Universidad de las Ciencias Informáticas
Facultad 6

Título: "Sistema Informático de Gestión de Información de los Recursos Humanos y las Actividades de las Coordinaciones Regionales de Prevención del Delito."

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

Autores: José Antonio León Rosales.

Jesús Manuel Roblejo Rondón.

Tutores: Ing. Liusmila Nieto Cervantes.

Ing. Dairon Freddy Calderio Hechavarria.

Junio 2010.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

José Antonio León Rosales

_____
Firma del Autor

Jesús Manuel Roblejo Rondón

_____
Firma del Autor

Ing. Liusmila Nieto Cervantes

_____
Firma del Tutor

Ing. Dairon F. Calderio Hechavarria

_____
Firma del Tutor

DATOS DE CONTACTO

Tutores:

Ing. Liusmila Nieto Cervantes.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: lnieto@uci.cu

Ing. Dairon Freddy Calderio Hechavarria.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: dfcalderio@uci.cu

Cotutor:

Ing. René Vega Gorgoso.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: rvega@uci.cu

Consultantes:

Ing. Aidacelys López Díaz.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: alopezd@uci.cu

Ing. Andres Ballester Marsal.

Universidad de las Ciencias Informática, Habana, Cuba.

Email: aballester@uci.cu

AGRADECIMIENTOS

A mis padres, por su constante apoyo y comprensión durante todos estos años; por su ejemplo de superación incansable; por lo que hemos sido y seremos gracias a ustedes, son los mejores padres del mundo.

A mi novia que es lo mejor que me ha pasado en estos años y a sus padres.

A nuestros tutores Liusmila y Dairon, por su asesoramiento científico, por su disposición permanente e incondicional en aclarar nuestras dudas, por su paciencia, esmero y ayuda.

A todos aquellos que de una forma u otra han hecho posible el desarrollo de la tesis, a mis amigos, en especial a Yoel que es mi hermano, es más que un amigo. A nuestros compañeros del proyecto que siempre estuvimos juntos en la pelea. A mi compañero de tesis que viene conmigo desde primer año.

Un agradecimiento especial a nuestro Comandante Fidel y a la Revolución Cubana.

José Antonio.

Agradezco sobre todas las cosas a mi madre por haber luchado tanto para que yo estuviera donde estoy ahora. A mi hermana, mi hermano, mi papá, mi abuela, y a toda mi familia por apoyo y comprensión. A mis amigos de la universidad por todos los momentos malos y buenos que pasamos juntos, en especial a mi hermano Leduar.

A nuestros tutores Liusmila y Dairon, por su asesoramiento científico, por su disposición permanente e incondicional en aclarar nuestras dudas, por su paciencia, esmero y ayuda. A todos los que me ayudaron de una forma u otra al desarrollo de la tesis a Danoy, Francisco, Alejandro, Osmani, Wilmar, Eneisi y Asdrubal.

A mi compañero de tesis Tony por su ayuda y su desempeño en el desarrollo de la tesis. Agradecer especialmente a Fidel y a la Revolución Cubana por lograr que mis sueños se convirtieran en realidad.

Jesús Manuel.

DEDICATORIA

Esta tesis se la dedico a mis padres ya que son los mejores padres del mundo, ellos han luchado muy fuerte para que este sueño se cumpla. A mi hermana que es la única persona que tiene mi misma sangre. También se la dedico a dos personas que ya no están conmigo, lamentablemente, pero que me quisieron mucho, a mi abuelo José Rosales "Pepe" y a mi tía María Caridad León. Abuelito esta tesis es tuya, se que soñaste mucho con este día, donde te encuentres tienes que saber que te quise mucho y nunca te olvidaré. Se la dedico a mi novia que ha sido más que una novia, ha sido mi todo desde que la conocí. Se la dedico a toda mi familia, a mis abuelas, a mis tías, en fin a toda mi familia que se que me quiere mucho.

José Antonio.

A mi madre, mi hermana Ibet, y a toda mi familia.

Jesús Manuel

RESUMEN

La presente investigación surge en el marco de trabajo del proyecto: “Prevención del Delito de la República Bolivariana de Venezuela”, aprobado en la Novena Mixta Cuba-Venezuela del convenio de colaboración de ambos países, y de la necesidad de un sistema informático que gestione la información referente a los recursos humanos y las actividades dirigidas a la prevención de la violencia y la criminalidad de las Coordinaciones Regionales.

En el desarrollo de la investigación se describió y argumentó la metodología de desarrollo a utilizar así como las herramientas y tecnologías que fueron empleadas durante la realización del sistema informático; definiendo a Rational Unified Procces como metodología a utilizar y Java como lenguaje de programación, apoyado en el Eclipse como Entorno de Desarrollo Integrado. El resultado final de la presente investigación fue una aplicación web capaz de gestionar la información referente a los recursos humanos y las actividades dirigidas a la prevención de la violencia y la criminalidad de las Coordinaciones Regionales de Prevención del Delito.

PALABRAS CLAVE: actividades, coordinación regional, prevención del delito, recursos humanos.

TABLA DE CONTENIDOS

INTRODUCCIÓN

El 15 de febrero de 1999, la historia de Venezuela cambió para siempre. Con el triunfo del Comandante Hugo Chávez, se comienza a tener en cuenta con orden priorizado una extensa agenda social, que incluía sectores como la salud, la educación, el empleo, las viviendas, la seguridad entre otros. Se comienzan a favorecer a los sectores más pobres y excluidos, pero para lograr todas estas metas sociales, era necesario tener una paz y una estabilidad nacional por lo que se hace necesario la disminución del índice delictivo existente hasta entonces en el país.

La situación económica, política y social acumulada en Venezuela en la década de los 90 generó un alto índice de desempleo y pobreza, produciéndose un incremento de la actividad delictiva, aumentando las necesidades de seguridad del colectivo y creando un ambiente de frustración y desconfianza hacia los organismos centrales del Estado. La población venezolana se ha visto afectada de manera significativa por el alto índice de delitos, problema que tiende a agudizarse de allí la necesidad de vislumbrar alternativas de solución.

En tal sentido el Ministerio del Poder Popular para Relaciones Interiores y Justicia de la República Bolivariana de Venezuela (MPPRIJ), atendiendo a su misión institucional de garantizar la seguridad ciudadana, mediante la formulación de políticas dirigidas al resguardo de la paz pública, desarrollo territorial equilibrado y la estabilidad de la nación y con la finalidad de reservar y fortalecer la democracia participativa y la nueva institucionalidad, promueve la formulación y puesta en marcha del proyecto “Solución Tecnológica Integral para el Perfeccionamiento del Sistema de Prevención del Delito de la República Bolivariana de Venezuela”, partiendo de la premisa de que la seguridad ciudadana es una condición necesaria para el desarrollo humano.

La Dirección General de Prevención del Delito (DGPD) está adscrita al Viceministerio de Seguridad Ciudadana perteneciente al Ministerio del Poder Popular para Relaciones Interiores y Justicia de la República Bolivariana de Venezuela. (Ver anexo 1, figura 1). Esta posee 22 Coordinaciones Regionales; 18 adscritas al MPPRIJ, ubicadas en los Estados: Anzoátegui, Aragua, Barinas, Bolívar, Carabobo, Cojedes, Falcón, Guárico, Lara, Mérida, Miranda, Monagas, Portuguesa, Táchira, Trujillo, Yaracuy, Zulia y en el Distrito Capital; además de cuatro que se han creado de forma descentralizada en los estados Delta Amacuro, Nueva Esparta, Sucre y Vargas. Estas Coordinaciones Regionales ejecutan los Programas de la Dirección General, además de apoyar los planes de los Ejecutivos Regionales. (Ver anexo 1, figura 2).

Introducción

Entre las funciones de la Dirección General de Prevención del Delito y de sus Coordinaciones Regionales se encuentra coordinar las actividades dirigidas a la prevención de la violencia y la criminalidad a través de instituciones públicas y privadas, además de elaborar el informe anual de sus actividades para el Despacho del Ministro del Interior y Justicia, así como también gestionar toda la información referente a sus recursos humanos. Actualmente en las Coordinaciones Regionales no existe una correcta estandarización y gestión de las informaciones referentes a sus recursos humanos y las actividades dirigidas a la prevención de la violencia y la criminalidad. Debido al cúmulo de información que dichas Coordinaciones registran y al volumen de documentación que se envía a la Dirección General, se ve afectada la retroalimentación desde las Coordinaciones Regionales hacia la Dirección General. Estas deficiencias conllevan a que exista pérdida de información y gastos innecesarios de recursos materiales. A continuación se mencionan algunas informaciones de las Coordinaciones Regionales que se ven afectada su gestión e intercambio:

- La información correspondiente a los datos personales y laborales de los trabajadores de las Coordinaciones Regionales de Prevención del Delito.
- Los documentos asociados a la asistencia diaria de los trabajadores e información que se envía a la Dirección General mensualmente, asociada a las horas extras de los mismos.
- La información correspondiente a las actividades dirigidas a la prevención de la violencia y la criminalidad que son realizadas por la Coordinación Regional. Además de los reportes que se envían de las mismas a la dirección general.

Por todo lo mencionado se plantea el siguiente **problema a resolver**: ¿Cómo contribuir al proceso de gestión de la información referente a los recursos humanos y las actividades dirigidas a la prevención de la violencia y la criminalidad de las Coordinaciones Regionales de Prevención del Delito?

El problema planteado se enmarca en el siguiente **objeto de estudio:** Proceso de Gestión de información de las Coordinaciones Regionales de la República Bolivariana de Venezuela pertenecientes a la Dirección General de Prevención del Delito.

El objeto de estudio delimita el **campo de acción:** Proceso de Gestión de información asociada a los recursos humanos y las actividades dirigidas a la prevención de la violencia y la criminalidad de las Coordinaciones Regionales de Prevención del Delito.

Para dar solución al problema planteado el desarrollo de la presente investigación tiene como **objetivo general:** Desarrollar un sistema informático que gestione la información referente a los recursos humanos y las actividades dirigidas a la prevención de la violencia y la criminalidad de las Coordinaciones Regionales de Prevención del Delito.

A partir del objetivo general se trazaron los siguientes **objetivos específicos:**

- Identificar las funcionalidades que debe cumplir el sistema informático.
- Diseñar las clases del sistema informático.
- Implementar el sistema informático.

Para lograr los objetivos específicos, se realizaran las siguientes **tareas de la investigación:**

1. Investigación de las herramientas y tecnologías existentes de código abierto para el desarrollo del sistema informático.
2. Definición de los requisitos funcionales y no funcionales del sistema informático.
3. Realización de los diagramas de clases y secuencias del diseño del sistema informático.
4. Diseño de la base de datos del sistema informático.
5. Realización de los diagramas de componentes del sistema informático.
6. Implementación de los componentes del sistema informático.
7. Realización de pruebas a nivel de desarrollador del sistema informático.

El presente trabajo de diploma consta de 4 capítulos, a continuación se hace una breve descripción de cada uno de ellos:

Capítulo 1: Fundamentación Teórica.

En este capítulo se realiza un estudio y análisis de los Sistemas de Gestión de Información. Además, incluye un estudio de las herramientas a utilizar, la metodología de desarrollo y las tecnologías en las que se apoya el desarrollo del Sistema Informático en función de un análisis de las tendencias actuales.

Capítulo 2: Características del Sistema.

En este capítulo se presenta la propuesta del sistema, se define el Modelo de Dominio, los requisitos funcionales y no funcionales. También se definen los casos de uso y las descripciones textuales de los mismos. Por último, se realiza el diagrama de Casos de Uso del Sistema.

Capítulo 3: Diseño del Sistema.

En este capítulo se traducen los requerimientos del sistema a una especificación que describe cómo implementar el sistema a través del diseño de clases, teniendo en cuenta los requisitos funcionales y no funcionales, es decir, se realizan los diagramas de clases de diseño y los diagramas de secuencias. También se realiza el diseño de la base de datos. Además, se fundamenta la arquitectura utilizada y los principales patrones de diseño a utilizar.

Capítulo 4: Implementación del Sistema.

En este capítulo se describe como fue desarrollado el sistema en términos de componentes, es decir, se lleva a cabo la realización de los diagramas de componentes, y la vista de implementación. Se muestran algunas pantallas del sistema. Además, se realizan pruebas a nivel de desarrollador para verificar sus funcionalidades.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA.

En este capítulo se describen algunos conceptos importantes para comprender que es un sistema de gestión de información computarizado. Se argumenta la metodología de desarrollo utilizada. Además, se especifican las herramientas y tecnologías en las cuales se va a apoyar el desarrollo del sistema informático.

1.1 Sistema de Gestión de Información Computarizado.

1.1.1 Gestión de la Información.

La gestión de la información es el proceso de analizar, utilizar, recuperar y almacenar la información que se ha obtenido y registrado, para permitir a los administradores tomar decisiones documentadas. [1]

1.1.2 Sistemas de Gestión de Información.

Es la gestión de todos los recursos de información que se generan dentro y fuera de la organización orientados a la obtención de ventajas competitivas y a proveer a la organización de servicios de calidad, mediante el uso de tecnologías, en función de satisfacer las necesidades de información de sus clientes y alcanzar el desarrollo estratégico y las metas de la organización. [2]

A partir de los temas antes mencionado se puede hacer alusión al concepto de Sistemas de Gestión de Información Computarizados, que no son más que sistemas con un soporte informático, es decir, se desarrollan en un entorno usuario-computadora, utilizando hardware y software computacional, redes de telecomunicaciones, técnicas de administración de base de datos computarizadas y otras formas de tecnología de información. [3]

1.2 Metodología de desarrollo de software.

Una metodología de desarrollo de software es un conjunto de procedimientos, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. [4]

Cuando se utiliza una metodología eficaz para el desarrollo de software se tiene una gran probabilidad de que el software presente una alta calidad, que se desarrolle en el tiempo planificado y con los costes establecidos. El uso de una metodología hace trabajar de forma organizada, controlando y documentando todo lo relacionado con el proyecto.

Rational Unified Process.

El proceso Unificado es un Proceso de Desarrollo de Software. Un Proceso de Desarrollo de Software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. Sin embargo, el Proceso Unificado es algo más que un simple proceso, en un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes aéreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. [5] (Ver Anexo 1, figura 3).

Los verdaderos aspectos definitorios del proceso Unificado se resumen en tres características principales: Dirigidos por Casos de Uso, Centrado en la Arquitectura e Iterativo e Incremental.

Los casos de uso son una herramienta para especificar los requisitos de un sistema, también guían su diseño, su implementación y prueba, es decir, guían el proceso de desarrollo basándose en el modelo de casos de uso. Los casos de uso se especifican, se diseñan y son la fuente a partir de la cual los ingenieros de pruebas construyen sus casos de pruebas.

La arquitectura surge de las necesidades de la empresa, como las perciben los usuarios y los inversores, y se refleja en los casos de uso. También se ve influida por varios factores, como la plataforma en la que tiene que funcionar el software (Arquitectura Hardware, Sistema Operativo, Sistema de gestión de base de datos, protocolos para comunicaciones de red), los bloques de construcción reutilizables de que se dispone por ejemplo un marco de trabajo. La arquitectura es una vista del diseño completo con las características más importantes resaltadas. Cada producto tiene tanto una función como una forma, ninguna es suficiente por sí misma. En esta situación la función corresponde a los casos de uso y la forma a la arquitectura, debe haber interacción entre ambos, es decir, los casos de uso deben encajar en la arquitectura.

El desarrollo de un producto software comercial supone un gran esfuerzo que puede durar entre varios meses hasta posiblemente un año o más. Es práctico dividir el trabajo en partes más pequeñas o mini proyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. Para una efectividad máxima, las iteraciones deben estar controladas, deben seleccionarse y ejecutarse de una forma planificada.

El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema, cada ciclo concluye con una versión del producto para los clientes. Cada ciclo se desarrolla a lo largo del

tiempo, este tiempo se divide a la vez en cuatro fases (Inicio, Elaboración, Construcción y Transición) y nueve flujos de trabajos. Los seis primeros son conocidos como flujos de ingeniería (Modelamiento del Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba y Despliegue) y los tres últimos como de apoyo (Gestión de Configuración y Cambio, Administración de Proyecto y Ambiente). Cada fase termina con un hito. Inicio [Objetivos (Visión)], Elaboración [Arquitectura], Construcción [Capacidad Operacional Inicial], Transición [Release del producto].

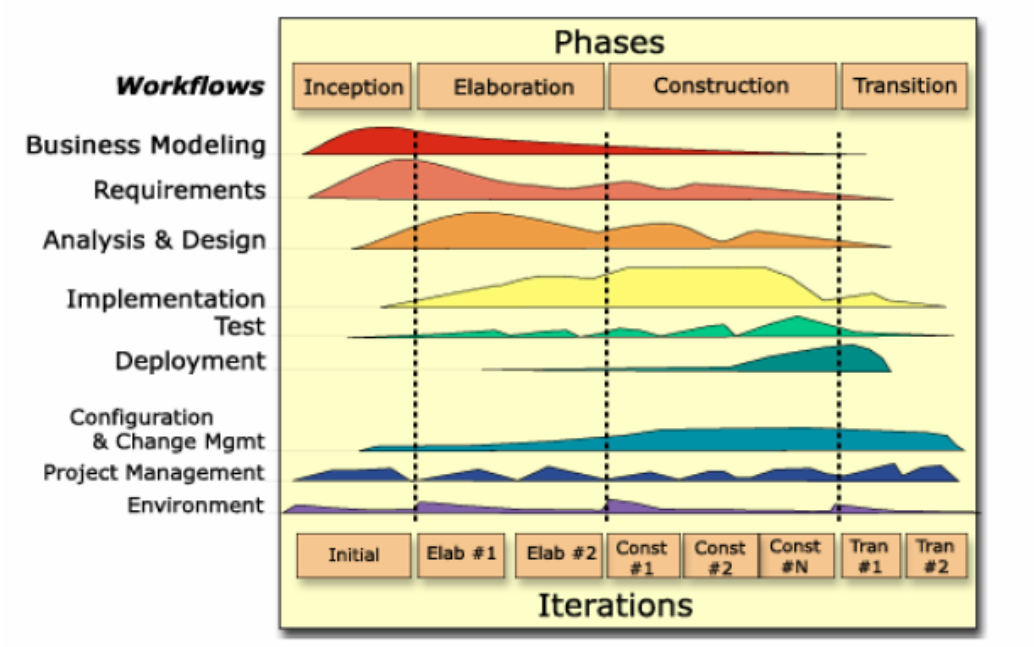


Figura 4. RUP

El Proceso Unificado utiliza el Lenguaje Unificado de Modelado (Unified Modeling Language, UML) para modelar los artefactos de un sistema de software. De hecho, UML es una parte esencial del proceso Unificado sus desarrollos fueron paralelos.

