Instituto Superior Minero Metalúrgico
"Dr. Antonio Núñez Jiménez"
Facultad: Geología – Minas
Departamento de Informática

# Trabajo de Diploma

para optar por el Título de

# Ingeniero Informático

**Título:** Infraestructura de desarrollo de Software para el Proyecto "Herramientas Inteligentes para la Gestión del Desarrollo Local en las Comunidades".

**Autor:** Robin Díaz Matos

**Tutor:** Ing. Eloy R. Jiménez Iglesias

Moa, 2012
"Año 54 de la Revolución"

# Declaración de autoría

**Declaro que soy el único autor de este trabajo y autorizo al Instituto Superior Minero Metalúrgico de Moa "Dr. Antonio Núñez Jiménez" y al Departamento de Informática para que hagan el uso que estimen pertinente con este trabajo.**

**Para que así conste firmamos la presente a los _____ días del mes de _____ del 2012.**

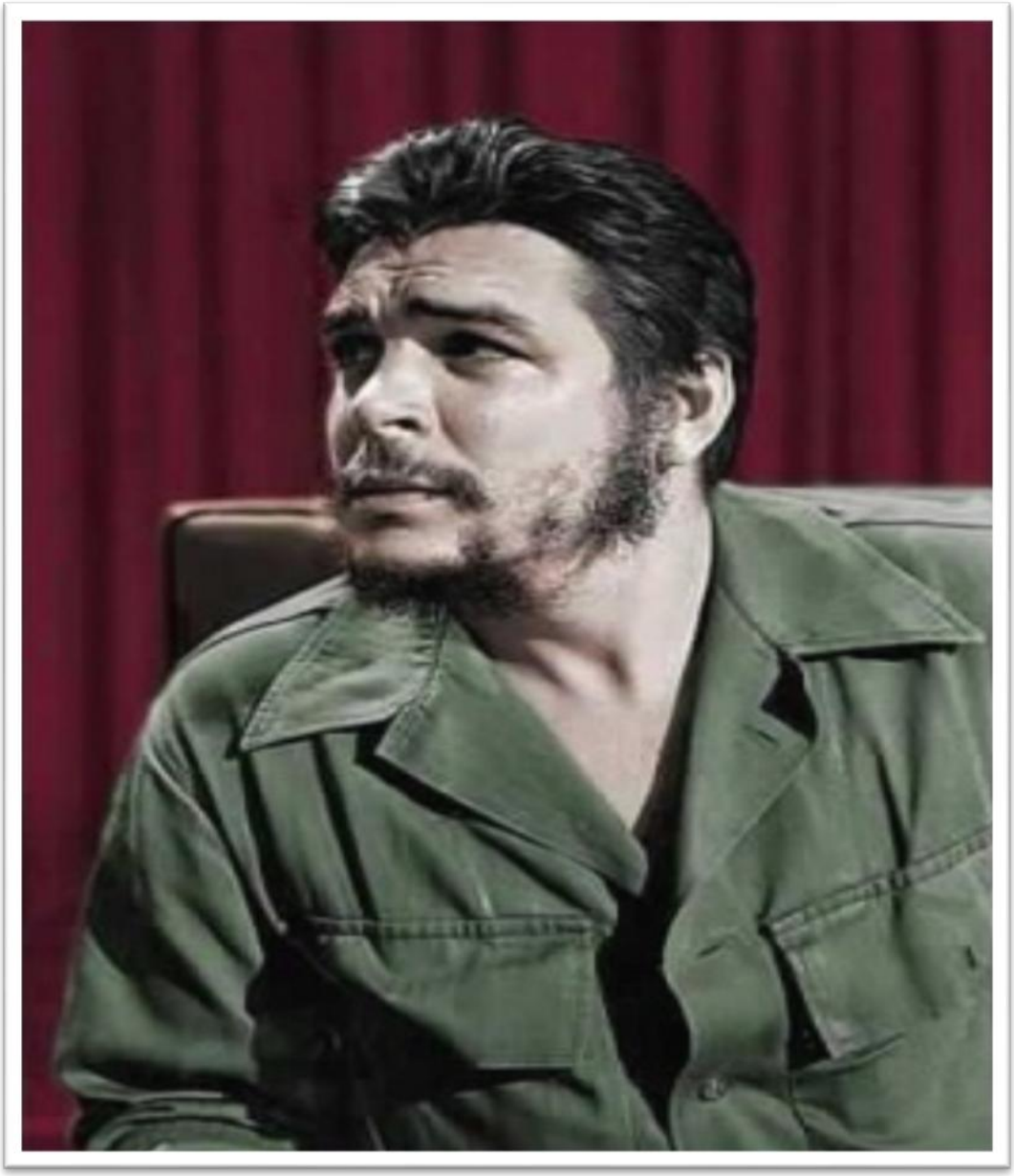
**Robin Díaz Matos** _____

**Firma autor**

**Ing. Eloy R. Jiménez Iglesias** _____

**Firma tutor**

# *Pensamiento*



***"Lo último que se pierde no son las esperanzas sino la fuerza de voluntad; el hombre que no tenga fuerza de voluntad que no sueñe con las esperanzas."***

***Ernesto Che Guevara***

# *Dedicatoria*

***Este trabajo va dedicado especialmente a mis Padres,***

***Teodoro Díaz, por haberme apoyado siempre en mis deseos de seguirme superándome y dar siempre lo mejor de sí para que mi sueño fuese posible.***

***Y a Doralis Matos, porque me enseñó que todo el mundo tiene algo que aprender.***

***A mis Abuelos Maternos que ya no están conmigo, pero que siempre me inculcaron los buenos principios.***

***A mis Abuelos Paternos, por darme siempre buenos consejos.***

***A mis tíos y tías, porque sé que son las personas que más confiaron en mi educación.***

***A mi hermana Sahily, quien siempre ha confiado en mí y siempre ha estado a mi lado en todo momento.***

***Y a todas las personas que de una forma u otra me ayudaron en el transcurso de mi desarrollo como ser humano.***

# *Agradecimientos*

***Primeramente quisiera agradecer a lo más importante de mi vida; mis padres y mi hermana, que sin ellos nada de esto hubiera sido posible.***

***A mi familia en general por darme siempre apoyo y confianza cuando la necesité.***

***A mi grupo Info07, que es sin duda alguna el mejor y más loco grupo del mundo.***

***A Eloy, por su ejemplar tutoría y a todos los profesores que de una forma u otra aportaron un granito en la realización de este trabajo.***

***A mis vecinos y amigos del barrio que siempre estuvieron atentos en mi desarrollo: Aurora, Nena, Cecilio, Cecilin, Yovito, Eider, Magali, Arelis, Jadier, Daimis, Adriano, Eidier, Alexei, Gotero, Yoa, Alexis, Yami, Jorgito, Indira, Rubiel.***

***Y a todas las personas que me brindaron su ayuda y apoyo durante todo mi desarrollo como estudiante.***

# Resumen

A medida que aumenta la complejidad de los sistemas de software surgen nuevos aspectos del desarrollo de aplicaciones que hasta ese momento no se habían tenido en cuenta, al menos de una forma explícita. La Ingeniería del Software ha ido respondiendo a esta situación con el desarrollo de nuevos modelos, notaciones, técnicas y métodos.

Dentro de esta tendencia se encuadra el creciente interés por los aspectos arquitectónicos del software. Estos aspectos se refieren a todo lo relativo a la estructura de alto nivel de los sistemas: su organización en subsistemas y la relación entre estos; el desarrollo de aplicaciones caracterizadas por presentar una arquitectura común; el mantenimiento y la evolución. En efecto, un aspecto crítico a la hora de desarrollar sistemas de software complejos es el diseño de su arquitectura, representada como un conjunto de elementos computacionales y de datos interrelacionados de un modo determinado.

Por tanto este trabajo tiene como objetivo concreto diseñar una infraestructura o arquitectura de software para el diseño y/o implementación de los módulos que conforman el Proyecto “Herramientas Inteligentes para la Gestión de Desarrollo Local en las Comunidades”. El proyecto se centra principalmente en la apariencia de los mismos, por lo que se realizó el análisis y diseño del módulo administrativo (seguridad), quedando expuestos en este trabajo un conjunto de elementos y un lenguaje común de comunicación entre las personas que participan en el proyecto. Se logra una estructura conceptual y tecnológica, la cual se basa en organizar y desarrollar los módulos que conforman este Proyecto.

# Summary

At the same time that the software complexity systems increase rise up some other new application aspects in development which up to this time were not taken into considerations, at the very least on an explicit manner. Software Engineering has found some answers to those new situations by developing new models, notations, techniques and methods.

In such tendency, it is encouraged current interests for architectural aspects of the software. These aspects shall refer to all high-level system structure related; their subsystem organization and the link or relationship between them, their maintenance and work progress evolution. In fact, a very critical aspect at the time of developing complex software system is the architecture's design, represented by a group of computer elements and interrelated data mode, in a determine manner.

That is why; our research target is to design an infrastructure or a software architecture for designing and/or module implementations that may integrate our Project. "Herramientas Inteligentes para la Gestión de Desarrollo Local en las Comunidades", by focussing mainly on their appearance, for such reason the analysis and the design administration module were executed, (security) so, we are exposing through out this research a group of elements and a mutual communication language for the persons who integrates this project, while achieving a conceptual and a technological structures to be the organization and development base for the modules which are integrating this Project.

# Índice

# Índice de Tablas

# Índice de Figuras

# Introducción

En el ámbito del software cada vez es más común escuchar el término “arquitectura de software”, y encontrar oportunidades de empleo para “arquitectos de software”. Aun así, este concepto tiende a ser malentendido y la falta de comprensión al respecto de sus principios frecuentemente repercute de manera negativa en la construcción de sistemas de software. La arquitectura de software se refiere a la estructuración del sistema que, idealmente, se crea en etapas tempranas del desarrollo. Esta estructuración representa un diseño de alto nivel del sistema que tiene dos propósitos primarios: satisfacer los atributos de calidad (desempeño, seguridad, modificabilidad), y servir como guía en el desarrollo. El no crear este diseño desde etapas tempranas del desarrollo puede limitar severamente a que el producto final satisfaga las necesidades de los clientes. Además, el costo de las correcciones relacionadas con problemas en la arquitectura es muy elevado. Es así que la arquitectura de software juega un papel fundamental dentro del desarrollo.

La arquitectura de software es de especial importancia porque la manera en que se estructura un sistema tiene un impacto directo sobre la capacidad de este para satisfacer lo que se conoce como los atributos de calidad del sistema. La manera en que se estructura un sistema permitirá o impedirá que se satisfagan los atributos de calidad. Dentro de un proyecto de desarrollo, e independientemente de la metodología que se utilice, se puede hablar de “desarrollo de la arquitectura de software”. Este desarrollo, que precede a la construcción del sistema, está dividido en las siguientes etapas: requerimientos, diseño, documentación y evaluación. Cabe señalar que las actividades relacionadas con el desarrollo de la arquitectura de software generalmente forman parte de las actividades definidas dentro de las metodologías de desarrollo. El objetivo principal de la Arquitectura de Software es aportar elementos que ayuden a la toma de decisiones y, al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la

comunicación entre los equipos que participen en un proyecto. Para conseguirlo, la Arquitectura de Software construye abstracciones, materializándolas en forma de diagramas comentados.

En la carrera de Informática del Instituto Superior Minero Metalúrgico de Moa se ha creado un grupo de trabajo para la realización de software, pero en estos momentos no se cuenta con una infraestructura que nos ayude a la toma de decisiones con respecto a la confección de software para el proyecto Herramientas Inteligentes para la Gestión del Desarrollo Local en las Comunidades. Por lo que no se puede garantizar que el producto final cumpla con todos los estándares de calidad y que satisfaga las necesidades finales de los usuarios, ya que ponerse a programar sin contar con los diagramas es una mala idea, porque las decisiones se toman a la hora de codificar y esto dificulta el trabajo en equipo. Si no se realiza una adecuada selección de los elementos, no se pueden establecer los principios y métodos para obtener un producto de modo rentable, fiable y que trabaje en máquinas reales, por otra parte para diseñar el sistema hay que tener en cuenta tres aspectos fundamentales, la presentación de la información, la funcionalidad de la aplicación y la arquitectura de software.

Por otra parte no realizar una arquitectura de software puede conllevar a que se dificulte el trabajo al ser usado por personas que no cuenten con conocimientos básicos de computación. Además, la interfaz de la aplicación puede ser un poco complicada dificultando la navegación entre las funcionalidades y que el sistema no pueda ser ejecutado en diferentes sistema operativos. No contar con la documentación necesaria puede provocar que no se puedan realizar labores de mantenimiento y mejoras en caso que fuera necesario, así como que el hardware (Cliente) no cuente con los requerimientos mínimos para su ejecución.

A partir de esta situación surge el **problema científico**:

¿Cómo favorecer la estructuración del software del proyecto Herramientas Inteligentes para la Gestión del Desarrollo Local en las Comunidades?

El **objeto** de esta investigación es:

- Frameworks para el desarrollo de Software.

Dicho objeto enmarca el **campo de acción** de esta investigación como:

- La Infraestructura de Software que facilite la creación de aplicaciones Desktop.

El **objetivo** principal de esta investigación es:

- Diseñar una Infraestructura de Software para la confección de software del Proyecto Herramientas Inteligentes para la Gestión del Desarrollo Local en las Comunidades.

Los **objetivos específicos** que se persiguen son:

- Elaborar el Marco Teórico Conceptual para la confección de Software.
- Realizar un estudio sobre las arquitecturas de software basadas en capas.
- Diseñar un Framework de Software para el Proyecto Herramientas Inteligentes para la Gestión del Desarrollo Local en las Comunidades.
- Implementar el Módulo Administrativo para la validación del framework diseñado.
- Documentar el sistema.

**Idea a defender**:

- Si se crea un Framework que posibilite una mejor estructuración del software, aumenta la calidad de los productos software y una mayor estandarización de los mismos.

Para lograr el objetivo propuesto se han trazado una serie de **tareas** a realizar:

- Buscar referencias bibliográficas.
- Estudiar las condiciones tecnológicas disponibles en el proyecto y en los posibles usuarios del software.
- Investigar las distintas arquitecturas de software basadas en capas.
- Seleccionar las tecnologías a utilizar en la confección del software.
- Construir la Framework de Software a utilizar.
- Realizar el modelo del negocio y levantamiento de requisitos.
- Elaborar el modelo de análisis y diseño del sistema.
- Diseñar la Base de Datos.
- Efectuar la implementación del Módulo Administrativo.
- Confeccionar el Manual de Usuario.

Para complementar estas tareas se han empleado **métodos** teóricos y empíricos de la investigación científica. Entre los métodos empíricos usados podemos citar la observación, entrevista y el análisis de documentos para la recopilación de la información. La observación se utilizó para ver la funcionalidad de los productos software y el comportamiento del problema. La entrevista; donde se realizó una conversación planificada con el fin de obtener información individual o colectiva y determinar los principales requerimientos del software. Mediante el análisis de documentos se supo cómo funcionan actualmente los procesos de construcción de software.

Los métodos teóricos proporcionaran calidad en la investigación. Entre los teóricos se usó el método histórico y lógico donde se realiza la búsqueda de antecedentes que pueda tener el software, si existe actualmente alguno en funcionamiento con las mismas características, la situación actual de los software de gestión y sus tendencias y las herramientas más potentes y usadas en el mundo para su confección. En el desarrollo del proceso de investigación se usaron el análisis y síntesis para la recopilación y el procesamiento de la información obtenida en los

métodos empíricos y arribar a las conclusiones de la investigación. La modelación permitió realizar un estudio de la construcción de software a nivel mundial y nacional.

El siguiente trabajo está compuesto por **Introducción**, **4 Capítulos**, **Conclusiones**, **Recomendaciones**, **Referencias Bibliográficas, Bibliografía, Glosario de Términos y Anexos**.

En el **Capítulo 1: Fundamentación Teórica**, se muestra en detalles el estudio realizado sobre las diferentes herramientas, tecnologías y metodologías que fueron analizadas con el fin de tomar decisiones para la futura construcción del sistema.

En el **Capítulo 2: Modelo de Negocio**, se muestra modelo del negocio, sus actores, trabajadores, los casos de uso del negocio y la descripción de los mismos, así como las especificaciones de los requerimientos funcionales y no funcionales.

En el **Capítulo 3: Análisis y Diseño**, se hace referencia al Modelo de Análisis y al Modelo de Diseño. Se definen los actores, se realizan las descripciones de los casos de uso del sistema, se muestra el diagrama de análisis, el diagrama de clases del diseño, el diagrama de secuencia y es definida la arquitectura que se empleará.

En el **Capítulo 4: Implementación**, se muestran el diseño de la Base de Datos, el Modelo de Despliegue y el Diagrama de Componentes para el Módulo Administrativo.

# Capítulo 1: Fundamentación Teórica

## 1.1 Introducción

En este Capítulo se muestra en detalles el estudio realizado sobre las herramientas, tecnologías y metodologías que fueron analizadas con el fin de realizar la futura construcción del sistema, así como un estudio de las arquitecturas de software basada en n-capas.

## 1.2 Estado del Arte

### 1.2.1 ¿Que es un Framework?

En el Desarrollo de Software es muy común encontrarse con el concepto de Framework cuya traducción seria “Marco de Trabajo o Infraestructura de Software”, pero sin embargo no es fácil de definir. Cualquier que cuente con experiencia programando es muy posible que esté utilizando su propio framework (aunque no lo llame así), por tanto se pude decir que un framework es un esquema (un esqueleto, un patrón) para el desarrollo y/o implementación de una aplicación. También es posible que el framework defina una estructura para una aplicación completa, o bien sólo se centre en un aspecto de ella.

En otras palabras un Framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones.

### 1.2.1.1 ¿Qué ventajas tiene utilizar un Framework?

1. El programador no necesita plantearse una estructura global de la aplicación, sino que el framework le proporciona un esqueleto que hay que "rellenar".
2. Facilita la colaboración. Cualquiera que haya tenido que "pelearse" con el código fuente de otro programador (¡o incluso con el propio, pasado algún tiempo!) sabrá lo difícil que es entenderlo y modificarlo; por tanto, todo lo que sea definir y estandarizar va a ahorrar tiempo y trabajo a los desarrollos colaborativos.
3. Es más fácil encontrar herramientas (utilidades, librerías) adaptadas al framework concreto para facilitar el desarrollo.
4. Facilita tanto el desarrollo como el mantenimiento de una aplicación.
5. La elección de un framework vendrá marcada por:
    - El tipo de aplicación a desarrollar.
    - El lenguaje de programación y otras tecnologías concretas como base de datos, sistema operativo etc.

## 1.2.2 ¿Qué es la Arquitectura del Software?

Existen muchas definiciones de Arquitectura del Software y no parece que ninguna de ellas haya sido totalmente aceptada. En un sentido amplio podríamos estar de acuerdo en que la Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de:

- Definir los módulos principales.
- Definir las responsabilidades que tendrá cada uno de estos módulos.
- Definir la interacción que existirá entre dichos módulos:
    - Control y flujo de datos.

- Secuenciación de la información Protocolos de interacción y comunicación.
- Ubicación en el hardware.

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

El modelo n-tier (n-capas) de informática distribuida ha emergido como la arquitectura predominante para la construcción de aplicaciones multiplataforma. Este cambio radical en los modelos de computación, desde los sistemas monolíticos basados en mainframe y los tradicionales sistemas cliente-servidor, hacia sistemas distribuidos multiplataforma altamente modulables, representa simplemente lo que está por llegar en el mundo del desarrollo de aplicaciones. Como tecnología, las arquitecturas de n-capas proporcionan una gran cantidad de beneficios para las empresas que necesitan soluciones flexibles y fiables para resolver complejos problemas inmersos en cambios constantes.

El crecimiento de la ciencia y la tecnología, combinado con el florecimiento de Internet y las economías basadas en mercados globales abiertos, contribuirán de forma conjunta a que la economía del Siglo XXI nos conduzca a una prosperidad sin precedentes. Algo obvio en nuestros días es que el futuro de la informática no es el ordenador de escritorio, sino Internet. La arquitectura emergente basada en los estándares Internet es "Navegador/Red", significando que desde ahora, el foco estará en la informática basada en Red. Serán aplicaciones que únicamente existan en las redes y que estarán disponibles para cualquiera, en cualquier lugar y en cualquier momento.

Todas las aplicaciones basadas en n-capas permitirán trabajar con clientes ligeros, tal como navegadores de Internet, WebTV, Teléfonos Inteligentes, PDAs (Personal Digital Assistants o Asistentes Personales Digitales) y muchos otros

dispositivos preparados para conectarse a Internet. De este modo, las arquitecturas de n-capas se están posicionando rápidamente como la piedra angular de los desarrollos de aplicaciones empresariales y las compañías están adoptando esta estrategia a una velocidad de vértigo como mecanismo de posicionamiento en la economía emergente que tiene su base en la red (lo que se ha venido a denominar "Nueva Economía").

Actualmente, la Red (Internet, intranets y extranets) es el ordenador o, como diría Sun Microsystems, el ordenador es la Red. Este paradigma está creando un cambio fundamental en los modelos de computación que, a su vez, proporciona desafíos y oportunidades como nunca antes se habían producido. Las arquitecturas basadas en n-capas permiten a los componentes de negocio correr en una LAN, WAN o Internet. Esto significa que cualquiera con un ordenador y conexión a la Red) posee toda la funcionalidad que tendría si se encontrase delante de su sistema de escritorio.

Las arquitecturas empresariales de n-capas se están convirtiendo en la nueva base para el desarrollo de aplicaciones de misión crítica y ofrecen la única arquitectura funcional para la siguiente generación de soluciones informáticas distribuidas basadas en Internet. Las aplicaciones industriales basadas en n-capas pueden ayudar a las compañías a desarrollar un nuevo núcleo de habilidades en prácticamente todo, desde la gestión del conocimiento hasta los sistemas relacionados con comercio electrónico.

La estrategia de desarrollo de aplicaciones para el siglo XXI
a diferencia de lo que se pudiera pensar, el desarrollo en n-capas no es un producto o un estándar, es un concepto estratégico que ayuda a la construcción y despliegue lógico de un sistema distribuido. Los sistemas de n-capas subdivididos ayudan a facilitar el desarrollo rápido de aplicaciones y su posterior despliegue, con beneficios incrementales fruto de los esfuerzos del desarrollo en paralelo coordinado resultando un enorme decremento del tiempo de desarrollo y de sus

costes. Muchas de las aplicaciones de e-business que se utilizan actualmente simplemente utilizan un navegador de Internet como cliente ligero que implementa una interfaz universal. Una arquitectura basada en clientes ligeros desplaza la capa de presentación de la aplicación en el lado del cliente, mientras que la lógica de negocio y los datos residen en el middleware y los servidores de back-end. El diseño para clientes ligeros minimiza los problemas de despliegue de las aplicaciones, mientras que maximiza la accesibilidad a la misma desde una amplia variedad de plataformas heterogéneas. Los frameworks basados en n-capas se crean para obtener las ventajas de los estándares abiertos de la industria que permiten a las aplicaciones resultantes operar en entornos distribuidos multiplataforma.

### 1.2.2.1 La Evolución

Las arquitecturas basadas en n-capas son el siguiente paso lógico en un proceso de evolución, el cual está basado en las arquitecturas convencionales cliente-servidor (2 y 3 capas). Los sistemas de n-capas utilizan técnicas de desarrollo basadas en componentes combinados con los estándares abiertos de Internet, para crear aplicaciones multiplataforma muy potentes con bajos costes, fáciles de mantener y con gran efectividad. Lo que realmente es nuevo en el modelo de n-capas es la posibilidad de distribuir objetos independientes sobre el número de capas que sean necesarias y enlazarlas dinámicamente, cuando sea necesario, para proporcionar una flexibilidad ilimitada a la aplicación.

### 1.2.2.2 La Revolución

N-Capas forma parte también de un revolucionario proceso, actualmente en desarrollo, basado en la aplicación de estas nuevas tecnologías (componentes y estándares de Internet). Estas tecnologías son los bloques para crear Software de Negocio y Sistemas de Información adaptables que ayuden a las empresas a integrar todos sus sistemas de Tecnologías de la Información, así como las inversiones realizadas en éstos, mientras que obtienen una ventaja clara en el uso

de Internet. Las empresas exitosas del futuro serán aquellas que se adapten mejor a un mundo conectado. Los framework de n-capas utilizan herramientas basadas en Internet que proporcionan a los clientes la adopción de las últimas y más potentes tecnologías que proporcionarán claros avances competitivos. Las empresas hoy en día (no importa dónde estén, qué tamaño tengan o en qué industria se encuentren) deben ser capaces de implementar las últimas prácticas de negocio, ventas y estrategias de distribución, procesos de fabricación, logística de la cadena de suministro, etc. Por eso, los sistemas basados en n-capas ayudan rápidamente a cambiar los negocios para experimentar la compartición sin restricciones de datos a lo largo de aplicaciones o fuentes de datos en la empresa.

El desarrollo de aplicaciones en n-capas es un proceso iterativo de división del problema en piezas manejables denominadas componentes. Estos componentes, o "Componentes de Negocio - Business Objects" son "modelos software" basados típicamente en la "vista" de un objeto real, evento o proceso de negocio. Los componentes software individuales pueden formar parte y adaptarse tanto de estructuras independientes como de sistemas colaborativos.

El diseño de aplicaciones en n-capas es ideal para la creación de sistemas adaptables, donde cada componente puede ser utilizado y reutilizado en nuevas combinaciones para satisfacer requisitos de negocio dinámicos. Esto permite a los desarrolladores y a las nuevas aplicaciones reutilizar componentes existentes que modelan lógica de negocio sobradamente probada. En un entorno tremendamente cambiante como el actual, utilizar aplicaciones basadas en diseños de n-capas posibilita a las empresas ser más ágiles y adaptables en proporcionar valor a sus clientes. Los sistemas basados en n-capas tienen el potencial de reducir drásticamente tanto el tiempo de las nuevas aplicaciones de negocio, como el coste total de mantenimiento, adaptando estos complejos y caros sistemas a las siempre cambiantes necesidades empresariales. Los sistemas basados en n-capas posibilitan un desplazamiento estratégico en el uso de Internet como el "Ordenador Global". Esta revolución global no sólo está cambiando la dirección de

la informática empresarial, sino que también está cambiado la naturaleza de cómo hacemos negocios. Por ahora, las empresas más progresistas han reconocido lo inevitable y tienen estrategias claras para abrazar Internet, más allá de proporcionar un navegador y una conexión a la Red a sus empleados. Utilizando la potencia de la información de Internet, se puede crear, mejorar y mantener relaciones con todas las partes de las que depende un negocio para alcanzar el éxito.

Como se ha podido ver hasta este momento, n-capas no es una tecnología, sino una estrategia de uso de las tecnologías para crear un negocio a la vez que se obtiene todo el potencial de éste inherente a Internet. La informática basada en n-capas no se refiere solamente al despliegue de clientes ligeros de bajo coste conectados a servidores de aplicaciones muy flexibles con balanceo de carga e integrados con bases de datos distribuidas existentes a lo largo de diferentes plataformas y localizaciones. En realidad tiene que ver con la aplicación de las tecnologías relacionadas con desarrollos en n-capas para mejorar el conocimiento de los negocios y proveer un servicio de valor mediante la aplicación de esta avanzada tecnología como una solución para envolver oportunidades del mundo real. Mediante la adopción de arquitecturas de aplicaciones basadas en n-capas se permitirá la integración, escalabilidad, enlace o reingeniería de los sistemas existentes para adaptarse continuamente a los constantes cambios en las necesidades de negocio y convertirse en una tarea mucho más manejable en el futuro.

