Universidad e las Ciencias Informáticas

Facultad 4

Título*: Diseño e implementación de la capa de presentación del módulo Visitas Institucionales.*

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

Autores: Renier Martínez Guzmán
Orestes Febles Díaz
Tutores: Dra. Vivian Estrada Sentí
Dr. Juan Pedro Febles Rodríguez

Mayo, 2008

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 _____.

Renier Martínez Guzmán

_____

Firma del Autor

Orestes Febles Díaz

_____

Firma del Autor

Vivian Estrada Sentí

_____

Firma del Tutor

Juan Pedro Febles Rodríguez

_____

Firma del Tutor

DATOS DE CONTACTO

Dra. Vivian Estrada Sentí

Graduado: Licenciada en Matemática-Especialista en Computación

Grado Científico: Doctora en Ciencias Técnicas.

Categoría Docente: Profesora Titular

Labor que desempeña: Asesora del Ministerio de Educación Superior (desde 1996)

Líneas de investigación y/o trabajo profesional que desarrolla y las investigaciones más importantes realizadas en los últimos cinco años:

Ha dirigido y desarrollado varios proyectos de investigación científica relacionados con la Tecnología Educativa, Aplicaciones Informáticas, la gestión del conocimiento, la inteligencia artificial (aplicada a la salud, la educación, el deporte,…), Informática en la salud y Ciencias de la Educación.

Ha tutorado de manera sistemática trabajos de diploma, tesis de maestría y doctorado tanto en Cuba como en el extranjero.

Imparte docencia en varias maestrías de Tecnología educativa, Informática y Educación, Gestión de la Información, Informática Aplicada, Ciencias de la Educación, Bioinformática, Computación Aplicada, Dirección y otras. Ha sido profesora invitada en varias universidades de Brasil y México y ha desarrollado actividades de investigación científica en el ámbito nacional e internacional. Ha sido asesora del rector en Universidad Federal de Roraima- Brasil, Universidad de Matanzas y asesora de la Superintendencia Nacional de la CNEC- Brasilia. Desde 1996 es asesora del Ministerio de Educación Superior de Cuba. Es profesora Titular adjunta de la Universidad de Guadalajara en México. Es miembro del grupo de expertos del Programa Nacional Científico – Técnico de Tecnología de la Información y miembro del grupo de expertos para la evaluación de los Proyectos Nacionales de Informática Educativa del Programa Ramal de Investigaciones Pedagógicas. Es miembro del claustro del doctorado curricular “La Información en la dirección estratégica y viabilidad de empresas” que se imparte por la Universidad de Málaga, España, del Doctorado en Educación de la Universidad Central de Las Villas y de la Universidad de La Habana y del Doctorado en Educación en la especialidad de Cultura Física en Cuba. Ha participado en Comisiones de carrera para el diseño curricular en Informática y también en el diseño de varias especializaciones y maestrías de perfil Informático, educación, Gestión, diseño curricular.

Ha participado como ponente en mas de 40 eventos internacionales y ha impartido conferencias en varios de ellos de reconocido prestigio (Cuba, España, México, Brasil, Santo Domingo, Guatemala, Colombia, eventos de la Unión Europea) en temáticas tales como: Inteligencia artificial, Tecnología educativa, Las TIC en la educación, Gestión de la Información y el conocimiento, Inteligencia Artificial, Multimedias Educativas, Informática Educativa, Internet y las perspectivas educativas, Minería de datos, Sistemas operativos distribuidos, etc. Dirige un proyecto Nacional de Tecnología educativa y es miembro de otros dos proyectos uno de ellos de carácter internacional (con España). Pertenece a la RedEVA (Espacios Virtuales de Aprendizaje) que coordina la Universidad Internacional de Andalucía (UNIA) y a la Red Internacional RedEspecial. Atendió por varios años el Programa Nacional de Computación en el país. Es miembro del Comité organizador del Congreso internacional de Informática en la Educación, del Congreso Internacional de Gestión Empresarial y Administración Pública, del Congreso RedEspecial de España y otros.

Es miembro del Tribunal Nacional de Grados Científicos de Computación y Control Automático. Es miembro de la Secretaría Ejecutiva de La Junta de Acreditación Nacional (de Carreras, Instituciones y Maestrías) y del Comité Técnico Evaluador de programas académicos de Maestrías en el país para la evaluación y acreditación.

Investigaciones:

- La gestión del conocimiento y la formación de gestores de conocimiento. Proyecto de investigación “Investigación de alternativas y desarrollo del Modelo para la Informatización de la gestión económica de medianas empresas”.
- Diseño e Implementación de Cursos Virtuales. Modelos Educativos.
- Tecnología Educativa. Metodología y elaboración de cursos y aplicaciones
- Informática aplicada a la salud.
- Gestión del Conocimiento. Centro Virtual (proyecto en colaboración con España)- La enseñanza de la Informática y el desarrollo de la creatividad y la Educación a Distancia. Creatividad informática.
- Informática Educativa:”La Informatización y las carreras de informática en la Universalización”. Educación a Distancia.
- Inteligencia Artificial y sus aplicaciones. El Razonamiento Basado en Casos en el tratamiento de temas médicos.

Dr. Juan Pedro Febles Rodríguez:

Ocupación actual: Director del Centro Nacional de Bioinformática

Tiempo en el cargo: 5 años

Formación académica:

- Licenciado en Matemática, especialización en Computación. Universidad Central de las Villas, 1974.
- Master en Informática 1979
- Doctor en Ciencias Técnicas, Especialidad Computación, IPSJAE, 1986.
- Profesor Titular, 1998.
- (Es Profesor Titular adjunto del ISPJAE, el CECAM, el ISCTN, la Universidad de la Habana y la Universidad de las Ciencias Informáticas.)

Actividades de carácter científico o profesional desarrolladas en los últimos 5 años.

Proyectos Concluidos

- Sistema Experto para la ayuda a la Medicina Tradicional Asiática (SAMTA). Bases de conocimiento que emplea razonamiento basado en casos como técnica de Inteligencia artificial.
- Sistema Experto para el Aprendizaje y Diagnóstico de Malformaciones Congénitas (MALCOM).
- Ingeniería de Software y metodologías de investigación en los Sistemas de Gestión en los CEMS.
- Aprendizaje significativo en la enseñanza de Tecnologías de la Información.
- La Educación a Distancia y la enseñanza de postgrado.
- La gestión del conocimiento en la educación Superior.
- Razonamiento basado en casos para la enseñanza problémica.
- Aprendizaje Automático en la biología molecular.
- Selección de datos relevantes en familias de proteínas de interés.

Dirección o participación en otros Proyectos o Programas Científico Técnicos.

- Es jefe del Programa Nacional científico tecnológico de Tecnología de la Información.

- Es experto del Programa Nacional de investigaciones básicas en Matemática, Física, Química y Computación.
- Es miembro del Tribunal Nacional para la defensa de Grados Científicos en la especialidad de Ingeniería Industrial.
- Es miembro del Consejo Científico del Instituto Superior de Ciencia y Tecnología Nuclear.
- Es miembro del consejo científico de la Universidad de las Ciencias Informáticas (UCI).
- Es experto del Ministerio de Educación Superior (MES) para acreditación de Programas de Maestrías.
- Es jefe del proyecto de investigación del Centro Nacional de Bioinformática, que pertenece al Programa Nacional de Tecnologías de la Información.
- Es presidente del comité académico de la maestría de Bioinformática

Principales asignaturas que imparte. (Principalmente en programas de doctorado y maestría)

- Metodología de la investigación (Maestría de Bioinformática, Maestría en Gestión de la Información, Doctorado de Tecnología Educativa, Ingeniería Informática).
- Inteligencia artificial (Maestría de gestión del conocimiento Facultad de Economía Universidad de la Habana, Doctorado en Ciencias del Deporte Universidad de Matanzas, Maestría en Informática Médica ISCM-H, Maestría en Bioinformática InSTEC, Maestría en informática aplicada Universidad de Matanzas, Maestría informática Universidad Mariano Gálvez de Guatemala).
- Metodología de la investigación. (Maestría en Bioinformática InSTEC).
- Redes y seguridad de redes TCP-IP (Maestría en Informática Universidad Mariano Gálvez, Guatemala).
- Programación en Java (Maestría en Informática Universidad Mariano Gálvez, Guatemala).
- Programación en Lenguaje PERL (Maestría en Informática Universidad Mariano Gálvez, Guatemala).
- Gestión del conocimiento (Maestría de Gestión del Conocimiento , UH)
- Minería de Datos (Maestría de Gestión del Conocimiento , UH)

Algunas publicaciones

- Febles JP, González A. La Minería de Datos y su aplicación en la Bioinformática. Revista médica ACIMED. 2002
- González A, Febles JP La Bioinformática: algunas precisiones conceptuales. Publicado en las Memorias del Congreso Informática 2002.
- Febles JP, González A. La bioinformática una ciencia de colaboración. Revista CINDE, volumen No 7. Editorial Academia. de la Torre V., Febles J. P. and *_Valles A._* (2004) Nuevas estrategias para representar secuencias peptídicas como entradas de un SVM y su influencia en la predicción de péptidos que se unen a MHC. /Memorias de la X Convención Internacional y Feria Informática 2004./ ISBN: 959-237-117-2
- Torres E., de la Torre V., *_Valles A._*, Espinosa V., Gómez R., Valencia A, Febles J. P. and Pons T. (2004) Sistema para la anotación y la inferencia de interacciones proteína-proteína. /Memorias de la X Convención Internacional y Feria Informática 2004./ ISBN: 959-237-117-2

AGRADECIMIENTOS

A nuestros tutores, por su experiencia y sabiduría.

A Ailyn, por sacar tiempo siempre.

A nuestro jefe de proyecto, por sus conocimientos, por su preocupación.

A Tahirí por las horas que nos dedicó.

A nuestros compañeros de trabajo, por sus sugerencias y críticas, por su afecto.

A la UCI por ser nuestro refugio estos 5 años.

DEDICATORIA

Renier

En primer lugar a mis padres, que gracias a ellos he logrado vencer una etapa importante de mi vida y he llegado hasta aquí.

A mi familia que con altas y bajas siempre ha estado presente.

A mis amigos y a todos aquellos que de una u otra forma tuvieron que ver con mi formación.

Orestes

A mis padres, orgullo infinito de cada mañana, responsables de lo que soy, de lo que seré.

A Amelia, que es la alegría de la casa.

A mi familia, que es mi razón de ser.

A Denise por la sonrisa.

Y a mis abuelos…. que no están.