Se selecciona RUP como metodología de desarrollo, por su adaptabilidad al proyecto y las condiciones de desarrollo que garantiza el cumplimiento de las metas trazadas. Además, por su buena concepción en cuanto a la organización del trabajo. Propone además, la generación de los artefactos necesarios para una buena documentación y el entendimiento entre los miembros del equipo.

1.3 Lenguaje de Modelado.

UML

El “Lenguaje Unificado de Modelado” (UML) se presentó oficialmente cuando Rumbaugh, Booch y Jacobson unificaron sus estudios con una semántica y notación, para lograr compatibilidad en el análisis y diseño orientado a objetos, permitiendo que los proyectos se asentaran en un lenguaje de modelado maduro, permitiendo a los constructores de herramientas enfocarse en producir características más útiles. Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. [6]

UML presenta una serie de características muy favorables las cuales se mencionan a continuación:

1. Permite modelar sistemas utilizando técnicas orientadas a objetos (OO).
2. Permite especificar todas las decisiones de análisis y diseño, construyéndose así modelos precisos, no ambiguos y completos.
3. Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, entre otros).
4. Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.
5. A pesar de tener gran expresividad esta notación es fácil de aprender.
6. UML es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura e iterativo e incremental.

1.4 Roles y Artefactos.

Un rol es una definición abstracta de un conjunto de actividades realizadas y de artefactos obtenidos. Los roles son realizados típicamente por un individuo, o un conjunto de individuos, trabajando juntos en equipo. Un miembro del equipo de proyecto cumple normalmente muchos roles. Los roles no son individuos; en lugar de ello, describen cómo los individuos se comportan en el negocio y qué responsabilidades tienen estos individuos. [7]

La metodología de desarrollo RUP, propone una serie de roles, pero por las necesidades del proyecto se desempeñaron los siguientes: Analista, Diseñador, Diseñador de base de datos, Arquitecto de software e Implementador.

Analista: agrupa los roles que están involucrados en la gestión de requisitos del sistema, que pueden estar representados por una o varias personas, entre los que se encuentran: Analista de Procesos de Negocio, Diseñador de Negocio, Analista del Sistema y especificador de Requerimientos. Los artefactos que el Analista construye se mencionan a continuación:

- **Modelo de Dominio**: captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que trabaja el sistema.
- **Modelo de casos de uso**: es un modelo del sistema que contiene actores, casos de usos y sus relaciones.
- **Especificación de requerimientos del software**: es la captura de los requerimientos del software para el sistema o una parte de este.
- **Glosario**: es un documento que define los términos comunes que se utilizan para describir el proyecto.

Diseñador: es el responsable de una parte del sistema que incluye: las restricciones de los requisitos, la arquitectura y el proceso de desarrollo del proyecto. Los artefactos a realizar desempeñando el rol de diseñador se describen a continuación:

- **Clase del diseño**: es una descripción de un grupo de objetos que comparten las mismas responsabilidades, relaciones, operaciones, atributos y semántica.
- **Realización de caso de uso**: describe como un caso de uso específico es realizado dentro del modelo de diseño en término de colaboración de objetos.

Diseñador de base de datos: es el responsable de la definición de los detalles del diseño de la base de datos incluyendo: tablas, índices, vistas, restricciones, disparadores, y procedimientos almacenados. Dicho rol tiene que asegurarse de que los datos persistentes son almacenados consistente y eficientemente y definir el comportamiento que debe ser implementado en la base de datos. El artefacto a realizar desempeñando el rol de diseñador de base de datos se describe a continuación:

- **Modelo de datos:** el modelo de datos describe la representación lógica y física de los datos persistentes utilizados por la aplicación.

Arquitecto de Software: el Arquitecto de Software es el responsable de la arquitectura de software, que incluye las decisiones técnicas claves que restringen el diseño global y la implementación para el proyecto. A continuación se describen los artefactos que se desarrollan por el arquitecto de software:

- **Modelo de diseño**: es un modelo de objeto que describe la realización de casos de uso, y sirve como una abstracción del modelo de implementación y el código fuente.
- **Modelo de despliegue**: un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos).
- **Modelo de implementación**: representa la composición física de la implementación en términos de subsistemas de implementación, y elementos de implementación (directorios y archivos, incluyendo código fuente, datos y archivos ejecutables). Específicamente de este modelo se realizará el diagrama de componentes (usado para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación).

Implementador: es el responsable del desarrollo y prueba de los componentes, en acuerdo con las normas adoptadas en el proyecto para la integración de subsistemas más grandes. Los artefactos a realizar desempeñando el rol de implementador se describen a continuación:

- **Elemento de implementación**: son los componentes físicos que conforman una implementación, que incluyen archivos y directorios. Incluyen los archivos de código de software (origen, binario o ejecutable), los archivos de datos y los archivos de documentación, como los archivos de ayuda en línea.
- **Artefactos de instalación**: se refiere al software y las instrucciones documentadas requeridas para instalar el producto.

1.5 Herramienta para el modelado.

El modelado de sistemas de software es una técnica para tratar con la complejidad inherente a estos sistemas. El uso de modelos ayuda al ingeniero de software a "visualizar" el sistema a construir. Además,

los modelos de un nivel de abstracción mayor pueden utilizarse para la comunicación con el cliente. Por último, las herramientas de modelado pueden ayudar a verificar la corrección del modelo.

Algunas de las herramientas CASE conocidas son el ArgoUML, Rational Rose, Visual Paradigm, Easy CASE, Xcase, CASE Studio 2, CASEWise entre otras. Dentro de las más utilizadas se encuentran el Rational Rose y el Visual Paradigm.

Visual Paradigm.

Visual Paradigm para UML es una herramienta CASE de modelado profesional que soporta el ciclo de vida completo del desarrollo de software. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación en formatos web y pdf. Esta herramienta brinda un ambiente amigable a los analistas del sistema y desarrolladores para analizar, diseñar y mantener aplicaciones de software. Se integra con las siguientes herramientas Java: Eclipse/IBM WebSphere, JBuilder, NetBeans IDE, Oracle JDeveloper, BEA Weblogic'k. Permite realizar ingeniería tanto directa como inversa, también es una herramienta colaborativa, es decir permite a múltiples usuarios trabajar sobre el mismo proyecto. [8]

Teniendo en cuenta las características antes mencionadas del Visual Paradigm, especialmente su buena integración con los IDE para el desarrollo de aplicaciones en java, y la ventaja de que presenta una interfaz de usuario de fácil uso y que permite realizar los diagramas y artefactos que se generan durante el desarrollo del software además de realizar un control de las versiones durante todo el ciclo de trabajo, el colectivo de autores decide utilizarla como herramienta de modelado.

1.6 Lenguaje de programación.

Un lenguaje de programación es un conjunto de instrucciones, órdenes, comandos y reglas que permite la creación de programas. Los ordenadores entienden señales eléctricas (valores 0 ó 1). Los lenguajes permiten al programador indicar lo que debe hacer un programa, sin tener que escribir largas cadenas de ceros y unos, sino palabras (instrucciones) más comprensibles por las personas.

Java.

Java es un lenguaje orientado a objeto, fue desarrollado por la compañía Sun Microsystems, con la idea original de usarlo para la creación de páginas web. [9]

Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente de la plataforma. Es decir, si se realiza un programa en Java podrá funcionar en cualquier ordenador del mercado. Es una ventaja significativa para los desarrolladores de software, pues antes tenían que hacer un programa para cada sistema operativo, por ejemplo Windows, Linux, Macintosh, entre otros. Esto lo consigue porque se ha creado una Máquina Virtual de Java para cada sistema que hace de puente entre el sistema operativo y el programa de Java.

Se ha decidido usar este lenguaje para la realización de la aplicación por las ventajas de ser un lenguaje multiplataforma y un software libre, razón por la que aparecen gran cantidad de aplicaciones realizadas en dicho lenguaje actualmente que van desde móviles, la web y aplicaciones desktop.

1.7 Plataforma de Desarrollo.

J2EE.

J2EE es una plataforma de programación, parte de la Plataforma Java para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de N niveles distribuidos, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

La plataforma J2EE añade a Java la funcionalidad necesaria para convertirse en un lenguaje orientado al desarrollo de servicios en Internet. Mediante JSP y Servlets se pueden desarrollar sitios web bajo la tecnología Java. [10]

1.8 Frameworks.

Framework

Un framework es una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta. [11]

Las principales ventajas de la utilización de un framework son:

- El desarrollo rápido de aplicaciones. Los componentes incluidos en un framework constituyen una capa que libera al programador de la escritura de código de bajo nivel.

- La reutilización de componentes software al por mayor. Los frameworks son los paradigmas de la reutilización.
- El uso y la programación de componentes que siguen una política de diseño uniforme. Un framework orientado a objetos logra que los componentes sean clases que pertenezcan a una gran jerarquía de clases, lo que resulta en bibliotecas más fáciles de aprender a usar. [11]

1.8.1 Framework de Presentación.

Dojo.

Dojo Toolkit es un framework Javascript, su creador y presidente fue Alex Russell. Es un framework Open Source de Javascript que ayuda a construir aplicaciones de internet. Dojo es licencia dual como AFL y BSD, puedes usarlo en proyectos (L) GPL. A continuación se mencionan algunas de las características más significativas: [12]

- Contiene APIS y Widgets.
- Es potente, portable, ligero.
- Tiene herramientas de testeo para construir interfaces dinámicas.
- Permite crear widgets rápidamente, transiciones animadas, peticiones Ajax con la mejor potencia y facilidad.
- Soporta bookmarking.
- Encapsula las molestias “crossbrowser”.

Para el desarrollo de la aplicación informática se utilizó la versión 1.3.

1.8.2 Framework de Aplicación.

Spring.

Spring es un framework de aplicación desarrollado por la compañía Interface 21, para aplicaciones escritas en el lenguaje de programación Java. Fue creado gracias a la colaboración de grandes programadores, entre ellos se encuentran como principales y líderes de este proyecto Rod Johnson y Jurger Holler. Estos dos desarrolladores, además de otros colaboradores que juntando toda su experiencia en el desarrollo de aplicaciones J2EE (Java 2 Enterprise Editions), incluyendo EJB (Enterprise JavaBeans), Servlets y JSP (Java Server Pages), lograron combinar dichas herramientas y otras más en

un solo paquete, para brindar una estructura más sólida y un mejor soporte para este tipo de aplicaciones. [13]

Este framework está adquiriendo un gran auge y una gran popularidad, una de las características que ayuda a este éxito, es que es una aplicación Open Source, por lo tanto, da la libertad a muchas empresas y desarrolladores a incursionar en la utilización de esta aplicación. Para el desarrollo de la aplicación informática se utilizó la versión 2.5.6.

Arquitectura de Spring.

Spring es un framework modular que cuenta con una arquitectura dividida en siete capas o módulos, lo cual permite tomar y ocupar únicamente las partes que interesen para el proyecto y juntarlas con gran libertad. [13]

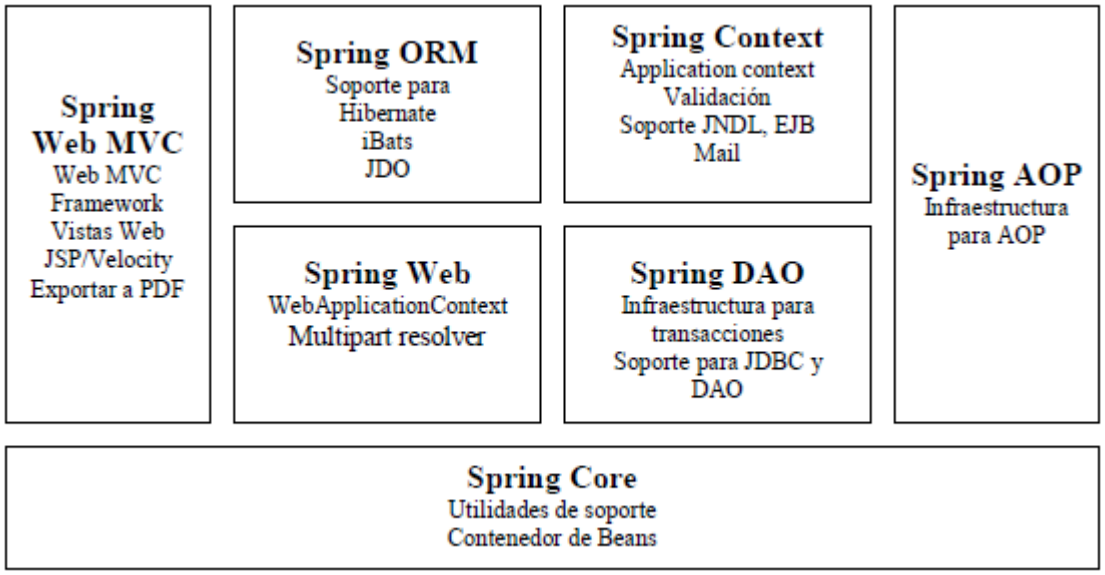


Figura 5. Arquitectura de Spring.

A continuación se describen los módulos de Spring**.**

Spring Core.

Esta parte es la que provee la funcionalidad esencial del framework está compuesta por el BenFactory, el cual utiliza el patrón de Inversión de Control (Inversion of Control) y configura los objetos a través de

Inyección de Dependencia (Dependency Injection). El núcleo de Spring es el paquete org.springframework.beans el cual está diseñado para trabajar con JavaBeans.

Spring Context.

El Spring Context es un archivo de configuración que provee de información contextual al framework general. Además provee servicios enterprise como JNDI, EJB, e-mail, validación y funcionalidad de agenda.

Spring AOP.

AOP es un enfoque diferente y un poco más complicado de acostumbrarse en comparación con OOP (Object Oriented Programming). Ron Johnson prefiere referirse a AOP como un complemento en lugar de como un rival o un conflicto. Spring implementa AOP utilizando proxies dinámicos. Además, se integra transparentemente con los BeanFactory que existen.

Spring ORM.

En lugar de que Spring proponga un propio módulo ORM (Object-Relational Mapping), para los usuarios que no se sientan confiados en utilizar simplemente JDBC, propone un módulo que soporta los frameworks ORM más populares del mercado entre ellos: Hibernate (2.1 y 3.0), iBatis SQL Maps (1.3 y 2.0), Apache OJB (1.0), entre otros como JDO (1.0 y 2.0) y Oracle TopLink.

Spring DAO.

El patrón DAO (Data Access Object) es uno de los patrones más importantes y usados en aplicaciones J2EE, y la arquitectura de acceso a los datos de Spring provee un buen soporte para este patrón.

Existen dos opciones para llevar a cabo el acceso, conexión y manejo de bases de datos: utilizar alguna herramienta ORM o utilizar el template de JDBC (Java DataBase Connectivity) que brinda Spring.

Spring Web.

El módulo web de Spring se encuentra en la parte superior del módulo de contexto, y provee el contexto para las aplicaciones web. Este módulo se encarga de diversas operaciones web como por ejemplo: las peticiones multi-parte que puedan ocurrir al realizar cargas de archivos y la relación de los parámetros de las peticiones con los objetos correspondientes (domain objects o business objects).

Spring Web MVC.

Spring brinda una arquitectura MVC (Model View Controller) para web bastante flexible y altamente configurable, se pueden desarrollar aplicaciones sencillas sin tener que configurar muchas opciones. Spring web MVC presenta algunas características muy importantes, las cuales se mencionan a continuación:

- Spring hace una clara división entre controladores, modelos de JavaBeans y vistas.
- El MVC de Spring está basado en interfaces y es bastante flexible.
- Spring no obliga a utilizar JSP como única tecnología View también se puede utilizar otras.
- Los controladores son configurados de la misma manera que los demás objetos en Spring, a través de IoC.

1.8.3 Framework de Acceso a Datos.

Hibernate.

Hibernate es un entorno de trabajo que tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales y al mismo tiempo la consulta de estas bases de datos para obtener objetos. [14]

Hibernate es una capa de persistencia objeto/relacional y un generador de sentencias SQL. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada se puede generar BBDD en cualquiera de los entornos soportados: Oracle, DB2, MySql, entre otros. Lo más importante de todo, es Open Source.

Para el desarrollo de la aplicación informática se utilizó la versión 3.5.2. Hibernate se integra en cualquier tipo de aplicación justo por encima del contenedor de datos. Una posible configuración básica de Hibernate es la siguiente:

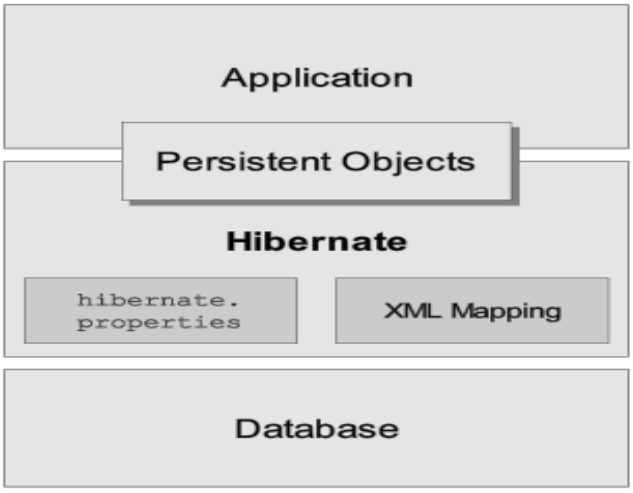


Figura 6. Arquitectura Base.

1.9 Sistema Gestor de base de datos.

Un Sistema Gestor de base de datos (SGBD) es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. Por tanto, debe permitir: [15]

- 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 SGBD.
- Manipular la base de datos: realizar consultas, actualizarla, generar informes.

PostgreSQL.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. [16]

Por sus características técnicas es uno de los sistemas de gestión de bases de datos más potentes y robustos del mercado. Con el paso del tiempo la estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. [16] Para el desarrollo de la aplicación se utilizó la versión 8.2.4.