## 1.3 Tendencias y Tecnologías

### 1.3.1 Software Libre

Según la Free Software Foundation (Fundación de Software Libre), el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el Software; de modo más preciso, se refiere a cuatro

libertades de los usuarios del software, la libertad de usar el programa, con cualquier propósito; de estudiar el funcionamiento del programa, y adaptarlo a las necesidades; de distribuir copias, con lo cual se puede ayudar a otros y de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad se beneficie, para la segunda y última libertad mencionadas, el acceso al Código fuente es un requisito previo.

#### 1.3.1.1 Libertades del Software Libre

- La libertad de ejecutar el programa para cualquier propósito.
- La libertad de estudiar cómo trabaja el programa y adaptarlo a sus necesidades (El acceso al código fuente es una condición necesaria).
- La libertad de redistribuir copias para que pueda ayudar al vecino.
- La libertad de mejorar el programa y publicar sus mejoras y versiones modificadas en general para que se beneficie toda la comunidad (El acceso al código fuente es una condición necesaria).

Las ventajas especialmente económicas que brindan las soluciones libres y las aportaciones de la comunidad de desarrollo han permitido un constante crecimiento del software libre hasta superar en ocasiones al mercado propietario. Estas ventajas hacen que nuestro país siga una política de migración hacia el software libre y como parte de este proceso se decide para el desarrollo de la aplicación la utilización de herramientas y tecnologías pertenecientes al software libre (Subirós, Junio 2009).

### 1.3.2 Lenguajes de Programación

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, especialmente una computadora. Permite a los programadores especificar sobre qué datos la computadora debe operar, cómo estos deben ser almacenados, y qué acciones debe tomar ante cada

circunstancia previamente definida. Al ser un estándar de escritura permite a más de un programador trabajar de forma colaborativa en la construcción de un programa (Gutiérrez, 2007).

En el transcurrir de los años y en la medida en que la tecnología ha ido avanzado, han venido surgiendo diferentes lenguajes de programación, cada uno con características y objetivos específicos distintos pero todos con la misma finalidad, la comunicación hombre-máquina a través de una estructura sintáctica similar al lenguaje común utilizado en la vida diaria.

### 1.3.2.1 Lenguaje de Programación C++

C++ es un lenguaje imperativo orientado a objetos derivado del C. En realidad un superconjunto de C, que nació para añadirle cualidades y características de las que carecía. El resultado es que como su ancestro, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción. Respecto a su antecesor, se ha procurado mantener una exquisita compatibilidad hacia atrás por dos razones: poder reutilizar la enorme cantidad de código C existente, y facilitar una transición lo más fluida posible a los programadores de C clásico, de forma que pudieran pasar sus programas a C++ e ir modificándolos (haciéndolos más "++") de forma gradual. De hecho, los primeros compiladores C++ lo que hacían en realidad era traducir (pre procesar) a C y compilar después. A pesar de todo, ha experimentado un extraordinario éxito desde su creación. De hecho, muchos sistemas operativos, compiladores e intérpretes han sido escritos en C++ (el propio Windows y Java). Una de las razones de su éxito es ser un lenguaje de propósito general que se adapta a múltiples situaciones.

Se puede decir que abarca tres paradigmas de la programación:

1. La programación estructurada.
2. La programación genérica.

3. La programación orientada a objetos.

**Ventajas**

- Lenguaje muy didáctico, gracias a este lenguaje puedes aprender muchos otros lenguajes con gran facilidad, como C#, Java, Visual Basic, Javascript, PHP, entre otros.
- Lenguaje de programación orientado a objetos.
- Es muy potente en lo que se refiere a creación de sistemas complejos, un lenguaje muy robusto.
- Permite elaborar aplicaciones sencillas como un "Hola Mundo" hasta sistemas operativos y mucho más, todo eso dependiendo del manejo del lenguaje.
- Actualmente, puede compilar y ejecutar código de C, ya viene con librerías para realizar esta labor.
- Es un lenguaje muy empleado, existen muchos tutoriales en línea, libros, códigos fuentes abiertos, hay material de sobra y basta para aprender lo necesario y mucho más con este lenguaje.
- Existen muchos algoritmos cuyo pseudocódigo se encuentra ya desarrollado en C++, de manera que puedes tomarlo y amoldarlo a tu solución (porque el que veas un fragmento de código no asegura que sea correcto al 100%).

## ¿Por qué utilizar C++?

Se ha decidido la utilización de C++ como lenguaje de programación principalmente porque posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel:

1. Posibilidad de redefinir los operadores (sobrecarga de operadores).
2. Identificación de tipos en tiempo de ejecución.

3. Está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel.

También porque abarca los tres paradigmas de la programación. Además de C++ fueron analizados los lenguajes JAVA (Roche, 2010) y Delphi (Roche, 2010), pero estos lenguajes se nos hacen difíciles para utilizarlos porque tendríamos que buscar una serie de librerías para poder conectarlos con la herramienta que emplearemos para realizar el diseño del sistema, esta herramienta está más familiarizada con C++ y tiene bastante ayuda con códigos en C++, además muchas de las actualizaciones de estos lenguajes son propietarias.

### 1.3.3 Gestores de Bases de Datos.

Una Base de Datos (BD) es un conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora, o sea, que puede considerarse una colección de datos variables en el tiempo. El software que permite la utilización y la actualización de los datos almacenados en una o varias bases de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina Sistema de Gestión de Bases de Datos (SGBD), cuyo objetivo fundamental consiste en suministrar al usuario las herramientas que le permitan manipular los datos en términos abstractos, de forma que no necesite conocer el modo de almacenamiento de los mismos en la computadora, ni el método de acceso empleado (Guerra, 2009).

**Un SGBD debe permitir**

- **Definir una base de datos:** especificar tipos, estructuras y restricciones de datos.
- **Construir la base de datos:** guardar los datos en algún medio controlado por el mismo Sistema Gestor de Base de Datos (SGBD).
- **Manipular la base de datos:** realizar consultas, actualizarla, generar informes.

**Características**

- **Abstracción de la información:** Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.
- **Independencia:** La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- **Redundancia mínima:** Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.
- **Consistencia:** En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.
- **Seguridad:** La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra protegida frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipularla o destruirla; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
- **Integridad:** Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o

cualquier otra circunstancia capaz de corromper la información almacenada.

Entre los SGBD comúnmente utilizados en el mundo tenemos MySQL y PostgreSQL, Oracle. Estos presentan un enfoque relacional con un buen basamento matemático centrado en el Álgebra Relacional.

### 1.3.3.1 PostgreSQL

Es un sistema gestor de base de datos objeto-relacional, bajo licencia BSD. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre. Es el sistema de gestión de bases de datos de código abierto más avanzado del mundo y en sus últimas versiones posee muchas características que solo se podían ver en productos comerciales de alto calibre. Se ejecuta en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos y Windows. Su documentación está muy bien organizada, pública y libre, y posee comentarios de los propios usuarios.

Soporte nativo para los lenguajes más populares del medio: PHP, C, C++, Perl, Python, .Soporte de todas las características de una base de datos profesional (triggers, store procedures, funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas, etc.). Altamente adaptable a las necesidades del cliente (CAVSI, 2004).

**Principales Características**

- **Atomicidad (Indivisible):** es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- **Consistencia:** es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.

- **Aislamiento:** es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generarán ningún tipo de error.
- **Durabilidad:** es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

**Ventajas**

- **Inhalación ilimitada:** Con PostgreSQL, nadie puede demandarlo por violar acuerdos de licencia, puesto que no hay costo asociado a la licencia del software.
- **Soporte:** Además de nuestras ofertas de soporte, tenemos una importante comunidad de profesionales y entusiastas de PostgreSQL de los que su compañía puede obtener beneficios y contribuir.
- **Ahorros considerables en costos de operación:** PostgreSQL ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que otros productos, conservando todas las características, estabilidad y rendimiento.
- **Estabilidad y Confiabilidad Legendarias:** PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad. Ni una sola vez. Simplemente funciona.
- **Extensible:** El código fuente está disponible para todos sin costo. Si su equipo necesita extender o personalizar PostgreSQL de alguna manera, pueden hacerlo con un mínimo esfuerzo, sin costos adicionales. Esto es complementado por la comunidad de profesionales y entusiastas de PostgreSQL alrededor del mundo que también extienden PostgreSQL todos los días.
- **Multiplataforma:** PostgreSQL está disponible en casi cualquier Unix (34 plataformas en la última versión estable), y ahora en versión nativa para Windows.

### ¿Por qué utilizar PostgreSQL?

Teniendo en cuenta las características analizadas de los gestores de base de datos PostgreSQL, Oracle (Sandó, 2009) y MySQL (Sandó, 2009), se opta por usar PostgreSQL por las mejoras de rendimiento, características y funciones de agregación de múltiples columnas, así como índices invertidos generalizados, lo cual constituye una forma más escalable y programable de indexar datos semi-estructurados y texto. Es el sistema de gestión de bases de datos de código abierto más avanzado del mundo y se ejecuta en casi todos los principales sistemas operativos, también fue objeto de investigación MySQL y Oracle pero estos gestores de base de datos son propietarios, además PostgreSQL es superior a MySQL en que:

- Ofrece una garantía de integridad en los datos mucho más fuerte.
- Presenta una mejor escalabilidad y rendimiento bajo grandes cargas de trabajo.

## 1.4 Metodología de Desarrollo de Sistemas Informáticos

Desarrollar un buen software depende de un sin números de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo en un determinado proyecto, es trascendental, para el éxito del producto. El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte se encuentran aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán (Guerra, 2009).

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Pueden ser comparadas con un plan de contingencias en el que se va indicando

paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además quienes deben participar en el desarrollo de las actividades y qué papel deben de tener. Detallan además la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla (Solís Álvarez, 2005).

### 1.4.1 Proceso Unificado de Rational (RUP)

Es un proceso de desarrollo de software, que junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de un software de alta calidad que satisfaga los requerimientos de los usuarios finales (respetando cronograma y presupuesto). Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. Además se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y dirigido por casos de usos (Jacobson, 2000).

**Principios fundamentales**

- **Adaptar el proceso:** El proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.
- **Balancear prioridades:** Los requerimientos de los diversos inversores pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un balance que satisfaga los deseos de todos.
- **Demostrar valor iterativamente:** Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.

- **Elevar el nivel de abstracción:** Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o marcos de referencia (frameworks) por nombrar algunos. Esto evita que los ingenieros de software vayan directamente de los requisitos a la codificación de software a la medida del cliente, sin saber con certeza qué codificar para satisfacer de la mejor manera los requerimientos y sin comenzar desde un principio pensando en la reutilización del código. Un alto nivel de abstracción también permite discusiones sobre diversos niveles y soluciones arquitectónicas. Éstas se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con el lenguaje UML.
- **Enfocarse en la calidad:** El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

**Principales Elementos**

1. **Trabajadores:** Definen el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
2. **Actividades:** Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
3. **Artefactos:** Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
4. **Flujo de actividades:** Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

**Principales Flujos de Trabajo**

1. Modelamiento del negocio.
2. Requisitos.
3. Análisis y diseño.
4. Implementación.
5. Prueba.
6. Despliegue.
7. Administración de configuración y cambios.
8. Administración del proyecto.
9. Ambiente.

Los 6 primeros son conocidos como flujos de ingeniería, los cuales guían todo el proceso de desarrollo del software y los tres últimos, como de apoyo, que están presentes durante todo el ciclo de vida del sistema. Además divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades. Estas fases son: inicio, elaboración, construcción o implementación y transición o prueba (Electrónica, 2001 ).

1. **Inicio**

   Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

2. **Elaboración**

   Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.

3. **Construcción o Implementación**

   Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene una o varias versiones del producto que han pasado las pruebas. Se ponen estas versiones a consideración de un subconjunto de usuarios.

4. **Transición o Prueba**

   La versión ya está lista para su instalación en las condiciones reales. Puede implicar reparación de errores.

**Ventajas de la aplicación de esta metodología**

1. Reconoce que las necesidades del usuario y sus requerimientos no se pueden definir completamente al principio.
2. Permite evaluar tempranamente los riesgos en lugar de descubrir problemas en la integración final del sistema.
3. Acelera el ritmo del esfuerzo de desarrollo en su totalidad debido a que los desarrolladores trabajan para obtener resultados claros a corto plazo.
4. Distribuye la carga de trabajo a lo largo del tiempo del proyecto ya que todas las disciplinas colaboran en cada iteración.
5. Facilita la reutilización del código teniendo en cuenta que se realizan revisiones en las primeras iteraciones.
6. Tolerable cambio en los requerimientos.
7. Los elementos son integrados progresivamente.
8. Los riegos son mitigados en etapas tempranas.
9. Permite a la organización aprender a improvisar.
10. Resulta un producto más robusto ya que los errores se van corrigiendo en cada iteración.
11. El proceso puede ser improvisado y refinado en el desarrollo.

**Diferencias de RUP con las demás metodologías**

Algunos aspectos que diferencian a RUP de las demás metodologías y que lo hace único son que en RUP, los casos de uso no son sólo una herramienta para especificar los requisitos del sistema, sino que también guían su diseño, implementación y prueba. Los casos de uso constituyen un elemento integrador y una guía del trabajo. Además de utilizar los casos de uso para guiar el proceso; se presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento. También este propone que cada fase se desarrolle en iteraciones.

## ¿Por qué utilizar RUP?

Después de realizar un análisis e investigación de diferentes metodologías de desarrollo de Software, se elige utilizar RUP para el desarrollo del software, porque es la más completa y abarcadora, pues como señalan algunos autores, las otras metodologías son casos particulares de esta. Además, XP (Guerra, 2009) y MSF (Guerra, 2009) que fueron otras metodologías analizadas, presentan algunas debilidades, lo que representa riesgos considerables, como dificultades a la hora de una buena obtención de requisitos para el sistema. RUP es adaptable, según el tipo de proyecto y las características del mismo, haciéndose énfasis en aquellos flujos de trabajo durante la vida del software, que reporten más importancia y sean indispensables. RUP contiene artefactos que son diseñados durante las diferentes fases, los cuales describen detalladamente las características del software desde que se realiza el análisis del problema hasta la entrega final del producto.

## 1.4.2 Herramientas CASE

Las Herramientas CASE (Computer-Aided Systems Engineering) se pueden definir como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Existen herramientas CASE de trabajo visuales

como el Rational Rose y el Visual Paradigm, entre otras que permiten realizar el modelado del desarrollo de los proyectos.

### 1.4.2.1 Visual Paradigm

Es una herramienta que interpreta UML profesional, que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este software ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Una de las características más importantes del Visual Paradigm es que es multiplataforma (Guerra, 2009).

**Propósito**

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta de software libre de probada utilidad para el analista. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos (Freedownloadmanager.org, 2004).

**Características**

- Software libre.
- Disponibilidad en múltiples plataformas.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Licencia: gratuita y comercial. Varios idiomas. Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.

**Ventajas**

- Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes.
- Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
- Genera modelos VP-UML instantáneamente a partir de código binario .Net.
- Generación de documentación en formatos HTML y PDF.
- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris o Java.
- Brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo Visio y Rational Rose.
- Generación de código e ingeniería inversa: brinda la posibilidad de generar código a partir de los diagramas, para las plataformas como .Net, Java y PHP, así como obtener los diagramas a partir del código.
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

## ¿Por qué utilizar Visual Paradigm?

Se ha decidido utilizar Visual Paradigm 6.4 como herramienta CASE, debido fundamentalmente a que es multiplataforma. Por su estabilidad de ejecución en diferentes sistemas operativos y la facilidad de abrir y trabajar con un modelo UML utilizando el mismo programa sin importar el sistema operativo y sin afectar en absoluto el trabajo hecho; además destacar que esta herramienta guarda todo el modelo en un sólo fichero, así de simple, y basta con copiarse sólo ese fichero y uno está seguro de que tiene todo el trabajo encapsulado en él. Este software ayuda a una más rápida construcción de aplicaciones de calidad, mejores y

a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas.

También se analizó la herramienta Rational Rose (Roche, 2010), pero la principal razón por la que se escogió Visual Paradimg es que Rational es una herramienta propietaria.

### 1.4.3 Lenguaje Unificado de Modelado (UML)

Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir (Vico, 2002). Capta la información sobre la estructura estática y el comportamiento dinámico de un sistema, no es un lenguaje de programación. Las herramientas CASE pueden ofrecer generadores de código de UML para una gran variedad de lenguaje de programación, así como construir modelos por ingeniería inversa a partir de programas existentes. UML es un lenguaje de propósito general para el modelado orientado a objetos, es también un lenguaje de modelamiento visual que permite una abstracción del sistema y sus componentes (Corp, 2006).

#### 1.4.3.1 Objetivos

1. Es un lenguaje de modelado de propósito general que pueden usar todos los modeladores. No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática.
2. No pretende ser un método de desarrollo completo. No incluye un proceso de desarrollo paso a paso, incluye todos los conceptos que se consideran necesarios para utilizar un proceso moderno iterativo, basado en construir una sólida arquitectura para resolver requisitos dirigidos por casos de uso.
3. Ser tan simple como sea posible pero manteniendo la capacidad de modelar toda la gama de sistemas que se necesita construir. Necesita ser

lo suficientemente expresivo para manejar todos los conceptos que se originan en un sistema moderno, tales como la concurrencia y distribución, así como también los mecanismos de la ingeniería de software, como son la encapsulación y componentes.

4. Ser un lenguaje universal, como cualquier lenguaje de propósito general.
5. Imponer un estándar mundial.

### 1.4.3.2 Principales Beneficios.

1. Mejores tiempos totales de desarrollo (de 50 % o más).
2. Modelar sistemas (y no sólo de software) utilizando conceptos orientados a objetos.
3. Establecer conceptos y artefactos ejecutables.
4. Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
5. Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
6. Mejor soporte a la planeación y al control de proyectos.
7. Alta reutilización y minimización de costos.

## 1.5 Arquitectura.

La Arquitectura es el esqueleto o base de una aplicación. Representa la organización fundamental de un sistema. Desde los pequeños programas hasta los sistemas más grandes poseen una estructura y un comportamiento que los hace clasificables según su "arquitectura" (Subirós, Junio 2009).

La arquitectura por capas es un estilo de arquitectura en la que el objetivo primordial es la separación de la lógica de negocio de la lógica de diseño, un ejemplo básico es separar la capa de datos, de la capa de presentación al usuario. La ventaja principal de este estilo, es que el desarrollo se puede llevar a cabo en

varios niveles y en caso de algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado (Janium, 2009).

Además permite distribuir el trabajo de creación de una aplicación por niveles, de este modo, cada grupo de trabajo está totalmente abstraído del resto de los niveles, simplemente es necesario conocer las API (interfaz de programación de aplicaciones o API (del inglés Application Programming Interface), que es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción) que existen entre niveles.

Como tecnología, las arquitecturas de n-capas proporcionan una gran cantidad de beneficios para las empresas que necesitan soluciones flexibles y fiables para resolver complejos problemas inmersos en cambios constantes, permiten a los componentes de negocio correr en una LAN, WAN o Internet. Esto significa que cualquiera con un ordenador y conexión a la Red posee toda la funcionalidad que tendría si se encontrase delante de su sistema de escritorio, las arquitecturas n capas son el siguiente paso lógico en un proceso de evolución, el cual, está basado en las arquitecturas convencionales cliente-servidor (2 y 3 capas) más la convergencia de dos tecnologías tremendamente potentes (Monografias, 2007):

- **Desarrollo de aplicaciones basadas en componentes:** relacionado directamente con la Programación Orientada a Objetos (Lenguajes y Técnicas).
- **Internet:** primer ejemplo de un sistema complejo de n-capas cliente-servidor.

Los sistemas de n-capas utilizan técnicas de desarrollo basadas en componentes combinados con los estándares abiertos de Internet, para crear aplicaciones multiplataforma muy potentes con bajos costes, fáciles de mantener y con gran efectividad. Lo que realmente es nuevo en el modelo de n-capas es la posibilidad de distribuir objetos independientes sobre el número de capas que sean necesarias y enlazarlas dinámicamente, cuando sea necesario, para proporcionar

una flexibilidad ilimitada a la aplicación. La arquitectura n-capas no es una tecnología, sino una estrategia de uso de las tecnologías para crear un negocio a la vez que se obtiene todo el potencial de éste.

**Capas o Niveles**

1. **Capa de presentación:** es la capa que le permite al usuario interactuar con el sistema, captura y le comunica la información al mismo, dando un mínimo de proceso, realiza un filtrado previo para comprobar que no hay errores de formato. Esta capa se comunica únicamente con la del negocio.
2. **Capa de lógica o de negocio:** es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio e incluso lógica del negocio, pues es aquí donde se establecen las reglas que deben cumplirse. Esta capa se comunica con la de presentación para recibir las solicitudes y presentar los resultados, y con la capa de datos para solicitar al gestor de bases de datos para almacenar o recuperar datos de él.
3. **Capa de datos:** es donde se ubican los datos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de los mismos, reciben solicitudes de almacenamiento o de recuperación de información desde la lógica del negocio.

## 1.5.1 Modelo en 3 capas

La arquitectura de tres capas, define cómo organizar el modelo de diseño en capas, que pueden estar físicamente distribuidas, lo cual quiere decir que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores. Este patrón es importante porque simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores. Además, nos ayuda a identificar qué puede

reutilizarse, y proporciona una estructura que nos ayuda a tomar decisiones sobre qué partes comprar y qué partes construir (Monografias, 2007).

Para enfrentarse a estos temas, la comunidad de software desarrolló la noción de una arquitectura de tres niveles. La aplicación se divide en tres capas lógicas distintas, cada una de ellas con un grupo de interfaces perfectamente definido. La primera capa se denomina capa de presentación y normalmente consiste en una interfaz gráfica de usuario de algún tipo.

La capa intermedia, o capa de empresa, consiste en la aplicación o lógica de empresa, y la tercera capa, la capa de datos, contiene los datos necesarios para la aplicación. La capa intermedia (lógica de aplicación) es básicamente el código al que recurre la capa de presentación para recuperar los datos deseados. La capa de presentación recibe entonces los datos y los formatea para su presentación. Esta separación entre la lógica de aplicación de la interfaz de usuario añade una enorme flexibilidad al diseño de la aplicación. Pueden construirse y desplegarse múltiples interfaces de usuario sin cambiar en absoluto la lógica de aplicación siempre que está presente una interfaz claramente definida a la capa de presentación.

**Los principales beneficios**

- **Mantenibilidad:** Debido a que cada nivel es independiente de los otros niveles, actualizaciones o cambios pueden llevarse a cabo sin afectar a la aplicación como un todo.
- **Escalabilidad:** Debido a que niveles se basan en el despliegue de las capas, la ampliación de una aplicación es relativamente sencilla.
- **Flexibilidad:** Debido a que cada nivel se pueden manejar o escalar de forma independiente, se incrementa la flexibilidad.
- **Disponibilidad:** Las aplicaciones pueden aprovechar la arquitectura modular de permitir que los sistemas que utilizan los componentes son fácilmente escalables, lo que aumenta la disponibilidad.

**Ventajas**

1. Los componentes de la aplicación pueden ser desarrollados en cualquier lenguaje general lo que posibilita que el grupo de desarrolladores no se centre en el uso de un solo lenguaje.
2. Los componentes están centralizados lo que posibilita su fácil desarrollo, mantenimiento y uso.
3. Los componentes de la aplicación pueden estar esparcidos en múltiples servidores permitiendo una mayor escalabilidad.
4. Los problemas de limitación para las conexiones a las bases de datos se minimizan ya que la base de datos solo es vista desde la capa intermedia y no desde todos los clientes. Además que las conexiones y los drivers de las bases de datos no tienen que estar en los clientes.

## ¿Por qué utilizar Modelo en 3 capas?

La aplicación se sustentará en una Arquitectura Cliente – Servidor de tres capas principalmente por los beneficios de Mantenibilidad, Escalabilidad, Flexibilidad, Disponibilidad que aporta esta arquitectura, además de esta arquitectura de n capas se analizó en 2 capas pero estas arrojaron una serie de desventajas como:

**En 2 capas:**

- Difíciles de mantener, esto viene dado por el hecho de que son difíciles de mantener las reglas de negocio de la lógica de aplicación ya que estas están programadas en cada cliente y esto implica que cualquier cambio tiene que ser redistribuido en todos los clientes.
- Se compromete la confidencialidad, al tener programada la lógica de aplicación en el cliente, este tiene a su disposición todas las reglas de negocio de la empresa.

# 1.6 Entorno Visual

## 1.6.1 Qt Creator

Es un Entorno Integrado de Desarrollo (IDE) multiplataforma, diseñado para hacer que el desarrollo en C++ de la aplicación Qt sea más rápido y fácil. Qt Creator no quiere ser un reemplazo de Eclipse ni Visual Studio, sino un IDE ligero pensado especialmente para el desarrollo en múltiples plataformas (Qt, 2009).

Qt es un framework de C++ y para C++, proporciona un conjunto de bibliotecas de clases con funcionalidades como interfaz gráfica de usuario, comunicación en red, conexión a base de datos y muchas otras. Está escrito en C++ estándar y principalmente se usa C++ para trabajar con él. Sin embargo, también extiende al lenguaje mediante algunas características y palabras clave, como señales y slots, el bucle foreach, manejo automático de memoria para QObject's y otras pocas.

### 1.6.1.1 Principales características

- Posee un avanzado editor de código C++.
- Además soporta los lenguajes: C#/.NET Languages (Mono), Python PyQt y PySide, Ada, Pascal, Perl, PHP y Ruby.
- Posee también una GUI integrada y diseñador de formularios.
- Herramienta para proyectos y administración.
- Ayuda sensible al contexto integrado.
- Depurador visual.
- Resaltado y auto-completado de código.
- Soporte para refactorización de código.

### 1.6.1.2 Características Útiles.

- **Smart Code Editor:** El editor de código ofrece resaltado de sintaxis, así como de código.
- **Qt4 Project Generating Wizard:** Este asistente permite al usuario generar un proyecto para una aplicación de consola, una aplicación GUI, o una librería de C++.
- **Qt Help Integration:** Toda la documentación se puede acceder fácilmente haciendo clic en el botón de Ayuda (Help).
- **Qt Designer Integration:** En la interfaz de usuario se pueden diseñar formas en Qt Creator.
- **Locator:** Una potente herramienta de navegación que permite al usuario localizar los archivos y el uso mínimo de clases de teclado.
- **Support for qmake's .pro file format:** El archivo .Pro se usa como descripción del proyecto.
- **Debugging Interface to GDB:** Las solicitudes pueden ser depuradas en Qt Creator utilizando una interfaz gráfico para el depurador GNU simbólico.

### ¿Por qué utilizar QTCreator?

Para la realización del entorno visual del Software se decidió utilizar QTCreator 4.7 principalmente por su condición de multiplataforma, permite el drag and drop más personalizable y fácil de usar, además de crear ventanas rápidamente y no estar lidiando exhaustivamente con el código.

## 1.7 Conclusiones del Capítulo

Luego del estudio realizado se arriba a la conclusión de que el sistema estará guiado por la metodología de desarrollo RUP, por ser la más usada para el diseño, implementación y documentación de sistemas, además está basada en UML. Por las potencialidades que ofrece se escoge la herramienta CASE Visual Paradigm 6.4 para llevar la documentación del sistema. Como Gestor de Base de Datos se determina el uso de PostgreSQL 8.4 porque es el sistema de gestión de Bases de Datos de código abierto más avanzado del mundo y se ejecuta en casi todos los principales sistemas operativos. Para el desarrollo del sistema se utilizará el lenguaje de programación C++ y para el desarrollo QTCreator que se integra muy bien a este lenguaje de programación. La aplicación se sustentará en una arquitectura en 3 capas.