RESUMEN

En el presente trabajo se hace un estudio del Sistema Penitenciario de la República Bolivariana de Venezuela *y se diseña e implementa la capa de presentación de un sistema informático que automatiza el proceso de planificación y ejecución de visitas institucionales* con el objetivo de mejorar el control de estas actividades para así resolver muchos de los problemas de funcionamiento que se presentan y hacen más acertada la toma de decisiones de las personas involucradas.

Dentro del desarrollo de este trabajo se hace una descripción de los métodos y herramientas utilizadas. En el documento se hace mención igualmente a la arquitectura empleada, así como a los patrones de diseño utilizados con el fin de lograr un mayor entendimiento de la solución. Para el desarrollo de estos módulos se tuvo como base los requisitos del usuario que quedan recogidos en las funcionalidades descritas dentro de este trabajo.

PALABRAS CLAVE

Sistema Penitenciario, Visitas Institucionales, Capa de Presentación, Implementación, SIGEP

TABLA DE CONTENIDOS

INTRODUCCIÓN

Debido a la incidencia de las telecomunicaciones y de la informática, en la oferta de servicios de información, se impone una cultura de gestión de la información de manera que esta esté disponible cuando y donde se necesite y sobre todo por el que la necesite para poder tomar buenas decisiones y que estas sean oportunas.

Los **sistemas de información** son muy importantes dentro de una organización fundamentalmente porque facilitan el trabajo de muchas personas, proporcionan información útil y precisa que es de gran ayuda para los directivos en la toma de decisiones. Algunos autores han dado distintas definiciones de lo que es un **sistema de información.**

- Laudon (1996): "conjunto de componente interrelacionados que capturan, almacenan, procesan y distribuyen la información para apoyar la toma de decisiones, el control, análisis y visión en una institución".
- Senn (1992): “Un conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio. Un sistema de información interactúa con el recurso humano, el cual está formado por las personas que utilizan el sistema, y el equipo computacional: el cual es el hardware necesario para que el sistema de información pueda operar.”

Un sistema de información realiza cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información y cumplen con tres objetivos básicos dentro de las organizaciones:

- Automatización de procesos operativos.
- Proporcionar información que sirva de apoyo al proceso de toma de decisiones.
- Lograr ventajas competitivas a través de su implantación y uso.

1.1. Antecedentes del Control de Visitas en el Sistema Penitenciario Venezolano

El sistema penitenciario venezolano en la actualidad presenta serias deficiencias con respecto al control de las visitas familiares e institucionales. Esta situación se enmarca dentro del estado de deterioro que heredó la República de años de inacción y falta de atención gubernamental a las cárceles venezolanas. En aras de revertir esta grave situación, que influye en todas las áreas del

funcionamiento de las prisiones, se ha concebido un proyecto integral de modernización y humanización del sistema penitenciario y que incluye el reordenamiento de los procesos claves de la institución y la informatización de los mismos.

1.2. Situación actual y descripción del negocio

Las visitas institucionales son aquellas que realizan a las prisiones tanto los funcionarios públicos, como miembros de organizaciones públicas o privadas que para su trabajo requieren ingresar a los establecimientos. En este grupo de visitantes se encuentran los abogados, fiscales y jueces que atienden el proceso penal de los internos y la ejecución de la pena. Por otro lado se incluyen los instructores educativos, deportivos y culturales que intervienen en las actividades de tratamiento que se llevan en los centros para lograr la reinserción social de los privados de libertad. Adicionalmente se incluyen dentro de las visitas institucionales aquellas otras que se realicen eventualmente por otras instituciones u organizaciones con un propósito específico: campañas de vacunación, instituciones de caridad o religiosas, organismos internacionales, equipos de asistencia técnica y otras.

En función de lo anterior, las visitas se pueden clasificar en permanentes, referentes a aquellas personas o instituciones que en cualquier momento se pueden presentar al penal (jueces, abogados y fiscales) y están autorizados a acceder y las no permanentes, aquellas visitas que requieren una autorización previa de algún nivel de dirección según el tipo de visita.

En la actualidad no existe una planificación de estas visitas. El penal es informado de estas por un mensaje de radio que es enviado desde la Dirección General de Custodia y Rehabilitación del Recluso. El registro de la ejecución de las mismas es llevado por la Guardia Nacional, ente armado que no se subordina a la Dirección de Prisiones por lo que esta institución no cuenta con un registro de visitantes. Esta información resulta de gran utilidad tanto para el tratamiento de los internos como para el control mismo de las personas que acceden al penal y la regularidad con que lo hacen.

En el caso de los abogados, muchos acceden al penal en busca de “clientes” y no se verifica que están accediendo a prestar los servicios legales de los internos que lo han nombrado como abogado, ya sea público o privado.

En este contexto de descontrol sobre los visitantes institucionales surge la necesidad de establecer un mecanismo para su planificación y control.

Tras la realización de un estudio a profundidad sobre este proceso fueron identificados una serie de **problemas** que se relacionan a continuación:

- No hay un registro estable de las visitas de diferentes tipos que se realizan al penal.
- La información que se registra en los diferentes centros penitenciarios no es uniforme lo que dificulta su control y análisis posterior.
- No existe un control estricto sobre la planificación y ejecución de las visitas de las diferentes instituciones al penal.
- No existe un registro histórico basado en las visitas institucionales realizadas al penal en cierto período de tiempo, lo cual atenta contra cualquier tipo de planificación y control que se desee realizar.

Para mejorar el funcionamiento de los penales en general se comenzó a desarrollar el proyecto SIGEP (Sistema de Gestión Penitenciaria), y para el perfeccionamiento de los procesos que se han comentado específicamente, se capturaron requisitos y se diseñó e implementó el Subsistema de Seguridad y Custodia y dentro de éste, el módulo de Visitas Institucionales hasta su capa de negocio. Ahora surge la necesidad de diseñar e implementar además la capa de presentación de dicho módulo.

Como propuesta de solución al problema en cuestión se propone como **Objetivo General**: *Realizar el diseño e implementación de la capa de presentación de un módulo perteneciente al subsistema de Seguridad y Custodia que automatice procesos de planificación y ejecución de visitas institucionales dentro del Sistema de Gestión Penitenciaria en Venezuela*.

Basado en lo antes expuesto se define como **objeto de estudio**: *Diseño e implementación de aplicaciones de gestión para la web basada en java y tecnologías Open Source*, siendo el campo de acción: *Ingeniería de software, disciplinas de diseño e implementación, Lenguaje Unificado de Modelado (UML), tecnologías web, programación en java, frameworks, arquitectura de software.*

Para dar solución al problema planteado se plantean las siguientes tareas:

- Asimilar las tecnologías para el proceso de desarrollo del sistema.
- Diseñar a partir de los requisitos de software y el Modelo de Negocio, la capa de presentación de la Planificación y Ejecución de Visitas Institucionales.
- Implementar el diseño realizado de la capa de presentación relacionado con los procesos de Planificación y Ejecución de Visitas Institucionales.
- Realizar pruebas unitarias y de integración.

1.3. Técnicas, métodos y herramientas utilizadas

Para el desarrollo de este trabajo nos guiamos por la metodología propuesta por RUP (Rational Unified Process), aunque en función de las necesidades reales de nuestro proyecto se organizó un flujo de trabajo para los programadores de interfaz de usuario que orienta el mismo hacia la arquitectura propuesta en el SIGEP.

Dentro del conjunto de herramientas y tecnologías empleadas cabe destacar que se utilizó Eclipse como Ambiente Integrado de Desarrollo con los plugins necesarios para darle funcionalidad a la capa de presentación, como herramientas de modelado fueron usados el Visual Paradigm Suite y el ER/Studio, y dentro del marco de tecnologías tuvo un aporte significativo la utilización de Ajax conjuntamente con Dojo.

Todas estas herramientas y tecnologías serán explicadas en el desarrollo de este trabajo.

DESARROLLO

2.1. Aporte de la solución, beneficios y resultados

Una vez realizados el diseño y la implementación de la capa de presentación quedó identificado un flujo de trabajo bien definido para los programadores de interfaz de usuario dentro de la arquitectura del proyecto SIGEP, así como un conjunto de componentes de interfaz y funcionalidades válidas para su reutilización. Dentro de la realización de las pruebas pilotos en los penales venezolanos se probó la solución, obteniéndose resultados satisfactorios que derivaron en un mejor manejo de la información, propiciando así una acertada toma de decisiones por parte de los involucrados.

2.2. Plataforma de Desarrollo

Como plataforma de desarrollo se utilizó la Plataforma Java, específicamente la versión empresarial. Java Platform, Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java con arquitectura de N niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. Java EE incluye varias especificaciones de Interfaz de Programación de Aplicaciones (API), tales como Conectividad de la Base de Datos de Java (JDBC), Invocación de Métodos Remotos (RMI), e-mail, Servicios de Mensajería de Java (JMS), Servicios Web, Lenguaje de Marcas Extensible (XML), entre otros, y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para componentes. Estas incluyen Enterprise JavaBeans, servlets, portlets (siguiendo la especificación de Portlets Java), JavaServer Pages y varias tecnologías de servicios Web. Esto permite al desarrollador crear una Aplicación de Empresa portable entre plataformas y escalable, a la vez integrable con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel. Estos fueron algunos de los principales elementos que indujeron el uso de esta plataforma de desarrollo.

2.3. Lenguaje de Programación

Como lenguaje de programación se usó el lenguaje Java, propio de la antes mencionada plataforma de desarrollo. Entre las ventajas más comunes de este lenguaje podemos encontrar las siguientes(Marañón 1999):

- Orientado a objetos

Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red.

- Distribuido

Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

- Interpretado y compilado a la vez

Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador.

Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time).

- Robusto

Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo de los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.

- Seguro

Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la Red, la seguridad se impuso como una necesidad de vital importancia. A nadie le gustaría ejecutar en

su ordenador programas con acceso total a su sistema, procedentes de fuentes desconocidas. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.

- Indiferente a la arquitectura

Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura, diseñado para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de problemas los soluciona el intérprete de Java.

- Portable

La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas. Estas dos últimas características se conocen como la Máquina Virtual Java (JVM).

- Multihebra

Hoy en día ya se ven como terriblemente limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.

- Dinámico

El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la Red.

2.4. Arquitectura

2.4.1. Arquitectura General

La arquitectura utilizada para desarrollar el módulo de Visitas Institucionales está regida por la arquitectura base del proyecto SIGEP, descrita en el Documento de Arquitectura de Software del mismo, dentro del cual se enmarcan los módulos antes mencionados.