A continuación se mencionan algunas de las características más importantes y soportadas por PostgreSQL:

- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows.
- Licencia BSD.
- Es una base de datos 100% ACID.
- Multi-Version Concurrency Control (MVCC).

1.10 Entorno de Desarrollo Integrado.

Un entorno de desarrollo integrado es una herramienta de soporte al proceso de desarrollo de software que integra las funciones básicas de edición de código, compilación y ejecución de programas con otras tales como: [17]

- Editor con corrección sintáctica y coloreo de la misma al momento de codificar.
- Herramientas gráficas.
- Soporte integrado para la compilación y ejecución de programas.
- Relación entre errores de compilación y el código fuente, para facilitar su corrección.
- Opciones de Debugging.

Eclipse.

Eclipse es una plataforma de desarrollo extensible, basada en Java y de tipo Open-Source (CPL). Originalmente fue desarrollada por Alphaworks, laboratorio de desarrollo de IBM, pero actualmente está manejado por un consorcio de varias empresas, (Borland, Hitachi, entre otros). [18]

Este entorno de desarrollo integrado ofrece, el control del editor de código, del compilador y del depurador desde una única interfaz de usuario. Su misión consiste en evitar tareas repetitivas, facilitar la escritura de código correcto, disminuir el tiempo de depuración e incrementar la productividad del desarrollador. Estas tareas pueden realizarse de muchas maneras distintas: mediante la inclusión de asistentes para las tareas más habituales y mecánicas, de editores que completen automáticamente el código y señalen los errores sintácticos y de gestores de archivos fuente. Eclipse no es un IDE más a añadir a la lista, el objetivo de IBM ha sido crear una plataforma de desarrollo modular que cualquier herramienta de desarrollo pueda

usar con cualquier lenguaje de programación. Para el desarrollo de la aplicación informática de utilizó el Eclipse 3.5.0.

1.11 Herramienta de Generación de Reportes.

Jasper Reports.

Jasper Reports es una herramienta desarrollada sobre plataforma Java. Puede ser usada por cualquier aplicación desarrollada en esa plataforma incluyendo J2EE o aplicaciones Web. Puede usar gran variedad de origen de datos, lo que posibilita extender las capacidades de crear reportes para casi cualquier tercera aplicación. Permite múltiples fuentes de información de múltiples tipos en un mismo reporte. Exporta a diferentes formatos como HTML, PDF, XML, XLS, RTF, CSV y TXT. Soporta imágenes y maneja fácilmente sub-reportes. [19] Dada la necesidad de la generación de reportes dinámicos se decidió utilizar Jasper Reports.

Conclusiones.

En este capítulo se han especificado algunos conceptos para comprender los sistemas de gestión de información computarizados. Se seleccionó la metodología de desarrollo, Rational Unified Process así como lenguaje de programación Java. También se abordaron los roles que se desempeñarán y los artefactos construidos por los mismos. Se argumentaron las herramientas y tecnologías en las que se desarrollará el sistema.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.

En este capítulo se describe la propuesta de solución del sistema según el problema descrito. Para ello se describe el funcionamiento del negocio mediante el modelo del dominio; además, se brinda una concepción general del sistema propuesto que se particulariza en las funcionalidades y características que tendrá el sistema a desarrollar. Se presenta el diagrama de casos de uso del sistema con las correspondientes especificaciones de los casos de uso asociados. También se argumentan los patrones de casos de uso utilizados.

2.1 Descripción del negocio.

RUP define en su primera fase de desarrollo la realización del modelo de negocio, con el objetivo de comprender el entorno del cliente y detectar las mejoras potenciales en los procesos de la organización. Cuando no es posible identificar claramente los procesos del negocio, RUP propone realizar un modelo de dominio, que es un subconjunto del modelo de negocio.

2.2 Modelo de Dominio.

El estudio del negocio realizado evidencia que los procesos que se llevan a cabo, no están bien definidos, por ello se procede a la realización del siguiente modelo de dominio.

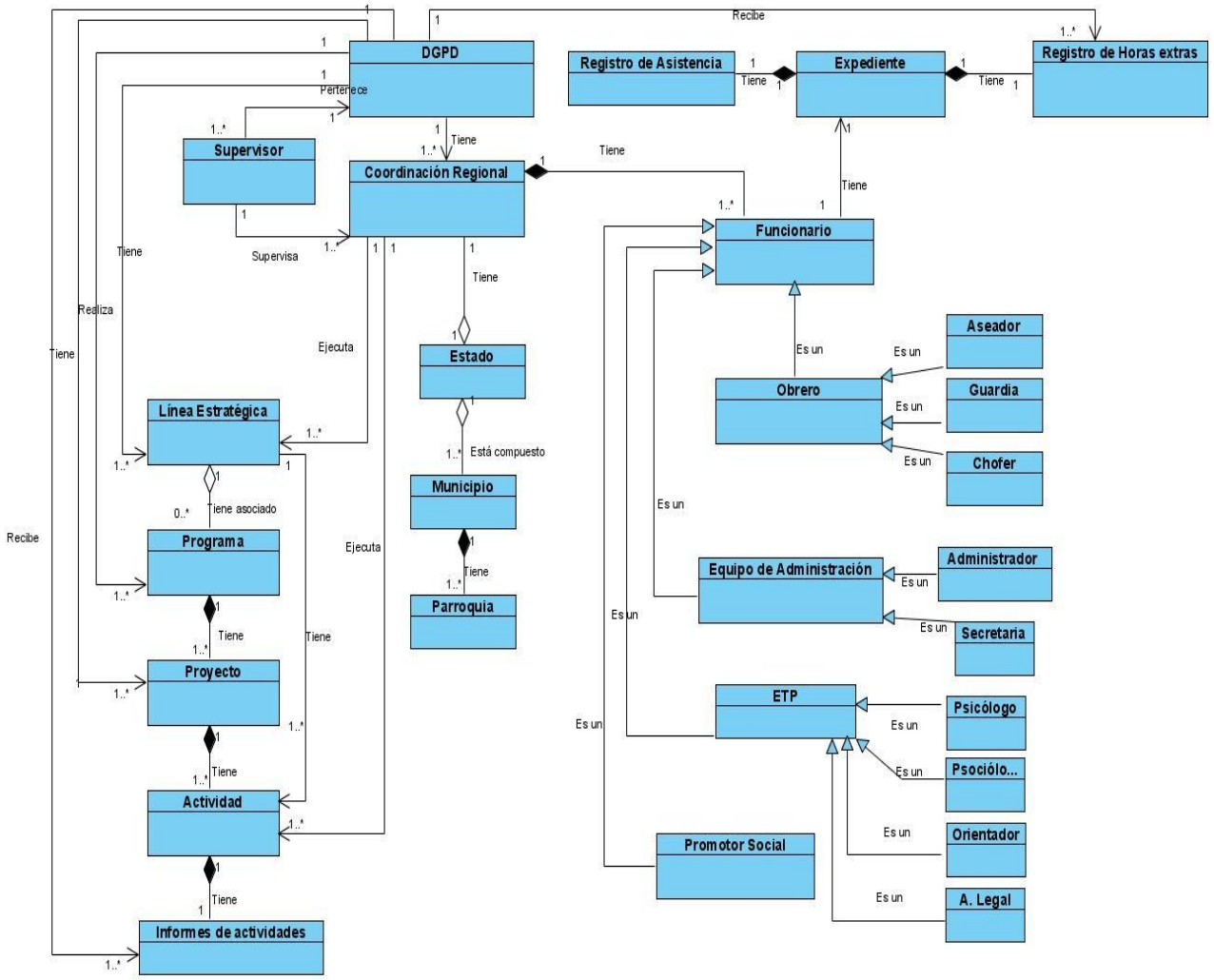


Figura 7. Modelo de Dominio.

Descripciones de los conceptos del Modelo de Dominio.

1- Clase DGPD.

Adscrita al Viceministerio de Seguridad Ciudadana del Ministerio del Poder Popular para Relaciones Interiores y Justicia de la República Bolivariana de Venezuela. Esta dirección planifica la política antidelictiva, cuyas funciones versan sobre el diseño y ejecución de política, planes y programas orientados a la prevención de la violencia y la criminalidad a través del ejercicio de la corresponsabilidad entre instituciones y comunidades y del fortalecimiento de la convivencia ciudadana.

2- Clase Coordinación Regional.

Es una oficina adscrita a la Dirección General de Prevención del Delito, conformada por un equipo de profesionales y técnicos que se encargan de la atención e implementación de los programas de prevención del delito, como el Programa de Educación y el de la Comunidad.

3- Clase Línea Estratégica.

La DGPD hasta el momento tiene definidas 11 líneas estratégicas para guiar el trabajo en función de la prevención del delito. Ejemplo de estas líneas son: plan nacional de prevención contra la corrupción, plan nacional contra la pornografía infantil, plan nacional contra la violencia de género, plan contra el consumo de especies alcohólicas y sustancias estupefacientes y psicotrópicas, entre otras.

4- Clase Programa.

Un programa es el resultado de una concepción filosófica, ideológica, funcional y operativa de lo que la Dirección General define como su "Función Social".

5- Clase Proyecto.

Un proyecto es un conjunto de actividades con un fin específico basado en los programas.

6- Clase Actividad.

Crear acciones preventivas, poner en marcha los convenios adquiridos en materia prevención, talleres, mítines, trabajos sociales, reuniones con las comunidades, mesas de trabajo y otras que son destinadas a incorporar a los miembros de la comunidad, en especial al sector juvenil. Otros ejemplos de actividades son capacitaciones a instituciones educativas, organismos de seguridad ciudadana y comunidades sobre la prevención de la delincuencia-violencia, consumo de drogas, lucha contra pornografía infantil, entre otros.

7- Informe de Actividades.

Este documento contiene un resumen de las actividades de la Coordinación Regional que se soliciten por la Dirección General, puede ser trimestral, mensual o las actividades relevantes en un periodo de tiempo dado.

8- Clases Estado, Municipio, Parroquia.

Distribución política geográfica que existe en la República Bolivariana de Venezuela, el estado está compuesto por Municipios y los Municipios por Parroquias.

9- Clase Supervisor.

Personal que trabaja en la Dirección General de Prevención del Delito, encargado de supervisar las coordinaciones regionales.

10- Clase Funcionario.

Trabajador o persona directamente vinculada a la CR que realiza actividades y procesos de interés para la CR.

11- Clase Obrero.

Trabajador de la coordinación general que no es graduado universitario. Existen diversos obreros entre los que se encuentran: aseador, guardia y chofer.

12- Clase Equipo de Administración.

Personal encargado de administrar los Recursos Humanos de la Coordinación Regional, estos pueden ser: Administrador y Secretaria.

13- Clase ETP.

Equipo Técnico Profesional, personal vinculado directamente a la coordinación regional que realiza una serie de procesos de interés. Estos pueden ser: servicios a la población (Psicológicos, Sociales y de Orientación) y la planificación y responsabilidad de la ejecución de las actividades realizadas con las comunidades.

14- Clase Promotor Social.

Personal vinculado a la Coordinación Regional, es el encargado de divulgar y promover las actividades a realizar por las coordinaciones.

15- Clase Expediente

Documento donde se archiva los datos personales y ocupacionales del trabajador de la Coordinación Regional**.** En este documento se archivan las asistencias diarias y las horas extras del trabajador.

16- Clase Registro de Asistencia

Documento donde se recoge la asistencia diaria de cada trabajador de la Coordinación Regional.

17- Registro de Horas Extras.

Documento donde se recoge las horas extras trabajadas por cada trabajador de la Coordinación Regional. Este registro es enviado a la DGPD para que sean pagadas dichas horas al trabajador.

2.3 Modelación del sistema.

Requisitos

Se define requerimiento como condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.

2.4 Requisitos Funcionales.

Son capacidades o condiciones que el sistema debe cumplir. A continuación se muestran los requisitos funcionales de la aplicación agrupados por casos de usos:

CUS Gestionar actividad

RF1. Insertar una actividad.
RF2. Actualizar una actividad.
RF3. Eliminar una actividad.

CUS Visualizar actividad

RF4. Visualizar actividad.
RF5. Imprimir datos de la actividad.

CUS Gestionar listado de actividades

RF6. Visualizar listado de actividades.
RF7. Buscar actividad.
RF8. Generar reportes de actividades.
RF9. Exportar el resultado del reporte de actividades a formato Excel o PDF.

CUS Gestionar listado de trabajadores

RF10. Visualizar listado de trabajadores.
RF11. Buscar trabajador.
RF12. Generar reportes de trabajadores.

RF13. Exportar el resultado del reporte del listado de trabajadores a formato Excel o PDF.

CUS Gestionar trabajadores

RF14. Insertar trabajador.

RF15. Actualizar trabajador.

CUS Visualizar trabajador

RF16. Visualizar trabajador.

RF17. Imprimir datos del trabajador.

CUS Gestionar reportes de asistencia del trabajador

RF18. Visualizar historial de asistencia del trabajador.

RF19. Buscar asistencia diaria del trabajador.

RF20. Generar reporte del historial de asistencia del trabajador.

RF21. Exportar el resultado del reporte de asistencia del trabajador a formato Excel o PDF.

CUS Gestionar término de relación laboral

RF22. Registrar término de relación laboral.

RF23. Actualizar término de relación laboral.

CUS Gestionar listado de controles de asistencia diaria.

RF24. Visualizar listado de controles de asistencia diaria.

RF25. Buscar controles de asistencia diaria.

RF26. Generar reporte de controles de asistencia diaria.

RF27. Exportar el resultado del reporte del listado de controles de asistencia diaria a formato Excel o PDF.

CUS Gestionar controles asistencia diaria

RF28. Insertar control de asistencia diaria.

RF29. Actualizar control de asistencia diaria.

CUS Visualizar controles asistencia diaria

RF30. Visualizar control de asistencia diaria.

RF31. Imprimir control de asistencia diaria.

CUS Gestionar horas extras

RF32. Insertar horas extras del trabajador.

RF33. Actualizar horas extras del trabajador.

RF34. Eliminar horas extras del trabajador.

RF35. Buscar horas extras del trabajador.

CUS Gestionar Informes de horas extras del trabajador

RF36. Visualizar listado de horas extras del trabajador.

RF37. Generar informe de horas extras del trabajador.

RF38. Exportar el resultado del reporte de informes de horas extras del trabajador a formato Excel o PDF.

CUS Autenticar usuario

RF39. Autenticar usuario.

CUS Gestionar usuario:

RF40. Insertar usuario.

RF41. Actualizar usuario.

RF42. Eliminar usuario.

RF43. visualizar usuario.

RF44. Visualizar listado de usuarios.

CUS Gestionar línea de estrategia:

RF45. Insertar línea estratégica.

RF46. Eliminar línea estratégica.

RF47. Actualizar línea estratégica.

RF48. Visualizar listado de línea estratégica.

CUS Gestionar programas:

RF49. Insertar programa.

RF50. Eliminar programa.

RF51. Actualizar programa.

RF52. Visualizar listado de programa.

CUS Gestionar proyectos:

RF53. Insertar proyecto.

RF54. Eliminar proyecto.

RF55. Actualizar proyecto.

RF56. Visualizar listado de proyecto.

CUS Gestionar aéreas de atención:

RF57. Insertar aérea de atención.

RF58. Eliminar aérea de atención.

RF59. Actualizar aérea de atención.

RF60. Visualizar listado de aérea de atención.

CUS Gestionar tipo de actividad:

RF61. Insertar tipo de actividad.

RF62. Eliminar tipo de actividad.

RF63. Actualizar tipo de actividad.

RF64. Visualizar listado de tipo de actividad.

CUS Gestionar temas de la actividad:

RF65. Insertar tema de la actividad.

RF66. Eliminar tema de la actividad.

RF67. Actualizar tema de la actividad.

RF68. Visualizar listado de tema de la actividad.

CUS Gestionar consejo comunal:

RF69. Insertar consejo comunal.

RF70. Eliminar consejo comunal.

RF71. Actualizar consejo comunal.

RF72. Visualizar listado de consejo comunal.

CUS Gestionar Adjuntos:

RF73. Visualizar archivo.

RF74. Adjuntar archivo.

RF75. Eliminar archivo.

RF76. Visualizar listado de archivo.

2.5 Requisitos No Funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener.

Los requisitos no funcionales del sistema se detallan a continuación:

RNF1 - Requerimientos de Apariencia o interfaz externa

Se deben utilizar imágenes y colores identificados con la Dirección General de Prevención del Delito y el Ministerio del Poder Popular para Relaciones Interiores y Justicia de la República Bolivariana de Venezuela.

RNF2 - Requerimientos de Usabilidad

Los usuarios deben tener conocimientos básicos de computación, el sistema tendrá una interfaz de autenticación donde el usuario debe registrar su usuario y contraseña, además presentará un menú donde estarán todas las funcionalidades de la aplicación.

NF3 - Requerimientos de soporte

Una vez terminado el desarrollo del software se deben tomar decisiones con motivo de asistir a los clientes para lograr un mejoramiento progresivo y evolutivo, por lo que se plantean como requerimientos de soporte.

- Reinstalación de la aplicación ante fallos.
- Instalación de nuevas versiones o actualizaciones de la aplicación.
- Solución de fallos de configuración de la tecnología asociada a la aplicación.

RNF4 - Requerimientos de Seguridad

- **Confidencialidad**

  La autenticación será la primera acción del usuario en el sistema y consistirá en proveer un nombre de usuario único y una contraseña que debe ser de conocimiento exclusivo de la persona que se autentica.

- **Integridad**

  Se debe garantizar que la información sensible sólo pueda ser vista por los usuarios con el nivel de acceso adecuado y que las funcionalidades del sistema se muestren de acuerdo con el usuario que esté activo.

- **Disponibilidad**

  El sistema debe estar disponible para su utilización las 24 horas del día, durante los siete días de la semana, con el menor tiempo posible de recuperación ante fallos.

RNF5 - Requerimientos Legales

La licencia para la comercialización del producto se emitirá por la Dirección de Servicios Legales de la Infraestructura Productiva, para la misma se tendrá en cuenta los componentes que se utilizaron para el desarrollo de la aplicación.

RNF6 - Requerimientos de software

- **Cliente**

- Debe tener instalado un navegador web. Se recomienda Mozilla Firefox 3.5 o superior o Google Chrome 4.0.
- Debe tener instalado Sistema Operativo: Windows XP o superior, o GNU Linux.