En este capítulo quedaron reflejados todos los conceptos relacionados con las tecnologías y herramientas informáticas, procesos de desarrollo de software, lenguajes de programación y tendencias que fundamentan la solución propuesta. Se hizo un análisis del estado del arte ampliando así los conocimientos sobre el problema a resolver.

# Capítulo 2: Modelo del Negocio

## 2.1 Introducción

En este capítulo se identifica el modelo del negocio, sus actores, trabajadores, la identificación de los casos de uso del negocio y la descripción de los mismos, así como las especificaciones de los requerimientos funcionales y no funcionales.

## 2.2 Modelo de Negocio.

El flujo actual de procesos en el negocio está dado a la necesidad de controlar y mantener actualizado los datos de los usuarios de la aplicación.

El negocio inicia cuando el administrador de la aplicación gestiona los usuarios, los grupos de usuarios y los trabajadores que tendrán acceso a las herramientas del sistema, así como los permisos que podrán tener en la aplicación.

### 2.2.1 Actores del Negocio.

Un Actor es cualquier persona, individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados.

**Tabla 1: Actores del Negocio.**

| Actor | Justificación |
|---|---|
| Administrador | Es el que inicia la gestión de los Usuarios (insertar, eliminar, modificar) y Grupos de Usuarios (insertar, eliminar) que hacen uso de la aplicación. |

### 2.2.2 Trabajadores del Negocio.

Los trabajadores del negocio son aquellos individuos, personas, máquinas o sistemas que trabajan para facilitar la realización de las actividades del negocio.

**Tabla 2: Trabajadores del Negocio.**

| Trabajador | Justificación |
|---|---|
| Administrador | Es quien gestiona los Usuarios (insertar, eliminar, modificar) y Grupos de Usuarios (insertar, eliminar, modificar) que hacen uso de la aplicación, otorgándole los respectivos permisos. |
| Usuario Avanzado | Hace uso de la aplicación y tiene los mismos privilegios que el Administrador. |
| Secretaria | Es quien hace uso de la aplicación y tiene como privilegio insertar o modificar datos de los Trabajadores de la aplicación. |

## 2.3 Casos de uso del Negocio.

Un proceso de negocio es un grupo de tareas relacionadas lógicamente que se llevan a cabo en una determinada secuencia y manera, que emplean los recursos de la organización para dar resultados en apopo a sus objetivos.

Un caso de uso del negocio representa a un proceso de negocio, por lo que se corresponde con una secuencia de acciones que producen un resultado observable para ciertos actores del negocio. Desde la perspectiva de un actor individual, define un flujo de trabajo completo que produce resultados deseables. Para identificar los procesos de negocio es muy importante tener en cuenta que deben generar un valor para el negocio o mitigar los costos del negocio.

## 2.3.1 Diagrama de caso de uso del Negocio.

Un diagrama de casos de uso del negocio representa gráficamente a los procesos del negocio y su interacción con los actores del negocio.

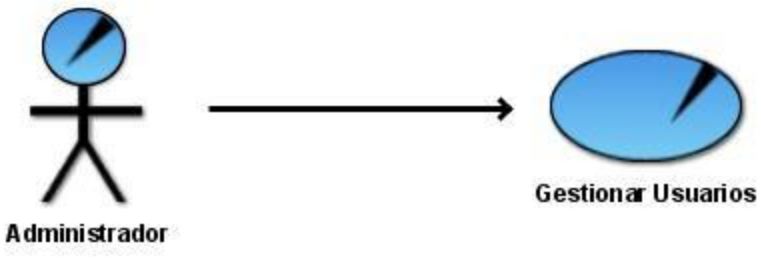



**Figura 1: Diagrama Caso de Uso del Negocio.**

## 2.3.2 Descripción de los casos de uso del Negocio.

**Tabla 3: Descripción del caso de uso Gestionar Usuarios.**

| **Caso de Uso:** | **Gestionar Usuarios.** |
|---|---|
| **Actores** | Administrador |
| **Trabajadores** | Administrador |
| **Descripción** | El caso de uso inicia cuando el Administrador de la aplicación precisa un (insertar, eliminar o modificar) de la aplicación. Estas operaciones son realizadas por él mismo, teniendo los datos del usuario opera sobre los registros de la aplicación actualizándolos. |
| **Precondiciones** | El caso de insertar precisa que no exista ningún usuario en el registro con los mismos datos.<br>En caso de eliminar o modificar precisa que el usuario esté en el registro. |
| **Flujo Normal de eventos** | |
| **Acción del Actor** | **Respuesta del Negocio** |
| 1. El administrador obtiene los | 1.1 Si la operación es insertar crea un nuevo |

| | |
|---|---|
| datos del usuario según la operación a realizar. | usuario en los registros<br>1.2 Si el eliminar, lo elimina de los registros de forma permanente<br>1.3 Si es modificar, renueva los datos en el registro quedando así actualizado. |

## 2.4 Diagrama de Clases.

El diagrama de clases, como artefacto que se construye para describir el modelo de objetos del negocio, muestra la participación de los trabajadores y entidades del negocio y la relación entre ellos. Aunque se puede construir un único diagrama, se recomienda confeccionarlo para cada caso de uso de negocio para una mejor claridad.

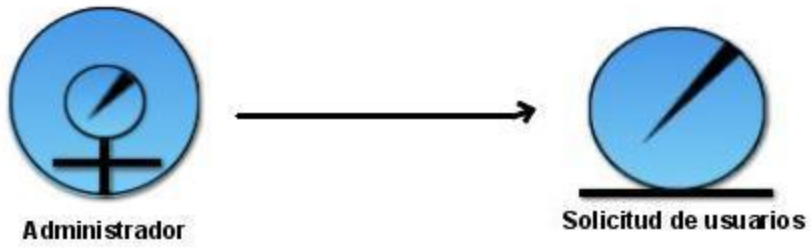



**Figura 2: Diagrama de clase de negocio.**

## 2.5 Captura de Requisitos.

Los requerimientos son una descripción de las necesidades o deseos de un cliente o un producto, su meta es identificar y documentar de una forma clara para el cliente y para el equipo de desarrollo lo que en realidad se necesita. Para la

captura de requisitos de este proyecto se contó con la colaboración de los usuarios finales, la dirección y el cliente.

### 2.5.1 Requisitos funcionales.

Una vez conocidos los conceptos que rodean al objeto de estudio, se puede comenzar a analizar ¿Qué debe hacer el sistema para que se cumplan los objetivos planteados al inicio de este trabajo?, para ello enumeramos a través de requerimientos funcionales las funciones que el sistema deberá ser capaz de realizar. Dentro de ellos se incluyen las acciones que podrán ser ejecutadas por el usuario, las acciones ocultas que debe realizar el sistema, y las condiciones extremas a determinar por el sistema. De acuerdo con los objetivos planteados el sistema debe ser capaz de:

**R1 Autenticarse.**

**R2 Conexión con Base de Datos.**

**R3 Modificar Contraseña.**

**R4 Gestionar Usuarios.**

1. Insertar Usuario.
2. Eliminar Usuario.
3. Modificar Usuario.
4. Buscar Usuario.

**R5 Gestionar Grupos de Usuarios.**

1. Insertar Grupo.
2. Eliminar Grupo.
3. Modificar Grupo.
4. Buscar Grupo.

**R6 Gestionar Trabajadores**

1. Insertar Trabajador
2. Eliminar Trabajador
3. Modificar Trabajador
4. Buscar Trabajador

**R7 Listar Usuarios.**

**R8 Listar Grupos de Usuarios.**

**R9 Listar Trabajador.**

## 2.5.2 Requisitos no funcionales.

Los requerimientos no funcionales como su nombre indica, son propiedades o cualidades que el producto debe tener. Estas propiedades se ven como las características que hacen al producto agradable, usable, rápido y confiable. A continuación se definen los siguientes:

**Apariencia o interfaz externa.**

- La interfaz a implementar debe ser sencilla, para que los usuarios que no son personas expertas en la rama de la informática, no necesiten mucho tiempo de adiestramiento.
- Por el uso diario y constante que tendrá el software, la interfaz debe ser agradable, que favorezca el estado de ánimo del cliente y que combine correctamente los colores, tipo de letra y tamaño.
- Deben utilizarse plantillas con el mismo estilo.

**Usabilidad.**

- El sistema debe ser de fácil manejo para los usuarios que tengan niveles básicos sobre la computación o trabajo con software.
- Debe tener una opción de ayuda sobre las principales operaciones que se realizan, para lograr un menor tiempo de adaptación.

**Rendimiento.**

- La aplicación debe estar concebida para el consumo mínimo de recursos.
- El sistema debe ser capaz de formular la respuesta lo más rápido posible.

**Soporte.**

- **Para el servidor de base de datos:**

  Se requiere que esté instalado un gestor de bases de datos que soporte grandes volúmenes de datos y velocidad de procesamiento.

- **Para el cliente:**

  Se requiere que esté instalado el Software.

**Portabilidad.**

- El sistema debe ser compatible con los SO más usados como UNIX (Linux), Windows (Versiones 2000 y XP).

**Hardware.**

- **Para las PC del cliente:**

  Se requiere tengan tarjeta de red.
  Se requiere tengan al menos 512 MB de memoria RAM.
  Se requiere procesador de 1.0 GHz como mínimo.

- **Para los servidores:**

  Se requiere tarjeta de red.
  Se requiere tenga al menos 1 GB de RAM.
  Se requiere al menos 80 GB de disco duro.
  Procesador 3.0 GHz como mínimo.

**Software**

- **Para las PC del cliente:**

  Se utilizará un servidor con el SO instalado Windows 2000 o superior, o con SO UNIX (Linux) preferiblemente.

- **Para los servidores:**

  Se necesita tecnología Postgre 8.4 o superior.

**Seguridad**

- El sistema debe comunicarse usando un protocolo seguro.
- Brindar servicio de autenticación.
- Garantizar que la información sea vista solo por quien tiene derecho a verla.
- Garantizar que las funcionalidades del sistema se muestren de acuerdo al nivel de acceso del usuario que esté activo.
- Mantener la integridad de la información, es decir que no se pierda durante su transporte o almacenamiento.
- El transporte y almacenamiento de la información se guardará en un fichero encriptado.

**Disponibilidad**

- El sistema deberá estar disponible las 24 horas del día para todos los usuarios.

**Confiabilidad**

- La información manejada por el sistema debe estar protegida de accesos no autorizados y divulgación.

## 2.6 Conclusiones de Capítulo

En este capítulo se comenzó a desarrollar la propuesta de solución, obteniéndose a partir del análisis de los procesos del negocio y la captura de Requerimientos.
En dicho proceso se encontraron:

- 9 Requerimientos Funcionales.
- 10 Requerimientos no Funcionales.

Se determinaron los Actores y Trabajadores del Negocio encontrándose:

- 1 Actor (Administrador).
- 2 Trabajadores (Secretaria y Usuario Avanzado).

Se presentaron los Diagramas de Casos de Uso del Negocio, Diagrama de Clases del Negocio y finalmente se describieron las acciones de los actores en los casos de uso con los que interactúan.

# Capítulo 3: Análisis y Diseño

## 3.1 Introducción

En este capítulo se hace referencia al Modelo de Análisis y al Modelo de Diseño. Se definen los actores, se realizan las descripciones de los casos de uso del sistema, se muestra el diagrama de análisis, el diagrama de clases del diseño, el diagrama de secuencia y es definida la arquitectura que se emplea.

## 3.2 Identificación de los actores del sistema.

Según Pressman (2005), los actores representan los usuarios del sistema e incluyen otros actores que puedan interactuar con él, es decir, representan terceros fuera del sistema que colaboran con este. Una vez identificados, cada actor desempeñará un papel bien definido, y en el contexto de ese papel debe tener interacciones útiles con el sistema, además se tiene delimitado el entorno externo del mismo.

**Tabla 4: Actores de la Aplicación.**

| Actor | Justificación. |
|---|---|
| Usuario | Persona que se autentifica para acceder al sistema con su respectivo permiso. |
| Administrador | Es el responsable de gestionar los datos de los usuarios que hacen uso de la aplicación, y cuenta además con todos los privilegios del sistema. |
| Usuario Avanzado | Persona encardada al igual que el administrador a gestionar el sistema, teniendo los mismos privilegios que el administrador. |
| Secretaria | Persona encargada de gestionar datos de los trabajadores, será la encargada de insertar o modificar los datos de los trabajadores. |

### 3.2.1 Diagrama de caso de uso del Sistema.

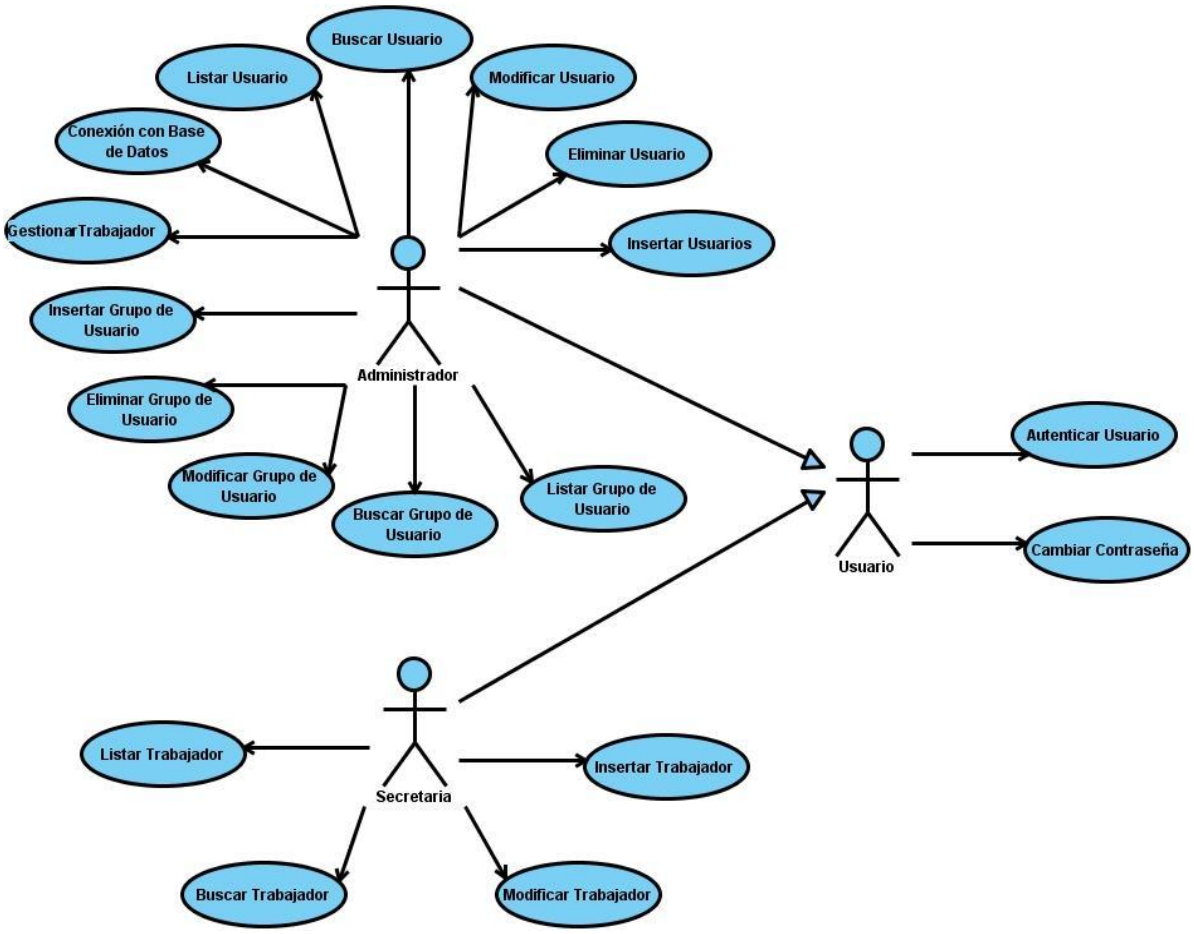



**Figura 3: Diagrama Caso de Uso del Sistema**

### 3.2.2 Descripciones de los casos de uso del Sistema.

La descripción de los Casos de Uso del Sistema le permitirá conocer a detalles cómo interactúa el usuario con el Sistema ante cada funcionalidad del mismo (Anexo 1).

**Tabla 5: Descripción del caso de uso Gestionar Usuario.**

| **Caso de Uso** | **Gestionar Usuario** |
|---|---|
| **Actores** | Administrador (inicia) |
| **Propósito** | Insertar un nuevo usuario para que interactúe con el sistema en dependencia del rol que se le asigne, modificar datos de estos usuarios o eliminar usuarios. |
| **Resumen** | En este caso de uso el administrador podrá mantener actualizada la información referente a los usuarios que interactúan con el sistema.<br>El caso de uso comienza cuando el administrador solicita "Gestionar Usuarios", ya sea para insertar un nuevo usuario, modificar datos o eliminar uno existente. Según la opción escogida, el sistema actualiza los cambios realizados en la base de datos, finalizando la realización de este caso de uso. |
| **Precondiciones** | En caso de que se desea modificar o eliminar un usuario determinado, este debe encontrarse registrado en el sistema. |
| **Curso Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1.El administrador accede a la interfaz de administración | 2. El sistema muestra una serie de acciones a realizar. |
| 3. El administrador elige la acción a realizar. | 4.- Si elige:<br>4.1 Insertar un nuevo usuario ir a la sección "Insertar Usuario"<br>4.2 Modificar un usuario ir a la sección "Modificar Usuario"<br>4.3 Eliminar un usuario ir a la sección "Eliminar Usuario". |
| **Sección "Insertar Usuario"** | |

| | |
|---|---|
| | 1. El sistema muestra la interfaz de insertar usuario, mostrando un formulario con los campos generales que se deben introducir. |
| 2. El administrador introduce los datos (nombre, contraseña, rol). | 3. El sistema verifica que no exista ese usuario en el sistema.<br>4. El sistema valida los datos introducidos.<br>5. Inserta el nuevo usuario en el sistema y muestra un mensaje ¨El Usuario ha sido Insertado Correctamente¨. |
| **Flujos Alternos** | |
| | 3.1 Si el usuario existe en la Base de Datos el sistema muestra un mensaje de error ¨El Usuario se encuentra registrado en el Sistema¨. |
| | 4.1 El sistema muestra un mensaje de error especificando que los datos son incorrectos. |
| **Sección “Modificar Usuario”** | |
| | 1. El sistema muestra un formulario con un listado de los usuarios existentes en la base de datos. |
| 2- El administrador selecciona el usuario que desea modificar. | 3. El sistema muestra la información del usuario seleccionado en un formulario. |
| 4- El administrador realiza los cambios pertinentes. | 5. El sistema verifica que los campos obligatorios estén llenos.<br>6. El sistema guarda los datos modificados en la base de datos y el |

| | |
|---|---|
| | sistema muestra un mensaje ¨Los Datos del Usuario han sido Modificados¨. |
| **Flujos Alternos** | |
| | 5.1 Si el Administrador deja algún campo obligatorio vacío, el sistema muestra el mensaje de error ¨Debe Completar los Campos Vacíos¨. |
| **Sección "Eliminar Usuario"** | |
| | 1. El sistema muestra en un formulario el listado de los usuarios existentes. |
| 2- El administrador elige el usuario a eliminar y presiona el botón ¨Eliminar¨. | 3- El sistema pide confirmación de que desea eliminar el usuario. |
| 4- El administrador confirma la acción. | 5- El sistema elimina el usuario seleccionado. |
| **Flujos Alternos** | |
| 4.1 El administrador cancela la acción. | 4.2 El sistema desmarca el usuario seleccionado. |
| **Poscondiciones** | Se registra un usuario, se modifican los datos del usuario o se elimina un usuario. |

**Tabla 6: Descripción del caso de uso Autenticar Usuario.**

| **Caso de Uso** | **Autenticar Usuario** |
|---|---|
| **Actores** | Usuario (inicia) |
| **Propósito** | Entrar a la aplicación para hacer uso de la misma. |
| **Resumen** | El caso de uso inicia cuando el usuario desea hacer uso de la aplicación y se identifica introduciendo su nombre y |

| | |
|---|---|
| | contraseña. El sistema comprobará si el usuario existe y si su contraseña es correcta. Si los datos son verídicos este habilitará las funciones de acuerdo al rol que desempeñe. |
| **Precondiciones** | El usuario debe estar registrado en el sistema. |
| **Curso Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El Usuario accede a la interfaz Inicio. | 2. El sistema muestra un formulario para autenticarse. |
| 3. El usuario llena los campos. | |
| **Flujos Alternos** | |
| | 3.1 Si el usuario o contraseña son incorrectos el sistema muestra un mensaje ¨El Nombre de Usuario o Contraseña son Incorrectos ¨. |
| **Poscondiciones** | Se abre la sesión del usuario registrado. El sistema sólo muestra las opciones disponibles para el tipo de usuario registrado. |

**Tabla 7: Descripción del caso de uso Modificar Contraseña.**

| **Caso de Uso** | **Modificar Contraseña** |
|---|---|
| **Actores** | Usuario (inicia) |
| **Propósito** | Permitir que el usuario modifique su contraseña. |
| **Resumen** | El caso de uso inicia cuando un usuario desea cambiar su contraseña actual. Para efectuar la operación el usuario debe introducir su contraseña actual y la nueva contraseña, el sistema solicitará una confirmación de esta última, de ser correcto este proceso el sistema procede a actualizar la contraseña, concluyendo así la ejecución del caso de uso. |
| **Precondiciones** | El usuario debe estar utilizando su cuenta en ese |

| | |
|---|---|
| | momento. |
| **Curso Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El Usuario accede a la interfaz Cambiar Contraseña. | 2. El sistema muestra un formulario con el usuario, la contraseña anterior y nueva contraseña, así como confirmar la nueva contraseña. |
| 3. El usuario llena los campos. | |
| **Flujos Alternos** | |
| | 3.1 Si los datos en los campos son incorrectos el sistema muestra un mensaje ¨Los Datos son Incorrectos, escriba nuevamente la Contraseña ¨. |
| **Poscondiciones** | Queda actualizada en la base de datos la nueva contraseña. |

## 3.3 Modelo de Análisis

Durante el análisis, se analizan los requisitos que fueron descritos en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar todo el sistema, incluyendo su arquitectura.

Aunque en el modelo del análisis hay un refinamiento de los requisitos, no se tiene en cuenta el lenguaje de programación a usar en la construcción, la plataforma en la que se ejecutará la aplicación, los componentes prefabricados o reusables de otras aplicaciones, entre otras características que afectan al sistema.

**Tabla 8: Trabajadores del Análisis.**

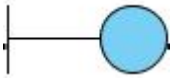
| Trabajadores | Justificación. |
|---|---|
| Arquitecto del Software. | Responde por la integridad del modelo de análisis y por la arquitectura del modelo de análisis. |
| Ingeniero de Casos de Uso. | Responde por la integridad de una o más realizaciones de casos de uso. |
| Ingeniero de Componentes. | Define y mantiene las clases y las relaciones entre ellas y la integridad de uno o varios paquetes del análisis. |

**Tabla 9: Artefactos del Análisis.**

| Trabajadores | Justificación. |
|---|---|
| Modelo de Análisis. | Contiene clases del análisis y sus objetos organizados en paquetes que colaboran. |
| Clases de Análisis. | Se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Tienen atributos y entre ellas se establecen relaciones de asociación, agregación / composición, generalización / especialización y tipos asociativos. |

En una aplicación cliente/servidor de tres capas, en la capa de usuario aparecen fundamentalmente clases interfaz ya que allí se ejecutan las aplicaciones del cliente. En la capa intermedia están las clases controladoras ya que en ellas se agrupan los servicios que son compartidos por múltiples aplicaciones. En la capa servidor estarían las clases entidad porque estas contienen los datos que persistirán en la base de datos. RUP propone clasificar a las clases en:

**Tabla 10: Clasificación de las Clases.**

| Nombre | Características | Representación |
|---|---|---|
| Entidad | Modelan información que posee larga vida y que es a menudo persistente. |  |
| Interfaz | Modelan la interacción entre el sistema y sus actores. |  |
| Control | Coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso. |  |

En la construcción del modelo de análisis se tienen que identificar las clases que describen la realización de los casos de uso, los atributos y las relaciones entre ellas. Con esta información se construye el Diagrama de clases del análisis, que por lo general se descompone para agrupar las clases en paquetes. Esta descomposición tiene impacto por lo general en el diseño e implementación de la solución.

## 3.3.1 Diagrama de clases del análisis.

Un Diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa el funcionamiento del mundo real, no de la implementación automatizada del mismo (Anexo 2).

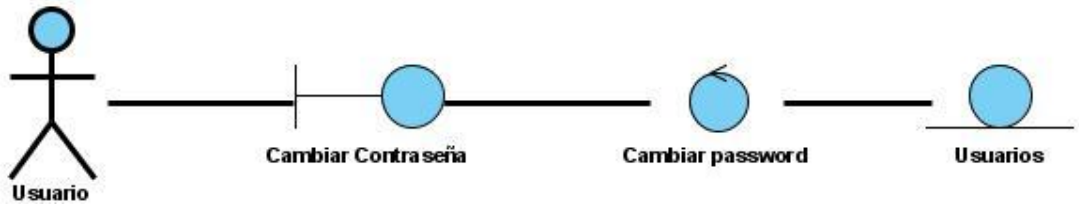



**Figura 4: Diagrama Clase del Análisis Cambiar Contraseña.**

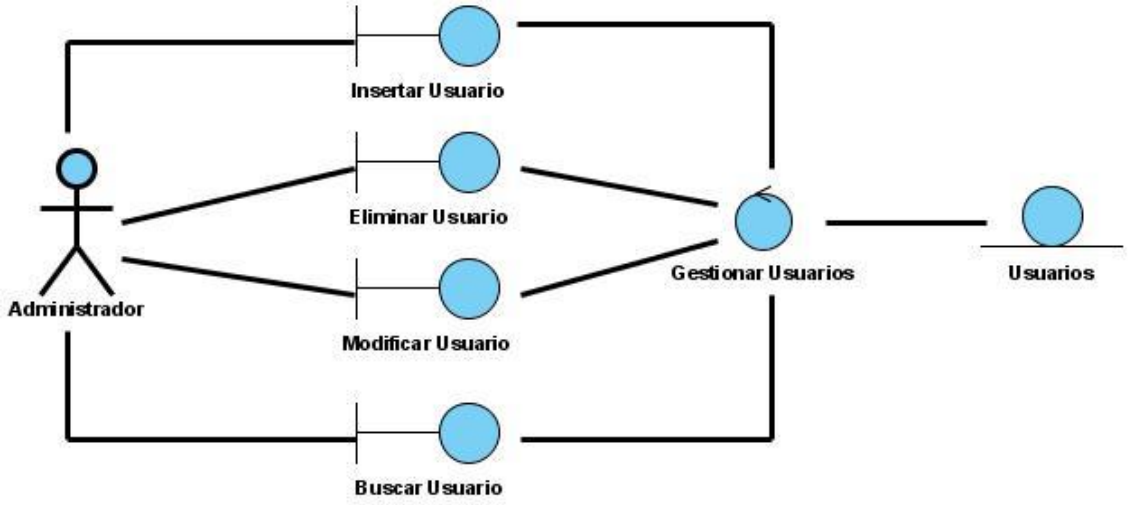



**Figura 5: Diagrama de clases del Análisis Gestionar Usuarios.**

## 3.3.2 Diagramas de Colaboración.

Una colaboración modela los objetos y los enlaces significativos dentro de una interacción. Los objetos y los enlaces son significativos solamente en el contexto proporcionado por la interacción. Un rol describe un objeto, y un rol en la asociación describe un enlace dentro de una colaboración. Un diagrama de colaboración muestra los roles en la interacción en una disposición geométrica. Los mensajes se muestran como flechas, ligadas a las líneas de la relación, que conectan a los roles (Anexo 3).