Esta está basada fundamentalmente en los estilos arquitectónicos Cliente-Servidor y Arquitectura en Capas, específicamente una arquitectura de tres capas lógicas fundamentales bien definidas(Pimentel and Rivero 2007) como se muestra en la *figura 1*.

Arquitectura Cliente-Servidor

Consiste básicamente en que un programa cliente realiza peticiones a otro programa servidor que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

La capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores Web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Arquitectura en capas

La plataforma de J2EE define un modelo de programación encaminado a la creación de aplicaciones basadas en n-capas. Típicamente una aplicación puede tener cinco capas diferentes:

- Capa de cliente: Representa la interfaz de usuario que maneja el cliente.
- Capa de presentación: Representa el conjunto de componentes que generan la información que se representará en la interfaz de usuario del cliente. Típicamente se creará a través de componentes basados en Servlets y JSP.
- Capa de lógica de negocio: Contiene nuestros componentes de negocio reutilizables. Normalmente se forma a partir de componentes EJB (Enterprise Java Beans).

- Capa de integración: Aquí se encuentran componentes que nos permiten hacer más transparente el acceso a la capa de sistemas de información. Por ejemplo este es el lugar idóneo para implementar una lógica de objetos de acceso a datos, DAO (Data Access Objects).
- Capa de sistemas de información: Esta capa engloba a nuestros sistemas de información: bases de datos relacionales, bases de datos orientadas a objetos, sistemas heredados, etc.

Las ventajas para la solución informática son palpables. Al tener las capas separadas existe poco acoplamiento entre las mismas, de modo que es mucho más fácil hacer modificaciones en ellas sin que interfieran en las demás. Todo esto redunda en la obtención de mejoras en cuanto a mantenibilidad, extensibilidad y reutilización de componentes.

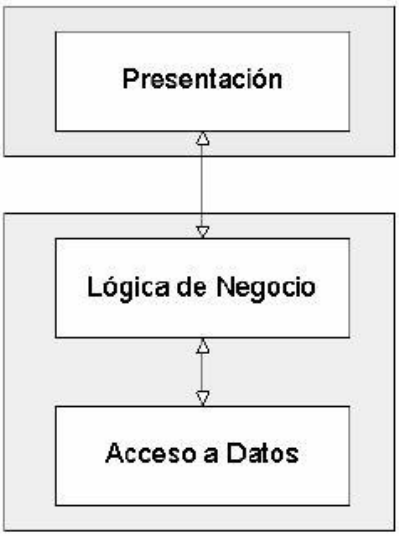

Figura 1 Diagrama de Arquitectura en 3 capas

2.4.1.1. Estructura de paquetes del SIGEP

En la estructura de paquetes del SIGEP se muestran todos los subsistemas (*figura 2*), dentro de los cuales se encuentran los diferentes módulos correspondientes a cada uno de ellos. En la estructura interna de cada módulo se encuentra una organización de paquetes (*figura 3*) en la que están presentes las tres capas lógicas: capa de acceso a datos, capa de lógica negocio, y capa de presentación.

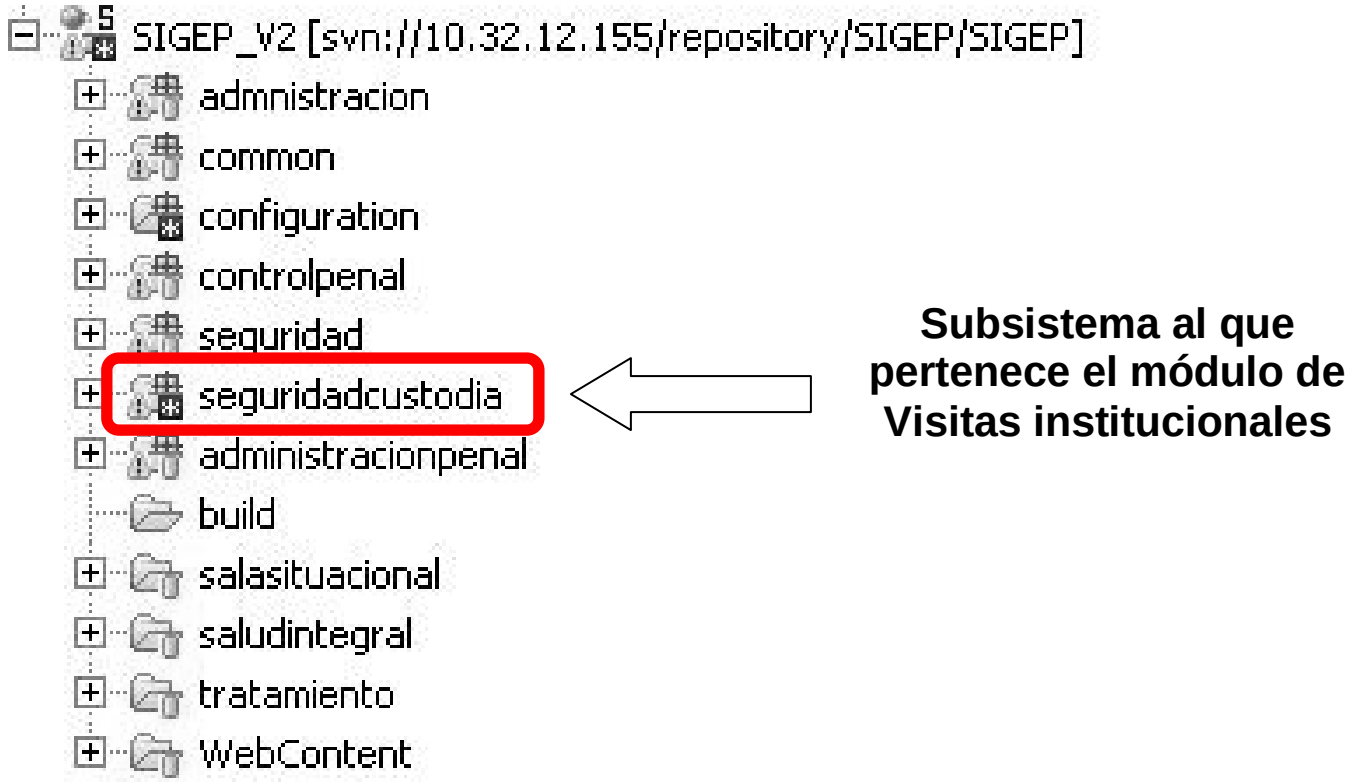

Figura 2 Estructura de paquetes del SIGEP

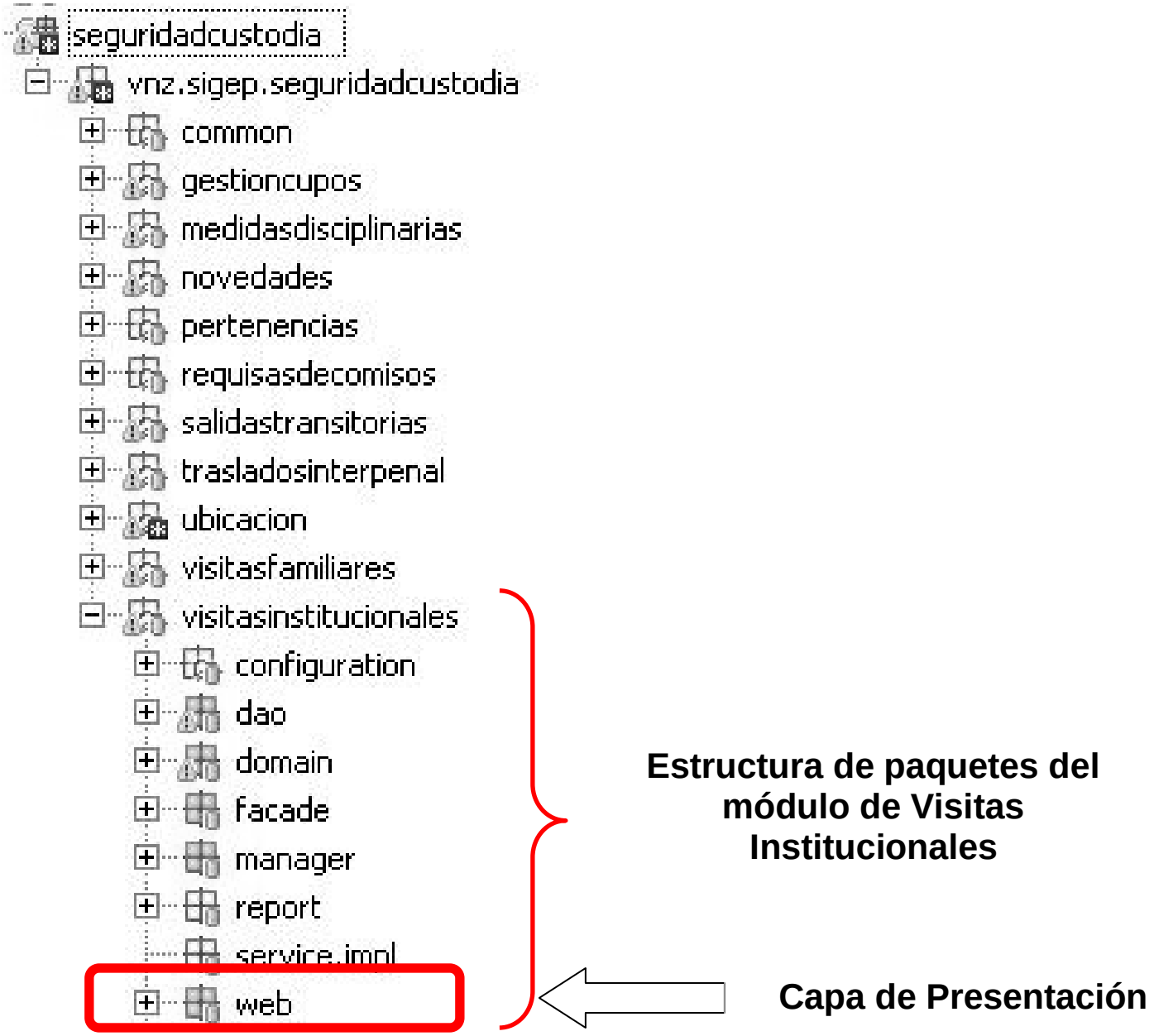

Figura 3 Estructura de paquetes del módulo Visitas Institucionales

2.4.2. Arquitectura de la capa de Presentación

En el subsistema de seguridad y custodia perteneciente al proyecto SIGEP, se encuentra el módulo de Visitas Institucionales y dentro de este existen una serie de carpetas donde se ubican los elementos necesarios para la implementación de la capa de presentación. En la configuración del módulo se incluye un fichero XML donde se mapea la ubicación y composición de los controladores con las peticiones a las que darán respuesta, así como los distintos resolvedores de vistas (viewresolvers) a utilizar (las vistas pueden ser mostradas en varios formatos: JSP, PDF, EXCEL). En la carpeta Web se ubican los controladores, los commands, validators, property editors a usar. Dentro del contexto WebContent están los ficheros js y más específicamente dentro de WEB-INF se encuentran las páginas jsp.

