Universidad de las Ciencias Informáticas

Sistema para la evaluación de la prevención, recuperación y vulnerabilidad de los sismos, realizada por la Consultoría Pro-Ambiente de la Empresa Nacional de Investigaciones Aplicadas

Trabajo de diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Autor: Jesús Rodríguez Herrera.

Tutora: MSc. Isyed de la Caridad Rodríguez Trujillo

La Habana

Junio, 2015

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor del trabajo “Sistema para la evaluación de la prevención, recuperación y vulnerabilidad de los sismos, realizada por la Consultoría Pro-Ambiente de la Empresa Nacional de Investigaciones Aplicadas” y autorizo a la Facultad 4 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Jesús Rodríguez Herrera

_____

Autor

Isyed de la Caridad Rodríguez Trujillo

_____

Tutor

AGRADECIMIENTOS

Ante todo dar gracias a mis padres y tutora que hicieron posible este momento, estoy agradecido de tener unos padres como ustedes. A mi familia que siempre me ha apoyado en las decisiones de mi vida y han estado presente para respaldarme en los momentos buenos y malos. Dar gracias a mis amigos que han sufrido y ayudado en este proceso tan importante en mi vida, gracias por estar ahí, a todos ustedes, Gracias.

RESUMEN

La presente investigación consiste en un sistema informático desarrollado mediante herramientas de software libre, con el objetivo de agilizar la realización de estudios sobre el peligro, vulnerabilidad y riesgo sísmico en escenarios físicos. El software facilita la recopilación y análisis de los datos de las diferentes estructuras ante la acción de un sismo y agiliza el trabajo llevado a cabo por los expertos de la unidad de Pinar del Río de la Consultoría Ambiental Pro-Ambiente de la Empresa Nacional de Investigaciones Aplicadas de Cuba.

La implementación del sistema está sustentada sobre la base teórica de la Metodología Peligro, Vulnerabilidad y Riesgo Sísmico desarrollada por el Centro Nacional de Investigaciones Sismológicas, CENAIS y guiada por la metodología de desarrollo de *software* AUP.

La aplicación desarrollada ofrece la posibilidad de evaluar cualitativamente y cuantitativamente la vulnerabilidad de las estructuras y calcular diferentes indicadores que permiten establecer un plan de contingencia y de prevención de daños. La presentación de los resultados obtenidos se ofrece a través de la generación de un informe, donde se detallan los elementos principales del estudio.

Palabras clave: *Software,* e*structura, riesgo símico, vulnerabilidad, prevención.*

ÍNDICE

INTRODUCCIÓN

Los principales fenómenos que azotan a nuestro país son los desastres naturales, entre ellos el primer puesto lo ocupan los ciclones tropicales ya que ocurren con mayor frecuencia; el segundo puesto lo ocupan los sismos que aunque no son muy fuertes tienen su potencia y ocurren periódicamente en distintas provincias del país. El Centro Nacional de Investigaciones Sismológicas en adelante CENAIS ha desarrollado una metodología denominada Peligro, Vulnerabilidad y Riesgo Sísmico en Escenarios Físicos (PVR); esta metodología permite saber cuan vulnerable es una estructura o un área al ocurrir este tipo de eventos naturales, de esta manera se pueden prevenir daños mayores, minimizar las pérdidas y evitar riesgos de tipos tecnológicos.

La dirección de Cuba le ha pedido a la Consultoría Ambiental Pro-Ambiente de la Empresa Nacional de Investigaciones Aplicadas (ENIA) adscrita al Ministerio de la Construcción (MICONS), debidamente acreditadas ante el Centro Inspección Control Ambiental (CICA) y amparadas por la Resolución 189/2012 y la Ley 81 del Ministerio de Ciencia, Tecnología y Medio Ambiente (CITMA), que la unidad de esta entidad ubicada en Pinar del Río, aplique la metodología PVR en dicha provincia. Hoy en día se emplean técnicas de construcción antisísmicas, es decir, que permiten que las construcciones sean más resistentes a estos eventos, pero la mayoría de las edificaciones de la provincia pinareña no son modernas por lo que es necesaria la realización de esta tarea, que posteriormente será extendida al resto de la isla.

Los especialistas encargados planean hacer todo el análisis de la provincia Pinar del Río para el año 2015, lo cual es un reto para ellos y se hace necesario informatizar este proceso de manera que se pueda avanzar más rápido en ver cuán seguras son las edificaciones en toda la provincia, tomando las medidas pertinentes. Lo anterior nos deja como principal problema la no existencia de un sistema informático capaz de unificar todas las herramientas empleadas por los especialistas y facilitar el desarrollo del proceso.

Analizando la situación actual de los sistemas respecto a la evaluación de la prevención, recuperación y vulnerabilidad de los sismos, se determina que:
Internacionalmente:

- Las aplicaciones desarrolladas presentan grandes divergencias en las metodologías usadas y la mayoría no presentan licencia gratuita, lo que repercute negativamente en el uso de las mismas para Cuba.

Nacionalmente:

- No existe variedad de productos informáticos y los existentes no cumplen con las expectativas de la Consultoría Ambiental Pro-Ambiente de la ENIA, en cuanto a metodología, funcionalidades y

que sean multiplataforma porque la empresa se encuentra en periodo de transición al software libre; por tal motivo el proceso regido por la metodología PVR se realiza de forma manual trayendo consigo:

- Lento y engorroso trabajo al realizar la evaluación de los sismos.
- Los pasos establecidos en la metodología PVR, se realizan mediante planillas que el especialista va rellenando, lo cual puede provocar pérdida de información y errores.
- Se utilizan diversas e independientes herramientas para recopilar la información de la estructura o del área a la que se le realiza el proceso de análisis, así como el mapeo de la zona en donde se esté trabajando.

- No existe almacenamiento automatizado para la evaluación de la prevención, recuperación y vulnerabilidad de los sismos, imposibilitando la reutilización de la información, el control y el análisis estadístico.

Lo anteriormente expuesto, constituye la situación problemática identificada que fundamentó la investigación, cuyo **problema científico** radica en: ¿Cómo contribuir al proceso de utilización de la metodología PVR de sismos, de forma tal que facilite el empleo de la misma para los especialistas?

Como **objeto de estudio** de la investigación se define: los sistemas informáticos para evaluación de la prevención, recuperación y vulnerabilidad de desastres naturales.

El **campo de acción** se enmarca en los sistemas informáticos para evaluación de la prevención, recuperación y vulnerabilidad de los sismos.

Se define como **objetivo general** de la investigación, desarrollar un sistema informático para la evaluación de la prevención, vulnerabilidad y recuperación de los sismos, realizada por la Consultoría Pro-Ambiente Empresa Nacional de Investigaciones Aplicadas.

De una manera más detallada se plantean los siguientes **objetivos específicos**:

1. Elaborar el marco teórico conceptual partiendo del estudio de las tendencias, tecnologías y metodologías más utilizadas en el desarrollo de este tipo de sistemas, que permita dar solución al problema planteado.
2. Diseñar un sistema para la evaluación de la prevención, vulnerabilidad y recuperación de los sismos.
3. Implementar un sistema para la evaluación de la prevención, vulnerabilidad y recuperación de los sismos.
4. Realizar pruebas al sistema para comprobar su correcto funcionamiento.

En correspondencia con lo planteado anteriormente se formula la siguiente **hipótesis**: La informatización del proceso para la evaluación de la prevención, recuperación y vulnerabilidad de los sismos llevado a cabo por la Consultoría Ambiental Pro-Ambiente de la Empresa Nacional de Investigaciones Aplicadas facilita el cumplimiento de la metodología *Peligro, Vulnerabilidad y Riesgo Sísmico en Escenarios Físicos,* en cuanto los cálculos realizados, empleo de la misma e integración de las herramientas utilizadas en la elaboración del proceso.

Para llevar a cabo la investigación se utilizan los **métodos científicos** detallados a continuación:

Métodos Teóricos:

- El método histórico-lógico se aplica en la investigación para conocer las diferentes herramientas que emplean la metodología PVR de sismos.
- El método analítico-sintético se utiliza para el análisis de la teoría y extracción de los principales conceptos a incluir en el marco teórico, además para el estudio y análisis de la información, lo que permite obtener los elementos principales relacionados con la metodología PVR de sismos mediante herramientas informáticas.

Métodos Empíricos:

- Se utiliza la observación en el procedimiento de recolección de datos e información, utilizando los sentidos para observar los hechos a la ocurrencia de sismos.
- Se utiliza un cuestionario para enfocar la necesidad del cliente y así trazar el curso de la investigación.

Para cumplir con lo expuesto anteriormente se citan un conjunto de **tareas investigativas**:

1. Realizar un análisis bibliográfico sobre la necesidad del sistema y el estado actual de la evaluación de los sismos en Cuba y en el resto del mundo, para de esta manera conformar la base teórica metodológica de la investigación.
2. Plantear los conceptos y terminologías fundamentales relacionadas con la evaluación de la prevención, recuperación y vulnerabilidad de los sismos, para lograr unificar criterios y evitar interpretaciones equivocadas.
3. Analizar, partiendo de la investigación previa, soluciones similares relacionadas con la evaluación de los sismos para obtener características afines y necesarias de utilizar.
4. Analizar la metodología PVR que guiará el procedimiento de la herramienta propuesta.
5. Valorar las metodologías de desarrollo de software y las herramientas que conformarán el ambiente en que se desarrollará la propuesta de solución.

6. Definir las funcionalidades apoyándose en la metodología PVR, así como las características no funcionales del sistema, que garanticen una herramienta que cumpla con las expectativas del cliente.
7. Elaborar los artefactos de diseño e implementación según la metodología de desarrollo de software.
8. Implementar la herramienta usando las técnicas apropiadas de programación para facilitar la lectura, comprensión y mantenimiento del código.
9. Realizar pruebas unitarias y de aceptación.

Estructura capitular

Capítulo 1 "Fundamentación teórica": Se expondrán los principales conceptos relacionados con el tema de investigación. Se analizarán las soluciones similares existentes en Cuba y el resto del mundo para obtener características comunes y necesarias de incorporar. Se explicará claramente la metodología PVR que guiará el proceso de evaluación. Por último se realizará un estudio de la metodología y herramientas necesarias para el desarrollo de la solución.

Capítulo 2 "Descripción de la solución propuesta": Se identificarán las funcionalidades y características del sistema. Se elaborarán los artefactos correspondientes en base a la metodología de desarrollo de software seleccionada utilizando las herramientas tecnológicas necesarias.

Capítulo 3 "Implementación y prueba": Se realizará la implementación de la solución propuesta y se generará los artefactos correspondientes a esta etapa. Por último se realizarán pruebas al software desarrollado con el objetivo de entregar un producto conforme a los requerimientos del cliente.

Capítulo 1 FUNDAMENTACIÓN TEÓRICA

En este capítulo se mencionan algunos conceptos para el entendimiento de aquellas palabras técnicas sobre la cuales no se tiene dominio. Se realiza un análisis de soluciones similares para tener idea de cómo están conformados estos tipos de software. Se expone la metodología PVR necesaria para la elaboración del sistema. Se estudian dos grupos de metodologías de desarrollo de software para seleccionar la que guiará el proceso de construcción de la aplicación. Se exponen las tecnologías necesarias y disponibles para la implementación.

1.1 Conceptos

Prevención

Conjunto de medidas cuyo objeto es impedir o evitar que sucesos naturales o generados por el hombre causen desastres. Estas acciones responden a la efectividad del cumplimiento de la legislación en lo que respecta a la Planificación urbana y física, así como la intervención directa del fenómeno (Rivera, 2008).

Actividades diseñadas para proveer protección permanente de un desastre. Incluye ingeniería y otras medidas de protección física, así como medidas legislativas para el control del uso de la tierra y ordenación urbana (William, 2002).

Como conclusión, la prevención es estar preparado para uno o varios eventos negativos, para que a la ocurrencia de estos los resultados negativos sean los menos posibles y poder volver a la normalidad, lo más rápido posible, una vez transcurrido el evento siguiendo los pasos pertinentes.

Reconstrucción

Es el proceso de recuperación a mediano y largo plazo, del daño físico, social y económico, nivel de desarrollo igual o superior al existente antes del desastre. Los efectos de un desastre repercuten tanto social, económica como ambientalmente. Por ello las acciones en reconstrucción buscan activar las fuentes de trabajo, reactivar la actividad económica de la zona o región afectada; reparar los daños materiales en especial en materia de vivienda y de infraestructura, incorporar las medidas de prevención y mitigación del riesgo en el proceso de desarrollo (Rivera, 2008).

Acciones tomadas para restablecer una comunidad después de un periodo de rehabilitación, subsecuente a un desastre. Las acciones incluirían construcción de viviendas permanentes, restauración total de todos los servicios y reanudar por completo el estado de pre-desastre (William, 2002).

Constituye un momento de transición entre la emergencia y un estado nuevo, se realiza en una primera instancia con la reorganización del entorno y el desarrollo de la economía, una vez superada la emergencia, evitando a la vez el encadenamiento de otras calamidades (William, 2002).

Por tanto la reconstrucción es restablecer las funcionalidades de una localidad o estructura después del paso de un desastre natural o provocado por el hombre evitando el desencadenamiento de otros males.

Vulnerabilidad

Entendida como la reducción de la capacidad a “acomodarse” a determinadas circunstancias. Dicha vulnerabilidad ha sido definida de diferentes maneras, entre las que se puede citar la siguiente: Grado de pérdida de un elemento o grupo de elementos en riesgo como resultado de la probable ocurrencia de un suceso desastroso, expresada en una escala desde 0 (sin daño) a 1 (pérdida total) (Sandy, 1983).

Es la probabilidad de ocurrencia de un suceso potencialmente desastroso durante cierto período de tiempo en un sitio dado (Sandy, 1983).

Se concluye que la vulnerabilidad no es más que la posibilidad de que cierto grupo pueda tener pérdidas materiales y daños a la salud ante la ocurrencia de algún evento de condiciones adversas en un sitio durante cierto periodo de tiempo.

Riesgo

Número esperado de pérdidas humanas, personas heridas, propiedad dañada e interrupción de actividades económicas debido fenómenos naturales particulares y por consiguiente, el producto de riesgos específicos y elementos de riesgo (William, 2002).

Probabilidad de que ocurra un desastre. Situación frente a un peligro. Condición real frente a una determinada amenaza (William, 2002).

Se puede concluir que riesgo no es más que la posibilidad de que ocurra una amenaza en contra del bienestar de una localidad o grupo de personas.

1.2 Soluciones similares

Carpa: Herramienta de software modular y de libre acceso para la evaluación del riesgo probabilístico de desastres naturales. Esta herramienta está fundamentada en un Sistema de Información Geográfica que permite realizar cálculos de riesgo en función de los datos de amenaza, exposición y vulnerabilidad física, a través de herramientas para el mapeo de amenaza, evaluación de riesgo y análisis costo-beneficio de los terremotos, maremotos, huracanes, inundaciones, deslizamientos y amenazas de erupciones volcánicas (CARPA, 2013).

Ventajas:

- Es modular.
- Modela la frecuencia e intensidad de varios tipos de desastres.

Desventajas:

- Se visualiza sobre las imágenes BlueMarble® de la NASA®.
- Necesita conexión a internet.
- Necesita la biblioteca jogl de Java OpenGL.

Sahana: Es una herramienta informática que permite solucionar los problemas más comunes de coordinación en momentos de desastre. Es una plataforma de software libre cuyo trabajo se basa, principalmente, en la gestión de desastres. Cuenta con varias funcionalidades básicamente se centra en saber quién está pidiendo qué y quién está ofreciendo qué (Howden, 2013).

Ventajas:

- Permite tener una fotografía mucho más real del impacto y de la ayuda.
- Las personas están informados en todo momento durante la catástrofe.

Desventajas:

- Necesita estar conectado para acceder a la información.
- Su especialidad son los tsunamis.

Earth Alerts: Es una herramienta desarrollada por la empresa South Wind Technologies que ofrece datos y avisos sobre terremotos, tsunamis, volcanes en erupción y ciclones alrededor del globo terráqueo. Usando fuentes de información oficiales a través de Internet, Earth Alerts te mantendrá al día de los últimos accidentes naturales incluyendo imágenes y mapas por satélite, noticias relacionadas y el nivel de alerta (López, 2014).

Ventajas:

- Información abundante y de fuentes fiables.
- Incluye mapas y fotografías.
- Búsqueda por zonas concretas.
- Es gratuita.

Desventajas:

- Necesita estar conectado a internet.
- Los datos se descargan de manera lenta.
- Los datos se muestran en inglés.

- No es código abierto.

Earth browser: Fue desarrollada por Earthbrowser para que puedas mantenerte informado, cuenta con unos excelentes gráficos, de todo lo que pasa por el mundo en lo referente a clima, pronósticos climáticos, terremotos, volcanes, cámaras web, entre otros. El programa contiene 650 ciudades de todo el mundo, pero puedes añadir cualquier zona que desees de manera muy sencilla (Softonic, 2014).

Ventajas:

- Gran volumen de información y de organismos certificados.
- Utiliza mapas, gráficas, videos y fotografías.
- Rapidez en la descarga de datos.
- Es multilenguaje.

Desventajas:

- Necesita estar conectado a internet.
- No es código abierto.
- Tienes que pagar para acceder a sus datos.
- Necesita Adobe Air que es otra herramienta privativa.

Earthquake 3D: Es una herramienta desarrollada por Starfield-screen-saver muy eficaz y muy interesante al mostrar los datos casi en tiempo real gracias a la información suministrada por USGS (Inspección Geológica de los Estados Unidos) vía Internet (Navarro, 2014).

Ventajas:

- Ver el planeta en 3D, girar y dar zoom.
- Permite generar reportes.
- Cuenta con distintas vistas combínales como:
    - Nocturnidad.
    - Ángulos y capas.
- Es gratuito.

Desventajas:

- Necesita estar conectado a internet.
- No es código abierto.
- Depende de la información de USGS.
- Problemas con el diseño gráfico.

Quakeshakes: Es una herramienta desarrollada por Bricbrac Software para estar al tanto de los eventos sísmicos. El software muestra los terremotos más recientes y ofrece información extra de los mismos a través de puntos sobre un mapa físico (Ferri-Benedetti, 2014).

Ventajas:

- Muestra terremotos recientes en el mapa.
- Capas con datos demográficos y geográficos.

Desventajas:

- Hay que pagar licencia.
- No es de código abierto.
- Sonidos y animaciones innecesarias.
- La navegación por el mapa es complicada.

1.2.1 Contribuciones al sistema a partir del estudio de soluciones homólogas realizadas

A partir de búsqueda bibliográfica y entrevistas con especialistas y directivos de la unidad consultora de Pinar del Río, se considera que el tema ha sido poco explotado en Cuba, por lo que no se encontró ningún sistema de este tipo en el país, de ahí la necesidad de este trabajo, ya que internacionalmente las aplicaciones desarrolladas no son de código abierto y no utilizan la metodología PVR debido a que esta es una metodología cubana. Sin embargo el estudio permitió identificar algunas funcionalidades de este tipo de sistemas como:

- Representar en un mapa la ubicación de las casas analizadas.
- Realizar un reporte.

1.3 Metodología PVR

PVR es una metodología elaborada por el CENAIS para la realización de los estudios de peligro, vulnerabilidad y riesgo de desastres por sismos. La misma es aplicable a estudios de Peligro Vulnerabilidad y Riesgo Sísmico Local (PVR-SL) en Escenarios Físicos para esto se analizan varios factores que a continuación se exponen:

1.3.1 Peligro Sísmico Local

Para la determinación del peligro sísmico local se valora el comportamiento y distribución de intensidades producidas por sismos perceptibles recientes y los efectos de terremotos fuertes en el escenario. La precisión del Peligro Sísmico Local del Escenario se realiza utilizando de base, las condiciones geólogo – litológicas de su entorno, los criterios de la NC 46:1999, los valores promedio pesados de estimados regionales y los criterios de árbol lógico en términos de aceleración pico del terreno, según las características del Escenario (CENAIS, 2009).

1.3.2 Estudios de Vulnerabilidad Sísmica

La Vulnerabilidad Sísmica es la susceptibilidad de un Escenario, sistema o elemento expuesto, a sufrir daños bajo la acción de un fenómeno peligroso o perturbador de una energía determinada. Puede expresarse desde el punto de vista matemático como un número entre cero (0) y uno (1). Esto implica que un valor 0 para un evento de determinada intensidad los daños son nulos y 1 los daños son totales, por tanto la vulnerabilidad puede expresarse mediante una función matemática o por las llamadas matrices de vulnerabilidad. La vulnerabilidad total resulta de la suma de todas las vulnerabilidades calculadas de forma independiente (CENAIS, 2009):

$V = Ve + Vne + Vf + Vs + Vecn$

Donde:

Ve: vulnerabilidad estructural.

Vne: vulnerabilidad no estructural.

Vf: vulnerabilidad funcional.

Vs: vulnerabilidad social.

Vecn: vulnerabilidad económica.

Se expresan los indicadores de vulnerabilidad con números enteros entre 0 – 100, siendo 100 el caso de mayor vulnerabilidad. El resultado final se dividirá entre 100 para ajustarse a los intervalos de vulnerabilidad establecidos ($0 \leq V \leq 1$). Los intervalos de vulnerabilidad se muestran en la tabla 1 (CENAIS, 2009).

Intervalos de Vulnerabilidad	
Intervalos	**Vulnerabilidad**
$V \leq 0.25$	Baja
$0.26 \leq V \leq 0.50$	Media
$0.51 \leq V \leq 1.00$	Alta

Tabla 1. Intervalos de vulnerabilidad.

1.3.3 Vulnerabilidad estructural

Se puede evaluar la vulnerabilidad estructural del nivel de detalle que se desee, se calcula la vulnerabilidad estructural como (CENAIS, 2009):

$Ve = 20*(V_{fh}) + 10*(V_{ei}) + 10*(V_P)$

Donde:

V_{fh}: Vulnerabilidad Fondo Habitacional.

V_{ei}: Vulnerabilidad Edificaciones Importancia.

V_P: Vulnerabilidad Puentes.

1.3.3.1 Vulnerabilidad fondo habitacional

Esta metodología se aplica a un gran número de construcciones de determinada tipología constructiva en un escenario, se emplean matrices de vulnerabilidad de las escalas macrosísmicas adaptadas a las condiciones de nuestro país. El CENAIS utiliza la Escala Macrosísmica Europea (EMS) de 1998, la cual clasifica los daños en las edificaciones teniendo en cuenta factores como el tipo constructivo, estado técnico y problemas de configuración estructural. La definición de diversas tipologías estructurales se basa fundamentalmente en el material predominante (hormigón armado, acero, mampostería o madera); el sistema resistente (pórticos o muros); la altura y la fecha de construcción.

La tabla 2 que se presenta continuación, permite categorizar la resistencia de las estructuras en una forma sencilla, tomando en consideración el tipo de estructura y otros factores. Cuando se le asigna la clase de vulnerabilidad a una estructura o a un grupo de estructuras, el examen del tipo de construcción permite encontrar la fila correcta de la tabla de vulnerabilidad. La decisión de cual clase deberá ser asignada depende de asociar los rasgos descritos arriba a los símbolos mostrados en el rango de clases posibles en la tabla, el círculo muestra la clase más probable, si no existen deterioros apreciables o reforzamiento esta es la clase que se debe asignar. Una línea continua muestra el rango probable hacia arriba o hacia abajo, una línea discontinua muestra los edificios cuando presenten deterioros o reforzamiento (CENAIS, 2009).

Tipo de estructura	Clase de vulnerabilidad					
	A	B	C	D	E	F
Adobe						
Mampostería no reforzada						
Mampostería no reforzada con pisos concreto reforzado						
Reforzada o confinada						
Pórticos hormigón armado sin diseño sismorresistente (DSR)						
Pórticos hormigón armado con nivel moderado de DSR						
Pórticos hormigón armado con alto nivel de DSR						
Muros hormigón armado sin DSR						
Muros hormigón armado con moderado nivel de DSR						
Muros hormigón armado con alto nivel de DSR						
Estructura de acero						
Estructuras de madera						

Tabla 2. Clases de vulnerabilidad.

Grado de daños	Vulnerabilidad	
Sin daño el 100% de las edificaciones	No vulnerable	0.0
Daño ligero (DL) hasta el 20% de las edificaciones	Baja vulnerabilidad	0.2
Daño moderado (DM) hasta el 40%	Moderada vulnerabilidad	0.4
Daño severo (DS) hasta el 60%	Alta vulnerabilidad	0.6
Daño completo (DC) hasta el 80%	Muy alta vulnerabilidad	0.8
Destrucción (DT) el 100%	Extrema vulnerabilidad	1.0

Tabla 3. Puntaje de vulnerabilidad sísmica para fondo habitacional.

1.3.3.2 Edificaciones de importancia

El propósito de la evaluación sísmica de las instalaciones importantes, las cuales son las que brindan los servicios básicos a la población luego de un desastre o ponen en peligro la vida de las personas, es clarificar si la función de estas instalaciones se conservará en el caso de que ocurra un terremoto, la metodología que se propone se modificó y adaptó a partir del sondeo visual rápido de la FEMA – 154. Esta metodología no requiere cálculo estructural y se basa en un sistema de puntuaciones según una plantilla, el cual se compara con un umbral igual a 2 (CENAIS, 2009).

Evaluación detallada (CENAIS, 2009):
Requiere (S<2)
No requiere (S>2)
Cuando las edificaciones no requieren una evaluación detallada se consideran seguras

Evaluación detallada	Vulnerabilidad	
Requiere	Alta	1.0
No requiere	Baja	0.0

Tabla 4. Puntaje de vulnerabilidad para edificaciones de importancia.

1.3.3.3 Puentes

Para estimar la vulnerabilidad se adopta un modelo estadístico basado en experiencias en Japón, debido a que no hay registros de información sobre el colapso de puentes por sismos en Cuba. Debe señalarse que si se estima que algún puente colapsa, se debe realizar un estudio sísmico detallado. El criterio para la estimación de los daños sísmicos de los puentes está basado en el método propuesto por Tsuneo Katayana y se tienen en cuenta aspectos relacionados con el tipo de terreno, licuación, características y

materiales estructurales del puente. En la tabla 9 se muestra el sistema de puntaje que se le asigna a cada aspecto del puente analizado (CENAIS, 2009).

Donde:

La longitud de apoyo se calcula según la (EMNDC, 1999). Construcciones sismo resistentes. Requisitos básicos para el diseño y construcción.

N = 203+1.67L+6.66H (mm)

L: longitud en metros del extremo del puente hasta la junta de expansión adyacente o hasta el extremo del puente.

H: altura promedio en metros de las columnas de apoyo del tablero hasta la próxima junta de expansión (para los estribos).

H: altura de la columna o pila en metros (para pilas).
Para puentes de una luz H = 0.

S = (Tipo de terreno)* (Licuación)* (Tipo de viga)* (Dispositivo de apoyo)* (Altura máxima de estribo o pilar)* (Número de tramos)* (Longitud de apoyo)* (Aceleración del terreno)* (Tipo de cimentación)* (Material de columnas de pilas o estribos abiertos).