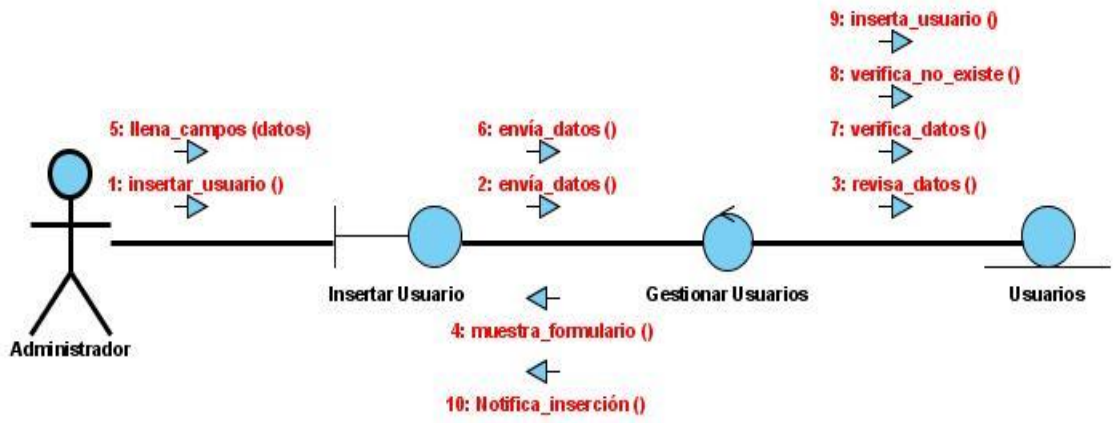



**Figura 6: Diagrama de Colaboración Insertar Usuario.**

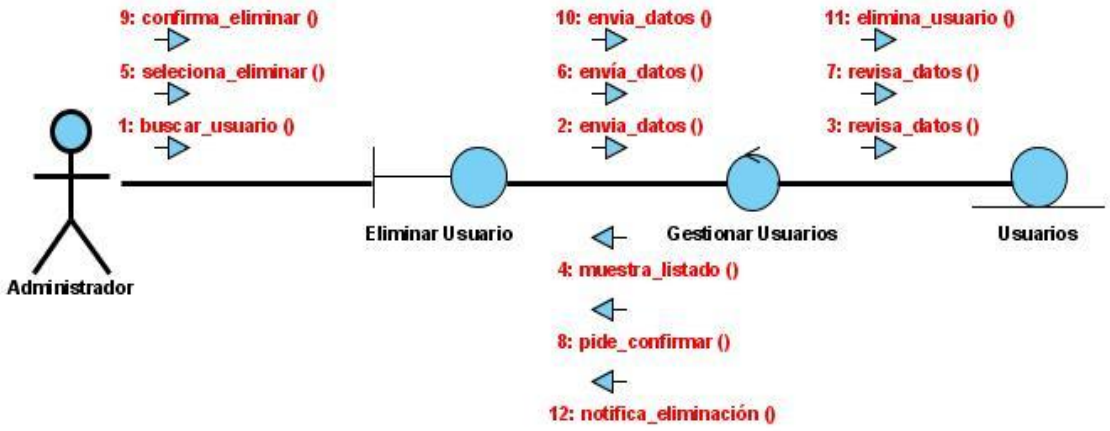



**Figura 7: Diagrama de Colaboración Eliminar Usuario.**

## 3.3.3 Diagramas de Secuencia.

Un diagrama de secuencia representa una interacción como un gráfico bidimensional. La dimensión vertical es el eje de tiempo, que avanza hacia abajo de la página. La dimensión horizontal muestra los roles de clasificador que representan objetos individuales en la colaboración. Cada rol de clasificador se representa mediante una columna vertical-línea de vida (Anexo 5).

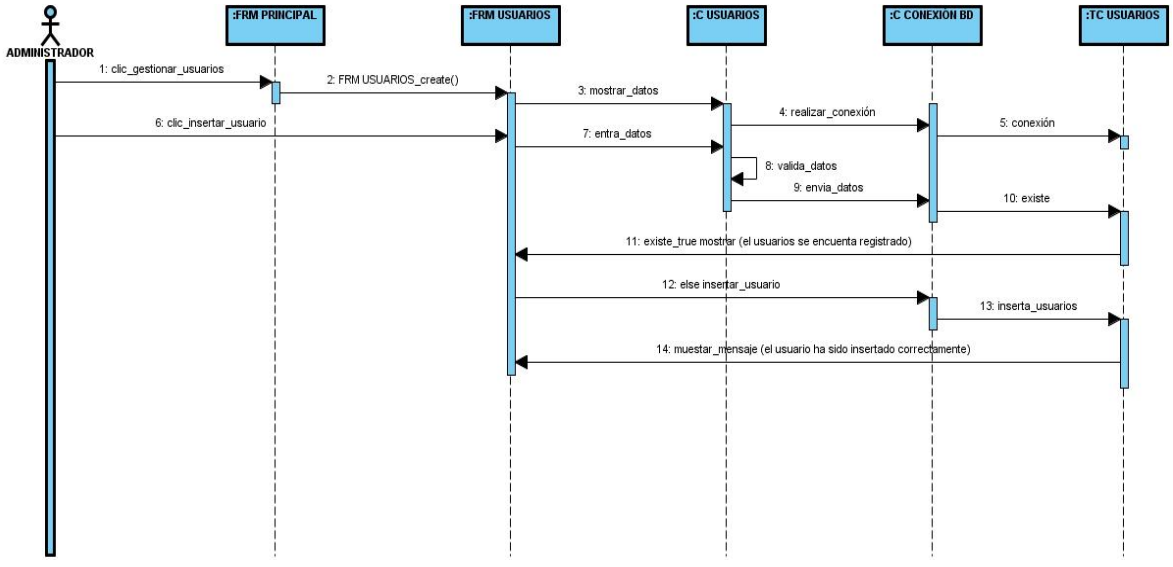



**Figura 8: Diagrama de Secuencia Insertar Usuarios.**

## 3.4 Modelo de Diseño

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Esto contribuye a una arquitectura estable y sólida, y crear un plano del modelo de implementación. Durante la fase de construcción, cuando la arquitectura es estable y los requisitos están bien entendidos, el centro de atención se desplaza a la implementación.

En el diseño modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos. Además, impone una estructura del sistema que debemos esforzarnos por conservar lo más fielmente posible cuando demos forma al sistema.

**Tabla 11: Trabajadores del Diseño.**

| Trabajadores | Justificación. |
|---|---|
| Arquitecto del Software. | Responsable de la arquitectura del software incluyendo las decisiones técnicas que restringen el diseño e implementación general del proyecto. Debe identificar y documentar los aspectos arquitectónicamente significativos de las vistas de requerimientos, diseño, implementación y despliegue. |
| Diseñador del Software. | El diseñador es el responsable de diseñar una parte del sistema cumpliendo con las restricciones de los requerimientos, arquitectura y proceso de desarrollo del proyecto. Debe asegurarse que el diseño es consistente con la arquitectura del software y que está detallado al |

| | |
|---|---|
| | punto que se puede proceder con la implementación. |
| Diseñador de la Base de Datos. | Es el responsable de diseñar el almacenamiento de la información persistente que se usará en el proyecto. Debe definir el diseño detallado de la Base de Datos (BD) incluyendo tablas, índices, vistas, restricciones, procedimientos almacenados y cualquier otro elemento que se necesite para almacenar, recuperar y eliminar objetos persistentes. |

**Tabla 12: Artefactos del Diseño.**

| Artefactos | Propósito. |
|---|---|
| Modelo de Diseño. | Es una abstracción de la implementación del sistema. Es usado para concebir un documento del diseño del sistema de Software (SW). |
| Modelo de Despliegue. | Capturar la configuración de los elementos de procesamiento, y las conexiones entre estos elementos en el sistema. |
| Documento de Arquitectura de Software | Provee una vista abarcadora de la arquitectura del sistema. Medio de comunicación entre el arquitecto de SW y otros miembros del equipo de proyecto que tienen acceso a las decisiones significativas para la arquitectura. |
| Clases del Diseño | Provee una vista detallada de cómo estará implementada la aplicación. |
| Modelo de Datos | Usado para describir la representación lógica y física de la información persistente manejada por el sistema. |

## 3.4.1 Diagramas de Clases del Diseño.

Este describe las clases que componen el sistema con mayor detalle. Incluye los atributos particulares de cada clase y las relaciones existentes entre ellas. Se crean para complementar los requisitos funcionales y no funcionales determinados.

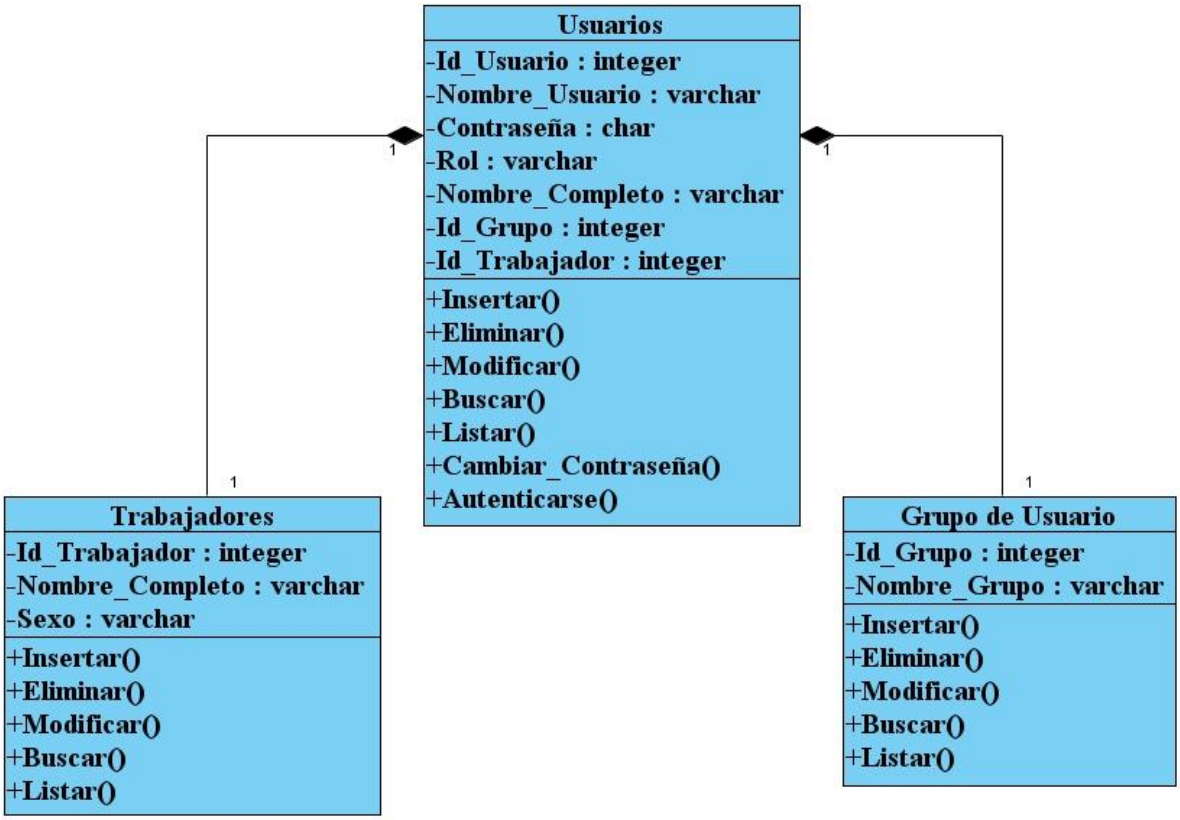



**Figura 9: Diagrama de clases del Diseño.**

## 3.4.2 Descripción de las clases del diseño.

Se describe cada clase representada en el diagrama anterior, Se define el tipo de clase que es cada una, las funciones que realizan, los atributos y métodos que la conforman (Anexo 6).

**Tabla 13: Descripción de la Clase Usuario.**

| **Nombre de la clase** | **Clase_Usuario** |
|---|---|
| **Tipo de clase:** | Entidad |
| **Resumen:** | Esta clase permite realizar la gestión de los datos de los usuarios del sistema. |
| **Atributos** | **Tipo** |
| Id_Usuario | integer |
| Nombre_Usuario | varchar |
| Contraseña | char |
| Rol | varchar |
| Nombre_completo | varchar |
| **Responsabilidades** | |
| **Nombre.** | **Descripción.** |
| Insertar_Usuario() | Este método se encarga de insertar los usuarios al sistema, no se debe repetir el nombre de usuario. |
| Eliminar_Usuario() | Este método se encarga de eliminar un usuario del sistema, debe estar registrado el usuario. |
| Modificar_Usuario() | Permite modificar datos del usuario. |
| Modificar_Contraseña() | Permite modificar la contraseña de un usuario. |
| Buscar_Usuario() | Permite buscar un usuario para realizarle cualquier acción. |
| Listar_Usuario() | Permite mostrar una lista de usuarios que hacen uso de la aplicación. |

## 3.5 Principios del Diseño.

A continuación se muestran algunas de las interfaces gráficas que han sido diseñadas para la aplicación final.

### 3.5.1 Interfaz de Usuarios.

Respondiendo a las solicitudes del cliente se escogen colores ligeros para el diseño del software, se emplean palabras familiarizadas con el entorno del cliente y se crea una interfaz sencilla pero con una apariencia que se agradable y que despierte curiosidad en el usuario, pero previendo siempre que el usuario no se pierda en la navegación del software.


**Figura 10: Interfaz para caso de uso Autenticar Usuario.**


Figura 11: Interfaz de Inicio de Sesión de Administrador.


Figura 11: Interfaz para el caso de uso Gestionar Grupo de Usuarios.


**Figura 12: interfaz para el caso de uso Gestionar Usuarios.**

## 3.5.2 Ayuda.

Se contará con un manual de ayuda para el usuario, en el que se explica detalladamente cómo se trabaja con la aplicación.

## 3.5.3 Tratamiento de Errores.

Para que un sistema sea confiable, este debe ser capaz de detectar la mayor cantidad de errores que sea posible y explicarle al usuario dónde es que se ha cometido el error. Los errores más frecuentes cometidos por los usuarios son:

- Insertar un elemento que ha insertado anteriormente.
- Eliminar o modificar uno que no está registrado en la base de datos.
- Dejar campos vacíos al realizar una de las acciones anteriores.

Cuando el usuario avanzado o el administrador introduce nuevos registros en la base de datos, se corre el riesgo de que los valores entrados provoquen

duplicidad de datos, para ellos se han definido las llaves primarias de cada tabla en la BD y el sistema impide la entrada de valores duplicados. Si el usuario incurre en este error el sistema muestra un mensaje.

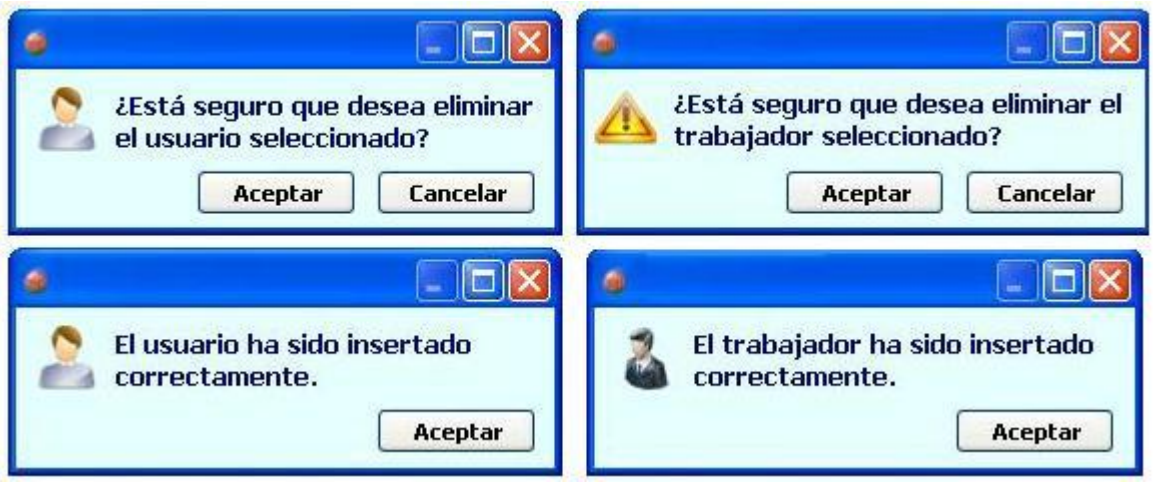



**Figura 13: Mensajes de Errores.**

## 3.6 Definición de la arquitectura.

A continuación se aborda el análisis detallado de la arquitectura, análisis que se divide en:

- Representación arquitectónica, la cual describe la arquitectura entre otras cosas.
- Objetivos y restricciones de la arquitectura, aquí se describen los principales objetivos y requisitos presentes en el sistema y que tienen impacto en la arquitectura.
- Vistas arquitectónicas, se hace una descripción técnica y profunda de la arquitectura.
- Dimensión y rendimiento, aborda una descripción de las características más importantes del sistema que afectan a la arquitectura.
- Calidad, explica como la arquitectura contribuirá al desarrollo de la aplicación o de cada una de las funcionalidades de la aplicación.

### 3.6.1 Representación Arquitectónica

Nuestro Sistema está basado en una arquitectura en 3 Capas y 2 niveles (ver Vista de Despliegue) para una aplicación de Desktop:

- **Capa de interfaz de Usuario o de Presentación:** es donde estarán nuestras interfaces de usuarios finales y sus controles visuales.
- **Capa de Lógica o del Negocio:** aquí irá todo el código que define las reglas de negocio (cálculos, validaciones). Surge de los procesos que hemos encontrado en el análisis.
- **Capa de datos (Origen de datos):** permite acceder a las fuentes de datos. Esencialmente trata sobre 4 operaciones básicas, llamadas CRUD (crear, eliminar, modificar y buscar). En nuestro caso está dado por el Gestor de Base de Datos PostgreSQL.

### 3.6.2 Objetivos y Restricciones

El sistema necesita controlar los usuarios que harán uso del sistema de una manera segura. Por tanto el sistema establece niveles de privilegios para garantizar seguridad e integridad en los datos.

Se utiliza la filosofía del software libre para la implementación del sistema por tanto se usa la tecnología C++ como herramienta de programación por sus facilidades de orientada a objetos y para el desarrollo de las interfaces se utilizará QTCreator que se integra perfectamente con C++. Como gestor de base de datos se utiliza PostgreSQL 8.4, ya que el mismo trae consigo grandes mejoras en cuanto a seguridad e incluye muchas funcionalidades. El conjunto de estas 2 herramientas incluyen componentes y funcionalidades con una eficacia notable si de seguridad hablamos.

Esto último introduce la ventaja de que nuestro sistema pueda ser desplegado en PC con sistemas operativos Windows o Linux (tecnología UNIX), por lo que es portable esto último se puede notar en la vista de despliegue.

En cuanto a la realización de los otros módulos que conforman las herramientas deben cumplir con las siguientes restricciones:

1. Los colores que se usaran serán:
    - background-color: rgb (0, 170, 255) (Azul Claro-Oscuro).
    - background-color: rgb (255, 85, 0) (Rojo Oscuro).
    - background-color: rgb (230, 255, 255) (Azul Claro).
    - color: rgb (0, 0, 0) (Negro)
    - color: rgb (0, 0, 255) (Azul Fuerte)
2. Los formatos de escritura serán:
    - MS Shell Dlg 2
    - Bold (hace la misma función de negrita)
    - Arial
    - El tamaño si es de decisión propia.
3. Los iconos e imágenes que puedan ser usadas ya han sido seleccionadas y solo se podrá utilizar alguna otra, como un logo para representar la herramienta, por lo que será de decisión propia del responsable de la herramienta a implementar, el uso de esa imagen como logo de su herramienta.
4. Para establecer la conexión con la base de datos se apoyaran en ***libpq***, la cual es un conjunto de librerías escritas en C que permite a un programa cliente enviarle consultas al servidor de PostgreSQL y recibir el resultado de estas. Debemos tener en cuenta incluir el archivo ***libpq-fe.h***, el cual es el encargado de incluir las librerías de ***libpq***.

## 3.6.3 Vistas

### 3.6.3.1 Vistas de caso de uso

Los casos de uso arquitectónicamente significativos son:

- Gestionar Usuarios.
    - Insertar Usuario.

- Eliminar Usuario.
- Modificar Usuario.
- Buscar Usuario.

➢ Gestionar Grupos de Usuarios.

- Insertar Grupo.
- Eliminar Grupo.
- Modificar Grupo.
- Buscar Grupo.

➢ Gestionar Trabajadores.

- Insertar Trabajador.
- Eliminar Trabajador.
- Modificar Trabajador.
- Buscar Trabajador.

➢ Listar Usuarios.
➢ Listar Grupo de Usuarios.
➢ Listar Trabajadores.

Para conocer con mayor detalle cuales son todos estos escenarios, referirse a Diagrama: caso de uso del sistema.

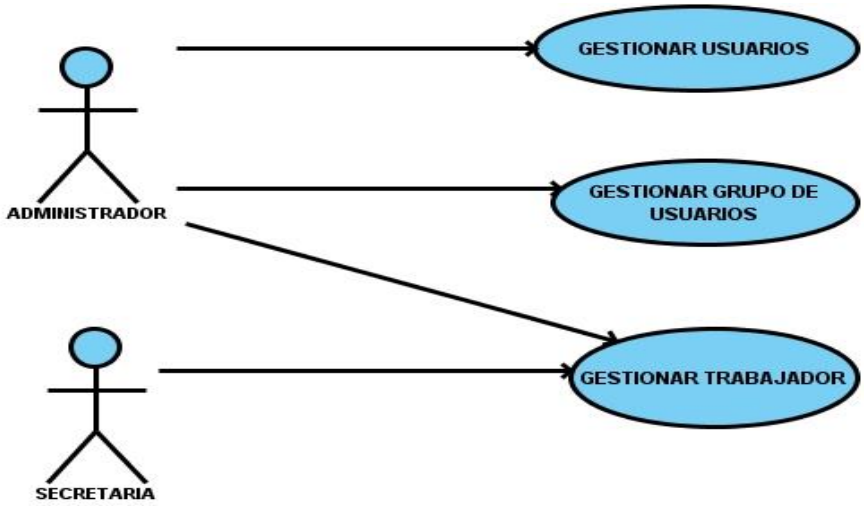



**Figura 14: Vista de caso de uso.**

### 3.6.3.2 Vista Lógica.

Se utiliza un estilo arquitectónico en capas siendo estas:

- Capa de Interfaz de Usuario.
- Capa Lógica del Negocio.
- Capa de Acceso a Datos.

Como se explicó anteriormente el sistema está basado en una arquitectura por capas las cuales conforman los paquetes donde irán ubicadas las clases según la capa a la que pertenecen, a continuación los paquetes con los subsistemas arquitectónicamente significativos:

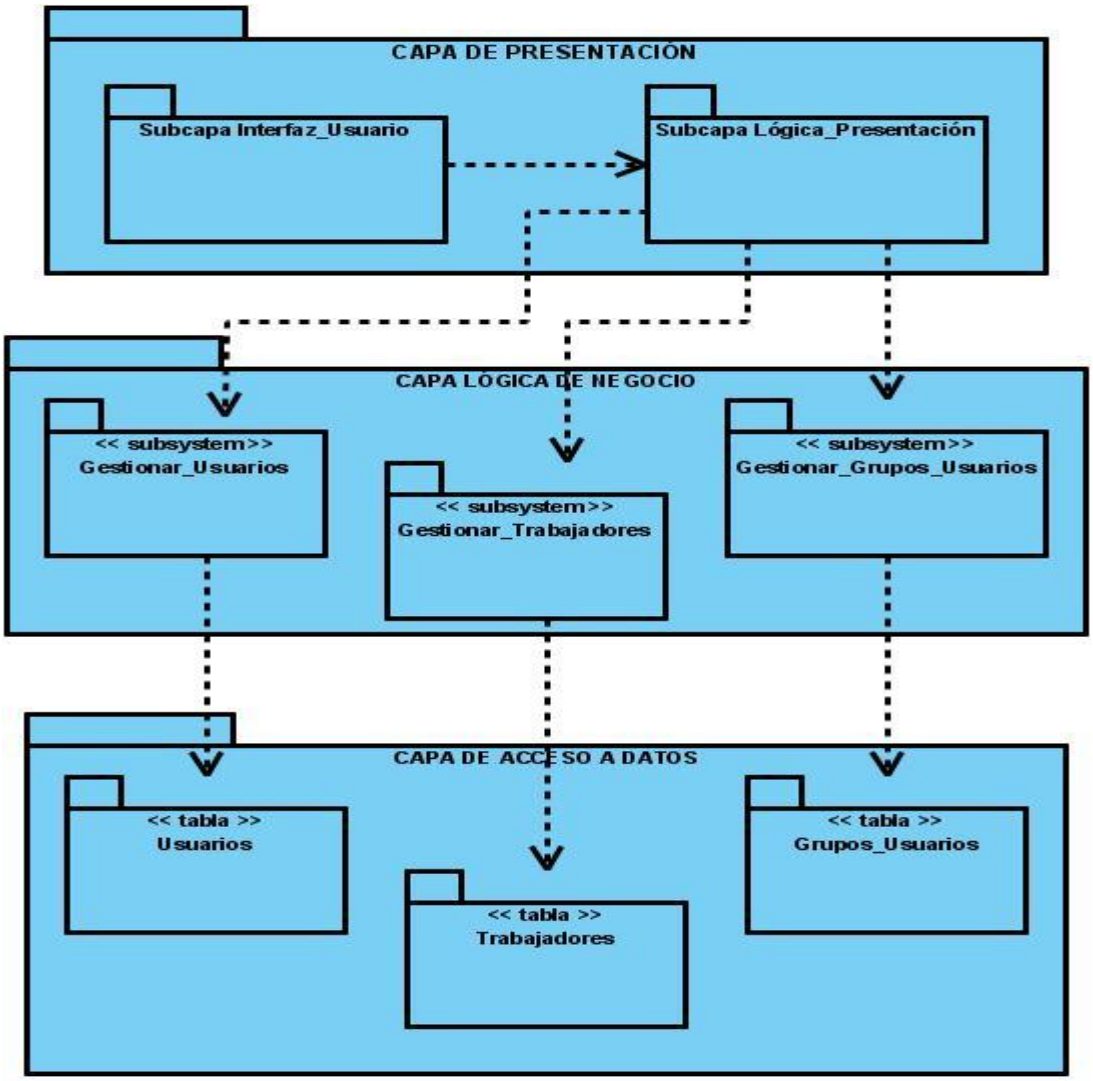



**Figura 15: Diagrama de Vista Lógica.**

**Interfaz de Usuario.**

- **Subcapa interfaz de usuario:** Agrupa las clases que interactúan directamente con el usuario final (Interfaces de Usuarios clientes), estas se comunican con las clases para interfaces de usuario servidoras.

- **Subcapa lógica de presentación:** Contiene las clases de interfaz (servidoras, se ejecutan del lado del servidor) que crean o construyen las clases de interfaz de usuario clientes, además de recoger los datos que estas envían y darle una forma (formato, agrupación) para enviarlos a la lógica de negocio.

**Lógica de Negocio.**

- **Subsistema Gestionar Usuario:** Encapsula la lógica de la gestión de los Usuarios y las acciones que se pueden realizar a los mismos (insertar, eliminar, ect). Toma las peticiones de la lógica de presentación y las satisface.

