Universidad de las Ciencias Informáticas

Facultad 2

Título: Desarrollo de los módulos Visitas Familiares y Visitas Institucionales del Sistema Informativo de la Dirección de Establecimientos Penitenciarios (SIDEP).

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

Autor(es):
Zulai Silva Reyes
Yasel Hernández Marin

Tutor:
Ing. Dasiel Cordero Morales

Co-Tutor:
Ing. Linet Lores Sanchez

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 _____.

Zulai Silva Reyes

_____

Firma de la autora

Yasel Hernández Marin

_____

Firma del autor

Ing. Dasiel Cordero Morales

_____

Firma del tutor

Agradecimientos

A mi familia, mis padres, a mi tía Migdalia y demás tíos, mis primos, mi hermana y a mi sobrinita a todos por estar siempre conmigo.

A mi mamá y mi tía, por darme las fuerzas necesarias en los momentos en que más las necesité y apoyarme durante todos estos años de estudios y confiar en mí.

A Manolo que ayudo a mami a formarme durante todo este tiempo, y es un ejemplo a seguir.

A Made por su apoyo en todo momento y por su amistad incondicional.

Agradecer a mi novia y compañera de tesis Zulai la cual ha estado estos cinco años apoyándome y dándome fuerza sin la cual no hubiera podido llegar hasta aquí.

A mi tutor, Dasiel, por su preocupación y sus consejos, su ayuda, su ejemplo, sus críticas constructivas durante todo este proceso de elaboración de la tesis.

A todos los miembros del proyecto, que cada cual aportó un granito de arena para construir este sueño. Especialmente a mis amigos del grupo. Yaidel, Apa, Dariel, Rene, Aldo y todos aquellos que de una o de otra forma han estado cuando los he necesitado.

A todas aquellas personas que me ayudaron de una forma u otra y se preocuparon siempre por mi bienestar y por mi dedicación a la carrera.

Yasel

A mi familia. Por todo el apoyo y la confianza que han depositado en mí durante toda la carrera.

A mami y papi por ser el motor impulsor de todos mis logros y apoyarme siempre.

A mis niños Fani y Yin que siempre tienen un abrazo y una sonrisa para mí.

A mi abuela Maira y mi tía Mayi por ser mis mayores consejeros y estar por estar presentes en cada momento de mi vida y a Yosvany por tener siempre una palabra de aliento para mí.

A mi abuela Lucia, mi tía Rosi y a Yani por cada granito de arena aportado su en este sueño y su apoyado incondicional.

A Nana y a Tata por incluirme en su lista de nietos y tratarme como tal.

A mi novio Yasel por soportarme y comprenderme, por todo el cariño y el amor que me ha bridado, por estar a mi lado dándome su apoyo. Te amo.

A mi tutor, Dasiel por guiarme y prestarme su ayuda.

A todos mis compañeros que durante el transcurso de mis años en la UCI me han soportado y sobre todo ayudado.

Zulai

Dedicatoria

En primer lugar este trabajo va dedicado a mi mamá Gisela que sin su ayuda y apoyo no hubiera podido terminar, por cumplir cada uno de mis sueños a lo largo de mi vida y por su sacrificio y dedicación para que llegara este día.

A mi tía Migdalia por su cariño y apoyo todo estos años, por sus dulces y por creer siempre en mi

A mi novia Zulai dedico este trabajo por su total entrega al trabajo y a mí.

A toda la familia que de una o de otra forma siempre han estado cuando los he necesitado.

A mi hermana y mi sobrinita para que sirva de estímulo a seguir adelante y no abandonar sus sueños.

Dedicatoria especial a Manolo que aunque no pudo ver este día sé que se siente orgulloso de mí en donde quiera que esté.

A todos mis abuelos, que siempre me entregaron su amor.

Yasel

A mi hermanita Fany espero servirle de ejemplo para su vida.

A mi mama que ha hecho posible todos mis sueños.

A mi novio Yasel por su entrega en cada momento.

A mi tío Derys me aconsejó durante toda su vida y sé que siente orgullo de mí y de mis logros.

A toda mi familia que han aportado su granito de arena en este sueño.

Zulai

Resumen

En la actualidad la proliferación de las Tecnologías de la Informática y las Comunicaciones (TIC) han permitido el desarrollo de tecnologías para informatizar todos los sectores de la sociedad. En Cuba el Ministerio del Interior (MININT) en conjunto con la Universidad de las Ciencias Informáticas (UCI) se han dado a la tarea de informatizar los procesos referentes al Sistema Penitenciario Nacional. Dicho sistema cuenta en la actualidad con el Sistema Automatizado para el Control de Recluso (SACORE) que a pesar de que se ha ido perfeccionando con los años, aún no satisface las demandas de los usuarios.

El presente trabajo de diploma consiste en el desarrollo de los módulos Visitas Familiares y Visitas Institucionales del Sistema de Información para la Dirección de Establecimientos Penitenciarios (SIDEP). Su objetivo principal es lograr una mejor gestión y control de los procesos referentes a las Visitas. Dando cumplimiento durante el desarrollo del mismo a las tareas y objetivos planteados, cumpliendo con los requerimientos del cliente y haciendo uso de las tecnologías y herramientas definidas en el proyecto.

Palabras claves: Sistema Penitenciario Nacional, SACORE, Visitas Familiares e Institucionales, MININT, SIDEP.

Índice de contenidos

Índice de tablas

Índice de figuras

Introducción.

En la era moderna, el aumento y proliferación de las Tecnologías de la Informática y las Comunicaciones (TIC) han permitido la creación de herramientas y sistemas informáticos para automatizar diferentes procesos, la mayoría de los países utilizan estas tecnologías con el fin de informatizar su sociedad. Cuba no ha quedado retrasada en este sentido pues; desde hace algún tiempo en el país, a nivel social se han informatizado un gran número de sectores, entre los que se encuentran: la salud, la educación, el deporte, el turismo, la economía, las Fuerzas Armadas Revolucionarias (FAR), el Ministerio del Interior (MININT) y dentro de este último el Sistema Penitenciario.

Antes del triunfo de la revolución en 1959, el sistema penitenciario en Cuba se caracterizaba por la corrupción y el tratamiento brutal al recluso en deterioro de su integridad y dignidad humana. Desde ese mismo año se comenzó un proceso de transformaciones con el fin de renovar el sistema penitenciario existente en aquel entonces y sustituirlo por uno basado en el respeto y el control riguroso de las leyes, reglamentos y políticas con el fin de reeducar y rehabilitar a cada persona recluida para su reinserción social. El sistema penitenciario cubano incluye entre sus pilares principales:

- La adopción y perfeccionamiento del sistema progresivo. El recluso avanza en diferentes regímenes penitenciarios hasta lograr su libertad condicional, a partir de su conducta y de plazos mínimos de cumplimiento de su sanción.
- El establecimiento de criterios de clasificación de la población penal que aseguran mejor tratamiento colectivo e individualizado (a partir de situación legal, sexo, edades, nacionalidad, características personales, niveles de peligrosidad, etc.).
- La construcción de locales adecuados para los establecimientos penitenciarios (celdas colectivas e individuales, con aire, luz, ventilación, servicios sanitarios y duchas).
- El otorgamiento de ayudas económicas a familiares de reclusos y de la Seguridad Social a los reclusos.
- La organización de un subsistema de atención médica y estomatológica, primaria y especializada, para la atención a reclusos.
- El desarrollo de actividades educativas, artísticas, deportivas y recreativas.
- La capacitación técnica y profesional y la superación permanente del personal penitenciario (juristas, psicólogos, pedagogos, defectólogos, sociólogos y funcionarios). (1)

Sobre las bases anteriormente expuestas y en correspondencia con las transformaciones desarrolladas en el sistema penitenciario, comienza en 1989 en Cuba el proceso de informatización de los centros penitenciarios, la cual tiene como principal objetivo la automatización de los principales datos del recluso y ciertos aspectos de control penal, pero no es hasta algunos años más tarde que se crea el un Sistema Automatizado para el Control del Recluso (SACORE), el cual comenzó a dar respuesta en gran medida a muchos de los problemas existentes en el momento. Este sistema culminó su desarrollo a finales del 2002, poniéndose en marcha a principios del 2003.

El SACORE se ha ido perfeccionando a partir de los requerimientos y solicitudes de los usuarios. Contiene como subsistemas principales Control Legal, Tratamiento Educativo y Orden Interior lo cual permite recoger los principales datos del recluso y los aspectos penales existentes. Este sistema contiene muchas de las funcionalidades necesarias para el trabajo con los internos en los centros penitenciarios, pero a pesar de llevar varios años de explotación aún quedan pendientes un grupo de requisitos que el sistema actual no cubre.

En el año 2009 la Jefatura del MININT en colaboración con la Universidad de las Ciencias Informáticas decide crear el proyecto Prisiones Cuba el cual tiene como objetivo informatizar el Sistema Penitenciario Nacional y automatizar los procesos de trabajo dentro de los centros penitenciaros mediante el Sistema Informativo de la Dirección de Establecimientos Penitenciarios (SIDEP), el cual cuenta con varios subsistemas divididos en módulos. Uno de los subsistemas lo constituye Seguridad Penitenciaria, que recoge aspectos del Orden Interior y la Seguridad Penal, siendo el encargado del control físico, la vigilancia y el orden disciplinario de los internos.

En este subsistema se recogen, además aspectos relacionados con las visitas realizadas al centro penitenciario, estas pueden ser familiares o institucionales. En los centros penitenciarios existen áreas destinadas para la organización y ejecución de dichas visitas. Estos constituyen locales a los cuales asiste los colectivos designados para la realización de la visita.

La visita familiar se establece como un derecho en el Sistema Penitenciario y la frecuencia de realización de las mismas está en dependencia del régimen de cumplimiento de las sanciones. El régimen es una clasificación que reciben los internos en dependencia del delito cometido, estos se pueden clasificar en: Mayor Severidad, Severo, Media Severidad y Mínima Severidad.

Las visitas institucionales constituyen visitas que realizan las instituciones al centro penitenciario, tal es el caso de jueces, fiscales, abogados, funcionarios y representantes de organismos de la administración central del estado y cualquier otra institución que solicite una visita.

El sistema existente (SACORE) a pesar de llevar varios años de explotación, no responde a las necesidades de los usuarios referentes a los procesos de visitas familiares e institucionales, pues no contiene todas las funcionalidades para la correcta gestión y control de los procesos de Visita dentro de los establecimientos penitenciarios. Este sistema no permite realizar la configuración de la visita en dependencia del centro penitenciario y no presenta la opción de crear planes de visita de forma automática. No existen funcionalidades que permitan registrar la información perteneciente a las visitas institucionales, como las consulares y de la comunidad, y en el caso de las visitas familiares, no se registran las visitas fuera del plan. Por todo lo anteriormente expuesto se hace necesario el perfeccionamiento de funcionalidades existentes y la inclusión de nuevos requisitos que automaticen aún más el trabajo del personal del Sistema Penitenciario Cubano.

Analizando la situación anteriormente expuesta, se impone el siguiente **problema a resolver**: ¿Cómo contribuir a la gestión de la información relacionada con los procesos de visitas familiares e institucionales del sistema penitenciario cubano?

Teniendo en cuenta lo anterior planteado se define como **objeto de estudio** la gestión de la información en los establecimientos penitenciarios, trazándose como **objetivo general** desarrollar los módulos Visitas Familiares y Visitas Institucionales para gestionar los procesos de visitas del Sistema Penitenciario Cubano, quedando enmarcado como **campo de acción** la gestión de las visitas familiares e institucionales en el Sistema Penitenciario Cubano.

La **idea a defender** durante la investigación estará definida por: El desarrollo de los módulos Visitas Familiares y Visitas Institucionales permite la gestión de los procesos de visitas familiares e institucionales en los centros penitenciarios.

Para complementar el objetivo general se han definido los siguientes objetivos específicos:

- Realizar la fundamentación teórica de la Investigación.
- Definir las funcionalidades de los módulos Visitas Familiares y Visitas Institucionales.
- Realizar el análisis y diseño de los módulos Visitas Familiares y Visitas Institucionales.
- Implementar los módulos Visitas Familiares y Visitas Institucionales.
- Realizar las pruebas de funcionalidad a los módulos Visitas Familiares y Visitas Institucionales.

De donde se conciben las siguientes tareas de la investigación:

- Análisis de soluciones similares a la que se quiere desarrollar.
- Descripción de las herramientas y metodología a utilizar en el desarrollo de los módulos Visitas Familiares y Visitas Institucionales del Sistema Penitenciario Cubano.

- Especificación de los requisitos funcionales de los módulos Visitas Familiares y Visitas Institucionales del Sistema Penitenciario Cubano.
- Estructuración del modelo del sistema para los módulos Visitas Familiares y Visitas Institucionales del Sistema Penitenciario Cubano.
- Realización del modelo de negocio de los módulos Visitas Familiares y Visitas Institucionales del Sistema Penitenciario Cubano.
- Realización del modelo de análisis de los módulos Visitas Familiares y Visitas Institucionales del Sistema Penitenciario Cubano.
- Realización del modelo de diseño para los módulos Visitas Familiares y Visitas Institucionales del Sistema Penitenciario Cubano.
- Realización del modelo de implementación de los módulos Visitas Familiares y Visitas Institucionales del Sistema Penitenciario Cubano.
- Realización de las pruebas de caja negra para los módulos Visitas Familiares y Visitas Institucionales del Sistema Penitenciario Cubano.

Para darle cumplimiento a las tareas anteriormente planteadas se utilizaron los siguientes **métodos de investigación**:

- **Analítico-Sintético:** Se utiliza en el análisis de documentos consultados, lo que permitió examinar los principales elementos relacionados con las visitas y así conocer todo lo respectivo a este proceso en el Sistema Penitenciario Cubano.
- **Histórico-Lógico:** Se utiliza para identificar las principales etapas por las que ha transitado el Sistema Penitenciario Cubano, permitiendo conocer la evolución del mismo en cuanto a los procesos de visitas y el estado actual de estos con respecto a las necesidades de los usuarios.

Este informe está estructurado en cuatro capítulos, como se describe a continuación:

Capítulo I: Fundamentación Teórica

Se realiza un estudio de algunas soluciones de software existentes relacionadas con los Sistemas Penitenciarios tanto a nivel nacional como internacional y al finalizar se hace referencia a la metodología de desarrollo de software y herramientas utilizadas durante el desarrollo de la investigación y definidas por el equipo de desarrollo del proyecto Prisiones Cuba.

Capitulo II: Características del sistema

Se describen las características del sistema, se modelan los procesos de negocio y se da una descripción de la solución propuesta, definiéndose los requisitos que deben cumplir la misma y los casos de uso del sistema.

Capítulo III: Análisis y Diseño

Se hace alusión a todo lo referente a las características del sistema, donde se plasman los requisitos definidos, se realiza la descripción de los casos de uso, se explica cómo está estructurada la arquitectura de la aplicación que se implementará, los patrones de diseño que se utilizarán. Se expone una representación gráfica de los diagramas de clases e interacción, así como el Diseño de la Base de Datos.

Capítulo IV: Implementación y Prueba

En este capítulo se generan los diagramas de despliegue y componentes, se implementa el sistema y se concluye realizando pruebas para comprobar si la aplicación cumple sus objetivos.

Capítulo I: Fundamentación Teórica.

1.1 Introducción

Este capítulo constituye la fundamentación teórica que sustenta el desarrollo del presente trabajo. Se exponen los conceptos principales del negocio en cuestión y se hace referencia a las principales características de soluciones de software relacionadas con el Sistema Penitenciario a nivel nacional e internacional. Por último se describirán las herramientas, técnicas y metodologías a utilizarse en el desarrollo de la solución que se propone.

1.2 Principales conceptos del negocio

Sistema Penitenciario: Es el encargado de garantizar el proceso de ejecución de la sanción de privación de libertad, de la sanción de trabajo correccional con internamiento, la medida de seguridad reeducativa de internamiento y la medida cautelar de prisión provisional. Es dirigido por la Dirección de Establecimientos Penitenciarios (DEP) del Ministerio del Interior, se sustenta en la integración de principios, conceptos, procedimientos, fuerzas y medios que garantizan el funcionamiento de los centros destinados al internamiento y el tratamiento a los internos (2).

Centro Penitenciario: Es la institución reconocida para el cumplimiento de la sanción perpetua o temporal de privación de libertad, se organizarán separadamente para hombres y mujeres. Pueden ser cerrados o abiertos (2).

Colectivo: Es la unidad básica fundamental del sistema penitenciario, donde conviven agrupados por los criterios de clasificación en el reglamento, los internos (2).

Interno: Denominación genérica que identifica a los acusados, sancionados o asegurados en el proceso de ejecución de la sanción de privación de libertad, medida cautelar de prisión provisional y mediada reeducativa de internamiento respectivamente (2).

Visita Familiar: Es la comunicación personal del familiar con el interno. Estas se planifican por colectivos, pueden ser:

- **Familiar:** Se planifican anualmente y asisten los familiares y amigos previamente declarados por el interno.
- **Conyugal:** Se planifican trimestralmente y asiste el conyugue del interno.
- **Al ingreso:** Constituye la primera visita que se le asigna al interno en la cual el familiar le entrega los artículos de aseo personal.

- **Fuera de plan:** Constituyen las visitas solicitadas por los familiares de un interno al Jefe de la Unidad y autorizadas por el mismo, no se encuentran dentro de la planificación de visitas del centro penitenciario.

Visita Institucional: Es la visita de instituciones al centro penitenciario, tal es el caso de jueces, fiscales, abogados, integrantes de comunidades, cónsules, así como las asociadas a un proyecto, pueden ser:

- **Consulares:** Constituyen las visitas a las que asiste el cónsul de un determinado país.
- **Institucionales:** Constituyen las visitas a las que asisten abogados, jueces, fiscales u otros integrantes de la Administración Central del Estado así como funcionarios de la DEP. Pueden estas asociadas a internos específicos o no.
- **Asociadas a un proyecto:** Constituyen las visitas que realizan diferentes proyectos, tales como culturales, deportivos o investigativos. Pueden estar asociada a internos específicos o no.
- **Comunidad:** Constituyen las visitas a las que asisten un grupo de personas perteneciente a una determinada comunidad.