Donde:

Aspecto	Categoría	Puntuación
Tipo de terreno	S1	0.50
	S2	1.00
	S3	1.50
	S4	1.80
Licuación	Ninguna	1.00
	Probable	2.00
Tipo de viga	Arco	1.00
	Continuo	2.00
	Simple	3.10
	Gelber	3.10
Dispositivo de apoyo	Si	0.60
	No	1.15
Altura máxima estribo	< 5 m	1.00
	5 – 10 m	INTERPOLAR

	> 10m	1.00
Número de tramos	1 tramo	1.70
	más de 1 tramo	1.75
Longitud de apoyo	Cumple longitud de apoyo	0.80
	No cumple longitud de apoyo	1.20
	Soporte Gelber	0.80
	En pilar	1.20
Aceleración del terreno	Zona 0	0.80
	Zona 1	1.00
	Zona 2	1.70
	Zona 3	2.40
Tipo de cimentación	Garantizando empale	1.00
	Sin empale	1.40
Material de columnas de pilas o estribos abiertos	Ladrillo o concreto simple	1.40
	Otros	1.00

Tabla 5. Puntaje para la evaluación de puentes.

Resistencia Sísmica	**Puntuación Total**	**Vulnerabilidad**
Alta resistencia	S < 26	0.0
Moderada resistencia	26 ≤ S <30	0.6
Inadecuada resistencia	S ≥ 30	1.0

Tabla 6. Puntaje de vulnerabilidad para puentes.

1.3.4 Vulnerabilidad no estructural

Un estudio de vulnerabilidad no estructural busca determinar la susceptibilidad a daños que presentan las líneas vitales del área de estudio, tales como: redes de acueducto, alcantarillado, eléctricas, de comunicaciones, instalaciones que almacenan sustancias peligrosas y carreteras. Se realiza una inspección donde se detalla el nivel de daño que pueden sufrir cada uno de los factores que inciden en la vulnerabilidad no estructural (CENAIS, 2009).

Vne = 10*(Acueducto y alcantarillado) + 10*(Redes Eléctricas y Comunicaciones) + 3*(Tanques de líquidos peligrosos) +3*(Tanques de gases peligrosos) +4*(Tanques de gases tóxicos)

Donde:

Aspecto	Indicador	Puntuación
Acueducto y alcantarillado	Sin daños	0.0
	Daños ligeros	0.1
	Daños moderados	0.2
	Daños considerables	0.6
	Daños totales	1.0
Redes Eléctricas y Comunicaciones	Ndp = 0	0
	0 < Ndp ≤ 3	0.5
	Ndp > 3	1
Instalaciones peligrosas		
Tanques de líquidos peligrosos	0.00 – 0.15	0.15
	0.15 – 0.45	0.45
	0.45 – 1.00	1.0
Tanques de gases peligrosos	0.00 – 0.05	0.05
	0.05 – 0.10	0.10
	> 0.10	1.00
Tanques de gases tóxicos	> 0.01	1.00

Tabla 7. Puntaje de la vulnerabilidad no estructural.

1.3.4.1 Redes de acueducto y alcantarillado

Un método estadístico para la estimación de daños de los acueductos principales de cada ciudad, son solamente aplicables cuando la información sobre los materiales, diámetros y longitudes son conocidas. Se asume que cuando no se cuente con esta información de las tuberías, la vulnerabilidad es igual a 1.0. Las suposiciones de esta metodología son (CENAIS, 2009):

Se estima solamente la ruptura y separación en las juntas de tuberías y no los daños causados por derrumbe o colapso de edificaciones.

El método está basado en las experiencias previas de sismos ocurridos en otros países.
Se estima el daño de tuberías (Nd) como: $Nd = C_1*C_2*C_3*R_1*L$.

Donde:

R_1: relación del daño estándar (puntos de daño /km) $R_1 = 2.24*10^{-3}(PGV-20)^{1.51}$

PGV es la velocidad máxima del terreno (cm/seg).

L: Longitud total de tuberías (km).

C_1: Factor de corrección por licuación.

C_2: Factor de corrección por tipo de material.

C_3: Factor de corrección por diámetro de tuberías.

Valor del Coeficiente C_1	
Potencia de licuación (PL)	**C_1**
PL = 0	1.0
0 <PL ≤5	1.2
5 <PL ≤15	1.5
PL > 15	3.0

Tabla 8. Valor del coeficiente C_1.

Valores de los Coeficientes C_2 y C_3			
Material	**C_2**	**C_3(mm)**	**Valor**
Hierro fundido dúctil	0.30	d < 75	2.0
		100 < d <450	1.0
		00 < d < 900	0.3
		d > 1000	0.15
Hierro fundido	1.00	d < 75	1.70
		100 <d < 250	1.20
		300 <d < 900	0.40
		d > 1000	0.15
Acero soldado	1.50	d < 75	2.80
		100 < d < 250	1.40
		d > 300	0.80
Cloro-etileno	1.50	d < 75	1.00
		d > 100	0.80
Asbestos	3.0	d < 75	2.30
		100 < d < 250	0.90
		d > 300	0.40

Tabla 9. Valor de los coeficientes C_2 y C_3.

Es necesario destacar que cuando no se cuente con datos del material y longitud de las tuberías la vulnerabilidad será igual a 1.

Puntaje de vulnerabilidad sísmica de redes de acueducto y alcantarillado

Descripción del daño	Daño en tuberías (Nd)	Vulnerabilidad
Sin daños	Nd = 0	0.0
Daños ligeros	0.01 ≤ Nd ≤ 0.10	0.1
Daños moderados	0.10 < Nd ≤ 0.21	0.2
Daños considerables	0.21 < Nd ≤ 0.37	0.6
Daños totales	Nd > 0.37	1.0

Tabla 10. Puntaje de vulnerabilidad sísmica de redes de acueducto y alcantarillado.

1.3.4.2 Redes de energía eléctrica y comunicaciones

Para estimar los daños que produciría un evento sísmico en las Redes de energía eléctrica y comunicaciones se analizan los postes eléctricos o de comunicaciones, el daño sísmico de un poste se define como el número de postes colapsados $Ndp = C_1 N\ R/100$ (CENAIS, 2009).

Donde:

N: número total de postes.

C1: factor de corrección por licuación.

Relación del daño.

Valor del coeficiente C_1	
Potencia de Licuación	**C_1**
PL = 0	1.0
0 <PL ≤5	1.1
5 <PL ≤15	1.3
PL > 15	2.1

Tabla 11. Valor del Coeficiente C_1

Relación del Daño	
Intensidad	**R (%)**
≤ 8	0.00
> 8	0.55

Tabla 12. Relación del daño

En la Tabla 13 se muestra el puntaje para la estimación de la vulnerabilidad sísmica, asumiendo que el daño a postes eléctricos significa el colapso. Es necesario destacar que cuando no se cuente con datos del material y longitud de las tuberías la vulnerabilidad será igual a 1 (CENAIS, 2009).

Daño en postes (Ndp)	Vulnerabilidad
Ndp = 0	0.0
0 < Ndp ≤ 3	0.5
Ndp > 3	1.0

Tabla 13. Puntaje de vulnerabilidad sísmica de redes de energía eléctrica y comunicaciones.

Durante un terremoto, las instalaciones peligrosas que sufren daños pueden generar serias afectaciones secundarias por incendios, explosiones y escape de sustancias tóxicas. Antes de que ocurra el evento sísmico, las instalaciones débiles que almacenen productos peligrosos y tóxicos tienen que ser definidas para mitigar los daños secundarios. En esta metodología las funciones de daños para instalaciones peligrosas se obtienen a partir de la correlación obtenida de los datos de terremotos pasados en otros países en términos de aceleración máxima del terreno y los daños identificados para cada categoría de instalaciones peligrosas (CENAIS, 2009).

Instalación	Tipo de daño	Aceleración (% g)					
		0.10	0.15	0.20	0.25	0.30	0.35
Tanques grandes de almacenamiento de líquidos inflamable	Derrames pequeños del tanque o las juntas	0.00	0.01	0.05	0.14	0.33	0.69
	Derrames continuos	0.00	0.00	0.01	0.03	0.08	0.17
Tanques y contenedores de gases inflamables	Fuga del gas por la junta de tuberías	0.00	0.00	0.01	0.02	0.06	0.11
	Fuga continua, peligro de exposición	0.00	0.00	0.00	0.00	0.01	0.03
Tanques gases tóxicos/ nitrógeno líquido	Fuga por la junta de tuberías o tanques	0.00	0.00	0.00	0.00	0.01	0.02
	Fuga continua	0.00	0.00	0.00	0.00	0.00	0.00

Tabla 14. Probabilidad de ocurrencia de daño en instalaciones peligrosas en (%).

Como se puede observar en la tabla anterior la probabilidad de que ocurran escapes de gases o líquidos en las instalaciones de almacenamiento es muy pequeña para aceleraciones bajas del terreno, en la tabla siguiente se muestra el puntaje para la estimación de la vulnerabilidad a partir de la probabilidad de daños (CENAIS, 2009).

Instalación	Probabilidad (%)	Vulnerabilidad
Tanques grandes de almacenamiento de líquidos inflamable	0.00-0.15	0.15
	0.15-0.45	0.45
	0.45-1.00	1.00
Tanques y contenedores de gases inflamable	0.00-0.05	0.05
	0.05-0.10	0.10
	> 0.10	1.00
Tanques de gases tóxicos/nitrógeno líquido	> 0.01	1.00

Tabla 15. Puntaje de vulnerabilidad sísmica de instalaciones peligrosas.

1.3.5 Vulnerabilidad funcional

Realización de un análisis administrativo – organizativo a partir de los resultados de la vulnerabilidad estructural y no estructural que permita definir la funcionabilidad o paralización de los servicios básicos ante un evento sísmico. En la tabla 20 se muestra el puntaje asignado a cada uno de los indicadores que se tienen en cuenta en el análisis de la vulnerabilidad funcional (CENAIS, 2009).

Vf = 4*(Preparación del sistema de salud) + 4*(Disponibilidad de energía) + 2*(Suministros básicos)

Donde:

Aspecto	Indicador	Puntuación
Preparación del sistema de salud	Preparado	0
	Moderadamente preparado	0.5
	Sin preparación	1.0
Disponibilidad de energía	100 % de disponibilidad	0
	50 % de disponibilidad	0.5
	Sin disponibilidad	1.0
Suministros básicos	100 % garantizados	0
	50 % garantizados	0.5
	0 % garantizados	1.0

Tabla 16. Puntaje de la vulnerabilidad funcional.

1.3.6 Vulnerabilidad social

Se evaluará el total de población expuesta, personas sin viviendas, sin asistencia médica, personas afectadas por sustancias peligrosas o incomunicadas. Se evalúa también la percepción del riesgo y el grado de preparación de la población, además el conocimiento, posibilidades y disponibilidad de actuación de los decisores ante los efectos extremos. En la tabla 17 se muestra el puntaje asignado a cada uno de los indicadores que se tienen en cuenta en el análisis de la vulnerabilidad social (CENAIS, 2009).

Vs = 6*(Densidad de población afectada) + 2*(Percepción del riesgo por la población) + 2*(Capacitación de la población)

Donde:

Aspecto	**Indicador**	**Puntuación**
Densidad de población afectada	0.10 – 0.25	0.1
	0.26 – 0.50	0.3
	0.51 – 0.75	0.6
	0.76 – 1.00	0.8
	> 1.00	1.0
Percepción del riesgo por la población	100 % con percepción	0
	50 % con percepción	0.5
	Sin percepción	1.0
Capacitación de la población	100 % preparados	0
	50 % preparados	0.5
	Sin preparación	1.0

Tabla 17. Puntaje de la vulnerabilidad social.

1.3.7 Vulnerabilidad económica

Se evaluarán los factores relacionados con las pérdidas de zonas industriales, zonas turísticas, así la incapacidad de producción o transportación de producciones importantes entre otras. En la Tabla 18 se muestra el puntaje asignado a cada uno de los indicadores que se tienen en cuenta en el análisis de la vulnerabilidad económica (CENAIS, 2009).

Vecn = (Zonas industriales en zonas de riesgo) + (Zonas turísticas en zonas de riesgo) + (Ejecución del presupuesto de reducción) + (Contabilización del costo de respuesta).

Donde:

Indicador	Puntaje
Zonas industriales en zonas de riesgo	3
Zonas turísticas en zonas de riesgo	3
Ejecución del presupuesto de reducción	2
Contabilización del costo de respuesta	2

Tabla 18. Puntaje de la vulnerabilidad económica.

1.4 Análisis de las metodologías de desarrollo de software

El desarrollo de software no es una tarea fácil, prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte se encuentran aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos (Gómez, López, Bacalla 2010).

Por otra parte como se muestra en la siguiente tabla, la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad (Letelier, Canos, Penadés 2004).

Metodología Ágil	Metodología Tradicional
Pocos artefactos. El modelado es prescindible, modelos desechables.	Más artefactos. El modelado es esencial, mantenimiento de modelos.
Pocos roles, más genéricos y flexibles.	Más roles, más específicos.
No existe un contrato tradicional, debe ser bastante flexible.	Existe un contrato prefijado.
Cliente es parte del equipo de desarrollo (además in-situ).	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Orientada a proyectos pequeños. Corta duración (o entregas	Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas/usadas en

frecuentes), equipos pequeños (< 10 integrantes).	proyectos grandes y con equipos posiblemente dispersos.
La arquitectura se va definiendo y mejorando en el trayecto.	Se promueve que la arquitectura se defina tempranamente en el proyecto.
Énfasis en los aspectos humanos: el individuo y el trabajo en equipo.	Énfasis en la definición del proceso: roles, actividades y artefactos.
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Se esperan cambios durante el proyecto.	Se espera que no ocurran cambios de gran impacto durante el proyecto.

Tabla 19. Diferencias entre metodologías ágiles y tradicionales (Letelier, Canos, Penadés 2004).

1.4.1 Metodologías tradicionales

Proceso Unificado de Desarrollo de Software (RUP)

Un proceso define quién está haciendo qué, cuándo y cómo para alcanzar un determinado objetivo. Un proceso de desarrollo de software es un conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto. Este conjunto de actividades, tiene la misión de transformar los requerimientos del usuario en un producto software.

Como RUP es un proceso, en su modelación define como sus principales elementos (Jacobson, 2000):

- Trabajadores (Quién): Define el comportamiento y responsabilidades de un individuo, grupo de individuos, sistema automatizado o máquina, los cuales se encargan de realizar las actividades y son los responsables de una serie de artefactos.
- Actividades (Cómo): Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- Artefactos (Qué): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades.
- Flujo de actividades (Cuándo): Secuencia de actividades realizadas por trabajadores y que producen un resultado de valor observable.

RUP divide su ciclo de vida en *cuatro fases* dentro de las cuales se llevan a cabo un grupo de iteraciones:

- Inicio: Se describe el negocio y se delimita el proyecto a describir y sus alcances con la identificación de los casos de uso del sistema.
- Elaboración: Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen.

- Construcción: Se obtiene un producto listo para su utilización que está documentado.
- Transición: La liberación del producto ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

Está compuesto por nueve Flujos de Trabajo de Ingeniería (Jacobson, 2000):

- Modelado del negocio: Describe los procesos de negocio e identifica quiénes participan y las actividades que requieren automatización.
- Requerimientos: Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- Análisis y diseño: Describe cómo el sistema será realizado a partir de las funcionalidades previstas y las restricciones impuestas, por lo que indica con precisión lo que se debe programar.
- Implementación: Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- Prueba: Busca identificar los defectos y corregirlos antes de la instalación final del sistema, se verifica la integración apropiada de los componentes y que se satisfacen los requerimientos de los clientes.
- Despliegue: Produce liberación del producto y realiza actividades (empaque, instalación, asistencia a usuarios, y otras) para entregar el software a los usuarios finales.
- Administración del proyecto: Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- Administración de configuración y cambios: Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto.
- Entorno: Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

RUP unifica los mejores elementos de metodologías anteriores, está preparado para desarrollar grandes y complejos proyectos, es orientado a objetos, utiliza UML como lenguaje de representación visual y su ciclo de vida se caracteriza por ser: dirigido por casos de uso, centrado en la arquitectura y además por ser iterativo e incremental (Jacobson, 2000).

Ventajas:

- Unifica los mejores elementos de metodologías anteriores.
- Está preparado para desarrollar grandes y complejos proyectos.
- Es orientado a objetos.

- Utiliza UML como lenguaje de representación visual.

Desventajas:

- Realizar cambios conlleva mucha documentación.
- Es para equipos grandes de desarrollo de software.
- Ralentiza los proyectos pequeños.
- Obliga a usar herramientas específicas

Marco de soluciones de Microsoft

Es un enfoque personalizable para entregar con éxito soluciones tecnológicas de manera más rápida, con menos recursos humanos y menos riesgos, pero con resultados de más calidad. Microsoft Solutions Framework (MSF) ayuda a los equipos a enfrentarse directamente a las causas más habituales de fracaso de los proyectos tecnológicos y mejorar así las tasas de éxito, la calidad de las soluciones y el impacto comercial (Microsoft, 2013).

Se centra en:

- Alinear los objetivos de negocio y de tecnologías
- Establecer de manera clara los objetivos, los roles y las responsabilidades
- Implementar un proceso iterativo controlado por hitos o puntos de control
- Controlar los riesgos de manera proactiva
- Responder con eficacia ante los cambios

MSF se asienta sobre una base de principios y actitudes que representan años de experiencia. Estos principios y actitudes, destilados en conceptos que se sostienen en los distintos modelos, procesos y disciplinas de MSF, son la base de MSF. Aunque son conceptos de sentido común, comprenderlos e implementarlos correctamente puede llegar a ser complicado. No obstante, una vez que se comprendan, el equipo podrá crear productos de calidad (Microsoft, 2013).

Los siguientes principios y conceptos de MSF sirven de guía al equipo de proyecto para entregar una solución de calidad. Cada miembro del equipo deberá comprender y aplicar estos principios en sus interacciones con otros miembros del equipo, con la organización y con las partes interesadas. MSF se basa en nueve principios fundamentales (Microsoft, 2013):

- Fomentar una comunicación abierta. Para que el equipo sea eficaz y eficiente, tanto usted como su equipo deben compartir niveles de información apropiados entre los miembros del equipo y en toda la empresa. El equipo debe comprender la naturaleza de lo que se debe hacer y el modo en que se comunican los miembros del equipo y los contactos externos. Lo difícil es determinar un nivel apropiado para cada relación y qué información se debe compartir.

- Intentar lograr una visión compartida. El hecho de tener una visión compartida empodera a los miembros del equipo y les permite actuar con agilidad para poder tomar decisiones rápidas pero bien fundadas con el objetivo de lograr una visión. Al tener una visión compartida, los miembros del equipo pueden ir satisfaciendo los requisitos a medida que se vayan detectando.

- Empodere a los miembros del equipo. Empoderar a los miembros del equipo no solo es una de las muchas maneras de sobrevivir en un entorno en constante cambio, sino que los miembros del equipo también aprenden a encontrar modos de alcanzar el Éxito de manera creativa y a ayudarse unos a otros. Si no se permite a los miembros del equipo dar lo mejor de sí mismos, no solo disminuye su creatividad, sino que también pueden sufrir de baja moral y ser incapaces de contribuir a crear un equipo de alto rendimiento.

- Establecer responsabilidades claras y compartidas. A menudo, los miembros del equipo empoderados se sienten más responsables de sus decisiones y están dispuestos a ser corresponsables de un proyecto. A mayor responsabilidad de los miembros del equipo, mayor calidad. Por ejemplo, si un miembro del equipo afirma que ha completado una tarea pero se detecta que no tiene el nivel de calidad adecuado, ese miembro del equipo es responsable de resolver este problema de manera que la tarea completada tenga los niveles de calidad indicados. Si se fomenta el crecimiento positivo y la responsabilidad en lugar de castigar tales deslices, el miembro del equipo comparte la responsabilidad de la solución general y sus entregas. Esto fomenta la motivación entre los miembros más sólidos del equipo para ayudarse mutuamente a dar lo mejor de sí mismos.

- Ofrecer valor incremental. Este principio tiene dos facetas:
    - Asegurarse de que lo que se entrega tiene un valor óptimo para las partes interesadas.
    - Determinar los incrementos óptimos en los que se entregará valor (o la "frecuencia de entrega").
- Mantenerse ágil, esperar cambios y adaptarse a ellos. Como los cambios pueden darse a menudo y en el peor momento posible, disponer de una manera ágil de manejarlos ayuda a minimizar los trastornos habituales que provocan. Mantenerse ágil significa que una organización está preparada para los cambios y puede adaptarse y ajustarse sin contratiempos.

- Invertir en la calidad. Muchas organizaciones adoptan el principio de calidad, a menudo con una definición bastante difusa, pero no saben cómo cuantificarla. La calidad es algo que se debe incorporar de manera proactiva al ciclo de vida de entrega de la solución y no es algo que aparezca de la nada.

- Aprender de todas las experiencias. Si todos los niveles de una organización no aprenden de lo que funcionó y lo que no funcionó anteriormente, ¿Cómo se puede esperar que mejoren la próxima vez? Los miembros del equipo deben comprender y darse cuenta de que el aprendizaje se da en todos los niveles:
    - A nivel de proyecto, como por ejemplo, al perfeccionar un proceso válido para todo el proyecto
    - A nivel individual, como por ejemplo, al buscar la manera de interactuar mejor con otros miembros del equipo
    - A nivel de la organización, como por ejemplo, al ajustar las métricas de calidad que se recopilan para cada proyecto

Colaborar con clientes internos y externos. Las probabilidades de éxito del proyecto aumentan cuando el cliente trabaja con el equipo del proyecto. Eso no quiere decir que los clientes tengan que hacer el trabajo de un equipo. Sin embargo, cuando los clientes colaboran estrechamente y de manera incremental con un equipo de entrega, la solución satisface mejor sus expectativas. Colaborar con los clientes es ventajoso para ambas partes, ya que ayuda a reducir la incertidumbre, reduce el tiempo necesario para resolver temas de requisitos y aumenta la comprensión por parte del equipo de las propuestas de valor de la solución por medio del contacto periódico (Microsoft, 2013).

Ventajas:

- Tiene como principios fundamentales la comunicación entre el cliente, el usuario y los integrantes del grupo de trabajo.
- Invierte en calidad.
- Demuestran resultados interactivamente e incrementalmente.
- Existe la colaboración del equipo de trabajo.

Desventajas:

- Equipos de trabajo grandes.
- No utiliza UML.
- Depende de otras herramientas para su correcta implementación.
- Documentación exhaustiva de todo el proyecto.
- Es necesario tener en el equipo de trabajo una persona experta en la aplicación.

1.4.2 Metodologías ágiles

SCRUM

Es una metodología ágil con un proceso liviano que sirve para administrar y controlar el desarrollo de software. El desarrollo se realiza en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora una nueva funcionalidad. Las iteraciones en general tienen una duración entre 2y 4 semanas. SCRUM se utiliza como marco para otras prácticas de ingeniería de software como RUP o Extreme Programming. Gestiona su evolución a través de reuniones breves diarias en las que todo el equipo revisa el trabajo realizado el día anterior y el previsto para el día siguiente (Valdés, 2011.).

Un principio clave de SCRUM es el reconocimiento de que durante un proyecto los clientes pueden cambiar de idea sobre lo que quieren y necesitan y que los desafíos impredecibles no pueden ser fácilmente enfrentados de una forma predictiva y planificada. Por lo tanto, SCRUM adopta una aproximación pragmática, aceptando que el problema no puede ser completamente entendido o definido y centrándose en maximizar la capacidad del equipo de entregar rápidamente y responder a requisitos emergentes (Pressman R. S., 2002).

Ventajas:

- Se adapta bien a los cambios.
- Constante interacción con el cliente.
- El equipo de trabajo tiene las metas a cumplir bien definidas en cada jornada laboral.

Desventajas:

- Muchas reuniones.
- Cada ciclo depende del anterior.
- Desperdicio del tiempo de trabajo.

Programación Extrema (XP)

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propicia un buen clima de trabajo. XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo

técnico. Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre (Cortizo Pérez, 2004).

Según (Cortizo Pérez, José Carlos, Expósito Gil, Diego y Ruiz Leyva, Miguel, 2004) las fases de XP son:

- Exploración: En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.
- Planificación de la entrega: En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.
- Iteraciones: Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible pues es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.
- Producción: La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).
- Mantenimiento: Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro el equipo y cambios en su estructura.

- Muerte del Proyecto: Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o no hay presupuesto para mantenerlo (Cortizo Pérez, 2004).

Ventajas:

- Se adapta bien a los cambios.
- Facilita los proyectos sencillos.
- Prioriza la funcionalidad del software.
- Constante interacción con el cliente.

Desventajas:

- Documentación escasa.

Proceso Unificado Ágil (AUP)

Metodología creada por Scott Ambler es una versión simplificada del Proceso Unificado de Rational (RUP); describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP (Flores, 2005).

AUP dentro de sus características fundamentales presenta (Miranda, 2010):

- Versión simplificada de la metodología RUP.
- Abarca siete flujos de trabajos, cuatro ingenieriles y tres de apoyo: Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyectos y Ambiente.
- El modelado agrupa los tres primeros flujos de RUP (Modelamiento del negocio, Requerimientos y Análisis y Diseño). Dispone de cuatro fases igual que RUP: Incepción o Creación, Elaboración, Construcción y Transición.

Descripción de los flujos de trabajo (Miranda, 2010):

- Modelado: es el flujo de trabajo que tiene el objetivo de entender el negocio de la organización, el problema de dominio que se aborda en el proyecto y determinar una solución viable para resolver el problema de dominio.
- Implementación: tiene como objetivo transformar su(s) modelo(s) en código ejecutable y realizar un nivel básico de las pruebas, en particular, la unidad de pruebas.

- Prueba: tiene como objetivo realizar una evaluación objetiva para garantizar la calidad. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, verificando que se cumplan los requerimientos.
- Despliegue: tiene como objetivo realizar un plan para la prestación del sistema y la ejecución de dicho plan, para que el sistema quede a disposición de los usuarios finales.
- Gestión de configuración: tiene como objetivo la gestión de acceso a herramientas de su proyecto. Esto incluye no sólo el seguimiento de las versiones con el tiempo, sino también el control y gestión del cambio para ellas.
- Gestión de proyectos: tiene como objetivo dirigir las actividades que se llevan a cabo en el proyecto. Esto incluye la gestión de riesgos, la dirección de personas (la asignación de tareas, el seguimiento de los progresos), coordinación con el personal y los sistemas fuera del alcance del proyecto para asegurarse de que es entregado a tiempo y dentro del presupuesto.
- Ambiente: tiene como objetivo apoyar el resto de los esfuerzos por garantizar que el proceso sea el adecuado, la orientación (normas y directrices), y herramientas (hardware, software) estén disponibles para el equipo según sea necesario.

Descripción de las fases del ciclo de desarrollo*:*

- Incepción: identificación del alcance y dimensión del proyecto, propuesta de la arquitectura y del presupuesto del cliente.
- Elaboración: confirmación de la idoneidad de la arquitectura.
- Construcción: desarrollo incremental del sistema, siguiendo las prioridades funcionales de los implicados.
- Transición: validación e implantación del sistema. Las técnicas ágiles que aplica AUP son (Miranda, 2010):
    - Desarrollo Dirigido por Pruebas (Test Driven Development - TDD).
    - Modelado Ágil.
    - Gestión de Cambios Ágil.
    - Refactorización de Base de Datos para mejorar la productividad.

