Universidad de las Ciencias Informáticas

Facultad 3

Diseño e Implementación de la capa de acceso a datos y la base de datos de un "Nodo Virtual de Procesos" (NVP)

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autores:

José Antonio Topiz Garcia

Pedro Enrique Piñero Ferrer

Tutor: Ing. Yalice Gámez Batista

Junio 2009

Declaración de Autoría.

Declaro que soy el único autor de este trabajo y autorizo a la Dirección de la Universidad de las Ciencias Informáticas (UCI) a hacer uso del mismo en su beneficio. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____.

José Antonio Topiz Garcia

Pedro Enrique Piñero Ferrer

_____

Ing. Yalice Gámez Batista

Dedicatoria

De Pedro

A:

Mi mamá por ser la persona más linda en mi vida, por tu amor, comprensión y por sobre todas las cosas haberme guiado por buen camino para convertirme en un hombre de bien.

Mi papá por enseñarme el valor de la humildad del hombre, que el conocimiento esta por encima de todo y ser un ejemplo a seguir desde que tengo uso de razón.

Mi hermanita Gaby lo mas lindo de la familia, a mi hermano Rafael porque sabes que nunca haz sido hermanastro sino un hermano para mi.

Toda la familia por su apoyo incondicional en todo momento al cabo de estos 5 años.

Todas las personas que me quieren y que esperan mucho de mí, este trabajo es para ustedes.

Dedicatoria

De Topiz

A:

Mi mamá Carmen, porque es la persona más importante para mí, gracias por la educación que me has dado y por confiar en mí en todo momento, lo hemos logrado. Esta tesis es más tuya que mía.

Mi papá Mario, que me ha inculcado los valores que debe tener toda persona para desenvolverse en la vida, por tener tanta paciencia conmigo. Eres mi ejemplo.

Mi hermanito querido Carlos Alberto, porque sin ti, en todos estos años de mi vida me hubiera sentido solo y gracias a ti tengo fuerzas para seguir adelante.

Mi abuelita Nieves por ser mi segunda madre. Gracias por estar siempre a mi lado y por tus consejos tan sabios.

Mis tías Tata y Rosi por su apoyo incondicional y por todo el sacrificio que han hecho por mí desde que nací.

Mi tío Felipe, que aunque ya no está te recordaré siempre.

Mis abuelos que no están, gracias por los padres que me han dado, son los mejores del mundo.

Agradecimientos

De Pedro

A:

Iriolis Ferrer Blanco, por ser amiga, hermana, padre y por sobre todo madre, gracias mamá.

Evaristo Piñero Nuez, no sabes la alegría y el orgullo que tengo cuando me dicen que me parezco a ti, gracias papá.

Mis hermanos Rafe y Gaby, mis abuelos, mi familia en Camagüey, mi primo Carlos Alberto que estoy seguro será el 1er Diploma de Oro de la familia, mi tío Sergio que dicen que soy zorro igual que el.

Mis dos hermanos habaneros Noel y Dayniel para ustedes no sería una hoja sino un libro de agradecimientos y grandes recuerdos, a mis negros por tantos momentos juntos, marchas, fiestas, colas del doble, Delio el ternerin, Omar, Lenon el súper rapero, Robert el Legendario, Anier, Yoel el Lider, Argenis mi hermano voy ahí, Carlos Enrique, Yoilan el niño, Chung, Julio y Pacho la vieja escuela, Wilson, Arnaldo, Koki.

Mis amigos en Manzanillo, Pedro por tu sinceridad, Félix por sus piedras, Alejandro el Butty mi enemigo - amigo, mi gente de Santa Clara, Arian que lo estoy soportando desde preescolar, Leniecer, Miguel Ángel.

Mi compañero de Tesis por su esmero, preocupación y esfuerzo, este trabajo es tuyo.

Mis amigos de aula Jairol y Gustavo tienen una casa en Santa Clara, Raúl con su falta de memoria que todo te salga bien mi hermano.

Todas mis amigas Yaniris, Yanet, a Lisandra por aguantarme 1año y 6 meses, Emili que en tan poco tiempo te haz ganado todo mi cariño, a mi niña por dejarme entrar en su vida.

Todo el que tuve el placer de conocer en esta gran Universidad aquí les dedico este trabajo.

Agradecimientos

De Topiz

A:

A mis padres por todo el amor que me han dado, los quiero mucho.

Al trío letal (Yoan, Annier y Abel), por todos los grandes momentos que hemos compartido.

A los amigos (Chung, Adrián, Yunier, Martín, Ornolis, el Cory, Hussein, Charly, Yuli) que han hecho posible que yo llegara hasta aquí, por su ayuda en estos 5 años, porque cada uno de ustedes aportó su granito de arena.

A mis primeros amigos en esta escuela (Thais y José Miguel), gracias por esta amistad verdadera que me han brindado.

A mis amigos de Mayarí (Javier, Jhonny y Espinosa) que me han acompañado en todo momento.

A mis amigas del 8302 (Sahily, Malena, Ivon, Ariana, Lisandra e Idaliana), gracias por su amistad.

A Neky por ser tan buena amiga y estar presente en todo momento.

Resumen

El caudal de servicios que brindan las nuevas tecnologías de la información y comunicaciones (TIC), posibilita la creación de modelos, procesos y aplicaciones donde se puede llevar el control informático de toda una serie de operaciones que simulan actividades reales o experimentales. Para ejecutar estas acciones de forma virtual, se propone la elaboración de un proyecto basado en la realización de un Nodo Virtual de Procesos Industriales (NVP); con este trabajo se persiguen objetivos fundamentales tales como implementar modelos de distintos procesos industriales, agrupar estos modelos por tipos posibilitando realizar simulaciones con los mismos o probando aplicaciones en tiempo real. La puesta en marcha de este nodo se basa en la tecnología cliente-servidor, donde aplicaciones clientes podrán conectarse al nodo utilizando un protocolo establecido.

Este trabajo se centra en el diseño y la implementación de la capa de acceso a datos y de una base de datos que permita gestionar la información del NVP, así como la seguridad del mismo, teniendo como características fundamentales, el soporte de altos niveles de concurrencia y requerimientos estrictos de respuesta temporal; pues el sistema debe permitir el almacenamiento en tiempo real. Se realiza un estudio de los principales gestores de bases de datos que existen, las herramientas más utilizadas para diseñar bases de datos y la capa de acceso a datos, para la selección de las herramientas que más se ajustan a las características del trabajo a desarrollar. Luego se describe la solución propuesta y finalmente se hace una valoración del trabajo realizado.

PALABRAS CLAVES

Diseño, base de datos (BD), capa de acceso a datos (CAD), modelo lógico y físico, nodo virtual de procesos (NVP).

Índice

Índice de Tablas

Índice de Figuras

Introducción

La simulación de procesos se ha utilizado ampliamente en aspectos prácticos de muchas disciplinas. Con esta se accede a la capacidad de experimentar independientemente del sistema real. La simulación permite obviar los riesgos inherentes a la experimentación, alcanzar una completa independencia temporal y repetir el experimento un número de veces arbitrario.

En la actualidad existen herramientas informáticas que permiten realizar simulaciones de procesos, pero todas ellas tienen limitaciones, unas debido a la cantidad de recursos que necesitan para su implementación, otras en cuanto al número de procesos simultáneos que se pueden simular. Para solucionar estas restricciones se han creado aplicaciones que usan nodos virtuales en los cuales se simulan los procesos.

Los países más desarrollados en la creación de nodos virtuales son, Estados Unidos, Inglaterra y Alemania. Por ejemplo: en Inglaterra, Universidad de Nottingham Trent, se creó una herramienta para la simulación de sistemas de agua (Hosseinzaman, Bargiela, 1995), en la Universidad de Stuttgart en Alemania se implementó la herramienta llamada “Network Emulation Testbed” (Maier, Herrscher, 2005), que permite simular redes y en el laboratorio Nacional de Lawrence Berkeley en EUA, se creó una herramienta para simulación numérica multifase en pozos de petróleo (Wu, 1999).

En Cuba, se conoce del desarrollo de sistemas de control industrial (SCI), un ejemplo de ello es el SCI “EROS” desarrollado en 1991 por ingenieros de la Unión del Níquel en la provincia de Holguín. En la Universidad de las Ciencias Informáticas (UCI) se está llevando a cabo un proyecto SCADA (Supervisory Control And Data Adquisition) en convenio con Venezuela para el control de los procesos que se desarrollan en las industrias, “El Guardián del Alba”, además del proyecto SIMPRO dedicado al desarrollo de simuladores. En ninguno de estos casos se han usado los nodos virtuales para la simulación de procesos. (Ortiz, Escobar. 2008)

De acuerdo a lo anteriormente expuesto se llega a la conclusión, que en Cuba no se tienen grandes experiencias en el desarrollo de nodos virtuales para la simulación de procesos.

La carencia de una aplicación que realice estas actividades, dificulta el desarrollo y la comprensión de los contenidos docentes en la carrera de Ingeniería Automática, carrera que se estudia en el Instituto Politécnico José Antonio Echeverría (CUJAE), en la Facultad de Ingeniería Eléctrica. En esta, se imparten una serie de contenidos que proveen a los estudiantes de toda la teoría necesaria para ejercer como profesionales de la rama, sin embargo no siempre estos conocimientos son fundamentados desde el punto de vista práctico con herramientas que fortalezcan la comprensión de los mismos.

En la UCI, durante el curso 2007-2008 se realizó el "Análisis y Diseño" (Ortiz, Escobar. 2008) y la "Arquitectura" (Gonce, 2008) de una herramienta para la simulación de procesos usando nodos virtuales, Nodo Virtual de Procesos (NVP). En el presente curso se está llevando a cabo su implementación. Tomando todo lo anterior se concluye que la **situación problémica** consiste en que se está llevando a cabo la implementación de una herramienta que le permita a los estudiantes de Ingeniería Automática interactuar con diferentes procesos configurados por ellos o por sus profesores con la cuál puedan probar posibles estrategias de control, pero no se cuenta con el diseño de una base de datos y de la capa de acceso que permita su correcto funcionamiento.

Según lo descrito anteriormente se puede llegar al siguiente **problema**:

¿Cómo desarrollar la capa de acceso a datos y la base de datos de una herramienta para la simulación de procesos que permita una mayor calidad en la gestión y almacenamiento de la información?

Por consiguiente el **objeto de la investigación** sería, el Proceso de Desarrollo de Software, centrando la atención en el almacenamiento de la información en el Proceso de Desarrollo de Software, identificado como campo de acción de la investigación.

Lo que conlleva a la siguiente **hipótesis**:

Si se logra realizar el diseño e implementación de la capa de acceso a datos y la base de datos de un NVP, entonces se podrá garantizar una mayor calidad en la gestión y almacenamiento de la información.

A partir de ésta se formulan los **objetivos** de este trabajo, los cuales se precisan a continuación:

Objetivo General:

Diseñar e implementar la capa de acceso a datos y la base de datos de una herramienta para la simulación de procesos: “Nodo Virtual de Procesos” (NVP).

Objetivos específicos:

1. Diseñar la base de datos para el NVP.
2. Implementar la base de datos para el NVP.
3. Diseñar el acceso a datos para el NVP.
4. Implementar la capa de acceso a datos del NVP.

Tareas de Investigación:

- Estudio del estado del arte.
- Estudio del Sistema Gestor de Base de Datos (SGBD) que se va a utilizar.
- Estudio del lenguaje y la plataforma que se utilizarán.
- Identificación y descripción de los elementos de diseño.
- Diseño e implementación de la capa de acceso a datos.
- Diseño e implementación de la base de datos.
- Validación de los elementos de diseño.
- Validación de la implementación.

Con el propósito de desarrollar las tareas planteadas, se utilizaron los **métodos** de investigación siguientes:

Métodos teóricos:

Histórico lógico: Se realiza un análisis de cómo han evolucionado las diferentes herramientas de simulación con el uso de nodos virtuales. Obteniendo una tendencia de cómo se deben comportar en la actualidad.

Sistémico: Se tienen en cuenta todas y cada una de las relaciones que se establecen entre los módulos dentro de la aplicación y las contradicciones que generan los mismos.

Hipotético-deductivo: A partir de la interpretación de la realidad se establecen posibles situaciones o resultados para llegar a conclusiones.

Métodos empíricos:

Consulta a especialistas: Se empleó para comprobar la necesidad y la funcionalidad práctica de la aplicación, Nodo Virtual de Procesos que se propone en la presente investigación.

La observación: Mediante guías de observación se le dará seguimiento al desarrollo de la aplicación.

Estructura de la tesis:

El trabajo está estructurado por tres capítulos, los cuales están conformados como se muestra a continuación:

En el capítulo 1 se realizará una breve reseña de los nodos virtuales. Se llevará a cabo un estudio sobre las herramientas a utilizar, para establecer una comparación que permita escoger las más adecuadas.

En el capítulo 2 se realizará la descripción y análisis de la solución propuesta. En el mismo se exponen los aspectos del diseño de la BD: los diagramas relacionales y la descripción de las entidades

correspondientes. Además, se explicarán los aspectos relacionados a la implementación de la capa de acceso a datos a través de diagramas y descripciones de las clases necesarias para el acceso.

En el capítulo 3 se realizará la validación de los resultados, mediante la aplicación de métricas y pruebas.

Capítulo 1: Fundamentación teórica

Introducción

Como propuesta de solución a la problemática planteada en la introducción de este trabajo se optó por un NVP. En este capítulo se abordarán aspectos y conceptos relacionados con el mismo. Se hará un análisis del comportamiento de las funcionalidades a realizar por el NVP en otros sistemas similares. Se explicarán las herramientas escogidas para dar solución a la problemática propuesta.

1.1. Nodo Virtual de Simulación de Procesos

Existen diversas definiciones y tipos de simulación. Para propósitos prácticos, se define simulación como el proceso de desarrollar un modelo de un problema y estimar medidas de su comportamiento llevando a cabo experimentos muestrales sobre el modelo (Jiménez, 2008).

Definiciones:

"Simulación es una técnica numérica para realizar experimentos en una computadora digital. Estos experimentos involucran ciertos tipos de modelos matemáticos y lógicos que describen el comportamiento de sistemas de negocios, económicos, sociales, biológicos, físicos o químicos a través de largos períodos de tiempo." (Castelán, 2006)

"Simulación es el proceso de diseñar y desarrollar un modelo computarizado de un sistema o proceso y conducir experimentos con este modelo con el propósito de entender el comportamiento del sistema o evaluar varias estrategias con las cuales se puede operar el sistema." (Shannon, 1992)

En la actualidad existen muchas herramientas informáticas (Maier, 2003), (Hosseinzaman, 1995), (Wu, 1999), (Molino, 2004), (Miyachi, 2006), que permiten realizar simulaciones de procesos industriales (en tiempo real o no), pero todas éstas tienen limitaciones, en cuanto al número de recursos que necesitan para su implementación o en cuanto al número de procesos simultáneos que se pueden simular. Incluso, en su mayoría, o simulan los procesos o permiten probar aplicaciones reales, nunca las dos prestaciones.

Para dar solución a esta problemática se optó por un NVP, que admite ejecutar diferentes instancias del software en un único nodo (nodo físico) y cada instancia del software trabaja en un entorno de ejecución independiente (nodo virtual).

Esta filosofía de trabajo proporciona la simulación simultánea de diferentes procesos concurrentes sin que interfieran uno con otros. La virtualización de nodos provee una vía de regular el acceso a recursos de hardware exclusivos de un determinado número de consumidores. En este caso los consumidores son los entornos de ejecución para cada proceso, los cuales están sujetos a las propiedades de la simulación. (Maier, 2003)

Del cual se derivan los siguientes requerimientos para la virtualización del nodo: (Gámez, 2008)

- El parámetro más importante es minimizar los gastos de virtualización para preservar los recursos para el proceso en ejecución.
- Cada entorno de ejecución introducido por la virtualización del nodo debe ser tan transparente como sea posible para los restantes. Esto es importante para que la medición de la implementación no sufra modificaciones en comparación con la real.

Hoy en día se conoce como NVP, a aquel software que permita implementar modelos de distintos procesos ya sea para su simulación o la prueba de aplicaciones en tiempo real, por lo que será necesario que varios procesos estén activos simultáneamente. Esta filosofía de trabajo permite la simulación simultánea de diferentes procesos sin que interfieran uno con otros. (Ortiz, Escobar. 2008).

Se establece como nodo a aquella estructura a la cual se interconectan varios elementos. No hay que pensar en un nodo como un elemento constituido solamente por una parte física, sino más bien considerarlo como una unidad funcional en donde tiene que haber tanto hardware como software. Por otra parte, al ser el punto de conexión de dos o más elementos, el nodo posee la capacidad de recibir información, procesarla y enrutarla a otro u otros nodos. De esta manera, un nodo puede ser el punto de conexión para transmitir los datos, el punto desde el cual se distribuye los datos hacia otros nodos y el punto final al que se transmiten los datos.

Para el establecimiento de pruebas se necesita de una fracción de recursos de un nodo de prueba y un número de aplicaciones dirigidas a dispositivos de pocos recursos de forma tal que se puedan ejecutar varios procesos en este nodo de prueba más conocido como nodo físico o pnodo y que cada proceso provea un entorno de ejecución del mismo de manera separada, a esto se le conoce como nodo virtual o vnodo. (Gámez, 2008)

1.2. El NVP en otros sistemas similares.

En la actualidad existen muchas aplicaciones que utilizan los nodos virtuales por las grandes ventajas que brindan en el aprovechamiento máximo de los recursos de las computadoras. Para la simulación de redes, en la Universidad de Stuttgart en Alemania, instituto especializado en sistemas distribuidos y paralelos, se creó la aplicación "Network Emulation Testbed" (Maier, 2003). Esta aplicación va dirigida a simular un entorno de redes configurable que permita reproducir un escenario real de cientos de nodos en comunicación. De esta forma posibilita medir de manera comparativa el comportamiento de una aplicación en diferentes entornos de redes o de varias aplicaciones en un mismo entorno de red. Esta aplicación es utilizada tanto en redes tradicionales como en un entorno de redes ad hoc inalámbrica. Permite a partir de una red de 64 computadoras traducirlo a un escenario de más de 1920 nodos. La principal limitante de esta aplicación está en que es diseñada para simular redes por lo que no es posible utilizarla para la simulación de procesos que tienen una dinámica más compleja y variada.

En la Universidad de Nottingham Trent, se creó un software para la simulación de sistemas de agua (Hosseinzaman, 1995). Surge por la necesidad que tenían en la industria del agua de adquirir y almacenar datos de estaciones remotas para la inspección ingenieril. A medida que iba creciendo el sistema, fue incrementándose la acumulación de grandes volúmenes de datos que debían ser procesados por los ingenieros. Para ayudar a reducir la carga de trabajo, aumentar la eficiencia y la efectividad del control del sistema se introdujo el software de simulación. Este software tiene como tareas analizar los parámetros del sistema y arribar a decisiones que pueden ser aceptadas o modificadas por el ingeniero responsable de realizar estas operaciones. La integración del sistema de supervisión con el software de simulación, para lograr un sistema capaz de tomar decisiones en tiempo real, conllevó a un incremento de los requerimientos computacionales para los algoritmos de simulación con un respectivo aumento de tamaño de la red física. Una solución clásica fue dividir la red en subredes para resolver las subredes de manera

aislada y de esta manera coordinar las soluciones de los subsistemas para encontrar la solución del sistema completo. Esta aplicación no está concebida para simular procesos independientes, sino que simula subprocesos de un sistema específico, para luego integrarlo en el proceso general como un todo.

En el laboratorio Nacional de Lawrence Berkeley en EUA, se enfrentaban al problema de tratar las condiciones límites en los pozos de petróleo en el momento de formular y codificar un simulador numérico multifase de la reserva. Esto se debe a la complejidad de las ecuaciones diferenciales que gobiernan el flujo de la superficie. Son una mezcla de tipo hiperbólica-parabólica y provocan problemas de convergencia computacional (Wu, 1999). El método convencional de tratamiento de pozos geotermales o de reservas de petróleo no es lo suficientemente riguroso y puede dar lugar a soluciones físicamente incorrectas para las diferentes capas del pozo. Por esta razón se utilizó el método de nodos virtuales donde a cada capa se le asignó un nodo virtual cuyo tratamiento está dado en dependencia de las características de la capa a la que represente para los cálculos del flujo. La solución en el pozo se obtendrá de resolver las ecuaciones del balance de masa para el nodo del pozo. Así se provee de un procedimiento numérico eficiente y consistente de forma física para el manejo de los problemas del flujo en los pozos. De la misma manera que el anterior este simulador fue diseñado para el estudio de las diferentes capas de un pozo petrolífero para luego integrarlo en el modelo del pozo.