1.3 Sistemas Penitenciarios

1.3.1. Soluciones nacionales

Sistema Automatizado para el Control del Recluso (SACORE)

Surge para dar cumplimiento a la Orden 43/99 del Vice-Ministro Primero del Ministerio del Interior de Cuba y con el objetivo de automatizar la gestión de los datos principales del recluso y algunos aspectos de control penal, este es el primer sistema informático que gestiona la información que generan las prisiones cubanas. Entre las principales funcionalidades que brinda están contempladas las siguientes:

- Respuestas inmediatas a las solicitudes de información de los diferentes órganos e instituciones del estado como son: Jefatura del MININT, Ministerio de Justicia, Tribunales, Fiscalías, entre otros.
- Los partes que se emiten son obtenidos de forma automatizada.
- Se recogen datos de medidas disciplinarias, visitas, ocupaciones de objetos no permitidos por lo establecido en el reglamento y requisas, entre otros (3).

Aunque este sistema está actualmente en uso en Cuba, no están totalmente informatizados y organizados los procesos de visitas familiares así como los procesos relacionados con las visitas institucionales como son el control de acceso, el registro de visitantes institucionales así como la

consulta de los detalles de las visitas realizadas al centro. Además, este sistema no incluye la realización de los planes de visitas familiares de forma automática, por lo que no cumple con las características necesarias que le permiten gestionar las visitas en los centros penitenciarios cubanos.

1.3.2. Soluciones extranjeras:

Sistema de Gestión Penitenciaria (SIGEP)

SIGEP es una aplicación desarrollada en Cuba, específicamente en la Universidad de las Ciencias Informáticas, en conjunto con el ministerio del interior para el Sistema Penitenciario venezolano, este se crea para dar respuesta a las necesidades de gestión, información y apoyo a la toma de decisiones de la Dirección Nacional de Servicios Penitenciarios (DNSP). Este sistema permite a los establecimientos penitenciarios recopilar y controlar la información que se genera en este tipo de centros. Presenta numerosas funcionalidades entre las que están, la gestión de ingresos, de permisos y solicitudes así como un módulo que se encarga de la gestión y control del proceso de visita.

El análisis de este sistema arroja que el mismo no se considera posible solución pues procesos como el control de acceso, la planificación de visitas familiares, la creación de calendarios y el registro de visitantes se realizan de forma diferente a como se realizan en el Sistema Penitenciario Nacional, se necesita un sistema que genere los planes de visita a partir de la configuración de las visitas previamente confeccionada y que permita la gestión de la información referente a los procesos de visita de la forma requerida por el Sistema Penitenciario cubano.

OFFENDERTRAK

Es una solución de software empresarial proporcionada por Motorola, que ofrece una solución completa para las necesidades operativas y administrativas del centro penitenciario de hoy. Permite la captura, almacenamiento y recuperación de todos los datos de consumo de los reclusos a través de la liberación. El sistema incluye un menú completo de todos los módulos que abordan los aspectos de las cárceles de hoy en día y centros penitenciarios, proporcionando una base de datos centralizada del infractor y los datos de las instalaciones (4).

Este sistema presenta como inconveniente que es propietario y Para ser utilizado requiere pagar elevado costo. Además en la documentación consultada no se encontró ninguna evidencia que demostrara la gestión de las visitas en este software, por lo que no se considera como posible solución a la problemática planteada.

Elite Jails

Elite Jails es una solución genérica proporcionada por Syscon. Contiene un modelo de seguridad basado en roles para garantizar que cada usuario solo tenga acceso a la información que tenga autorización. Puede ser personalizado para satisfacer los requisitos de un negocio específico (5). Este sistema permite:

- Controla el ingreso o el reingreso de un interno en el sistema, mantiene un historial penal central de todos los presos, incluyendo datos demográficos detallados.
- Graba imágenes únicas o múltiples de los delincuentes (la cara, de perfil, las señales físicas). La imagen captada se almacena automáticamente con todos los datos de otro delincuente en un registro.
- Registra detalles de los incidentes, tales como el tipo de incidente, los delincuentes y personal involucrado, la ubicación, y las medidas adoptadas (5).

Este sistema contiene además funcionalidades que gestionan las visitas del interno, los horarios de estas y una lista de visitantes autorizados. El análisis del mismo arrojó que no constituye una posible solución a la problemática planteada porque no permite generar planes de visita. En la documentación consultada sobre este sistema no se encontraron aspectos relacionados con el registro de visitas por estímulo y fuera del plan, además de las relacionadas con visitas institucionales, tales como: consulares, de la comunidad y las asociadas a un proyecto, por lo que no constituye una posible solución al problema planteado

Spillman Corrections Management System/Sistema de Gestión Correcional Spillman

Spillman Corrections Management System es un software de gestión penitenciaria que desarrolló la empresa Spillman y se utiliza en algunos centros penitenciarios de los Estados Unidos. Cuenta con más de 40 módulos que permiten personalizar un sistema de gestión penitenciaria para satisfacer las necesidades individuales de las instalaciones. En la esfera de la comunicación permite compartir información con entidades que no pertenecen al centro penitenciario como los departamentos de policía, los centros de comunicación y departamentos de bomberos para la interoperabilidad total de la seguridad pública (6).

En la información disponible en el sitio oficial de la compañía Spillman, referente al producto Spillman Corrections Management System no se menciona nada referente al manejo de las visitas, por lo que este no se considera como posible solución a la problemática planteada.

Sistema Penitenciario del Gobierno de Panamá

El Sistema Penitenciario del Gobierno de Panamá fue creado a principios de año 1997, como resultado de un proyecto financiado por las Naciones Unidas y el gobierno español en coordinación con la Dirección General de Sistemas Penitenciarios de Panamá (DGSP). Este software tiene como objetivo mantener en una base de datos los registros de los internos que están detenidos en los centros penales a nivel nacional. Entre estos registros se encuentran los datos personales del interno, jurídicos, antecedentes médicos y los datos socioeconómicos (7).

La información se registra en una base de datos radica en la sede central y la cual está disponible para los diferentes departamentos que tiene acceso al sistema. Esto garantiza que se refleje de forma inmediata los cambios en el expediente del interno tales como traspasos de autoridad, traslados de un centro a otro, libertades, diligencias médicas, jurídicas y la realización del cómputo automático de la sentencia (7).

Este sistema contiene funcionalidades para la gestión de las visitas, permite registrar los datos de las visitas familiares y conyugales para las cuales el interno declara una lista de visitantes que pueden asistir a estas visitas, pero no se especifica la forma en que se realiza la gestión de las visita de familiares y ni de ningún aspecto relacionado con las visitas institucionales por lo que no constituye una posible solución a la problemática planteada.

Jail Management System (JMS)/Sistema de Gestión de Cárcel

El JMS es una solución genérica que pertenece al Software Development and Services Corporation (SDSC) el cual tiene como misión el desarrollo e implementación de sistemas que gestionan información de justicia. Este sistema proporciona una gestión para el registro de los datos de internos. Incluye los datos demográficos del interno y permite a los usuarios capturar, recuperar y visualizar fotos de los mismos (8)

El JMS incluye funcionalidades relacionadas con la programación de las visitas. Permite registrar la entrada y salida de los visitantes, así como guardar y vincular a los visitantes frecuentes con los reclusos.

El análisis realizado a este sistema arroja que el mismo no constituye una posible solución porque a pesar de contener funcionalidades similares a los elementos de la problemática planteada, la gestión de los procesos de visitas familiares se realiza de manera diferente, debido a que en este sistema la relación de los visitantes con los internos se realiza por la frecuencia de visitas de estos al centro, mientras que en el sistema penitenciario cubano el interno define previamente a la realización de la

visitas la relación de vínculos a los cuales designa como posibles visitantes. Por otra parte en la documentación consultada acerca de este sistema no se describen aspectos relacionados con las visitas institucionales.

Sistema de Identificación Automatizado (SIA).

El SIA es un sistema español que permite con rapidez la verificación de la identidad de los privados de libertad del Sistema Penitenciario de España. Permite la captura de las reseñas dactilares, fotográficas, verificación de la identificación, y demás datos de filiación, procesales y penales, además de la actualización de todos los datos previstos en la ficha de los internos. (9).

Este sistema está diseñado fundamentalmente para el control documental legal y los datos de los internos, en el análisis del mismo no se encontraron funcionalidades referentes al registro y control de los visitantes a los centros penitenciarios, por lo que no constituye una posible solución porque no responde a la problemática planteada.

Jail Administration and Management System (JAMS). /Sistema de Gestión y Administración de Cárcel

JAMS es una solución genérica para la gestión de la información sobre los encarcelados. Proporciona informes fiables sobre todos los aspectos de la gestión de la cárcel. Contiene un módulo relacionado con las visitas y mantiene dos tipos de registros relacionados con estos módulos. La primera es una lista de visitantes para cada interno que contiene los nombres, números de teléfono y la relación. El segundo es un registro de visitantes que contiene las transacciones de todas las visitas a los reclusos. La información incluye el identificador del interno asignado, un número para cada visita que se genera automáticamente, el nombre del interno, el nombre del visitante, la fecha de visita, hora de inicio y finalización, y una línea de comentarios de la naturaleza de la visita (10).

El análisis realizado a este sistema arroja que tiene características similares al sistema penitenciario cubano, pero no satisface las necesidades respecto a la gestión de las visitas, debido a que no se generan planes de visitas familiares y no se registran visitas por estímulo. Además en la documentación consultada no se encontró ningún aspecto referente a las visitas institucionales, por lo que no constituye una posible solución.

La investigación realizada a los sistemas automatizados existentes, mostró que dichos sistemas brindan la posibilidad de identificar a cada uno de los individuos que ingresan al centro penitenciario, registrando sus datos generales. El objetivo de dichos sistemas es facilitar el control de los reclusos y gestionar los procesos esenciales en los establecimientos penitenciarios.

El análisis realizado a cada uno de los sistemas mencionados anteriormente así como a las funcionalidades que brindan arroja que no constituyen una solución posible debido a que sus características no se adaptan a las necesidades del sistema penitenciario cubano porque no gestionan todos los procesos que intervienen en las visitas familiares y no contienen funcionalidades para registrar los datos de las visitas institucionales que responden a la problemática planteada. Por otra parte estos sistemas son privativos por lo que para poder utilizar sus servicios es necesario pagar un alto precio.

Se requiere una aplicación que soporte la información que se genera como parte de la realización de las visitas dentro del sistema penitenciario; que permita la configuración de las visitas en dependencia del centro penitenciario, la planificación de las visitas familiares de los internos, así como el registro de la entrada y salida de visitantes, además tener un control de las visitas de instituciones registrando los datos de cada uno de los visitantes. También que admita consultar detalles de las visitas realizadas, planificadas o que se encuentren en ejecución dentro del centro penitenciario.

1.4 Metodología, herramientas y tecnologías utilizadas en el desarrollo.

La metodología, herramientas y tecnologías definidas para la construcción del sistema, son producto de un estudio realizado por el equipo de Arquitectura del proyecto Prisiones Cuba, y establecidas como políticas del proyecto, por lo que la selección de las mismas queda fuera del alcance del presente trabajo.

1.4.1. Metodología de desarrollo de software

Rational Unified Process (RUP)

RUP es una metodología de desarrollo de software cuyo objetivo es lograr un software de eficiencia y calidad. Es muy factible para el desarrollo de soluciones informáticas complejas y extensas en cronogramas de ejecución.

RUP se utilizó como guía durante todo el proceso de desarrollo del sistema, pues posee ventajas en cuanto a la generación de la documentación durante todo el ciclo de desarrollo del software facilitando el desarrollo en los constantes cambios y la fácil interpretación de sus artefactos a través de UML. La aplicación de estos aspectos se evidencia en la confección de los casos de uso para especificar los requisitos del sistema que son los encargados de guiar el ciclo de vida del proyecto.

1.4.2. Lenguaje de modelado

Lenguaje Unificado de Modelado.

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software (11). Su finalidad es describir modelos de sistemas, compuestos por artefactos, diagramas y documentos.

Durante el desarrollo del sistema se utilizaron los diagramas definidos por el lenguaje UML para mostrar diferentes aspectos de las entidades representadas. Además permite confeccionar y documentar los artefactos que permite confeccionar y documentar en el ciclo de desarrollo, tales como diagramas de caso de uso para mostrar la interacción entre las funcionalidades y los actores del sistema, diagramas de clases para comprender la estructura de la aplicación mostrando sus clases, atributos y las relaciones entre ellos y diagramas de componentes para mostrar la vista física del sistema y las dependencias entre los componentes del mismo.

1.4.3. Herramientas de modelado

Business Process Modeling Notation (BPMN)

La Notación para el Modelado de Procesos del Negocio (*Business Process Modeling Notation*, BPMN) es un estándar de modelado de procesos de negocio, su principal objetivo es proporcionar una notación entendible por todos los involucrados en el negocio (12). La utilización de BPMN permitió explicar gráficamente las diferentes etapas de los procesos de visitas. Para el modelado de los procesos se realizaron diagramas en los que se definieron las relaciones entre los actores y las actividades que realiza cada uno. El modelado de los procesos de negocio permitió comprender la forma de cómo se desarrollan las visitas en el sistema penitenciario cubano definiendo el flujo de las actividades realizadas y los usuarios responsables.

Visual Paradigm for UML 5.0

Visual Paradigm es una herramienta fácil de instalar, actualizar y usar. Soportar el ciclo de vida completo del proceso de desarrollo del software y permite la representación de todo tipo de diagramas. Soporta UML y BPMN lo que permite la documentación y generación de los artefactos correspondientes a cada fase de la metodología RUP. Se utiliza esta herramienta durante el ciclo de vida del desarrollo de software para generar los artefactos pertinentes en las fases de modelación de los procesos del negocio, análisis, diseño e implementación.

Axure RP 5.5

Axure RP es una herramienta muy factible para generar prototipos de usabilidad. La utilización de esta herramienta permitió crear los diferentes prototipos en forma de páginas web posibilitando la utilización del código HTML generada en estas páginas en la implementación del sistema. Estos prototipos

forman parte de las descripciones de los casos de uso y permiten mostrarle al usuario la interfaz del futuro sistema logrando realizar una validación de los requisitos, la cual permitió conocer que el diseño de la propuesta de solución esté acorde a lo que necesita el cliente.

ER/Studio 8.0.0

Esta herramienta permite diseñar y construir bases de datos a nivel físico y lógico. Además permite la transformación automática de un diseño lógico a un diseño físico para una plataforma específica de base de datos y permite comparar el diseño con la estructura real física de las tablas. La utilización de esta herramienta permitió la confección del diagrama entidad-relación de los módulos de visitas en el cual contiene las relaciones entre las tablas que intervienen en el modelo de datos, dicho diseño permitió la normalización de la base de datos.

1.4.4. Herramientas de desarrollo

NetBeans 7.0

Es una herramienta para programadores en la que se puede escribir, compilar, depurar y ejecutar programas. Está escrito completamente en Java usando la plataforma NetBeans, pero puede servir para cualquier otro lenguaje de programación. Tienen integración con el framework de desarrollo Grails y trae incluido el uso de Groovy. Este lenguaje de programación es el utilizado para la implementación de los módulos de visita.

Apache Tomcat 6.0

Es un contenedor Web escrito en Java, por lo que funciona en cualquier sistema operativo que disponga de una máquina virtual Java. Es útil para el desarrollo y despliegue de aplicaciones y servicios web. Este viene incluido como módulo plugin en Grails y posee como características principales.

- **Rápido y ligero:** Tomcat es un contenedor ligero y altamente optimizado.
- **Flexible y escalable:** Ejecuta sus aplicaciones con mayor rapidez. Permite construir de abajo hacia arriba infraestructuras altamente personalizados a las necesidades del servicio. (13)

1.4.5. Sistema Gestor de Base de Datos

Oracle 11g

Es un Sistema Gestor de Bases de Datos con características objeto-relacionales, utilizando tecnología cliente/servidor. Permite la gestión de grandes bases de datos, se caracteriza por su estabilidad y

escalabilidad, es multiplataforma. Se utiliza para almacenar los datos necesarios para el correcto funcionamiento de la aplicación.

1.4.6. Tecnologías

Grails 1.3.7

Grails es un framework web de código abierto compatible con Java. Utiliza el lenguaje dinámico Groovy y los frameworks de código abierto Hibernate (para mapeo objeto-relacional) y Spring (para inyección de dependencias).

Grails se basa en dos principios fundamentales:

- **Convención mejor que configuración:** En lugar de tener que escribir archivos de configuración en formato XML, Grails se basa en una serie de convenciones para que el desarrollo de la aplicación sea mucho más rápido y productivo. Por ejemplo, todas las clases de la carpeta grails-app/controllers serán tratadas como Controladores, y se mapearán convenientemente a las urls de la aplicación (14).
- **DRY: Don't repeat yourself (¡No te repitas!):** La participación de Spring Container en Grails permite la inyección de dependencias mediante el uso de inversión de control, de forma que cada actor en la aplicación deba definirse una única vez, haciéndose visible a todos los demás de forma automática (14).

Grails al estar integrado con un lenguaje de programación dinámico permite la utilización de sintaxis de otros lenguajes. Además mediante la inyección de dependencias permite la reutilización del código y el uso de las convecciones hace más fácil y productivo el desarrollo, debido a que no es necesario configurar la aplicación mediante ficheros XML. Genera la Base de Datos a partir del dominio mediante la utilización del gestor de persistencia GORM (Grails Object Relational Mapping) que se encarga de relacionar las clases del dominio con tablas de la base de datos.

Dojo 1.5

Dojo es una librería de clases JavaScript que contiene controles para facilitar el desarrollo de aplicaciones web que utilicen tecnología AJAX. Se utiliza por sus estilos visuales además cuenta con un mecanismo de creación de componentes que permite modificar los existentes y permitir crear nuevos componentes en correspondencia con las necesidades del proyecto.

1.4.6. Lenguajes

JavaScript

Es un lenguaje de programación ligero y orientado a objetos. Con este lenguaje script se pueden generar páginas dinámicamente en función de las preferencias del usuario, validar los datos introducidos en un formulario o modificar dinámicamente el contenido de la página. Se hace uso de este lenguaje para la validación del lado del cliente.

Groovy 1.6

Es un lenguaje de programación orientado a objetos, ágil y dinámico. Usa sintaxis muy parecidas a Java. La mayor parte de código escrito en Java es totalmente válido en Groovy por lo que este lenguaje es de muy fácil adopción para programadores Java. Es un complemento perfecto para Grails, debido a ser integración con Java y su dinamismo.