Principios en los que se basa AUP (Miranda, 2010):

- Simplicidad: todo se describe concisamente utilizando poca documentación.
- Agilidad: el ajuste a los valores y principios de La Alianza Ágil.
- Centrarse en actividades de alto valor: la atención se centra en las actividades que en realidad lo requieren, no en todo el proyecto.

- Independencia de la herramienta: se puede usar cualquier conjunto de herramientas que desea

Con AUP se sugiere utilizar las herramientas más adecuadas para el trabajo, que a menudo son las herramientas simples o incluso herramientas de código abierto (Miranda, 2010).
AUP es una metodología bastante completa con la cual el código y la documentación van emparejadas.
Ventajas:

- Versión simplificada de la metodología RUP.
- Describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software.
- Permite usar cualquier conjunto de herramientas.
- Genial para equipos de desarrollo pequeños.

1.4.3 Selección de la Metodología

Como el equipo de desarrollo está compuesto por una persona y se esperan cambios durante el ciclo de construcción, debido a que es necesario que el cliente se vea integrado en el proceso de confección del proyecto se optó por usar una metodología ágil que son las que satisfacen estas necesidades y permiten mantener la documentación actualizada para obtener un proceso organizado.
Después de analizar las metodologías ágiles, AUP cumple con estas características, por ser una versión simplificada de RUP lo cual permite para proyectos pequeños y de pocos integrantes, trazar las directrices a seguir durante el proceso de desarrollo.

1.5 Análisis de las tecnologías para el desarrollo de software

1.5.1 Lenguajes de Modelado

Las ramas más antiguas de la ingeniería han encontrado útil, desde hace mucho tiempo, representar los diseños mediante dibujos. Desde los inicios del software, los programadores han encapsulado sus conceptos en diversos tipos de dibujos o, más ampliamente, de modelos (IBM, 2010).
Los modelos de ingeniería tienen como propósito ayudar a resolver un problema complejo, comunicar ideas acerca de un problema o solución y guiar la implementación. Para que un modelo sea eficaz debe satisfacer las siguientes características (IBM, 2010):

- Abstracto: Debe enfatizar los elementos importantes y ocultar los irrelevantes.
- Comprensible: Debe ser fácil de entender para los observadores.
- Preciso: Representar de forma fiel el sistema que modela.
- Predictivo: Se puede usar para deducir conclusiones sobre el sistema que modela.
- Barato: Mucho más barato y sencillo de construir que el sistema que modela.

El Lenguaje Unificado de Modelado (UML) es un lenguaje estándar de modelado para software – un lenguaje para la visualización, especificación, construcción y documentación de los artefactos de sistemas en los que el software juega un papel importante. Básicamente, UML permite a los desarrolladores visualizar los resultados de su trabajo en esquemas o diagramas estandarizados (IBM, 2010).

UML es un lenguaje de modelado de propósito general que pueden usar todos los modeladores. No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática (Pressman, 2002).

Es importante destacar que UML es un lenguaje de modelado, no un método o un proceso, se emplea para definir, detallar y documentar los artefactos de un sistema de software. En la última versión se adicionaron diversas novedades que resuelven carencias desde el punto de vista práctico fundamentalmente.

Entre los diagramas que propone UML para modelar un sistema se encuentran (Pressman, 2002):

- Diagramas de estructura (clases, componentes, objetos, despliegue, paquetes).
- Diagramas de comportamiento (actividades, casos de uso, estado).
- Diagramas de interacción (secuencia, comunicación).

1.5.2 Herramientas CASE

CASE es una sigla, que corresponde a las iniciales de: Computer Aided Software Engineering; y en su traducción al español significa Ingeniería de Software Asistida por Computación. El concepto de CASE es muy amplio; y una buena definición genérica, que pueda abarcar esa amplitud de conceptos, sería la de considerar a la Ingeniería de Software Asistida por Computación (CASE), como la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación. Estas herramientas permiten organizar y manejar la información de un proyecto informático. Permitiéndoles a los participantes de un proyecto, que los sistemas (especialmente los complejos), se tornen más flexibles, más comprensibles y además mejorar la comunicación entre los participantes (Larman, 1999).

1.5.2.1 ArgoUML

Es una herramienta desarrollada en Java que permite crear modelos UML compatibles con los estándares de la versión 1.4 de este lenguaje. Incluye una interfaz muy intuitiva, estable y de sencillo manejo. Los tipos de diagramas que se pueden crear son: diagramas de clases, de estados, de actividad, de casos de uso, de colaboración, de despliegue y de secuencia (Linperial International Systems S.A de C.V, 2010).

1.5.2.2 Visual Paradigm

Suite completa de herramientas CASE que da soporte al modelado visual con UML 2.0 ofreciendo distintas perspectivas del sistema. Independiente de la plataforma y dotada de una buena cantidad de productos o módulos para facilitar el trabajo durante la confección de un software así como garantizar la calidad del producto final. Además es importante destacar que tiene versiones con licencia libre para el uso de la misma (Carvajal Pérez, 2009).

Posee entre sus principales características las siguientes (Carvajal Pérez, 2009):

- Es profesional: brinda la posibilidad de crear un conjunto bastante amplio de artefactos utilizados con mucha frecuencia durante la confección de un Software. Todos estos, cumpliendo con el Standard UML 2.0.
- Es amigable: puede ser utilizado en varios idiomas, sus componentes se encuentran relacionados, por lo que se hace muy fácil la creación de cualquier tipo de diagrama, ya que cada componente utilizado en el diagrama que se esté creando, sugiere nuevos posibles componentes a utilizar, por lo que ya no es necesario localizarlos en la barra donde pueden aparecer un número grande de componentes.
- Brinda un número considerable de estereotipos a utilizar, lo que permite un mayor entendimiento de los diagramas.
- Facilidades para redactar especificaciones de casos de uso: es posible crear plantillas para las especificaciones de casos de uso y describirlos, por lo que no se necesita de una herramienta externa como editor de texto.
- Generación de código e ingeniería inversa: brinda la posibilidad de generar código a partir de los diagramas, para plataformas como .Net, Java y PHP, así cómo obtener diagramas a partir del código.
- Integración con distintos Ambientes de Desarrollo Integrados (IDE): se integra fácilmente con varios IDEs, entre ellos el de Visual Studio, el Eclipse y el NetBeans.
- Interoperabilidad con otras aplicaciones: brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo Visio y Rational Rose. Además permite importar y exportar XML y XMI.
- Generación de código ORM: permite generar a partir de un Diagrama de Entidad Relación una Base de Datos Relacional y el código necesario para acceder a esta base de datos utilizando Java, PHP, C# o Enterprise Object Framework.
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, Vista, Seven y 8), Linux, Mac OS X, Solaris o Java.

1.5.2.3 Rational Rose

Herramienta de diseño de software destinada al modelado visual. Proporciona un lenguaje común de modelado para el equipo que facilita la creación de software de calidad más rápidamente. Dos rasgos importantes del Rational Rose son su habilidad para ofrecer desarrollo iterativo e ingeniería bidireccional.

El Rational Rose permite a los diseñadores aprovecharse del desarrollo iterativo, porque la nueva aplicación puede ser creada por etapas con la salida de una iteración como entrada de la próxima. Además Rose permite la generación de código a partir de un diseño en UML en lenguajes como C++,
VisualBasic, Java, Ada, genera IDL's para aplicaciones CORBA. Soporta realizar ingeniería inversa por lo que se puede obtener un diseño a partir del código de un programa. Está disponible en la plataforma Windows: en MicrosoftWindows NT 4.0, Windows 95, Windows 98, Windows Seven; y la licencia es exclusivamente propietaria (Carvajal Pérez, 2009).

Proporciona una herramienta de modelado visual para capturar y compartir los requerimientos del negocio y el seguimiento de ellos a medida que cambian a lo largo del proceso. Está basado en modelos de desarrollo para Java y entornos de aplicación J2EE, arquitectos y desarrolladores de software de apoyo. Permite el desarrollo de aplicaciones de software, modelado de datos, servicios de diseño web, modelado de negocios, y la extensión de aplicaciones heredadas (Visconti, 2003). La desventaja que tiene es que es una herramienta propietaria.

1.5.2.4 Selección de la Herramienta Case

Todas las herramientas CASE analizadas son bastante potentes. Se decidió utilizar como herramienta CASE para el modelado de la aplicación el Visual Paradigm en su versión 8.0 por ser multiplataforma, porque da soporte al modelado visual con UML 2.0 ofreciendo distintas perspectivas del sistema y tener licencia de uso libre lo que permitirá transferir los modelos obtenidos al cliente, al finalizar el desarrollo del sistema.

1.5.3 Sistema Gestor de Base de Datos

Sistema Gestor de Base de Datos (SGBD), en inglés DBMS (Data Base Management System), es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos. Su objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones.

Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta (Domínguez, 2010).

1.5.3.1 Oracle

Es un Sistema Gestor de Base de Datos relacional fabricado por la empresa de software Oracle Corporation, es considerado en la actualidad como unos de los más completos, se destaca por su gran soporte de transiciones, estabilidad, escalabilidad, además de ser un programa multiplataforma. Oracle cuenta con una buena administración de usuarios que a la vez propicia una administración de roles y tiene procedimientos almacenados, además de soportar disparadores (Oracle, 2010).

1.5.3.2 PostgreSQL

Es un Sistema Gestor de Base de Datos objeto-relacional basado en el proyecto POSTGRES creado por el Departamento de Investigación de la Universidad de California en Berkeley. Aunque la licencia es propiedad de dicha universidad, está distribuido bajo la licencia BSD y con su código fuente disponible libremente; es decir, es libre para utilizar, copiar, modificar y distribuir, sin importar para los fines que se utilice. PostgreSQL utiliza como modelo la arquitectura cliente/servidor, además de utilizar multiprocesos en vez de multihilos, este propicia que un fallo en uno de los procesos del sistema no afecte el resto del mismo y continúe funcionando; presenta confiabilidad, estabilidad y mantenimiento de la integridad de los datos; permite definir un nuevo tipo de tabla, a partir de otra previamente definida, todo esto gracias a que posee características de la Programación Orientada a Objetos (POO), como puede ser la herencia, tipos de datos, funciones, restricciones y disparadores, reglas e integridad transaccional (Postgre SQL, 2010).

1.5.3.3 Selección del Sistema Gestor de Base de Datos

PostgreSQL debido a que es una herramienta libre, de código abierto y gratis; cuenta con varios años de desarrollo activo y con buenos resultados, además de mantener la integridad de los datos y corrección; mientras que Oracle es propietario. Se decide utilizar PostgreSQL en su versión 9.1.1 ya que las características mencionadas son útiles para el sistema a implementar.

1.5.4 Lenguaje de programación

Un lenguaje de programación es una notación o conjunto de símbolos y caracteres combinados entre sí de acuerdo con una sintaxis ya definida que posibilita la transmisión de instrucciones a la Unidad Central de Procesamiento (CPU) (Catalina, 2003).

La aplicación era necesario que fuera desarrollada con tecnología Desktop debido a la mala infraestructura que posee la empresa. Los lenguajes que soportan esta tecnología se decidió java por ser libre, de código abierto, multiplataforma y además por su robustez.

1.5.5 Entornos Desarrollo Integrado

Los Entornos Integrados de Desarrollo (Integrated Development Environment), tal y como su nombre indican, son entornos de desarrollo integrados. En un mismo programa es posible escribir el código Java, compilarlo y ejecutarlo sin tener que cambiar de aplicación. Algunos incluyen una herramienta para realizar Debug gráficamente, frente a la versión que incorpora el JDK basada en la utilización de una consola (denominada habitualmente ventana de comandos de MS-DOS, en Windows NT/95/98) bastante difícil y pesada de utilizar. Estos entornos integrados permiten desarrollar las aplicaciones de forma mucho más rápida, incorporando en muchos casos librerías con componentes ya desarrollados, los cuales se incorporan al proyecto o programa. Como inconvenientes se pueden señalar algunos fallos de compatibilidad entre plataformas, y archivos resultantes de mayor tamaño que los basados en clases estándar (Sanabria, 2003).

1.5.5.1 Eclipse

Es un IDE de código abierto y multiplataforma, para desarrollar “Aplicaciones de Cliente Enriquecido”, este emplea un conjunto de plugins para sus funcionalidades; a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no; este tipo de mecanismo le permite a Eclipse ser extensible cuando de lenguaje de programación se habla, como son C++ y Java. De forma general Eclipse es únicamente un gran armazón sobre el cual se pueden montar herramientas de desarrollo, presenta Licencia Pública Eclipse (LPE) (Geocities, 2010).

1.5.5.2 NetBeans

NetBeans IDE es el IDE oficial de Java 8. Con sus editores, analizadores de código y convertidores, puede actualizar de forma rápida y sin problemas sus aplicaciones para utilizar las nuevas construcciones del lenguaje Java 8, operaciones funcionales, y las referencias a métodos. Los analizadores por lotes y convertidores se proporcionan al buscar a través de múltiples aplicaciones al mismo tiempo, haciendo coincidir los patrones para la conversión a las nuevas construcciones del lenguaje Java 8. Con su mejora constante el Editor Java, muchas características se optimizan y una amplia gama de herramientas, plantillas y muestras, NetBeans IDE establece el estándar para el desarrollo de tecnologías de vanguardia de la caja. Y cuenta con las siguientes funciones (IDE NetBeans, 2010):

- Reemplaza tabulaciones por espacios.
- Proporciona el autocompletado con el apoyo y sugerencia para funciones y clases de todo el proyecto
- Compatible con SVN y GIT, sistemas de control de versiones.

1.5.5.3 Selección del IDE de desarrollo

Eclipse y NetBeans son herramientas poderosas que se pueden utilizar debido a que los dos IDE de soportan java, pero se decide usar NetBeans por ser el IDE por defecto de java además de que permite que el fichero de salida sea multiplataforma y facilita el trabajo con las interfaces.

1.6 Conclusiones parciales

Durante el capítulo se realizó un resumen de los principales conceptos asociados al dominio del problema, con el objetivo de lograr un mejor entendimiento del mismo. Se estudiaron y analizaron las soluciones similares existentes en el mundo junto con la metodología PVR como apoyo a la solución que se pretende desarrollar para el grupo de medio ambiente UIC Pinar del Río. Se evidencia la necesidad de este trabajo, ya que internacionalmente las aplicaciones desarrolladas no satisfacen las necesidades del cliente y nacionalmente no existe ningún software que el cliente pueda utilizar pues el tema no ha sido explotado en la isla.

Se explicó la metodología PVR como base teórica para el sistema para la evaluación de la prevención, recuperación y vulnerabilidad de los sismos.

Según las tendencias actuales se selecciona las herramientas y metodología para el desarrollo de software, AUP como metodología ágil, UML como lenguaje de modelado, Visual Paradigm en su versión 8.0 como herramienta CASE, como lenguaje de programación Java e IDE NetBeans en su versión 8.0 con las bibliotecas necesarias y como gestor de bases de datos Postgres SQL en su versión 9.1.

Capítulo 2 Descripción de la solución propuesta

En este capítulo se realiza y describe los principales conceptos del modelo de dominio para un mejor entendimiento del problema a desarrollar y permitir obtener los requerimientos tanto funcionales como los no funcionales que componen las funcionalidades que va a tener el software SISπ. Se realiza el diagrama de caso de uso del sistema junto con una breve descripción de cada caso en una tabla resumen. A partir de lo anterior se diseña los prototipos de interfaz del sistema para que el usuario tenga una idea de la interfaz visual para cada opción dentro del software y se familiarice con el mismo.

2.1 Modelo de dominio

Se describen primeramente los conceptos que se representarán en el diagrama Modelo de Dominio:

- Provincia: Parte del país en la cual se va a realizar el estudio.
- Técnica antisísmica: Forma empleada a la hora de la fabricación de la estructura para minimizar los efectos negativos de los terremotos.
- Localidad: Región de la provincia en la cual se van a analizar las edificaciones (municipios).
- Construcción: Estructura que se está analizando.
- Vulnerabilidad: Forma en la cual la estructura se verá afectada a la ocurrencia de un evento adverso.
- Vulnerabilidad Estructural: Se evalúa la vulnerabilidad estructural del fondo habitacional, de las edificaciones de importancia y los puentes de carreteras, teniendo en cuenta aspectos relacionados con la tipología estructural, estado técnico, material estructural, problemas de configuración, entre otros factores (CENAIS, 2009).
- Vulnerabilidad No Estructural: Determina la susceptibilidad a daños que presentan las líneas vitales del área de estudio, tales como: redes de acueducto, alcantarillado, eléctricas, de comunicaciones, instalaciones que almacenan sustancias peligrosas y carreteras. Se realiza una inspección donde se detalla el nivel de daño que pueden sufrir cada uno de los factores que inciden en la vulnerabilidad no estructural (CENAIS, 2009).
- Vulnerabilidad Funcional: Permite definir la funcionabilidad o paralización de los servicios básicos ante un evento sísmico (CENAIS, 2009).
- Vulnerabilidad Social: Total de población expuesta, personas sin viviendas, sin asistencia médica, personas afectadas por sustancias peligrosas o incomunicadas (CENAIS, 2009).

- Vulnerabilidad Económica: factores relacionados con las pérdidas de zonas industriales, zonas turísticas, así la incapacidad de producción o transportación de producciones importantes entre otras (CENAIS, 2009).

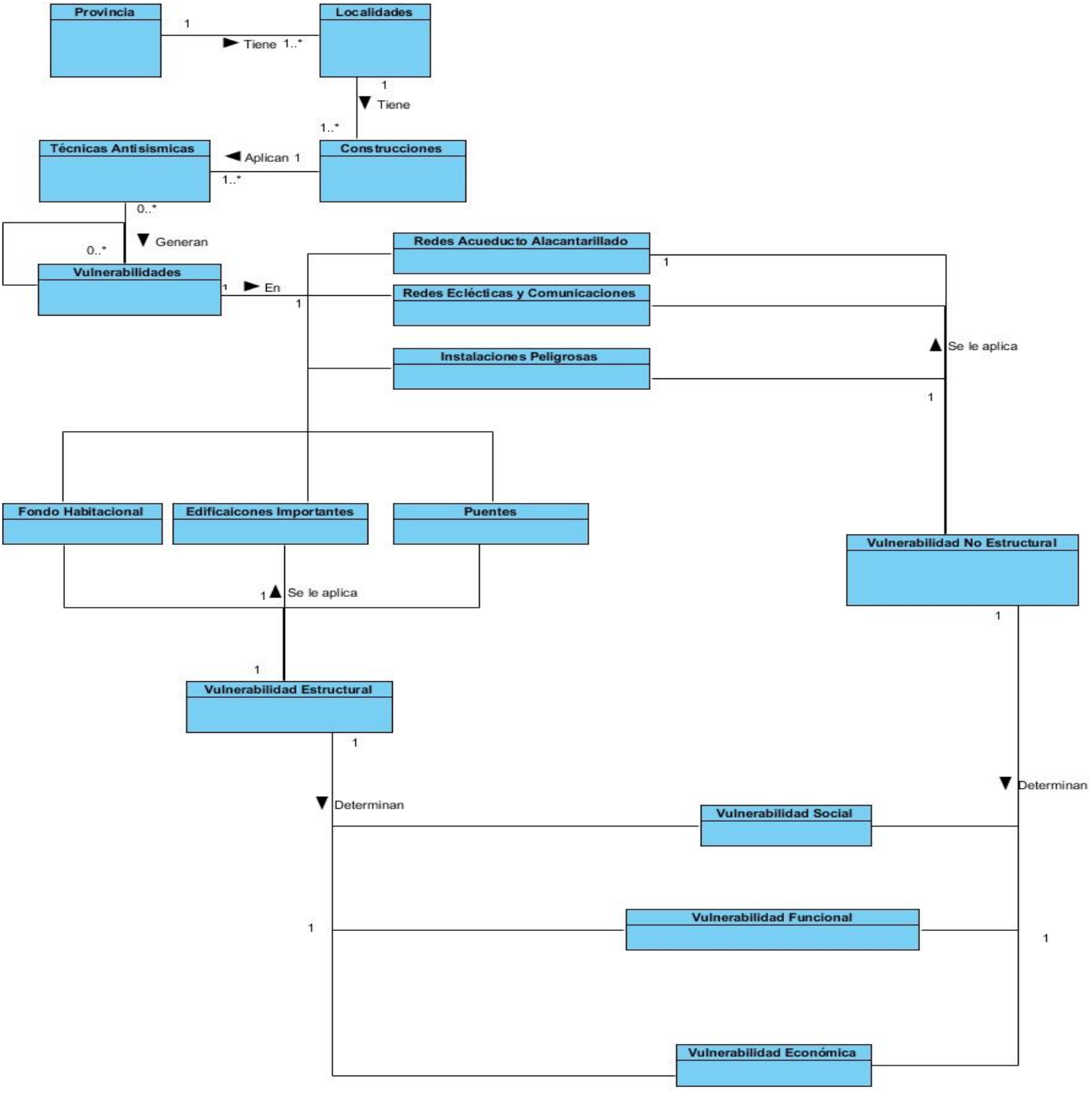

Figura 1. Modelo de dominio.

2.2 Requerimientos

2.2.1 Requerimientos funcionales (RF)

El sitio oficial de AUP recomienda el tratamiento de requisitos como una lista priorizada que evolucione en el tiempo (esto apoya la gestión de cambios).

Los requerimientos funcionales componen las distintas actividades que el software debe hacer por tanto según los objetivos del trabajo serían:

RF 1. Evaluar la vulnerabilidad estructural.

RF 1.1. Evaluar la vulnerabilidad de puentes.

RF 1.2. Evaluar la vulnerabilidad edificaciones importantes.

RF 1.3. Evaluar la vulnerabilidad fondo habitacional.

RF 2. Evaluar la vulnerabilidad social.

RF 3. Evaluar la vulnerabilidad económica.

RF 4. Evaluar la vulnerabilidad funcional.

RF 5. Evaluar la vulnerabilidad no estructural.

RF 5.1. Evaluar la vulnerabilidad redes eléctricas y comunicaciones.

RF 5.2. Evaluar la vulnerabilidad redes acueducto y alcantarillado.

RF 5.3. Evaluar la vulnerabilidad instalaciones peligrosas.

RF 6. Calcular índice de vulnerabilidad.

RF 7. Representar en mapa la ubicación de las casas analizadas.

RF 8. Insertar nuevos estudios.

RF 9. Modificar estudios existentes.

RF 10. Eliminar estudios existentes.

RF 11. Generar un reporte.

RF 12. Obtener cuantitativamente el costo de recuperación ante los sismos.

RF 13. Obtener cuantitativamente el costo de prevención ante los sismos.

RF 14. Insertar medidas preventivas.

RF 15. Modificar medidas preventivas.

RF 16. Eliminar medidas preventivas.

RF 17. Insertar medidas recuperación.

RF 18. Modificar medidas recuperación.

RF 19. Eliminar medidas recuperación.

RF 20. Autenticar Usuario.

2.2.2 Requerimientos no funcionales

Los requisitos no funcionales están enfocados a destacar las cualidades que debe tener el software y se clasifican en los siguientes grupos:

Usabilidad

RNF1. Interfaz sugerente y manejable para usuarios con escasos conocimientos informáticos puedan manipular el software.

RNF2. Fácil acceso a las funcionalidades del sistema.

RNF3. Tipografía amena para todo tipo de usuarios.

Confiabilidad

RNF5. La información manejada y generada por el sistema estará almacenada en la Base de Datos.

RNF6. Solicitar confirmación a los usuarios antes de realizar cualquier operación que comprometa la integridad de los datos.

RNF7. Los usuarios deben tener acceso (según sus permisos y roles otorgados) en todo momento a la información solicitada.

Soporte

RNF8. Ser generado en tecnología Desktop para trabajar sin conexión.

RNF9 Utilizar las buenas prácticas de programación a la hora de la implementación.

RNF10. El software debe ser multiplataforma.

Restricciones de diseño

RNF11. El lenguaje de programación a emplear debe ser java 8.

RNF12. Como IDE se empleará NetBeans 8.

RNF13. El SGDB deberá ser PostgreSQL 9.1.1.

Interfaces de usuario

RNF14. Interfaz de usuario sugerente para los usuarios finales obtengan una navegación dentro del sistema sencilla con un mínimo de información.

Interfaces de hardware

RNF15. 500 MHz de Microprocesador.

RNF16. 128 Mb de RAM.

RNF17. 2 GB de HDD mínimo, depende de la cantidad de información se quiera almacenar.

Legales, Derecho de autor y otros

RNF18. Una vez terminada la aplicación debe ser sometida a una evaluación y certificación por parte de los clientes del producto.

Seguridad

RNF19. El acceso al sistema estará restringido por usuario y contraseña.

2.3 Diagrama de Casos de Uso del Sistema

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa. En el diagrama de casos de uso se representa también el sistema como una caja rectangular con el nombre en su interior. Los casos de uso están en el interior de la caja del sistema, y los actores fuera, y cada actor está unido a los casos de uso en los que participa mediante una línea (Pareades, 2011).

A continuación se muestra el diagrama de caso de uso del software SISπ.

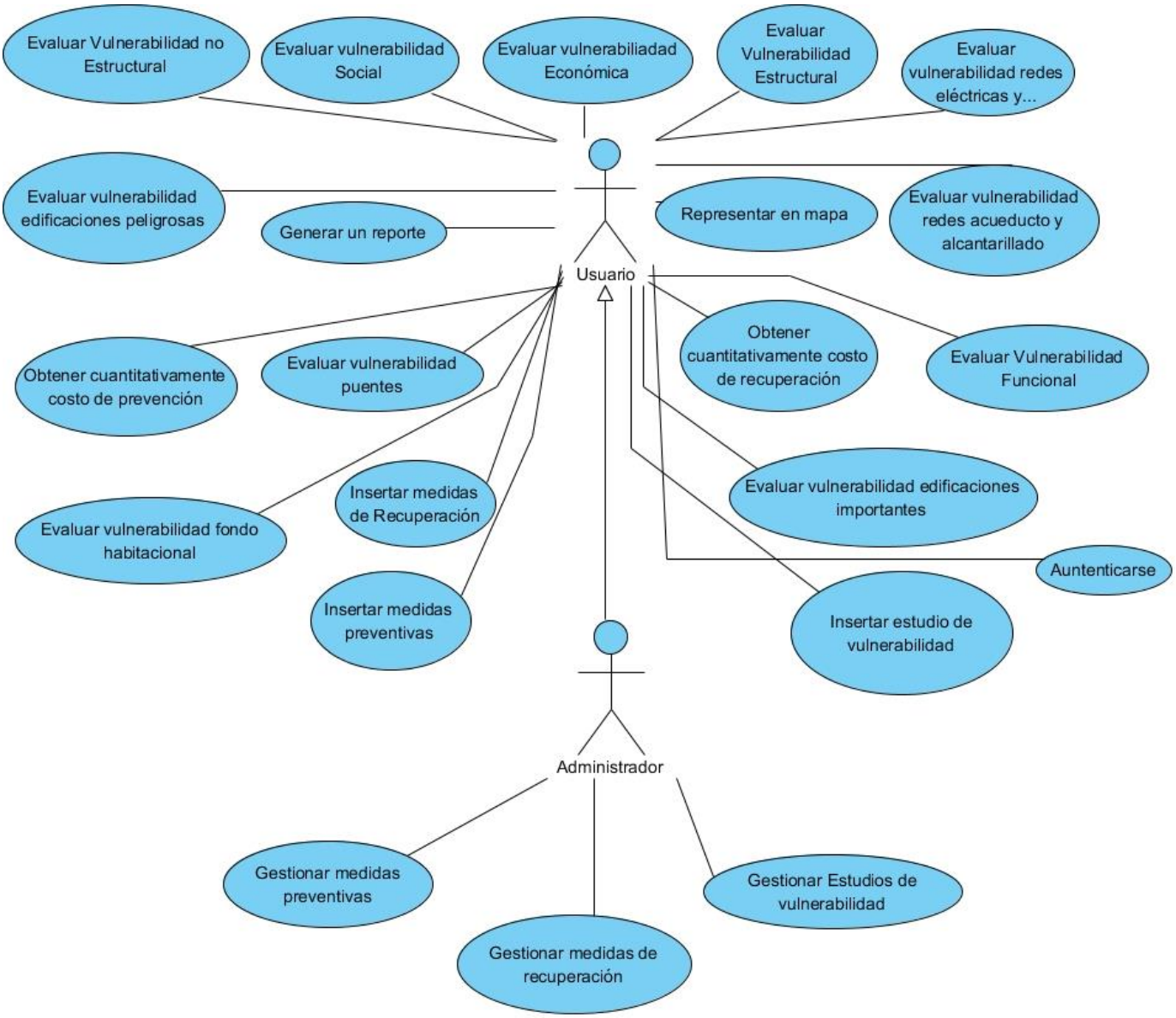

Figura 2. Diagrama Caso de uso del sistema.