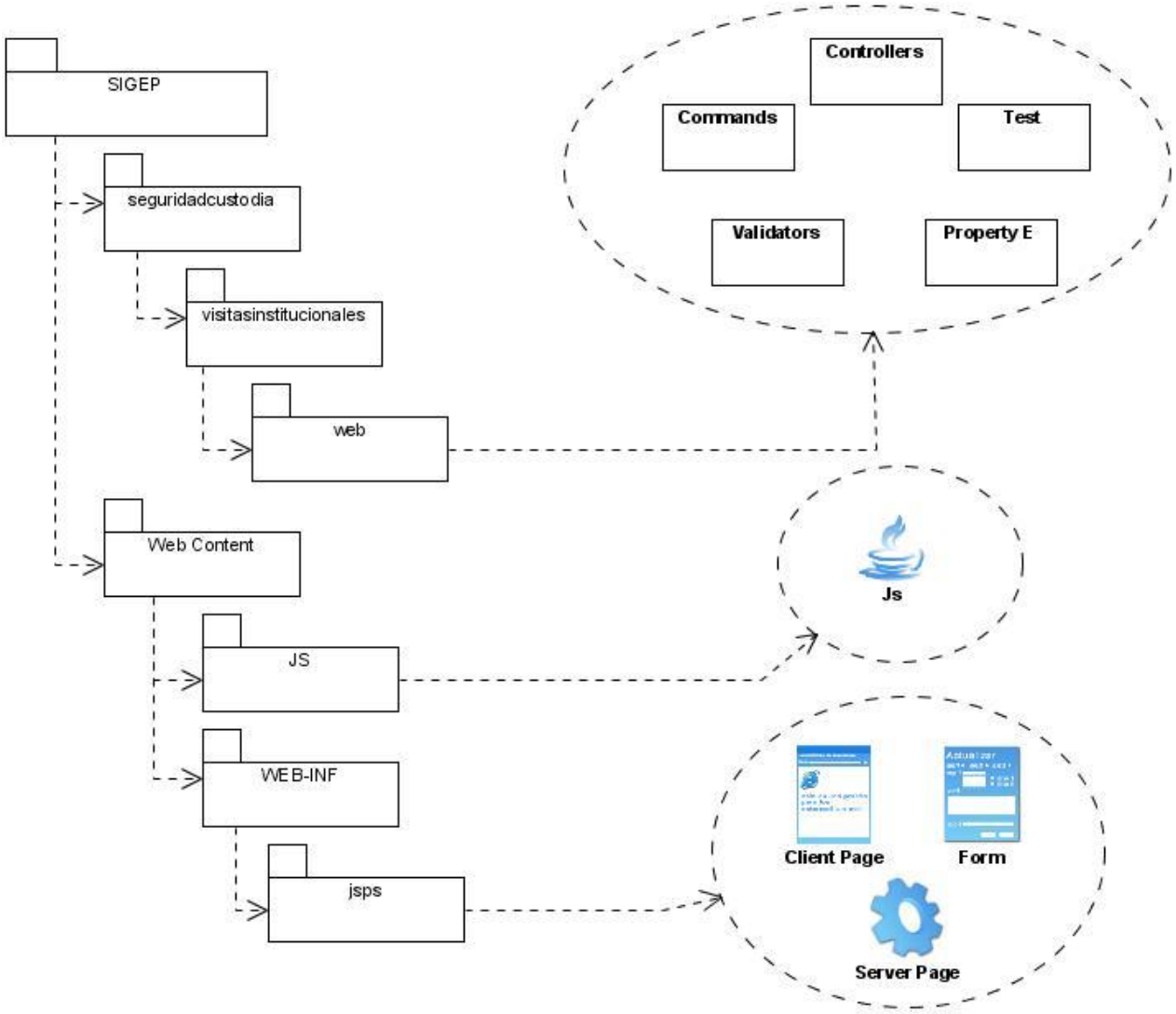

Figura 4 Arquitectura de la capa de presentación

2.5. Patrones

Dado el hecho de la aparición de varios problemas que se presentan de forma recurrente durante el proceso de desarrollo de software, se ha decidido hacer uso de los patrones de diseño de software con la finalidad de explotar las soluciones que los mismos nos. Estos patrones ofrecen soluciones que ya han sido probadas en numerosas ocasiones y logran darle solución al problema planteado de una manera profesional, posibilitando de esta forma un código claro y robusto.

Algunas definiciones de patrones según otros autores:

"Un patrón describe un problema de diseño recurrente, que surge en contextos específicos de diseño, y presenta un esquema genérico probado para la solución de este. El esquema de la solución describe un conjunto de componentes, responsabilidades y relaciones entre éstos, y formas en que dichos componentes colaboran entre sí."Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal. [BU96]

Los patrones de diseño son la representación escrita de las técnicas que proporcionan la visión de alto nivel de un sistema de software. Un patrón es un conjunto de información que aporta la solución a un problema que se presenta en un contexto determinado. Para elaborarlo se aíslan sus aspectos esenciales y se añaden cuantos comentarios y ejemplos sean necesarios. Alejandro Ramirez. [RA04].

A continuación se hará mención a algunos patrones que se usan en el desarrollo de este trabajo:

Modelo Vista Controlador (MVC)

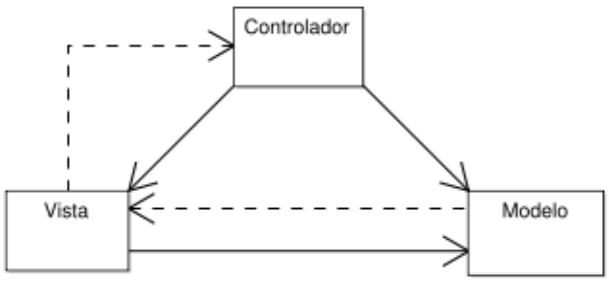

Figura 5 Esquema del Modelo Vista Controlador

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la Lógica de negocio.

Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.

Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos. En MVC corresponde al modelo.

Facade

El patrón Fachada sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas. Entre sus prestaciones este patrón tiene:

- Simplifica el uso y comprensión de una librería software.
- Encapsula una interfaz de librería poco amigable en un interfaz más coherente o mejor estructurado.
- Centraliza las dependencias externas hacia la librería en uno o pocos puntos de entrada.
- Reduce la dependencia de código externo en los trabajos internos de una librería, ya que la mayoría del código lo usa Facade, permitiendo así más flexibilidad en el desarrollo de sistemas.
- Los clientes no necesitan conocer las clases que hay tras la clase FACADE.
- Se pueden cambiar las clases “ocultadas” sin necesidad de cambiar los clientes. Sólo hay que realizar los cambios necesarios en FACADE.

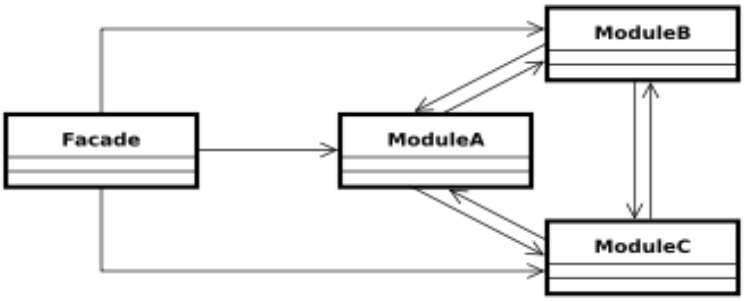

Figura 6 Esquema del Patrón Fachada

2.6. Ambiente de Desarrollo

Como algo inherente al desarrollo de software tenemos al ambiente de desarrollo (Development Enviroment en inglés). Dentro de este se definen todo el conjunto de herramientas y tecnologías que intervienen en un proceso de desarrollo de software.

Se relacionan a continuación todo el conjunto de herramientas usadas en el diseño y la implementación de la capa de presentación en nuestro trabajo, tales como: un ambiente de desarrollo integrado o IDE por sus siglas en inglés (Integrated Development Enviroment) y sus plugins, las herramientas de modelado, los frameworks, así como las tecnologías. Se hará referencia a las características y principales ventajas, para así proporcionar un mejor entendimiento de sus capacidades y comprender su utilidad.

2.6.1. Ambiente de desarrollo integrado (IDE)

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Estos proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, Visual Basic, Object Pascal, etcétera.

Seguidamente se relacionan un conjunto de herramientas que se usaron en el ambiente de desarrollo integrado:

Eclipse

Eclipse es un entorno de desarrollo integrado de código abierto independiente de una plataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados, como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

2.6.1.1. Plugins

Un plugin (plug-in, en inglés enchufar) es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica. Esta aplicación adicional es ejecutada por la aplicación principal. En el caso del Eclipse le dan un conjunto de funcionalidades que lo hacen más extensible.

Web Tool Platform (WTP)

La plataforma de herramientas Web de Eclipse o WTP provee un conjunto de herramientas para desarrollar aplicaciones Web y Java EE. Incluye editores gráficos y código fuente para una gran variedad de lenguajes, con el objetivo de simplificar el desarrollo, así como herramientas y APIs para soporte de despliegue, ejecución, y pruebas de aplicaciones.

Soporta integración con servidores Web dentro de Eclipse como ambiente de ejecución de primera clase para aplicaciones Web. También incluye la configuración de servidores y su asociación con los proyectos Web, permitiendo la depuración sobre el servidor de los recursos y las clases.

Spring IDE

Spring IDE es un plugin que sirve como interfaz de usuario gráfica para la configuración de los archivos usados por Spring Framework. Permite el completamiento de etiquetas, valores de atributos y elementos en estos archivos de configuración.

Analiza gramaticalmente expresiones de punto de cortes de AspectJ (AspectJ Pointcut Expressions) y @AspectJ-style pointcut expressions, mostrando errores si están incorrectas sintácticamente.

Muestra un árbol con todos los proyectos de Spring y sus archivos de configuración. Permite hacer búsquedas de beans en dichos archivos. Muestra gráficamente los beans y sus dependencias.

