniero Londoño es nuevo e innovador, permite al ingeniero experto en diseño de base de datos tener una vista general del sistema para coordinar el diseño de software con el de la base de datos. Propone simplificar el modelamiento de bases de datos, pues al iniciar los pasos de ésta se aplican directamente los principios de normalización. La metodología no renuncia a los fundamentos tradicionales de las bases de datos, sino que concibe la teoría de Armstrong como un referente en la definición de la lógica del sistema.

Se hace necesario crear una herramienta que permita apoyar el diseño de bases de datos contemplando la teoría anteriormente descrita, para lo cual este trabajo se hace pertinente porque apoya una metodología naciente que complementa la manera cómo se ha llevado a cabo el proceso de diseño de bases de datos; por ende desarrollar una herramienta CASE aplicable a esta metodología resulta trascendental como quiera que se use dentro de un marco de la Ingeniería de Software; entendiendo que la metodología ofrece simplicidad, inmediatez, eficacia y confiabilidad.

El desarrollo del proyecto de investigación busca construir el cuerpo práctico del modelo seudomatemático para el diseño de bases de datos relacionales propuesto por el ingeniero John

Jairo Londoño Pérez en el que el producto final es un script DDL para la implementación en el motor de bases de datos que se elija.

1.5. Hipótesis

En el entendimiento de la necesidad de contar con una herramienta que asista el diseño de bases de datos relacionales desde la aplicación del modelo propuesto por el ingeniero John Jairo Londoño Pérez y que desde este modelo se puede representar el sistema analizado en el contexto, se plantea la siguiente hipótesis:

La teoría del modelo seudomatemático para el diseño de las bases de datos relacionales es susceptible de ser implementada en un prototipo CASE que sirva de apoyo para el efectivo diseño de las mismas.

1.6. Marco referencial

1.6.1. Marco teórico

1.6.1.1. Bases de datos relacionales

Las bases de datos relacionales son un tipo de base de datos que pretenden representar los datos en forma de tablas y relaciones, es uno de los modelos más utilizados y populares, creado por Edgar Frank Codd a finales de los años 60. Según el profesor Henry F. Korth en su libro *“Fundamentos de bases de datos”* las bases de datos relacionales consisten en un conjunto de tablas, a las cuales se les asigna un nombre exclusivo; tiene como estructura la selección de atributos para cada tabla y un dominio asociado a ese atributo. Estos modelos son basados en fundamentos matemáticos ya que las relaciones de las tablas se ven como subconjuntos del producto cartesiano en la lista de dominios. Se utilizan los términos matemáticos relación y tupla en lugar de los términos tabla y fila. Una variable tupla es una variable que representa a una tupla; en otras palabras, una tupla que representa al conjunto de todas las tuplas.

1.6.1.2. Dependencias funcionales

Una dependencia funcional según Date (2004) afirma que “una dependencia funcional (abreviada DF) es un vínculo muchos a uno que va de un conjunto de atributos a otro dentro de una determinada varrel o tabla”. Dada esta definición si tenemos una tabla **X** donde se encuentren subconjunto $A \rightarrow B$, se dice que hay una dependencia de B hacia A si se encuentran

tuplas o registros con el mismo valor; es decir que los valores de A se encuentren también en B, siendo A el subconjunto determinante y B el subconjunto dependiente.

1.6.1.3. Normalización de bases de datos

La normalización de bases de datos son una serie de reglas que se deben tener en cuenta para hacer el diseño de las mismas. Para que una tabla cumpla con estas reglas se deben tener en cuenta 5 formas normales y una denominada Boyce-Cood.

- **Primera forma normal (1FN):** “una tabla esta en 1FN si y solo si, en cada valor valido de esa tabla, todo el registro contiene exactamente un valor para cada atributo” (Date, 2004, p.357).
- **Segunda forma normal (2FN):** esta forma normal está definida por Date (2004) como: “una varrel está en 2FN si y sólo si está en 1FN y todo atributo que no sea clave es dependiente irreduciblemente de la clave primaria” (p.360).
- **Tercera forma normal (3FN):** “Una varrel está en 3FN si y sólo si está en 2FN y los atributos que no son clave son dependientes en forma no transitiva de la clave primaria” (Date, 2004, p.362).
- **Forma normal de Boyce/Codd (FNBC):** “Una varrel está en FNBC si y solamente si toda DFi trivial, irreducible a la izquierda, tiene una clave candidata como su determinante” (Date, 2004, p.362).
- **Cuarta forma normal (4FN):** “una relación está en 4FN cuando se tiene la forma FNBC y no se tienen dependencias multivaluadas” (Kroenke, 2003).
- **Quinta forma normal (5FN):** “Una varrel R está en 5FN —también llamada forma normal de proyección-junta (FN/PJ) — si y solamente si cada dependencia de junta no trivial válida para R está implicada por las claves candidatas de R (Date, 2004, p.398).

1.6.1.4. Herramientas CASE

CASE es un acrónimo para Computer-Aided Software Engineering, principalmente son herramientas informáticas que permiten ayudar al diseño de modelos en la ingeniería del software, ya sean de bases de datos, estructurales o de comportamiento. Según [Ken05] las herramientas CASE deben tener los siguientes elementos:

- **Repositorio:** debe tener una estructura definida para que sea aplicable en un sistema manejador de bases de datos SDBD.
- **Meta modelo:** debe especificar las técnicas aplicadas en la metodología a aplicar según el tipo de base de datos a manejar.
- **Generador de documentación:** esta característica debe ofrecer documentos necesarios y requeridos en los procesos de ingeniería del software.
- **Carga o descarga de datos:** debe ofrecer la posibilidad de descargar scripts o archivos que permitan facilitar la creación en este caso de las bases de datos.
- **Comprobación de errores:** Son facilidades que permiten detectar inconsistencias en diseño estructuración del diseño creado por la herramienta.
- **Interfaz de usuario:** Debe permitir a un usuario por medio del uso de iconos, menús, pantallas, etc. permitir que se diseñen los modelos, diagramas o productos esperados en un proceso de ingeniería de software que estén incluidas dentro de una metodología establecida.

Estas características deben ser aplicables en la manera en cómo se implemente la metodología aplicada al tipo de base de datos que se quiera manejar, sin embargo no todas las CASE deben tener obligatoriamente estos componentes. Las CASE se pueden categorizar en tres productos principales:

- **Case de alto nivel:** de acuerdo a [Alb00] se entiende por una CASE de alto nivel cuando ésta sirve como herramienta en los procesos iniciales o las primeras fases en el ciclo de vida de un software, son utilizadas para la gestión de proyectos, análisis y diseño.
- **Case de bajo nivel:** estas están orientadas principalmente a las bases de datos, programas o pruebas
- **Case integrado:** estas CASE combinan las de alto y bajo nivel

1.6.1.5. Modelo seudomatemático para el diseño de bases de datos relacionales

Este es un modelo creado por el Ingeniero John Jairo Londoño Pérez el cual muestra una metodología para que se diseñen las bases de datos a partir de dos únicas preguntas: “¿Qué se quiere controlar?” y “¿Para quién se quiere controlar?”, estas preguntas se tienen en cuenta para

abstraer de la forma más precisa el contexto del sistema al que se quiere diseñar una solución en bases de datos. Esta metodología parte de los fundamentos matemáticos como lo son los axiomas de Armstrong, aplicables a las formas de normalización de las bases de datos, con la diferencia en que ésta propone nuevos términos más accesibles que resume de alguna manera el gran número de preguntas que se formulan originalmente dentro de los diferentes axiomas nombrados.

Dentro de los términos soportados por la metodología se tiene al sujeto el cual responde a la pregunta inicial “para quién se quiere controlar” y el grupo transaccional que responde al “qué se quiere controlar”.

Con base en lo anterior se propone realizar una cadena lógica del negocio que según Londoño (2011) *“permite a partir de las dependencias funcionales básicas, alinear todo lo concerniente con la ingeniería del software y renueva de alguna manera el pensamiento hasta hoy utilizado para contextualizar la problemática de los sistemas de información”*.

Se proponen entonces 5 esquemas para diseñar las bases de datos que se utilizan de acuerdo al contexto que de un sistema, todas ellas basadas en la relación que existe entre un sujeto y su grupo transaccional. Para poder identificar dichas relaciones se deben tener en cuenta las siguientes preguntas:

- ¿existe dependencia funcional, exclusiva del grupo transaccional respecto al sujeto?
- ¿existe dependencia funcional no exclusiva del grupo transaccional respecto al sujeto?
- ¿qué variables dependen funcional y estrictamente de los dos componentes de la llave?
- ¿qué variables dependen funcional y estrictamente del segundo componente de la llave?

1.6.1.6. Diseño de bases de datos normalización vs cadena lógica de negocio del sistema - dependencias funcionales (DFE – DFNE)

El proceso de normalización es una técnica en la que se proponen una serie de reglas para lograr diseñar el modelo relacional de una base de datos, a través del cual se busca eliminar la redundancia de los datos, garantizar la integridad de la información a manejar y optimizar el proceso de actualización de datos.

En el presente documento se muestra a través del proceso de normalización históricamente usado, la solución a los problemas de diseño definida por Modelo de Cadena Lógica de Negocio del Sistema - Dependencias Funcionales (Exclusivas y No Exclusivas) propuesto por el ingeniero Jhon Jairo Londoño.

Para efectos de ilustración se lleva a cabo la solución de un problema sencillo que consiste en diseñar el modelo de bases de datos que permita el almacenamiento de la información de clientes, facturas y productos.

El proceso de normalización se define a través del cumplimiento de varias etapas conocidas como Formas Normales, a saber:

1.6.1.6.1. Primera forma normal

Una tabla se encuentra en primera forma normal cuando:

- Los atributos de la tabla son atómicos, es decir, no tienen posibilidad de ser más divididos, por ejemplo el nombre de una persona, se puede separar en nombres y apellidos.
- La tabla tiene definida un campo que permite identificar de forma única e inequívoca todo el registro. Este campo se conoce como clave principal o llave primaria (PRIMARY KEY).

Cedula	Nombre	Ciudad	Factura	Fecha	Producto	Cantidad	Vr Unitario
123	Juan Pérez	Bogotá	1	15/07/15	Teclado	2	15000
123	Juan Pérez	Bogotá	1	15/07/15	Mouse	1	14000
234	Daniel Mendoza	Bogotá	2	16/07/15	Monitor	1	285000
234	Daniel Mendoza	Bogotá	2	16/07/15	Teclado	1	15000
234	Daniel Mendoza	Bogotá	2	16/07/15	Mouse	1	14000
345	Maria Perdomo	Cali	3	17/07/15	Monitor	1	285000
456	Claudia Forero	Villavicencio	4	18/07/15	Monitor	1	285000
456	Claudia Forero	Villavicencio	4	18/07/15	Teclado	1	15000
456	Claudia Forero	Villavicencio	4	18/07/15	Mouse	1	14000
456	Claudia Forero	Villavicencio	4	18/07/15	Parlantes	1	12000
456	Claudia Forero	Villavicencio	4	18/07/15	CPU	1	850000

Figura 1. Información antes de convertir a Primera Forma Normal

Cedula	Nombre	Apellido	Ciudad
123	Juan	Pérez	Bogotá
234	Daniel	Mendoza	Bogotá
345	Maria	Perdomo	Cali
456	Claudia	Forero	Villavicencio

Producto	Vr Unitario
Teclado	15000
Mouse	14000
Monitor	285000
Parlantes	12000
CPU	850000

Factura	Fecha
1	15/07/15
2	16/07/15
3	17/07/15
4	18/07/15

Ciudad
Bogotá
Cali
Villavicencio

Figura 2. Tablas en Primera Forma Normal

Se puede apreciar que cada tabla ahora cuenta con sus atributos separados y la asignación de una clave primaria candidata.

1.6.1.6.2. Segunda forma normal

Los atributos de la tabla que no son clave principal tienen correspondencia única con la clave principal, ejemplo, el nombre de una persona tiene correspondencia única con su cédula.

Cedula	Nombre	Apellido	CodCiudad
123	Juan	Pérez	1
234	Daniel	Mendoza	1
345	Maria	Perdomo	2
456	Claudia	Forero	3

CodCiudad	Ciudad
1	Bogotá
2	Cali
3	Villavicencio

CodProducto	Producto	Vr Unitario
1	Teclado	15000
2	Mouse	14000
3	Monitor	285000
4	Parlantes	12000
5	CPU	850000

Factura	Fecha
1	15/07/15
2	16/07/15
3	17/07/15
4	18/07/15

Figura 3. Tablas en Segunda Forma Normal

1.6.1.6.3. Tercera forma normal

Los atributos no clave deben ser completamente dependientes de la clave principal. Para el cumplimiento de esta FN, se crean las tablas que relacionan Clientes con Facturas y Facturas Con Productos.

Cedula	Nombre	Apellido	CodCiudad
123	Juan	Pérez	1
234	Daniel	Mendoza	1
345	Maria	Perdomo	2
456	Claudia	Forero	3

CodCiudad	Ciudad
1	Bogotá
2	Cali
3	Villavicencio

CodProducto	Producto	Vr Unitario
1	Teclado	15000
2	Mouse	14000
3	Monitor	285000
4	Parlantes	12000
5	CPU	850000

Factura	Fecha	Cedula	Productos
1	15/07/15	123	1,2
2	16/07/15	234	3,1,2
3	17/07/15	345	3
4	18/07/15	456	3,1,2,4,5

Figura 4. Tablas en Tercera Forma Normal

1.6.1.6.4. Cuarta forma normal

Los atributos que pueden tener multivalores se separan en dos o más relaciones independientes. El ejemplo se da para el caso de muchos productos en una factura y un producto en varias facturas.

Cedula	Nombre	Apellido	CodCiudad
123	Juan	Pérez	1
234	Daniel	Mendoza	1
345	Maria	Perdomo	2
456	Claudia	Forero	3

CodCiudad	Ciudad
1	Bogotá
2	Cali
3	Villavicencio

CodProducto	Producto	Vr Unitario
1	Teclado	15000
2	Mouse	14000
3	Monitor	285000
4	Parlantes	12000
5	CPU	850000

Factura	Fecha	Cedula
1	15/07/15	123
2	16/07/15	234
3	17/07/15	345
4	18/07/15	456

Factura	CodProducto	Cantidad	Vr Unitario
1	1	2	15000
1	2	1	14000
2	3	1	285000
2	1	1	15000
2	2	1	14000
3	3	1	285000
4	3	1	285000
4	1	1	15000
4	2	1	14000
4	4	1	12000
4	5	1	850000

Figura 5. Tablas en Cuarta Forma Normal

De cara al modelo del ingeniero Londoño (2011) se identifican los elementos de la Cadena Lógica de Negocio del Sistema como sigue:

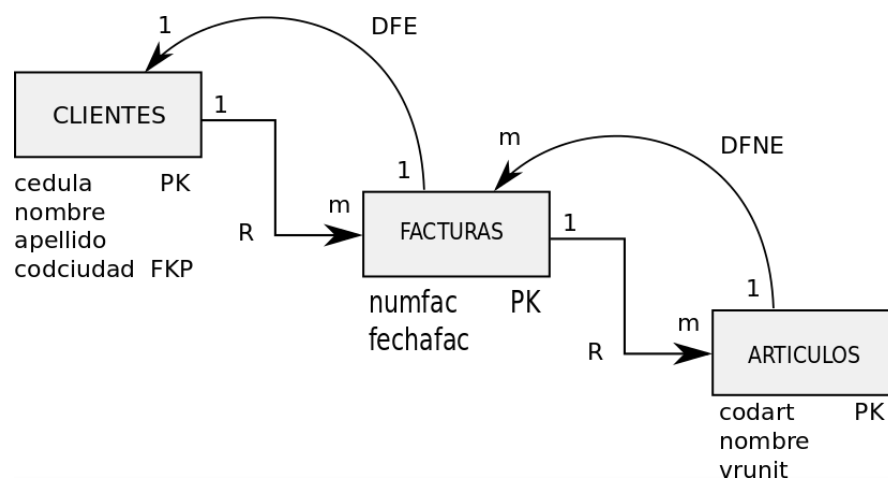


Figura 6. Cadena Lógica de Negocio del Sistema

Desde el análisis que se puede hacer para confrontar el cumplimiento del modelo propuesto por el ingeniero Londoño de las Formas Normales, se encuentra que:

- En la definición de los atributos de cada Sujeto o Grupo Transaccional, se definen como atómicos

- En cada uno se define un atributo como clave primaria PK

Estos elementos permiten obtener 1FN y 2FN

- Los atributos son completamente dependientes de la PK
- Los atributos definidos como Foreign Key Default FKD permite relacionar las tablas para Dependencias Funcionales Exclusivas y No Exclusivas.

Lo anterior nos permite obtener 3FN y 4FN

1.6.1.7. PROGRAMACIÓN EXTREMA

La programación extrema es un tipo de metodología ágil para el desarrollo de proyectos de software creada y formulada por el ingeniero Kent Beck en el año 1999, el primer libro sobre la metodología la escribió el mismo autor la cual la define como: “un cambio social en el cual se deja lado los hábitos y patrones del pasado para adoptar una nueva manera de optimizar el trabajo. Se trata de ser abierto acerca de lo que somos capaces de hacer y luego hacerlo. Y, lo que permite esperar que los demás hagan lo mismo” (Beck, 2005).

XP como estilo de software se enfoca en excelentes técnicas de programación, clara comunicación y el trabajo en equipo. La metodología según Beck (2005) propone los siguientes aspectos:

- Una filosofía de desarrollo de software basada en valores de comunicación, retroalimentación, simplicidad, coraje y respeto.
- Un conjunto de prácticas probadas para mejorar el desarrollo de software, prácticas que se complementan entre sí, amplificando sus efectos. Son elegidos como expresiones de los valores.
- Un conjunto de principios complementarios, técnicas intelectuales para traducir los valores en práctica.
- Una comunidad que comparte esos valores y muchos de las mismas prácticas.

1.6.1.7.1. Valores de XP

Los valores en XP son comportamientos que el equipo de trabajo debe adoptar para beneficio en la productividad del proyecto, Beck (2005) los define en su libro "Extreme Programming Explained" de la siguiente manera:

- **Comunicación:** el primer valor de XP es la comunicación. Los problemas con los proyectos invariablemente se remontan a alguien que no habla con otro sobre algo importante. A veces un programador no le dice a otra persona acerca de un cambio fundamental en el diseño. A veces un programador no le pide al cliente la pregunta correcta, por lo que se sople una decisión de dominio crítico. A veces, un administrador no le pide a un programador la pregunta correcta, y está mal informado del progreso del proyecto.
- **Simplicidad:** el entrenador XP pide al equipo, "¿Cuál es la cosa más simple que podría funcionar?" La simplicidad no es fácil. Es lo más difícil en el mundo no mirar hacia las cosas que necesitará para poner en práctica mañana y la próxima semana y el mes siguiente. Pero pensando compulsivamente por delante está escuchando el miedo del coste exponencial de la curva de cambio.
- **Retroalimentación:** la retroalimentación funciona en diferentes escalas de tiempo. En primer lugar, la retroalimentación trabaja en la escala de minutos y días. Los programadores escriben pruebas unitarias para toda la lógica en el sistema que podría romper. Los programadores tienen minuto a minuto comentarios sobre el estado de su sistema. Cuando los clientes escriben nuevas "historias", los programadores sin demora a ellos atienden a ellas en el menor tiempo posible, haciendo uso de los valores anteriores.
- **Coraje:** en el contexto de los tres primeros valores de la comunicación, la simplicidad, y la retroalimentación, es hora de revisar cómo el equipo está preparado para asumir los cambios que se presenten a medida que avanza el proyecto; estos cambios pueden ser en tiempo de ejecución de un producto, se debe tener valentía para saber cuándo desechar un código y adoptar una nueva solución para un problema del dominio.

1.6.1.7.2. Principios de XP

Los principios en XP ayudarán a elegir entre alternativas, los principios principalmente usan los valores, sin embargo se difieren de estos en la medida en que cada persona del equipo los

adopta de diferentes maneras; el principio es más concreto, y propone una serie de normas que el equipo debe asumir para el desarrollo de los proyectos. Los siguientes según Beck (2005) son las prácticas fundamentales, agrupadas en 3 categorías principales:

- **Retroalimentación:** esta categoría que contiene inmersas 4 prácticas ; la primera es un plan de *pruebas* en el cual se debe adoptar un periodo de pruebas de aceptación de una funcionalidad de la aplicación, en consecuencia se debe probar lo que debe hacer el software; la segundo es la *planificación* que asume las necesidades que tiene el proyecto para ser convertidas en actividades que el sistema debe realizar, esta fase contiene las historias de usuario como input del plan de liberación, el cual sirve para definir tiempos de entrega de la aplicación, y así poder recibir la retroalimentación del cliente; la tercer práctica es el *cliente in-situ* que consiste en la selección de un representante del cliente que hará parte del equipo de trabajo, permitiendo ayudar en la resolución de requisitos y aprobación de las soluciones; Por último tenemos la *programación por parejas* que consiste en escribir código con dos personas del equipo de trabajo compartiendo una sola máquina.
- **Proceso continuo en lugar de bloques:** esta categoría propone la *integración continua* que consiste en implementar nuevas funcionalidades del software si es necesario, con el fin de crear versiones estables para cada solución propuesta; otro principio de esta categoría es la *refactorización*, termino comúnmente usado en el ámbito del desarrollo de software el cual permite optimizar al máximo el código escrito. Por último las *entregas pequeñas* como práctica consiste en la evaluación del producto en el transcurso de 2, máximo 3 semanas para entregar a producción.
- **Entendimiento compartido:** esta última categoría propone en primer lugar el *diseño simple* que consiste en entregarle al cliente la solución que sea más simple y que cumpla con las historias propuestas; la *metáfora* como segunda práctica de esta categoría propone medir el éxito del proyecto mediante la definición de objetivos en una historia de usuario; *la propiedad colectiva del código* es una práctica que propone hacer del código una propiedad de todos, es decir que cada solución le pertenece al equipo y no al programador; *el estándar de programación* como en otras metodologías es una práctica sobre el cual todo proyecto debe basarse ya que define las normas para escribir y documentar código, de esta manera se puede entregar un producto de calidad; por último

se tiene el principio de *bienestar del programador* que consiste en que cada programador no sobrepase las 40 horas semanales de trabajo, permitiéndole generar código de más calidad.

1.6.1.7.3. Roles de la metodología XP

Los siguientes son los principales roles que se presentan en un proyecto basado en programación extrema en donde cada uno es responsable del éxito del proyecto.¹

- **Programador:** Escribe el código del software
- **Cliente:** Es la persona nombrada como representante del cliente para ayudar en la escritura de las historias de usuario.
- **Tester:** junto al cliente es quien realiza las pruebas del software.
- **Tracker:** Es el que proporciona la realimentación al equipo.
- **Coach:** Es el responsable del proceso global.
- **Consultor:** Es un miembro externo del equipo con un conocimiento específico en algún tema que es necesario para el proyecto
- **Big boss:** Es el vínculo entre clientes y programadores, se centraliza en la coordinación del cliente y el programador.