- **Servidor**

- Se debe instalar TOMCAT como servidor web y PostgreSQL como gestor de base de datos.
- Debe estar instalado el Java Runtime Environment (JRE) versión 1.6 o superior.
- Debe estar instalado Sistema Operativo: Windows XP o superior, o GNU Linux.

RNF7 - Requerimientos de Hardware

Para el funcionamiento del componente se requiere de máquinas con los siguientes requisitos:

- Para las PC clientes: procesador Pentium o superior, 256 Mb de RAM y al menos 10Gb de capacidad del disco duro.
- El servidor de aplicaciones debe tener las siguientes características: capacidad de disco duro superior a 40 GB, microprocesador Pentium IV superior a 2.0 GHz y como mínimo 1.0 GB de RAM.
- El servidor de base de datos debe tener las siguientes características: capacidad de disco duro superior a 80 GB, microprocesador Pentium IV superior a 2.0 GHz y como mínimo 1.0 GB de RAM.

RNF8 - Requerimientos de restricciones en el diseño y la implementación

- Los componentes del sistema deben desarrollarse siguiendo el principio de alta cohesión y bajo acoplamiento.
- La lógica de presentación constituirá una capa independiente de la lógica de negocio, centrando su función en la interfaz de usuario y validaciones de los datos de entrada.

2.6 Diagrama de Casos de Uso del Sistema.

El Diagrama General de Casos de Uso describe la división del sistema en paquetes, así como la interacción de los actores con los casos de uso.

A continuación se presenta el Diagrama de Casos de Uso del Sistema Informático de Gestión de Información de los Recursos Humanos y las Actividades de las Coordinaciones Regionales de Prevención del Delito.

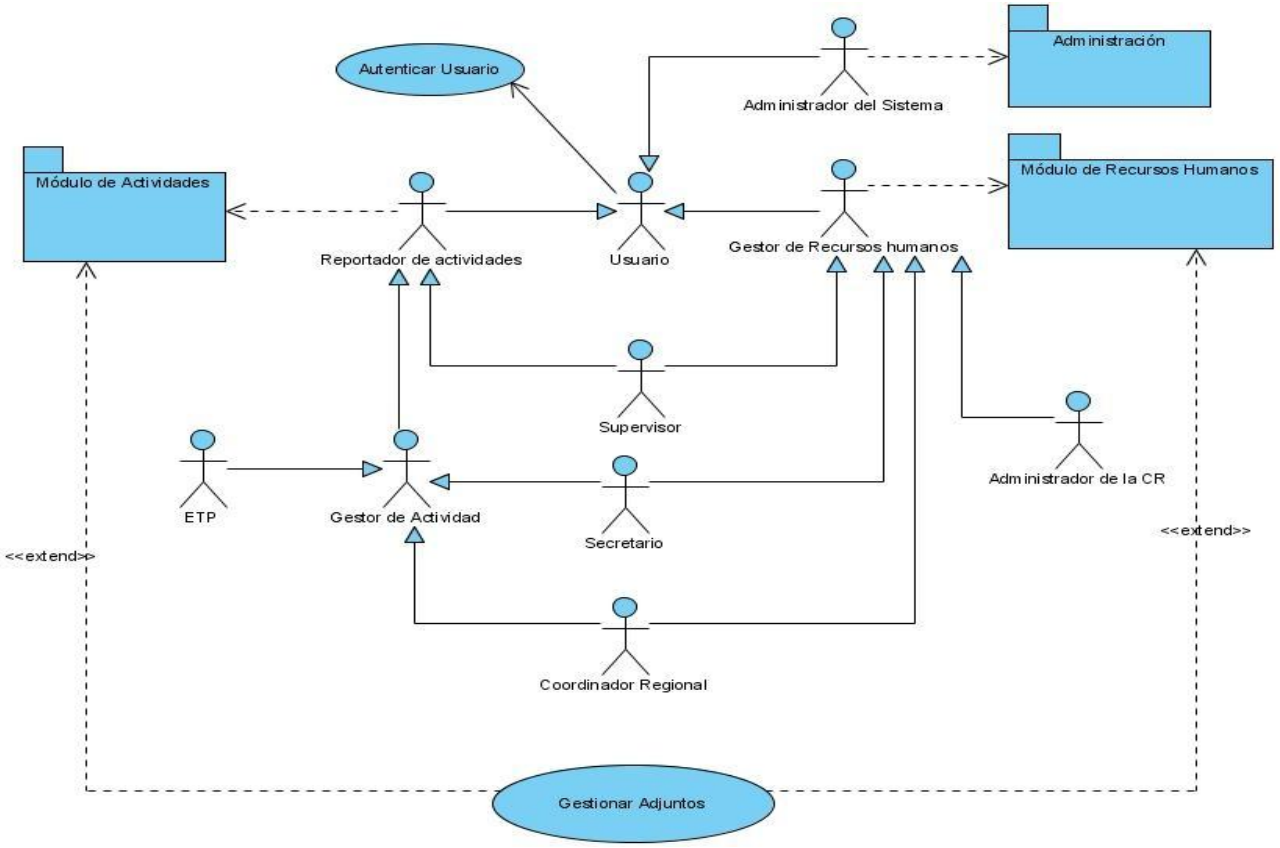


Figura 8. Diagrama de Casos de Uso general.

A continuación se presenta el Diagrama de Casos de Uso del Sistema del paquete de Administración.

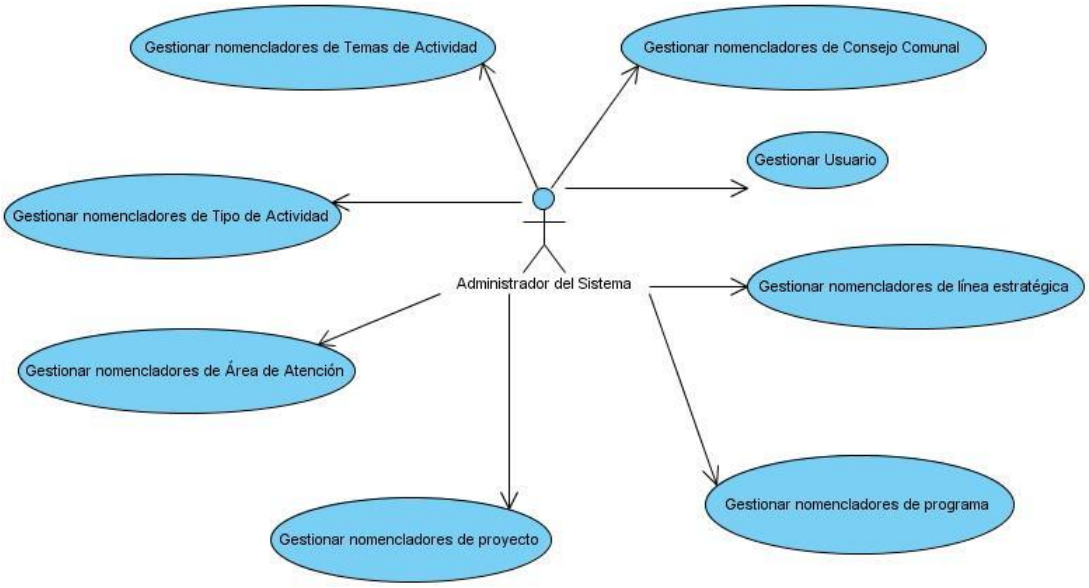


Figura 9. Diagrama de Casos de uso del Sistema. Módulo de Administración.

A continuación se presenta el Diagrama de Casos de Uso del Sistema del paquete de Actividades.

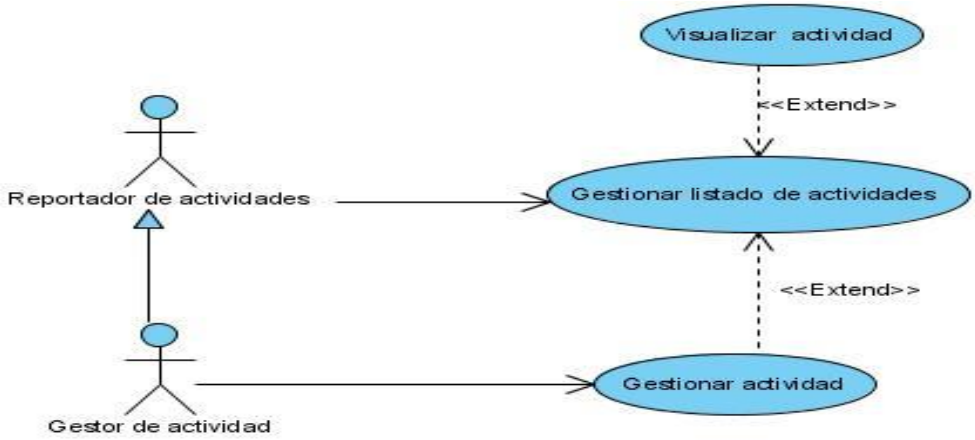


Figura 10. Diagrama de Casos de Uso del Sistema. Módulo de Actividades.

A continuación se presenta el Diagrama de Casos de Uso del Sistema del paquete de Recursos Humanos.

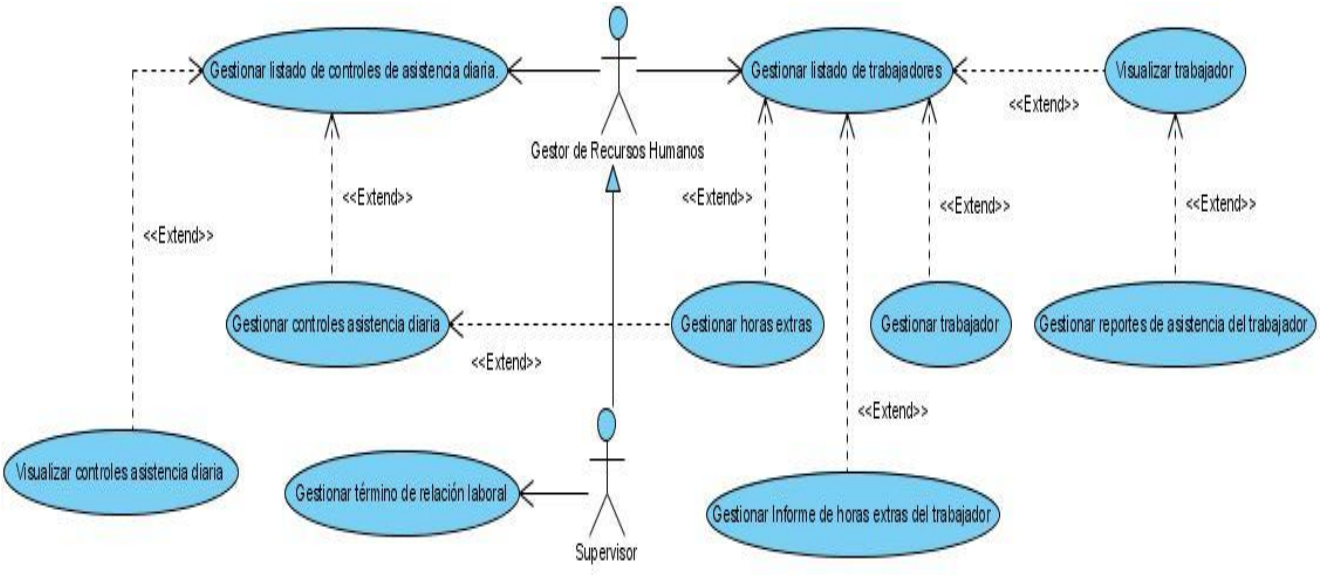


Figura 11. Diagrama de Casos de Uso del Sistema. Módulo de Recursos Humanos.

2.7 Patrones de Casos de Uso utilizados.

Múltiples actores

Roles comunes.

Puede suceder que los dos actores jueguen el mismo rol sobre el CU. Este rol es representado por otro actor, heredado por los actores que comparten este rol. Es aplicable cuando, desde el punto de vista del caso de uso, solo exista una entidad externa interactuando con cada una de las instancias del caso de uso.

A continuación se presenta un ejemplo donde se evidencia dicho patrón. En este caso los actores “Reportador de actividades” y “Gestor de Recursos Humanos”, ambos son usuarios del sistema por lo que para acceder al mismo deben ejecutar el caso de uso “Autenticar Usuario”.

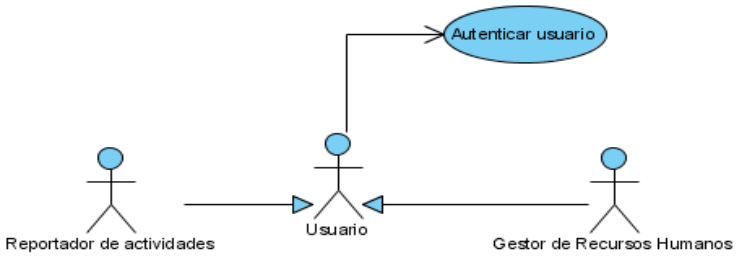


Figura 12. Ejemplo del Patrón Múltiples actores.

CRUD (Creating, Reading, Updating, Deleting)

Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual.

Este patrón consta de un caso de uso, llamado Información CRUD o Gestionar información, modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples.

A continuación se presenta un ejemplo donde se evidencia dicho patrón. En este caso existe un actor llamado “Administrador del Sistema” que ejecuta el caso de uso Gestionar Usuario. Este caso de uso agrupa un conjunto de funcionalidades asociadas a un mismo tipo de información, como: insertar, eliminar, actualizar y visualizar usuario.

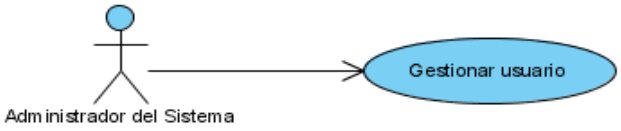


Figura 13. Ejemplo del Patrón CRUD.

Concordancia (Commonality)

Extrae una subsecuencia de acciones que aparecen en diferentes lugares del flujo de casos de uso y es expresado por separado.

- **Adición**

En el caso de este patrón alternativo, la subsecuencia común de casos de uso, extiende los casos de uso compartiendo la subsecuencia de acciones. Los otros casos de uso modelan el flujo que será expandido con la subsecuencia. Este patrón es preferible usarlo cuando otros casos de uso se encuentran propiamente completos, o sea, que no requieren de una subsecuencia común de acciones para modelar los usos completos del sistema.

A continuación se presenta un ejemplo donde se evidencia dicho patrón. En este caso existe un caso de uso base llamado “Gestionar listado de controles de asistencia diaria”, el mismo extiende a los casos de usos “Visualizar controles de asistencia diaria” y “Gestionar controles de asistencia diaria”. Es decir, que los casos de usos “Visualizar controles de asistencia diaria”, y “Gestionar controles de asistencia diaria”, dependen para ejecutarse del caso de uso Gestionar listado de controles de asistencia diaria.

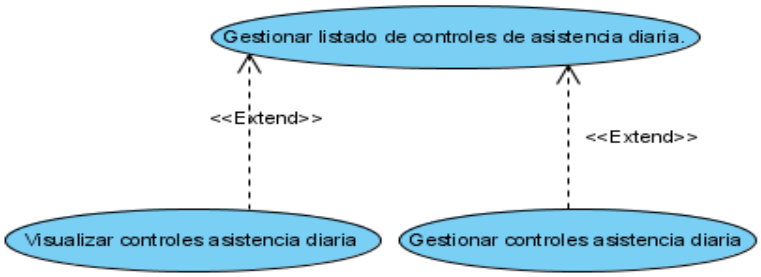


Figura 14. Ejemplo del Patrón Concordancia (Adición)

2.8 Especificación de los Casos de Uso del Sistema.

A continuación se muestran las descripciones textuales de los casos de uso: Gestionar listado de trabajadores y Gestionar Trabajador.

CUS: Gestionar listado de Trabajadores.

Nombre del CU	**Gestionar listado de Trabajadores.**
Actores	Gestor de Recursos Humanos.
Propósito	Este caso de uso tiene como propósito gestionar el listado de los trabajadores.
Resumen	El caso de uso se inicia cuando el Gestor de Recursos Humanos decide gestionar el listado de los trabajadores. El caso de uso da la opción de gestionar los datos de los trabajadores, buscar trabajadores en el listado, generar reportes a partir de las búsquedas realizadas y exportar a formato Excel o PDF el resultado de la búsqueda.
Referencias	RF10, RF11, RF12, RF13
Precondiciones	El usuario tiene que estar autenticado con el rol de Gestor de Recursos Humanos.
Poscondiciones	Se genera un reporte y/o se buscan trabajadores.

