UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

TÍTULO: "ARQUITECTURA DE DATOS PARA LA PLATAFORMA MODULAR DE IDENTIFICACIÓN Y CONTROL DE ACCESO DE LA UCI."

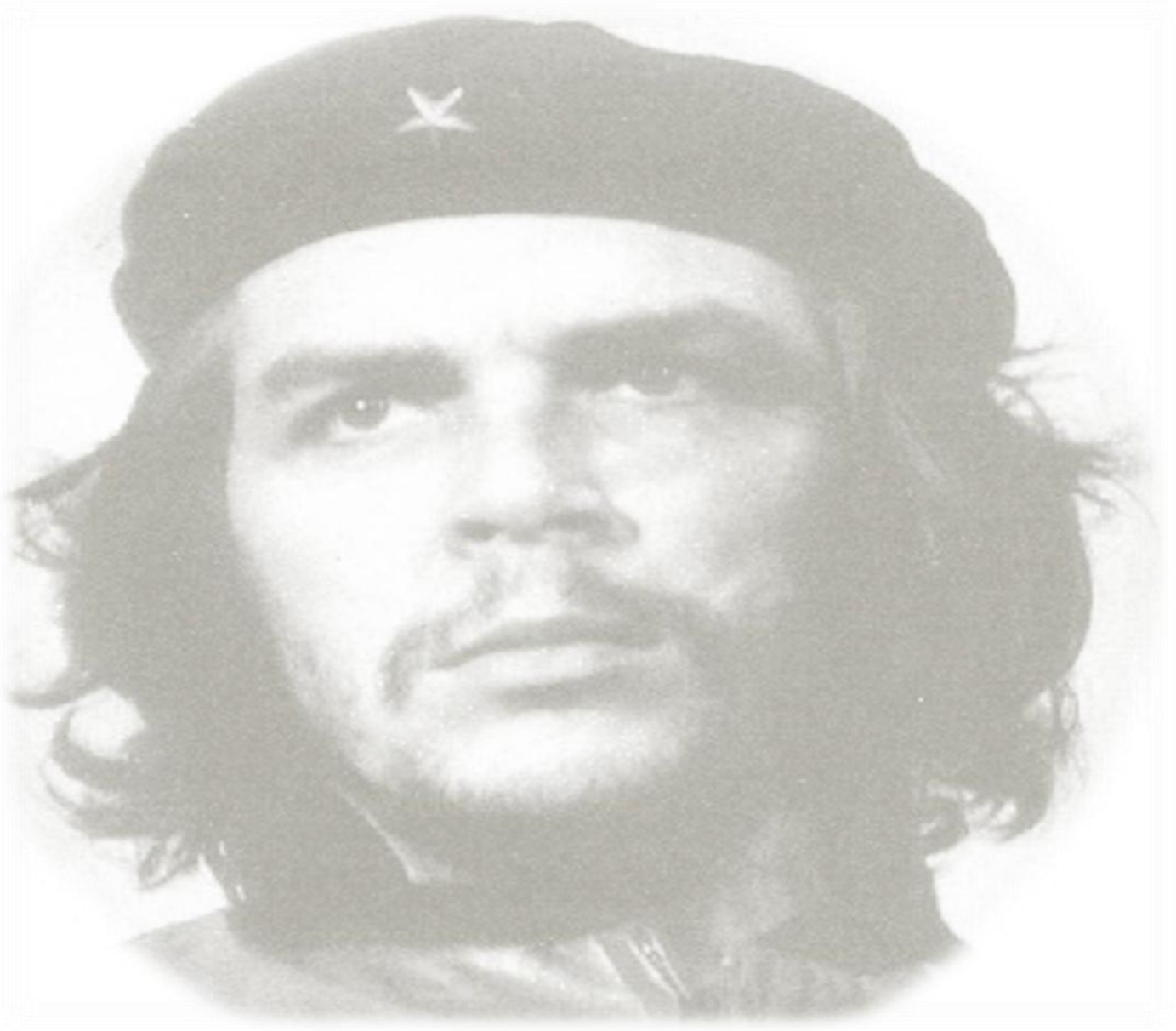
TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

AUTORES: YORDANYS ALBELO TALAVERA.
LEIDYS RIVERO GARCÍA.

TUTOR: LIC. YADIER PERDOMO CUEVAS.

CIUDAD DE LA HABANA, CUBA.

JUNIO, 2013.



El aspecto fundamental en el cual la juventud debe señalar el camino es precisamente en el aspecto de ser vanguardia en cada uno de los trabajos que le compete.

Ernesto Che Guevara.

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del trabajo titulado: *"Arquitectura de datos para la Plataforma Modular de Identificación y Control de Acceso de la UCI"*, y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los _____días le mes de _____ del año____.

Leidys Rivero García.

_____

Autor.

Yordanys Albelo Talavera.

_____

Autor.

Lic. Yadier Perdomo Cuevas.

_____

Tutor.

AGRADECIMIENTOS

Quisiera aprovechar este espacio para agradecerles a las personas que de una forma u otra me ayudaron con la realización de este trabajo de diploma. En primer lugar, a mi familia, por apoyarme en el transcurso de estos cinco años, en especial a mi madre Odalys, que ha sido mi principal guía y fuente de inspiración; a mi padre Carlos por tanto sacrificio, esfuerzo y consagración, y a mis hermanos Carlitos y Dayana por darme su apoyo incondicionalmente. Otro agradecimiento muy especial va dedicado a todos mis tíos maternos y paternos, por estar a mi lado en el momento que más necesité de su ayuda, especialmente a Rosario, Riguito, Dania y Tania y a mis primas por apoyarme en cada momento. Agradezco además, a Yoanis por dedicarse de lleno a mi familia en los momentos que no estuve presente.

Agradecer además, a mi novio Ricardo por su paciencia, dedicación y amor; y a su familia por brindarme su cariño y apoyo. A mis amistades que siempre estuvieron a mi lado en los momentos difíciles; a mis compañeros de aula, a todos los profesores que de un modo u otro me brindaron su ayuda y sobre todo a mi tutor Yadier Perdomo Cuevas por su preocupación en la elaboración correcta del trabajo. A mi compañero de tesis Yordanys, muchas gracias por apoyarme y ayudarme durante toda la realización del trabajo. A todos muchas gracias.

Leidys.

Agradezco mis padres Odalys y Juan por ser el mayor ejemplo que un hijo puede tener; a mi hermana por quererme tanto y darme todo su amor y apoyo; además, agradezco a mis tíos por brindarme toda su ayuda cuando necesité de ellos y sobre todo a mi abuelos por guiarme y aconsejarme en todo momento.

Otro agradecimiento muy especial va dedicado a mis compañeros y amistades que me dieron todo su apoyo y me llenaron de aliento para seguir adelante. También, a todos los educadores que de una forma u otra me brindaron su ayuda para lograr el éxito de éste trabajo de diploma, sin dejar de mencionar al profesor Yendry y a mi tutor Yadier Perdomo, a todos muchas gracias.

Yordanys.

DEDICATORIA

Dedico esta investigación a toda mi familia en general, incluyendo mi novio y en especial a mis padres, que son lo más grande que tengo en la vida y sin ellos no hubiera sido lo que soy hoy.

Leidys.

Dedico este trabajo de diploma a mis padres que son lo más bonito que tengo en la vida, a mi hermana y a toda la familia en general por todo el apoyo que me han brindado siempre.

Yordanys.

Resumen

El Centro de Identificación y Seguridad Digital (CISED) surge como un departamento destinado a la identificación y el control de acceso de personas a la universidad, aunque no siempre se realiza este proceso de forma correcta, dando paso a las deficiencias en la seguridad que requiere la institución.

Debido a que se hace necesario mejorar este proceso de identificación y control de acceso, se desarrolla la Plataforma Modular de Identificación y Control de Acceso (PMICA) y con ella la creación de una arquitectura de datos que cumpla el objetivo de diseñar e implementar una base de datos y una capa de acceso a datos que separe la lógica del negocio del acceso a datos.

Para dar cumplimiento al mismo, en la investigación se estudian las principales definiciones referentes a las bases de datos, los sistemas gestores de bases de datos más comunes en la actualidad, así como las formas normales y otras restricciones que se imponen a un modelo de datos relacional. Se especifica el diseño detallado de la base de datos, incluyendo tablas, índices, llaves primarias, foráneas, utilizando patrones de diseño, garantizando la unidad y estabilidad de los datos. También se analiza la propuesta de solución, conjuntamente con temas de seguridad, tanto en la base de datos como en PostgreSQL, dando paso a la optimización del sistema. Se migró para PostgreSQL el servicio de persistencia que posee el Bison de Oracle. Además se efectuaron pruebas al modelo de datos de la solución propuesta validando el correcto funcionamiento del sistema.

Palabras Clave:
Bases de datos, motor de persistencia, modelo de datos.

Tabla de contenido

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

En organizaciones donde circulan diariamente un gran cúmulo de personas, resulta complicado mantener la identificación y control de acceso del personal a las diferentes áreas o locales de una empresa. La Universidad de las Ciencias Informáticas (UCI), es uno de los centros de mayor desarrollo tecnológico del país con una notable circulación del personal, lo que pone en riesgo la seguridad de los medios y bienes que posee la institución.

Son varios los sistemas que se han desarrollado en la UCI para la identificación y el control de acceso de personas. Un ejemplo de esto es el Sistema Único de Identificacion Nacional (SUIN) encargado de la identificación de ciudadanos cubanos y extranjeros residentes en el país. Además se destaca el Bison Framework, el cual engloba un conjunto de actividades y servicios encargados de instanciar los flujos de trabajos relacionados con la identificación y acceso de personas a la universidad. Ambos sistemas están orientados a la persistencia de los datos utilizando Entity Framework y como Sistema Gestor de Base de Datos Oracle.

Actualmente la universidad no cuenta con un sistema de identificación y control de acceso que se ajuste a otros clientes con otros tipos de actividades en sus procesos. El uso del Bison permitirá el trabajo con diferentes tipos de acciones, pero utilizando Oracle como gestor de base de datos privativo y costoso en cuanto a temas de licencias. Por este motivo, se hace necesaria la creación de una arquitectura de datos para la Plataforma Modular de Identificación y Control de Acceso (PMICA) desarrollada por el Centro de Identificación y Seguridad Digital (CISED), elaborando además una base de datos en PostgreSQL y la capa de acceso a datos utilizando NHibernate, migrando el servicio de persistencia que posee el Bison de Oracle hacia PostgreSQL.

Todo esto origina el siguiente **problema de la investigación** que queda expresado con la siguiente pregunta: ¿Cómo garantizar la persistencia de datos para la Plataforma Modular de Identificación y Control de Acceso de la UCI de manera que separe la lógica de negocio del acceso a datos?

Como consecuencia de lo antes planteado se deriva el siguiente **objeto de estudio**: los procesos o sistemas de identificación y control de acceso de personas y recursos asociados.

El **campo de acción** está enmarcado específicamente en la capa de persistencia de datos para sistemas de identificación y control de acceso.

Para la solución al problema se plantea como **objetivo general**: garantizar el almacenamiento de los datos mediante el diseño e implementación de la capa de persistencia para la PMICA de manera que separe la lógica del negocio del acceso a datos.

De los que se derivan los siguientes **objetivos específicos**:

- Elaborar el estado del arte de la investigación basándose en las tendencias actuales del desarrollo de los sistemas gestores de bases de datos y bases de datos relacionales.
- Analizar los diferentes patrones de diseño, garantizando el estudio de los mismos.
- Seleccionar las herramientas, tecnologías y metodología que garanticen la calidad máxima de la investigación.
- Diseñar la base de datos para la Plataforma Modular de Identificación y Control de Acceso de la UCI.
- Implementar la capa de persistencia de datos para la PMICA.
- Diseñar y ejecutar pruebas a todos los componentes de la capa de acceso a datos.

Para dar cumplimiento a los objetivos trazados se desarrollaron las siguientes **tareas de investigación**:

- Análisis y estudio de los sistemas de bases de datos y bases de datos relacionales.
- Estudio de los diferentes modelos y patrones de diseño existentes para su utilización en la investigación.
- Análisis y estudio de los sistemas gestores de bases de datos a utilizar en el trabajo.
- Diseño e implementación de la capa de persistencia de datos para la PMICA.
- Implementación de la base de datos en PostgreSQL.
- Ejecución de pruebas de integridad a los modelos elaborados.
- Ejecución de pruebas de volumen, carga y estrés a la base de datos.

Para guiar la investigación se plantea como **idea a defender** que: diseñando e implementando la capa de persistencia de datos para PMICA y migrando el servicio de persistencia del Bison, se logrará una arquitectura que persista los datos de dicha plataforma en PostgreSQL, garantizándose con ello un correcto almacenamiento y control de la información, separando la lógica del negocio del acceso a datos.

Como **posible resultado** se espera la implementación de la base de datos y la capa de persistencia en PostgreSQL para la Plataforma Modular de la UCI, de manera que garantice un apropiado almacenamiento de los datos.

A lo largo del trabajo se ponen de manifiesto métodos investigativos. A continuación se explican los que han sido seleccionados:

- **Histórico – Lógico**: Este método fue usado en la investigación para estudiar el progreso de los sistemas de almacenamiento y las diferentes bases de datos existentes que se han ido perfeccionando en nuestros días, así como el estudio de motores de persistencia y tecnologías que influyen en el aprendizaje del proceso de identificación y control de acceso de la UCI.

- **Analítico – Sintético**: Se utilizó a la hora de analizar los diferentes sistemas gestores de bases de datos, precisando las características de cada uno de ellos. Además facilitó el estudio de los patrones de diseño para su futura implementación.
- **Modelación**: Permitirá identificar en los requerimientos del software entidades o elementos que conformarán el modelo de dominio, el modelo de datos y otros modelos que destacaran sus relaciones entre estas entidades, además de verse expresado en el diseño de la base de datos para la PMICA.

El documento consta de tres capítulos, desarrollados a partir del estudio realizado. La descripción de los mismos se presenta a continuación:

- **Capítulo 1: Fundamentación Teórica**

En este capítulo se realiza un estudio del estado del arte de la investigación; se abordan los principales conceptos que serán presentados a lo largo de todo el trabajo, así como un análisis de las principales herramientas, tecnologías y metodología que son utilizadas en el desarrollo del trabajo, especificando las empleadas en la implementación de la base de datos y justificando su uso.

- **Capítulo 2: Diseño de la base de datos y Propuesta de Solución.**

En este capítulo se elabora una propuesta de solución, donde se ilustran algunos de los modelos creados para dar una perspectiva del diseño de la base de datos. Se describen los principales conceptos y se exponen las relaciones más fuertes entre entidades.

- **Capítulo 3: Implementación y Prueba de la solución propuesta.**

En este capítulo se define el ambiente donde será desplegada la solución propuesta. Se ofrecerá información acerca de la validación teórica y funcional realizada a la solución, con el objetivo de examinar su diseño y comportamiento al ser gestionado mediante consultas. Además, se llevarán a cabo diferentes tipos de pruebas, así como temas de seguridad y optimización con que contará la base de datos.

CAPÍTULO 1: Fundamentación Teórica.

Introducción

El presente capítulo muestra la fundamentación teórica de la investigación, dando paso posteriormente a la solución del problema planteado. Se explican las características de los sistemas gestores de bases de datos. Además, se exponen un conjunto de conceptos significativos en el área de las bases de datos y que son claves en el desarrollo de este trabajo de diploma. Por último se realiza un estudio de las diversas herramientas, tecnologías de programación y metodología de desarrollo de software que serán usadas en la implementación y pruebas de la base de datos.

1.1 Base de Datos (BD)

Las técnicas de almacenamiento de datos han ido evolucionando a lo largo de los años. Las tarjetas perforadas, las cintas magnéticas y los discos fijos, fueron sistemas innovadores de la década del 50. Posteriormente fueron apareciendo los primeros temas relacionados con el concepto de base de datos, hasta nuestros días que se reflejan impresiones más certeras relacionadas con su significado.

Dentro de estos criterios relacionados con el significado de las BD se destacan el de varios autores que dan sus versiones, como por ejemplo:

- “Una BD no es más que un conjunto de dados interrelacionados entre sí, almacenados con carácter más o menos permanente en una computadora, o sea, puede considerase una colección de datos variables en el tiempo” (Mato García, 2005).
- “Se define una base de datos como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular” (Pérez Valdés, 2007).

De todas las definiciones expuestas la más acertada en cuanto a lo que se quiere expresar sobre una base de datos es la que a continuación se expresa: “una base de datos se define como un archivo en el que se almacena información en campos de las tablas, teniendo acceso a ellas posteriormente, tanto de forma separada, como conjunta. Se utiliza fundamentalmente para recoger grandes cantidades de información” (Vallez Pérez, 2009). Optando por la última definición, se tiene una idea más aceptada de lo expresa una base de datos y cuáles son los beneficios que aporta:

1. Manipulan:
    a. Grandes volúmenes de datos.

b. Información de alto nivel de complejidad.
c. Bajo tiempo de acceso requerido.

2. Reducen espacio de almacenamiento.
   a. Reducen la redundancia.
   b. Evitan inconsistencias.
   c. Comparten información.
   d. Seguridad e integridad.

1.1.1 Pautas para el diseño de una base de datos

La clave para obtener un apropiado diseño de base de datos radica en comprender exactamente qué información es la que se desea guardar y la forma en que un sistema almacena los datos. Al organizar los datos de forma adecuada, se tiene la posibilidad de combinar y presentar información de muchas formas diferentes, proporcionando con esto una característica fundamental a la hora del trabajo con datos, flexibilidad entre los distintos tipos de información. A continuación se indican pasos que hay que seguir en el proceso de diseño de una base de datos (Microsoft, 2010):

- Determinar el propósito de la base de datos: permite estar capacitado para realizar los demás pasos del diseño.
- Determinar las tablas necesarias: ofrece una información detallada de cada entidad o tema principal que luego pasará a ser una tabla.
- Determinar los campos necesarios: se decide qué información se almacenará en cada tabla, convirtiéndose en un campo y posteriormente mostrándose como una columna de la tabla.
- Definir las calves principales: permiten identificar inequívocamente cada fila de una tabla.
- Determinar las relaciones entre tablas: permite examinar cada tabla y decidir cómo se relacionan los datos de una tabla con las restantes, añadiéndose nuevos campos o tablas para corregir las relaciones según sea necesario.
- Perfeccionar el diseño: este proceso se realiza con el objetivo de detectar errores en el diseño y realizar los ajustes necesarios para corregirlos.
- Aplicar las reglas de normalización a los datos: permite comprobar si los datos de las tablas están estructurados correctamente

El diseño de la base de datos en cuestión está regido por 6 fases (Pérez Gamboa, 2012):

1. Recopilación y análisis de los requisitos: Este proceso lo realizan los analistas del proyecto donde recogen toda la información necesaria relacionada con el sistema a elaborar. También identifican los datos y usuarios reales y posibles que contendrá la BD.
2. Diseño conceptual: Mediante este diseño se permite descubrir el significado de los datos con los que se irá a trabajar. Este esquema permite transmitir todos los conocimientos adquiridos a las entidades del modelo, siendo éste una fuente de información para el diseño lógico de la BD.
3. Elección de un sistema gestor de bases de datos: con esta elección se garantiza el cumplimiento de diferentes factores de rendimiento en la BD.
4. Diseño lógico: permite al diseñador construir un esquema lógico teniendo como base un modelo conceptual, además de posibilitar la correcta representación de los datos en caso de cambios o modificaciones. En este segmento del diseño se realiza la normalización de la base de datos para garantizar la validez del esquema lógico.
5. Diseño físico: en este proceso se describe la implementación de la BD. Entre el diseño lógico y el físico existe una retroalimentación ya que si efectúan cambios para mejorar las prestaciones en el diseño lógico se puede afectar la estructura del esquema físico.
6. Implementación de la BD: Mediante la creación y compilación de varios esquemas y ficheros de datos se obtiene como resultado una base de datos que corresponde al diseño previamente efectuado.

Con el uso de estas pautas para el diseño de una base de datos se garantiza solidez en los modelos de datos, además de una guía por la cual regirse para desempeñar un apropiado trabajo.

1.2 Bases de Datos Relacionales

Una base de datos relacional es una base de datos en donde todos los datos visibles al usuario están organizados estrictamente como tablas de valores, y en donde todas las operaciones de la base de datos operan sobre estas tablas. En términos tradicionales una relación se asemeja a un archivo, una tupla a un registro, y un atributo a un campo (Mato García, 2005).

Dentro de los objetivos de las bases de datos relacionales se tienen los de:

- Proteger el valor de los datos.
- Hacer que las fuentes de datos sean responsables de los cambios de las necesidades de información.
- Habilitar que la organización que procesa datos logre un mejor control y seguimiento de sus planes de negocios y metas.

- Reducir los costos de optimización del desempeño.

Beneficios de las bases de datos relacionales:

- Puede reducirse la repetición de datos.
- Puede evitarse los errores al insertar datos, problemas de actualización y establecer restricciones en el borrado.
- Los datos pueden compartirse. No solo significa que las aplicaciones existentes pueden compartir los datos de la base de datos, sino también que es factible desarrollar nuevas aplicaciones que operen con los mismos datos almacenados.
- Control centralizado de la base de datos.
- Pueden aplicarse restricciones de seguridad.
    a. Asegurar que el único medio de acceso a la base de datos sea a través de los canales establecidos.
    b. Definir controles de autorización para que se apliquen cada vez que se intente el acceso a datos sensibles.
- Es compacto. No hacen falta archivos de papeles que pudieran ocupar mucho espacio.
- Es rápido. La máquina puede obtener y modificar datos con mucha mayor velocidad que un ser humano, así es posible satisfacer con rapidez consultas de casos particulares, sin necesidad de búsquedas visuales o manuales que requieren mucho tiempo.

1.3 Sistemas Gestores de Bases de Datos (SGBD)

Es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez. 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 (Mato García, 2005).

Los SGBD deben cumplir con un conjunto de objetivos generales de modo que faciliten el proceso de diseño de aplicaciones y que los tratamientos sean más eficientes y rápidos, dando la mayor flexibilidad posible a los usuarios. A continuación se nombran algunos de estos objetivos generales:

- Independencia de los datos y los programas de aplicación.
- Minimización de la redundancia.
- Integración y sincronización de las bases de datos.
- Integridad de los datos.

- Seguridad y recuperación.

