SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

OPCIÓN I.- TESIS

TRABAJO PROFESIONAL

"DESARROLLO DE UNA HERRAMIENTA PARA LA GENERACIÓN DE DIAGRAMAS DEL MODELO CLIENTE Y NAVEGACIONAL BASADA EN ARTEFACTO CON GRAFOS Y TEORÍA DE CONJUNTOS".

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:

I.I. José Fernando Carreón Díaz de León

DIRECTOR DE TESIS:

M.C. Silvestre Gustavo Sergio Peláez Camarena

CODIRECTOR DE TESIS:

Dr. Ulises Juárez Martínez

INSTITUTO TECNOLÓGICO DE ORIZABA

ORIZABA, VERACRUZ, MÉXICO. **DICIEMBRE 2018**

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

FECHA: 05/12/2018
DEPENDENCIA: POSGRADO
ASUNTO: **Autorización de Impresión**
OPCIÓN: **I**

C. **JOSE FERNANDO CARREON DIAZ DE LEON**
CANDIDATO A GRADO DE MAESTRO EN:
SISTEMAS COMPUTACIONALES

De acuerdo con el Reglamento de Titulación vigente de los Centros de Enseñanza Técnica Superior, dependiente de la Dirección General de Institutos Tecnológicos de la Secretaría de Educación Pública y habiendo cumplido con todas las indicaciones que la Comisión Revisora le hizo respecto a su Trabajo Profesional titulado:

"DESARROLLO DE UNA HERRAMIENTA PARA LA GENERACION DE DIAGRAMAS DEL MODELO CLIENTE Y NAVEGACIONAL BASADA EN ARTEFACTO CON GRAFOS Y TEORIA DE CONJUNTOS".

Comunico a Usted que este Departamento concede su autorización para que proceda a la impresión del mismo.

C.A. TITULACIÓN

A T E N T A M E N T E

P.A.

DR. RUBEN POSADA GOMEZ
JEFE DE LA DIV. DE ESTUDIOS DE POSGRADO

SECRETARIA DE EDUCACIÓN PÚBLICA
INSTITUTO TECNOLÓGICO DE ORIZABA

ggc

Avenida Oriente 9 Núm. 852, Colonia Emiliano Zapata, C.P. 94320 Orizaba, Veracruz, México
Teléfonos: (272) 7 24 40 96 Fax. (272) 7 25 17 28
e- mail: Orizaba@itorizaba.edu.mx www.itorizaba.edu.mx

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

FECHA : 19/11/2018

ASUNTO: **Revisión de Trabajo Escrito**

C. DR. RUBEN POSADA GOMEZ
JEFE DE LA DIVISION DE ESTUDIOS
DE POSGRADO E INVESTIGACION.
P R E S E N T E

Los que suscriben, miembros del jurado, han realizado la revisión de la Tesis del (la) C. :

JOSE FERNANDO CARREON DIAZ DE LEON

la cual lleva el título de:

"DESARROLLO DE UNA HERRAMIENTA PARA LA GENERACION DE DIAGRAMAS DEL MODELO CLIENTE Y NAVEGACIONAL BASADA EN ARTEFACTO CON GRAFOS Y TEORIA DE CONJUNTOS".

Y concluyen que se acepta.

A T E N T A M E N T E

PRESIDENTE :	**M.C. SILVESTRE GUSTAVO SERGIO PELAEZ**	FIRMA
SECRETARIO :	**DR. ULISES JUAREZ MARTINEZ**	FIRMA
VOCAL :	**M.C. MARIA ANTONIETA ABUD FIGUEROA**	FIRMA
VOCAL SUP. :	**M.C. CELIA ROMERO TORRES**	FIRMA

EGRESADO(A) DE LA MAESTRIA EN **SISTEMAS COMPUTACIONALES**

OPCION: **I** **Tesis**

Avenida Oriente 9 Núm. 852, Colonia Emiliano Zapata, C.P. 94320 Orizaba, Veracruz, México
Teléfonos: (272) 7 24 40 96 Fax. (272) 7 25 17 28
e- mail: Orizaba@itorizaba.edu.mx www.itorizaba.edu.mx

Agradecimientos

En primera instancia agradezco a Dios por darme sabiduría e inteligencia para desenvolverme en esta área de la ingeniería, a mis padres por todo el apoyo otorgado durante la realización de mi maestría, así como a mi hermano quién se encuentra realizando su maestría en ciencias de la electrónica. A Edith Verdejo Palacios quién aportó ideas y alternativas de solución para la mejora de este trabajo de tesis.

A mi director de tesis, el M. C. Silvestre Gustavo Peláez Camarena por su apoyo, tiempo y esfuerzo para ofrecer sus consejos y observaciones al trabajo que se realizó y durante las presentaciones de seminario.

A los miembros del jurado, el Dr. Ulises Juárez Martínez, a la M. C. María Antonieta Abud Figueroa y a la M. C. Celia Romero Torres, por dedicarle su valioso tiempo durante la revisión del presente documento, así como sus retroalimentaciones para mejorarlo en virtud de la superación académica.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por la beca otorgada para la realización de mis estudios de posgrado.

Contenido

Índice de Figuras

Índice de Tablas

Resumen

La fase del análisis de requerimientos en el desarrollo de software propone la captura de las necesidades de los clientes. Los diagramas que se generan en esta fase sirven como punto de partida para el desarrollo de las demás fases del desarrollo de software tales como el diseño, la especificación, la implementación y las pruebas.

Con base en esta situación, el objetivo de esta tesis es generar una herramienta que soporte los diagramas del modelo del cliente y navegacional para facilitar a los clientes el entendimiento de estos diagramas y saber si se aproximan a lo que ellos buscan, y en caso contrario, ofrecer alternativas de solución o guías que permitan a los clientes atacar estos puntos débiles y ofrecer un producto final que cubra al cien por ciento sus exigencias.

La tesis se centra en los diagramas del modelo del cliente y el modelo navegacional. El modelo del cliente se enfoca en la obtención de conceptos que forman parte del modelado o el contexto de negocio, mientras que el modelo navegacional se centra en la navegación entre las diversas páginas que conforman la aplicación Web.

Para el desarrollo del sistema que consiste en la generación de diagramas del modelo cliente y navegacional con base en un lenguaje propuesto se utilizó la metodología OOHDM (*Oriented Object Hypermedia Design Method*, Método de Diseño Hipermedia Orientado a Objetos), los lenguajes PHP (*PHP Hypertext Preprocessor*, Procesador de Hipertexto de PHP), JavaScript, HTML5 (*HyperText Markup Language 5*, Lenguaje de marcado de hipertexto 5), CSS3 (*Cascading Stylesheets 3*, Hojas de estilo en cascada 3), Canvas y bibliotecas que extienden de JavaScript como jQuery y jQueryUI para el manejo de grafos que soporten animaciones y el manejo de eventos de arrastrar y soltar. Además, se utilizaron archivos XMI (*XML Metadata Interchange*, Intercambio de metadatos de XML) para la gestión de los metamodelos y la generación de los modelos del cliente y navegacional.

Abstract

The requirements analysis phase of the software development proposes capturing customer needs The diagrams generated in this phase allow to starting point to the other phases of software development such as design, specification, implementation and proofs.

Based on this situation, this thesis proposes to generate a tool whose support to create costumer and navigational diagrams in order to help custimers understand these diagrams and knows if they are looking for it, otherwise, offer another solutions who allow emphasize this weak points and improve the software specification who will cover all customer's needs.

This thesis is center on the customer & navigational models. The customer's model is based on get concepts from modeling or business context, while the navigational model is center on the investigation between diverse Web pages which make up the Web application.

In order to develop the system consisting of generate the customer & navigational model diagram it will be based on proposed language and it will be use the OOHDM methodology, PHP, JavaScript, HTML5, CSS3, Canvas and libraries who extend of JavaScript as jQuery & jQueryUI to manipulate graphs who support animations and event handler like a drag & drop. Furthermore it will be used XMI files in order to manage metamodels and customer & navigational models generation.

Introducción

Esta tesis se basa en [1] donde se propusieron los modelos del cliente para facilitar a los clientes el entendimiento de los modelos a través de la generación de diagramas del cliente y navegacional, y ofrecer una perspectiva simple sobre lo que se realizará con sus aplicaciones Web y dar retroalimentaciones desde la fase de análisis del desarrollo de software.

En la tesis anterior, los diagramas que se diseñaron permitieron a los desarrolladores tomar los caminos correctos para la construcción de sus aplicaciones en lo que se refiere a la arquitectura y la distribución de los elementos lógicos y físicos del sistema (bases de datos, bibliotecas externas, servicios Web, entre otros).

En esta tesis se desarrolló una herramienta para la generación de diagramas y la utilización de archivos XMI que fungirán como trabajo futuro para su conversión en bosquejos o arquitecturas físicas básicas que faciliten la construcción de las aplicaciones Web a través de un lenguaje apropiado.

Para ofrecer una visión completa de esta investigación y las partes que lo integran, el trabajo de tesis se compone de cinco capítulos

En el primer capítulo se presentan los conceptos que anteceden a esta investigación, el planteamiento del problema, los objetivos generales y específicos y la justificación que funge como motivación para ofrecer una solución al problema. En el capítulo dos se presentan los trabajos relacionados, se presenta un análisis comparativo en el que se muestra de forma resumida las herramientas que se generaron y que se toman como base en esta investigación y se presentan las alternativas de solución, así como la solución que se propone. En el capítulo tres se describe la arquitectura del sistema, la función de cada capa y módulos que la conforman, se muestran los procesos y la lógica del negocio. En el capítulo cuatro se ejemplifican dos casos de estudio que describen la utilidad del sistema, así como sus respectivas pruebas. Finalmente, en el capítulo cinco se presentan las conclusiones, productos académicos y recomendaciones, así como trabajo futuro.

Capítulo 1. Antecedentes

En este capítulo se muestran los elementos que sirven como base para este proyecto de tesis que incluyen los objetivos a realizar, la justificación y el planteamiento del problema.

1.1.Marco teórico

En esta sección se muestran los conceptos que tienen relación con el presente proyecto de tesis.

1.1.1. Grafo

Un grafo es una descripción visual que muestra bajo un contexto espacial información específica, como puntos entre dos coordenadas geográficas y la generación de trazos que conforman figuras geométricas. Visto de otra manera, Balakrishnan [2] define a los grafos como un conjunto de objetos que se relacionan entre sí para formar relaciones entre ellos; se componen de vértices, enlaces, y nodos que forman un conjunto de grafos dirigidos y no dirigidos que, con base en las necesidades y aplicaciones específicas, se utilizará uno u otro.

1.1.2. Teoría de grafos

La teoría de grafos, según la interpretación de Gross y Yellen [3], es un área de las ciencias exactas y ciencias de la computación que comprende y profundiza en los detalles de los grafos para su aplicación en diferentes áreas como álgebra, teoría de conjuntos, topologías y aritmética básica. Estos grafos se representan con dos variables V y E. Donde V es el número de nodos y E el número de enlaces entre ellos para formar una representación gráfica [4]. En la Figura 1.1 se representa gráficamente un grafo con nodos V = {1, 2, 3, 4, 5} y E = {{1,2}, {2,4}, {3,4}, {2,5}, {1,3}}.

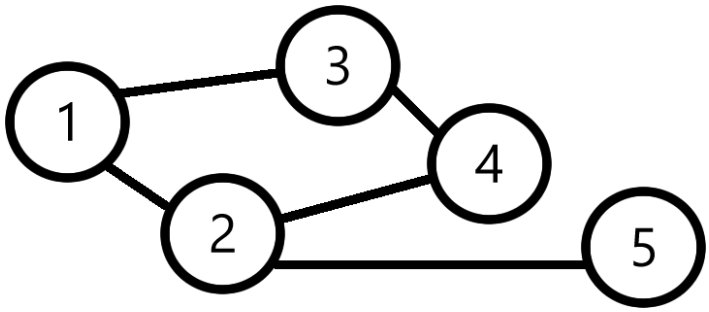


Figura 1.1 Representación de un grafo simple con variables V, E.

Existen los grafos orientados, grafos mezclados, multi-grafos, grafos simples, grafos dirigidos y grafos no dirigidos. Sin embargo, se muestran los grafos dirigidos que representan el contexto fundamental de este trabajo.

Según Elmasri et al. [5], el diseño conceptual de una base de datos ofrece un panorama más completo sobre las relaciones entre los datos y su consistencia en el conjunto de datos. En la Figura 1.2 se muestra un esquema de base de datos en el que cada tabla representa un nodo y sus relaciones representan los enlaces del grafo.

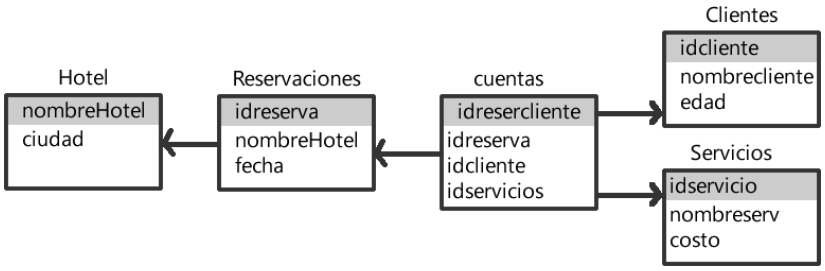


Figura 1.2 Diagrama entidad relación.

1.1.3. Modelo

Un modelo, según Rumbaugh et al. [6], es la representación abstracta de elementos tangibles o no tangibles que simplifican su entendimiento previo a su construcción. Estos modelos se utilizan en etapas de planificación y diseño de software para ofrecer múltiples vistas del sistema aplicando notaciones, grafos, esquemas e información adicional para comprenderlos.

Para crear un modelo bien formado se necesita definir correctamente el problema a resolver e identificar los elementos generales a resolver. La abstracción consiste en simplificar los desafíos

a través de la obtención de las características principales de un objeto y manejarlos a grandes rasgos dejando sus detalles fuera de contexto.

Por ejemplo, si se desea obtener la abstracción de un carro se obtendrán el número de ruedas, el número de puertas, el color y la posición de la silla del piloto. Sin embargo, no se necesita saber en primera instancia el tipo de combustible que utiliza, los kilómetros que recorrió, su peso o su alineación. Estos detalles se atacarán a medida que se realiza una evaluación a fondo del carro y se aplicarán en etapas de desarrollo.

1.1.4. Modelo del cliente

Con base en el trabajo propuesto por Lima Gámez, et al. [1], se utilizó el modelo del cliente que describe un grupo de objetos que comparten características en común (atributos) y su comportamiento (métodos). Este modelo de clases se basa en la teoría de conjuntos. En la Figura 1.3 se describe una notación que sigue las siguientes reglas equivalentes a los números que se muestran en ella:

1. Representa el nombre de clase y su uso es obligatorio.
2. Representa el conjunto de atributos que tiene la clase.
3. Representa el conjunto de métodos que tiene la clase.

Para los puntos 2 y 3 no se permiten el uso de acentos.

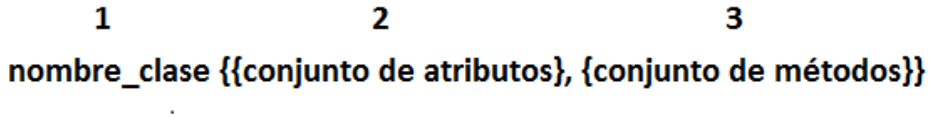


Figura 1.3 Notación del modelo del cliente.

Para generar el modelo del cliente se consideran las siguientes reglas:

1. La primera letra de la clase es mayúscula y cada atributo o método se separará por comas.
2. Si no hay atributos o métodos se representarán a través del conjunto vacío.
3. Si se trata de una clase abstracta el nombre de clase iniciará con “Abs_”.
4. Si se trata de una interfaz el nombre de la interfaz iniciará con “In_”.

5. Para representar una clase que implementa una interfaz se utilizará el símbolo de implicación.
6. Para representar la herencia se utilizará el símbolo de unión de conjuntos.
7. Para representar la composición y agregación de clases se utilizarán los símbolos $\in$ y $\subset$, respectivamente.
8. Para establecer la asociación entre clases, se utilizará el símbolo $\cap$ con sus diferentes multiplicidades (1-1, 1-*, *-*).
9. Para representar una clase asociación se utilizará la concatenación de la primera clase, seguido de la clase de asociación y finalizando con la segunda clase.

El modelo del cliente bajo un esquema de diagrama, cuenta con los siguientes elementos.

1. Rectángulo: representan a los nodos de concepto.
2. Círculos/óvalos: representan a los nodos de relación.
3. Flechas (rectas/curvas): representan el enlace entre nodos de concepto y relación.

La Figura 1.4 muestra la representación de los elementos descritos anteriormente:

Figura 1.4 Modelo del cliente.

1.1.5. Modelo navegacional

Con base en el trabajo propuesto por Lima Gámez, et al. [1], los modelos navegacionales representan un conjunto de rutas que un usuario sigue bajo un contexto de operación natural. Es decir, el seguimiento de cada actividad hasta su finalización. En la Tabla 1.1 se describen las notaciones que conforman el trabajo anterior y sirven como punto de partida para este trabajo.

Tabla 1.1. Notaciones con grafos del modelo navegacional

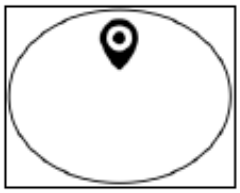
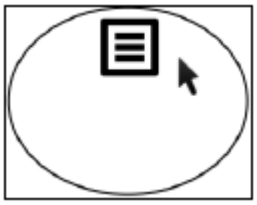
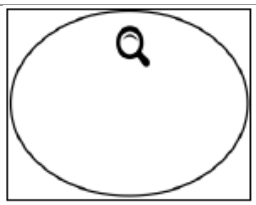
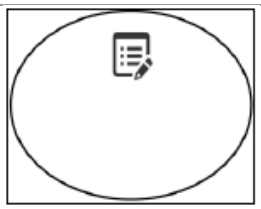
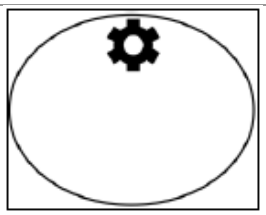
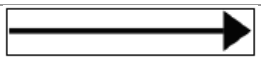
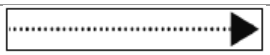
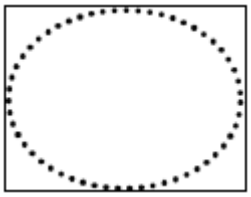
Nombre	Figura	Descripción
Nodo navegacional		Es el nodo de la navegación que trabaja con la información y recupera o modifica datos en la aplicación.
Nodo menú		Es el nodo de navegación que maneja los caminos alternativos de navegación.
Nodo consulta		Es el nodo de navegación donde la aplicación solicita información al usuario previo a continuar con la navegación.
Nodo lista		Es el nodo de navegación donde al usuario se le muestra una lista de resultados a visualizar.
Nodo proceso		Es el nodo de navegación donde se presentan procesos o tareas a realizar por la aplicación.
Enlace de navegación		Es un enlace de navegación entre un nodo de navegación y otro. Éste no se referencia a nodos de proceso.
Enlace proceso		Es un enlace de navegación entre nodos de proceso exclusivamente.

Tabla 1.1. Notaciones con grafos del modelo navegacional (continuación)

Nombre	Figura	Descripción
Acceso Usuario		Es un símbolo que indica un acceso a un destino de navegación.
Nodo inicio		Es un nodo de navegación que describe el punto inicial del modelo con grafos o el inicio de la aplicación.

La representación de cada nodo utilizando una representación formal se conforma de las siguientes reglas:

1. Para representar a un nodo se utiliza "(nombre_nodo)".
2. Para representar el tipo de nodo (ver Tabla 1.1) se utiliza "{propiedades}".
3. Para representar los enlaces entre nodos se tienen dos variantes:
    a. Si se trata de un enlace de navegación se utiliza "-[EnlaceNavegacional]->".
    b. Si se trata de un enlace de proceso se utiliza "-[EnlaceProceso]->"

Con base en las reglas anteriores, en la Figura 1.5 se muestra una relación simple entre un videoclub y un menú principal:

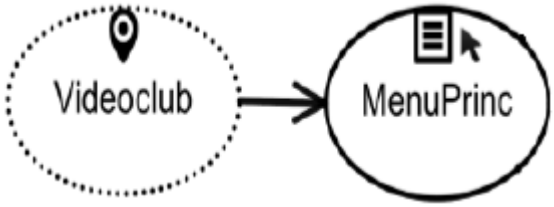


Figura 1.5 Modelo navegacional.

Con base en las reglas descritas anteriormente y en la Figura 1.5, se describe su notación equivalente:

```
(Videoclub){NodoNavegacional}-[EnlaceNavegacional]->(MenuPrinc){NodoMenu};
```

1.1.6. Diagrama

Un diagrama según las definiciones de Weilkiens y Oestereich y de Duc [7], es la representación gráfica de un modelo que extiende una estructura, comportamiento o combinaciones entre los diversos conceptos de los sistemas. Estos incluyen elementos como un nombre, atributos, variables y operaciones. En el ámbito computacional existen diversos diagramas tales como los diagramas de UML [8] que en conjunto ofrecen una perspectiva visual de lo que se realiza o se resuelve. Un modelo representa una serie de pasos que sirven como punto de partida para realizar acciones repetibles como por ejemplo la fabricación de un vaso a partir de un modelo. La diferencia entre estos dos conceptos es que el modelo se aplica en todas las etapas de desarrollo de software, mientras que el diagrama realiza una interpretación visual de alguna parte del sistema.

Por ejemplo, Radoaca [9] define a los diagramas Venn como la representación gráfica de las relaciones entre conjuntos para realizar las operaciones (intersección, inclusión y unión) sin cambiar su posición en el universo. La Figura 1.6 describe un diagrama de Venn de intersección.

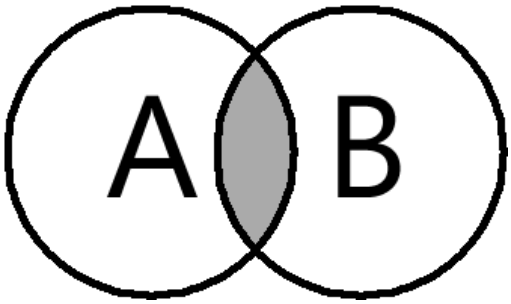


Figura 1.6 Diagrama de Venn para la intersección.

1.1.7. Aplicación Web

Una aplicación Web, según Moseley [10], es un sitio Web que contiene páginas con contenido estático parcial, total o nulo. La página Web se completa cuando se solicita el archivo al servidor

Web y retornará un resultado diferente en función de las variables del entorno como el visualizador que se esté utilizando (Google Chrome, Firefox, Internet Explorer, Safari), la dirección IP, el país de origen y las variables de sesión, por lo que al final se obtiene una página Web dinámica. Sus usos principales son:

1. Localizar información con facilidad.
2. Gestionar los datos que se obtienen de los visitantes.
3. Actualizar sitios Web cuyo contenido es volátil o tolerante a cambios.

La Figura 1.7 muestra el flujo principal de solicitud de una aplicación Web. Este flujo consiste en enviar una solicitud al servidor a través de Internet. El servidor Web reenvía la solicitud al servidor de aplicaciones Web adecuado. El servidor de aplicaciones obtiene los datos de la base de datos y lo retorna al servidor de aplicaciones y éste lo envía al servidor Web para que retorne una respuesta al cliente.

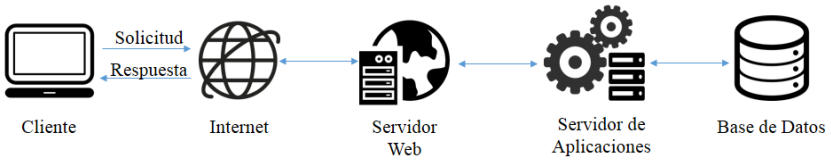


Figura 1.7 Flujo principal de una aplicación web.

Las aplicaciones Web se basan en la ingeniería Web para realizar aplicaciones Web utilizando un enfoque metodológico. La funcionalidad de las aplicaciones Web tiene en cuenta los aspectos de usabilidad, funcionalidad, fiabilidad, seguridad, eficiencia y mantenibilidad. Algunos archivos que se ejecutan del lado del servidor y fungen como la base para la aplicación Web son PHP, CGI (*Common Gateway Interface*, Interfaz de Entrada Común), Perl, ASP (*Active Server Pages*, Páginas activas del servidor), JSP (*JavaServer Pages*, Páginas de servidor Java), entre otros.

1.1.8. XML

XML (*eXtensible Markup Language*, Lenguaje de Marcado Extensible), según Boulanger [11], es un formato de texto jerárquico o con forma de árbol que contiene información particular para

describir elementos como configuraciones, datos y facturas, para su entendimiento por aplicaciones que recopilan su contenido para diferentes propósitos.

Un documento XML posee las siguientes reglas:

1. Sensible a mayúsculas y minúsculas.
2. La eliminación de espacios en los elementos.

Sintaxis de XML

A continuación, se muestra la sintaxis básica de un documento XML

```
<?xml version="1.0" encoding="UTF-8"?>
<raiz>
	<elemento>Contenido</elemento>
	<time id="2017-01-01 16:32">Domingo 1 de enero de 2017 a las 16:32</time>
	<hijo nombre="Fernando" edad="60">
		<nieto nombre="Juan Manuel" edad="35">
			<bisnieto nombre="Carlos" edad="5"></bisnieto>
		</nieto>
	</hijo>
</raiz>
```

1.1.9. XMI

XMI es una extensión de XML que valida los metamodelos que se representan un documento de XML [12]. Cada documento XMI se compone de una breve descripción de los elementos generales que contiene el modelo, así como la descripción de cata atributo y elemento que conforma al documento, a esta estructura se le conoce como esquema. Este esquema ofrece a los validadores de XML verificar la sintaxis y la semántica de un documento XML y su uso es completamente opcional.

Cada esquema XMI cumple con los siguientes puntos.

a. Todos los elementos de un documento XML que se definieron como un metamodelo en XMI se necesitan importar.
b. Las declaraciones de los metamodelos se declaran de tipo simple (*SimpleType*). En algunas ocasiones se utilizan declaraciones de tipo compleja (*ComplexType*).
c. Todos los elementos XML que se incluyan tienen que hacer referencia en alguna declaración del esquema.

Ejemplo de la estructura básica de un documento XMI

Se tiene un ejemplo en el que se requiere modelar un objeto auto que tenga atributos marca y modelo. La Figura 1.8 muestra el objeto a modelar.

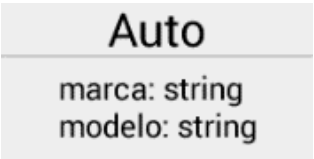


Figura 1.8 Objeto auto a modelar.

A continuación, se muestra el documento XMI que representa el modelo de un auto

```
<?xml version= "1.0" encoding= "UTF-8"?>
<xsd:schema
	xmlns:xmi="http://www.omg.org/XMI"
	targetNamespace="http://www.ejemplo.com/EspacioAutos"
	xmlns:cds="http://www.ejemplo.com/EspacioAutos">
	<xsd:complexType name="Auto">
		<xsd:choice minOccurs="0" maxOccurs="unbounded">
			<xsd:element name="marca" type="xsd:string" />
			<xsd:element name="modelo" type="xsd:string" />
			<xsd:element ref="xmi:Extension" />
		</xsd:choice>
		<xsd:attribute ref="xmi:id" />
		<xsd:attribute name="marca" type="xsd:string" />
		<xsd:attribute name="modelo" type="xsd:string" />
	</xsd:complexType>
	<xsd:element name="Autos" type="cds:Auto" />
</xsd:schema>
```

La representación de este documento, es el equivalente a la Figura 1.8.

1.1.10. Canvas

El elemento <canvas> se utiliza generalmente para dibujar gráficos utilizando lenguajes de *script* como JavaScript. La introducción de esta etiqueta es una alternativa a las tecnologías de Adobe Flash Player, dado que esta última no se ejecuta en dispositivos móviles como tabletas o teléfonos inteligentes. Sin embargo, *canvas* realiza una representación de contenido dinámico con gráficos 2D y mapas de bits, así como sus correspondientes animaciones. *Canvas* se utiliza generalmente para la construcción de gráficos, juegos, animaciones 2D y 3D, entre otros [13].

El *canvas* es una región en la que se dibujan elementos con base en el límite de ancho y de altura. Normalmente, JavaScript utiliza esta etiqueta para acceder a su contenido y generar un gráfico.

1.1.11. SVG

SVG (*Scalable Vector Graphics*, Gráficos vectoriales escalables) es un formato gráfico de vectores en 2D que se representan estáticos o dinámicos en formato *XML* que se definió por el *W3C* en 1999 [14]. El dibujado de un elemento SVG se realiza de forma programática o través de *JavaScript*. Posee compatibilidad nativa con el DOM (*Document Object Model*, Modelo del Objeto del Documento) por lo que reacciona a eventos *onClick* y *onMouseOver*.

1.1.12. PHP

PHP es un lenguaje de programación que se ejecuta en el lado del servidor que interpreta y genera un resultado con base en un archivo PHP (`.php, .php3, .php4, .phps`, entre otros formatos de archivo compatibles con PHP) que se muestra en un visualizador o en otros mecanismos tales como servicios Web, llamadas *AJAX*, entre otros. Este lenguaje se considera imperativo, funcional, reflexivo y orientado a objetos, por lo que entra en la categoría multi-paradigma. Actualmente se encuentra en la versión 7.0 con una vida de soporte por parte del grupo de PHP hasta el 3 de noviembre de 2018 [28], [29].

Dentro de sus ventajas destacan su uso en múltiples plataformas. Además, PHP posee la capacidad de ampliarse a través de extensiones como *Symfony*, *OpenSSL*, *PDF*, entre otros.

1.1.13. OOHDM

OOHDM es una metodología que gestiona la construcción de una aplicación a través de la especificación de meta-modelos conceptuales, navegacionales y de interfaz del usuario. Cada meta-modelo consiste en la descripción de una parte de la aplicación Web que funge como base en la generación de código a tiempo de ejecución con los cambios que se realicen en cada fragmento de la aplicación [43].