- **Subsistema Gestionar Grupo de Usuarios:** Encapsula la lógica de la gestión de los Grupo de Usuarios y las acciones que se pueden realizar a los mismos (insertar, eliminar, ect). Toma las peticiones de la lógica de presentación y las satisface.

- **Subsistema Gestionar Trabajadores:** Encapsula la lógica de la gestión de los Trabajadores y las acciones que se pueden realizar a los mismos (insertar, modificar, ect). Toma las peticiones de la lógica de presentación y las satisface.

**Acceso a Datos.**

- **Tabla Usuarios:** contiene todos los datos de los usuarios que hacen uso de la aplicación, que además contiene los id de trabajador y grupo de usuarios.

- **Tabla Trabajadores:** contiene todos los datos de los trabajadores registrados en la aplicación.

- **Tabla Grupo de Usuarios:** contiene los datos de todos los grupos de usuarios que se encuentran registrados en la aplicación.

### 3.6.3.3 Vista de Despliegue.

Para que el sistema realice las funciones se necesita:

- Una máquina (PC Cliente) donde estará instalado nuestro sistema para brindar los servicios.
- Una máquina (PC Servidor) que haga función de servidora de bases de datos, es decir, donde radicarían los datos persistentes, máquina con la cual la PC Cliente se conecte mediante el protocolo TCP/IP y puedan gestionar entre ambas la actualización de los datos.


**Figura 16: Diagrama de la Vista de Despliegue.**

## 3.6.4 Dimensiones y Rendimiento

Las características del sistema que influyen en la arquitectura se mencionan a continuación:

- Es un sistema concebido bajo la filosofía de software libre.
- El sistema será desarrollado como una aplicación de desktop, por tanto se utilizará una arquitectura en 3 capas para aplicaciones Desktop.
- El sistema necesita mantener los datos de forma centralizada de forma tal de que si el sistema es instalado en diferente máquinas (PC), se puedan seguir accediendo a los datos sin ningún problema.
- El núcleo del sistema consiste en la gestión de los **Usuarios** (creación, modificación y eliminación de los distintos **Usuarios**) para autorizar el uso del sistema, así como la gestión de los **Grupos de Usuarios** y los **Trabajadores** (modificación, creación ect).

### 3.6.5 Calidad

Para el desarrollo del sistema en cuestión se hace necesaria la definición y establecimiento de una arquitectura o una infraestructura (framework) para regir el desarrollo del sistema basándose principalmente en la apariencia de los módulos que conformarán el proyecto.

La arquitectura propuesta ayudará a que se le puedan dar cumplimiento a todos los requisitos funcionales y los no funcionales con una alta eficiencia, garantizando de esta forma que el sistema sea portable, tenga un grado (relativamente pequeño) de reusabilidad, sea estable, y que pueda ser mantenido con cierta facilidad.

## 3.7 Conclusiones del Capítulo.

Concluido este capítulo hemos finalizado la etapa de análisis y diseño del sistema, obteniendo como resultado los diagramas que reflejan la estructura del sistema y la forma en la que están distribuidas las partes fundamentales, que sin ellas resultaría difícil obtener resultados satisfactorios en el desarrollo de la aplicación.

Se confeccionó la arquitectura a para el desarrollo de los módulos que conforman la Herramientas, determinándose una serie de restricciones que deberán cumplir los módulos y que se encontrará en cada capa de la arquitectura logrando con esto un lenguaje común entre los desarrolladores del sistema.

Se determinó los Diagramas de Clase del Análisis y Diagramas de Diseño, así como los Diagramas de Secuencia y Colaboración. Se puede asegurar que se cumplió con los objetivos trazados para este Capítulo.

# Capítulo: 4 Implementación

## 4.1 Introducción

En este Capítulo se hace referencia y se muestran el diseño de la Base de Datos y el Diagrama de Clases Persistentes, utilizando este último para generar el modelo físico de datos con el uso de la herramienta Visual Paradimg, se describe cada tabla de la base de datos con sus respectivos atributos. Por otro lado, se ofrece una representación gráfica del modelo de despliegue y el de componente.

## 4.2 Diseño de la Base de datos.

Las Bases de Datos necesitan de una definición de su estructura que le permitan almacenar datos, reconocer el contenido y recuperar la información. La estructura tiene que ser desarrollada para la necesidad de las aplicaciones que la usarán, esto nos puede ayudar a realizar un proceso del negocio para alcanzar un valor agregado para el cliente. La puesta en práctica de la Base de Datos es el paso final en el desarrollo de aplicaciones de soporte del negocio. Tiene que conformarse con los requisitos del proceso del negocio, que es la primera abstracción de la vista de la Base de Batos.

### 4.2.1 Propósito de la actividad Diseñar la Base de Datos:

- Garantizar que los datos persistentes son almacenados consistente y eficientemente.
- Definir el comportamiento que debe ser implementado en la Base de Datos.

**Tabla 14: Trabajadores de la Base de Datos.**

| Trabajadores | Justificación. |
|---|---|
| Diseñador de la Base de Datos. | Es el responsable de la definición de los detalles del diseño de la Base de Datos. |

**Tabla 15: Papel del Diseñador de la Base de Datos.**

| Trabajadores | Responsable de: | Conocimientos sobre: |
|---|---|---|
| Diseñador de la Base de Datos. | **1.** Tablas.<br>**2.** Índices.<br>**3.** Vistas.<br>**4.** Restricciones.<br>**5.** Disparadores.<br>**6.** Procedimientos almacenados. | **1.** Modelado de datos.<br>**2.** Diseño de Bases de Datos.<br>**3.** Técnicas de análisis y diseño orientado a objetos.<br>**4.** Arquitectura de sistemas.<br>**5.** Administración de Bases de Datos. |

## 4.3 Modelo lógico de datos.

En el diagrama de clases persistentes mostrado, se aprecian las relaciones de asociación entre las clases y se incluyen las llaves primarias de cada clase.

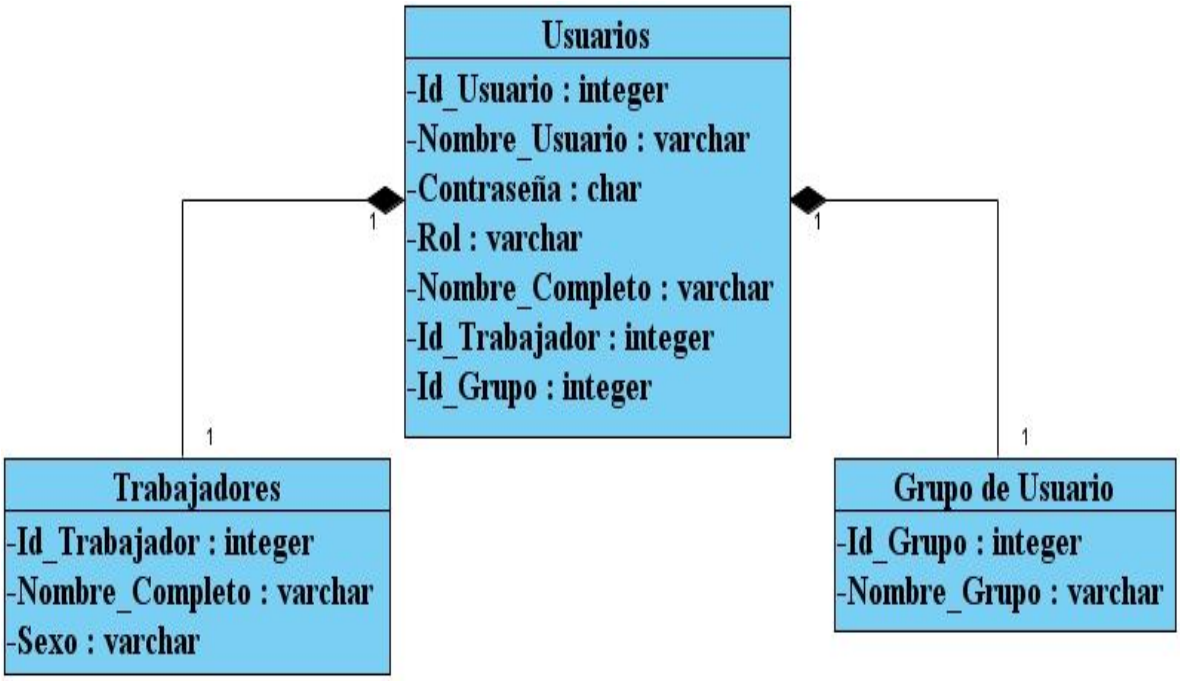



**Figura 17: Diagrama de Clase Persistentes.**

## 4.4 Modelo Físico de datos.

Este modelo describe la estructura lógica de la información que será almacenada en la Base de Datos. Es generado a partir del diagrama de clases persistentes. En la Figura 10, se detallan la estructura de cada tabla que conforma al modelo.

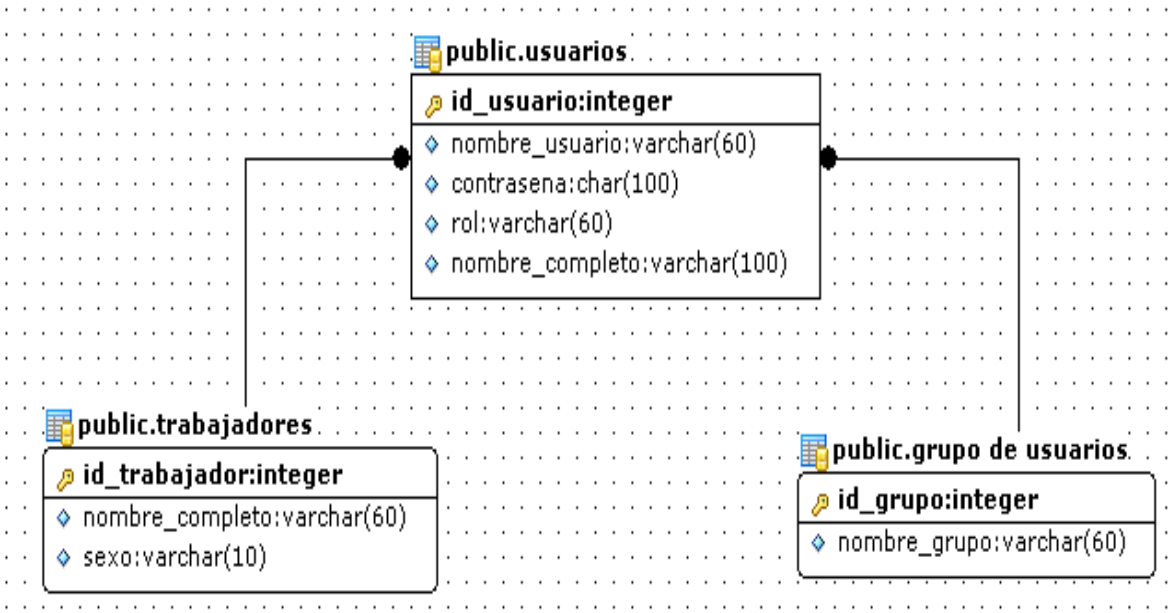



**Figura 10: Modelo Físico de los Datos**

## 4.4.1 Descripción de las tablas.

**Tabla 16: Descripción de la Tabla Clase Usuario.**

| **Nombre de la tabla** | **Tabla_Clase_Usuarios.** | |
|---|---|---|
| **Descripción.** | Almacenar datos de los usuarios que hacen uso de la aplicación. | |
| **Atributos** | **Tipo** | **Descripción de los atributos.** |
| Id_Usuario | integer | Identificador de usuario. |
| Nombre_Usuario | varchar | Nombre con el que la persona se identifica al acceder a la herramienta. |
| Contraseña | char | Contraseña utilizada para entrar a la aplicación. |
| Rol | varchar | Papel que desempeña el usuario al usar el software. |
| Nombre_completo | varchar | Nombre de la persona que hace uso de la herramienta. |

**Tabla 17: Descripción de la Tabla Clase Grupo Usuarios.**

| **Nombre de la tabla** | **Tabla_Clase_Grupo_Usuarios.** | |
|---|---|---|
| **Descripción.** | Almacenar datos de los usuarios que hacen uso de la aplicación. | |
| **Atributos** | **Tipo** | **Descripción de los atributos.** |
| Nombre_Grupo_ Usuarios | varchar | Nombre con el que el grupo se identifica para acceder a la herramienta. |
| Id_Grupo_Usuario | integer | Identificador del grupo. |

**Tabla 18: Descripción de la Tabla Clase Trabajadores**

| **Nombre de la tabla** | **Tabla_Clase_Trabajadores.** | |
|---|---|---|
| **Descripción.** | Almacenar datos de los Trabajadores que hacen uso de la aplicación. | |
| **Atributos** | **Tipo** | **Descripción de los atributos.** |
| Id_trabajador | integer | Identificador del trabajador. |
| Nombre-completo | varchar | Nombre del trabajador en la aplicación |
| sexo | varchar | Tipo de sexo del trabajador. |

## 4.5 Modelo de Despliegue.

Es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño. El software en desarrollo es una aplicación de escritorio (Desktop), que dispone de una Base de Datos, en la que se encuentra almacenada la información requerida para el funcionamiento de la aplicación, a la que pueden estar accediendo

disímiles de usuarios al unísono. Por esta razón se propone contar con un servidor donde la Base de Datos queda instalada y pueda accederse a ella de forma simultánea desde distintas ubicaciones.

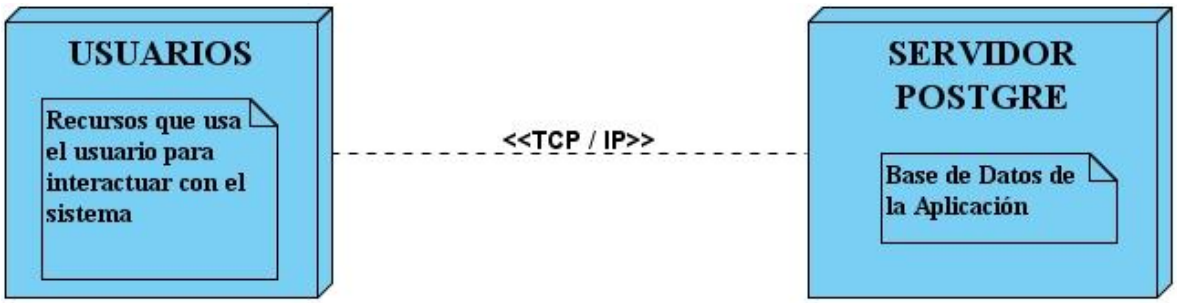



**Figura 18: Diagrama del Modelo de Despliegue.**

## 4.6 Diagrama de Componentes.

Es un diagrama que muestra un conjunto de elementos del modelo tales como componentes, subsistemas de implementación y sus relaciones. Se utiliza para modelar la vista estática de un sistema y muestra la relación y las dependencias lógicas entre un conjunto de componentes de software (Anexo 5).

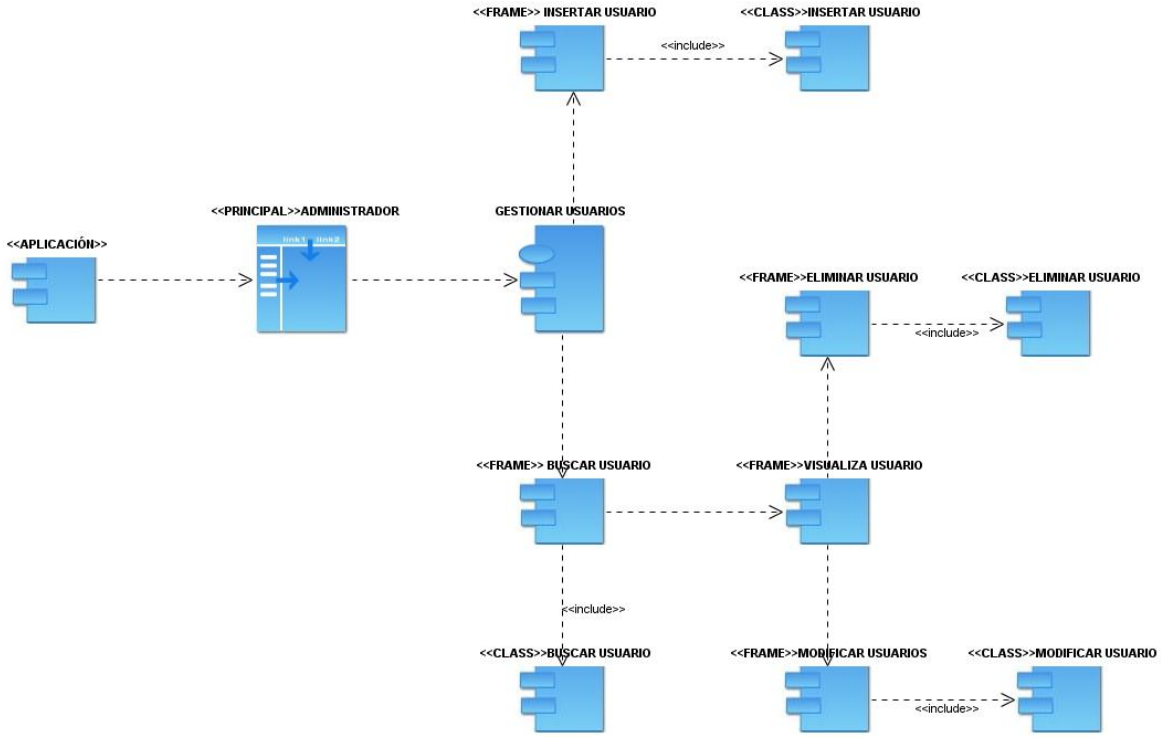



**Figura 19: Diagrama de Componentes Gestionar Usuarios.**

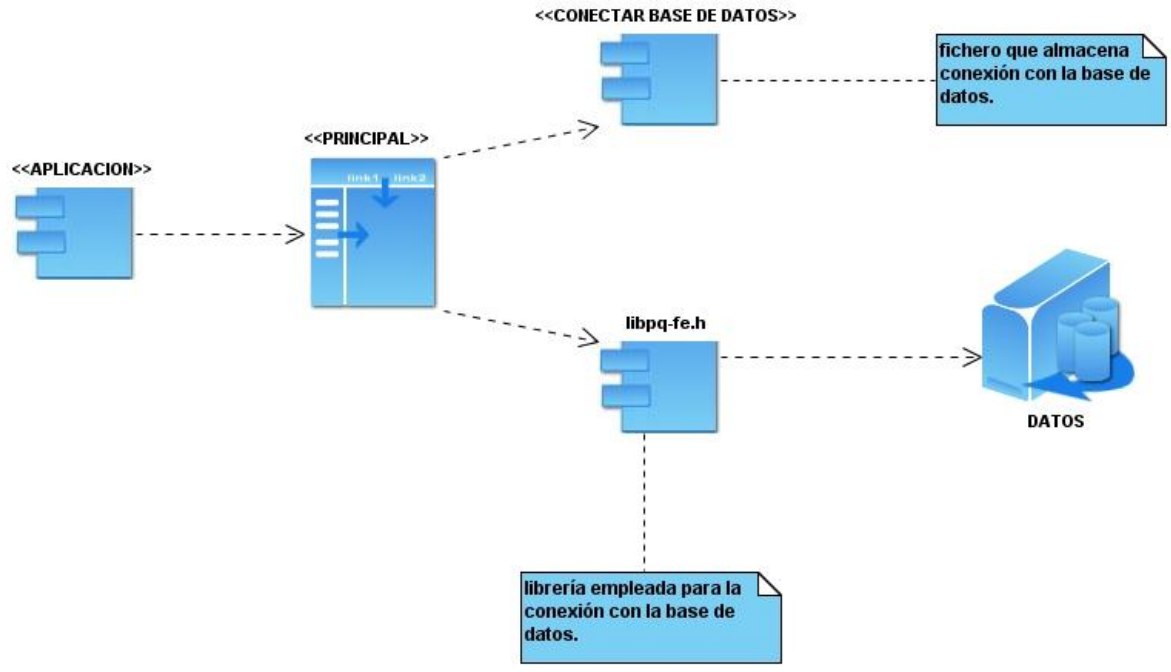



**Figura 20: Diagrama de Componentes Conectar Base de Datos.**

## 4.7 Conclusiones del Capítulo.

Concluido este capítulo hemos finalizado la etapa de diseño e implementación de la Base de Datos, arrojando los Modelos Lógicos y Físicos, además los Diagramas de Despliegue y el de Componentes.

En este capítulo se confeccionaron 3 Tablas que contendrán toda la información referente a los Usuarios, Grupo de Usuarios y Trabajadores registrados en el Sistema, así como las relaciones que existen entre estas Tablas.

Con el Diseño de la Base de Datos se logró una estructura para almacenar datos, reconocer el contenido y recuperar la información Se pude asegurar que se cumplió con todos los objetivos trazados en este capítulo.

# Conclusiones Generales

Culminada la etapa de trabajo se concluye con la realización y confección de una infraestructura de desarrollo de Software para el proyecto “Herramientas Inteligentes para la Gestión del Desarrollo Local en las Comunidades”, cumpliéndose así el objetivo general definido en este trabajo.

Con el desarrollo de este trabajo se determinó:

- Las Tecnologías, Herramientas y Metodología que se emplearán para la construcción de los módulos que conforman el proyecto “Herramientas Inteligentes para la Gestión del Desarrollo Local en las Comunidades”.
- Que el modelo en capas se ha convertido en la arquitectura predominante para la construcción de aplicaciones multiplataforma.
- Los Objetivos y Restricciones que deberán cumplir los módulos, así el Rendimiento y Dimensiones del módulo administrativo, por la que se regirán los otros módulos que conforman las Herramientas.
- El análisis, diseño e implementación del módulo de seguridad del proyecto.

De manera general

Como resultado de este trabajo se obtuvo un framework o infraestructura de desarrollo para la aplicación completa, centrándose principalmente es el aspecto de la misma, además se proporcionó una estructuración y una metodología para organizar, desarrollar y unir los diferentes componentes del proyecto.

# Recomendación

A partir de los resultados obtenidos de la presente investigación se concluye con la necesidad de proponer:

- Utilizar de inmediato la arquitectura propuesta como apoyo para la confección de los módulos que conformas el proyecto “Herramientas Inteligentes para la Gestión de Desarrollo Local en las Comunidades”.
- Continuar con el desarrollo de software, para alcanzar el objetivo final, que es poder diseñar todos los módulos que conforman el proyecto a partir de la arquitectura diseñada.
- Introducir los resultados de este trabajo en la carrera de Informática, para que los estudiantes lo utilicen para la confección de cualquier módulo de seguridad, ya que el presente trabajo se convierte en el primer antecedente de la carrera.
- Realizar una continua actualización de las herramientas utilizadas en el framework.

# Referencias Bibliográficas


- **CAVSI. 2004.** ¿Qué es un Sistema Gestor de Bases de Datos o SGBD? *http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.* [En línea] 2004.

- **Corp, Neuron. 2006.** ¿Qué es UML? *http://www.neuronsrl.com.ar/training/uml/uml_intro.html.* [En línea] 2006.

- **Electrónica, Laboratorio III de. 2001 .** *Anotaciones RUP .* 2001

- **Freedownloadmanager.org. 2004.** Visual Paradigm for UML. *http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3.* [En línea] 2004.

- **Guerra, Maité Sosa Veranes y Vladimir Gonzales. 2009.** *Tesis ¨Sistema de Gestión de la Facultad 2¨.* 2009. págs. 38 y 50 - 54.

- **Gutiérrez, Jorge A. Saavedra. 2007.** El Mundo Informático. Software Libre. *http://jorgesaavedra.wordpress.com/2007/05/05/lenguajes-de-programacion/.* [En línea] 2007.
  http://es.answers.yahoo.com/question/index. [En línea]

- **Jacobson, I., Booch, G., & Rumbaugh, J. 2000.** El proceso unificado de desarrollo del software. Capitulo 1.
  *http://www.histaintl.com/servicios/consulting/rup*
  *http://www.encamina.com/boletines/ENCAMINA%20y%20las%20metodolog%C3%ADas%20software.* [En línea] 2000.

- **Janium. 2009.** Aplicaciones basadas en Web. *http://www.janium.com/page2/page1/page6/page7/page7.html.* [En línea] 2009.

- **Monografias. 2007.** Definición arquitectura cliente servidor. *http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml.* [En línea] 2007.

- **Qt. 2009.** http://pixelcoblog.com/qt-creator-completo-entorno-de-desarrollo-multiplataforma/. *http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/.* [En línea] 2009.

- **Roche, Katiusca Jiménez. 2010.** *Tesis ¨Obtención del Modelo geométrico para el diseño y explotación de canteras de materiales de construcción.¨.* 2010. págs. 17 - 18.

- **Sandó, Lesther Delgado Pérez y José Rolando Pérez. 2009.** *Tesis ¨Herramientas para la revisión y seguimiento de errores de documentación en los Proyectos de la Facultad 7 ¨.* 2009. págs. 25 - 30.

- **Solís Álvarez, Camilo Javier and Figueroa Díaz, Roberth Gustavo. 2005.** *Metodologías Tradicionales vs. Metodologías Ágiles.* 2005.

- **Subirós, Dariel Raúl. Junio 2009.** *Tesis ¨Desarrollo de una interfaz gráfica de usuario para el Procesador Meteorológico¨.* Junio 2009. pág. 9 y 15.

- **Vico. 2002.** Unifiel Modeling Language. *http://www.vico.org/FormMentorOutsourcingUML.pdf.* [En línea] 2002.

# Bibliografía


1 **BOOCH, Grady, RUMBAUGH, James, JACOBSON y Ivar. 2000..** *"El lenguaje unificado de modelado. Manual de referencia".* s.l. : Addison Wesley. Capítulos 4-5, 8, 12 Páginas 41-64, 93-102, 147-158, 2000.

2 **CAVSI. 2004.** ¿Qué es un Sistema Gestor de Bases de Datos o SGBD? *http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.* [En línea] 2004.

3 **Corp, Neuron. 2006.** ¿Qué es UML? *http://www.neuronsrl.com.ar/training/uml/uml_intro.html.* [En línea] 2006.

4 **Electrónica, Laboratorio III de. 2001 .** *Anotaciones RUP .* 2001 .

5 **Freedownloadmanager.org. 2004.** Visual Paradigm for UML. *http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3.* [En línea] 2004.

6 **Guerra, Maité Sosa Veranes y Vladimir Gonzales. 2009.** *Tesis ¨Sistema de Gestión de la Facultad 2¨.* 2009. págs. 38 y 50 - 54.

7 **Gutiérrez, Jorge A. Saavedra. 2007.** El Mundo Informático. Software Libre. *http://jorgesaavedra.wordpress.com/2007/05/05/lenguajes-de-programacion/.* [En línea] 2007. http://es.answers.yahoo.com/question/index. [En línea]

8 **Jacobson, I., Booch, G., & Rumbaugh, J. 2000.** El proceso unificado de desarrollo del software. Capitulo 1. *http://www.histaintl.com/servicios/consulting/rup http://www.encamina.com/boletines/ENCAMINA%20y%20las%20metodolog%C3%ADas%20software.* [En línea] 2000.

9 **JACOBSON, Ivar, RUMBAUGH, James y BOOCH, Grady,. .2000. .** *"El proceso unificado de desarrollo".* s.l. : Addison Wesley. capítulos 8 Páginas 165-181, 185-204, .2000.

10 **Janium. 2009.** Aplicaciones basadas en Web. *http://www.janium.com/page2/page1/page6/page7/page7.html.* [En línea] 2009.