1.5. Conclusiones parciales

En el capítulo se realizó un estudio detallado del sistema penitenciario nacional, lo que permitió analizar el estado actual del mismo en todo lo referente a la gestión de las Visitas, y la necesidad de automatizar estos procesos. Con el análisis de Sistemas Penitenciarios existentes se demostró que estos no satisfacen las necesidades que actualmente tienen los centros penitenciarios de Cuba con respecto a las visitas. Se abordaron además todas las herramientas y tecnologías escogidas por el proyecto para el desarrollo del módulo Visitas del SIDEP así como la metodología de desarrollo, que en su conjunto permiten el desarrollo de la solución.

Capítulo II: Característica del sistema

2.1. Introducción

Este capítulo proporcionará una visión de cómo funciona el negocio a modelar y una descripción general del sistema para que cumpla con los requerimientos y procesos del negocio.

2.2. Modelo de negocio

Con la realización del modelado de negocio se pretende comprender la estructura de los procesos de visitas familiares y visitas institucionales, con el objetivo de definir los procesos y subprocesos que interactúan en el negocio, así como los roles y las responsabilidades de cada funcionario que interviene en la realización de las visitas, determinando los requisitos que necesita el cliente. El modelado de negocio del sistema fue realizado con BPMN.

2.2.1. Procesos de negocio identificados

El estudio del negocio permitió identificar los elementos fundamentales que interactúan en las visitas a los centros penitenciarios. Todas estas actividades, ligadas a la gestión de las visitas a estos centros de forma general y por el tipo de vista que constituyen pueden dividirse en dos procesos como son:

- Realización de Visita Familiar
- Realización de Visitas institucionales

Realización de Visita Familiar

La visita familiar como parte de los derechos de los internos constituye un proceso importante dentro de los Centros Penitenciarios, el cual es establecido por el régimen de cada colectivo es planificado y controlado por el centro penitenciario. Con este objetivo se debe registrar la fecha y hora de la visita, además de proporcionar una planificación trimestral para las visitas conyugales y anual para las familiares. Este plan contendrá las visitas planificadas a cada colectivo según el régimen pertinente, con una fecha y en turno asignado.

Realización de Visitas institucionales

La visita institucional constituye una visita de una determinada institución al Establecimiento Penitenciario con determinados fines, entre los que se encuentran la visita de abogados, cónsules y comunidades, así como las visitas asociadas a un proyecto. Por lo que la llegada del visitante al centro requiere el registro de sus datos para llevar el control de las visitas de este tipo y permitir la gestión de tal información en el centro penitenciario.

2.2.2. Descripción de los procesos

Para la especificación de las actividades que intervienen en las visitas, se realizó la descripción de los procesos en los que se define: objetivo, responsable, clientes, entradas, salidas y las actividades que intervienen en cada proceso.

Descripción del proceso Realización de Visita Familiar

La tabla que se muestra a continuación contiene los elementos que intervienen en la descripción del proceso **Realización de Visita Familiar.**

Nombre:	Realización de visita familiar
Objetivos:	Planificar y registrar la realización de la visita familiar.
Evento(s) que lo generan:	Beneficio de un interno a visita familiar
Precondiciones:	-
Poscondiciones:	Se realizó la visita familiar a un interno.
Reglas de Negocio:	
Responsables:	Jefe de Orden Interior
Clientes internos:	Funcionario de Orden Interior
Clientes externos:	Visitante
Entradas:	Listado de colectivos del centro y su última fecha de visita si la tiene.
Salidas:	Plan de Visitas familiares
Actividades:	Elaborar plan de visitas familiares Mostrar plan de visitas familiares

Tabla 1. Descripción del proceso Realización de Visita Familiar.

Como parte de la descripción de los procesos de visitas fueron confeccionados los diagramas correspondientes, los cuales constituyen una representación gráfica de los pasos que se siguen en toda una secuencia de actividades dentro de un proceso. Estos diagramas se dividen en calles que representan a los actores que interactúan en el sistema, dentro de cada calle se representan las actividades (rectángulos de color amarillo) y subprocesos (rectángulos de color naranja) que intervienen en el proceso a describir. Para el desarrollo de las actividades se utilizan objetos de datos que pueden ser creados o consultados. El inicio y fin del flujo de actividades están marcados con círculos de color verde y rojo respectivamente.

En la siguiente figura se presenta el diagrama del proceso **Realización de visita familiar** en el que se muestra el flujo de actividades que comienza cuando el Jefe de Orden Interior elabora el plan de visita familiar, para ello es necesario consultar la lista de colectivos del centro y a partir de esta se crea el plan de visitas familiares con el objetivo de mostrarlo para permitir su posterior consulta. Como parte

del flujo interviene el subproceso Ejecutar visita familiar el cual agrupa las actividades que se desarrollan en un día de visita.

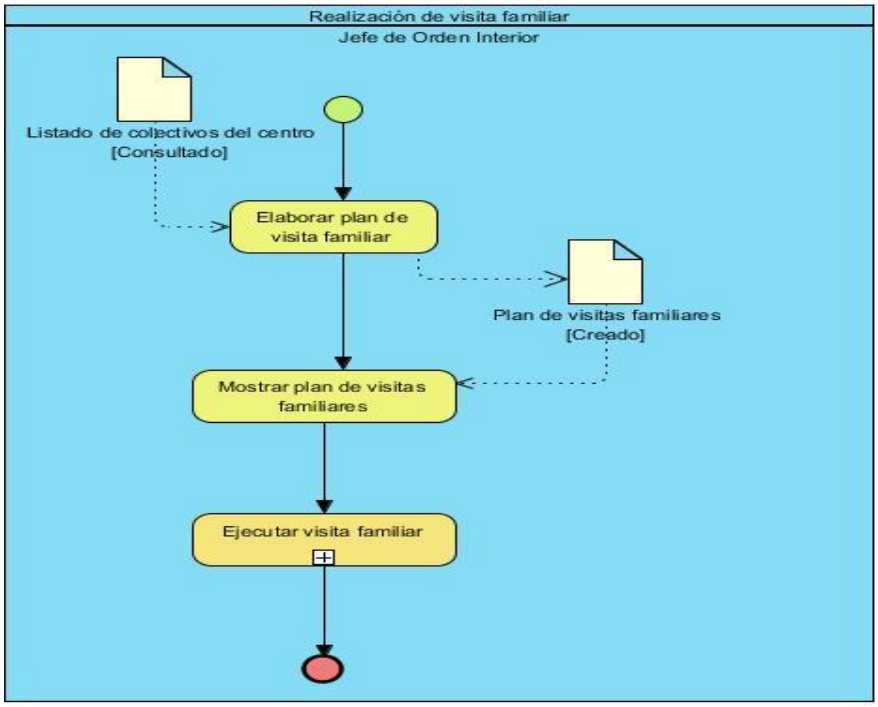

Figura 1. Diagrama del proceso Realización de Visita Familiar.

La descripción del subproceso Ejecutar visita familiar está en el Anexo 1

Descripción del proceso Realización de visitas institucionales

La siguiente tabla muestra la descripción de los elementos que intervienen en el proceso Realización de Visitas institucionales.

Nombre:	Registrar visitante institucional
Objetivos:	El objetivo de este proceso es registrar los datos del visitante institucional
Evento(s) que lo generan:	Una institución solicita al Establecimiento Penitenciario efectuar una visita ya sea para algún proyecto, programa y cualquier otra actividad y registrar a los visitantes institucionales.
Precondiciones:	-
Poscondiciones:	Se registra los visitantes institucionales
Reglas de Negocio:	
Responsables:	Funcionario de Orden Interior
Clientes internos:	Funcionario de Orden Interior
Clientes externos:	Visitante
Entradas:	Documentos de identificación

Salidas:	-
Actividades:	Recibir al visitante Solicitar documento de identificación Entregar pase de visitante institucional. Autorizar entrada al centro

Tabla 2. Descripción del proceso Realización de Visita Familiar.

En la siguiente figura se presenta el diagrama del proceso Realización de Visitas institucionales en el que se muestra el flujo de actividades pertenecientes a este donde una vez que llega el visitante al centro se le solicita el documento de identificación, se le entrega el pase de visitante institucional y se autoriza la entrada del mismo al establecimiento penitenciario.

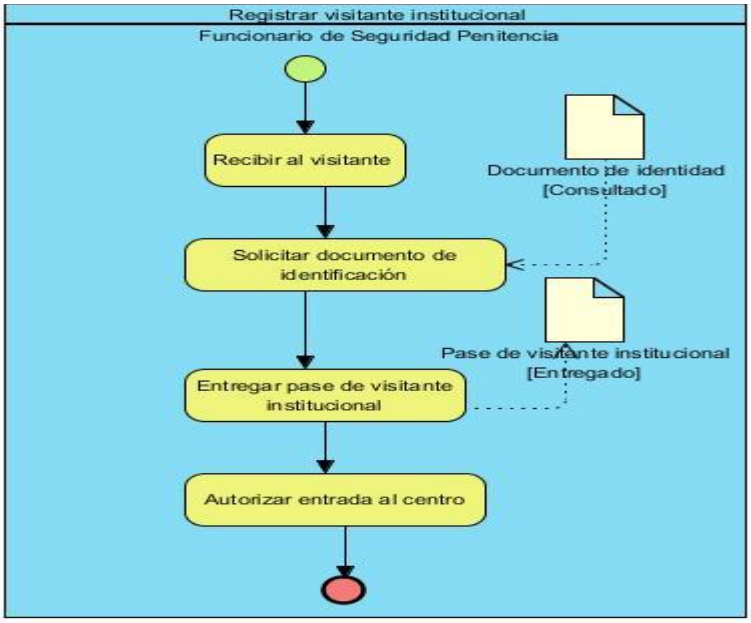

Figura 2. Diagrama del proceso Realización de Visitas institucionales.

Los procesos descritos anteriormente así como sus respectivos diagramas constituyen los procesos principales dentro del marco de este trabajo así como del negocio. Los restantes diagramas y sus descripciones correspondientes a los procesos y subprocesos del negocio en cuestión pueden ser consultados en el Anexo 1.

2.3. Modelo de sistema.

El modelo de sistema incluye los requisitos funcionales del sistema que responden a las necesidades de los clientes. Estos son especificados en la descripción de los casos de uso evidenciándose las relaciones entre los usuarios del sistema con las diferentes funcionalidades.

2.3.1. Requisitos funcionales del sistema.

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, describen lo que el sistema debe hacer (15).

2.3.1.1 Técnicas de captura de requisitos.

Existen varias técnicas para la captura de requisitos de las cuáles solo son descritas a continuación las utilizadas en la obtención de los requisitos del futuro sistema. La captura de requisitos constituye una etapa muy importante para el desarrollo de un software debido a que es el hilo conductor de todo desarrollo de software. Una obtención de requisitos con calidad demuestra que el trabajo realizado culminará con éxito.

- **Entrevistas:** Los analistas hacen preguntas a los clientes sobre el sistema que utilizan y sobre el sistema a desarrollar. Los requerimientos provienen de las respuestas a estas preguntas. Las entrevistas pueden ser de dos tipos (15):
  - **Entrevistas cerradas:** Donde los clientes responden un grupo predefinidos de preguntas.
  - **Entrevistas abiertas:** Donde no hay un programa definido de preguntas, se examinan un conjunto de cuestiones con los clientes para una mejor comprensión de sus necesidades.
- **Lluvia de ideas:** Es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios. Puede ayudar a generar una gran variedad de vistas del problema y a formularlo de diferentes formas, sobre todo al comienzo del proceso de captura. (16).
- **Casos de uso:** Es una técnica que se basa en escenarios para la obtención de los requerimientos. Un caso de uso identifica el tipo de interacción y los actores involucrados (15).

En la captura de los requisitos para los módulos Visitas Familiares y Visitas Institucionales se realizaron entrevistas con los clientes en las que mediante preguntas y aplicando la técnica de lluvia de ideas se obtuvieron parte de los requisitos. Con el análisis del sistema existente, SACORE, se determinó la necesidad de perfeccionar funcionalidades existentes y además de la extracción del resto de los requisitos para realizar posteriormente la descripción de los casos de uso.

2.3.1.2. Requisitos funcionales identificados

Módulo Visitas Familiares

- Configurar visitas familiares.
- Configurar visitas conyugales.
- Consultar plan de visitas familiares reglamentarias.
- Consultar plan de visitas conyugales reglamentarias.
- Registrar visitas familiares por estímulo.
- Registrar vistas conyugales por estímulo.
- Registrar visitas fuera de plan autorizadas por el Jefe de Unidad.
- Registrar visitas al ingreso.
- Registrar visitas familiares.
- Registrar visitas conyugales.
- Consultar histórico de visitas familiares.
- Consultar internos beneficiados con visitas.
- Consultar calendario de vistas.

Módulo Visitas Institucionales

- Registrar visitas de la comunidad.
- Registrar visitas consulares.
- Registrar visitas institucionales.
- Registrar visitas institucionales asociada a un proyecto.
- Consultar histórico de visitas institucionales.

2.3.1.3. Requisitos no funcionales.

Los requisitos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales. Los requisitos no funcionales se identificaron en reuniones del líder y el equipo de arquitectura del proyecto Prisiones Cuba con el cliente.

Requisitos de Apariencia o interfaz externa

- El sistema brindará una interfaz amigable para sus usuarios, con predominio de color verde en diversas tonalidades y gris. Las letras serán legibles con formato Arial.
- Los errores que sean visibles al usuario, indicarán las posibles causas y sus soluciones

Requisitos Software

- La PC donde se ejecute la aplicación debe tener instalado Firefox 3.X o superior. Debe tener alojado Oracle 11g o contar con dicho servidor externo.

- En caso de ser externo el servidor de base de datos, debe tener instalado Oracle 11g.
- La Computadora que sirva como servidor de la aplicación, debe tener instalado el Apache Tomcat como servidor Web, preferentemente una versión superior a 6.X y la Máquina Virtual de Java.

Requisitos de Hardware

- Es necesario contar con una computadora con un Microprocesador con velocidad superior a los 1.6 GHz, y con una memoria RAM superior a los 256 Mega Bytes, si en ella no se encuentra el servidor de Base de Datos.
- Si el servidor de base de datos es externo, debe estar alojado en una computadora con 2 GB de RAM, 500 HDD y tener la red disponible con una tarjeta de 10/100/1000 Mb/s.

Requisitos de diseño

- El sistema implementado será una aplicación web.
- El sistema se implementará usando la plataforma JAVA.
- El sistema estará basado en un estilo arquitectónico en capas.

Requisitos de Seguridad

- Seguridad y control a nivel de usuarios y contraseñas, garantizando el acceso de los mismos sólo a los niveles establecidos de acuerdo a la función que realicen.

2.3.1.4 Técnicas de validación de requisitos.

La validación de los requisitos muestra que estos realmente definen el sistema que el cliente desea, con esta se pretende encontrar errores en los requisitos. La validación es muy importante debido a que errores en los requisitos pueden provocar problemas en el desarrollo del sistema final. Para la validación de los requisitos capturados para los módulos de visitas se aplicó la técnica de construcción de prototipos.

- **Construcción de Prototipos:** Se construye una maqueta del futuro sistema software a partir de los requisitos recogidos y muestra a los usuarios y clientes para ver si cumplen con sus necesidades reales (15).

Los prototipos se incluyen en la descripción de casos de uso de cada módulo. La siguiente figura muestra el prototipo de la funcionalidad configurar visita, en el que se muestran los diferentes campos que compondrán la funcionalidad en la futura solución.

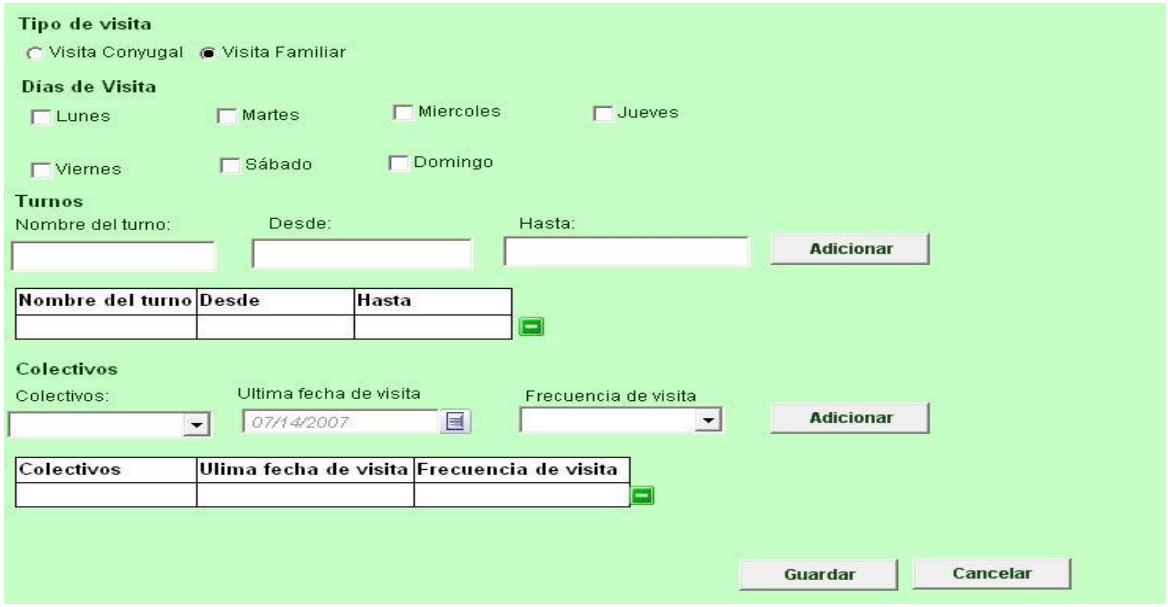

Figura 3. Prototipo de interfaz de la funcionalidad Configurar Visita.

La obtención de los prototipos para cada funcionalidad permitió mostrar una versión inicial del sistema y los elementos por los que está compuesta cada funcionalidad, como se mostró en la figura 3. El resto de los prototipos pueden ser consultados en el Anexo 2.

2.4. Actores

Los actores son un conjunto de roles que los usuarios de casos de uso desempeñan cuando interactúan con estos (17). La siguiente tabla muestra los actores que intervienen en el proceso de visita y las acciones que realizan.

Actores	Justificación
Jefe de Orden Interior	Es el encargado de planificar las visitas familiares y conyugales, así como registrar los datos para la realización de la visitas por estímulos y puede consultar los datos de las diferentes visitas.
Funcionario de Orden Interior	Es el encargado de registrar los datos de las diferentes visitas, como por ejemplo las no planificadas, las consulares, entre otras.