Suponga que tiene una aplicación de renta de autopartes y cada parte se compone por su marca, tamaño, características físicas, entre otros. Las relaciones que se tienen con otras autopartes son de pieza compuesta o de relación con otra autoparte. A continuación, se describen los pasos que detallan la construcción de una aplicación Web:

- **Análisis de requerimientos:**

  En este paso se obtienen los requerimientos de la aplicación tales como los actores y sus tareas a través del uso de UML que incluyen los diagramas de casos de uso y diagramas de interacción de usuario. En conjunto, estos diagramas ofrecen una perspectiva visual de las interacciones que se realizarán y permiten validar la aplicación de todos los requerimientos antes de continuar con los pasos posteriores
- **Diseño conceptual:**

  En este paso se construye un modelo conceptual que se centra en el dominio de la aplicación y no en quienes las utilizarán. Para ello se desarrolla un diagrama de clases
- **Diseño navegacional:**

  En este paso se generan vistas navegacionales en la que cada actor y su relación con cada tarea se describe como un lugar de interacción entre el usuario y la aplicación Web. OOHDM define algunos nodos de hipermedia como nodos, enlaces, estructuras de acceso, entre otros. Existen nodos que involucran accesos obvios de la aplicación (buscar, registrar, modificar o eliminar una autoparte) y se consideran dentro del contexto de la navegación
- **Diseño de interfaz:**

  La interfaz del usuario se compone del diseño estructural y de comportamiento. Cada diseño ofrece un bosquejo del cómo se representará la información del modelo del dominio de la aplicación brindando un panorama que sirve como ruta a seguir cuando se llegue al paso de la implementación. El diseño estructural utiliza las ADV (*Abstract Data View*, Vistas de datos abstractas) y representa lo que el cliente desea como resultado interactivo en su aplicación Web y los ADO (*Abstract Data Object*, Objetos de datos abstractos)
- **Implementación:**

En este paso, la implementación consiste en codificar lo que se ha realizado y utilizar toda la tecnología, arquitectura y bases de datos que se requieran para completar el proceso de desarrollo de la aplicación

Finalmente, se obtiene una aplicación Web completa. Una recomendación es seleccionar la tecnología antes de realizar el diagrama de clases para refinar el proceso de desarrollo y evitar la gestión de posibles inconvenientes con la aplicación.

1.1.14. PHPStorm

PhpStorm es una aplicación de escritorio de paga disponible para *Microsoft Windows*, *Mac OS X* y *Linux* que gestiona un editor para PHP, HTML y JavaScript con análisis del código fuente en tiempo real y accesos directos que simplifican la tarea de construcción de aplicaciones. Estas tareas implican la generación código, aplicación de segmentos de código de propósito general y verificación de código duplicado tanto para *PHP* y *JavaScript*. Además, se integra de forma automática con editores de *SQL* para generar resultados en la misma aplicación sin recurrir a las aplicaciones gráficas de los SGBD (Sistema Gestor de Bases de Datos) [32].

1.2. Planteamiento del problema

En el modelado de aplicaciones Web existen diversas metodologías orientadas a objetos que utilizan UML como lenguaje semi-formal para construir diversas aplicaciones. Este lenguaje implica un alto conocimiento por parte de los desarrolladores para su implementación y dificulta la comunicación entre los clientes en el contexto de asegurar que se satisficieron sus necesidades. Como alternativa, Lima Gámez, et al. [1] desarrollaron una “Propuesta de artefactos basados en grafos y conjuntos para el modelado conceptual de aplicaciones Web”, para establecer una notación formal como sustento a los modelos del cliente y navegacional, que faciliten la visualización del proceso de desarrollo por los patrocinadores del proyecto de manera clara y amigable.

Por lo tanto, como siguiente etapa al procedimiento alternativo para el desarrollo de aplicaciones se requiere desarrollar una herramienta que, basada en la notación para los modelos propuestos, permita la generación de estos modelos de manera gráfica.

1.3. Objetivo general y específicos

A continuación, se muestran los objetivos generales y específicos del presente proyecto de tesis.

1.3.1. Objetivo general

Desarrollar una herramienta soportada en los artefactos con base en grafos y teoría de conjuntos, que permita el modelado conceptual de aplicaciones Web, para la generación de los Modelos del Cliente y Navegacional propuestos para estos artefactos.

1.3.2. Objetivos específicos

1. Estudiar los artefactos propuestos para el modelado de aplicaciones alternativo.
2. Identificar y proponer tecnologías a utilizar para el desarrollo de la herramienta.
3. Analizar metodologías para el desarrollo de aplicaciones y seleccionar la más adecuada.
4. Proponer el procedimiento metodológico para el desarrollo de la herramienta.
5. Desarrollar dos modelos de estudio basados en la herramienta para el modelado de aplicaciones para validar y probar la funcionalidad del producto obtenido.
6. Generar la notación formal XMI para permitir la portabilidad de los modelos obtenidos.

1.4.Justificación

El desarrollo de aplicaciones Web requiere la gestión de un modelo que le ofrezca a un cliente una idea general de cómo funcionará la aplicación que requiere. Existen diagramas de UML que describen estos modelos. Aunque los desarrolladores comprenden el funcionamiento de estos diagramas, son complejos de entender por parte de los clientes, por lo que no tienen una idea desde su punto de vista de cómo se desarrollará su aplicación y de si cumple o no con sus requerimientos. Dada esta problemática, se necesita una herramienta que con base en la notación propuesta en [1] a través de los modelos cliente y navegacional, generen los diagramas de una forma más comprensible para los usuarios a través del desarrollo de un modelo de cliente y navegacional y que al mismo tiempo, sea funcional para los desarrolladores en las siguientes etapas de desarrollo según la metodología que elija cada uno.

Capítulo 2. Estado de la práctica

En este capítulo se presenta el análisis del estado de la práctica que muestra los trabajos que se relacionan con este proyecto de tesis.

2.1.Trabajos relacionados

El proceso de desarrollo de software orientado a servicios se posicionó a través de los años sustituyendo a los sistemas de información tradicionales, llevando todos los modelos de negocio y sus operaciones a entornos Web donde se acceden de forma transparente. Sin embargo, el desarrollo y despliegue de estos servicios Web se complican a medida que los sistemas crecen debido a la necesidad de obtener en su concepto natural, la flexibilidad y agilidad de interacción con los componentes. Por tal motivo, Baghdadi et al. [19] propusieron la generación de una herramienta CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) que represente un meta-modelo para determinar perspectivas de resumen y facilite la descripción de servicios Web para la generación de artefactos descritos en dicho meta-modelo que monitoricen los servicios Web de manera implícita. Con el desarrollo de la herramienta CASE se generaron cinco componentes (decisiones de diseño, abstracción de nombres, flexibilidad, agilidad y arquitectura) que se interrelacionaron entre sí y que brindaron dos perspectivas a soluciones de encapsulación y funcionamiento de manera independiente o embebida.

El diseño es la etapa que determina el éxito en el desarrollo de software. El diagrama de actividades de UML es el modelo principal que describe el comportamiento secuencial del software, así como el comportamiento errático del mismo y los mecanismos para salvaguardarlo sin generar alguna inestabilidad al sistema en cuestión. Es por esto que Hettab et al. [20] propusieron la generación de dos gramáticas de grafos que disminuyeron el tiempo y costo de desarrollo de software para mejorar su calidad desde la etapa de diseño. El primer grafo y modelo EADG (*Extended Activity Dependency Graph*, Gráfico extendido de dependencia de actividad) abstrae la esencia del diagrama de actividades. El segundo grafo genera un conjunto de casos de prueba con base en el modelo EADG y utilizando los criterios de cobertura.

Finalmente, este enfoque simplifica a los desarrolladores la capacidad de detectar errores en su etapa de diseño, y en un trabajo futuro el modelo EADG evaluará las métricas de complejidad del diseño de software para desarrollar aplicaciones con un bajo acoplamiento y una alta cohesión de forma automática.

Actualmente los lenguajes de dominio específico se desarrollan para resolver los problemas de un grupo particular de usuarios, dificultando la reutilización de conceptos y elementos básicos de los lenguajes base. Por esta razón, Melo, Sánchez, y Villalobos [21] plantearon un medio para describir la adaptación y composición de la sintaxis gráfica de los lenguajes de modelado a partir del uso de los elementos existentes y la generación de herramientas de lenguajes compuestos. Se desarrolló una herramienta que combina la composición de la sintaxis gráfica y abstracta, para describir lenguajes de modelado de dominio específico y la generación de editores.

La interpretación de los diagramas dibujados a mano representa un proceso largo para analistas y programadores, ya que no se reporta una herramienta potente capaz de reconocer e interpretar los trazos realizados por el usuario y convertirlos en representaciones simbólicas que capturen la intención del usuario. Para solucionar este problema, Freeman y Plimmer [22] sugirieron analizar la semántica de los conectores pertenecientes a los grafos no dirigidos, dirigidos y organigramas por medio del módulo de reconocimiento de formas perteneciente a la herramienta InkKit. Para ello fue necesario explorar los conectores de los gráficos anteriormente mencionados, a través de enfoques alternativos. Finalmente, identificaron los requisitos generales de los conectores e integraron un complemento de conectores dentro del módulo de reconocimiento de formas en InkKit.

La generación automática de código fuente a través de los modelos bien formados en UML representa un reto fundamental para los investigadores y centros de desarrollo, quienes con la evolución del software generan herramientas cada vez más sofisticadas que coadyuvan a minimizar el error entre los diagramas propuestos por el equipo de diseño y la traducción a código fuente por parte del equipo de desarrollo. Con base en esto, Kundu et al. [23] propusieron un enfoque que genera un código fuente a partir de diagramas UML o BDs (*Behavioral diagrams*, Diagramas de comportamiento) mediante la construcción de un modelo SIG

(*Sequence Integration Graph*, Gráfico de integración de secuencia) que integra las tecnologías XML y XMI. Este modelo establece una serie de reglas de mapeo que usando técnicas de escaneo, integración y validación integran un conjunto de artefactos de código útiles para cualquier propósito. Finalmente, tras realizar pruebas experimentales utilizando el modelo SIG se observó que se generaron cerca del 48% de líneas de código más robustas y con menor complejidad asintótica que otros generadores de código disponibles en herramientas CASE actuales.

Actualmente, las herramientas CASE como Visual Paradigm generan diagramas que disminuyen la complejidad de un problema de desarrollo de software. Sin embargo, los diagramas brindan mejores perspectivas de solución en la fase de diseño y desarrollo de software y las herramientas CASE comúnmente generan códigos fuente poco flexibles y genéricos que motivan a los desarrolladores a utilizar otros mecanismos para solucionar sus problemas. Por tal motivo, Canovas y Cugnasca [24] propusieron una extensión de un meta-modelo de un lenguaje de modelado de dominio específico para programas adaptables a través de la implementación de MDE (*Model Driven Engineering*, Ingeniería dirigida por modelo) y de BADAL (*Basic Adaptive Language*, Lenguaje adaptable básico), que en conjunto generan una cantidad parcial de código fuente como resultado de la interpretación de los elementos del meta-modelo y de sus relaciones con otros meta-modelos. Como resultado, se generó una herramienta CASE para diversos programas adaptables con lo que se aprovechó el máximo potencial de la tecnología MDE para obtener la máxima abstracción de los elementos para una clase de aplicación.

Durante el ciclo de vida del desarrollo del software los niveles en las abstracciones de los requerimientos son una pieza fundamental para el éxito de una aplicación. A pesar de que UML contribuye a diseñar un conjunto considerable de elementos dinámicos, se necesita reducir la brecha entre los problemas del software y de su implementación. Con base en esta problemática, Schumacher et al. [25] propusieron la reconstrucción de los meta-modelos de los diagramas de secuencia de UML a través de la inserción de OCL (*Object Constraint Language*, Lenguaje de restricción de objetos) que altera la descripción XMI. Con esto se desarrolló una herramienta CASE para generar de forma automática un código fuente consistente que cubriera la mayor cantidad de los requerimientos iniciales. En resultados preliminares, la herramienta CASE

generó un esquema XML intercambiable con un contenido más enriquecido sobre las restricciones y comportamiento del sistema en cuestión y se determinó que, a través de una comparación con un diagrama de secuencia original, los parámetros coincidieron en su mayoría con los requerimientos originales descritos en la etapa de diseño del software.

La generación de notaciones gráficas semi-formales como UML, diagramas entidad-relación o diagramas de bloque funcionales facilitan a los desarrolladores la tarea de diseñar y construir software con un aseguramiento de la calidad cada vez mejor. La validación y comprobación de estas notaciones se realizan sin ningún problema por separado. Sin embargo, la verificación en conjunto de estas notaciones que compruebe de forma simultánea la integridad del software representa un reto de calidad y de integración considerable. Dada esta situación, Jiang et al. [26] propusieron un enfoque que transformó notaciones semi-formales y formales en bloques lógicos que validaron la especificación de requerimientos contra la integridad de los modelos a evaluar a través de un entorno de modelado de requerimientos y de un lenguaje para describir los requerimientos recolectados de manera precisa. Tras finalizar las investigaciones se desarrolló una herramienta CASE en modo alpha que recolectó en principio los requerimientos de forma precisa y generó prototipos de requerimientos en forma animada. Como trabajo futuro se investigarán en detalle las notaciones semi-formales, como UML y el modelo entidad-relación, y las notaciones formales (*statechart*, Petri Net) para construir un dominio semántico que realice una evaluación completa y sin tendencia a fallos en cuestión a la interpretación de las mismas.

El análisis de requerimientos es la etapa principal del proceso de desarrollo de software que diseña, implementa y prueba posteriormente la aplicación resultante. Sin embargo, un mal entendimiento en la etapa inicial representa una amenaza potencial a todos los equipos que intervienen en la construcción del software. Tomando como principio el diseño orientado a objetos, Krishnan y Samuel [27] propusieron una metodología para generar diagramas de clases en UML a través de la especificación de requerimientos o del lenguaje natural, con el propósito de generar modelos o esqueletos completamente consistentes y con el menor número de fallas posibles. Finalmente, se construyó una metodología REM (*Relative Extraction Methodology*, Metodología de extracción relativa) y se generó una representación de grafo intermedio DG (*Dependency graph*, grafo de dependencia) que coadyuvó a la generación de tres nodos (de

concepto, valor y acción) para generar dicho diagrama de clases. Como trabajo futuro se planea extender la metodología con relaciones avanzadas que incluyen agregaciones y dependencias entre las clases, así como sus multiplicidades.

La sintaxis consiste en una serie de reglas gramaticales que generan programas, modelos y diagramas para un sinfín de propósitos. Los modelos tradicionales se centran en la revisión meticulosa de la semántica, sintaxis y léxico para que cumplan con su propósito. Sin embargo, estas estructuras de sentencias se convierten en tecnologías cada día más obsoletas a medida que los tiempos de entrega y costos disminuyen y requieren nuevas formas de traducir problemas en propuestas de solución. Bajo estas limitaciones, Mayfield [28] propuso la creación de un diagrama de sentencia tomando como base un grafo de dependencia. Dicho grafo consiste en la escritura de textos en lenguaje natural (inglés) que extrae el sujeto, predicado, objeto y complemento para formar relaciones entre las palabras y generar un análisis de requerimientos más completo. A través de la utilización de oraciones de prueba se detectó una precisión promedio del 92% y se generaron los grafos correspondientes que agruparon de igual forma los requerimientos en un diagrama de sentencias. Sin embargo, aún se afrontan retos como el vicio del lenguaje y comprender hasta qué punto un margen de error será suficiente para el usuario del programa, por lo que estos elementos se consideran para un trabajo futuro.

La generación de aplicaciones utilizando metodologías ágiles y de fácil construcción, constituye un nuevo camino hacia la competitividad entre las empresas. Los diseñadores y desarrolladores expertos construyen software bajo los estándares de UML y utilizan herramientas CASE para la generación de diagramas en tiempos cada vez más cortos. Sin embargo, los estudiantes o recién graduados de ingenierías con un enfoque hacia el desarrollo de software encuentran una brecha muy fuerte entre los problemas que se abordaron en clases y los que se les solicitan en las empresas, por lo que necesitan nuevas herramientas que les generen sus diagramas de una forma más práctica. Con base en esta situación, Segundo et al. [29] propusieron la construcción de una herramienta CASE SDG (*UML Sequence Diagram Generator*, Generador de diagramas de secuencia UML) que genera diagramas utilizando mecanismos de inteligencia artificial y técnicas especializadas en el entendimiento del lenguaje natural. Con esto, se obtuvo un software que recibió en primera instancia un actor y en segunda instancia la descripción del problema

para generar un diagrama con opciones de exportación JPG o XML y obtener con estos elementos una aplicación con un enfoque al aprendizaje de análisis de sistemas y diseño.

El entendimiento del lenguaje natural constituye en el ser humano el pilar de la comunicación con otros seres humanos y define su comportamiento desde una edad temprana. Algunos problemas con el aprendizaje se relacionan con situaciones delicadas que afectan esta capacidad y dificultan su aprendizaje sobre todo en pacientes con diversos problemas psicológicos y sociales. Las nuevas tecnologías AT (*Assistive Technology*, Tecnología asistida) coadyuvan a los humanos a comunicarse a través de una CB (Communication Board, tableta de comunicación). Bajo estas circunstancias, Franco et al. [30] propusieron la generación de una herramienta CASE CBCASE que utilice un vocabulario de comunicación CB para interpretar elementos categóricos y relaciones de cohesión a través del desarrollo de un lenguaje de modelado CBML (*Communication Board Extensible Markup Language*, Lenguaje de marcado extensible de la tableta de comunicación). Con estos elementos, se desarrolló una aplicación con un enfoque GUI (*Graphic User Interface*, Interfaz gráfica de usuario) que entendió en una prueba experimental lo que el paciente expresó. Sin embargo, el vocabulario se les complicó a algunos pacientes, por lo que se pretende en un futuro mejorar algunas vistas del CB, así como la integración de la herramienta CASE y el generador CB para qué en conjunto, integren un nuevo paradigma derivado de AT y CB.

En el desarrollo de aplicaciones de alto riesgo, el tiempo de ejecución y sincronización con otras aplicaciones es un elemento principal para la coordinación y el buen funcionamiento de sistemas industriales, militares, de aviación, entre otros. Las anotaciones MARTE (*Modeling and Analisys of Real-time and Embedded Systems,* modelado y análisis de sistemas en tiempo real y embebidos) modelan diagramas de tiempo que les ofrecen una perspectiva sobre qué ocurre en cada segundo de ejecución de su aplicación. A pesar de que MARTE ofrece un soporte completo para sistemas electrónicos, la inclusión de software que coordine sus operaciones con dichos sistemas es una tarea riesgosa y requiere capturar a tiempo de ejecución la información de la línea del tiempo, sus marcas de verificación, valores, estados y duraciones de eventos. Bajo estas condiciones, Choi et al. [31] propusieron el desarrollo de un diagrama de tiempo con anotaciones de MARTE bajo los diagramas de UML que se enfocan principalmente en diagramas de

secuencia y diagramas de máquinas de estado, para generar modelos bien formados que garanticen una comunicación consistente en este tipo de aplicaciones de alto riesgo. Finalmente, se desarrolló un software que integró los elementos de entrada de los SDs/MARTE (*Sequence Diagrams with MARTE*, Diagramas de secuencia con MARTE) y SMDs/MARTE (*State Machine Diagrams with MARTE*, diagramas de máquinas de estado con MARTE), así como los diagramas de UML que crearon modelos bien formados a través de reglas de transformación. Como trabajo futuro se pretende extender el software para tratar con tipos más complejos de diagramas de secuencia y desarrollar una herramienta automatizada que soporte el trabajo de la extensión del software.

Un PLC (*Programmable Logic Controller*, controlador lógico programable) es un equipo de cómputo que se especializa en la automatización de los procesos industriales a través de señales o circuitos electrónicos que facilitan el control de la ingeniería automática. Existen diversas aplicaciones de software que gestionan la edición de diagramas LD (*ladder diagram*, diagramas de escalera) a través del uso de instrucciones, compuertas lógicas, símbolos gráficos o conexiones topológicas. Sin embargo, la edición se limita en cuanto a la posición gráfica de los elementos y el número de diagramas de escaleras que se permiten para cada aplicación. Con estas limitaciones, Hu et al. [32] propusieron el desarrollo de un software de edición de diagramas LD que se enfoca en almacenar información sobre una estructura o tabla de datos con el propósito de enriquecerlos y ofrecer un panorama más completo sobre las acciones que realiza dicho diagrama LD para su posterior aplicación en lenguajes orientados a objetos como C++. Finalmente, se lograron eliminar las restricciones de elementos, el establecimiento de parámetros adicionales directamente sobre los diagramas y la expresión de su complejidad para cada dispositivo.

En la descripción de requerimientos, los clientes que buscan automatizar sus procesos de negocio utilizan un patrón a la hora de expresar sus necesidades que se centran en el ¿Quién?, ¿Qué? y ¿Por qué? Estos clientes aumentan sus necesidades conforme las metodologías ágiles siguen su camino y provocan en la aplicación cambios no deseados que afectarían su mantenimiento o actualización en el futuro. Aunque las US (*User Stories*, historias de usuario) mitigan parte del problema, no cubren un soporte para el cambio de los requerimientos por parte

de los clientes. Dada esta problemática, Wautelet et al. [33] propusieron el desarrollo de una herramienta CASE que consiste en diagramas de relación personalizados y notaciones gráficas que utilizan las US en un ciclo iterativo para adaptarse de mejor manera a los cambios durante el proceso de desarrollo y mitigar el impacto sobre el producto final. Con estos elementos se construyó una aplicación que se espera obtenga la aprobación y validación como un nuevo enfoque de desarrollo.

2.2.Análisis Comparativo

La Tabla 2.1 muestra un análisis comparativo de las contribuciones de los trabajos relacionados y ofrece una perspectiva y diferencia sobre lo que se desarrolló con este proyecto de tesis.

Tabla 2.1 Tabla comparativa de los trabajos relacionados

Art.	Objetivo	Tecnologías	Resultado	Estado
[15]	Facilitar la administración de servicios Web como un conjunto de tecnologías de la información empresariales.	WSDL (Web *Service Description Language*, Lenguaje de descripción de servicios Web), XML, UDDI (*Universal Description, Discovery and Integration*, Descripción, descubrimiento e integración universal) SOAP (*Simple Object Access Protocol*, Protocolo simple de acceso a objetos)	Se desarrolló una herramienta CASE que simplificó la separación de componentes en un entorno de servicios Web.	Terminado
[16]	Disminuir el tiempo y costo de desarrollo de software en la etapa de diseño del modelado UML.	AToM^3 UML	Se construyeron dos grafos que contribuyeron a simplificar el proceso de desarrollo de una aplicación.	Terminado /con trabajo futuro.

Tabla 2.1 Tabla comparativa de los trabajos relacionados (continuación)

Art.	Objetivo	Tecnologías	Resultado	Estado
[17]	Describir la composición de la sintaxis gráfica de los lenguajes de modelado.	FUSION PICTURE GROMP	Se desarrolló una herramienta capaz de combinar la sintaxis gráfica y abstracta.	Terminado
[18]	Analizar la semántica de los conectores pertenecientes a grafos no dirigidos, dirigidos y organigramas.	InkKit	Se identificaron los requisitos generales en los conectores de diagramas.	Desarrollo
[19]	Desarrollar un enfoque a través del modelo SIG que genere código fuente a partir de diagramas de comportamiento de UML.	XML XMI	Se generaron mediante pruebas experimentales cerca del 48% de líneas de código de clases y métodos.	Terminado /Futuro.
[20]	Proponer una extensión de un meta-modelo de un lenguaje de modelado de dominio específico para programas adaptables.	BADAL MDE UML GPML	Se desarrolló una herramienta CASE para diversos programas adaptables que permitieran realizar una abstracción completa de los requerimientos.	Desarrollo.

Tabla 2.1 Tabla comparativa de los trabajos relacionados (continuación)

Art.	Objetivo	Tecnologías	Resultado	Estado
[21]	Reconstruir el modelo XMI del diagrama de secuencias de UML para generar un código fuente más completo.	UML XMI OCL	Se construyó un modelo y una herramienta CASE que incluyó un nuevo código fuente que proveyó una mejor calidad en su contenido.	Terminado
[22]	Construir un enfoque que valide notaciones gráficas formales y semi-formales para asegurar el cumplimiento de la especificación de requerimientos.	SREE RTRSM	Se desarrolló una herramienta CASE en modo *alpha* que colecta en principio los requerimientos de forma precisa y genera prototipos de requerimientos en forma animada.	Desarrollo.
[23]	Desarrollar una metodología que genere diagramas de clases UML a través de una representación gráfica que tome como base la representación del LN.	UML Java Eclipse Flex	Se construyó una herramienta CASE que muestra un diagrama de clases compatible con UML.	Terminado /Futuro
[24]	Generar un diagrama de sentencia tomando como base un grafo de dependencia de lenguaje natural.	Ninguna	Se obtuvo un algoritmo que agrupó requerimientos en un	Desarrollo.

			diagrama con una precisión promedio del 92%.	

Tabla 2.1 Tabla comparativa de los trabajos relacionados (continuación)

Art.	Objetivo	Tecnologías	Resultado	Estado
[25]	Desarrollar un diagrama de secuencia UML a través del procesamiento de la descripción de casos de uso utilizando mecanismos de inteligencia artificial y lenguaje natural.	Java UML JPG XML	Se desarrolló una aplicación que tradujo una descripción en un diagrama compatible para mejorar el aprendizaje de análisis de sistemas y diseño.	Terminado
[26]	Desarrollar una herramienta CASE para el entendimiento del vocabulario CB en pacientes con problemas de comunicación del habla y entendimiento del lenguaje natural.	XML BoardMaker Software	Se construyó una aplicación CBCASE que permitió interpretar los mensajes de un paciente a través de gráficos GUI.	Terminado /plan futuro.
[27]	Generar una aplicación que combine las anotaciones de MARTE con diagramas de UML	MARTE UML	Se desarrolló un software que integró SDs/MARTE, SMDs/MARTE con UML para crear modelos bien formados.	Terminado /plan futuro.

	para obtener información en tiempo real.			

Tabla 2.1 Tabla comparativa de los trabajos relacionados (continuación)

Art.	Objetivo	Tecnologías	Resultado	Estado
[28]	Construir una aplicación que reduzca las limitaciones de la edición de diagramas LD para ampliar el alcance de edición y desarrollar aplicaciones orientadas a objetos.	C++	Se obtuvo un software que eliminó la restricción de dispositivos PLC por cada diagrama LD.	Terminado
[29]	Descripción una aplicación que obtenga US durante el ciclo de vida de las metodologías ágiles y se adapten a los cambios sin afectar el desarrollo del software.	SCRUM UML	Se desarrolló un diagrama de relación y una herramienta CASE que coleccionaron las US centrándose en las preguntas principales propias del análisis de requerimientos.	Terminado

2.3.Propuesta de solución

En esta sección se presenta la propuesta de solución para la elección de las tecnologías de información.

2.3.1. Propuesta de solución

Tomando en cuenta la disponibilidad de los recursos tecnológicos para el desarrollo de la herramienta, así como el grado y curva de aprendizaje para la interacción entre el usuario y el cliente del visualizador Web, se determinó la utilización de PHP, OOHDM y PHPStorm como tecnologías para el desarrollo de la herramienta de esta tesis.

2.3.2. Justificación de la solución propuesta

La solución propuesta se enfoca en desarrollar una aplicación Web que genere modelos del cliente y navegacional para aplicaciones Web.

Como lenguaje de programación se eligió a PHP gracias a la implementación de extensiones para la gestión y control de los archivos XMI, y al soporte para las comunicaciones asíncronas que envían al cliente un formato en estilo JSON.

Como metodología se eligió OOHDM dado que su implementación incluye el modelo navegacional que se tiene para el diseño de la herramienta que se enfocará en la interacción completa entre el usuario y el conjunto de componentes que la albergan.

Finalmente, como IDE se eligió a PHPStorm debido a la revisión en tiempo real de la sintaxis y a su especialización e inclusión de bibliotecas para JavaScript y PHP.

Finalmente, para la gestión de los datos se utilizarán archivos XMI que guardarán la configuración de los nodos tales como posición, relación con otros nodos, dimensiones, nombre, entre otros. Además, se utilizarán directorios para obtener su contenido y ofrecer independencia entre cada proyecto y organizarlo para una mejor presentación.

Capítulo 3. Aplicación de la metodología

En el presente capítulo se describe el diseño del sistema de la herramienta conforme a la metodología OOHDM, de manera que a través de ellas se llegó a la arquitectura propuesta, a la descripción de cada capa y a los componentes que se presentan en este trabajo de tesis. Además, se presentan los algoritmos que se utilizaron para el desarrollo de la herramienta.

En la metodología OOHDM con respecto en la especificación de requerimientos, se utilizaron diagramas de UML en lugar de UIDs (*User Interaction Diagrams*, Diagramas de interacción de usuario) dado que en UML, la representación de las acciones es más abstracta y se complementan estos casos con sus especificaciones. En la generación de interfaces abstractas, la utilización de maquetas en lugar de las ADVs (*Abstract Data Views,* vistas abstractas de datos) funge como representación general de la interfaz de la herramienta dado que, en las siguientes páginas, se explica detalladamente cada parte del programa.

3.1.Aplicación de las fases de OOHDM para el desarrollo de la herramienta

Para dar solución al problema identificado en este trabajo de tesis, se describe la aplicación de las fases de la metodología OOHDM que consiste en el análisis de requerimientos, diseño conceptual, diseño navegacional, diseño de la interfaz y la implementación. Con base en estas etapas, se generó la herramienta para la generación de diagramas del modelo del cliente y navegacional basada en artefacto con grafos y teoría de conjuntos.

3.1.1. Análisis de requerimientos

El análisis de requerimientos es el primer paso de la metodología OOHDM y en ella se obtienen los requerimientos de la aplicación tales como los actores y sus tareas a través del uso de UML que incluyen los diagramas de casos de uso y diagramas de interacción de usuario.

En conjunto, estos diagramas ofrecen una perspectiva visual de las interacciones que se realizan y validar la aplicación a través de los requerimientos antes de continuar con los pasos posteriores.