11 **LARMAN, C. 2004.** *UML y patrones. Introducción al análisis y diseño orientado a objeto.3 ed. .* La Habana: : Editorial Félix Varela, . 507 p., 2004.

12 **Monografias. 2007.** Definición arquitectura cliente servidor. *http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml.* [En línea] 2007.

13 **PRESSMAN, R.S. 2005..** *Ingeniería de Software: Un enfoque práctico. .* La Habana. : Editorial Félix Varela., 2005.

14 **Qt. 2009.** http://pixelcoblog.com/qt-creator-completo-entorno-de-desarrollo-multiplataforma/. *http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/.* [En línea] 2009.

15 **Roche, Katiusca Jiménez. 2010.** *Tesis ¨Obtención del Modelo geométrico para el diseño y explotación de canteras de materiales de construcción.¨.* 2010. págs. 17 - 18.

16 **RUMBAUGH, James, JACOBSON, Ivar y BOOCH, Grady,. .2000..** *"El lenguaje unificado de modelado. Manual de referencia".* s.l. : Addison Wesley. Capítulos 4 y 13 Páginas 42-48, 122-125, 131-143, 156, 162-170, 191-197, 211, 299-303, 367-373, 399-402, 427, 434-435, 446-447., .2000.

17 **Sandó, Lesther Delgado Pérez y José Rolando Pérez. 2009.** *Tesis ¨Herramientas para la revisión y seguimiento de errores de documentación en los Proyectos de la Facultad 7 ¨.* 2009. págs. 25 - 30.

18 **Solís Álvarez, Camilo Javier and Figueroa Díaz, Roberth Gustavo. 2005.** *Metodologías Tradicionales vs. Metodologías Ágiles.* 2005.

19 **Subirós, Dariel Raúl. Junio 2009.** *Tesis ¨Desarrollo de una interfaz gráfica de usuario para el Procesador Meteorológico¨.* Junio 2009. pág. 9 y 15.

20 **Vico. 2002.** Unifiel Modeling Language. *http://www.vico.org/FormMentorOutsourcingUML.pdf.* [En línea] 2002.

# Glosario de Términos

- **Herramienta CASE:** Programas que se utilizan para crear los modelos de datos. Mediante esta es posible desplazarse por todas las etapas del ciclo de desarrollo de un sistema, documentar las ideas y conceptos que se le ocurran, y llevar al día los convenios de denominación.
- **SGBD:** Sistema Gestor de Base de datos, conjunto de programas que permiten crear y mantener una base de datos garantizando la seguridad, confidencialidad e integridad.
- **Multiplataforma:** Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.
- **Metodologías de Desarrollo:** Se define como un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas e información.
- **RUP:** El Proceso Unificado Racional o RUP (Rational Unified Process), es un proceso desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos y roles.
- **UML:** es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software.
- **Software:** es la suma total de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo.
- **Software Libre:** es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente, aunque conserve su carácter de libre, puede ser vendido comercialmente.

- **API:** una interfaz de programación de aplicaciones o API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
- **Mainframe:** Tipo de computadora más potente y más rápida que existen en un momento dado. Es de gran tamaño, la más grande entre sus paredes. Puede procesar enormes cantidades de información en poco tiempo, pudiendo ejecutar millones de instrucciones por segundo. Está destinada a una tarea específica y poseen una capacidad de almacenamiento enorme.
- **Framework:** En el desarrollo de Software, un framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.
- **C++:** Es un lenguaje híbrido, que se puede compilar. Las principales características son abstracción (encapsulación), el soporte para programación orientada a objetos (polimorfismo) y el soporte de plantillas o programación genérica (templates). Es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.
- **TCP:** Transmission Control Protocol, en español Protocolo de Control de Transmisión, garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto.
- **IP:** El Protocolo de Internet (en inglés Internet Protocol) es un protocolo no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados.

# Anexos

## Anexo 1: Descripción de Caso de Uso del Sistema.

**Tabla 1.1: Descripción del caso de uso Buscar Usuarios.**

| **Caso de Uso** | **Buscar Usuarios** |
|---|---|
| **Actores** | Administrador (inicia) |
| **Propósito** | Permitir ver un usuario existente en la aplicación. |
| **Resumen** | El Caso de Uso se inicia cuando el Administrador desea acceder a la información de un usuario del sistema para realizarse cualquier cambio. |
| **Curso Normal de los Eventos** | |
| **Acciones del Actor** | **Respuesta del Sistema** |
| 1. El Administrador selecciona la opción Buscar Usuarios.<br>2. El Administrador llena los campos. | 1.1 El sistema muestra unos campos para llenar.<br>2.1 El sistema muestra los datos del usuario. |
| **Flujo Alterno** | |
| | 1.1 Si el Administrador deja algún campo obligatorio vacío, el sistema muestra el mensaje de error ¨Debe completar los campos vacíos¨. |
| 2. El administrador cancela la acción. | 2.1 El sistema cancela la acción. |

**Tabla 1.2: Descripción del caso de uso Listar Usuarios.**

| **Caso de Uso** | **Listar Usuarios** |
|---|---|
| **Actores** | Administrador (inicia) |
| **Propósito** | Permitir ver listado de los usuarios existentes en la aplicación. |

| **Resumen** | El Caso de Uso se inicia cuando el Administrador desea acceder al listado de Usuarios del Sistema. |
|---|---|
| **Curso Normal de los Eventos** | |
| **Acciones del Actor** | **Respuesta del Sistema** |
| 1. El Administrador selecciona la opción Listar Usuarios. | 1.1 El sistema muestra el listado de los Usuarios de la Aplicación y termina así el caso de uso. |
| **Flujo Alterno** | |
| 2. El administrador cancela la acción. | 2.1 El sistema cancela la acción. |

**Tabla 1.3: Descripción del caso de uso Buscar Grupo de Usuarios.**

| **Caso de Uso** | **Buscar Grupo de Usuarios** |
|---|---|
| **Actores** | Administrador (inicia) |
| **Propósito** | Permitir ver un grupo de usuarios existente en la aplicación. |
| **Resumen** | El Caso de Uso se inicia cuando el Administrador desea acceder a la información de un grupo de usuarios del sistema para realizarse cualquier cambio. |
| **Curso Normal de los Eventos** | |
| **Acciones del Actor** | **Respuesta del Sistema** |
| 1. El Administrador selecciona la opción Buscar Grupo de Usuarios.<br>2. El Administrador llena los campos. | 1.1 El sistema muestra unos campos para llenar.<br><br>2.1 El sistema muestra los datos del grupo de usuarios. |
| **Flujo Alterno** | |

| | |
|---|---|
| | 1.1 Si el Administrador deja algún campo obligatorio vacío, el sistema muestra el mensaje de error ¨Debe completar los campos vacíos¨. |
| 2. El administrador cancela la acción. | 2.1 El sistema cancela la acción. |

**Tabla 1.4: Descripción caso de eso Listar Grupo de Usuarios**

| **Caso de Uso** | **Listar Grupo de Usuarios** |
|---|---|
| **Actores** | Administrador (inicia) |
| **Propósito** | Permitir ver listado de los grupos de usuarios existentes en la aplicación. |
| **Resumen** | El Caso de Uso se inicia cuando el Administrador desea acceder al listado de Grupos de Usuarios. |
| **Curso Normal de los Eventos** | |
| **Acciones del Actor** | **Respuesta del Sistema** |
| 1. El Administrador selecciona la opción Listar Grupo de Usuarios. | 1.1 El sistema muestra el listado de los Grupos de Usuarios de la Aplicación y termina así el caso de uso. |
| **Flujo Alterno** | |
| 2. El administrador cancela la acción. | 2.1 El sistema cancela la acción. |

**Tabla 1.5: Descripción del caso de uso Gestionar Grupos de Usuarios.**

| **Caso de Uso** | **Gestionar Grupos Usuario** |
|---|---|
| **Actores** | Administrador (inicia) |
| **Propósito** | Insertar, eliminar o modificar un nuevo grupo de usuarios |
| **Resumen** | En este caso de uso el administrador podrá mantener actualizada la información referente a los grupos de |

| | |
|---|---|
| | usuarios que interactúan con el sistema.<br>El caso de uso comienza cuando el administrador solicita "Gestionar Grupos de Usuarios", ya sea para insertar un nuevo grupo, eliminar uno existente o modificar sus datos. Según la opción escogida, el sistema actualiza los cambios realizados en la base de datos, finalizando la realización de este caso de uso. |
| **Precondiciones** | En caso de que se desea eliminar o modificar un grupo de usuarios determinado, este debe encontrarse registrado en el sistema. |

| **Curso Normal de Eventos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| 1.El administrador accede a la interfaz de administración | 2. El sistema muestra una serie de acciones a realizar. |
| 3. El administrador elige la acción a realizar. | 4.- Si elige:<br>4.1 Insertar un nuevo grupo ir a la sección "Insertar Grupo de Usuarios"<br>4.2 Eliminar un grupo ir a la sección "Eliminar Grupo de Usuarios".<br>4.3 Modificar un grupo ir a la sección "Modificar Grupo de Usuarios". |
| **Sección "Insertar Grupo de Usuarios"** | |
| | 1. El sistema muestra la interfaz de insertar grupo, mostrando un formulario con los campos generales que se deben introducir. |
| 2. El administrador introduce los datos (id_grupo de usuario, nombre del grupo de usuario) | 3. El sistema verifica que no exista ese grupo en el sistema.<br>4. El sistema valida los datos introducidos. |

| | |
|---|---|
| | 5. Inserta el nuevo grupo en el sistema y muestra un mensaje ¨el grupo ha sido insertado correctamente¨. |
| **Flujos Alternos** | |
| | 3.1 Si el grupo existe en la Base de Datos el sistema muestra un mensaje de error ¨El grupo se encuentra registrado en el sistema¨. |
| | 4.1 El sistema muestra un mensaje de error especificando que los datos son incorrectos. |
| **Sección "Modificar Grupo de Usuarios"** | |
| | 1. El sistema muestra un formulario con un listado de los Grupos existentes en la base de datos. |
| 2- El administrador selecciona el grupo que desea modificar. | 3. El sistema muestra la información del grupo seleccionado en un formulario. |
| 4- El administrador realiza los cambios pertinentes. | 5. El sistema verifica que los campos obligatorios estén llenos.<br>6. El sistema guarda los datos modificados en la base de datos y el sistema muestra un mensaje ¨los datos del grupo han sido modificados¨. |
| **Flujos Alternos** | |
| | 5.1 Si el Administrador deja algún campo obligatorio vacío, el sistema muestra el mensaje de error ¨Debe completar los campos vacíos¨. |
| **Sección "Eliminar Grupo de Usuarios"** | |

| | |
|---|---|
| | 1. El sistema muestra en un formulario el listado de los grupos existentes. |
| 2- El administrador elige el grupo a eliminar y presiona el botón ¨Eliminar¨. | 3- El sistema pide confirmación de que desea eliminar el grupo. |
| 4- El administrador confirma la acción. | 5- El sistema elimina el grupo seleccionado. |
| **Flujos Alternos** | |
| 4.1 El administrador cancela la acción. | 4.2 El sistema desmarca el grupo seleccionado. |
| **Poscondiciones** | Se registra un grupo, se modifica un grupo o se elimina un grupo. |

**Tabla 1.6: Descripción caso de uso Gestionar Trabajadores.**

| **Caso de Uso** | **Gestionar Trabajadores** |
|---|---|
| **Actores** | Secretaria y Administrador (inicia) |
| **Propósito** | Insertar un trabajador para que interactúe con el sistema modificar datos de estos trabajadores. |
| **Resumen** | En este caso de uso la Secretaria podrá mantener actualizada la información referente a los trabajadores que interactúan con el sistema.<br>El caso de uso comienza cuando la secretaria solicita "Gestionar Trabajadores", ya sea para insertar un nuevo trabajador o modificar datos uno existente. Según la opción escogida, el sistema actualiza los cambios realizados en la base de datos, finalizando la realización de este caso de uso. |
| **Precondiciones** | En caso de que se desea modificar un trabajador determinado, este debe encontrarse registrado en el |

| | |
|---|---|
| | sistema. |
| **Curso Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. La Secretaria accede a la interfaz de Secretaria. | 2. El sistema muestra una serie de acciones a realizar. |
| 3. La Secretaria elige la acción a realizar. | 4.- Si elige:<br>4.1 Insertar un nuevo trabajador ir a la sección “Insertar trabajador”<br>4.2 Modificar un trabajador ir a la sección “Modificar trabajador” |
| **Sección “Insertar Usuario”** | |
| | 1. El sistema muestra la interfaz de insertar trabajador, mostrando un formulario con los campos generales que se deben introducir. |
| 2. El administrador introduce los datos (id_trabajador, nombre completo y sexo). | 3. El sistema verifica que no exista ese trabajador en el sistema.<br>4. El sistema valida los datos introducidos.<br>5. Inserta el nuevo trabajador en el sistema y muestra un mensaje ¨el trabajador ha sido insertado correctamente¨. |
| **Flujos Alternos** | |
| | 3.1 Si el trabajador existe en la Base de Datos el sistema muestra un mensaje de error ¨El trabajador se encuentra registrado en el sistema¨. |
| | 4.1 El sistema muestra un mensaje de error especificando que los datos |

| | |
|---|---|
| | son incorrectos. |
| **Sección "Modificar Usuario"** | |
| | 1. El sistema muestra un formulario con un listado de los trabajadores existentes en la base de datos. |
| 2- La Secretaria selecciona el trabajador que desea modificar. | 3. El sistema muestra la información del trabajador seleccionado en un formulario. |
| 4- La Secretaria realiza los cambios pertinentes. | 5. El sistema verifica que los campos obligatorios estén llenos.<br>6. El sistema guarda los datos modificados en la base de datos y el sistema muestra un mensaje ¨los datos del trabajador han sido modificados¨. |
| **Sección "Eliminar Trabajador"** | |
| | 1. El sistema muestra en un formulario el listado de los trabajadores existentes. |
| 2- El administrador elige el trabajador a eliminar y presiona el botón ¨Eliminar¨. | 3- El sistema pide confirmación de que desea eliminar el trabajador. |
| 4- El administrador confirma la acción. | 5- El sistema elimina el grupo seleccionado. |
| **Flujos Alternos** | |
| 4.1 El administrador cancela la acción. | 4.2 El sistema desmarca el trabajador seleccionado. |
| **Poscondiciones** | Se registra un trabajador, se modifica un trabajador o se elimina un trabajador. |

**Tabla 1.7: Descripción del caso de uso Buscar Trabajador.**

| **Caso de Uso** | **Buscar Trabajador** |
|---|---|
| **Actores** | Administrador y Secretaria (inicia) |
| **Propósito** | Permitir ver los trabajadores existentes en la aplicación. |
| **Resumen** | El Caso de Uso se inicia cuando el Administrador o Secretaria desea acceder a la información de un trabajador del sistema para realizarse cualquier cambio. |
| **Curso Normal de los Eventos** | |
| **Acciones del Actor** | **Respuesta del Sistema** |
| 1. El Administrador o Secretaria selecciona la opción Buscar Trabajador.<br>2. El Administrador o Secretaria llena los campos. | 1.1 El sistema muestra unos campos para llenar.<br><br>2.1 El sistema muestra los datos del trabajador. |
| **Flujo Alterno** | |
| | 1.1 Si el Administrador deja algún campo obligatorio vacío, el sistema muestra el mensaje de error ¨Debe completar los campos vacíos¨. |

**Tabla 1.8: descripción caso de uso Listar Trabajador.**

| **Caso de Uso** | **Listar Trabajador** |
|---|---|
| **Actores** | Administrador o Secretaria(inicia) |
| **Propósito** | Permitir ver listado de los trabajadores existentes en la aplicación. |
| **Resumen** | El Caso de Uso se inicia cuando el Administrador o Secretaria desea acceder al listado de trabajadores. |
| **Curso Normal de los Eventos** | |
| **Acciones del Actor** | **Respuesta del Sistema** |

| | |
|---|---|
| 1. El Administrador o Secretaria selecciona la opción Listar Trabajador. | 1.1 El sistema muestra el listado de los Trabajadores de la Aplicación y termina así el caso de uso. |
| **Flujo Alterno** | |
| 2. El administrador cancela la acción. | 2.1 El sistema cancela la acción. |

**Tabla 1.9: Conexión con Base de Datos**

| **Caso de Uso** | **Conexión Base de Datos** |
|---|---|
| **Actores** | Administrador (inicia) |
| **Propósito** | Permitir la conexión con la base de datos. |
| **Resumen** | El Caso de Uso se inicia cuando el Administrador desea acceder a la base de datos del sistema. |
| **Curso Normal de los Eventos** | |
| **Acciones del Actor** | **Respuesta del Sistema** |
| 1. El Administrador selecciona la opción conexión base de datos.<br>2. El Administrador llena los campos (host, nombre base de datos, usuario y contraseña de acceso a la base de datos). | 1.1 El sistema muestra unos campos para llenar.<br><br>2.1 El sistema conecta con la base de datos. |
| **Flujo Alterno: -** | |
| **Poscondición** | La conexión se guarda en un fichero. |

# Anexo 2: Diagramas de Clase del Análisis.

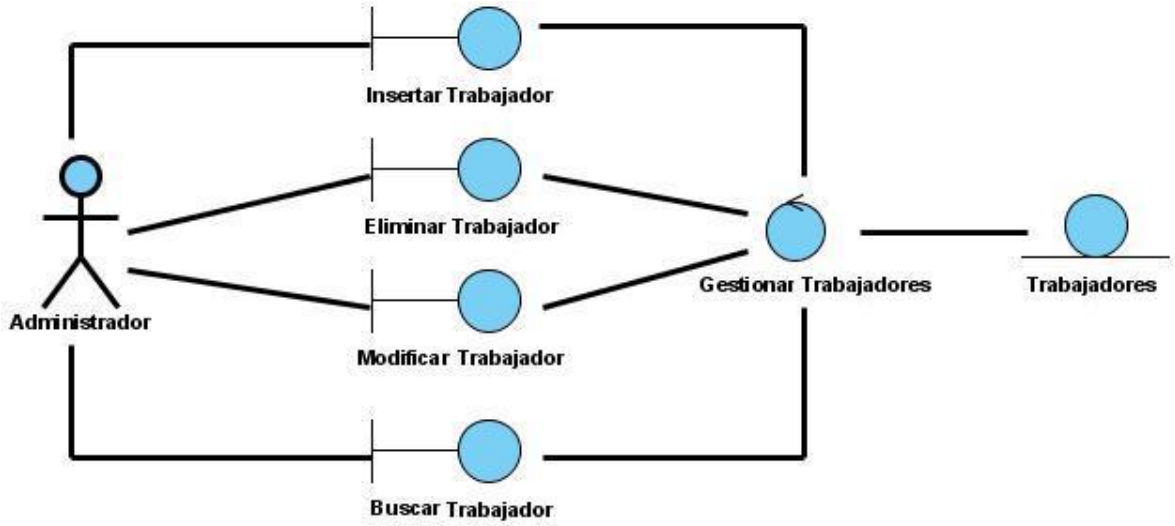



Figura 2.1: Diagrama Gestionar Trabajadores.

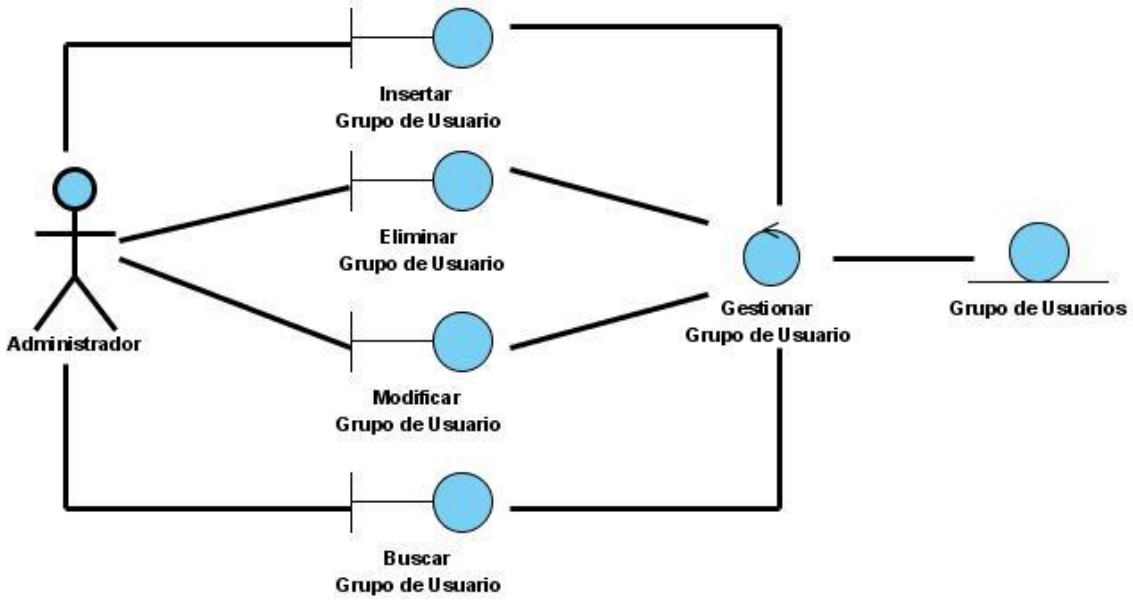



Figura 2.2: Diagrama Gestionar Grupo de Usuarios.

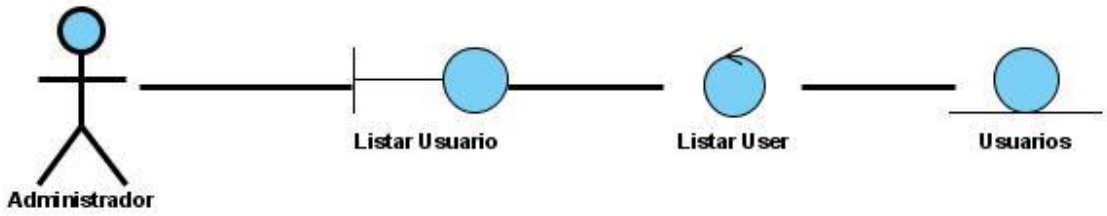



Figura 2.3: Diagrama Listar Usuarios.

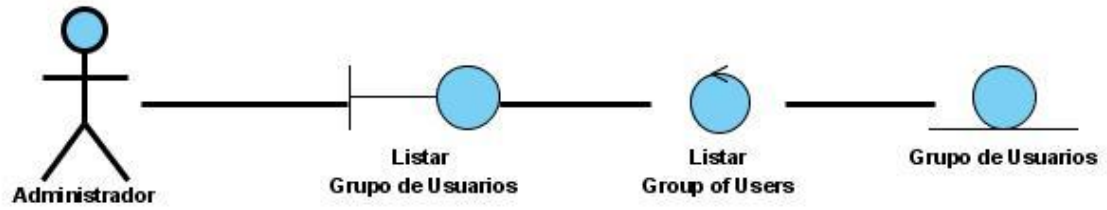



Figura 2.4: Diagrama Listar Grupo de Usuario.

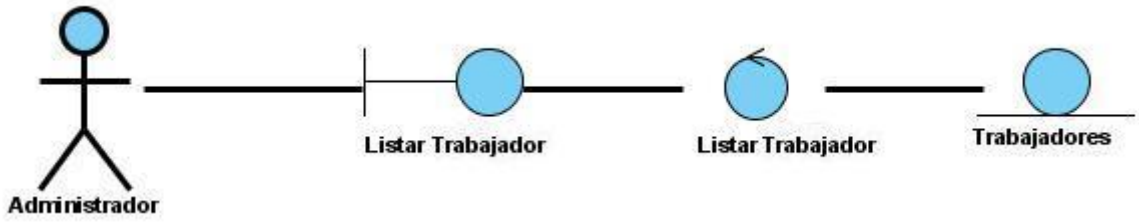



Figura 2.5: Diagrama Listar Trabajadores.