Tabla 3. Descripción de los actores que intervienen en el sistema.

2.4.1. Diagrama de Caso de Uso del sistema

Los Casos de Uso son “fragmentos” de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. Especifica una secuencia de acciones que el sistema puede llevar a cabo, interactuando con sus actores (17).

En las figuras 4 y 5 se muestran los diagramas de casos de uso para los módulos Visitas Familiares y Visitas Institucionales respectivamente en los que se ponen de manifiesto las relaciones entre los actores y las funcionalidades que interactúan en estos módulos.

Diagrama de caso de uso del módulo Visitas Familiares

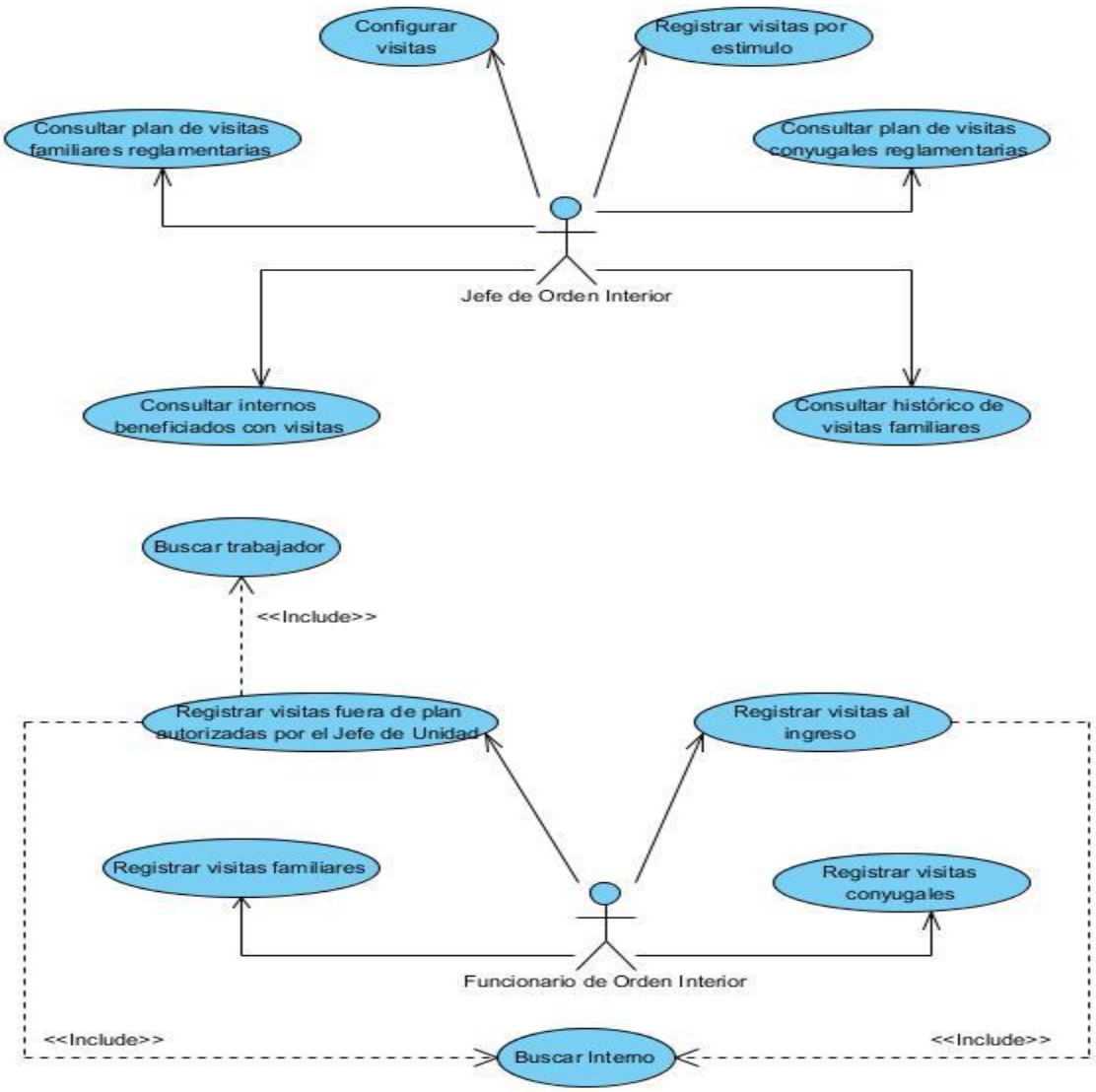

Figura 4 Diagrama de Caso de Uso del sistema Vistas Familiares.

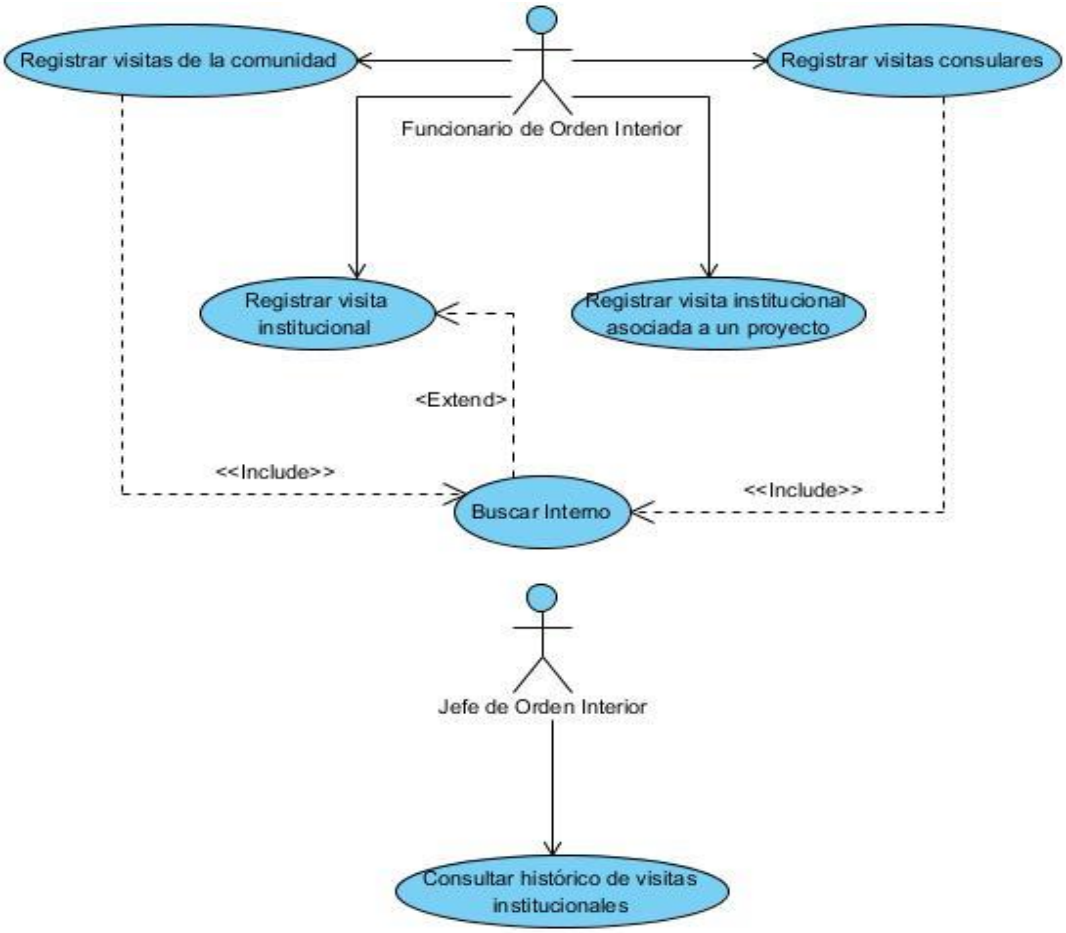

Figura 5 Diagrama de Caso de Uso del sistema Vistas Institucionales.

2.4.2. Patrones de Casos de Uso.

Los patrones de casos de uso permiten el desarrollo de modelos de casos de uso reutilizables (18). Estos patrones describen como deberían ser estructurados y organizados los casos de uso.

Extensión Concreta o Inclusión: Este patrón consiste en dos casos de uso y una relación de extensión o inclusión entre ellos.

- **Extensión:** Consiste en dos casos de uso y una relación extendida entre ellos. Este patrón se aplica cuando un flujo puede extender el flujo de otro caso de uso así como ser realizado en sí mismo (18). Se utiliza este patrón cuando no es obligatorio la participación de un interno en una visita. La siguiente figura presenta un fragmento del caso de uso del módulo Visitas Institucionales que evidencia el uso de este patrón, el cual muestra que una visita institucional puede no estar asociada a un interno.

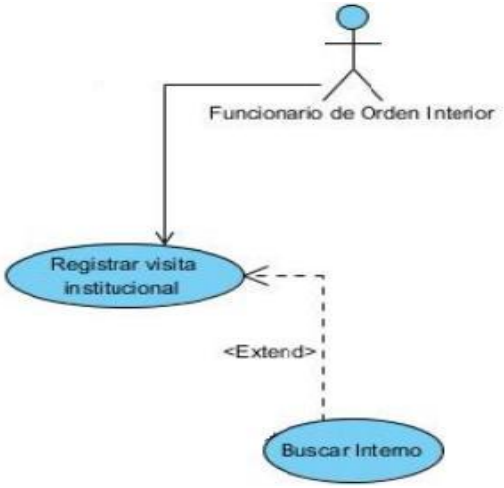

Figura 6. Fragmento del diagrama de caso de uso del módulo Visitas Institucionales donde se evidencia el uso del patrón Extensión Concreta.

- **Inclusión:** Se incluye una relación del caso de uso base al caso de uso de inclusión. El último puede ser instalado en sí mismo (18). Se utiliza este patrón cuando es obligatoria la participación de un trabajador en una visita. La siguiente figura presenta un fragmento del caso de uso del módulo Visitas Familiares que evidencia el uso de este patrón, mostrándose que en la realización de una visita fuera del plan siempre se requiere la presencia de un trabajador.

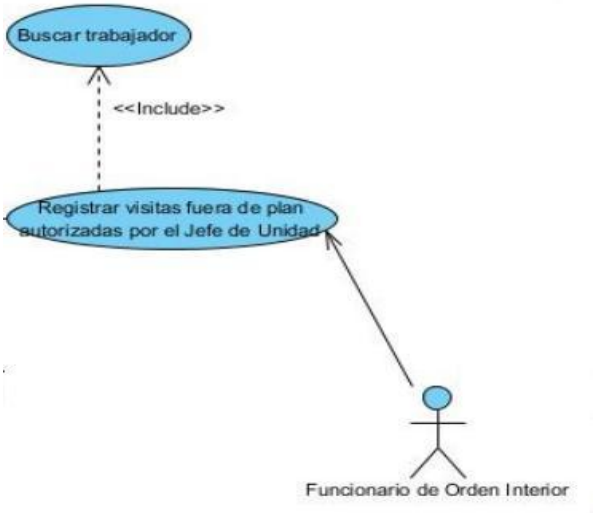

Figura 7. Fragmento del diagrama de caso de uso del módulo Visitas Familiares donde se evidencia el uso del patrón Inclusión.

Concordancia: Extrae una subsecuencia de acciones que aparecen en diferentes lugares del flujo de casos de uso y es expresado por separado (18).

- **Reuso:** Constituye una variante del patrón Concordancia. Consta de 3 casos de uso. El primero llamado subsecuencia común, modela una secuencia de acciones que aparecerán en múltiples casos de uso en el modelo. Los otros casos de uso modelan el uso del sistema que comparte la subsecuencia común de acciones (18). Se utiliza este patrón cuando en la realización de las visitas, es obligatoria la participación de los internos. La siguiente figura presenta un fragmento del diagrama de caso de uso del módulo Visitas Familiares donde se evidencia el uso de este patrón, evidenciándose que en la realización de una visita fuera del plan y en una al ingreso, siempre es necesario la presencia de al menos un interno.

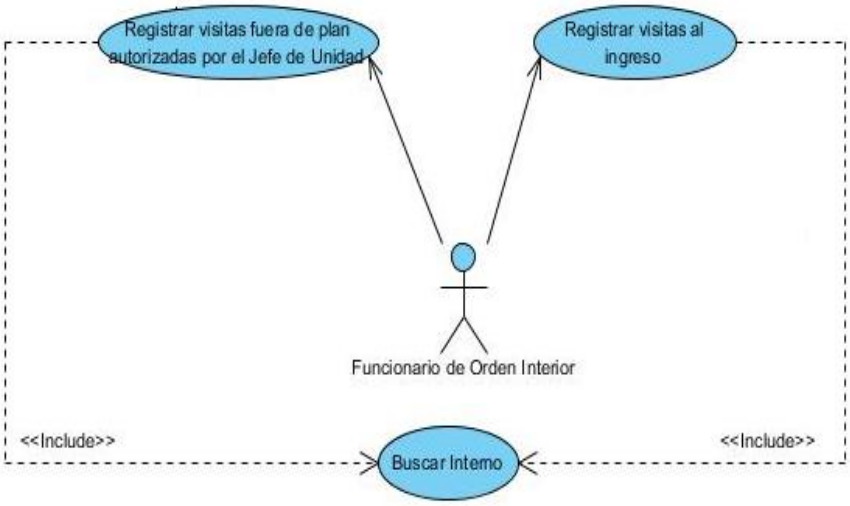

Figura 8. Fragmento del diagrama de caso de uso del módulo Visitas Familiares donde se evidencia el uso del patrón Concordancia en su variante Reuso.

2.4.3. Descripción de Casos de Uso

Las descripciones de los casos de uso constituyen un documento narrativo que describen la secuencia de eventos de un actor (agente externo) que utiliza un sistema para completar un proceso (19).

A continuación se presenta la descripción del caso de uso Configurar Visitas que permite seleccionar el tipo y los días de visita, así como crear los turnos en que se realizarán las mismas. Una vez guardada la configuración, se generará de forma automática el plan de visitas según el tipo de visita seleccionado en la configuración (familiar o conyugal).

Objetivo	Configurar las visitas, turnos de visita y generar el plan de visita en el Centro Penitenciario.
Actores	Jefe de Orden Interior.

Resumen	El caso de uso se inicia cuando el Jefe de Orden Interior necesita configurar las visitas y turnos de visitas en el Centro Penitenciario, el sistema permite realizar estas acciones, registra la información y termina el CU.	
Complejidad	Alta	
Prioridad	Normal.	
Precondiciones	El Jefe de Orden Interior debe estar autenticado en el sistema y con los permisos para registrar esta operación.	
Postcondiciones	Se configura las visitas en el Centro Penitenciario.	
Flujo de eventos		
Flujo básico Configurar visita		
	Actor	**Sistema**
1.	Selecciona la opción “Configurar visita” en el menú dentro de la opción “Visitas”.	
2.		Muestra una interfaz para configurar las visitas y permite seleccionar que tipo de visita desea configurar: • Visitas Familiares • Visitas Conyugales
3.	Selecciona un tipo de visita.	
4.		Permite seleccionar: • Días de visita Y permite crear además los turnos de visita con los siguientes datos: • Nombre del turno • Desde • Hasta Y registrar de cada colectivo la fecha de su última visita: • Colectivo • Fecha última visita • Frecuencia de visita
5.	Si selecciona la opción Eliminar, ver flujo alterno 5a.”Eliminar Turno de Visita”.	
6.		Valida los datos introducidos. Si faltan datos obligatorios ver flujo alterno 6a. “Faltan datos obligatorios”.
7.		Guarda los datos de la configuración de la visita y a su vez genera el plan de visitas para el tipo de visita que se configuró.

		Si la visita es conyugal ver flujo alterno 7a. "Alerta sobre creación de nuevo plan".
8.		Termina el CU.
Flujos alternos		
*** Cancelar**		
	Actor	**Sistema**
1.	Selecciona el botón "Cancelar".	
2.		No registra la información y regresa al paso 2 del flujo básico.
Flujos alternos		
5a. Eliminar Turno de Visita Familiar		
	Actor	**Sistema**
1.	Selecciona la opción "Eliminar".	
2.		Muestra el siguiente mensaje "Está seguro que desea eliminar el elemento seleccionado".
3.	Oprime y el botón "Si".	
4.		Elimina el turno de visita y regresa a paso 4 del flujo básico.
6a. Faltan Datos Obligatorios		
	Actor	**Sistema**
1.		Muestra el mensaje "Los campos en rojo son obligatorios" indicando los campos obligatorios que faltaron por llenar.
2.	Introduce los datos obligatorios que faltan en el sistema y oprime el botón "Aceptar".	
3.		Regresa al paso 6 del flujo básico.
7a. Alerta sobre creación de nuevo plan		
	Actor	**Sistema**
1.		Muestra una alerta informando que la creación del plan de visitas conyugales para el próximo trimestre.
2.		Regresa al paso 6 del flujo básico.

Relaciones	**CU Incluidos**	No aplicable
	CU Extendidos	No aplicable
Requisitos no funcionales	No aplicable	
Asuntos pendientes	No aplicable	

Tabla 4. Descripción del caso de uso Configurar visitas.

La siguiente tabla muestra un resumen de los casos de uso que no fueron descritos anteriormente.

Casos de uso del módulo Visita Familiar

Casos de uso	Descripción
Registrar visitas por estímulo.	**Actor:** Jefe de Orden Interior. Permite registrar las visitas que son otorgadas a internos por estímulo, en él se muestra un listado de los internos estimulados y la disponibilidad de turnos del centro una vez planificada la visita y permite registrar este tipo de visita.
Mostrar plan visitas familiares.	**Actor:** Jefe de Orden Interior. Permite mostrar el plan de visitas familiares del año vigente, con todos los colectivos, los días y el mes que le tocó la visita al mismo.
Mostrar plan visitas conyugales.	**Actor:** Jefe de Orden Interior. Permite mostrar el plan de visitas conyugales del trimestre vigente, con todos los colectivos, los días y el mes que le tocó la visita al mismo.
Registrar visitas fuera del plan.	**Actor:** Funcionario de Orden Interior. Permite registrar los datos de los visitantes y los internos de visitas que no estén planificadas, así como la fecha y hora de la misma.
Registrar visitas al ingreso.	**Actor:** Funcionario de Orden Interior. Permite registrar los datos del interno y el visitante, esta constituye la primera visita que tiene el interno.
Registrar visitas conyugales.	**Actor:** Funcionario de Orden Interior. Permite registrar los datos de la entrada y salida de los visitantes en un día de visita conyugal.
Consultar internos beneficiados con visita.	**Actor:** Jefe de Orden Interior. Permite consultar los días que los internos tendrán visita según lo planificado por día, mes, año y por número de expediente.
Consultar histórico de visitas familiares	**Actor:** Jefe de Orden Interior. Permite consultar los datos de las diferentes visitas familiares realizadas al centro por día, mes y año y los detalles de las mismas.