En la Universidad de Stamford se creó un algoritmo de nodos virtuales para el trabajo con imágenes en tres dimensiones (Molino, 2004). Un elemento es fragmentado para crear varias réplicas del elemento y se le asigna una porción real del material a cada réplica. Como resultado se obtienen elementos que contienen material real y regiones vacías, el resto está contenido en otra u otras copias. El algoritmo de nodo virtual determina automáticamente el número de réplicas así como la asignación de material para cada una. Provee además los grados de libertad requeridos para simular el material parcial o completamente fragmentado en una imagen consistente con la geometría. Aprovecha las posibilidades de la simulación de una geometría compleja con una simple mezcla. Este simulador tampoco satisface las necesidades planteadas.

En Japón se creó una herramienta “StarBED” para dar solución al vacío que existe entre internet y los entornos para experimentación, atendiendo a los aspectos de escala, complejidad y realidad (Miyachi, 2006). Es una aplicación de pruebas basada en lotes de nodos que tiene como objetivos construir

entornos de experimentaciones reales, complejos y de gran escala. Está diseñado para 512 computadoras en las cuales se ejecutan diez computadoras virtuales. Es capaz de soportar una topología de experimentación por encima de los 512 pero resulta difícil manejar y controlar todos esos nodos. Además se implementó “SpringOS” para soportar las simulaciones de acuerdo con la configuración de los usuarios. Ya ha sido probada con efectividad en diferentes experimentos como son:

- Test de rendimiento de switchs L2 con tráfico multiemitido.
- Observación del comportamiento del TCP.
- Comparación entre el comportamiento entre routers de hardware manufacturado por vendedores populares y routers de software en código abierto ejecutándose en una PC.
- Simulación de redes inalámbricas en redes cableadas.
- Simulación de redes de teléfonos celulares G4.

Estas herramientas fueron creadas para evitar las influencias de los servicios críticos de internet en las aplicaciones de simulación. Se utilizan para la simulación de diferentes tipologías de redes y no están concebidas para la simulación de procesos. Por esta razón no satisface los requerimientos de la aplicación planteada. De manera general se puede concluir que en la actualidad los nodos virtuales son potencialmente usados para trabajar en aplicaciones diseñadas para la simulación de redes, para la simulación de procesos que por su complejidad no pueden ser realizadas por las herramientas tradicionales y para el tratamiento de imágenes. Por estas razones es necesario que el Nodo Virtual de Procesos incorpore muchas características que no poseen dichas aplicaciones. (Gámez, 2008)

1.3. Aspectos Generales de las Bases de Datos

Una base de datos es una colección de información organizada de forma que un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite. Una base de datos es un sistema de archivos electrónico. Las bases de datos tradicionales se organizan por campos, registros y archivos. Un campo es una pieza única de información; un registro es un sistema completo de campos; y un archivo

es una colección de registros. Por ejemplo, una guía de teléfono es análoga a un archivo. Contiene una lista de registros, cada uno de los cuales consiste en tres campos: nombre, dirección, y número de teléfono. Los Sistema de Gestión de Base de Datos (SGBD) son un tipo de software muy específico dedicado a servir de interfaz entre la base de datos y las aplicaciones que la utilizan (el usuario). Se compone de un lenguaje de definición de datos (LDD), de un lenguaje de manipulación de datos (LMD) y de un lenguaje de consulta (LC). Tiene como propósito general manejar de manera clara, sencilla y ordenada un conjunto de información. Se destacan entre sus funciones la definición de los datos en los distintos niveles de abstracción, la manipulación de los datos, el mantenimiento de la integridad de la BD y el control de la seguridad de dichos datos. (Masadelante, 2006)

Clasificación de las Bases de Datos

De acuerdo a varios criterios, las bases de datos pueden ser clasificadas de diferentes formas:

Según la forma en que cambia la información almacenada:

Bases de datos estáticas: Son aquellas bases de datos en las que no se puede modificar la información que está guardada, son utilizadas preferentemente para guardar datos históricos.

Base de datos dinámicas: Son aquellas donde la información puede ser cambiada con el tiempo, permitiendo operaciones como actualización y edición de datos, además de las operaciones de consultas. (Alburquerque, 2007)

1.4. Fases de diseño de datos

1.4.1. Diseño Conceptual

Un esquema conceptual es una descripción de alto nivel de la estructura de la base de datos, independientemente del Sistema Gestor de Base de Datos (SGBD) que se vaya a utilizar para manipularla. Un modelo conceptual es un lenguaje que se utiliza para describir esquemas conceptuales. El objetivo del diseño conceptual es describir el contenido de información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar esta información. Los modelos

conceptuales deben ser buenas herramientas para representar la realidad, por lo que deben poseer las siguientes cualidades:

- **Expresividad:** Suficientes conceptos para expresar perfectamente la realidad.
- **Simplicidad:** Deben ser simples para que los esquemas sean fáciles de entender.
- **Minimalidad:** Cada concepto debe tener un significado distinto.
- **Formalidad:** Todos los conceptos deben tener una interpretación única, precisa y bien definida. (Márquez, 2001)

1.4.2. Diseño Lógico

El diseño lógico parte del esquema conceptual y da como resultado un esquema lógico. Un esquema lógico es una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD. Un modelo lógico es un lenguaje usado para especificar esquemas lógicos (Modelo Relacional, Modelo de Red, etc.) El diseño lógico depende del tipo de SGBD que se vaya a utilizar, no depende del producto concreto. (Márquez, 2001)

El esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, pues permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la base de datos. (Fábregas, 2007)

1.4.3. Diseño Físico

El objetivo del diseño físico es conseguir una instrumentación lo más eficiente posible del esquema lógico, para esto se analizan aspectos como las características del sistema operativo, el SGBD, la herramienta para realizar el diseño, aspectos relacionados con el rendimiento y los requisitos de procesos así como las características del hardware, en fin, cualquier factor cercano con la computadora, logrando optimizar el consumo de recursos, minimizar el espacio de almacenamiento, proporcionar la seguridad máxima, disminuir los tiempos de respuesta y evitar las reorganizaciones. (Fábregas, 2007)

1.5. Modelo de datos

El modelado de datos puede ser descrito como el proceso de información de una aplicación de uso intensivo. Construir una representación de la aplicación que capture las propiedades estáticas y dinámicas requeridas para dar soporte a los procesos deseados. Modelos de datos clásicos:

Modelo Jerárquico

Los modelos de datos jerárquicos almacenan la información de forma jerárquica, los datos se organizan en forma similar a un árbol donde un padre tiene muchos hijos. Este modelo es especialmente útil en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos, permitiendo crear estructuras estables y de gran rendimiento. El modelo de datos jerárquico presenta importantes inconvenientes, que provienen principalmente de su rigidez, por la falta de capacidad de las organizaciones jerárquicas para representar sin redundancias ciertas estructuras muy difundidas en la actualidad. La poca flexibilidad de este modelo puede obligar a la introducción de redundancias cuando es preciso instrumentar, mediante el modelo jerárquico, situaciones del mundo real que no responden a una jerarquía. (Moraga, 2001)

Modelo de Red

El modelo de datos en red general representa las entidades en forma de nodo de un grafo, y las asociaciones o interrelaciones entre éstas mediante los arcos que unen dichos nodos. El modelo de red evita esta redundancia en la información, a través de la incorporación de un tipo de registro denominado el conector, que en este caso pueden ser las calificaciones que obtuvieron los alumnos de cada profesor. La dificultad surge al manejar las conexiones o ligas entre los registros y sus correspondientes registros conectores. Es un modelo poco utilizado. (Fábregas, 2007)

Modelo Relacional

El modelo de datos relacional es el más utilizado en la actualidad para representar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en

los modelos de base de datos. Su idea fundamental es el uso de “relaciones”, estas podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". A pesar de ser la teoría de las BD relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar, concibiendo en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la BD. La información puede ser recuperada o almacenada mediante "consultas", que ofrecen una amplia flexibilidad y eficiencia para administrar la información. El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales. Durante su diseño, una base de datos relacional pasa por un proceso conocido como normalización de una BD. (Alarcón, 2006)

Modelo Orientado a Objetos

Las BD orientadas a objetos se crearon para tratar de satisfacer las necesidades de las nuevas aplicaciones. La orientación a objetos ofrece flexibilidad para manejar algunos de estos requisitos y no está limitada por los tipos de datos y los lenguajes de consulta de los sistemas de bases de datos tradicionales. Una característica clave de las Bases de Datos Orientadas a Objetos (BDOO) es la potencia que proporcionan al diseñador al permitirle especificar tanto la estructura de objetos complejos, como las operaciones que se pueden aplicar sobre dichos objetos. Otro motivo de la creación de estos modelos es el creciente uso de los lenguajes orientados a objetos.

El modelo seleccionado para el desarrollo de la BD fue el relacional, pues se adapta al problema a resolver, es un modelo maduro, ha tenido un buen desempeño y ha alcanzado un gran éxito reflejado en su amplio mercado. El tipo de BD será dinámica, debido a que la información guardada en la aplicación podrá ser modificada.

1.6. Importancia de la Capa de Acceso a Datos (CAD).

Actualmente, los equipos de desarrollo, en su mayoría, escogen arquitecturas multicapa para el desarrollo de aplicaciones Web, de modo que utilicen una CAD que encapsule todas las interacciones con la base de datos. Así, se simplifican las llamadas desde la capa de presentación, y permite encapsular la lógica de acceso a datos, siendo transparentes las operaciones de la CAD a la capa lógica del negocio. En la CAD es donde se implementan los componentes que interactúan con la BD y donde se encapsula el acceso a datos con estos componentes de una manera fácil, para lograr un rendimiento óptimo. Permite que las demás capas, componentes e interfaces no interactúen directamente con el servidor de BD y así excluir las complejidades del acceso a datos en el código fuente de otras capas.

La CAD juega un importante papel dentro de la arquitectura de las aplicaciones, ella provee muchas ventajas claves entre las que tenemos:

- **Separación entre la capa de negocio y los gestores de BD:** los componentes de la CAD implementan aquellas funciones especializadas en el acceso a los diferentes servidores de BD y brindar esos datos, en un formato adecuado a la capa de negocio de la aplicación. Este nivel de aislamiento protege a los componentes de las otras capas, específicamente de la capa de negocio, de los cambios potenciales que puedan ocurrir en el servidor de BD.
- **Mejoras en el mantenimiento de la aplicación:** encapsula toda la gestión de acceso a datos en una sola capa, que ofrece la ventaja de reutilizar componentes, reduciendo la cantidad de código fuente a desarrollar y mantener.

1.7. Diseño

En el contexto de la Ingeniería de Software, el diseño se centra en cuatro áreas importantes de interés: datos, arquitectura, interfaces y componentes. El diseño de software es uno de los pilares fundamentales de la Ingeniería de Software. Es la única forma de convertir exactamente los requisitos de un cliente en un producto o sistema de software finalizado. Entre las tareas fundamentales del diseño están: producir un diseño de datos, un diseño arquitectónico, un diseño de interfaz y un diseño de componentes. (Pressman, 2005)

1.7.1. Calidad del diseño

La importancia del diseño de software se puede describir con una sola palabra “calidad”. El diseño es el lugar en donde se fomentará la calidad en la Ingeniería de Software. Existen tres características fundamentales que se deben tener en cuenta a la hora de realizar un buen diseño. (Pressman, 2005)

Características:

- El diseño deberá implementar todos los requisitos explícitos del modelo de análisis, y deberá ajustarse a todos los requisitos implícitos que desea el cliente.
- El diseño deberá ser una guía legible y comprensible para aquellos que generan código y para aquellos que comprueban y consecuentemente, dan soporte al software.
- El diseño deberá proporcionar una imagen completa del software, enfrentándose a los dominios de comportamiento, funcionales y de datos desde una perspectiva de implementación.

Si se tienen en cuenta estas características cuando se realice el diseño se está garantizando que éste tenga calidad. Además existen otros aspectos que se deben tener en cuenta a la hora de diseñar.

Existen varios criterios por los cuales se puede evaluar la calidad del diseño:

- Un diseño deberá presentar una estructura jerárquica que facilite el trabajo con los componentes del software.
- Un diseño deberá ser modular, es decir que deberá dividirse de forma lógica en elementos que realicen funciones específicas.
- Un diseño deberá contener distintas representaciones de datos y procedimientos.
- Un diseño deberá conducir a módulos que tengan características funcionales independientes.
- Deberá conducir a interfaces que reduzcan la complejidad de las conexiones entre los módulos y el entorno externo.
- Deberá producir un diseño mediante un método que pudiera repetir la información obtenida durante el análisis de los requisitos del software.

1.7.2. Principios del diseño

Roger Pressman en su libro: “Ingeniería del Software. Un enfoque práctico” plantea una serie de principios básicos del diseño que hacen posible que el ingeniero de software navegue por el proceso de diseño, y que hay que tener en cuenta a la hora de diseñar ya que ellos garantizan el correcto funcionamiento del sistema.

- En el proceso de diseño no deberá utilizarse “orejeras”. (Pressman, 2005)
- El diseño deberá poderse rastrear hasta el modelo de análisis.
- El diseño no deberá inventar nada que ya esté inventado.
- El diseño deberá minimizar la distancia intelectual entre el software y el problema como si de la vida real se tratara.
- El diseño deberá presentar uniformidad e integración.
- El diseño deberá estructurarse para admitir cambios.
- El diseño deberá estructurarse para degradarse poco a poco, incluso cuando se enfrenta con datos, sucesos o condiciones de operación aberrantes.
- El diseño no es escribir código y escribir código no es diseñar.
- El diseño deberá evaluarse en función de la calidad mientras se va creando, no después de terminado.
- El diseño deberá revisarse para minimizar los errores conceptuales (semánticos).

1.8. Metodología de desarrollo de software

El término Metodología se define como un conjunto de métodos eficientes orientados a conseguir un objetivo propuesto. Son un conjunto de procesos que organizados dan una secuencia de pasos a seguir para obtener los hitos propuestos y finalmente el producto final. Las metodologías de desarrollo de software constituyen un factor importante para lograr productos con calidad en el tiempo requerido. (López Barrio, 2005)

1.8.1. Programación Extrema (Extreme Programming, XP)

La Programación Extrema surge gracias a la idea de Kent Beck, como proceso de creación de software diferente al convencional. XP es una metodología ágil de desarrollo de software en la que se postula que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos, por tanto, ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos. XP se emplea en proyectos cortos y con poco personal de desarrollo. Basada en la retroalimentación continúa entre el cliente y el equipo de desarrollo. Se caracteriza por la simplicidad, la comunicación, la realimentación o reutilización del código desarrollado así como las pruebas unitarias. Según Beck: "XP es una metodología ligera, eficiente, con bajo riesgo, flexible, predecible y divertida para desarrollar software". Es una de las metodologías de desarrollo de software más utilizadas y exitosas en la actualidad. Una de las razones que la hacen peculiar es la presencia constante e implicación del usuario final como parte del equipo de trabajo, factor vital para el éxito del proyecto. (Ortiz, Escobar. 2008)

1.8.2. Scrum

Es una metodología de desarrollo ágil desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle, indicada para proyectos con un rápido cambio de requisitos. Las principales características que la definen es que el desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento del producto que se muestra al cliente. El marco para la gestión de proyectos que define se ha utilizado con éxito durante los últimos 10 años. Otra cuestión peculiar de Scrum son las reuniones a lo largo del proyecto. De todas ellas la más importante se efectúa diariamente durante 15 minutos por parte del equipo de desarrollo para coordinar e integrar el trabajo. Su práctica es trabajar con un solo representante, el dueño del producto final, aunque últimamente se estila crear un grupo de clientes finales para darle agilidad al proceso (Larman, 2003).

1.8.3. Rational Unified Process (RUP)

RUP es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organización, diferentes niveles de aptitud y diferentes tamaños de proyecto. Está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes de software interconectados a través de interfaces bien definidas y utiliza el Lenguaje Unificado de Modelado (Unified Modeling Language, UML) como soporte a la metodología. El Proceso Unificado de Rational está dirigido por casos de uso, centrado en la arquitectura y es iterativo e incremental (Jacobson, 2000). RUP se caracteriza por dividir el ciclo de vida de la producción del software en 4 fases:

1. Inicio: es donde se determina la visión del proyecto, o sea se comprende el entorno y se determina el alcance del producto.

2. Elaboración: en esta etapa se determinan los cimientos de la arquitectura y se analiza el dominio del problema.

3. Construcción: en esta fase se obtiene la capacidad operacional inicial del producto.

4. Transición: se obtiene el release o liberación del producto y se pone en manos de los usuarios finales. Las fases antes mencionadas se llevan a cabo desarrollando nueve flujos de trabajo principales. Los seis primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo:

- Modelado de Negocio
- Requerimientos
- Análisis y Diseño
- Implementación
- Prueba
- Despliegue
- Configuración y Control de Cambios
- Gestión de Proyectos

- Entorno

RUP identifica claramente a los profesionales (actores) involucrados en el desarrollo del software y sus responsabilidades en cada una de las actividades. Además, explícitamente indica qué actor es responsable de qué artefacto en cada actividad.

Justificación de la metodología seleccionada.

Después de realizado el estudio de las principales metodologías de desarrollo de software por parte del equipo de desarrollo se optó por la utilización de RUP por ser una metodología diseñada para proyectos, además de hacer énfasis en realizar una buena captura de los requisitos. El NVP no es un sistema de compleja construcción, pero requiere que su ejecución sea a largo plazo y con un equipo de desarrollo numeroso. Este presenta un considerable conocimiento de la metodología seleccionada proporcionando una mayor calidad del producto final.

1.9. Principales Gestores de Bases de Datos

No es posible hablar de las bases de datos sin mencionar a los SGBD. Los SGBD son un conjunto de programas que permiten la implementación, acceso y mantenimiento de la base de datos. El SGBD, junto con la base de datos y los usuarios, constituyen el Sistema de Base de Datos. Los SGBD se pueden agrupar en tres grandes grupos: SGBD Relacionales, SGBD Orientado a Objetos y SGBD Objeto/Relacional. El objetivo fundamental de un SGBD consiste en suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos, o sea, de forma que no le sea necesario conocer el modo de almacenamiento de los datos en la computadora, ni el método de acceso empleado. Los programas de aplicación operan sobre los datos almacenados en la base utilizando las facilidades que brindan los SGBD, los que, en la mayoría de los casos, poseen lenguajes especiales de manipulación de la información que facilitan el trabajo de los usuarios. (Alburquerque, 2007)

1.9.1. SQL Server

SQL Server fue desarrollado por Microsoft y permite la gestión de un entorno de base de datos relacional. A través de una interfaz cómoda y fácil de trabajar. SQL Server abarca tanto el área de diseño como la de

administración. Es nombrado SQL porque hace uso de este lenguaje para la definición y manejo de los datos, y se llama Server porque dispone de una parte servidora que es responsable de atender a los procesos clientes, que son aquellos que realizan las peticiones al servidor, cumpliendo con una arquitectura cliente-servidor.

La arquitectura cliente-servidor organiza o estructura el sistema en tres capas: una capa de presentación o capa de los clientes, una capa de lógica del negocio o red y una capa de datos o servidora. En la capa de presentación se encuentra una interfaz de usuario y los métodos que utiliza para la conexión con el servidor de datos. Las aplicaciones que se desarrollan haciendo uso de SQL Server pueden utilizar diferentes métodos de conexión entre los que se encuentran ODBC (Open Database Connectivity o Conectividad Abierta de la Base de Datos), OLE DB (Object Linking and Embedding for Databases) y ADO.NET (ActiveX Database Objects). SQL (Structured Query Lenguaje), Lenguaje de Consultas Estructurado, es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Reúne las características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de una base de datos de una forma sencilla. (Woody, 2006)

1.9.2. MySQL

MySQL es una idea originaria de la empresa opensource MySQL AB establecida inicialmente en Suecia en el año 1995 y cuyos fundadores son David Axmark, Allan Larsson, y Michael "Monty" Widenius. El objetivo que persigue esta empresa consiste en que MySQL cumpla el estándar SQL, pero sin sacrificar velocidad, fiabilidad o usabilidad. MySQL es muy utilizado en aplicaciones web y en plataformas como Linux y Windows. Su popularidad está muy ligada a PHP, con el cual aparece muy a menudo en combinación. Se han desarrollado varias versiones, siendo la última la 5.0.22 y una de las más usadas en el mundo.