Descripción del actor del sistema

Actor	Descripción
Usuario	Persona con conocimientos sobre sismos que puede insertar estudios de vulnerabilidad, generar un reporte y plotear la ubicación de las casas.

Administrador	Persona con conocimientos sobre sismos que puede insertar, modificar o eliminar estudios de vulnerabilidad, generar un reporte y plotear la ubicación de las casas.

2.4 Descripción de los casos de uso del sistema

Nombre del caso de uso	Gestionar estudios de vulnerabilidad
Objetivo	Modificar o eliminar un estudio de vulnerabilidad
Actores	Administrador: Modifica o elimina un estudio existente
Complejidad	Alta
Nivel	Administrador
Precondiciones	Debe estar autenticado en el sistema
Poscondiciones	Se modificó o eliminó un estudio existente
Descripción el caso de uso	
El caso de uso se inicia cuando el administrador selecciona los datos que corresponden a la estructura que se está analizando. El administrador puede modificar o eliminar los datos de un estudio de vulnerabilidad (EV). En caso de que seleccione la opción de cargar un EV el sistema permitirá realizar una búsqueda por distintos criterios como nombre del EV, o identificador con el que fue guardado, se le permite modificar los datos cargados. Si el usuario elige la opción de eliminar un EV, el sistema eliminará de la BD el EV seleccionado.	
Requisitos Funcionales	
RF8, RF9, RF10.	

Tabla 20. Resumen de la descripción del CU Gestionar Estudios de vulnerabilidad.

Nombre del caso de uso	Insertar estudio de vulnerabilidad
Objetivo	Insertar los datos de un estudio de vulnerabilidad
Actores	Usuario: Inserta un estudio de vulnerabilidad
Complejidad	Alta
Nivel	Usuario
Precondiciones	Debe estar autenticado en el sistema
Poscondiciones	Se Insertó un nuevo estudio de vulnerabilidad
Descripción el caso de uso	
El caso de uso se inicia cuando el usuario selecciona los datos que corresponden a la estructura que se está analizando. El usuario puede insertar un nuevo estudio de vulnerabilidad (EV) en la BD.	
Requisitos Funcionales	
RF8.	

Tabla 21. Resumen de la descripción del CU Insertar Estudios de vulnerabilidad.

Nombre del caso de uso	Evaluar la vulnerabilidad estructural
Objetivo	Evaluar cuantitativamente la vulnerabilidad no estructural
Actores	Usuario, Administrador: selecciona los datos necesarios para que el software realice los cálculos
Complejidad	Media
Nivel	Usuario
Precondiciones	Los datos estén correctamente seleccionados
Poscondiciones	La evaluación de la vulnerabilidad cuantitativamente.
Descripción el caso de uso	
El caso de uso se inicia cuando el usuario o el administrador seleccionen todos los datos de cualquiera de las plantillas que trae el sistema vinculadas a esta vulnerabilidad para posteriormente calcular la vulnerabilidad estructural.	
Requisitos Funcionales	
RF1, RF1.1, RF1.2, RF1.3.	

Tabla 22. Resumen de la descripción del CU Evaluar la vulnerabilidad estructural.

Nombre del caso de uso	Evaluar la vulnerabilidad no estructural.
Objetivo	Evaluar cuantitativamente la vulnerabilidad no estructural.
Actores	Usuario, Administrador: selecciona los datos necesarios para que el software realice los cálculos.
Complejidad	Media
Nivel	Usuario
Precondiciones	Los datos estén correctamente seleccionados
Poscondiciones	La evaluación de la vulnerabilidad no estructural cualitativamente.
Descripción el caso de uso	
El caso de uso se inicia cuando el usuario o el administrador seleccionen todos los datos de cualquiera de las plantillas que trae el sistema vinculadas a esta vulnerabilidad para posteriormente calcular la vulnerabilidad no estructural.	
Requisitos Funcionales	
RF5, RF5.1, RF5.2, RF5.3.	

Tabla 23. Resumen de la descripción del CU Evaluar la vulnerabilidad no estructural.

Nombre del caso de uso	Evaluar la vulnerabilidad social.
Objetivo	Evaluar cuantitativamente la vulnerabilidad social.
Actores	Usuario, Administrador: selecciona los datos necesarios para que el software realice los cálculos.
Complejidad	Media
Nivel	Usuario
Precondiciones	Los datos estén correctamente seleccionados
Poscondiciones	La evaluación de la vulnerabilidad social.
Descripción el caso de uso	
El caso de uso se inicia cuando el usuario o el administrador seleccionen todos los datos de la plantilla vulnerabilidad social el sistema calcula cuantitativamente la vulnerabilidad ante este evento adverso para la población.	
Requisitos Funcionales	
RF2.	

Tabla 24. Resumen de la descripción del CU Evaluar la vulnerabilidad social.

Nombre del caso de uso	Evaluar la vulnerabilidad funcional.
Objetivo	Evaluar cuantitativamente la vulnerabilidad funcional.
Actores	Usuario, Administrador: selecciona los datos necesarios para que el software realice los cálculos.
Complejidad	Media
Nivel	Usuario
Precondiciones	Los datos estén correctamente seleccionados
Poscondiciones	La evaluación de la vulnerabilidad social.
Descripción el caso de uso	
El caso de uso se inicia cuando el usuario o el administrador seleccionen todos los datos de la plantilla vulnerabilidad funcional el sistema calcula cuantitativamente la vulnerabilidad ante este evento adverso y poder determinar si la estructura permanece en funcionamiento o no.	
Requisitos Funcionales	
RF4.	

Tabla 25. Resumen de la descripción del CU Evaluar la vulnerabilidad funcional.

Nombre del caso de uso	Evaluar la vulnerabilidad Económica.
Objetivo	Evaluar cuantitativamente la vulnerabilidad económica.
Actores	Usuario, Administrador: selecciona los datos necesarios para que el software realice los cálculos.
Complejidad	Media
Nivel	Usuario
Precondiciones	Los datos estén correctamente seleccionados
Poscondiciones	La evaluación de la vulnerabilidad social.
Descripción el caso de uso	
El caso de uso se inicia cuando el usuario o el administrador seleccionen todos los datos de la plantilla vulnerabilidad económica el sistema calcula cuantitativamente la vulnerabilidad de la economía ante este evento adverso y poder determinar medidas de prevención y de recuperación.	
Requisitos Funcionales	
RF3.	

Tabla 26. Resumen de la descripción del CU Evaluar la vulnerabilidad económica.

Nombre del caso de uso	Generar reporte
Objetivo	Generar un documento con los datos del estudio.
Actores	Usuario, Administrador: selecciona los datos necesarios para que el software realice los cálculos e introduce las medidas preventivas y de recuperación a tomar para esa estructura.
Complejidad	Alta
Nivel	Usuario
Precondiciones	Haberse realizado un estudio de vulnerabilidad.
Poscondiciones	Un documento con los datos del estudio realizado.
Descripción el caso de uso	
El caso de uso se inicia cuando el usuario o el administrador seleccionen la opción de generar reporte después de haberse realizado el estudio de vulnerabilidad correctamente. El reporte contendrá los atributos y los valores de la vulnerabilidad analizada así como una breve explicación según la metodología PVR.	
Requisitos Funcionales	
RF11.	

Tabla 27. Resumen de la descripción del CU Generar reporte.

Nombre del caso de uso	Representar en mapa
Objetivo	Representar las casas analizadas en un mapa para saber su ubicación exacta.
Actores	Usuario, Administrador: selecciona los datos necesarios para que el software realice los cálculos
Complejidad	Alta
Nivel	Usuario
Precondiciones	Haberse realizado un estudio de vulnerabilidad.
Poscondiciones	Un mapa con la representación de las casas analizadas
Descripción el caso de uso	
El caso de uso se inicia cuando el usuario o el administrador seleccionen la opción de Plotear en la cual el sistema muestra en el mapa de la provincia las casas analizadas según las coordenadas que hayan sido insertadas en las planillas.	
Requisitos Funcionales	
RF7.	

Tabla 28. Resumen de la descripción del CU Representar en un mapa las casas analizadas.

Nombre del caso de uso	Obtener cuantitativamente el costo de prevención
Objetivo	Saber la cantidad de presupuesto a destinar para realizar las medidas de prevención para los sismos a una determinada estructura.
Actores	Usuario, Administrador: Inserta o selecciona las medidas que desea tomar así como al tipo a que pertenece
Complejidad	Media
Nivel	Usuario
Precondiciones	Haberse realizado un estudio de vulnerabilidad.
Poscondiciones	Las medidas de recuperación a tomar y el costo en CUP.
Descripción el caso de uso	
Al realizarse el estudio de vulnerabilidad de la estructura el usuario o el administrador seleccionan las medidas más comunes que cumplan con la necesidad de la unidad en cuestión siempre que la misma las puede costear, sino, hay que tomar otras medidas. Tomando como base casos de estructuras similares, o puede introducir otras que el usuario desee seleccionando el grupo a la que pertenece la medida será insertada y almacenada en la base de datos.	
Requisitos Funcionales	
RF13.	

Tabla 29. Resumen de la descripción del CU Obtener cuantitativamente el costo de prevención.

Nombre del caso de uso	Obtener cuantitativamente el costo de recuperación
Objetivo	Saber la cantidad de presupuesto a destinar para realizar las medidas de recuperación para los sismos a una determinada estructura.
Actores	Usuario, Administrador: Inserta o selecciona las medidas que desea tomar así como al tipo a que pertenece
Complejidad	Baja
Nivel	Usuario
Precondiciones	Haberse realizado un estudio de vulnerabilidad.
Poscondiciones	Las medidas de prevención a tomar y el costo en CUP.
Descripción el caso de uso	
Al realizarse el estudio de vulnerabilidad de la estructura el usuario o el administrador seleccionan las medidas más comunes que cumplan con la necesidad de la unidad en cuestión siempre que la misma las puede costear, sino hay que tomar otras medidas. Tomando como base casos de estructuras	

similares, o puede introducir otras que el usuario desee seleccionando el grupo a la que pertenece la medida será insertada y almacenada en la base de datos.
Requisitos Funcionales
RF12.

Tabla 30. Resumen de la descripción del CU Obtener cuantitativamente el costo de recuperación.

Nombre del caso de uso	Gestionar medidas preventivas
Objetivo	Modificar o eliminar acciones preventivas contra los sismos
Actores	Administrador: Modifica o elimina acciones preventivas contra los sismos
Complejidad	Alta
Nivel	Administrador
Precondiciones	Debe haberse insertados medidas previamente
Poscondiciones	Se modificó o eliminó acciones preventivas contra los sismos.
Descripción el caso de uso	
El caso de uso se inicia cuando el administrador selecciona la cargar Medidas. El administrador puede modificar o eliminar medidas preventivas contra los sismos. El sistema permitirá además realizar una búsqueda por distintos criterios como nombre de la medida y costo.	
Requisitos Funcionales	
RF15, RF16	

Tabla 31. Resumen de la descripción del CU Gestionar acciones preventivas contra los sismos.

Nombre del caso de uso	Insertar medidas de prevención
Objetivo	Insertar acciones de prevención de los sismos
Actores	Usuario, Administrador: Inserta acciones de prevención de los sismos
Complejidad	Alta
Nivel	Usuario
Precondiciones	Debe haberse autenticado en el sistema.
Poscondiciones	Se insertó acciones de prevención de los sismos
Descripción el caso de uso	

El caso de uso se inicia cuando el usuario selecciona la opción Medidas. El usuario puede seleccionar alguna de las medidas que el sistema ofrece o insertar las suyas propias si desea.
Requisitos Funcionales
RF17.

Tabla 32. Resumen de la descripción del CU Insertar medidas de prevención.

Nombre del caso de uso	Gestionar medidas de recuperación
Objetivo	Modificar o eliminar acciones de recuperación de los sismos
Actores	Administrador: Modifica o elimina acciones de recuperación de los sismos
Complejidad	Alta
Nivel	Administrador
Precondiciones	Debe haberse insertados medidas previamente.
Poscondiciones	Se modificó o eliminó acciones de recuperación de los sismos
Descripción el caso de uso	
El caso de uso se inicia cuando el administrador selecciona la opción cargar Medidas. El usuario puede modificar o eliminar medidas de recuperación contra los sismos. El sistema permitirá además realizar una búsqueda por distintos criterios como nombre de la medida y costo.	
Requisitos Funcionales	
RF18, RF19.	

Tabla 33. Resumen de la descripción del CU Gestionar medidas de recuperación de los sismos.

Nombre del caso de uso	Insertar medidas de recuperación
Objetivo	Insertar acciones de recuperación de los sismos
Actores	Usuario, Administrador: Inserta acciones de recuperación de los sismos
Complejidad	Alta
Nivel	Usuario
Precondiciones	Debe haberse autenticado en el sistema
Poscondiciones	Se insertó acciones de recuperación de los sismos
Descripción el caso de uso	
El caso de uso se inicia cuando el usuario selecciona la opción Medidas. El usuario puede seleccionar alguna de las medidas que el sistema ofrece o insertar las suyas propias si desea.	

Requisitos Funcionales
RF17.

Tabla 34. Resumen de la descripción del CU Insertar medidas de recuperación.

Nombre del caso de uso	Autenticar usuario
Objetivo	Obtener acceso a la aplicación
Actores	Usuario, Administrador: Se autentica en el sistema
Complejidad	Alta
Nivel	Usuario
Precondiciones	Abrir el sistema.
Poscondiciones	Acceso a la funcionalidades del sistema
Descripción el caso de uso	
El caso de uso se inicia cuando el usuario abre el sistema. El usuario deberá autenticarse para poder acceder a las funcionalidades del sistema según los permisos que posea.	
Requisitos Funcionales	
RF20.	

Tabla 35. Resumen de la descripción del CU Autenticar Usuario

2.5 Prototipo de la interfaz principal de usuario

Para mostrar una vista preliminar de la solución se crearon prototipos de interfaz no funcionales que básicamente incluyen las características definidas de la propuesta de solución, y son de fácil modificación a partir de la retroalimentación del cliente. Los prototipos iniciales sufrieron muchos cambios a través de las iteraciones en el desarrollo del sistema, la interfaces finales pueden ser consultadas en el Anexo 1.

A continuación se muestra el prototipo de la pantalla principal del sistema donde a partir de las opciones brindadas se desplegarán diferentes pantallas que permitirán introducir los datos del estudio a realizar.

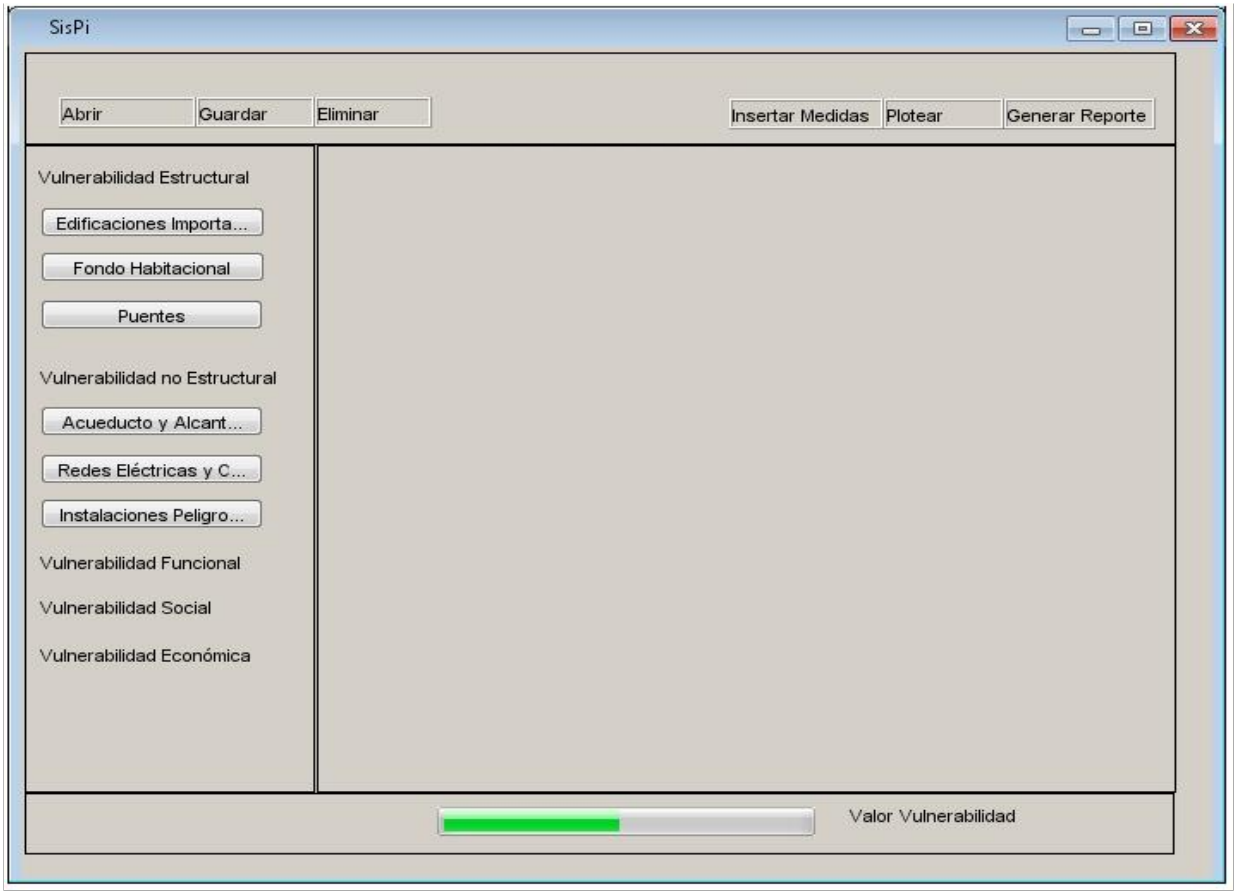

Figura 3. Prototipo interfaz de usuario principal.

2.6 Conclusiones Parciales

El análisis realizado permitió definir, entender y diseñar la propuesta de solución del software que lleva por nombre SISπ a partir del modelo de dominio. La modelación permitió la captura de veintiséis requerimientos funcionales, agrupados en diecisiete casos de uso y establecer dieciocho requerimientos no funcionales. Se realiza una breve descripción de las funcionalidades del software junto con los prototipos de interfaces para cada una.

Capítulo 3 Implementación y Pruebas

La etapa más importante de un software es el momento de la implementación de todos los requisitos funcionales que pasan a ser las funcionalidades que el sistema va a tener. El empleo de una metodología ágil para guiar el proceso de desarrollo permite que la implementación sea de manera continua para facilitarle al cliente la familiarización con la herramienta y poder sugerir algunas no conformidades e incrementar la visión de los desarrolladores y clientes sobre el programa. En el presente capítulo se detallan los patrones de diseño utilizados tanto los de tipo GRASP como los GoF, la arquitectura usada en la construcción del software así como los diagramas pertinentes para una mejor comprensión del sistema a desarrollar. Se describen las pruebas realizadas al sistema para tener constancia del alcance, las funcionalidades y calidad que el cliente necesita y espera del trabajo.

3.1 Patrón de Arquitectura

Un patrón de arquitectura de software no es más que un esquema genérico probado para solucionar un problema específico recurrente que surge en un cierto contexto (Crawler, 2013).

3.1.1 Patrón Modelo-Vista-Controlador

Patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos (Alvarez, 2012):

- Modelo: Es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- Vista: Presenta el modelo en un formato adecuado para interactuar con este, usualmente la interfaz de usuario.
- Controlador: Este responde a eventos, usualmente acciones del usuario, e invoca cambios en el modelo y, probablemente, en la vista.

El Patrón MVC se encuentra frecuentemente en aplicaciones Web, aunque también es aplicable en software de tipo de escritorio. Donde la Vista lo componen las interfaces de usuario y los códigos que modifican estas interfaces, esto se evidencia en las clases del paquete *gui*; en el Modelo se encuentran agrupados los datos necesarios para realizar las funcionalidades para las cuales el sistema se implementó como son las clases Puentes, Edificaciones por citar algún ejemplo; el Controlador es el responsable de recibir los eventos de entrada desde la Vista y resolver estos eventos asignándoselos a la clase que pueda cumplir con dichos eventos, la clase responsable de esta tarea en el software SISπ lo lleva la clase Control. Además se decide utilizar esta arquitectura por las ventajas siguientes (Álvarez, 2012):

- Facilita la organización de la aplicación.
- Es fácil crear representaciones de los mismos datos.
- Facilita la realización de pruebas unitarias para todos los componentes.
- Permite reutilizar los componentes.

Desventajas:

- La distribución de componentes ayuda a crear un mayor número de ficheros.

Se muestra un ejemplo de la relación entre algunas clases del sistema mediante esta arquitectura:

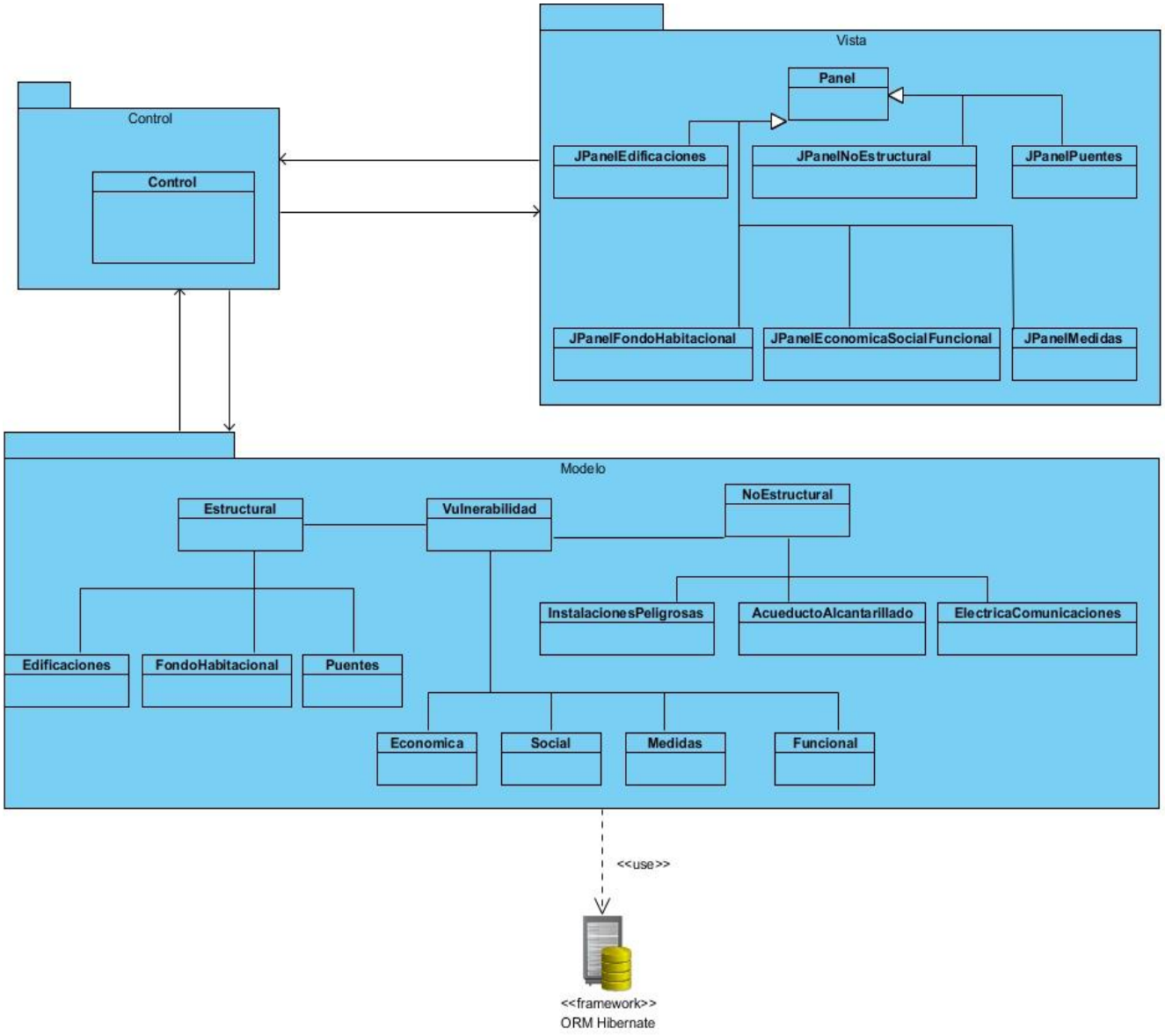

Figura 4. Ejemplo práctico del patrón MVC en el sistema *SISπ*.

3.2 Patrones de diseño

3.2.1 Patrones GRASP

Los Patrones Generales de Software para Asignación de Responsabilidades, GRASP por sus siglas en Ingles, describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable (Crawler, 2013).

Algunos de los patrones utilizados para la confección del sistema son:
Experto: Tiene la responsabilidad de realizar una labor. Es de la clase que tiene o puede tener los datos involucrados. Una clase contiene toda la información necesaria para realizar la labor que tiene encomendada (Savedra, 2013). Este patrón se evidencia en las clases del paquete modelo cada una contiene la información referente a la entidad que representa y es responsable de realizar la labor que tiene encomendada.

Controlador: Asigna a clases específicas la responsabilidad de controlar el flujo de eventos del sistema. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no las realiza, sino las delega en otras clases con las que mantiene un modelo de alta cohesión (Savedra, 2013). Este patrón se ve reflejado en la clase Control. La arquitectura Modelo Vista Controlador MVC brinda una capa específicamente para los controladores, que son el núcleo de este, y especifica la presencia de este patrón.