Existen otros objetivos que deben cumplir los SGBD que en muchos casos dependen de las condiciones o requerimientos específicos de utilización del sistema. Todos los SGBD brindan lenguajes e interfaces adaptadas para cada tipo de usuario. Dichos lenguajes se clasifican de la siguiente manera:

- Lenguaje de definición de datos (DDL), el cual garantiza la definición o descripción de los objetos de la base de datos. Se utiliza para especificar el esquema de la base de datos (esquema conceptual e interno), las vistas y las estructuras de almacenamiento.
- Lenguaje de manipulación de datos (DML), garantiza la manipulación o procesamiento de esos objetos. Se utiliza para leer y actualizar los dados así como para ser utilizados por los usuarios en las funciones de creación, inserción, modificación, eliminación de consultas.

1.3.1 Oracle

Sistema Gestor de Base de Datos utilizado por grandes corporaciones o empresas, ya que sus características ofrecen la posibilidad de gestionar, de una manera avanzada, un elevado número de usuarios e información (Brea, 2011).

Son diversos los beneficios que aporta este gestor, destacándose que:

- Puede ejecutarse en todas las plataformas, desde una computadora (PC) hasta un supercomputador.
- Permite el uso de particiones para la mejora de la eficiencia, de replicación e incluso ciertas versiones admiten la administración de bases de datos distribuidas.
- El software del servidor puede ejecutarse en multitud de sistemas operativos. Existe incluso una versión personal para Windows 9x, siendo un punto a favor para los desarrolladores que llevan trabajo a casa.

También hay que mencionar que ante todas estas ventajas Oracle tiene la desventaja de que:

- Desde el lanzamiento original de la versión 8.0 de Oracle sobrevinieron varias versiones con correcciones, hasta alcanzar la estabilidad en la 8.0.3. El motivo de tantos fallos fue, al parecer, la remodelación del sistema de almacenamiento por causa de la introducción de extensiones orientadas a objetos.

Este gestor es utilizado en la investigación con el objetivo de migrar hacia PostgreSQL los procedimientos almacenados que utiliza el servicio de persistencia que posee el Bision.

1.3.2 PostgreSQL

PostgreSQL es un Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos, de código abierto; brinda un control de concurrencia multi-versión (MVCC por sus siglas en inglés) que permite trabajar con grandes volúmenes de datos. Además soporta gran parte de la sintaxis SQL y cuenta con un extenso grupo de enlaces con lenguajes de programación. Posee características significativas del motor de datos, entre las que se pueden incluir las subconsultas, los valores por defecto, las restricciones a valores en los campos (constraints) y los disparadores (triggers) (Montenegro Bernot, 2011). Los SGBD reflejan de cierto modo una elevada importancia en sus condiciones de uso, como así se expresa a continuación:

- PostgreSQL se caracteriza por ser un sistema estable, de alto rendimiento, gran flexibilidad ya que funciona en la mayoría de los sistemas Unix, además tiene características que permiten entender fácilmente el sistema.
- PostgreSQL puede ser integrada al ambiente Windows permitiendo de esta manera a los desarrolladores, generar nuevas aplicaciones o mantener las ya existentes.
- Permite desarrollar o migrar aplicaciones para que utilicen a PostgreSQL como servidor de BD.

Por lo expuesto, PostgreSQL en su versión 9.1.2 se convierte en una gran alternativa al momento de decidirse por un sistema gestor de bases de datos a utilizar en la investigación, ofreciendo una replicación sincrónica de datos, es decir, proporciona a los desarrolladores algo de inmenso valor: replicación sin pérdida de datos. Incluso se puede controlar la durabilidad de cada transacción y todos los niveles de durabilidad pueden co-existir en la misma aplicación. Además, como parte de las exigencias del cliente, en este caso la Universidad de las Ciencias Informáticas, se elige a PostgreSQL como el gestor idóneo para el desarrollo de la base de datos. El Centro de Tecnologías de Gestión de Datos (DATEC) es de gran importancia para el trabajo en cuestión, ya que posibilitaría un asesoramiento de la investigación en caso que fuese necesario, pues basa su labor en la creación de bienes y servicios informáticos relacionados con la gestión de datos, utilizando PostgreSQL como gestor de base de datos.

1.4 BisonFramework

BisonFramework es un framework para la organización de procesos de negocio basado en la tecnología Windows Workflow Fundation. Su función principal es proporcionar un componente

que permita gestionar las instancias de workflow[1]. Además encapsula un conjunto de actividades y servicios que brindan un mayor dinamismo al desarrollo de sistemas (Galens Ameneiro, y otros, 2010).

Son muchas las ventajas que ofrece BisonFramework, pero dentro de las más relevantes se pueden mencionar las siguientes:

- Proporciona una mayor aproximación a los usuarios de negocio.
- Propone rapidez y flexibilidad para modelar y cambiar los procesos según las necesidades.
- Aporta escalabilidad y propone una arquitectura donde se encuentran definidas las capas de presentación y negocio.
- Posee actividades y servicios especializados, permitiendo la definición del flujo de trabajo dentro del workflow.

Con el desarrollo del trabajo se espera migrar el servicio de persistencia de datos que ofrece el Bison hacia el PostgreSQL, sin dejar de mencionar que este servicio cuenta con procedimientos almacenados que de igual manera son migrados de Oracle hacia PostgreSQL.

¿Qué es Window Workflow Fundation (WWF)?

Es una tecnología extensible para desarrollar soluciones de workflow sobre la plataforma .NET. Proporciona una interfaz de programación de aplicaciones API[2] y herramientas para el desarrollo y la ejecución de aplicaciones basadas en flujos de trabajo. La mayoría de estos flujos están organizados dentro del contexto tecnológico para la automatización de sus procedimientos.

1.5 Optimización de bases de datos

En el desarrollo de un sistema de bases de datos la optimización juega un papel fundamental, ya que permite mejorar el rendimiento, la eficiencia y las funcionalidades de dicho sistema en condiciones dadas de acuerdo a sus características (Mariluz Hernández, y otros, 2009). Es válido aclarar que, para que exista una eficaz optimización de las bases de datos, debe existir también un correcto diseño inicial, tanto lógico como físico. El objetivo de la optimización consiste en minimizar el tiempo de respuesta para cada petición y maximizar el rendimiento de todo el sistema disminuyendo el tráfico de red, el acceso al disco duro y el tiempo de CPU[3].

[1] Workflow: Flujo de trabajo.
[2] API: conjunto de funciones residentes en sistema operativo permitiendo que una aplicación se ejecute.
[3] CPU (Central Process Unit) Unidad Central de Procesamiento.

Para trabajar en el perfeccionamiento de una BD existen dos clasificaciones:

1. El afinamiento proactivo: emprendido con mayor profundidad en las primeras etapas del diseño, responsabilizándose entre otras cosas, de aplicar técnicas para mejorar el rendimiento, así como de establecer qué configuración de hardware será la apropiada para responder y cumplir con los requisitos y expectativas trazadas.
2. El afinamiento reactivo: puesto en práctica para mejorar sistemas en explotación, proporcionando un mecanismo que identifique y combata los riesgos a los que está expuesto el sistema.

El rendimiento del sistema debe ser monitorizado y evaluado regularmente después de iniciar las funciones de la BD. Este es un paso muy importante a tener en cuenta, ya que si se detectan fallos en el rendimiento o demoras en el tiempo de respuesta de una petición sería demasiado tarde para rediseñar la BD, entonces lo único que se podría hacer para mejorar estos problemas es reasignar más memoria y mejorar los parámetros de entrada y salida de disco.

1.5.1 Técnicas de optimización

Un paso fundamental para la optimización de una BD es contar con técnicas que logren hacer de la implementación un escenario perfecto, para lograr el éxito en cuanto a rendimiento se trata. A continuación se definen algunas técnicas de optimización:

1. **Diseño de bases de datos.**

Con el correcto diseño se garantiza la validez en el cumplimiento de los objetivos de la base de datos, razón por la cual es de máxima importancia el estudio de los principios y aspectos fundamentales de un correcto diseño.

2. **Normalización de bases de datos.**

Se utiliza la normalización con el objetivo de crear estructuras de datos que eviten las anomalías que ocurren durante la actualización de los datos; permitiendo eliminar la información redundante en una BD, y que trae consigo un grupo de fases, que al completarse, se dice que la relación está en una de las formas normales que se muestran a continuación (Mato García, 2005):

- Primera Forma Normal (1FN): una relación se encuentra en 1FN cuando todos los atributos que componen sus tuplas son atómicos.

- Segunda Forma Normal (2FN): una relación se encuentra en 2FN si, y solo si, se encuentra en 1FN y los atributos no llaves, son funcional y completamente dependientes de la clave primaria. Una relación que posea una única llave primaria se encuentra en 2FN.
- Tercera Forma Normal (3FN): una relación se está en 3FN si, y solo si, se encuentra en 2FN y no existe dependencia transitiva entre los atributos no llaves respecto a la llave primaria. Este proceso elimina la transitividad entre los atributos.
- Forma Normal de Boyce-Codd (FNBC): la relación está en FNBC si, y solo si, cada determinante[4] es una súper-llave[5] (candidata[6] o primaria).

Algunos diseñadores normalizan los modelos hasta la 3FN debido a su cómodo modo de aplicación, lo que no quiere decir, que las bases de datos no sean vulnerables, al contrario, aún pueden ser frágiles a algunas anomalías menos comunes, lo que requiere de una normalización más compleja, terminando en complicados modelos de datos, difíciles de implementar y de mantener.

3. Índices.

Son estructuras de datos que mejoran la velocidad de las operaciones en las tablas asociadas, permitiendo búsquedas de información más rápidas al proporcionar un orden e indicador de búsqueda para las consultas, evitando que el gestor revise todos los datos disponibles para devolver el resultado (Pérez Gamboa, 2012).

La indexación, tanto de llaves primaria como extranjeras, se puede obtener del modelo relacional, ya que las claves primarias identifican únicamente a cada elemento de una tupla, y las claves extranjeras marcan las relaciones entre tablas. Poseer índices en los campos adecuados de las tuplas de la base de datos optimizará los resultados, es por esto que se muestran a continuación algunas prácticas estudiadas para el desarrollo del trabajo:

- Crear un índice sobre los campos que son utilizados en las búsquedas, los que aparecen en las cláusulas *WHERE o JOIN*, para mejorar una consulta *(SELECT).*
- Utilizar índices sobre los campos con valores únicos.
- Minimizar la longitud del texto de los índices.
- Evitar la creación de índices innecesarios.

[4] Determinante: cualquier atributo o conjunto de atributos del cual depende funcional y completamente otro atributo.
[5] Súper-llave: conjunto de atributos que permite identificar a cada tabla de forma única.
[6] Llave candidata: subconjunto formado a partir de una súper-llave que no puede ser súper -clave.

El escaneo completo de una tabla ocurre cuando el gestor tiene que leer todos los registros de dicha tabla de manera secuencial para resolver una consulta, trayendo consigo:

- **Sobrecarga de CPU:** esto ocurre cuando el procesador tiene que chequear un gran volumen de registros en la tabla.
- **Concurrencia:** mientras un gestor de BD está leyendo los datos de una tabla, éste la bloquea, de tal manera que nadie más puede escribir en ella, aunque si pueden leerla. De forma similar ocurre cuando se está actualizando o eliminado filas de una tupla, con la diferencia que aquí no se puede leer.
- **Sobrecarga de disco:** esta sobrecarga ocurre cuando se escanean tablas muy grandes que consumen una gran cantidad de entrada/salida en el disco, provocando que se caliente el disco duro del servidor de base de datos, especialmente si se tiene un disco antiguo.

La acción de indexación ordena los registros de una colección de datos de acuerdo con los campos utilizados como llave primaria e incrementa sensiblemente la velocidad de ejecución de varias de estas acciones sobre la colección de datos. Para cada archivo de datos debe existir un índice cuya llave de indexación sea igual a su llave primaria, llamándose índice primario. Por otra parte, también existen índices secundarios, asociados a uno o varios atributos, diferentes a la llave primaria (Pérez González , y otros, 2011).

4. **Optimización de consultas.**

Cuando se tienen consultas muy largas que no se construyen de forma avanzada, o cuando las consultas pasan directamente a ser ejecutadas sin ser analizadas por el servidor, el uso de los procedimientos almacenados proporciona una ayuda en cuanto a rendimiento se trata, permitiendo acotar las trasferencias de datos. Existen muchos tipos de optimización de consultas, pero solo se muestran las que formaron parte del estudio para el desarrollo de la investigación:

- Optimización basada en reglas: heurísticas generales en las que el programador puede especificar reglas que guíen al optimizador en su trabajo.
- Optimización basada en costos: estimación de los costos de realizar una operación física.

La herramienta más usada para realizar la optimización de consultas de una BD es el comando *SQL EXPLAIN ANALYZE,*[7] destacándose por mostrar el registro de cada consulta SQL que es realizada y como PostgreSQL procesa cada una de ellas, una vía para optimizar consultas es usar índices en columnas específicas y combinaciones de columnas que corresponden a las consultas más frecuentes (Pérez Gamboa, 2012).

1.5.2 Optimización para PostgreSQL

PostgreSQL como gestor de base de datos a utilizar en la investigación cuenta con parámetros de configuración, los cuales tienen como meta hacer que el gestor corra más rápido en la plataforma en que se encuentra instalado, perfeccionando su rendimiento o añadiendo más o mejor hardware. Entre los aspectos importantes para garantizar la optimización de una base de datos en PostgreSQL, se destaca la correcta configuración de archivos como:

- **Postgresql.conf:** es el archivo encargado de guardar toda la configuración del gestor y optimizar parámetros como la memoria compartida, la paginación y la caché, vistos como: shared_buffers, cache_miss, wal_buffers, full_page_writes, effective_cache_size. (Martínez, 2007).
- **Afinamiento de consultas:** este optimizador debe garantizar un plan de ejecución que adquiera la efectividad de la consulta y en su comprobación las pruebas toman valor significativo que reflejan el camino de la ejecución de consultas, los tiempos asociados a las mismas y mejoras en caso de que los resultados no respondan a los requisitos requeridos (Pérez González , y otros, 2011).
- **Optimizador genérico de consultas:** no es más que un algoritmo genérico que puede ser habilitado o deshabilitado, incorporado en el gestor PostgreSQL para la optimización de consultas que incorporan la cláusula *JOIN o WHERE* (Tumax , 2012).
- **Autovacuum:** proceso de limpieza encargado de revisar periódicamente las tablas con modificaciones considerables a sus tuplas, recuperando espacio de disco ocupado por las filas modificadas o eliminadas.

1.6 Herramientas para la Gestión de Configuración de Bases de Datos

En la investigación se utilizaron varias herramientas para la gestión de la configuración de bases de datos entre las que se destacan Embarcadero ER/Studio 8.0, SQL Manager 2011 for PostgreSQL, Visual Paradigm 8.0 y Microsoft Visual Studio 2010. A continuación se muestra una revisión más detallada de estas herramientas.

[7]SQL EXPLAIN ANALYZE: es un comando que se utiliza para chequear la exactitud del plan de ejecución, lo muestra en una forma más detallada, lo que el planificador Postgre genera para la consulta dada. Muestra los costos promedio para cada iteración y el costo total.

1.6.1 Embarcadero ER/Studio 8.0

Embarcadero ER/Studio es una solución intuitiva y visual de modelado de datos para el diseño y mantenimiento de bases de datos transaccionales, de ayuda a la toma de decisiones y para la Web. El diseño multi-nivel de ER/Studio permite a los profesionales de bases de datos controlar, documentar y desplegar rápidamente cambios en el diseño en las principales plataformas de bases de datos (Rodríguez Martín, 2011). Entre las funcionalidades que tiene esta herramienta se pueden destacar las siguientes:

- Función de diseño multi-nivel.
- Transformación automática.
- Mapeado de diseños no autorizados.
- Diseño de notaciones estándares.

Además se enuncian funciones en el soporte completo del ciclo de vida de las bases de datos como: la ingeniería inversa, la ingeniería directa y la modificación de las bases de datos.

Por lo planteado se puede llegar a la conclusión de que ER/Studio en su versión 8.0 no solo tolera funcionalidades de soporte, gestión, integración, sino que también permite el diseño de bases de datos de calidad permitiendo funcionalidades como:

- Migración automática de claves externas.
- Asistentes de validación.
- Seguridad/Permisos de modelado.
- Planificación de capacidad.

Esta herramienta posibilitará la creación del modelo de datos de la investigación, expresando de forma gráfica las relaciones entre las tuplas y sus atributos.

1.6.2 SQL Manager 2011 for PostgreSQL

Es una herramienta gráfica, ligera y fácil de usar para la administración de PostgreSQL. SQL Manager 2011 for PostgreSQL funciona para cualquier versión, admitiendo las últimas características de este gestor de bases de datos incluyendo tablas y argumentos de funciones con nombres (Member, 2012). Destaca además, características que hacen de esta herramienta de gestión una opción de uso más codiciada por todos. A continuación se mencionan algunas de ellas:

- Interfaz de usuario gráfica potente y fácil de usar.
- Rápida administración y navegación de base de datos.

- Fácil gestión de todos los objetos de PostgreSQL.
- Herramientas de manipulación de datos avanzadas para la creación de consultas.
- Gestión de la seguridad efectiva.

El desempeño con SQL Manager 2011 for PostgreSQL permite la creación de la base de datos asociada a la investigación, en conjunto con otras herramientas.

1.6.3 Visual Paradigm versión 8.0

Es una de las herramientas UML CASE[8], considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Fue creada para el ciclo de vida completo del software, permitiendo entre diversas funciones la de diseñar e implementar modelos físicos para el trabajo con las bases de datos, creando a su vez la definición a bases de datos a partir de los esquemas de clases (Martínez, 2013). El trabajo con esta herramienta ofrece un diseño agradable del modelo de dominio y expresa de manera dinámica y visual las relaciones entre las entidades que lo componen.

1.6.4 Microsoft Visual Studio 2010 versión 8.0

Es un entorno de desarrollo integrado que soporta varios lenguajes de programación tales como C++, C# y otras extensiones para muchos otros lenguajes. Permite a los desarrolladores crear aplicaciones, sitios web y otros servicios que soporte la plataforma .NET. Microsoft Visual Studio 2012 es la versión más reciente de esta herramienta, acompañada por .NET Framework 4.5 (Criptoy, 2012). Además se debe destacar el uso del Microsoft Visual Studio 2010 en la implementación de la capa de acceso a datos de la investigación.

1.6.5 PgAdmin III

Es una herramienta grafica para gestionar el gestor de base de datos PostgreSQL. Es capaz de gestionar versiones a partir de PostgreSQL 7.3 ejecutándose en cualquier plataforma. PgAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos compuestas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita la administración. Además la herramienta incluye el editor SQL, un editor de código de la parte del servidor, un agente para lanzar script programados, entre otras facilidades. La conexión entre el servidor puede establecerse entre diferentes protocolos o puede encriptarse mediante SSL para mayor seguridad (Pérez Urrutia, 2010).

[8] CASE (Computer Aided Software Engineering): Conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de Software y desarrolladores durante todos los paso de vida de desarrollo de software.

1.6.6 EMS Data Generator for PostgreSQL 2005

Es una poderosa herramienta para generar datos a tablas de bases de datos en PostgreSQL de una vez. La aplicación permite generar tablas y campos para generar datos, configurar valores de rango, cargar archivos para campos de tipo objetos binarios, obtener listas de valores desde consultas SQL y muchas otras características para generar datos de prueba de manera simple y directa. Además provee una aplicación consola, que permite generar datos de un "clic" usando plantillas de generación (SQLManager, 2010).

1.6.7 Apache JMeter versión 2.3.1

Apache JMeter es un software de código abierto, un 100% Java. Es una aplicación de escritorio diseñada con el objetivo de realizar pruebas de cargas funcionales, ensayos de comportamientos e interpretación de los resultados. Originalmente fue diseñado para probar las aplicaciones web, pero desde entonces ha ido evolucionando.

Puede ser utilizado para probar el rendimiento tanto en estática como en dinámica de archivos como: base de datos y consultas, servidores FTP y muchos más. Además puede hacer un análisis gráfico del rendimiento y comportamiento del objeto con cargas pesadas concurrentes (Pérez Urrutia, 2010).

1.7 Tecnologías de programación

En la investigación se decide utilizar diversas tecnologías de programación que faciliten el desarrollo del trabajo, haciéndose un previo estudio de cada una de ellas seguidamente.

1.7.1 Motor de persistencia

Los motores de persistencias, que en el mundo de la programación no son más que un componente de software (una capa de programación), también conocido como "capa de datos", "capa de persistencia" o "correspondencia ORM[9]", son las que permiten establecer una capa intermedia entre el sistema orientado a objetos y las bases de datos relacionales donde se almacenará toda la información del mismo (Palacios, 2012). Esta solución brinda ventajas como:

- Por una parte, se puede programar con orientación a objetos, aprovechando las ventajas de flexibilidad, mantenimiento y reusabilidad.
- Por otra parte, el poder utilizar una base de datos relacional, aprovechándose de su madurez, la estandarización y las herramientas relacionales que hay para ello.

[9] ORM: Object Relation Mapper del inglés. Proceso que permite convertir datos utilizando el lenguaje de programación orientado a objetos, utilizando una base de datos relacional.

La persistencia es la parte más crítica en una aplicación de software. Si la aplicación está diseñada bajo la orientación a objetos, la persistencia se logra con la serialización del objeto o almacenado en una base de datos la información referente. Dentro de los motores de persistencia estudiados se destacan NHibernate y Entity Framework, argumentados más adelante.

1.7.2 NHibernate

Es una herramienta que facilita el mapeo de los atributos entre una base de datos relacional y el modelo de los objetos de una aplicación, que permite establecer estas relaciones. NHibernate (NH) es la conversión de Hibernate de lenguaje Java a C#, para su integración en la plataforma .NET. NHibernate se destaca por:

- Poder utilizar los objetos definidos por el usuario.
- No utilizar técnicas como generación de código a partir de descriptores del modelo de datos.
- Abrir las puertas a utilizar herencia en el modelo de datos.

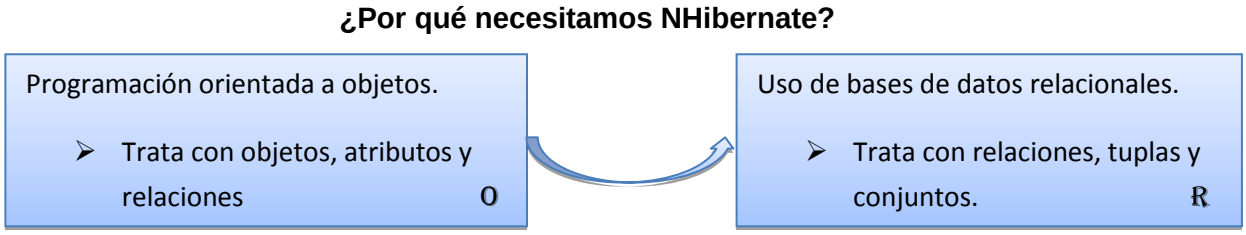


Figura 1: Objeto-Relacional-Mapeado.