1.6.1.7.4. Ciclo de vida de un software según XP

El ciclo de vida de un software en XP es basado en 5 procesos principales:

- **Planeación:** se escriben las historias de usuario o requerimientos del cliente.
- **Gestión:** el equipo adopta la tecnología, se definen a grandes rasgos historias de usuario y se hace seguimiento al equipo por medio de reuniones diarias.
- **Diseño:** parte de la práctica de simplicidad, se revisan las historias de usuario y se programa la refactorización cuando sea necesario
- **Codificación:** el equipo de pares codifica la historia de usuario.
- **Pruebas:** el equipo prueba con el cliente la historia de usuario.

¹Extreme Roles, tomado el 20 de abril de : <http://c2.com/cgi/wiki?ExtremeRoles>

En la siguiente figura se puede observar una vista general del ciclo de vida de XP.

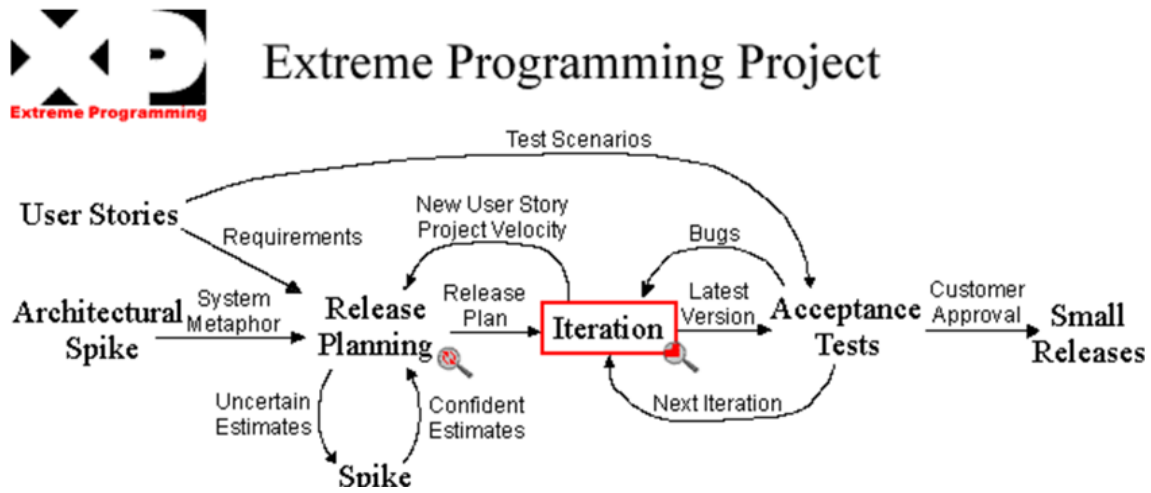


Figura 7. Ciclo de Vida XP tomada de <http://www.extremeprogramming.org/map/project.html>

1.6.1.8. Diseño de prototipos de software

Según Pressman (1993) un prototipo es un proceso que facilita al programador la creación de un modelo de software a construir, este proceso hace parte de los modelos evolutivos en la construcción del software, por lo general son iterativos y su principal característica es que permiten desarrollar versiones diferentes en cada iteración.

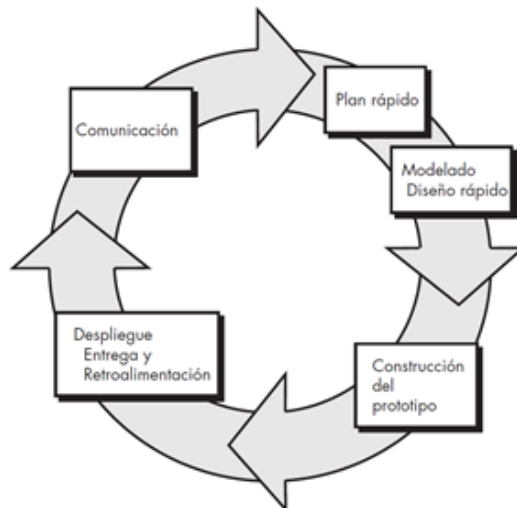


Figura 8. El paradigma de hacer prototipos, Pressman 2010

El paradigma de hacer prototipos comienza con comunicación, que es la fase inicial en donde se toman los requerimientos o se abstraen las necesidades iniciales del cliente, la fase siguiente nos lleva a un plan en donde se establecen los objetivos generales y las características básicas con las que el software debe contar, las dos fases siguientes construyen en código una versión inicial del prototipo que es desplegado para que reciba una retroalimentación por parte de los involucrados en el proyecto, para posteriormente realizar otra versión la cual mejore y afine los requerimientos. La iteración ocurre a medida de que el prototipo es afinado para satisfacer las necesidades de distintos participantes, y al mismo tiempo le permite a entender mejor lo que se necesita hacer. El ideal es que el prototipo sirva como mecanismo para identificar los requerimientos del software. Si va a construirse un prototipo, pueden utilizarse fragmentos de programas existentes o aplicar herramientas (por ejemplo, generadores de reportes y administradores de ventanas) que permitan generar rápidamente programas que funcionen (Pressman, 2010).

1.6.1.9. Webapps

Las aplicaciones web (webapps) es un tipo de software orientado esencialmente a la web. Con el desarrollo de la Web 2.0 y los lenguajes de programación orientados a la web, las webapps han ido creciendo en su confiabilidad y robustez, tanto así que hoy día las organizaciones confían en ellas para dar respuesta a sus crecientes necesidades. Poseen todas las características técnicas de calidad y diseño como cualquier software tradicional (Pressman, 2010)

1.6.1.10. La arquitectura MVC

La arquitectura MVC (controlador de la vista del modelo) es una arquitectura usada para el desarrollo de aplicaciones web, su función principal es separar el negocio de las interfaces gráficas al usuario (vistas); esto con el fin de hacer las aplicaciones más mantenibles y confiables, ya que estructurando un proyecto con esta arquitectura se aseguran los datos en la capa de negocio. El modelo contiene todo el contenido y la lógica de procesamiento específicos de la aplicación, incluso todos los objetos de contenido, acceso a fuentes de datos o información externos y todas las funciones de procesamiento que son específicas de la aplicación. La vista contiene todas las funciones específicas de la interfaz gráfica y permite la presentación de contenido y lógica de procesamiento, incluidos todos los objetos de contenido, el acceso a fuentes

de datos o información del exterior y todas las funciones de procesamiento que requiere el usuario final. El controlador administra el acceso al modelo y a la vista, y coordina el flujo de datos entre ellos (Pressman, 2010). En la siguiente figura se puede apreciar la estructura básica de un MVC.

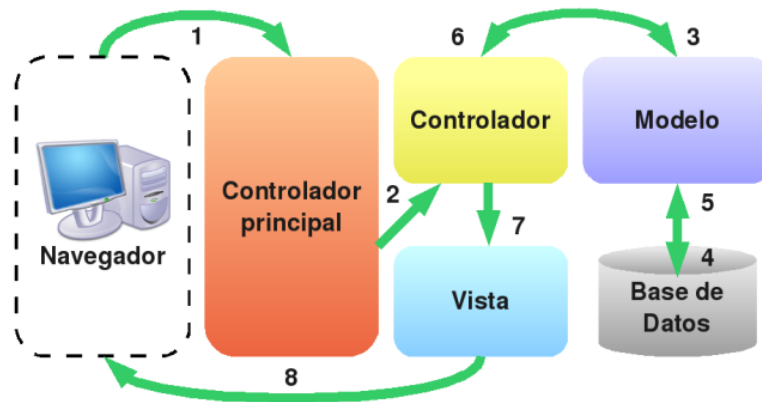


Figura 9. Estructura básica de un MVC.

Fuente: Bogotá, L.A. (2009, enero). Proyecto Red social para deportistas. Recuperado el 18 de septiembre de 2012, del sitio Web de la Universidad Politécnica de Madrid <http://www.upm.es/institucional>.

1.6.1.11. Lenguaje unificado de modelado (UML)

Según Grady Booch (2003) UML se define como un único lenguaje de modelamiento que permite especificar, visualizar y construir los artefactos de los sistemas de software. Es un sistema notacional destinado a los sistemas de modelado que utilizan conceptos orientados a objetos.

El lenguaje UML estandariza los artefactos y la notación, pero no define procesos oficiales de desarrollo. Larman (2003) define el UML como:

Un lenguaje para construir modelos; no guía al desarrollador en la forma de realizar el análisis y diseño orientado a objetos ni le indica cual proceso de desarrollo adoptar.

1.6.2. Marco conceptual

1.6.2.1. Cadena lógica del negocio del sistema (CLNS)

La cadena lógica del negocio permite determinar la dimensión del producto de software que se pretende construir y en consecuencia deberá alinear con todo componente que se involucre en su

desarrollo [Lon11], En lo que atañe al Diseño de la base de datos, es a partir de la cadena del negocio, como se hace posible aplicar los conceptos de las dependencias funcionales, resumidos en cuatro preguntas de la metodología del modelo pseudomatemático hablado. La CLNS se abstrae del contexto del sistema, no permite variaciones en su estructura ya que lo que reconoce precisamente es el modelamiento del comportamiento del sistema en una realidad, a partir de la definición de sujetos y grupos transaccionales.

1.6.2.2. Sujeto

Teniendo en cuenta la teoría del modelo pseudomatemático para el diseño de las bases de datos, se habla de la definición de un sujeto para poder diseñar una base de datos de acuerdo al contexto de un sistema; el sujeto se presenta como una entidad que responde a la pregunta de “¿para quién se quiere controlar?” (Londoño, 2011). Respondiendo a dicha pregunta el sujeto definido se presenta como principal entidad de la cadena lógica del negocio que se define del contexto del sistema; que no es más que una abstracción de una realidad. Cuando tengo claro en un diseño de bases de datos para qué entidad se va a controlar puedo definir claramente qué exactamente quiero controlar por medio de ese sujeto.

1.6.2.3. Grupo Transaccional

Los grupos transaccionales como respuesta a la pregunta “¿qué se va a controlar?” determinan otra entidad que conforma la cadena lógica del negocio. Se habla de grupos ya que pueden ser muchos los que son controlados por un solo sujeto en la CLNS (Londoño, 2011). En consecuencia para una CLNS se pueden tener un sujeto con un grupo transaccional, un sujeto con dos grupos transaccionales independiente, un sujeto con un grupo transaccional anidado, un sujeto con dos grupos transaccionales independientes y subgrupos transaccionales para cada grupo inicial y un sujeto que contenga dos grupos transaccionales independientes y uno de ellos contiene un subgrupo transaccional. Todas estas combinaciones se pueden dar en una CLNS la selección de cada una dependerá del análisis del contexto del sistema.

1.6.2.4. Dependencia Funcional

En un diseño de bases de datos existen relacionales las cuales sirven para conectar una o más entidades; dichas conexiones se establecen entre los atributos de cada entidad. Para el modelo pseudomatemático las relaciones se dan por medio de los atributos del sujeto y el grupo o los

grupos transaccionales. Las dependencias funcionales juegan un papel importante en la definición de las relaciones entre entidades, ya que estas establecen el grado de dependencia que tiene un sujeto con un grupo transaccional; se hace posible aplicar los conceptos de las Dependencias Funcionales en el modelo seudomatemático resumido en cuatro preguntas principales (Londoño, 2011), las cuales son:

- ¿Define la llave primaria del grupo transaccional una dependencia funcional exclusiva de la llave primaria del sujeto?
- ¿Define la llave primaria del grupo transaccional una dependencia funcional no exclusiva de la llave primaria del sujeto?
- ¿Qué variables del grupo transaccional definen una dependencia funcional estricta de la llave primaria del sujeto + la llave primaria del grupo transaccional?
- ¿Qué variables del grupo transaccional definen una dependencia funcional estricta de la llave primaria del grupo transaccional?

1.6.2.5. Dependencia Funcional Exclusiva

Cuando se tienen relaciones entre entidades que requieran que un grupo transaccional dependa directamente de un sujeto se presentan *dependencias funcionales exclusivas*; este término es propio del modelo seudomatemático creado por el Ing. Londoño para referenciar los atributos del ente transaccional con respecto al sujeto, está dada, en comparación con la teoría de bases de datos, como una relación de uno a muchos.

1.6.2.6. Dependencia Funcional No Exclusiva

Cuando se establece en la cadena lógica del negocio del sistema que un grupo transaccional depende del sujeto directamente y ese sujeto también tiene una dependencia con el grupo transaccional se determina que estas dos entidades tienen una dependencia funcional no exclusiva; en comparación con la teoría de bases de datos estas relaciones se ven como relaciones de muchos a muchos, con el modelo seudomatemático se resuelven siempre con la llave compuesta (Londoño, 2011).

1.6.2.7. Prototipo

Los prototipos de software sirven para comprender rápidamente aspectos que un cliente quiera conocer sobre la solución que se ofrecerá; la idea principal es minimizar el riesgo y la

incertidumbre del producto final, ya que a medida que el cliente va solicitando requerimientos se van entregando versiones de la aplicación que resolverá el problema; esto ayudará, en gran medida, a conocer cuál será el resultado del desarrollo desde el principio del proyecto. Los prototipos vienen definidos dentro de un proceso evolutivo de software, hay que aclarar que un prototipo no es un sistema completo sino que establece, dentro de su marco de trabajo, el versionamiento de los requerimientos del cliente, ya que dentro de los procesos evolutivos se establecen iteraciones que se van refinando a medida que el cliente va probando la solución dada. Según [Pre10] “un prototipo es aquel que describe una interacción entre la máquina y el hombre, facilitando una comprensión de la forma en cómo se da esa interacción, dando también una visión de las funciones que el programa existente puede contener para así lograr una función deseada al finalizar el mismo”.

1.6.2.8. Laravel

Laravel es un framework de php basado en la arquitectura MVC el cual provee una serie de herramientas y utilidades para realizar aplicaciones de manera rápida, segura y con una sintaxis elegante. “Laravel intenta eliminar las molestias del desarrollo facilitando las tareas comunes que se utilizan en la mayoría de los proyectos web, como la autenticación, enrutamiento, sesiones, y el almacenamiento en caché”².

1.6.2.9. PHP

PHP por sus siglas hypertext preprocessor es un lenguaje de programación de código abierto utilizado para desarrollos web, según la documentación oficial de php: “es un lenguaje de propósito general ampliamente utilizado que es especialmente adecuado para el desarrollo web y puede ser embebido en páginas HTML. Su sintaxis es similar a C, Java y Perl y es fácil de aprender. El objetivo principal del lenguaje es permitir a los desarrolladores web escriban páginas web generadas dinámicamente con rapidez”³.

1.6.2.10. Postgres

Postgre Sql es una base de datos relacional orientada a objetos, es de código abierto por lo que cuenta con una comunidad de desarrolladores que han versionado la aplicación desde el año 1986

² Laravel documentation, consultado el 20 de abril de 2016 de: <http://web.archive.org/web/20140920185439/http://laravel.com/docs/introduction>

³ Php documentation, consultado el 20 de abril de 2016 de: <https://secure.php.net/manual/en/preface.php>

hasta la fecha cuya última versión estable es la 9.5.1 del 11 de febrero de 2016. Según la documentación oficial de PostgreSQL “es un sistema de base de datos de gran alcance, de código abierto objeto-relacional. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de fiabilidad, integridad de datos y la corrección. Se ejecuta en todos los principales sistemas operativos, incluyendo Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows. Es totalmente compatible con ACID, tiene soporte completo para claves foráneas, combinaciones, vistas, triggers y procedimientos almacenados (en varios idiomas). Incluye más de SQL: 2008 tipos de datos, incluyendo entero, numérico, Boolean, CHAR, VARCHAR, DATE, INTERVALO y TIMESTAMP. También es compatible con el almacenamiento de objetos binarios grandes, como imágenes, sonidos, o de vídeo. Tiene interfaces de programación nativo de C / C ++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre otros, y la documentación excepcional”.⁴

1.7. Metodología de la investigación

1.7.1. Tipo de estudio

El tipo de estudio planteado para esta investigación será descriptiva; ya que se pretende caracterizar una metodología existente proponiendo una solución que ayude como herramienta en la aplicación y uso de la misma.

1.7.2. Método de investigación

En esta investigación se acude al método de análisis, proponiendo inicialmente la descomposición de la metodología de estudio; en nuestro caso “el modelo seudomatemático para el diseño de las bases de datos relacionales”. Esta metodología parte del análisis de las teorías de Boyce-Codd y Armstrong para lo cual se mostrará la aplicación de las mismas en la solución planteada. La información recabada va a servir de entrada para la definición de los requerimientos funcionales de la aplicación a desarrollar, una vez se pueda caracterizar esto, se analiza de la teoría de desarrollo de procesos evolutivos de software, específicamente el desarrollo de prototipos, la cual apoya en la construcción del objeto de estudio.

⁴ Postgre Sql documentation, consultada el 20 de abril de 2016 de: <http://www.postgresql.org/about/>

1.7.3. Fuentes y técnicas para la recolección de la información

La principal fuente de la que se toma la teoría principal es el libro del Ingeniero John Londoño, ya que es el creador de la metodología mencionada en este documento y de la que se pretende desarrollar un prototipo de software que sirva de apoyo para la misma.

El libro de ingeniería de software de Roger Pressman 2010; se convierte en fuente de información secundaria ya que de él se toman las secciones que hablan sobre el desarrollo de prototipos, requerimientos, calidad y gestión de proyectos.

La principal técnica de recolección de información consiste en la realización de entrevistas, una de las cuales se usa con el creador de la metodología de estudio, el ingeniero John Londoño, pues sirve para aclarar dudas acerca de la solución que se propone en cuanto a la herramienta.

También se planea efectuar reuniones en las cuales el Ingeniero Londoño pruebe de cierta manera si la solución que se está proponiendo facilita la aplicación de su metodología de diseño de las bases de datos relacionales.

1.7.4. Tratamiento de la información

Las entrevistas realizadas al ingeniero Londoño sirven para comprender los objetivos iniciales de su metodología; variables, condiciones y pensamiento acerca del objeto de estudio; esto facilita, en gran medida, la recopilación de información del propio participante, los cuales se plasmarán en los requerimientos funcionales del prototipo propuesto.

Se espera que la información obtenida de la teoría de ingeniería de software se vea aplicada en los diferentes diagramas UML que se establezcan para la solución planteada.

1.8. Organización del trabajo de grado

INTRODUCCIÓN.....	7
PARTE I. CONTEXTUALIZACIÓN DE LA INVESTIGACIÓN.....	8
1. CAPÍTULO I. DESCRIPCIÓN DE LA INVESTIGACIÓN.....	9
1.1. Planteamiento/identificación del problema	9
1.2. Formulación del problema.....	10
1.3. Objetivos	10
1.4. Justificación del trabajo/investigación	11
1.5. Hipótesis.....	12
1.6. Marco referencial.....	12
1.7. Metodología de la investigación.....	29
1.8. Organización del trabajo de grado	31
PARTE II. DESARROLLO DE LA INVESTIGACIÓN.....	32
2. CAPÍTULO II. CARACTERIZACIÓN DEL MODELO SEUDOMATEMÁTICO PARA EL DISEÑO DE LAS BASES DE DATOS RELACIONALES.....	33
2.1. La cadena lógica del negocio	33
2.2. CLNS Monofuncional Simple con Dependencia Funcional Exclusiva (DFE).....	34
2.3. CLNS Monofuncional Compuesta con Dependencia Funcional Exclusiva (DFE).....	37
2.4. CLNS Bifuncional Simple con Dependencia Funcional Exclusiva (DFE).....	41
2.5. CLNS Monofuncional Compuesta con Dependencia Funcional No Exclusiva (DFNE)	45
2.6. CLNS Bifuncional Simple con Dependencia Funcional No Exclusiva (DFNE).....	49
2.7. CADENA DEL NEGOCIO HIBRIDO “GENÉRICO” (SIMPLE Y COMPUESTA).....	53
BIBLIOGRAFÍA	58

PARTE II. DESARROLLO DE LA INVESTIGACIÓN

2. CAPÍTULO II. CARACTERIZACIÓN DEL MODELO SEUDOMATEMÁTICO PARA EL DISEÑO DE LAS BASES DE DATOS RELACIONALES

2.1. La cadena lógica del negocio

La cadena del negocio del sistema permite tener un contexto sobre la lógica del negocio al cual va a apuntar la solución propuesta en el diseño de nuestras bases de datos. Teniendo en cuenta lo anterior tenemos los siguientes temas a abordar:

Tabla 1 *Descripción del Modelo Seudomatemático*

En el sistema	En la base de datos
Tenemos la CLNS(Cadena Lógica del Negocio del Sistema)	<ul style="list-style-type: none">• Para las dependencias funcionales exclusivas DFE.• Para las dependencias funcionales no exclusivas DFNE.
<ul style="list-style-type: none">• Monofuncional simple.• Monofuncional compuesta.• Bifuncional simple.• Bifuncional compuesta.• Bifuncional híbrida.	Se soporta el diseño con 4 modelos principales: <ul style="list-style-type: none">• Modelo de dependencias funcionales.• Modelo relacional.• Modelo entidad relación.• Modelo DLL (lenguaje de definición de datos).

Nota: Decir que se desarrollan sistemas es erróneo, ya que el sistema debe existir antes de que se dé un diseño de software; el sistema existente debe contextualizarse y comprenderse a través de 2 preguntas principales:

1. ¿Para qué se va a controlar? → sujeto.
2. ¿Qué se va a controlar? → grupo transaccional

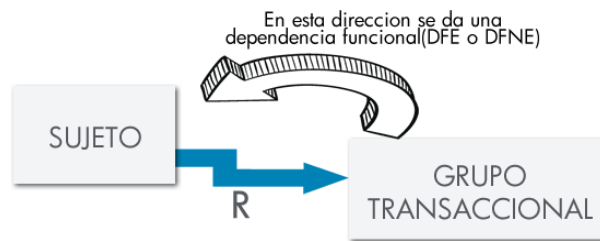


Figura 10. *Representación gráfica del sujeto y el grupo transaccional*

Nota: la dependencia que existe entre el sujeto y el grupo transaccional se llama relación(R). El tipo de relación la determina el tipo de dependencia funcional.

2.2. CLNS Monofuncional Simple con Dependencia Funcional Exclusiva (DFE)

Una DFE se define cuando un registro de elementos del componente sucesor de la CLNS sólo depende de un registro del componente antecesor.

El concepto de pertinencia se refleja a medida que creamos las cadenas ya que corresponde a los atributos son efectivamente los que corresponden a cada componente.

En este orden de ideas es importante manejar las siguientes abreviaturas para poner los atributos de nuestras entidades:

- FKD – Foreign Key por Defecto.
- FKP – Foreign Key por Proceso. El componente sucesor hereda la PK del componente antecesor.
- PKE – Primary Key Emergente.

La siguiente es la estructura de una cadena lógica del negocio monofuncional simple con DFE, según Londoño (2011) es monofuncional simple porque solo se tiene en la cadena un sujeto con un grupo transaccional.