Requerimientos funcionales

En la herramienta se identifican los requisitos funcionales y a un único actor que lleva por nombre usuario. La Figura 3.1 muestra los casos de uso de la herramienta:

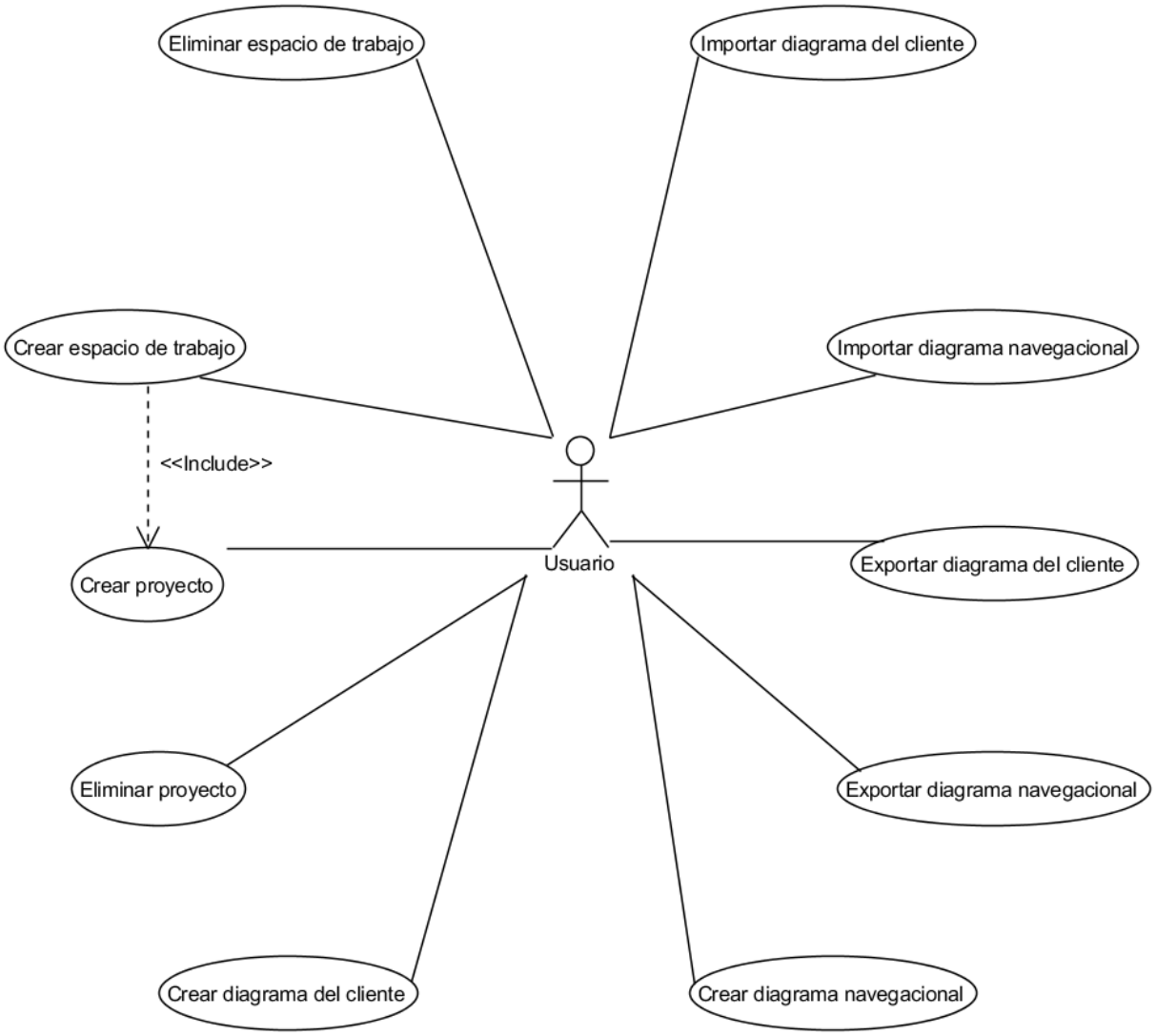


Figura 3.1 Diagrama de casos de uso.

A continuación, se muestran los escenarios para cada caso de uso. En cada escenario se realiza la especificación del mismo, así como el diagrama de interacción de usuario (o diagrama de secuencia) que corresponde a los detalles de la fase de análisis de requerimientos de la metodología OOHDM.

1. Crear espacio de trabajo (CU-001). La Tabla 3.1 muestra su especificación.

Tabla 3.1 Especificación del caso de uso crear espacio de trabajo

CU-001	**Crear espacio de trabajo**	
Actor	Usuario.	
Descripción	El usuario crea un espacio de trabajo estableciendo un nombre con el formato correcto. Solo se permiten caracteres en la primera letra, y caracteres y números a partir de la segunda letra. No se permite otro tipo de caracteres como espacio, ni símbolos especiales.	
Pre-condición	Ninguna.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario abre la herramienta.
	2	El usuario escribe un nombre de un campo de texto.
	3	El sistema crea una carpeta con el nombre del espacio de trabajo dentro de la carpeta principal.
	4	El sistema indica que la creación se realizó.
Post-condición	El espacio de trabajo se ha creado.	
Excepciones	**Paso**	**Acción**
	3	Si el formato del espacio de trabajo es incorrecto, el sistema vuelve al paso 2.
	3	Si el nombre del espacio de trabajo ya existe en la herramienta, el sistema vuelve al paso 2.
Importancia	Alta	

La Figura 3.2 muestra el diagrama de interacción de este caso de uso:

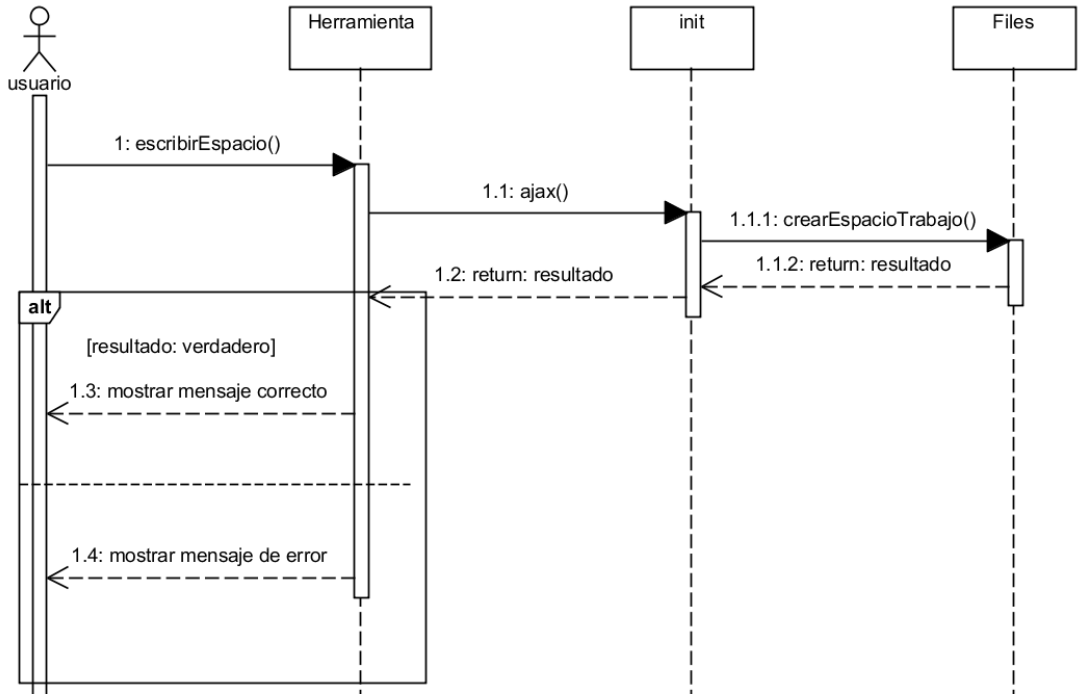


Figura 3.2 Diagrama de casos de uso.

1. Crear proyecto (CU-002). En la Tabla 3.2 se muestra la especificación del caso de uso.

Tabla 3.2 Especificación del caso de uso crear proyecto

CU-002	**Crear proyecto**	
Actor	Usuario	
Descripción	El usuario crea un proyecto estableciendo un nombre con el formato correcto. Solo se permiten caracteres en la primera letra, y caracteres y números a partir de la segunda letra. No se permite otro tipo de caracteres como espacio, ni símbolos especiales.	
Pre-condición	El usuario selecciona y abre un espacio de trabajo.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario abre la interfaz de proyecto.
	2	El usuario escribe un nombre.
	3	El sistema crea una carpeta con el nombre del proyecto dentro de la carpeta del espacio de trabajo activo.

Tabla 3.2 Especificación del caso de uso crear proyecto

CU-002	**Crear proyecto**	
	4	El sistema indica que la creación se realizó.
Post-condición	El proyecto se ha creado.	
Excepciones	**Paso**	**Acción**
	3	Si el formato del proyecto es incorrecto, el sistema vuelve al paso 2.
	3	Si el nombre del proyecto ya existe en la herramienta, el sistema vuelve al paso 2.
Importancia	Alta	

La Figura 3.3 muestra el diagrama de interacción de este caso de uso:

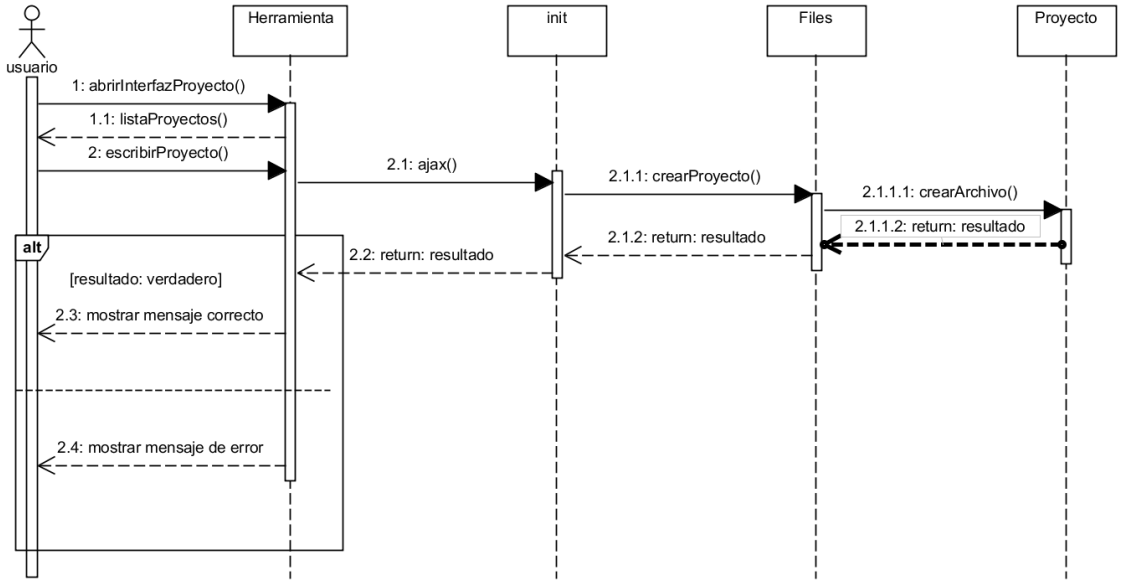


Figura 3.3 Diagrama de casos de uso.

2. Eliminar proyecto (CU-003). La Tabla 3.3 muestra la especificación del caso de uso.

Tabla 3.3 Especificación del caso de uso eliminar proyecto

CU-003	**Eliminar proyecto**	
Actor	Usuario	
Descripción	El usuario elimina un proyecto de la lista de proyectos disponibles dentro del espacio de trabajo.	
Pre-condición	El usuario selecciona y abre un espacio de trabajo.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario selecciona un proyecto de la lista de proyectos del espacio de trabajo activo.
	2	El usuario solicita al sistema la eliminación del proyecto.
	3	El sistema confirma la eliminación del proyecto.
	4	El sistema elimina el proyecto.
Post-condición	El proyecto se elimina del espacio de trabajo activo.	
Excepciones	**Paso**	**Acción**
	3	El usuario cancela la confirmación de eliminación.
Importancia	baja	

La Figura 3.4 muestra el diagrama de interacción de este caso de uso:

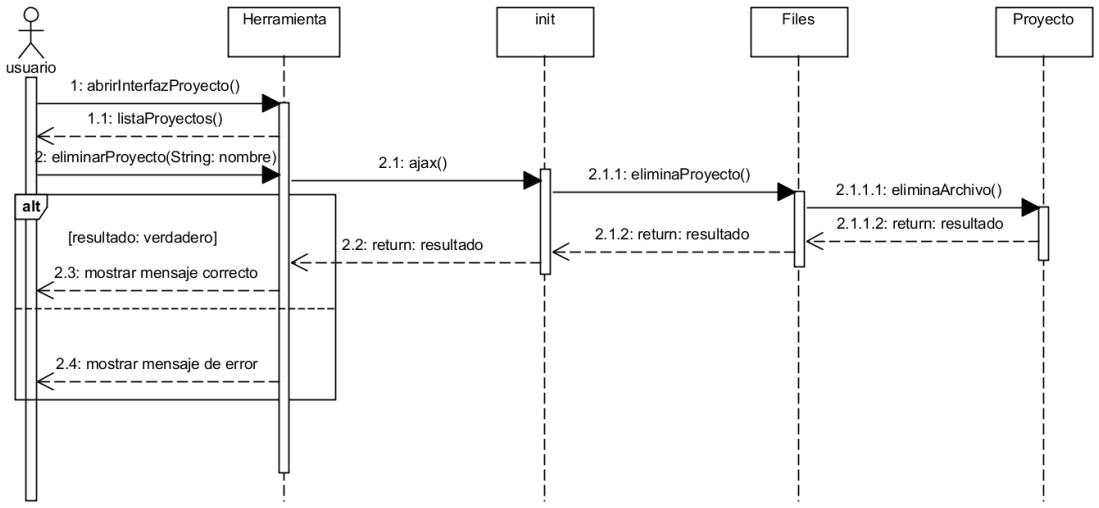


Figura 3.4 Diagrama de casos de uso.

3. Eliminar espacio de trabajo (CU-004). En la Tabla 3.4 se muestra su especificación.

Tabla 3.4 Especificación del caso de uso eliminar espacio de trabajo

CU-004	**Eliminar espacio de trabajo**	
Actor	Usuario	
Descripción	El usuario elimina un espacio de trabajo. Con ello, se eliminan también los proyectos asociados a dicho espacio de trabajo.	
Pre-condición	El usuario selecciona y abre un espacio de trabajo.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario selecciona un espacio de trabajo de la lista de espacios de trabajo.
	2	El usuario solicita la eliminación del espacio de trabajo.
	3	El sistema confirma la eliminación.
Post-condición	El espacio de trabajo se elimina de la herramienta.	
Excepciones	Ninguna.	
Importancia	Alta	

La Figura 3.5 muestra el diagrama de interacción de este caso de uso:

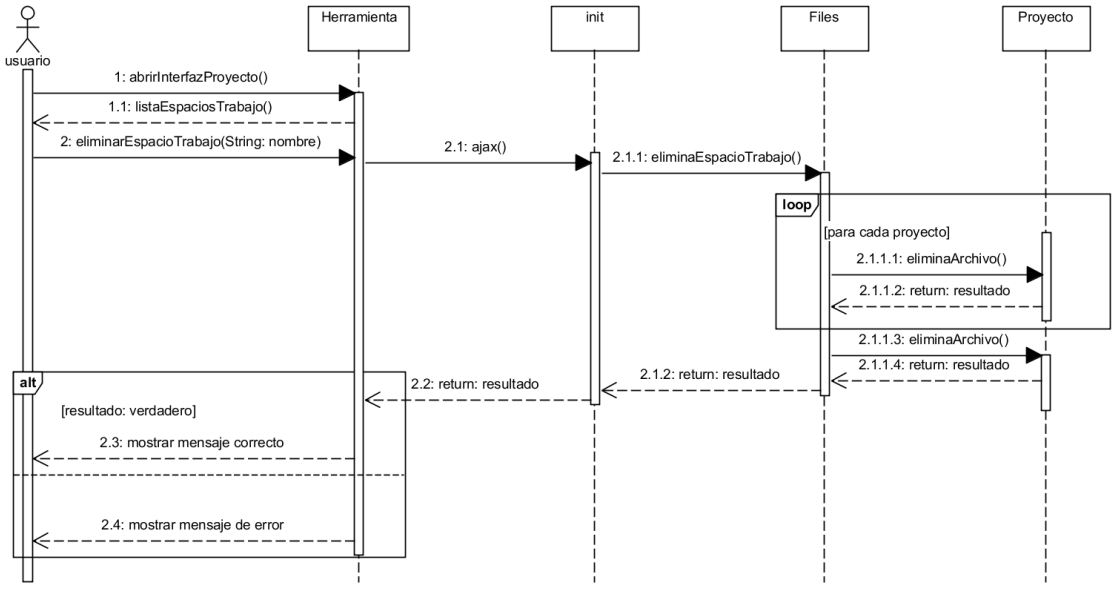


Figura 3.5 Diagrama de casos de uso.

4. Crear diagrama del cliente (CU-005). La Tabla 3.5 muestra su especificación.

Tabla 3.5 Especificación del caso de uso crear diagrama del cliente

CU-005	**Crear diagrama del cliente**	
Actor	Usuario	
Descripción	El usuario crea el diagrama del cliente.	
Pre-condición	El usuario selecciona y abre un espacio de trabajo. Posteriormente selecciona y abre un proyecto.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema la creación del diagrama del cliente.
	2	El sistema crea el diagrama del cliente.
Post-condición	El diagrama del cliente se ha creado	
Excepciones	Ninguna.	
Importancia	Alta	

La Figura 3.6 muestra el diagrama de interacción de este caso de uso:

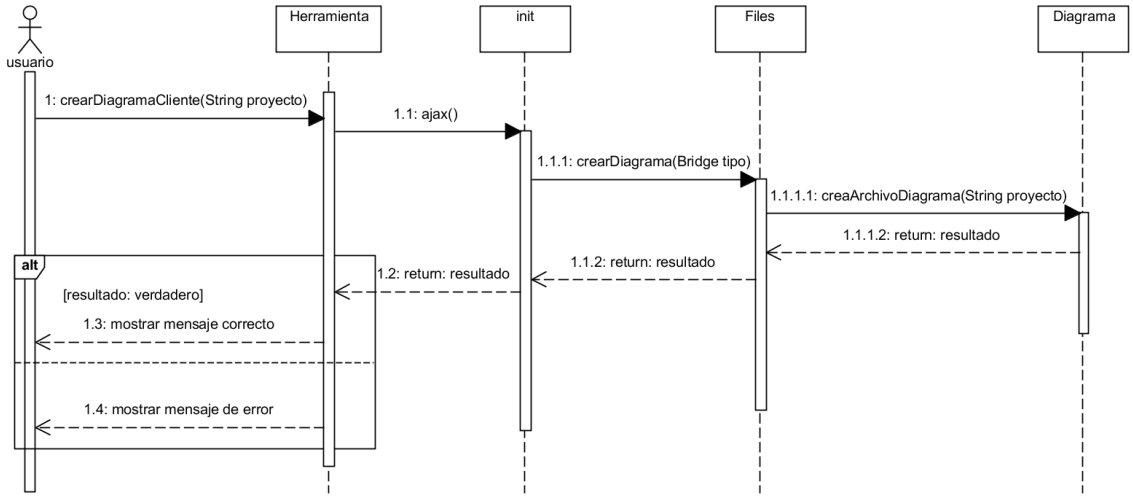


Figura 3.6 Diagrama de casos de uso.

5. Exportar diagrama del cliente (CU-006) En la Tabla 3.6 se muestra su especificación.

Tabla 3.6 Especificación del caso de uso exportar diagrama del cliente

CU-006	**Exportar diagrama del cliente**	
Actor	Usuario	
Descripción	El usuario exporta el diagrama resultante eligiendo la configuración para su exportación en JPG o XMI.	
Pre-condición	El usuario selecciona y abre un espacio de trabajo. Posteriormente selecciona y abre un proyecto. El usuario gestiona el diagrama activo.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita la exportación del diagrama activo.
	2	El sistema solicita las alternativas de exportación para el diagrama activo las cuales son: exportar el diagrama en formato JPG y el modelo en XMI.
	3	El usuario selecciona las alternativas se exportación adecuadas.
	4	El sistema con base en las alternativas seleccionadas, genera un archivo en formato ZIP con el contenido del diagrama.
Post-condición	El diagrama del cliente se ha exportado.	
Excepciones	**Paso**	**Acción**
	2	El sistema detecta que no hay contenido para exportar en el diagrama del cliente. Regresa al paso 1.
	4	El usuario no selecciona ninguna alternativa de exportación para el diagrama del cliente. Regresa al paso 3.
Importancia	Alta	

La Figura 3.7 muestra el diagrama de interacción de este caso de uso:

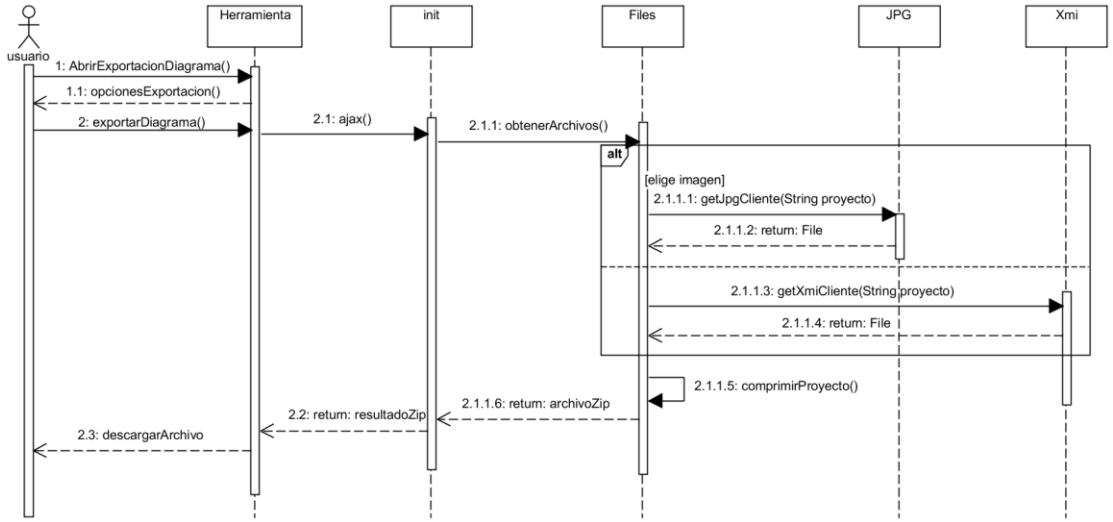


Figura 3.7 Diagrama de casos de uso.

6. Importar diagrama del cliente (CU-007). La Tabla 3.7 muestra su especificación.

Tabla 3.7 Especificación del caso de uso importar diagrama del cliente

CU-007	**Importar diagrama del cliente**	
Actor	Usuario	
Descripción	El usuario importa un archivo XMI válido para el diagrama del cliente.	
Pre-condición	El usuario selecciona y abre un espacio de trabajo. Posteriormente selecciona y abre un proyecto.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario abre la interfaz de importación del diagrama del cliente.
	2	El sistema solicita el archivo XMI.
	3	El usuario adjunta y carga el archivo XMI.
	4	El sistema valida e importa el archivo XMI.
Post-condición	El diagrama del cliente se ha importado.	
Excepciones	**Paso**	**Acción**

Tabla 3.7 Especificación del caso de uso importar diagrama del cliente

CU-007	**Importar diagrama del cliente**	
	3	El usuario cancela la importación del archivo XMI.
	4	El sistema verifica que el archivo XMI no es archivo XMI válido para el sistema. Regresa al paso 2.
	4	El sistema verifica que el archivo XMI es válido, pero posee errores en su sintaxis. Regresa al paso 2.
	4	El sistema verifica que el archivo XMI es válido, pero posee errores en la congruencia de los elementos. Regresa al paso 2.
Importancia	Alta	

La Figura 3.8 muestra el diagrama de interacción de este caso de uso:

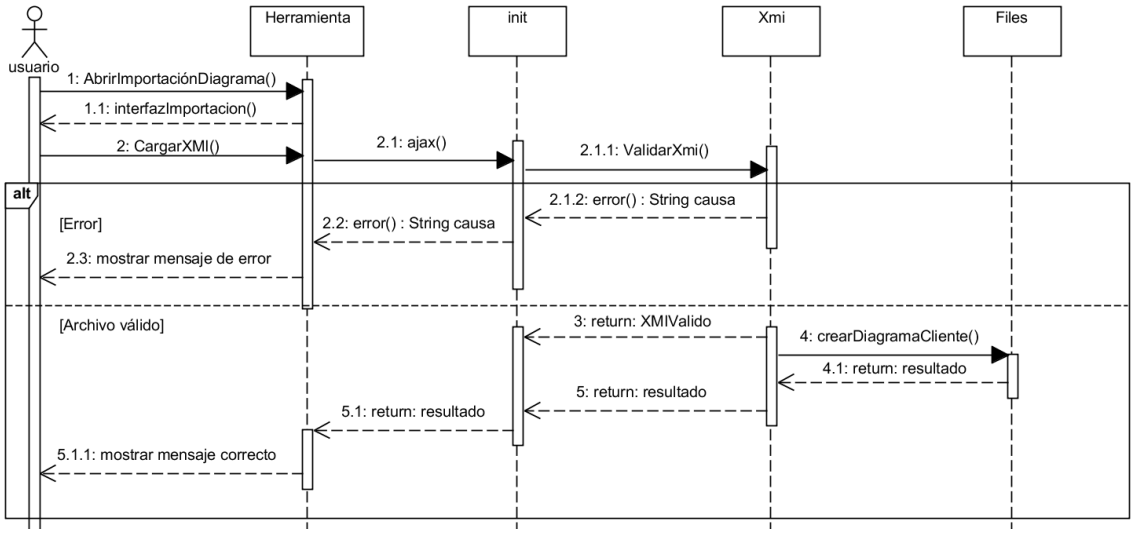


Figura 3.8 Diagrama de casos de uso.

7. Crear diagrama navegacional (CU-008). En la Tabla 3.8 se muestra la especificación del caso de uso.

Tabla 3.8 Especificación del caso de uso exportar diagrama del cliente

CU-008	**Crear diagrama navegacional**	
Actor	Usuario	
Descripción	El usuario crea el diagrama navegacional.	
Pre-condición	El usuario selecciona y abre un espacio de trabajo. Posteriormente selecciona y abre un proyecto.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema la creación del diagrama del cliente.
	2	El sistema crea el diagrama del cliente.
Post-condición	El diagrama navegacional se ha creado	
Excepciones	Ninguna.	
Importancia	Alta	

La Figura 3.9 muestra el diagrama de interacción de este caso de uso:

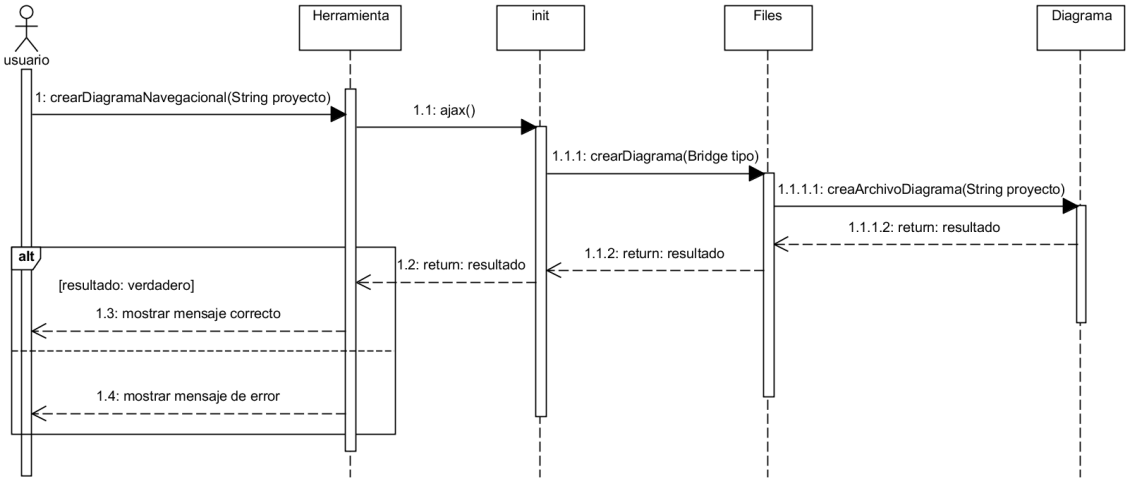


Figura 3.9 Diagrama de casos de uso.

8. Exportar diagrama navegacional. La Tabla 3.9 muestra la especificación del caso de uso.

Tabla 3.9 Especificación del caso de uso exportar diagrama del cliente

CU-009	**Exportar diagrama navegacional**	
Actor	Usuario	
Descripción	El usuario exporta el diagrama resultante eligiendo la configuración para su exportación en JPG o XMI.	
Pre-condición	El usuario selecciona y abre un espacio de trabajo. Posteriormente selecciona y abre un proyecto. El usuario gestiona el diagrama activo.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita la exportación del diagrama activo.
	2	El sistema solicita las alternativas de exportación para el diagrama activo las cuales son: exportar el diagrama en formato JPG y el modelo en XMI.
	3	El usuario selecciona las alternativas se exportación adecuadas.
	4	El sistema con base en las alternativas seleccionadas, genera un archivo en formato ZIP con el contenido del diagrama.
Post-condición	El diagrama navegacional se ha exportado.	
Excepciones	**Paso**	**Acción**
	2	El sistema detecta que no hay contenido para exportar en el diagrama navegacional. Regresa al paso 1.
	4	El usuario no selecciona ninguna alternativa de exportación para el diagrama navegacional. Regresa al paso 3.
Importancia	Alta	

La Figura 3.10 muestra el diagrama de interacción de este caso de uso:

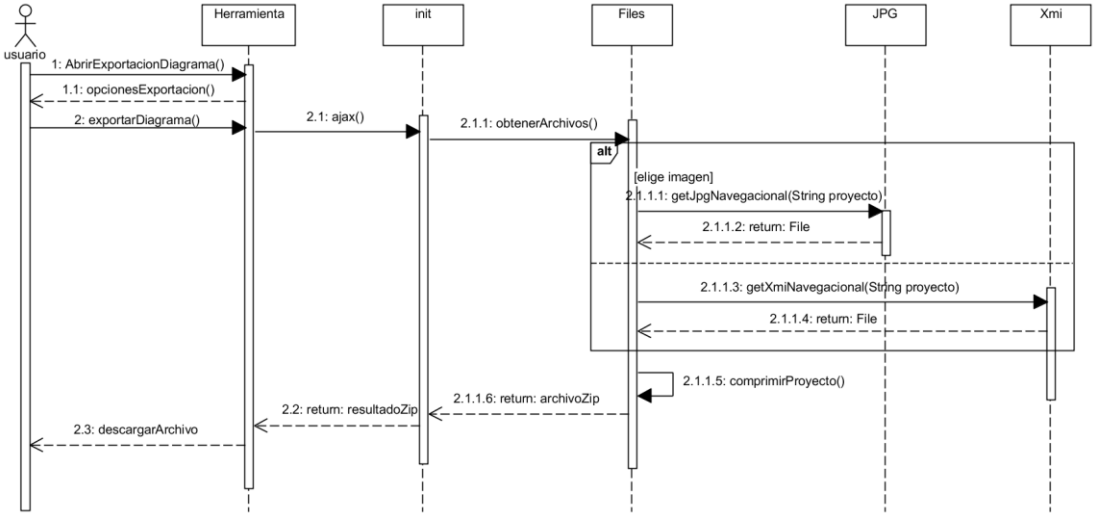


Figura 3.10 Diagrama de casos de uso.