Presenta un editor gráfico con completamiento y validaciones para los archivos de definiciones del flujo Web usado por el Spring Web Flow.

Subclipse

Subclipse es un plugin para Eclipse que adiciona integración para el control de versiones (Subversion, específicamente), permitiendo operaciones de sincronización, actualización, entre otras. Permite bloqueos a recursos para que otros usuarios no puedan modificarlo. Dispone de una vista de comparación entra el recurso local y remoto en caso que exista conflicto entre la versión del recurso local con el remoto. Muestra una vista del historial de versiones de los recursos con un conjunto de atributos de las acciones realizadas sobre el recurso. Soporta conectarse a varios repositorios de control de versiones al mismo tiempo, permitiendo hacer operaciones sobre el repositorio directamente. Es un plugin muy útil para el desarrollo colaborativo, en el que intervienen un conjunto de desarrolladores trabajando sobre el mismo proyecto, poniendo a disposición del equipo de desarrollo facilidades para el trabajo en equipo.

2.6.2. Herramientas de Modelado

El lenguaje Unificado de modelado (UML) utilizado para visualizar, especificar, construir y documentar un sistema de software por medio de conceptos orientados a objetos, ha constituido la base principal para la construcción de diagramas en las diferentes herramientas de modelados.

Visual Paradigm Suite

Visual Paradigm Suite es un conjunto de herramientas que permiten realizar el modelado dentro del proceso de desarrollo de software.

Dentro de su suite se encuentran:

- Visual Paradigm for UML Enterprise Edition
- Visual Paradigm Smart Development Environment (SDE) Enterprise Edition.

- Visual Paradigm DB Visual Architect Frameworks

ER/Studio

ER/Studio es una herramienta de modelado para diseñar bases de datos Con un soporte de ida y vuelta de bases de datos, los arquitectos tienen el poder para analizar las fuentes de datos existentes así como el diseño e implementación de bases de datos de alta calidad.

2.6.3. Frameworks

Un framework es una estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado. Típicamente, un *framework* puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Spring Framework

Es un marco de trabajo de aplicación de código abierto que ayuda a hacer el desarrollo en JEE mucho más fácil. Ayuda a estructurar aplicaciones completas en una manera consistente y productiva para crear arquitecturas coherentes. (JOHNSON 2005)

Es el más popular y el más ambicioso de todos los framework de peso ligero. Es el único framework que interviene en todas las capas arquitectónicas de una aplicación JEE. Además está diseñado para facilitar una flexibilidad arquitectónica. (JOHNSON 2005)

Dentro de este framework que abarca una gran variedad de módulos, se utilizó específicamente el **Framework Web MVC** del que se dará una breve descripción a continuación:

- **Framework Web MVC**: Su estructura esta implementada sobres clases interface haciéndola sumamente flexible a la vez que define claramente elementos como Controllers, Models y Views(González 2007). De dichos controllers, Spring MVC proporciona a los desarrolladores una serie de implementaciones, las cuales pueden factorizar el comportamiento común en el manejo de múltiples requests y configuran mediante *Inversión de Control* los demás objetos, haciéndolos fáciles de probar e integrables con otros objetos que estén en el contexto de Spring framework. A la hora de mostrar las vistas, Spring MVC facilitó no solo el formato JSP

(Java Server Pages), sino otros formatos como PDF, Excel, XLST o la opción de implementar una vista propia.

Dojo

Framework que contiene APIs y widgets para facilitar el desarrollo de aplicaciones Web que utilicen tecnología AJAX (Bates 2006). Contiene un sistema de empaquetado inteligente, los efectos de interfaz de usuario, drag and drop APIs, widget APIs, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX. Es conocido como "la navaja suiza del ejército de las bibliotecas Javascript". Proporciona una gama más amplia de opciones en una sola biblioteca Javascript y es compatible con navegadores antiguos. Dojo brindó una serie de componentes visuales que se convirtieron prácticamente en la cara de la aplicación Web, maneja peticiones tanto sincronías como asíncronas (muy usado para gestionar información sin la necesidad de refrescar toda la vista), componentes muy útiles que se usaron en la validación del lado del cliente, para listar, filtrar información, calendarios, los cuales hasta hace poco solo eran privilegios de las aplicaciones de ventanas.

2.6.4. Tecnologías Empleadas

Programación Orientada a Objetos (POO)

La tecnología Orientada a Objetos en nuestros días se torna prácticamente indispensable, se aplica tanto en el análisis y diseño como en la implementación y hasta en las bases de datos. (Morero 2000) La POO es la forma más popular de programar actualmente, permite a los programadores escribir instrucciones de forma que esté organizado en la misma manera que el problema que trata de modelar, y casi no se imagina un proyecto de software que no la utilice de alguna manera, pues existe una tendencia al crecimiento del nivel de complejidad de los programas empresariales con complicadas reglas de negocio. Para el caso particular de este proyecto el uso de la POO brindó muchas ventajas:

- Permitió crear un software con un alto grado de complejidad.
- Posibilitó la reutilización y extensión de código.
- Agilizó enormemente todo el proceso de desarrollo de software.
- Facilitó en gran medida el trabajo en equipo.

- Facilitó e todo momento el mantenimiento del software implementado.

Javascript:

Es un lenguaje de programación interpretado utilizado principalmente en páginas Web, con una sintaxis semejante a la del lenguaje Java. Se hizo gran uso de este lenguaje debido a que las vistas requieren en buena medida de ficheros Javascript para su correcto funcionamiento. Proporcionó rapidez, solvencia y ligereza a la hora de validar datos o realizar operaciones del lado del cliente así como para modificar los elementos de la Web pertenecientes al DOM (Document Object Model por sus siglas en inglés).

Ajax:

Asynchronous Javascript And XML (Javascript asíncrono y XML), es una técnica de desarrollo Web para crear aplicaciones interactivas o RIA (Rich Internet Applications).(Pérez 2008) Estas se ejecutan en el cliente, es decir, en el navegador de los usuarios y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma. AJAX es una combinación de cuatro tecnologías ya existentes:

- XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño.
- Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como Javascript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto XMLHttpRequest para intercambiar datos asincrónicamente con el servidor Web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.
- XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

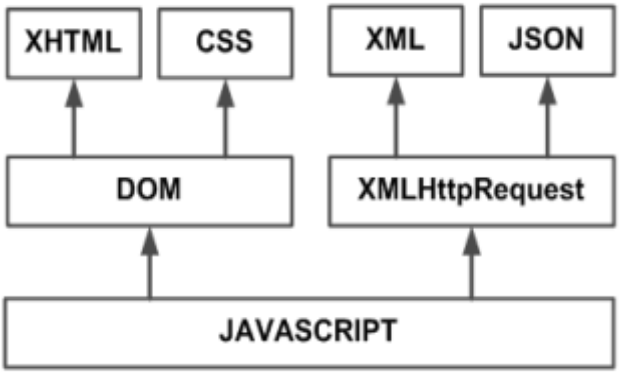

Figura 7 Tecnologías que conforman AJAX

La difusión de AJAX en los líderes de la industria de internet prueba que el mercado acepta y valida el uso de esta tecnología. Todo el mundo está migrando hacia AJAX incluyendo Google, Yahoo, Amazon, Microsoft (por nombrar unos pocos).

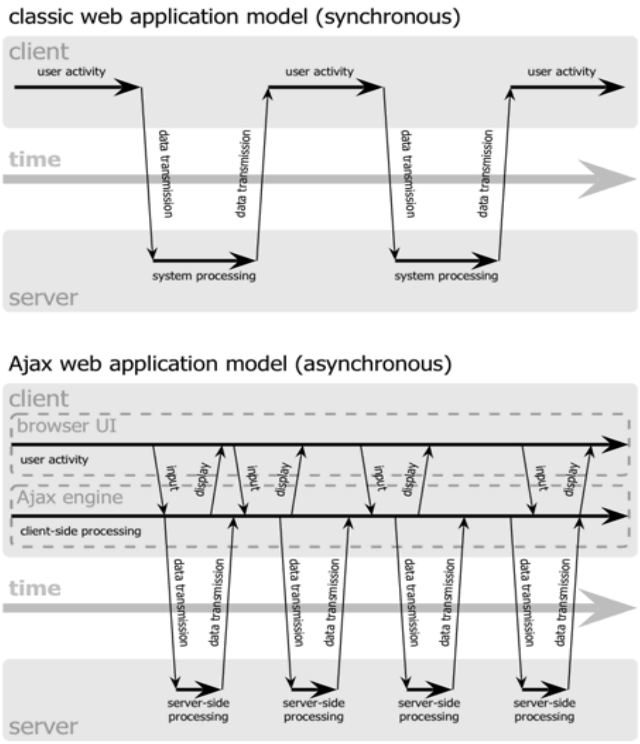

Figura 8 Web app clásica/Web app con AJAX

2.7. Flujo de Trabajo para Programador de Interfaz de Usuario

Para el desarrolllo de la solución informática nos basamos en el flujo de trabajo definido en el proyecto SIGEP para el rol de Programador de Interfaz de Usuario(PIU), lo que fue de suma importancia pues organizó completamente el desarrollo al tener plasmado cada paso necesario en un orden lógico y conciso.