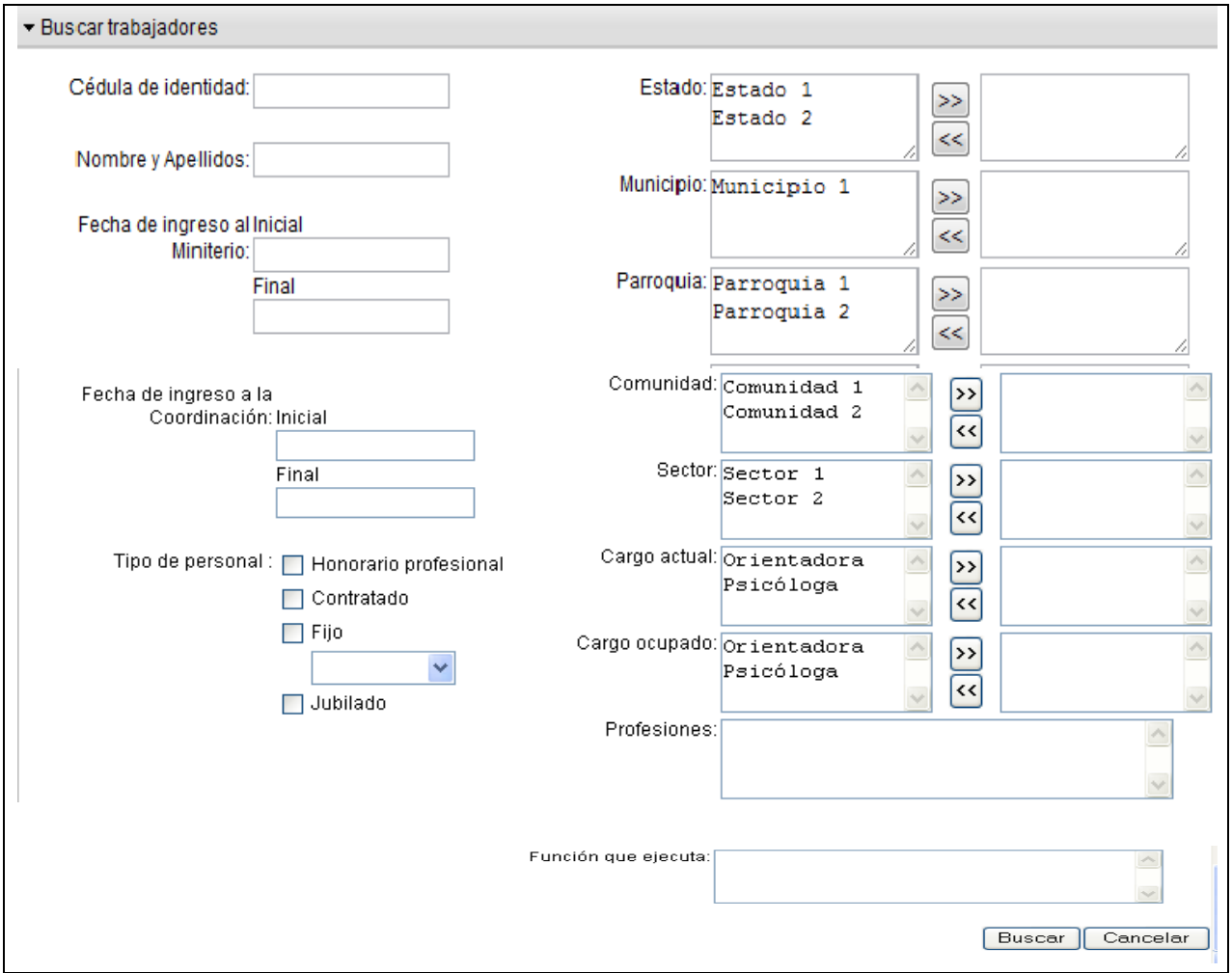
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El Gestor de Recursos Humanos decide gestionar listado trabajadores.	2. El sistema verifica que existan trabajadores registrados.
	3. El sistema muestra un listado con todos los trabajadores existentes y da la posibilidad de registrar, actualizar, visualizar y buscar trabajador, generar reportes, y visualizar horas extras.

4. El Gestor de Recursos Humanos selecciona una de las opciones. a) Registrar trabajador b) Actualizar trabajador c) Visualizar trabajador d) Buscar trabajador e) Generar Reporte del trabajador f) Visualizar Horas extras	5. El sistema ejecuta alguna de las siguientes opciones en dependencia de lo seleccionado: a) Si decide registrar un trabajador, ir al caso de uso "Gestionar Trabajador", la "Sección Registrar Trabajador". b) Si decide actualizar un trabajador, ir al caso de uso "Gestionar Trabajador", la "Sección Actualizar Trabajador". c) Si decide visualizar un trabajador, ir al caso de uso "Visualizar Trabajador". d) Si decide buscar trabajador, ir a la "Sección Buscar Trabajador". e) Si decide generar reporte del listado de trabajadores, ir a la "Sección Generar reporte del listado de trabajadores". f) Si decide visualizar horas extras, ir al caso de uso "Gestionar Horas extras".
Curso Alternativo de los eventos	

Acciones del Actor	Respuesta del Sistema
	3.1 Si no existen trabajadores registrados entonces el sistema muestra el listado vacío y solamente da la posibilidad de registrar trabajador.
3.2 El usuario indica registrar trabajador.	3.3 Se ejecuta la acción 5.a del Curso Normal de los Eventos.

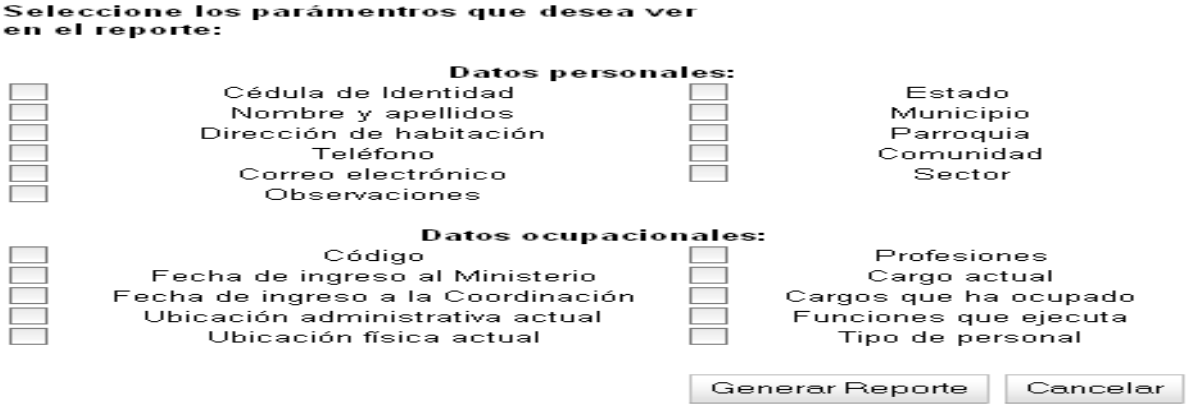
Sección "Buscar Trabajador"	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
	1. El sistema muestra un formulario con los datos que debe llenar.
2. El Gestor de Recursos Humanos introduce los datos.	3. El sistema verifica que no se hayan introducido datos inconsistentes.

	4. El sistema realiza la búsqueda y verifica si existen trabajadores que coincidan con los datos indicados.
	5. El sistema muestra los resultados encontrados en el listado de trabajadores.
Cursos Alternativos de los eventos	
Acciones del Actor	**Respuesta del Sistema**
	4.1 Si el Gestor de Recursos Humanos introdujo datos inconsistentes, el sistema emite un mensaje de error.
	5.1 Si no existen trabajadores que coincidan con los datos indicados, el sistema muestra un mensaje informativo.



Sección "Generar reporte del listado de Trabajadores"	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
	1. El sistema muestra un formulario para seleccionar los parámetros que desea ver en el reporte.

2. El Gestor de Recursos Humanos selecciona los datos.	3. El sistema verifica que el Gestor de Recursos Humanos haya seleccionado algún campo.
	4. El sistema genera el reporte según los campos seleccionados por el Gestor de Recursos Humanos y da la posibilidad de exportarlo a Excel o PDF.
5. El Gestor de Recursos Humanos selecciona una de las opciones: • Exportar a Excel. • Exportar a PDF.	6. Si el Gestor de Recursos Humanos selecciona la opción de exportar a Excel el sistema exporta a Excel, si el Gestor de Recursos Humanos selecciona la opción de exportar a PDF el sistema exporta a PDF.
Cursos Alternativos de los eventos	
Acciones del Actor	**Respuesta del Sistema**
	4.1 Si el Gestor de Recursos Humanos no selecciona ningún campo, el sistema muestra un listado que tiene por defecto los campos del listado inicial de trabajadores, además da la opción de exportar a Excel o PDF.
	4.2 Se ejecuta la acción 5 del Curso Normal de eventos.

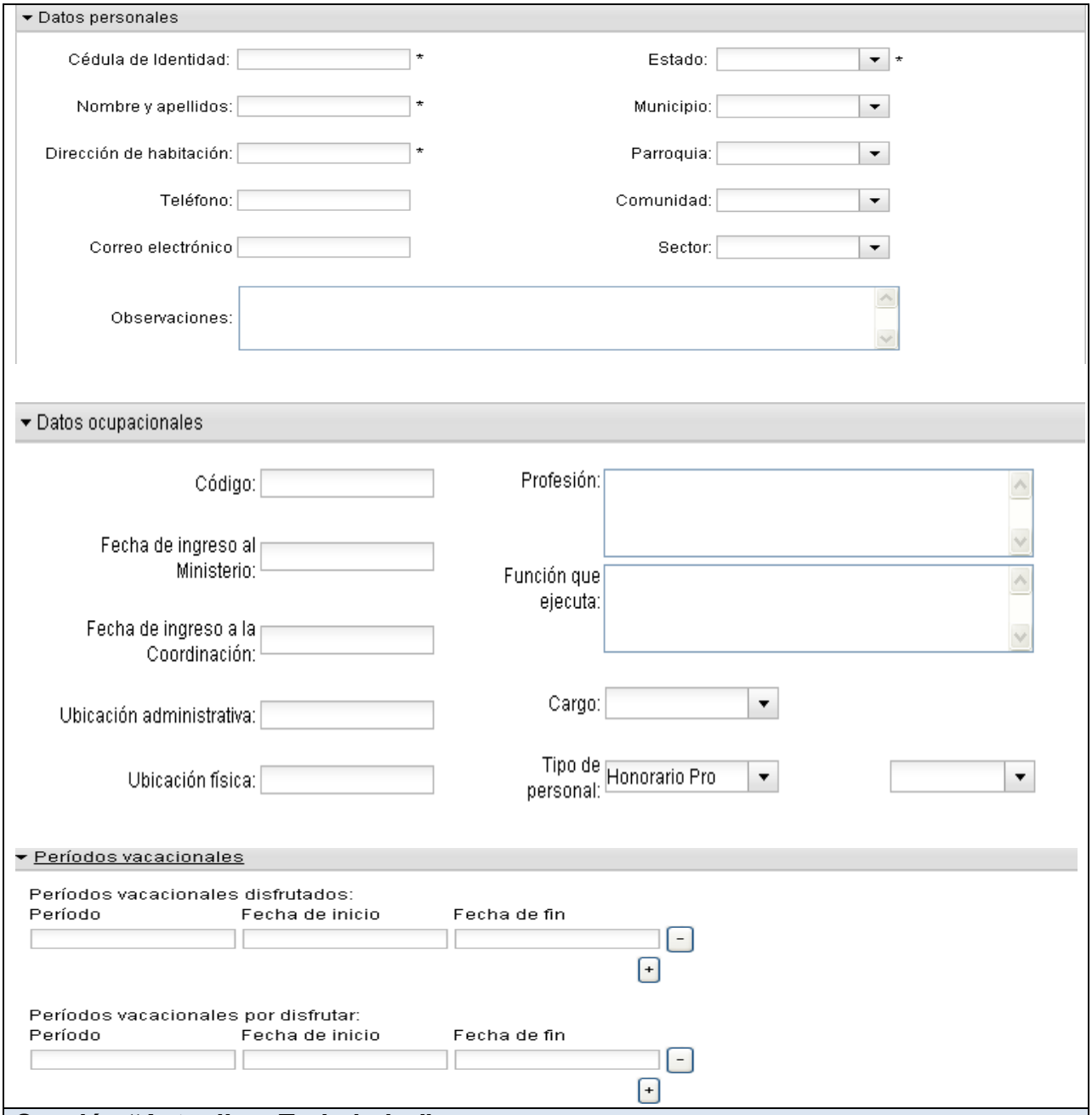


Prioridad	Crítico.

CUS: Gestionar Trabajadores.

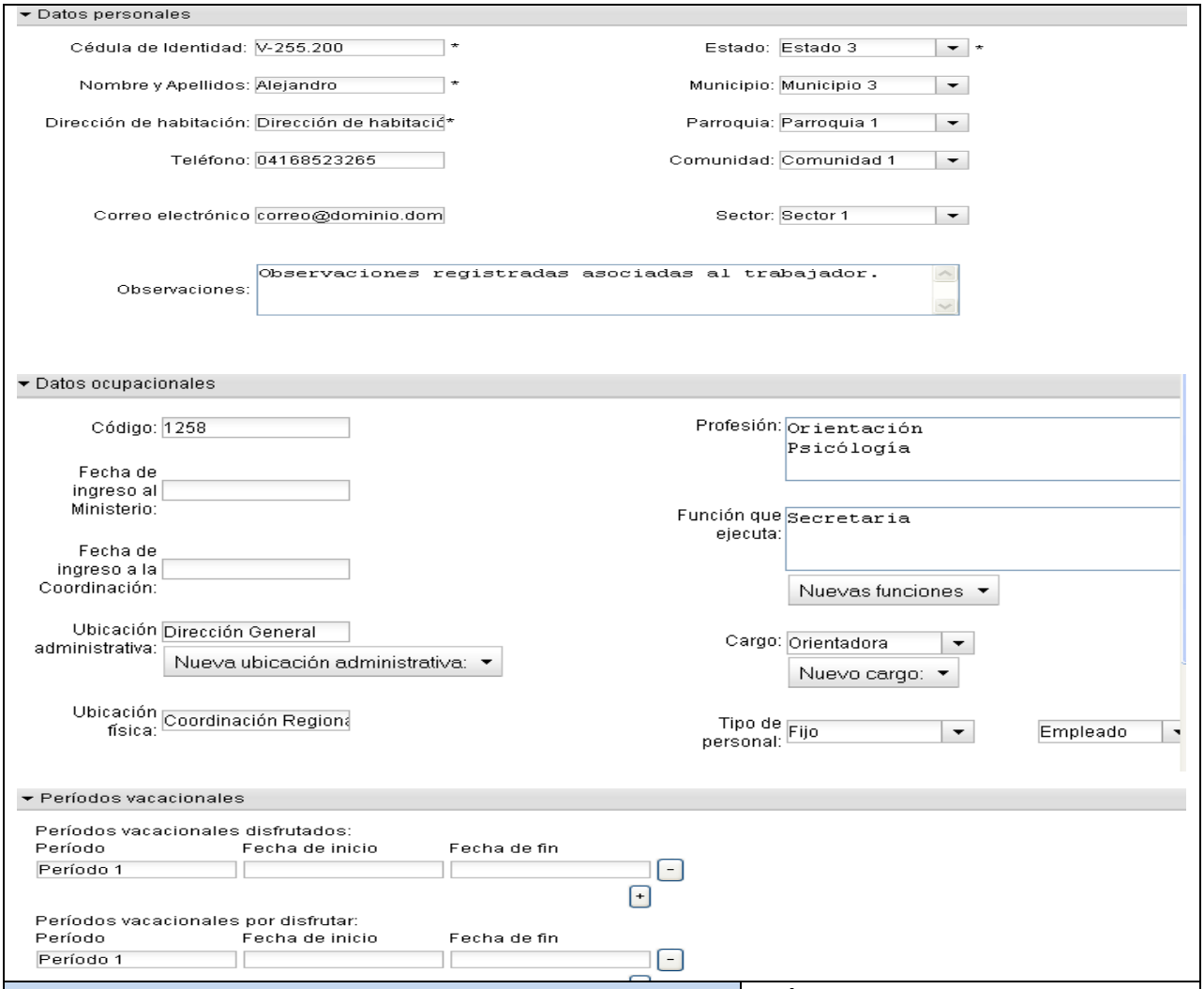
Nombre del CU	**Gestionar Trabajadores.**
Actores	Gestor de Recursos Humanos.
Propósito	Este caso de uso tiene como propósito gestionar trabajadores.
Resumen	El caso de uso se inicia cuando el Gestor de Recursos Humanos decide gestionar trabajadores. El caso de uso da la opción de registrar, visualizar, actualizar y eliminar trabajadores.
Referencias	RF14, RF15
Precondiciones	El usuario tiene que estar autenticado con el rol de Gestor de Recursos Humanos.
Poscondiciones	Quedan registrados y/o actualizados los datos de los trabajadores.
Sección “Registrar Trabajador”	
Curso Normal de los Eventos	

Acciones del Actor	**Respuesta del Sistema**
	1. El sistema muestra un formulario con los datos que debe llenar.
2. El usuario introduce los datos.	3. El sistema verifica que no hayan introducido datos inconsistentes y además que todos los campos obligatorios estén llenos.
	4. El sistema verifica que no exista ese trabajador.
	5. El sistema realiza la inserción y muestra la página donde se visualiza el listado de trabajadores y finaliza el caso de uso.
Cursos Alternativos de los eventos	
Acciones del Actor	**Respuesta del Sistema**
	4.1 Si hay datos inconsistentes o además no todos los campos obligatorios no están llenos; emite un mensaje de error.
	5.1 Si el trabajador ya existe, el sistema emite un mensaje de error informando que ya existe ese trabajador y no realiza la inserción.



Sección "Actualizar Trabajador"	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**

	1. El sistema muestra un formulario con los datos del trabajador.
2. El usuario cambia los datos que desea.	3. El sistema verifica que no hayan introducido datos inconsistentes ni campos obligatorios vacíos.
	4. El sistema actualiza el trabajador y finaliza el caso de uso.
Curso Alternativo de los eventos	
Acciones del Actor	**Respuesta del Sistema**
	4.1 Si el sistema verifica que hay datos inconsistentes y además que no todos los campos obligatorios están llenos; emite un mensaje de error.



Prioridad	Crítico.

Conclusiones.

En el presente capítulo fueron elaborados y descritos brevemente algunos de los artefactos propuestos por RUP para el desarrollo de software, tales como el modelo de dominio, definición de los requisitos funcionales y no funcionales, diagrama de casos de uso del sistema y la descripción textual de los casos de uso del sistema. También se ejemplificaron los patrones de casos de uso utilizados.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

En este capítulo se fundamenta la arquitectura utilizada y los principales patrones de diseño a utilizar. Se traducen los requerimientos del sistema a una especificación que describe como implementar el sistema a través del diseño de clases, teniendo en cuenta los requisitos funcionales y no funcionales. Se realiza el diagrama de clases del diseño del caso de uso Gestionar Trabajadores y los diagramas de secuencias por escenarios del mismo caso de uso. También se realiza el diseño de la base de datos.

3.1 Estilo Arquitectónico utilizado.

1. Arquitectura de Software

La arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. [20]

2. Estilo Arquitectónico

Define las reglas generales de organización en términos de un patrón y las restricciones en la forma y la estructura de un grupo de sistemas de software**.**

De acuerdo con las características del sistema a implementar se decidió utilizar el estilo arquitectónico en Capas (arquitectura en tres capas), perteneciente al estilo de Llamada y Retorno.

Este estilo define cómo organizar el modelo de diseño a través de 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. Ver figura 20.