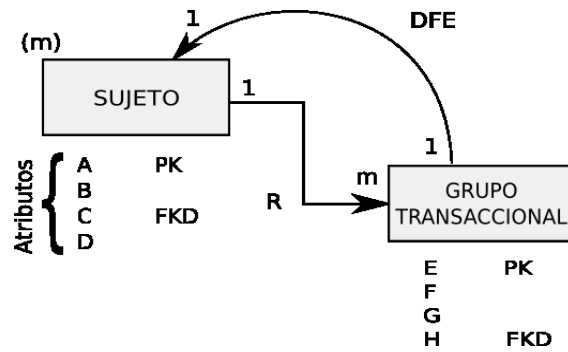


Figura 11. Cadena Lógica del Negocio Monofuncional Simple con DFE

Diseño por Modelo de Dependencias Funcionales (DFE)

Este diseño corresponde a la vista de las tablas que se crean a partir del diseño de una CLNS. En el desarrollo de una CLNS se crean las tablas principales que corresponden al sujeto y grupo transaccional, en el diseño monofuncional serían 2; las tablas auxiliares corresponden a las creadas a partir de la definición de atributos que sean de tipo FKD; en el caso de la figura anterior dos tablas auxiliares de las llaves C Y H.

Tablas Estructurales del
Diseño

Tabla_1: {A, B, C, D}

PK FKD

Tabla_2: {E, F, G, H, A}

PK FKD FKP

Tablas Auxiliares
del Diseño

Tabla_3: {C, na1_3, na2_3}

PKE

Tabla_4: {H, na1_4, na2_4}

PKE

Diseño por Modelo Relacional

El diseño del modelo relacional hace referencia al modelo tradicional de Edgar Frank Codd en 1970, sin embargo según la metodología del modelo pseudomatemático para diseñar ese modelo se debe partir del orden de la CLNS partiendo de la tabla que contenga el último elemento de la cadena, seguido de las tablas auxiliares de esta y continuando con el elemento sucesor de la CLNS.

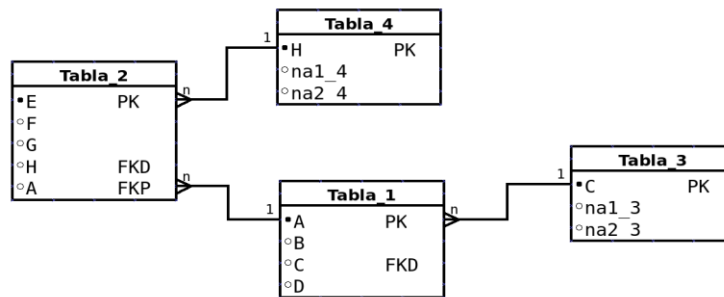


Figura 12. *Modelo Relacional Monofuncional Simple con DFE*

Cabe destacar que para diseñar este modelo no deben existir cruces entre las relaciones. Esto con el fin de tener un modelo más legible y entendible.

Las artistas de contacto son únicamente, la arista izquierda de la tabla de la derecha y la arista derecha de la tabla de la izquierda.

Diseño por Modelo Entidad – Relación

El diseño del modelo entidad relación corresponde al modelo de Peter Chen sin embargo en el modelo pseudomatemático se hace un cambio en cuanto a la manera como se ordenan las entidades; en este caso las mismas del modelo relacional explicado en el ítem anterior

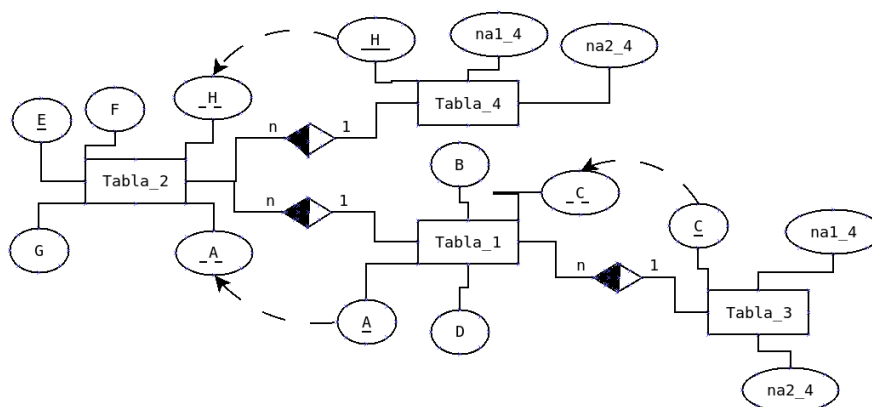


Figura 13. *Modelo Entidad Relación Monofuncional Simple con DFE*

Diseño Modelo DDL

El modelo DDL corresponde a la codificación de las entidades como resultado del diseño de una CLNS, esta codificación en un script en SQL estándar que según Londoño (2011) debe funcionar y servir para cualquier motor de bases de datos. La creación de este script va conforme al desarrollo de la cadena, puesto que parte de las tablas del último elemento, con sus auxiliares y finaliza en el primero; esto permite que el motor al crear la base de datos no encuentre inconsistencias en la integridad referencial de las tablas.

```
CREATE TABLE Tabla_3
(
    C            int,
    na1_3        varchar[20],
    na2_3        varchar[10],
    primary key (C)
);
CREATE TABLE Tabla_4
(
    H            int,
    na1_4        varchar[15],
    na2_4        int,
    primary key (H)
);
CREATE TABLE Tabla_1
(
    A            int,
    B            varchar[20],
    C            int,
    D            int,
    primary key (A),
    foreign key (C) references Tabla_3 (C)
);
CREATE TABLE Tabla_2
(
    E            int,
    F            int,
    G            varchar[15],
    H            int,
    A            int,
    primary key (E),
```


foreign key (H) references Tabla_4 (H),
foreign key (A) references Tabla_1 (A)
);

2.3. CLNS Monofuncional Compuesta con Dependencia Funcional Exclusiva (DFE)

Esta cadena hace referencia a los elementos de un sujeto que tiene dos grupos transaccionales, esto lo podemos observar en la siguiente figura.

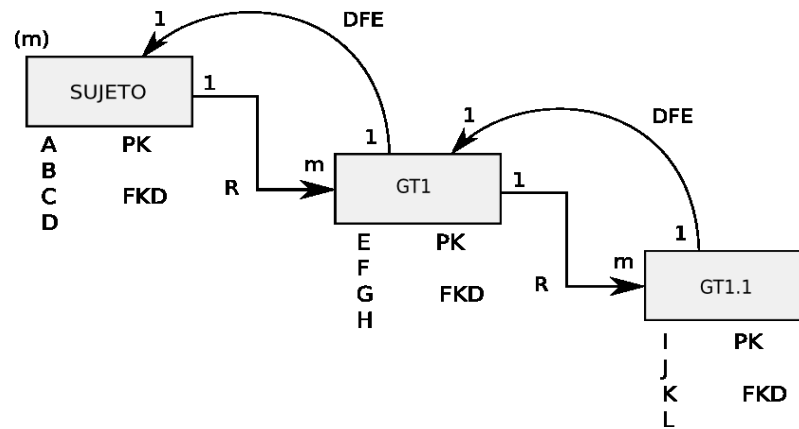
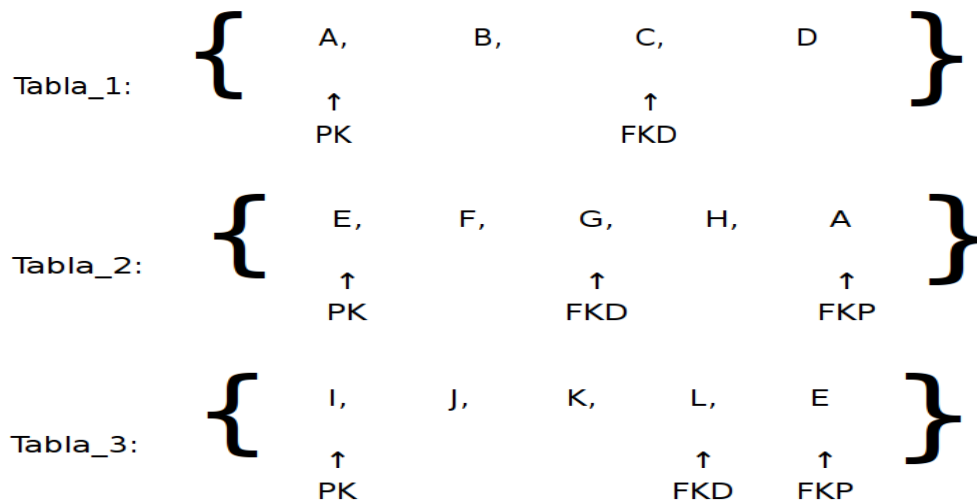


Figura 14. *CLNS Monofuncional Compuesta con Dependencia Funcional Exclusiva (DFE)*

Cabe resaltar que el sujeto tiene un grupo transaccional directo GT1 que se compone de otro; es decir que el sujeto controla GT1.1 a través de GT1.

Diseño Modelo de Dependencias Funcionales (DFE)

Para este caso como mínimo debemos tener 3 tablas principales que corresponden al sujeto y los 2 grupos transaccionales



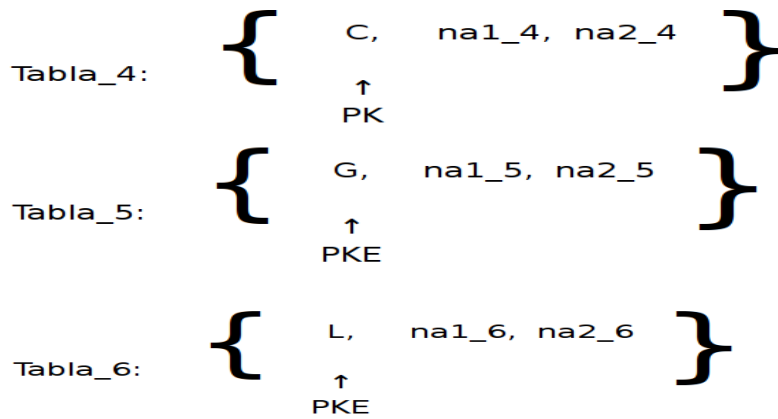


Figura 15. *Modelo de Dependencias Funcionales Monofuncional Compuesta con DFE*

Diseño Modelo Relacional

Todos los tipos de cadena comparten los mismos principios del modelo seudomatemático, esto quiere decir que tanto el modelo relacional como el entidad – relación parten del principio de diseñar primero las tablas correspondientes al último elemento de la cadena.

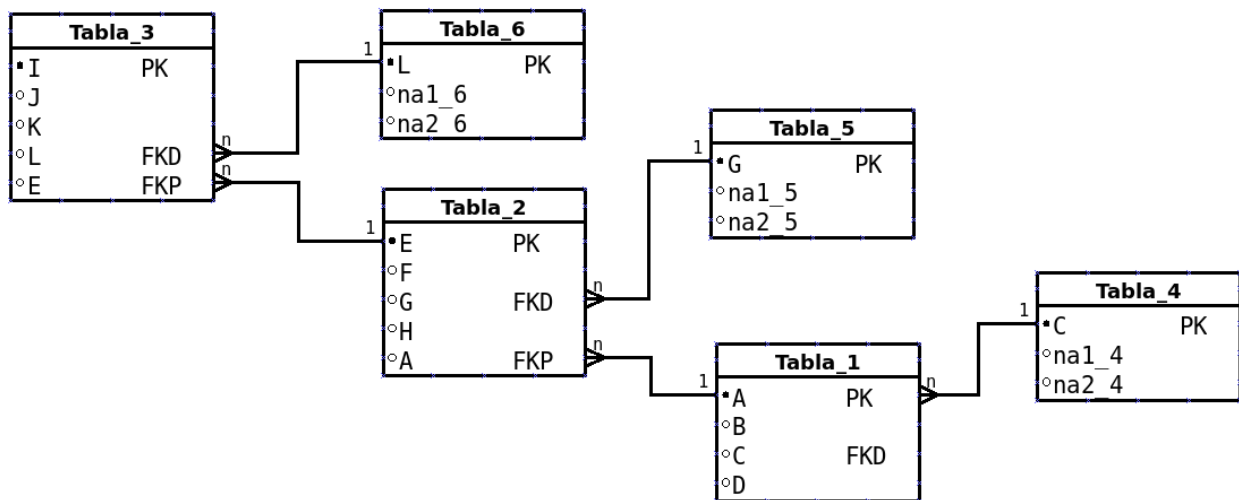


Figura 16. *Modelo Relacional Monofuncional Compuesta con DFE*

Diseño Modelo Entidad – Relación

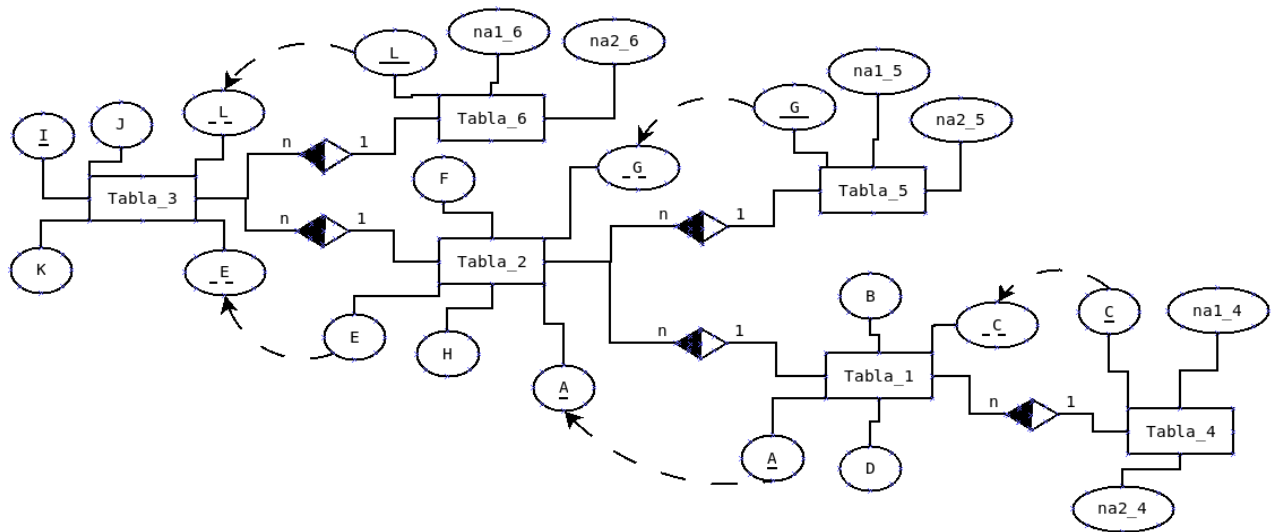


Figura 17. Modelo Entidad Relación Monofuncional Compuesta con DFE

Diseño Modelo DDL

```
CREATE TABLE Tabla_4
(
  C      int,
  na1_4 varchar[30],
  na2_4 int,
  primary key (C),
);
```

```
CREATE TABLE Tabla_5
(
  G      int,
  na1_5 varchar[30],
  na2_5 int,
  primary key (G),
);
```

```
CREATE TABLE Tabla_1
(
  A      int,
  B      varchar[30],
  C      int,
  D      int,
```

```
primary key (A),
foreign key (C) references Tabla_4(C),
);
```

```
CREATE TABLE Tabla_6
(
  L      int,
  na1_6 varchar[30],
  na2_6 int,
  primary key (L),
);
```

```
CREATE TABLE Tabla_2
(
  E      int,
  F      varchar[30],
  G      int,
  H      int,
  A      int,
  primary key (E),
  foreign key (G) references Tabla_5(G),
  foreign key (A) references Tabla_1(A),
);
```

```
CREATE TABLE Tabla_3
(
  I      int,
  J      varchar[30],
  K      int,
  L      int,
  E      int,
  primary key (I),
  foreign key (L) references Tabla_6(L),
  foreign key (E) references Tabla_2(E),
);
```

2.4. CLNS Bifuncional Simple con Dependencia Funcional Exclusiva (DFE)

La característica de esta cadena es la dependencia directa de un sujeto con dos grupos transaccionales, en este caso el sujeto controla dos grupos que son indiferentes uno del otro.

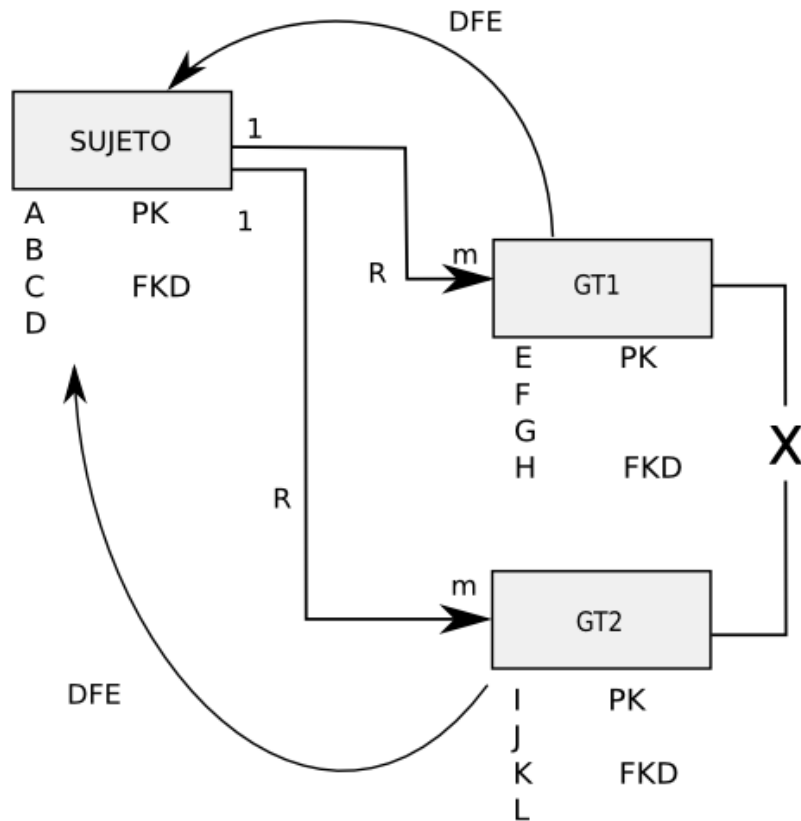


Figura 18. *CLNS Bifuncional Simple con Dependencia Funcional Exclusiva (DFE)*

Notemos en la figura anterior que GT1 Y GT2 son dos grupos que no se conectan entre sí, pero que comparten un elemento controlador

Diseño por Modelo de Dependencias Funcionales (DFE)

Tabla_1: {A, B, C, D}

PK FKD

Tabla_2: {E, F, G, H, A}

PK FKD FKP

Tabla_3: {I, J, K, L, A}

PK FKD FKP

Tabla_4: {C, na1_4, na2_4}

PKE

Tabla_5: {G, na1_5, na2_5}

PKE

Tabla_6: {K, na1_6, na2_6}

PKE

Diseño Modelo Relacional

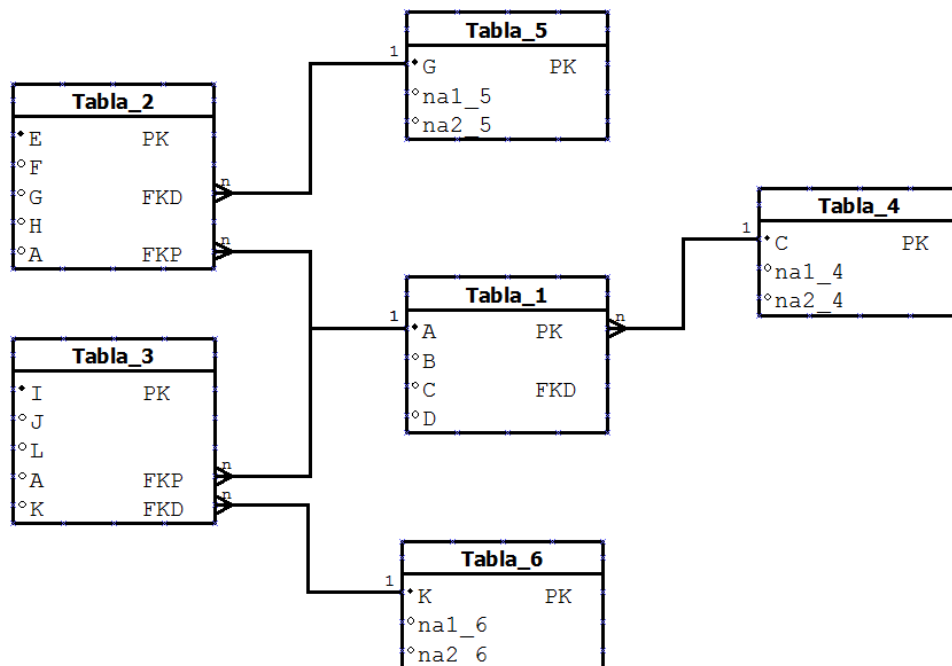


Figura 19. Modelo Relacional Bifuncional Simple con Dependencia Funcional Exclusiva (DFE)

Diseño Modelo Entidad – Relación

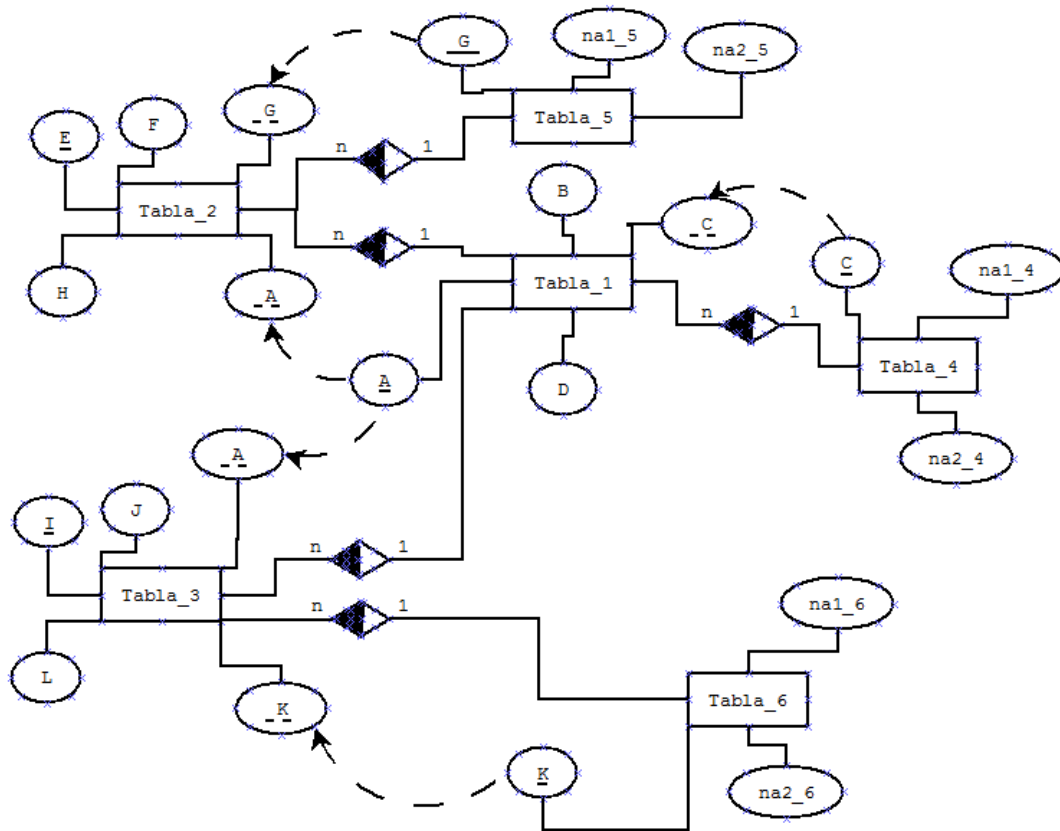


Figura 20. Modelo Entidad Relación Bifuncional Simple con Dependencia Funcional Exclusiva (DFE)

Diseño Modelo DDL

```
CREATE TABLE Tabla_4
(
    C          int,
    na1_4      varchar[20],
    na2_4      varchar[20],
    primary key (C)
);
```

```
CREATE TABLE Tabla_6
(
    K          int,
    na1_6      varchar[20],
    na2_6      varchar[20],
```

```
        primary key (K)
);
```

```
CREATE TABLE Tabla_5
(
    G            int,
    na1_5        varchar[20],
    na2_5        varchar[20],
    primary key (G)
);
```

```
CREATE TABLE Tabla_1
(
    A            int,
    B            varchar[20],
    C            int,
    D            int,
    primary key (A),
    foreign key (C) references Tabla_4 (C)
);
```

```
CREATE TABLE Tabla_2
(
    E            int,
    F            varchar[20],
    G            int,
    H            int,
    A            int,
    primary key (E),
    foreign key (G) references Tabla_5 (G),
    foreign key (A) references Tabla_1 (A)
);
```

```
CREATE TABLE Tabla_3
(
    I            int,
    J            varchar[20],
    K            int,
    L            int,
    A            int,
    primary key (I),
    foreign key (K) references Tabla_6 (K),
    foreign key (A) references Tabla_1 (A)
);
```


2.5. CLNS Monofuncional Compuesta con Dependencia Funcional No Exclusiva (DFNE)

En el ejemplo dado ya conocemos las cadenas monofuncionales compuestas con dependencia exclusiva, lo cual en el lenguaje de base de datos se conoce como una relación de uno a muchos del elemento sujeto al grupo. Para este tipo de cadena se tienen la misma monofuncionalidad pero con la **no** exclusividad entre los elementos de las cadenas, lo que indica que puedo tener relaciones de uno a muchos del grupo al sujeto.