- Definir con el diseñador gráfico las_jsps que se van obtener, los campos que tendrán, etc., y/o los modelos de cómo deben quedar los reportes en caso de que los haya, que se van a obtener.
- Realizar diagrama de navegación (por páginas, vistas e interacciones cliente servidor).
- Dado el diagrama anterior, definir tipos de controladores que se van a usar utilizando un mapa de controladores predefinidos.
- Elaborar tabla (controlador-petición-vista). Se definen los nombres de las peticiones y las respuestas por controlador.
- Declarar e Implementar controladores sin lógica.
- Declarar los controladores en los ficheros xml de configuración de Spring.
- Crear Commands, Validators y PropetyEditors (posible interacción con el diseñador, necesitan las entidades para utilizarlas como Command o PropertyEditor) y se machean los atributos del jsp con los del command en caso de hacer falta.
- Implementar lógica de los controladores. (posible interacción: debe estar definida las Fachadas de Negocio y las entidades, y se piden métodos necesarios en caso de que no existan).
- Machear vista con modelo y programar lógica en el cliente
- Realizar pruebas a la interfaz para verificar el correcto funcionamiento e interacción de esta con el supuesto usuario final.
- Implementar validaciones en el cliente. (Gómez 2006)

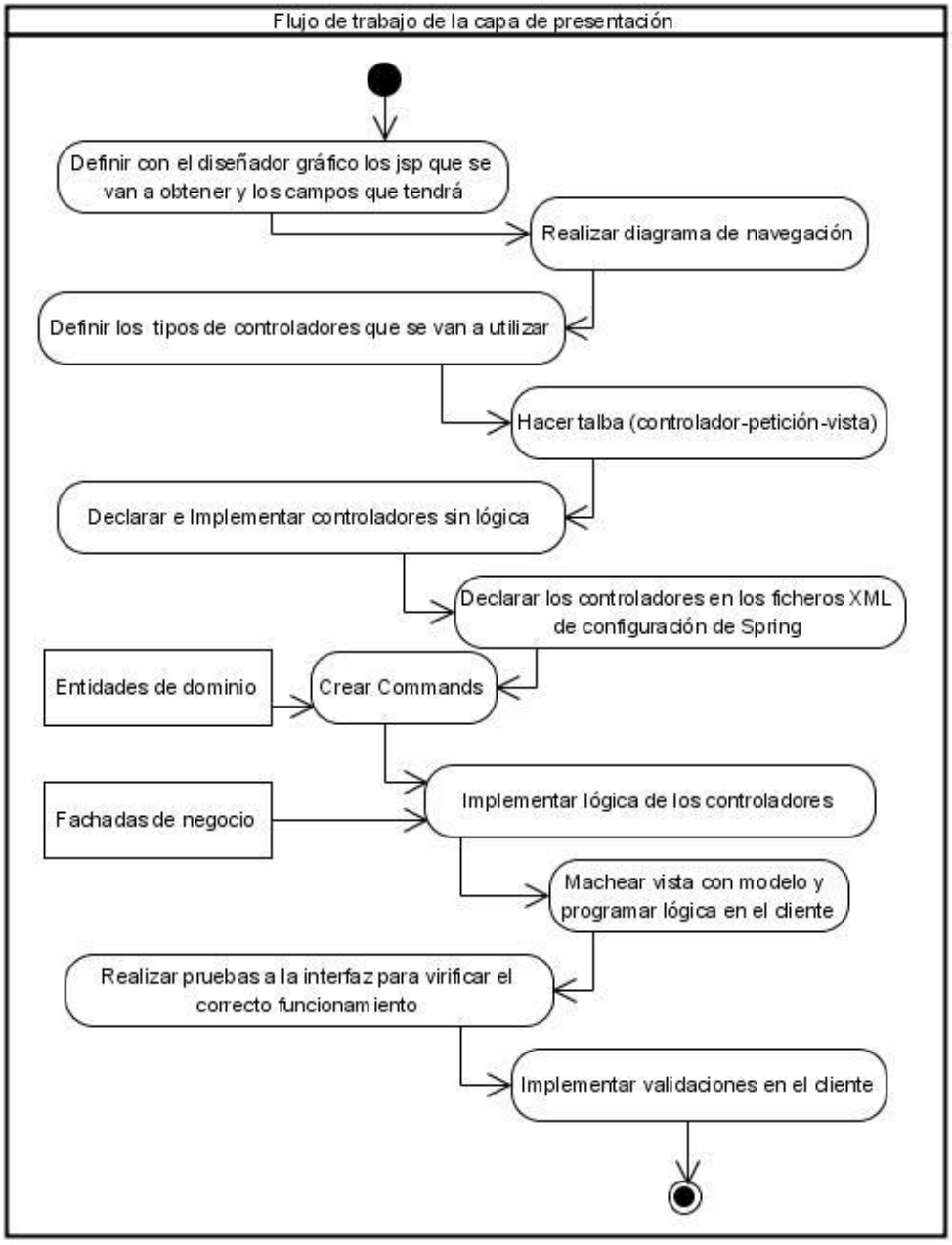

Figura 9 Diagrama del Flujo de Trabajo del PIU

2.8. Diseño de la solución

El módulo de visitas institucionales tiene dos procesos fundamentales, Planificación de Visitas Institucionales y Ejecución de Visitas Institucionales. Dentro del proceso de Planificación se gestiona todo lo referente al registro de una nueva visita, las que se dividen en dos grupos: las permanentes y

las no permanentes. En el caso del proceso de Ejecución enmarca lo relacionado con el registro de la entrada y la salida de las visitas previamente planificadas.

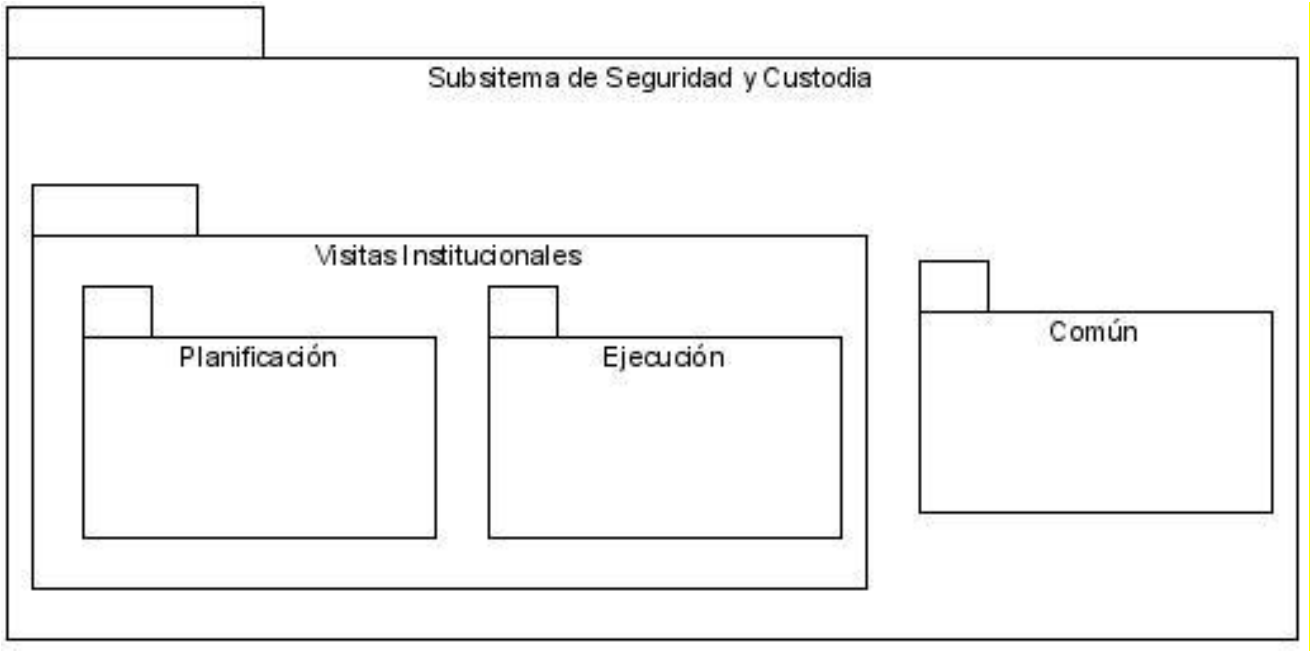

Figura 10 Diseño de Paquetes

2.8.1. Componentes utilizados

Según las necesidades de las pantallas, del prototipo a desarrollar, se utilizaron varios componentes visuales de la librería de Dojo Toolkit, los llamados widgets de Dojo. A pesar de que solo se usaron algunos de ellos, estos proporcionaron a las vistas mucha estética y agilidad en el trabajo con la información. Los componentes utilizados fueron:

Navegable Table: Tabla que pide sus datos al servidor mediante una propiedad llamada dataUrl y a la que en el desarrollo se le implementó un Controlador especial (AbstractLoadTableController) que la provee de datos en cada una de sus peticiones .Permite paginado y filtrado. Puede mostrar varios tipos de datos como String, Integer, Boolean, Float, así como código HTML.

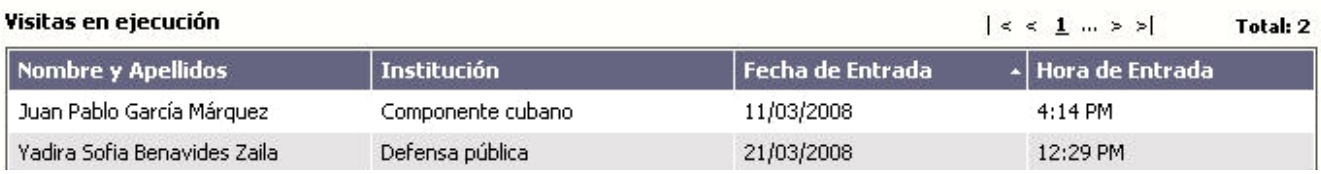

Figura 11 Navegable Table

DatePicker: Calendario que puede mostrar y salvar la fecha en diferentes formatos. Entre sus propiedades están las acotaciones de fecha inicial y final.

Figura 12 Date Picker

SpinnerTime: Componente que permite el cambio de hora y que valida su formato de entrada.

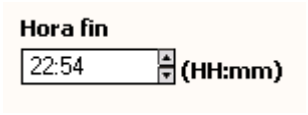

Figura 13 Spinner Time

RegexpTextbox: Valida la entrada de texto ante una expresión regular, muestra mensajes si el texto es requerido o está incorrecto.

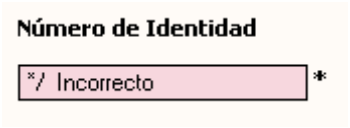

Figura 14 Regexp Textbox

2.8.2. Funcionalidades más complejas. Solución

Para el desarrollo del módulo de Visitas Institucionales se desarrollaron varias funcionalidades con determinado peso dentro de la capa de presentación. Para la planificación, se implementaron: Gestionar Visitas Institucionales Permanentes Planificadas que permite al usuario ver un listado de todas las visitas permanentes planificadas hasta el momento, así como planificar una nueva visita, modificarla y eliminarla. Con las mismas características del anterior se desarrolló Gestionar Visitas Institucionales No Permanentes Planificadas.

Gracias a un flujo de trabajo bien definido para el rol de programador de interfaz de usuario, con especificaciones de que hacer en cada momento y a la existencia de un equipo de trabajo con buenas prácticas de desarrollo, las labores de implementación transcurrieron sin grandes percances aunque se presentaron algunos casos de funcionalidades que necesitaban para su implementación elementos interesantes. Para su solución se realizaron estudios de casos similares y pruebas que dieran seguridad sobre su funcionamiento. Como ejemplo está la funcionalidad de Registrar Entrada de Visitas Planificadas perteneciente a la Ejecución de Visitas Institucionales , la que incluye toma de datos al Visitante, tanto datos escritos como imagen de dicho visitante. Esto sugirió el trabajo con dispositivos externos, específicamente una WebCam. Para ello fue necesario diseñar y desarrollar un applet (WebCam-Applet) que implementa al API JTwain. JTwain permite a los desarrolladores capturar imágenes de un escáner y/o cámara digital mediante la utilización de código Java.