9. Importar diagrama navegacional (CU-010). En la Tabla 3.10 se muestra la especificación del caso de uso.

Tabla 3.10 Especificación del caso de uso exportar diagrama del cliente

CU-010	**Importar diagrama navegacional**	
Actor	Usuario	
Descripción	El usuario importa un archivo XMI válido para el diagrama del cliente.	
Pre-condición	El usuario selecciona y abre un espacio de trabajo. Posteriormente selecciona y abre un proyecto.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario abre la interfaz de importación del diagrama navegacional.
	2	El sistema solicita el archivo XMI.
	3	El usuario adjunta y carga el archivo XMI.
	4	El sistema valida e importa el archivo XMI.
Post-condición	El diagrama navegacional se ha importado.	
Excepciones	**Paso**	**Acción**

Tabla 3.10 Especificación del caso de uso exportar diagrama del cliente

CU-010	**Importar diagrama navegacional**	
	3	El usuario cancela la importación del archivo XMI.
	4	El sistema verifica que el archivo XMI no es archivo XMI válido para el sistema. Regresa al paso 2.
	4	El sistema verifica que el archivo XMI es válido, pero posee errores en su sintaxis.
	4	El sistema verifica que el archivo XMI es válido, pero posee errores en la congruencia de los elementos.
Importancia	Alta	

La Figura 3.11 muestra el diagrama de interacción de este caso de uso:

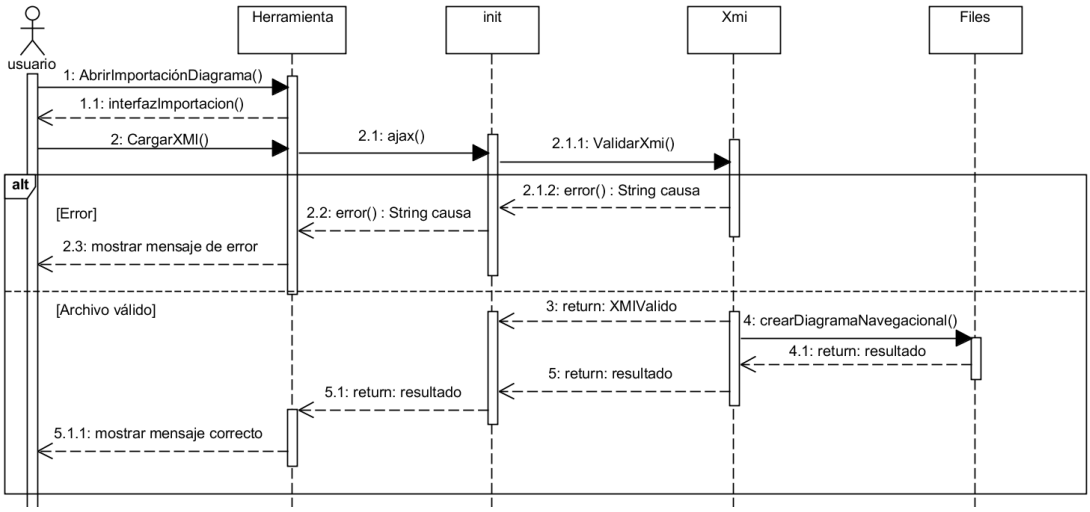


Figura 3.11 Diagrama de casos de uso.

Requerimientos no funcionales

A continuación, se listan los requerimientos no funcionales:

1. La herramienta se utiliza en un visualizador Web.

2. La herramienta se instala en un servidor apache compatible con PHP versión 6.0.
3. La herramienta necesita al menos una resolución de 1024 x 768.

3.1.2. Diseño conceptual

En este paso se construye un modelo conceptual que se centra en el dominio de la aplicación y no en quienes la utilizarán. Para ello de desarrolla un diagrama de clases solo con los atributos.

Diagrama de clases

En la Figura 3.12 se muestran las clases necesarias para representar el modelo de la aplicación. Las relaciones expresan lo siguiente: La clase `Gral` se relaciona con la mayoría de las otras clases (excepto `XmlToJson`). Las clases `Diagrama`, `Proyecto`, `XmlToJson`, `Cliente` y `Navegacional` se relacionan con la clase `Xmi`. Las clases `Model`, `CliModel`, `NavModel`, `Cliente` y `Navegacional` representan al patrón de diseño Bridge. Se utilizó este patrón dada la similitud de representación para la generación de los archivos XMI. La clase `Jpg` garantiza la conversión del formato base64 a formato de imagen para su almacenamiento en los directorios del servidor.

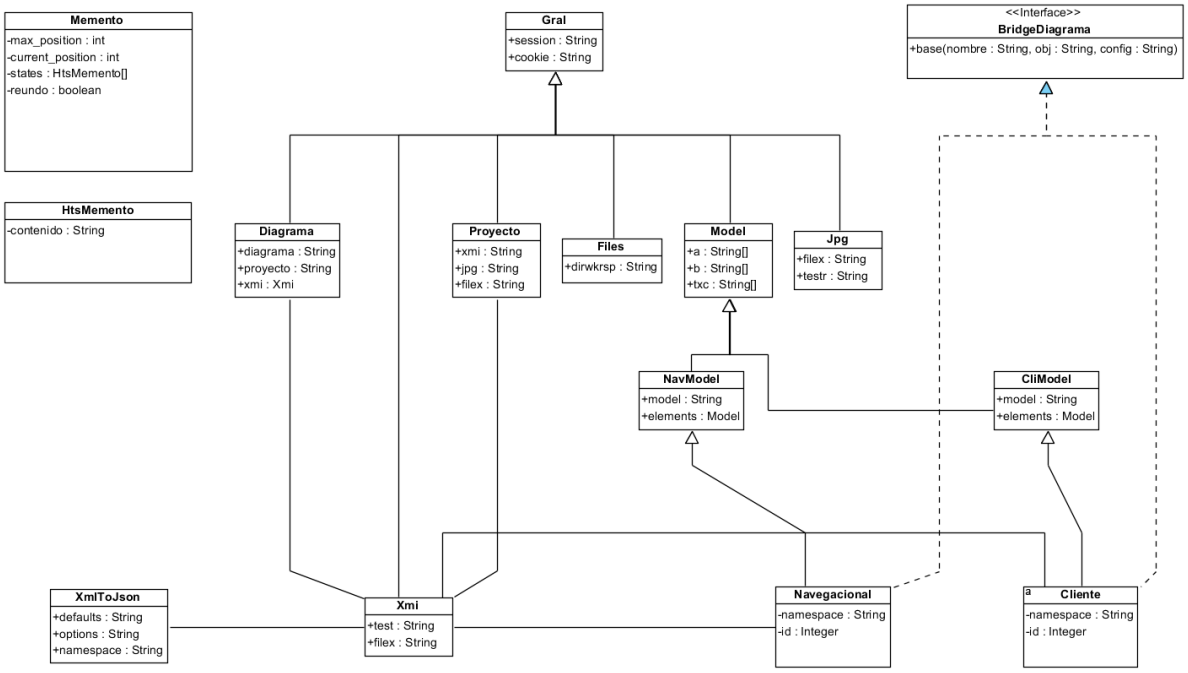


Figura 3.12 Diagrama de clases.

3.1.3. Diseño navegacional

Con base en el diagrama de clases, se generan las vistas navegacionales en la que el autor y su relación con cada tarea se describe como un lugar de interacción y la aplicación Web. OOHDM define algunos nodos de hipermedia como nodos, enlaces o estructuras de acceso, así como el contexto de la navegación. Para ello de desarrolla un diagrama de clases extendido (que incluye los métodos). En la Figura 3.13 se muestra el diagrama de clases extendido.

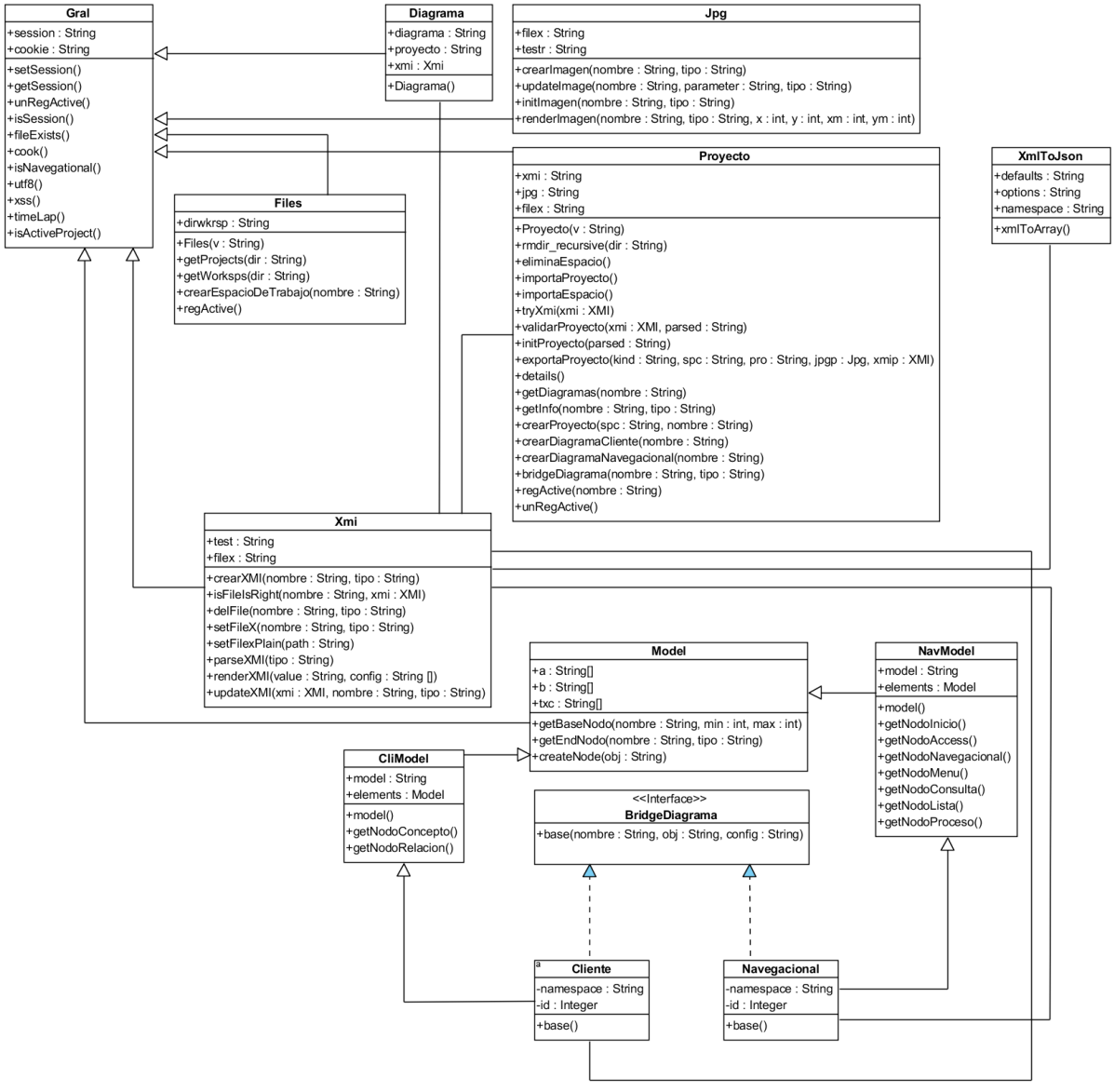


Figura 3.13 Diagrama de clases extendido.

A continuación, se definen los objetos navegacionales:

1. Inicio.
2. Selección de espacio de trabajo.
3. Creación de espacio de trabajo.
4. Selección de proyecto.
5. Creación de proyecto.
6. Generación del diagrama del cliente.
7. Creación del diagrama navegacional.
8. Lienzo.

Después se generan los contextos navegacionales como se indica a continuación:

1. Exportar diagrama.
2. Importar diagrama.
3. Configurar diagrama.
4. Ejemplos de ayuda.

Esquema navegacional

A continuación, de la Tabla 3.11 a la Tabla 3.22, se resumen las principales características de los objetos navegacionales.

Tabla 3.11 Inicio

Nombre	**Inicio**
Clase conceptual (CC)	*Gral*
Atributos	Ninguno
Descripción	Se encarga de mostrar la interfaz inicial de la herramienta donde se seleccionará un espacio de trabajo existente o la creación de un nuevo espacio de trabajo.
Enlaces	1. selección de espacio de trabajo. 2. Selección de proyecto. 3. Creación de proyecto.

Tabla 3.12 Selección de espacio de trabajo

Nombre	**Selección de espacio de trabajo**
Clase conceptual (CC)	*Gral*
Atributos	1. Espacio de trabajo seleccionado.
Descripción	Se encarga de mostrar un formulario para seleccionar el espacio de trabajo deseado.
Enlaces	1. Selección de proyecto. 2. Creación de proyecto.

Tabla 3.13 Creación de espacio de trabajo

Nombre	**Creación de espacio de trabajo**
Clase conceptual (CC)	*Gral, Files*
Atributos	1. Nombre del nuevo espacio de trabajo.
Descripción	Se encarga de mostrar un formulario donde se ingresa el nombre del nuevo espacio de trabajo.
Enlaces	1. Selección de proyecto.

Tabla 3.14 Selección de proyecto

Nombre	**Selección de proyecto**
Clase conceptual (CC)	*Gral, Proyecto*
Atributos	1. Nombre del proyecto.
Descripción	Se encarga de mostrar un formulario para la selección de un proyecto activo dentro del espacio de trabajo seleccionado.
Enlaces	1. Generación del diagrama del cliente. 2. Generación del diagrama navegacional.

Tabla 3.15 Creación de proyecto

Nombre	**Creación de proyecto**
Clase conceptual (CC)	*Gral, Proyecto*
Atributos	1. Nombre del nuevo proyecto.
Descripción	Se encarga de mostrar un formulario para ingresar el nombre del nuevo proyecto.
Enlaces	1. Generación del diagrama del cliente. 2. Generación del diagrama navegacional.

Tabla 3.16 Generación del diagrama del cliente

Nombre	**Generación del diagrama del cliente**
Clase conceptual (CC)	*Gral, Proyecto, Diagrama, Model, CliModel, Cliente, XMI*
Atributos	Ninguno
Descripción	Se encarga de generar el diagrama del cliente a través de la creación del archivo XMI asociado a dicho diagrama.
Enlaces	1. Lienzo.

Tabla 3.17 Generación del diagrama navegacional

Nombre	**Generación del diagrama navegacional**
Clase conceptual (CC)	*Gral, Proyecto, Diagrama, Model, NavModel, Navegacional, XMI*
Atributos	Ninguno
Descripción	Se encarga de generar el diagrama del cliente a través de la creación del archivo XMI asociado a dicho diagrama.
Enlaces	1. Lienzo

Tabla 3.18 Lienzo

Nombre	**Lienzo**
Clase conceptual (CC)	*Gral, Diagrama, Proyecto, Files, Model, Jpg CliModel, NavModel, Cliente, Navegacional, Xmi, XmiToJson, Mememto, HTSMemento.*
Atributos	Ninguno
Descripción	Se encarga de mostrar un formulario (lienzo) donde se genera el diagrama del cliente o navegacional.
Enlaces	1. Exportar diagrama

Tabla 3.19 Exportar diagrama

Nombre	**Exportar diagrama**
Clase conceptual (CC)	*Gral, Diagrama, proyecto, Files, Model, JPG, NavModel, CliModel, Cliente, Navegacional, XMI.*
Atributos	1. Campo de exportación del archivo JPG. 2. Campo de exportación del archivo XMI.
Descripción	Se encarga de mostrar un formulario para exportar el diagrama.
Enlaces	Ninguno

Tabla 3.20 Importar diagrama

Nombre	**Importar diagrama**
Clase conceptual (CC)	*Gral, Proyecto, Diagrama, Files, Model, CliModel, Cliente, NavModel, Navegacional, XMI, XmiToJson.*
Atributos	1. Diagrama bajo el modelo XMI.
Descripción	Se encarga de mostrar la interfaz inicial de la herramienta donde se seleccionará un espacio de trabajo existente o la creación de un nuevo espacio de trabajo.
Enlaces	Ninguno.

Tabla 3.21 Configurar diagrama

Nombre	**Configurar diagrama**
Clase conceptual (CC)	*Gral, Diagrama, Files, Model, CliModel, Cliente, NavModel, Navegacional, XMI. XmiToJson.*
Atributos	1. Autoguardado. 2. Activación de reglas. 3. Activación de cuadrículas. 4. Guardado en Google Drive.
Descripción	Se encarga de mostrar un formulario para la activación o desactivación de diversas opciones del diagrama.
Enlaces	1. Lienzo

Tabla 3.22 Ejemplos de ayuda

Nombre	**Ejemplos de ayuda**
Clase conceptual (CC)	*Gral, Files.*
Atributos	Ninguno
Descripción	Se encarga de mostrar una lista de diagramas de ayuda para la mejora en la comprensión del uso de los diagramas del cliente y navegacional.
Enlaces	1. Lienzo.

En la Figura 3.14 se aprecia el modelo de clases navegacionales, en el que, a través de la clase Inicio se acceden a los demás nodos.

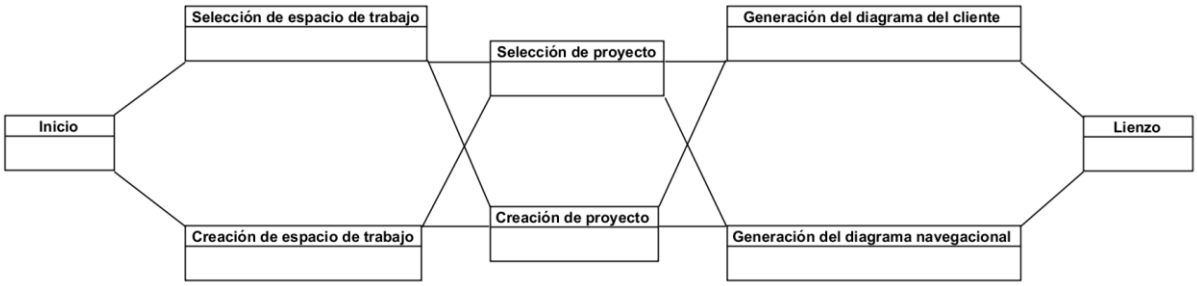


Figura 3.14 Modelo de clases navegacionales de la herramienta.

En la Figura 3.15 se muestra el esquema de contextos navegacionales de la herramienta.

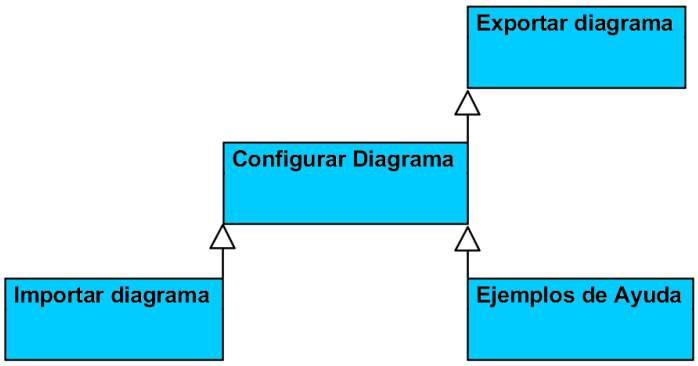


Figura 3.15 Esquema de contextos navegacionales.

Arquitectura del sistema

Para el desarrollo de esta herramienta, se utilizó la arquitectura bajo un enfoque basado en tres capas. Cada capa ofrece un nivel de abstracción que separa las responsabilidades en actividades pequeñas. En la Figura 3.16 se presenta el esquema de la arquitectura de la herramienta.

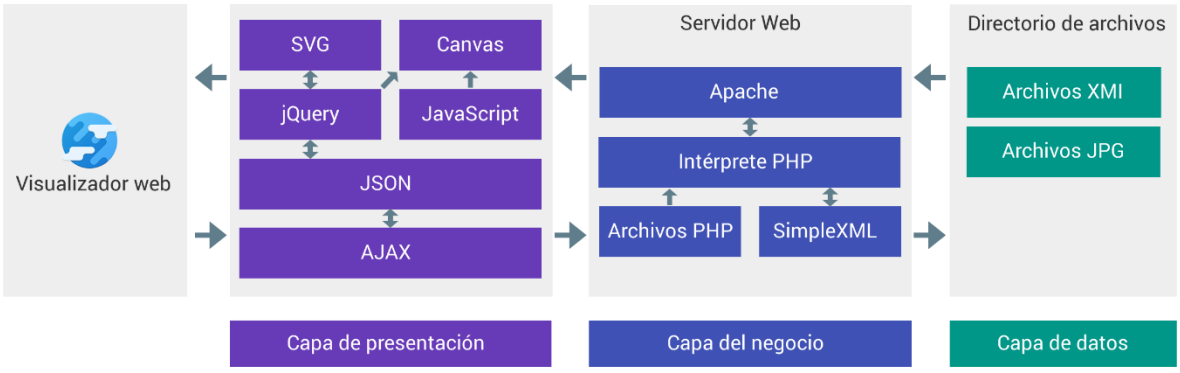


Figura 3.16 Arquitectura del sistema.

Capa de presentación: La capa de presentación muestra al usuario una interfaz gráfica para la gestión de las herramientas que generen los diagramas del modelo cliente y navegacional. Esto se logra a través de bibliotecas que se ejecutan del lado del cliente utilizando las API (*Application Programming Interface*, Interfaz de programación de aplicaciones) de jQuery, Canvas y SVG que, en conjunto con JavaScript, se encargan de la gestión de los diagramas antes

mencionados. AJAX (*Asynchronous JavaScript And XML*, JavaScript asíncrono y XML) se centra en la comunicación con la capa de negocio y retorna como resultado un elemento JSON (*JavaScript Object Notation*, Notación de objetos de JavaScript) que sirve como base para la construcción de toda la interfaz del usuario.

Capa del negocio: La capa del negocio se gestiona en un servidor Web y sirve como medio de entrada para acceder a los archivos PHP que se procesarán bajo un intérprete PHP en su versión 5.6. Los archivos PHP se comunican con la capa de datos para recuperar los archivos XMI y entregarlos en la capa de presentación en formato JSON. Además, se utilizará la extensión de PHP SimpleXML que, a través de un archivo XMI seleccionado convierta su contenido en un conjunto de arreglos de propósito general.

Capa de datos: La capa de datos contiene los directorios y archivos de los diagramas que se distribuyen en función del nombre del proyecto. Los archivos XMI contienen la información del meta-modelo y del modelo requeridos para su interpretación y gestión en la capa del negocio y de presentación. Finalmente, los archivos JPG contienen una vista previa del diagrama que se utilizará como recurso de propósito general.

3.1.4. Diseño de la interfaz

La interfaz del usuario se compone del diseño estructural y de comportamiento. Cada diseño ofrece un bosquejo del cómo se representará la información del modelo del dominio de la aplicación brindando un panorama que sirve como ruta a seguir cuando se llegue al paso de la implementación. El diseño estructural utiliza las ADV (*Abstract Data View*, Vistas de datos abstractas) y representa lo que el cliente desea como resultado interactivo en una aplicación Web.

Vista abstracta del nodo inicio

En la Figura 3.17 se muestra la vista abstracta del nodo inicio y representa la interfaz inicial de la herramienta. Cuenta con el nombre de la herramienta y un botón para crear un espacio de trabajo.

Vista abstracta del nodo de selección de espacios de trabajo

La Figura 3.18 muestra la vista abstracta del nodo de selección de espacios de trabajo. Además de contar con los elementos del punto 3.1.4.1, posee una lista de espacios de trabajo activos, así como un botón para abrir un espacio de trabajo seleccionado con base en dicha lista.

Vista abstracta del nodo de creación de espacios de trabajo

En la Figura 3.19 se muestra la vista abstracta del nodo de creación de espacio de trabajo. Cuenta con un campo para escribir el nombre del espacio de trabajo, un botón para crear un espacio de trabajo, así como un campo de texto para mostrar información en caso de error.

Vista abstracta del nodo de selección de proyecto

La Figura 3.20 muestra la vista abstracta del nodo de selección de proyecto. Cuenta con una lista de proyectos dentro del espacio de trabajo y muestra para cada proyecto su nombre, tamaño y última fecha de modificación.

Vista abstracta del nodo de creación de proyectos

En la Figura 3.21 se muestra la vista abstracta del nodo de creación de un proyecto. Cuenta con un campo para escribir el nombre del proyecto, un botón para crear un proyecto, así como un campo de texto para mostrar información en caso de error.

Vista abstracta del nodo de generación del diagrama del cliente

La Figura 3.22 muestra la vista abstracta del nodo de generación del diagrama del cliente. Cuenta con dos diagramas. Cuando el diagrama del cliente no se creó, posee un botón para crearlo. En caso contrario se muestra la imagen con la última modificación del mismo.

Vista abstracta del nodo de generación del diagrama navegacional

En la Figura 3.23 se muestra la vista abstracta del nodo de generación del diagrama navegacional. Cuenta con dos diagramas. Cuando el diagrama navegacional no se creó, posee un botón para crearlo. En caso contrario se muestra la imagen con la última modificación del mismo.

Vista abstracta del nodo del lienzo

La Figura 3.24 muestra la vista abstracta del nodo del lienzo. Cuenta con un menú, el campo para la última modificación y tamaño del lienzo, elementos del zoom, la lista de nodos en función del tipo del diagrama, así como el lienzo correspondiente.

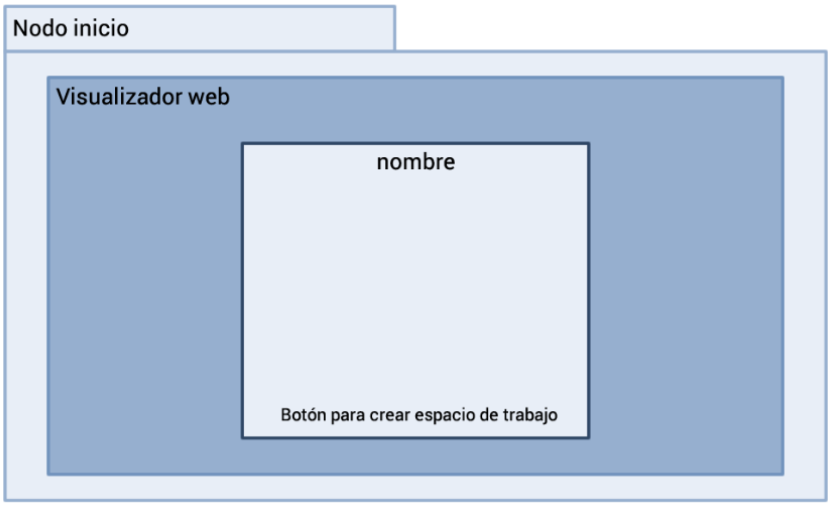


Figura 3.17 Vista abstracta del nodo inicio.

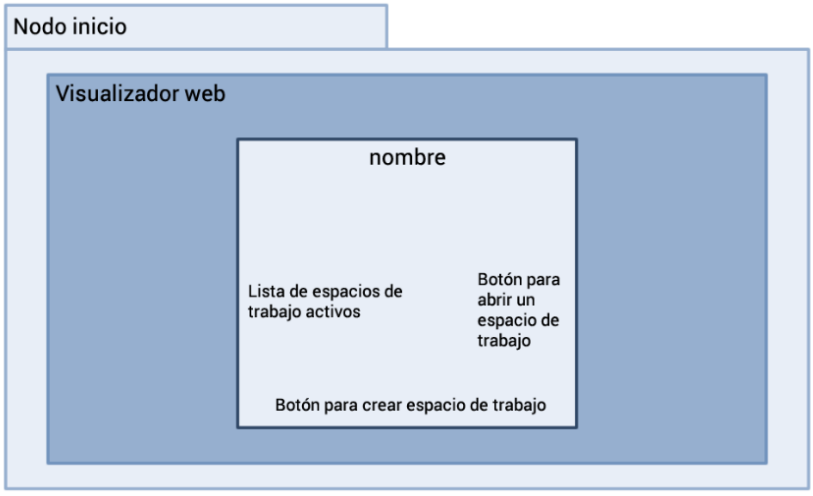


Figura 3.18 Vista abstracta del nodo selección de espacio de trabajo.

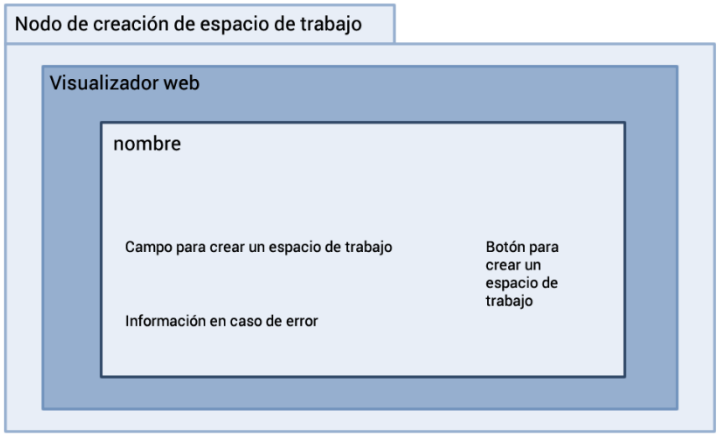


Figura 3.19 Vista abstracta del nodo de creación de espacio de trabajo.

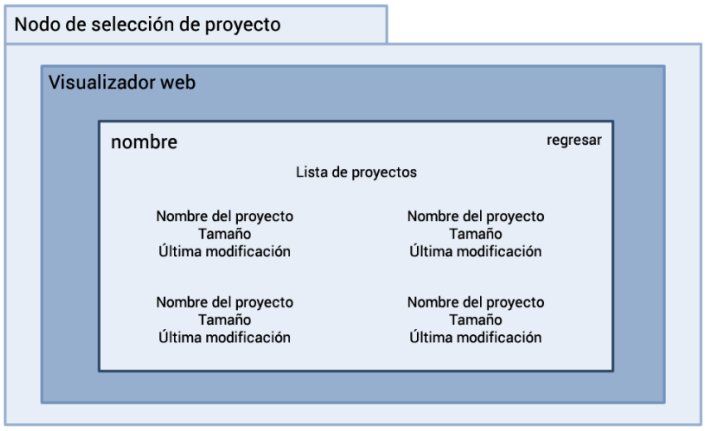


Figura 3.20 Vista abstracta para el nodo de selección de proyecto.