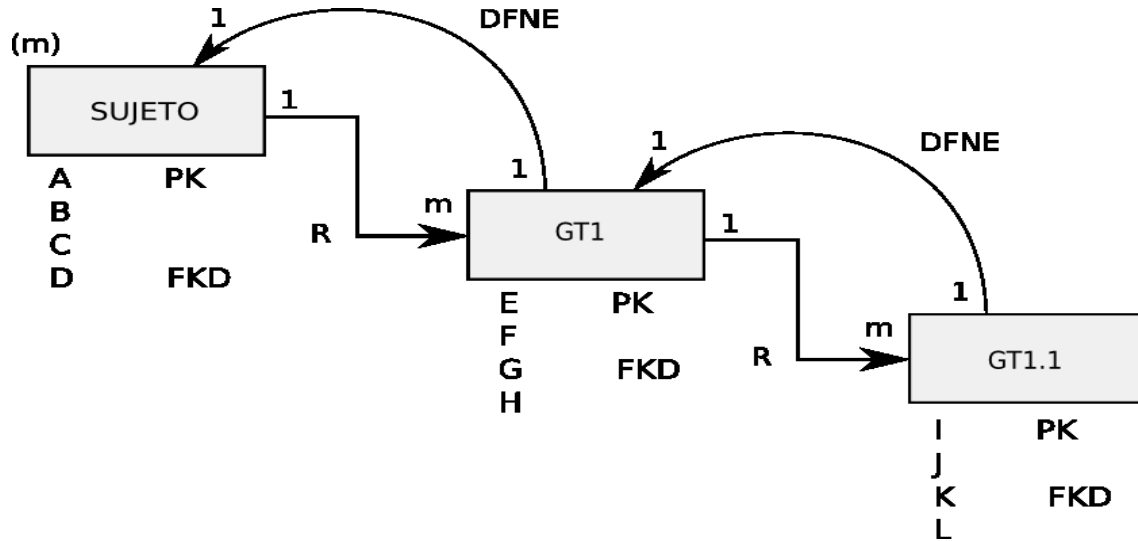


Figura 21. CLNS Monofuncional Compuesta con Dependencia Funcional No Exclusiva (DFNE)

Diseño por Modelo de Dependencias Funcionales

Este tipo de cadenas se solucionan por medio de creación llaves compuestas las cuales se construyen a partir de las llaves primarias de las dos tablas relacionadas cuando contienen dependencias funcionales no exclusivas.

$$\text{Tabla_1: } \left\{ \begin{array}{c} \underline{A}, \\ \uparrow \\ \text{PK} \end{array} \quad B, \quad C, \quad \begin{array}{c} \underline{D} \\ \uparrow \\ \text{FKD} \end{array} \right\}$$

$$\text{Tabla_2: } \left\{ \begin{array}{c} \underline{A+E}, \\ \uparrow \\ \text{PKE} \end{array} \quad F, \quad \begin{array}{c} \underline{G} \\ \uparrow \\ \text{FKD} \end{array} \right\}$$

$$\text{Análisis 1} \quad A + E \leftarrow F, G \text{ (Suposición)}$$

$$\text{Tabla_3: } \left\{ \begin{array}{c} \underline{E}_r \\ \uparrow \\ \text{PK} \end{array} \quad H \right\}$$

Analysis 2 $E \leftarrow H$ (Suposicion)

$$\text{Tabla_4: } \left\{ \begin{array}{c} \underline{A+E+I}_r \\ \uparrow \\ \text{PKE} \end{array} \quad \begin{array}{c} \underline{K}_r \\ \uparrow \\ \text{FKD} \end{array} \quad L \right\}$$

Analysis 2 $A+E+I \leftarrow K, L$ (Suposicion)

$$\text{Tabla_5: } \left\{ \begin{array}{c} \underline{I}_r \\ \uparrow \\ \text{PK} \end{array} \quad J \right\}$$

Analysis 2 $I \leftarrow J$ (Suposicion)

$$\text{Tabla_6: } \left\{ \begin{array}{c} \underline{D}_r \\ \uparrow \\ \text{PKE} \end{array} \quad \text{na1_6,} \quad \text{na2_6} \right\}$$

$$\text{Tabla_7: } \left\{ \begin{array}{c} \underline{G}_r \\ \uparrow \\ \text{PKE} \end{array} \quad \text{na1_7,} \quad \text{na2_7} \right\}$$

$$\text{Tabla_8: } \left\{ \begin{array}{c} \underline{K}_r \\ \uparrow \\ \text{PKE} \end{array} \quad \text{na1_8,} \quad \text{na2_8} \right\}$$

Figura 22. *Modelo de Dependencias Funcionales Monofuncional Compuesta con (DFNE)*

Diseño Modelo Relacional

En el caso de las dependencias funcionales no exclusivas se diseñan conforme a los principios anteriormente descritos con la diferencia en que las relaciones se identifican mediante una línea vertical que cruza la relación en los vértices de “muchos”.

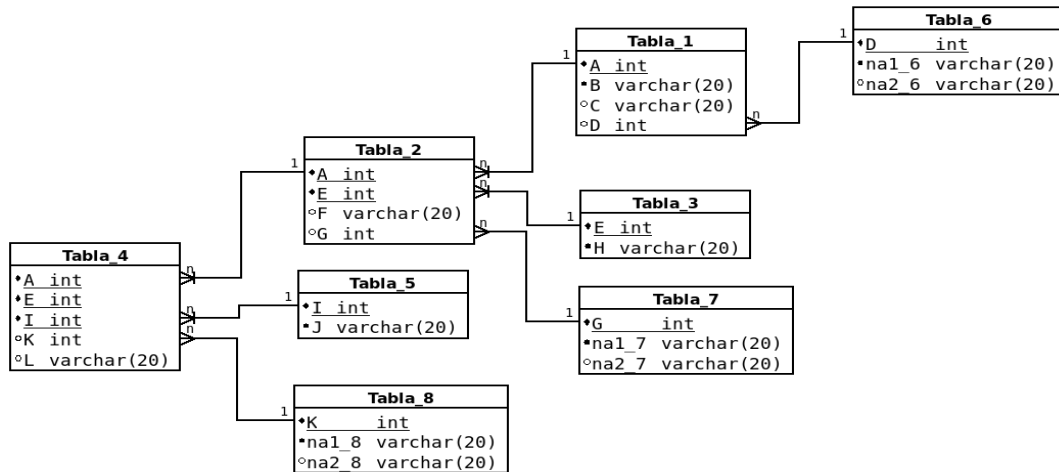


Figura 23. Modelo Relacional CLNS Monofuncional Compuesta con (DFNE)

Diseño Modelo Entidad – Relación

Para diseñar un modelo entidad relación con DFNE se debe marcar las relaciones fuertes - es decir, las que contienen las llaves compuestas - con un rombo indicando la relación con un vértice que ingresa directamente a la entidad.

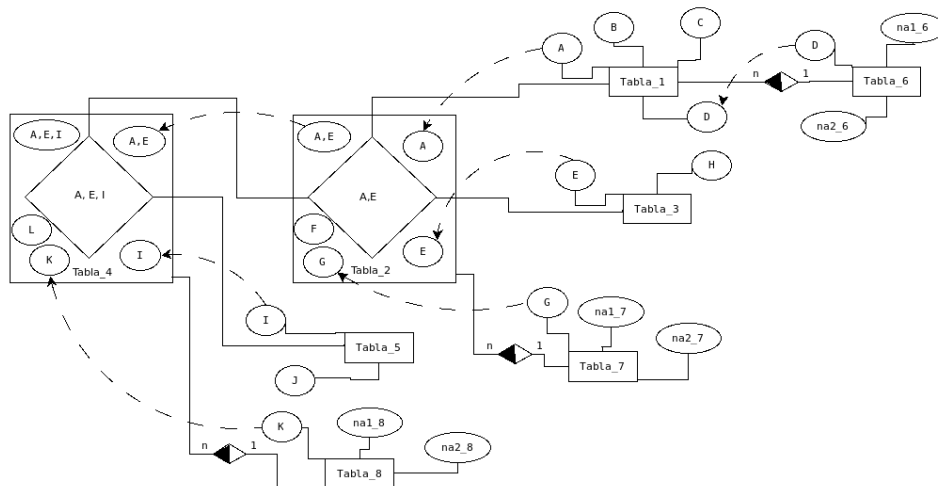


Figura 24. Modelo Entidad Relación Monofuncional Compuesta con (DFNE)

Como podemos observar en la figura anterior las relaciones débiles, que no son llave compuesta, no ingresan directamente al rombo sino que se expresan por fuera de la entidad.

Diseño Modelo DDL

Las dependencias funcionales no exclusivas no interfieren en el orden que ya hemos establecido para crear el script, ya que se continúa partiendo del último elemento de la cadena y finalizando en el primero.

```
CREATE TABLE Tabla_6
(
  D      int,
  na1_6 varchar[20],
  na2_6 varchar[15],
  primary key (D)
);
```

```
CREATE TABLE Tabla_1
(
  A      int,
  B      varchar[20],
  C      int[8],
  D      int,
  primary key (A),
  foreign key (D) references Tabla_6(D)
);
```

```
CREATE TABLE Tabla_3
(
  E      int,
  H      varchar[20],
  primary key (E)
);
```

```
CREATE TABLE Tabla_7
(
  G      int,
  na1_7 varchar[20],
  na2_7 varchar[15],
  primary key (G)
);
```

```
CREATE TABLE Tabla_2
(
  A      int,
```

```

E      int,
F      varchar[20],
G      int[8],
primary key (A,E),
foreign key (G) references Tabla_7(G)
);

```

```

CREATE TABLE Tabla_5
(
I      int,
J      varchar[20],
primary key (I)
);

```

```

CREATE TABLE Tabla_8
(
K      int,
na1_8 varchar[20],
na2_8 varchar[15],
primary key (K)
);

```

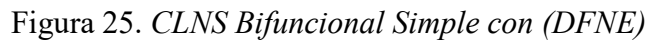
```

CREATE TABLE Tabla_4
(
A      int,
E      int,
I      int,
K      int,
L      varchar[20],
primary key (A,E,I)
);

```

2.6. CLNS Bifuncional Simple con Dependencia Funcional No Exclusiva (DFNE)

Para este modelo tenemos un sujeto que tiene una relación de dependencia no exclusiva con sus grupos transaccionales lo que conlleva a realizar llaves compuestas para cada relación tal como se explicó en el modelo anterior.



Tabla_1:

	{	A,	B,	C,	D	}
		↑			↑	
		PK			FKD	

Tabla_2: { $\frac{A+E}{\uparrow \text{PKE}}$, F, $\frac{H}{\uparrow \text{FKD}}$ }

Tabla_3: $\left\{ \begin{array}{c} \underline{E}, \\ \uparrow \\ \text{PK} \end{array} \quad G \right\}$

Tabla_4: $\left\{ \begin{array}{c} \underline{A+I_r} \\ \uparrow \\ \text{PKE} \end{array} \right\}$

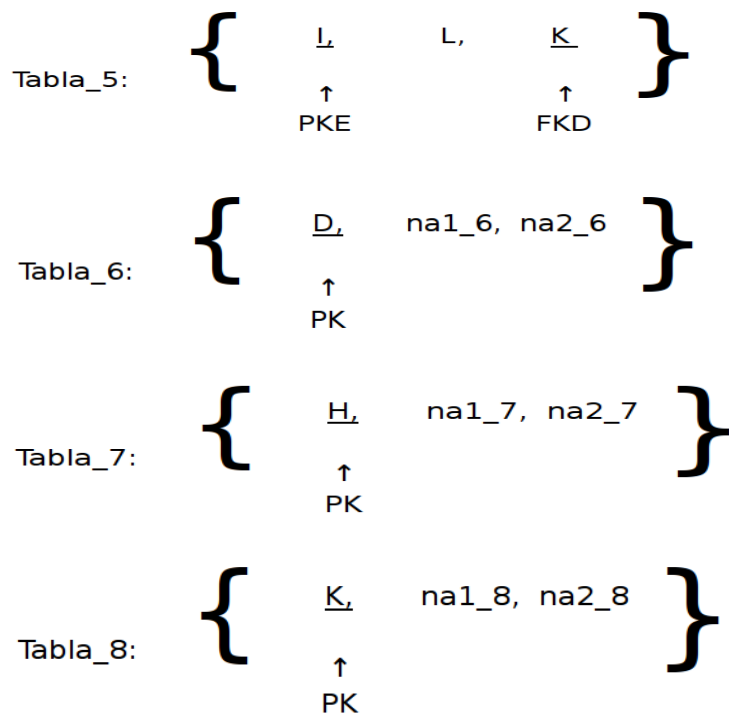


Figura 26. Modelo de Dependencias Funcionales Bifuncional Simple con (DFNE)

Diseño Modelo Entidad Relación

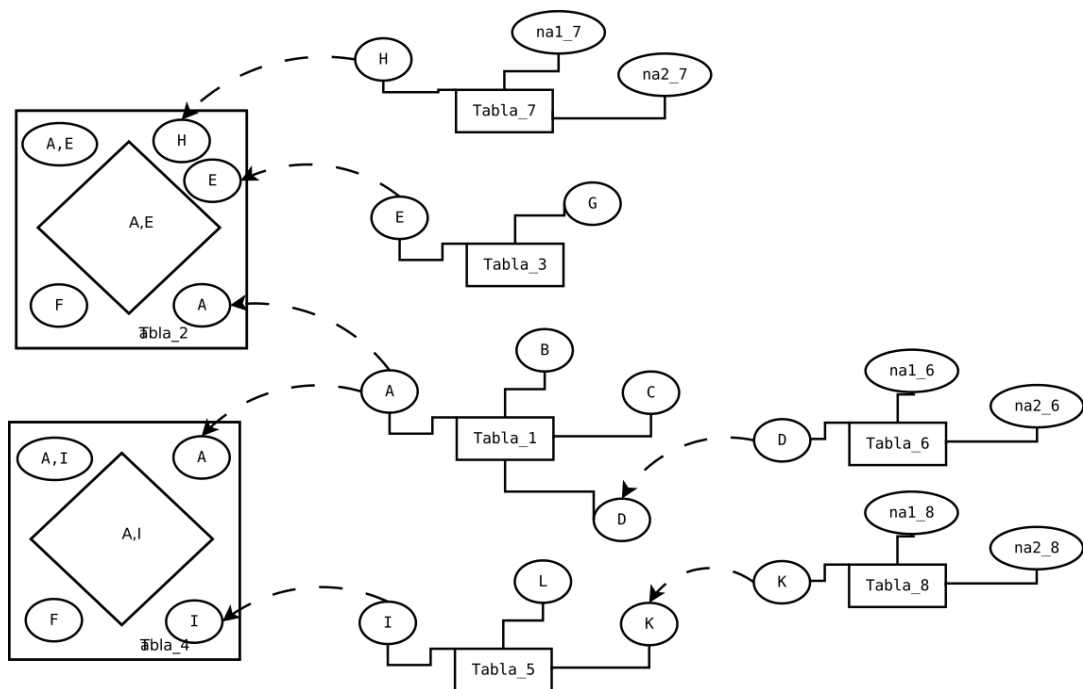


Figura 27. Modelo Entidad Relación Bifuncional Simple con (DFNE)

Diseño Modelo Relacional

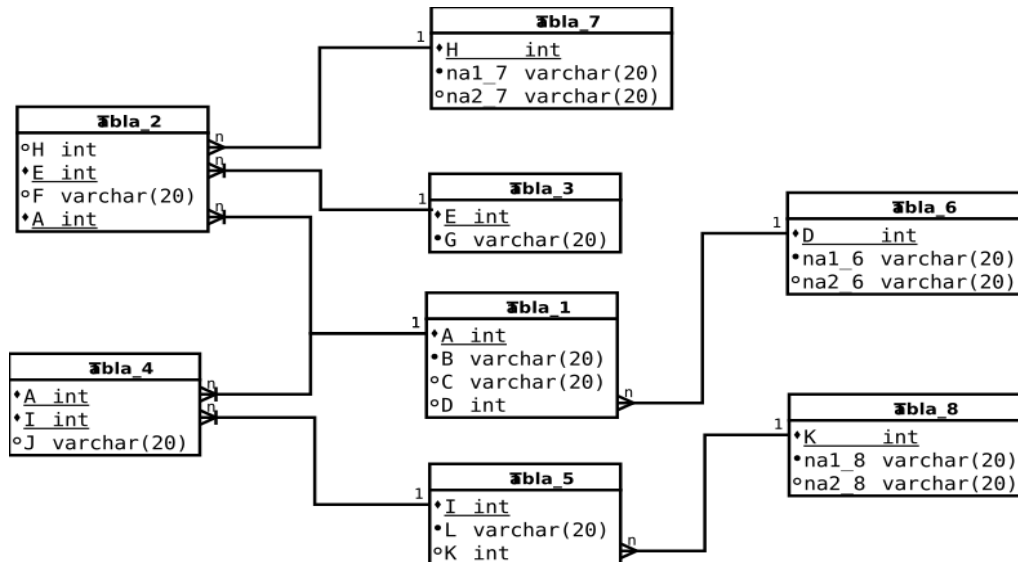


Figura 28. *Modelo Relacional Bifuncional Simple con (DFNE)*

Diseño Modelo DDL

```

CREATE TABLE Tabla_1
(
  A    int,
  B    varchar[20],
  C    varchar[30],
  D    int[15],
  primary key (A),
  foreign key (D) references Tabla_6(D)
);
  
```

```

CREATE TABLE Tabla_2
(
  A    int,
  E    int,
  F    varchar[20],
  H    int,
  primary key (A),
  primary key (E),
  foreign key (H) references Tabla_7(H)
);
  
```



```
CREATE TABLE Tabla_4
(
  A+I    int,
  J      varchar[20],
  primary key (A+I)
);
```

```
CREATE TABLE Tabla_5
(
  I      int,
  L      varchar[20],
  K      int,
  primary key (I),
  foreign key (K) references Tabla_8(K)
);
```

```
CREATE TABLE Tabla_6
(
  D      int,
  na1_6  varchar[20],
  na2_6  varchar[20],
  primary key (D)
);
```

```
CREATE TABLE Tabla_7
(
  H      int,
  na1_7  varchar[20],
  na2_7  varchar[20],
  primary key (H)
);
```

```
CREATE TABLE Tabla_8
(
  K      int,
  na1_8  varchar[20],
  na2_8  varchar[20],
  primary key (K)
);
```

2.7. CLNS Híbrida “Genérico” (Simple y Compuesta)

La lógica simple compuesta podrá aparecer en la parte superior o inferior de la cadena. Podemos expresarlo mediante el siguiente ejemplo:

Cadena lógica del negocio del sistema.

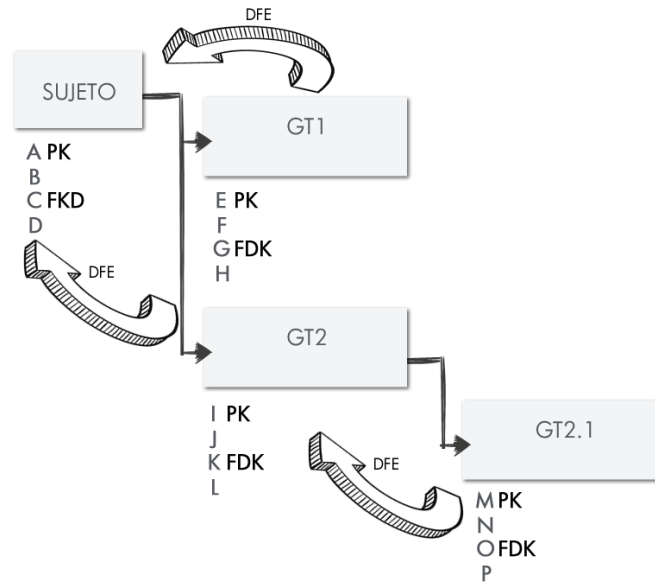


Figura 29. CLNS Híbrida “Genérico” (Simple y Compuesta)

Notemos que esta cadena contiene una bifuncionalidad compuesta entre elementos, y sus relaciones pueden darse con dependencias funcionales exclusivas y no exclusivas; lo que indica que tengo que crear llaves compuestas solo para las relaciones que contengan dependencias no exclusivas DFNE.