3.2.2 Patrones GoF

Los patrones Pandilla de los Cuatro GoF por sus siglas en Ingles, se clasifican en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

Para el diseño de la herramienta se tuvieron en cuenta los siguientes patrones de comportamiento:
Cadena de responsabilidades: Proporcionar a más de un objeto la capacidad de atender una petición, para así evitar el acoplamiento con el que objeto que hace la petición. Se forma con estos objetos una cadena, en la cual cada objeto o satisface la petición o la pasa al siguiente (Universidad Politécnica de Madrid, 2011). Este patrón se ve reflejado en la clase NoEstructural debido a que permite a los objetos de las clases hijas atender la petición de la vulnerabilidad.

Estado: Permitir a un objeto modificar su comportamiento a medida que su estado interno va cambiando, dando así la impresión de que el objeto “cambia de clase” (Universidad Politécnica de Madrid,

2011). Este patrón se ve reflejado en las clases hijas de NoEstructural y Estructural debido a mientras que varíen los atributos de la clase en el cálculo se ejecuta de manera diferente.

Mediador: Define un objeto que encapsula la forma en que interactúan un grupo de objetos, promoviendo así un acoplamiento débil al evitar las referencias explícitas entre los objetos y permitiendo, por tanto, que su interacción se modifique de forma independiente (Universidad Politécnica de Madrid, 2011).Este patrón se ve reflejado en la clase Control debido a que se encarga de asignar qué clase interactúa con qué objeto.

3.3 Bibliotecas utilizadas

Para la generación de los reportes se utiliza la biblioteca iText. Debido a que sin intermedio de herramientas de diseño, al ser utilizada directamente mantiene gran parte de sus potencialidades como organización por secciones, capítulos, utilización de diferentes tipologías de letras, inserción de tablas e imágenes. iText es una biblioteca de código abierto de Java para la generación y la manipulación de PDF. Puede ser utilizado para crear documentos en este formato a partir de cero, para rellenar formularios PDF interactivos, para acabar nuevos contenidos en documentos PDF existentes así como para dividir y combinar documentos PDF existentes (Lowagie, 2013).

3.4 Estándares de codificación

Se utiliza como estándar de codificación el llamado CamelCase que es la práctica de escribir frases o palabras compuestas eliminando los espacios y poniendo en mayúscula la primera letra de cada palabra. El nombre viene del parecido entre las demás letras, con las jorobas de los camellos (Cunningham, 2015). Existen dos tipos de CamelCase (Cunningham, 2015):

- UpperCamelCase, cuando la primera letra de todas está escrita en mayúscula.
  Ejemplo clase SISπ: AcueductoAlcantarillado
- LowerCamelCase, cuando la primera letra de todas está escrita en minúscula.
  Ejemplo variable SISπ: _tipoSuelo

CamelCase es el estándar más utilizado para los proyectos de código java pues propone nombrar las clases usando siempre nombres significativos permitiendo que cualquier persona que lea el código pueda comprenderlo. Este estándar propone usar sangría a la hora de escribir para hacer la lectura más cómoda así como apoyarse en el uso de comentarios para poder obtener información adicional en una línea de código. Crear la documentación Javadoc para facilitar la continuación, o soporte, por otros desarrolladores al programa. Otro aspecto importante en cuanto a los estándares que se ha tomado como convención es declarar las variables de las clases privadas y obtener acceso a ellas mediante métodos públicos como los get y set por citar algún ejemplo. Otro estándar optado en el sistema fue hacer uso de los paquetes

donde se organizaron cada clase en un paquete para facilitar la organización y evidenciar la arquitectura y las sentencias de importación en los paquetes que la necesitan se encuentran en el inicio de las clases.

3.5 Diagrama clases de diseño

Una clase del diseño y sus atributos participan en varias realizaciones de casos de uso. En un diagrama de clases del diseño se exponen las clases que intervienen en las realizaciones de los casos de uso del sistema. En este tipo de diagrama se representa un nivel de detalle más alto que en los diagramas de clases del análisis, relacionándose con el lenguaje de programación del cual se hará uso en la implementación del sistema (Carvajal, 2009).

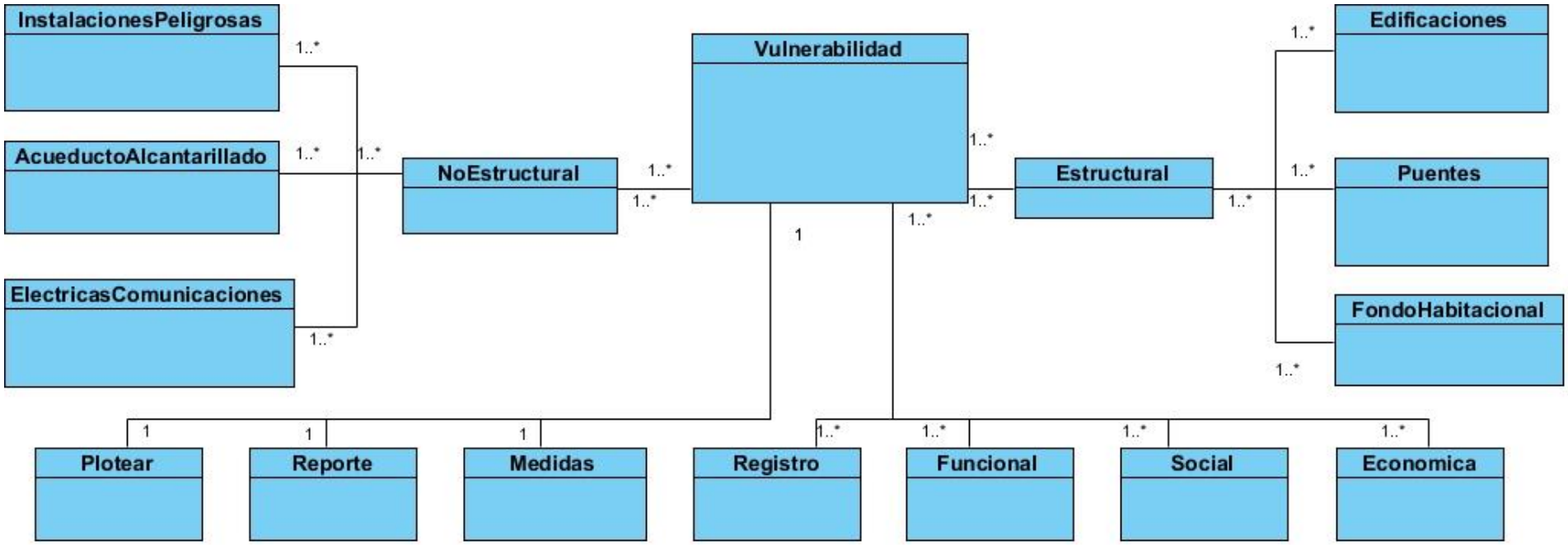

Figura 5. Diagrama de clases del diseño del sistema SISπ[1].

[1] Ver Clases en el anexos 4.

3.6 Modelo de datos del sistema

Para el modelado de los datos se generó el artefacto Diagrama Entidad-Relación, conformado por veintitrés tablas.

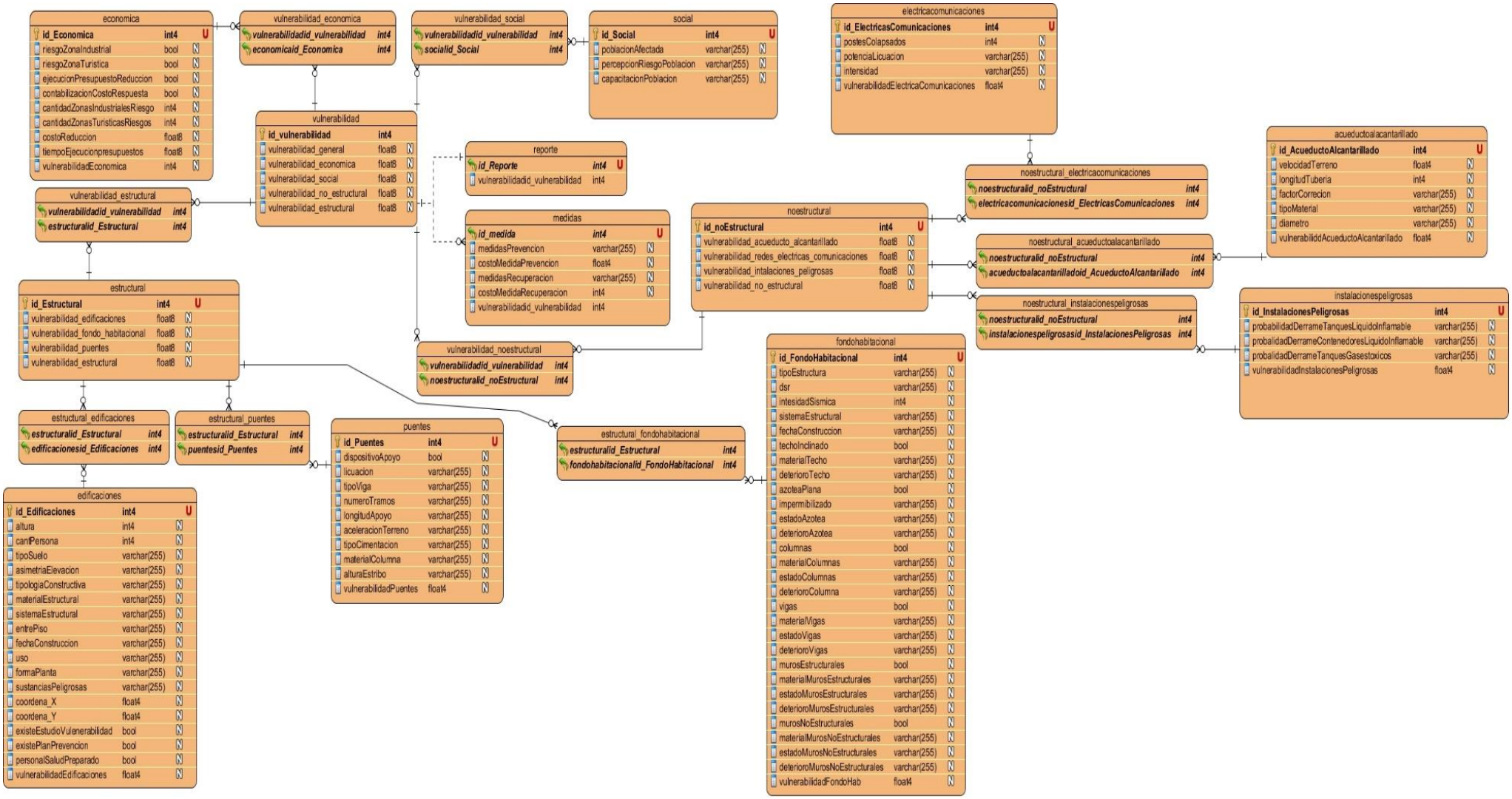

Figura 6. Diagrama Entidad-Relación

3.7 Pruebas

Durante la fase de pruebas del ciclo de ingeniería de software se hace necesario realizar pruebas, automatizarlas es una garantía y una ventaja para el desarrollador, facilitando enfocarse en la prueba y el resultado esperado, y dejando a la herramienta la creación de las clases que permiten coordinar las pruebas. Para ello se seleccionó JUnit; un conjunto de clases (framework) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. En función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que devolvió el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

3.7.1 Pruebas unitarias automatizadas

Las pruebas unitarias o también llamadas pruebas de caja blanca se basan en realizar pruebas al código del sistema. Para llevar a cabo esta tarea se comprueban los caminos lógicos de la aplicación mediante casos de prueba, que pongan a prueba los algoritmos implementados. Las pruebas unitarias no se le pueden realizar a todo el código de la aplicación, ya que el número de caminos lógicos puede llegar a crecer de manera exponencial lo cual imposibilita realizar casos de pruebas para todos estos caminos y mucho menos procesarlos todos (Extreme Testing, 2013). Por este motivo las pruebas de caja blanca se realizan a los principales algoritmos o procedimientos.

Uno de los métodos o técnicas de prueba unitarias, es la prueba del camino básico. Esta técnica permite obtener una medida de la complejidad de un procedimiento o algoritmo y un conjunto básico de caminos de ejecución de este, los cuales luego se utilizan para obtener los casos de prueba. Esta técnica será la utilizada para desarrollar los casos de pruebas unitarias de la herramienta desarrollada. De igual manera existen varias métricas de software para realizar pruebas unitarias, entre estas se encuentra la complejidad ciclomática, la cual será utilizada junto a la técnica explicada anteriormente. Esta métrica proporciona una medición cuantitativa de la complejidad lógica de un procedimiento.

La complejidad ciclomática cuando se utiliza en el contexto del método de prueba del camino básico, el valor que se calcula como complejidad ciclomática define el número de caminos independientes dentro de un fragmento de código y determina la cota superior del número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez (Pressman R. , 2002).

Para realizar las pruebas unitarias existen distintas herramientas, en específico para las pruebas al código en lenguaje Java existe JUnit y TestNG, ambos son frameworks utilizados para automatizar las pruebas;

permitiendo que el esfuerzo y el trabajo en la fase de pruebas se reduzcan.

A continuación se explica, mediante un ejemplo, todo el procedimiento que se siguió para obtener los casos de prueba utilizando la técnica del camino básico junto con la métrica complejidad ciclomática.
Este algoritmo lleva como nombre “Calcular”, y tiene como finalidad calcular la vulnerabilidad no estructural en las redes de acueducto y alcantarillado.

A partir del algoritmo, se dibuja el grafo de flujo asociado.

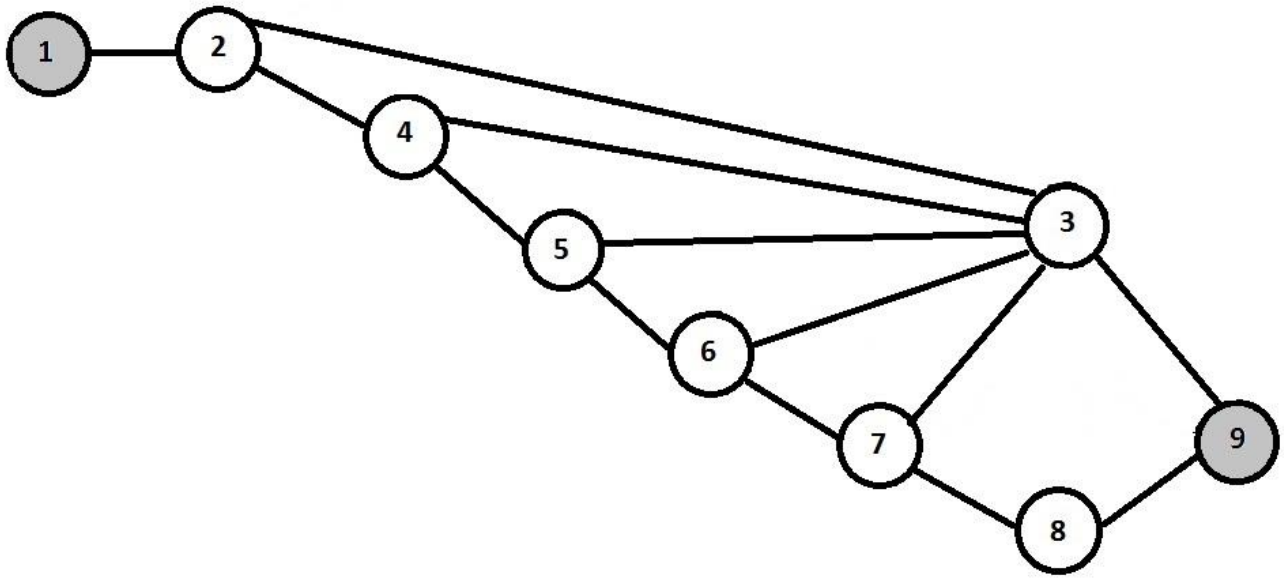

Figura 7. Grafo flujo funcionalidad Calcular.

Luego se determina la complejidad ciclomática V(G) del grafo resultante G, para esto se utiliza la siguiente fórmula:
V(G) = A - N + 2
Donde:

- A es el número de aristas del grafo
- N es el número de nodos.

V(G) = 13 – 9 + 2 = 6.

El número de caminos independientes de la estructura del programa es igual a 6, y los caminos independientes son:
Camino 1: 1,2,4,5,6,7,8,9

Camino 2: 1,2,3,9

Camino 3: 1,2,4,3,9

Camino 4: 1,2,4,5,3,9

Camino 5: 1,2,4,5,6,3,9

Camino 6: 1,2,4,5,6,7,3,9

Luego se definen 6 casos de prueba para el código del algoritmo, uno para cada camino. Para realizar estos casos de prueba se utilizó la siguiente plantilla. En la sección correspondiente a los Anexos se muestran los casos de pruebas unitarias correspondientes a este algoritmo. (Ver Anexo 2).

Caso de prueba unitaria
Camino: nombre
Caso de prueba: nombre
Entrada: descripción textual de lo que ocurre en el mundo real que hace necesario ejecutar el caso de prueba, precisando la data de entrada y los comandos a dar por el actor. Descripción textual del estado de la información almacenada
Resultado: descripción textual del estado en el que queda la información y las alertas que puedan generarse, una vez ejecutado el caso de uso con los valores y el estado especificado en la entrada
Condiciones: condiciones que deben cumplirse mientras se ejecuta el caso de prueba

Tabla 36. Plantilla de caso de prueba unitaria.

3.7.2 Pruebas aceptación

Las pruebas de aceptación son un tipo de prueba de caja negra orientadas a evaluar las distintas tareas en las que ha sido dividido un caso de uso. Para asegurar el funcionamiento final de un determinado caso de uso las pruebas son creadas y usadas por los clientes para comprobar que éstas cumplen su cometido.

Un caso de uso puede tener una o varias pruebas de aceptación y no está completo hasta no haberlas pasado todas. El cliente es el máximo responsable de verificar cada una de las pruebas y de priorizar la corrección de las pruebas fallidas.

La utilización de estas pruebas fue de vital importancia en el proceso de desarrollo ya que permitió al programador tener una idea más clara sobre la calidad del trabajo, y se garantizó la entrega de un producto acorde con las necesidades del usuario final.

A continuación se muestra la plantilla de caso de prueba de aceptación definida por el cliente:

Caso de prueba aceptación	
Código: código que identifica la prueba.	**Caso de uso**: nombre del caso de uso relacionado con la prueba que se realiza.
Descripción: breve descripción del objetivo con que se realiza la prueba.	
Condiciones de Ejecución: condiciones que deben satisfacerse para que pueda realizarse la prueba.	
Entrada/Pasos de Ejecución: se describen los pasos de ejecución de la prueba en cuestión.	
Resultado Esperado: proporciona las expectativas ideales para las cuales fue pensada la prueba.	
Evaluación de la Prueba: calificación que recibe la prueba de acuerdo con los resultados obtenidos.	

Tabla 37. Plantilla de caso de prueba aceptación.

Iteración 1

Para la primera iteración se realizaron 19 casos de pruebas de aceptación, identificando 15 no conformidades, 9 no significativas y 6 significativas. Debido a la baja complejidad de éstas, fueron todas resueltas y no quedaron casos de pruebas pendientes para la siguiente iteración.

Iteración 2

Para esta iteración se realizaron 10 casos de pruebas de aceptación, identificando 4 no conformidades, de ellas 1 significativa y 3 no significativa. Estas no conformidades se solucionaron dando fin al proceso de pruebas de aceptación y obteniendo resultados satisfactorios.

Las no conformidades, no significativas, se centraron en errores ortográficos como: omisiones de tildes, paréntesis, cambio de mayúscula por minúscula, espacio entre palabras y aceptar caracteres donde se esperaban valores numéricos, y las significativas, en errores de validación, cambios en el diseño y 2 fórmulas que presentaron errores matemáticos. Estas últimas se corrigieron primero debido a que tenían alta prioridad.

En el Anexo 3 se muestran un caso de prueba de aceptación correspondiente a cada iteración.

3.7.2.1 Resultados de las pruebas de aceptación

	Iteración 1	Iteración 2
CP de Aceptación	19	10
No conformidades	15	4
Significativas	6	1
No significativas	9	3
Resueltas	15	4
Pendientes	0	0

Tabla 38. Resultado de las pruebas de aceptación.

3.7.3 Pruebas a partir de casos de estudio

Luego de haber realizado las pruebas que define la metodología de desarrollo de software seleccionada, se decide además, acorde a las peticiones del cliente y de la tutora del trabajo de diploma, realizar pruebas con estudios reales. El objetivo principal de este tipo de pruebas es garantizarle al cliente un producto terminado que posea como principal característica la confiabilidad.

Según los estudios que llevan a cabo las consultorías de la ENIA actualmente, fue posible comprobar el funcionamiento de la herramienta. Por la situación actual que presenta el cliente, descrita al principio de la investigación, solo fue posible rescatar dos de los estudios realizados hasta el momento, acorde con la capacidad total del estudio. Estos dos estudios son los que brindan todos los datos necesarios para comprobar los resultados arrojados por la herramienta en cuanto al amplio análisis que brinda la herramienta. Por lo que el cliente estuvo de acuerdo probarlo con solo dos estudios asumiendo que para el resto arrojará resultados satisfactorios.

Estudios	Plantillas rellenadas	Cálculos realizados	Valorados correctamente	Valorados incorrectamente
Ueb Ron El Valle	10	58	58	0
Centro Meteorológico Provincial	10	58	58	0

Tabla 39. Pruebas a partir de casos de estudio.

3.8 Conclusiones Parciales

En el presente capítulo se aborda aspectos esenciales de la fase de diseño, implementación y pruebas empezando por la selección de la arquitectura para la cual se escoge el patrón MVC, luego se hace alusión a los patrones de diseño Grasp y GoF poniendo ejemplos prácticos de su utilización en el desarrollo del sistema. Se realizó un diagrama de clases de diseño compuesto por dieciséis clases para darle cumplimiento a las funcionalidades definidas en la fase de análisis. A partir del diseño se realiza el modelo de datos mostrando la estructura de la base de datos.Por último se describe el proceso de pruebas, ejecutándose pruebas unitarias automatizadas a los principales procedimientos del código, Veintinueve pruebas de aceptación por parte del cliente y en el sistema fueron probados dos casos de estudio, obteniendo resultados satisfactorios que garantizan que el producto final cumpla con las expectativas del cliente.

CONCLUSIONES GENERALES

Con el fin de esta investigación se concluye los siguiente:

- Se obtuvo una comprensión sobre la situación y las tendencias de los sistemas para la evaluación de la prevención, vulnerabilidad y recuperación de los sismos, permitiendo así encontrar una solución al problema planteado.
- El análisis de los sistemas homólogos evidenció la necesidad de desarrollar un nuevo sistema y permitió incluir funcionalidades comunes en este tipo de herramientas.
- El uso de la metodología AUP para guiar el proceso de desarrollo de la herramienta y los artefactos generados en cada una de sus fases propició una mejor comunicación con el cliente, logrando obtener un mejor funcionamiento del sistema.
- Se desarrolló una herramienta que resuelve las limitantes que presenta la Consultoría Pro-Ambiente de la ENIA de Pinar del Río, con respecto a la pérdida y recopilación de información, evaluación de la vulnerabilidad ante los sismos y el almacenamiento automatizado de los estudios.
- Se comprobó la funcionalidad y confiabilidad de la aplicación mediante la realización de pruebas basadas en casos de estudio, pruebas unitarias automatizadas y de aceptación.

RECOMENDACIONES

Como resultado del proceso de investigación y desarrollo del software, han surgido ideas que serían recomendables tener en cuenta para su futuro perfeccionamiento, por lo que se le recomienda:

A los futuros desarrolladores que trabajen en el sistema:

- Incorporar análisis estadísticos como por ejemplo las zonas de mayor vulnerabilidad estructural ante un sismo.
- Utilizar la funcionalidad de localización geográfica de manera nativa en la aplicación.
- Crearle una ayuda a la herramienta para facilitar su uso en próximos despliegues.

Al cliente:

- Realizar una investigación para determinar nuevas funcionalidades que se puedan agregar a la herramienta.
- Extender el uso de la aplicación a todas las entidades de la ENIA con el objetivo de reducir costos durante la prestación de este importante servicio así como mejorar la calidad del mismo en cada una de ellas.

A la Universidad de las Ciencias Informáticas:

- Usar este trabajo como material de estudio en la creación de alguna herramienta similar.

REFERENCIAS BIBLIOGRÁFICAS

1. **Álvarez, M. A.** (23 de marzo de 2012). Desarrollo web. Disponible en: http://desarrolloweb.com

2. **Carpa.** (3 de febrero de 2013). Software de Desastres. Disponible en: http//www.ecapra.org/es/software

3. **CENAIS.** (2009). *Determinación del peligro, vulnerabilidad y riesgo*. Ministerio, tegnología y medio ambiente, 4-29.

4. **Catalina, E. Q.** (2003). *Sistemas operativos y lenguajes de programación*. Madrid, España: Paraninfo.

5. **Carvajal Pérez, E.** (2009). *Análisis y diseño del subsistema de Análisis de Resultados de un Simulador de Procesos Químicos*. La Habana.

6. **Cortizo Pérez, J. C.** (2004). EXtreme Programming. España: s.n. (Geocities, 2010) Sanabria, E. C. Java Básico. Universidad Católica de Colombia, 6.

7. **Crawler, W.** (25 de abril de 2013). Scribd. Disponible en: http://es.scribd.com/doc/34279909/PuntosExamen-Patrones-Arquitectonicos-Secc01

8. **Crawler, W.** (25 de abril de 2013). Willydev. Disponible en: http://www.willydev.net/InsiteCreation/v1.0/WillyCrawler/2008.05.01.Articulo.Introduccion%20a%20la%20

9. **Crawler, Willy.** (25 de abril de 2013). Scribd. Disponible en: http://es.scribd.com/doc/34279909/PuntosExamen-Patrones-Arquitectonicos-Secc01

10. **Cunningham, W.** (27 de febrero de 2015). Diclib. Disponible en: http://www.diclib.com/cgibin/d1.cgi?l=es&base=es_wiki_10&page=showid&id=30673

11. **Domínguez Pérez, J**. (2010). *Introducción a los sistemas Gestores de Base de Datos*. Octubre 2010. ISSN: 1988 6047.

12. **Eclipse**. -- Sitio Oficial Disponible en: www.eclipse.org

13. **EMNDC.** (1999). *Normas para la proyección y ejecución de las medidas tecnicos ingenieras de defensa civil*. Manual de Defensa Civil, 112-170.

14. **Extreme Testing.** (1999). [Consultado el: 22 de marzo de 2013.] Disponible en: http://www.xprogramming.com/publications/SP99 Extreme for Web.pdf

15. **Ferri-Benedetti, F**. (14 de junio de 2014). Softonic. Disponible en: http://www.Quakeshakes.com

16. **Flores, L**. (2000). *Metodologías ágiles (Proceso Unificado Ágil).* Bolivia.

17. **Howden, M.** (5 de junio de 2013). Sahana Foundation. Obtenido de Sahana Foundation: http://sahanafoundation.org/

18. **IDE NetBeans**. - Sitio Oficial. Disponible en: https://netbeans.org/features/index.html

19. **Jacobson Ivar, G. B.** (2000). *El proceso unificado de desarrollo de software*. Madrid.