El modelo relacional trata con relaciones, tuplas y conjuntos y es muy matemático por naturaleza. Sin embargo, el paradigma orientado a objetos trata con objetos, sus atributos y relaciones. Cuando se desea que los objetos sean persistentes utilizando para ello una base de datos relacional, pues surge un problema de diferencia objeto-relación. Un mapeador objeto-relacional (ORM) ayudaría a evitar esa diferencia, por lo que NHibernate es una vía para solucionar dicho problema.

1.7.3 Configuraciones de Mapeo

Para poder conocer la correspondencia que existe entre los objetos y las tablas, *NHibernate* lo hace por medio de la configuración de mapeo que se puede ver reflejada por dos vías diferentes, como a continuación se enuncian:

- *XML NHibernate*: Una de estas vías para lograrlo consiste en archivos de mapeo XML. Estos archivos poseen la información necesaria para poder saber en qué tablas se

tienen que guardar cada objeto o en qué tabla tengo que buscar para obtener un objeto. Pero esta vía de trabajo nos es tan cómoda como el trabajo con *Fluent NHibernate*, otra alternativa al mapeo de datos.

- *Fluent NHibernate*: Una de las dificultades y que más intimidan a la hora de trabajar con *NHibernate* es crear su fichero de configuración y los ficheros para mapear cada tabla en un objeto. Para hacer el trabajo más fácil se utiliza en la investigación *Fluent NHibernate*, que es una librería que se encarga de llevar a cabo todo el trabajo que tendríamos que hacer a mano para conseguir mapear las tablas en objetos. *Fluent NHibernate* ofrece una alternativa al estándar de los archivos de asignación *NHibernate XML*. En lugar de escribir documentos en XML, *Fluent NHibernate* permite escribir fuertes asignaciones en código C#. Esto permite una fácil refactorización y un código más conciso.

¿Cómo funciona el trabajo con Fluent NHibernate?

- Sus asignaciones o mapeos se mueven a código real, es decir, que están compilados junto con el resto de la aplicación.
- Cambiar el nombre de sus refactorizaciones alterará sus asignaciones y el compilador colapsará.
- Dispone de un sistema de configuración convencional, donde se pueden especificar patrones para anular las convenciones de nomenclatura, entre otras cosas.
- Establece cómo las cosas deben ser nombradas una vez, luego *Fluent NHibernate* hace el resto.

1.7.4 Entity Framework

Framework de Microsoft ADO.NET Entity es una plataforma para la programación con los datos que eleva el nivel de abstracción del nivel relacional al conceptual (entidad), y por lo tanto reduce significativamente las diferencias entre aplicaciones y servicios centrados en datos (Microsoft, 2013). Las aplicaciones de *Entity Framework* ofrecen las siguientes ventajas:

- Las aplicaciones pueden funcionar en términos de un modelo conceptual más centrado en la aplicación que incluyendo herencia, miembros complejos y relaciones.
- Las aplicaciones están libres de dependencias de codificación rígida de un motor de datos o de un esquema de almacenamiento.
- Las asignaciones entre el modelo conceptual y el modelo específico de almacenamiento puede cambiar sin tener que cambiar el código de la aplicación.

- Los programadores pueden trabajar con un modelo de objeto de aplicación coherente que se puede asignar a diversos esquemas de almacenamiento.
- Se pueden asignar varios modelos conceptuales a un único esquema de almacenamiento.

Una vez estudiadas estas dos tecnologías, *NHibernate y Entity Framework*, se puede llegar la conclusión, de que basándose en las características propias de cada una de ellas, se utiliza NHibernate como herramienta adoptada para el desarrollo de la investigación, ya que ayuda a tener las particulares propias de la orientación a objetos para el trabajo con las bases de datos, además de permitir cambiar de gestor de base de datos sin tener que realizar cambios en la aplicación.

1.7.5 Lenguaje de Programación C#

C# es un lenguaje de programación que, desde su creación, trata de implementar las mejores características de los lenguajes C, C++ y Java; debido a un diseño mejorado y su fuerte tipado (los tipos deben ser declarados en tiempo de compilación) se ha convertido en uno de los lenguajes más utilizados en diferentes ambientes, pasando por Windows Forms hasta complejos desarrolladores en ambientes web, con tecnologías Asp.Net. C# mantiene diversas características, que a continuación se mencionan:

- Sencillez de uso: es uno de los lenguajes más fáciles y prácticos que existen actualmente en el mercado.
- Orientados a objetos: desde su concepción, C# estuvo diseñado con paradigmas de la orientación a objetos.
- Recolección de basura: una de sus fortalezas más destacadas es su importante tecnología en la recolección de objetos que ya no se utilizan en los hilos de programación.
- Seguridad de tipos: mantiene un fuerte tipado los cuales ayudan a encontrar errores que pueden ser difíciles de encontrar.

Estas características que se han mencionado, son las más importantes, pero existen más ventajas por las cuales se ha seleccionado este lenguaje.

- Puede correr en diferentes plataformas, como en Linux y Windows.
- C# incorpora los estándares ECMA-334 y ECMA-3355.
- Similitud de sentencias entre C++ y java, hacen que sea más sencillo el aprendizaje del mismo, lo que agiliza el tiempo de desarrollo de los proyectos.

- Existen innumerables librerías para el desarrollo con diferentes tipos de Base de Datos (MySql, Oracle, PostgreSQL).

1.7.6 Lenguaje de Programación PL/pgSQL

PostgreSQL puede tener disponibles varios lenguajes de programación, uno de estos es PL/pgSQL. Este lenguaje es muy parecido a PL/SQL utilizado por Oracle y es uno de los mejores lenguajes de procedimientos que podemos usar en PostgreSQL, es más fácil de aprender y muy potente. Con la creación de PL/pgSQL se establecieron objetivos para su desarrollo, los cuales son:

- Poder ser usado para crear funciones y disparadores (triggers).
- Añadir estructuras de control al lenguaje SQL.
- Poder realizar cálculos complejos.
- Heredar todos los tipos, funciones y operadores definidos por el usuario.
- Fácil de usar.

Con el paso del tiempo, desde sus inicios, estos objetivos han ido creciendo, hasta los días de hoy que se le adicionan nuevas metas para que su desarrollo sea más preciso y completo. Esto significa que este lenguaje es de cierta forma superior al SQL, ya que se pueden agrupar bloques de cálculos y varias series de consultas dentro del servidor de base de datos, obteniendo de esta manera el poder de un lenguaje procedimental, no siendo de igual forma si se utilizan las sentencias SQL, ya que deben ser ejecutadas individualmente por el servidor. Además, PL/pgSQL ahorra una gran cantidad de tiempo debido a que no tiene la sobrecarga de la comunicación completa cliente/servidor.

1.7.7 Lenguaje de programación PL/SQL

PL/SQL es el lenguaje de programación que proporciona Oracle para extender el SQL estándar con todo tipo de instrucciones. Es un lenguaje procedimental para trabajar con las base de datos, que soporta todos los comandos de consulta y manipulación de datos, aportando al SQL las estructuras de control y diversos elementos propios del lenguaje. Además, es un lenguaje estructurado en bloques, donde dichos bloques se almacenan como objetos de la BD que pueden ser ejecutados repetitivamente. (Faci, 2004).

También el uso de paquetes que pueden contener procedimientos y funciones y que a su vez pueden ser ejecutados desde otros subprogramas. Se incluye también dentro del PL/SQL los disparadores (triggers). Como parte del proceso de desarrollo, Oracle da la posibilidad de programar de forma modular, clara y eficiente, permitiendo no sólo mejorar la calidad de diseño

de la aplicación, sino también optimizar el desempeño de la misma (Alcalde, 2013). A continuación se destacan algunas de las ventajas que ofrece este lenguaje:

- Permite modulizar el diseño de la aplicación.
- Otorga flexibilidad al momento de diseñar la aplicación.
- Permite ocultar los detalles de implementación.
- Agrega mayor funcionalidad al desarrollo.
- Introduce mejoras en el rendimiento.

1.8 Metodología de desarrollo de software

En un proyecto de desarrollo de software la metodología define quién debe hacer qué, cuándo y cómo debe hacerlo. Las características de cada proyecto exigen que el proceso sea configurable. Estas imponen un proceso disciplinado sobre el desarrollo del software con el propósito de hacerlo más eficiente y previsible. Seguidamente se expone un estudio de la metodología elegida para guiar el desarrollo de la investigación.

1.8.1 Metodología Desarrollo Basado en Rasgos o Funciones (FDD)

El Desarrollo Basado en Rasgos o Funciones (FDD por sus siglas en inglés) es un enfoque ágil para el desarrollo de sistemas. Se enfoca en iteraciones cortas que entregan funcionalidad tangible (Collorana, 2009). Esta metodología consta de cinco procesos los cuales se mencionan a continuación:

- Desarrollar un Modelo Global.
- Construir una Lista de los Rasgos.
- Planear por Rasgo.
- Diseñar por Rasgo.
- Construir por Rasgos.

Los primeros tres procesos se hacen al principio del proyecto. Los últimos dos se hacen en cada iteración. Cada proceso se divide en tareas y se da un criterio de comprobación. Los programadores jefe son los desarrolladores más experimentados, actúa como coordinadores, diseñadores líderes y mentores, mientras los dueños de clases hacen gran parte de la codificación del rasgo. Lo estudiado de esta metodología de desarrollo de software destaca las siguientes ventajas:

- El equipo de desarrollo no malgasta el tiempo y el dinero del cliente desarrollando soluciones innecesarias generales y complejas que en realidad no son un requisito del cliente.

- Cada componente del producto final ha sido probado y satisface los requerimientos.
- Rápida respuesta a cambios de los requisitos a lo largo del desarrollo.
- Entrega continua y en plazos cortos de software funcional.
- Trabajo conjunto entre el cliente y el equipo de desarrollo.

Se concluye puntualizando que toda metodología debe ser adaptada al contexto del proyecto (recursos técnicos y humanos, tiempo de desarrollo, tipo de sistema), exigiendo un esfuerzo considerable para ser adaptadas, sobre todo en proyectos pequeños y con requisitos muy cambiantes. Las metodologías ágiles ofrecen una solución casi adecuada para una gran cantidad de proyectos.

1.8.2 Gestión de cambios en las bases de datos

La gestión de cambios en bases de datos es el proceso que determina qué cambios se tienen que realizar en una BD, especificarlos, evaluarlos y desplegarlos. El hecho de modificar objetos de una base de datos, independientemente de su tipo, no suele ser una operación trivial. Los cambios suelen afectar a objetos dependientes y en ocasiones a los datos subyacentes. El proceso de analizar y mantener estas dependencias se ve reflejado en la realización de tres tareas fundamentales para lograr el éxito (IBM, 2013).

Realizar estas tres tareas es de gran importancia para lograr una visión general de la gestión de cambios en una base de datos. Reflejadas dichas tareas en la **Figura 2**, ilustran una secuencia de pasos necesarios para la gestión del cambio.

Figura 2: Pasos necesarios para la gestión de cambios.

El primer paso es la especificación de cambios, proceso en el cual se definen un conjunto de cambios que se desean aplicar a la base de datos. Esta fase incluye definir los cambios que se desean realizar, analizar el impacto que tendrán y generar las operaciones para implementar los cambios en la base de datos. Además de especificar dichos cambios, se ejecutan y en ocasiones necesarias, se eliminan los mandatos de cambios.

El segundo paso a realizar es el despliegue de los cambios. Este proceso tiene como objetivo cambiar la base de datos para reflejar las condiciones especificadas en la fase anterior. El despliegue comprende el proceso de aplicar los cambios en la base de datos real, y si es necesario deshacer los cambios. Además, este paso trae consigo la realización de un documento o historial de cambios, incluyendo script, modelos, registros e informes para capturar el cambio desde su inicio.

Finalmente se realiza una auditoría con el propósito de ver si un cambio se ha realizado para comprender la evolución del modelo de base de datos. Este paso es importante ya que permite realizar un seguimiento de quién ha realizado el cambio, ayudándole a evitar el acceso desconocido a los datos.

Para una mayor profundización acerca del tema expuesto ver Documento de Gestión de Cambios en la BD, archivada dentro del Expediente de Proyecto referente al trabajo de diploma.

Conclusiones parciales

Con el avance de las tecnologías se han ido incrementando las herramientas que de una forma u otra contribuyen a mejorar el almacenamiento de la información de manera más estructurada. Actualmente se considera que los SGBD han ido mejorando los servicios que brindan garantizando, entre otras cosas, el acceso de la información.

En este capítulo se hizo un estudio de la información referente a los SGBD, escogiendo por sus potentes características y facilidad de uso a PostgreSQL como el gestor apropiado para la creación de la BD, incluyendo PL/pgSQL como lenguaje de programación, además de usar otras tecnologías que profundizan la fundamentación del trabajo. Como metodología para guiar el desarrollo de la investigación, se utilizará FDD, aportando funcionalidades tangibles. La base de datos además debe estar diseñada para interactuar con varios gestores de base de datos, entre los que se priorizará al gestor Oracle.

CAPÍTULO 2: Diseño de la base de datos y propuesta de solución.

Introducción

En este capítulo se propone una solución al problema de la investigación, utilizando la metodología de desarrollo de software Basado en Rasgos o Funciones (FDD). Las fases con que cuenta esta metodología mostrarán la evolución de la solución. Además se generarán los entregables: modelos de dominio, modelo de datos del sistema y diccionario de datos.

2.1 Descripción de la propuesta de solución

Se propone la elaboración de una base de datos que acumule tola la información referente al proceso de identificación de un ciudadano o usuario de la UCI. Dicho proceso se realiza mediante citas o solicitudes de creación de un documento de identificación a un ciudadano, el cual valide su relación y cobertura por la institución. Todas las identificaciones a elaborar son diferentes, así como también los roles que se les asignan a los usuarios de la universidad para la obtención de permisos específicos del rol asociados.

El proceso de captación de datos es el paso siguiente que el ciudadano debe seguir para la realización de este documento. El proceso finaliza con la entrega del documento al usuario correspondiente, previamente supervisado y personalizado. Se propone además, la implementación de la capa de persistencia de datos, posibilitando la integración entre el negocio y la base de datos. Para dar cumplimiento a la solución propuesta, se elaboran a lo largo de la investigación modelos que facilitan el diseño de la BD y que más adelante son expuestos.

2.2 Modelo de dominio o conceptual

El modelo de dominio puede ser visto como el punto de partida para el diseño del sistema. Como objetivo principal, este modelo tiene las funciones de captar y expresar el entendimiento en un área bajo análisis. Además, es utilizado por el analista como un medio para comprender el sector de negocios al cual el sistema va a servir. La **Figura 3** muestra la representación del modelo de dominio.

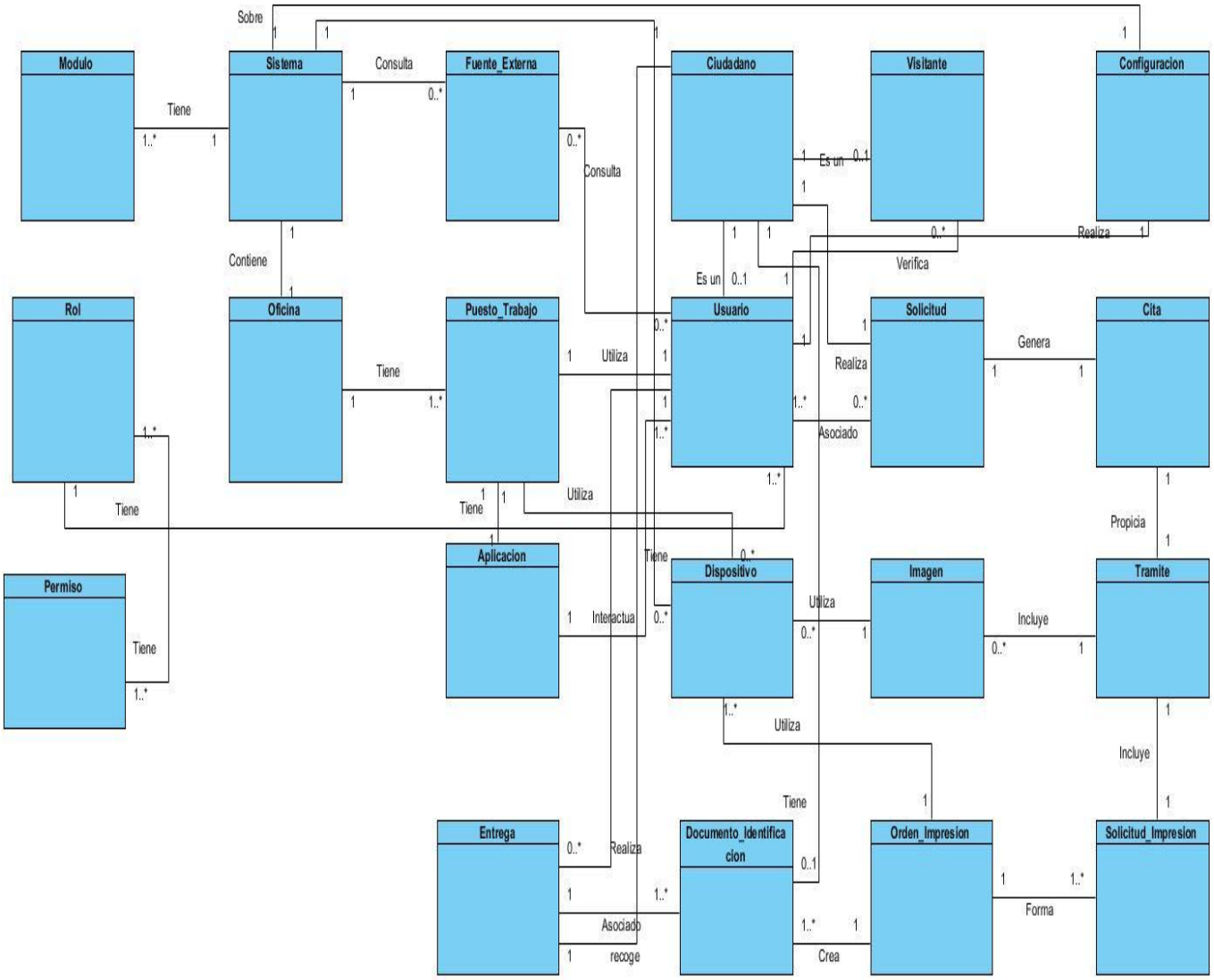


Figura 3: Representación del modelo de dominio.

De dicho modelo, los conceptos más significativos a tratar son:

- **Sistema:** Programa informático donde se integran todos los módulos con la base de datos, teniendo como objetivo informatizar la Plataforma Modular de Identificación y Control de Acceso de la UCI.
- **Rol:** Un rol es una manera de agrupar usuarios acordando a qué partes de la aplicación tienen acceso. Este acceso es asignado o revocado a través de la administración en dependencia de lo que deba hacer en ese momento.
- **Entrega:** Acción en la que el ciudadano obtiene su documento de identificación.
- **Permiso:** Son asignados a los usuarios con el objetivo de dar acceso a diferentes recursos en dependencia del rol que posea el mismo. En el Modelo de Control de Acceso Basado en Roles (RBAC) los permisos disponibles para un usuario son el

conjunto de permisos asignados a los roles que están activados en todas las sesiones del usuario, sin tener en cuenta las sesiones establecidas por otros usuarios en el sistema.

- **Fuente externa**: Lugar de donde se obtendrán datos a utilizar en el sistema. Puede ser una base de datos, fichero, servicio web, entre otros.
- **Aplicación**: Producto con el que interactúan los usuarios para el proceso de emisión de documentos. Accesible o instalada desde los puestos de trabajo y configurable para cualquier organización que precise de ella.
- **Ciudadano:** Es cualquier persona que de alguna forma esté vinculado a la institución que necesite un documento de identificación para acceder a la misma.
- **Usuario:** Es el nombre único que recibe la persona cuando es registrada en el sistema a través del cual podrá autenticarse y dependiendo de sus roles, podrá efectuar determinadas operaciones y tener acceso a los diferentes recursos, en función de los permisos que le sean asignados.
- **Dispositivo:** Dispositivos conectados al sistema (cámara, webcam, lector de huellas, lector de firma digital, impresora y cualquier otro dispositivo que sea necesario en el proceso de captura de imágenes).
- **Documento de identificación:** Es lo que va a permitir identificar al ciudadano.
- **Visitante:** Es una persona a la cual se le autoriza a acceder a la institución pero no posee un documento de identificación, a esta persona se le entrega un pase provisional.
- **Solicitud:** Necesidad del ciudadano de la creación de un documento de identificación planteada de manera formal (escrita o digital).
- **Imagen:** Es donde se capturan las imágenes de las personas a las cuales se le está haciendo el trámite.
- **Orden de impresión:** Es uno de los pasos a seguir a la hora de realizar un trámite, este paso permite la personalización y creación del documento de identificación.
- **Configuración:** Es la configuración con que cuenta el sistema.
- **Cita:** Es la convocatoria que se le hace a una persona con la fecha para comenzar el trámite para la creación de su documento de identificación.
- **Trámite:** Es donde se realizan los procesos de captura de datos, captura de imágenes, supervisión, personalización, control de calidad y entrega y se verifica que todos estos procesos se hayan realizado de forma efectiva.
- **Solicitud de impresión:** Son las solicitudes que hace el sistema para la creación de un documento de identificación.

2.3 Modelo de datos

Es una representación abstracta de los datos de una organización y las relaciones entre ellos, es decir, en cierta medida un modelo de datos describe una organización (Salazar, 2012) (en este caso la organización representa la PMICA). De forma lógica se podría encaminar por dos vías:

- Formalización: Permite definir formalmente las estructuras permitidas y sus restricciones a fin de representar los datos, además, de establecer las bases para un lenguaje de datos.
- Diseño: Este modelo es uno de los elementos básicos en el diseño de una base de datos.

En las **Figuras (4, 5 y 6)** se puede ver representada de forma parcial o por submodelos de elaboración la propuesta del modelo de datos de la investigación.

El primer paso que se lleva a cabo es el proceso de gestión de cita o solicitud, donde un ciudadano solicita la creación de un documento de identificación y la administración correspondiente. Posteriormente se pasa a la captación de datos de dicho ciudadano, que contribuye al proceso anterior; finalizando con la entrega del documento de identificación requerido.

En el **Anexo1** se muestra la relación en conjunto de todos estos procesos, expresados mediante el modelo de datos de la solución propuesta.