Modelo de dependencias funcionales

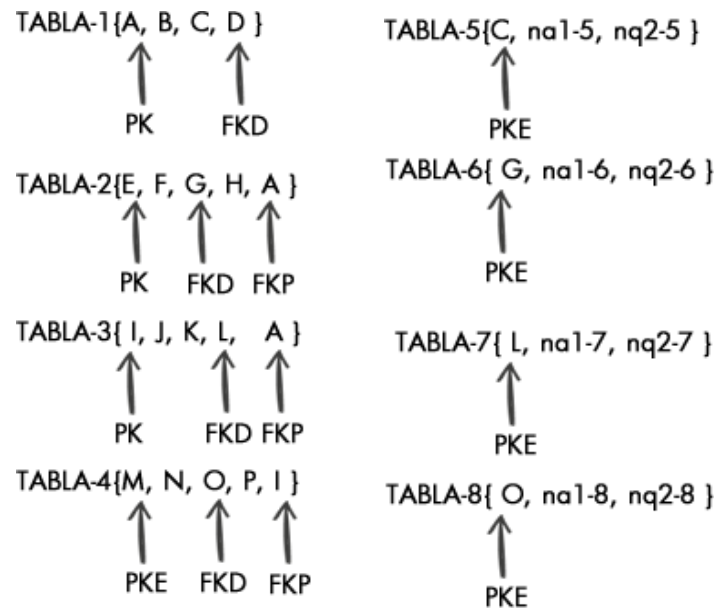


Figura 30. *Modelo de Dependencias Funcionales Híbrida “Genérico” (Simple y Compuesta)*

Modelo Relacional

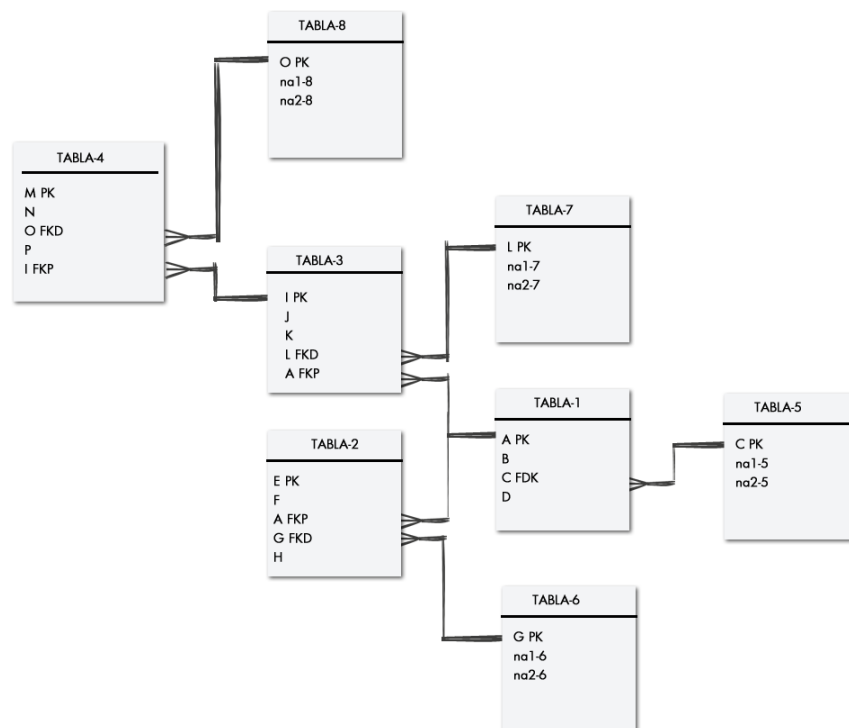


Figura 31. *Modelo Relacional Híbrida “Genérico” (Simple y Compuesta)*

Modelo Entidad Relación

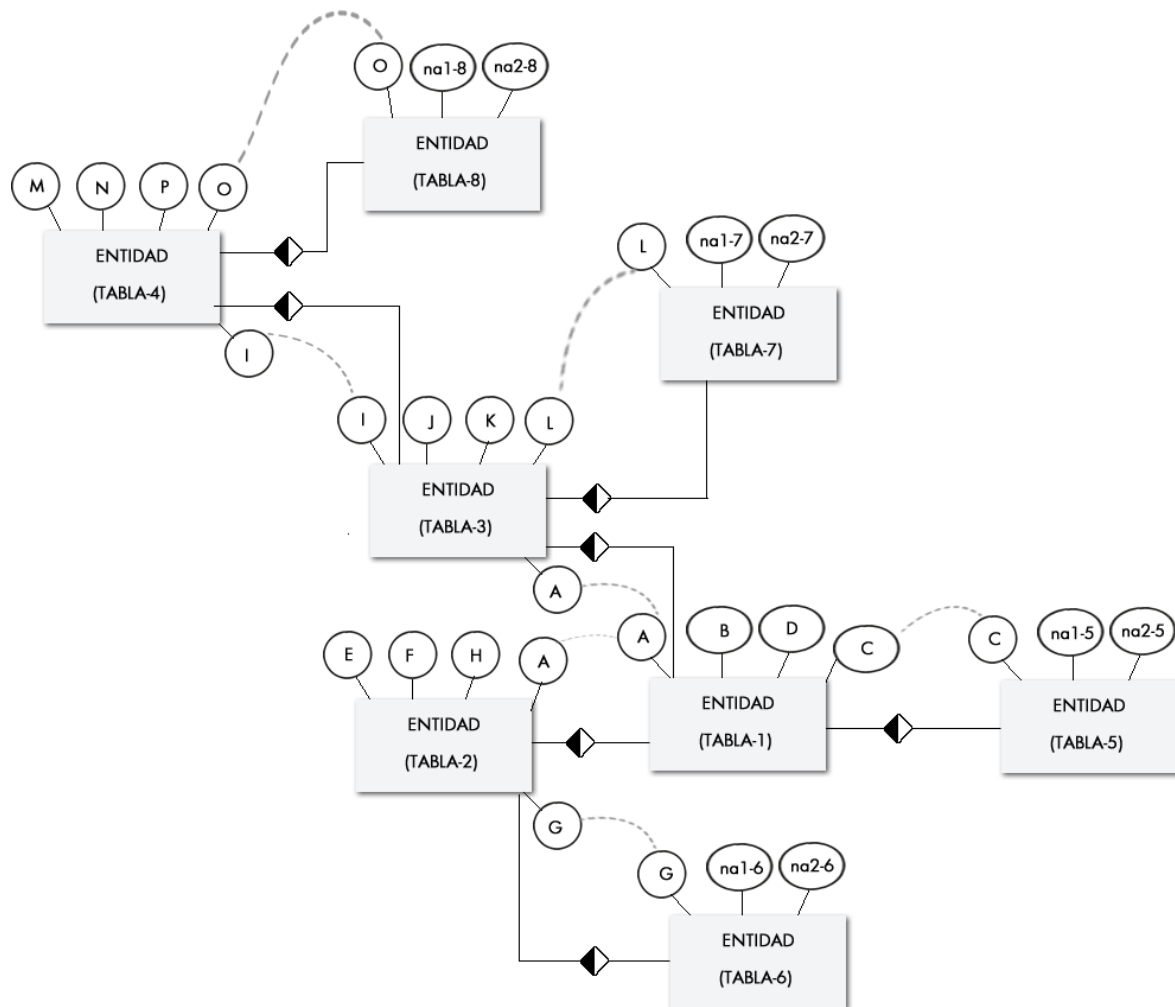


Figura 32. Modelo Entidad Relación Híbrida "Genérico" (Simple y Compuesta)

Modelo DDL

```
create table tabla-5 {
    C int,
    na1-5 varchar[20],
    na2-5 varchar[10],
    primary key(C)
};
create table tabla-6 {
    G int,
    na1-6 varchar[20],
    na2-6 varchar[10],
    primary key(G)
};
create table tabla-1 {
```

```

        A int,
        B int
        C int
        D int
        primary key(A),
        foreing key(C) references tabla-5(C)
    };
create table tabla-7{
    L int,
    na1-6 varchar[20],
    na2-6 varchar[10],
    primary key(L),
};
create table tabla-2{
    E int,
    F int
    G int
    H int
    A int
    primary key(E),
    foreing key(A) references tabla-1(A),
    foreing key(G) references tabla-6(G)
};
create table tabla-3 {
    I int
    J int
    K int
    L int
    A int
    primary key(I),
    foreing key(A) references tabla-1(A),
    foreing key(L) references tabla-7(L)
};
create table tabla-8{
    O int,
    na1-6 varchar[20],
    na2-6 varchar[10],
    primary key(O),
};
create table tabla-4{
    M int
    N int
    O int
    P int
    I int
    primary key(M),
    foreing key(I) references tabla-3(I),
    foreing key(O) references tabla-8(O)
};

```

3. CAPÍTULO III. GESTIÓN DEL PROYECTO

De acuerdo a la metodología ágil XP un proyecto de software debe cumplir con las fases: planificación del proyecto, diseño, codificación y pruebas. Los ítems que se muestran a continuación contienen algunos de los entregables oportunos para cada fase.

3.1. Requerimientos funcionales

La siguiente tabla presenta los requerimientos funcionales del prototipo, con una descripción de la funcionalidad de cada uno; para revisar en detalle cada uno nos debemos dirigir al anexo correspondiente.

Tabla 2 *Requerimientos Funcionales*

REQUERIMIENTOS FUNCIONALES PARA EL PROTOTIPO DEL MODELO SEUDOMATEMÁTICO PARA EL DISEÑO DE LAS BASES DE DATOS RELACIONALES		
Nombre requerimiento	Descripción	ANEXO (anexos.xls)
Registrar usuarios	El usuario debe contar con un módulo en el sistema que le permita obtener acceso a la herramienta	ANEXOS/anexo.xls/RFN001
Gestionar sesiones de usuario	Se debe generar un módulo que permita gestionar los usuarios para el acceso, la salida y el recordatorio de contraseñas	ANEXOS/anexo.xls/RFN002
Gestionar proyectos	Se debe crear un módulo que permita crear, editar y borrar proyectos de bases de datos en la herramienta	ANEXOS/anexo.xls/RFN003
Gestionar Cadena Lógica de Negocio del Sistema	Se debe crear una interfaz que permita la gestión de los elementos propios de la Cadena Lógica de Negocio del Sistema	ANEXOS/anexo.xls/RFN004
Generar Modelo de Dependencias Funcionales	Se debe crear la opción de generar el Modelo de Dependencias Funcionales	ANEXOS/anexo.xls/RFN005
Generar Modelo Relacional	Se debe crear la opción para obtener el Modelo Relacional	ANEXOS/anexo.xls/RFN006
Generar script en SQL	Se debe generar un opción para la descarga de un archivo de texto en formato .sql que contenga el script de creación de las tablas	ANEXOS/anexo.xls/RFN007

3.2. Requerimientos no funcionales

El prototipo de software se realizó de acuerdo a los requerimientos no funcionales que se presentan en la siguiente tabla.

Tabla 3 *Requerimientos No Funcionales*

Nombre requerimiento	Descripción
Interfaz	Se debe contar con una presentación que sea agradable al usuario y fácil de entender
Disponibilidad	Se debe garantizar la disponibilidad de la herramienta al momento de realizar los diseños de bases de datos.
Portabilidad	Se debe garantizar que la ejecución de la herramienta sea independiente de plataforma
Recursos de maquina mínimos	<ul style="list-style-type: none">- Memoria RAM de 512 MB.- Procesador de cualquier gama de 600MHz en adelante.- Conexión a internet de mínimo 1MB de ancho de banda para acceso remoto.- Conexión a red LAN para uso local.- Disco duro de mínimo 10GB.

3.3. Fase1: planificación del proyecto

El desarrollo del prototipo se realizó con base en el cronograma de actividades del proyecto, a continuación se pueden observar las historias de usuario obtenidas a partir de los requerimientos abstraídos de la metodología del modelo seudomatemático, estos requerimientos fueron aprobados por el autor de la misma; en un segundo aspecto se puede observar el Release Planing para dar a conocer las fechas estipuladas para las entregas de las historias de usuario; seguidamente se presentan las iteraciones realizadas para dar cumplimiento a cada una de ellas; por último se muestra con qué velocidad de trabajo se pudo lograr el desarrollo de las iteraciones realizadas.

3.3.1. Historias de usuario

La siguiente tabla representa las historias de usuario establecidas para el desarrollo de prototipo de software del modelo pseudomatemático para el diseño de las bases de datos. Para ver la especificación de cada una ir al anexo correspondiente.

Tabla 4 *Historias de Usuario*

HISTORIAS DE USUARIO PARA EL PROTOTIPO DEL MODELO SEUDOMATEMÁTICO PARA EL DISEÑO DE LAS BASES DE DATOS RELACIONALES		
Nombre historia	Descripción	ANEXO (anexos.xls)
Registrar usuarios - Gestionar sesiones de usuario	El usuario debe contrar con un módulo en el sistema que le permita obtener acceso a la herramienta	ANEXOS/anexo.xls/HU001
Gestionar proyectos	Se debe crear un módulo que permita crear, editar y borrar proyectos de bases de datos en la herramienta	ANEXOS/anexo.xls/HU002
Gestionar Cadena Lógica de Negocio del Sistema	Se debe crear una interfaz que permita la gestión de los elementos propios de la Cadena Lógica de Negocio del Sistema	ANEXOS/anexo.xls/HU003
Generar Modelo de Dependencias Funcionales	Se debe crear la opción de generar el Modelo de Dependencias Funcionales	ANEXOS/anexo.xls/HU004
Generar Modelo Relacional	Se debe crear la opción para obtener el Modelo Relacional	ANEXOS/anexo.xls/HU005
Generar script en SQL	Se debe generar un opción para la descarga de un archivo de texto en formato .sql que contega el script de creación de las tablas	ANEXOS/anexo.xls/HU006

3.3.2. Release planning

Tabla 5 *Release Planning*

[illegible]

[illegible]

Según el cuadro del Iteration planning el proyecto tuvo un desfase de una semana con respecto a la planeación.

3.3.4. Velocidad del proyecto

El siguiente cuadro presenta una relación de horas de trabajo con respecto a las iteraciones realizadas para el cumplimiento de las historias de usuario.

Tabla 7 *Velocidad de Proyecto*

VELOCIDAD DEL PROYECTO				
	Iteración 1	Iteración 2	Iteración 3	Iteración 4
Horas	25	35	33	27
Semanas	3	3	3	3
Horas semanales	32	40	42	38
Historias de usuario (Velocidad del proyecto)	2	1	2	2

3.4. Fase2: diseño

3.4.1. Tarjetas C.R.C.

La siguiente tabla presenta las diferentes clases que se plantearon para el prototipo del modeloseudomatemático para el diseño de las bases de datos, estas clases se categorizan de acuerdo a la responsabilidad que realiza cada una en el prototipo y en que parte del mismo colaboran para llevar a cabo una determinada función.

Tabla 8 *Tarjetas C.R.C.*

Nombre de la clase: User	
Responsabilidad	Colaboración
controlar el ingreso de usuarios al sistema	
validar el correo y contraseña de un usuario	
registrar un usuario en el sistema	
Nombre de la clase: Project	
Responsabilidad	Colaboración
mostrar los proyectos creados por un usuario	User

permitir crear un proyecto a un usuario	
abrir proyectos existentes de un usuario	
Nombre de la clase: Clns	
Responsabilidad	Colaboración
muestra entidades de un proyecto	project
actualiza entidades de un proyecto	
crea entidades para un proyecto	
Nombre de la clase: Attribute	
Responsabilidad	Colaboración
crea atributos a una cadena	clns
edita atributos de una cadena	
elimina atributos de una cadena	
Nombre de la clase: AuthController	
Responsabilidad	Colaboración
verifica las credenciales de un usuario	user
Nombre de la clase: ProjectController	
Responsabilidad	Colaboración
envía información al modelo para crear, editar y eliminar proyectos	user project
Nombre de la clase: ClnsController	
Responsabilidad	Colaboración
comunica la vista de cadenas con el modelo para crear, editar y eliminar entidades	project clns
verifica el estado de una entidad	
Nombre de la clase: AttributeController	
Responsabilidad	Colaboración
comunica la vista de atributos con el modelo de creación, eliminación y actualización de un atributo de una entidad	clns clnscontroller atribute
Nombre de la clase: routes	
Reponsabilidad	Colaboración
enviar y recibir peticiones de las vistas para ejecutar una acción en un controlador	projectController clnsController userController atributeController scriptController defpfunController
Nombre de la clase: userController	
Responsabilidad	Colaboración

controlar el ingreso de usuarios al sistema	User
validar el correo y contraseña de un usuario	
registrar un usuario en el sistema	
alertar a la vista de un error	

3.5. Fase3: codificación

La estructura de un proyecto basado en el Framework Laravel contienen los siguientes directorios. Para este proyecto se hizo uso de la versión 5.2

- app/
- bootstrap/
- config/
- database/
- public/
- resources/
- storage/
- tests/
- vendor/
- .env
- .env.example
- .gitattributes
- .gitignore
- artisan
- composer.json
- composer.lock
- gulpfile.js
- package.json
- phpspec.yml
- phpunit.xml
- readme.md
- server.php

La carpeta “app”

Esta carpeta contiene los modelos usados en el proyecto es usada para almacenar los modelos de nuestro proyecto, es decir en donde se ejecutan las operaciones principales del negocio; esta carpeta contiene otros directorios de gran importancia como lo son el Http en el cual están los controladores, Middlewares; que filtran nuestras peticiones y los request. Existen también un archivo denominado routes.php que es el que maneja las rutas por las cuales se hacen las peticiones GET, POST, PUT Y DELETE.

La siguiente imagen ilustra una vista general del proyecto.

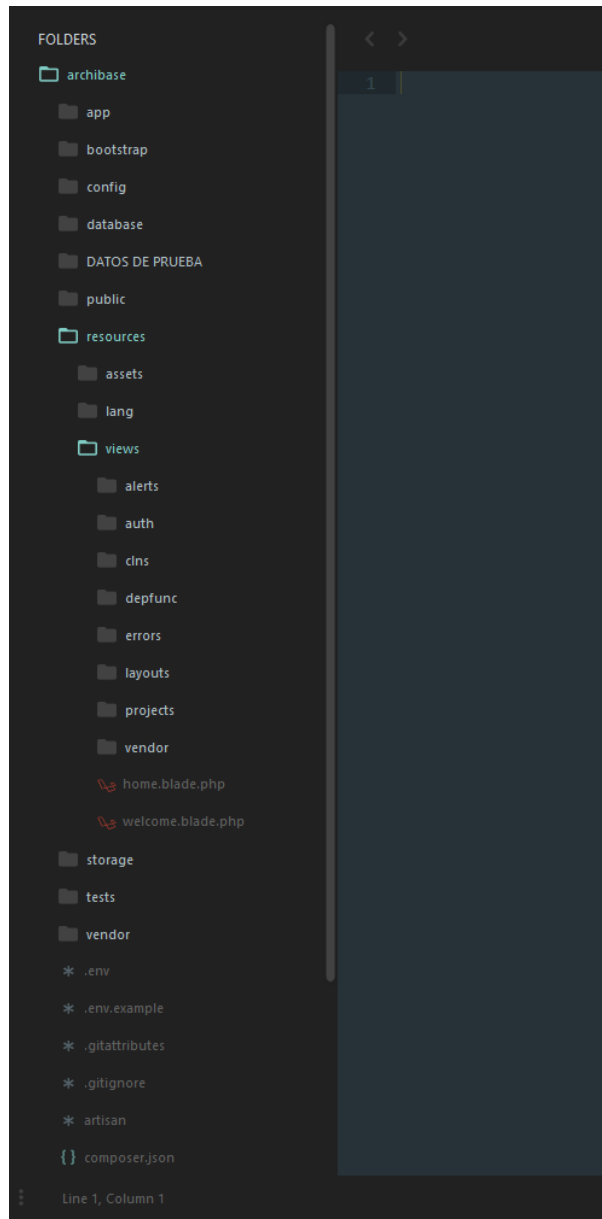


Figura 33. *Directorios del proyecto Archibase*

La carpeta “config”

Como el nombre del directorio menciona, esta carpeta contiene la configuración general de nuestra aplicación, el archivo más importante de este directorio es el llamado `app.php` el cual lista las clases principales del framework, establece entornos de prueba y hace un llamado al archivo `database.php` de conexión de la bases de datos.

La carpeta “database”

Los archivos más importantes de esta carpeta son los llamados migrations, ya que contienen los atributos y tablas que se conectarán al ORM de Laravel para crear una tabla por cada clase declarada. En la siguiente figura podemos observar las migraciones creadas para el proyecto Archibase

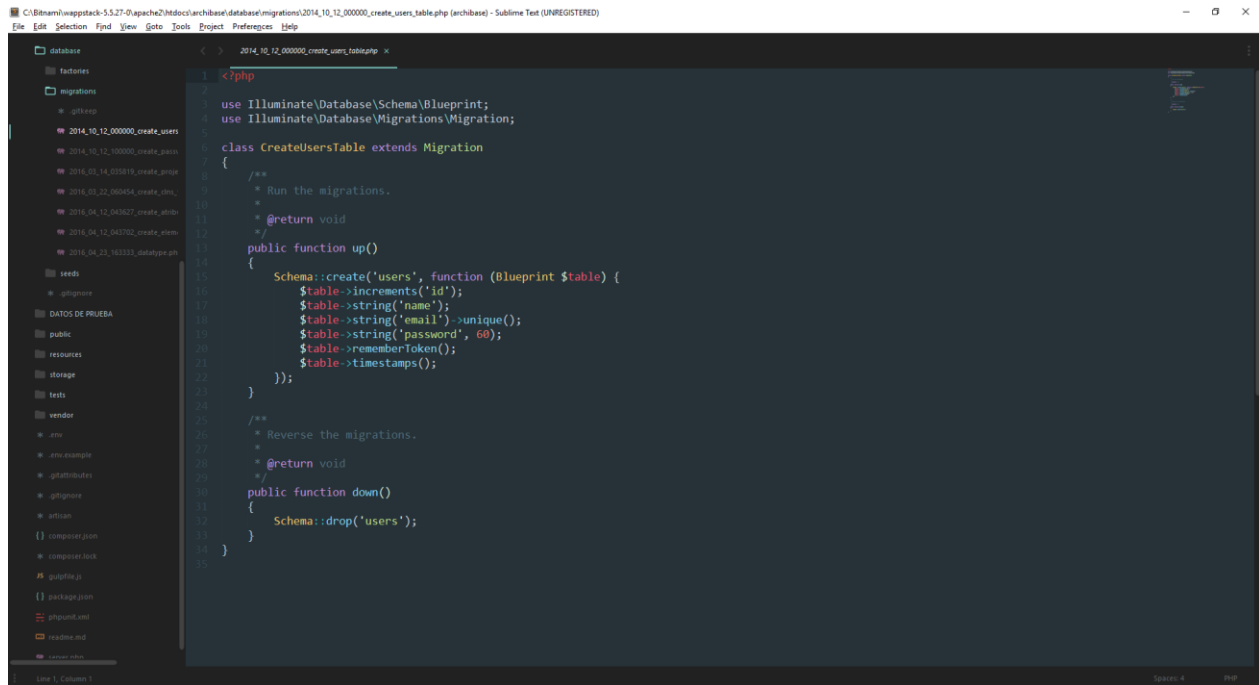


Figura 34. Directorio database del proyecto archibase

La carpeta “public”

Dentro de esta carpeta se muestran los recursos de la aplicación, es decir los archivos de estilos .css, .js, imágenes y tipografías utilizadas en la aplicación. directorio colocaremos todos los recursos estáticos de nuestra aplicación, es decir, archivos css, js, imágenes y fuentes. Una característica de laravel es que ya incluye el conjunto de herramientas de Bootstrap listas para usar. La siguiente imagen ilustra el directorio public.