Entre las características disponibles en las últimas versiones podemos destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.

- Diferentes opciones de almacenamiento, si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto. (Pabón, 2005)

1.9.3. PostgreSQL

PostgreSQL es un servidor de base de datos objeto relacional libre, liberado bajo la licencia BSD. Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo, dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

Entre las principales características de PostgreSQL están:

- Alta concurrencia: Mediante un sistema denominado MVCC (Acceso Concurrente Multiversión, por sus siglas en inglés). Permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.
- Soporta casi toda la sintaxis SQL, el uso de foreign keys, joins, views, triggers, y stored procedures (en múltiples lenguajes).
- Cliente/Servidor: usa una arquitectura proceso-por-usuario cliente/servidor. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.
- Lenguajes Procedurales: tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al de Oracle, PL/SQL. Otra ventaja es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido, además de C, C++ y, Java.
- Interfaces con lenguajes de programación: La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas

Interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, Pike, etc.

- Herencia de tablas.
- Incluye la mayoría de los tipos de datos SQL92 y SQL99 (INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, y TIM ESTAMP), soporta almacenamiento de objetos grandes binarios, además de tipos de datos y operaciones geométricas.
- Puntos de recuperación a un momento dado, tablespaces, replicación asincrónica, transacciones jerarquizadas (savepoints), copia de seguridad en línea.
- Un sofisticado analizador/optimizador de consultas.
- Soporta juegos de caracteres internacionales, codificación de caracteres multibyte. (Kuroki, 2005)

Dadas las características, ventajas y desventajas de estos tres SGBD se seleccionó PostgreSQL teniendo en cuenta los lineamientos de la UCI de la migración a Software Libre, por el gran rendimiento y estabilidad a la hora de manejar grandes cantidades de datos y ser crucial en el manejo de altos niveles de concurrencia con rapidez.

1.10. Clientes para administrar PostgreSQL

1.10.1. PgAdmin3

PgAdmin 3 es una aplicación gráfica para el uso del gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usar en Linux, FreeBSD, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL como Pervasive Postgres, EnterpriseDB, Mammoth Replicator y SRA PowerGres.

PgAdmin 3 está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar

scripts programados, soporte para el motor de replicación Slony-I y mucho más. La conexión al servidor puede hacerse mediante conexión TCP/IP o Unix Domain Sockets (en plataformas *nix), y puede encriptarse mediante SSL para mayor seguridad. (pgAdmin, 2009)

1.10.2. PhpPgAdmin

Es una herramienta de administración PostgreSQL, basada en una interface Web, entre sus principales características ofrece la posibilidad de administrar varios servidores, soporte a múltiples versiones de PostgreSQL, administración de usuarios, grupos, bases de datos, esquemas, etc. Manipulación sencilla de datos, exportar los datos a diferentes formatos, importar sentencias SQL y otras características. (LinuxPartyGroup, 2008)

De estos dos clientes se hace uso de PgAdmin3, porque permite una fácil administración de la base de datos mediante una interfaz gráfica sencilla, además de ser el más usado en el mundo para la administración PostgreSQL.

1.11. Herramientas CASE que se utilizan para el diseño de bases de datos

Las herramientas CASE son de gran ayuda para el desarrollo de un software. Son un grupo de programas que utilizan las personas que intervienen en el desarrollo de un software, como es el caso de los diseñadores, desarrolladores, analistas, entre otros, durante las fases del desarrollo del software (análisis, diseño, implementación), para agilizar y facilitar su trabajo, ya que dichas herramientas proveen de métodos, técnicas y utilidades que ayudan al perfeccionamiento del desarrollo de sistemas de información, de forma total o parcialmente.

Entre las herramientas CASE más utilizadas para el diseño de base de datos se encuentran:

1.11.1. ERwin

Es una herramienta de diseño de base de datos. Brinda productividad en diseño, generación, y mantenimiento de aplicaciones. Desde un modelo lógico de los requerimientos de información, hasta el modelo físico perfeccionado para las características específicas de la BD diseñada. Permite visualizar la

estructura, los elementos importantes, y optimizar el diseño de la BD. Genera automáticamente las tablas y miles de líneas de stored procedure y triggers para los principales tipos de BD. Realizar un diseño fácil de una BD. Los diseñadores de BD sólo apuntan y pulsan un botón para crear un gráfico del modelo E-R (Entidad-Relación) de todos sus requerimientos de datos y capturar las reglas de negocio en un modelo lógico, mostrando todas las entidades, atributos, relaciones, y llaves importantes. Más que una herramienta de dibujo, ERwin automatiza el proceso de diseño de una manera inteligente. Soporta principalmente BD relacionales SQL y BD que incluyen Oracle, Microsoft SQL Server, Sybase, DB2, e Informix. El mismo modelo puede ser usado para generar múltiples BD, o convertir una aplicación de una plataforma de BD a otra. (Geocities, 2009)

1.11.2. Rational Rose Data Modeler

Es una herramienta de modelado visual que posibilita que miembros de un equipo de desarrollo trabajen juntos, capturando y compartiendo los requerimientos de negocio y dándoles seguimiento a medida que cambian a través del proceso. Proporciona una realización de la metodología E-R usando la notación UML para unificar a los diseñadores de BD con el equipo de desarrollo de software. Con UML, el diseñador de bases de datos puede capturar información tal como restricciones, triggers e índices directamente en el diagrama en lugar de tener que representarlos con propiedades ocultas por fuera. Rational Rose Data Modeler da la libertad de pasar del objeto al modelo de datos y sacar ventaja de los tipos de transformación básicos como las relaciones muchos-a-muchos. Esta herramienta brinda una forma intuitiva de visualizar la arquitectura de las bases de datos y la forma en que se vincula con la aplicación. (GSInnova, 2009)

1.11.3. Visual Paradigm

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Características:

- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento Modelado colaborativo con CVS y Subversion.
- Ingeniería inversa - Código a modelo, código a diagrama.
- Ingeniería inversa Java, C++, Esquemas XML, XML, .NET exe/dll, CORBA IDL.
- Generación de código - Modelo a código, diagrama a código.
- Soporte ORM - Generación de objetos Java desde la base de datos.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- Generador de informes para generación de documentación (Manager, 2009).

Considerando todas las facilidades brindadas por el Visual Paradigm, que es un producto de calidad, muy fácil de instalar y actualizar, que está disponible en varias ediciones entre las que se encuentran Enterprise, Professional, Community, Standard, Modeler y Personal ofreciéndoles a los usuarios la posibilidad de escoger según sus necesidades. Es multiplataforma, además la UCI pagó la licencia para su uso lo que constituye una poderosa ventaja.

1.12. Fundamentación de los lenguajes de programación

Para el desarrollo del NVP se compararon numerosas alternativas de desarrollo. Desde usar un único lenguaje de alta eficiencia como C++ y basarse en librerías intermedias para lograr portabilidad, hasta lenguajes que no dependieran en absoluto de la máquina de ejecución como puede ser Java. Pero sin perder de vista que para el objetivo fundamental de las operaciones también es necesario altas capacidades de cálculo en punto flotante y una eficiente gestión de la memoria. (Lucas Rodríguez F., 2005)

Los lenguajes que se tuvieron en cuenta, por su uso difundido fueron: Java, C++ y C#. (Gámez, 2008)

1.12.1. Lenguaje C++

Desde sus inicios, C++ intentó ser un lenguaje que incluyera completamente al lenguaje C (quizá el 99% del código escrito en C es válido en C++) pero al mismo tiempo incorpora muchas características sofisticadas no incluidas en aquel, tales como: programación orientada a objetos, excepciones, sobrecarga de operadores, templates o plantillas. (Gámez, 2008)

1.12.2. Lenguaje Java

La sintaxis del lenguaje Java heredó características de C y C++, explícitamente eliminando aquellas que para muchos programadores (según los diseñadores) resultan excesivamente complejas e inseguras. En la actualidad su uso es promovido para el desarrollo de aplicaciones empresariales del lado del servidor, especialmente a través del estándar J2EE, así como en dispositivos móviles (a través del estándar J2ME.) En realidad, Java hace referencia a un conjunto de tecnologías entre las cuales el lenguaje Java es sólo una de ellas. Por tal motivo muchas veces se habla de la "plataforma Java", la cual es indesligable del lenguaje. (Gámez, 2008)

1.12.3. Lenguaje C#

El lenguaje C# es una versión avanzada de C y C++ y se ha diseñado especialmente para el entorno .NET. Es un nuevo lenguaje orientado a objetos empleado por programadores de todo el mundo para desarrollar aplicaciones que se ejecuten en la plataforma .NET. C# es un paso muy importante en la evolución de los lenguajes de programación, y es una solución ideal para las aplicaciones de alto nivel. Con C# se puede desarrollar todo tipo de proyectos de aplicaciones cliente/servidor. (Gámez, 2008)

C# amplía las capacidades de C, C++, Visual Basic y Java para proporcionar un completo entorno de desarrollo en el que crear aplicaciones. C# mezcla la potencia de C, las capacidades de orientación a objetos de C++ y la interfaz gráfica de Visual Basic. (Geetanjali A., 2007)

En la tesis de Maestría “Herramienta interactiva para la enseñanza en la carrera de Ingeniería Automática: Nodo Virtual de Procesos.” de la Ingeniera Automática Yalice Gámez Batista, a la cual tributa el presente Trabajo de Diploma, se realizó un estudio comparativo sobre los lenguajes antes mencionados en el que se tuvieron en cuenta factores como: portabilidad, capacidades 2d/3d, matemática de precisión compleja,

gestión de memoria, velocidad de ejecución, licencia, eficiencia y modularidad; asignando un peso a cada factor obteniéndose como resultado la selección del lenguaje C++ para implementar el NVP.

1.13. Fundamentación de los entornos integrados de desarrollo

Un entorno de desarrollo integrado, (IDE Integrated Development Environment) es un entorno de programación empaquetado como un programa, que brinda una serie de herramientas y facilidades a los programadores y pueden centrarse en uno o a varios lenguajes.

1.13.1. NetBeans

NetBeans IDE es un entorno de desarrollo - una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. También está disponible NetBeans Platform; una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones. Ambos productos son de gratuitos y de código abierto para uso tanto comercial como no comercial. (NetBeans, 2009)

1.13.2. Eclipse

Eclipse, se ha convertido en un potencial IDE de desarrollo para los programadores de aplicaciones Java, por las capacidades de implementación que éste les brinda. Es una plataforma universal para integrar herramientas de desarrollo, con una arquitectura abierta y basada en plug-ins. Además da soporte a todo tipo de proyectos que abarcan desde el ciclo de vida del desarrollo de aplicaciones, incluyendo soporte para modelado. (Fernández de la Pera, 2007)

La arquitectura de plug-ins permite integrar diversos lenguajes sobre un mismo IDE o introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo, como: herramientas UML, editores visuales de interfaces y ayuda en línea para librerías. Conservan el registro de las

versiones y generan y mantienen la documentación de cada etapa del proyecto. Además incluye asistentes para la creación de interfaces y clases, y proporciona una integración perfecta con el sistema open source CVS (Concurrent Version System), que sirve de gran utilidad para llevar el control de la versiones con las que se trabaja, y conocer en todo momento sus respectivas diferencias. (javaHispano, 2009)

1.13.3. QtCreator

Es un IDE creado por Trolltech para el desarrollo de aplicaciones con las bibliotecas Qt, requiriendo su versión 4.x. Los sistemas operativos que soporta en forma oficial son:GNU/Linux 2.6.x, para versiones de 32 y 64 bits con Qt 4.x instalado. Además existe una versión para Linux con gcc 3.3.

- Mac OS X 10.4 o superior, requiriendo Qt 4.x
- Windows XP y Vista, requiriendo el compilador MinGW y Qt 4.4.3 para MinGW
- QtCreator está desarrollado en C++ y el rendimiento es mucho mejor que NetBeans y Eclipse.
- Posee una interfaz gráfica sencilla.
- Es sencillo de usar.

Tiene acceso a las claves derivadas de QObject con acceso a todos sus miembros y a todos los archivos de cabecera de esta librería. El sistema de construcción está basado en qmake.
El API (Application Programming Interface) de la biblioteca hace uso de las QT Development Tools, que son herramientas para acceder a bases de datos mediante SQL, así como uso de XML para el manejo de ficheros, además de estructuras de datos tradicionales desde el mismo QtCreator. Además de importantes novedades como una mejora sustancial de rendimiento, motores de renderizado intercambiables, actualización de la versión de WebKit incluida. (Expressions, 2009)
Incluye:

- Una avanzada de código C++ editor.
- Integrado GUI diseño y formas de diseño.
- Proyecto y construcción de instrumentos de gestión.
- Integrado, ayuda sensible al contexto del sistema.
- Depurador de Visual.

- Soporta múltiples plataformas.

Se escogió QtCreator como IDE de desarrollo por utilizar el lenguaje de programación C++ escogido anteriormente, es multiplataforma y brinda grandes ventajas para la conexión a cualquier BD.

Conclusiones parciales

En este capítulo se hace un análisis de los principales aspectos y conceptos relacionados con el Nodo Virtual, cómo las aplicaciones existentes implementan las funcionalidades propuestas para el NVP. También se analizan temas y conceptos de importancia dentro del diseño de software, pretendiendo dejar sentadas las bases teóricas del trabajo a desarrollar. Se realiza un estudio de las posibles herramientas a utilizar. Como resultado de este análisis se determina lo siguiente:

- Es necesaria la construcción de un Nodo Virtual de Procesos, ya que esta herramienta será de valor incalculable para que los estudiantes desarrollen habilidades y pongan en práctica conocimientos adquiridos.
- Existe la necesidad de desarrollar un producto multiplataforma, teniendo en cuenta que el sistema es para la simulación de procesos (actividad que genera mucha información). Una de las características principales es la velocidad para realizar las operaciones, lo cual incidirá en la calidad del trabajo del sistema.
- Debido a los anteriores requerimientos se utilizará RUP como metodología de desarrollo, PostgreSQL en su versión 8.3 como gestor de base de datos, el PgAdmin3 para la administración del PostgreSQL, se empleará como herramienta de modelado Visual Paradigm en su versión 6.1, C++ como lenguaje de programación y el IDE a utilizar será QtCreator.

Capítulo 2: Descripción y análisis de la solución propuesta.

Introducción

En este capítulo se definen las actividades más importantes que posibilitaron la construcción de la capa de acceso a datos y la base de datos, teniendo como resultado final el diseño e implementación de las mismas.

Primero se seleccionan y argumentan los requisitos funcionales y no funcionales del sistema propuesto. Se realiza un estudio del diagrama de clases persistente obtenido a partir del diagrama de clases del diseño por su importancia para la confección del modelo de datos. Se confecciona el diagrama entidad relación de la BD y se describen las tablas que la componen.

2.1. Requisitos funcionales y no funcionales

Para que un sistema de software funcione correctamente se debe lograr una comunicación efectiva entre los usuarios y el equipo de proyecto con el objetivo de llegar a un entendimiento de lo que hay que hacer, lo que constituye la clave del éxito en la producción de un software. Aquí radica la importancia que en los últimos años se le ha dado a la identificación de los requerimientos como parte del proceso de desarrollo del software.

En la tesis titulada “Análisis y Diseño de una herramienta interactiva de simulación de procesos: Nodo Virtual de Procesos. Segunda Iteración”, de los autores Eliober Cleger Despaigne y Annarella M. Tornés Montes de Oca, se realizaron los artefactos que sirven de entrada al diseño e implementación de la capa de acceso a datos y la base de datos del NVP que se desea construir. A continuación se presentan los requisitos funcionales y no funcionales, que se proponen en dicho trabajo.

La IEEE define un requisito como:

- Una condición o capacidad necesaria para un usuario para resolver un problema o alcanzar un objetivo.

- Una condición o capacidad que debe ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento formalmente impuesto.
- Una representación documentada de una condición o capacidad dada en los puntos 1 o 2. (IEEE, 1993)

2.1.1. Requisitos Funcionales

En esta sección se presentan los requisitos funcionales fundamentales del sistema, que constituyeron un paso esencial para el diseño de las entidades de la BD propuesta.

Los requisitos funcionales fueron elicitados mediante la aplicación de técnicas de obtención de requisitos como: (entrevista, arqueología de documentos, mapas conceptuales, tormentas de ideas, cuestionarios y desarrollo conjunto de aplicaciones (Joint Application Development, JAD)). Los mismos fueron analizados y especificados mediantes los diagramas de Casos de Uso de Sistema y la Especificación de Requisitos de Software. Estos requisitos fueron validados por el prototipado de interfaces no funcionales y mediante la aplicación de métricas (calidad de especificación de los requisitos). (Cleger, Tornés.2009)

A continuación se presentarán los requisitos funcionales:

RF 1. Gestionar Conexión al Sistema.

RF 1.1. Registrar.

RF 1.2. Autenticar.

RF 2. Gestionar Conexión a Proceso

RF 2.1 Conectar un usuario a un proceso.

RF 2.2 Desconectar usuario de un proceso.

RF 3. Gestionar Usuario

RF 3.1 Adicionar un Usuario

RF 3.2 Modificar datos de un Usuario

RF 3.3 Eliminar un Usuario

RF 3.4 Mostrar un listado de los Usuarios.

RF 3.5 Mostrar listado de usuarios conectados a un proceso.

RF 4. Gestionar Tipos de Procesos

RF 4.1 Adicionar un tipo de proceso.

RF 4.2 Modificar datos un tipo de proceso.

RF 4.3 Eliminar un tipo de proceso.

RF 4.4 Mostrar listado de los tipos de procesos existentes.

RF 5. Gestionar Procesos

RF 5.1 Adicionar un proceso.

RF 5.2 Modificar datos de un proceso.

RF 5.3 Eliminar un proceso

RF 5.4 Mostrar un listado de los procesos existentes.

RF 5.5 Mostrar listado de procesos activos.

RF 6. Gestionar Controladores

RF 6.1 Adicionar controladores.

RF 6.2 Modificar datos de controladores.

RF 6.3 Eliminar controladores.

RF 6.4 Mostrar listado de los controladores.

RF 7 Gestionar Límite

RF 7.1 Adicionar un límite.

RF 7.2 Modificar datos de un límite.

RF 7.3 Eliminar límite.

RF 7.4 Mostrar listado de los límites.

RF 8. Gestionar Variables

RF 8.1 Adicionar variables.

RF 8.2 Modificar datos de las variables.

RF 8.3 Eliminar variables.

RF 8.4 Mostrar listado de las variables.

RF 9. Gestionar Reportes

RF 9.1 Mostrar listado usuarios conectados al sistema.

RF 9.2 Mostrar listado de usuarios conectados a un proceso.

RF 9.3 Mostrar listado de los procesos activados por un usuario.

RF 9.4 Mostrar un listado de los procesos por tipo de proceso.

2.1.2. Requisitos No Funcionales

A continuación se presentarán los requisitos no funcionales:

Hardware

- Servidor
  - Discos duros: SCCSI.
  - Microprocesador: 3GHz como mínimo.
  - Memoria RAM: 1Gb.
  - PC PENTIUM IV o superior.
  - 3Gb de espacio libre en el disco duro.
- . PC clientes
  - Microprocesador: 2 GHz como mínimo.

- Memoria RAM: 512 MB.
- PC PENTIUM IV o superior.
- Impresora.

Software

- Servidor

  - Sistema Operativo: Ubuntu Server.
  - Gestor de Base de datos: PostgreSQL.

- . PC cliente

  - Sistema Operativo: Windows 2000 o superior, Linux.

Seguridad

- Cada usuario accederá sólo a la información que le corresponda según su nivel de acceso.
- Se llevará a cabo tratamiento de excepciones.
- El tipo de tecnología para la configuración que tendrán los discos duros en el servidor será Raid.

Redes

- La red existente en las instalaciones debe de soportar la tecnología multicast.
- Las transacciones y recuperación de los datos en la comunicación servidor - PC cliente, se realizara a través del protocolo TCP/ IP.
- Protocolos de comunicación con controladores: TCP/IP, DCOM, DDE y OPC.

Portabilidad

- La aplicación deberá correr sobre cualquier Sistema Operativo.