Tabla 5. Resumen de los casos de uso del módulo Visitas Familiares.

Casos de uso del módulo Visitas Institucionales

Casos de uso	Descripción

Registrar visitas de la comunidad.	**Actor:** Funcionario de Orden Interior Permite registrar los datos de visitas de las comunidades al centro penitenciario, se recogen los datos de los visitantes.
Registrar visitas institucionales.	**Actor:** Funcionario de Orden Interior Permite registrar los datos de visitas de instituciones al centro penitenciario, se recogen los datos de cada visitante.
Registrar visitas institucional asociada a un proyecto.	**Actor:** Funcionario de Orden Interior Permite registrar los datos de visitas de proyectos al centro penitenciario, se recogen los datos del proyecto y de cada integrante.
Registrar visitas consulares.	**Actor:** Funcionario de Orden Interior Permite registrar los datos de cónsules al centro, se recoge el país del mismo.
Consultar histórico de visitas institucionales.	**Actor:** Jefe de Orden Interior Permite consultar los datos de las diferentes visitas institucionales realizadas al centro por día, mes y año y los detalles de las mismas.

Tabla 6. Resumen de los casos de uso del módulo Visitas Institucionales.

Las descripciones de los casos de usos mencionados anteriormente se encuentran en el Anexo 3.

2.6. Conclusiones parciales

En este capítulo se explicaron las características del sistema y del negocio en cuestión. Además se identificaron los procesos de negocio, los cuales se modelaron en diagramas de procesos para un mejor entendimiento de la lógica definida en ellos. Se realizó una captura de requisitos teniendo en cuenta las técnicas apropiadas para dicho procedimiento, tales como, el estudio del negocio y las entrevistas con los clientes, donde quedaron definidos los actores y los requisitos correspondientes a los procesos de visitas familiares e institucionales, mediante la obtención de los casos de uso del sistema y sus descripciones.

Capítulo III. Análisis y diseño del sistema.

3.1. Introducción.

El presente capítulo recoge los aspectos fundamentales del análisis y diseño del sistema. En el análisis se estructuran los requisitos obtenidos con anterioridad, profundizando en el dominio de la aplicación. Se definirán las clases para modelar la solución lo que permitirá una mayor comprensión del sistema. Por otra parte se reflejará todo lo referente al diseño de la aplicación en aras de dar cumplimiento a los requisitos funcionales del sistema.

3.2. Análisis

El objetivo del análisis es comprender los requisitos del software y lograr una descripción de estos de manera que ayude a estructurar el sistema.

3.2.1. Diagrama de clases de análisis

A continuación se presenta el diagrama de clases del análisis del caso de uso Configurar visita, en el que se muestran las relaciones entre las diferentes clases que intervienen en esta funcionalidad. En el diagrama se muestran las clases Principal y ConfigurarVisita, las cuales representan clases interfaces que muestran la interacción entre el sistema y los actores. Mediante las clases controladoras SeguridadPenitenciariaInicioController y ConfiguraciónVisitaController se manejan y coordinan las acciones entre las interfaces y las entidades Plan_visitas_familiares, Plan_visitas_conyugales, Configurar_visita, Turno, Colectivo, Días_visita y Fecha_colectivo, como se muestra en la figura 9.

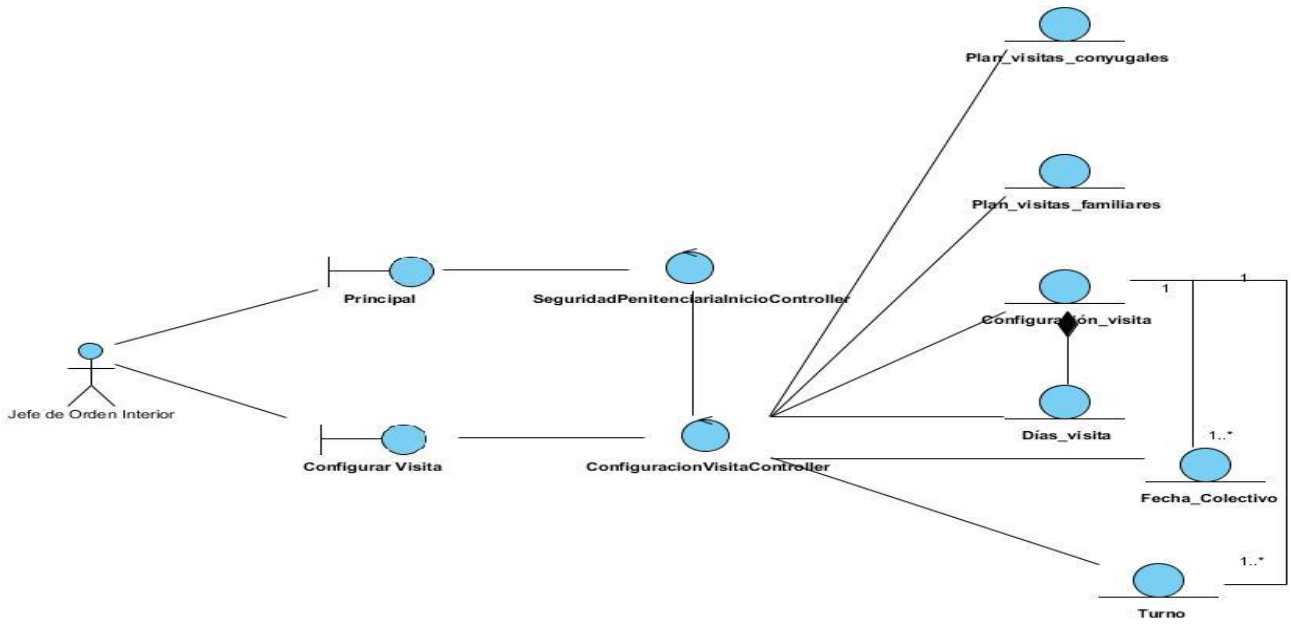

Figura 9. Diagrama de clases del análisis del caso de uso Configurar visitas.

La confección de los diagramas de clases del análisis constituye una parte fundamental de la disciplina de Análisis porque se utilizan como punto de partida en el desarrollo de la solución, para consultar los restantes diagramas de clase del análisis, ver Anexo 4.

3.3. Diseño

En el diseño se modela el sistema. Se crea una entrada apropiada y un punto de partida para las actividades de la implementación, se descomponen los trabajos de implementación en partes más manejables que pueden ser llevados a cabos por diferentes equipos de desarrollo (17).

3.3.1. Arquitectura del sistema

La arquitectura es la estructura jerárquica de los componentes del programa, la manera en que los componentes interactúan y la estructura de datos que van a utilizar los componentes (20).

La arquitectura del sistema se basa principalmente en las funcionalidades y facilidades del framework de desarrollo Grails en su versión 1.3.7 que incluye componentes que van desde los flujos webs hasta la capa de acceso a datos, conocida como “Arquitectura en capas según Grails”. Esta arquitectura en capas está basada en el patrón Modelo-Vista-Controlador. A continuación se explica la función de cada capa.

- **Capa de presentación:** Los componentes de esta capa son responsables de mostrar al usuario el estado actual del modelo de datos, y representarle las distintas acciones disponibles (14). Contienen plantillas Groovy Server Page (gsp), las cuales son las encargadas de mostrar en el navegador el contenido que se encuentra en el modelo, mediante código HTML para mostrar los componentes de la página, JavaScript para definir el comportamiento de los componentes y Cascading Style Sheets (CSS) para los estilos de las páginas.
- **Capa de control:** Contiene los componentes que reciben las órdenes del usuario, gestionan la lógica de negocio sobre el modelo de datos, y determinan que vista debe mostrarse a continuación (14). Grails brinda la facilidad de manejar la lógica de negocio fuera de la capa de control, dejándole la responsabilidad de establecer y mantener el flujo de comunicación con el usuario y manejar estas peticiones.
- **Capa de datos:** Contiene los componentes que representan y gestionan los datos manejados por la aplicación. En el caso más típico, los objetos encargados de leer y escribir en la base de datos (14). En esta capa se crean las clases de dominio de Grails. Que son utilizadas para la abstracción de datos mediantes el gestor de persistencia Grails Object Relational Mapping (GORM) con el objetivo de controlar el ciclo de vida de las

entidades y proporcionar una serie de métodos dinámicos que faciliten el acceso a los datos.

Esta especificación del Modelo-Vista-Controlador contiene otro componente que se adiciona al modelo o capa de datos para implementar la lógica del negocio, los servicios de Grails. Sobre estos recae la responsabilidad de proporcionar un punto de acceso común en forma de fachada de servicios a través de los controladores.

- **Servicios:** Contiene componentes encargados de implementar la lógica de negocio de la aplicación siendo los encargados de realizar los cambios en el modelo.

La siguiente figura muestra la organización de los componentes del MVC de Grails, en el que se evidencia los diferentes elementos que lo componen.

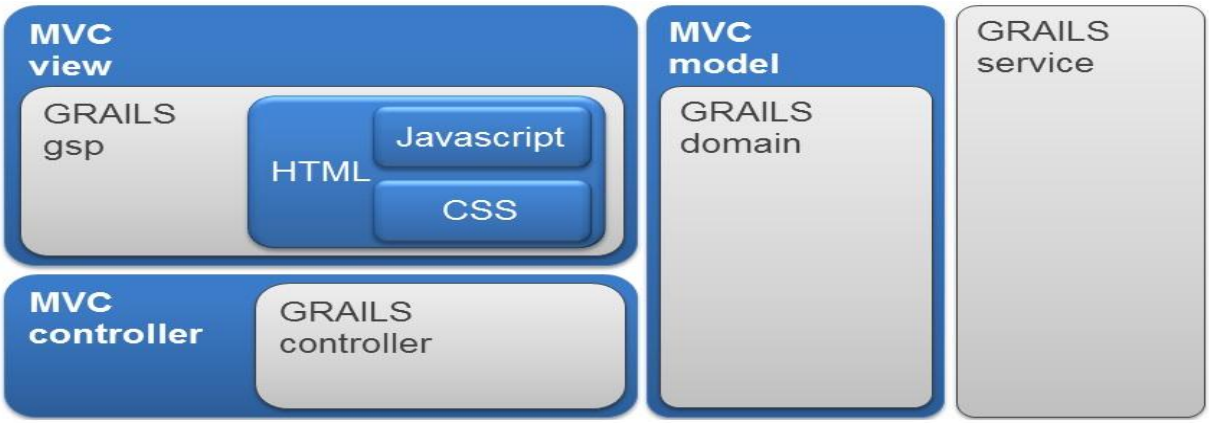

Figura 10. Arquitectura del sistema.

1.3.2. Patrones de diseño

Representan una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos. Ofrecen orientación sobre como asignar las responsabilidades a los objeto (19).

Patrones GRASP

Representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones (19). GRASP es el acrónimo para General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades).

- **Experto:** Asigna una responsabilidad al experto en información, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad (19).

El uso de este patrón permite que se conserve un bajo acoplamiento. En el diseño de las clases del dominio se utilizó este patrón al generarse las clases con las responsabilidades debidamente asignadas. Un ejemplo es la clase ConfigurarVisita.groovy, ella contiene la información referente a la configuración de las visitas.

- **Controlador:** Asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase (19).

Se utilizó este patrón en el diseño de los controladores de Grails. Las clases controladoras son las encargadas de gestionar todas las funcionalidades referentes a una entidad. Un ejemplo del uso de este patrón es la clase controladora configuraciónVisitaController.groovy, ella es la encargada de manejar a través de acciones el flujo de información entre las vistas, los servicios y a su vez con el modelo de datos.

- **Alta Cohesión:** La cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme (19).

Se hace uso de este patrón en el registro de las visitas. Teniendo en cuenta que son diferentes tipos de visitas, es indispensable la utilización de un controlador y un servicio asociado a él, que tienen la responsabilidad de registrar solo los datos de la visita con la que está relacionado, y estos solo tienen relación con los componentes que necesarios para registrarla. Un ejemplo de esto son las clases configurarVisitaController.groovy y configurarVisitaService.groovy que son las encargadas de guardar solo los datos de la configuración de visita.

- **Bajo Acoplamiento:** Asigna una responsabilidad para mantener bajo acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras (19).

Con este patrón las clase se diseñan lo menos ligadas entre si posibles, lo que permite que en caso de producirse una modificación en alguna de estas, repercuta lo menos posible en el resto de ellas. Durante el desarrollo de los módulos Visitas Familiares y Visitas Institucionales se hace uso de este patrón en el diseño de las clases del dominio.

3.3.3 Diagrama de clases del diseño

Los diagramas de clases del diseño muestran una abstracción de clases similar a la implementación del sistema. A continuación se presenta el diagrama de clase del diseño basado en el caso de uso

Configurar Visita, en el que interviene la clase Principal.html encargada de mostrar al usuario el menú que proporciona la opción seleccionar la opción Configurar visita. Esta solicitud es enviada a la clase seguridadPenitenciariaInicioController que redirecciona al controlador configuraciónVisitaController para mostrar la clase configurarVisita.gsp muestra la información al usuario construyendo la vista configuracionVisita.html la cual contiene un formulario que permite introducir y enviar los datos que son validados mediante un archivo JavaScript. Estos datos son recepcionados por el controlador configuraciónVisitaController que los envía al servicio ConfiguraVisitasService quien realiza las operaciones necesarias para salvar la configuración de la visita registrando los datos en las entidades correspondientes.

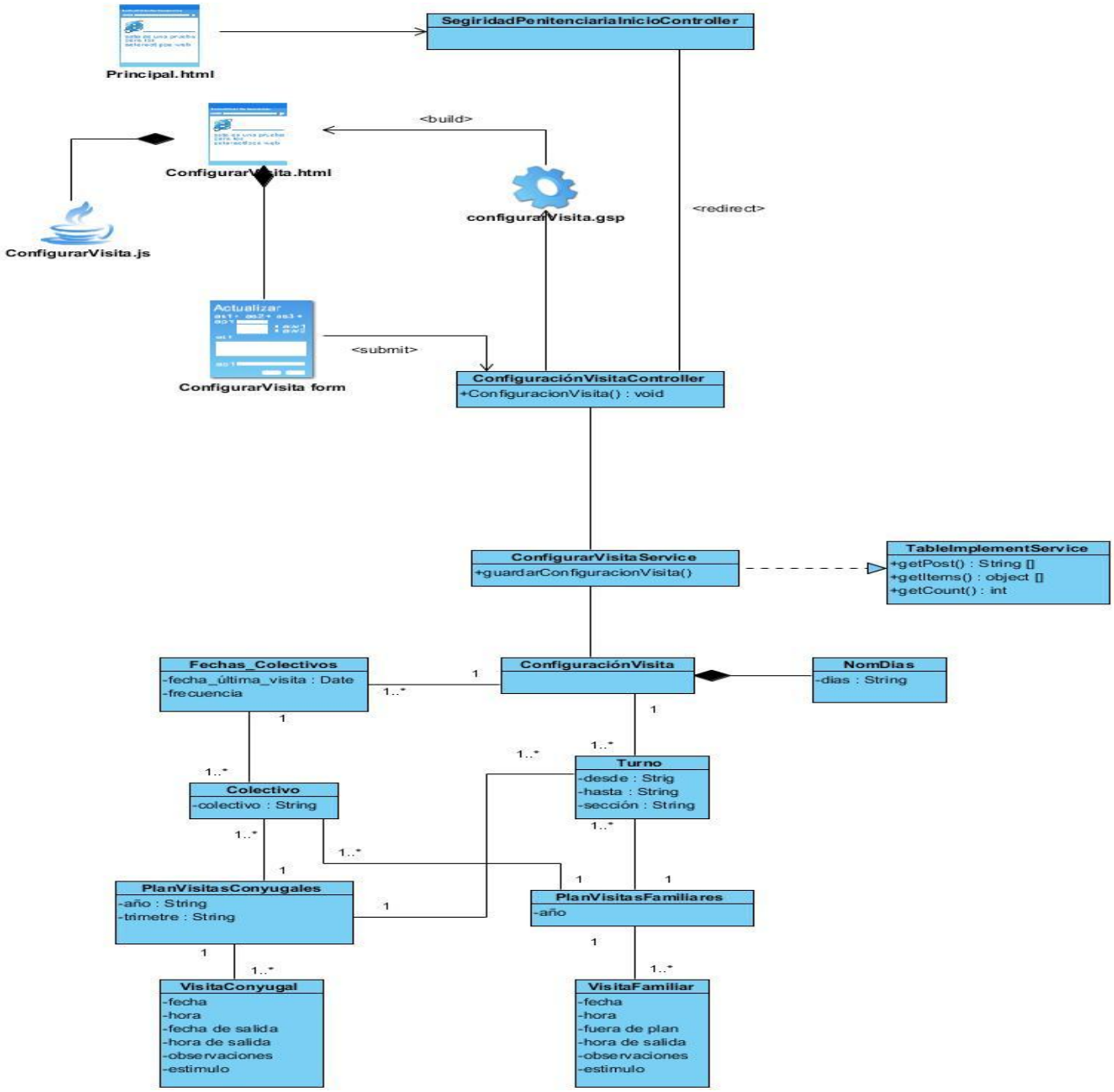

Figura 11. Diagrama de clases del diseño del caso de uso Configurar visita.

En el diagrama de clases expuesto en la figura 11 se evidencia el uso de los patrones de diseño explicados en el epígrafe **3.3.2.** La entidad ConfiguraciónVisita y la controladora ConfiguraciónVisitaController implementan el uso del patrón Experto y Controlador respectivamente. El patrón Alta Cohesión se pone de manifiesto en las clases ConfiguraciónVisitaController y ConfigurarVisitaService. Por otra parte las relaciones entre las entidades FechaColectivo, PlanVisitasFamiliares, PlanVisitasConyugales, VisitaFamiliar, VisitaConyugal, Turnos, ConfiguraciónVisita y Colectivo evidencian el uso del patrón Bajo Acoplamiento.