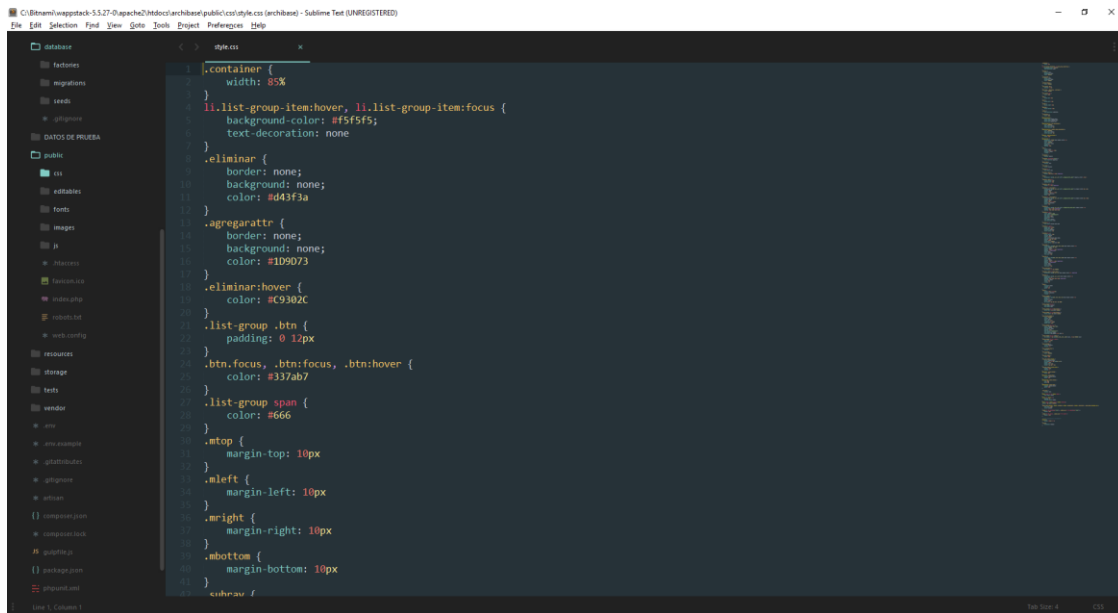


Figura 35. Carpeta public del framework laravel para el proyecto archibase

La carpeta “resources”

Dentro de este directorio lo más significativo es la carpeta *views*, que contiene toda la parte gráfica o de interfaz de nuestra aplicación, cada formulario, botón o ventana emergente se incluye dentro de estas carpetas. Algo característico de laravel es el uso de plantillas que se puede reutilizar en todo el software, estas plantillas son programadas en la carpeta *layouts*. A continuación se presenta una imagen con las diferentes vistas del proyecto archivase.

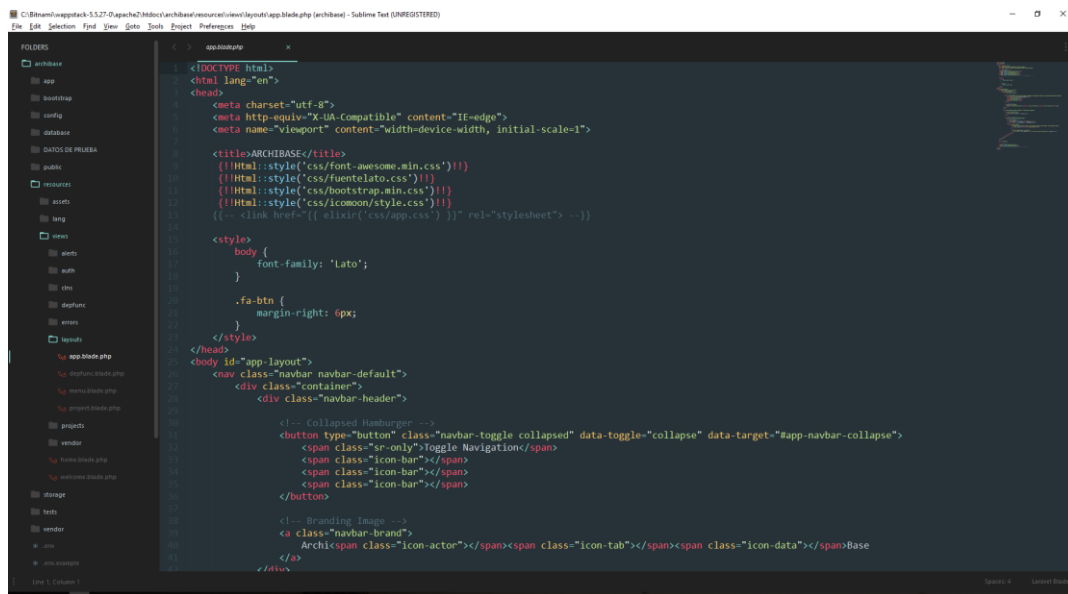


Figura 36. Carpeta resources del proyecto archibase utilizando Laravel

3.6. Fase4: pruebas

El siguiente caso de prueba se realizó de acuerdo a una de las historias de usuario principales realizada para dar cumplimiento a los requerimientos de la metodología de modelo pseudomatemático para el diseño de las bases de datos relacionales. Para ver todos los casos de prueba consultar los anexos en las pestañas con nombre “CP”

Tabla 9 Caso de prueba Cadena Lógica del Negocio

CASO DE PRUEBA		
Número de caso de prueba:	1	Número de historia de usuario: 3
Nombre del caso de prueba:	Gestionar cadena lógica del negocio del sistema	
Descripción:	Permitir crear una cadena lógica del negocio a partir de la creación de sujetos, grupos transaccionales y la dependencia funcional entre los mismos.	
Condiciones de ejecución:	El usuario debe estar logeado en el sistema y haber creado un proyecto en el módulo de creación de proyectos	
Entradas:	<p>El actor pulsa el botón de "+" en la sección para quién controlo?, para crear una entidad de tipo sujeto en una ventana emergente.</p> <p>El actor pulsa el botón de "+" en la sección qué quiero controlar?, para crear una entidad de tipo grupo transaccional en una ventana emergente.</p> <p>El actor pulsa un radiobutton para seleccionar si el grupo transaccional es exclusivo del sujeto, esto para seleccionar la dependencia funcional existente.</p> <p>El actor pulsa el botón "crear" para formar una cadena de negocio con el grupo y el sujeto creados</p>	
Resultado esperado:	<p>Las entidades sujeto y grupo transaccional son almacenadas en la base de datos con la dependencia existente.</p> <p>El sistema debe mostrar un grafo en el que se muestre cada entidad creada y su unión por medio de una fecha descendiente, apuntando de izquierda a derecha, siendo la entidad izquierda el sujeto creado y derecha el grupo transaccional.</p> <p>Cada grupo transaccional debe representar el tipo de dependencia existente con respecto a su sujeto (dfe, dfne).</p> <p>El sistema debe mostrar por cada entidad un botón para poder agregar atributos a la misma. Con el fin de</p>	

	crear las tablas auxiliares correspondientes a los atributos que sean de tipo FKD
Evaluación:	El sistema cumple con los resultados esperados, ya que representa la cadena lógica del negocio al crear un sujeto y grupo transaccional, sin embargo existen inconvenientes al momento de querer cambiar los grupos transaccionales de sujeto.

3.6.1. Pruebas de rendimiento con WAPT

Las pruebas de rendimiento del prototipo del software Archibase se han llevado a cabo mediante el uso de la herramienta WAPT 9.0 el cual permite realizar test de aplicaciones web y de escritorio. Esta aplicación permite simular usuarios virtuales apuntando a las URL's o páginas que se vayan navegando a medida que se usa el sistema. Dichos usuarios simulan concurrencia y escalabilidad.

Para el prototipo Archibase, la aplicación WAPT se utilizó con el fin de responder a los siguientes supuestos:

- Cantidad de usuarios simultáneos que puede manejar la aplicación web.
- Tiempo de respuesta de la aplicación web.

Esta prueba se realizó con una carga inicial de 5 usuarios con un límite de 250 usuarios, en un tiempo de ejecución de 10 segundos. Una vez ejecutada esta prueba el software WAPT arroja los resultados mostrados en la siguiente imagen:

Test result: success

Pass/Fail Criteria

Name

Session error rate for each profile

Result

success

Comment

Summary

Profile	Successful sessions	Failed sessions	Successful pages	Failed pages	Successful hits	Failed hits	Other errors	Total Kbytes sent	Total Kbytes received	Avg response time, sec (with page elements)
JHCN	2990	251	10	0	0	success	0	1276	1270	3.45(2.01)

Number of active users

Profile	0:00:00 - 0:01:00	0:01:00 - 0:02:00	0:02:00 - 0:03:00	0:03:00 - 0:04:00	0:04:00 - 0:05:00	0:05:00 - 0:06:00	0:06:00 - 0:07:00	0:07:00 - 0:08:00	0:08:00 - 0:09:00	0:09:00 - 0:10:00
JHCN	5	5	5	5	5	5	5	5	5	5
Total	5	5	5	5	5	5	5	5	5	5

Successful sessions (Failed sessions)

Profile	0:00:00 - 0:01:00	0:01:00 - 0:02:00	0:02:00 - 0:03:00	0:03:00 - 0:04:00	0:04:00 - 0:05:00	0:05:00 - 0:06:00	0:06:00 - 0:07:00	0:07:00 - 0:08:00	0:08:00 - 0:09:00	0:09:00 - 0:10:00	Total
JHCN	0(300)	0(300)	0(300)	0(295)	0(300)	0(300)	0(297)	0(296)	0(300)	0(300)	0(2990)
Total	0(300)	0(300)	0(300)	0(295)	0(300)	0(300)	0(297)	0(296)	0(300)	0(300)	0(2990)

Successful pages (Failed pages)

Profile	0:00:00 - 0:01:00	0:01:00 - 0:02:00	0:02:00 - 0:03:00	0:03:00 - 0:04:00	0:04:00 - 0:05:00	0:05:00 - 0:06:00	0:06:00 - 0:07:00	0:07:00 - 0:08:00	0:08:00 - 0:09:00	0:09:00 - 0:10:00	Total
JHCN	0(300)	0(300)	0(300)	0(295)	0(300)	0(300)	0(297)	0(296)	0(300)	0(300)	0(2990)
Total	0(300)	0(300)	0(300)	0(295)	0(300)	0(300)	0(297)	0(296)	0(300)	0(300)	0(2990)

Successful hits (Failed hits)

Profile	0:00:00 - 0:01:00	0:01:00 - 0:02:00	0:02:00 - 0:03:00	0:03:00 - 0:04:00	0:04:00 - 0:05:00	0:05:00 - 0:06:00	0:06:00 - 0:07:00	0:07:00 - 0:08:00	0:08:00 - 0:09:00	0:09:00 - 0:10:00	Total
JHCN	0(300)	0(300)	0(300)	0(295)	0(300)	0(300)	0(297)	0(296)	0(300)	0(300)	0(2990)
Total	0(300)	0(300)	0(300)	0(295)	0(300)	0(300)	0(297)	0(296)	0(300)	0(300)	0(2990)

Figura 37. Resultados de prueba de carga al prototipo Archibase

Según la imagen el sistema soporta hasta 2990 sesiones en 10 páginas consultadas con un tiempo de respuesta promedio de tres segundos por consulta realizada. Esta prueba fue ejecutada con una velocidad de internet de 1276Kbytes como punto de partida.

4. Capítulo IV análisis y diseño del prototipo

4.1. Casos de uso

El siguiente caso de uso representa de manera general el funcionamiento del prototipo de software.

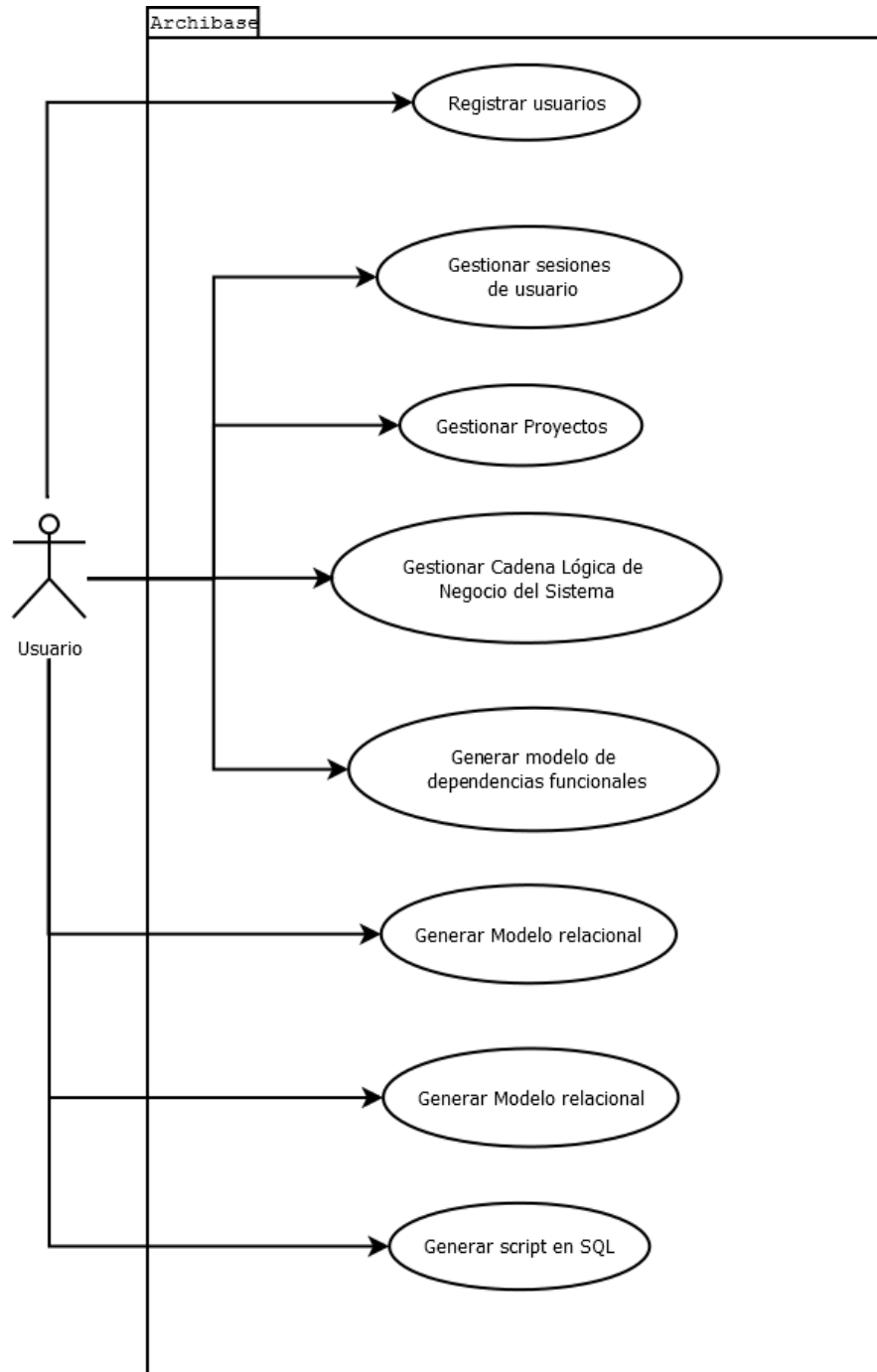


Figura 38. Caso de uso general

4.1.1. Especificación de casos de uso prototipo CASE

La especificación de cada uno de los casos de uso que representan los requerimientos del sistema se nombran en la siguiente tabla, para observar con detalle a cada uno ver los anexos en las pestañas con nombre “CU”.

Tabla 10 *Especificación de casos de uso*

Especificación de Casos de Uso			
Nombre del caso de uso		Descripción	Anexo
1	Iniciar sesión	Describe la forma en que un usuario se registra e ingresa al sistema Se requiere el desarrollo de una interfaz que permita al usuario registrarse para poder usar el software y una interfaz para ingresar al software luego de haberse registrado	ANEXOS/ANEXOS.xls/CU001
2	Administrar proyectos	Se muestra la forma en que deben crear, editar y borrar proyectos de bases de datos en la herramienta	ANEXOS/ANEXOS.xls/CU002
3	Administrar Cadenas Lógicas de Negocio	Presenta la forma en que se crean, editan y eliminan proyectos de bases de datos	ANEXOS/ANEXOS.xls/CU003
4	Generar Modelo de Dependencias Funcionales	Se presenta la manera en que se genera el Modelo de Dependencias Funcionales	ANEXOS/ANEXOS.xls/CU004
5	Generar Modelo Relacional	Se presenta la manera en que se genera el Modelo Relacional	ANEXOS/ANEXOS.xls/CU005
6	Generar Script en SQL	Se presenta la manera en que se genera el Script SQL	ANEXOS/ANEXOS.xls/CU006

4.2. Diagrama de clases

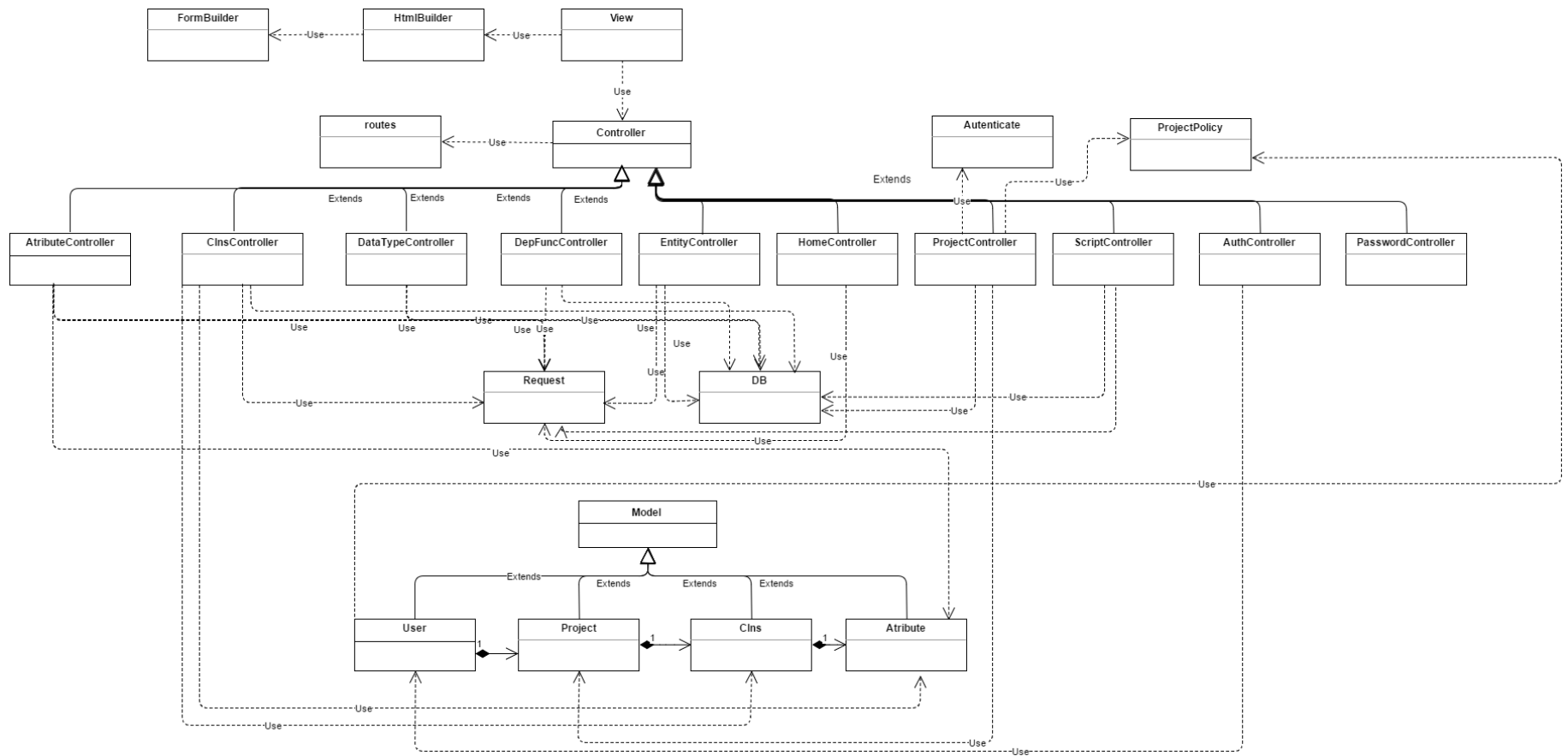


Figura 39. *Diagrama de clases*

4.3. Diagramas de estado

La especificación de cada uno de los diagramas de estado representan los distintos cambios que puede tener un objeto con respecto a los requerimientos del sistema. A continuación se puede observar el diagrama de estados principal “administrar cadenas”, para observar con detalle a cada uno ver los anexos en las pestañas con nombre “DIAGEST”.

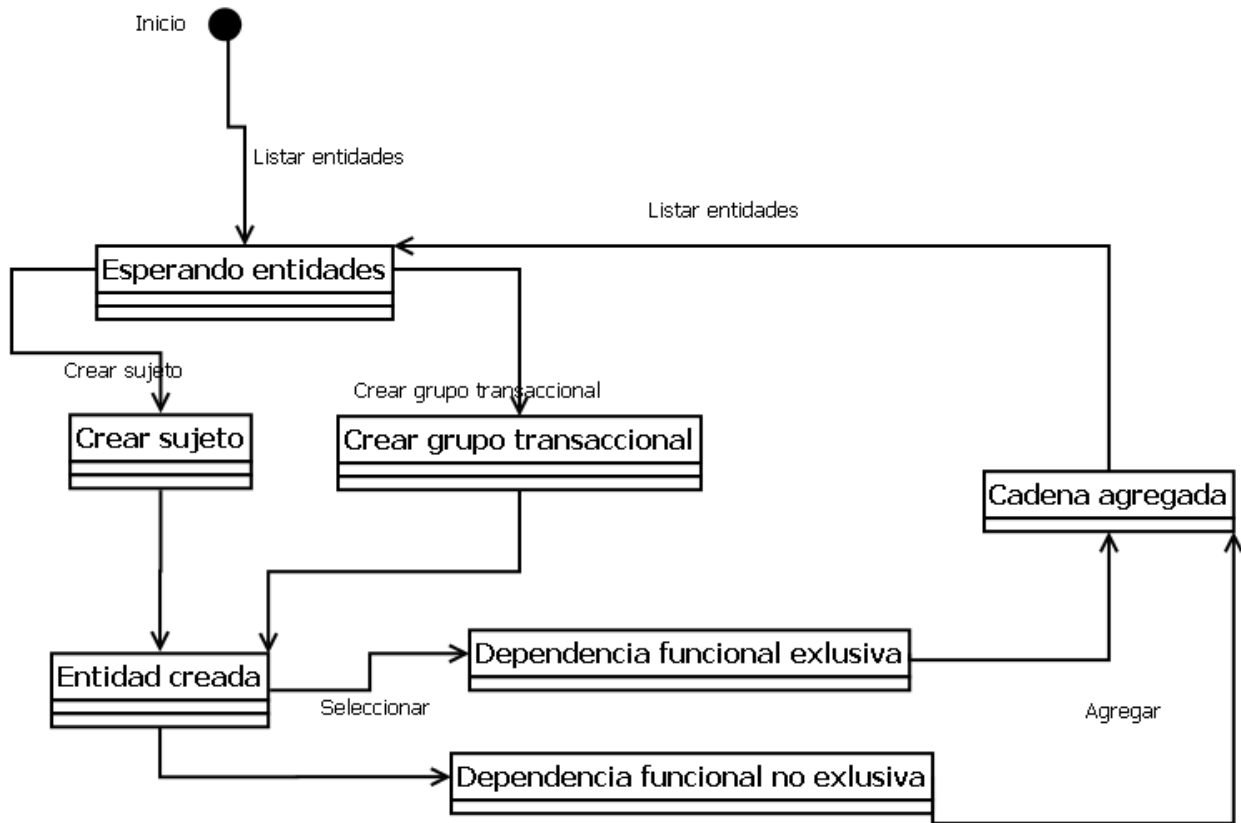


Figura 40. *Diagrama de estados administrar cadenas*

Tabla 11 *Diagramas de estado*

Diagramas de estados			
Nombre del diagrama		Descripción	Anexo
1	Iniciar sesión	Estados del objeto User al iniciar sesión en el sistema	ANEXOS/ANEXOS.xls/DIAGE ST001
2	Administrar proyectos	Estados del objeto Project cuando un usuario desea crear, editar o eliminar un proyecto en el sistema	ANEXOS/ANEXOS.xls/DIAGE ST002
3	Administrar Cadenas Lógicas de Negocio	Estados del objeto Clns cuando un usuario desea crear, editar o eliminar una cadena lógica del negocio en el sistema	ANEXOS/ANEXOS.xls/DIAGE ST003
4	Administrar atributos	Estados del objeto Attribute cuando un usuario desea crear, editar o eliminar un atributo de una entidad en el sistema	ANEXOS/ANEXOS.xls/DIAGE ST004

4.4. Diagramas de secuencia

El siguiente diagrama muestra la secuencia de objetos del módulo de cadena lógica del negocio, una de las funcionalidades más representativas del prototipo CASE.