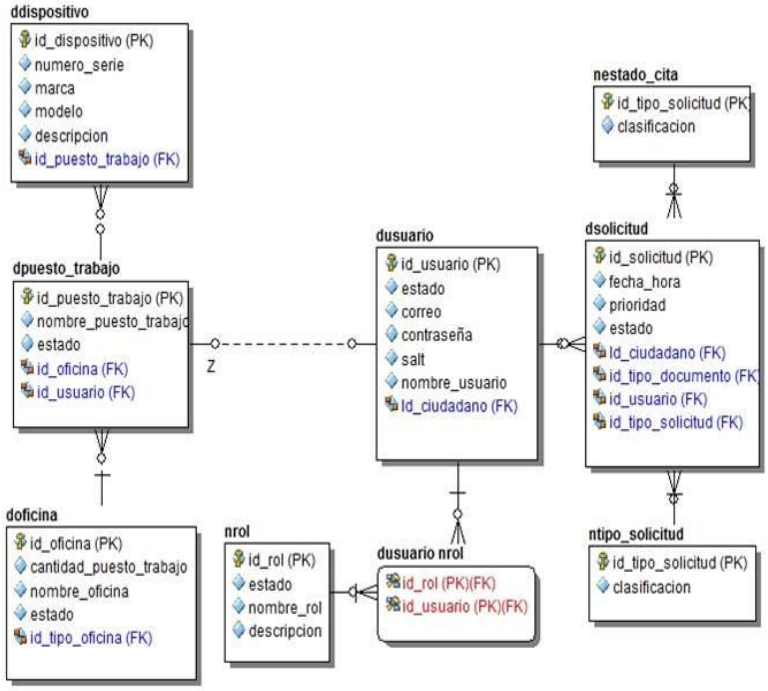


Figura 4: Submodelo Administración y Gestión de Cita o Solicitud.

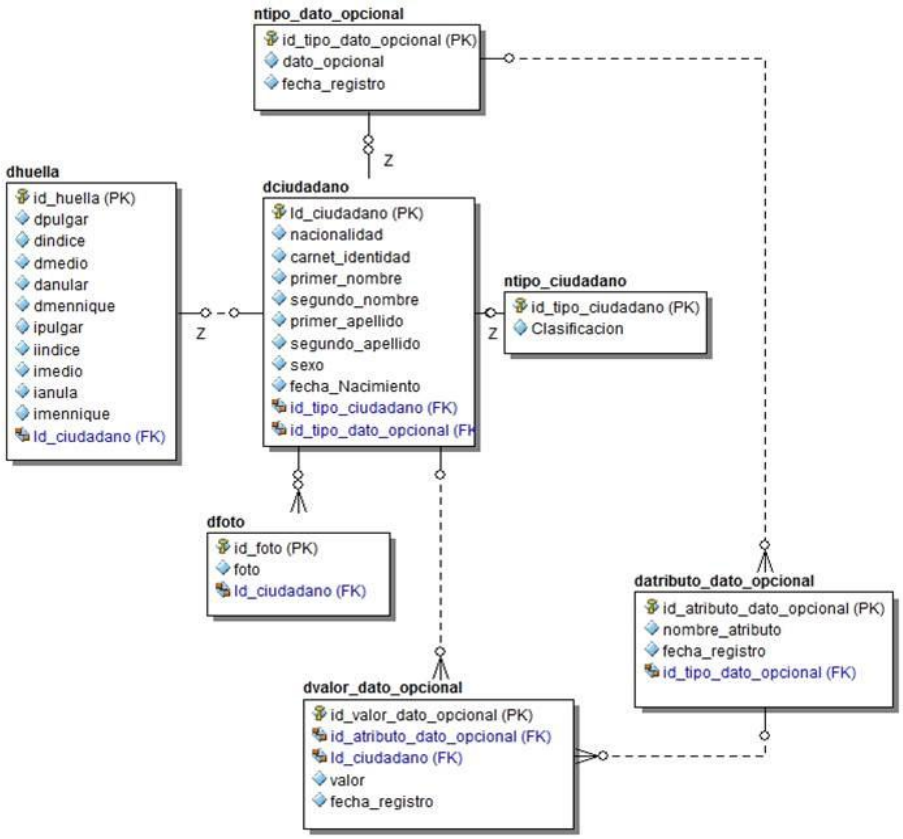


Figura 5: Submodelo Captación de Datos.

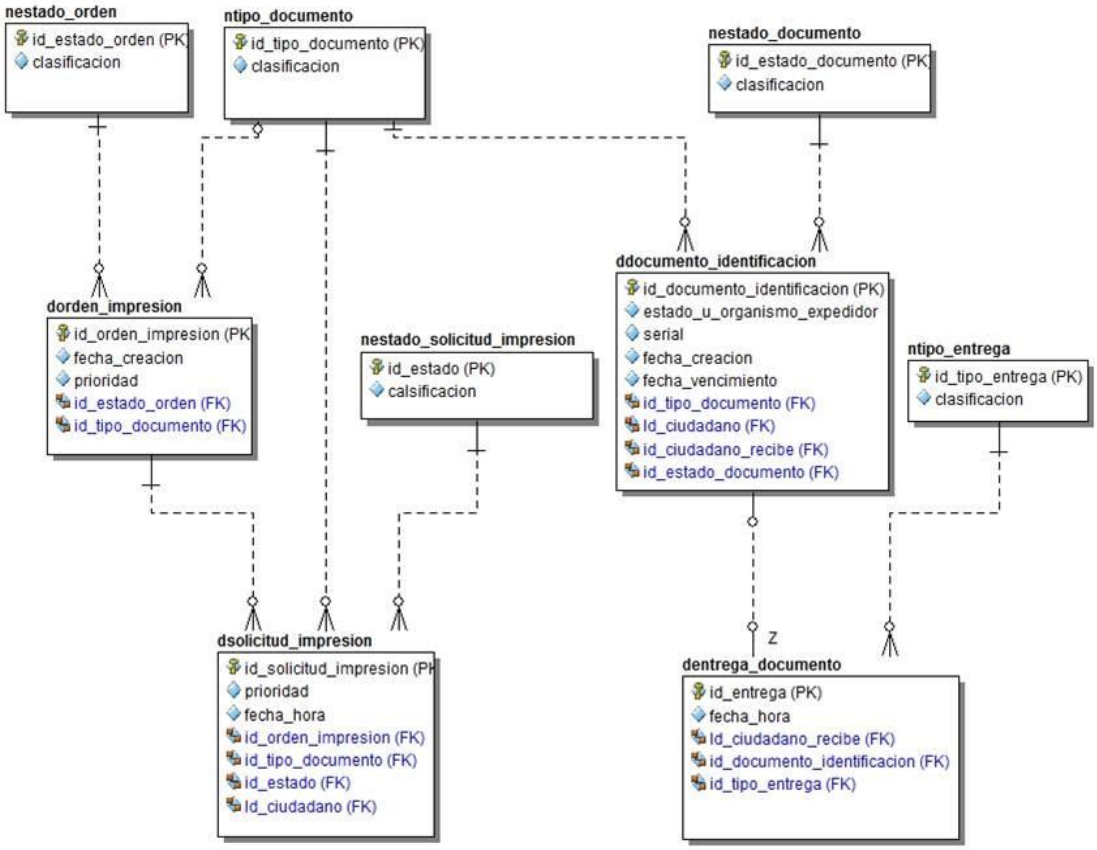


Figura 6: Submodelo Supervisión y Entrega.

2.4 Diccionario de datos

"Un diccionario de datos es una herramienta de importancia para el administrador de la base de datos, es un catálogo accesible para el usuario de datos relacionados con la base de dato." (David M. Kroenke). El diccionario de datos almacena información acerca de la estructura de las bases de datos y las relaciones entre dichos datos. En el **Anexo 2** se representan las principales descripciones de este diccionario de datos.

2.5 Nomenclatura del modelo de datos

En el diseño de la base de datos se utilizó una nomenclatura capaz de mejorar la comprensión y organización de los elementos del modelo. El nombre de las tablas será descriptivo y estará escrito con letra en minúscula, además de estar en singular. De ser un conjunto de palabras las que componen el nombre de la tabla, se escribirán cada una de ellas en minúscula, separadas por el carácter guión bajo "_". Una tabla hija llevará el nombre de la tabla padre delante. Los prefijos están descritos en la siguiente tabla:

Prefijo	Elemento
d(nomb_tabla)	Tabla de datos
n(nomb_tabla)	Tabla nomencladora
id_nomb_tabla	Llave primaria
Id_tabla_referenciada	Llave foránea

Tabla 1: Nomenclatura del modelo de dato.

Los nombres de las tablas que almacenan datos o información propia de los ciudadanos comienzan con el prefijo *(d),* un ejemplo es: *dciudadano*, *dusuario, drol*, entre otras que se muestras en modelo de datos. Además se pueden observar tablas nomencladoras como: *ntipo_atributo, ntipo_documento, ntipo_solicitud,* a las cuales se les antecedió el prefijo *(n)* antes del nombre de la entidad. Los nombres de las llaves primarias y foráneas están compuestos por un conjunto de letras, separadas por guión bajo, antecediéndole a cada una de ellas el prefijo *(id)* y escritos en letra minúscula, como por ejemplo: *id_ciudadano*, que representa el identificador único por el cual se distingue cada ciudadano; *id_atributo_dato_opcional*, representando en este caso la clave única de la tabla *datributo_dato_opcional.*

2.6 Normalización de la BD

En la elaboración del diseño de la base de datos, el proceso de normalización juega un papel fundamental, pues es el encargado de evitar toda redundancia existente en la información, protege la integridad de los datos ante cualquier acción realizada en las tablas, permitiendo optimizar los procesos de gestión de la información almacenada. A la hora de aplicar la normalización a la BD, se trató de que cada entidad cumpliera con los controles y restricciones establecidas para cada una de las formas normales, alcanzándose una normalización correcta en todas las entidades.

De manera general el modelo de datos se encuentra normalizado hasta la 3FN, como se muestra en el **Anexo1**. Estas entidades cumplen con las restricciones establecidas por la 1FN, pues posee una llave primaria y sus dominios no tienen elementos que sean conjuntos. Cumplen además con las restricciones de la 2FN, ya que toda relación que se encuentra en 1FN y que además su llave primaria no sea compuesta, se encuentra en 2FN y también se encuentran en 3FN pues no existen dependencias transitivas entre los atributos no llaves.

Sin embargo, en el modelo existen entidades que por cuestiones de rendimiento se encuentran en 2FN. Un ejemplo de esto es la entidad *dentrega_documento*, la cual no cumple con las restricciones propuestas por la 3FN, que plantea que la relación debe de estar en 2FN y además se deben eliminar las dependencias transitivas entre los atributos respecto a la llave

primaria. La **Figura 7** muestra la dependencia transitiva que existe entre los atributos *id_ciudadano_recibe (FK)* e *id_documento_identificacion (FK)*, debido a que un documento de identificación tiene un ciudadano encargado de recoger este documento, y con el objetivo de ganar en rendimiento, se decidió incluir en la tabla *dentrega_documento* el atributo *id_ciudadano_recibe,* para de esta forma no tener que recurrir a la tabla *ddocumento_identificacion* para obtener la información personal del ciudadano que recibe el documento de identificación implicando otra búsqueda sobre otra tabla del modelo.

Figura 7: Dependencia transitiva entre atributos.

De esta forma queda evidenciado cómo la base de datos para la PMICA queda normalizada de manera general hasta la 3FN, permitiendo un mínimo de redundancia y una correcta optimización de las consultas.

2.7 Restricciones del modelo

Este modelo cumple con un grupo de restricciones con el objetivo de apoyar la integridad de la base de datos:

- La integridad de la entidad se garantiza para todas aquellas entidades que poseen más de un atributo como llave primaria, pues ninguno de ellos puede tomar valores nulos.
- No existe más de una tupla perteneciente a una relación con igual llave.
- Todas las relaciones entre entidades se encuentran sujetas a referencias entre entidades del modelo. Con ello se garantiza que una llave foránea apunta a una entidad con una tupla, fila o registro existente dentro de la base de datos, permitiendo minimizar la redundancia de los datos y con ello aumentar la integridad de los mismos.

2.8 Indexación

Un índice en una estructura auxiliar que permite acelerar el rendimiento de las búsquedas siempre que la consulta tenga condiciones asociadas a los índices y puedan ser creados utilizando una o más columnas de una tabla, aunque una incorrecta estrategia de indexado puede repercutir negativamente en el rendimiento, pues para aquellas tablas que posean un

número pequeño de tuplas, resulta más costoso realizar una búsqueda indexada que secuencial, además las operaciones de inserción y actualización se tornan más lentas, ya que requieren de una modificación del índice. (Alarcón Medina, 2006). PostgreSQL promete cuatro tipos de índices: GIST, B-Tree, Hash y GIN. Cada uno de ellos utiliza un algoritmo diferente para determinados tipos de consultas. Por defecto PostgreSQL utiliza los índices del tipo B-Tree por ser el más utilizado.

En el modelo de datos propuesto para la PMICA, el tipo de índice utilizado fue el B-Tree por su fácil manipulación, indexando aquellas columnas que pueden llegar a contener un alto número de registros y sobre las cuales se realizan más operaciones de búsquedas, haciendo un uso eficiente del índice. Dentro de los campos indexados, primeramente se destacan las columnas que a la hora de la creación se clasificaron como atributos únicos *(UNIQUE),* además de la indexación que realiza PostgreSQL automáticamente cuando se define una llave primaria.

Luego como parte de un análisis que se hace para verificar el rendimiento de la base de datos y la creación de índices innecesarios, se decide dejar los índices que proporciona el gestor y colocar otros en las tablas de mayores registros. Un ejemplo es la tabla *dusuario*, donde se indexó el atributo *nombre_usuario,* y de la tupla *dsolicitud* el campo *fecha_hora* fue el indexado. Este simple proceso permitirá a la Plataforma Modular un camino de búsqueda más rápido ante eventos de gran cúmulo de datos.

2.9 Patrones de Diseños

Un patón de diseño es una descripción de las clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular (García, y otros, 2010). Dichos patrones exponen los siguientes beneficios:

1. Son soluciones concretas:
    a. Un catálogo de patrones es un conjunto de recetas de diseño.
    b. Aunque se pueden clasificar, cada patrón es independiente del resto.
2. Son soluciones técnicas:
    a. Dada una determinada situación, los patrones indican cómo resolverla.
    b. Existen patrones específicos para un lenguaje determinado, y otros de carácter más general.
3. Se aplican en situaciones muy comunes:
    a. Proceden de la experiencia.
4. Son soluciones simples:
    a. Indican cómo resolver un problema particular utilizando un pequeño número de clases relacionadas de forma determinada.

b. No indican cómo diseñar un sistema completo, sino sólo aspectos puntuales del mismo.

5. Facilitan la reutilización de las clases y del propio diseño:
    a. Los patrones favorecen la reutilización de clases ya existentes y la programación de clases reutilizables.
    b. La propia estructura del patrón es reutilizada cada vez que se aplica.

En la investigación se destacan patrones de diseño como: el patrón entidad-atributo-valor (EAV) manifestando sus múltiples usos y sus grandes ventajas; patrón asociación, el patrón modelo para la seguridad de aplicaciones y el patrón de llaves subrogadas. También se destacan otros como el patrón repositorio, el patrón iterador, instancia única y el patrón unidad de trabajo caracterizados a continuación.

2.9.1 Patrón entidad-atributo-valor (EAV)

El modelo entidad-atributo-valor es la representación de un modelo flexible donde se pueden representar objetos con sus atributos, es un acercamiento al modelo orientado a objeto representado en el modelo relacional, donde se representan clases, atributos de las clases, instancias de las clases, y los valores de cada atributo para cada objeto dado. (Strappazzon, 2010). Se emplea este patrón en la tupla ciudadano, donde un ciudadano puede tener datos opcionales. (Ver **Figura 8**).

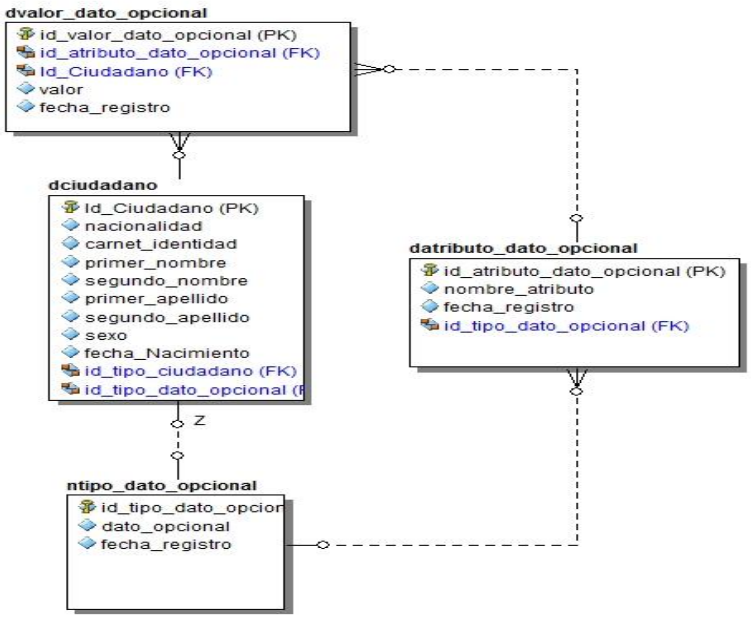


Figura 8: Patrón entidad -atributo-valor. .

2.9.2 Patrón asociación

Los patrones de asociación simbolizan las relaciones entre entidades. Estos fueron utilizados para representar las asociaciones de tipo muchos a muchos. Este tipo de asociación no es soportada por las bases de datos relacionales, por lo que una de las vías de solución a dicha traba es la agregación de una tabla de asociación a dicho modelo, cumpliendo la función de representar este tipo de relación entre varias relaciones. La tabla resultante estará compuesta por las llaves primarias de las entidades a relacionar.

En el modelo de base de datos que se propone, este patrón se pone de manifiesto a la hora de relacionar usuarios y roles del sistema, ya que un usuario puede tener uno o varios roles, mientras que un rol puede pertenecer a más de un usuario. Su representación se muestra en la **Figura 9.**

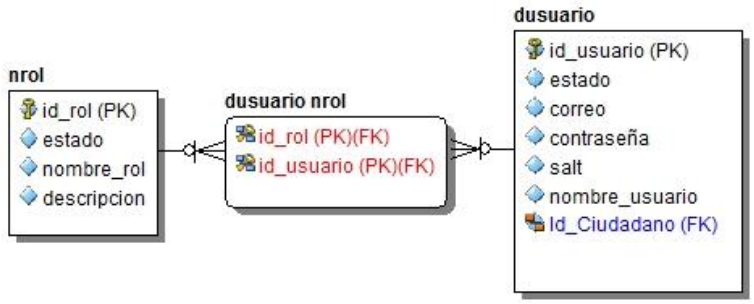


Figura 9: Patrón asociación.

2.9.3 Patrón modelo para la seguridad de aplicaciones

Este patrón utiliza el modelo de control de acceso basado en roles (RBAC). A través de él, los permisos para ejecutar ciertas operaciones son asignados a roles específicos, y estos a su vez, son asignados a los usuarios del sistema, es decir, que el manejo de los permisos de cada usuario está determinado por la asignación del rol apropiado. Este patrón se pone de manifiesto en el ejemplo del patrón anterior, donde a cada ciudadano o usuario se le es asignado un rol específico de acuerdo a la función que vaya a realizar, y se le asignan permisos requeridos a ese rol.

2.9.4 Patrón de llaves subrogadas

Este patrón es utilizado cuando se desea generar una llave primaria única para cada entidad. Generalmente en los modelos de datos se utilizan números enteros en columnas auto-

incrementales o en este caso identificadores del tipo UUID [10]. Este patrón es muy utilizado por las entidades que integran el modelo de datos, ya que representan en su mayoría identificadores del tipo UUID.

2.9.5 Patrón repositorio

Este patrón no es más que una fachada que abstrae el dominio(o capa de lógica del negocio) de la persistencia, es decir, es una clase que servirá como intermediario entre la aplicación y los datos, ofreciendo una interfaz *CRUD (Add, Update, Delete)* (siendo éstos los métodos que se utilizan en la investigación) (Escobar, 2013). La idea de este patrón es que un objeto repositorio actúe como una colección de objetos en memoria, y a que a su vez se puedan añadir o quitar elementos y además realizar búsquedas filtradas.

2.9.6 Patrón iterador

Teniendo en cuenta lo propuesto por el anterior patrón, se hace uso además del patrón iterador. El propósito de este patrón es proporcionar acceso secuencial a los elementos de un objeto agregado sin tener que exponer su representación interna. Permite además, la creación de una interfaz uniforme para recorrer distintos tipos de listas o arreglos (Trillo Lado, 2010).

2.9.7 Patrón instancia única

Este patrón tiene como propósito asegurar que una determinada clase solo tenga una instancia única y que proporcione un punto de acceso a dicha instancia. Tanto el patrón iterador, como el patrón instancia única forman parte de los patrones *Gang of Four* (por sus siglas en inglés GoF) o (pandilla de los cuatro) brindando un modelo elegante y robusto. (Larman, 1999). Estos dos patrones forman parte de la clasificación de los patrones GoF, según su propósito, siendo el iterador un patrón de comportamiento, que refleja la forma en que interactúan diferentes objetos o clases distribuyendo sus responsabilidades. Por otra parte, instancia única es un patrón de creación que cuenta con el propósito de crear nuevos objetos de forma transparente al sistema (Trillo Lado, 2010).

2.9.8 Patrón unidad de trabajo

El modelo de unidad de trabajo mantiene una lista de los objetos afectados por una transacción empresarial y coordina la escritura fuera de los cambios y la resolución de problemas de simultaneidad (Fowler, 2005). Dicho en otras palabras, la unidad de trabajo se ocupa de controlar los cambios que sufren las entidades, es decir, todas las altas, bajas y modificaciones que se deben producir en las bases de datos después de hacer una acción.

[10] Un UUID (Universally Unique Identifiers) es un entero de 128 bits representado en notación hexadecimal que se puede utilizar siempre que se requiera de un identificador único.

Conclusiones parciales

Con el transcurso del tiempo se ha hecho cada vez más necesario e imprescindible el uso de las base de datos, permitiendo el desarrollo y aumento de tecnologías para el tratamiento de las mismas. El proceso de diseño de una BD se puede presentar como un proceso complicado, que pasa por varias etapas hasta conseguir el objetivo perseguido.

Con el diseño de la base de datos propuesta para la Plataforma Modular de Identificación y Control de Acceso de la UCI, se detallaron los principales conceptos de los modelos elaborados, identificándose un total de 26 entidades, con ayuda del ER/Studio y Visual Paradigm como herramientas para el modelado de datos. El uso de patrones de diseño facilitó un desarrollo cómodo del diseño de la base de datos, adquiriendo en cada caso los conocimientos de cada patrón.