JTwain es una suite desarrollada por LAB Asprise. JTwain comprende 2 componentes esenciales:

- Una librería nativa (AspriseJTwain.dll) la cual debe estar presente físicamente en la máquina que va a realizar el proceso de escaneo donde está conectado el dispositivo de escaneo.
- Varios paquetes de Java:* com.asprise.util.jtwain. E paquete principal; contiene las clases esenciales para la captura de imágenes.

Esta pantalla permite dar entrada a un visitante institucional después de haber planificado la visita, entrando sus datos por escrito y tomando su foto mediante el uso del dispositivo externo. Además permite acceder a un buscador con los visitantes registrados y seleccionarlo, acto seguido aparecerán en el formulario sus datos y la imagen registrados anteriormente.

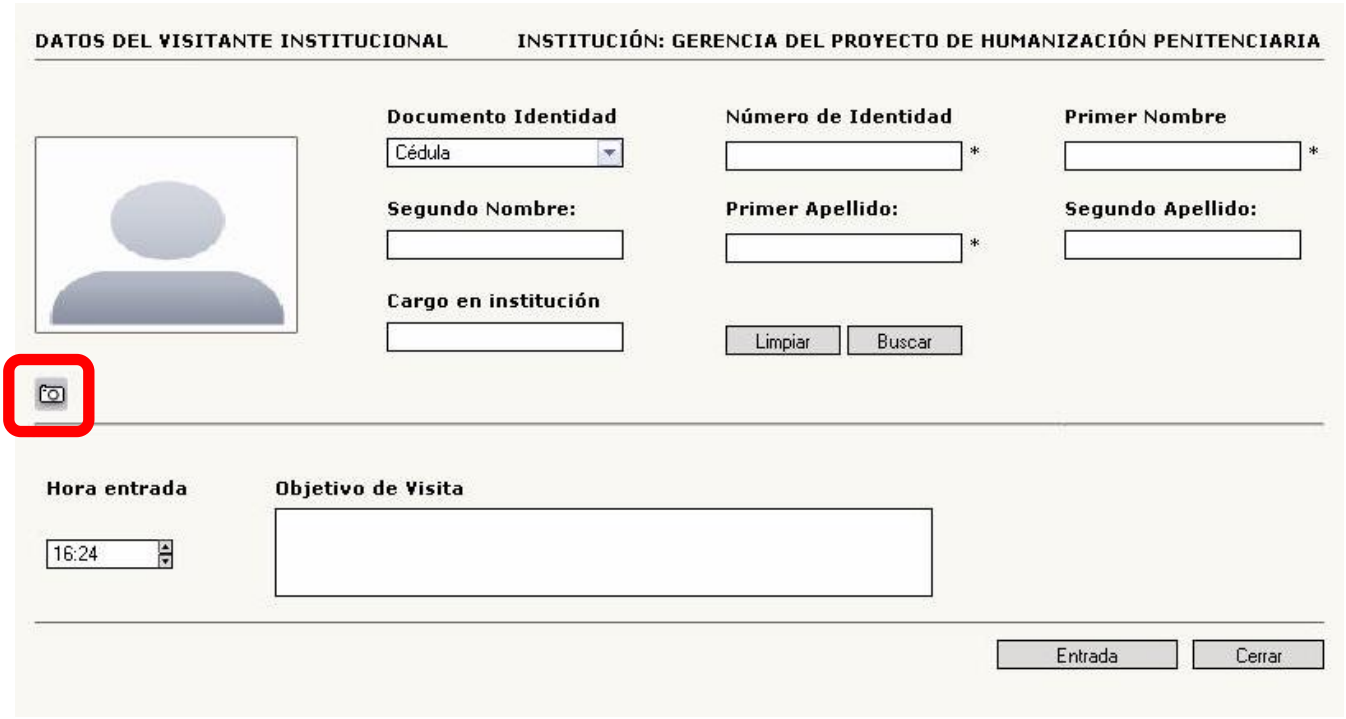

Figura 15 Entrada a visitante institucional

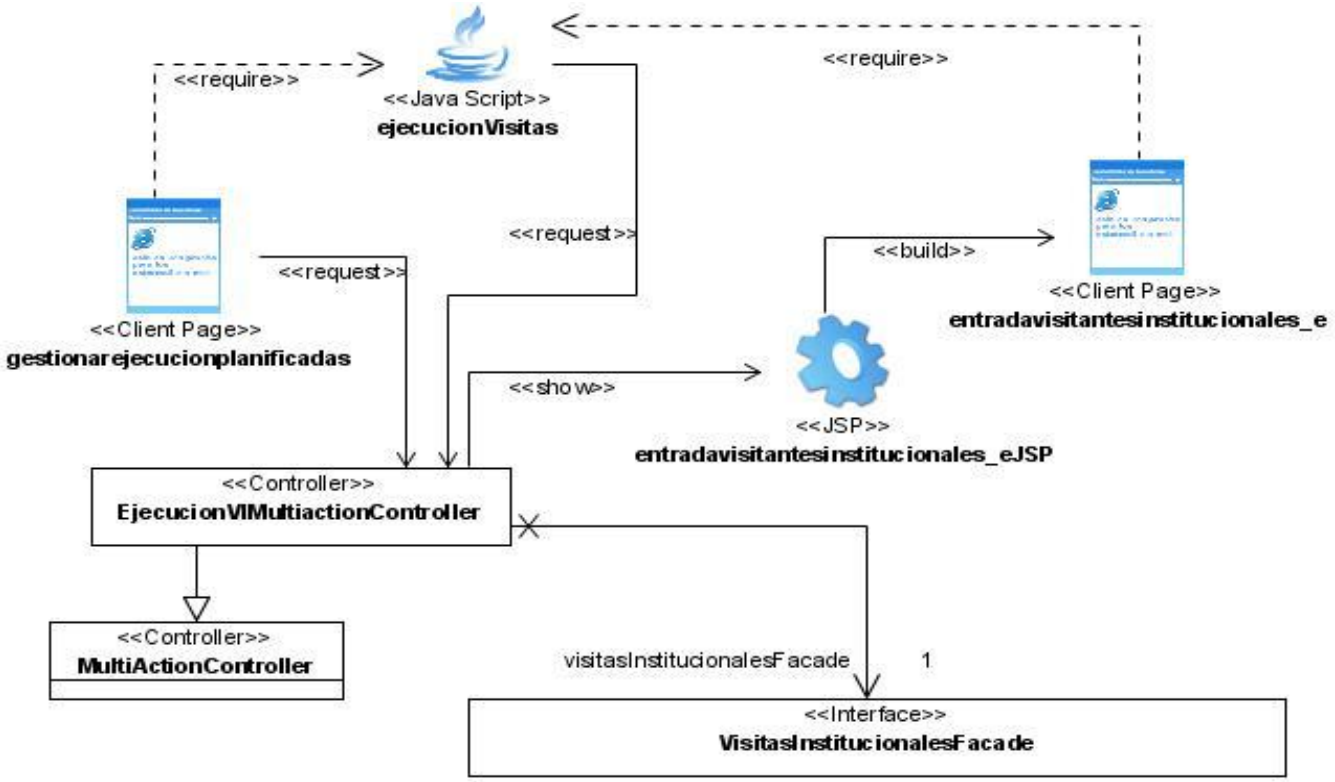

Figura 16 Diagrama de clases Web: Entrada de Visita Institucional Planificada

Otra de las complejidades que se presentó fue la del filtrado de datos para la tabla del Histórico de las Visitas Ejecutadas. Para este caso se definieron filtros en el fichero XML de configuración y se colocó un widget de la librería de Dojo para soportar los criterios para el filtrado.

En esta pantalla al seleccionar el vínculo Filtrar aparece una sección desplegable que puede mostrar tantos criterios de filtrado como datos tenga la tabla. Automáticamente después de seleccionar los criterios se actualiza la tabla con los datos que desea el usuario que sean mostrados.

Visitas que han sido ejecutadas

Filtrar | < < 1 2 3 ... > >| Total: 247

Nombre y Apellidos	Institución	Fecha de Entrada	Hora de Entrada	Fecha de Salida	Hora de Salida
Heysell Jaime Piñera	**Componente Cubano. Humanización Penitenciaria**	**14/12/2007**	**8:10 AM**	**14/12/2007**	**11:00 PM**
Javier López Del Castillo Caymares	Componente Cubano. Humanización Penitenciaria	14/12/2007	8:10 AM	14/12/2007	11:00 PM
Luis Alberto Pimentel González	Componente Cubano. Humanización Penitenciaria	14/12/2007	8:09 AM	14/12/2007	11:00 PM
Arturo César Arias Orizondo	Componente Cubano. Humanización Penitenciaria	14/12/2007	8:09 AM	14/12/2007	11:00 PM
Esteban Reynol Cuba Tápanes	Componente Cubano. Humanización Penitenciaria	14/12/2007	8:35 AM	14/12/2007	11:00 PM
Deny Lester Collado Coello	Componente Cubano. Humanización Penitenciaria	14/12/2007	8:14 AM	14/12/2007	11:00 PM
Yandry Alberto Terry	Componente Cubano. Humanización Penitenciaria	14/12/2007	8:11 AM	14/12/2007	11:00 PM
Jorge Ernesto Martínez Cabrera	Componente Cubano. Humanización Penitenciaria	14/12/2007	8:11 AM	14/12/2007	11:00 PM
Aura Cárdenas Morales	MIJ	14/12/2007	11:10 AM	14/12/2007	11:00 PM
Fabricio Perez	Visita gubernamental	14/12/2007	9:33 AM	14/12/2007	11:00 PM
Yanet Vega Miniet	Componente Cubano. Humanización Penitenciaria	14/12/2007	8:12 AM	14/12/2007	11:00 PM
Julio Serra Santiesteban	Componente Cubano. Humanización Penitenciaria	14/12/2007	10:22 AM	14/12/2007	11:00 PM
Esteban Reynol Cuba Tápanes	Componente Cubano. Humanización Penitenciaria	17/12/2007	2:05 PM	17/12/2007	4:01 PM
Martín Antonio Soto Reyes	Defensa pública	17/12/2007	4:51 PM	17/12/2007	5:20 PM
Luis Alberto Pimentel González	Componente Cubano. Humanización Penitenciaria	17/12/2007	2:42 PM	17/12/2007	5:00 PM