# Anexo 3: Diagramas de Colaboración.

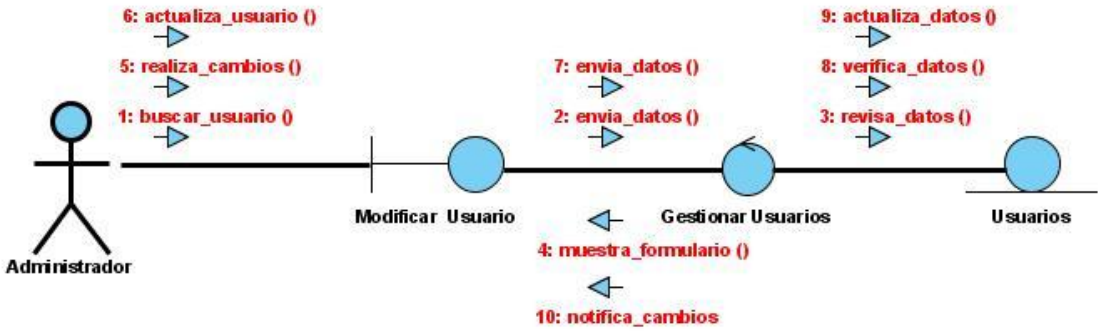



**Figura 3.1: Diagrama Modificar Usuario.**

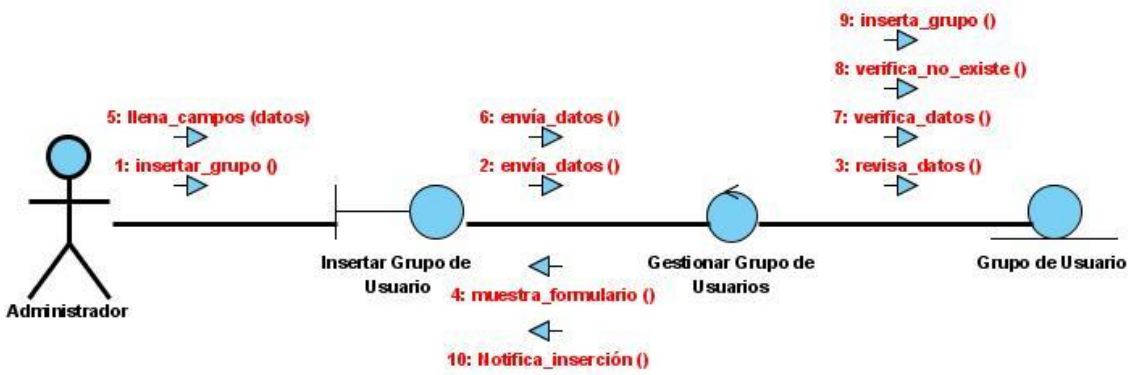



**Figura 3.2: Diagrama Insertar Grupo de Usuarios.**

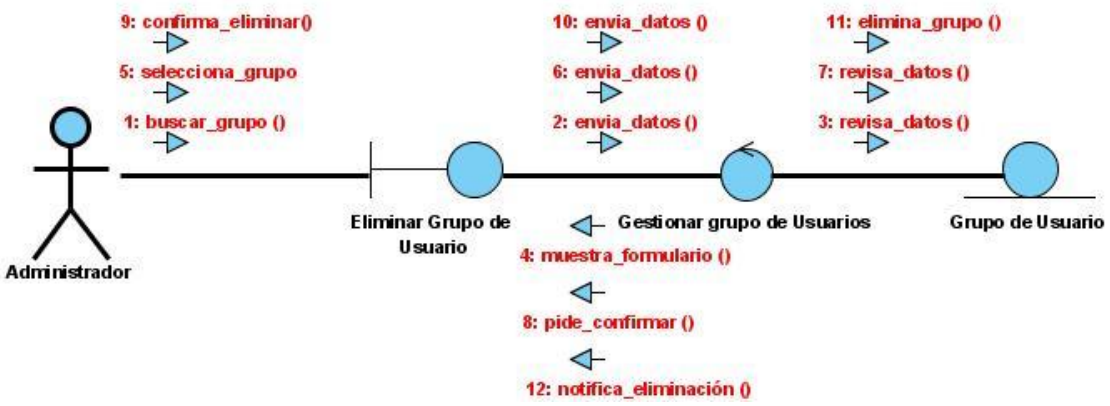



**Figura 3.3: Diagrama Eliminar Grupo de Usuarios.**

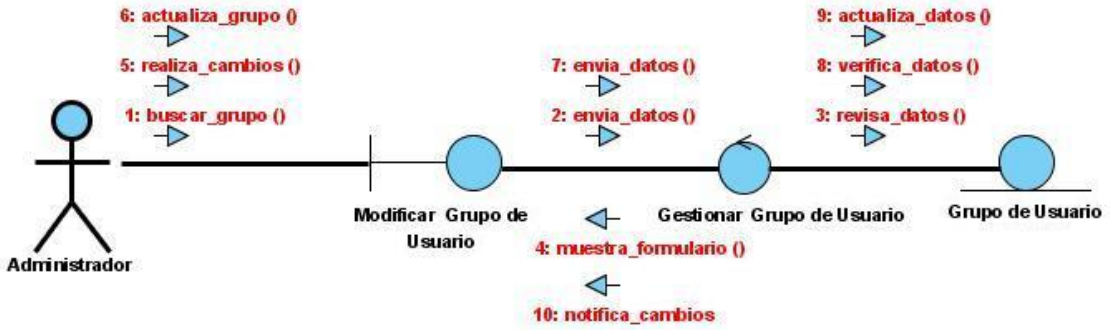



**Figura 3.4: Diagrama Modificar Grupo de Usuarios.**

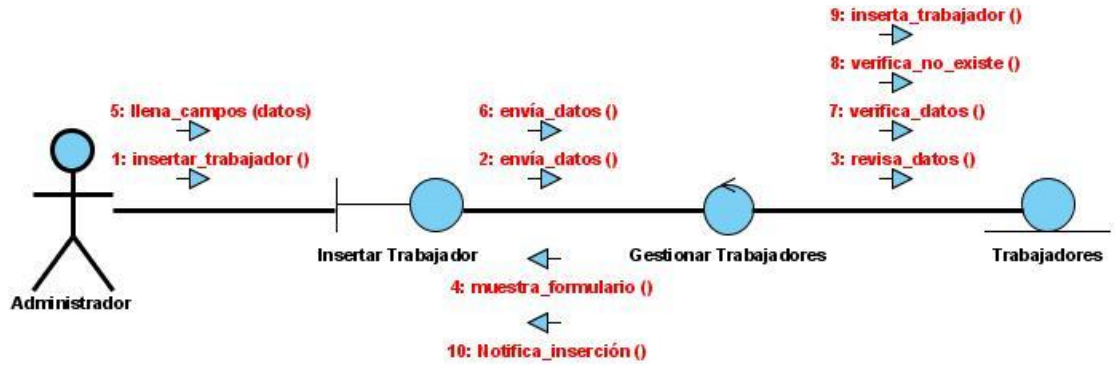



**Figura 3.5: Diagrama Insertar Trabajador.**

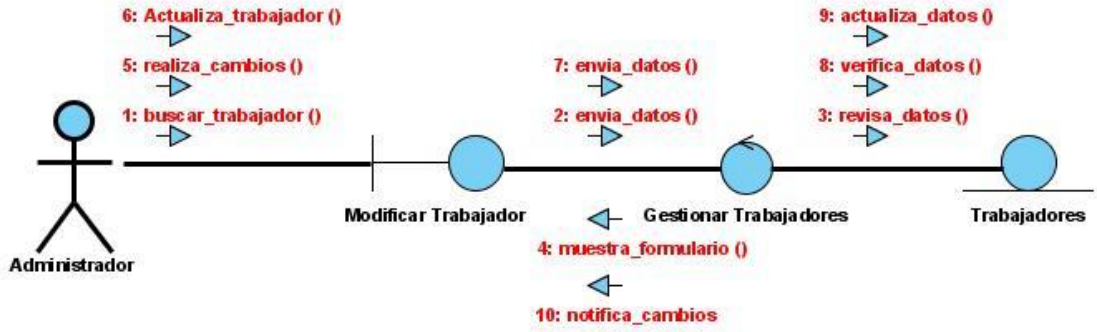



**Figura 3.6: Diagrama Modificar Trabajador.**

# Anexo 4: Diagrama de Componentes.

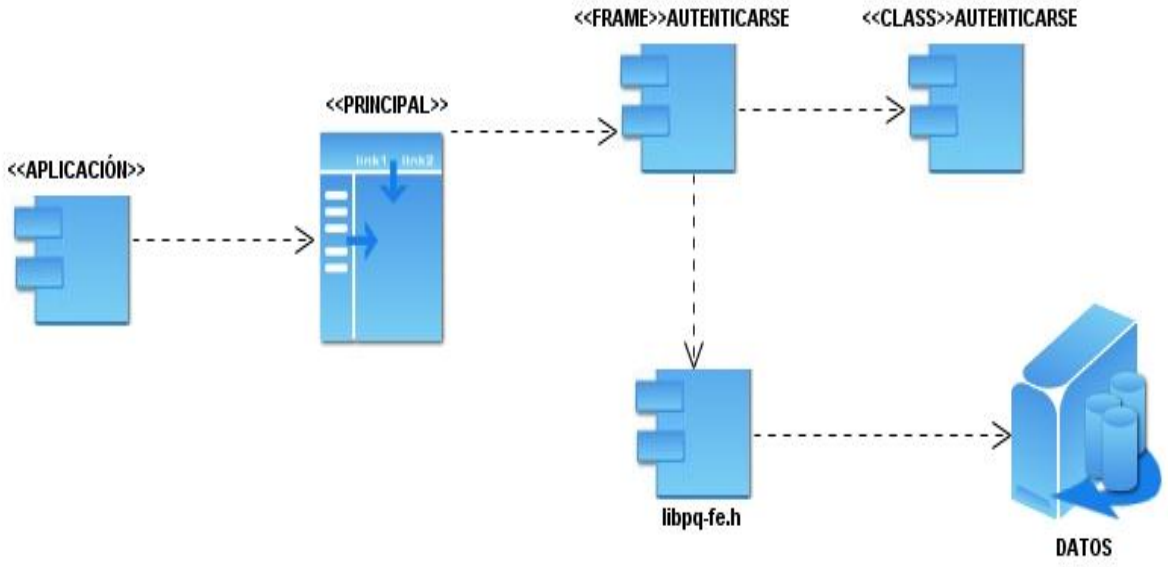



**Figura 4.1: Autenticar Usuario.**

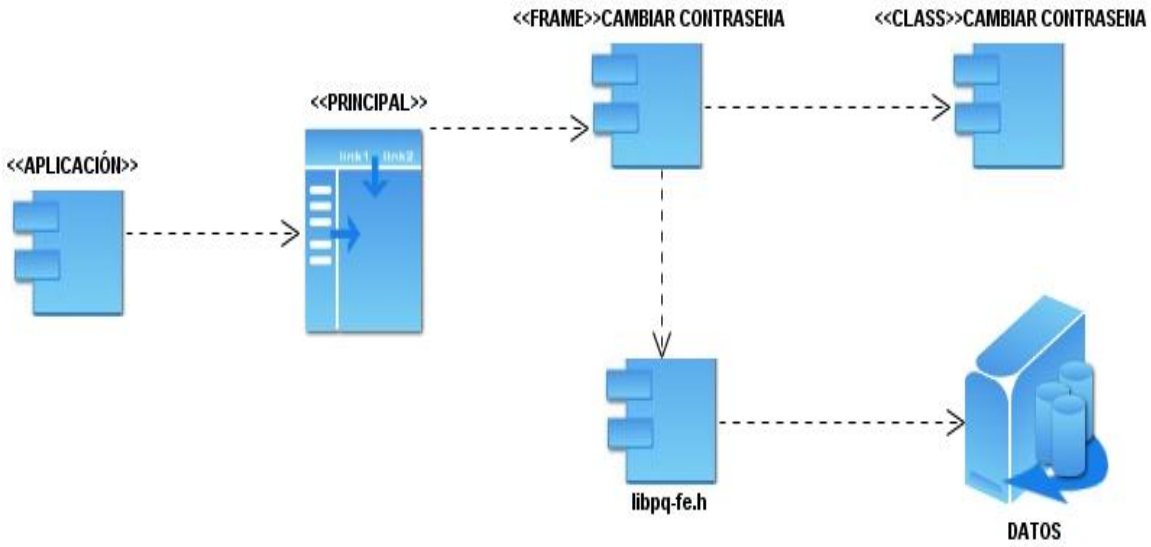



**Figura 4.2: Cambiar Contraseña.**

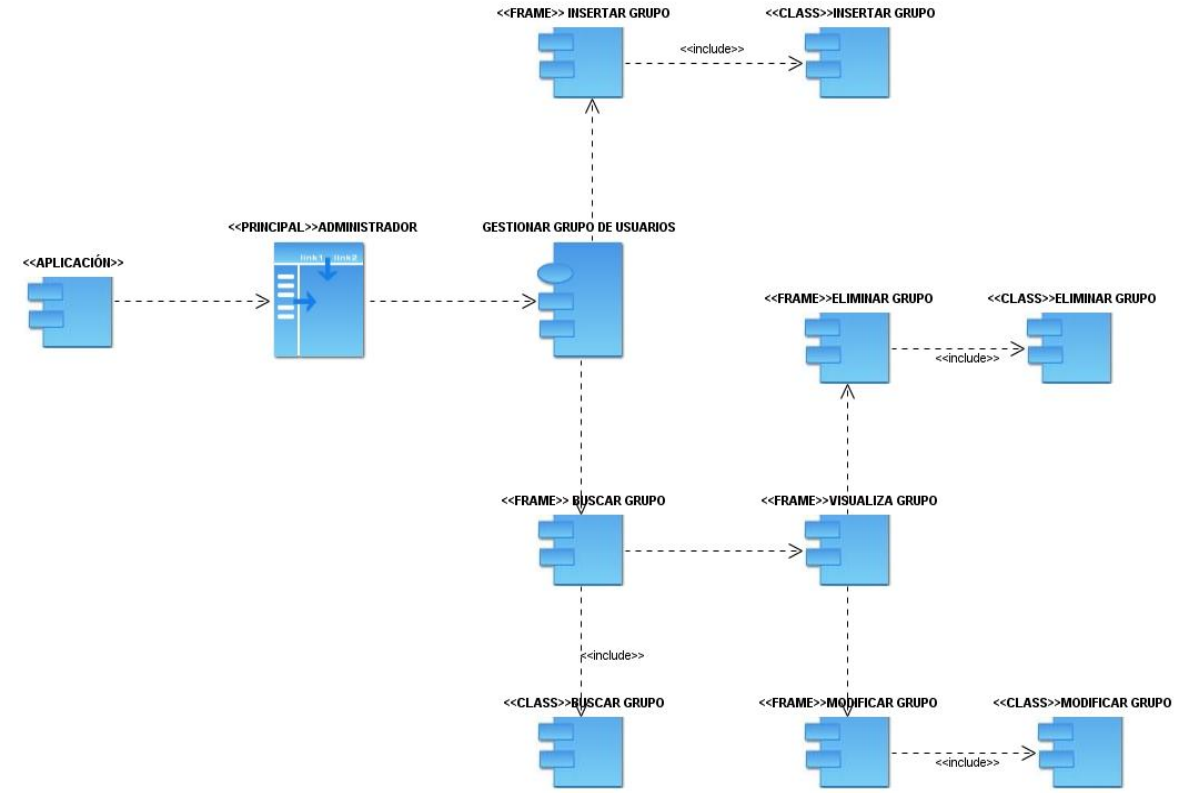



**Figura 4.3: Gestionar Grupo de Usuarios.**

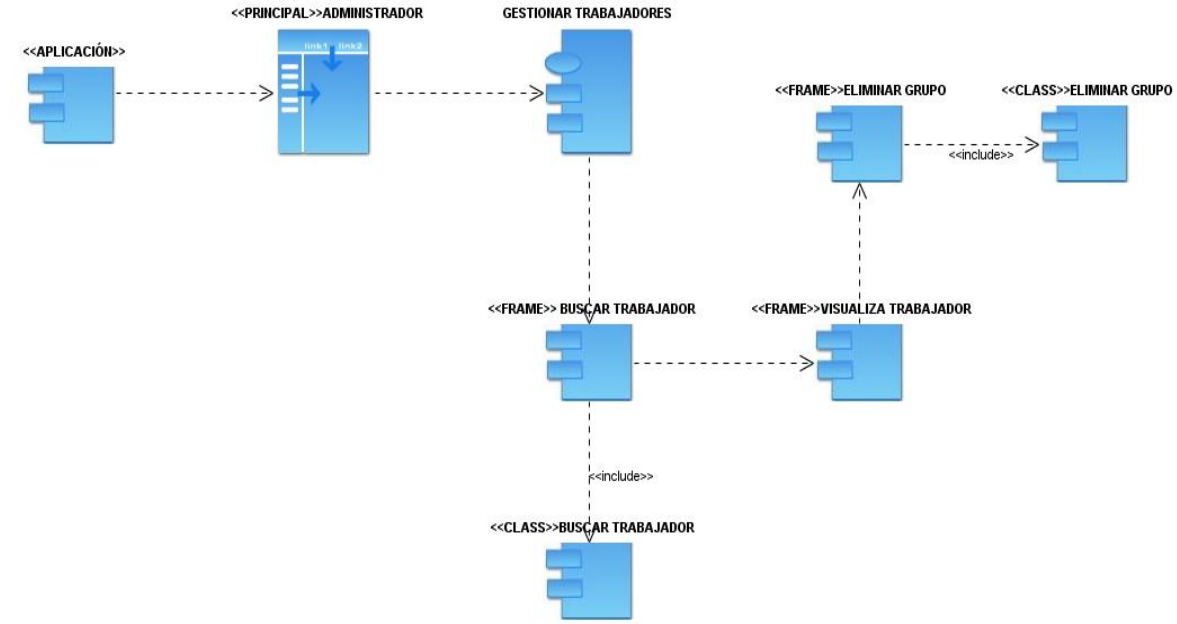



**Figura 4.4: Gestionar Trabajador (Administrador).**

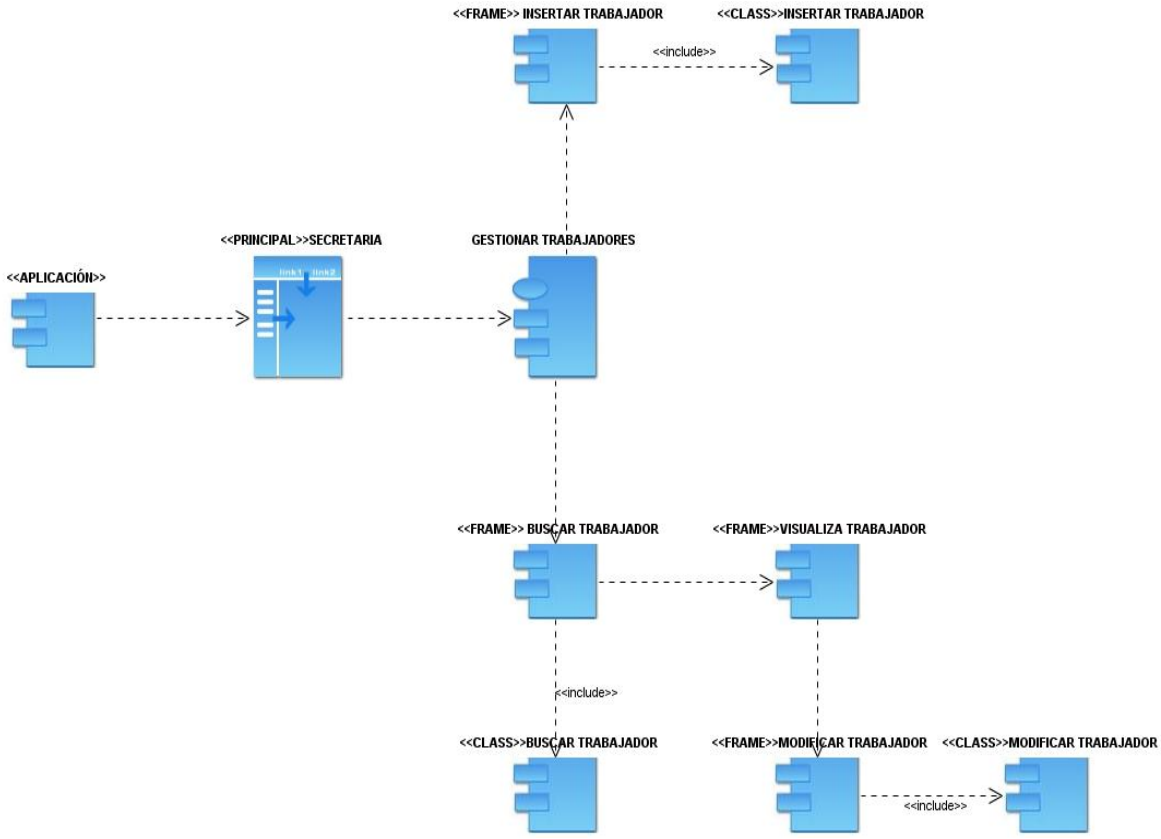



**Figura 4.5: Gestionar Trabajador (Secretaria).**

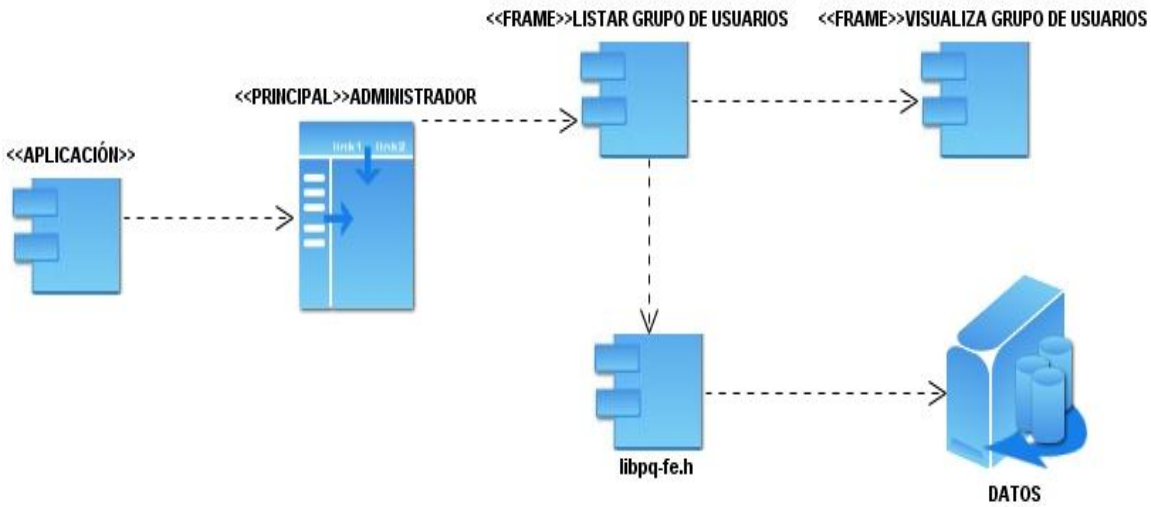



**Figura 4.6: Listar Grupo de Usuarios.**

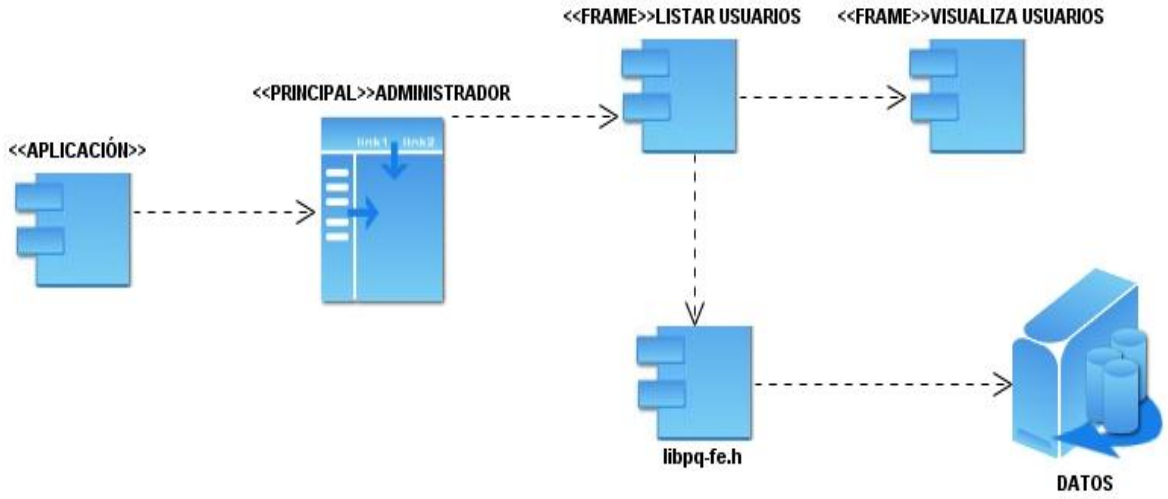



Figura 4.7: Listar Usuarios.

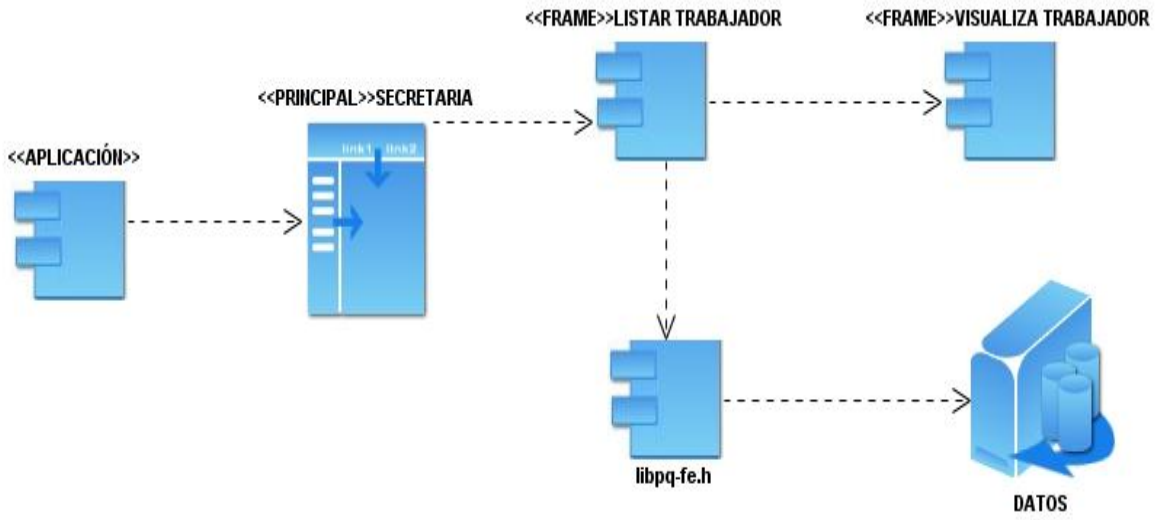



Figura 4.8: Listar Trabajadores.