Al culminar este capítulo se obtuvo, entre otros resultados, el modelo de datos de la investigación y la base de datos correspondiente, la cual se logró normalizar, extrayendo las dependencias funcionales y a partir de estas se normalizó hasta llegar a la 3FN, evitando todos los problemas de redundancia que podrían existir, dando paso a la implementación de la capa de persistencia de datos en el próximo capítulo.

CAPÍTULO 3: Implementación y prueba de la solución propuesta.

Introducción

Una vez alcanzado el diseño correcto de la BD, se instala el sistema gestor de base de datos PostgreSQL y se realizan las configuraciones pertinentes, para lo que es de vital importancia definir el ambiente donde se desplegará el software. Por otra parte, la seguridad es un aspecto fundamental en las bases de datos, ya que la información puede extraviarse o modificarse, es por eso que es necesario tener en cuenta estos aspectos, tanto en el diseño como en la implementación. Las pruebas permiten comprobar si un sistema se comporta según lo esperado, por lo que aplicando pruebas de integridad al modelo y a la solución propuesta, permitirán verificar su funcionamiento y validez una vez terminada la implementación.

3.1 Propuesta de despliegue para la solución

El despliegue es la etapa de desarrollo que describe la configuración del sistema para su ejecución en un ambiente del mundo real. El modelo despliegue muestra la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Un nodo no es más que un elemento físico que existe en tiempo de ejecución y representa un recurso computacional. Estos a su vez, se utilizan para modelar la topología del hardware sobre el que se ejecuta el sistema.

Dichos nodos son los elementos donde se ejecutan y se representa el despliegue físico de los componentes. Por otra parte, los componentes no son más que elementos que participan en la ejecución de un sistema, representando el empaquetamiento físico de los elementos lógicos (Rubiano, 2009).

La relación entre un nodo y el componente de despliegue puede mostrarse con una relación de dependencia, generalización o asociación de nodos desplegados en un compartimiento adicional dentro del nodo. El diagrama mostrado en la **Figura 10** representa la distribución de los nodos físicos a utilizar en la Plataforma Modular y la comunicación entre ellos.

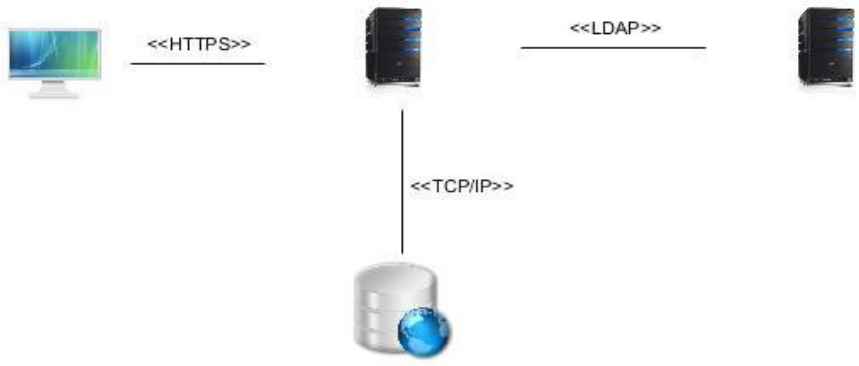


Figura 10: Diagrama despliegue de la solución propuesta.

El sistema cuenta con una PC cliente donde cualquier usuario puede acceder al servidor de aplicaciones para crearse o solicitarse una cita, para que se le realice un documento de identificación, autenticándose con ayuda del servidor LDAP.uci.cu., estableciéndose una conexión de tipo LDAP. La conexión existente ente PCs y servidor de aplicaciones se establece mediante el protocolo de comunicación HTTPS. Por otra parte, se tiene que el servidor de aplicaciones es el nodo que contiene todas aquellas aplicaciones relacionadas con la documentación del certificado de identificación que se desea crear. Mediante el protocolo de comunicación TCP/IP se establece la vínculo del servidor de aplicaciones con el servidor de bases de datos, encargándose de la relación entre datos que se espera tramitar.

3.2 Patrones empleados en la implementación

Una de las vías de solución de problemas es el trabajo con patrones de diseño. El uso de estos patrones ayuda a obtener un software de calidad, facilitando la reutilización de clases ya existentes y la programación de clases reutilizables.

3.2.1 Patrón repositorio

Uno de los patrones que se utilizan en la implementación de la base de datos es el patrón repositorio. Este se ve empleado como intermediario entre la aplicación y los datos, además de verse reflejado en la interfaz *IPersistenRepository agrupando métodos como: Add, Update y Delete.*

Por otra parte, se tiene la interfaz *IReadOnlyRepository* que contiene métodos como: *FindBy (parámetro)* en sus dos versiones, la primera, busca un objeto según su identificador (id) y la segunda, según una expresión; en ambos casos, tanto el id como la expresión se le pasa por parámetro al método. Estas dos interfaces son implementadas por la clase *Repository,* donde se desarrollan cada uno de sus funciones. La **Figura 11** muestra una representación de estas clases e interfaces de forma gráfica.

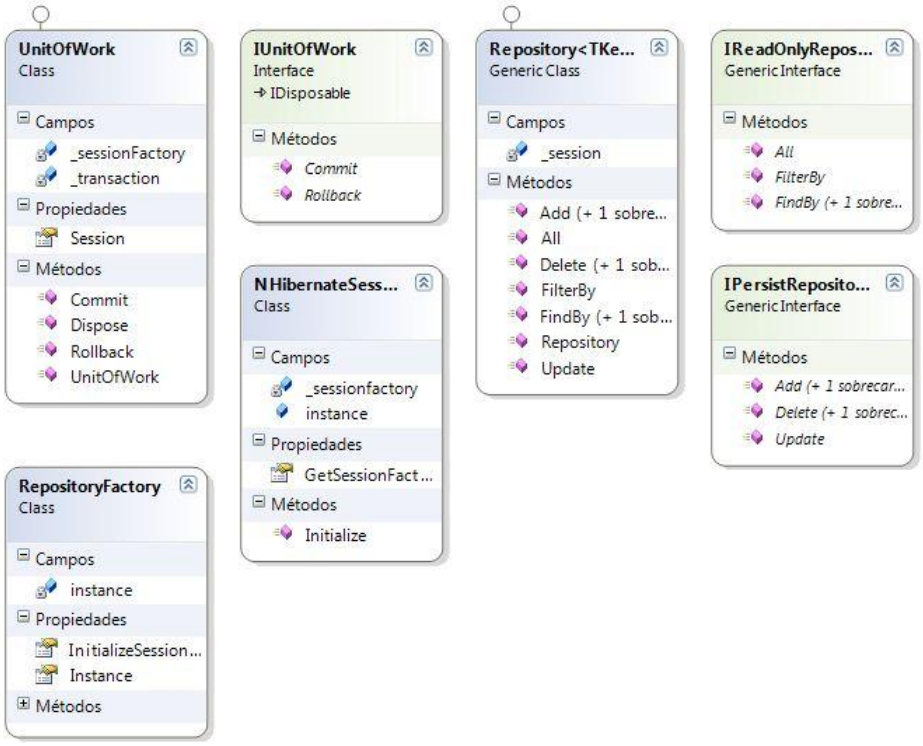


Figura 11: Representación gráfica de clases e interfaces.

3.2.2 Patrón iterador

Cuando alguna clase en el sistema necesita tener acceso a un objeto de entidad, se puede emplear simplemente *IPersistenRepository* para recuperar esa entidad por su identificador. Por otra parte, si se utiliza dicha clave que proporciona la biblioteca de *NHibernate*, extrae una cadena de conexión de la memoria, carga las definiciones de asignación, crea el I*SessionFactory* (una sola vez porque es una función de gran rendimiento) y ajusta la interfaz *ISession* bajo el nivel de *NHibernate.*

En la desarrollo del trabajo se destaca además la implementación de *IColection*, que puede incluir funciones como las de añadir y remover un objeto de la colección. Es aquí donde el iterador hace su función, recorriendo esta colección de objetos y realizando sus funciones predeterminadas.

3.2.3 Patrón instancia única

Este patrón se utiliza a la hora de crear nuevos objetos que son invisibles al sistema y que a su vez sean instancias únicas. Se ve implementado en la clase *NHibernateSessionManager.cs* con el propósito de crear una sola instancia de *ISessionFactory*, permitiendo la conexión a la base de datos.

3.2.4 Patrón unidad de trabajo

Diversas son las responsabilidades que tiene el patrón unidad de trabajo dentro de las que se destacan:

- La administración de transacciones.
- El orden de las acciones que se realizan a las base de datos (insertar, eliminar y actualizar).
- Prohibición de las actualizaciones duplicadas.

En la implementación de la base de datos se crea la clase *UnitofWork*, conteniendo *ITransaction*, I*SessionFactory e ISession.* Además, dispondrá de procedimientos para confirmar o distribuir todos los cambios. Esta clase dispone de métodos como *Commit(),* haciendo efectiva las acciones sobre las transacciones; *Rollback(),* desasiendo todas las operaciones sobre la BD en una transacción errónea y *Dispose(),* cumpliendo con la función de cerrar todas aquellas sesiones en las que se está trabajando. De esta forma, se puede considerar la unidad de trabajo como un lugar para volcar el control de transacciones de todo el código (Miller, 2011).

3.3 Migración hacia PostgreSQL del servicio de persistencia del Bison para Oracle

Bison Framework se encarga de persistir en el SGBD Oracle todas las actividades manejadas por los *workflow*, mediante el servicio de persistencia que ofrece dicho gestor. Para realizar este proceso, el *framework* realiza llamadas a procedimientos almacenados, los cuales se encuentran estructurados dentro de paquetes en la base de datos.

Durante la migración de este servicio de persistencia hacia PostgreSQL se realizaron diferentes transformaciones, dentro de las que se destacan:

- Utilización del SGBD PostgreSQL en su versión 9.1.2.
- La utilización de Npgsql como librería proveedora de datos para el gestor PostgreSQL.
- Se migraron tipos de datos nativos que utiliza Oracle hacia PostgreSQL.
- Se migraron un conjunto de procedimientos almacenados, funciones y triggers.
- Se creó en PostgreSQL el esquema bison_suin_001 para almacenar todos los objetos referentes al Bison.
- Se realizaron cambios durante la implementación en llamadas a objetos pertenecientes al bison_suin_001.

- Se hizo uso de la cadena de conexión de PostgreSQL, la cual facilitó el enlace con la base de datos.

El **Anexo 3** representa una relación de los tipos de datos migrados del servicio de persistencia hacia PostgreSQL.

3.4 Seguridad en las bases de datos

La seguridad en las bases de datos es un mecanismo fundamental, ya que todo sistema informático está expuesto a cualquier tipo de amenazas o daños, que de cierta manera, causan la perdida de la confidencialidad de los datos. Es decir, asegurar la confidencialidad significa prevenir, detectar e impedir la relevación impropia de la información. Por otra parte, "la integridad de los datos proporciona un medio de asegurar que los cambios que se hacen en las bases de datos por usuarios autorizados no resultan en una pérdida de consistencia de los datos" (Alarcón Medina, 2006).

3.4 Seguridad en PostgreSQL

En PostgreSQL la seguridad puede ser materializada en diferentes aspectos, en primer lugar debe garantizarse la seguridad en la manipulación de los ficheros, es decir, que el sistema se puede controlar con tres ficheros de configuración que se encuentran en el directorio de datos *(DATA)* donde inicializamos el clúster de la base de datos. Estos tres ficheros son:

- **pg_hba.conf:** Este fichero se utiliza para definir los diferentes tipos de accesos que un usuario puede tener en el clúster.
- **pg_ident.conf:** Este fichero se utiliza para definir la información necesaria en el caso que se utilice el tipo de acceso ident en pg_hba.conf.
- **postgresql.conf:** En este fichero se puede cambiar todos los parámetros de configuración que afectan al funcionamiento y al comportamiento de PostgreSQL en la PC.

Otro aspecto importante a tener en cuenta es la seguridad en los accesos de los clientes, ya que es necesario definir qué usuarios se conectarán a la base de datos, desde qué equipos y por cuál método de autentificación[11] lo harán.

[11] La autentificación es el proceso por el que el servidor de base de datos establece la identidad del cliente mediante un usuario y una contraseña, y por ende determinar si a la aplicación del cliente (o al usuario que ejecuta la aplicación del cliente) se le permite conectarse al servidor PostgreSQL.

La seguridad en este nivel se realiza mediante el fichero pg_hba.conf (de sus siglas en inglés host-based authentication), fichero de autentificación basada en host (Fisher, 2013). Este archivo contiene registros que tienen el siguiente formato:

[Tipo de conexión][DATABASE][USER][CIDR-ADDRESS][METHOD][Opciones]

Donde el primer campo indica el *tipo de conexión*: "local" es una conexión al equipo local, "host" es una conexión con o sin encriptación SSL mediante el socket TCP/IP; "hostssl" es una conexión encriptada SSL mediante el socket TCP/IP, mientas que "hostnossl" es una conexión TCP/IP que no está encriptada.

El campo *DATABASE* indica la base de datos a la que se pueden conectar los usuarios, tomando valores como: "all" para especificar que son todas o "sameuser" para especificar que se dé acceso a las bases de datos a las que el usuario identificado tenga acceso.

El campo *USER* indica qué usuarios tendrán acceso a las bases de datos que se le sean permitidas. De igual forma se hace referencia a todos con la palabra reservada "all".

El campo *CIDR-ADDRESS* especifica las direcciones de los equipos o el conjunto de equipos (red) que se comunicarán con el servidor. Está estructurado con la dirección IP del equipo y la máscara de red a la que pertenece, que se representa con un entero (entre 0 y 32 para IPv4 o 128 para IPv6) y especifica el número de bits significativos en la máscara de red. Para el caso de una conexión local se deja en blanco este campo.

METHOD, define el tipo de autentificación que el servidor debería usar para un usuario que intenta conectarse a PostgreSQL. Existen varios métodos posibles a utilizar, entre los que están: trust, md5 e ident. Si se pone "trust" en dicho campo, se permite la conexión incondicionalmente al servidor PostgreSQL, sin la necesidad de contraseña.

En cambio, si se utiliza el método "md5" se le exige al usuario que proporcione una contraseña encriptada MD5[12] para la autentificación. En otro caso, si se emplea "ident" se especifica el nombre del usuario del sistema operativo en la maquina cliente, y se verifica si dicho usuario se identifica con el nombre del usuario de la base de datos a la que se quiere acceder mediante una consulta al archivo *pg_ident.conf*. Lugo de realizadas estas configuraciones sobre el archivo *pg_hba.conf* se reinicia el servicio de PostgreSQL para que los cambios surjan efecto.

[12] MD5 (acrónimo de Message-Digest-Algorithm, o Algoritmo de Resumen de Mensaje) es un algoritmo de reducción criptográfico de 128 bits ampliamente usado para realizar autentificaciones de usuarios.

3.5 Optimización

Cada día se necesita procesar mayor cantidad de datos y obtener de manera más rápida y precisa la información. Para lograr que esto suceda, se realiza la optimización de la base de datos, comenzando con los diseños de los modelos que aportan relaciones y atributos reflejándose en un modelo lógico final.

La normalización de las tablas como el siguiente paso de la optimización, permite reducir al máximo la información repetida, no excluyéndola totalmente ya que puede afectar el rendimiento del sistema. También, ajustar al máximo el tamaño de los campos ayuda a no desperdiciar espacio en la base de datos, eliminando en tal caso todo campo que no sea de utilidad, ya que por más que contenga datos genera retrasos. Como buena práctica a seguir se decidió que, entre los campos candidatos para realizarle la indexación, estarían los campos de mayor frecuencia de acceso, así como los más empleados como criterio de comparación en consultas *WHERE*, entre otros.

PostgreSQL brinda entre sus múltiples ventajas la posibilidad de contar con procedimientos almacenados que permiten ahorrar un grupo de pasos al ejecutar diversas consultas. En el caso de la base de datos en cuestión se expresa mediante la utilización de *triggers*, favoreciendo notablemente el rendimiento del sistema.

La configuración de los parámetros relevantes al rendimiento del archivo *postgresql.conf*, forma parte de los aspectos básicos para una correcta optimización del rendimiento de sistema. Esta configuración está relacionada con las características del hardware de la PC en que está instalado el servidor, y el sistema operativo con que se obrará. Se configuró el *shared_buffers* proporcionándole 450MB de la capacidad de la memoria RAM (memoria de acceso aleatorio, 1GB), indicando el número de bloques o buffers de 8KB que el gestor reservará como zona de trabajo en el momento de inicio para procesar las consultas. Se le asignó a *max_connections* un máximo de 100 conexiones permitidas a la base de datos. Se concedió 512MB de la RAM al parámetro *efective_cache_size*, encargado de exponerle al optimizador de PostgreSQL la cantidad de memoria disponible para el almacenamiento de datos en la caché, además, se destinó un valor muy por debajo de la RAM a la memoria destinada a cada usuario concurrente, conocido como *work_men* (8MB). Para operaciones del tipo *VACUUM, CREATE INDEX, ALTER TABLE* se le asignó 512MB al parámetro *maintenance_work_men*. Además se modificó el parámetro *max_prepare_transaction* para establecer el número máximo de transacciones simultáneas. Desde el punto de vista del proceso de pruebas, resultaron favorables estas configuraciones.

El planificador de consultas de PostgreSQL es quien se encarga de decidir cuál será la manera más válida de ejecutar una consulta y hacerlo en un tiempo lo suficientemente rápido. Esto se ve evidenciado en el momento en que se realizan los *EXPLAIN ANALYZE* a la tabla *dsolicitud.*

En un primer ejemplo se muestran los costos y el tiempo total de rutina, seleccionando de dicha tabla las solicitudes que cumplen con una fecha determinada. **(Ver Figura 12)**.

En el segundo ejemplo se realiza la misma consulta del caso anterior, evidenciándose la utilización de un índice para mejorar la velocidad de las operaciones. Este índice fue creado al atributo *fecha_hora* de la tabla *dsolicitud.* **(Ver Figura 13)**.

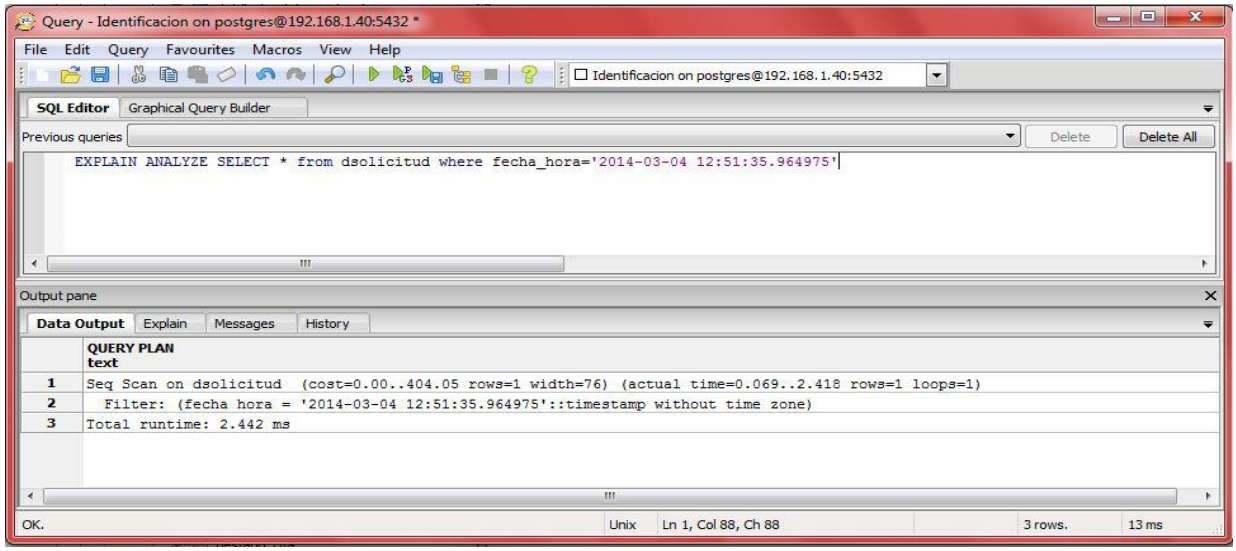


Figura 12: Resultado de la consulta realizada a la tabla *dsolicitud.*

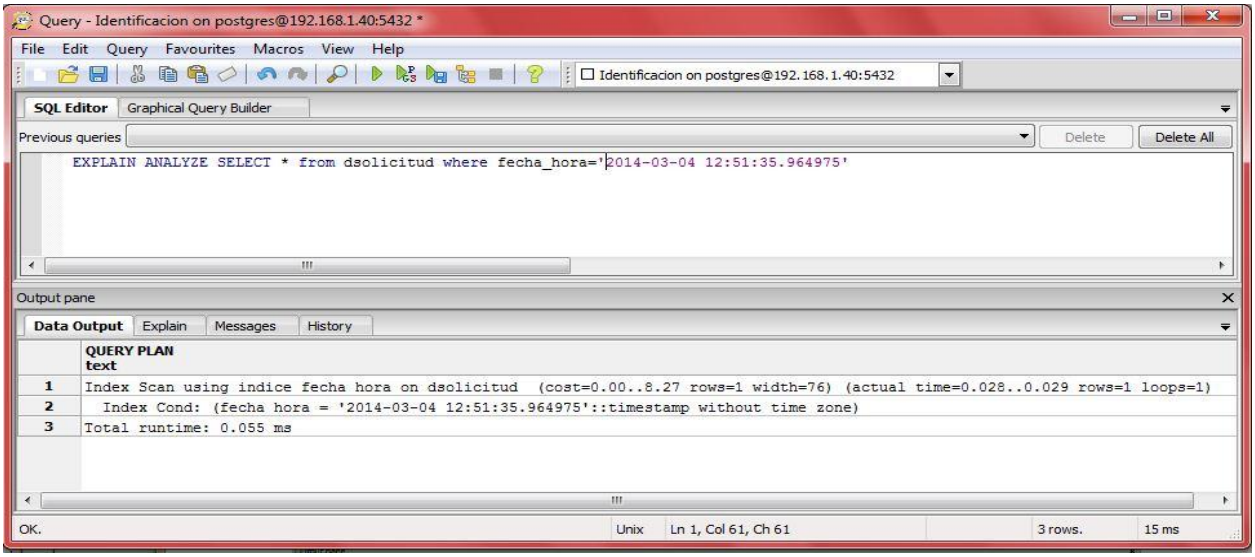


Figura 13: Resultados arrojados por el planificador de consultas utilizando la indexación.

Haciendo un análisis de los resultados mostrados en cada ejemplo, se puede llegar a la conclusión de que, la utilización de índices, en este caso, necesarios para la base de datos en cuestión, garantiza una mayor validez en cuanto al tiempo de respuesta de las operaciones realizadas, disminuyendo tanto los valores del costo de las sentencias SQL, como el tiempo total de rutina y el tiempo expresado en milisegundos (ms) de la operación en general.