20. **López, J. M.** (15 de junio de 2014). Softonic internacional Disponible en: http://www.EarthAlerts.com

21. **Lowagie, B.** (12 de abril de 2013). Programmable PDF Software. Obtenido de Programmable PDF Software: http://itextpdf.com

22. **Microsoft.** (11 de diciembre de 2013). Microsoft Solutions Framwork. Disponible en: http://msdn.microsft.com

23. **Miranda, Y. R.** (15 de mayo de 2010.). Desarrollo de la Ampliación del Portal Web del CICPC. La Habana, Cuba.

24. **Navarro, D.** (14 de junio de 2014). Softonic. Disponible en: http://www.earthquake3d.com

25. **PostgreSQL**.- Sitio Oficial. Disponible en: http://www.postgresql.org

26. **Pressman, R**. (2002). *Ingeniería del Software. Un enfoque práctico* (5ta ed.). [Consultado el 30 de mayo de 2013], Disponible en: http://boards5.melodysoft.com/UBV_INGS/re-tema-metodologias-agiles-de-desarrollo-44.html.

27. **Rivera, M.** (2008). *Evacuación de la población de la ciudad de esmeraldas ante una emergencia*. Instituto de altos estudios nacionales IAEN, 7-60.
28. **Softonic**, **E. D**. (15 de junio de 2014). Softonic. Disponible en: http://www.EarthBrowser.com

29. **Sandi, H**. (1983). *"Earthquake risk and earthquake preparedness: some qualitative aspects and quantification possibilities"*, Proceedings of the Seminar on Earthquake Preparedness. UNDP/UNESCO/UNDRO Project for Earthquake Risk Reduction in the Balkan Region, Athens, 79-93.

30. **Savedra, J.** (25 de abril de 2013). El Mundo Informático. Disponible en: http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman.

31. **Universidad Politécnica de Madrid.** (2011). *Patrones del "Gang of Four.* Madrid.

32. **Valdés, M. E.** (2011.). *Desarrollo del módulo iTopics de la plataforma*. La Habana, Cuba.

33. **William, M.** (2002). *Vocabulario Controlado sobre Desastres*. Centro Regional de Información sobre Desastres, 74-100.

ANEXOS

Anexo 1: Interfaces de usuario

En la siguiente interfaz de usuario se puede observar que se han introducido los datos para realizar un estudio de una edificación de importancia este estudio se lleva a cabo para tener en cuenta cuan vulnerable es esta edificación que se considera importante ante la ocurrencia de un sismo evaluándose la altura de la instalación, el tipo de suelo en el que se encuentra construida entre otras es necesario señalar que para poder realizar un análisis de cualquiera de las vulnerabilidades que el sistema permite calcular es necesario contar con un usuario con acceso al sistema. Existen actualmente dos tipos de permisos en la aplicación el **usuario** que solo tiene permisos de guardar el trabajo realizado y puede ser creado por cualquiera con acceso mientras que el otro tipo de permiso es el **administrador** que solo puede ser concedido por otro administrador y tiene un control total sobre el software.

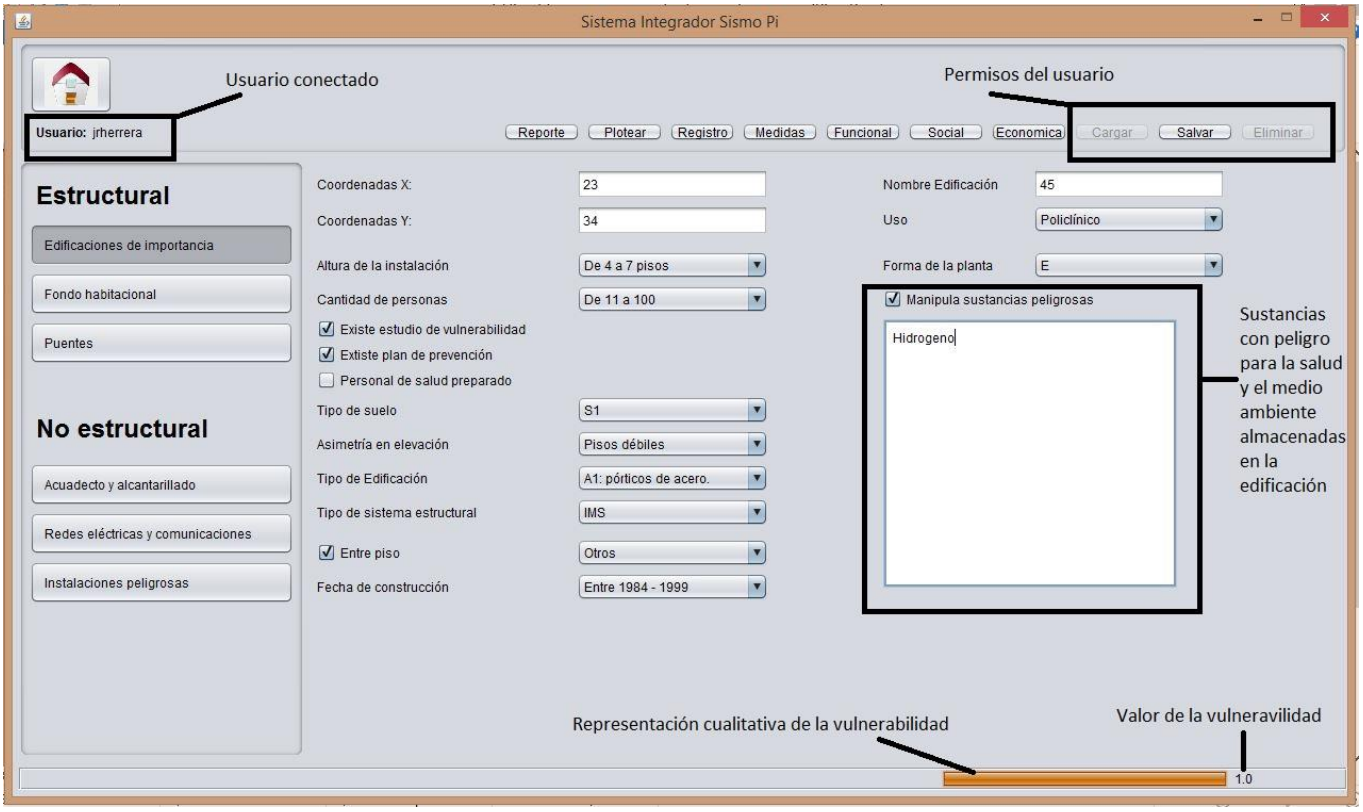

Figura 1.1 Datos de un proyecto en SISπ.

En la interfaz siguiente aparece la lista de medidas con la cual cuenta la aplicación para facilitar el trabajo a los especialistas involucrados en el estudio la cual se habilita con la opción seleccionar medidas estas medidas están separadas en dos grupos recuperación y prevención. El software también permite insertar otras medidas así como el costo de llevarlas a cabo por si el especialista necesita otras medidas que se ajusten a la instalación en cuestión.

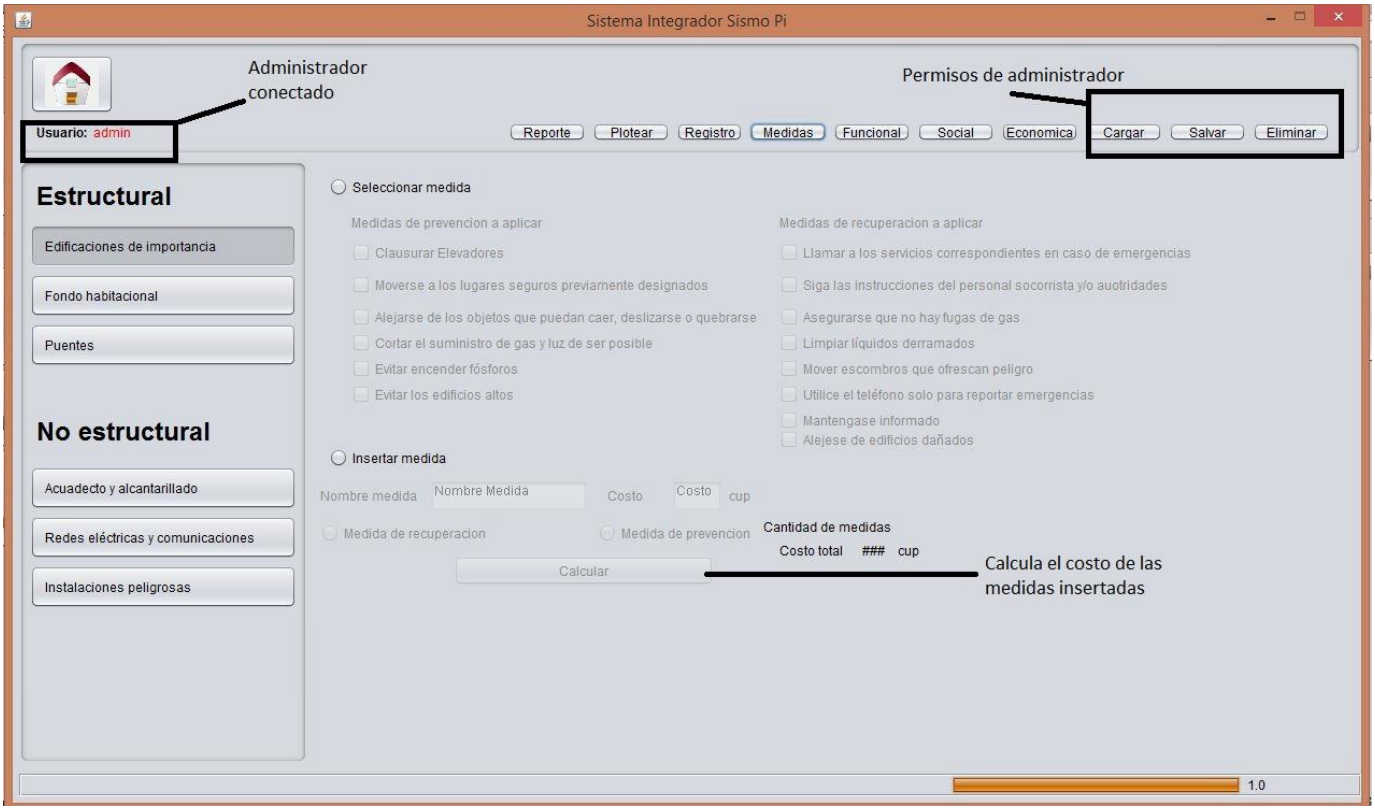

Figura 1.2. Medias a aplicar en un estudio.

Anexo 2: Casos de pruebas unitarias

Caso de prueba unitaria
Camino: *1,2,4,5,6,7,8,9*
Caso de prueba: *prueba unitaria camino 1*
Entrada: *El actor introduce la velocidad del terreno, la longitud de tubería que queda en la zona afectada por el sismo y selecciona el factor de corrección de la tubería así como el tipo de material y el diámetro del mismo entonces el software se encarga de calcular los valores asociados a cada uno de los datos y procede a calcular la vulnerabilidad que presentan las redes de acueducto y alcantarillado al presenciarse un sismo de esa magnitud.*
Resultado: *El sistema devuelve como salida la vulnerabilidad en un valor de -5 porque los valores obtenidos se salen de los parámetros de la metodología esto le permite al especialista a cargo del estudio tener una idea de lo que sucede con este sismo ya sea porque los valores son muy alarmantes o insignificantes de cualquier manera es una advertencia para el especialista.*
Condiciones: *Se rellenen los datos de los formularios*

Tabla 2.1. Caso de prueba unitaria del camino 1.

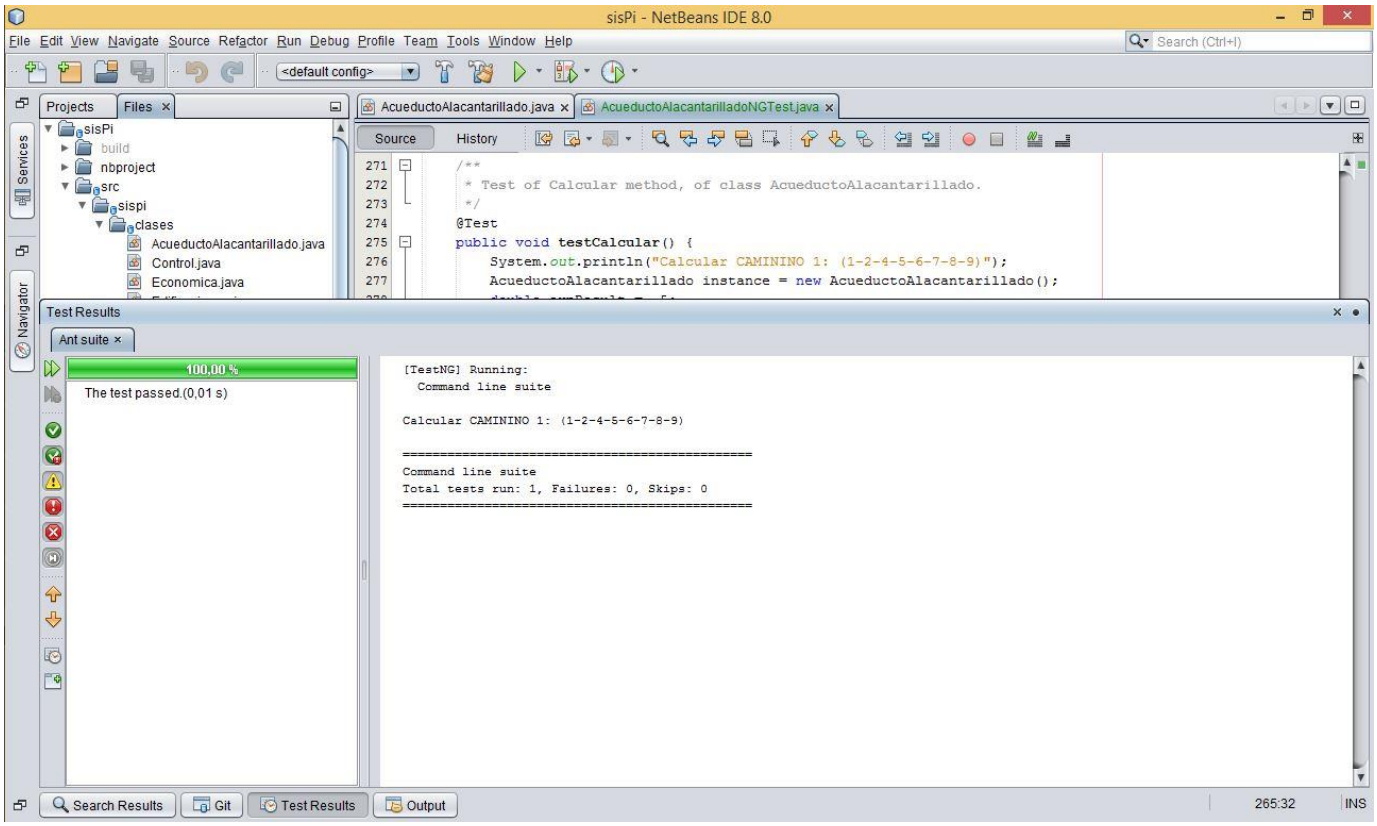

Figura 2.1. Prueba unitaria automatizada del camino 1.

Caso de prueba unitaria
Camino: *1,2,3,9*
Caso de prueba: *prueba unitaria camino 2*
Entrada: *El actor introduce la velocidad del terreno, la longitud de tubería que queda en la zona afectada por el sismo y selecciona el factor de corrección de la tubería así como el tipo de material y el diámetro del mismo entonces el software se encarga de calcular los valores asociados a cada uno de los datos y procede a calcular la vulnerabilidad que presentan las redes de acueducto y alcantarillado al presenciarse un sismo de esa magnitud.*
Resultado: *El sistema devuelve como salida la vulnerabilidad en un valor de 0 siendo no vulnerables las redes ante este tipo de evento adverso*
Condiciones: *Se rellenen los datos de los formularios*

Tabla 2.2. Caso de prueba unitaria del camino 2.

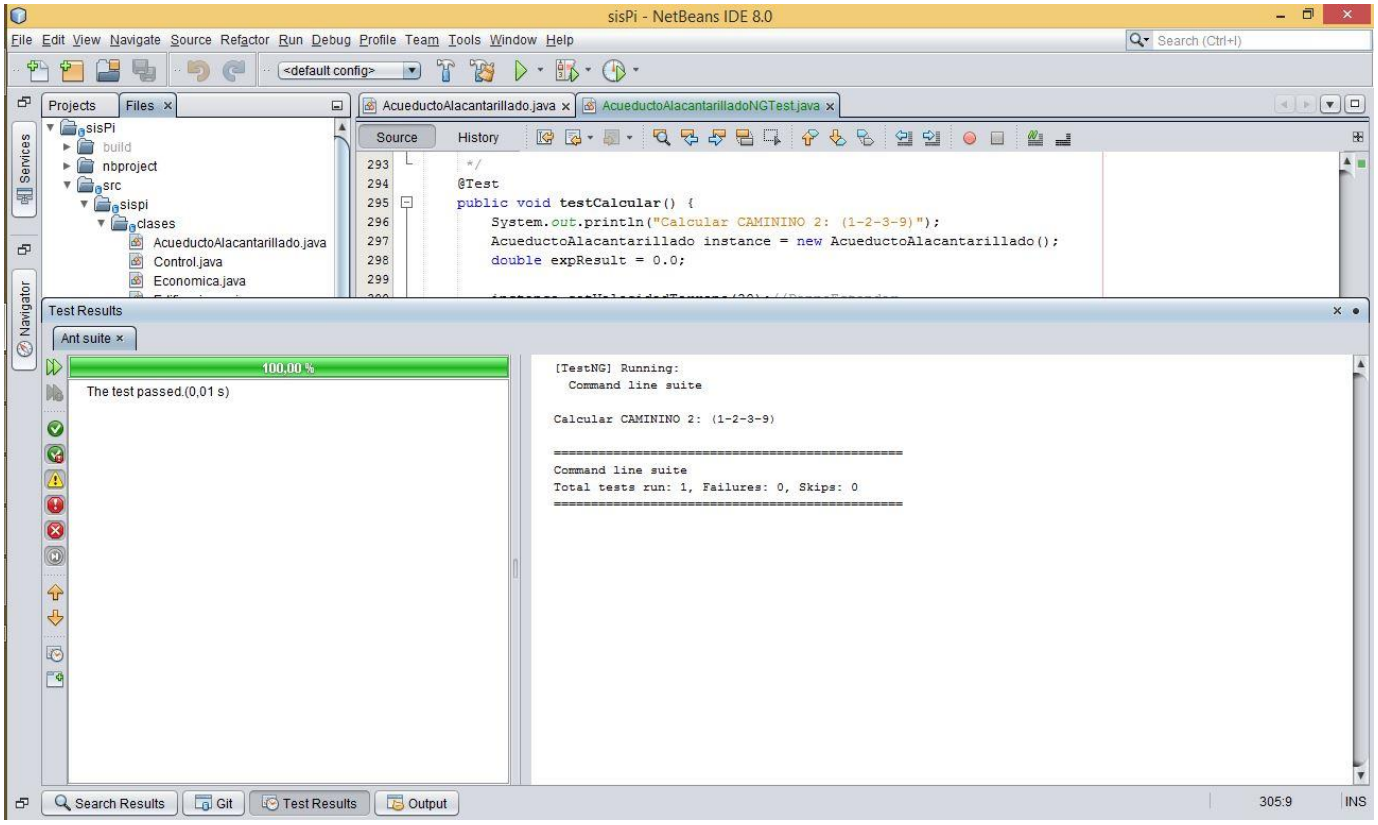

Figura 2.2. Prueba unitaria automatizada del camino 2.

Caso de prueba unitaria
Camino: *1,2,4,3,9*
Caso de prueba: *prueba unitaria camino 3*
Entrada: *El actor introduce la velocidad del terreno, la longitud de tubería que queda en la zona afectada por el sismo y selecciona el factor de corrección de la tubería así como el tipo de material y el diámetro del mismo entonces el software se encarga de calcular los valores asociados a cada uno de los datos y procede a calcular la vulnerabilidad que presentan las redes de acueducto y alcantarillado al presenciarse un sismo de esa magnitud.*
Resultado: *El sistema devuelve como salida la vulnerabilidad en un valor de 0.10 teniendo una leve vulnerabilidad.*
Condiciones: *Se rellenen los datos de los formularios*

Tabla 2.3. Caso de prueba unitaria del camino 3.

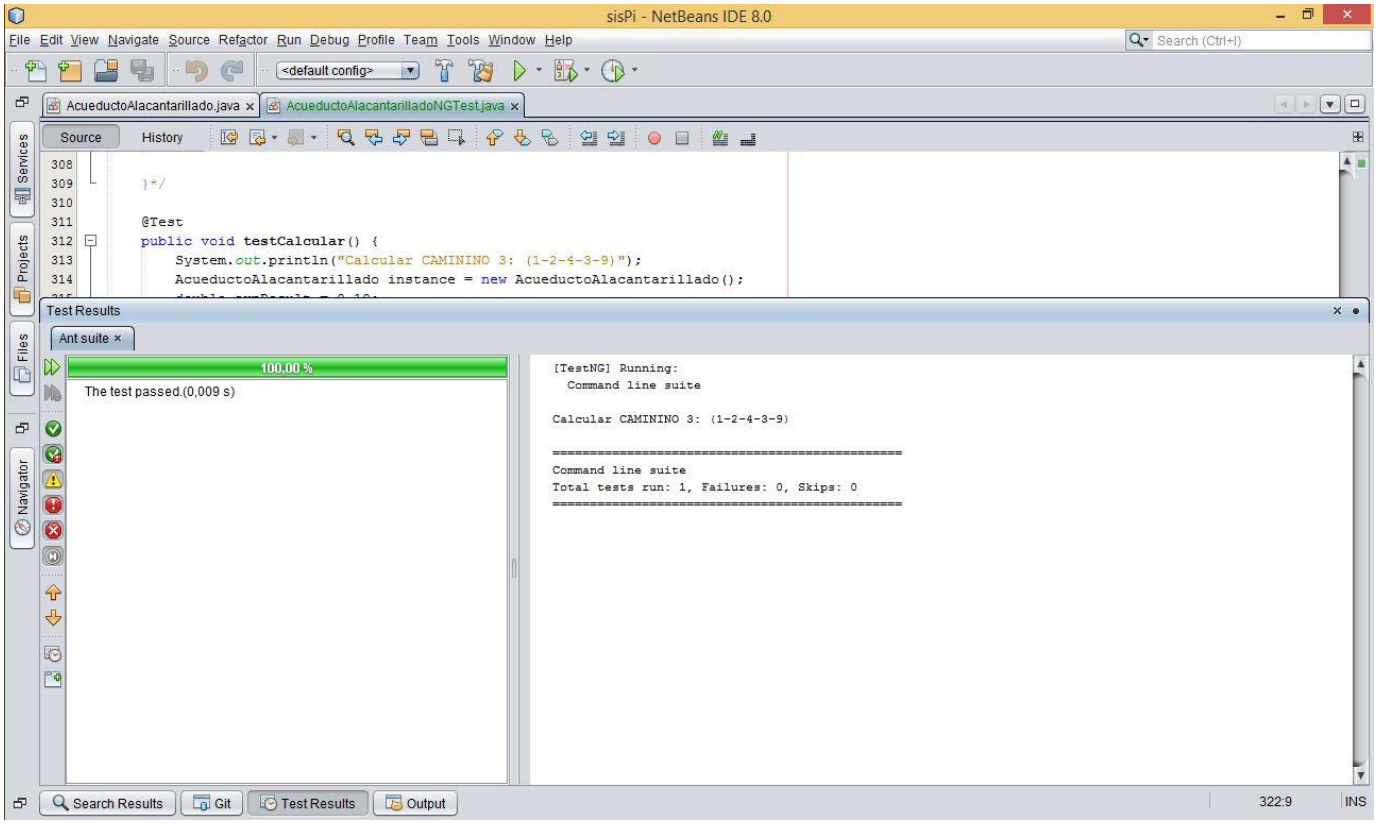

Figura 2.3. Prueba unitaria automatizada del camino 3.

Caso de prueba unitaria
Camino: *1,2,4,5,3,9*
Caso de prueba: *prueba uitaria camino 4*
Entrada: *El actor introduce la velocidad del terreno, la longitud de tubería que queda en la zona afectada por el sismo y selecciona el factor de corrección de la tubería así como el tipo de material y el diámetro del mismo entonces el software se encarga de calcular los valores asociados a cada uno de los datos y procede a calcular la vulnerabilidad que presentan las redes de acueducto y alcantarillado al presenciarse un sismo de esa magnitud.*
Resultado: *El sistema devuelve como salida la vulnerabilidad en un valor de 0.20 teniendo una vulnerabilidad moderada ya debe llamar la atención de los especialistas y empezar a buscar alternativas.*
Condiciones: *Se rellenen los datos de los formularios*

Tabla 2.4. Caso de prueba unitaria del camino 4.

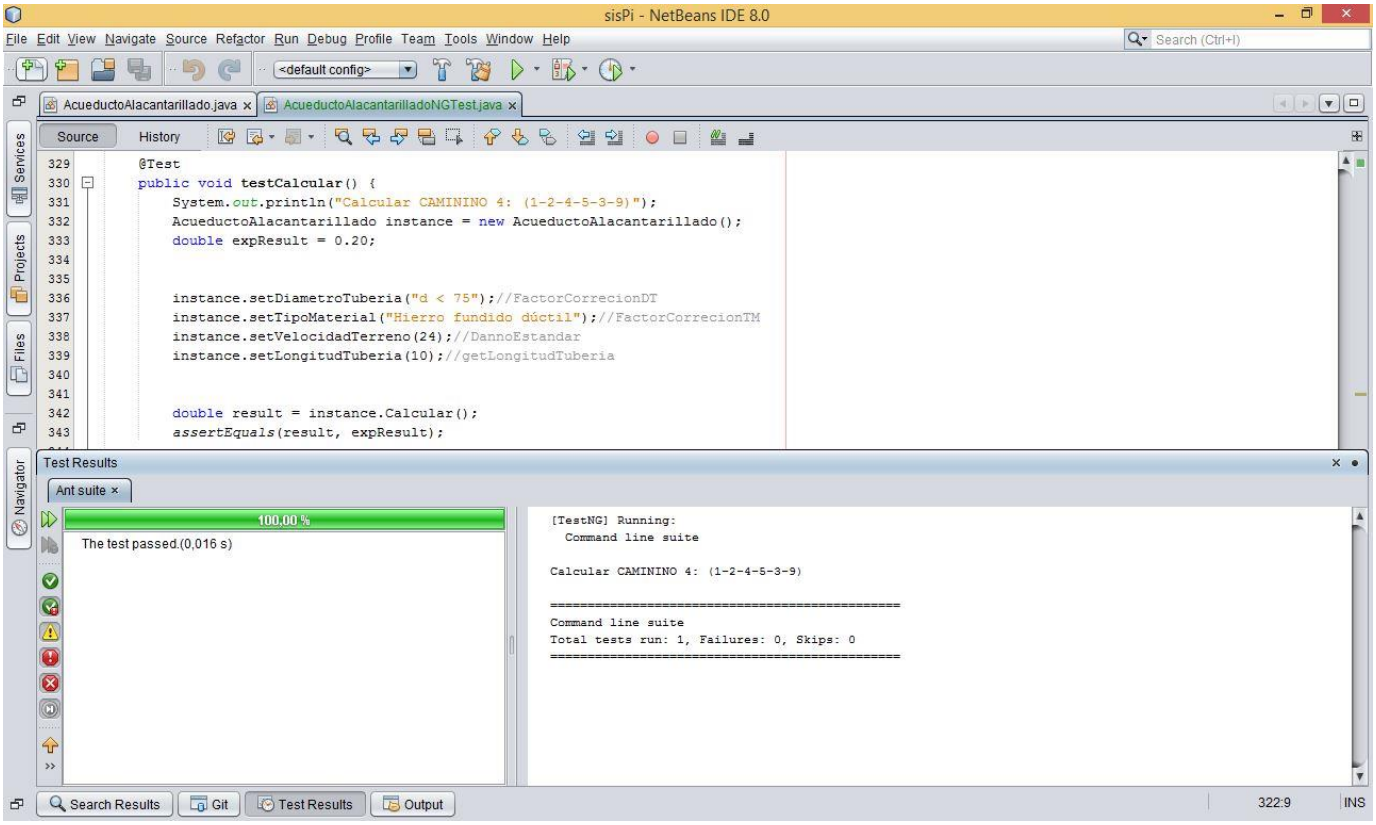

Figura 2.4. Prueba unitaria automatizada del camino 4.