Los diagramas de clase del diseño muestran una abstracción de las clases que intervienen en el sistema, parte de la implementación del mismo depende del correcto diseño de estos diagramas, para consultar los restantes diagramas de clase del diseño, en el Anexo 5.

3.3.4 Descripción de las clases. Diagrama de clases del diseño Configurar Visita.

En las tablas que se muestran a continuación se describen las clases que intervienen en el diagrama de clases del diseño del caso de uso Configurar Visita, en la que se describe el tipo de clases, acciones y atributos que intervienen en cada una, así como sus descripciones. Este conjunto de clases constituyen las clases más relevantes que intervienen en la creación de las visitas familiares y conyugales.

Nombre: ConfiguraciónVisitaController	
Tipo de clase: Controladora	
Descripción: Controla el flujo de información entre las interfaces y los servicios.	
Acciones	**Descripción**
Index()	Muestra la vista configuracionVisita.gsp
configuracionVisita()	Crea una instancia de la clase configurarVisita.groovy

Tabla 7. Descripción de la clase ConfiguraciónVisitaController.

Nombre: ConfiguraciónVisitaService	
Tipo de clase: Controladora	
Descripción: Almacena en la base de datos las instancias de la clase Configuración. Hereda de TableService con el objetivo de listar los turnos creados, así como los colectivos con su última fecha de visita.	
Acciones	**Descripción**
guardarConfigurarVisita()	Crea una instancia de la clase ConfiguracionVisita, la valida y la almacena en la base de datos.

Tabla 8. Descripción de la clase ConfiguraciónVisitaService.

Nombre: ConfigurarVisita.html		
Tipo de clase: Interfaz		
Descripción: Permite insertar los datos de la configuración de la visita mediante un formulario y contiene un archivo javascript para la validación de los datos en caso de errores.		
Atributos	**Tipo**	**Descripción**
tipoVisita	String	Registra el tipo de visita que se vaya configurar, puede ser visitas familiares o visitas conyugales
desde	String	Registra la hora de comienzo del turno de visita.
hasta	String	Registra la hora de fin del turno de visita.
nombre_turno	String	Registra el nombre del turno de visita.
días	NomDias	Registra los días de visita del Establecimiento Penitenciario.

fechaUltimaVisita	Date	Registra la fecha de la última visita de un colectivo determinado.
colectivo	String	Registra el nombre del colectivo al que se le registra la fecha de la última visita

Tabla 9. Descripción de la clase ConfigurarVisita.html.

Nombre: configuraciónVisita.js	
Tipo de clase: Interfaz	
Descripción: Valida los campos referentes a una configuración de visita.	
Componentes	**Descripción**
mostrarDatos(params)	Le asigna valores a los campos.
enviar()	Envía los datos del formulario.
validar()	Es la función que valida los campos correspondientes a una configuración, verificando que no se envíen en blanco y que se seleccione algún valor.

Tabla 10. Descripción de la clase configuraciónVisita.js.

Nombre: ConfiguraciónVisita		
Tipo de clase: Entidad		
Descripción: Guarda los datos configuración de visita.		
Atributos	**Tipo**	**Descripción**
tipoVisita	String	Guarda el tipo de visita que se vaya a configurar, puede ser visitas familiares o visitas conyugales.
días	NomDias	Guarda los días de visita del Establecimiento Penitenciario.

Tabla 11. Descripción de la clase ConfiguraciónVisita.

Nombre: FechaColectivo		
Tipo de clase: Entidad		
Descripción: Guarda la fecha de la última visita de un colectivo determinado.		
Atributos	**Tipo**	**Descripción**
fechaUltimaVisita	Date	Guarda la fecha de la última visita de un colectivo determinado.

Tabla 12. Descripción de la clase FechaColectivo.

Nombre: Turno		
Tipo de clase: Entidad		
Descripción: Guarda los turnos de vistas creados en la configuración de visita.		
Atributos	**Tipo**	**Descripción**
desde	String	Guarda la hora de comienzo del turno de visita.
hasta	String	Guarda la hora de fin del turno de visita.
sección	String	Guarda el nombre del turno de visita.

Tabla 13. Descripción de la clase Turno.

Nombre: PlanVisitasFamiliares		
Tipo de clase: Entidad		
Descripción: Guarda el plan de visitas familiares.		
Atributos	**Tipo**	**Descripción**
año	String	Guarda el año del plan de visitas familiares.

Tabla 14. Descripción de la clase Plan de visitas familiares.

Nombre: PlanVisitasConyugales		
Tipo de clase: Entidad		
Descripción: Guarda el plan de visitas conyugales.		
Atributos	**Tipo**	**Descripción**
año	String	Guarda el año del plan de visitas conyugales.
trimestre	String	Guarda el trimestre del plan de visitas conyugales.

Tabla 15. Descripción de la clase Plan de visitas conyugales.

Nombre: VisitaFamiliar		
Tipo de clase: Entidad		
Descripción: Guarda los datos de las visitas familiares.		
Atributos	**Tipo**	**Descripción**
fecha	Date	Guarda la fecha de la visita familiar.
hora	String	Guarda la hora de la visita familiar.
fuera_plan	Booleano	Guarda si visita familiar es fuera del plan o no.
hora salida	String	Guarda la hora de salida de la visita familiar.
observaciones	String	Guarda las observaciones de la visita familiar.
estímulo	Booleano	Guarda si la visita es por estímulo o no.

Tabla 16. Descripción de la clase Visita familiar.

Nombre: VisitaConyugal		
Tipo de clase: Entidad		
Descripción: Guarda los datos de las visitas conyugales.		
Atributos	**Tipo**	**Descripción**
fecha	Date	Guarda la fecha de la visita conyugal.
hora	String	Guarda la hora de la visita conyugal.
hora salida	String	Guarda la hora de salida de la visita familiar.
fecha salida	Date	Guarda la fecha de salida de la visita familiar.

observaciones	String	Guarda las observaciones de la visita familiar.
estímulo	Booleano	Guarda si la visita es por estímulo o no.

Tabla 17. Descripción de la clase Visita conyugal.

3.3.5 Diagramas de interacción. Diagramas de colaboración

En los diagramas de interacción se representan las interacciones entre las diferentes clases que intervienen en los casos de uso. El diagrama presentado a continuación está en correspondencia con el mostrado anteriormente y representa el diagrama de colaboración de la funcionalidad Configurar Visita, en el que se muestra mediante mensajes la secuencia de pasos a seguir por los actores para guardar la configuración.

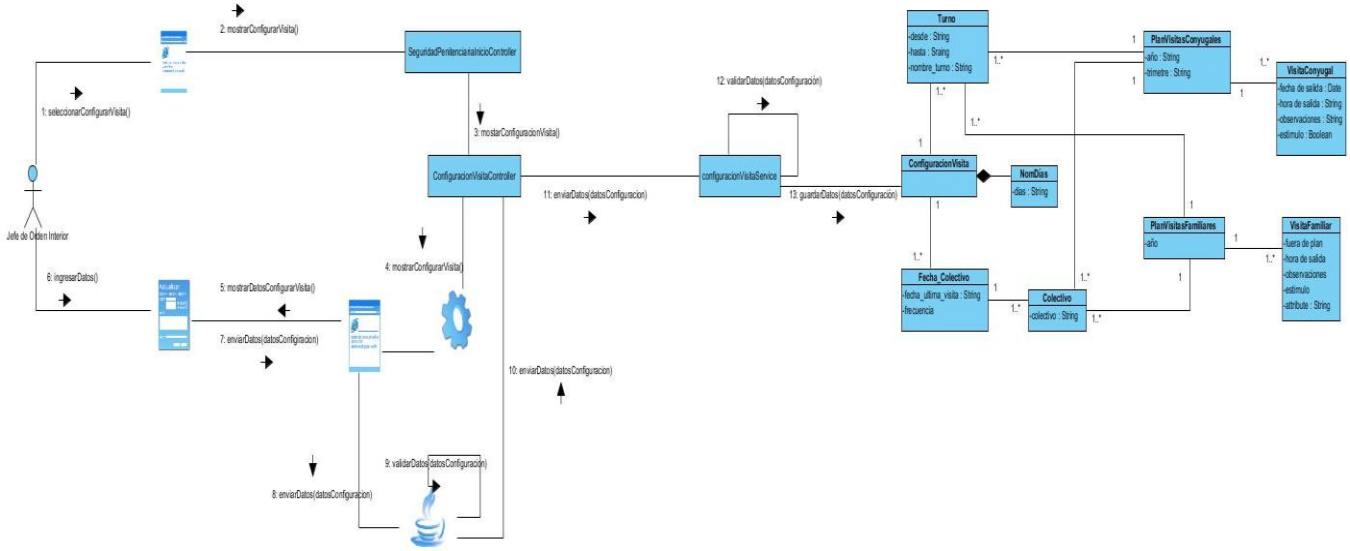

Figura 12. Diagrama de colaboración Configurar visita.

Los diagramas de colaboración son diagramas de interacción que muestran la comunicación y el envío de mensajes entre las clases que intervienen en cada funcionalidad del sistema. Para consultar los restantes diagrama de colaboración ver Anexo 6.

3.3.6 Diseño de la BD

El diseño de la base de datos es de suma importancia porque su objetivo principal es generar un conjunto de esquemas de relaciones que permitan almacenar la información y a la vez faciliten la recuperación de la misma. Permite además la creación del modelo entidad relación el cual expresa las entidades relevantes para el sistema así como sus relaciones.

En la siguiente imagen se presenta el diseño de la Base de datos de los módulos Visitas Familiares y Visitas Institucionales, en el que se muestran las entidades que intervienen en estos módulos y las

relaciones entre ellas. Las principales entidades que intervienen son: Visita y ConfigurarVisita. De Visita heredan VisitaFamiliar, VisitaConyugal y Proyecto. ConfigurarVisita mantiene una relación de uno a muchos con FechaColectivo, Días y Turnos, además intervienen las entidades PlanVisitasFamiliares y PlanVisitasConyugales encargadas de guardar los planes de visita.

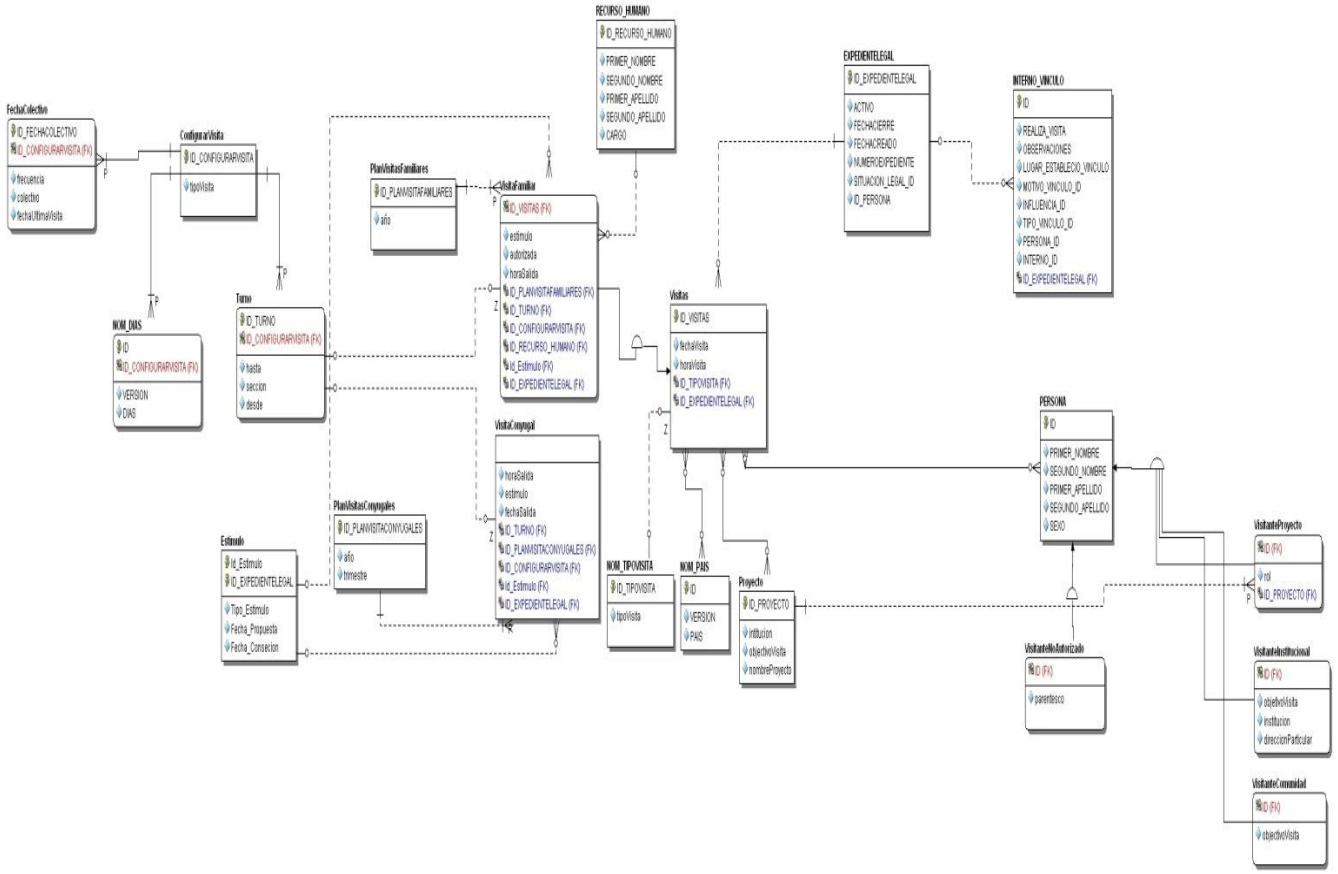

Figura 13. Diagrama Entidad Relación de los módulos Visitas Familiares y Visitas Institucionales.

Para que el nombre de las entidades no excediera el máximo número de caracteres admisibles por Oracle fue necesario truncarlos, tal es el caso de ConfigurarVisita, VisitaFamiliar, VisitaConyugal, PlanVisitaFamiliar y PanVisitaFamiliar modificadas a VisConfigurar, VisFamiliar, VisConyugal, VisPlanVF y VisPlanVC respectivamente.

3.3.7 Descripción de tablas

En las tablas que se muestran a continuación se describen las entidades principales que intervienen en los módulos Visitas Familiares y Visitas Institucionales, las cuales contienen los diferentes atributos, el tipo de cada uno de ellos, así como su descripción.

Nombre: VISCONFIGURAR		
Descripción: Esta clase guarda la configuración de la visita.		
Atributos	**Tipo**	**Descripción**
VISCONFIGURAR_ID	NUMBER(19,0)	Guarda el identificador de la configuración de la visita.
TIPO_VISITA_ID	NUMBER(19,0)	Guarda el identificador del tipo de visita que se vaya a configurar.

Tabla 18. Descripción de la tabla VISCONFIGURAR.

Nombre: VISCONYUGAL		
Descripción: Esta clase guarda las visitas conyugales.		
Atributos	**Tipo**	**Descripción**
VISCONYUGAL_ID	NUMBER(19,0)	Guarda el identificador de la visita conyugal.
FECHA_SALIDA	VARCHAR(255)	Guarda la fecha de salida de la visita conyugal.
HORA_SALIDA	VARCHAR(255)	Guarda la hora de salida de la visita conyugal.
OBSERCAVIONES	VARCHAR(255)	Guarda las observaciones de la visita conyugal.
ESTÍMULO	NUMBER(1,0)	Guarda si la visita es por estímulo o no.
POR_PLAN	NUMBER(1,0)	Guarda si la visita es por el plan o no.
CERRADA	NUMBER(1,0)	Guarda si la visita está cerrada o no.
ACTIVA	NUMBER(1,0)	Guarda si la visita está activa o no.

Tabla 19. Descripción de la tabla VISCONYUGAL.

Nombre: VISFAMILIAR		
Descripción: Esta clase guarda las visitas familiares.		
Atributos	**Tipo**	**Descripción**
VISFAMILIAR_ID	NUMBER(19,0)	Guarda el identificador de la visita familiar.
HORA_SALIDA	VARCHAR(255)	Guarda la hora de salida de la visita conyugal.
AUTORIZA_ID	NUMBER(19,0)	Guarda le identificador del funcionario que autoriza la visita.
FUERA_PLAN	NUMBER(1,0)	Guarda si la visita familiar es fuera de plan.
OBSERCAVIONES	VARCHAR(255)	Guarda las observaciones de la visita conyugal.
ESTÍMULO	NUMBER(1,0)	Guarda si la visita es por estímulo o no.
POR_PLAN	NUMBER(1,0)	Guarda si la visita es por el plan o no.
CERRADA	NUMBER(1,0)	Guarda si la visita está cerrada o no.
ACTIVA	NUMBER(1,0)	Guarda si la visita está activa o no.

Tabla 20. Descripción de la tabla VISFAMILIAR.

Nombre: VISFECHAC

Descripción: Esta clase la fecha de la última vista de cada colectivo.		
Atributos	**Tipo**	**Descripción**
VISFECHAC_ID	NUMBER(19,0)	Guarda el identificador de la fecha de la última visita de cada colectivo.
COLECTIVO_ID	NUMBER(19,0)	Guarda el identificador del colectivo que le pertenece la fecha de la última visita.
CONFIGURARV_ID	NUMBER(19,0)	Guarda el identificador de la configuración de la visita.
FECHA_ULTIMA_VISITA	DATE	Guarda la fecha de la última visita de cada colectivo.
FRECUENCIA	NUMBER(19,0)	Guarda la frecuencia de visita del colectivo.

Tabla 21. Descripción de la tabla VISFECHAC.

Nombre: VISITA		
Descripción: Esta clase las visitas realizadas al centro.		
Atributos	**Tipo**	**Descripción**
VISITA_ID	NUMBER(19,0)	Guarda el identificar de la visita.
FECHA_VISITA	VARCHAR(255)	Guarda la fecha de la visita.
HORA_VISITA	VARCHAR(255)	Guarda la hora de la visita.
TIPO_VISITA_ID	NUMBER(19,0)	Guarda el identificador del tipo de visita.

Tabla 22. Descripción de la tabla VISITA.

Nombre: VISPLANVC		
Descripción: Esta clase guarda los planes de visitas conyugales.		
Atributos	**Tipo**	**Descripción**
VISPLANVC_ID	NUMBER(19,0)	Guarda el identificador del plan de vistas conyugales.
AÑO	VARCHAR(255)	Guarda el año del plan.
TRIMESTRE	VARCHAR(255)	Guarda el trimestre del plan.