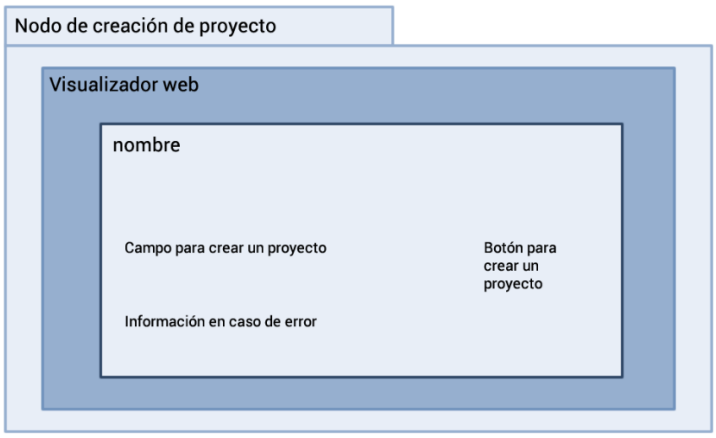


Figura 3.21 Vista abstracta para el nodo de creación de proyecto.

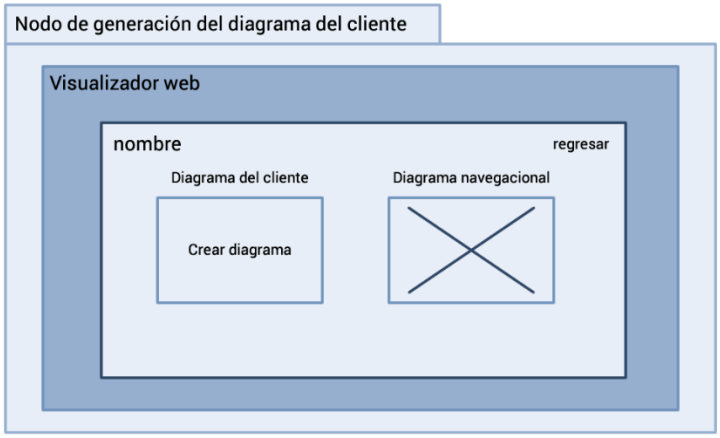


Figura 3.22 Vista abstracta para el nodo de generación del diagrama del cliente.

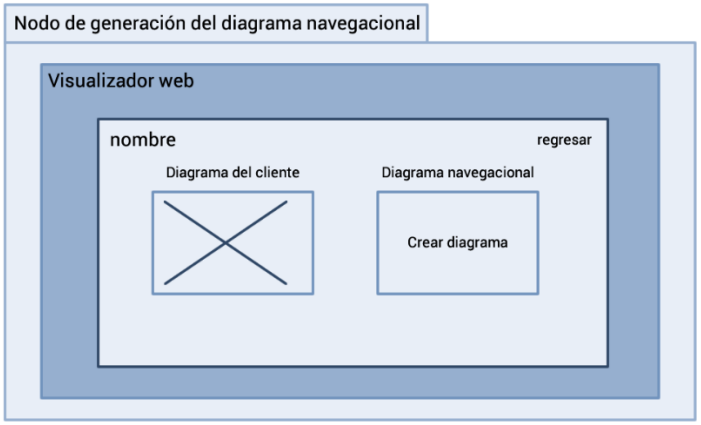


Figura 3.23 Vista abstracta para el nodo de generación del diagrama navegacional.

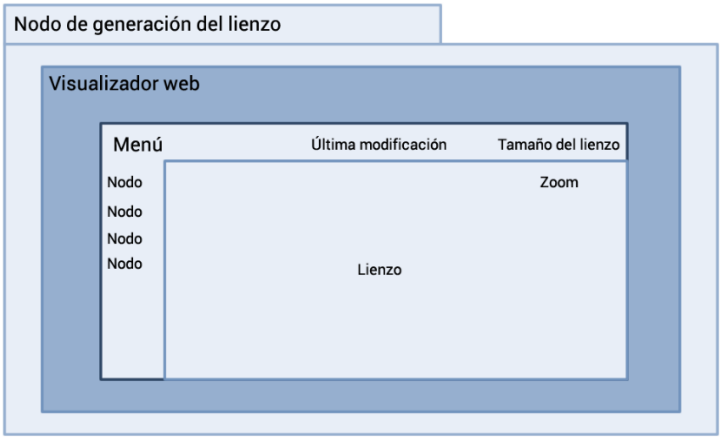


Figura 3.24 Vista abstracta para el nodo del lienzo.

Diagrama de paquetes de la herramienta

Con base en las fases anteriores de la metodología OOHDM, en la Figura 3.25 se muestra el diagrama de paquetes de la herramienta. Esta distribución se conforma de los directorios css, js, php. No se incluyeron los directorios de imágenes ni del espacio de trabajo dado que el número de archivos es extenso.

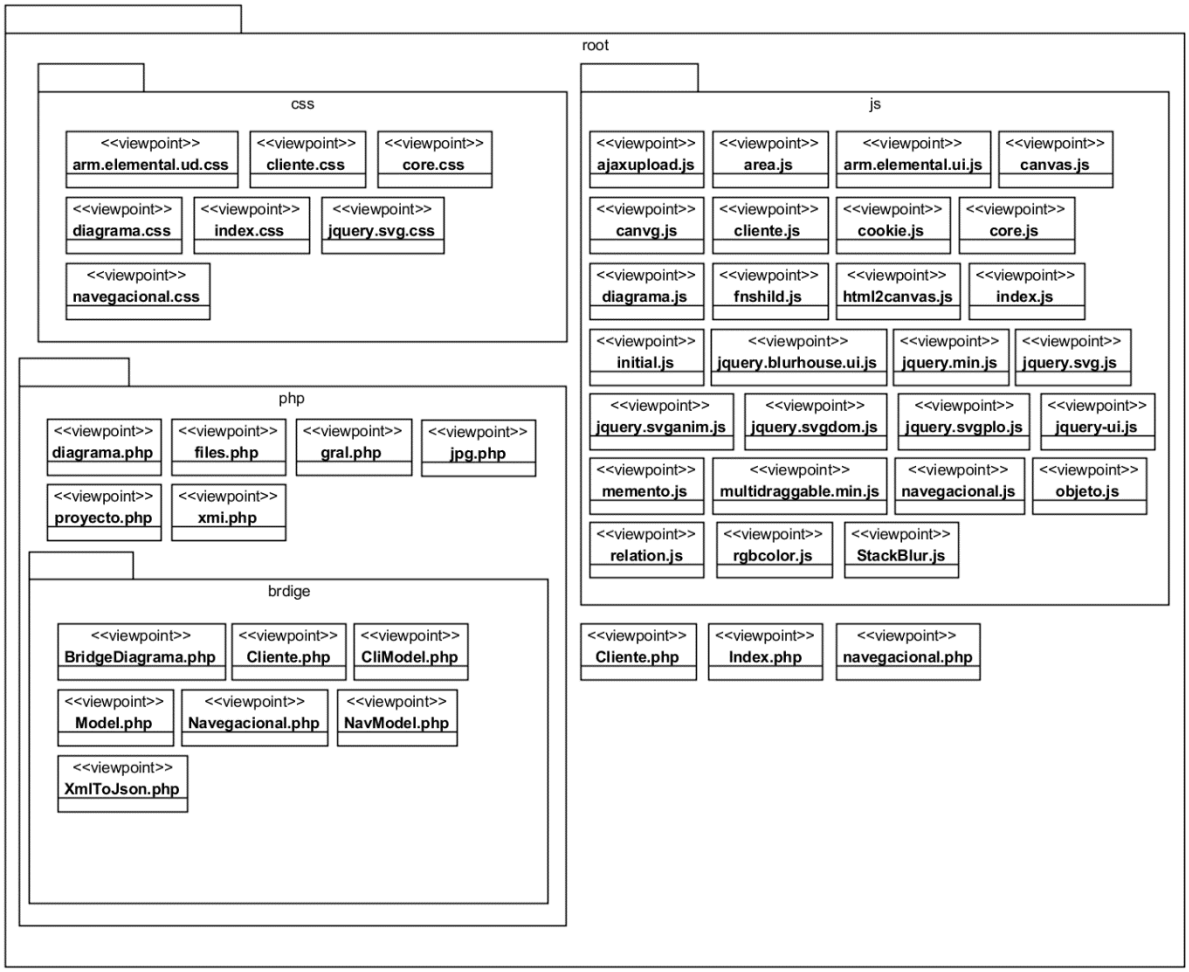


Figura 3.25 Diagrama de paquetes de la herramienta.

Características de diseño

El diseño se basa en la utilización de iconos para la representación de los nodos y los colores blanco y morado como tema principal del sistema. En el lienzo se utilizan los colores negro, blanco y azul. A continuación, se muestra la representación hexadecimal de cada color:

1. Blanco: *#FFFFFF*.
2. Morado: *#6146A7*.
3. Azul: *#0099FF*.
4. Negro: *#000000*.
5. Gris oscuro. *#666666*.

Consideraciones de diseño gráfico.

La fuente que se usa es *Roboto* (propiedad de *Google* bajo licencia de uso y distribución libre).

3.1.5. Implementación.

La fase de implementación consiste en codificar lo que se realizó y utilizar todo lo que se describió en etapas anteriores para completar el proceso de desarrollo de la aplicación Web. Con base en la descripción de tecnologías a utilizar y que se describieron en el capítulo 1, se utilizó como IDE PHPStorm, en el cual se aplicó la arquitectura basada en tres capas.

Generación del meta-modelo de los diagramas del cliente y navegacional

El meta-modelo es la descripción de un modelo que garantiza la aplicación de esquemas de organización, reglas y elementos que rigen la estructura de un modelo. La estructura de un modelo se compone por las siguientes partes:

1. Atributos: Cada atributo de un meta-modelo posee elementos que los identifican.
2. Tipo de dato: Los atributos que se definen en las operaciones simples o complejas cuentan con un tipo de dato (int, boolean, String, entre otros) que permite al verificador del modelo garantizar que los tipos de datos se cumplan y mantenerlo consistente.
3. Tipo simple: Estos tipos representan una acción en el modelo. Por ejemplo: el nombre de un autor que gestiona el modelo. A continuación, se muestra el meta-modelo en XMI que representa un tipo simple:

   Código XMI

   ```
   <xsd:element name="autor" type="xsd:String"></autor>
   ```

4. Tipo complejo: Estos tipos representan un conjunto de acciones en el modelo. Por ejemplo: La definición de propiedades de un modelo como son *lastEdit*, *grid*, *autosave*, *rule* y *fileopen*. A continuación, se muestra el meta-modelo en XMI que representa al tipo complejo propiedades:

   Código XMI

```
<xmi:Properties>
       <xsd:element name="lastEdit" type="xsd:integer"/>
       <xsd:element name="grid" type="xsd:boolean"/>
       <xsd:element name="autosave" type="xsd:boolean"/>
       <xsd:element name="rule" type="xsd:boolean"/>
       <xsd:element name="fileopen" type="xsd:boolean"/>
</xmi:Properties>
```

Para la implementación del archivo XMI se utilizó la versión de XML 1.0 con codificación UTF-8 y se utilizó la versión 2.1 de XMI con dos espacios de trabajo:

1. Xmlns:sodra: Este espacio de trabajo aplica para la definición del meta-modelo de los nodos de los diagramas del cliente y navegacional y sus correspondientes relaciones.
2. Xmlns:xmi: Este espacio de trabajo pertenece al estándar del w3c para la definición de los tipos datos, atributos, elementos, elecciones así como la definición de los tipos simples y complejos.

A continuación, se muestra el código XMI con la definición de los espacios de trabajo y las versiones de trabajo:

Código XMI

```
<?xml version="1.0" encoding="utf-8"?>
<xmi:XMI xmi:version="2.1" xmlns:sodra="https://sodra.gov.uk/metadata"
xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
xmlns:xsd="http://www.w3c.org/2001/XMLSchema">
       <!-- Contenido -->
</xmi:XMI>
```

Para la definición del meta-modelo del cliente y navegacional, se definieron algunas características en común como son las propiedades del modelo (última modificación, aplicación de cuadrículas, guardado automático, reglas y archivo abierto). A continuación, se muestra el código XMI con la definición del meta-modelo de las propiedades del modelo:

Código XMI

```
<xmi:Properties exporter="Sodra">
```

```
        <xmi:Extension extender="Sodra">
                <xsd:element name="lastEdit" type="xsd:integer"/>
                <xsd:element name="grid" type="xsd:boolean"/>
                <xsd:element name="autosave" type="xsd:boolean"/>
                <xsd:element name="rule" type="xsd:boolean"/>
                <xsd:element name="fileopen" type="xsd:boolean"/>
        </xmi:Extension>
</xmi:Properties>
```

Con base en el código XMI, a continuación, se describen las funcionalidades de cada tipo simple dentro del tipo complejo propiedades:

1. **lastEdit**: Indica la última modificación del archivo.
2. **Grid**: muestra la cuadrícula en la herramienta.
3. **Autosave**: Activa el autoguardado del diagrama del cliente o navegacional en la herramienta.
4. **Rule**: Gestiona la visibilidad de las reglas vertical y horizontal de la herramienta.
5. **Fileopen**: Determina si el archivo se encuentra abierto con el propósito de evitar que se modifique el diagrama en dos pestañas de un mismo visualizador o de diferentes visualizadores.

Para la definición del meta-modelo de los nodos se consideró homogeneizar las diferencias entre el diagrama del cliente y navegacional, por lo que se generó un tipo complejo con 11 tipos simples. Los nombres nodo_cliente/nodo_navegacional se reemplazan con los nombres que se definen en el punto 3.3.1. A continuación se muestra el código XMI que corresponde a la representación del modelo de los nodos:

Código XMI

```
<sodraModel name="nodo_cliente/nodo_navegacional" id="nodo_cliente/nodo_navegacional">
    <xsd:complexType name="nodo_cliente/nodo_navegacional">
        <xsd:choice minOccurs="1" maxOccurs="1">
            <xsd:element name="idobj" type="xsd:integer" />
            <xsd:element name="x" type="xsd:integer" />
            <xsd:element name="y" type="xsd:integer" />
            <xsd:element name="isnodo" type="xsd:integer" />
            <xsd:element name="extender" type="xsd:integer" />
            <xsd:element name="action" type="xsd:string" />
            <xsd:element name="form" type="xsd:string" />
            <xsd:element name="in" type="xsd:string" />
            <xsd:element name="out" type="xsd:string" />
            <xsd:element name="name" type="xsd:string" />
```

```
            <xsd:element name="locked" type="xsd:boolean" />
        </xsd:choice>
        <xsd:attribute name="idobj" type="xsd:integer" />
        <xsd:attribute name="x" type="xsd:integer" />
        <xsd:attribute name="y" type="xsd:integer" />
        <xsd:attribute name="isnodo" type="xsd:integer" />
        <xsd:attribute name="extender" type="xsd:integer" />
        <xsd:attribute name="action" type="xsd:string" />
        <xsd:attribute name="form" type="xsd:string" />
        <xsd:attribute name="in" type="xsd:string" />
        <xsd:attribute name="out" type="xsd:string" />
        <xsd:attribute name="name" type="xsd:string" />
        <xsd:attribute name="locked" type="xsd:boolean" />
    </xsd:complexType>
    <xsd:element name="nodo_cliente/nodo_navegacional"
type="cliente/navegacional:nodo_cliente/nodo_navegacional" />
</sodraModel>
```

Con base en el código XMI, a continuación, se describen las funcionalidades de cada tipo simple dentro del tipo complejo en función de los nombres que se definen en el punto 3.3.1.

1. **idobj**: Define un identificador único del nodo.
2. **x**: Establece la coordenada X del nodo.
3. **y**: Muestra la coordenada Y del nodo.
4. **isnodo**: Determina si se trata de un nodo del cliente o navegacional.
5. **extender**: Define al nodo como sucesor de otro.
6. **action**: Establece el nombre del nodo en función de los nombres del punto 3.3.1.
7. **form**: Muestra la forma gráfica del nodo (square/circle).
8. **in**: Determina el identificador de los nodos del cuál es destino.
9. **out**: Define el identificador de los nodos del cuál es origen.
10. **name**: Establece el nombre del nodo.
11. **locked**: Determina si un nodo se permite editar o mover.

Definición de los nombres de los nodos.

El diagrama del cliente cuenta con tres elementos y le antecede el prefijo cte_nodo_:

1. **cte_nodo_concepto**: Representa un nodo de concepto que se muestra en la Figura 3.26.

Figura 3.26 Nodo de concepto.

2. **cte_nodo_relacion**: Representa un nodo de relación que se muestra en la Figura 3.27.

Figura 3.27 Nodo de relación.

3. **cte_nodo_nota**: Representa un nodo de nota que se muestra en la Figura 3.28.

Figura 3.28 Nodo de nota.

El diagrama navegacional cuenta con siete elementos y le antecede el prefijo nav_nodo:

1. **nav_nodo_inicio**: Representa un nodo de inicio que se muestra en la Figura 3.29.

Figura 3.29 Nodo de inicio.

2. **nav_nodo_acceso**. Representa un nodo de acceso que se muestra en la Figura 3.30.

Figura 3.30 Nodo de acceso.

3. **nav_nodo_navegacional**. Representa un nodo navegacional que se muestra en la Figura 3.31.

Figura 3.31 Nodo navegacional.

4. **nav_nodo_menu**. Representa un nodo de menú que se muestra en la Figura 3.32.

Figura 3.32 Nodo de menú.

5. **nav_nodo_consulta**. Representa un nodo de consulta que se muestra en la Figura 3.33.

Figura 3.33 Nodo de consulta.

6. **nav_nodo_lista**: Representa un nodo de lista que se muestra en la Figura 3.34.

Figura 3.34 Nodo de lista.

7. **nav_nodo_proceso**: Representa un nodo de proceso que se muestra en la Figura 3.35.

Figura 3.35 Nodo de proceso

8. **nav_nodo_nota**: Representa un nodo de nota que se muestra en la Figura 3.36.

Figura 3.36 Nodo de nota

Ejemplo del diagrama del cliente con su modelo

En este ejemplo se define un diagrama del cliente que resuelve este enunciado: Una escuela tiene como atributo un nombre. En la Figura 3.37 se muestra el diagrama resultante:

Figura 3.37 Diagrama del cliente.

A continuación, se muestra el modelo equivalente al diagrama de la Figura 3.27:

```
<sodra:Diagram diagramType="Cliente" name="ejemplo1" toolName="Sodra" xmi:id="0">
    <xmi:Extension extender="Sodra">
        <properties>
            <lastEdit value="20171113005037"/>
            <grid value="false"/>
            <autosave value="false"/>
            <rule value="false"/>
            <fileopen value="false"/>
        </properties>
    </xmi:Extension>
    <sodra:Diagram.elements>
        <sodra:DiagramElement type="square" subject="cte_nodo_concepto" xmi:id="9600891">
            <xmi:Extension extender="Sodra">
                <idobj value="9600891"/>
                <x value="76px"/>
                <y value="123px"/>
                <isnodo value="1"/>
                <extender value="0"/>
                <action value="cte_nodo_concepto"/>
                <form value="square"/>
                <in value=""/>
                <out value="10383405"/>
                <name value="Escuela"/>
                <locked value="false"/>
            </xmi:Extension>
        </sodra:DiagramElement>
        <sodra:DiagramElement type="circle" subject="cte_nodo_relacion" xmi:id="10383405">
            <xmi:Extension extender="Sodra">
```

```
                <idobj value="10383405"/>
                <x value="360px"/>
                <y value="123px"/>
                <isnodo value="1"/>
                <extender value="0"/>
                <action value="cte_nodo_relacion"/>
                <form value="circle"/>
                <in value="9600891"/>
                <out value="10037881"/>
                <name value="Tiene"/>
                <locked value="false"/>
            </xmi:Extension>
        </sodra:DiagramElement>
        <sodra:DiagramElement type="square" subject="cte_nodo_concepto" xmi:id="10037881">
            <xmi:Extension extender="Sodra">
                <idobj value="10037881"/>
                <x value="554px"/>
                <y value="123px"/>
                <isnodo value="1"/>
                <extender value="0"/>
                <action value="cte_nodo_concepto"/>
                <form value="square"/>
                <in value="10383405"/>
                <out value=""/>
                <name value="Nombre"/>
                <locked value="false"/>
            </xmi:Extension>
        </sodra:DiagramElement>
    </sodra:Diagram.elements>
</sodra:Diagram>
```

En la Figura 3.38 se muestra la notación equivalente a la Figura 3.37:

```
Definición de clases:
- Escuela{{Nombre},{Ø}};

Relaciones entre clases:
Sin relaciones entre clases.
```

Figura 3.38 Notación equivalente al diagrama del cliente.

Ejemplo del diagrama navegacional con su modelo

En este ejemplo se define un diagrama navegacional que resuelve este enunciado: Se necesita un sitio de autos que permita consultar autos, agregar un nuevo auto y mostrar todos los autos disponibles a partir de un menú de selección. En la Figura 3.39 se muestra el diagrama resultante:

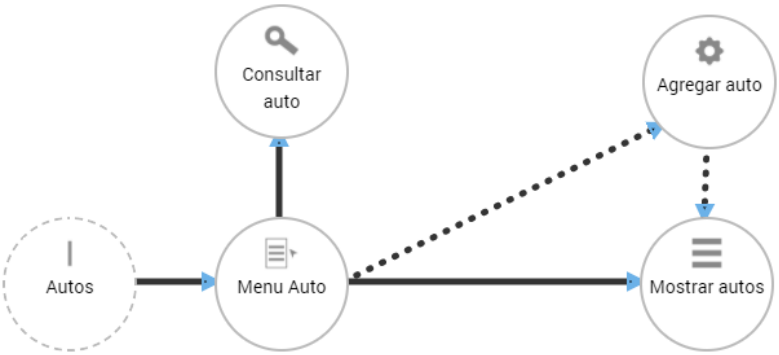


Figura 3.39 Diagrama navegacional.

A continuación, se muestra el modelo equivalente al diagrama de la Figura 3.39:

```
<sodra:Diagram diagramType="Navegacional" name="ejemplo1" toolName="Sodra" xmi:id="0">
    <xmi:Extension extender="Sodra">
        <properties>
            <lastEdit value="20171113005952"/>
            <grid value="false"/>
            <autosave value="false"/>
            <rule value="false"/>
            <fileopen value="false"/>
        </properties>
    </xmi:Extension>
    <sodra:Diagram.elements>
        <sodra:DiagramElement type="circle" subject="nav_nodo_inicio" xmi:id="2838752">
            <xmi:Extension extender="Sodra">
                <idobj value="2838752"/>
                <x value="233px"/>
                <y value="245px"/>
                <isnodo value="1"/>
                <extender value="1"/>
                <action value="nav_nodo_inicio"/>
                <form value="circle"/>
                <in value=""/>
                <out value="12474121"/>
                <name value="Autos"/>
                <locked value="false"/>
            </xmi:Extension>
        </sodra:DiagramElement>
        <sodra:DiagramElement type="circle" subject="nav_nodo_menu" xmi:id="12474121">
            <xmi:Extension extender="Sodra">
                <idobj value="12474121"/>
                <x value="397px"/>
                <y value="245px"/>
                <isnodo value="1"/>
                <extender value="1"/>
                <action value="nav_nodo_menu"/>
                <form value="circle"/>
                <in value="2838752"/>
                <out value="2595219,10611856,7048927"/>
                <name value="Menu Auto"/>
                <locked value="false"/>
```

```
            </xmi:Extension>
        </sodra:DiagramElement>
        <sodra:DiagramElement type="circle" subject="nav_nodo_consulta" xmi:id="2595219">
            <xmi:Extension extender="Sodra">
                <idobj value="2595219"/>
                <x value="584px"/>
                <y value="89px"/>
                <isnodo value="1"/>
                <extender value="1"/>
                <action value="nav_nodo_consulta"/>
                <form value="circle"/>
                <in value="12474121"/>
                <out value=""/>
                <name value="Consultar auto"/>
                <locked value="false"/>
            </xmi:Extension>
        </sodra:DiagramElement>
        <sodra:DiagramElement type="circle" subject="nav_nodo_proceso" xmi:id="7048927">
            <xmi:Extension extender="Sodra">
                <idobj value="7048927"/>
                <x value="583px"/>
                <y value="246px"/>
                <isnodo value="1"/>
                <extender value="1"/>
                <action value="nav_nodo_proceso"/>
                <form value="circle"/>
                <in value="12474121"/>
                <out value="10611856"/>
                <name value="Agregar auto"/>
                <locked value="false"/>
            </xmi:Extension>
        </sodra:DiagramElement>
        <sodra:DiagramElement type="circle" subject="nav_nodo_lista" xmi:id="10611856">
            <xmi:Extension extender="Sodra">
                <idobj value="10611856"/>
                <x value="580px"/>
                <y value="396px"/>
                <isnodo value="1"/>
                <extender value="1"/>
                <action value="nav_nodo_lista"/>
                <form value="circle"/>
                <in value="12474121,7048927"/>
                <out value=""/>
                <name value="Mostrar autos"/>
                <locked value="false"/>
            </xmi:Extension>
        </sodra:DiagramElement>
    </sodra:Diagram.elements>
</sodra:Diagram>
```

En la Figura 3.40 se muestra la notación equivalente a la Figura 3.39:

```
(Autos){NodoInicio}-[EnlaceNavegacional]->(Menu Auto){NodoMenu};
(Menu Auto){NodoMenu}-[EnlaceNavegacional]->(Consultar auto){NodoConsulta};
(Menu Auto){NodoMenu}-[EnlaceNavegacional]->(Mostrar autos){NodoLista};
(Menu Auto){NodoMenu}-[EnlaceProceso]->(Agregar auto){NodoProceso};
(Agregar auto){NodoProceso}-[EnlaceProceso]->(Mostrar autos){NodoLista};
```

Figura 3.40 Notación equivalente al diagrama navegacional.

Herramienta SODRA

SODRA (Sistema de Objetos de Diagramación Relacional Amigable) es una aplicación Web que genera los diagramas del cliente y navegacional propuestos en [1] de forma interactiva.

A continuación, se muestran los algoritmos de la herramienta SODRA

Aplicación del patrón Memento

Para construir el patrón memento, se generaron dos prototipos en JavaScript (equivalente a una clase en Java) que son `HtsMemento` y `Memento`. En las Figuras 3.41 y 3.42 se muestran sus representaciones.

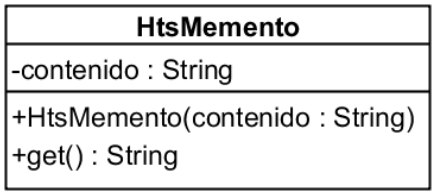


Figura 3.41 Prototipo HtsMemento.

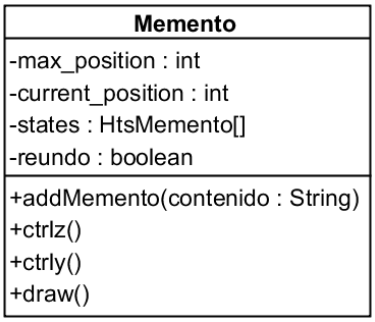


Figura 3.42 Prototipo Memento.

La clase `HtsMemento` actúa como un *bean* que encapsula el estado del lienzo y que a través de clase `Memento` se agrega a un conjunto de estados conforme se realice algún cambio. El usuario

en algún punto utilizará los métodos *ctrlz* y *ctrly* para deshacer y rehacer acciones, respetivamente. Desde la interfaz gráfica, la llamada a estos métodos se realiza con la combinación de teclas “ctrl + z” y “ctrl + y” o a través del menú Edición bajo los nombres deshacer y rehacer.

Marcador de nodos en las reglas

El marcador de nodos muestra la posición y longitud de los nodos en el plano cartesiano del lienzo en las coordenadas *X*, *Y*. En la Figura 3.43 se muestra el marcador de dos nodos y en la Figura 3.44 se muestra el marcador de cinco nodos:

Figura 3.43 Nodos con marcas separadas en las reglas.

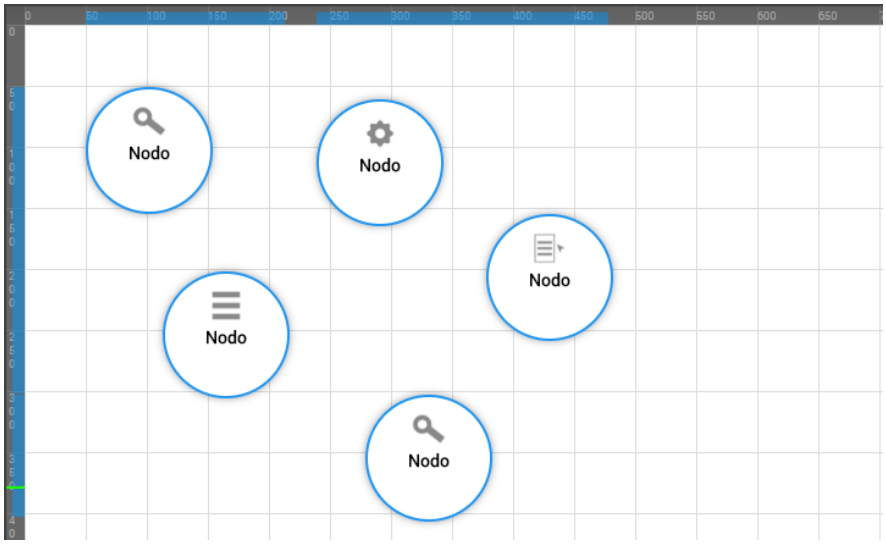


Figura 3.44 Nodos con marcas juntas en las reglas.

Con base en estas figuras se definieron las siguientes reglas:

1. Para cada nodo dentro del plano cartesiano se calcula su punto inicial y su longitud.

2. Si se encuentran dos o más nodos dentro del rango de otro u otros, se organizarán desde la posición más cercana a las coordenadas 0,0 hasta las coordenadas *N*, *M* y se ajustará la dimensión de los nodos (excepto el primero) en las coordenadas *X*, *Y*, tal que cada nodo en la marca aparente ser uno sólo. En las Figuras 3.45 y 3.46 se muestra el antes y el después de la aplicación de la regla 2, respectivamente.