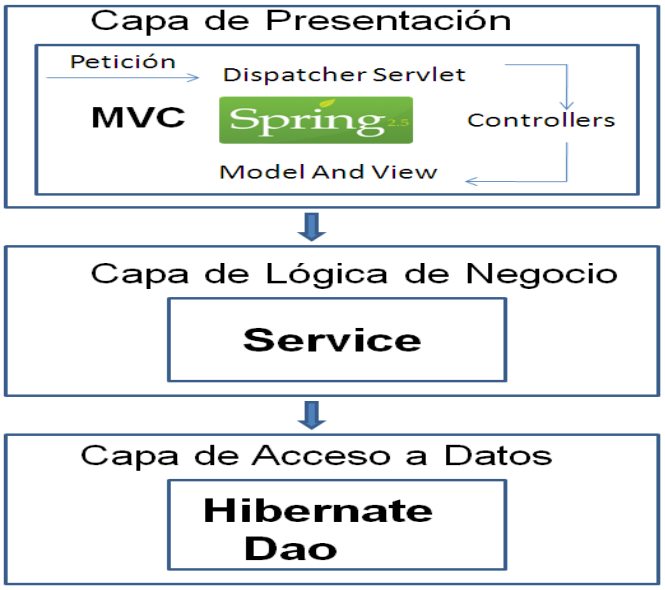


Figura 15. Arquitectura en Capas.

Capa de Presentación.

La capa de presentación define e implementa todo lo relacionado con la interfaz gráfica de usuario. Aquí residen la definición de las peticiones que el usuario puede realizar sobre la aplicación, los controladores que manejan el flujo web y la comunicación con las interfaces de la capa de negocio. Además, se encuentran las vistas HTML, XML, PDF, XLS que se muestran al usuario y la implementación del comportamiento dinámico de los documentos HTML a través de JavaScript. Las responsabilidades principales de la capa de presentación son: la navegabilidad del sistema, el formateo de los datos de salida, la validación de los datos de entrada y la construcción de la interfaz gráfica de usuario.

En esta capa va a estar presente el patrón arquitectónico Modelo Vista Controlador ya que Spring lo utiliza en su funcionamiento. A continuación se explica como funciona dicho framework:

Todas las peticiones son recibidas por el DispatcherServlet, este con el HandlerMapping identifica al controlador que procesará dicha petición, le pasa el control y obtiene como respuesta un ModelAndView, de este toma el nombre de la vista y utilizando el ViewResolver encuentra la vista física (archivo JSP), a la que le envía los datos extraídos del ModelAndView para que sea dibujada (rendered), resultado de lo cual obtiene el código HTML que es enviado como respuesta al cliente.

Capa de Lógica del Negocio.

La capa de negocio define e implementa las funcionalidades que responden directamente a los requisitos de manera que se conserve la integridad del sistema y de los datos. La misma está constituida por los servicios.

Capa de Acceso a Datos.

La capa de acceso a datos es la responsable de recobrar y persistir información desde y hacia la base de datos y de la comunicación con el gestor de base de datos. Esta capa contiene los objetos que encapsulan la lógica de acceso a datos (DAO) e interfaces brindadas para ser accedida desde la capa de negocio. Las implementaciones de los DAOs extienden de clases de soporte del framework Spring para el uso de este patrón usando el framework ORM Hibernate, mientras que las interfaces se mantienen independientes de Spring e Hibernate. Se encuentran además en esta capa las clases entidades que representan las clases del dominio.

3.2 Patrones de Diseño a utilizar.

Patrón de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Se debe tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios). [21]

A continuación se describen los patrones de diseño utilizados en la implementación del sistema informático.

Data Access Object (DAO)

Consiste en utilizar un objeto de acceso a datos para abstraer y encapsular todos los accesos a la fuente de datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos. [22]

El uso de este patrón en el sistema se evidencia en la capa de Acceso a Datos donde se encuentra un DAO Genérico encargado de realizar la gestión de los datos entre las clases que contienen la lógica de negocio (servicios) y el ORM Hibernate.

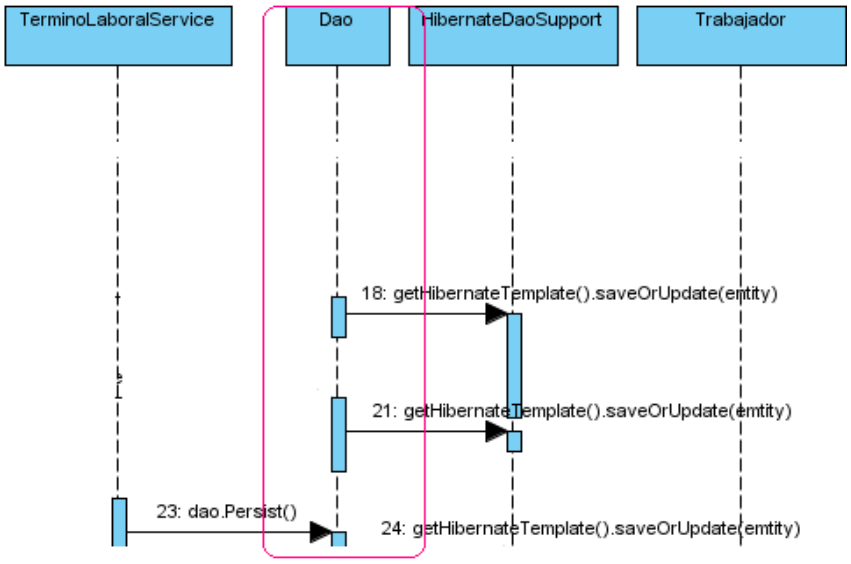


Figura 16. Ejemplo del Patrón DAO.

Inyección de dependencias

La inyección de dependencias radica en resolver las dependencias de cada clase (atributos) generando los objetos cuando se arranca la aplicación y luego inyectarlos en los demás objetos que los necesiten a través de métodos set o bien a través del constructor, pero estos objetos se instancian una vez, se guardan en una factoría y se comparten por todos los usuarios (al menos en el caso de Spring) y nos evitamos tener que andar extendiendo clases. [23]

En el desarrollo de la aplicación se pone en práctica el uso del patrón, ya que a todos los controladores se le inyectan los objetos correspondientes para su funcionamiento mediante el uso de la a*n*otación @Resource, la cual provee de atributos a clases mediantes los métodos set, especificando en el archivo XML el directorio donde se hará uso de anotaciones.

Front-Controller (Controlador Frontal)

El patrón propone utilizar un controlador como el punto inicial de contacto para manejar las peticiones del usuario en una aplicación. El controlador maneja el control de peticiones, incluyendo la invocación de los servicios de seguridad como la autentificación y autorización, la elección de una vista apropiada, el manejo de errores, y el control de la selección de estrategias de creación de contenido. Este patrón es utilizado por Spring MVC a través del DispatcherServlet.

En el desarrollo de la aplicación se evidencia el patrón de la siguiente manera: cada vez que entra una petición por la URL es recibida por el DispatcherServlet, el mismo posee un mecanismo para determinar cual controlador ejecuta la petición. El controlador ejecuta la acción y devuelve un ModelAndView al DispatcherServlet el que se encarga de despachar la petición a la vista.

Controller (Controlador)

Este patrón propone asignar la responsabilidad de controlar el flujo de eventos de un sistema a clases específicas llamadas controladores. Los controladores no ejecutan las tareas sino que las delegan en otras clases con las que mantiene un modelo de alta cohesión. Las clases a las cuales se les asignan estas responsabilidades se encuentran dentro de paquetes con la siguiente estructura nombre del paquete.controllers.

El patrón se evidencia en la aplicación de la siguiente manera: cuando llega una tarea a resolver al controlador este delega la responsabilidad a otras clases, en este caso los servicios que son los que tienen implementado la lógica de negocio.

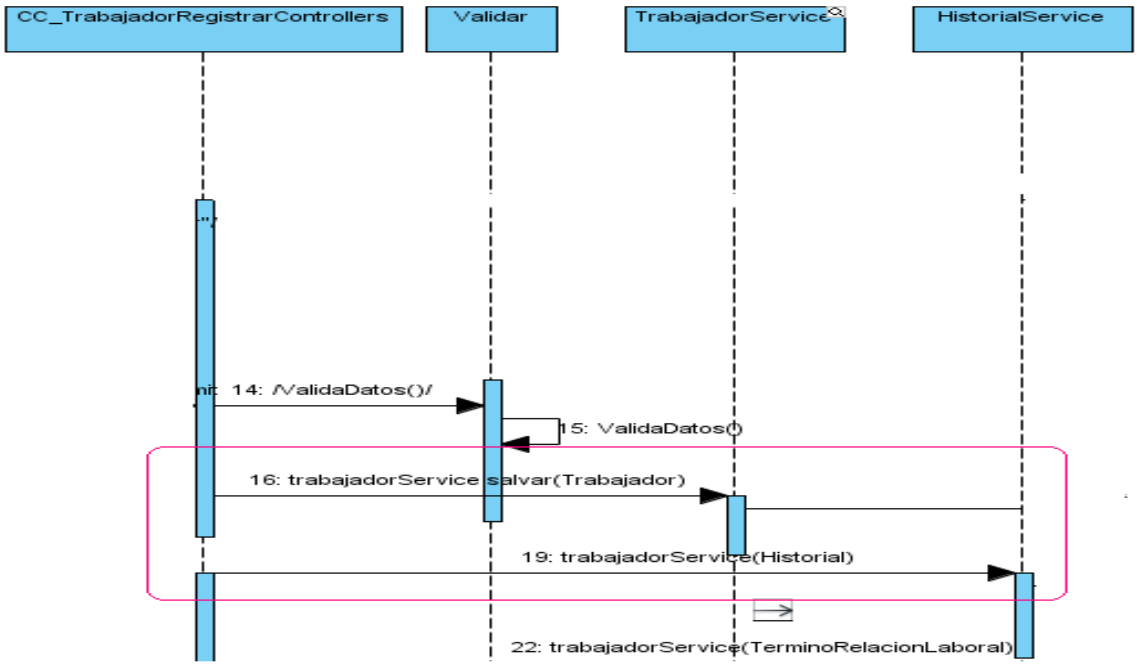


Figura 17. Ejemplo del patrón controllers.

3.3 Diagrama de Diseño Web.

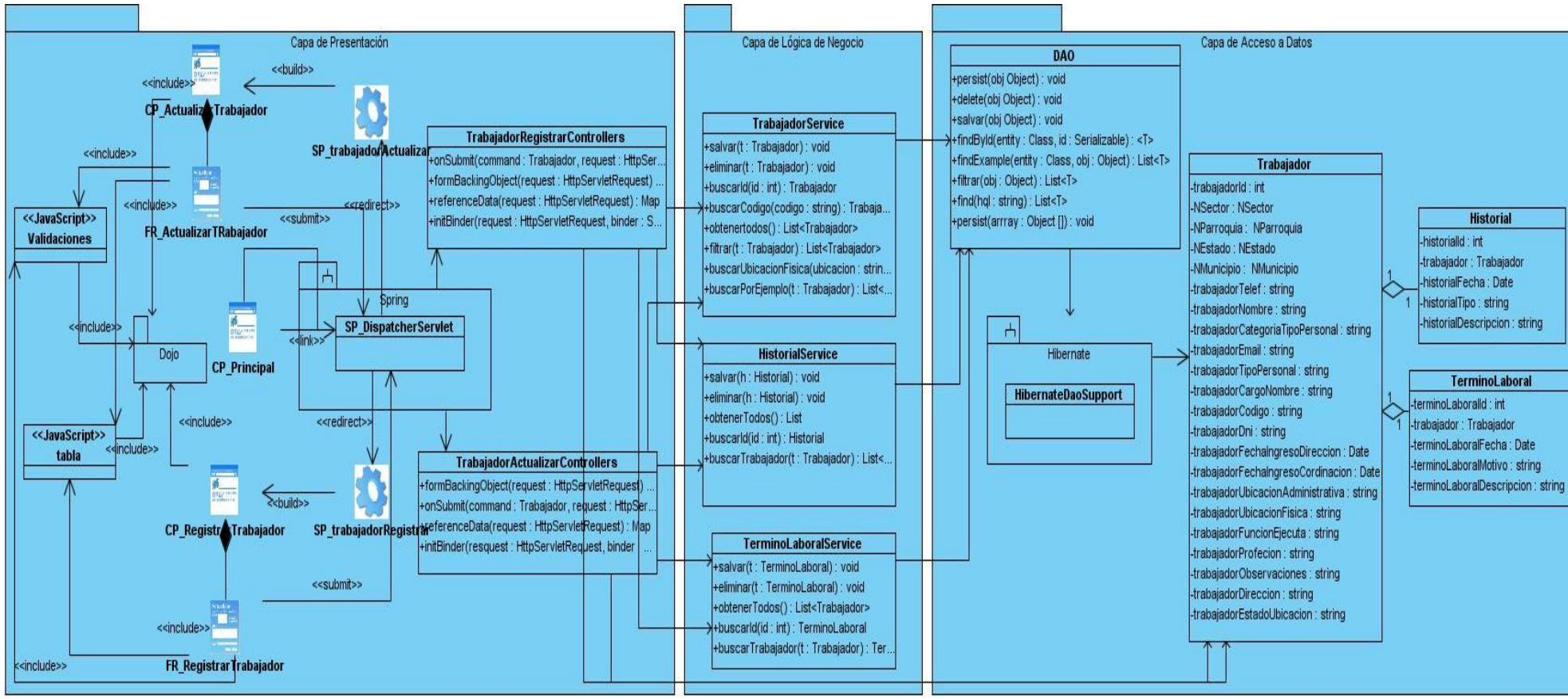


Figura 18. Diagrama de Diseño Web. CU Gestionar Trabajadores.

3.4 Diagramas de Secuencias.

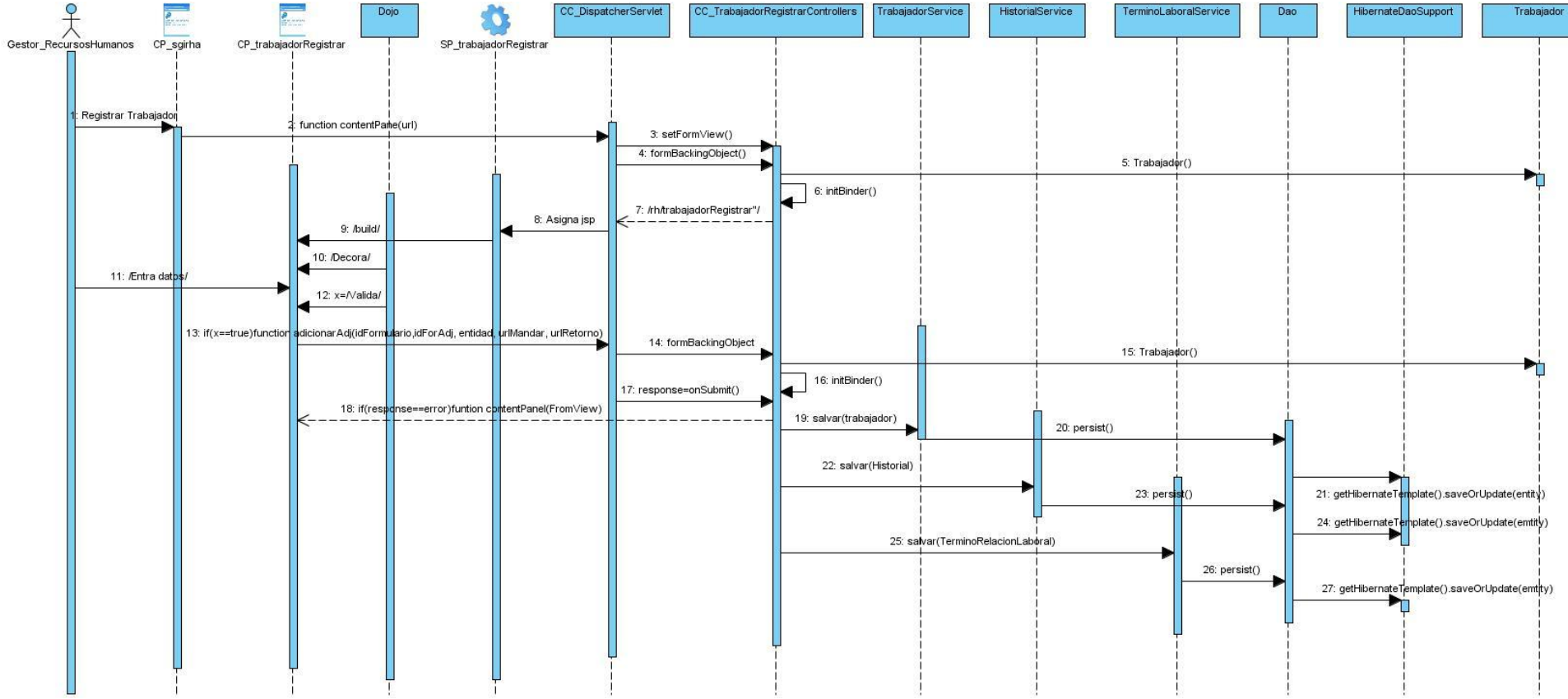


Figura 19. Diagrama de Secuencia del CU: Gestionar Trabajadores (Registrar).