Restricciones de diseño e implementación

- Uso de plataforma libre.
- Se utilizará el lenguaje de programación C++.
- El IDE a usar será QtCreator.
- Empleo de la filosofía de la Programación Orientada a Objetos (POO).

2.2. Estrategia de integración de la solución con otros módulos

La herramienta NVP está estructurada en cinco módulos: Simulación, Reporte, Conexión, Graficar y Gestión; a través de los cuales se garantiza la gestión de toda la información necesaria para el funcionamiento del sistema. Los datos que provienen de estos módulos se almacenarán en una BD. Las clases persistentes se identificaron a partir del modelo de diseño que se obtuvo del trabajo realizado por los analistas. Se integraron las clases persistentes de cada uno de los módulos en un único diagrama de clases permitiendo la creación del diagrama Entidad-Relación para la representación de la estructura lógica y física de la BD, a través de las relaciones entre las diferentes entidades obtenidas de cada módulo.

Todas las peticiones a la BD son manejadas por las clases obtenidas a partir de las persistentes, que conforman la capa de acceso a datos, mediante consultas SQL que se encuentran en las mismas. Esto permite conservar la sintaxis del lenguaje de consultas SQL tanto como sea posible. Además, es preciso mencionar que las consultas no se construirán ni se guardarán en la BD lo, que posibilita que si se decide cambiar el SGBD no se deban construir de nuevo.

2.3. Descripción de la arquitectura y fundamentación

La definición oficial de Arquitectura de Software es la que propone la IEEE y plantea que: “La arquitectura de software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán y los principios que sustentan su diseño y evolución”.

Para el diseño del NVP se hizo uso del estilo arquitectónico en Capas como nivel de abstracción más alto de la estructura del sistema, que corresponden con las capas presentación negocio y acceso a datos

Capa de Presentación

Es la que interactúa directamente con el usuario, captura la información de entrada por éste (realiza un filtrado previo para comprobar que no hay errores de formato) y hace las peticiones a la capa inferior mostrando al usuario la respuesta proveniente de ésta. Únicamente se comunica con la capa de negocio. (García, Plasencia. 2007)

- **Capa de Negocio**

Está conformada por subsistemas, que se ajustan a los requisitos y casos de uso arquitectónicamente significativos. Desde el punto de vista de diseño, esta capa es contenedora de las clases entidades y controladoras. Únicamente se comunica con la capa de Acceso a Datos. (García, Plasencia. 2007)

- **Capa de Acceso a Datos**

Contiene clases que interactúan con la base de datos y permiten, utilizando los procedimientos almacenados generados, realizar todas las operaciones con la base de datos de forma transparente para la capa de negocio. (García, Plasencia. 2007)

En el caso específico del Nodo Virtual la capa de presentación estaría compuesta por todos los elementos de la interfaz: representaciones gráficas de los diferentes procesos que se simulan, tablas con el historial de los valores de entrada y salida, gráficos que muestren el comportamiento del proceso. La capa de negocio abarca los módulos de Simulación, Reporte, Conexión, Graficar y Gestión. Por otro lado todo lo concerniente a la gestión de la persistencia de los datos está enmarcado en la capa de acceso datos. Esta capa cuenta con una fachada que contiene todos los métodos que serán utilizados en la capa de negocio estableciéndose de esta forma la comunicación entre ambas capas.

La distribución de los módulos por capa se refleja en la siguiente figura:

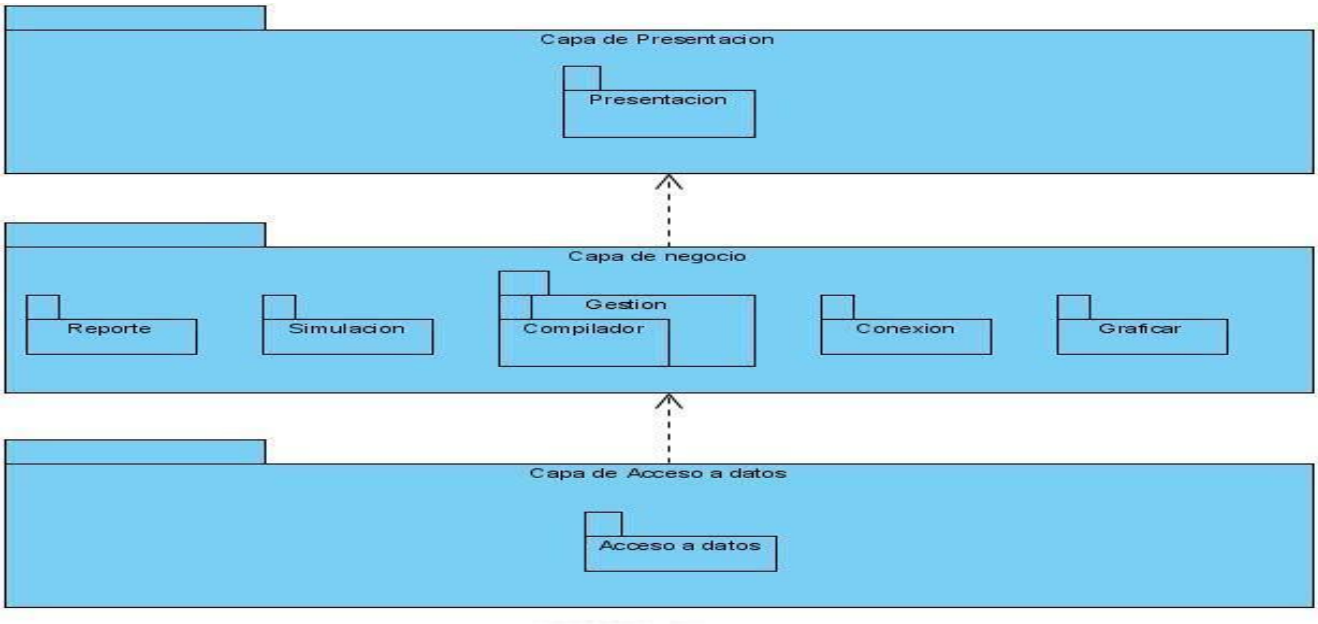


Figura 1 Arquitectura en Capas de NVP

2.4. Patrones de arquitectura

Un patrón es una solución probada que se puede ser aplicada con éxito a un determinado tipo de problemas que aparecen repetidamente en algún campo.

Por tanto, un patrón codifica conocimientos específicos acumulados por la experiencia, describiendo un problema que ocurre una y otra vez en nuestro ambiente y luego el núcleo de la solución a ese problema (Perera. 2007).

Durante la implementación de la capa de Acceso a Datos del NVP se aplica el patrón DAO para dar solución a la misma, el cual se describe a continuación.

- Nombre: Data Access Object (DAO)

- Problema: Este patrón encapsula la forma de acceder a la fuente de datos, resolviendo el problema de contar con diversas fuentes de datos como son las bases de datos, los archivos, los servicios externos. Su uso se extiende al problema de ocultar la forma de acceder a los datos. (Lago, Ramiro. 2007).

- Solución: La creación de los Objetos de Acceso a Datos (DAO) permite acceder a los datos a través de estos objetos y no tener que cambiar los objetos de negocio.

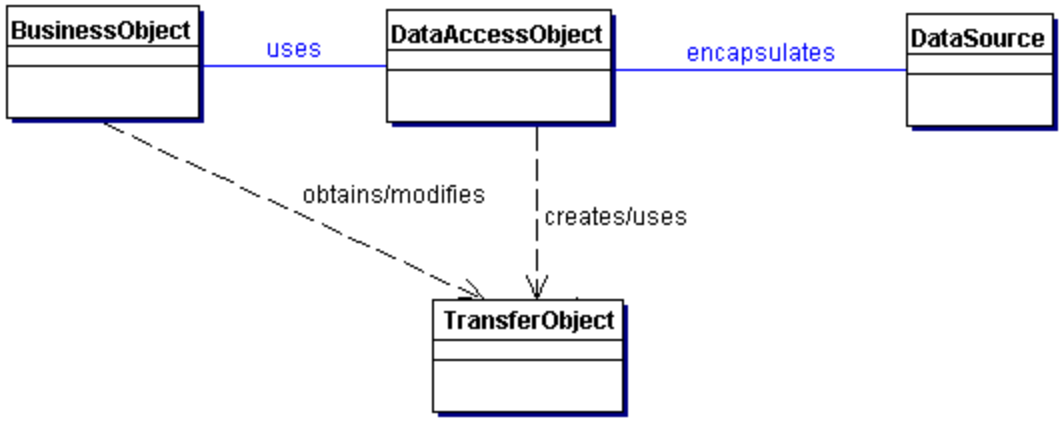


Figura 2 Patrón DAO

2.5. Diseño

El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. En el diseño se modela el sistema y se define una arquitectura que soporte todos los requisitos. Concretamente se pueden definir como propósitos del diseño:

- Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario.
- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.

Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo. (Jacobson, 2004)

2.5.1. Diagrama de clases del diseño

Los diagramas de clases representan las relaciones entre las clases y sus objetos. Las clases están compuestas por varios atributos y métodos que permiten darle solución a las funcionalidades de cada caso de uso. (**Ver Anexos 1-7)**.

2.5.2. Diagrama de clases persistentes

Las clases persistentes por lo general tienen como origen las clases clasificadas como entidad porque ellas modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto del mundo real o un suceso. Todas las clases identificadas en el dominio del análisis no son persistentes. La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. El Diagrama de Clase se utiliza para modelar la estructura lógica de la BD, con clases representando tablas, y atributos de clase representando columnas. Las clases persistentes y sus atributos hacen referencia directamente a las entidades lógicas y a sus atributos. (**Ver Anexo 8)**

2.5.2.1. Representación de las clases persistentes

A continuación se representan las clases persistentes con sus atributos, tipos de datos y la descripción de las mismas.

- **Usuario:** Persona que va a interactuar con el sistema.
- **Controlador:** Dispositivo físico para regular procesos industriales.
- **Tipo de Proceso:** Clasificación en la que se agrupan los procesos.
- **Modelo de Proceso:** Conjunto de datos de un proceso definidos por el administrador, como son: modelo matemático, controlador y variables de entrada y salida.
- **Proceso:** Modelo de Proceso configurado para realizar la simulación.
- **Límite:** Permite al administrador modificar los datos de los límites en cuanto a la cantidad de usuarios conectados al sistema o a un proceso y la cantidad de procesos activos simultáneamente.

- **Variables:** Datos de entrada ó salida.
- **Protocolo:** Permitir la conexión con dispositivos físicos Ejemplo: TCP/IP
- **Estados del Proceso:** Activo o Inactivo.

Nombre: Dlimite	
Tipo de Clase: Entidad	
Atributo	**Tipo**
idlimite	int
cantidadusuarios	int
cantidadprocesos	int

Tabla 1 Descripción de la clase Dlimite

Nombre: Dusuario	
Tipo de clase: Entidad	
Atributo	**Tipo**
idusuario	int
nombreusuario	varchar
contrasenna	varchar
nombrepersona	varchar
primerapellido	varchar
rol	int

Tabla 2 Descripción de la clase Dusuario

Nombre: Dproceso	
Tipo de clase: Entidad	
Atributo	**Tipo**
idproceso	int
nombre	varchar

ecuacion	varchar
tiemposimulacion	double
cantidadve	int
cantidadvs	int
cantidadusuarios	int
cantidadreplicas	int
editor	varchar
descripcion	varchar
id_usuario_activo	int

Tabla 3 Descripción de la clase Dproceso

Nombre: Ntipoprotocolo	
Tipo de clase: Nomenclador	
Atributo	**Tipo**
idtipoprotocolo	int
nombre	varchar
descripcion	varchar

Tabla 4 Descripción de la clase Ntipoprotocolo

Nombre: Nprotocolo	
Tipo de clase: Nomenclador	
Atributo	**Tipo**
idprotocolo	int
nombre	varchar
cadena	varchar
puerto	int

Tabla 5 Descripción de la clase Nprotocolo

Nombre: Ncontrolador	
Tipo de clase: Nomenclador	
Atributo	**Tipo**
idcontrolador	int
nombre	varchar
direccionip	varchar

Tabla 6 Descripción de la clase Ncontrolador

Nombre: Ntipocontrolador	
Tipo de clase: Nomenclador	
Atributo	**Tipo**
idtipocontrolador	int
nombre	varchar
descripcion	varchar

Tabla 7 Descripción de la clase Ntipocontrolador

Nombre: Nestadoproceso	
Tipo de clase: Nomenclador	
Atributo	**Tipo**
idestado	int
nombre	varchar
descripcion	varchar

Tabla 8 Descripción de la clase Nestadoproceso

Nombre: Nmetodonumerico	
Tipo de clase: Nomenclador	
Atributo	**Tipo**
idmetodonumerico	int
nombre	varchar
descripcion	varchar

Tabla 9 Descripción de la clase Nmetodonumerico

Nombre: Ntipoproceso	
Tipo de clase: Nomenclador	
Atributo	**Tipo**
idtipoproceso	int
nombre	varchar
descripcion	varchar
cantidad	int

Tabla 10 Descripción de la clase Ntipoproceso

Nombre: Nvariable	
Tipo de clase: Nomenclador	
Atributo	**Tipo**
idvariable	int
idproceso	int
idtipovariable	int
descripcion	varchar

Tabla 11 Descripción de la clase Nvariable

Nombre: Ntipovariable	
Tipo de clase: Nomenclador	
Atributo	**Tipo**
idvariable	int
nombre	varchar
descripcion	varchar

Tabla 12 Descripción de la clase Ntipovariable

Nombre: Dusuario_Dproceso	
Tipo de clase: Entidad	
Atributo	**Tipo**
idusuario	int
idproceso	int

Tabla 13 Descripción de la clase Dusuario_Dproceso

Nombre: Ncontrolador_Nvariable	
Tipo de clase: Nomenclador	
Atributo	**Tipo**
idcontrolador	int
idvariable	int

Tabla 14 Descripción de la clase Ncontrolador_Nvariable

2.6. Clases para el acceso a la Base de Datos

El diagrama de clases de acceso a datos (DAO) es usado para modelar la estructura lógica de la capa de acceso a datos.

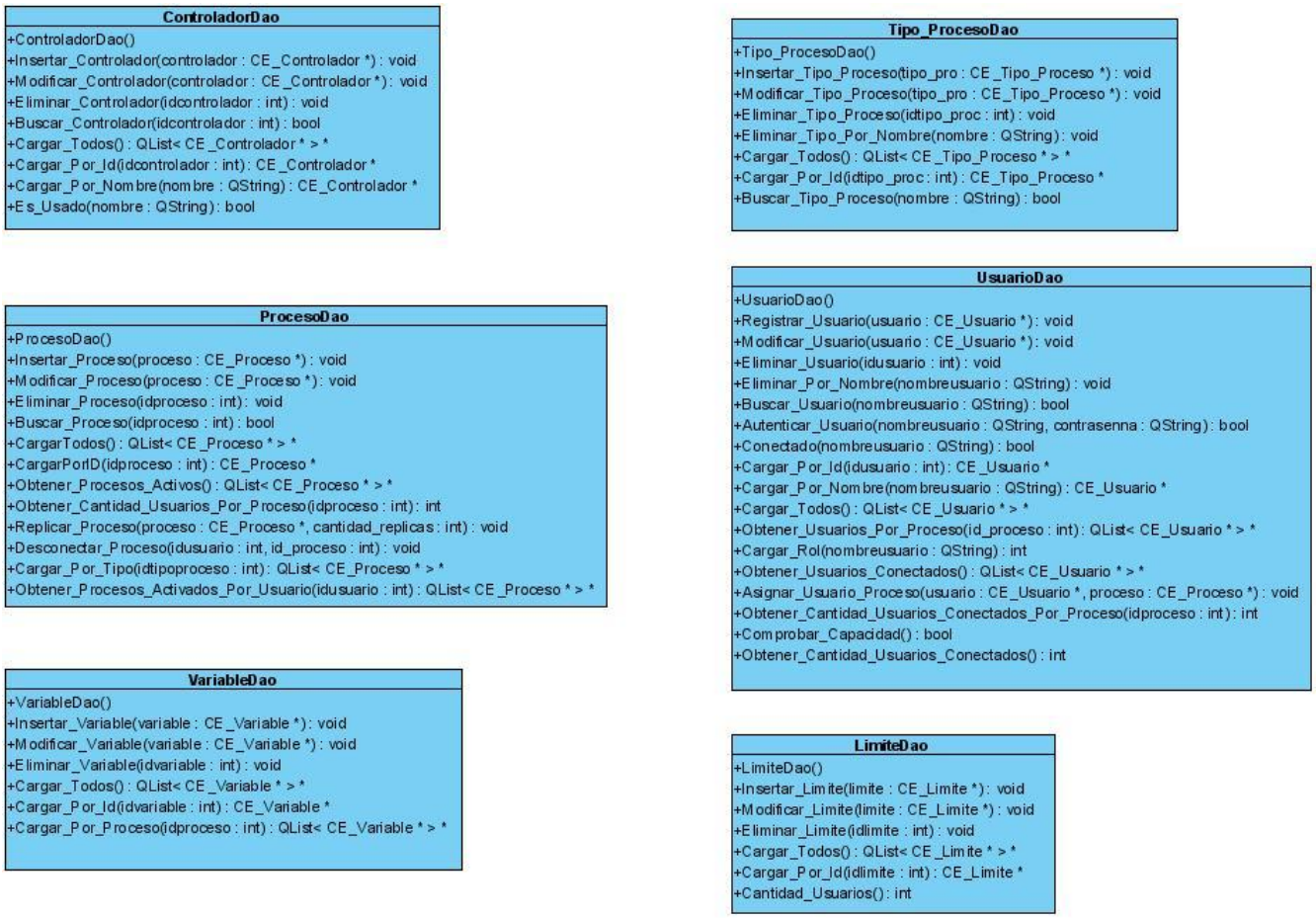


Figura 3 Diagrama de clases DAO

2.6.1. Descripción de las clases de acceso a datos

A continuación se representará la descripción de las clases DAO, haciendo referencia a las clases con sus respectivos métodos, con una breve descripción de lo que realiza.

Nombre: UsuarioDao	
Descripción: Clase que guarda los métodos de acceso a datos de los usuarios	
Métodos	**Descripción**
Registrar_Usuario()	Inserta los datos de un usuario en la aplicación.
Modificar_Usuario()	Modifica los datos de un usuario en la aplicación.
Eliminar_Usuario()	Elimina los datos de un usuario en la aplicación.
Buscar_Usuario()	Busca si existe un usuario en la aplicación.
Autenticar_Usuario()	Permite al usuario entrar al sistema.
Conectado()	Verifica si el usuario está conectado al sistema.
Cargar_Todos()	Devuelve una lista de todos los usuarios registrados.
Obtener_Usuarios_Por_Proceso()	Devuelve un lista de todos los usuarios conectados a un proceso
Obtener_Cantidad_Usuarios_Conectados_Por_Proceso()	Devuelve la cantidad de usuarios conectados a un proceso.
Comprobar_Capacidad()	Comprueba si existe capacidad para que un usuario se conecte a un proceso.
Obtener_Usuarios_Conectados()	Devuelve una lista de los usuarios autenticados en la aplicación.

Tabla 15 Descripción de la clase UsuarioDao

Nombre: ProcesoDao	
Descripción: Clase que guarda los métodos de acceso a datos de los procesos	
Métodos	**Descripción**
Insertar_Proceso()	Inserta los datos de un proceso en la aplicación.

Modificar_Proceso()	Modifica los datos de un proceso en la aplicación.
Eliminar_Proceso()	Elimina los datos de un proceso en la aplicación.
Buscar_Proceso()	Busca si existe un proceso en la aplicación.
Obtener_Procesos_Activos()	Devuelve una lista de todos los procesos activos en la aplicación.
Obtener_Cantidad_Usuarios_Por_Procesos()	Devuelve la cantidad de usuarios que pueden conectarse a un proceso.
Cargar_Todos()	Devuelve una lista de todos los procesos.
Replicar_Proceso()	Inserta una cantidad de procesos específica de un proceso ya existente.
Desconectar_Proceso()	Elimina los de usuarios conectados a un proceso y este pasa a inactivo.
Cargar_Por_Tipo()	Devuelve una lista de los procesos que pertenecen a un tipo de proceso.
Obtener_Procesos_Activados_Por_Usuario()	Devuelve una lista de los procesos que han sido activados por un usuario.