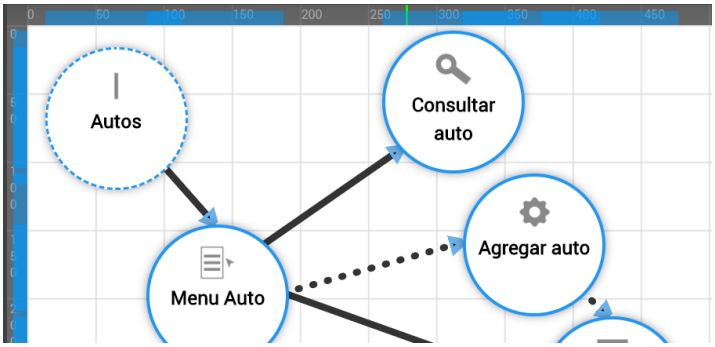


Figura 3.45 Marcas antes de aplicar la regla 2.

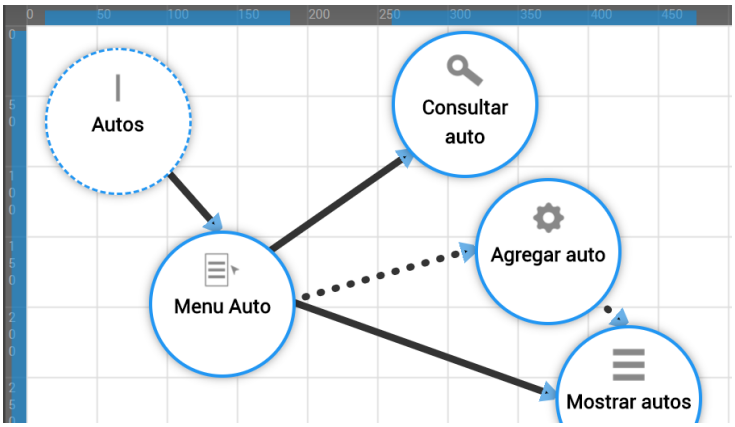


Figura 3.46 Marcas después de aplicar la regla 2.

En la Figura 3.35 se aprecian que las marcas azules se encuentran unas sobre las otras ocasionando que en algunos casos las tonalidades se vean más fuertes. En la Figura 3.36, las marcas se encuentran homogeneizadas y aparentan ser una sola marca azul.

3. Si la diferencia entre dos o más nodos es mayor a cero pixeles (ver Figura 3.33), se genera una nueva marca y se aplica la regla 2.

Cálculo de puntos imaginarios en los nodos

Cada nodo dentro del lienzo cuenta con 8 puntos imaginarios que sirven como base para la unión entre otro nodo a través de los nodos de relación. En la Figura 3.47 se muestra la presentación de un nodo rectangular con sus 8 puntos:

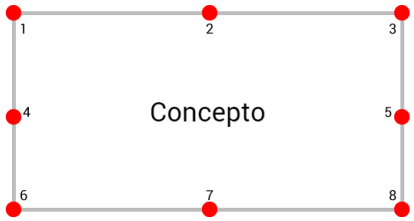


Figura 3.47 Representación de un nodo rectangular con sus ocho puntos.

Para obtener las coordenadas de los 8 puntos de un nodo rectangular se obtienen las coordenadas originales del punto 1 y se obtiene el largo y alto del nodo. Posteriormente para los puntos 2 al 7 se realizan los cálculos con base en las coordenadas y en la dimensión del nodo. Por ejemplo, un nodo en las coordenadas 30,50 con 200px de largo y 100px de alto. La Tabla 3.23 muestra el cálculo de los 7 puntos restantes siendo $x = 30$, $y = 50$, $w = 200$ y $h = 100$.

Tabla 3.23 Fórmulas para el cálculo de los ocho puntos imaginarios del nodo.

Punto	Fórmula	Sustitución	Coordenadas
1	$X1 = x$, $Y1 = y$	$X1 = 30$, $Y1 = 50$	(30, 50)
2	$X2 = x + (w/2)$, $Y2 =$ y	$X2 = 30 + (200/2)$, $Y2 = 50$	(130, 50)
3	$X3 = x + w$, $Y3 =$ y	$X3 = 30 + 200$, $Y3 = 50$	(230, 50)
4	$X4 =$ x, $Y4 = y + (h / 2)$	$X4 = 30$, $Y4 = 50 + (100/2)$	(30,100)
5	$X5 = x + w$, $Y5 = y + (h / 2)$	$X5 = 30 + 200$, $Y5 = 50 + (100/2)$	(230,100)
6	$X6 = x$, $Y6 = y + h$	$X6 = 30$, $Y6 = 50 + 100$	(30,150)
7	$X7 = x + (w / 2)$, $Y7 = y + h$	$X7 = 30 + (200/2)$, $Y7 = 50 + 100$	(130, 150)
8	$X8 = x + w$, $Y8 = y + h$	$X8 = 30 + 200$, $Y8 = 50 + 100$	(230,150)

Para los nodos circulares, se aplican las mismas fórmulas que en la Tabla 3.1. Sin embargo, el cálculo de los puntos 1, 3, 6 y 8 (puntos fuera de rango) se realizan con una conversión de radianes a grados dado que todos los elementos del árbol DOM son rectangulares. En la Figura 3.48 se aprecia esta limitación.

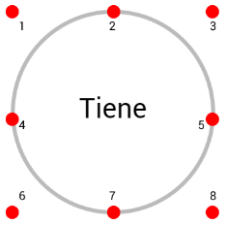


Figura 3.48 Nodo circular con limitaciones en 4 puntos.

De no hacerse el cálculo correcto de los puntos fuera de rango, se presentará un espacio en blanco en el nodo de relación como se muestra en la Figura 3.49 (ver punto 3.4.1.5):

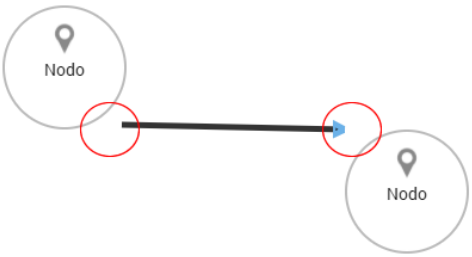


Figura 3.49 Relación incompleta entre dos nodos.

Los cálculos adicionales que se necesitan para los puntos fuera de rango son los siguientes:

$$sindeg\ (x) = Seno\ (\theta\ /\ 180\ *\ 3.14159265)$$

$$cosdeg\ (y) = Coseno\ (\theta\ /\ 180\ *\ 3.14159265)$$

Adicional a ello, se obtiene el ángulo de inclinación de los puntos fuera de rango como se muestra en la Tabla 3.24:

Tabla 3.24 Ángulos de inclinación de los puntos fuera de rango.

Punto	**Ángulo de inclinación**
1	225°
3	135°
6	315°
8	45°

Aplicando la fórmula de conversión de radianes a grados, el cálculo adicional a los puntos fuera de rango se realiza con los ángulos de la Tabla 3.2. La Tabla 3.3 muestra la posición real de los

puntos fuera de rango tomando como datos de prueba $x = 30$, $y = 50$, $w = 100$, $h = 100$, *centrox* = $x + (w/2)$ y *centroy* = $y + (h/2)$:

Tabla 3.25 Posición real de los puntos fuera de rango.

Punto	Fórmula adicional	Sustitución	Coordenada final
1	*X1* = *centrox* + (50 * *sindeg*(225)) *Y1* = *centroy* + (50 * *cosdeg*(225))	*X1* = 80 + (50 * -0.7071) *Y1* = 100 + (50 * -0.7071)	(44.64, 64.64)
3	*X3* = *centrox* + (50 * *sindeg*(135)) *Y3* = *centroy* + (50 * *cosdeg*(135))	*X3* = 80 + (50 * 0.7071) *Y3* = 100 + (50 * -0.7071)	(115.35, 64.64)
6	*X6* = *centrox* + (50 * *sindeg*(315)) *Y6* = *centroy* + (50 * *cosdeg*(315))	*X6* = 80 + (50 * -0.7071) *Y6* = 100 + (50 * 0.7071)	(44.64, 135.35)
8	*X8* = *centrox* + (50 + *sindeg*(45)) *Y8* = *centroy* + (50 + *cosdeg*(45))	*X8* = 80 + (50 * 0.7071) *Y8* = 100 + (50 * 0.7071)	(115.35, 135.35)

Con base en los cálculos adicionales de los puntos fuera de rango más los cálculos de los puntos 2, 4, 5 y 7, se obtienen los ocho puntos imaginarios como se muestra en la Figura 3.50:

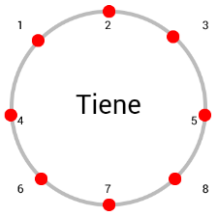


Figura 3.50 Cálculo de los puntos imaginarios de un nodo circular.

Estos cálculos unen los dos nodos correctamente como se muestra en la Figura 3.51 (ver punto 3.4.1.5).

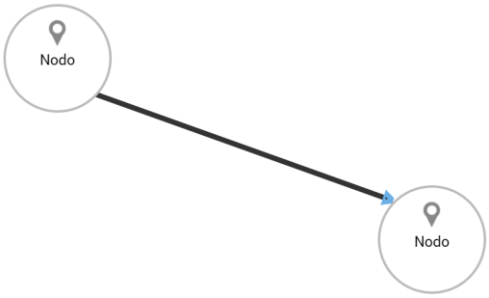


Figura 3.51 Relación completa entre dos nodos.

Selección múltiple de nodos

Cuando un nodo dentro del lienzo se selecciona, este cambia de color a azul. Esto le brinda la habilidad de arrastrarse (excepto cuando se bloquea) y de ubicar sus coordenadas a través de las reglas. En la Figura 3.52 se muestra un ejemplo de un nodo seleccionado.

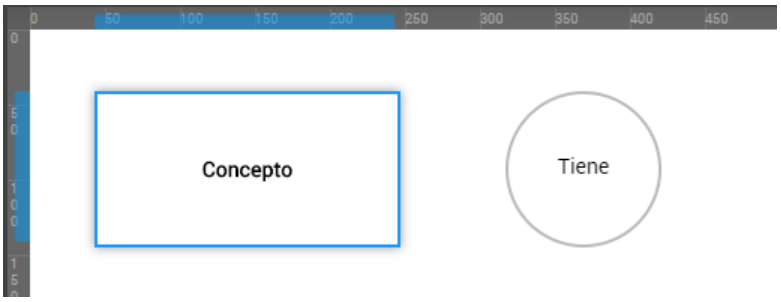


Figura 3.52 Diferencia entre un nodo seleccionado y no seleccionado.

La herramienta SODRA cuenta con un selector de nodos de color azul que se activa cuando se realiza un arrastre con el ratón sobre los elementos a seleccionar. Si el selector de nodos toca alguno de los 8 puntos imaginarios de un nodo en particular, este se seleccionará. La Figura 3.53 muestra la selección múltiple de nodos:

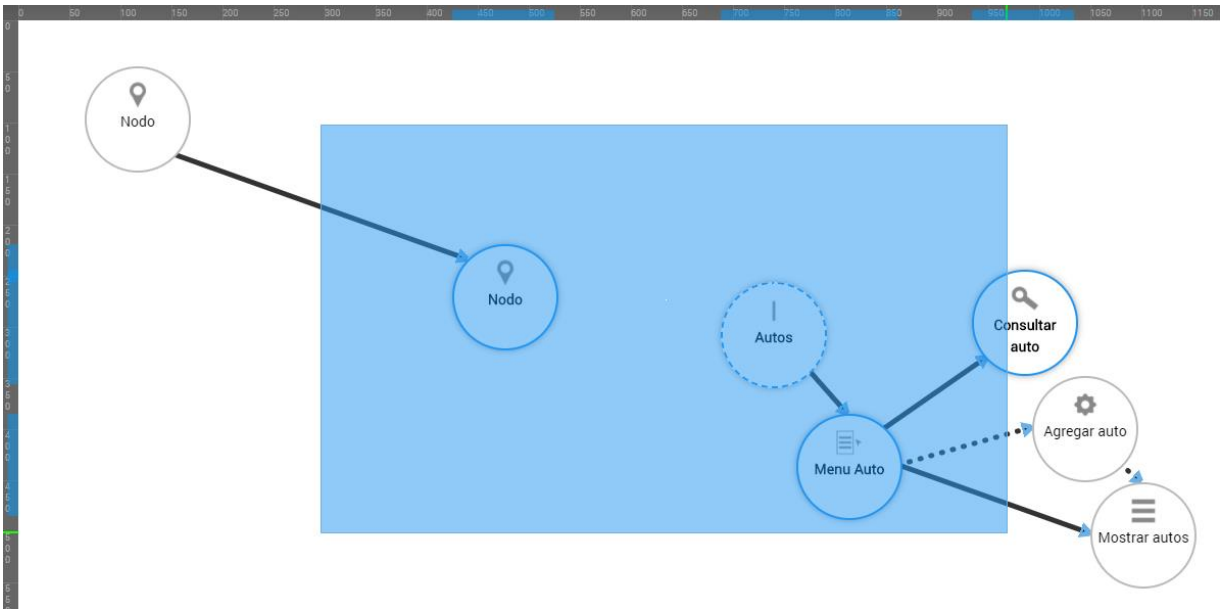


Figura 3.53 Selección múltiple de nodos.

En términos técnicos, el selector de nodos es un elemento de SVG que toma como coordenadas *x0* y *y0* el lugar donde se hizo clic. A medida que se arrastra el cursor por el lienzo, el largo y el ancho cambia por lo que con cada movimiento se actualizan los valores de cálculo que se definen por la coordenada x, la coordenada y, el largo (*width*) y el alto (*height*). Con base en estas posiciones se define la siguiente regla:

Para cada nodo dentro del lienzo se obtendrán sus 8 puntos imaginarios. Si algún punto se encuentra dentro del selector de nodos, el nodo concordante se seleccionará (como se aprecia en la Figura 3.43).

Cálculo de la distancia entre dos puntos dados de un sistema de coordenadas

El cálculo de la distancia entre dos puntos es un algoritmo que, dada una relación entre 2 nodos, encuentra la menor distancia entre cada conjunto de 8 puntos imaginarios (ver punto 3.4.1.3) y dibuja una línea en SVG que los une. A medida que se mueven estos nodos, el algoritmo se actualiza para encontrar de manera automática la distancia más óptima. En la Figura 3.54 se muestra un ejemplo de 2 conjuntos de 8 puntos imaginarios:

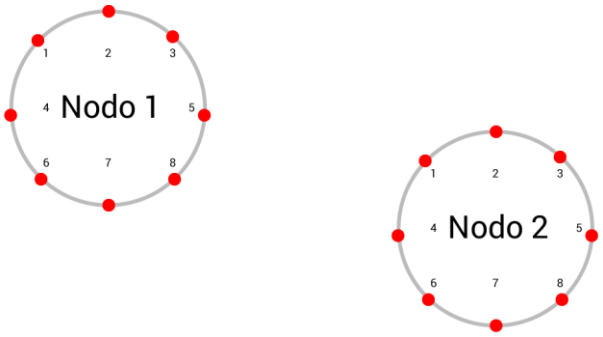


Figura 3.54 2 conjuntos de 8 puntos imaginarios.

A simple vista, la distancia más corta entre los nodos es el punto 8 del nodo 1 y el punto 4 del nodo 2. Sin embargo, para que la herramienta SODRA identifique la distancia más óptima, se utiliza la distancia euclidiana [30]. Para cada dos nodos se realizan 64 posibles combinaciones aplicando la siguiente fórmula que se muestra en la Figura 3.55 siendo "*i*" alguno de los 8 puntos del nodo 1 y "*j*" alguno de los 8 puntos del nodo 2:

$$\mathrm{d(i,}j) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)}^2$$

Figura 3.55 Fórmula de la distancia euclidiana.

Una vez que se identificaron los puntos de unión más óptimos, se pasan las coordenadas de dichos puntos a una recta en SVG. En la Figura 3.56 se muestra la aplicación de este algoritmo:

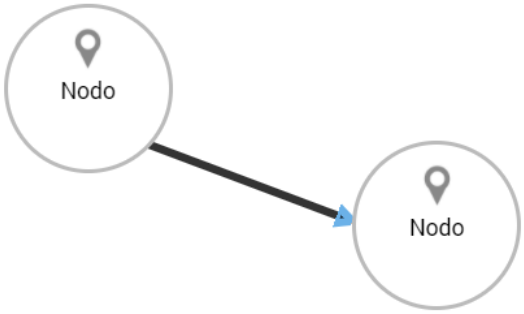


Figura 3.56 Aplicación del algoritmo con la distancia euclidiana.

Utilización de Canvas y SVG para la generación de imágenes

El elemento Canvas permite una interacción más fluida con el usuario sobre una aplicación Web sobre el comportamiento y manejo de eventos de los nodos. El elemento SVG funge como generador de rectas que unen a los nodos y se entrelaza con las coordenadas de Canvas para trabajar de manera conjunta. En la Figura 3.57 se muestra la combinación de estas dos tecnologías a través de una interfaz de usuario del diagrama navegacional:

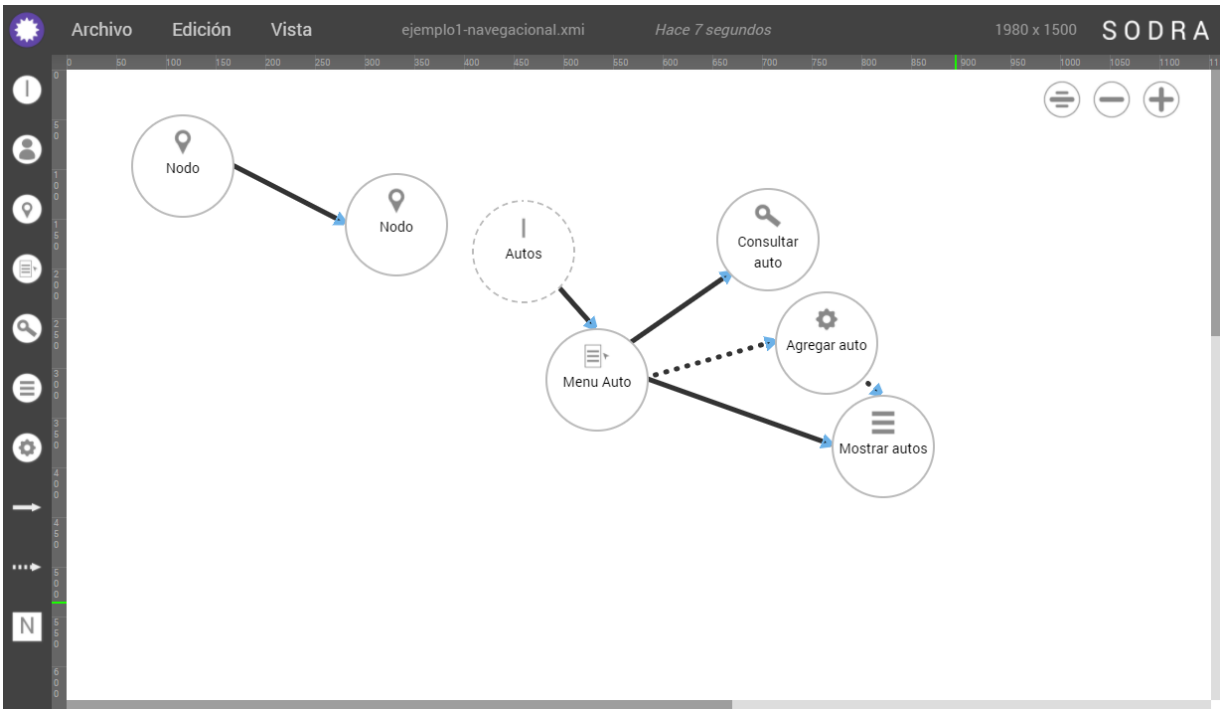


Figura 3.57 Trabajo conjunto de los elementos Canvas y SVG.

Conforme se utilizan más elementos del lienzo, Canvas aumenta el tamaño disponible para la diagramación y a través del patrón de diseño *Observer*, notifica a SVG que el tamaño del lienzo se modificó para que realice nuevos cálculos para que se mantenga actualizado y transparente al usuario.

Recorte de imagen

El recorte de la imagen elimina el espacio en blanco en las partes superior, inferior, izquierda y derecha del lienzo, dejando el contenido (los nodos y sus relaciones) como la imagen resultante. De forma predeterminada, el tamaño mínimo del lienzo es de *1980px* por *1500px*. En la Figura 3.58 se muestra el proceso de recorte:

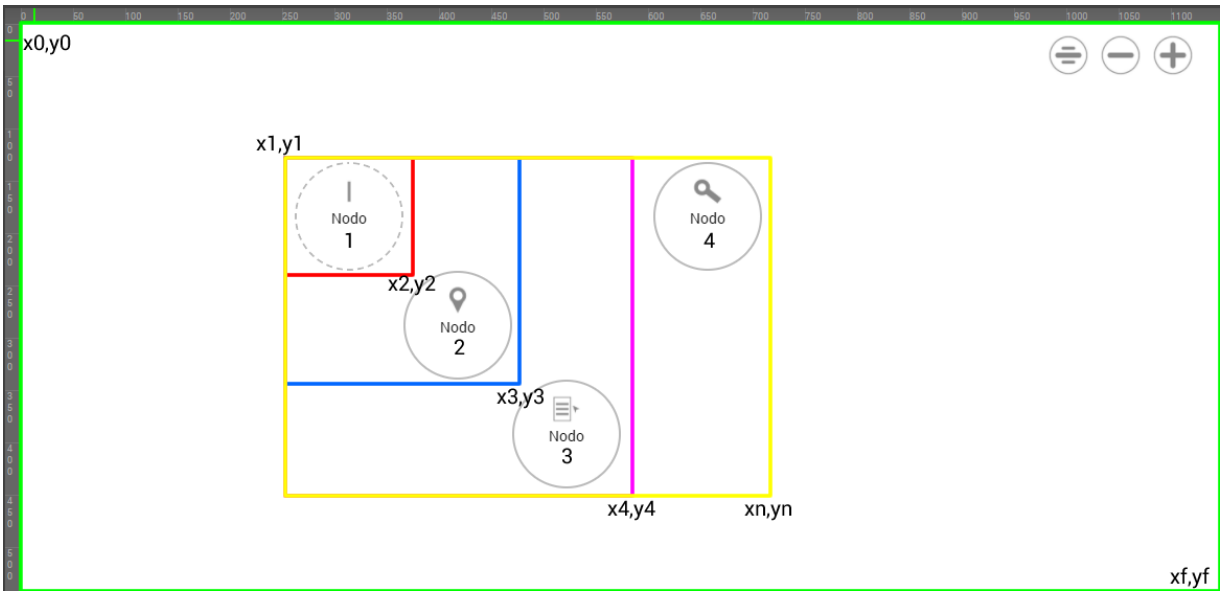


Figura 3.58 Proceso de recorte de la imagen.

Como punto de partida se tienen las siguientes incógnitas:

1. ***x***: Representa la coordenada *x* del punto de origen.
2. ***y***: Define la coordenada *y* del punto de origen.
3. **largo**: Establece el largo (*width*) del lienzo a recortar.
4. **Alto**: Especifica el alto (*height*) del lienzo a recortar.

En la primera iteración se obtiene *x*, *y*, largo y alto que corresponden a los valores 0, 0, 1980 y 1500, respectivamente. En la segunda iteración (línea roja) se determina el punto más cercano a los valores *x*, *y*, que corresponde al punto 1 del nodo 1, por lo que los valores *x*, *y* se reemplazan con *x1*, *y1*. Para la obtención del largo y alto, se calcula con la posición del punto más alejado de 0,0 conforme al nodo 1 y se realiza una diferencia entre el punto *x*, *y*, y el punto *x2*, *y2*. En la tercera iteración (línea azul) se determina el punto más cercano a los valores *x*, *y*, que para este caso no se cumple. Para la obtención del largo y alto, se calcula con la posición del punto más alejado de 0,0 conforme al nodo 2 y se realiza una diferencia entre el punto *x*, *y*, y el punto *x3*, *y3*. Este proceso concluye hasta que se obtengan todos los nodos del lienzo (línea amarilla). Posteriormente, estos valores se envían en formato JSON en conjunto con la representación base64 del lienzo al archivo `jpg.php` donde se realiza el recorte de la imagen. En la Figura 3.59 se muestra el resultado del recorte de la imagen:

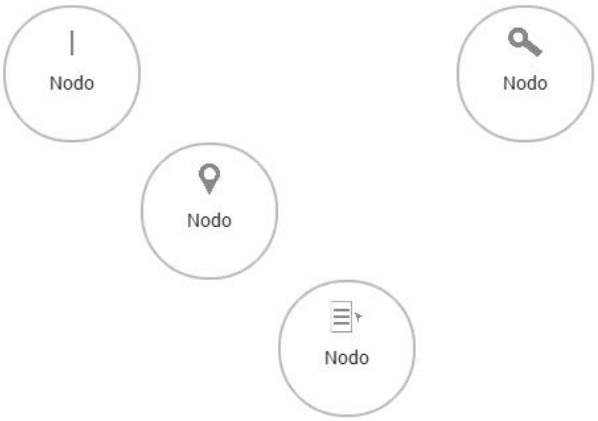


Figura 3.59 Resultado del proceso de recorte de imagen.

Esta imagen sirve para mostrar una vista previa del diagrama (ver Figura 3.60) y para su utilización bajo la modalidad de propósito general.

Figura 3.60 Vista previa del diagrama navegacional.

Capítulo 4. Resultados

Este capítulo tiene como objetivo mostrar la interfaz principal de la herramienta SODRA, así como dos casos de estudio. Cada caso de estudio comprende la especificación de requerimientos y posteriormente la generación del diagrama del cliente y navegacional.

4.1. Interfaz principal de SODRA

En esta sección, se describen las interfaces del diagrama del cliente y navegacional de SODRA que sirven como guía para la comprensión y ubicación de los elementos que posee en función de la generación de los diagramas y notaciones que se muestran en los casos de estudio 1 y 2.

4.1.1. Interfaz común entre ambos diagramas

A continuación, se muestra la interfaz común de los diagramas que se muestra en la Figura 4.1:

Figura 4.1 Interfaz del diagrama del cliente de SODRA.

4.1.1.1. Color rojo: Barra de menú

En esta sección se encuentran los siguientes menús:

El menú archivo que tiene las siguientes acciones:

1. Guardar (ctrl + S). Guarda el diagrama actual.
2. Guardar en Drive (ctrl + G). Almacena el diagrama en local y abre la interfaz de *Google Drive* para su almacenaje en la nube.

3. Exportar (ctrl + S). Abre la interfaz para exportar el proyecto y define la configuración para exportar el proyecto que abarca la exportación del archivo XMI y el archivo JPG o alguno de ellos.
4. Salir (ctrl + B). Regresa a la interfaz de proyectos de SODRA.

El menú Edición que posee los siguientes elementos:

1. Deshacer (ctrl + Z). Deshace una acción del lienzo.
2. Rehacer (ctrl + Y). Rehace una acción del lienzo.
3. Seleccionar todo (ctrl + A). Selecciona todos los nodos del lienzo.

El menú Vista que agrupa las siguientes funciones:

1. Vista previa (ctrl + P). Ilustra el progreso del diagrama en una imagen única.
2. Ver notación (ctrl + O). Muestra el progreso del diagrama en su notación correspondiente.
3. Ver cuadrículas (ctrl + L). Activa o desactiva las cuadrículas del lienzo.
4. Auto guardar (ctrl + Q). Habilita o inhabilita el autoguardado del lienzo que se realiza cada minuto.
5. Ver reglas (ctrl + R). Establece o remueve las reglas del lienzo.

4.1.1.2. Color verde: nombre del archivo

Establece el nombre del archivo tomando como base el nombre del proyecto y el tipo de diagrama con el que se trabaja. Cuando el nombre incluya el carácter "*" significa que dicho diagrama no se encuentra guardado.

4.1.1.3. Color azul: última vez guardado

La primera vez que se abre un diagrama aparece el texto "Sin guardar". Cuando se guarde el diagrama este texto cambia a: "hace unos momentos", "hace unos minutos", entre otros tiempos en función de la última vez que se guarde dicho diagrama.

4.1.1.4. Color amarillo: Tamaño del lienzo

De forma predeterminada, el tamaño inicial del lienzo es de 1980 por 1500 pixeles. A medida que se agregan los nodos al lienzo el tamaño aumenta hasta un límite de 10001 por 10001 pixeles.

4.1.1.5. Color cian: zoom del lienzo

Reestablece el zoom, disminuye o aumenta como se muestran en el orden de los íconos de la Figura 4.2.

Figura 4.2 Íconos para el manejo del zoom en el lienzo.

4.1.2. Diagrama del cliente

En esta interfaz se muestran los elementos principales del diagrama del cliente que se muestra en la Figura 4.3.

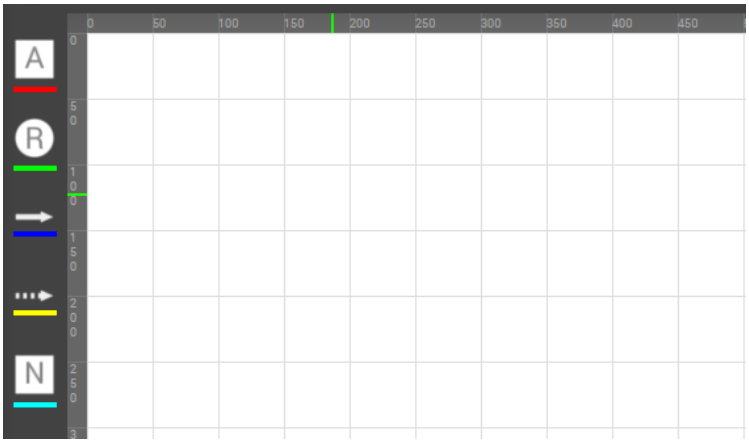


Figura 4.3 Interfaz principal del diagrama del cliente.

A continuación, se describen los significados de los colores de la Figura 4.3:

1. Color rojo: Nodo de concepto.
2. Color verde: Nodo de relación.

3. Color azul: Enlace de nodos.
4. Color amarillo: Enlace de clases de asociación.
5. Color cian: Nodo de notas.

4.1.3. Diagrama navegacional

En la Figura 4.4 se muestra la interfaz de los elementos principales del diagrama navegacional.