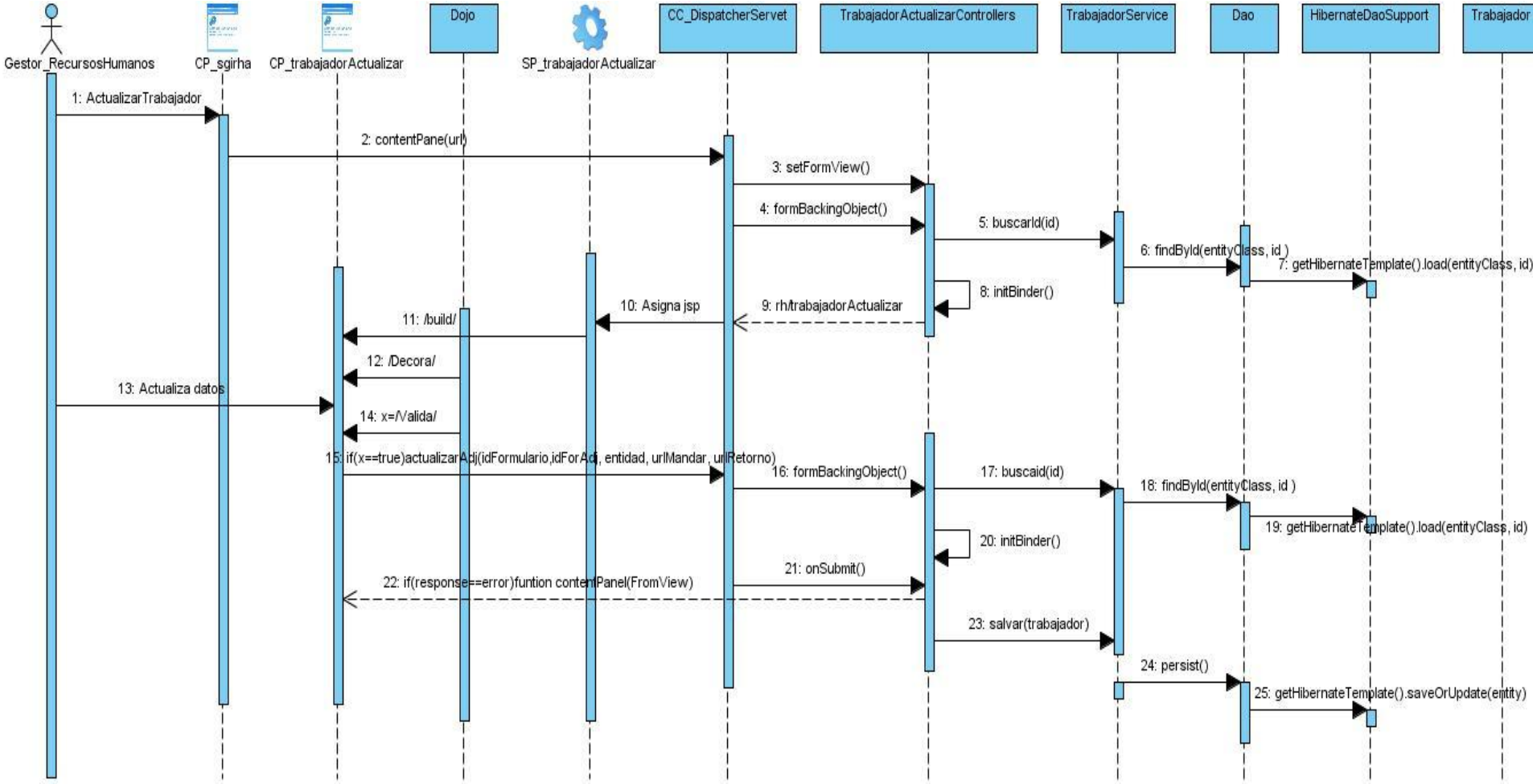


Figura 20. Diagrama de Secuencia del CU: Gestionar Trabajadores (Actualizar).

3.5 Base de Datos.

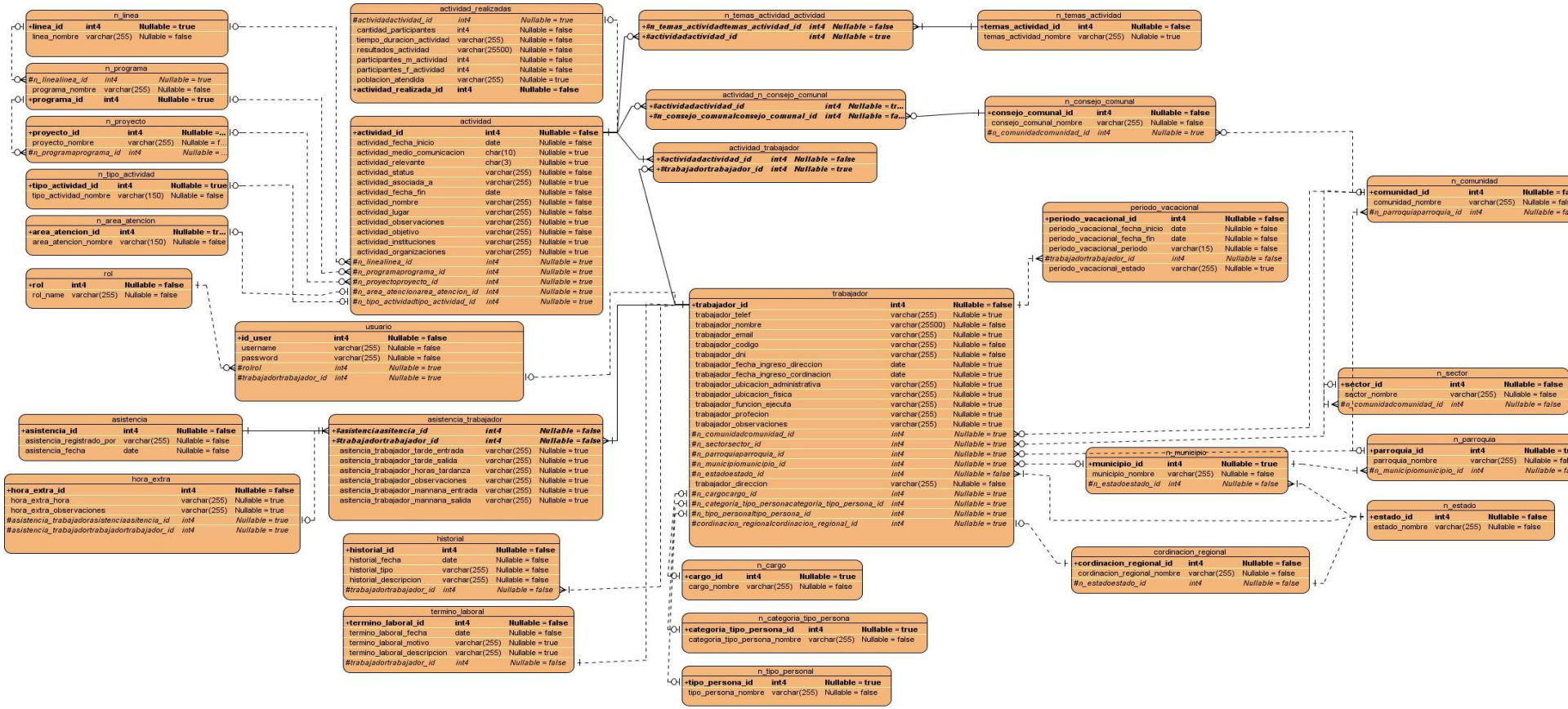


Figura 21. Diagrama de Entidad Relación.

Normalización de la base de datos.

El proceso de normalización de bases de datos consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional.

Las bases de datos relacionales se normalizan para:

- Evitar la redundancia de los datos.
- Evitar problemas de actualización de los datos en las tablas.
- Proteger la integridad de los datos.

La base de datos desarrollada se encuentra normalizada hasta la tercera forma normal ya que en ella se incluye la eliminación de todos los grupos repetidos (1ra FN), se asegura que todas las columnas que no son llave sean completamente dependientes de la llave primaria (2FN), y se elimina cualquier dependencia transitiva, es decir, se eliminan las dependencias entre las columnas que no son llave y que a su vez son dependientes de otras columnas que tampoco son llave (3FN).

Conclusiones.

En el presente capítulo se especificó el estilo arquitectónico utilizado así como los principales patrones de diseño. Se realizó el diagrama de clases del diseño del caso de uso Gestionar Trabajadores y los diagramas de secuencias por escenarios del mismo caso de uso. Además, se realizó el diseño de la base de datos.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

En este capítulo se describe cómo fue implementado el sistema. Se presenta el diagrama de componentes del caso de uso Gestionar Trabajadores, que muestra la distribución física de los componentes para la implementación. También se ejemplifica la organización de los ficheros y las validaciones. Además, se aborda sobre las pruebas realizadas a la aplicación y se muestran algunas interfaces del sistema.

4.1 Diagrama de Componente.

UML define el diagrama de componentes para la representación de un sistema de software en componentes físicos (por ejemplo archivos, cabeceras, módulos, paquetes) y establecer las dependencias entre estos componentes. [24]

A continuación se presenta el diagrama de Componente del CU: Gestionar Trabajadores.

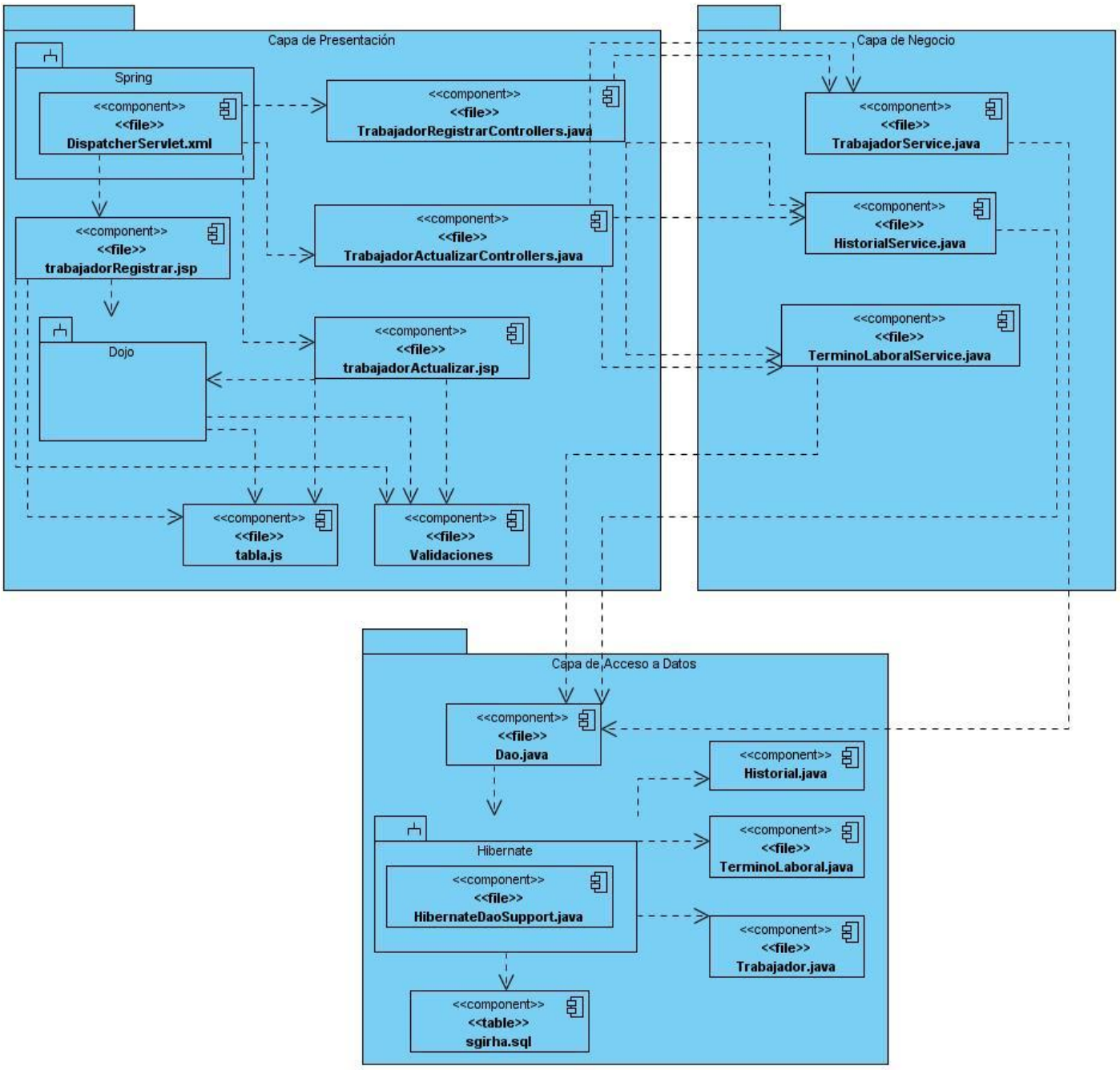


Figura 22. Diagrama de componente del CU Gestionar Trabajadores.

4.2 Organización de ficheros

Los ficheros se agruparon por paquetes en correspondencia a la capa donde pertenecían. La capa de Presentación cuenta con el paquete controllers y este a su vez agrupa las clases por módulos. En la capa de Negocio ocurre lo mismo que en la capa de Presentación, se agrupan las clases por módulos dentro del paquete service. La capa de Acceso a Datos cuenta con un único paquete Dao, que cuenta con las clases encargadas del acceso a los datos.

4.3 Validaciones

Dada la necesidad que el usuario que utilice el sistema inserte los datos correctamente, el sistema se validó del lado del cliente haciendo uso de las validaciones implementadas por el componte Dojo y con la utilización del lenguaje Java Script, para la validación de los campos del formulario. Se validó además posibles errores que pueden existir dentro del servidor como duplicación de códigos y procesos de eliminación incorrectos.

4.4 Pruebas a nivel de desarrollador.

A la aplicación informática se le desarrollaron pruebas a nivel de desarrollador. Tradicionalmente estas pruebas han sido consideradas solo para la prueba de unidad, enfocada a los elementos testeables más pequeños del software. Verifica que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. Como resultado de las pruebas realizadas se corrigieron errores ortográficos, campos sin validar, entre otros errores detectados en la interfaz de la aplicación, así como algunas líneas de código defectuosas.

4.5 Interfaces de la aplicación.

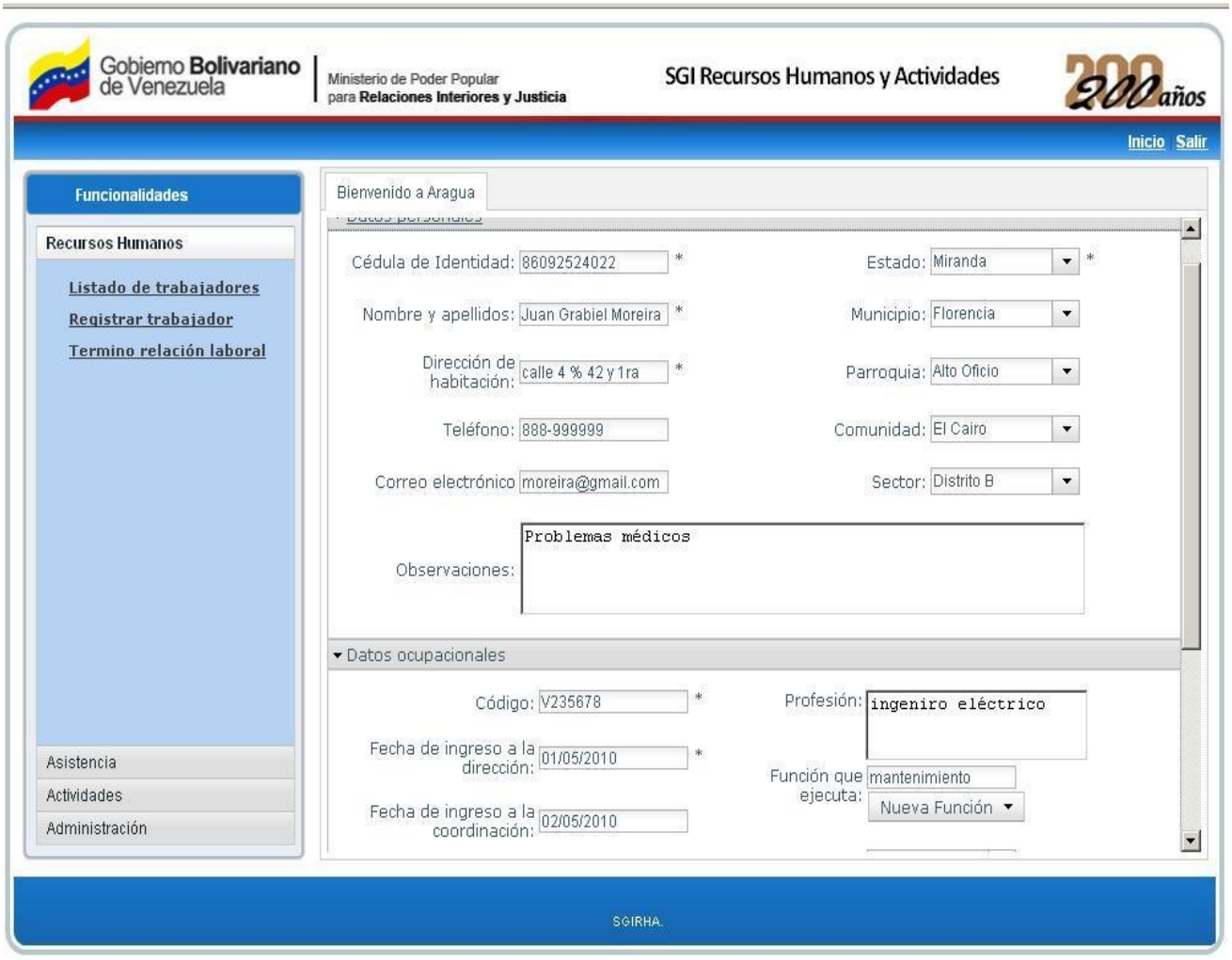


Figura 23. Interfaz Registrar Trabajador.