Tabla 16 Descripción de la clase ProcesoDao

Nombre: ControladorDao	
Descripción: Clase que guarda los métodos de acceso a datos de los controladores	
Métodos	**Descripción**
Insertar_Controlador()	Inserta los datos de un controlador en la aplicación.
Modificar_Controlador()	Modifica los datos de un controlador en la aplicación.
Eliminar_Controlador()	Elimina los datos de un controlador en la aplicación.
Buscar_Controlador()	Busca si existe un controlador en la aplicación.
Cargar_Todos()	Devuelve una lista de todos los controladores.

Tabla 17 Descripción de la clase ControladorDao

Nombre: Tipo_ProcesoDao	
Descripción: Clase que guarda los métodos de acceso a datos de los tipo de procesos	
Métodos	**Descripción**
Insertar_Tipo_Proceso()	Inserta los datos de un tipo de proceso en la aplicación.
Modificar_Tipo_Proceso()	Modifica los datos de un tipo de proceso en la aplicación.
Eliminar_ Tipo_Proceso ()	Elimina los datos de un tipo de proceso en la aplicación.
Buscar_ Tipo_Proceso ()	Busca si existe un tipo de proceso en la aplicación.
Cargar_Todos()	Devuelve una lista de todos los tipos de proceso.

Tabla 18 Descripción de la clase Tipo_ProcesoDao

Nombre: LimiteDao	
Descripción: Clase que guarda los métodos de acceso a datos de los límites del sistema	
Métodos	**Descripción**
Insertar_Limite()	Inserta los datos de un límite en la aplicación.
Modificar_Usuario()	Modifica los datos de un límite en la aplicación.
Eliminar_Usuario()	Elimina los datos de un límite en la aplicación.
Cargar_Todos()	Devuelve una lista de todos los límites.

Tabla 19 Descripción de la clase LimiteDao

Nombre: VariableDao	
Descripción: Clase que guarda los métodos de acceso a datos de las variables	
Métodos	**Descripción**
Insertar_Variable()	Inserta los datos de una variable en la aplicación.
Modificar_Variable()	Modifica los datos de una variable en la aplicación.
Eliminar_Variable()	Elimina los datos de una variable en la aplicación.
Cargar_Todos()	Devuelve una lista de todas las variables.

Tabla 20 Descripción de la clase VariableDao

2.7. Diseño de la BD

Las Bases de Datos necesitan de una definición de su estructura que le permita almacenar datos, reconocer el contenido, y recuperar la información. La estructura tiene que ser desarrollada para satisfacer las necesidades de las aplicaciones que la usarán.

La puesta en práctica de la Base de Datos es el paso final en el desarrollo de aplicaciones de soporte del negocio. Se conforman con los requisitos del proceso del negocio, que son la primera abstracción de la vista de la Base de Datos.

Para el diseño de la base de datos se realizó el diagrama entidad relación en el Visual Paradigm.

Una clase persistente es una clase entidad que tiene la capacidad de mantener su valor en el espacio y en el tiempo. Lo contrario son las clases temporales que son manejadas y almacenadas por el sistema en tiempo de ejecución por lo que dejan de existir cuando termina el programa.

En el diagrama entidad relación se especificaron los detalles físicos de la Base de Datos.

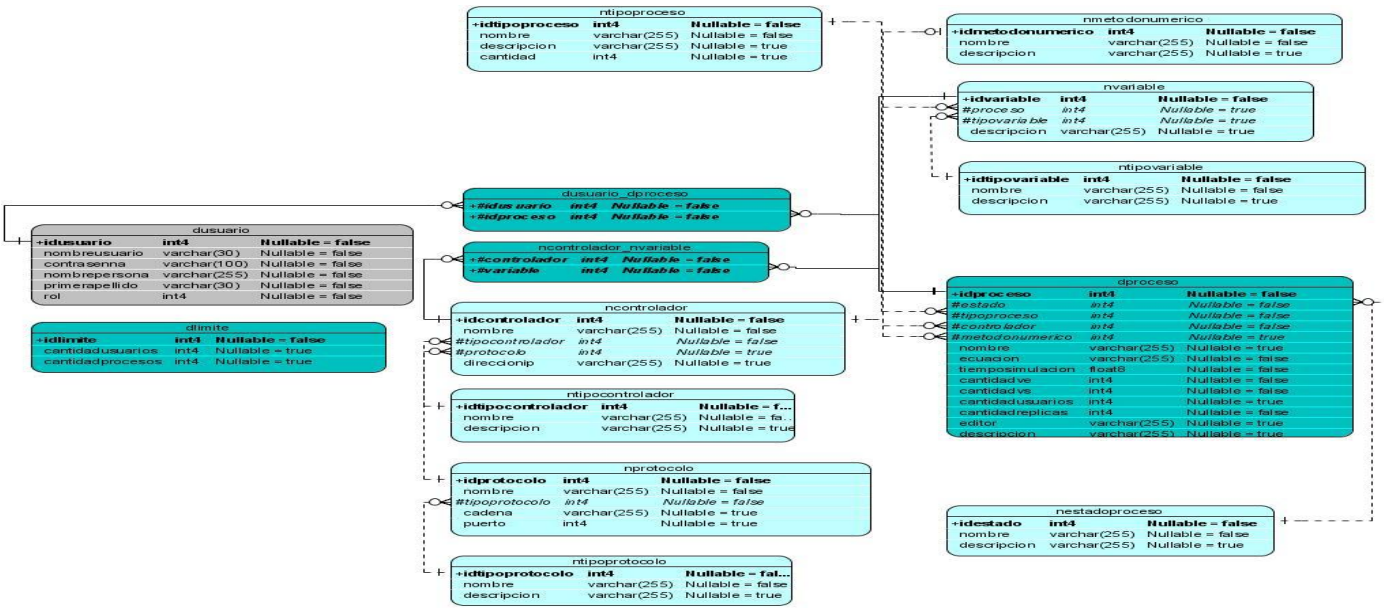


Figura 4 Modelo de datos

2.7.1. Descripción de las tablas

En esta sección se describen de forma general los datos que se almacenan en cada tabla del diagrama entidad relación de la (Base de Datos de una herramienta para la simulación de procesos usando nodos virtuales, Nodo Virtual de Procesos (NVP), al igual que se explica brevemente lo que representan todos sus atributos.

Nombre: Dlimite		
Descripción: Almacena los datos del límite		
Atributo	**Tipo**	**Descripción**
idlimite	int	Número identificador del límite
cantidadusuarios	int	Número de usuarios conectados
cantidadprocesos	int	Número de procesos activos

Tabla 21 Descripción de la tabla Dlimite

Nombre: Dusuario		
Descripción: Almacena los datos de los usuarios		
Atributo	**Tipo**	**Descripción**
idusuario	int	Número identificador del usuario
nombreusuario	varchar	Nombre del usuario
contrasenna	varchar	Contraseña del usuario
nombrepersona	varchar	Nombre de la persona que se registra
primerapellido	varchar	Primer apellido de la persona que se registra
rol	int	Número que identificará el rol que tendrá cada usuario en el sistema.

Tabla 22 Descripción de la tabla Dusuario

Nombre: Dproceso		
Descripción: Almacena los datos de los procesos		
Atributo	**Tipo**	**Descripción**
idproceso	int	Número identificador de un proceso
nombre	varchar	Nombre del proceso
ecuacion	varchar	Ecuación para un proceso
tiemposimulacion	double	Tiempo de simulación de un proceso
cantidadve	int	Cantidad de variables de entradas de un proceso
cantidadvs	int	Cantidad de variables de salida de un proceso
cantidadusuarios	int	Cantidad de usuarios para un proceso
cantidadreplicas	int	Cantidad de réplicas de un proceso
editor	varchar	Editor de un proceso
descripcion	varchar	Descripción de un proceso.
id_usuario_activo	int	Identificador del usuario que activó un proceso.

Tabla 23 Descripción de la tabla Dproceso

Nombre: Ntipoprotocolo		
Descripción: Tipo de protocolo de un controlador utilizado en un proceso		
Atributo	**Tipo**	**Descripción**
idtipoprotocolo	int	Número identificador del tipo de controlador
nombre	varchar	Nombre del tipo de protocolo
descripcion	varchar	Descripción del tipo de protocolo

Tabla 24 Descripción de la tabla Ntipoprotocolo

Nombre: Nprotocolo		
Descripción: Almacena los datos del protocolo		
Atributo	**Tipo**	**Descripción**
idprotocolo	int	Número identificador del protocolo
nombre	varchar	Nombre del protocolo
cadena	varchar	
puerto	int	Descripción del protocolo

Tabla 25 Descripción de la tabla Nprotocolo

Nombre: Ncontrolador		
Descripción: Almacena los datos del controlador		
Atributo	**Tipo**	**Descripción**
idcontrolador	int	Número identificador del controlador
nombre	varchar	Nombre del controlador
direccionip	varchar	Dirección ip del controlador

Tabla 26 Descripción de la tabla Ncontrolador

Nombre: Ntipocontrolador		
Descripción: Tipo de controlador utilizado por un proceso		
Atributo	**Tipo**	**Descripción**
idtipocontrolador	int	Número identificador del tipo de controlador
nombre	varchar	Nombre del tipo de controlador
descripcion	varchar	Descripción del tipo de controlador

Tabla 27 Descripción de la tabla Ntipocontrolador

Nombre: Nestadoproceso		
Descripción: Estado de un proceso en el sistema		
Atributo	**Tipo**	**Descripción**
idestado	int	Número identificador del estado del proceso
nombre	varchar	Nombre del estado del proceso
descripcion	varchar	Descripción del estado del proceso

Tabla 28 Descripción de la tabla Nestadoproceso

Nombre: Nmetodonumerico		
Descripción: Método numérico utilizado por un proceso		
Atributo	**Tipo**	**Descripción**
idmetodonumerico	int	Número identificador de un método numérico
nombre	varchar	Nombre de un método numérico
descripcion	varchar	Descripción de un método numérico

Tabla 29 Descripción de la tabla Nmetodonumerico

Nombre: Ntipoproceso		
Descripción: Tipos de procesos automáticos		
Atributo	**Tipo**	**Descripción**
idtipoproceso	int	Número identificador del tipo de proceso
nombre	varchar	Nombre del tipo de proceso
descripcion	varchar	Descripción del tipo de proceso
cantidad	int	Cantidad de tipos de procesos a priorizar

Tabla 30 Descripción de la tabla Ntipoproceso

Nombre: Dusuario_Dproceso		
Descripción: Tabla generada por la relación de mucho a mucho entre usuario y proceso		
Atributo	**Tipo**	**Descripción**
idusuario	int	Número identificador del usuario
idproceso	int	Número identificador de un proceso

Tabla 31 Descripción de la tabla Dusuario_Dproceso

Nombre: Ncontrolador_Nvariable		
Descripción: Tabla generada por la relación de mucho a mucho entre controlador y variable		
Atributo	**Tipo**	**Descripción**
idcontrolador	int	Número identificador del controlador
idvariable	int	Número identificador de la variable

Tabla 32 Descripción de la tabla Ncontrolador_Nvariable

Nombre: Ntipovariable		
Descripción: Tipo de variables de un proceso		
Atributo	**Tipo**	**Descripción**
idtipovariable	int	Número identificador del tipo de variable.
nombre	varchar	Nombre del tipo de variable.
descripcion	varchar	Descripción del tipo de variable.

Tabla 33 Descripción de la tabla Ntipovariable

Nombre: Nvariable		
Descripción: Variables de un proceso		
Atributo	**Tipo**	**Descripción**
idvariable	int	Número identificador de la variable.
idtipovariable	int	Número identificador del tipo de variable.
idproceso	int	Número identificador del proceso al que pertenece la variable.
descripcion	varchar	Descripción del tipo de variable.

Tabla 34 Descripción de la tabla Nvariable

2.8. Implementación de la Capa Acceso a Datos

En el desarrollo de la implementación de la capa de acceso a datos se tuvo presente la utilización de los diferentes componentes de prototipos .dll o librerías de enlace dinámico; las cuales son: NVP.Clases.dll, DAO.dll y QSQLDatabse.dll. En NVP.Clases, se recogen las clases persistentes; para acceder a los datos se hace necesaria la utilización de la librería DAO, que está compuesta por las clases que proporcionan a la capa de negocio el acceso a la base de datos y una clase fachada que brinda los servicios que utiliza la capa del negocio. Este componente utiliza además la librería QSQLDatabase, la cual se encarga del acceso físico a la base de datos.

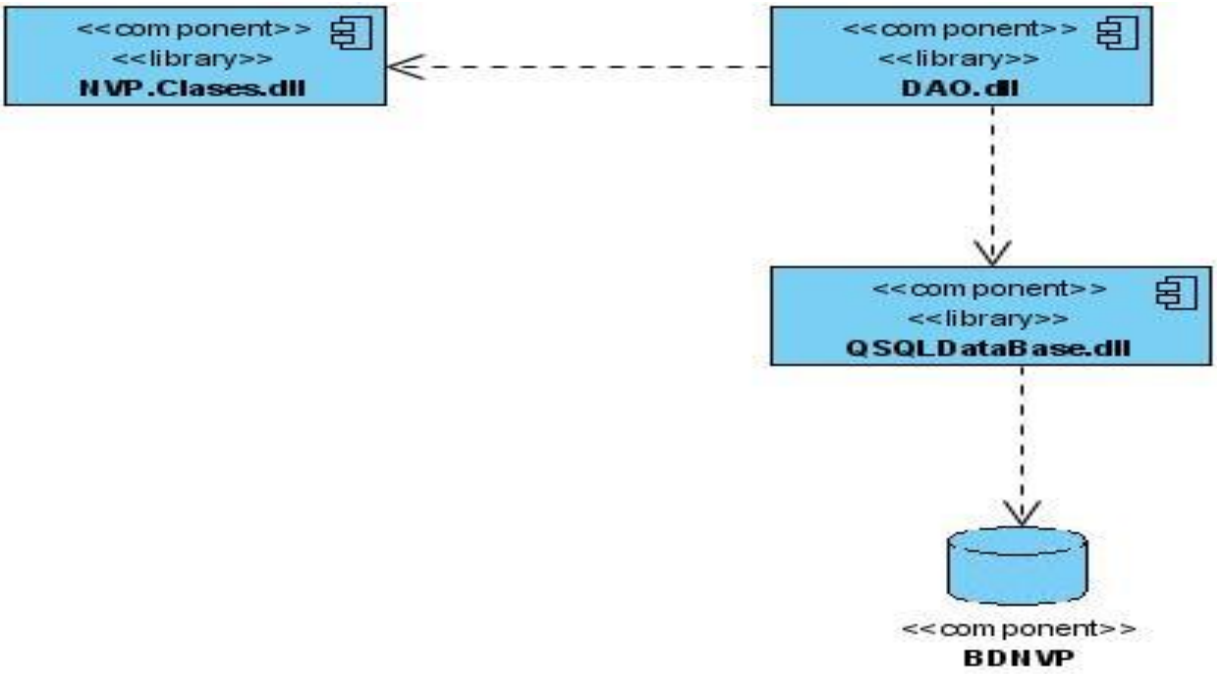


Figura 5 Diagrama de componentes de la Capa de Acceso a Datos

Conclusiones parciales

En este capítulo fueron descritos los requisitos funcionales y no funcionales. Se crearon los diagramas de clases del diseño, de clases persistentes, de acceso a datos y el modelo entidad-relación. Se realizó la descripción de cada entidad o clase necesaria para dichos diagramas, logrando obtener finalmente la construcción de la base de datos propuesta, a través del gestor de base de datos seleccionado para ello, en este caso PostgreSQL.

Capítulo 3: Validación de los Resultados

Introducción

En el presente capítulo se hará referencia a la validación del diseño de clases y de la base de datos, mediante la aplicación de métricas y teniendo en cuenta la descripción de la integridad relacional de datos, el análisis de la redundancia de la información que la conforma, además de hacer alusión a las políticas de seguridad aplicadas. Se realizarán pruebas de unidad que permitirán validar la implementación propuesta.

3.1. Métrica para evaluar el diseño. Tamaño de clase (TC)

La métrica de tamaño de clase pertenece a la familia de métricas propuesta por Lorenz y Kidd. Ellos proponen que para medir el tamaño de clases se deben tener en cuenta los siguientes aspectos.

Total de operaciones, ya sean las de la clase o las heredadas de las clases padres o interfaces que implementen.

- Cantidad de atributos, al igual que el anterior, tanto los de ella, como los de los padres.
- Promedio general de los dos anteriores para el sistema completo.

Un TC grande afecta los indicadores de calidad que fueron definidos para esta métrica por los especialistas:

- **Reutilización:** reduce la reutilización de la clase.
- **Implementación:** complica la implementación.
- **Responsabilidad:** la clase debe tener bastante responsabilidad.

Clases	Nº. de atributos	Nº. de operaciones
1- CE_Controlador	5	10
2- CE_Estado_Proceso	3	6
3- CE_Limite	3	6
4- CE_Metodo_Numerico	3	6
5- CE_Proceso	15	30
6- CE_Protocolo	5	10
7- CE_Tipo_Controlador	3	6
8- CE_Tipo_Proceso	4	8
9- CE_Tipo_Protocolo	3	6
10- CE_Tipo_Variable	3	6
11- CE_Usuario	7	14
12- CE_Variable	4	8
13- ControladorDao	0	14
14- LimiteDao	0	6
15- ProcesoDao	0	12
16- Tipo_ProcesoDao	0	7

17- Usuario_ProcesoDao	**0**	**1**
18- UsuarioDao	**0**	**17**
19- VariableDao	**0**	**6**

Tabla 35 Clases con No de atributos y operaciones

De esta forma el umbral queda con los datos mostrados a continuación:

Umbral	Tamaño	Cantidad de Clases
<=20	**Pequeño**	**18**
>20<=30	**Medio**	**1**
>30	**Grande**	**0**

Tabla 36 Umbral

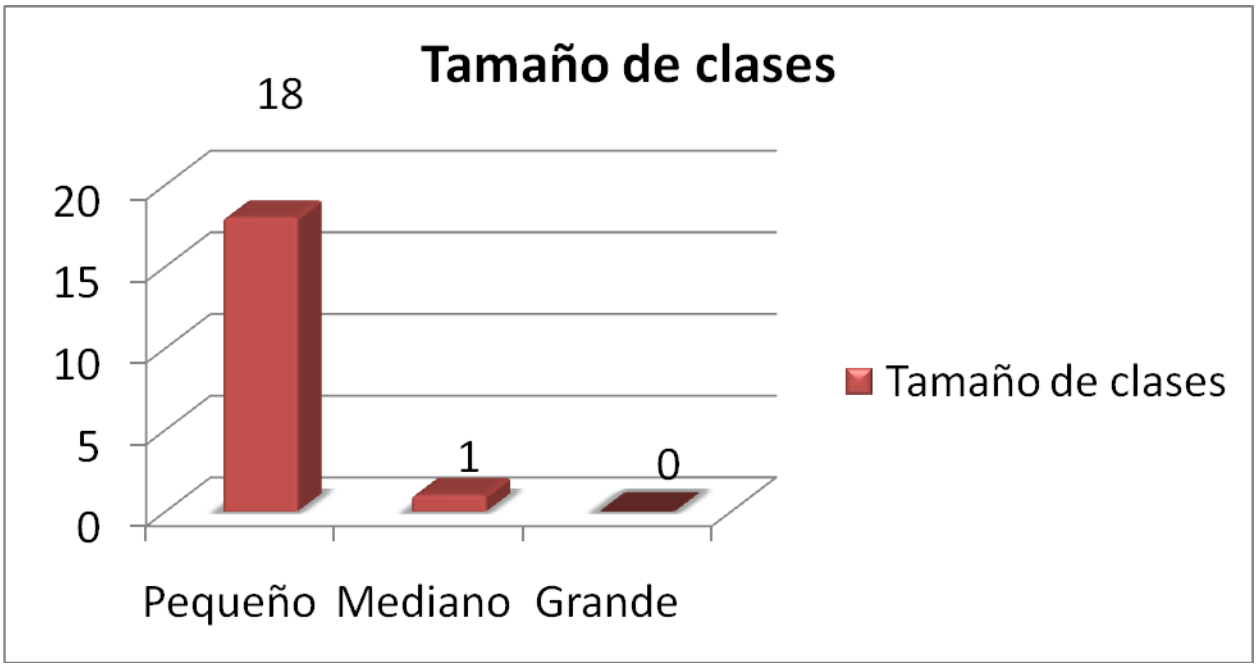


Figura 6 Número de clases por categorías.