La práctica de estas acciones de forma eficiente, influye positivamente en la optimización de la BD para la Plataforma Modular. No se deben realizar solamente estas acciones cuando existen otras prácticas a seguir que contribuyen a la configuración adecuada de la base de datos en busca de rendimiento:

- Optimizar las características del hardware del servidor: paso fundamental para garantizar la calidad de los servicios que ofrezca el servidor, ya que por lo generar están destinados a consumir gran volumen de recursos.
- Aumentar el volumen de resultados aún más con cada consulta
- Conocer las características de rendimiento entre las sentencias de lenguaje: este proceso se garantiza evitando el uso de cláusulas que pueden ser más costosas.
- Programación de vacunas: por lo menos se debe realizar algún día del mes para realizar un mantenimiento de las tablas más actualizadas.
- Reindexación: recomendable realizarla diariamente en horarios donde la base de datos tenga una menor gestión de usuarios, garantizado con esta práctica la reorganización de índices alterados por los procesos de limpieza.
- Realizar *ANALYZE* a las tablas: la realización de este proceso manualmente sobre las tablas de mayor número de inserciones y actualizaciones favorece los planes de consultas a la BD.

3.6 Pruebas para la validación de la BD

Para la validación funcional de la base de datos se realizará un conjunto de pruebas con el objetivo de comprobar la integridad de cada una de las entidades que la componen, además de simular una carga de producción real y posteriormente observar el comportamiento que tiene la BD bajo dicha carga. Este proceso posibilitará darle solución a los problemas de rendimiento existentes en la base de datos, antes de ser utilizada por otros usuarios. Las herramientas utilizadas para las pruebas de carga y estrés fueron *EMS Data Generator for PostgreSQL* y *Apache JMeter 2.3.1*

3.6.1 Pruebas de integridad al modelo

Con el objetivo de garantizar la integridad de los datos almacenados, se realizarón un conjunto de pruebas, entre las que se destacan las pruebas de integridad de entidad, e integridad referencial. Con las pruebas de integridad de entidad, se verifica que las claves primarias de las entidades del modelo no tomen valores nulos, ni duplicados; pues para lograr que esto no suceda PostgreSQL asigna automáticamente restricciones *UNIQUE*, y *NOT NULL* sobre las llaves primarias de las entidades. Un ejemplo de esto se evidencia en la **Figura 14**, donde al tratar de insertar un valor nulo en la llave primaria de la entidad *dciudadano* (*id_ciudadano*), el gestor lanza un error, igualmente ocurre si se trata de insertar una llave duplicada, como se muestra en la **Figura 15**.

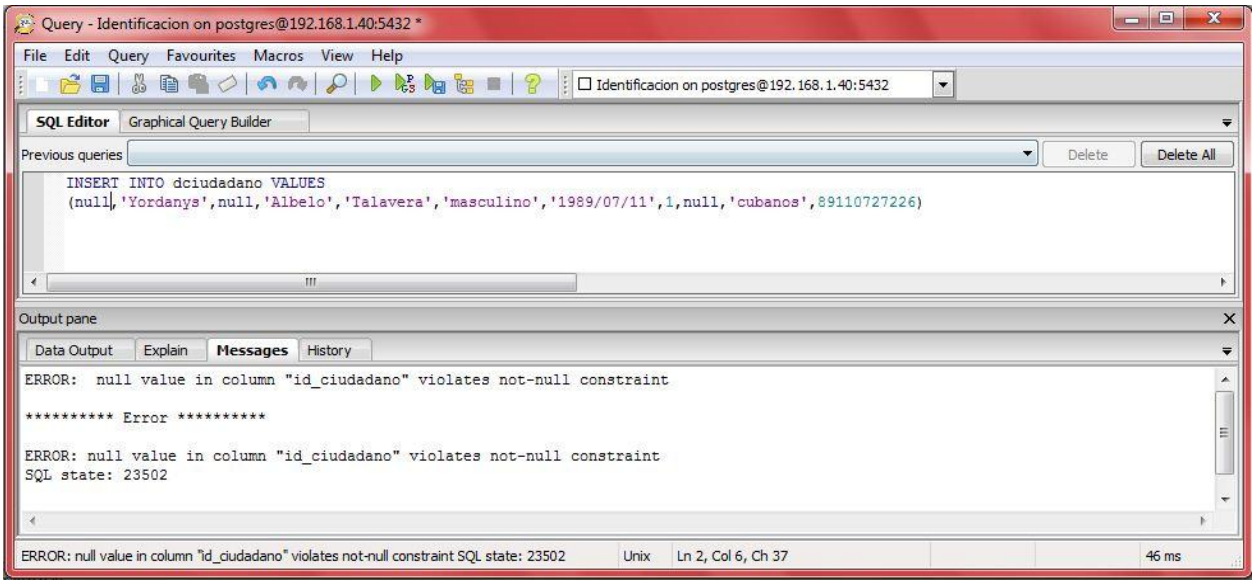


Figura 14: Error mostrado al insertar una llave primaria nula.

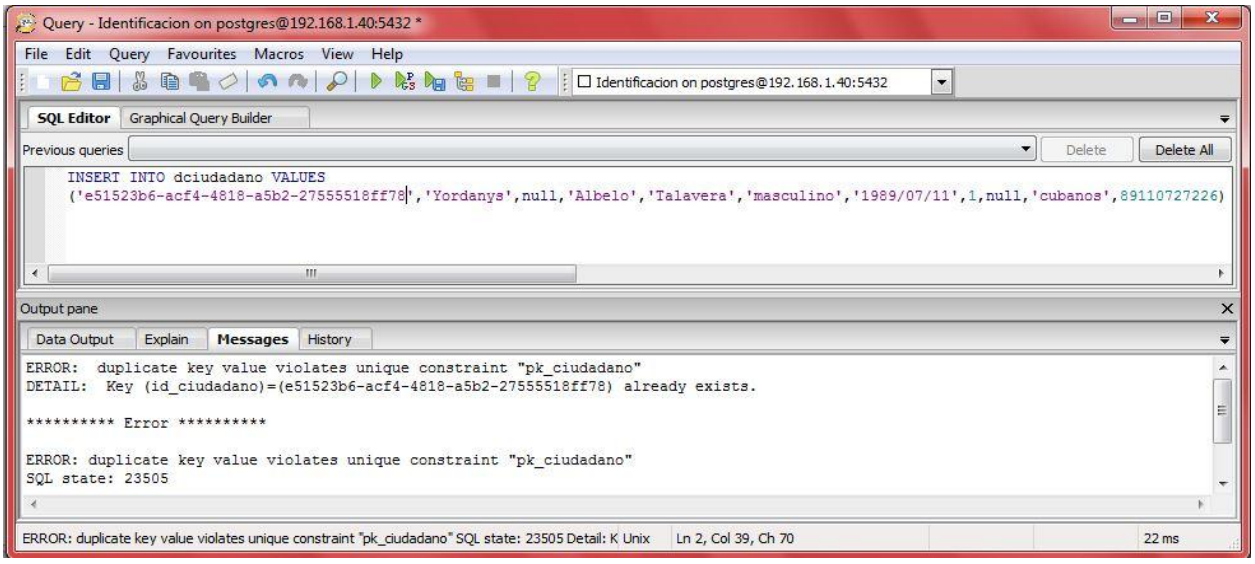


Figura 15: Error mostrado al insertar una llave primaria duplicada.

Algunos campos de la base de datos están sujetos a dominios, con el objetivo de garantizar la validez de las entradas para una columna determinada. Mediante las pruebas de integridad de dominio, se garantiza que cada valor que se desea modificar en las tablas cumpla con el dominio requerido. Un ejemplo de ello se muestra en la **Figura 16** donde al tratar de hacer un inserción sobre la entidad *dciudadano*, el gestor lanza un error, indicando que el valor viola la restricción implementada por el dominio *dom_carnet_identidad*, el cual solo admite números de once dígitos.

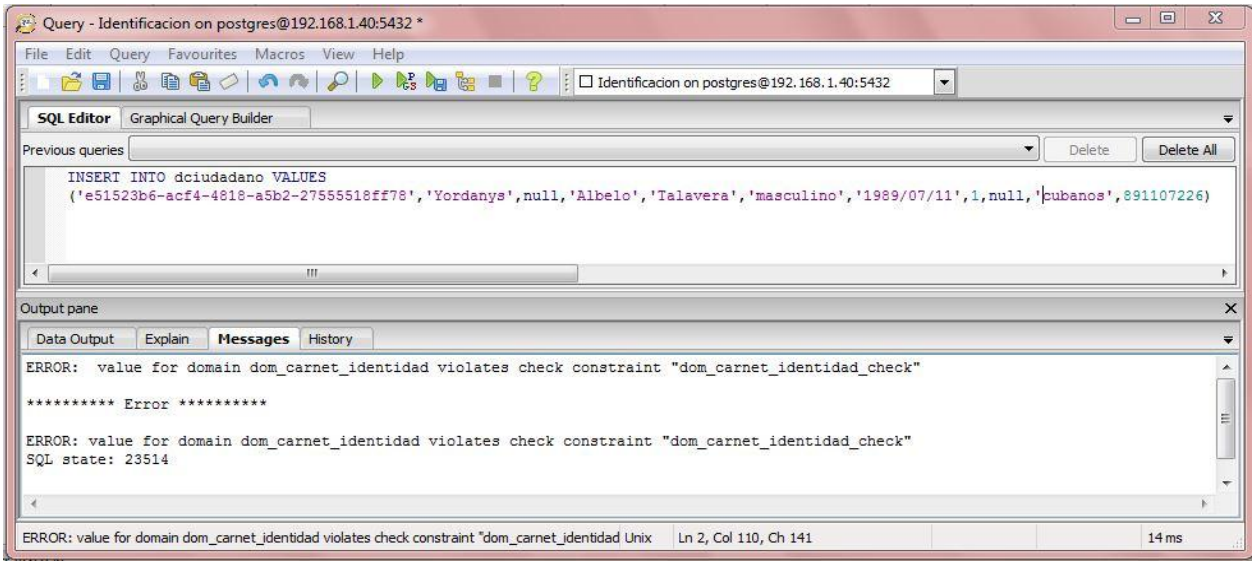


Figura 16: Error mostrado al insertar un ciudadano con valores incompletos del carnet de identidad.

Con la integridad referencial se les impide a los usuarios agregar o cambiar filas en una tabla relacionada si no hay ninguna fila asociada en la tabla principal; además modificar o eliminar filas de una tabla principal cuando hay filar relacionadas coincidentes (Durand , 2010).

Expresándolo de otra manera, se puede decir que la integridad referencial garantiza que los valores de las claves sean coherentes entre diferentes tablas en el modelo, o sea, al modificar una llave que es referenciada desde otras tablas, esta debe actualizarse en toda la base de datos, teniendo en cuanta que no debe de existir referencias a valores inexistentes. Un ejemplo de ello lo demuestra la **Figura 17**.

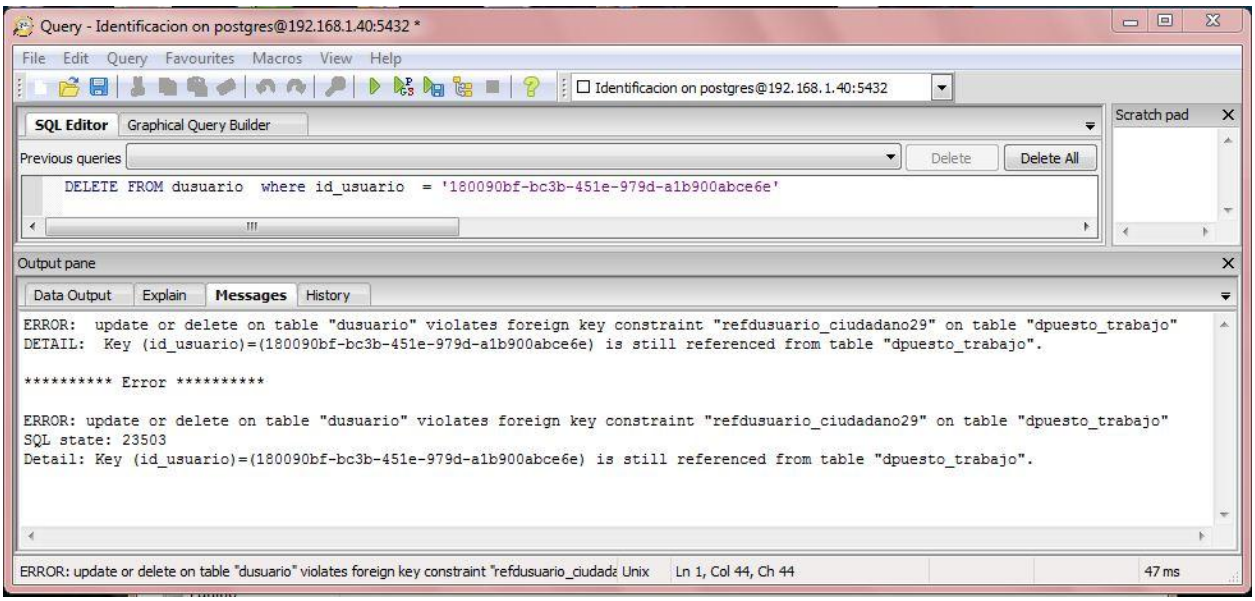


Figura 17: Error mostrado al tratar de eliminar una fila de una tabla referenciada en otras tuplas del modelo.

3.6.2 Pruebas unitarias

Las pruebas unitarias constituyen el primer paso para la detección de errores en el código, pues se concentran en la menor unidad de diseño del software: una clase o un método de una clase. Su objetivo primordial, es el de detectar errores en el código de los métodos de una clase, al ser ejecutados independientemente del resto de los componentes de software (Ramos Roman, y otros, 2007).

Estas pruebas permiten verificar mediante la ejecución de un fragmento de código si la BD cumple con las especificaciones esperadas. En la **Figura 18** se muestra la prueba realizada a varias funcionalidades, destacándose "*CiudadanoCarnet",* arrojando lo siguientes resultados.

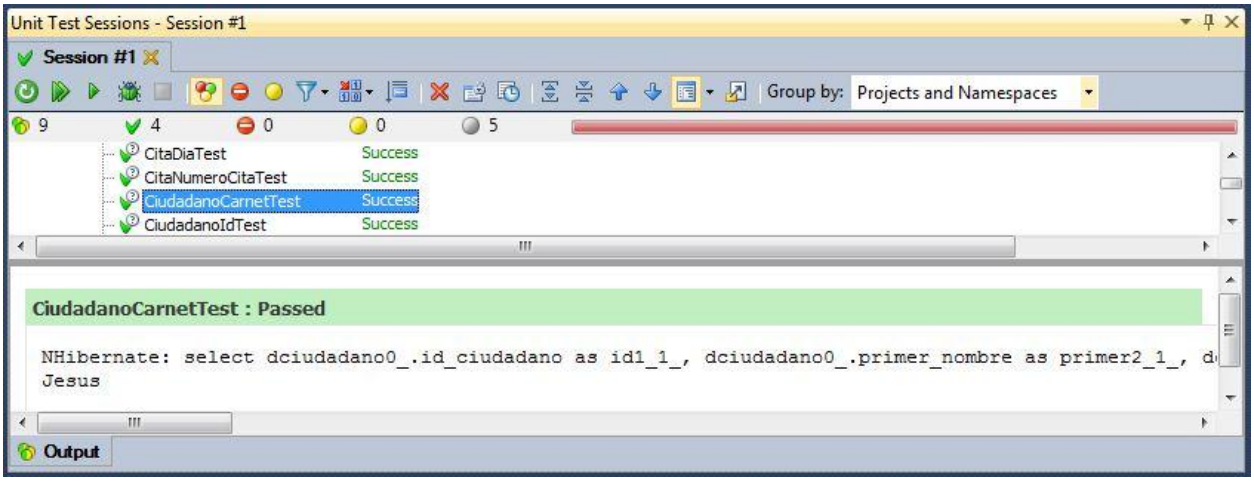


Figura 18: Resultados de la prueba unitaria a la funcionalidad *CiudadanoCarnet*.

Con la ejecución de esta prueba se obtiene el ciudadano correspondiente al carnet de identidad dado, completándose satisfactoriamente el test aplicado, ilustrándose con un icono verde. En caso contrario, la prueba se mostraría con un icono de color rojo, expresando fallos en la ejecución del método.

3.6.3 Prueba de volumen a la BD

Las pruebas de volumen se hacen con el objetivo de analizar el comportamiento de la BD, con volúmenes de datos lo más similar posible a los esperados en la explotación real del sistema. Estas se realizaron con la herramienta *EMS Data Generator for PostgreSQL,* permitiendo el llenado en todas las tablas al mismo tiempo de forma visual y natural, dando la posibilidad de elegir el rango de datos que quiere especificar según el tipo de columna.

Este tipo de prueba es especialmente útil si se quiere tener un parecido de la información estructural con la información real. Se probó la herramienta, generando un total de 20 000 tuplas a la base de datos, y la misma no presentó problemas, ya sea de límite de capacidad o volumen de datos, desbordamiento de columnas, atributos o tipos de datos, garantizando con esto que el diseño de la BD y el gestor utilizado soportan el almacenamiento de los niveles de información requeridos para el inicio del funcionamiento de la BD.

3.6.4 Pruebas de rendimiento

Entre los distintos tipos de pruebas de rendimiento se encuentran las pruebas de carga a la base de datos. Esta prueba cuenta con el objetivo de asegurar que el sistema pueda procesar una carga esperada, durante un tiempo determinado, que se va incrementando regularmente hasta que el rendimiento del sistema se hace inaceptable. Este test demuestra por una parte que el sistema satisface sus requerimientos; y por otra que descubre problemas y defectos en dicho sistema (Bustos Thames, 2011).

Las pruebas de estrés forman parte también de las pruebas de rendimiento al sistema. Su objetivo es estresar al sistema (de ahí su nombre) realizando demandas fuera de los límites del diseño de software. Consisten en ir creando pruebas, acercándose a la máxima carga del diseño del sistema, hasta que el sistema falla, verificando en tal caso que las sobrecargas en el sistema provocan fallos en lugar de colapsarlo.

Para la realización de dichas pruebas se elige a la herramienta *Apache JMeter*, el cual fue configurado para conectarse a la base de datos. Las pruebas se realizaron a una misma consulta pero aumentando la cantidad de usuarios diferentes, por lo que arrojaron resultados distintos que se presentan a continuación a través de las siguientes variables:

- Media: es el tiempo promedio de respuesta de todas las peticiones.
- Mediana: tiempo promedio de la mitad de los resultados, la otra mitad tomará un tiempo entre la mediana y el valor máximo.
- Mínimo: mínimo de tiempo de respuesta de un petición a la base de datos.
- Máximo: máximo de tiempo de respuesta de una petición.

Prueba: Devolver todos los datos de un ciudadano donde su carnet de identidad sea 89110727228. (Ver **Anexo 4**).

select dciudadano0_.id_ciudadano as id1_1_, dciudadano0_.primer_nombre as primer2_1_,dciudadano0_.segundo_nombre as segundo3_1_, dciudadano0_.primer_apellido as primer4_1_, dciudadano0_.segundo_apellido as segundo5_1_, dciudadano0_.fecha_nacimiento as fecha6_1_, dciudadano0_.sexo as sexo1_, dciudadano0_.carnet_identidad as carnet8_1_,dciudadano0_.nacionalidad as nacional9_1_, dciudadano0_.id_tipo_ciudadano as id10_1_, dciudadano0_.id_tipo_dato_opcional as id11_1_ from dciudadano dciudadano0_ where dciudadano0_.carnet_identidad= 89110727228;

Después de validar la consulta se procede a analizar los resultados arrojados por la herramienta elegida con las variables antes descritas.

Cantidad de usuarios concurrentes	Media(ms)	Mediana(ms)	Mínimo(ms)	Máximo(ms)
150	10	4	2	94

Tabla 2: Resultados de la prueba para 150 hilos.

Cantidad de usuarios concurrentes	Media(ms)	Mediana(ms)	Mínimo(ms)	Máximo(ms)
5000	4042	2874	2	14548

Tabla 3: Resultados de la prueba para 5000 hilos.

Es importante resaltar que estas pruebas se realizaron sobre un servidor, con las siguentes caracteristicas de hardware:

- 2CPU (Dual-Core).
- 2GB de RAM.
- 20GB de disco duro.

Los resultados evidencian la efectividad de los cambios realizados en la BD en aras de lograr su optimización. En el análisis realizado a la primera prueba se representa una conexión

concurrente de 150 usuarios y una más elevada en la segunda prueba, con 5000 usuarios concurrentes, mostrándose una gestión rápida de la información.

Finalmente, analizando los indicadores de rendimiento (rendimiento medio en base a peticiones por segundos y en Kilobytes por segundos) se percibe la agilidad de la gestión de peticiones a la base de datos, favoreciendo la optimización que se propone en el presente trabajo de diploma.

Conclusiones parciales

Con la validación teórica del diseño de la base de datos propuesta, se comprobó que dicho sistema cumpliera con la integridad de sus datos en cada una de sus entidades, efectuándose en un ambiente de despliegue cómodo para desarrollar la solución. Además se alcanzó un nivel de trabajo placentero con la ayuda de los patrones de diseño implementados, obteniéndose la capa de persistencia de datos para la PMICA. Además se tuvo en cuenta también la validación funcional de la BD mediante las pruebas de volumen, carga y estrés, las cuales permitieron conocer el comportamiento de la misma ante una sobrecarga de información y grandes conexiones de usuarios concurrentemente.

Conclusiones Generales.

Las bases de datos constituyen uno de los elementos fundamentales en el funcionamiento de cualquier empresa, garantizando su objetivo principal: el almacenamiento de la información valiosa que se maneja en la misma.

Con el presente trabajo de diploma se le proporciona a la Universidad de las Ciencias Informáticas una base de datos creada fundamentalmente para la Plataforma Modular de Identificación y Control de Acceso de personas a la institución.

Con el diseño e implementación de la capa de la capa de persistencia de datos se cumplió el objetivo general planteado. Con la normalización hasta la 3FN de cada una de las entidades del modelo de datos se logró eliminar las incoherencias y redundancias de los datos.

Finalmente, esbozando una correcta estructura de datos se alcanzó la optimización de la BD, la cual fue validada mediante diferentes pruebas, arrojando como resultados una base de datos resistente y con un excelente rendimiento.