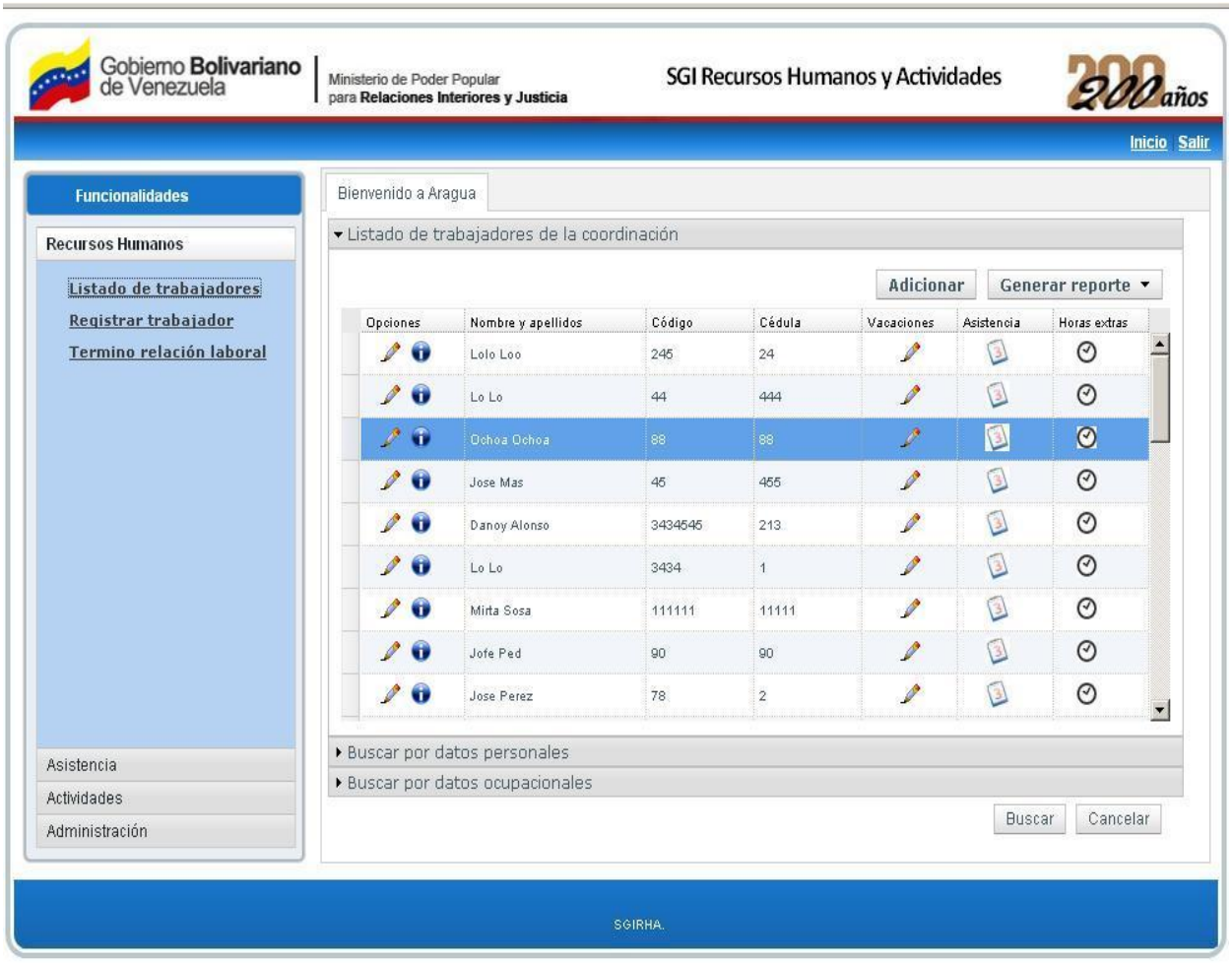


Figura 24. Interfaz Listar Trabajadores.

Conclusiones.

En el presente capítulo se realizó el diagrama de componente del caso de uso Gestionar Trabajadores, se hizo una breve descripción sobre las pruebas realizadas a la aplicación, también se argumentó sobre la organización de los ficheros. Seguidamente se hizo una breve descripción sobre las validaciones realizadas y finalmente se mostraron algunas interfaces relacionadas con los casos de uso Gestionar Trabajadores y Gestionar Listado de Trabajadores.

CONCLUSIONES

1. Se definieron las funcionalidades que garantizan la gestión de información referente a los Recursos Humanos y las Actividades de Prevención del Delito.
2. Se diseñaron las clases para la posterior implementación del sistema informático.
3. Se implementó una aplicación informática para la Gestión de Información de los Recursos Humanos y las Actividades de las Coordinaciones Regionales de Prevención del Delito.

RECOMENDACIONES

1. Se recomienda que se integren las aplicaciones que gestionan la información relacionada con los “Bienes Materiales, Programas y Proyectos”; “Denuncia, Comunidad y Citas” con la aplicación desarrollada en el marco de la presente investigación (“Recursos Humanos y Actividades”) como un sistema general.
2. Se recomienda que se centralice la base de datos.
3. Se recomienda que se le agreguen funcionalidades a la aplicación para mejorar la gestión de los Recursos Humanos y las Actividades.

REFERENCIAS BIBLIOGRAFICAS

1. SlideShare Inc. All rights reserved, 2009, disponible en: http://www.slideshare.net/ivonnetrabajosocial/gestion-de-la-informacion-sigeuis-3477938, consultado el 2 de marzo de 2010.
2. Grupo de investigación eumednet (SEJ-309) de la Universidad de Málaga, 2009 disponible en: http://www.eumed.net/libros/2009d/620/SISTEMAS%20DE%20GESTION%20DE%20INFORMACION%20EN%20INSTALACIONES%20HOTELERAS.htm , consultado el 2 de marzo de 2010.
3. Cátedra de Organización y Métodos Administrativos, 2008, disponible en: http://www.ccee.edu.uy/ensenian/catoym/material/sist_info_comp.pdf, consultado el 2 de marzo de 2010.
4. Piattini Velthuis, Mario (1996), Metodologías de desarrollo de software. Disponible en**:**

   http://www.itba.edu.ar/archivos/secciones/c19-icie99-ingenieriasoftwareeducativo.pdf, consultado el 1 de marzo de 2010.
5. Jacobson, Booch, Rumbaugh, El Proceso Unificado de Desarrollo de Software. Disponible en:

   http://biblioteca.uci.cu/, consultado el 25 de febrero de 2010.
6. Costales, Guirola, 2007. Trabajo de diploma: Predicción de actividad anticancerígena de compuestos orgánicos partiendo de descriptores, utilizando programación genética.
7. Ayuda extendida de Rational Rose Enterprise Edition 2003, consultado el 26 de febrero de 2010.
8. Visual Paradigm International. 2005, disponible en: http://www.visual-paradigm.com/product/vpuml/provides/, consultado el 28 de febrero de 2010.
9. Programación Java. 2009, disponible en: http://www.lenguajes-de-programacion.com/programacion-_java.shtml , consultado el 2 de marzo de 2010.
10. Ciberaula International Training, S.L, 2010, disponible en: http://www.ciberaula.com/curso/java2/, consultado el 8 de marzo de 2010.
11. Msc. Javier J. Gutiérrez, disponible en:

    http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf, 2009**,** consultado el 11 de marzo de 2010.
12. López Razo, José Alberto (2006), Ajax utilizando Dojo, disponible en: http://www.gruslic.org.mx/component/option,com_remository/Itemid,26/func,showdown/id,21/ , consultado el 5 de marzo de 2010.
13. Sánchez Rico, Mario Alfredo, Spring, un framework de aplicación**,** disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_r_ma/capitulo3.pdf, consultado el 5 de marzo de 2010.

14. Héctor Suárez González, 2003. Manual de Hibernate, disponible en: http://www.javahispano.org/contenidos/archivo/77/ManualHibernate.pdf , consultado el 15 de marzo de 2010.
15. Miguel Sanchiz, profesor titular de la Universidad Jaume I, España, disponible en: http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php, 2004, consultado el 10 de marzo de 2010.
16. PosgreSQL-es.org - Rafael Martínez, 2010, disponible en: http://www.postgresql-es.org/sobre_postgresql, consultado el 2 de marzo de 2010.
17. Notas técnicas de java. 2003, disponible en: http://www.teknoda.com/tips/java/java01.pdf , consultado el 3 de marzo de 2010.
18. The Eclipse Foundation, 2010, disponible en: www.eclipse.org , consultado el 5 de marzo de 2010.
19. Martínez Hernández, Lafaurie Olivares. Sistema para la Generación de Reportes en la plataforma alasGRATO. Consultado el 20 de marzo de 2010.
20. Aleksander González, Marizé Mijares, Luis E. Mendoza, Anna Grimán, María Pérez, LISI, Universidad Simón Bolívar, 2010, disponible en: http://www.lisi.usb.ve/publicaciones/03%20evaluacion/evaluacion_09.pdf , consultado el 2 de mayo de 2010.
21. Nicolás Tedeschi, 2010, disponible en: http://msdn.microsoft.com/es-es/library/bb972240.aspx consultado el 3 de mayo de 2010.
22. Universidad de Antioquia, 2009, disponible en: http://aprendeenlinea.udea.edu.co/lms/moodle/mod/resource/view.php?id=57215, consultado el 5 de mayo de 2010.
23. Programanía, Raúl Vicente, 2006, disponible en: http://www.programania.net/diseno-de-software/inyeccion-de-dependencias-con-spring/, consultado el 5 de mayo de 2010.
24. Jacobson, Booch, Rumbaugh, El Proceso Unificado de Desarrollo de Software, disponible en: http://biblioteca.uci.cu/, consultado el 6 de mayo de 2010.

BIBLIOGRAFÍA

- Ayuda extendida de Rational Rose Enterprise Edition 2003, consultado el: 26 de febrero de 2010.
- Costales, Guirola, 2007. Trabajo de diploma: Predicción de actividad anticancerígena de compuestos orgánicos partiendo de descriptores, utilizando programación genética.
- Cátedra de Organización y Métodos Administrativos, disponible en: http://www.ccee.edu.uy/ensenian/catoym/material/sist_info_comp.pdf, consultado el 2 de marzo de 2010.
- Ciberaula International Training, S.L, disponible en: http://www.ciberaula.com/curso/java2/, consultado el: 8 de marzo de 2010.
- Desarrollado por la Dirección General de Informática del Ministerio del Poder Popular para Relaciones Interiores y Justicia, http://www.mij.gov.ve/
- Grupo de investigación eumednet (SEJ-309) de la Universidad de Málaga, disponible en: http://www.eumed.net/libros/2009d/620/SISTEMAS%20DE%20GESTION%20DE%20INFORMACION%20EN%20INSTALACIONES%20HOTELERAS.htm , consultado el 2 de marzo de 2010.
- Héctor Suárez González, 2003. Manual de Hibernate. Disponible en: http://www.javahispano.org/contenidos/archivo/77/ManualHibernate.pdf , consultado el: 15 de marzo de 2010.
- Jacobson, Booch, Rumbaugh, El Proceso Unificado de Desarrollo de Software. Disponible en:

  http://biblioteca.uci.cu/, consultado el: 25 de febrero de 2010.
- López Razo, José Alberto (2006), Ajax utilizando Dojo. Disponible en: http://www.gruslic.org.mx/component/option,com_remository/Itemid,26/func,showdown/id,21/ , consultado el: 5 de marzo de 2010.
- Martínez Hernández, Lafaurie Olivares. Sistema para la Generación de Reportes en la plataforma alasGRATO. Consultado el: 20 de marzo de 2010.
- Notas técnicas de java. Disponible en: http://www.teknoda.com/tips/java/java01.pdf , consultado el: 3 de marzo de 2010.
- Nicolás Tedeschi, 2010, disponible en: http://msdn.microsoft.com/es-es/library/bb972240.aspx consultado el 3 de mayo de 2010.
- PosgreSQL-es.org - Rafael Martínez, disponible en: http://www.postgresql-es.org/sobre_postgresql, consultado el: 2 de marzo de 2010.
- Piattini Velthuis, Mario (1996), Metodologías de desarrollo de software. Disponible en:

http://www.itba.edu.ar/archivos/secciones/c19-icie99-ingenieriasoftwareeducativo.pdf, consultado el: primero de marzo de 2010.

- Programación Java. Disponible en: http://www.lenguajes-de-programacion.com/programacion-java.shtml , consultado el: 2 de marzo de 2010.
- Programanía, Raúl Vicente, 2006, disponible en: http://www.programania.net/diseno-de-software/inyeccion-de-dependencias-con-spring/, consultado el 5 de mayo de 2010.
- SlideShare Inc. All rights reserved, 2009, disponible en: http://www.slideshare.net/ivonnetrabajosocial/gestion-de-la-informacion-sigeuis-3477938, consultado el 2 de marzo de 2010.
- Sánchez Rico, Mario Alfredo, Spring, un framework de aplicación**,** disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_r_ma/capitulo3.pdf, consultado el: 5 de marzo de 2010.
- The Eclipse Foundation**,** disponible en: www.eclipse.org , consultado el: 5 de marzo de 2010.
- Universidad de Antioquia, 2009**,** disponible en: http://aprendeenlinea.udea.edu.co/lms/moodle/mod/resource/view.php?id=57215, consultado el 5 de mayo de 2010.
- Visual Paradigm International, disponible en: http://www.visual-paradigm.com/product/vpuml/provides/, consultado el: 28 de febrero de 2010.

ANEXOS

Anexo 1:

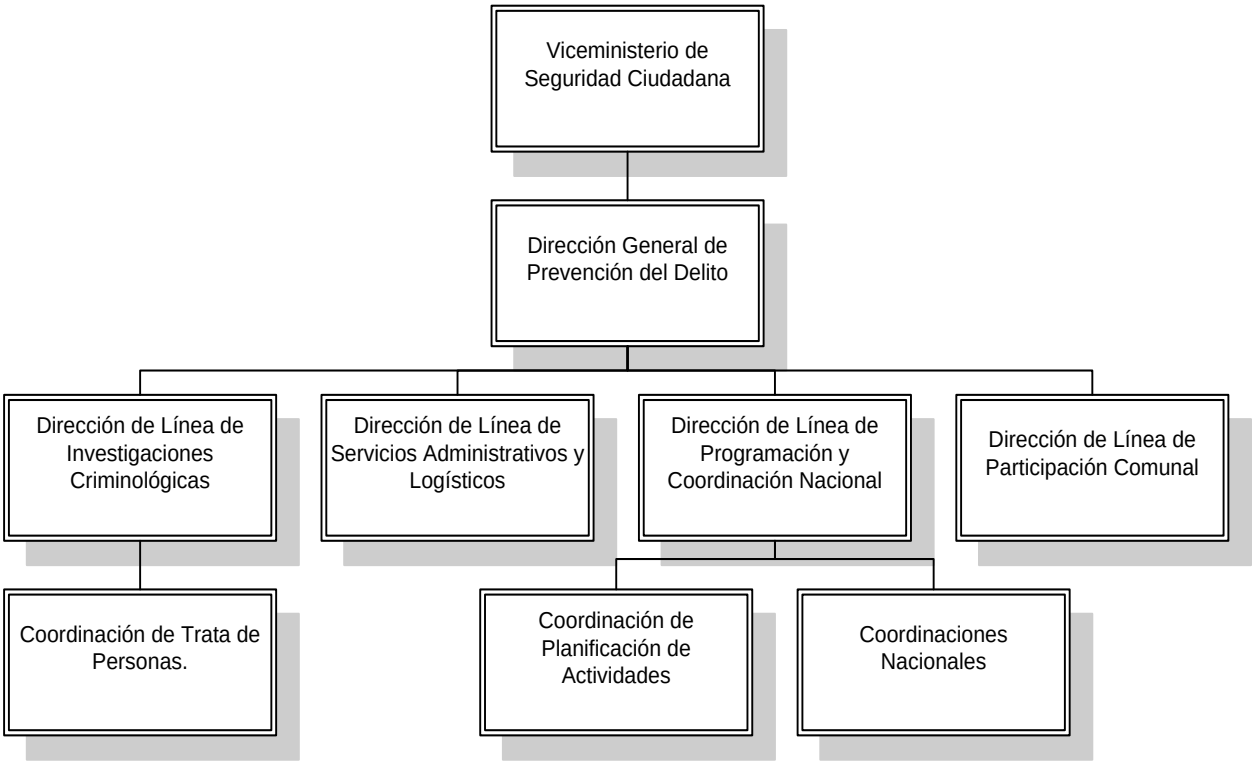


Figura 1: Organigrama de la Dirección General de Prevención del Delito.

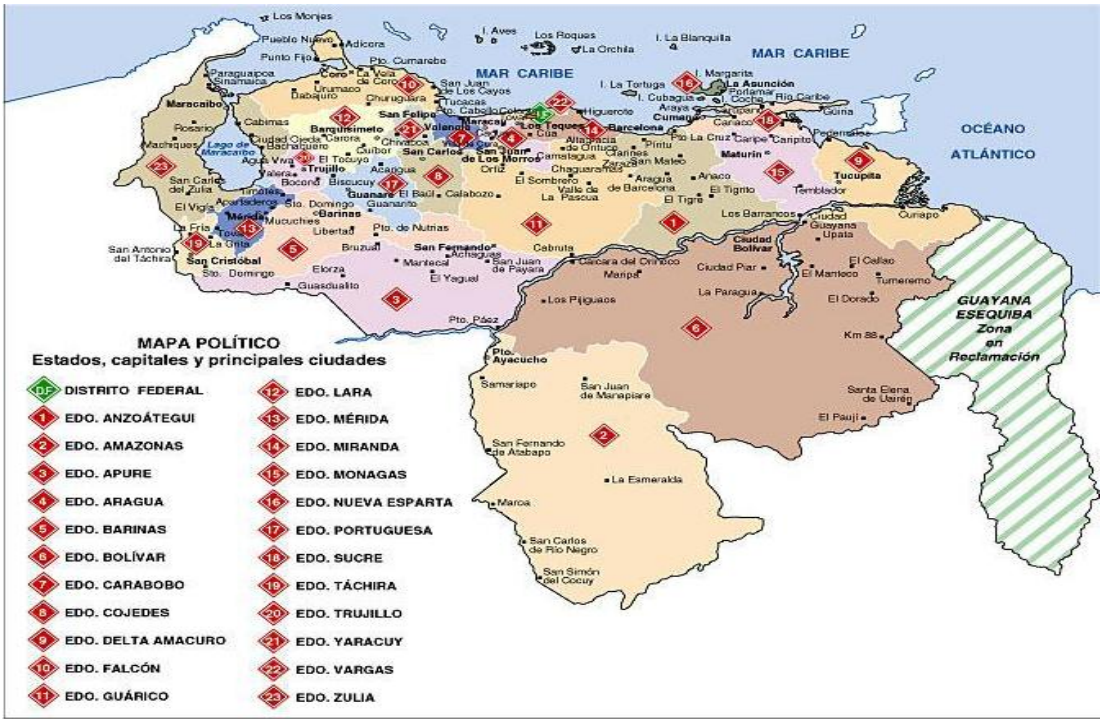


Figura 2. Coordinaciones regionales.

Figura 3. Proceso de desarrollo de Software.

GLOSARIO

API: Interfaz de programación de aplicaciones.

AOP: Programación Orientada a Aspectos.

BSD: Son las iniciales de Berkeley Software Distribution (en español, Distribución de Software Berkeley) y se utiliza para identificar un sistema operativo derivado del sistema Unix nacido a partir de los aportes realizados a ese sistema por la Universidad de California en Berkeley.

GNU: Licencia Pública General de GNU, está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

IDE: Entorno de desarrollo integrado.

J2EE: Java 2 Enterprise Edition.

JSP: Java Server Pages.

MPPRIJ: Ministerio del Poder Popular para Relaciones Interiores y Justicia de la República Bolivariana de Venezuela.

MVC: Modelo Vista Controlador**.**

ORM: Object Relational Mapping.

RUP: Rational Unified Process**.**

SGBD: Sistema Gestor de Base de Datos.

UML: Lenguaje Unificado de Modelado.