# Anexo 5: Diagramas de Secuencia.

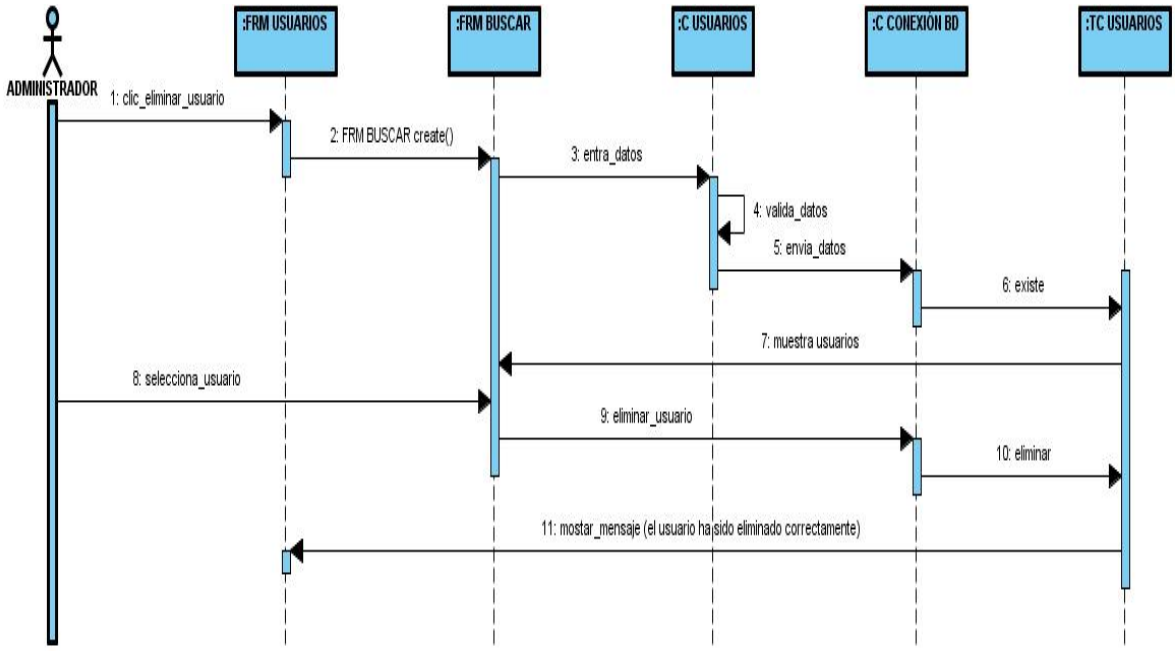



**Figura 5.1 Eliminar Usuario**

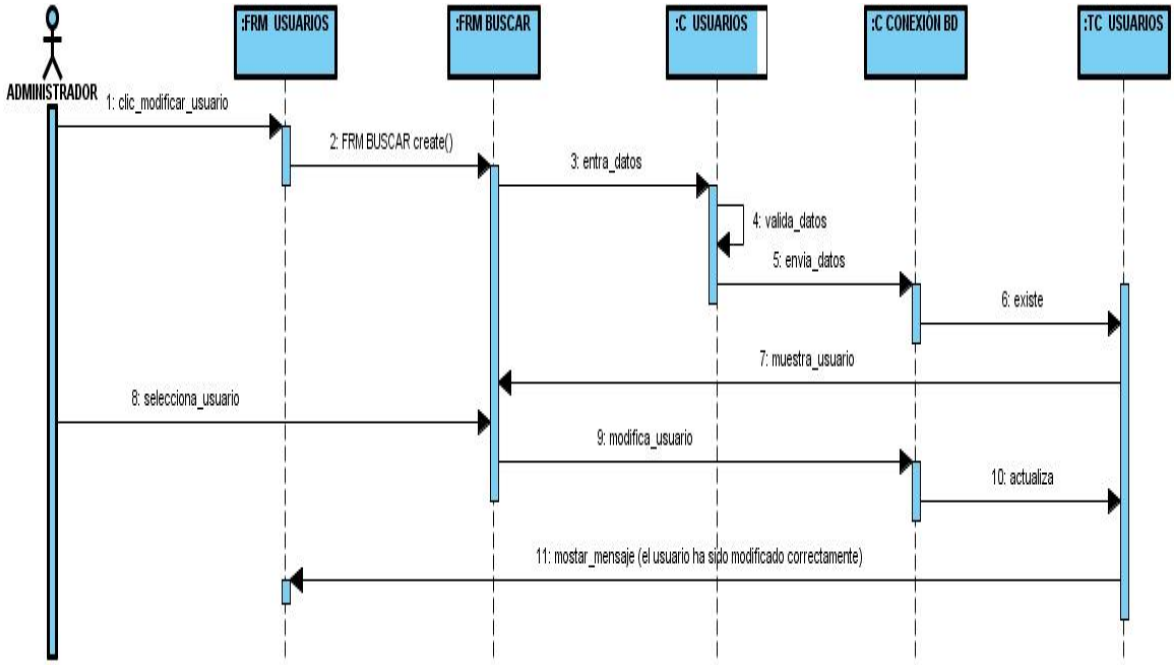



**Figura 5.2 Modificar Usuario**

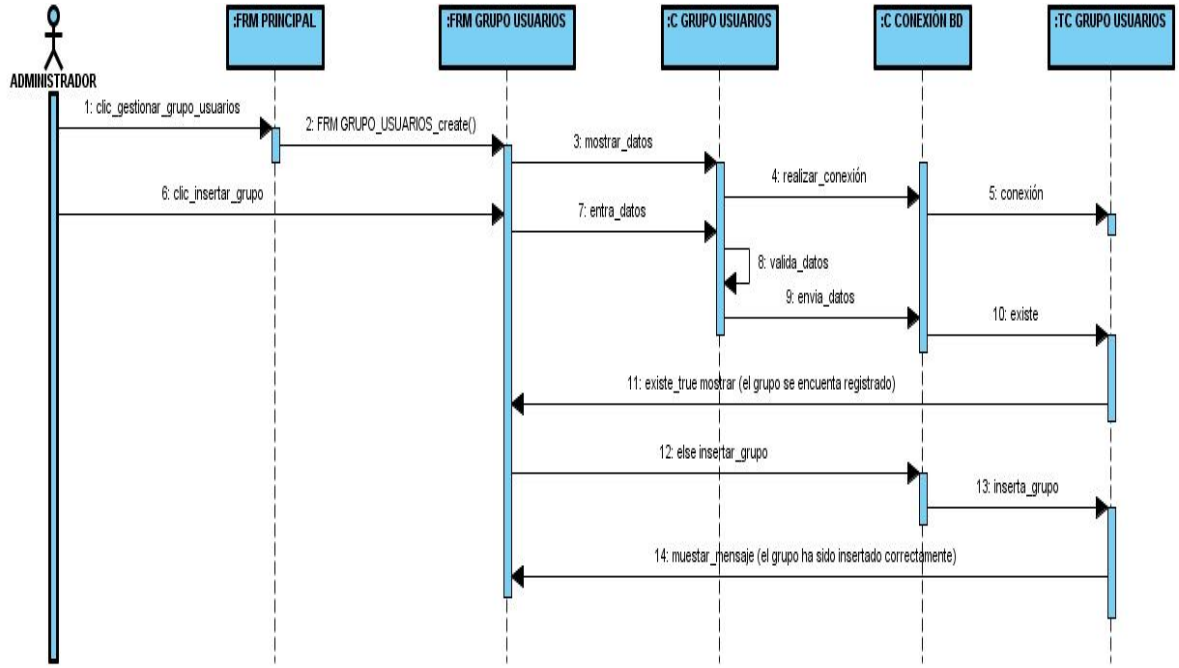



**Figura 5.3 Insertar Grupo de Usuarios**

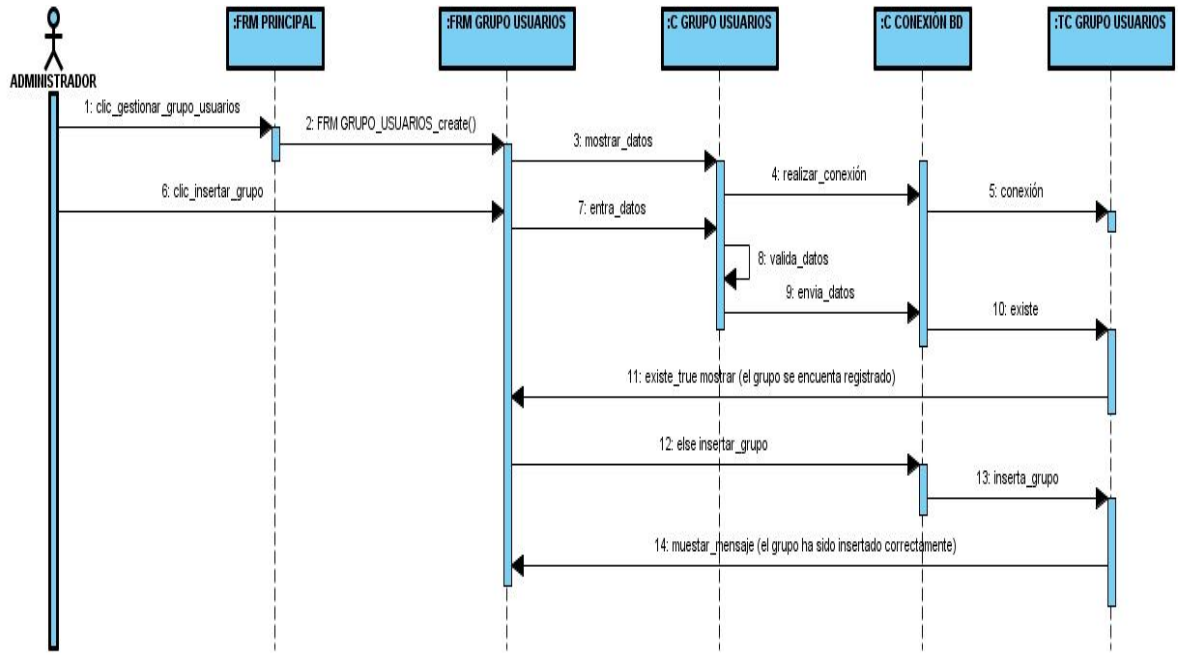



**Figura 5.4 Eliminar Grupo de Usuarios**

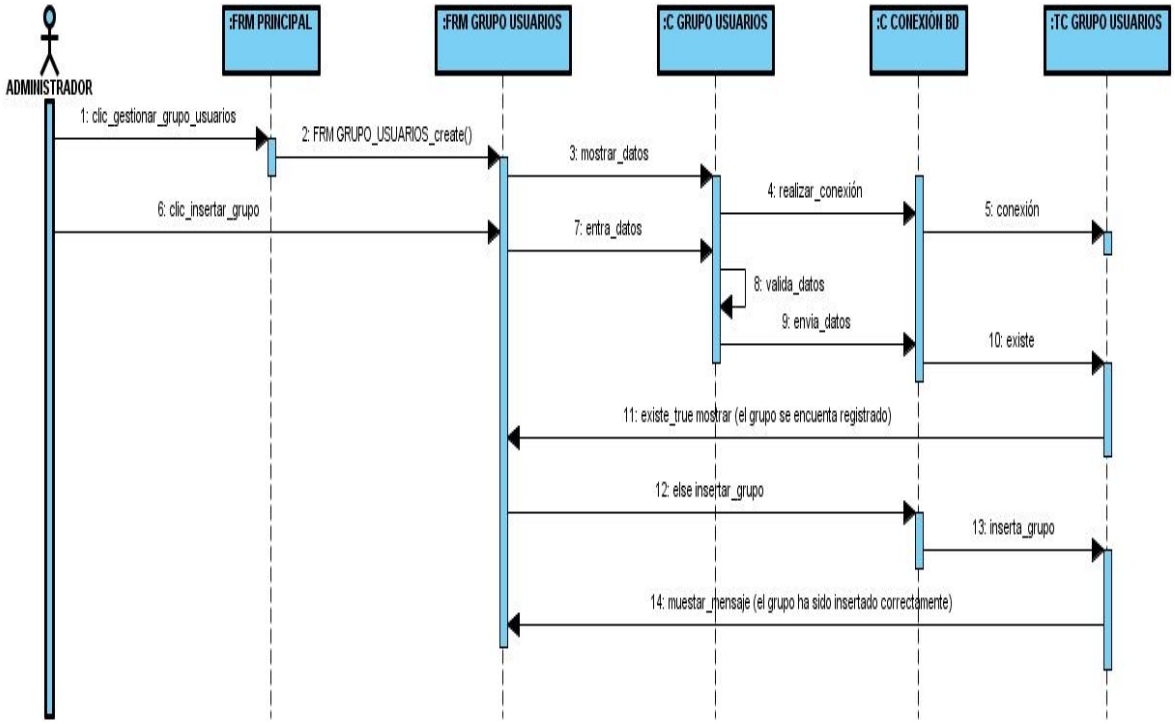



**Figura 5.5 Modificar Grupo de Usuarios**

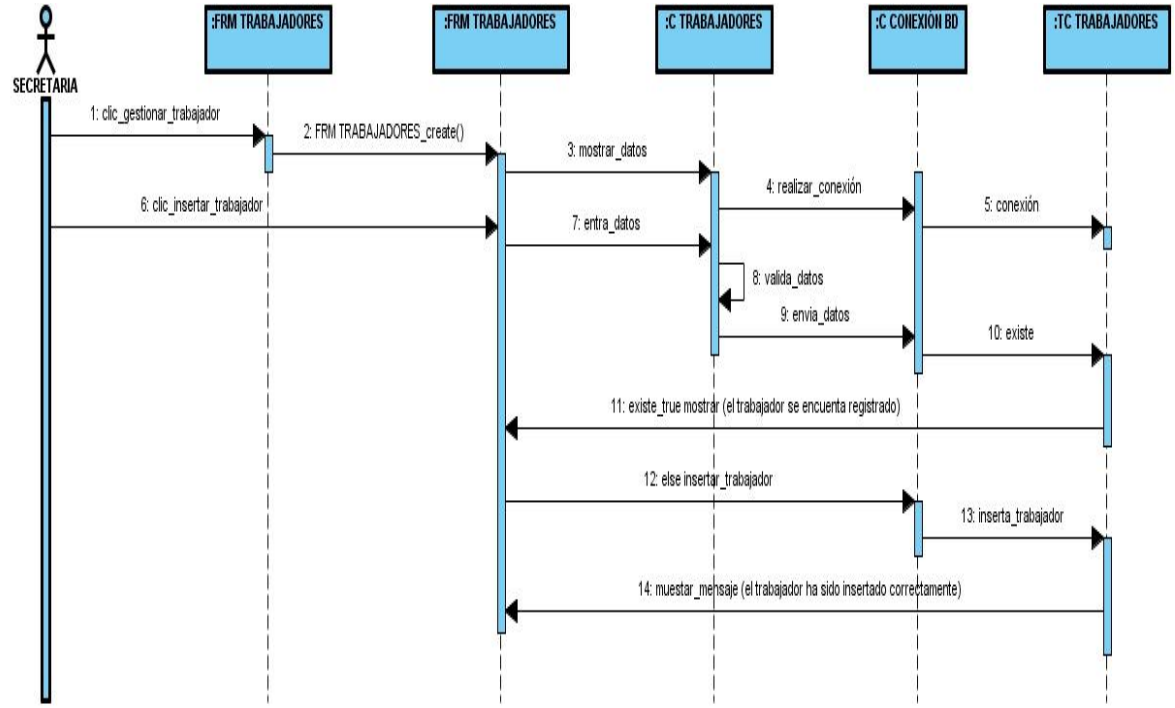



**Figura 5.6 Insertar Trabajador**

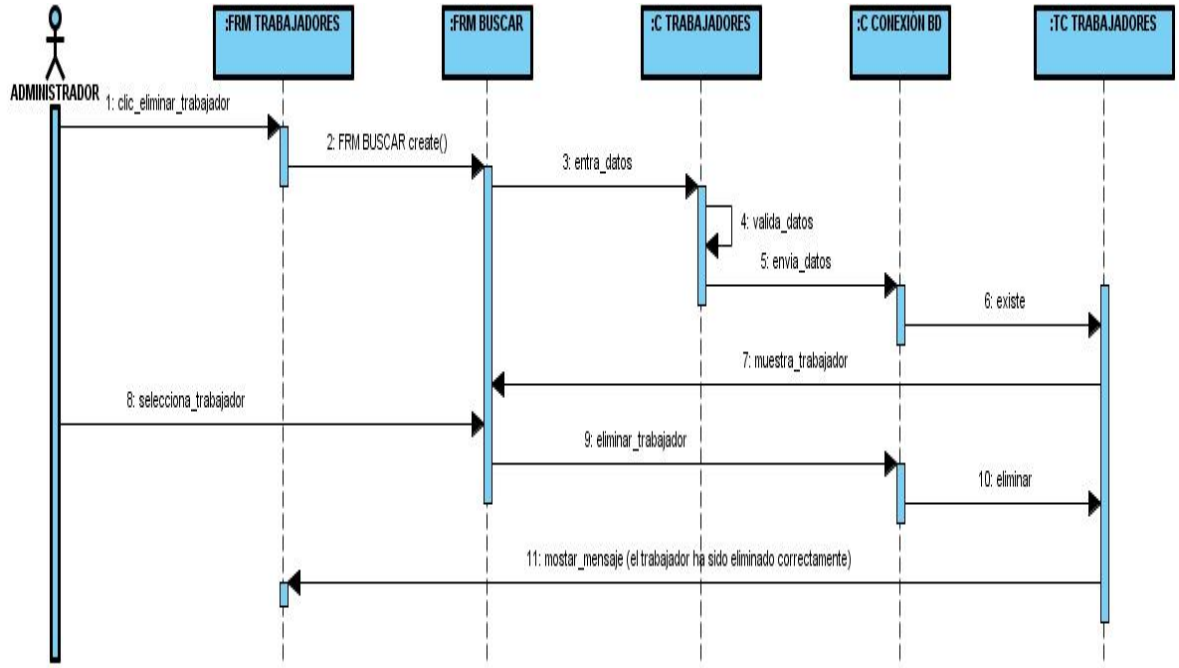



**Figura 5.7 Eliminar Trabajador**

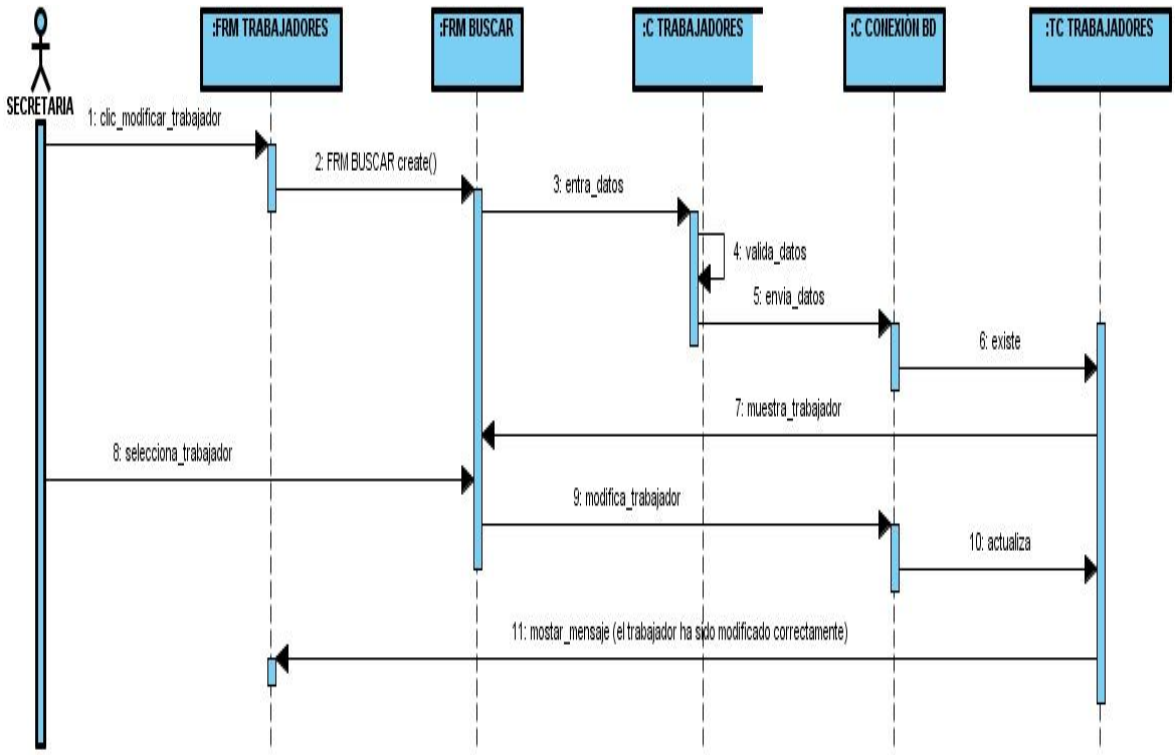



**Figura 5.8 Modificar Trabajador**

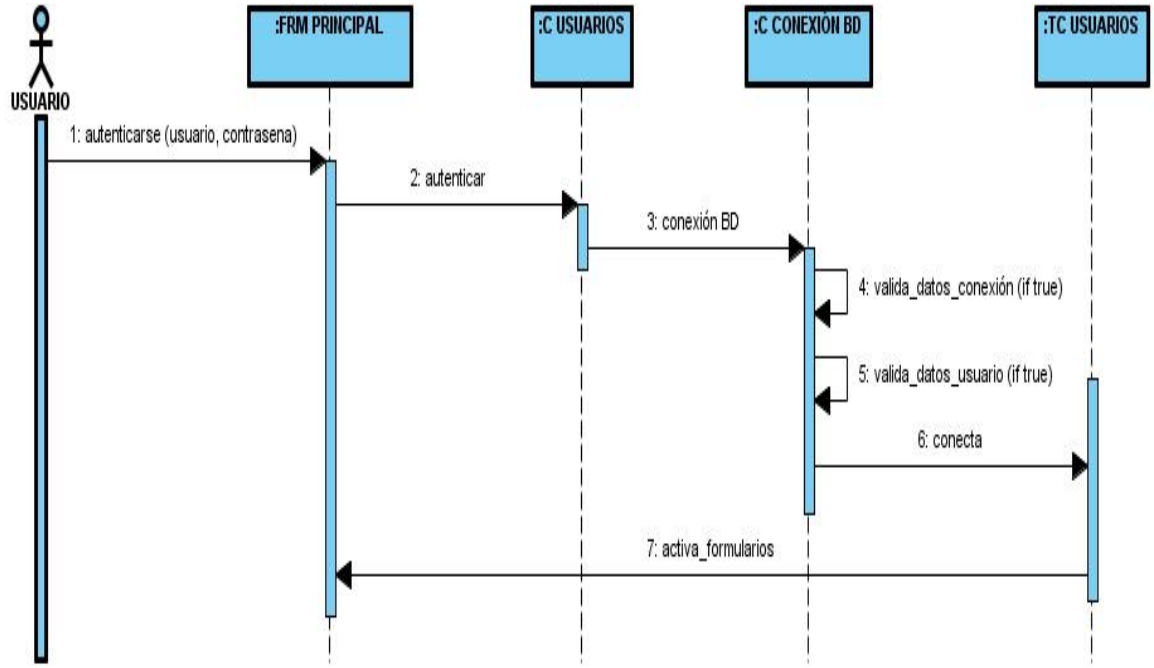



**Figura 5.9 Autenticar Usuarios**

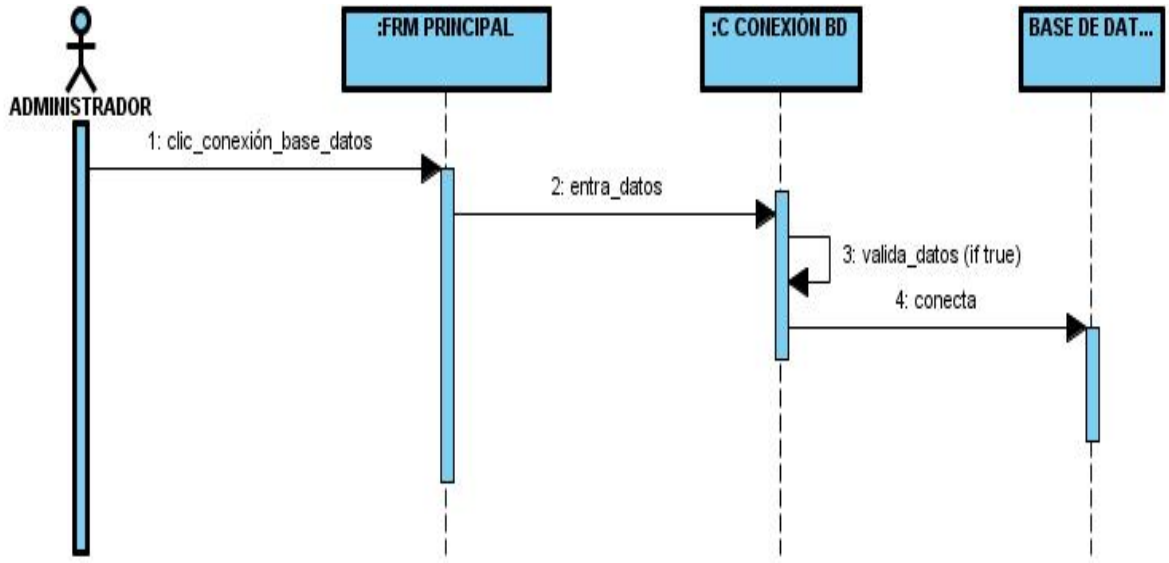



**Figura 5.11 Conexión Base de Datos**

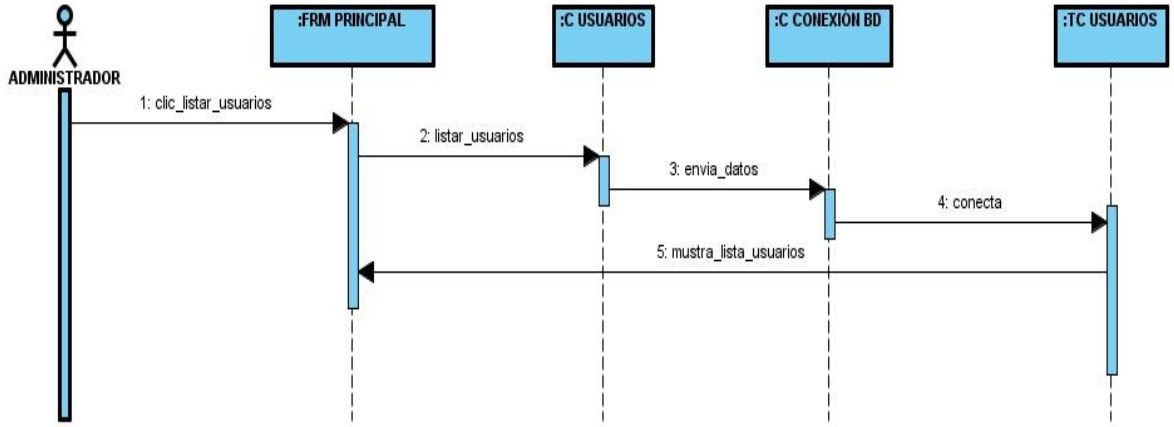



**Figura 5.12 Listar Usuarios**

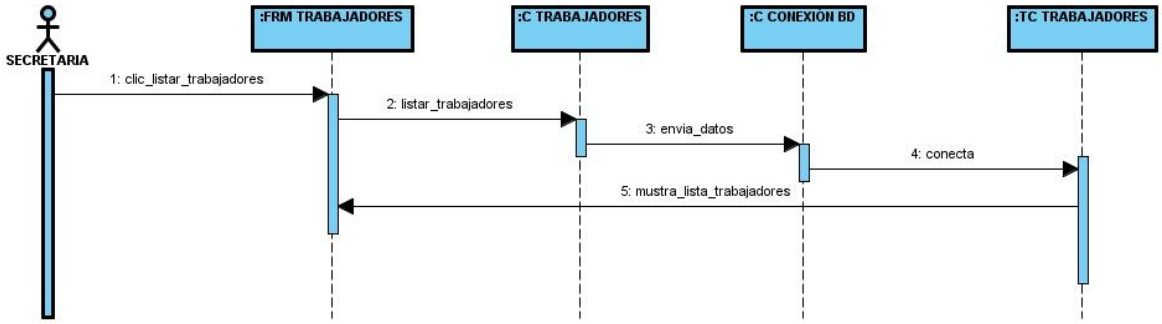



**Figura 5.13 Listar Grupo de Usuarios**

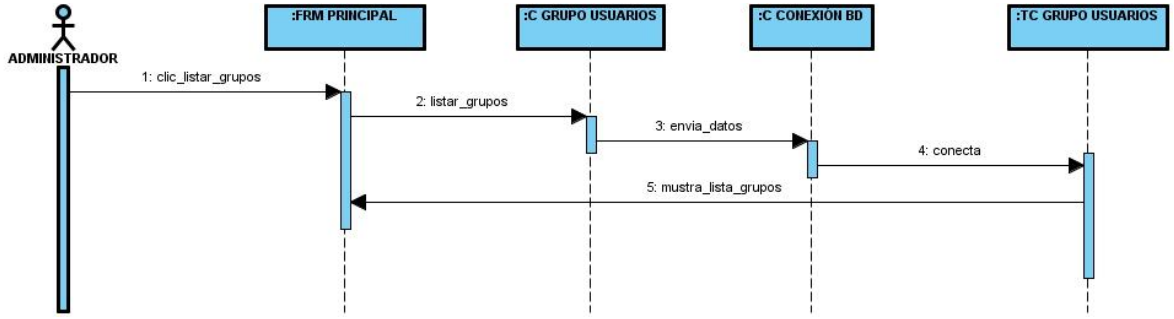



**Figura 5.14 Listar Trabajadores**

# Anexo 6: Descripción de Clases del Diseño.

**Tabla 6.1: Descripción de la Clase Grupo Usuario.**

| **Nombre de la clase:** | **Clase_Grupo_Usuarios** |
|---|---|
| **Tipo de clase:** | entidad |
| **Resumen:** | Esta clase permite realizar la gestión de los datos de los Grupos de Usuarios. |
| **Atributos** | **Tipo** |
| Id_Grupo_Usuarios | integer |
| Nombre_Grupo_Usuarios | varchar |
| **Responsabilidades** | |
| **Nombre.** | **Descripción.** |
| Insertar_Grupo_Usuarios () | Este método se encarga de insertar el Grupo de Usuario. |
| Eliminar_Grupo_Usuarios() | Este método se encarga de eliminar el Grupo de Usuario. |
| Modificar_Grupo_Usuarios() | Este método se encarga de modificar el Grupo de Usuario. |
| Buscar_Grupo_Usuarios() | Este método se encarga de buscar los grupos de usuarios para realizarle una determinada acción. |
| Listar_Grupo_Usuarios() | Permite mostrar una lista de los grupos de usuarios en la aplicación. |

**Tabla 6.2: Descripción de la Clase Trabajadores.**

| **Nombre de la clase:** | **Clase_Trabajadores** |
|---|---|
| **Tipo de clase:** | entidad |
| **Resumen:** | Esta clase permite realizar la gestión de los datos de los Trabajadores. |
| **Atributos** | **Tipo** |

| | |
|---|---|
| Id_Trabajador | integer |
| Nombre_Completo | varchar |
| Sexo | varchar |
| **Responsabilidades** | |
| **Nombre.** | **Descripción.** |
| Insertar_Trabajador () | Este método se encarga de insertar el Trabajador. |
| Eliminar_Trabajador () | Este método se encarga de eliminar un Trabajador. |
| Modificar_ Trabajador () | Este método se encarga de modificar el Trabajador. |
| Buscar_ Trabajador () | Este método se encarga de buscar los Trabajador para realizarle una determinada acción. |
| Listar_Trabajador | Permite mostrar una lista de trabajadores registrados en la aplicación. |