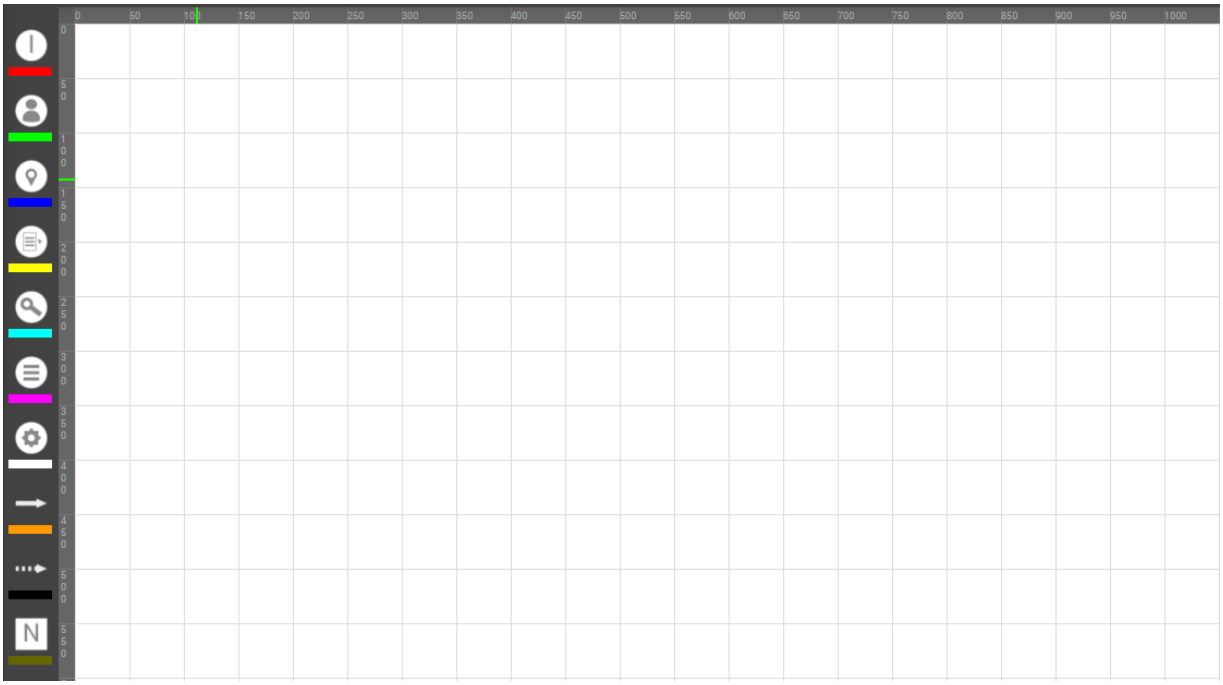


Figura 4.4 Interfaz principal del diagrama navegacional.

A continuación, se describen los significados de los colores de la Figura 4.4:

1. Color rojo. Nodo inicio.
2. Color verde. Nodo de acceso.
3. Color azul. Nodo de navegación.
4. Color amarillo. Nodo de menú.
5. Color cian. Nodo de consulta.
6. Color magenta. Nodo de lista.
7. Color blanco. Nodo de proceso.
8. Color naranja. Enlace de navegación.

9. Color negro. Enlace de proceso.
10. Color verde militar. Nodo de notas.

4.2. Caso de estudio 1: Agenda de contacto

Este caso de estudio se tomó de [1] cuyo propósito es realizar la comparación entre el caso de estudio de dicho trabajo de tesis anterior y este trabajo. Tomando como base este antecedente, en el primer caso de estudio se requiere de una aplicación de una agenda de contactos cuyas tareas son las siguientes:

1. Buscar contactos.
2. Eliminar contactos.
3. Crear contactos nuevos.
4. Actualizar contactos existentes.

A continuación, se determinan las siguientes clases:

1. Agenda. Atributos: descripción.
2. Contacto.
3. Dirección. Atributos: ciudad, país, dirección y código postal.
4. Teléfono. Atributos: código de área, lada y número.
5. Foto. Atributos: Alto y ancho.

Finalmente se determinan las acciones

1. Buscar.
2. Eliminar.
3. Crear.
4. Actualizar.

Para el desarrollo del diagrama del cliente se utilizará la herramienta SODRA. Los pasos que a continuación se muestran, llevarán como resultado el diagrama que se muestra en la Figura 4.17.

SODRA necesita un espacio de trabajo para agrupar un conjunto de proyectos que, a su vez, poseen los diagramas del cliente y navegacional en función de las necesidades del usuario que

utilice dicha herramienta. Para este caso de estudio se crea un nuevo espacio de trabajo bajo el nombre “casoestudio”.

A continuación, se crea el espacio de trabajo haciendo clic en el botón “Crear nuevo espacio de trabajo” encerrado en un cuadro rojo, tal como se muestra en la Figura 4.5:

Figura 4.5 Interfaz principal de SODRA.

Posteriormente, se define el nombre del espacio de trabajo como se muestra en la Figura 4.6. El campo para escribir el espacio de trabajo solo admite letras sin espacios ni acentos ni caracteres diferentes al alfabeto además de que la primera letra no admite un número. Se finaliza dando clic al botón “Crear”:

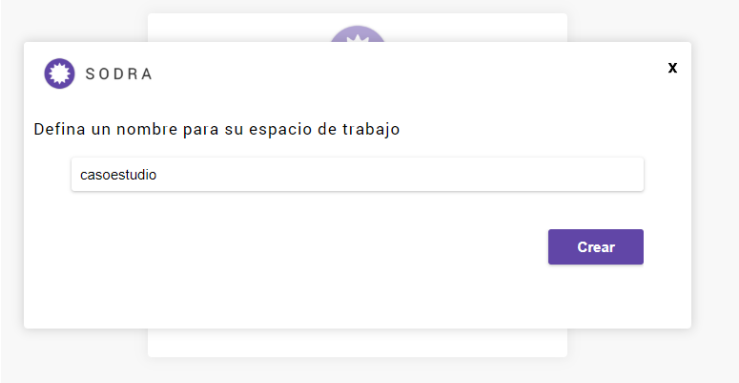


Figura 4.6 Interfaz de definición de espacio de trabajo de SODRA.

Una vez que se configure el espacio de trabajo, inmediatamente se abre la interfaz de definición de nombre de proyectos, que para este caso de estudio en particular se utiliza el nombre “caso1” y se marcan las opciones de crear diagrama del cliente y navegacional. Finalmente, se concluye el proceso dando clic en el botón “Crear” tal y como se muestra en la Figura 4.7:

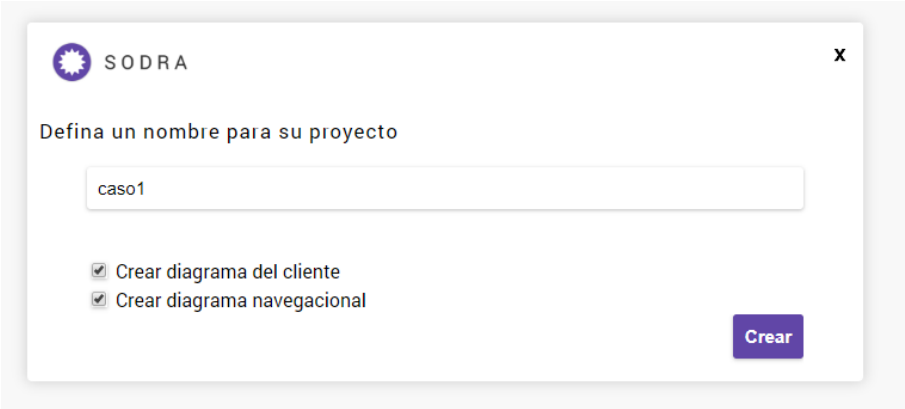


Figura 4.7 Interfaz de definición de nombre para proyectos.

Después de que SODRA realice las configuraciones correspondientes, se generan los diagramas del cliente y navegacional en blanco como se muestra en la Figura 4.8.

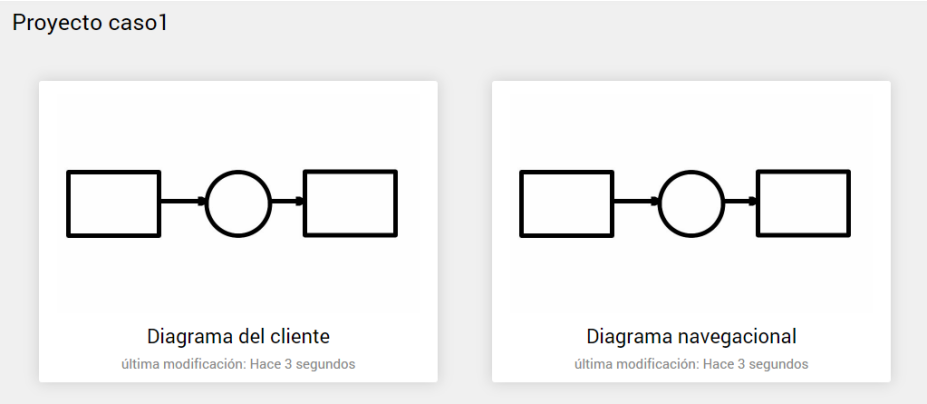


Figura 4.8 Interfaz que muestra los diagramas del cliente y navegacional en blanco.

Si se desea crear otro proyecto dentro del espacio de trabajo “casoestudio” se da clic al botón “Crear” que se encuentra en el menú “Proyecto” como se muestra en la Figura 4.9 y posteriormente, se sigue el paso que se muestra en la Figura 4.7.

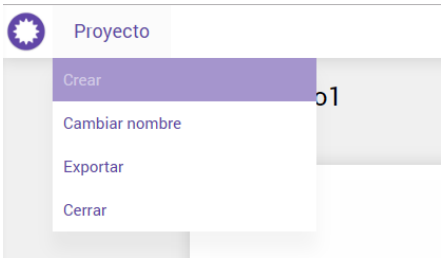


Figura 4.9 Interfaz para crear otro proyecto.

Para iniciar el desarrollo del diagrama del cliente se hace clic en el cuadro “Diagrama del cliente” que se encuentra en la Figura 4.8. Este proceso abre la interfaz del lienzo de dicho diagrama que se muestra en la Figura 4.3.

Para construir el diagrama primero se agregan las 5 clases. Para cada clase se hace clic en “Nodos de concepto”, luego hace clic en el lugar destino, posteriormente se edita el diagrama con el nombre correcto de cada clase como se muestra en la Figura 4.10:

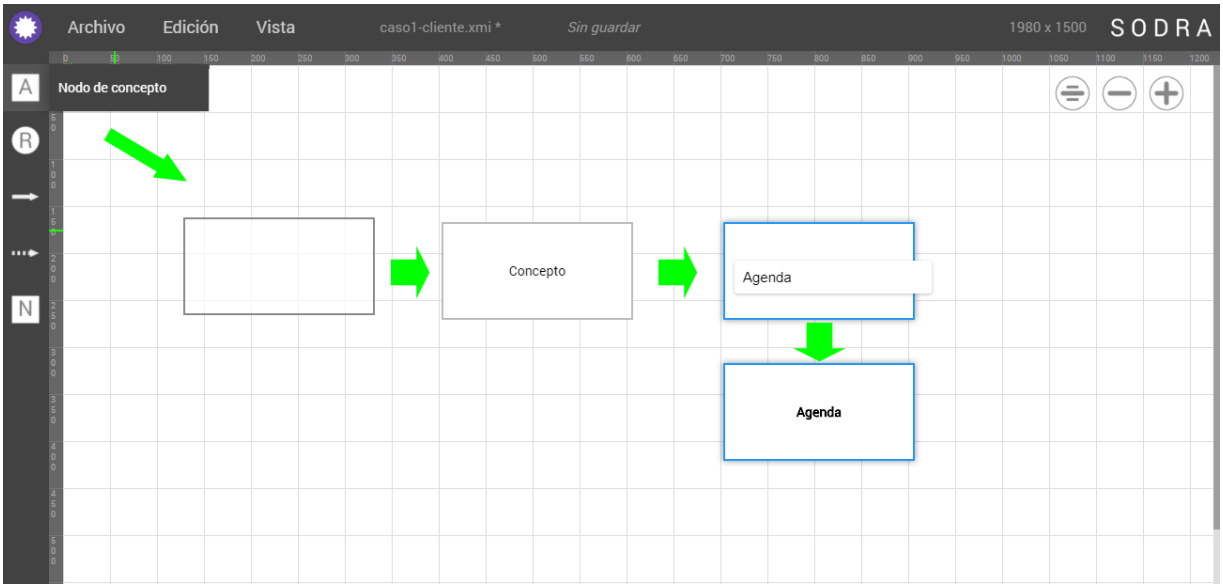


Figura 4.10 Pasos para agregar las clases al diagrama del cliente.

Posteriormente se agregan los atributos de las clases siguiendo el mismo procedimiento de la Figura 4.10. Después de realizar la adición de estos elementos, el diagrama es similar al que se muestra a la Figura 4.11:

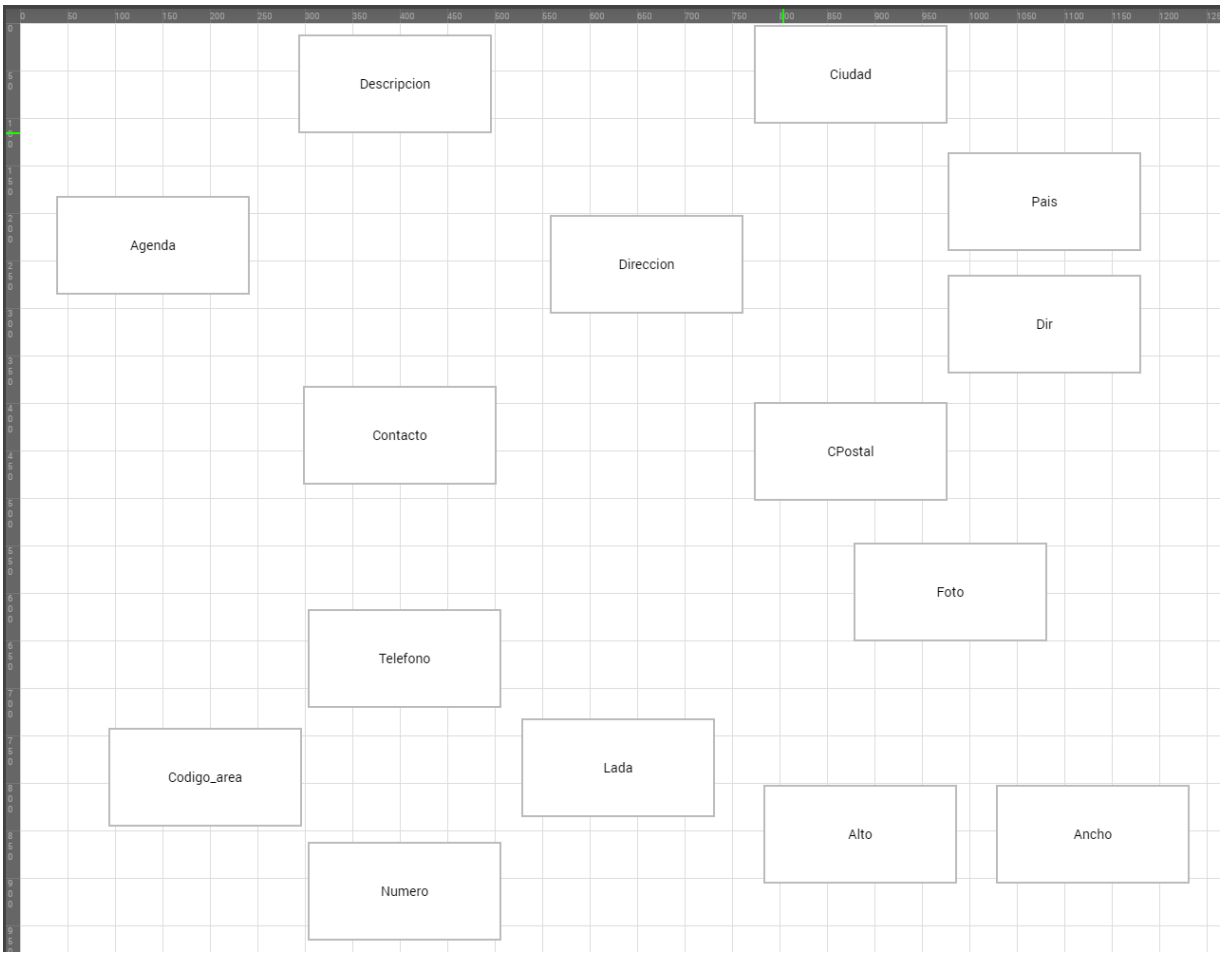


Figura 4.11 Clases y atributos en el diagrama del cliente.

Después se agregan las relaciones de composición siguiendo los pasos de la Figura 4.10 pero ahora se elige el nodo de relación. El diagrama con estos nodos se asemeja a la Figura 4.12. Debido a razones de espacio solo se muestra una fracción del diagrama.

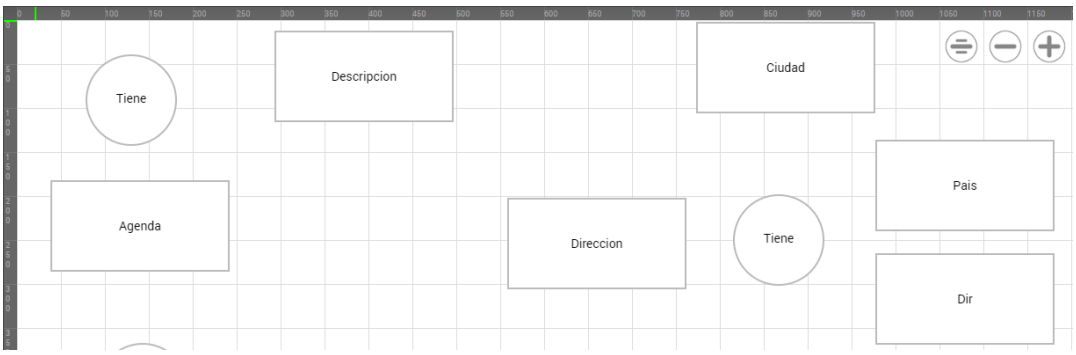


Figura 4.12 Clases y atributos en el diagrama del cliente.

Finalmente se agregan las relaciones entre los nodos. Para ello se siguen los siguientes pasos:

1. Se selecciona el enlace de nodos como se muestra en la Figura 4.13:

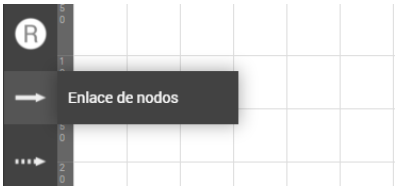


Figura 4.13 Selección del enlace de nodos.

2. Se hace clic en el nodo origen (Agenda) como se muestra en la Figura 4.14:

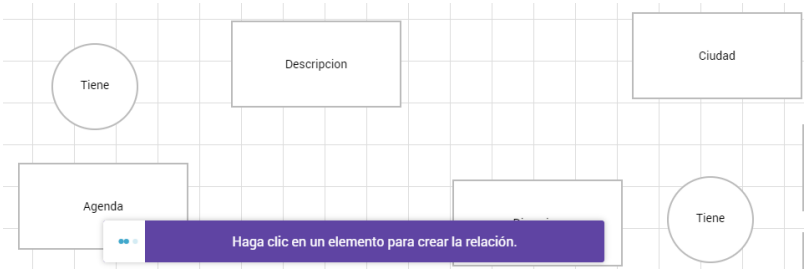


Figura 4.14 Selección del nodo Agenda.

3. Se hace clic en el nodo destino (Tiene) como se muestra en la Figura 4.15:

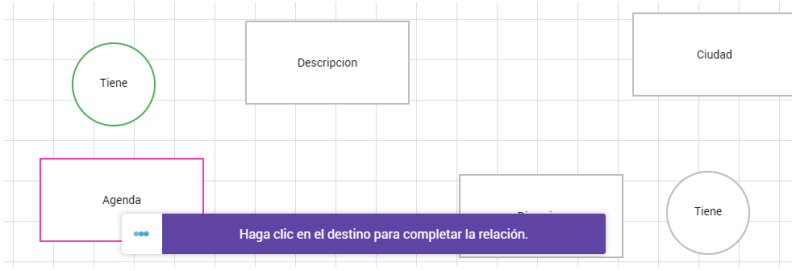


Figura 4.15 Selección del nodo Tiene.

4. La relación se forma entre Agenda y tiene como se muestra en la Figura 4.16:

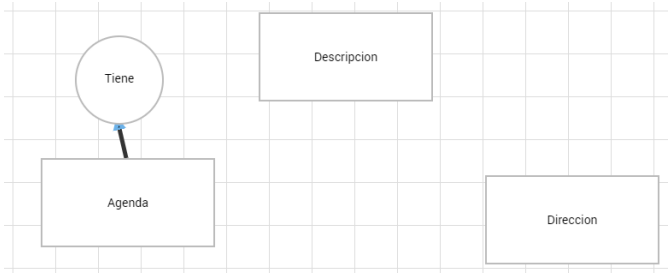


Figura 4.16 Selección del nodo Tiene.

Estos pasos se repiten hasta relacionar todos los nodos del diagrama del cliente.

A continuación, se muestra el diagrama resultante en la Figura 4.17:

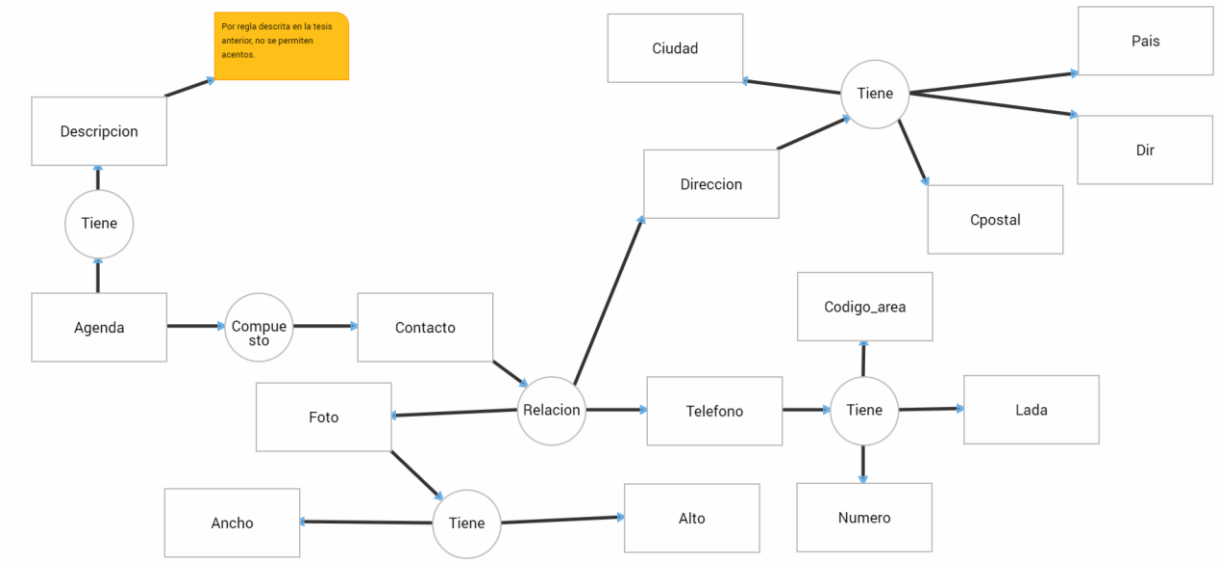


Figura 4.17 Diagrama del cliente del caso de estudio 1.

En la Figura 4.17 se determinaron algunos atributos, así como las relaciones entre la agenda y contacto, así como entre contacto y dirección, teléfono y Foto. Para obtener la notación de este diagrama se hace clic en el menú vista y se selecciona el elemento “ver notación” o se accede a la configuración de teclas ctrl + O. En la Figura 4.18 se muestra la notación propuesta en [1] que simplifica las relaciones entre los nodos:

```
Definición de clases:
- Agenda{{Descripcion},{Ø}};
- Contacto{{Ø},{Ø}};
- Direccion{{Ciudad,Pais,Dir,Cpostal},{Ø}};
- Telefono{{Codigo_area,Lada,Numero},{Ø}};
- Foto{{Alto,Ancho},{Ø}};

Relaciones entre clases:
- Agenda ∈ Contacto
- Contacto ∩ Direccion
- Contacto ∩ Telefono
- Contacto ∩ Foto

Comentarios de las clases:
- Descripcion = "Por regla descrita en la tesis anterior, no se permiten acentos.";
```

Figura 4.18 Notación del diagrama del cliente del caso de estudio 1.

Dentro de las herramientas de SODRA se encuentra la exportación de los diagramas en los formatos XMI y JPG. El XMI es un archivo que permitirá como trabajo futuro, servir como punto de partida para la generación de aplicaciones Web basados en diversos lenguajes de programación. Por otra parte, el archivo JPG es un archivo que sirve como ilustración al resultado del desarrollo del diagrama y su implementación en diversos programas como *Microsoft Word*, *Microsoft Power Point*, entre otros.

Para exportar el diagrama, se accede al menú Archivo seguido del botón exportar o se realiza la combinación de teclas ctrl + E. En la Figura 4.19 se muestra la interfaz para decidir los formatos de archivos a exportar.

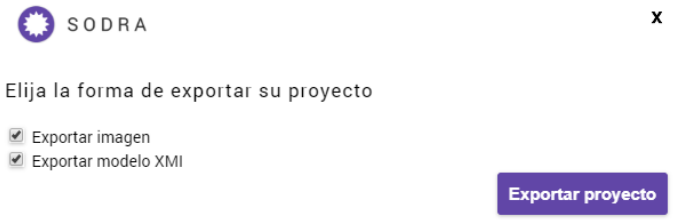


Figura 4.19 Interfaz de exportación de diagramas de SODRA.

Dadas las características de este caso de estudio se seleccionan los dos formatos para ejemplificar el proceso de generación del archivo XMI y JPG. En la Figura 4.20 se muestran los dos archivos que se exportaron de SODRA.

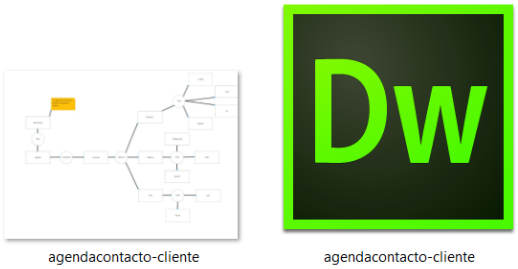


Figura 4.20 Archivos resultantes de la exportación del diagrama del cliente.

Para la generación del diagrama navegacional, cada clase se convierte en un nodo de navegación y los enlaces se realizan a través de enlaces navegacionales, así como la inclusión las acciones que se definieron anteriormente.

En la herramienta, para iniciar su desarrollo se hace clic en el cuadro “Diagrama navegacional” que se encuentra en la Figura 4.8. Este proceso abre la interfaz del lienzo de dicho diagrama que se muestra en la Figura 4.4. Para agregar un nodo al lienzo se selecciona el nodo correspondiente, posteriormente se hace clic en el lugar destino del lienzo y se finaliza con la edición del diagrama con su nombre correspondiente como se muestra en la Figura 4.21:

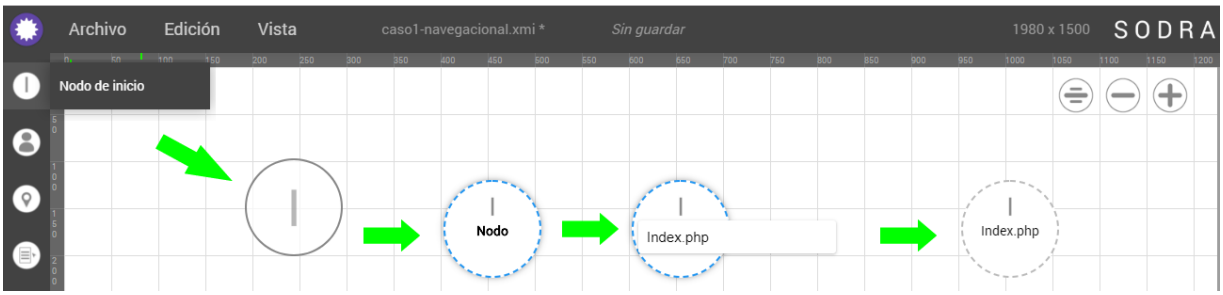


Figura 4.21 Pasos para agregar los nodos al diagrama navegacional.

Para relacionar los nodos se siguen los pasos de las Figuras 4.13, 4.14, 4.15 y 4.16. Dado que se modela un diagrama navegacional se necesita un nodo menú para acceder a los recursos de la aplicación Web. A continuación, se muestra el diagrama navegacional en la Figura 4.22.

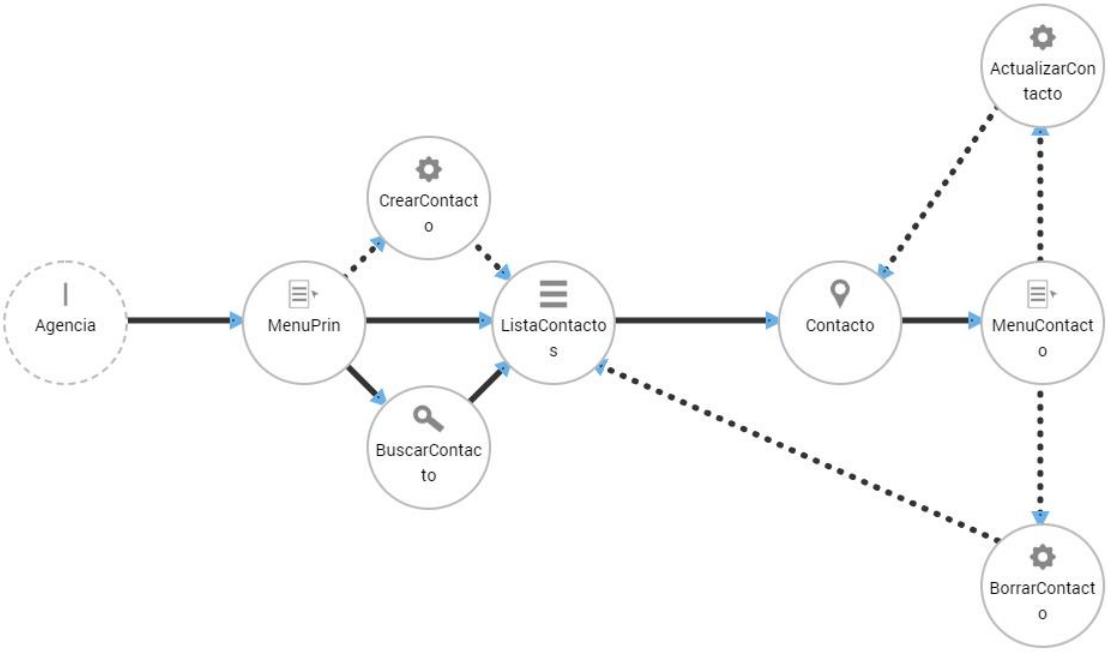


Figura 4.22 Diagrama navegacional del caso de estudio 1.