4.4.1. Diagrama de secuencia Administrar cadenas

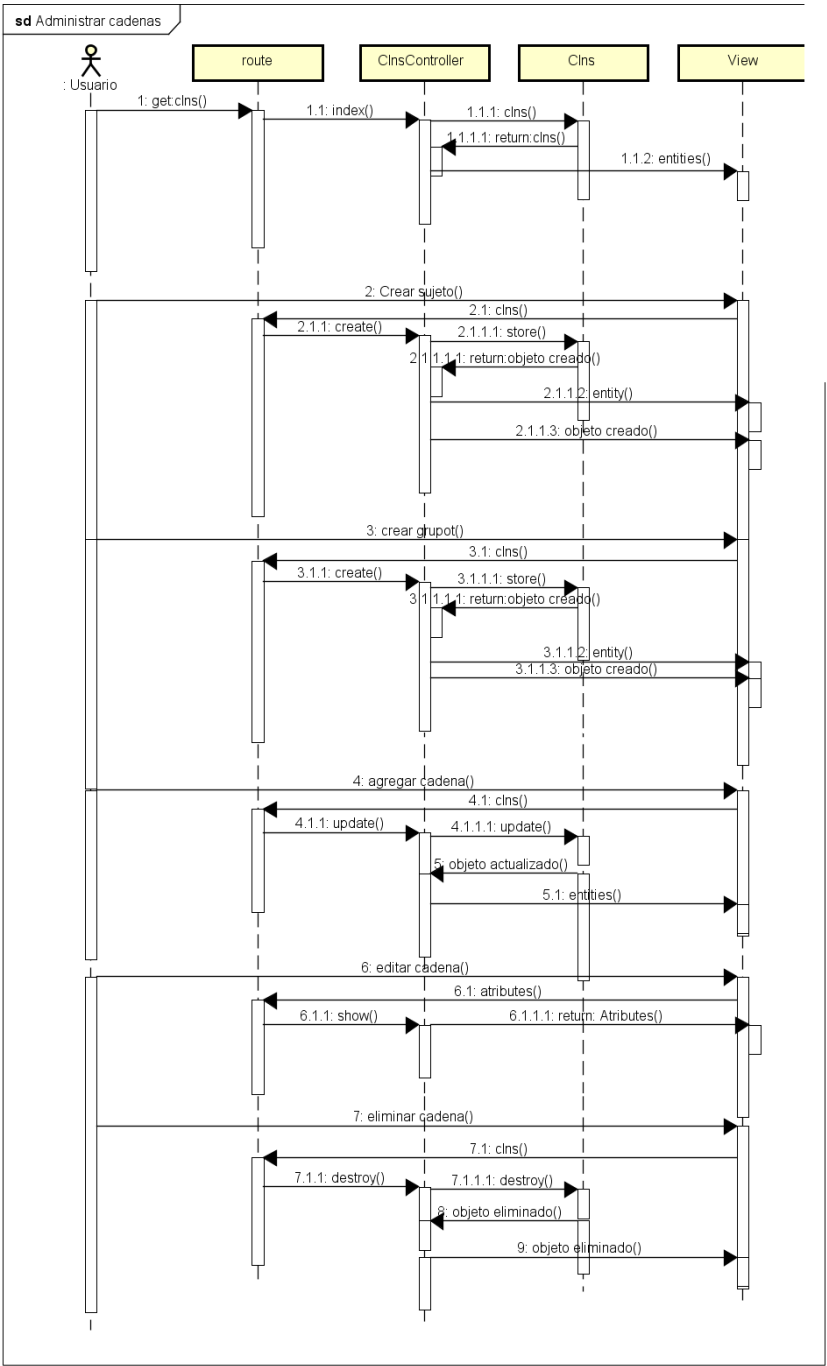


Figura 41. *Diagrama de secuencia administrar cadenas*

La siguiente tabla presenta los demás diagramas realizados para el diseño del prototipo, para ver la especificación de cada uno puede ir a los anexos en las hojas nombradas “DIAGSEC”.

Tabla 12 *Lista de diagramas de secuencia*

Diagramas de secuencia		
Nombre del diagrama		Descripción
		Anexo
1	Iniciar sesión	Describe la secuencia de objetos para iniciar sesión en el sistema
2	Administrar proyectos	Describe la secuencia de objetos para crear, editar y eliminar proyectos en el sistema
3	Administrar Cadenas Lógicas de Negocio	Describe la secuencia de objetos para crear, editar y eliminar cadenas lógicas en el sistema
4	Generar Modelo de Dependencias Funcionales	Describe la secuencia de objetos para presentar el modelo de dependencias funcionales
5	Generar script SQL	Describe la secuencia de objetos para descargar el script SQL de la cadena lógica del negocio.

4.5. Diagrama de componentes

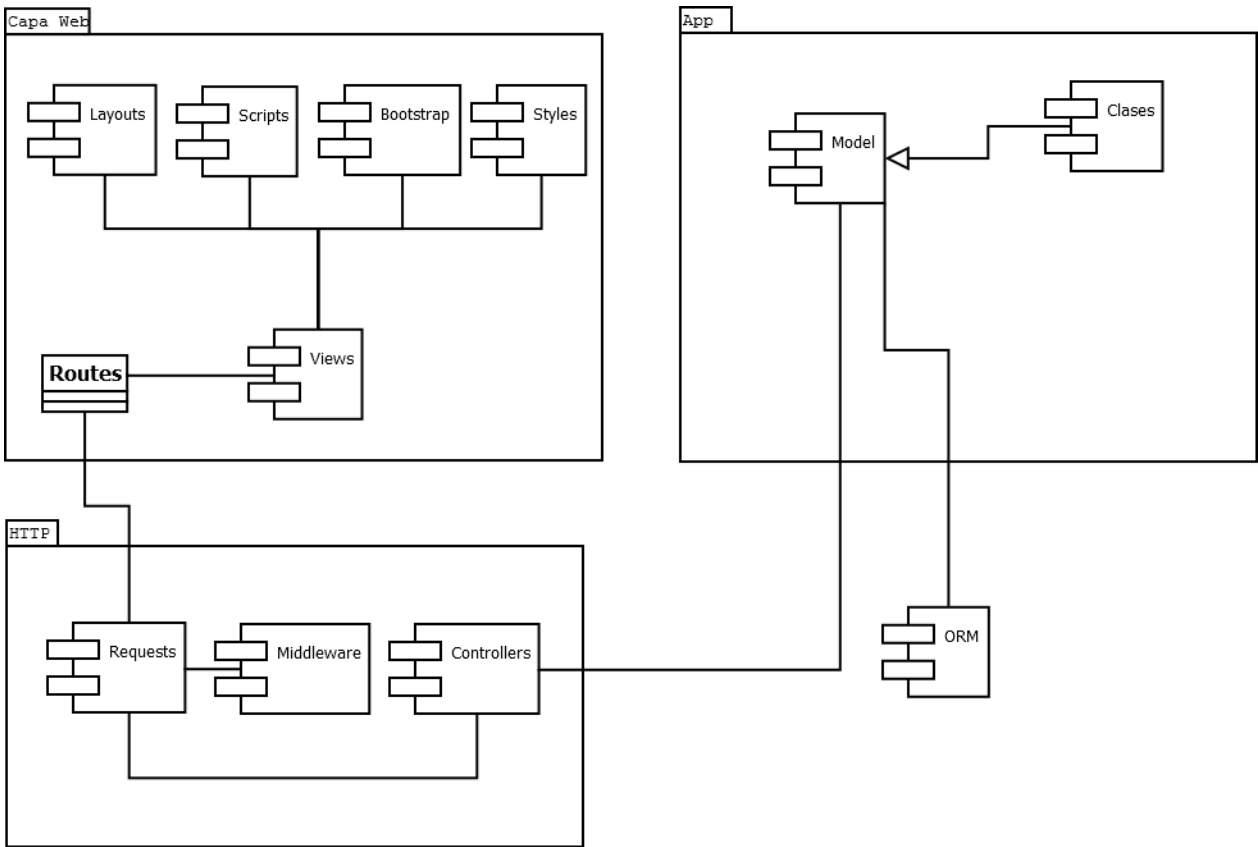


Figura 42. Diagrama de Componentes

4.6. Diagrama de despliegue

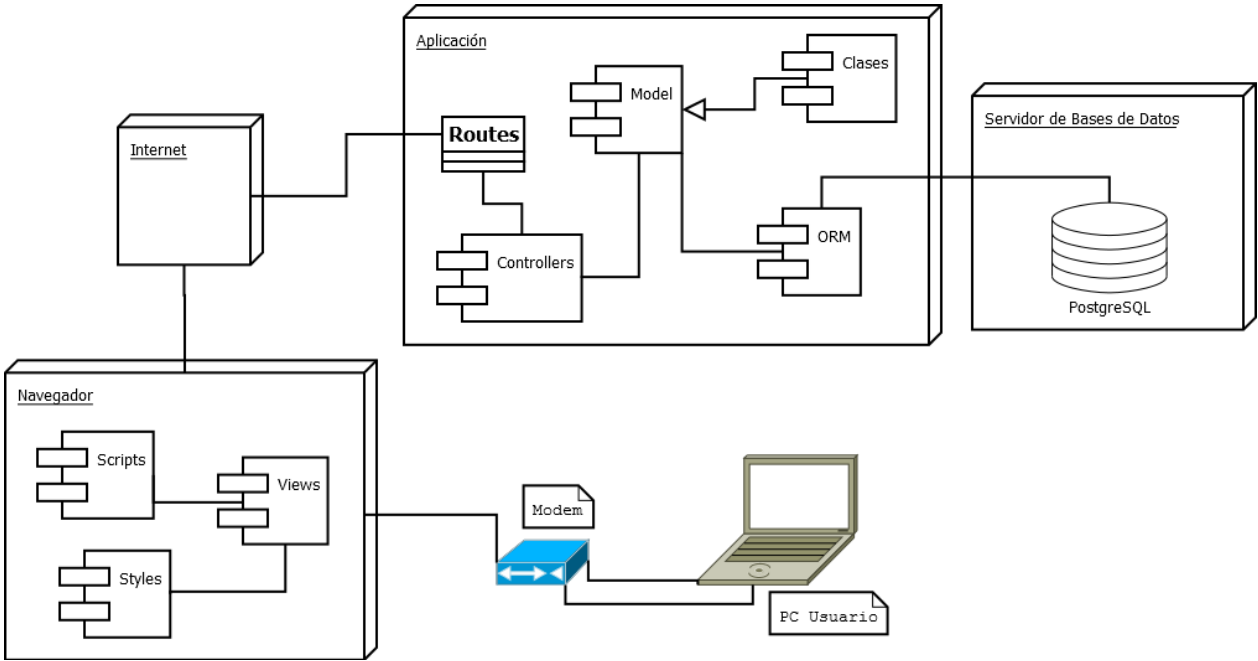


Figura 43. Diagrama de Despliegue

PARTE III CIERRE DE LA INVESTIGACIÓN

5. Capítulo V RESULTADOS Y DISCUSIÓN

En el artículo resultado del proyecto de investigación, se presenta un panorama general del modelo seudomatemático propuesto por el ingeniero Londoño, mediante el cual se ilustra al lector sobre la metodología y sus componentes. También se lleva a cabo una aplicación de la metodología mediante un ejemplo sencillo, con el cual, se busca comprobar que el proceso de normalización de bases de datos se cumple a cabalidad. Al final, se ilustra el desarrollo de un ejemplo usando el prototipo de software como la descripción de un prototipo de software que aplica dicha metodología con la esperanza última de obtener un archivo en formato SQL que contenga un script que pueda ser llevado a cualquier motor de bases de datos y ejecutar la creación de las tablas diseñadas desde el prototipo.

6. Capítulo VI CONCLUSIONES

6.1.1. Verificación

Para llevar a cabo el proceso de verificación del prototipo diseñado para el modelo seudomatemático para el diseño de las bases de datos relacionales se ha llevado a cabo el siguiente caso de estudio, el cual es una adaptación del documento publicado por la Facultad de Administración y Dirección de Empresas de la Universidad Politécnica de Valencia bajo la licencia Creative Commons Reconocimiento 3.0 (CC BY), disponible en http://users.dsic.upv.es/asignaturas/fade/inf/es/solucion_BD.pdf.

Descripción general

La empresa INCA_AUTOS se dedica al alquiler de vehículos y busca controlar el estado de sus autos. La empresa cuenta con sucursales en diferentes ciudades las cuales se identifican con un número único, además se requiere consultar su información de ubicación: dirección y teléfono.

Cada sucursal cuenta con un conjunto de vehículos que identifica por su placa, agrupa por marca, tipo de vehículo, modelo, capacidad de pasajeros y tipo de combustible que utiliza.

El control de cada alquiler se lleva en una planilla en la que se consigna la información del cliente: cédula, dirección, teléfono, celular, número de tarjeta de crédito. También se relacionan la fecha, el valor del alquiler y el número de días que dura el alquiler.

Las imágenes siguiente representan la resolución del problema anterior mediante el uso del prototipo, en este se asume que ya existe un usuario y un proyecto creado.

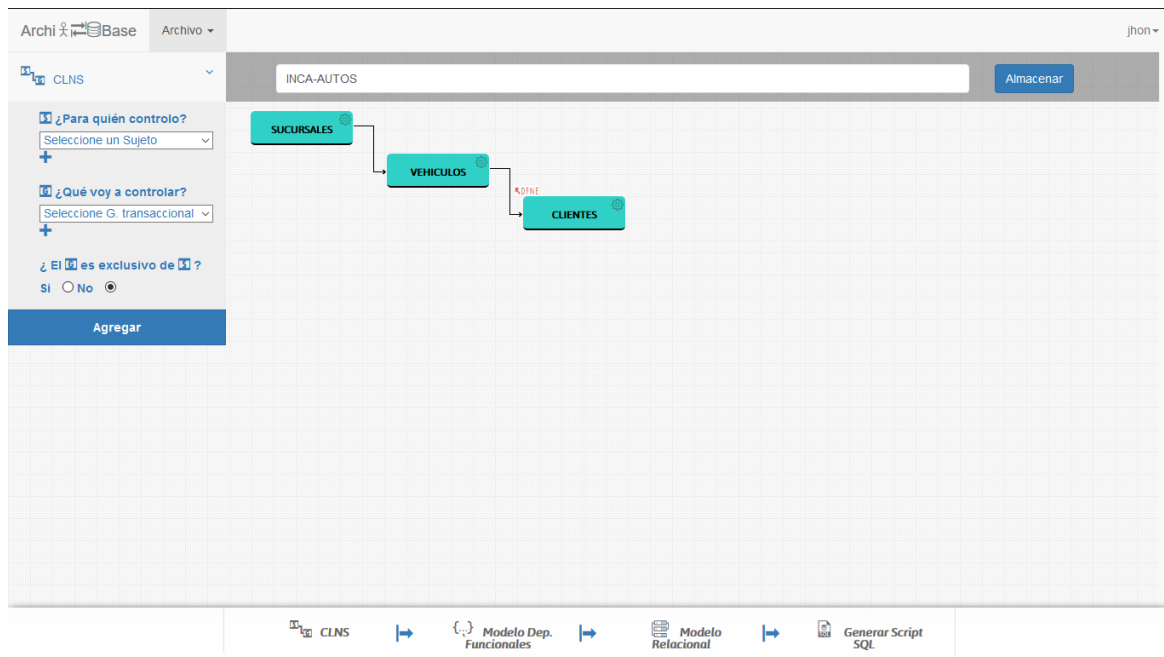


Figura 44. CLNS usando el prototipo CASE

En anterior figura se representan las entidades principales, resolviendo primero las preguntas del negocio: ¿Qué se quiere controlar? y ¿Para quién se quiere controlar?

Una vez se tienen resueltas las principales entidades del negocio, se agregan los atributos correspondientes a cada entidad.

Columna	Tipo dato	Tamaño	Precisión	Tipo índice	Agregar	Editar	Eliminar
ID_SUCURSALES	integer	null	null	PK			
codciudad	integer	0	0	FKD			
direccion	character varying	100	0	--			
telefono	integer	0	0	--			

Figura 45. Adición Atributos a la Entidad SUCURSALES usando el prototipo CASE

El sistema detecta cuando se quiere agregar una llave foránea e inmediatamente sugiere crear una tabla auxiliar.

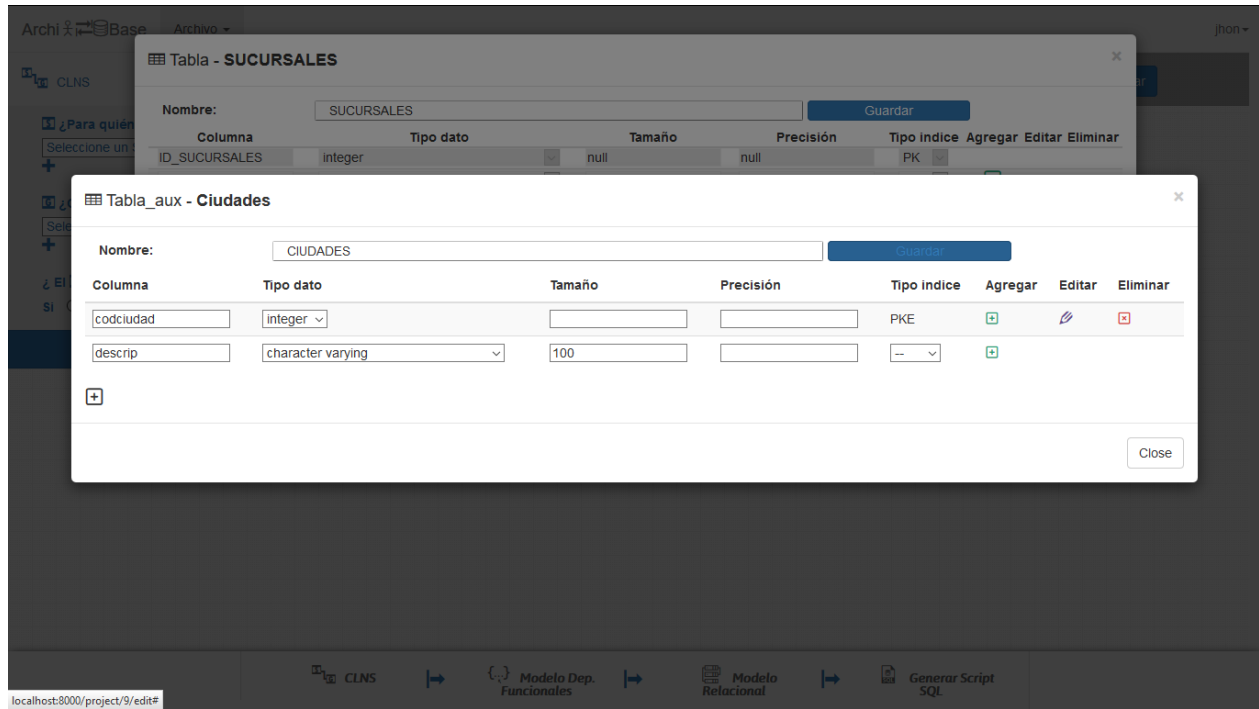


Figura 46. Creación de Tabla Auxiliar CIUDADES usando el prototipo CASE

Este proceso debe repetirse para cada una de las entidades principales del negocio.

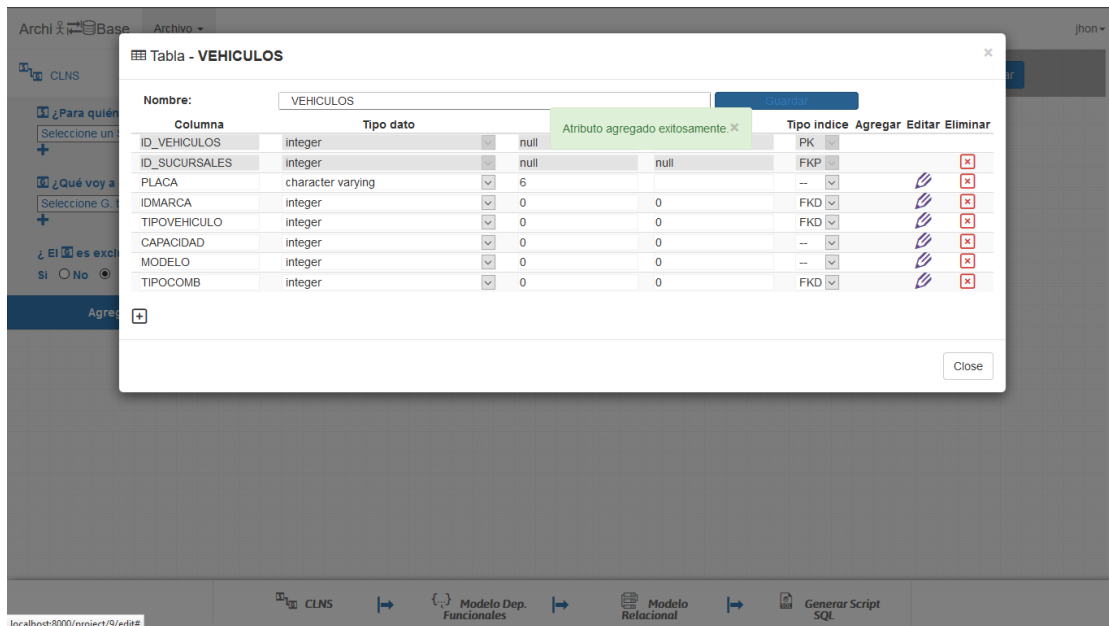


Figura 47. Adición Atributos a la Entidad VEHICULOS usando el prototipo CASE

Las tablas auxiliares se activan cuando se marca un atributo con el tipo de índice FKD.