Tabla 23. Descripción de la tabla VISPLANVC.

Nombre: VISPLANVF		
Descripción: Esta clase guarda los planes de visitas familiares.		
Atributos	**Tipo**	**Descripción**
VISPLANVF_ID	NUMBER(19,0)	Guarda el identificador del plan de vistas familiares.
AÑO	VARCHAR(255)	Guarda el año del plan.

Tabla 24. Descripción de la tabla VISPLANVF.

Nombre: VISPROY		
Descripción: Esta clase guarda los visitantes de un proyecto que realice la visita.		
Atributos	**Tipo**	**Descripción**

VISPROY_ID	NUMBER(19,0)	Guarda el identificador del visitante de proyecto.
PROYECTO_ID	NUMBER(19,0)	Guarda el identificador del proyecto al que pertenece el visitante.
ROL	VRACHAR(255)	Guarda el rol del visitante.

Tabla 25. Descripción de la tabla VISPROY.

Nombre: VISPROYECTO		
Descripción: Esta clase guarda las visitas asociadas a un proyecto.		
Atributos	**Tipo**	**Descripción**
VISPROYECTO_ID	NUMBER(19,0)	Guarda el identificador de la visita del proyecto.
INSTITUCION	VRACHAR(255)	Guarda la institución del proyecto
NOMBRE_PROYECTO	VRACHAR(255)	Guarda el nombre del proyecto
OBJETIVO_VISITA	VRACHAR(255)	Guarda el objetivo de la visita del proyecto

Tabla 26. Descripción de la tabla VISPROYECTO.

Nombre: VISTURNO		
Descripción: Esta clase guarda los turnos de visita.		
Atributos	**Tipo**	**Descripción**
VISTURNO_ID	NUMBER(19,0)	Guarda el identificador del turno.
SECCION	VRACHAR(255)	Guarda el nombre del turno.
DESDE	VRACHAR(255)	Guarda la hora de inicio del turno.
HASTA	VRACHAR(255)	Guarda la hora de fin del turno.
DISPONIBLE	NUMBER(1,0)	Guarda si el turno está disponible o no.

Tabla 27. Descripción de la tabla VISTURNO.

3.4 Conclusiones parciales

En este capítulo se expuso la arquitectura del sistema para una mejor comprensión de la estructura del mismo. Se realizó el análisis y diseño del sistema, obteniéndose los diagramas de clases del análisis, los diagramas de clases del diseño y el modelo de entidad relación, con el objetivo de sentar las bases para la posterior implementación.

Capítulo IV. Implementación y pruebas.

4.1. Introducción.

Partiendo de los resultados del diseño, en el presente capítulo se analiza la disciplina de implementación, en conjunto con los diagramas de despliegue y componentes del sistema. Se definen las pruebas de software como elemento crítico para garantizar la calidad de la solución propuesta y la revisión final del cumplimiento de las especificaciones del diseño.

4.2 Implementación.

En esta disciplina los elementos del diseño, fundamentalmente las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, librerías, entre otros. También se describe la dependencia física entre los componentes del sistema.

4.2.1. Diagramas de componentes.

Los diagramas de componentes muestran un conjunto de componentes y sus relaciones (17). Son empleados para mostrar la estructura del modelo de implementación y las relaciones entre elementos correspondientes a este. En la siguiente figura se muestra el diagrama de componentes para los módulos Visitas Familiares y Visitas Institucionales.

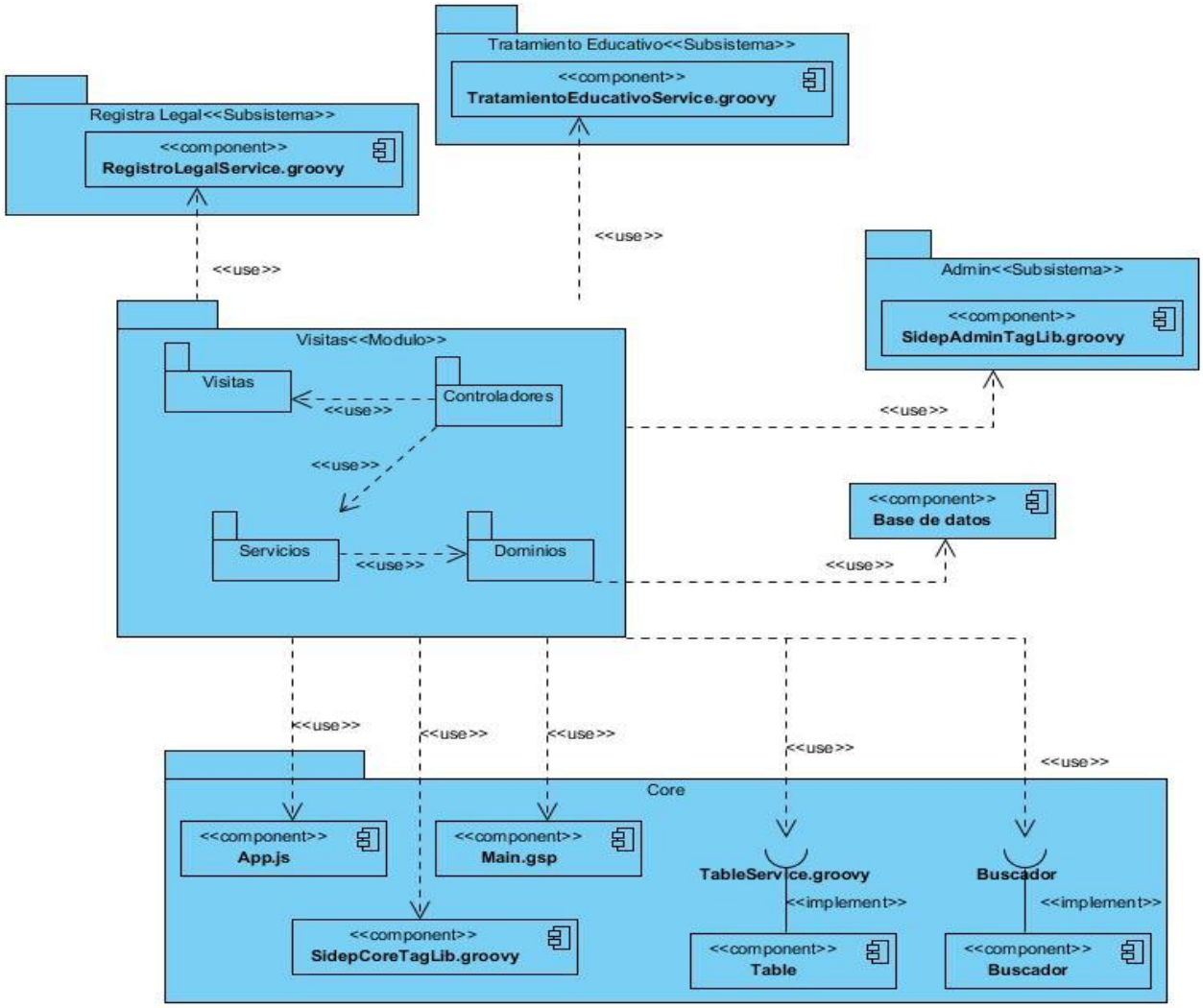

Figura 14. Diagrama de Componentes de los módulos Visitas Familiares y Visitas Institucionales.

En la implementación, los módulos se convierten en paquetes de componentes que interactúan entre ellos. Para el caso del módulo implementado se muestra el paquete Visitas que agrupa los paquetes de componentes Vistas, Controladores, Servicios y Dominios. Además se muestran las relaciones con componentes externos al módulo, que mostrado desde la implementación constituye una agrupación de componentes en forma de paquete.

En la implementación de Visitas se utilizaron algunos componentes de otros subsistemas, tales como: el componente Buscador para realizar la búsqueda de internos y funcionarios, el componente Table que posibilita generar y trabajar con las tablas, el Main.gsp para referenciar los archivos que sirven como plantilla para el SIDEP además de diferentes funciones del componente App.js. Todos los componentes mencionados anteriormente pertenecen al subsistema Core.

Se hizo uso además del SidepAdminTagLib.groovy, componente que permite que los diferentes usuarios puedan iniciar sesión en el sistema y que solo tengan acceso a las diferentes áreas según su rol. Este se encuentra dentro del subsistema Admin. Del subsistema Registro Legal, se necesita la capacidad de los locales de visitas para la generación del plan de visitas, la comunicación con este subsistema se establece mediante el servicio RegistroLegalService.groovy. Además, se obtiene del subsistema Tratamiento Educativo un listado de colectivos del centro para planificarle la visita familiar a estos y los internos estimulados con visitas familiares o conyugales para registrar las visitas por estímulo, la comunicación con esta interfaz se realiza mediante el servicio TratamientoEducativoService.groovy.

La distribución de componentes se realizó en paquetes distribuidos en las diferentes capas que caracterizan la arquitectura del sistema.

4.2.2. Paquete Vistas.

Grails incluye librerías de etiquetas (tags de presentación) que permiten una mejor definición de la lógica de presentación. Además brinda la posibilidad al desarrollador de crear este tipo de etiquetas para mostrar la sección de la vista que cumpla con la condición. En la implementación de las vistas se hizo uso de etiquetas para formularios y las etiquetas lógicas, estas las incorpora Grails

Etiquetas lógicas.

Estas etiquetas permiten la inclusión opcional de código en las vistas. La siguiente figura muestra un fragmento de la implementación de la vista mostrarConsulta.gsp donde se evidencia el uso de esta etiqueta.

```
<b>Tipo de visita:</b> ${visita?.tipoVisita}  <br/>
<b>Fecha de visita:</b> ${visita?.fechaVisita}  <br/>
<b>Hora de Visita:</b> ${visita?.horaVisita}  <br/>

<g:if test="${visita?.tipoVisita.id==1}">
    <b>Hora de Salida:</b> ${visita?.horaSalida}<br/>
    <b>Observaciones:</b> ${visita?.observaciones}<br/>
</g:if>

<g:if test="${visita?.tipoVisita.id==2}">
    <b>Hora de Salida:</b> ${visita?.horaSalida}<br/>
    <b>Fecha de Salida:</b> ${visita?.fechaSalida}<br/>
    <b>Observaciones:</b> ${visita?.observaciones}<br/>
</g:if>

<g:if test="${visita?.tipoVisita.id==4}">
    <b>Funcionario que autoriza:</b> ${visita?.autoriza.toString()}<br/>
</g:if>


<g:if test="${visita?.tipoVisita.id==7}">
    <b>Nombre del Proyecto:</b> ${visita?.nombreProyecto}<br/>
    <b>Institución del Proyecto:</b> ${visita?.institucion}<br/>
    <b>Objetivo de la visita:</b> ${visita?.objectivoVisita}<br/>
</g:if>
```

Figura 15. Uso de la etiqueta g: if en la vista mostrarConsulta.gsp.

La etiqueta*<g: if> </g: if>* comprueba el tipo de visita para mostrar sus detalles en dependencia del tipo.

Etiquetas para formularios.

Estas etiquetas permiten agrupar los campos y enviar los datos contenidos en éstos a los controladores. La siguiente figura muestra un fragmento de la implementación de la vista configuracionVisitas.gsp que evidencia el uso de esta etiqueta, para agrupar todos los campos necesarios para guardar la configuración de la visita, tales como el tipo y los días de visita.

```
<g:form id="configuracionVisita" name="configuracionVisita" method="post">

    <br>
    <s:title title="Tipo de Visita">
        <br/>
        <div id="tipoVisita" class="formpanel2">
            <input dojoType="dijit.form.RadioButton" name="tipo" id="conyugal" value="Conyugal" type="radio"/>
            <label for="conyugal">Vista Conyugal</label>
            <input dojoType="dijit.form.RadioButton" name="tipo" id="familiar" value="Familiar" type="radio"
                   checked="true"/>
            <label for="familiar">Visita Familiar</label>
        </div>
```

Figura 16. Uso de la etiqueta para formulario g: form en la vista configuracionVisitas.gsp.

4.2.3 Paquete Controladores

En los controladores las operaciones se definen como acciones que se ejecutan como parte de la clase. En estas se puede realizar llamadas al método *render* para enviar respuestas al cliente. Durante el desarrollo del software, los parámetros de este método que se utilizaron fueron:

- ***View***: Para mostrar vistas al usuario.
- ***Model***: Un mapa con el modelo para usar en la vista (14).

La siguiente figura muestra un fragmento de la implementación de la clase controladora consultarVisitasFamiliaresController.groovy que evidencia el uso de estos parámetros.

```
def show = {

    def vis = consultarVisitasFamiliaresService.getVisita()
    if (!consultarVisitasFamiliaresService.idVisita) {
        redirect(action: "index")
        return
    }

    render view: "/visitas/mostrarConsulta", model: [visita: consultarVisitasFamiliaresService.getVisita()]
}
```

Figura 17. Ejemplo del uso de los parámetros del método render en la clase controladora consultarVisitasFamiliaresController.groovy.

La clase controladora consultarVisitasFamiliaresController.groovy es la encargada de enviar los datos obtenidos en el servicio consultarVisitasFamiliaresService.groovy a la vista mediante un modelo.

4.2.4. Paquete Dominios

Grails utiliza GORM para controlar el ciclo de vida de las clases contenidas en este paquete. Todas estas clases serán tratadas por GORM como entidades, este se encarga de generar la tabla correspondiente en la base de datos con los campos necesarios para almacenar la información y proporcionar métodos que permitan manipular la entidad.

4.2.4.1. Relaciones entre entidades

Relaciones Uno-A-Uno

Las relaciones de este tipo existen, por ejemplo, entre un visitante de proyecto (VisitanteProyecto) y un proyecto, en la siguiente figura que se muestra la representación esta relación en la clase de dominio VisitanteProyecto.

```
class VisitanteProyecto extends Persona {

    String rol

    static belongsTo = [proyecto: Proyecto]

    static mapping = {
        table 'visProy'
        id column: 'visProy_ID'
    }

    static constraints = {
        rol nullable: false
        proyecto nullable: false
    }
}
```

Figura 18. Clase del dominio VisitanteProyecto.groovy.

La propiedad estática *belongsTo* establece que existe una relación de subordinación entre el visitante de proyecto y el proyecto, de manera que desaparece un proyecto se eliminaran también sus visitantes.

Relaciones Uno-A-Muchos

Las relaciones de este tipo se evidencian, entre una VisitaFamiliar y sus visitantes, ya que una visita familiar puede tener uno o muchos visitantes. La siguiente figura muestra la forma de representar esto en las clases de dominio, en el que se hace uso de la propiedad *hasMany.*

```
class VisFam extends Visita {

    String horaSalida
    String observaciones
    Boolean fueraPlan = false
    Funcionario autoriza
    Boolean estimulo = false
    Boolean porPlan = false
    Boolean cerrada = false

    static hasMany = [vistantes: Vinculo]

    static mapping = {
        table 'visFamiliar'
        id column: 'visFamiliar_ID'
        tablePerHierarchy(false)
    }

    static constraints = {
        horaSalida nullable: true
        observaciones(nullable: true)
        autoriza(nullable: true)
    }
```

Figura 19.Clase del dominio VisFam.groovy.

4.2.4.2. Operaciones sobre el modelo de datos

Para la manipulación de las entidades de la aplicación a continuación se explican los métodos usados y se muestran ejemplos de su utilización:

- ***save():*** Se utiliza para insertar el registro en la base de datos del sistema o para actualizarlo si ya existía. Para enviar los datos en el mismo momento en que se invoca este método, se utiliza el argumento *flush:true* (14). La figura que se muestra a continuación evidencia el uso de este método en la clase guardarVisitaConsularService.groovy específicamente en el método guardarVisitaConsular el cual es el encargado de salvar una visita consular.

```
boolean guardarVisitaConsular(params) {
List expedientes = agregarInternosConsularService.listaInternos

    VisitaConsular visitaConsular = new VisitaConsular()

    visitaConsular.fechaVisita = new SimpleDateFormat("yyyy-MM-dd").parse(params.fechaConsular)
    visitaConsular.horaVisita = params.horaInicioConsular
    visitaConsular.pais = NomPais.get(params.long("paisConsul"))
    visitaConsular.tipoVisita = NomTipoVisita.get(5)

     for(it in expedientes) {
        visitaConsular.addToInternos(it)
    }

    if (visitaConsular.validate()) {
        visitaConsular.save(flush: true)
    }
    else {
        visitaConsular.errors.allErrors.each {println it}
        return false
    }
    return true
```

Figura 20. Uso del método save().

- ***delete():*** Se utiliza para eliminar un registro. La figura que se muestra a continuación evidencia el uso de este método en la configurarVisitasService.groovy específicamente en el método salvarConfiguracion, en el cual para registrar una configuración de visita primero se verifica si existe alguna y en caso de existir se elimina para guardar la nueva.

```
boolean salvarConfiguracion (ConfigurarVisita conf, def listaDias) {
    try {

        def existe = ConfigurarVisita.findByTipoVisita(conf.tipoVisita)

        if (existe) {
            existe.delete()
        }

        def listaTurnos = guardarTurnoService.items
        def listaColectivo = guardarColectivosService.items

        listaColectivo.each {
            String dias = it.frecuencia
            def dia = dias.split(" ")
            def l = Colectivo.get(Long.parseLong(it.idColectivo))
            def c = new FechaColectivo(frecuencia: dia[0], fechaUltimaVisita: it.fecha,
            conf.addToFechaColectivo(c)
        }
```

Figura 21. Uso del método delete().

Dynamic Finders(“localizadores dinámicos”)

Estos buscadores son muy potentes para realizar consultas básicas, pero no permiten hacer búsquedas avanzadas. En algunas clases servicios se utilizaron los métodos:

- ***findAll()*:** Devuelve la primera instancia que cumpla la primera condición de búsqueda.
- **findAllBy():** Devuelve una lista con todos los resultados que correspondan

En la figura que se presentan a continuación muestra un fragmento de la implementación de la clase registrarVisitasFueraPlanService.groovy específicamente del método guardarVisitaFamiliar donde se evidencia el uso de *findBy()* para buscar al interno con número de expediente igual al enviado por la vista como parámetro.

```
boolean guardarVisitaFamiliar(params) {

    Expedientelegal expelegal = new Expedientelegal()
    expelegal = Expedientelegal.findByNumeroexpediente(params.expedientelegal.numeroexpediente)
    Funcionario func=Funcionario.get(params.funcionarioID.toString().toLong())
```

Figura 22.Uso del método findBy().