RECOMENDACIONES

Durante el desarrollo del trabajo de diploma surgieron ideas que pueden servir como recomendación para el perfeccionamiento de la base de datos. Ellas son:

- Creación de índices a medida que la BD sea poblada con una cantidad mayor de datos.
- Repetir el proceso de *ANALIZE* a las consultas de mayor gestión de información, una vez poblada la BD.
- Uso de las pautas propuestas para el diseño correcto de una BD y procedimientos del control de cambio para próximas iteraciones.
- Continuar con las pruebas e iteraciones en el desarrollo del servicio de persistencia buscando la optimización y perfección del mismo.

Trabajos citados

Alarcón Medina, José Manuel. 2006. PostgreSQL. *Manua de administración de SGBD PostgreSQL.* [En línea] noviembre de 2006. http://es.scribd.com/doc/32185176/144/Tipos-de-indices.

Alcalde, Alejandro. 2013. PL/SQL- Paquetes (Packages). [En línea] 9 de abril de 2013. http://elbauldelprogramador.com/programacion/basededatos/plsql-paquetes-packages/.

Brea. 2011. Aplicaciones Ofimáticas MR. *SGBD mas usados.* [En línea] 11 de abril de 2011. [Citado el: 10 de 11 de 2012.] http://docentes.educacion.navarra.es/bromerog/apof/index.php?option=com_content&view=article&id=48:04-base-de-datos&catid=37:bases-de-datos&Itemid=50.

Bustos Thames, Juan Pablo. 2011. Slideshare Present Yourself. *Pruebas del Software.* [En línea] 24 de noviembre de 2011. [Citado el: 2 de abril de 2013.]

Collorana, José R. 2009. *FDD (Features Driven Development) Desarrollo Basado en Funcionalidades. Ingeniería Académica.* La Paz – Bolivia : UNIVERSIDAD UNION BOLIVARIANA: 12, 2009.

Criptoy, Jimmy. 2012. Microsoft Visual Studio 2010. [En línea] 2 de junio de 2012. [Citado el: 15 de nobiembre de 2012.] http://www.intercambiosvirtuales.org/software/microsoft-visual-studio-2010-ultimate-espanol.

Durand , Darwin . 2010. Slideshare Present Yourself. *Integridad de datos.* [En línea] 5 de mayo de 2010. [Citado el: 1 de abril de 2013.] http://www.slideshare.net/sistemasddm/integridad-sql-server.

Escobar, Fernando. 2013. Programando en .NET. [En línea] 10 de Enero de 2013. [Citado el: 25 de marzo de 2013.] http://msdn.microsoft.com/es-es/library/cc466455(v=vs.71).aspx.

Faci, Santigo. 2004. *Lenguaje PL/SQL.* 2004.

Fisher, Hanton . 2013. Scribd. *5.7.2 Archivo pg_hba.conf for Bade de Datos.* [En línea] 2013. [Citado el: 30 de marzo de 2013.] http://es.scribd.com/doc/131967385/72/Archivo-pg-hba-conf.

Fowler, Martín. 2005. *Patterns of Enterprise Application Architecture.* s.l. : Addison-wesley, 2005.

Galens Ameneiro, Linnett y Sierra Obregón, Dannier . 2010. *Sistema de Personalización de Documentos de Identificación de la República de Cuba.* Ciudad de La Habana : s.n., 2010.

García, Yupanqui y Jasper, Glen. 2010. Patrones. *Diseño de sistemas.* [En línea] 2010. [Citado el: 15 de enero de 2012.] http://es.scribd.com/doc/3930805/Patrones-de-Diseno.

IBM. 2013. Visión general de la gestión de cambios en una base de datos. [En línea] 2013. [Citado el:]

Larman, Craig. 1999. *Uml y Patrones, introducción al análisis y diseño orientado a objetos.* México : Mexicana, 1999. Núm. 1524.

Mariluz Hernández, Perdomo y Fernández, José Enrique . 2009. *Guía para la optimización de servidores de bases de datos de PostgreSQL.* Ciudad de La Habana : s.n., 2009.

Martínez, Alexei. 2013. EcuRed. *Visual Paradigm.* [En línea] 2013. [Citado el: 15 de noviembre de 2012.] http://www.ecured.cu/index.php/CASE.

Martínez, Rafael. 2007. Introducción a PostgreSQL - Configuración. [En línea] 21 de agosto de 2007. http://www.linux-es.org/node/660.

Mato García, Rosa María. 2005. *Diseños de Bases de Datos.* La Habana : Pueblo y Educación, 2005.

Member, Junior . 2012. SQL Manager 2011 for PostgreSQL. *SQL Manager Lite for PostgreSQL (formerly EMS SQL Manager for PostgreSQL Lite) 5.3.0.1.* [En línea] 2012. http://www.softpedia.es/programa-EMS-PostgreSQL-Manager-Lite-22778.html.

Microsoft. 2013. Microsoft. [En línea] 2013. http://msdn.microsoft.com/es-es/library/bb399572.aspx.

Miller, Jeremy. 2011. MSDN Magazine. *La unidad de modelo de trabajo y persistencia Ignorance.* [En línea] 2011. [Citado el: 30 de marzo de 2013.] http://msdn.microsoft.com/es-es/magazine/dd882510.aspx#id0420003.

Montenegro Bernot, Yoandrys. 2011. EcuRed. *Sistemas Gestores de Bases de Datos.* [En línea] 11 de septiembre de 2011. [Citado el: 10 de noviembre de 2012.] http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos.

Palacios, Damian. 2012. Motor de persistencia NHibernate. [En línea] 7 de junio de 2012. [Citado el: 20 de noviembre de 2012.] http://www.slideshare.net/DamianPalacios/motor-de-persistencia-nhibernate.

Pérez Gamboa, Alexander. 2012. *Diseño de la Base de Datos para el Sitema de Integracion Bancarria.* La Habana : s.n., 2012.

Pérez González , Irael y Sosa Díaz, Madelaine. 2011. *Propuesta de optimización de la base de datos para el proyecto Sistema de Informatización de la Gestión de las Fiscalías.* Ciudad de la Habana : s.n., 2011.

Pérez Urrutia, Margelys. 2010. *Diseño e Implementación de la Base de Datos de la Plataforma de Servicios a Pozos Petroleros.* s.l. : La Habana, 2010.

Pérez Valdés, Damian. 2007. ¿Qué son las bases de datos? [En línea] 26 de octubre de 2007. http://www.maestrosdelweb.com/editorial/%C2%BFque-son-las-bases-de-datos/.

Ramos Roman, Isabel y Dolado Cos, Javier . 2007. *Técnica Cuantitativa para la gestión en la ingeniría del software.* 2007.

Rodríguez Martín, Róger. 2011. Ecured. [En línea] 15 de abril de 2011. http://www.ecured.cu/index.php/ER/Studio.

Rubiano, Martha. 2009. UML Diagramas de despliegue. [En línea] 9 de octubre de 2009. [Citado el: 30 de marzo de 2013.] http://es.scribd.com/doc/19611312/diagramas-de-despliegue-2222.

Salazar, Cristian. 2012. Modelo de datos. [En línea] 28 de septiembre de 2012. [Citado el: 10 de enero de 2013.] http://www.slideshare.net/csalazarc/modelo-de-datos-14506949.

Soriano, Amelia. Tipos de prueba. [En línea] http://carolina.terna.net/ingsw3/datos/Tipos_Prueba.pdf.

SQLManager. 2010. Sitio de descargas de software. [En línea] 2010. http://www.freedownloadmanager.org/es/downloads/Generador_de_Datos_de_SME_2005_para_PostgreSQL_37202_p/.

Strappazzon, Nicola C. 2010. Artículos de interes para el desarrollo de sistemas WEB. *Uso del modelo Atributos & Tipos (EAV) en el diseño de Base de Datos.* [En línea] 30 de octubre de 2010. http://www.swapbytes.com/2010_10_01_archive.html.

Trillo Lado, Raquel. 2010. *Taller de patrones de diseño.* s.l. : Universidad de Zaragoza., 2010.

Tumax , Oto. 2012. Optimización De Consultas. *Procesamiento y Optimizaciónde Consultas SQL.* [En línea] 1 de mayo de 2012. http://www.slideshare.net/etumax/optimizacion-de-consultas.

Vallez Pérez, Paqui. 2009. Emagister. *Informatica en la Administración pública.* [En línea] 31 de marzo de 2009. [Citado el: 15 de noviembre de 2011.] http://www.emagister.com/curso-informatica-administracion-publica-3/bases-datos-concepto-caracteristicas-funcionalidades.

Bibliografía

Tumax , Oto. 2012. Optimización De Consultas. *Procesamiento y Optimizaciónde Consultas SQL.* [En línea] 1 de mayo de 2012. http://www.slideshare.net/etumax/optimizacion-de-consultas.

Alarcón Medina, José Manuel. 2006. PostgreSQL. *Manua de administración de SGBD PostgreSQL.* [En línea] noviembre de 2006. http://es.scribd.com/doc/32185176/144/Tipos-de-indices.

Alcalde, Alejandro. 2013. PL/SQL- Paquetes (Packages). [En línea] 9 de abril de 2013. http://elbauldelprogramador.com/programacion/basededatos/plsql-paquetes-packages/.

Brea. 2011. Aplicaciones Ofimáticas MR. *SGBD mas usados.* [En línea] 11 de abril de 2011. [Citado el: 10 de 11 de 2012.] http://docentes.educacion.navarra.es/bromerog/apof/index.php?option=com_content&view=article&id=48:04-base-de-datos&catid=37:bases-de-datos&Itemid=50.

Bustos Thames, Juan Pablo. 2011. Slideshare Present Yourself. *Pruebas del Software.* [En línea] 24 de noviembre de 2011. [Citado el: 2 de abril de 2013.]

Collorana, José R. 2009. *FDD (Features Driven Development) Desarrollo Basado en Funcionalidades. Ingeniería Académica.* La Paz – Bolivia : UNIVERSIDAD UNION BOLIVARIANA: 12, 2009.

Criptoy, Jimmy. 2012. Microsoft Visual Studio 2010. [En línea] 2 de junio de 2012. [Citado el: 15 de nobiembre de 2012.] http://www.intercambiosvirtuales.org/software/microsoft-visual-studio-2010-ultimate-espanol.

Durand , Darwin . 2010. Slideshare Present Yourself. *Integridad de datos.* [En línea] 5 de mayo de 2010. [Citado el: 1 de abril de 2013.] http://www.slideshare.net/sistemasddm/integridad-sql-server.

Escobar, Fernando. 2013. Programando en .NET. [En línea] 10 de Enero de 2013. [Citado el: 25 de marzo de 2013.] http://msdn.microsoft.com/es-es/library/cc466455(v=vs.71).aspx.

Faci, Santigo. 2004. *Lenguaje PL/SQL.* 2004.

Fisher, Hanton . 2013. Scribd. *5.7.2 Archivo pg_hba.conf for Bade de Datos.* [En línea] 2013. [Citado el: 30 de marzo de 2013.] http://es.scribd.com/doc/131967385/72/Archivo-pg-hba-conf.

Fowler, Martín. 2005. *Patterns of Enterprise Application Architecture.* s.l. : Addison-wesley, 2005.

Galens Ameneiro, Linnett y Sierra Obregón, Dannier . 2010. *Sistema de Personalización de Documentos de Identificación de la República de Cuba.* Ciudad de La Habana : s.n., 2010.

García, Yupanqui y Jasper, Glen. 2010. Patrones. *Diseño de sistemas.* [En línea] 2010. [Citado el: 15 de enero de 2012.] http://es.scribd.com/doc/3930805/Patrones-de-Diseno.

Gibert Ginestà, Marc y Pérez Mora, Oscar . 2006. *Bases de datos en PostgreSQL.* 2006.

Hernández, José. 2011. Usar NHibernate con FluentNHibernate. [En línea] 14 de abril de 2011. http://jhernandz.es/noticia/usar-nhibernate-con-fluentnhibernate.

IBM. 2013. Visión general de la gestión de cambios en una base de datos. [En línea] 2013. [Citado el:]

Larman, Craig. 1999. *Uml y Patrones, introducción al análisis y diseño orientado a objetos.* México : Mexicana, 1999. Núm. 1524.

Mariluz Hernández, Perdomo y Fernández, José Enrique . 2009. *Guía para la optimización de servidores de bases de datos de PostgreSQL.* Ciudad de La Habana : s.n., 2009.

Martínez, Alexei. 2013. EcuRed. *Visual Paradigm.* [En línea] 2013. [Citado el: 15 de noviembre de 2012.] http://www.ecured.cu/index.php/CASE.

Martínez, Rafael. 2007. Introducción a PostgreSQL - Configuración. [En línea] 21 de agosto de 2007. http://www.linux-es.org/node/660.

Mato García, Rosa María. 2005. *Diseños de Bases de Datos.* La Habana : Pueblo y Educación, 2005.

Member, Junior . 2012. SQL Manager 2011 for PostgreSQL. *SQL Manager Lite for PostgreSQL (formerly EMS SQL Manager for PostgreSQL Lite) 5.3.0.1.* [En línea] 2012. http://www.softpedia.es/programa-EMS-PostgreSQL-Manager-Lite-22778.html.

Microsoft. 2013. Microsoft. [En línea] 2013. http://msdn.microsoft.com/es-es/library/bb399572.aspx.

Miller, Jeremy. 2011. MSDN Magazine. *La unidad de modelo de trabajo y persistencia Ignorance.* [En línea] 2011. [Citado el: 30 de marzo de 2013.] http://msdn.microsoft.com/es-es/magazine/dd882510.aspx#id0420003.

Montenegro Bernot, Yoandrys. 2011. EcuRed. *Sistemas Gestores de Bases de Datos.* [En línea] 11 de septiembre de 2011. [Citado el: 10 de noviembre de 2012.] http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos.

Palacios, Damian. 2012. Motor de persistencia NHibernate. [En línea] 7 de junio de 2012. [Citado el: 20 de noviembre de 2012.] http://www.slideshare.net/DamianPalacios/motor-de-persistencia-nhibernate.

Pérez Gamboa, Alexander. 2012. *Diseño de la Base de Datos para el Sitema de Integracion Bancarria.* La Habana : s.n., 2012.

Pérez González , Irael y Sosa Díaz, Madelaine. 2011. *Propuesta de optimización de la base de datos para el proyecto Sistema de Informatización de la Gestión de las Fiscalías.* Ciudad de la Habana : s.n., 2011.

Pérez Urrutia, Margelys. 2010. *Diseño e Implementación de la Base de Datos de la Plataforma de Servicios a Pozos Petroleros.* s.l. : La Habana, 2010.

Pérez Valdés, Damian. 2007. ¿Qué son las bases de datos? [En línea] 26 de octubre de 2007. http://www.maestrosdelweb.com/editorial/%C2%BFque-son-las-bases-de-datos/.

PostgreSQL – La historia hasta ahora | 2ndQuadrant. [En línea] [Citado el: 10 de noviembre de 2012.] http://www.2ndquadrant.com/es/postgresql-la-historia-hasta-ahora/.

Ramos Roman, Isabel y Dolado Cos, Javier . 2007. *Técnica Cuantitativa para la gestión en la ingeniría del software.* 2007.

Rodríguez Martín, Róger. 2011. Ecured. [En línea] 15 de abril de 2011. http://www.ecured.cu/index.php/ER/Studio.

Rodríguez Yunta, Luis. 2001. *Base de datos documentales: estructura y principios de uso.* Madrid : s.n., 2001.

Rubiano, Martha. 2009. UML Diagramas de despliegue. [En línea] 9 de octubre de 2009. [Citado el: 30 de marzo de 2013.] http://es.scribd.com/doc/19611312/diagramas-de-despliegue-2222.

Salazar, Cristian. 2012. Modelo de datos. [En línea] 28 de septiembre de 2012. [Citado el: 10 de enero de 2013.] http://www.slideshare.net/csalazarc/modelo-de-datos-14506949.

Smith, Gregory. 2010. *PostgreSQL 9.0.* s.l. : Team Leader, 2010.

Soriano, Amelia. Tipos de prueba. [En línea] http://carolina.terna.net/ingsw3/datos/Tipos_Prueba.pdf.

SQLManager. 2010. Sitio de descargas de software. [En línea] 2010. http://www.freedownloadmanager.org/es/downloads/Generador_de_Datos_de_SME_2005_para_PostgreSQL_37202_p/.

Strappazzon, Nicola C. 2010. Artículos de interes para el desarrollo de sistemas WEB. *Uso del modelo Atributos & Tipos (EAV) en el diseño de Base de Datos.* [En línea] 30 de octubre de 2010. http://www.swapbytes.com/2010_10_01_archive.html.

Trillo Lado, Raquel. 2010. *Taller de patrones de diseño.* s.l. : Universidad de Zaragoza., 2010.

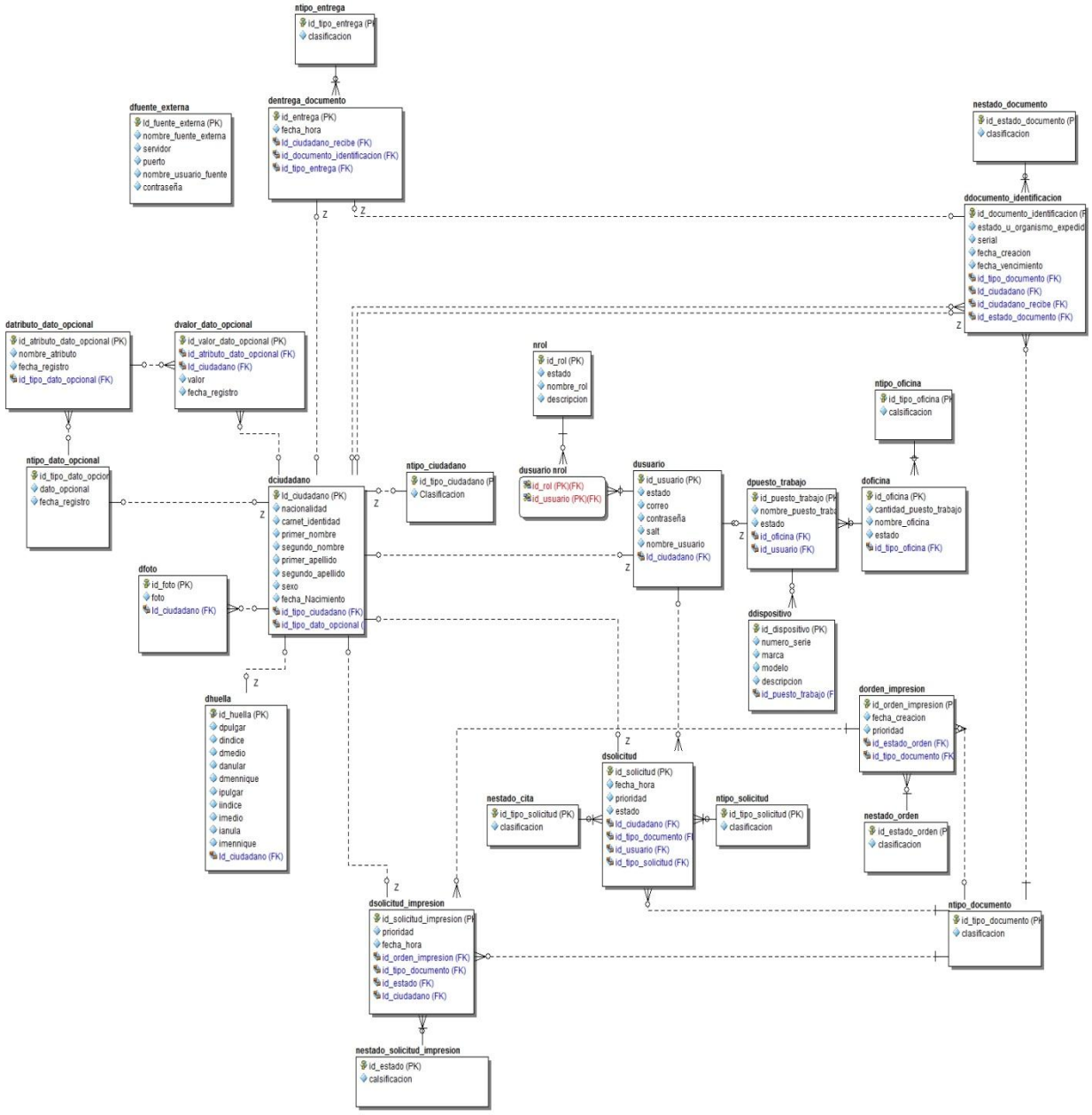
Vallez Pérez, Paqui. 2009. Emagister. *Informatica en la Administración pública.* [En línea] 31 de marzo de 2009. [Citado el: 15 de noviembre de 2011.] http://www.emagister.com/curso-informatica-administracion-publica-3/bases-datos-concepto-caracteristicas-funcionalidades.

Glosario de términos

- **API** o interfaz de programación de aplicaciones (del inglés Application Programming Interface) e un conjunto de funciones o procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como capa de abstracción.
- **Autentificación:** proceso por el cual el servidor de una base de datos establece la identidad del cliente mediante un usuario y una contraseña y por ende determinar si a la aplicación del cliente se le permite conectase al servidor PostgreSQL.
- **CASE** o conjunto de programas y ayudas (del inglés Computer Aided Software Engineering), que dan asistencia a los analistas, ingenieros y desarrolladores durante el ciclo de vida del software.
- **CPU:** unidad central de procesamiento (del inglés Central Process Unit).
- **Determinante:** es cualquier atributo o conjunto de atributos del cual depende funcional y completamente otro atributo.
- **Framework:** es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, para ayudar a desarrollar y unir los diferentes componentes de un proyecto.
- **MD5** o (del acrónimo Message-Digest-Algorithm)**:** es un algoritmo de reducción criptográfico compuesto por 128 bits y ampliamente utilizado por usuarios para la autentificación.
- **ORM** o (Object Relation Mapper, en inglés): es una técnica de programación que permite convertir datos entre el sistema de tipos utilizado en el lenguaje de programación orientado a objetos y el de una BD relacional.
- **UUID** o identificador universal único: es un identificador utilizado en el desarrollo de software, compuesto por 128 bit o 32 dígitos hexadecimales.
- **Workflow:** es un flujo o estación de trabajo, cómoda para la plataforma ASP.NET.

Anexos

Anexo 1: Modelo de Datos.



Anexo 2: Diccionario de Datos.