Caso de prueba unitaria
Camino: *1,2,4,5,6,3,9*
Caso de prueba: *prueba uitaria camino 5*
Entrada: *El actor introduce la velocidad del terreno, la longitud de tubería que queda en la zona afectada por el sismo y selecciona el factor de corrección de la tubería así como el tipo de material y el diámetro del mismo entonces el software se encarga de calcular los valores asociados a cada uno de los datos y procede a calcular la vulnerabilidad que presentan las redes de acueducto y alcantarillado al presenciarse un sismo de esa magnitud.*
Resultado: *El sistema devuelve como salida la vulnerabilidad en un valor de 0.60 teniendo una vulnerabilidad severa ya debe llamar la atención de los especialistas y empezar a buscar alternativas.*
Condiciones: *Se rellenen los datos de los formularios*

Tabla 2.5. Caso de prueba unitaria del camino 5.

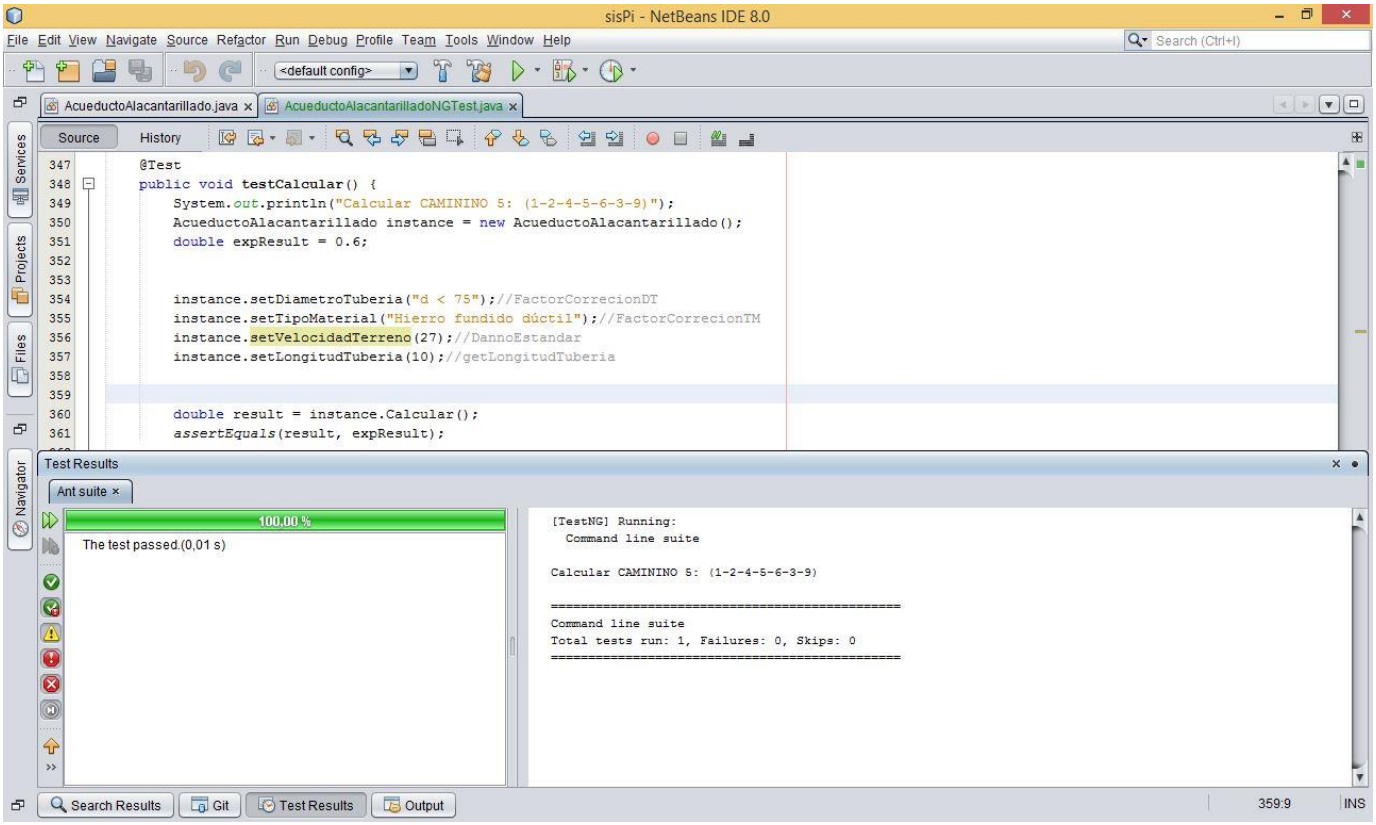

Figura 2.5 Prueba unitaria automatizada del camino 5.

Caso de prueba unitaria
Camino: *1,2,4,5,6,7,3,9*
Caso de prueba: *prueba unitaria camino 6*
Entrada: *El actor introduce la velocidad del terreno, la longitud de tubería que queda en la zona afectada por el sismo y selecciona el factor de corrección de la tubería así como el tipo de material y el diámetro del mismo entonces el software se encarga de calcular los valores asociados a cada uno de los datos y procede a calcular la vulnerabilidad que presentan las redes de acueducto y alcantarillado al presenciarse un sismo de esa magnitud.*
Resultado: *El sistema devuelve como salida la vulnerabilidad en un valor de 1 teniendo una vulnerabilidad total que es casi seguro que termine en destrucción de la infraestructura.*
Condiciones: *Se rellenen los datos de los formularios*

Tabla 2.6. Caso de prueba unitaria del camino 6.

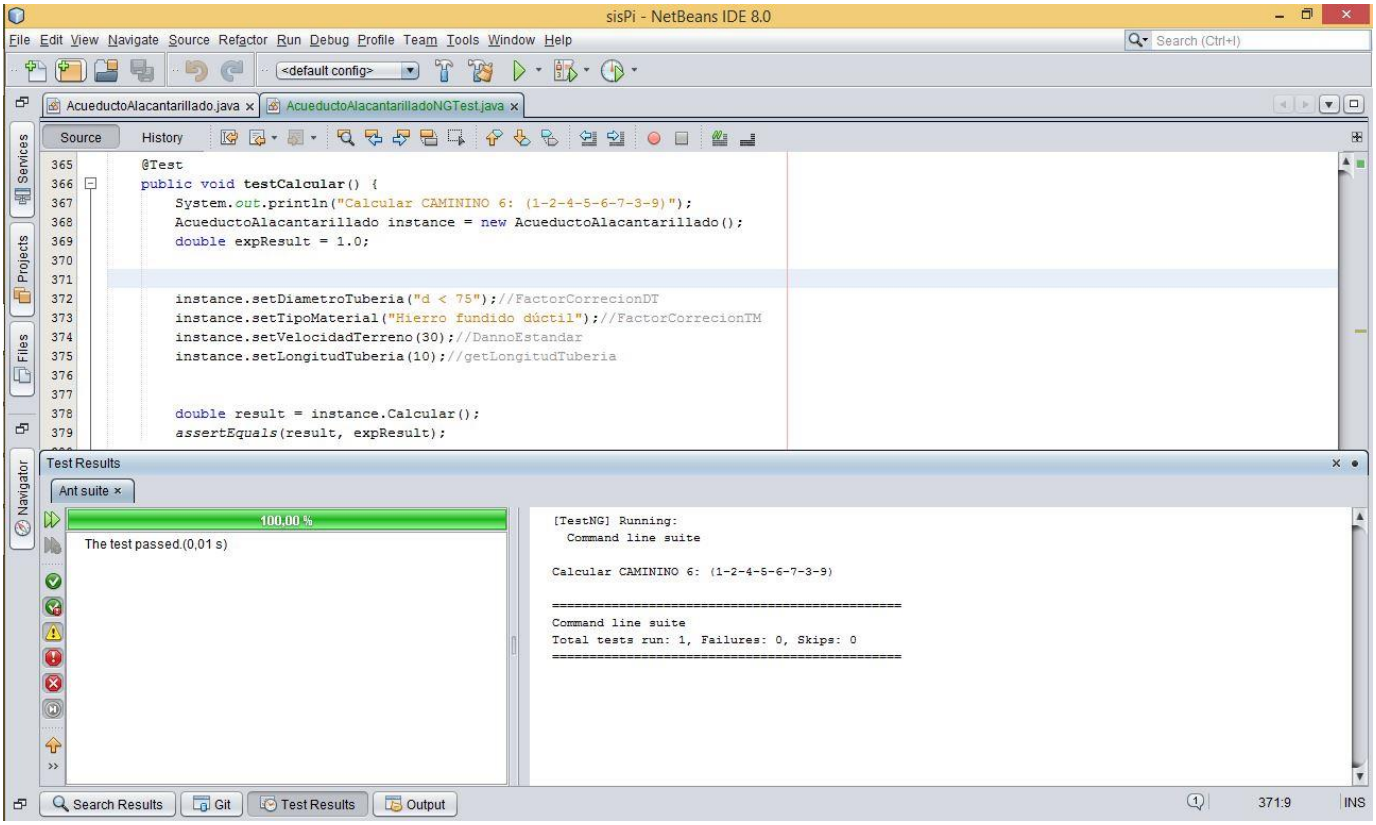

Figura 2.6. Prueba unitaria automatizada del camino 6.

Anexo 3: Casos de pruebas de aceptación

Caso de prueba aceptación	
Código: P1	**Caso de uso**: Gestionar estudios de vulnerabilidad
Descripción: El objetivo es comprobar que se puede introducir los datos de un estudio, que se permita modificar los datos una vez creado, así como eliminar el estudio.	
Condiciones de Ejecución: En caso de modificar o eliminar un estudio que exista previamente.	
Entrada/Pasos de Ejecución: Después de creado un estudio el usuario mediante la opción cargar o eliminar puede gestionar los mismos donde se le permite interactuar con todos los estudios de la base de datos.	
Resultado Esperado: El sistema permite introducir los datos iniciales de un estudio y realizar diferentes acciones sobre el mismo.	
Evaluación de la Prueba: Exitosa	

Tabla 3.1. Caso de prueba de aceptación del caso de uso gestionar estudio.

Caso de prueba aceptación	
Código: P16	**Caso de uso**: Generar reportes
Descripción: Comprobar que se genera un reporte con todos los elementos requeridos.	
Condiciones de Ejecución: Debe haberse creado al menos un estudio.	
Entrada/Pasos de Ejecución: Se selecciona la opción Reporte y se crea un pdf con los datos del estudio que se está trabajando.	
Resultado Esperado: Se muestra un reporte con el logo de la empresa el autor del reporte, la fecha de creación del reporte, una breve explicación de la vulnerabilidad analizada en el reporte y los datos que conllevaron a esa vulnerabilidad.	
Evaluación de la Prueba: Exitosa	

Tabla 3.2. Caso de prueba de aceptación del caso de uso generar reporte.

Anexos 4:Clases del sistema

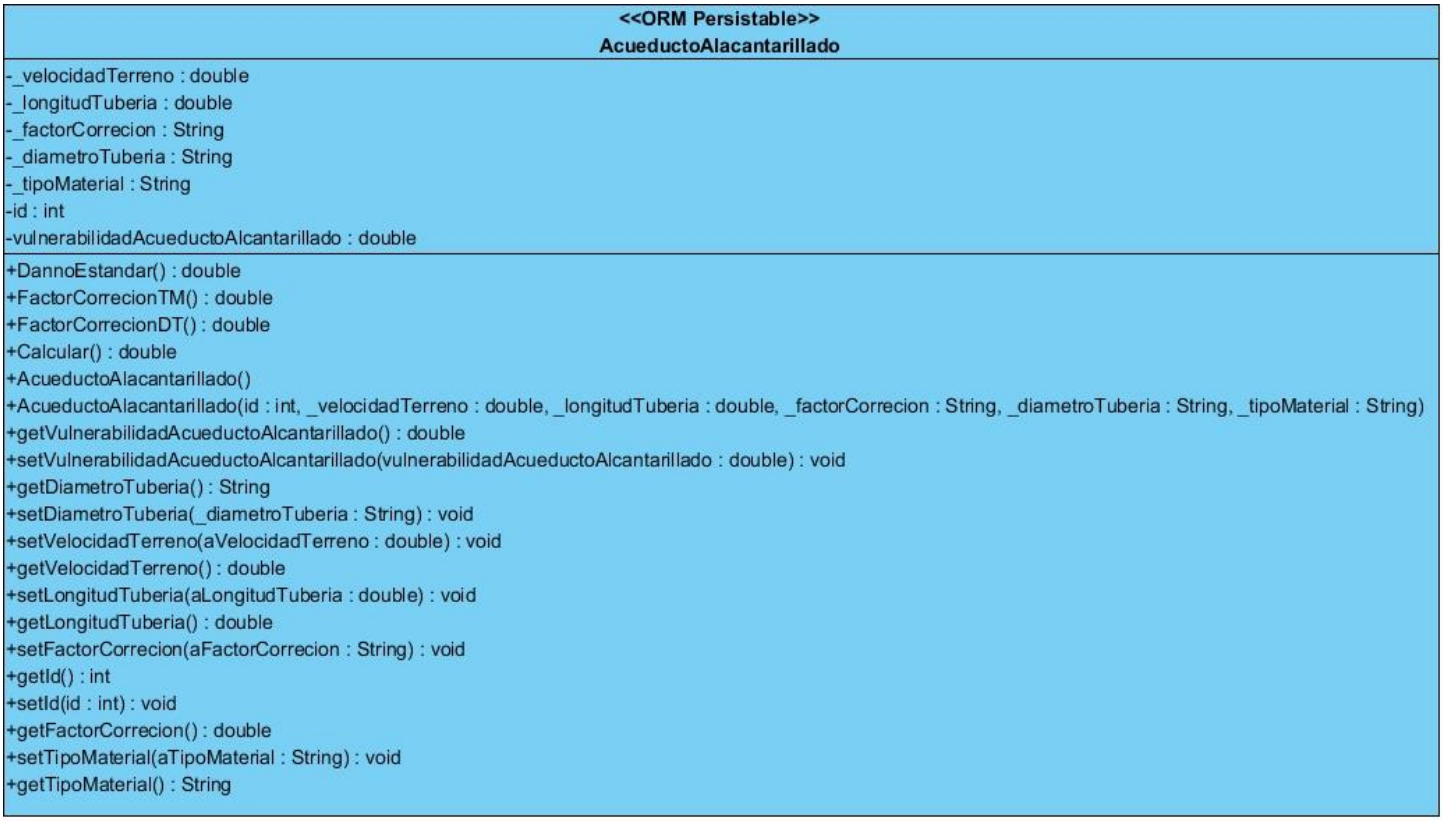

Figura 3.1. Clase Acueducto Alcantarillado SISπ.

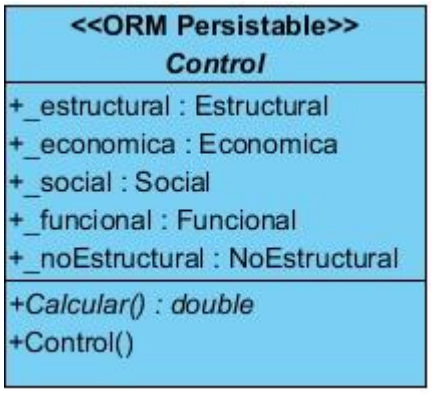

Figura 3.2. Clase Control SISπ.

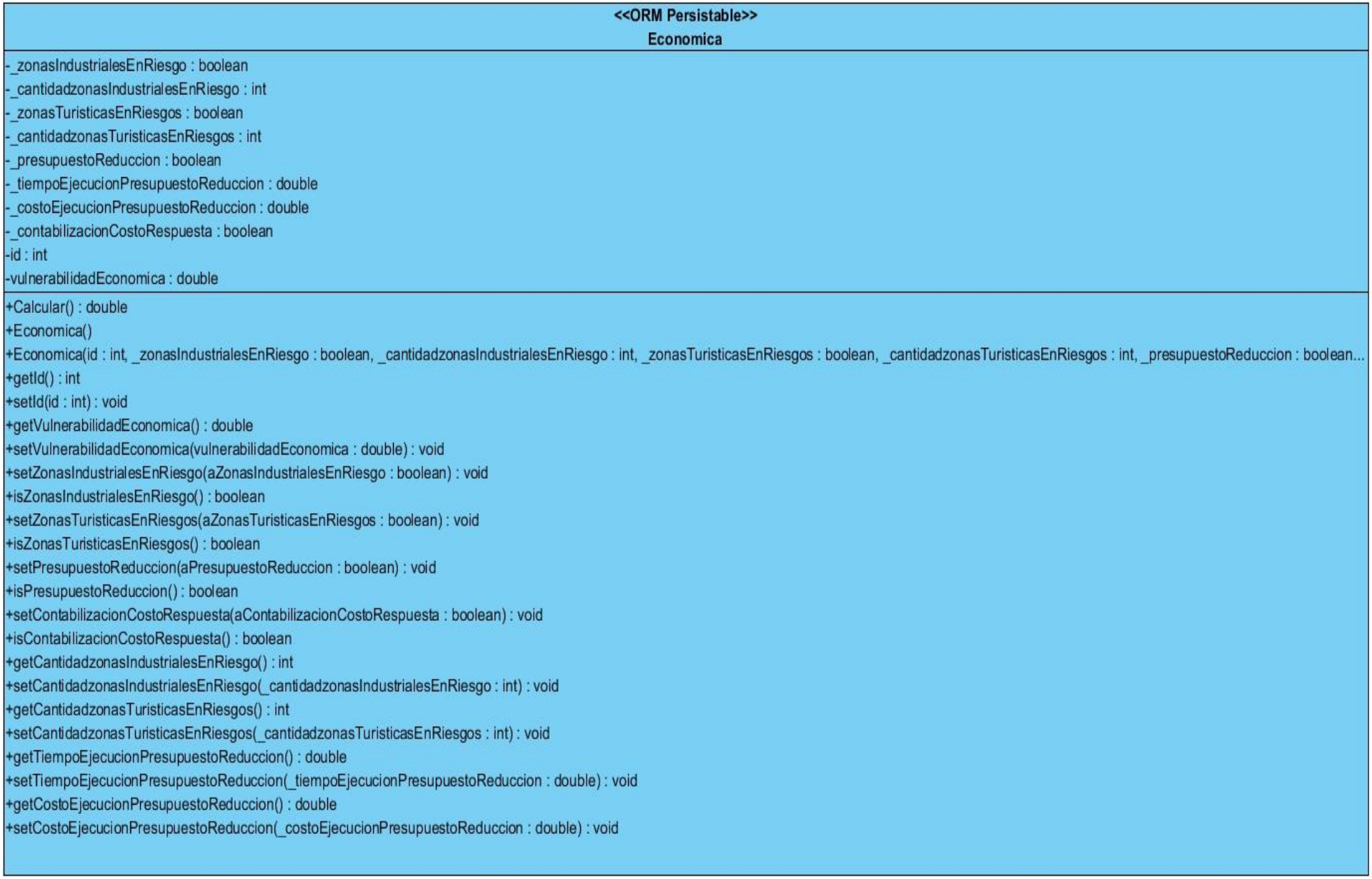

Figura 3.3. Clase Economica SISπ.

Figura 3.4. Clase Edificaciones SISπ.

Figura 3.5. Clase FondoHabitacional SISπ.

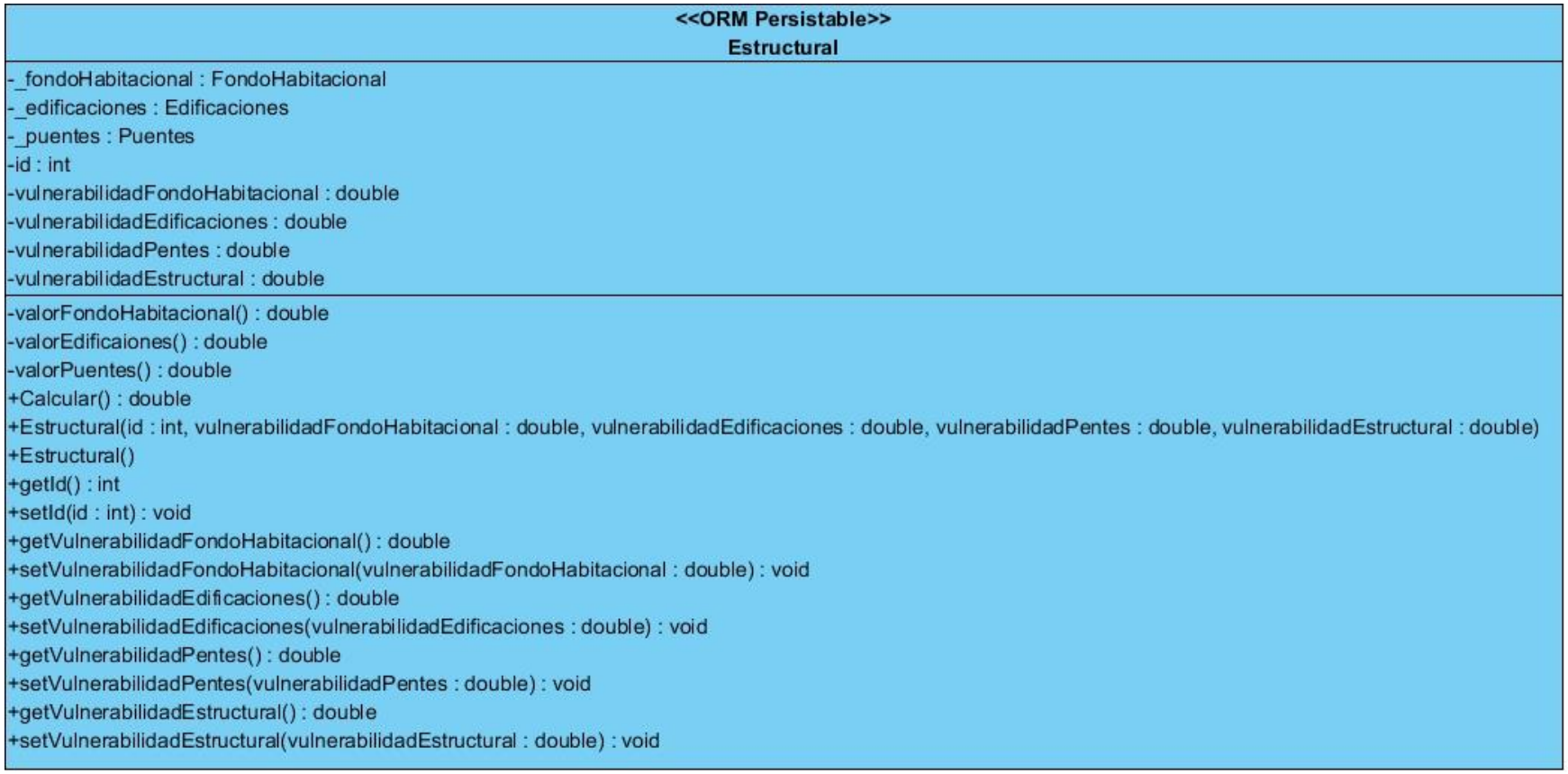

Figura 3.6. Clase Estructural SISπ.

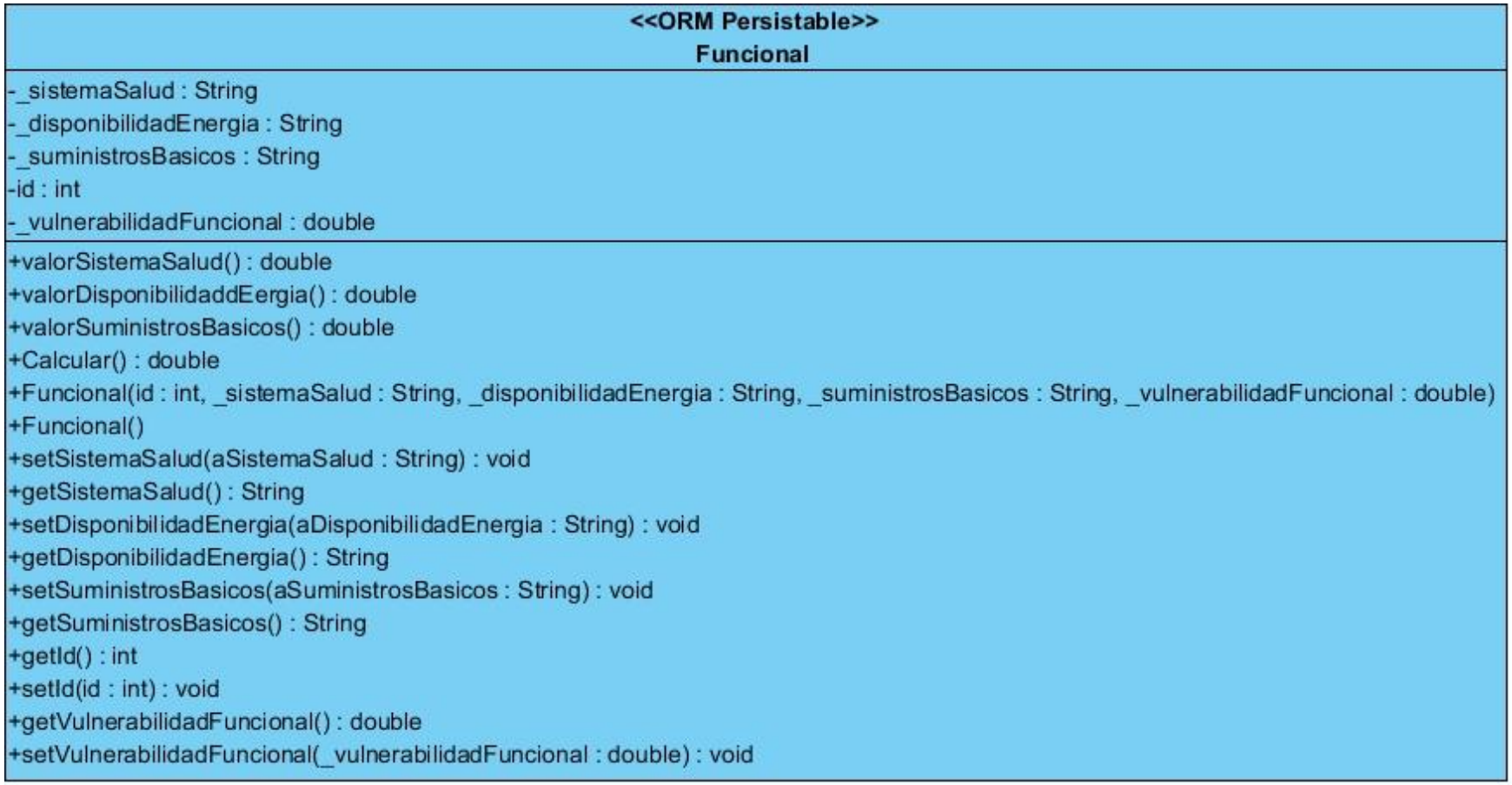

Figura 3.7. Clase Funcional SISπ.