The screenshot shows the CASE prototype interface. In the background, the 'Tabla - VEHICULOS' window is visible with a table containing one row: ID_VEHICULOS (integer, null, null, PK). In the foreground, the 'Tabla_aux - MARCAS' window is open, showing a table with two rows: IDMARCA (integer, 0, 0, PKE) and DESCRIP (character varying, 1009, 0, --). The 'Guardar' button is visible in both windows. The bottom toolbar includes icons for CLNS, Modelo Dep. Funcionales, Modelo Relacional, and Generar Script SQL.

Columna	Tipo dato	Tamaño	Precisión	Tipo indice	Agregar	Editar	Eliminar
ID_VEHICULOS	integer	null	null	PK			

Columna	Tipo dato	Tamaño	Precisión	Tipo indice	Agregar	Editar	Eliminar
IDMARCA	integer	0	0	PKE	+		-
DESCRIP	character varying	1009	0	--	+		

Figura 48. Creación de Tabla Auxiliar MARCAS usando el prototipo CASE

The screenshot shows the CASE prototype interface. In the background, the 'Tabla - VEHICULOS' window is visible with a table containing one row: ID_VEHICULOS (integer, null, null, PK). In the foreground, the 'Tabla_aux - TIPOVEHICULO' window is open, showing a table with two rows: TIPOVEHICULO (integer, 0, 0, PKE) and DESCRIP (SERIAL, empty, empty, --). The 'Guardar' button is visible in both windows. The bottom toolbar includes icons for CLNS, Modelo Dep. Funcionales, Modelo Relacional, and Generar Script SQL.

Columna	Tipo dato	Tamaño	Precisión	Tipo indice	Agregar	Editar	Eliminar
ID_VEHICULOS	integer	null	null	PK			

Columna	Tipo dato	Tamaño	Precisión	Tipo indice	Agregar	Editar	Eliminar
TIPOVEHICULO	integer	0	0	PKE	+		-
DESCRIP	SERIAL			--	+		

Figura 49. Creación de Tabla Auxiliar TIPOVEHICULO usando el prototipo CASE

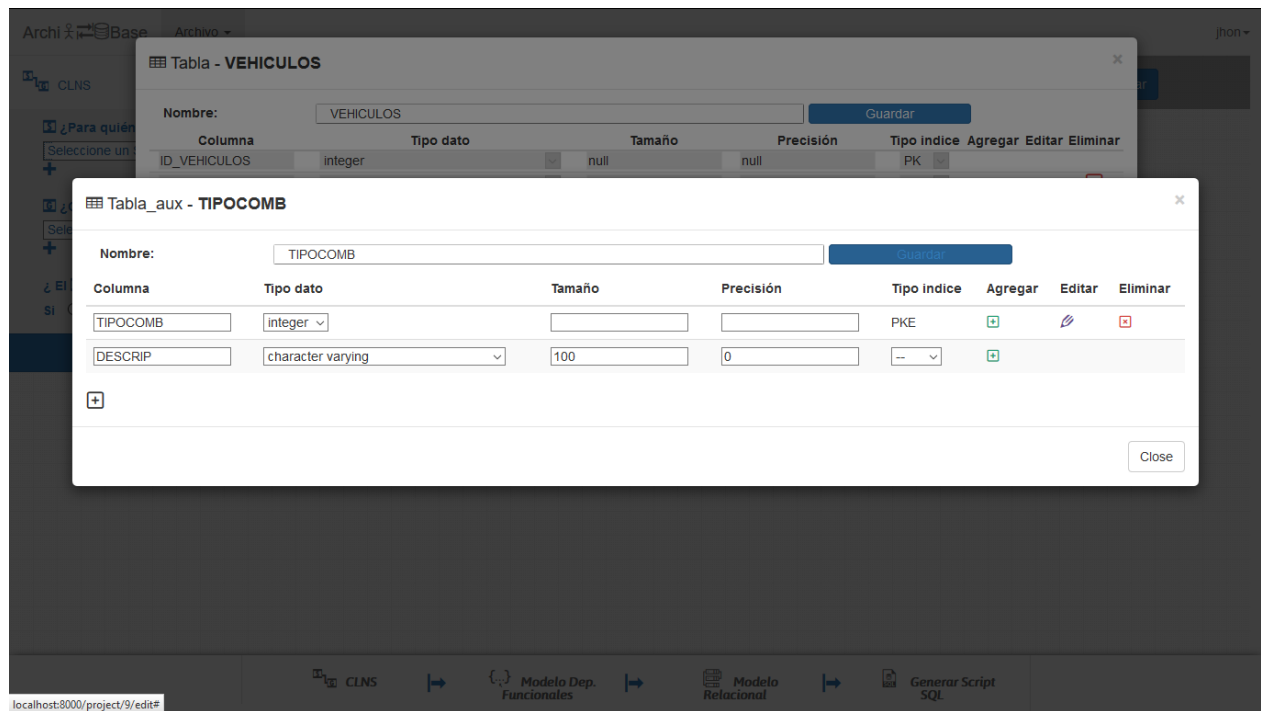


Figura 50. Creación de Tabla Auxiliar TIPOCOMB usando el prototipo CASE

Una de las características del modelo seudomatemático para el diseño de las bases de datos es la definición de dependencias funcionales no exclusivas, como se puede observar en la figura de la CLNS se cuenta con una DFNE entre las entidades CLIENTES Y VEHICULOS, esto quiere decir que existe en ellas una relación de muchos a muchos lo cual se resuelve mediante una llave compuesta; el sistema detecta esta dependencia y propone la llave compuesta.

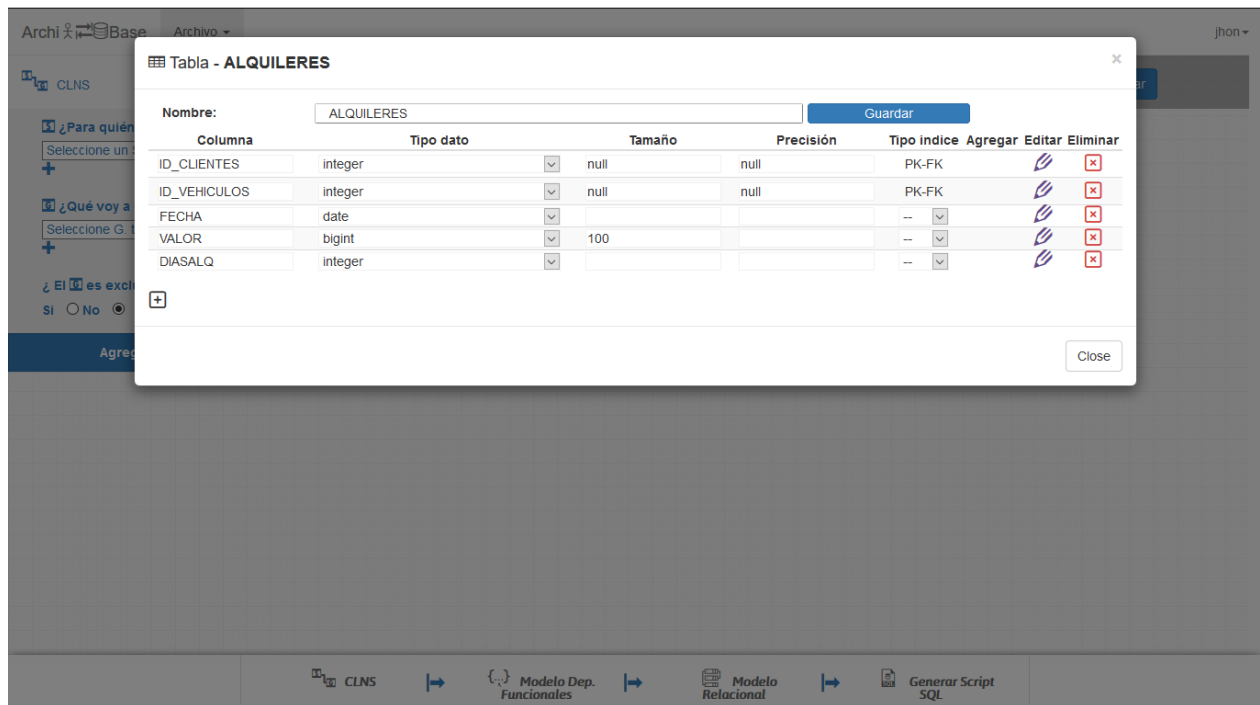


Figura 51. Creación de Tabla que resuelve DNFE usando el prototipo CASE

Una vez se han resuelto todos los atributos, es posible observar las tablas que se han creado mediante el modelo de dependencias funcionales.

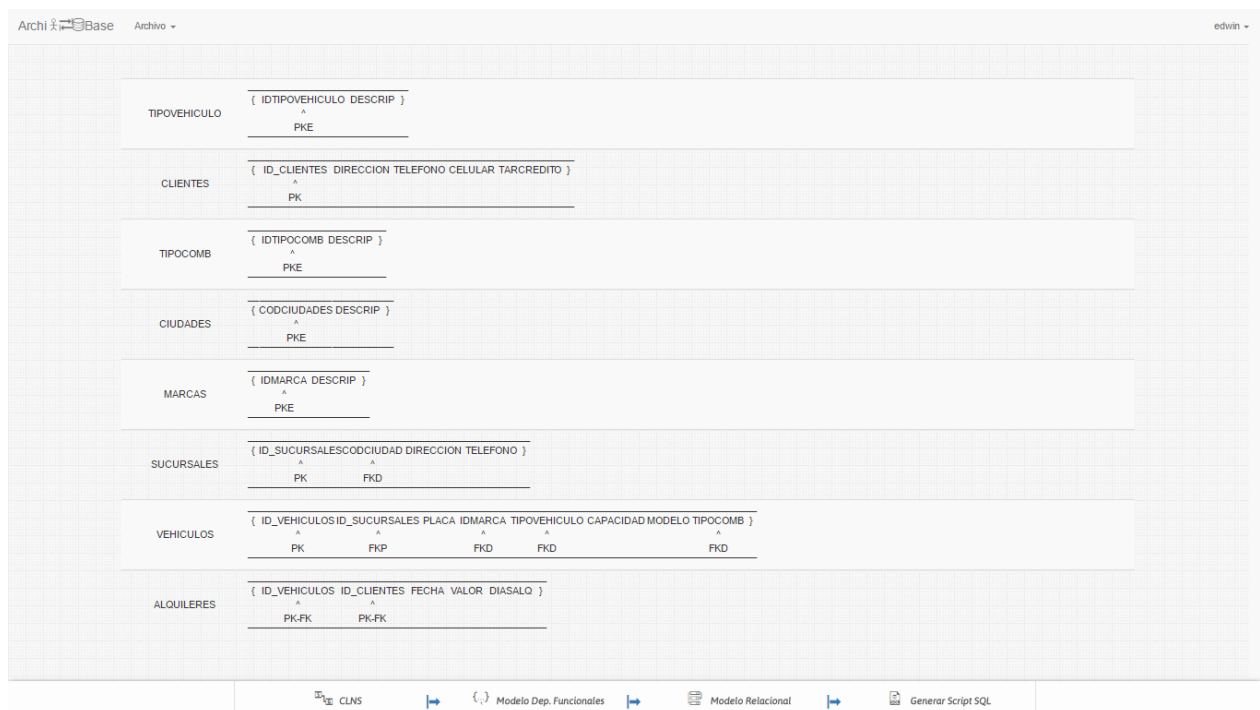
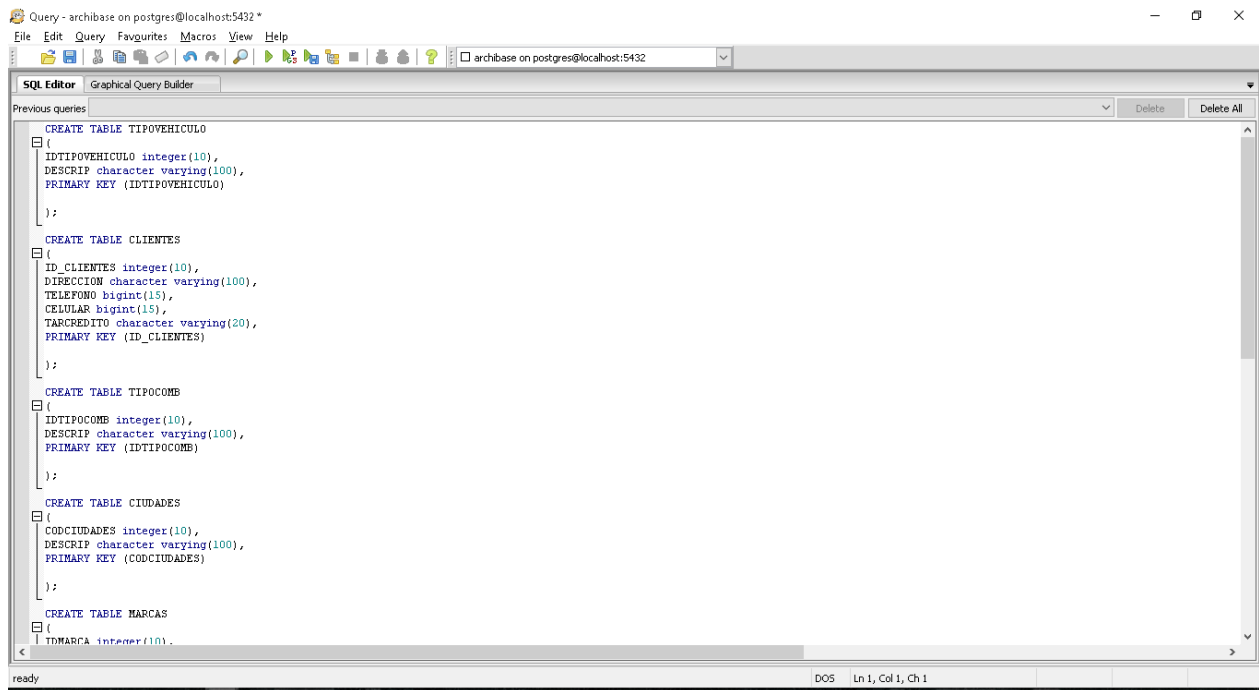


Figura 52. Modelo de Dependencias Funcionales usando el prototipo CASE

Si se tiene certeza sobre la resolución de la base de datos propuesta por el sistema, se cuenta entonces con la posibilidad de descargar el script de bases de datos, el cual se puede llevar a cualquier motor de bases de datos, para la creación de las tablas; el resultado del ejercicio se presenta en la siguiente imagen.



```
Query - archibase on postgres@localhost:5432 *
File Edit Query Favorites Macros View Help
SQL Editor Graphical Query Builder
Previous queries
CREATE TABLE TIPOVEHICULO
(
  IDTIPOVEHICULO integer(10),
  DESCRIP character varying(100),
  PRIMARY KEY (IDTIPOVEHICULO)
);

CREATE TABLE CLIENTES
(
  ID_CLIENTES integer(10),
  DIRECCION character varying(100),
  TELEFONO bigint(15),
  CELULAR bigint(15),
  TARCPREDITO character varying(20),
  PRIMARY KEY (ID_CLIENTES)
);

CREATE TABLE TIPOCOMB
(
  IDTIPOCOMB integer(10),
  DESCRIP character varying(100),
  PRIMARY KEY (IDTIPOCOMB)
);

CREATE TABLE CIUDADES
(
  CODCIUDADES integer(10),
  DESCRIP character varying(100),
  PRIMARY KEY (CODCIUDADES)
);

CREATE TABLE MARCAS
(
  TDMARCA integer(10),
```

Figura 53. Vista parcial del script SQL obtenido del prototipo CASE

6.1.2. Contraste

Haciendo referencia a la manera en que se diseña comúnmente una base de datos se presenta la resolución del mismo problema abordando el enfoque tradicional.

Lo primero es realizar una lista de entidades candidatas para la base de datos

Tabla 13 *Lista de Entidades*

Nombre entidad	Descripción
Vehículos	Características de un vehículo
Clientes	Características de un cliente

Sucursales	Características de una sucursal
Marcas	Tipos de marcas de un vehículo
Ciudades	Listado de ciudades
Tipo vehículo	Tipos de vehículos para alquiler
Tipo combustible	Tipos de combustible para el vehículo
Alquileres	Alquileres para un vehículo

En segunda instancia se deben dar los posibles atributos para cada entidad, así como la llave primaria candidata.

Tabla 14 *Lista de Atributos por Entidad*

Nombre entidad	Atributos
Vehículos	Placa
	Marca
	Tipo Vehículo
	Capacidad
	Modelo
	Tipo Combustible
Clientes	Cédula
	Dirección
	Teléfono
	Celular
	Número Tarjeta de Crédito
Sucursales	Id. Sucursal
	Cód. Ciudad
	Dirección
	Teléfono
Marcas	Id. Marca

	Descripción
Ciudades	Id. Ciudad
	Descripción
Tipo vehículo	Id. Tipo Vehículo
	Descripción
Tipo combustible	Id. Tipo Combustible
	Descripción
Alquileres	Placa
	Cédula
	Fecha
	Valor
	Días Alquiler

Por último defino qué relaciones pueden existir entre mis entidades y las diagramo en un modelo relacional

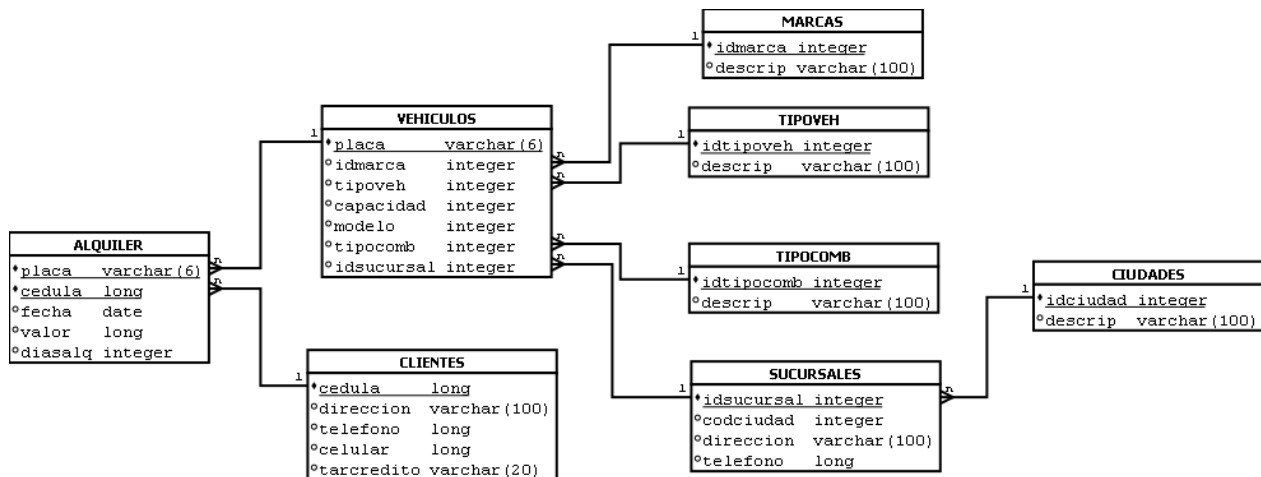


Figura 54. Caso de Estudio Modelo Entidad Relación - Método Tradicional

Cabe mencionar que gran parte de los ejercicios de bases de datos tradicionales se realizan directamente en el motor de bases de datos o mediante el uso de herramientas de gestión de bases de datos por lo que no es necesaria la obtención como tal un script porque ya la bases de datos se

han creado en el motor. Sin embargo, la base de datos se ha creado bajo el criterio del diseñador sin una metodología o guía que permita que la base de datos se realice pensando en las entidades propias del negocio; esto en comparación con el prototipo que aplica el modelo seudomatemático para la creación de bases de datos relacionales, el cual a través de la Cadena Lógica de Negocio del Sistema, representa los componentes principales del negocio que se está modelando.

6.1.3. Evaluación de los objetivos

A partir del estudio y aplicación de la metodología se muestra que la solución a los problemas de diseño definida por Modelo de Cadena Lógica de Negocio del Sistema - Dependencias Funcionales (Exclusivas y No Exclusivas) propuesto por el ingeniero John Jairo Londoño cumplen con los axiomas de Armstrong así como las formas normales.

Desde el entendimiento de los fundamentos teóricos planteados por el ingeniero John Jairo Londoño Pérez en su modelo seudomatemático para el diseño de bases de datos relacionales se han podido establecer los requerimientos funcionales para el diseño del prototipo CASE.

Se ha desarrollado un prototipo CASE que es funcional y permite obtener el script en lenguaje SQL para la creación de tablas en cualquier motor de bases de datos.

Más allá del cabal cumplimiento del objetivo principal, la construcción del prototipo, se ha podido encontrar también que la herramienta, cumpliendo con las reglas que se satisfacen usando el enfoque tradicional, permite entender el negocio en su dominio.

7. Capítulo VII Prospectivo Del Trabajo

7.1.1. Líneas de investigación futuras

Luego de concluido el trabajo, entendiendo que el modelo seudomatemático para el diseño de bases de datos relacionales se fundamenta en el entendimiento de la realidad y el contexto que se modela, se puede plantear un avance que se oriente a la aplicación de los lineamientos de la Ingeniería de software dirigida por modelos, en donde la comprensión del dominio es parte fundamental en el proceso de la misma.

7.1.2. Trabajos de investigación futuros

Dentro del proceso de evolución a partir del presente desarrollo del prototipo, se plantea la posibilidad de implementar un tutorial en línea de la metodología que guíe paso a paso al usuario en cada etapa que compone el modelo seudomatemático para el diseño de bases de datos relacionales.

De igual manera, se puede plantear la necesidad de implementar el reporte adicional, que consiste en la obtención del Modelo relacional y entidad relación para obtener vistas adicionales dentro del prototipo.

8. BIBLIOGRAFÍA

- [Bec05] BECK, KENT. Extreme Programing Explained. Second Edition. Pearson Education. U.S. 2005.
- [Boo99] Booch, Grady, (1999) The Unified Modeling Language User Guide, (1 ra. Ed), septiembre. Addison-Wesley Editorial.
- [Dat04] DATE, C.J. Introducción a los sistemas de bases de datos. Séptima Edición. Pearson Educación. México, 2004. 960.
- [Ken05] Análisis Y Diseño De Sistemas 3ª. Edición Kendall & Kendall Páginas 15.16.17.18
- [Kro03] KROENKE, D.M. procesamiento de bases de datos. Octava edición. Pearson educación. México, 2003.
- [Lon11] Londoño John, modelo pseudomatemático para el diseño de las bases de datos relacionales paginas 2.3.4
- [Pre93] PRESSMAN S., Roger: Ingeniería del Software – Un enfoque práctico, España, Mc Graw Hill, 3ª Edición, 1993, p 28.
- [Pre10] PRESSMAN S., Roger: Ingeniería del Software – Un enfoque práctico, España, Mc Graw Hill, 7ª Edición, 2010, p 37.38.

Referencias web

- [Alb00] Albizuri-Romero, M. B. 2000. A retrospective view of CASE tools adoption. SIGSOFT Software. <http://doi.acm.org/10.1145/346057.346071>

ANEXOS