Resultados:

Se aplicó la métrica tamaño de clases por Lorenz y Kidd, la cual fue realizada a la capa de acceso a datos, conformada por 18 clases de tamaño pequeño y una clase de tamaño mediano, este valor demuestra que los indicadores de reutilización, implementación y responsabilidad no se van a ver afectados.

3.2. Validación teórica del diseño

Un buen diseño en una base de datos se logra sobre todo cuando se tienen en cuenta algunos aspectos de gran relevancia tales como: la integridad de los datos, la redundancia de información y sobre todo la seguridad, la cual es base principal de los aspectos anteriormente mencionados.

3.2.1. Integridad

La integridad de una BD se refiere a la consistencia de los datos almacenados de acuerdo a un conjunto de restricciones semánticas.

Integridad de Entidad: Establece que la llave primaria de una tabla debe tener un valor único para cada fila de la tabla, sino la base de datos perderá su integridad. Para lograr satisfacer este parámetro se hace uso de secuencias, las cuales garantizan que en cada inserción en una tabla de la base de datos, la llave del registro insertado sea única.

Datos Requeridos: Establece que al realizar una operación de INSERT sobre una fila, una columna marcada como NOT NULL no debe recibir valores nulos.

Chequeo de Validez: Al crear una nueva tabla cada columna se describe con un tipo de datos y el SGBD garantiza que sólo los datos del tipo especificado sean ingresados en la tabla.

Integridad Referencial: Garantiza que una entidad (fila o registro) siempre se relaciona con otras entidades válidas, es decir, que existen en la base de datos. Implica que en todo momento dichos datos sean correctos, sin repeticiones innecesarias, datos perdidos y relaciones mal resueltas. Todas las bases de datos relacionales gozan de esta propiedad gracias a que el software gestor de base de datos vela por su cumplimiento.

3.2.2. Análisis de la redundancia de la información

El diseño de una BD debe ser realizado cuidadosamente, procurando permitir un fácil acceso a la información, facilitando un alto rendimiento, una buena velocidad de respuesta y una gran consistencia de los datos. Al diseñar una BD se debe tener en cuenta que la información almacenada ocupará irremediablemente un espacio en memoria; es de vital importancia eliminar la posibilidad de almacenar datos repetidos que podrían conducir a que exista inconsistencia en la información. A este almacenamiento de información repetida se le conoce como redundancia de la información. (Avendaño, 2009)

Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula. En algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias que en ocasiones resulta mejor para ganar en simplicidad de las consultas y lograr un mejor tiempo de respuesta.

3.2.3. Análisis de seguridad de la BD

Las BD están bajo constante amenaza de sufrir ataques de personas o sistemas no autorizados que ponen en riesgo su integridad y confidencialidad; por eso deben tomarse una serie de medidas que impidan estos sucesos y permitan conservar salvas de la BD para restaurar los datos si se pierden o corrompen por alguna razón.

Los SGBD permiten definir autorizaciones o derechos de acceso teniendo en cuenta los usuarios, ubicaciones desde donde se puede acceder, así como asignar privilegios que tendrán los usuarios una vez autentificados. Lo primero que debe garantizarse es que sólo puedan acceder a los datos los usuarios autorizados. PostgreSQL, permite configurar los permisos de los usuarios utilizando tres niveles de acceso. En el primer nivel se configuran los permisos de conexión para los host y los usuarios a la o las BD en el archivo pg_hba.conf, se define qué dirección o direcciones IP tendrán acceso a cuál o cuáles BD, y en qué modo podrán conectarse: conexión sin contraseña, validando el usuario y la contraseña o que rechace cualquier conexión desde el IP o rangos IP y usuarios seleccionados. En el segundo nivel, se configuran a qué BD pueden acceder determinados usuarios, utilizando las opciones del archivo

pg_ident.conf. En el tercer nivel, permite configurar los accesos de los usuarios a las tablas de la BD, utilizando las sentencias GRANT y REVOKE se pueden asignar o denegar respectivamente los permisos para insertar, eliminar y actualizar datos, entre otros. En la BD propuesta se establecen controles de seguridad para los datos almacenados, garantizando que sólo los usuarios autorizados puedan efectuar operaciones correctas sobre algunas tablas o sobre la BD. Para evitar la pérdida de los datos en caso de ocurrir alguna falla, PostgreSQL permite realizar salvas de la BD mediante el volcado (dump), pueden realizarse de forma automática o manualmente.

Para el volcado de la BD se utiliza el comando pg_dump, que extrae una base de datos PostgreSQL en un archivo de comandos o en otro tipo de archivo. El mismo cuenta con una serie de parámetros adicionales para indicar el nombre de la BD, el usuario y contraseña para conectarse a ella, el nombre que tomará el archivo de salva y donde se desea guardarla. El volcado de la BD puede realizarse con el servicio PostgreSQL corriendo, se puede realizar a una o varias BD y mantiene compatibilidad entre versiones; este proceso puede ser un poco lento cuando se hacen salvas a BD de gran volumen. Para restaurar las salvas, se utiliza el comando pg_restore indicando la salva que deseamos restaurar entre otros parámetros.

3.3. Prueba de unidad en C++

La prueba de unidad es uno de los métodos con el que se puede mejorar la calidad de los sistemas de software.
En un sistema de calidad de software (o Quality Assurance), existen principalmente dos tipos de pruebas:

- **Pruebas unitarias (o de aceptación)**: comprobaciones que se aplican a las unidades lógicas de la aplicación. Se verifica que una unidad funciona correctamente por sí misma, sin tener en cuenta las relaciones que pueda tener con otras partes del sistema.
- **Pruebas funcionales (o pruebas de sistema o integración)**: se comprueba el sistema globalmente, haciendo énfasis en las colaboraciones entre unidades. Se prueba cada una de las opciones (o casos de uso) que ofrece el sistema, pudiendo ser procesos automáticos, acciones sobre el interfaz gráfico, etc.

Debido a que uno de los pasos más importantes que debe tener en cuenta el rol de programador es evaluar o probar los componentes resultantes de implementar los elementos de diseño, se llevaron a cabo pruebas de unidad utilizando el framework CppUnit. CppUnit es un estándar para desarrollar pruebas en C++. Las pruebas de unidad utilizando CppUnit se realizaron a través de casos de prueba que incluyeron una o varias pruebas de unidad, logrando así examinar el comportamiento individual de cada una de las unidades de trabajo.

La siguiente tabla muestra los aspectos que se tuvieron en cuenta para seleccionar y desarrollar las pruebas de unidad, a determinadas unidades de trabajo utilizando el framework CppUnit.

Casos de prueba

Caso de Prueba	**Probar alternativas**			
Unidad de Trabajo	**TestRegistrar_Usuario**			
Objetivo	**Valores de Entrada**	**Resultado Esperado**	**Resultado Real**	**Tipo de Resultado**
Insertar el objeto de la clase CE_Usuario en la tabla correspondiente en la base de datos.	Objeto de la clase CE_Usuario.	Se espera que el objeto de la clase CE_Usuario sea insertado satisfactoriamente en la tabla correspondiente.	Se insertó el objeto de la clase CE_Usuario satisfactoriamente. Se insertaron correctamente los valores definidos en las propiedades del objeto en las columnas de la tabla correspondiente en la base de datos.	Correcto
Unidad de Trabajo	**TestModificar_Usuario**			

Objetivo	Valores de Entrada	Resultado Esperado	Resultado Real	Tipo de Resultado
Modificar el objeto de la clase CE_Usuario en la tabla correspondiente en la base de datos.	Objeto de la clase CE_Usuario.	Se espera que el objeto de la clase CE_Usuario sea modificado satisfactoriamente en la tabla correspondiente.	Se modificó el objeto de la clase CE_Usuario satisfactoriamente. Se modificó la tupla especificada por el objeto en las columnas de la tabla correspondiente en la base de datos.	Correcto
Unidad de Trabajo	**TestEliminar_Usuario**			
Objetivo	**Valores de Entrada**	**Resultado Esperado**	**Resultado Real**	**Tipo de Resultado**
Eliminar el objeto de la clase CE_Usuario en la tabla correspondiente en la base de datos.	Objeto de la clase CE_Usuario.	Se espera que el objeto de la clase CE_Usuario sea eliminado satisfactoriamente en la tabla correspondiente.	Se eliminó el objeto de la clase CE_Usuario satisfactoriamente. Se eliminó la tupla especificada por el objeto en las columnas de la tabla correspondiente en la base de datos.	Correcto
Unidad de Trabajo	**TestInsertar_Controlador**			
Objetivo	**Valores de**	**Resultado**	**Resultado Real**	**Tipo de**

	Entrada	Esperado		Resultado
Insertar el objeto de la clase CE_Controlador en la tabla correspondiente en la base de datos.	Objeto de la clase CE_Controlador.	Se espera que el objeto de la clase CE_Controlador sea insertado satisfactoriamente en la tabla correspondiente.	Se insertó el objeto de la clase CE_Controlador satisfactoriamente. Se insertaron correctamente los valores definidos en las propiedades del objeto en las columnas de la tabla correspondiente en la base de datos.	Correcto
Unidad de Trabajo	**TestModificar_ Controlador**			
Objetivo	**Valores de Entrada**	**Resultado Esperado**	**Resultado Real**	**Tipo de Resultado**
Modificar el objeto de la clase CE_Controlador en la tabla correspondiente en la base de datos.	Objeto de la clase CE_Controlador.	Se espera que el objeto de la clase CE_Controlador sea modificado satisfactoriamente en la tabla correspondiente.	Se modificó el objeto de la clase CE_Controlador satisfactoriamente. Se modificó la tupla especificada por el objeto en las columnas de la tabla correspondiente en la base de datos.	Correcto
Unidad de Trabajo	**TestEliminar_Controlador**			

Objetivo	Valores de Entrada	Resultado Esperado	Resultado Real	Tipo de Resultado
Eliminar el objeto de la clase CE_Controlador en la tabla correspondiente en la base de datos.	Objeto de la clase CE_Controlador.	Se espera que el objeto de la clase CE_Controlador sea eliminado satisfactoriamente en la tabla correspondiente.	Se eliminó el objeto de la clase CE_Controlador satisfactoriamente. Se eliminó la tupla especificada por el objeto en las columnas de la tabla correspondiente en la base de datos.	Correcto
Unidad de Trabajo	**TestInsertar_Proceso**			
Objetivo	**Valores de Entrada**	**Resultado Esperado**	**Resultado Real**	**Tipo de Resultado**
Insertar el objeto de la clase CE_Proceso en la tabla correspondiente en la base de datos.	Objeto de la clase CE_Proceso.	Se espera que el objeto de la clase CE_Proceso sea insertado satisfactoriamente en la tabla correspondiente.	Se insertó el objeto de la clase CE_Proceso satisfactoriamente. Se insertaron correctamente los valores definidos en las propiedades del objeto en las columnas de la tabla correspondiente en la base de datos.	Correcto

Unidad de Trabajo	**TestModificar_Proceso**			
Objetivo	**Valores de Entrada**	**Resultado Esperado**	**Resultado Real**	**Tipo de Resultado**
Modificar el objeto de la clase CE_Proceso en la tabla correspondiente en la base de datos.	Objeto de la clase CE_Proceso.	Se espera que el objeto de la clase CE_Proceso sea modificado satisfactoriamente en la tabla correspondiente.	Se modificó el objeto de la clase CE_Proceso satisfactoriamente. Se modificó la tupla especificada por el objeto en las columnas de la tabla correspondiente en la base de datos.	Correcto
Unidad de Trabajo	**TestEliminar_Proceso**			
Objetivo	**Valores de Entrada**	**Resultado Esperado**	**Resultado Real**	**Tipo de Resultado**
Eliminar el objeto de la clase CE_Proceso en la tabla correspondiente en la base de datos.	Objeto de la clase CE_Proceso.	Se espera que el objeto de la clase CE_Proceso sea eliminado satisfactoriamente en la tabla correspondiente.	Se eliminó el objeto de la clase CE_Proceso satisfactoriamente. Se eliminó la tupla especificada por el objeto en las columnas de la tabla correspondiente en la base de datos.	Correcto

Tabla 37 Aspectos que se tuvieron en cuenta para seleccionar y desarrollar las pruebas de unidad.

Pruebas realizadas mediante el framework CppUnit

Para implementar los casos de pruebas, el framework CppUnit ofrece la clase QTest, que es heredada por las clases que van a ser probadas, conteniendo los métodos que permiten probar las unidades de trabajo.

A continuación se muestra la implementación del caso de prueba UsuarioDaoTest, incluyendo las unidades de trabajo: testRegistrar_Usuario, testModificar_Usuario, testEliminar_Usuario:

```
void Test::testRegistrar_Usuario()

{

    UsuarioDao* usuario=new UsuarioDao();

    CE_Usuario* nuevo=new CE_Usuario("jatopiz","uci","jose","topiz",1);

    bool resultado = usuario->Registrar_Usuario(nuevo);

    QCOMPARE(resultado, true);

    resultado = usuario->Registrar_Usuario(nuevo);

    QCOMPARE(resultado, false);

}

void Test::testModificar_Usuario()

{

    UsuarioDao* usuario=new UsuarioDao();

    CE_Usuario* nuevo=new CE_Usuario("yrsanchez","feu","yunier","ricardo",3);
```

```
    nuevo->Set_Estado(2);

    bool resultado = usuario->Modificar_Usuario(nuevo);

    QCOMPARE(resultado, true);

}

void Test::testEliminar_Usuario()

{

    UsuarioDao* usuario=new UsuarioDao();

    bool resultado = usuario->Eliminar_Usuario(1);

    QCOMPARE(resultado, true);

}
```

Para verificar los resultados de la prueba ejecutada, se muestran los reportes generados por el framework CppUnit. (**Ver Anexo 9, 10 y 11**)

Conclusiones parciales

En este capítulo se han analizado aspectos a tener en cuenta para realizar un buen diseño de clases y de BD, así como una correcta implementación.

- Aplicación de la métrica del tamaño de clases para validar el diseño propuesto dando como resultado que el mismo se realizó con calidad, permitiendo la reutilización de estas clases.
- Se realizó el análisis de la seguridad de la BD, mediante un control del acceso a los datos y limitando los permisos de los usuarios, logrando así una BD segura y confiable.
- Se utilizó el framework CppUnit para la realización de las pruebas de unidad, permitiendo testear el código de la aplicación brindando resultados satisfactorios.

Conclusiones

- El diseño e implementación de la base de datos de una herramienta interactiva para la simulación de procesos se obtuvo como resultado la posibilidad de gestionar la información de forma eficiente, eliminando la posibilidad que se trabaje con información no válida, que no exista información redundante en la base de datos y que la información esté organizada para que facilite el trabajo con los datos que se manejan.
- El diseño de la capa de acceso a datos del NVP se logró la implementación del acceso de dicha capa de la aplicación a la base de datos para brindar las funcionalidades a los usuarios, eliminando la posibilidad de que exista pérdida de información, o se realicen cambios no deseados.
- La terminación de este trabajo permite plantear que se le ha dado cumplimiento a todos los objetivos del mismo.

Recomendaciones

Con la realización del presente trabajo se recomienda:

- Mantener sobre la base de datos un estricto cumplimiento del proceso de mantenimiento y copias de seguridad periódicas, logrando así que se mantenga la fiabilidad y funcionamiento óptimo de la base de datos.
- La utilización del diseño y la implementación de la capa de acceso a datos para futuras versiones del NVP.
- Desarrollar para futuras versiones la conexión de la aplicación con el SGBD PostgreSQL en el sistema operativo Windows.
- Otra recomendación válida es que una vez obtenida esta aplicación se debe poner a disposición de todos los centros de nivel superior del país donde se estudie la carrera Automática.

Bibliografía

Alarcón, Jose Manuel. 2006. Administración de SGBD PostgreSQL. 2006.

Alburquerque, Amado. 2007. Sistema Integrador de Gestión Estadística(SIGE). 2007.

Anónimo. 2006. Sistemas de Información. 2006.

Avendaño, D. E. P. 2009. Ciencias de la Computación. [En línea] 2009. [Citado el: 28 de enero de 2009.] http://www.cs.buap.mx/~dpinto/bd.

Gámez Batista, Ing. Yalice. 2008. Herramienta interactiva para la enseñanza en la carrera de Ingeniería Automática: Nodo Virtual de Procesos. 2008.

Castelán, Dr. Mario. 2006. Métodos cuantitativos y simulación.Introducción al modelado y simulación de sistemas. 2006.

Ortiz Azahares, Leydis A. y Escobar Pompa, Maylen Edith 2008. Análisis y Diseño de un Nodo Virtual de Procesos. 2008.

Expressions, Never Ordinay. 2009. Never Ordinay Expressions. [En línea] 2009. [Citado el: 26 de abril de 2009.] http://www.raul-art.blogspot.com.

Fábregas, Yuniesky y Fernández, Daniel. 2007. Administración, configuración y optimización de un Sistema de Base de Datos Descentralizado en Oracle Database 10g release 2. 2007.

Fernández de la Pera, Elian Luis y López Naranjo, Danis. 2007. Desarrollo de la capa de acceso a datos para los modulos de administracion y nomencladores del sistema de gestion de inventario de almacenes SIGIA. 2007.

Geetanjali A., Balasubramaniam A. 2007. Programación Microsoft C#. La Habana : Félix Varela, 2007.

Geocities. 2009. Geocities. [En línea] 2009. [Citado el: 12 de marzo de 2009.] http://es.geocities.com/johanmontanez/e3/Herramientas.html.

GSInnova. 2009. GSInnova. [En línea] 2009. [Citado el: 25 de enero de 2009.] http://www.rational.com.ar/herramientas/rosedatamodeler.html.

Hosseinzaman, A. y Bargiela, A. 1995. ADA's Virtual Node based Water System. 1995.

IEEE. 1993. Standards Collection: Software Engineering. s.l. : IEEE Standard 610.12-1990. 1993.

Jacobson, I, Booch, G y Rumbaugh, J. 2004. El Proceso Unificado de Desarrollo de Software. La Habana : Félix Varela, 2004.

javaHispano. 2009. javaHispano. [En línea] 2009. [Citado el: 26 de febrero de 2009.] http://www.javahispano.org/contenidos/es/el_archipielago_eclipse__2_parte11.

Jiménez, Walter Ramiro Toro. 2008. Modelo de simulación prospectiva de la demanda de servicios de salud para enfermedades de alto costo: aplicación para una entidad promotora de salud Colombiana.2008.

Kuroki C. 2005. PostgreSQL 8.Migración a PostgreSQL desde otras bases de datos. 2005.

LinuxPartyGroup. 2008. LinuxPartyGroup. [En línea] 2008. [Citado el: 17 de abril de 2009.] http://www.linux-party.com.

López Barrio, C. 2005. Metodología de Desarrollo: Programación Extrema. 2005.

Lucas Rodríguez F., Cabello Quintero A. 2005. Simulación de procesos de información y criptografía cuántica. 2005.

Maier, S y Herrscher, K. 2003. On node virtualization for scalable Network Emulation. 2003.

Manager, Free Download. 2009. Free Download Manager. [En línea] 2009. [Citado el: 3 de abril de 2009.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

Márquez, María Mercedes. 2001. Historia de los Sistemas de Base de Datos. 2001.

Masadelante. 2006. [En línea] 2006. [Citado el: 21 de febrero de 2009.] http://www.masadelante.com.

Miyachi, Toshiyuki y Chinen, Ken-ichi. 2006. StarBED and SprinOS: Large scale general purpose network testbed and supporting software. 2006.

Molino, Neil, Bao, Z. y R. Fedkiw, A. 2004. Virtual Node Algorithm for Changing Mesh Topology During Simulation. 2004.

Moraga, M. Angeles. 2001. El Modelo de datos Jerárquico. Escuela Superior de Informática. 2001.

NetBeans. 2009. NetBeans. [En línea] 2009. [Citado el: 13 de marzo de 2009.] http://www.netbeans.org/index_es.html.

Pabón, M. A. G. C. W. R. 2005. Comparación entre Sistemas de Gestión de Bases de Datos (SGBD). 2005.

pgAdmin. 2009. pgAdmin. [En línea] 2009. [Citado el: 15 de enero de 2009.] http://www.pgadmin.org.

Pressman, R. 2005. Ingeniería del Software. Un enfoque práctico. La Habana : Félix Varela, 2005.

Shannon, Robert E. 1992. Introduction to Simulation. 1992.