Nombre de la entidad		*dciudadano*					
Descripción de la entidad		*Es cualquiera persona que de alguna forma está vinculado a la institución que necesite un documento de identificación para acceder a la misma.*					
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
id_cuidadano	*Llave Primaria*	*uuid*	*no*			*no*	*si*
sexo		*char*	*no*	*1*		*no*	*no*
fecha_nacimiento		*timestamp*	*no*			*no*	*no*
primer_nombre		*varchar*	*no*	*15*		*no*	*no*
segundo_nombre		*varchar*	*si*	*15*		*no*	*no*
primer_apellido		*varchar*	*no*	*15*		*no*	*no*
segundo_apellido		*varchar*	*no*	*15*		*no*	*no*
carnet_identidad		*varchar*	*no*	*11*		*no*	*si*
nacionalidad		*varchar*	*si*	*20*		*si*	*no*
id_tipo_ciudadano	*Llave Foránea*	*integer*	*no*			*si*	*no*
Id_tipo_dato_opcional	*Llave foránea*	*integer*	*si*			*si*	*no*

Tabla 4: Relación de atributos de un ciudadano.

Nombre de la entidad		*ntipo_ciudadano*					
Descripción de la entidad		*Da una clasificación de cada ciudadano que está registrado en el sistema.*					
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
id_tipo_ciudadano	*Llave Primaria*	*serial*	*no*			*no*	*Si*
clasificacion		*varchar*	*no*	*20*		*no*	*si*

Tabla 5: Relación de atributos de un tipo de ciudadano.

Nombre de la entidad		*dusuario*					
Descripción de la entidad		*Es el nombre único que recibe la persona cuando es registrado en el sistema a través del cual podrá autenticarse y dependiendo de sus roles, podrá efectuar determinadas operaciones y tener acceso a los diferentes recursos, en función de los permisos que le sean asignados.*					
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
id_usuario	*Llave Primaria*	*uuid*	*no*			*no*	*si*
correo		*varchar*	*no*	*40*		*no*	*si*
contraseña		*varchar*	*si*	*20*		*si*	*no*
salt		*varchar*	*si*	*5*		*si*	*no*
nombre_usuario		*varchar*	*si*	*15*		*no*	*si*

Tabla 6: Relación de atributos de un usuario.

Nombre de la entidad		*dsolicitud*					
Descripción de la entidad		*Necesidad del ciudadano de la creación de un documento de identificación planteada de manera formal (escrita o digital).*					
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
id_solicitud	*Llave Primaria*	*uuid*	*no*			*no*	*si*
fecha_hora		*timestamp*	*no*			*si*	*no*
prioridad		*integer*	*si*			*si*	*no*
estado		*char*	*si*	*1*		*si*	*no*
id_tipo_documento	*Llave Foránea*	*integer*	*no*			*si*	*no*
id_cuidadano	*Llave Foránea*	*uuid*	*no*			*si*	*no*
id_usuario_ciudadano	*Llave Foránea*	*uuid*	*no*			*si*	*no*
id_tipo_solicitud	*Llave Foránea*	*integer*	*no*			*si*	*no*

Tabla 7: Relación de atributos de una solicitud.

Nombre de la entidad		*ddocumento_identificación*					
Descripción de la entidad		Es lo que va a permitir identificar al ciudadano.					
Nombre del atributo	**Descripción**	**Tipo**	**Puede ser nulo**	**Restricciones**		**Criterio de Selección**	
				Clases válidas	**Clases no válidas**	**Múltiple**	**Única**
id_documento_identificación		*uuid*	*no*			*no*	*si*
serial		*varchar*	*no*	*20*		*no*	*si*
fecha_creación		*timestamp*	*no*			*no*	*si*
fecha_vencimiento		*timestamp*	*no*			*no*	*si*
id_cuidadano	*Llave Foránea*	*uuid*	*no*			*no*	*si*
id_estado_documento	*Llave Foránea*	*serial*	*no*			*no*	*si*
id_tipo_documento	*Llave Foránea*	*serial*	*no*			*no*	*si*

Tabla 8: Relación de atributos de un documento de identificación.

Para más información remitirse a la Plantilla del Diccionario de Datos del expediente de proyecto de Centro de Identificación y Seguridad Digital (CISED).

Anexo 3: Tipos de datos migrados.

Parámetros modificados.	Oracle	PostgreSQL	Función en PostgreSQL
Proveedor de datos	OracleDataAcces	Npgsql	
Parámetro	OracleParameter	NpgsqlParameter	Representa un parámetro
Comand	OracleComand	NpgsqlComand	Representa una función que se ejecuta en la base datos la cual posee parámetros
DbType	OracleDbType	NpgsqlDbType	Tipos de datos que posee la librería
Refcursor	DbType.RefCursor	DbType.Refcursor	Cursor
Tipo de dato	BLOB	Bytea	Se utiliza para almacenar archivos grandes(imágenes)
Tipo de dato	Varchar2	Varchar	Cadena String
Tipo de dato	Int16	numeric	Cadena de números
Funciones utilizadas en los lenguajes PL/SQL y PL/pgSQL	ROWNUM	limit	Limita el número de registros de una consulta
Funciones utilizadas en los lenguajes PL/SQL y PL/pgSQL	NVL	FOUND	Devuelve true si se devuelve al menos una columna y false si es null
Tipo de dato	Sysdate	Current_date	Fecha actual
Tipo de dato	number	numeric	Cadena de números
Tipo de dato	raw	Bytea	Almacena datos binarios
Tipo de dato	CLOB	Text	Modo texto

Tabla 9: Tipos de datos modificados.

Anexo 4: Guía de pasos para realizar las pruebas de carga y estrés a la BD utilizando Apache JMeter.

- Paso 1: Se debe contar con la herramienta Apache JMeter en cualquiera de sus versiones, en este caso se utilizó la versión 2.3.1 y su driver correspondiente.
- Paso 2: Se adicionan al Plan de Prueba los directorios o jar requeridos como lo muestra la siguiente imagen.

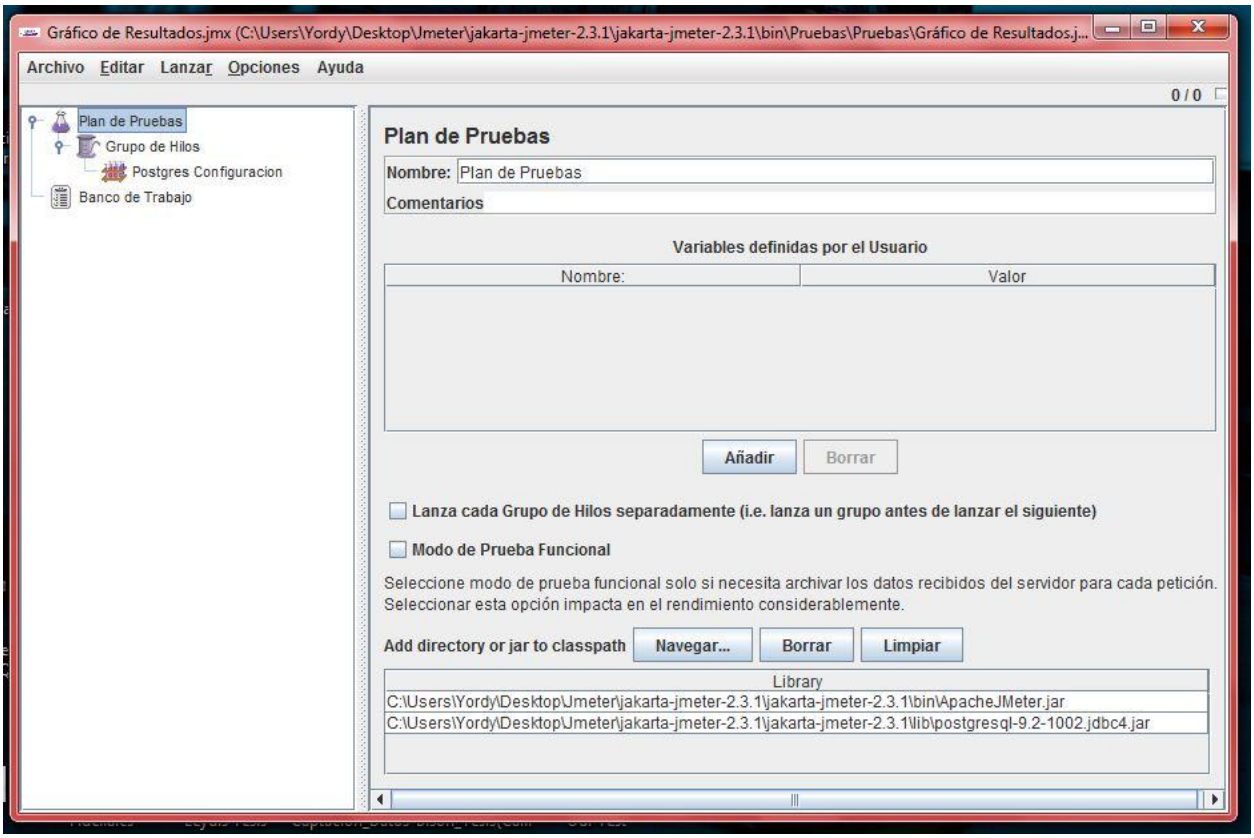


Figura 19: Creación del Plan de Prueba.

- Paso 3: Se sigue la ruta Plan de Prueba/Añadir/Grupo de Hilos, donde se define la cantidad de usuarios concurrentes o número de hilos, además del período de subida(en segundos), que no es más que el tiempo en segundo que se va a demorar el JMeter en lanzar todos los hilos (en este caso se seleccionan 150 hilos y el período de subida es de 1 segundo, entonces cada hilo se ejecutará 0,1 segundo después de que el hilo anterior haya sido lanzado); se le dio valor 1 al controlador de bucles, encargado de realizar tantas veces como se le especifica con el número.

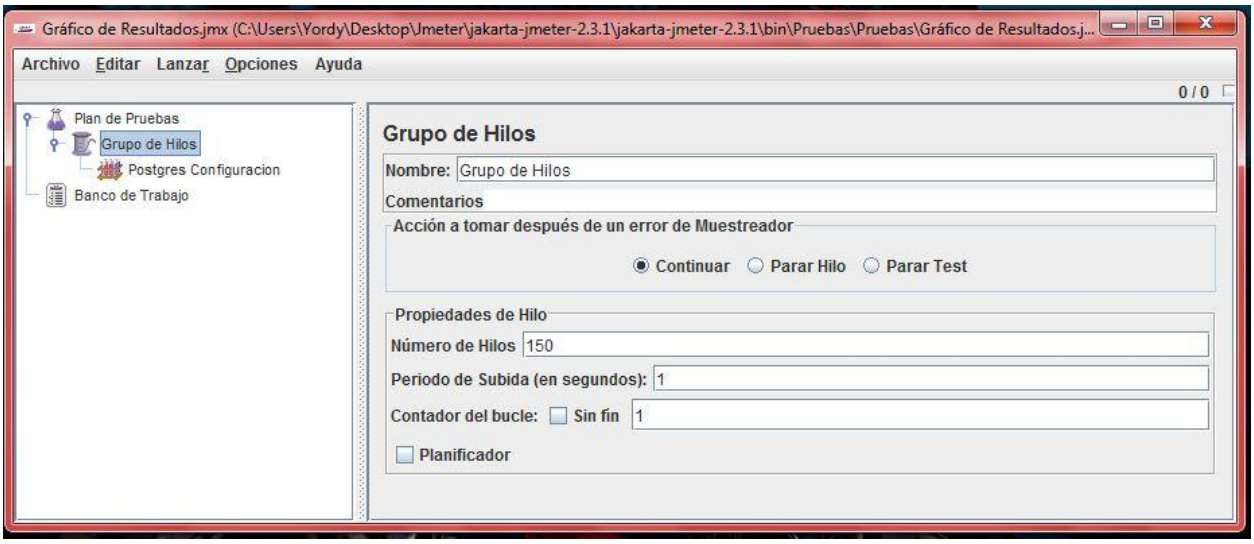


Figura 20: Creación del Grupo de Hilos.

- Se traza la ruta Grupo de Hilos/Añadir/Elementos de Configuración/Configuración de la Conexión JDBC.

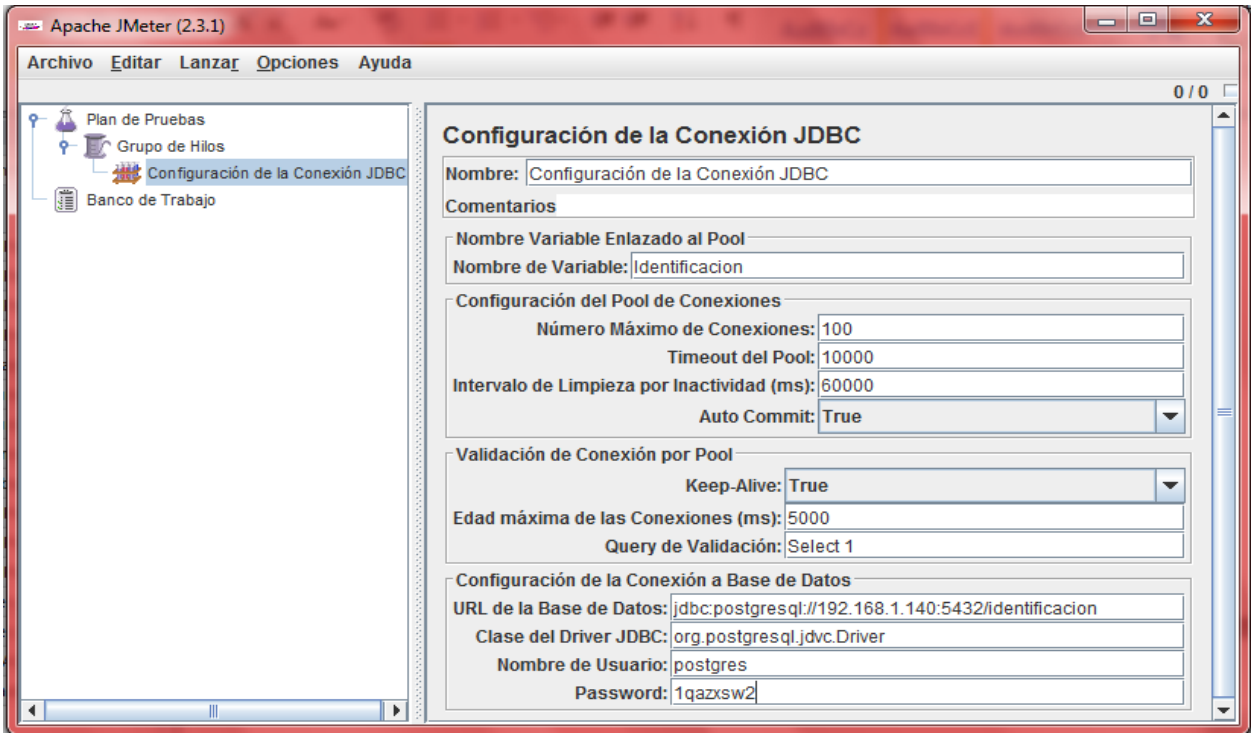


Figura 21: Configuración de la conexión JDBC.

- Se traza la ruta Grupo de Hilos/Añadir/Muestreador/Petición JDBC. Se elige Prueba 1 como Nombre de la Petición e Identificación como Nombre de Variable. Posteriormente se especifica el tipo de consulta o query a realizar.

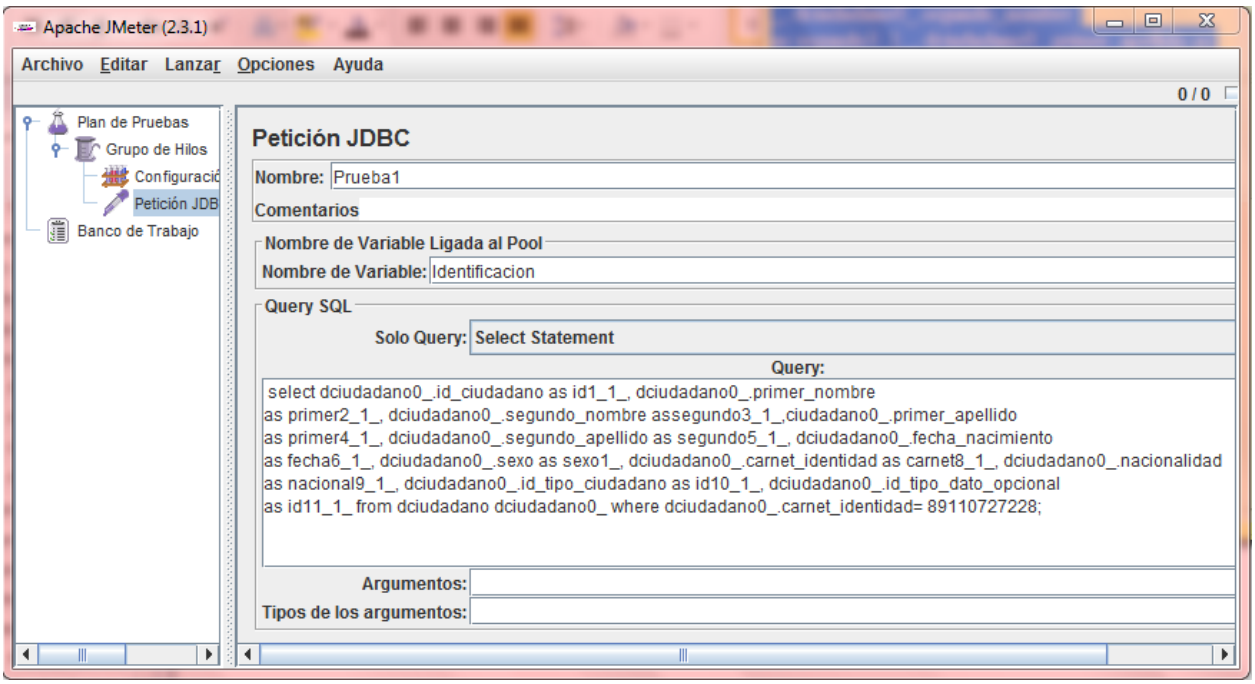


Figura 22: Petición JDBC.

- Se elige la ruta Grupo de Hilos/Añadir/Listener/ [Informe Agregado, Grafico de Resultados, Ver Árbol de Resultados]. Estas opciones permiten graficar de forma fácil los resultados mediante las etiquetas definidas a la hora de realizar las pruebas de carga y estrés.
- Finalmente se ejecuta la herramienta y se valoran los resultados expuestos.

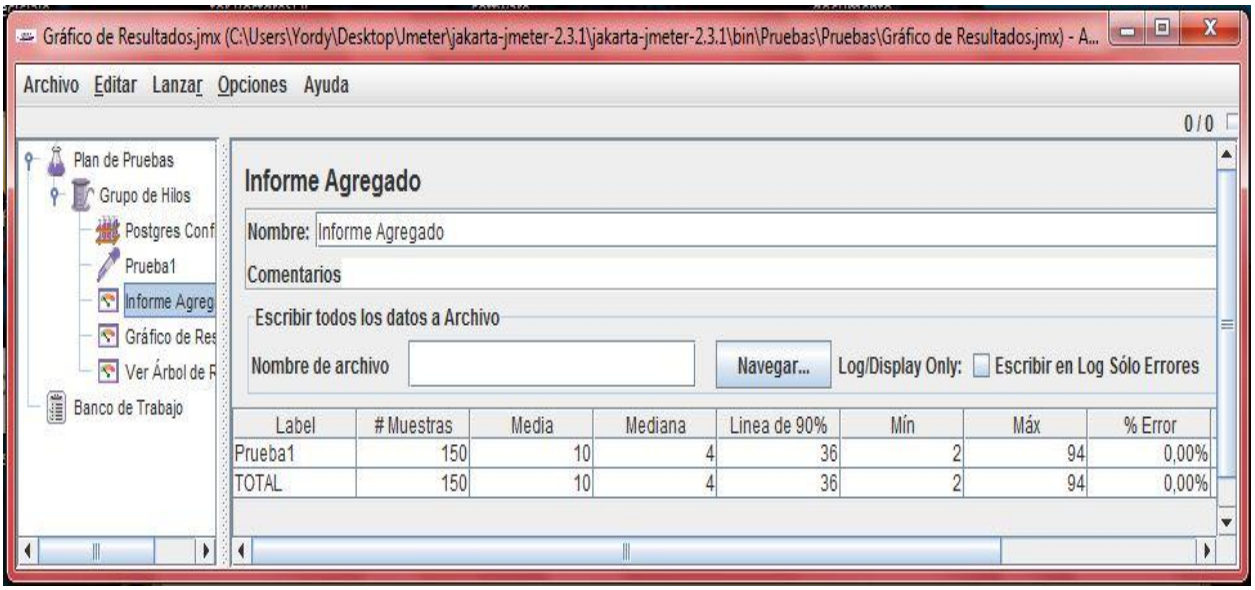


Figura 23: Informe Agregado (Primera Parte).

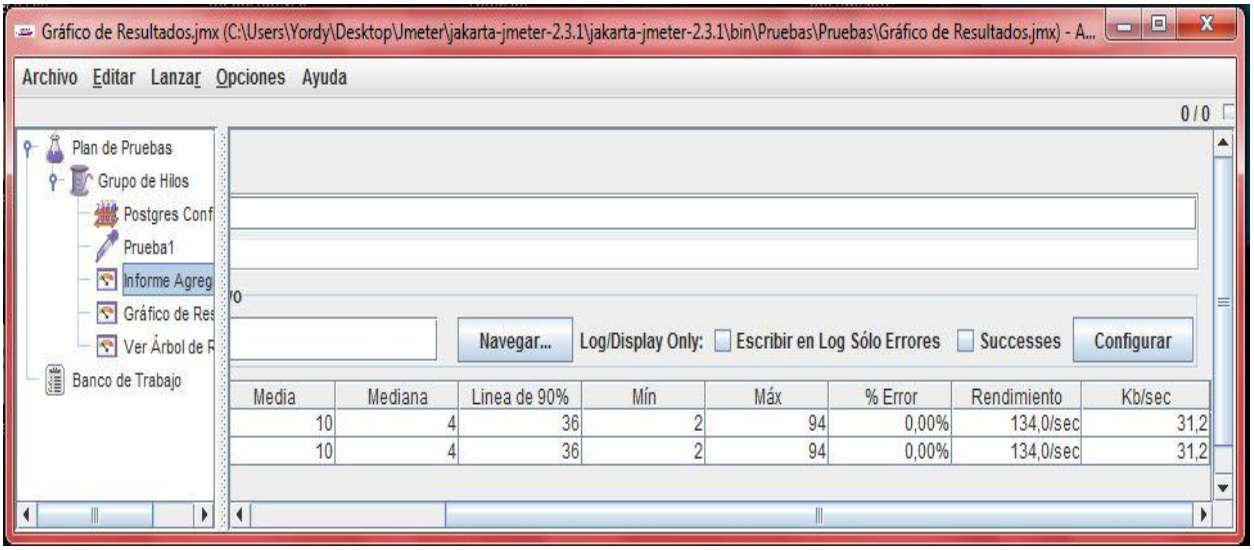


Figura 24: Informe Agregado (Segunda Parte).