Posteriormente, Para obtener la notación de este diagrama se hace clic en el menú vista y se selecciona el elemento “ver notación” o se accede a la configuración de teclas ctrl + O. En la Figura 4.23 se muestra la notación propuesta para este diagrama.

```
(Agencia){NodoInicio}-[EnlaceNavegacional]->(MenuPrin){NodoMenu};
(MenuPrin){NodoMenu}-[EnlaceProceso]->(CrearContacto){NodoProceso};
(MenuPrin){NodoMenu}-[EnlaceNavegacional]->(BuscarContacto){NodoConsulta};
(MenuPrin){NodoMenu}-[EnlaceNavegacional]->(ListaContactos){NodoLista};
(CrearContacto){NodoProceso}-[EnlaceProceso]->(ListaContactos){NodoLista};
(BuscarContacto){NodoConsulta}-[EnlaceNavegacional]->(ListaContactos){NodoLista};
(ListaContactos){NodoLista}-[EnlaceNavegacional]->(Contacto){NodoNavegacional};
(Contacto){NodoNavegacional}-[EnlaceNavegacional]->(MenuContacto){NodoMenu};
(MenuContacto){NodoMenu}-[EnlaceProceso]->(ActualizarContacto){NodoProceso};
(MenuContacto){NodoMenu}-[EnlaceProceso]->(BorrarContacto){NodoProceso};
(ActualizarContacto){NodoProceso}-[EnlaceProceso]->(Contacto){NodoNavegacional};
(BorrarContacto){NodoProceso}-[EnlaceProceso]->(ListaContactos){NodoLista};
```

Figura 4.23 Notación del diagrama navegacional del caso de estudio 1.

A continuación, se realiza la exportación de este diagrama con los archivos XMI y JPG que se mostraron en la Figura 4.19. En la Figura 4.24 se muestran los archivos resultantes de la exportación del diagrama navegacional:


agendacontacto-navegacional

agendacontacto-navegacional

Figura 4.24 Archivos resultantes de la exportación del diagrama navegacional.

A continuación, se muestra un fragmento del modelo navegacional del diagrama que se exportó con SODRA en la Figura 4.24:

```
<sodra:Diagram diagramType="Navegacional" name="agendacontacto" toolName="Sodra" xmi:id="0">
        <xmi:Extension extender="Sodra">
                <properties>
                        <lastEdit value="20180530225013" />
                        <grid value="false" />
                        <autosave value="false" />
                        <rule value="false" />
```

```
                        <fileopen value="false" />
                        <idgoogle value="" />
                </properties>
        </xmi:Extension>
        <sodra:Diagram.elements>
                <sodra:DiagramElement type="circle" subject="nav_nodo_inicio"
xmi:id="7173370">
                        <xmi:Extension extender="Sodra">
                                <idobj value="7173370" />
                                <x value="91px" />
                                <y value="265px" />
                                <isnodo value="1" />
                                <extender value="1" />
                                <action value="nav_nodo_inicio" />
                                <form value="circle" />
                                <in value="" />
                                <out value="9460006" />
                                <name value="Agencia" />
                                <locked value="false" />
                        </xmi:Extension>
                </sodra:DiagramElement>
                . . .
        </sodra:Diagram.Elements>
</sodra:Diagram>
```

4.3.Caso de estudio 2: Desarrollo de una aplicación Web autoadministrable.

Como parte de una aplicación real del uso de SODRA, en la empresa GSI Soluciones S.A. de C.V. requiere de una aplicación Web que le permita gestionar los productos y servicios que ofrece, así como la descarga de versiones de prueba de sus productos e información de contacto de propósito general o atendiendo un propósito específico, ya sea la adquisición de licencias de los productos o cotizaciones de los servicios que ofrecen. Para ello se ofrecen las siguientes características:

1. Ofrecer productos.
2. Ofrecer servicios.
3. Datos de contacto.
4. Pruebas y descargas de productos.
5. Información de contacto.

Con base en estas características se determinan las clases del diagrama del cliente

1. Producto. Atributos: nombre, descripción, foto, precio.

2. Servicio. Atributos: título, descripción del servicio.
3. Pruebas y descargas. Atributos: versión, tamaño y compatibilidad.
4. Contacto. Atributos: nombre del cliente, elemento a solicitar, correo y teléfono.

Finalmente se determinan las acciones:

1. Gestionar productos, servicios, descargas y contactos.
2. Solicitar cotización.
3. Descargar producto.
4. Ver producto.
5. Ver servicio.

Dado que anteriormente se creó el espacio de trabajo "casoestudio", se accede a él desde la interfaz principal que se muestra en la Figura 4.5 y se siguen los pasos que acompaña a la Figura 4.9 bajo el nombre “caso2” y se siguen los pasos de las Figuras 4.10, 4.11, 4.12, 4.13, 4.14, 4.15 y 4.16 para desarrollar el diagrama del cliente.

A continuación, en la Figura 4.25 se muestra el diagrama del cliente que corresponde a este caso de estudio.

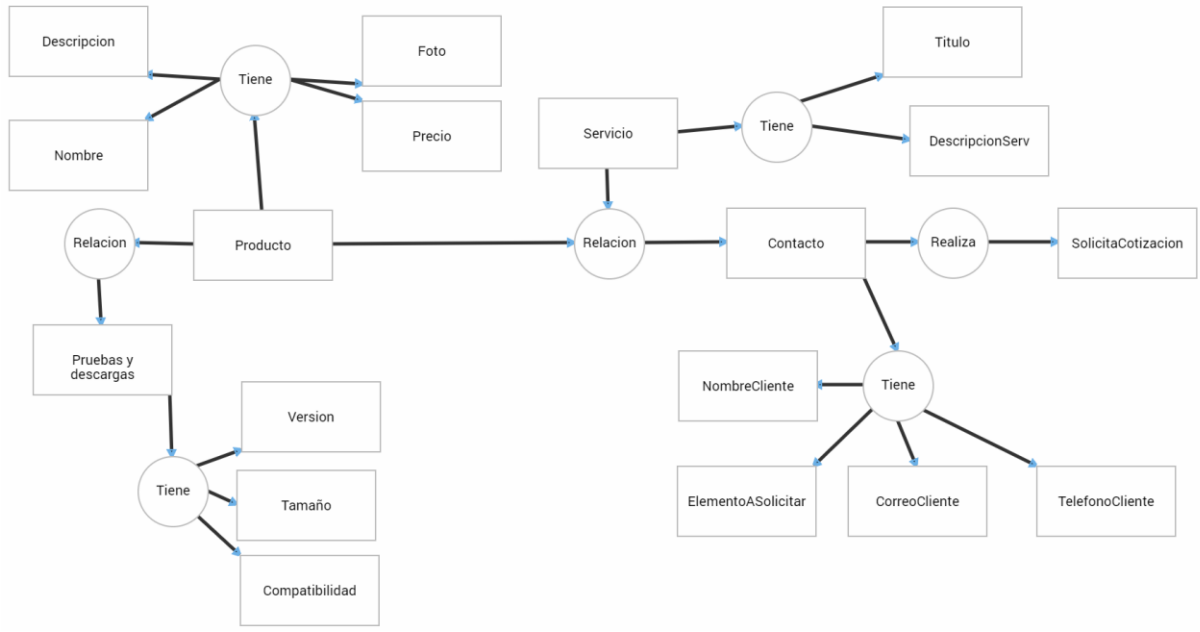


Figura 4.25 Diagrama del cliente del caso de estudio 2.

En la Figura 4.25 se determinaron algunos atributos, así como las relaciones entre producto y servicio con contacto, así como producto con pruebas y descargas. Para obtener la notación de este diagrama se hace clic en el menú vista y se selecciona el elemento “ver notación” o se accede a la configuración de teclas ctrl + O. En la Figura 4.26 se muestra la notación propuesta en [1] que simplifica las relaciones entre los nodos.

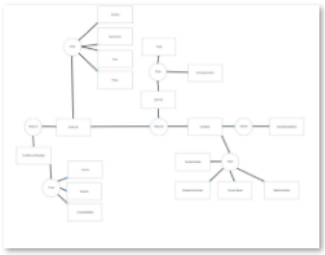
```
Definición de clases:
- Producto{{Nombre,Precio,Descripcion,Foto},{Ø}};
- Pruebas y descargas{{Version,Tamaño,Compatibilidad},{Ø}};
- Servicio{{Titulo,DescripcionServ},{Ø}};
- Contacto{{NombreCliente,TelefonoCliente,CorreoCliente,ElementoASolicitar},{SolicitaCotizacion}};

Relaciones entre clases:
- Producto n Pruebas y descargas
- Producto n Contacto
- Servicio n Contacto

Comentarios de las clases:
```

Figura 4.26 Notación del diagrama del cliente del caso de estudio 2.

Tras la realización de este diagrama, el resultado se exporta con los archivos XMI y JPG que se mostraron en la Figura 4.19. En la Figura 4.27 se muestran los archivos resultantes de la exportación del diagrama del cliente:



sitioweb-cliente

sitioweb-cliente

Figura 4.27 Archivos resultantes de la exportación del diagrama del cliente.

Tomando como base la descripción de la generación del diagrama navegacional en el caso de estudio 1, se procede a convertir las clases en nodos de navegación y a la generación del nodo menú que será el punto de partida para la navegación dentro del sitio Web.

En la herramienta, para iniciar su desarrollo se hace clic en el cuadro “Diagrama navegacional” que se encuentra en la Figura 4.8. Este proceso abre la interfaz del lienzo de dicho diagrama que se muestra en la Figura 4.4. Para agregar un nodo al lienzo se selecciona el nodo correspondiente, posteriormente se hace clic en el lugar destino del lienzo y se finaliza con la edición del diagrama con su nombre como se muestra en la Figura 4.19 ilustradas anteriormente: A continuación, se muestra el diagrama navegacional en la Figura 4.28.

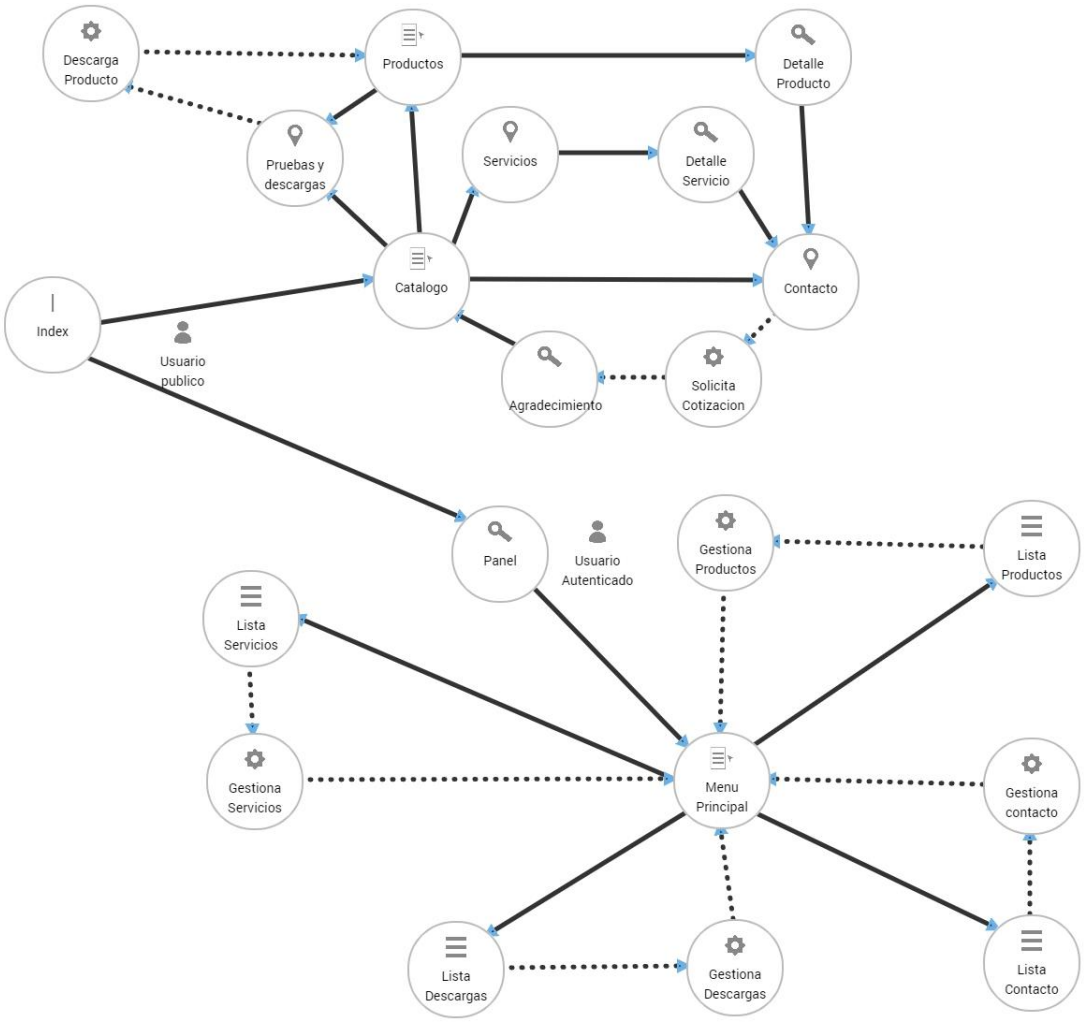


Figura 4.28 Diagrama navegacional del caso de estudio 2.

Posteriormente en la Figura 4.29 a través de la herramienta SODRA, se obtiene la notación de este diagrama se hace clic en el menú vista y se selecciona el elemento “ver notación” o se accede a la configuración de teclas ctrl + O.

```
(Index){NodoInicio}-[EnlaceNavegacional]->(Catalogo){NodoMenu};
(Index){NodoInicio}-[EnlaceNavegacional]->(Panel){NodoConsulta};
(Catalogo){NodoMenu}-[EnlaceNavegacional]->(Productos){NodoMenu};
(Catalogo){NodoMenu}-[EnlaceNavegacional]->(Servicios){NodoNavegacional};
(Catalogo){NodoMenu}-[EnlaceNavegacional]->(Pruebas y descargas){NodoNavegacional};
(Catalogo){NodoMenu}-[EnlaceNavegacional]->(Contacto){NodoNavegacional};
(Productos){NodoMenu}-[EnlaceNavegacional]->(Detalle Producto){NodoConsulta};
(Productos){NodoMenu}-[EnlaceNavegacional]->(Pruebas y descargas){NodoNavegacional};
(Servicios){NodoNavegacional}-[EnlaceNavegacional]->(Detalle Servicio){NodoConsulta};
(Pruebas y descargas){NodoNavegacional}-[EnlaceProceso]->(Descarga Producto){NodoProceso};
(Contacto){NodoNavegacional}-[EnlaceProceso]->(Solicita Cotizacion){NodoProceso};
(Solicita Cotizacion){NodoProceso}-[EnlaceProceso]->(Agradecimiento){NodoConsulta};
(Agradecimiento){NodoConsulta}-[EnlaceNavegacional]->(Catalogo){NodoMenu};
(Detalle Producto){NodoConsulta}-[EnlaceNavegacional]->(Contacto){NodoNavegacional};
(Detalle Servicio){NodoConsulta}-[EnlaceNavegacional]->(Contacto){NodoNavegacional};
(Panel){NodoConsulta}-[EnlaceNavegacional]->(Menu Principal){NodoMenu};
(Descarga Producto){NodoProceso}-[EnlaceProceso]->(Productos){NodoMenu};
(Menu Principal){NodoMenu}-[EnlaceNavegacional]->(Lista Productos){NodoLista};
(Menu Principal){NodoMenu}-[EnlaceNavegacional]->(Lista Servicios){NodoLista};
(Menu Principal){NodoMenu}-[EnlaceNavegacional]->(Lista Descargas){NodoLista};
(Menu Principal){NodoMenu}-[EnlaceNavegacional]->(Lista Contacto){NodoLista};
(Gestiona Productos){NodoProceso}-[EnlaceProceso]->(Menu Principal){NodoMenu};
(Gestiona Servicios){NodoProceso}-[EnlaceProceso]->(Menu Principal){NodoMenu};
(Gestiona Descargas){NodoProceso}-[EnlaceProceso]->(Menu Principal){NodoMenu};
(Gestiona contacto){NodoProceso}-[EnlaceProceso]->(Menu Principal){NodoMenu};
(Lista Productos){NodoLista}-[EnlaceProceso]->(Gestiona Productos){NodoProceso};
(Lista Servicios){NodoLista}-[EnlaceProceso]->(Gestiona Servicios){NodoProceso};
(Lista Descargas){NodoLista}-[EnlaceProceso]->(Gestiona Descargas){NodoProceso};
(Lista Contacto){NodoLista}-[EnlaceProceso]->(Gestiona contacto){NodoProceso};
```

Figura 4.29 Notación del diagrama navegacional del caso de estudio 2.

El resultado de diagrama se exporta siguiendo las configuraciones de XMI y JPG que se mostraron en la Figura 4.19. En la Figura 4.30 se muestran los archivos resultantes de la exportación del diagrama del cliente:

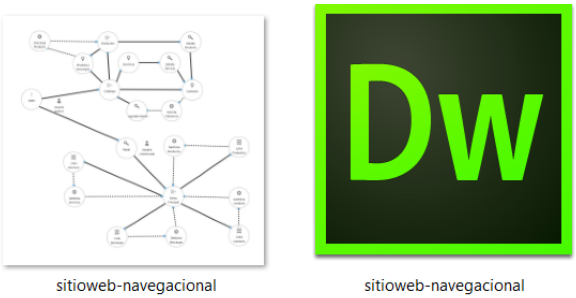


Figura 4.30 Archivos resultantes de la exportación del diagrama del cliente.

Con base en las necesidades de GSI Soluciones S.A. de C.V. se realizó la aplicación Web tomando como base el análisis de requerimientos que se realizó con la herramienta SODRA.

Después de generar los diagramas del cliente y navegacional para el segundo caso de estudio, se utilizaron las siguientes tecnologías para el desarrollo de la aplicación Web:

1. *PHP*.
2. *MySQL*.
3. *JavaScript*.
    a. Marco de trabajo *jQuery*.
    b. Marco de trabajo *Material Design* de *Google*.
4. Marco de trabajo Hadrón.

Con estas tecnologías se desarrolló la aplicación Web. Finalmente, en las Figuras 4.31, 4.32 y 4.33 se muestran algunas capturas de pantalla de la aplicación Web resultante.

Figura 4.31 Aplicación Web GSI Soluciones S.A. de C.V. versión de escritorio.

Figura 4.32 Aplicación Web GSI Soluciones S.A. de C.V. versión móvil.

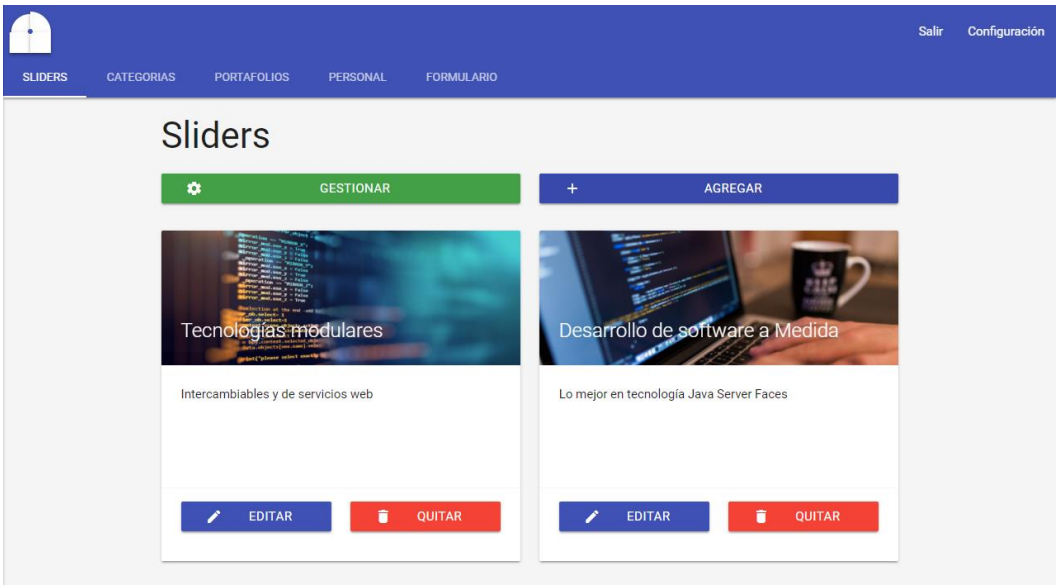


Figura 4.33 Aplicación móvil GSI Soluciones S.A. de C.V. versión panel de administración.

Capítulo 5. Conclusiones y recomendaciones

En este capítulo se muestran las conclusiones y recomendaciones para este trabajo de tesis

5.1. Conclusiones

En el desarrollo de este trabajo de tesis que corresponde a la etapa 2, se hicieron nuevas adiciones a lo que estableció en la etapa 1 como el nodo nota para plasmar el punto de vista del desarrollo de los diagramas del cliente y navegacional en la herramienta SODRA.

La generación de una herramienta Web para la creación de dichos diagramas y sus correspondientes notaciones representan la concretización de las ideas descritas en [1]. En el apartado de la generación de la notación propuesta en [1], se mostraron los ejemplos prácticos para cada caso de estudio. Dicha notación funge como un breve resumen de la organización y estructura de los diagramas del cliente y navegacional.

Aunado a ello y dado el creciente auge de herramientas Web que satisface cada vez más necesidades que se completaban en aplicaciones de escritorio, se otorgaron herramientas de guardado en el servicio de alojamiento en la nube de *Google Drive*. Con ello, la posibilidad de trabajo en equipo a través de la herramienta SODRA se facilita dados los mecanismos de sincronización nativos para este servicio de alojamiento de datos en *Google Drive*.

5.2. Trabajo futuro

Para el desarrollo de la etapa 3 de este proyecto de tesis, se sugiere utilizar un mecanismo, que con base en el diagrama del cliente y navegacional, genere tanto código en un lenguaje de servidor orientado a objetos (Java, PHP) y código del lado del cliente (HTML con JavaScript).

Además, se sugiere la implementación de una herramienta que permita la transformación de los modelos del cliente y navegacional a diagrama de clases y diagrama de UWE, o en otras palabras realizar la transformación de los archivos XMI de SODRA a los XMI de Visual Paradigm o Enterprise Architect. Esto con el propósito de brindar más herramientas que cumplan con el estándar internacional de modelación y hacer a la herramienta compatible con estos programas.

Productos Académicos

José Fernando Carreón Díaz de León, Silvestre Gustavo Peláez Camarena, Ulises Juárez Martínez, María Antonieta Abud Figueroa, Celia Romero Torres. "SODRA: Arquitectura de una herramienta para crear diagramas del modelo cliente y navegacional". VII Congreso Nacional de Ciencias de la Computación, CONACIC2017. ISBN: 978-607-525-473-9.

INDAUTOR
Instituto Nacional del Derecho de Autor

Para los efectos de los artículos 13, 162, 163 fracción 9, 164 fracción I, 168, 169, 209 fracción III y demás relativos de la Ley Federal del Derecho de Autor, se hace constar que la **OBRA** cuyas especificaciones aparecen a continuación, ha quedado inscrita en el Registro Público del Derecho de Autor, con los siguientes Datos:

Título: SODRA.

Rama: Software

Titular: José Fernando Carreón Díaz de León (50%)

Autore: Silvestre Gustavo Peláez Camarena (50%)

No. De registro: Continúa en trámite.

Bibliografía

[1] Peláez-Camarena, S. G., Gámez, C. D. L., Juárez-Martínez, U., Abud-Figueroa, M. A., & Torres, C. R. (2015). Propuesta de artefactos basados en una notación con grafos y conjuntos para el modelado conceptual de aplicaciones Web. Research in Computing Science, 107, 41-50.

[2] Balakrishnan, V. K. (1997-02-01). Graph Theory (1st ed.). McGraw-Hill. ISBN 0-07-005489-4.

[3] Gross, Jonathan L.; Yellen, Jay (2004). Handbook of graph theory. CRC Press. p. 35. ISBN 978-1-58488-090-5.

[4] Chein, Michel; Mugnier, Marie-Laure (2009). Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs. Springer. doi:10.1007/978-1-84800-286-9. ISBN 978-1-84800-285-2.

[5] Elmasri, R., Navathe, S. B., Castillo, V. C., Pérez, G. Z., & Espiga, B. G. (2002). Fundamentos de sistemas de bases de datos. Addison-Wesley.

[6] Rumbaugh, James, "Análisis y Diseño Orientado a Objetos", Segunda Edición, México, Editorial Addison Wesley Longman, 1996, 638 pp.

[7] B. M. Duc, Real-Time Object Uniform Design Methodology with UML. Springer Science & Business Media, 2007.

[8] T. Baar, A. Strohmeier, and A. Moreira, UML 2004 - The Unified Modeling Language: Modeling Languages and Applications. 7th International Conference, Lisbon, Portugal, October 11-15, 2004. Proceedings. Springer, 2004.

[9] A. Radoaca, "Venn Diagrams for Multisets," in 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2016, pp. 187–194.

[10] R. Moseley, Desarrollo de aplicaciones Web. Anaya Multimedia, 2007.

[11] Boulanger, T. (2015). XML práctico: Bases esenciales, conceptos y casos prácticos (2ª edición). Ediciones ENI.

[12] "XMI". [Online]. Available: http://www.omg.org/spec/XMI/ISO/19503/PDF/. [Accessed: 28-Apr-2017].

[13] "HTML Standard." [Online]. Available: https://html.spec.whatwg.org/#canvas. [Accessed:

25-Feb-2017].

[14] “Scalable Vector Graphics” [Online]. Available: https://www.w3.org/Graphics/SVG. [Accessed: 13-Oct-2017].

[15] C. Cosentino, Advanced PHP for Web Professionals. Prentice Hall Professional, 2003.

[16] "PHP: rfc:php7timeline." [Online]. Available: https://wiki.php.net/rfc/php7timeline. [Accessed: 25-Feb-2017].

[17] Urbieta, M. M. (2012). Metodología dirigida por modelos para el diseño de Funcionalidad Volátil en aplicaciones Web. Doctoral dissertation, Facultad de Informática, 25-34.

[18] "PhpStorm IDE :: JetBrains PhpStorm," JetBrains. [Online]. Available: https://www.jetbrains.com/phpstorm/. [Accessed: 04-Mar-2017].

[19] Y. Baghdadi, B. Alani, and Z. Al-Khanjari, “A CASE tool for Web Services: Towards development and management of service-oriented software,” in *2012 International Conference on Innovations in Information Technology (IIT)*, 2012, pp. 288–293.

[20] A. Hettab, E. Kerkouche, and A. Chaoui, “A Graph Transformation Approach for Automatic Test Cases Generation from UML Activity Diagrams,” in *Proceedings of the Eighth International C* Conference on Computer Science & Software Engineering*, New York, NY, USA, 2008, pp. 88–97.

[21] I. Melo, M. Sánchez, and J. Villalobos, “Composing Graphical Languages,” in *Proceedings of the First Workshop on the Globalization of Domain Specific Languages*, New York, NY, USA, 2013, pp. 12–17.

[22] I. J. Freeman and B. Plimmer, “Connector Semantics for Sketched Diagram Recognition,” in *Proceedings of the Eight Australasian Conference on User Interface - Volume 64*, Darlinghurst, Australia, Australia, 2007, pp. 71–78.

[23] D. Kundu, D. Samanta, and R. Mall, “Automatic code generation from unified modelling language sequence diagrams,” *IET Software*, vol. 7, no. 1, pp. 12–28, Feb. 2013.

[24] S. R. M. Canovas and C. E. Cugnasca, “Extending a Metamodel for Adaptive Programs: Specifying Adaptive Functions,” *IEEE Latin America Transactions*, vol. 14, no. 4, pp. 1923–1929, Apr. 2016.

[25] F. Schumacher, M. Holzer, T. Greiner, and W. Rosenstiel, “Modeling and code generation of recursive algorithms with extended UML Activity Diagrams,” in *Proceedings of 21st*

International Conference Radioelektronika 2011, 2011, pp. 1–4.

[26] M. Jiang, C. Zhou, F. Zhang, and S. Chen, "Interpretation, Transformation and Model Checking of Semi-formal Diagram Notations," in *2008 International Conference on Computer Science and Software Engineering*, 2008, vol. 2, pp. 263–266.

[27] H. Krishnan and P. Samuel, "Relative Extraction Methodology for class diagram generation using dependency graph," in *2010 INTERNATIONAL CONFERENCE ON COMMUNICATION CONTROL AND COMPUTING TECHNOLOGIES*, 2010, pp. 815–820.

[28] E. Mayfield, "Sentence Diagram Generation Using Dependency Parsing," in *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*, Stroudsburg, PA, USA, 2009, pp. 45–53.

[29] L. M. Segundo, R. R. Herrera, and K. Y. P. Herrera, "UML Sequence Diagram Generator System from Use Case Description Using Natural Language," in *Electronics, Robotics and Automotive Mechanics Conference (CERMA 2007)*, 2007, pp. 360–363.

[30] N. M. Franco, R. N. Fidalgo, E. A. Silva, T. F. Cacalcante, and P. H. S. Brito, "Modeling language and CASE tool for communication board customization," in *2014 IEEE 16th International Conference on e-Health Networking, Applications and Services (Healthcom)*, 2014, pp. 335–340.

[31] J. Choi, E. Jee, and D. H. Bae, "Toward Systematic Construction of Timing Diagrams from UML/MARTE Behavioral Models for Time-Triggered Embedded Software," in *2012 IEEE Sixth International Conference on Software Security and Reliability*, 2012, pp. 118–127.

[32] F. Hu, L. Fu, L. Liu, and G. Zhang, "A Soft PLC Ladder Diagram Edit Software Implemented Based on Table Technology," in *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, 2008, vol. 1, pp. 817–821.

[33] Y. Wautelet, S. Heng, M. Kolp, I. Mirbel, and S. Poelmans, "Building a rationale diagram for evaluating user story sets," in *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, 2016, pp. 1–12.

[34] Danielsson, P.E, "Euclidean distance mapping" in *Computer Graphics and image processing*, 1980, pp. 227–248.