Woody, B. 2006. Electronic Source: Administrator's Guide to SQL Server 2005. Addison Wesley Professional. 2006.

Wu, Y. 1999. A virtual node method for treatment of wells in modeling multiphase Flow in reservoirs. . 1999.

García de la Puente, Sergio Jesús y Plasencia Crespo, Mileisys. “Sistema para la descarga y procesamiento automatizado de patentes: Predictor. Rol de arquitecto”. 2007

Larman, Craig. 1999. UML Y PATRONES. Introducción al análisis y diseño orientado a objetos. Prentice Hall Hispanoamericana, 1999.

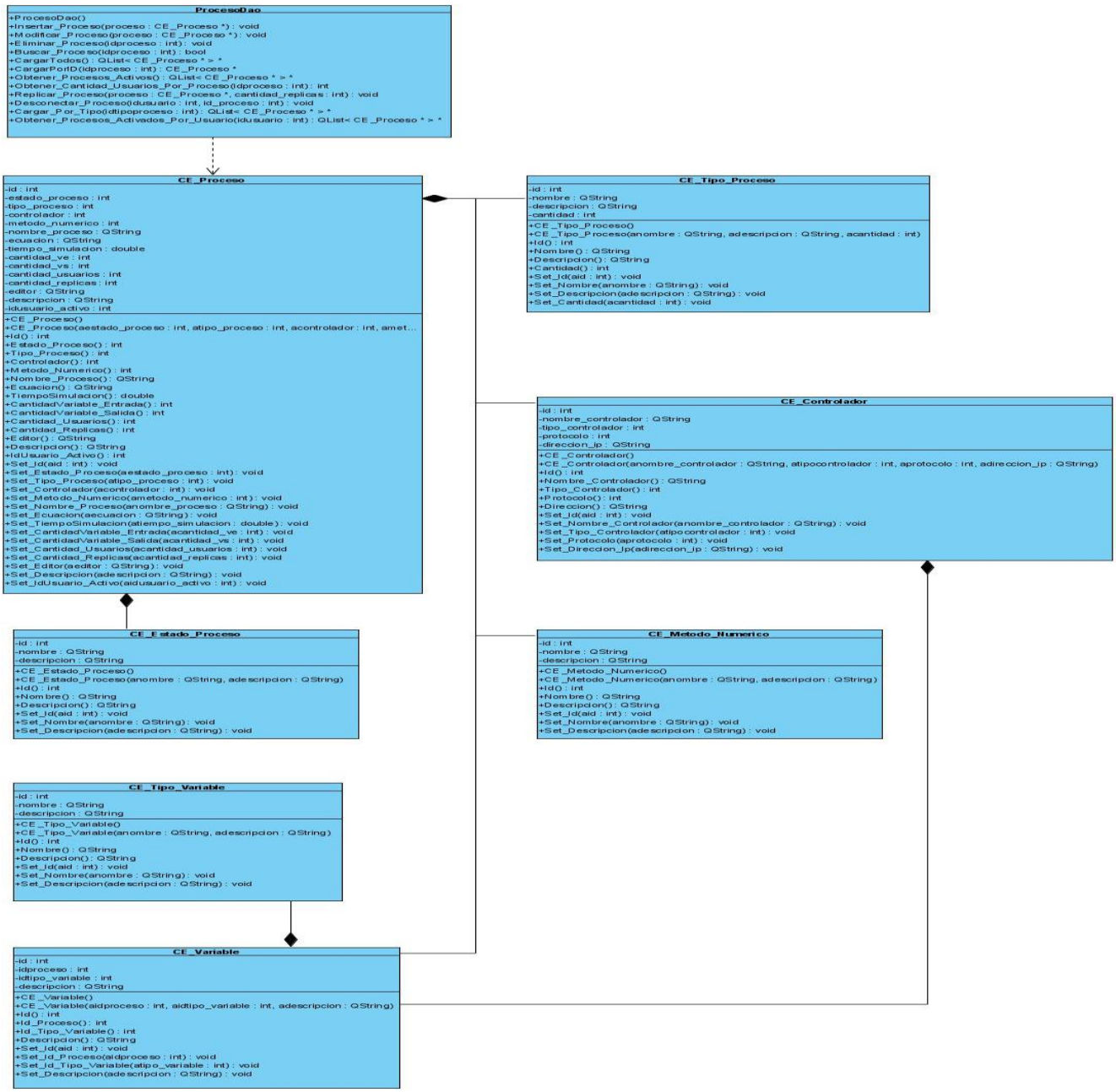
Perera Morales, José Raúl. “Arquitectura de software para un sistema Gestión de Inventarios”. 2007.

Lago, Ramiro. “Patrón: Data Access Object”. 2007. Disponible en: http://www.proactiva-calidad.com/java/**patrones**/DAO.html

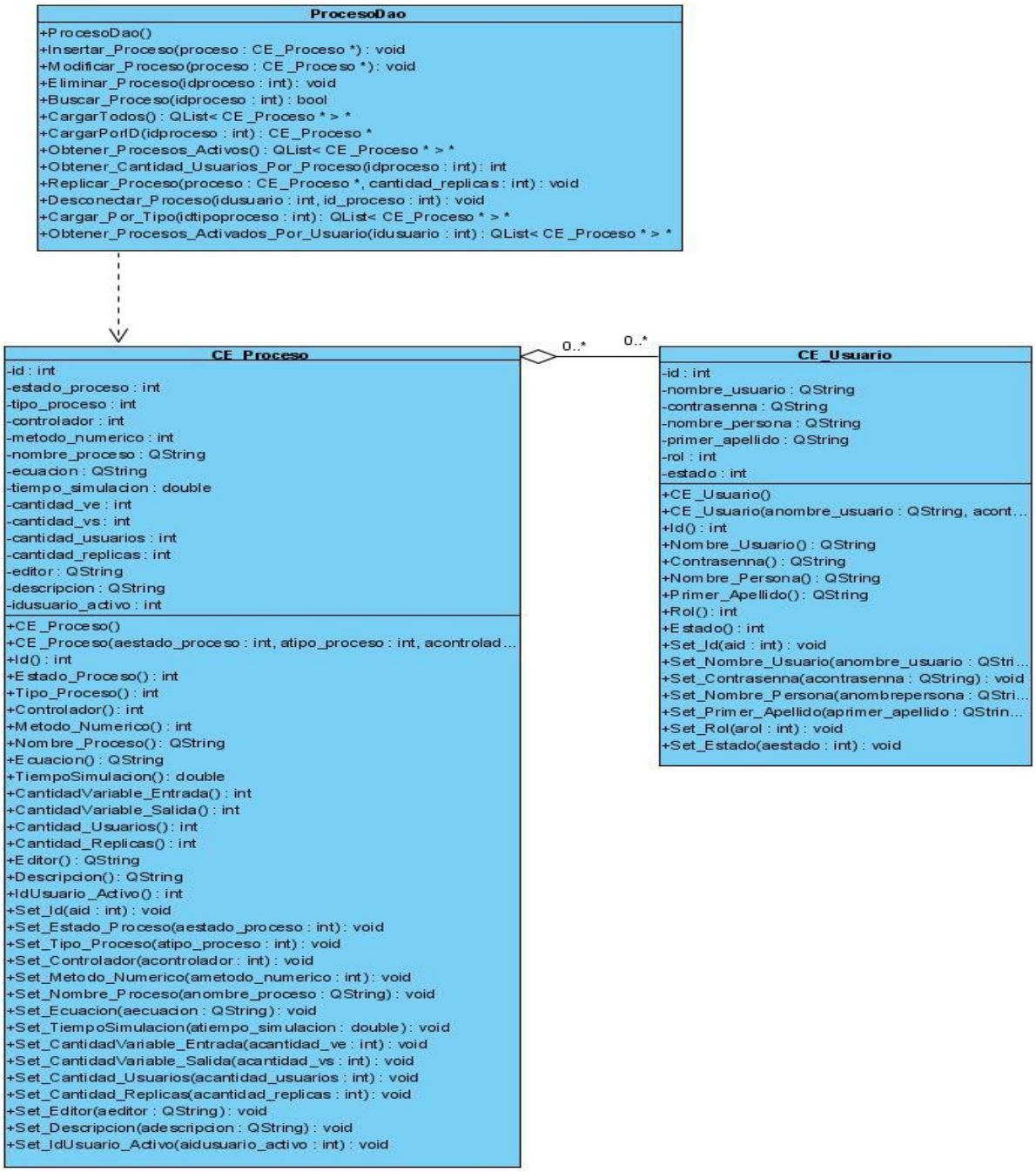
Cleger Despaigne, Eliober y Tornés Montes de Oca, Annarella María. “Análisis y Diseño de una herramienta interactiva de simulación de procesos: Nodo Virtual de Procesos. Segunda Iteración.” 2009.

Anexos

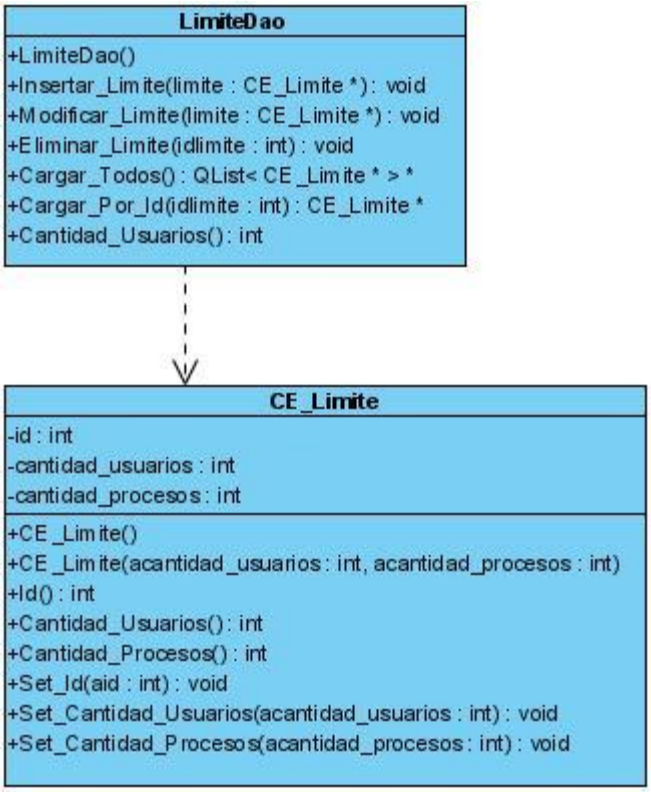
Anexo 1: Diagrama de clases de Diseño. **Configurar Proceso**



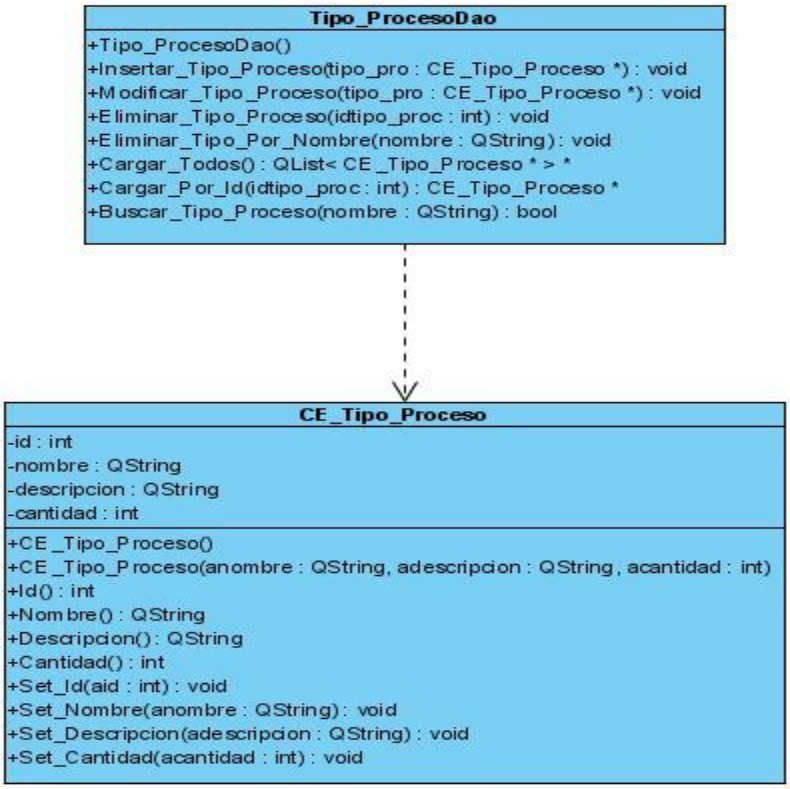
Anexo 2: Diagrama de clases de Diseño. **Desconectar Usuario-Proceso**



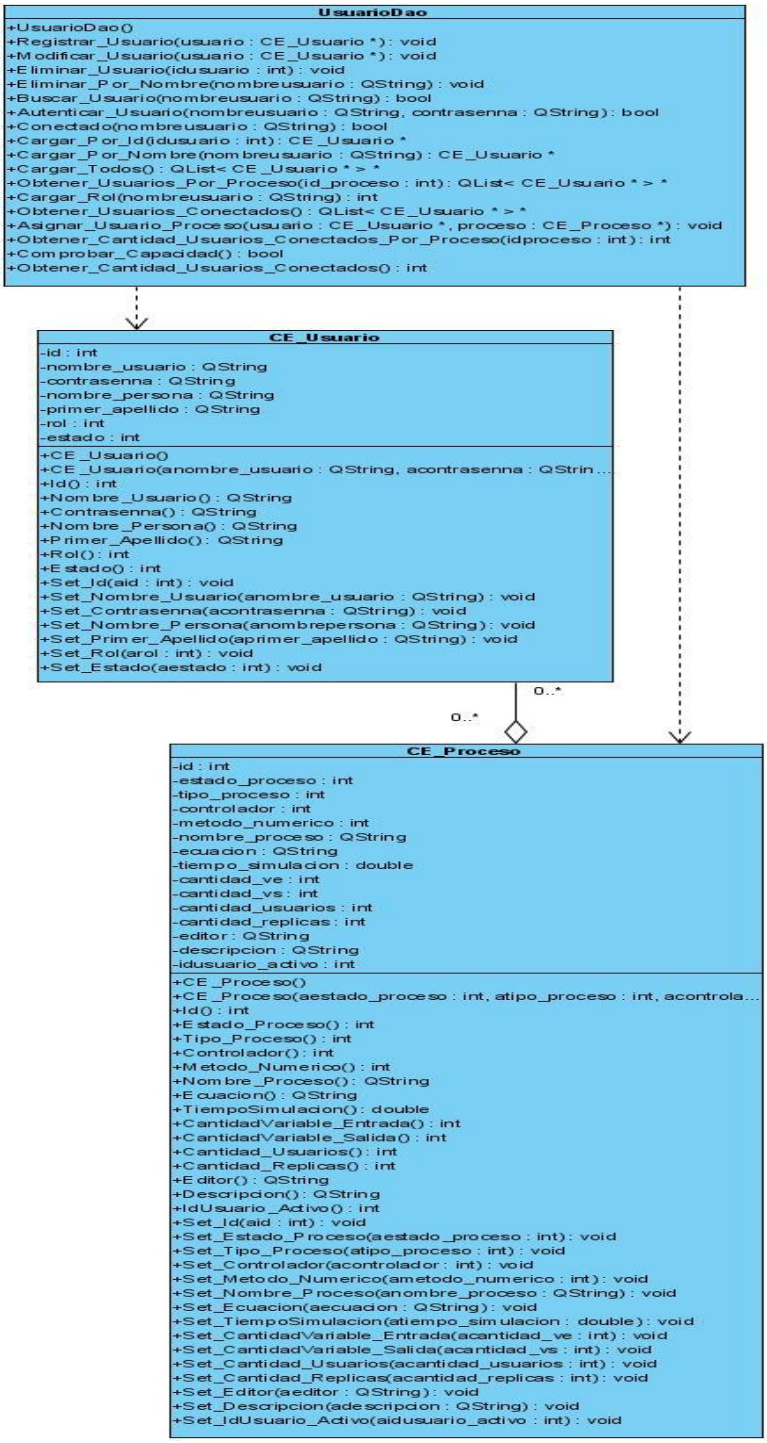
Anexo 3: Diagrama de clases de Diseño. **Establecer Límites del Sistema**



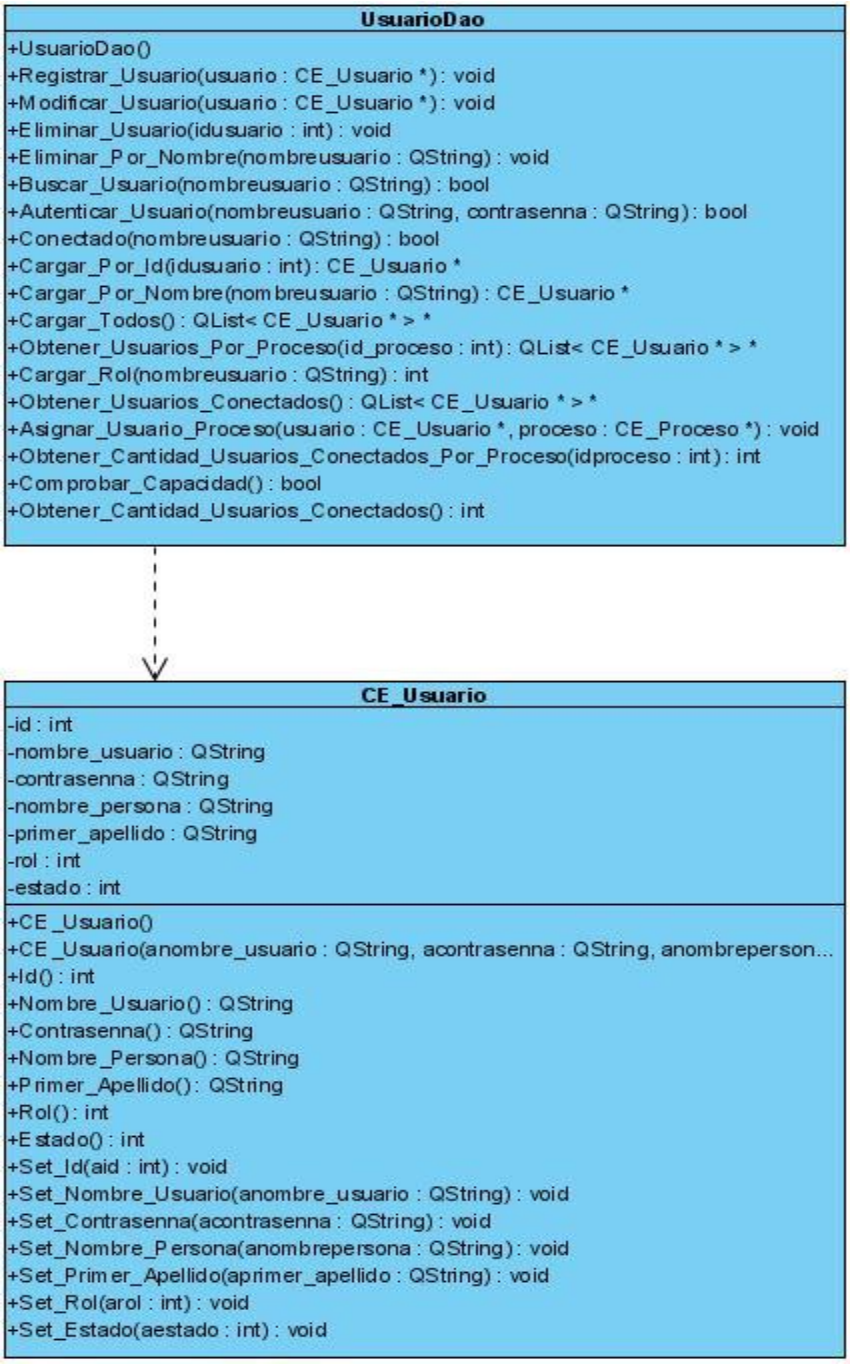
Anexo 4: Diagrama de clases de Diseño. **Gestionar Tipo de Proceso**