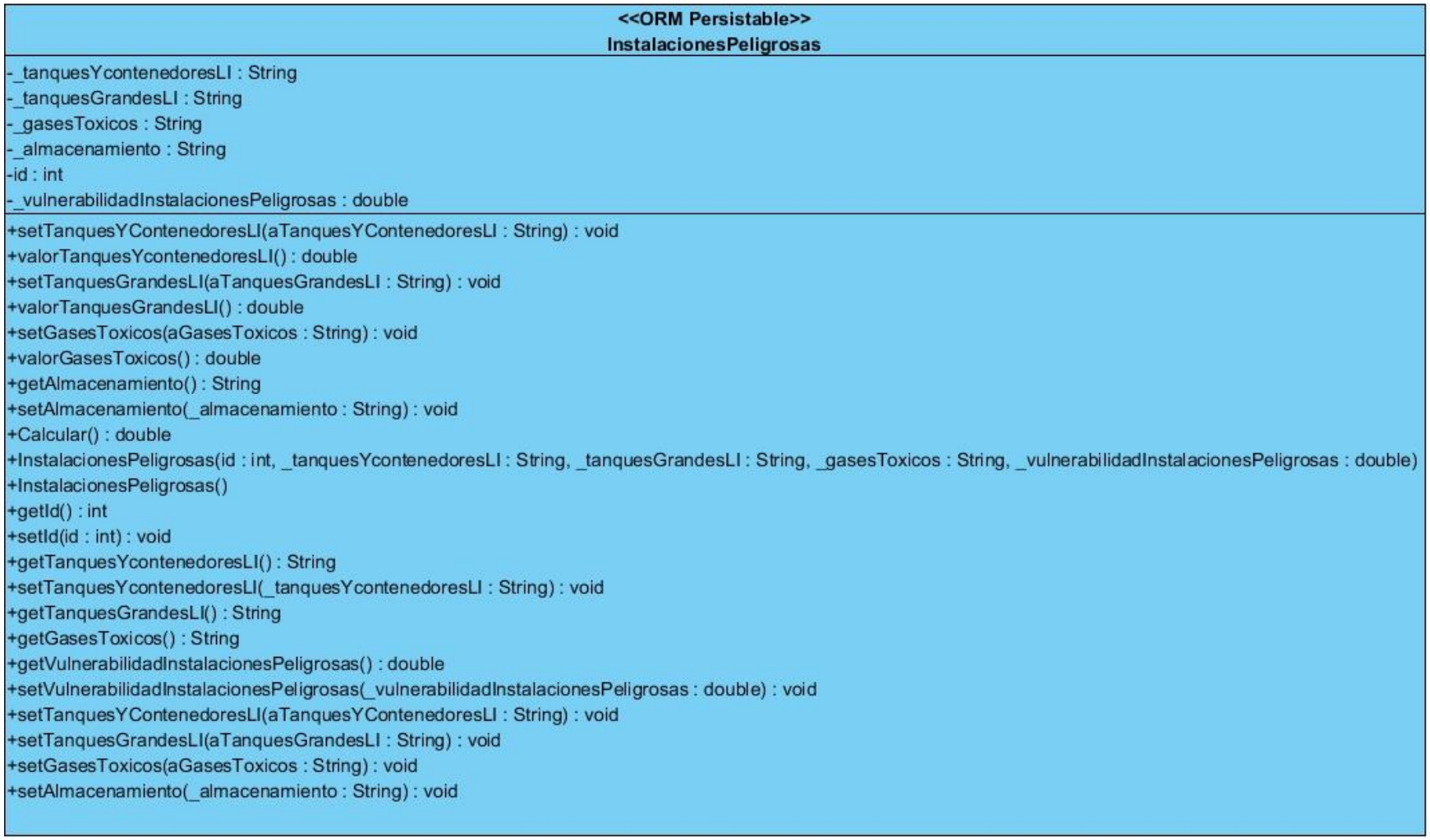

Figura 3.8. Clase InstalacionesPeligrosas SISπ.

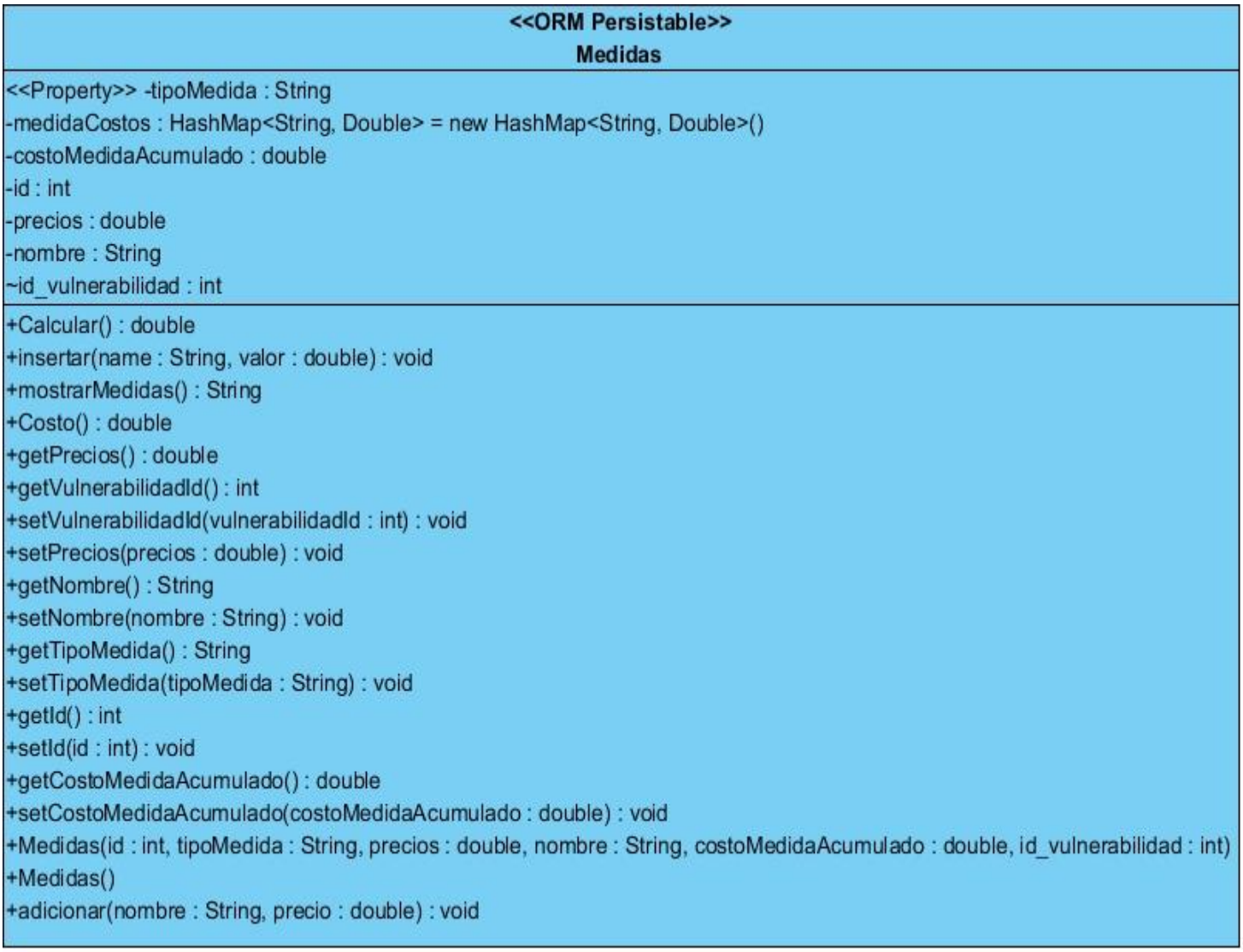

Figura 3.9. Clase Medidas SISπ.

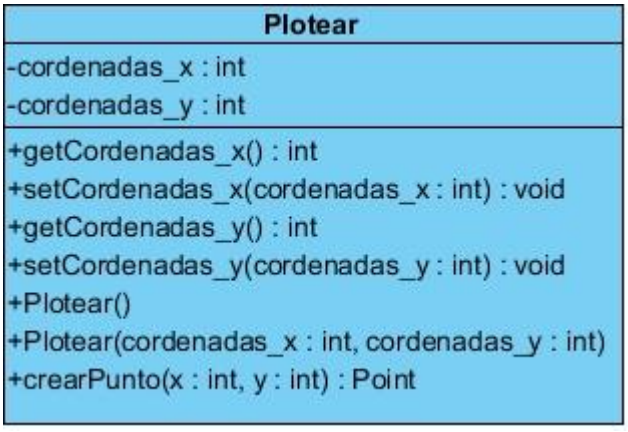

Figura 3.10. Clase Plotear SISπ.

Figura 3.11. Clase Registro SISπ.

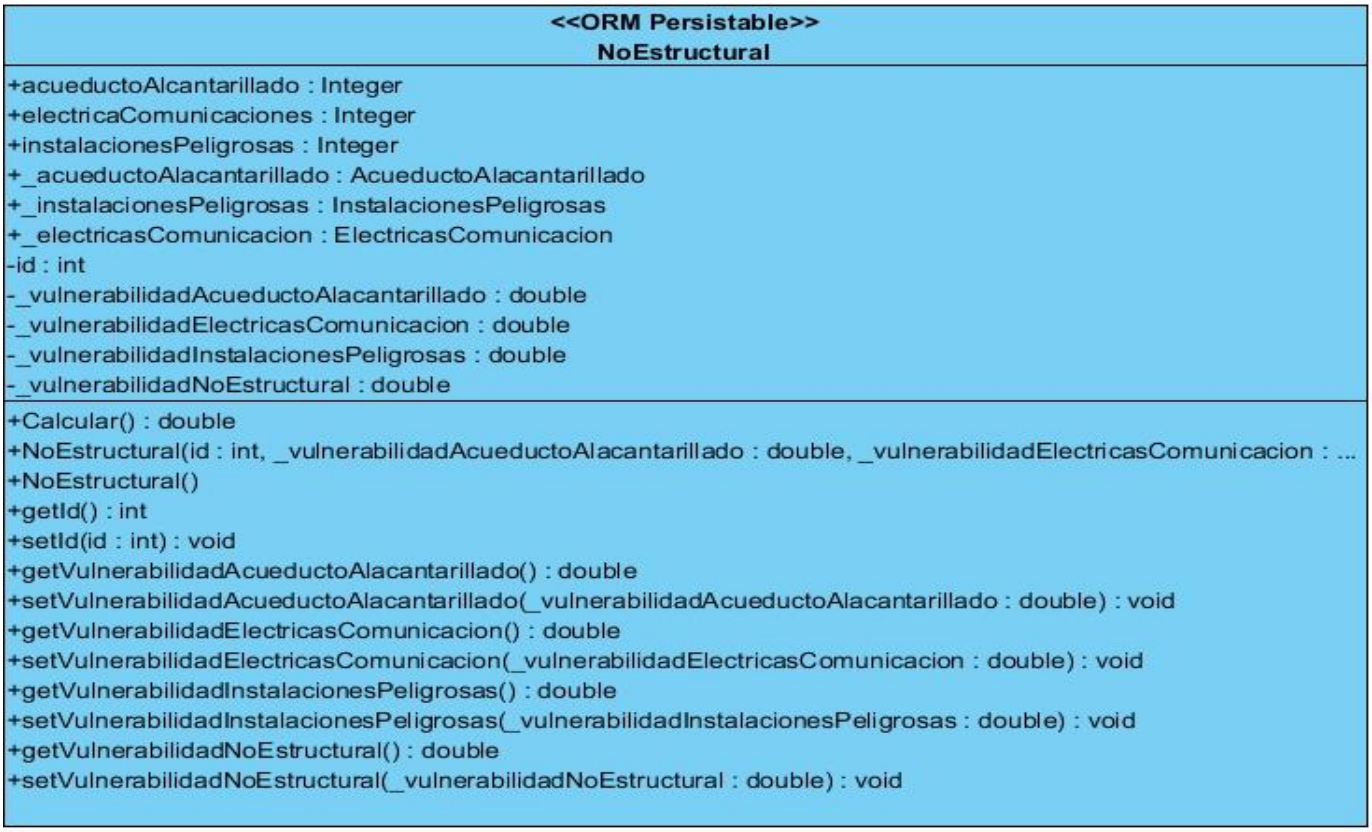

Figura 3.12. Clase NoEstructural SISπ.

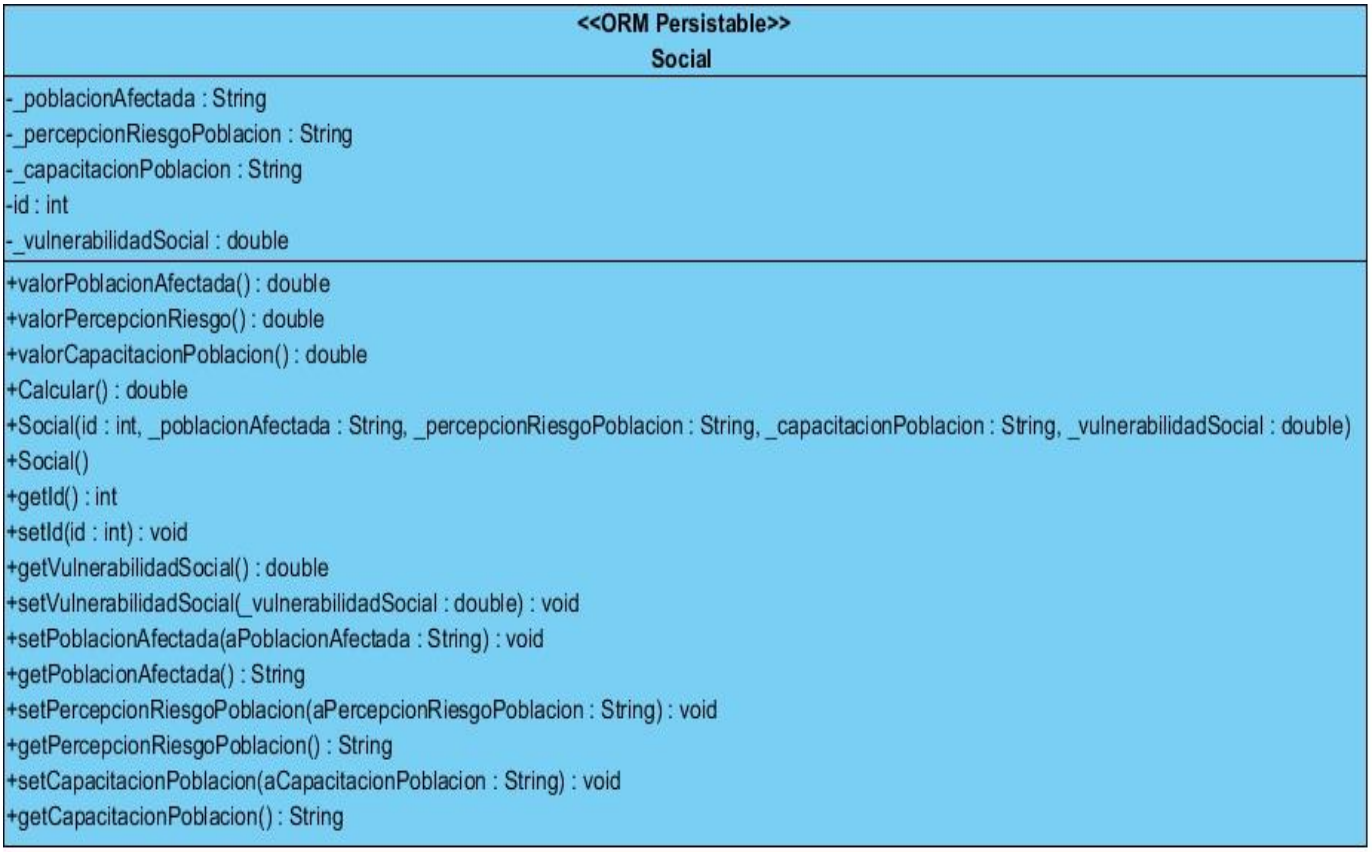

Figura 3.13. Clase Social SISπ.

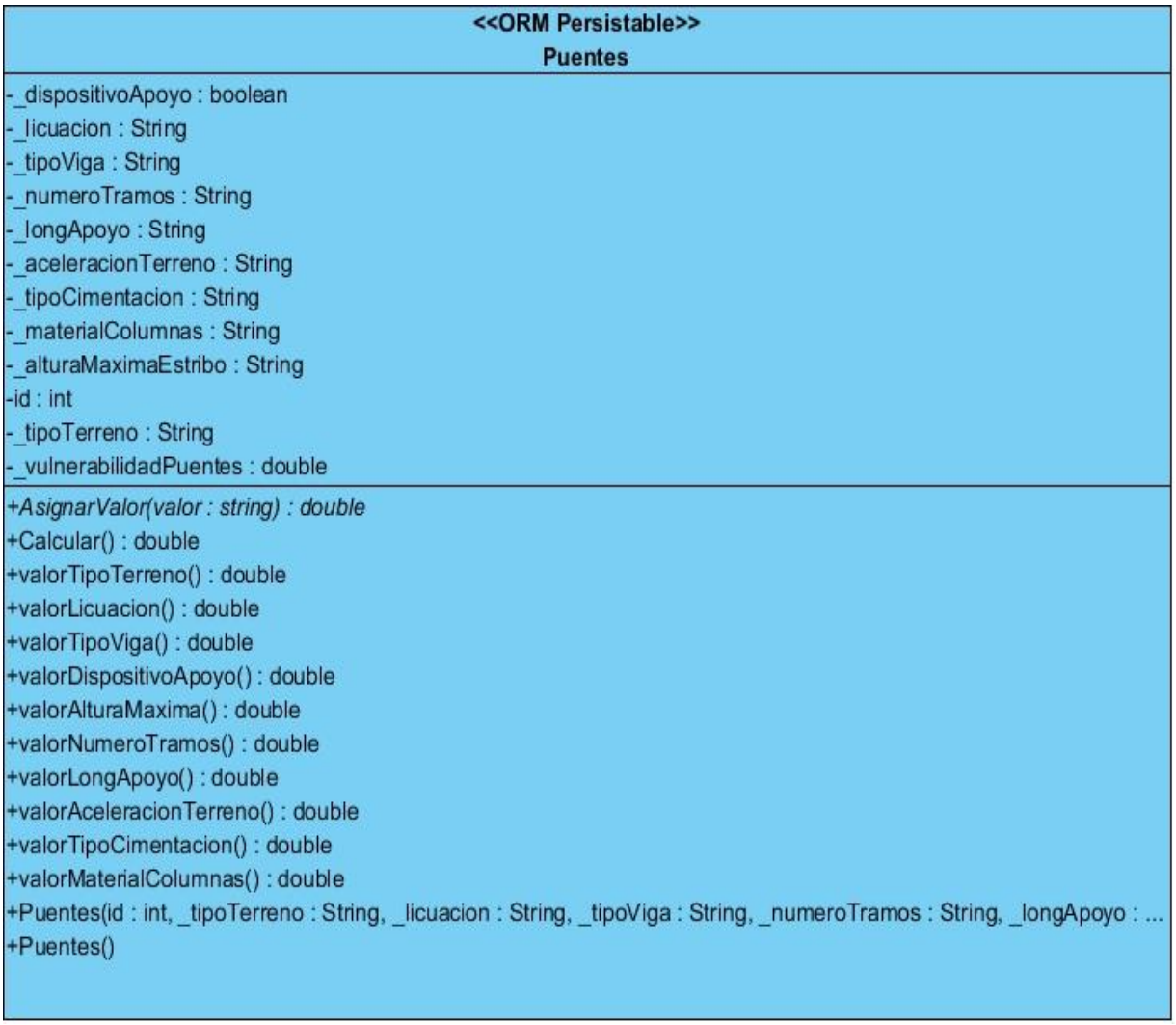

Figura 3.14. Clase Puentes SISπ.

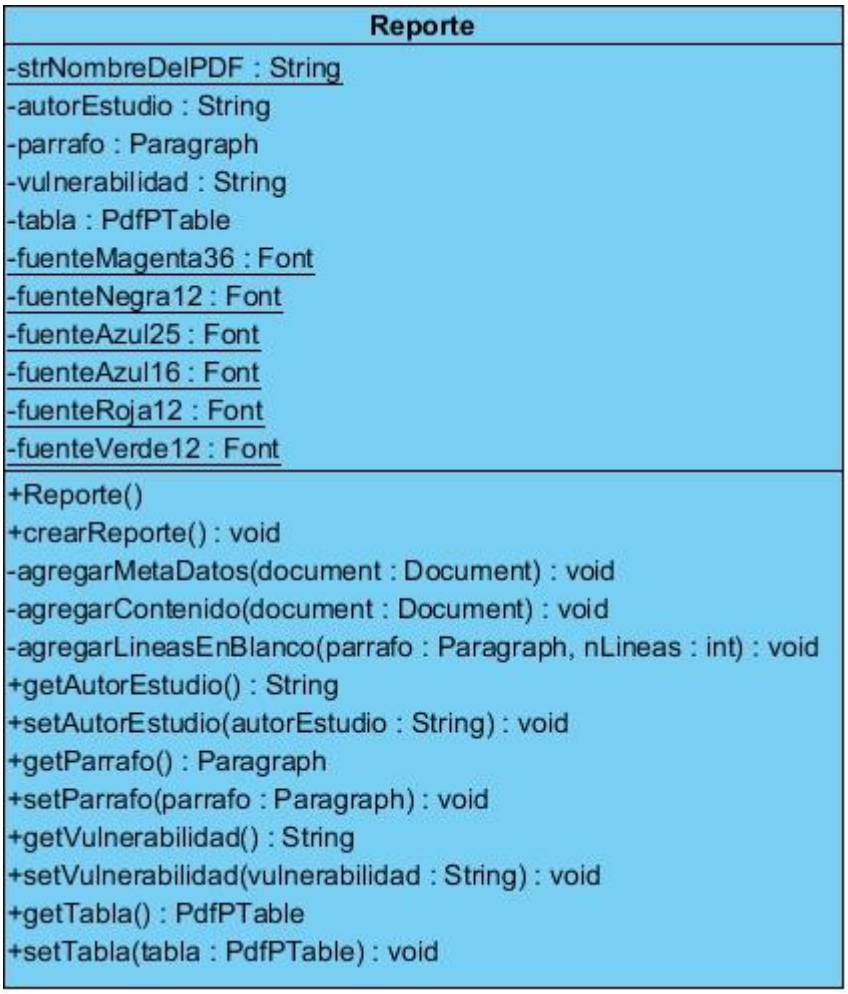

Figura 3.15. Clase Reporte SISπ.

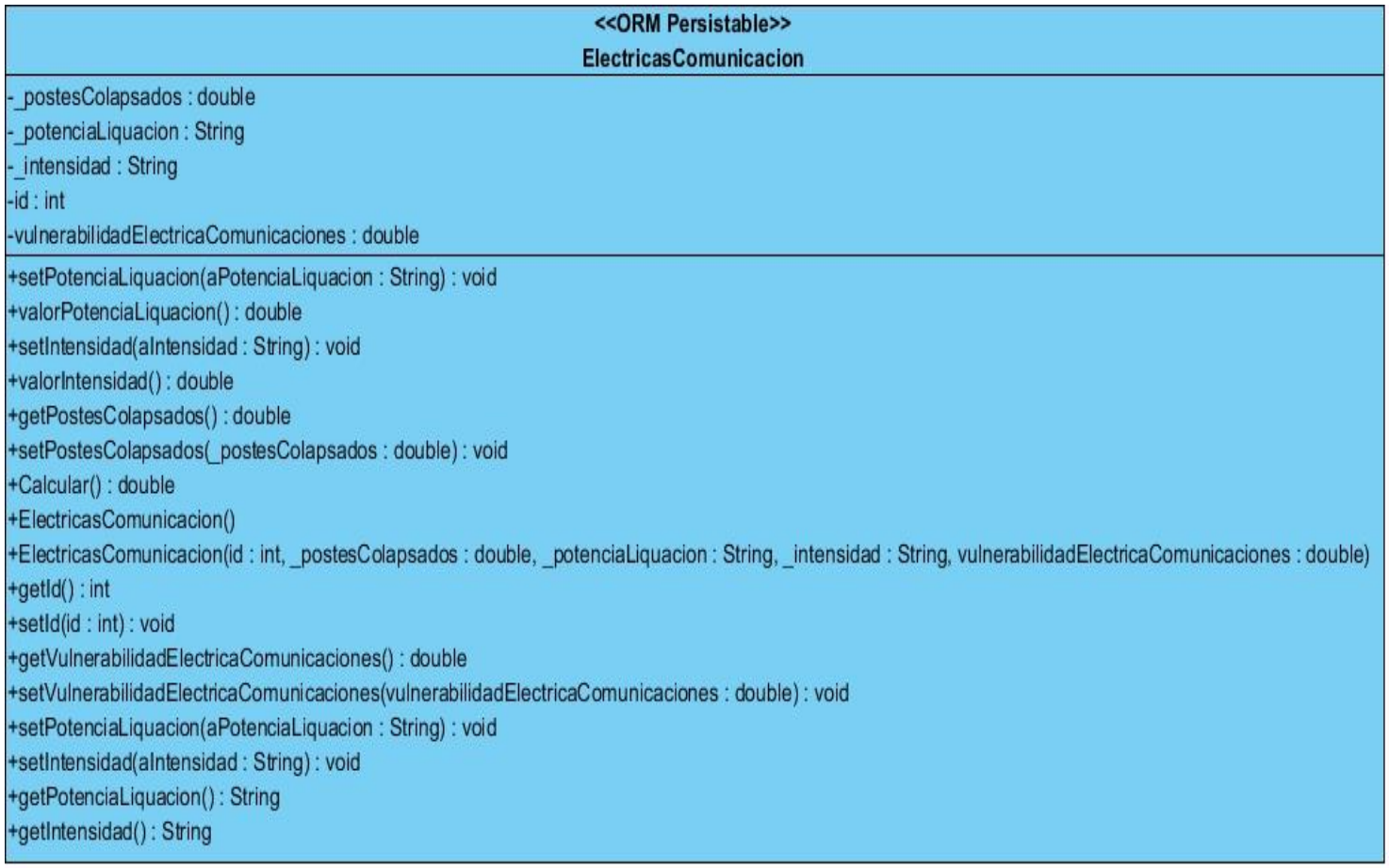

Figura 3.16. Clase ElectricasComunicaciones SISπ