En la figura que se presentan a continuación muestra un fragmentos de la implementación de la clase familiarVinculosXInternoService.groovy específicamente el método visitantesPorInterno, en el cual se utiliza *findAllBy()* para buscar a todos los familiares que tienen derecho de visitar a un interno determinado, este devuelve una lista con todos los resultados.

```
List visitantesPorInterno(def idInterno) {
    /* obteniendo listado de internos-vinculos a partir del id del individuo */
    def vinculos = InternoVinculo.findAllByInterno(Interno.get(idInterno))
    List items = []
     if(vinculos)
    for(InternoVinculo var : vinculos){
```

Figura 23. Uso del método findAllBy().

Criteria

Permite hacer búsquedas avanzadas que incluyen varios campos y comparaciones complejas (14). La siguiente figura muestra un fragmento de la implementación de la clase consultarVisitasInstitucionalesService.groovy, en la cual utiliza el Criteria con el objetivo de buscar todas las visitas que cumplan con los criterios de búsqueda, devuelve una lista con todos resultados.

```
def resultado = Visita.createCriteria().list {
    and {
        if (params.consulta == "Dias") {
            eq("fechaVisita", new SimpleDateFormat("yyyy-MM-dd").parse(params.fecha))
        } else {
            between("fechaVisita", inicio, fin)
        }
         tipoVisita {
            or {
                eq("id", (long) 8)
                eq("id", (long) 7)
                eq("id", (long) 6)
                eq("id", (long) 5)
            }
        }
    }
```

Figura 24. Criteria aplicado a la entidad Visita.groovy.

4.2.5. Servicios

Los servicios son los responsables de implementar la lógica de negocio de la aplicación, es decir, son los encargados de recibir la información enviada por los controladores y realizar operaciones con estas sobre las entidades. La utilización de los servicios posibilita la reutilización del código, ya que se puede utilizar el mismo servicio para realizar varias actividades Por defecto todos los servicios utilizan el patrón singleton: solo existe una instancia de la clase, lo que trae como consecuencia que todos los controladores vean la misma instancia del servicio y el mismo valor. Este comportamiento se puede modificar declarando una variable “scope” en el servicio con el valor:

- session: Se creará una instancia del servicio para cada sesión del usuario.

La siguiente figura muestra un fragmento de código del controlador configuracionVisitaController.groovy que mediante la inyección de dependencia crea una instancia del servicio configurarVisitasService.groovy para acceder a las funcionalidades del mismo a utilizar en el controlador.

```
class ConfiguracionVisitaController {

    def configurarVisitasService
    def guardarColectivosService
    def guardarTurnoService
    def generarPlanVisitaFamiliarService
```

Figura 25. Instancia del servicio configurarVisitaService.groovy

La figura que se muestra a continuación representa un fragmento de código del servicio configurarVisitaService.groovy, en el que muestra el uso de otros servicios y la implementación de las diferentes operaciones necesarias para guardar la configuración.

```
class ConfigurarVisitasService {

    static scope = "session"
    def guardarColectivosService
    def guardarTurnoService

    List getColectivos () {
        def result = Colectivo.list()
        def colectivos = []
        colectivos << [id: -1, descripcion: "Seleccione..."]
        result.each {
            colectivos << [id: it.id, descripcion: it.numeroColectivo]
        }
        return colectivos
    }

    boolean salvarConfiguracion (ConfigurarVisita conf, def listaDias) {...}

    private def getValidDates (def listaDias) {...}
```

Figura 26. Fragmento de código del servicio configurarVisitaService.groovy

4.2.6. Diagrama de despliegue.

El diagrama de despliegue muestra un conjunto de nodos y sus relaciones, muestra un sistema desde el punto de vista estático (17). Describen la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. En la siguiente figura se muestra el diagrama de despliegue De la solución propuesta.

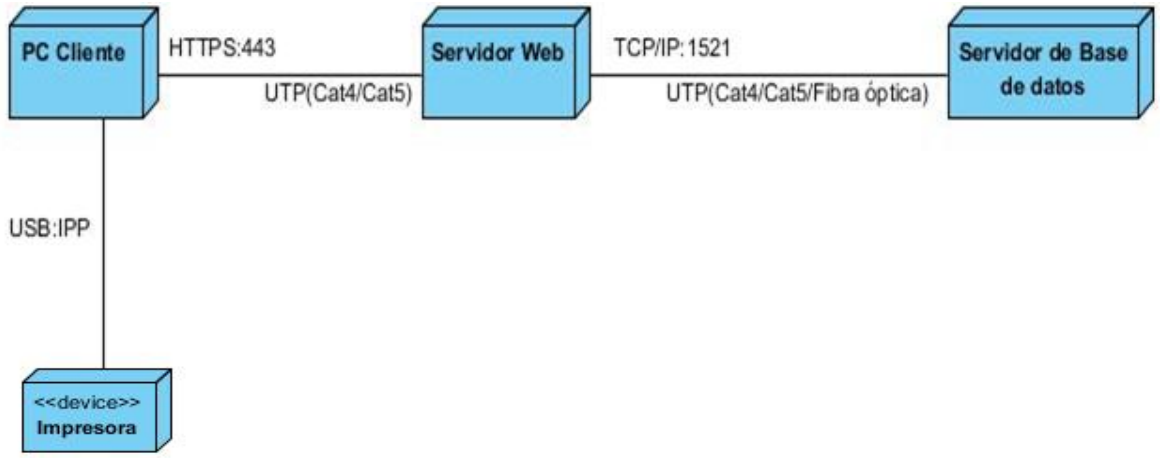

Figura 27. Diagrama de despliegue.

Para el correcto despliegue de los módulos se necesita una estructura tecnológica, donde radican el Servidor de aplicaciones y el de Base Datos.

Servidor de Aplicación:

- Trabajará sobre la distribución de Linux Suse Enterprise Server 10.0 SP2 y el servidor web Apache Tomcat 6.0.3.
- Microprocesador: 4 núcleos, 3 GHz
- RAM: 4 GB
- Espacio necesario para la instalación: 250 MB
- Espacio libre requerido: 250 GB
- JDK 1.6

Servidor de Bases de datos:

- Microprocesador: 4 núcleos, 3 GHz
- RAM: 4 GB
- Espacio necesario para la instalación: 2 GB
- Espacio libre requerido: 1 TB
- JDK 1.6
- Se utilizará Oracle 11G Enterprise Edition como sistema gestor de base de datos.

El cliente hará uso de la aplicación a través de clientes ligeros con sistema operativo Windows XP SP3 y navegador Mozilla Firefox 3.6. Se necesitará una impresora para la impresión de los planes de visita.

La estructura anteriormente documentada fue propuesta por el equipo de arquitectura del SIDEP.

4.3 Pruebas.

Las pruebas de software se integran dentro de las diferentes fases del ciclo del software permiten verificar y revelar la calidad de un software. Las pruebas se realizan para identificar posibles fallos de implementación o usabilidad del sistema informático. Existen diferentes tipo y métodos de pruebas que juntos conforman la estrategia de prueba a seguir para lograr la calidad del producto final.

4.4 Estrategia de Prueba.

La estrategia de prueba describe el enfoque y los objetivos generales de las actividades de la prueba. Incluye los niveles, el tipo y los métodos de prueba a aplicar.

4.4.1 Niveles de Pruebas.

Para lograr una mayor efectividad, las pruebas de software se realizan a diferentes niveles, en las pruebas realizadas a los módulos Visitas Familiares y Visitas Institucionales se distinguen los siguientes niveles de pruebas:

- **Prueba de desarrollador:** Es la prueba diseñada e implementada por el equipo de desarrollo.
- **Pruebas de Sistema**: Este tipo de pruebas tiene como propósito ejercitar profundamente el sistema para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones adecuadas (21).

4.4.2 Tipos de prueba.

Como parte de la estrategia de prueba, se definió como tipo de prueba:

- **Funcionalidad:** Determinan que la aplicación satisface los requisitos funcionales esperados. Este proceso simulará varios escenarios para confirmar que todos los resultados satisfacen las expectativas establecidas (22).

4.4.3 Métodos de Pruebas.

Los métodos de prueba son utilizados con el propósito de descubrir fallos y para descubrir fallos en el funcionamiento del software. En las pruebas realizadas a la aplicación se hizo uso del método de Caja Negra.

- **Pruebas de Caja Negra:** Estas pruebas se centran en los requisitos funcionales. Su objetivo es saber qué es lo que hace el sistema sin dar importancia a cómo lo hace y permiten encontrar funciones incorrectas o ausentes y errores de interfaz, estructuras de datos, inicialización y terminación (20).

Como parte de la estrategia de prueba se diseñaron los casos de pruebas para cada funcionalidad, con el objetivo de probar que el sistema funciona correctamente. Para consultar el caso de prueba de la funcionalidad Registrar visitas familiares, ver Anexo 7.

4.4.4 Resultados de las pruebas

Para evaluar la solución desarrollada se realizaron pruebas en las cuales se aplicaron los niveles, método y tipos de prueba expuestos anteriormente, mediante el uso de diseño de casos de prueba lo que arrojó resultados visibles. En estas pruebas se detectaron 9 no conformidades de las cuales todas fueron resueltas.

No.	No conformidad	Ubicación	Estado
1.	Validar que la hora fin del turno sea mayor que la hora inicio	Configuración de Visita	Resuelta
2.	Cambiar mensaje “Debe seleccionar el Tipo y Días de Visitas”	Configuración de Visita	Resuelta
3.	Cambiar error ortográfico en el campo “Días de Visita”	Configuración de Visita	Resuelta
4.	Cambiar error ortográfico en el campo “Sábado”	Configuración de Visita	Resuelta
5.	Validar que se seleccione al menos un vínculo para guardar la visita	Visita Familiar, visita Fuera de Plan	Resuelta
6.	Agregar botón “Cancelar”	Visita Fuera de Plan, visita al ingreso	Resuelta
7.	Cambiar el campo observaciones que puede ser nulo	Visita Familiar y Visita Conyugal	Resuelta
8.	Validar que las tablas tengan al menos un elemento para guardar la visita.	Todas las visitas	Resuelta
9.	Validar que la hora de salida sea mayor que la hora de la visita	Visita Familiar y Visita Conyugal	Resuelta

Tabla 28. Resultados de las pruebas.

4.4. Conclusiones parciales

Con la implementación del sistema se logró satisfacer las necesidades de los usuarios de acuerdo a los requisitos capturados. Además se realizaron los diagramas de componentes y de despliegue como parte de la disciplina de implementación. Se realizaron pruebas a la aplicación con el objetivo de detectar deficiencias en la misma. Las deficiencias encontradas fueron solucionadas lográndose el correcto funcionamiento del sistema.

Conclusiones generales

Con la realización de este trabajo se logró el desarrollo de los módulos Visitas Familiares y Visitas Institucionales, alcanzándose la gestión de las visitas en el SIDEP, por lo que se puede concluir que:

- El análisis de soluciones informáticas existentes en Sistemas Penitenciarios, tanto nacionales como internacionales permitió demostrar que estas no se correspondían con las características de las visitas del Sistema Penitenciario Cubano por lo que fue necesario implementar un nuevo sistema.
- Durante el análisis del sistema se obtuvieron las características de la aplicación a desarrollar, así como las principales funcionalidades de la misma, lo que posibilitó la elaboración de los artefactos derivados del flujo de análisis con el objetivo de tener una tener una idea clara y precisa sobre el diseño a realizar.
- El diseño de los módulos Visitas Familiares y Visitas Institucionales, demostró que este constituye un paso fundamental para un desarrollo correcto de la implementación, además mostró una vista lógica del sistema y los elementos que componen el mismo, para la posterior implementación de los módulos dando respuesta a los requisitos previstos con el cliente.
- Aplicando los métodos y tipos de pruebas, se logró una revisión detallada de cada uno de los componentes, lo que permitió detectar y corregir errores existentes.

Recomendaciones

Como resultado del presente trabajo se obtuvo un sistema informático que resuelve la problemática planteada inicialmente, a continuación se exponen algunas recomendaciones a tener en cuenta para futuras versiones del producto:

- Agregarle nuevas funcionalidades que brinden otros tipos de informaciones y servicios al usuario.
- Realizar Pruebas de aceptación y de liberación a los módulos con el objetivo de desplegar la aplicación en los centros penitenciarios.
- Tener en cuenta este trabajo para proyectos futuros, siempre y cuando se cumpla con las normas de confidencialidad establecidas.

Referencias bibliográficas

1. MINREX. [En línea] [Citado el: 21 de 5 de 2012.] http://www.cubaminrex.cu/derechos%20humanos/articulos/consejoderechoshumanos/Informe/Espannol/Sist_Penitenciario.html.

2. **DEP y MININT.** *Reglamento Penitenciario Cubano.*

3. **MININT.** *Informe de factibilidad e ideas preliminares sobre la informatización del sistema penitenciario cubano.* Habana : s.n.

4. **Corrections.com.** Offendertrak Corrections Management System. [En línea] Corrections Media, 18 de enero de 2001. [Citado el: 8 de 6 de 2012.] www.corrections.com/articles/7279.

5. **Syscon.** Syscon. [En línea] [Citado el: 10 de 6 de 2012.] http://www.syscon.net/Products/elite-jails.

6. **Spillman .** Spillman Corrections Management System. [En línea] [Citado el: 10 de 6 de 2012.] http://www.spillman.com/corrections/.

7. Dirección General del Sistema Penitenciario. [En línea] [Citado el: 10 de 6 de 2012.] http://www.sistemapenitenciario.gob.pa.

8. SDSC. [En línea] [Citado el: 5 de 6 de 2012.] http://www.sdscnet.org/jms.html.

9. [En línea] [Citado el: 4 de 6 de 2012.] http://www.institucionpenitenciaria.es/web/export/sites/default/datos/descargables/instruccionesCirculares/I-5-2007_SISTEMA_IDENTIFICACION_AUTOMATIZADO.pdf.

10. CISCO. [En línea] [Citado el: 5 de 6 de 2012.] http://www.cisco-ps.com/jams/.

11. **Jacobson, Ivor, Brooch, Gravy y Baugh., James Rum.** *El Lenguaje Unificado de Modelado .*

12. Object Management Group Business Process Model and Notation. [En línea] [Citado el: 10 de 6 de 2012.] http://www.omg.org/bpmn/Documents/Introduction_to_BPMN.pdf.

13. MuleSoft. [En línea] 2009. [Citado el: 4 de 6 de 2012.] http://www.mulesoft.com/understanding-apache-tomcat.

14. **Brito Calahorro, Nacho.** *Manual de desarrollo web con Grails.* 2009.

15. **Sommerville, Ian.** *Ingeniería de software.*

16. SCIELO. [En línea] [Citado el: 8 de 6 de 2012.] http://www.scielo.org.co/scielo.php?pid=S1794-12372007000100003&script=sci_arttext.

17. **Jacobson, Ivor, Brooch, Gravy y Baugh. , James Rum.** *El Proceso Unificado de Desarrollo de Software.*

18. **ÖVERGAARD, GUNNAR.** *Use Cases Patterns and Blueprints.Addison Wesley Professional.*

19. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 1999. págs. 189-424. Vol. 1ra Edicion.

20. **Pessman, Roger S.** *Ingeniería de Software un enfoque práctico.5ta Edición.* 1997.

21. **Natalia Juristo, Ana M. Moreno, Sira Vegas.** *Técnicas de evaluación de software.* 2005.

22. CSOFT. [En línea] [Citado el: 11 de 6 de 2012.] http://es.csoftintl.com/solutions_test_software.php.

Bibliografía

1. MINREX. [Online] [Cited: 5 21, 2012.] http://www.cubaminrex.cu/derechos%20humanos/articulos/consejoderechoshumanos/Informe/Espannol/Sist_Penitenciario.html.

2. **DEP and MININT.** *Reglamento Penitenciario Cubano.*

3. **MININT.** *Informe de factibilidad e ideas preliminares sobre la informatización del sistema penitenciario cubano.* Habana : s.n.

4. **Corrections.com.** Offendertrak Corrections Management System. [Online] Corrections Media, e nero 18, 2001. [Cited: 6 8, 2012.] www.corrections.com/articles/7279.

5. **Syscon.** Syscon. [Online] [Cited: 6 10, 2012.] http://www.syscon.net/Products/elite-jails.

6. **Spillman .** Spillman Corrections Management System. [Online] [Cited: 6 10, 2012.] http://www.spillman.com/corrections/.

7. Dirección General del Sistema Penitenciario. [Online] [Cited: 6 10, 2012.] http://www.sistemapenitenciario.gob.pa.

8. SDSC. [Online] [Cited: 6 5, 2012.] http://www.sdscnet.org/jms.html.

9. [Online] [Cited: 6 4, 2012.] http://www.institucionpenitenciaria.es/web/export/sites/default/datos/descargables/instruccionesCirculares/I-5-2007_SISTEMA_IDENTIFICACION_AUTOMATIZADO.pdf.

10. CISCO. [Online] [Cited: 6 5, 2012.] http://www.cisco-ps.com/jams/.

11. **Jacobson, Ivor, Brooch, Gravy and Baugh., James Rum.** *El Lenguaje Unificado de Modelado .*

12. Object Management Group Business Process Model and Notation. [Online] [Cited: 6 10, 2012.] http://www.omg.org/bpmn/Documents/Introduction_to_BPMN.pdf.

13. MuleSoft. [Online] 2009. [Cited: 6 4, 2012.] http://www.mulesoft.com/understanding-apache-tomcat.

14. **Brito Calahorro, Nacho.** *Manual de desarrollo web con Grails.* 2009.

15. **Sommerville, Ian.** *Ingeniería de software.*

16. SCIELO. [Online] [Cited: 6 8, 2012.] http://www.scielo.org.co/scielo.php?pid=S1794-12372007000100003&script=sci_arttext.

17. **Jacobson, Ivor, Brooch, Gravy and Baugh. , James Rum.** *El Proceso Unificado de Desarrollo de Software.*

18. **ÖVERGAARD, GUNNAR.** *Use Cases Patterns and Blueprints.Addison Wesley Professional.*

19. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 1999. pp. 189-424. Vol. 1ra Edicion.

20. **Pessman, Roger S.** *Ingeniería de Software un enfoque práctico.5ta Edición.* 1997.

21. **Natalia Juristo, Ana M. Moreno, Sira Vegas.** *Técnicas de evaluación de software.* 2005.

22. CSOFT. [Online] [Cited: 6 11, 2012.] http://es.csoftintl.com/solutions_test_software.php.