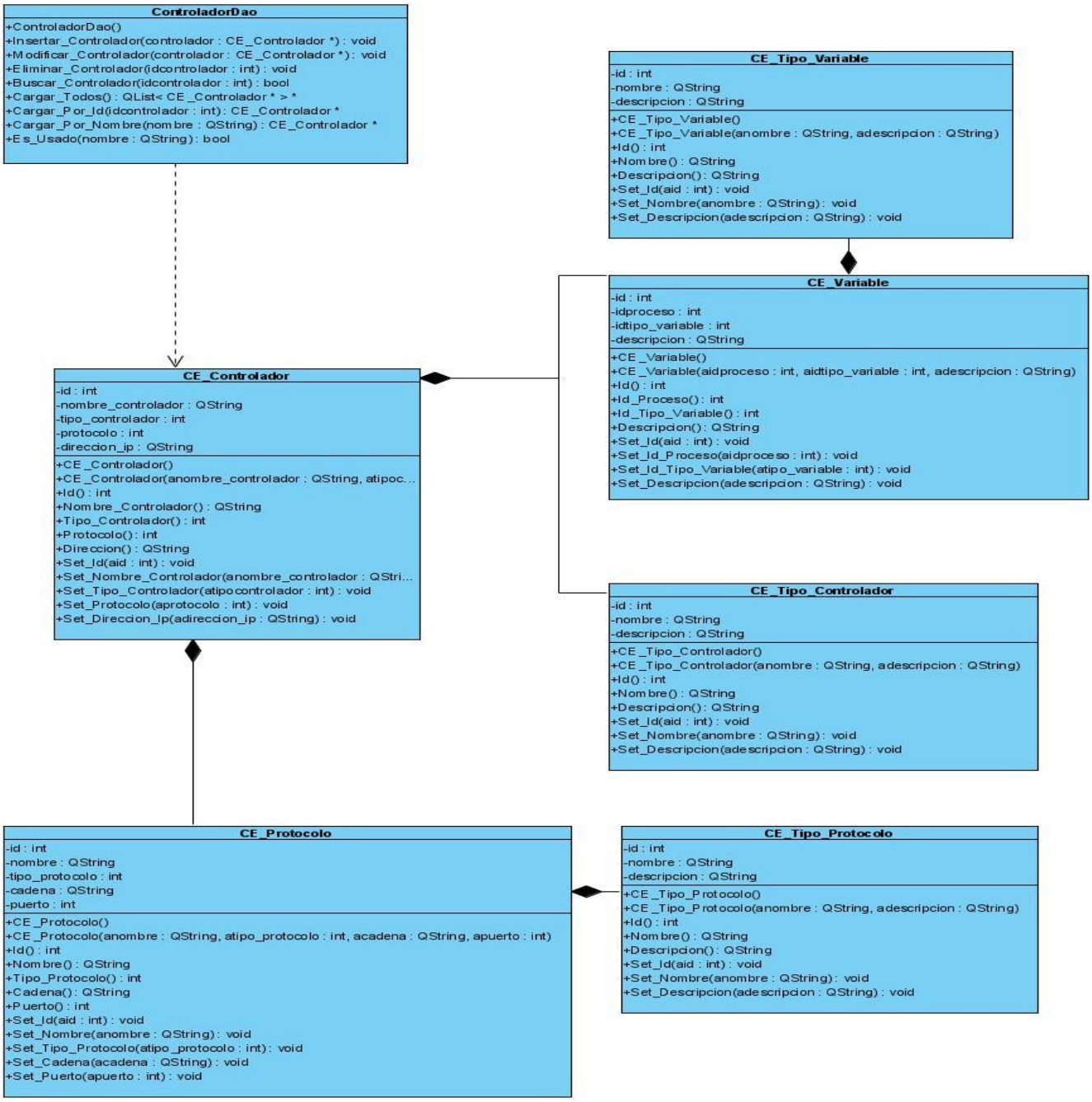
Anexo 5: Diagrama de clases de Diseño. **Gestionar Conexión a Proceso**



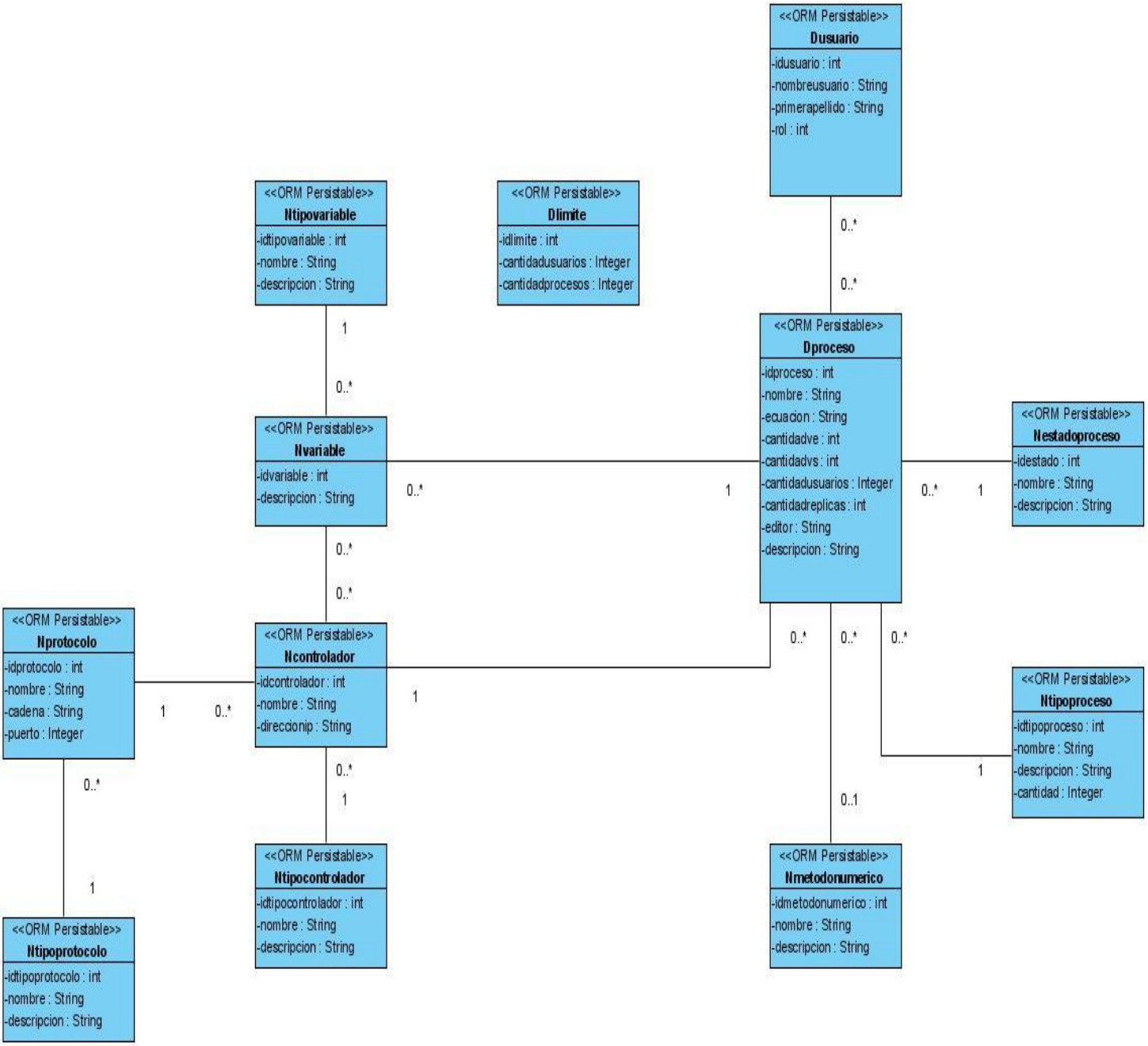
Anexo 6: Diagrama de clases de Diseño. **Gestionar Conexión al Sistema**



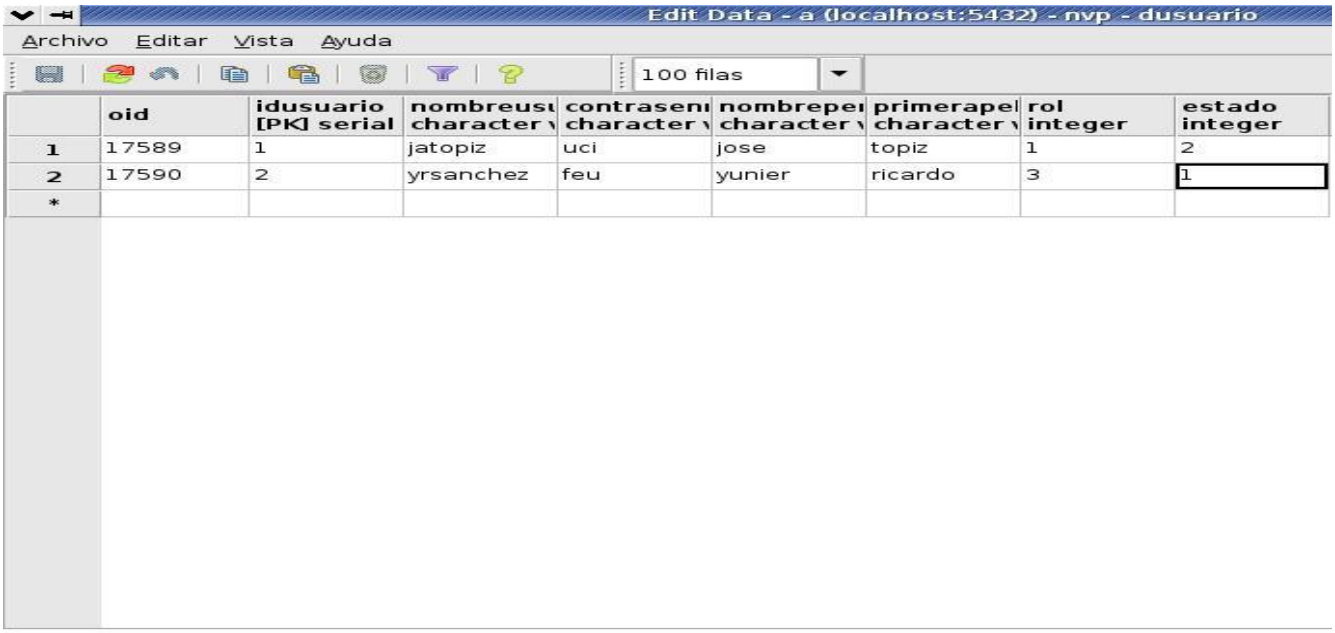
Anexo 7: Diagrama de clases de Diseño. **Gestionar Controlador**



Anexo 8: **Diagrama de Clases Persistentes**



Anexo 9: **Resultados después de insertar datos**



	oid	idusuario [PK] serial	nombreusu character	contraseni character	nombrepe character	primerapel character	rol integer	estado integer
1	17589	1	jatopiz	uci	jose	topiz	1	2
2	17590	2	yrsanchez	feu	yunier	ricardo	3	1
*								

Figura 7 Datos insertados en la BD

Anexo 10: **Resultados de las pruebas después de modificar y eliminar datos**

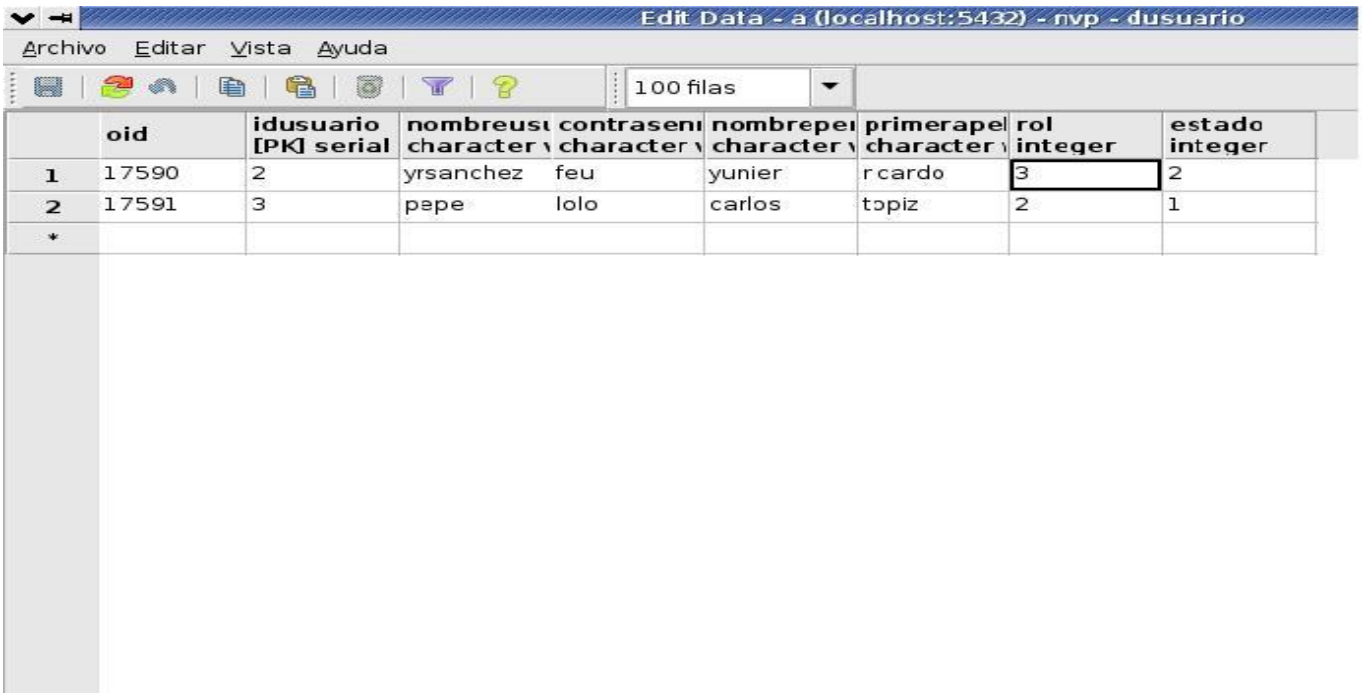


Figura 8 Datos de la BD

Anexo 11: **Resultados de las pruebas de unidad**

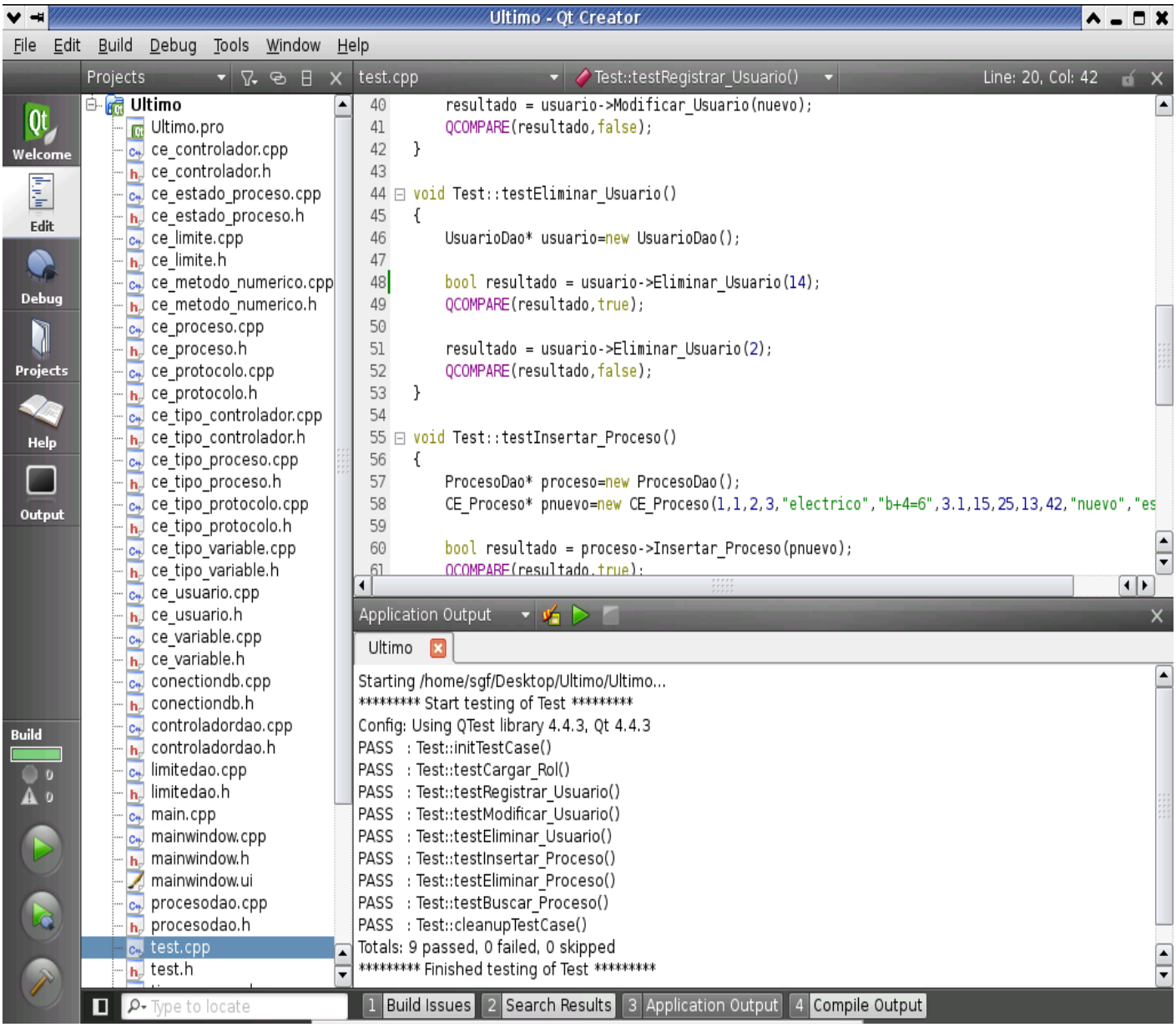


Figura 9 Resultados de las pruebas de unidad

Glosario de Términos

Herramienta CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador): Las Herramientas CASE son aplicaciones informáticas empleadas para a aumentar la productividad en el desarrollo de software disminuyendo el coste de las mismas en términos de tiempo y de dinero.

SGBD (Sistema Gestor de Base de Datos): Conjunto de programas, procedimientos y lenguajes, que suministran tanto a usuarios no informáticos como a los analistas, programadores o al administrador, los medios precisos para describir, restaurar y manejar los datos, conservando su integridad, confidencialidad y seguridad.

Hardware: Componentes físicos de un ordenador o de una red, en contraposición con los programas o elementos lógicos que los hacen funcionar.

Software: Componente intangible de una computadora, es decir, un conjunto de programas y procedimientos esenciales para hacer posible la realización de una tarea específica.

Software libre: Es el software que, una vez obtenido, se puede usar, copiar, estudiar, modificar y redistribuir libremente.

SQL (Structured Query Language): Es un lenguaje de acceso a las BD, permite especificar todas las operaciones sobre la BD como por ejemplo: inserción, borrado y actualización. Usa características de álgebra y cálculo relacional proporcionando de esta forma efectuar consultas a la BD de forma sencilla.

Multiplataforma: Es usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que funcionen en diversas plataformas. Una plataforma es una combinación de hardware y software utilizada para ejecutar aplicaciones; en su forma más simple consiste exclusivamente de un sistema operativo, una arquitectura, o una combinación de ambos.

RAID (Redundant Array Of Independent/Inexpensive Disks). Es un método de combinación de varios discos duros para formar una única unidad lógica en la que se recogen los datos de forma redundante. Ofrece mayor tolerancia a fallos y más altos niveles de rendimiento que un sólo disco duro o un grupo de discos duros independientes.

DAO (Data Access Object): Objeto de Acceso a Datos, es un componente de software que facilita una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una BD o un archivo.

Linux: Es un sistema operativo multi-usuario, multi-tarea que puede ejecutarse en muchas plataformas. Tiene una buena interoperabilidad con otros sistemas operativos, incluyendo los de Apple, Microsoft y Novell. Soporta una variada gama de software, incluyendo el sistema de ventanas X y redes TCP/IP.

Open-Source: Código abierto, es el término con el que se conoce al software distribuido y desarrollado libremente.

Programación orientada a objeto: La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación. Con la POO tenemos que aprender a pensar las cosas de una manera distinta, para escribir nuestros programas en términos de objetos, propiedades, métodos y otras cosas que veremos rápidamente para aclarar conceptos y dar una pequeña base que permita soltarnos un poco con este tipo de programación.

Consultas o querys: petición al SGBD para que procese un determinado comando SQL. Incluye tanto peticiones de datos como creación de bases de datos, tablas, modificaciones, inserciones, etc.

Tuplas: Las filas de una tabla.