Detalles

Figura 17 Histórico de Visitas Ejecutadas

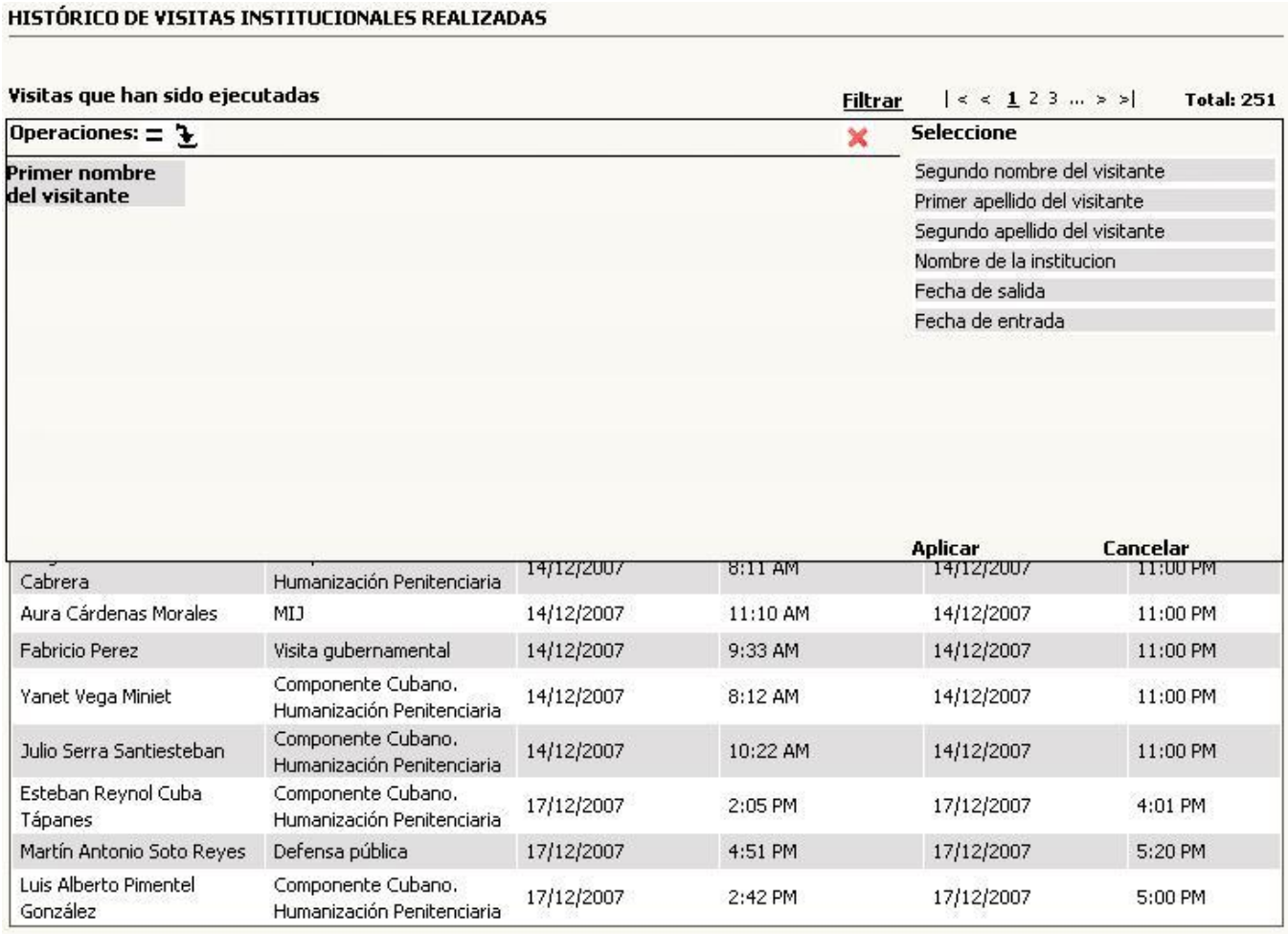

Figura 18 Filtro de la tabla de Histórico de Visitas Ejecutadas

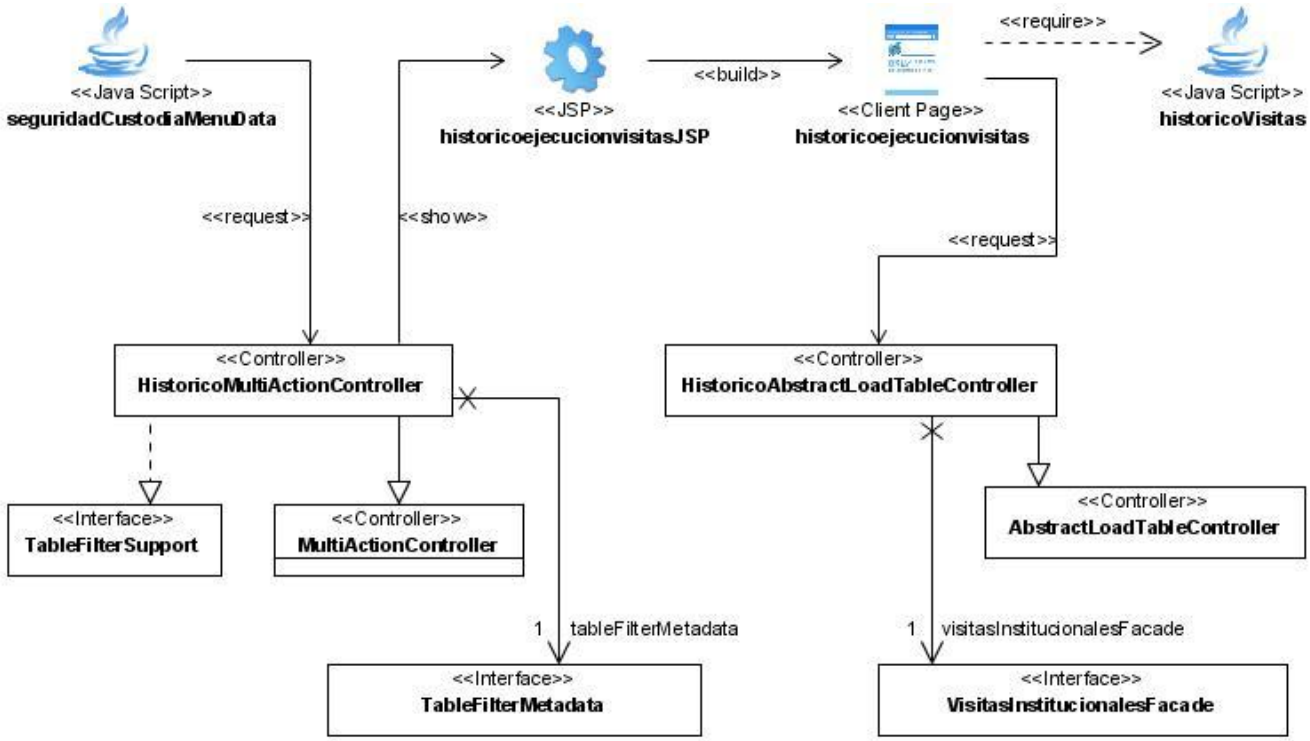

Figura 19 Diagrama de la funcionalidad: Histórico de visitas institucionales ejecutadas

Como parte de la implementación de la solución se tuvo también la necesidad de desarrollar un algoritmo para la generación de turnos en las visitas institucionales. Fue necesario desarrollar una clase (GeneracionTurnoVIUtil) que se encargara de manejar todo el volumen de información desde el servidor de aplicaciones por cuestiones de rendimiento en el cliente. Una vez procesados estos datos un controlador (TurnosPlanificadosVNPTableController) se encargaría de enviarlos a la vista para posteriormente mostrarlos en la página cliente.

A modo general se diseñaron 21 controladores, 22 páginas jsp, 10 ficheros Javascript, 3 commands y 2 Property Editors. En los Anexos aparecen gráficas que presentan los diseños con estereotipos y las pantallas del módulo implementado.

CONCLUSIONES

Las principales conclusiones a las que se pudo arribar en este trabajo son las siguientes:

- Se diseñó, a partir de los requisitos de software y el Modelo de Negocio, la capa de presentación de la Planificación y Ejecución de Visitas Institucionales cumpliendo con las necesidades de los clientes.
- Se implementó el diseño realizado de la capa de presentación relacionado con el proceso de Planificación y Ejecución de Visitas Institucionales, lo que facilitó el tratamiento de la información, su procesamiento y almacenamiento.
- Se desarrolló un producto informático que se ejecuta satisfactoriamente y que da apoyo al control del sistema penitenciario venezolano el que constituye una gran ayuda para la toma de decisiones de los directivos.
- Se realizaron las pruebas unitarias y de integración las que arrojaron resultados satisfactorios.

RECOMENDACIONES

Se recomienda analizar el flujo de trabajo desarrollado, y valorar su reutilización durante el diseño y la implementación de la capa de presentación de sistemas desarrollados bajo la plataforma JEE.

BIBLIOGRAFÍA

- PIMENTEL, L.A. and I. P RIVERO. *ArBaWeb: Arquitectura base sobre la Web*, 2007.
- PEREZ, J E. *Introducción a AJAX*. 2008.
- BATES, S*. The Dojo Book*, 0.9, 2006.
- CHARLAND*, A. Top Ten Reasons AJAX is Here to Stay*, 2002. Disponible en: http://www.developer.com/design/article.php/3567706
- GÓMEZ, J. C. *Flujo de trabajo para programador de UI*, 2006.
- GONZÁLEZ, R. V. *Spring MVC*, 2007.
- MARAÑÓN, G. Á*. Características del lenguaje Java*, 1999.
- MORERO, F. *Introducción a la OOP*., 2000.
- ROD JOHNSON, J. H., ALEF ARENDSEN, COLIN SAMPALEANU, ROB HARROP, THOMAS RISBERG, DARREN DAVISON, DMITRIY KOPYLENKO, MARK POLLACK, THIERRY TEMPLIER, ERWIN VERVAET, PORTIA TUNG, BEN HALE, ADRIAN COLYER, JOHN LEWIS, COSTIN LEAU, RICK EVANS. The Spring Framework - Reference Documentation. Disponible en: http://www.springframework.org/docs/reference/new-in-2.html

ANEXOS

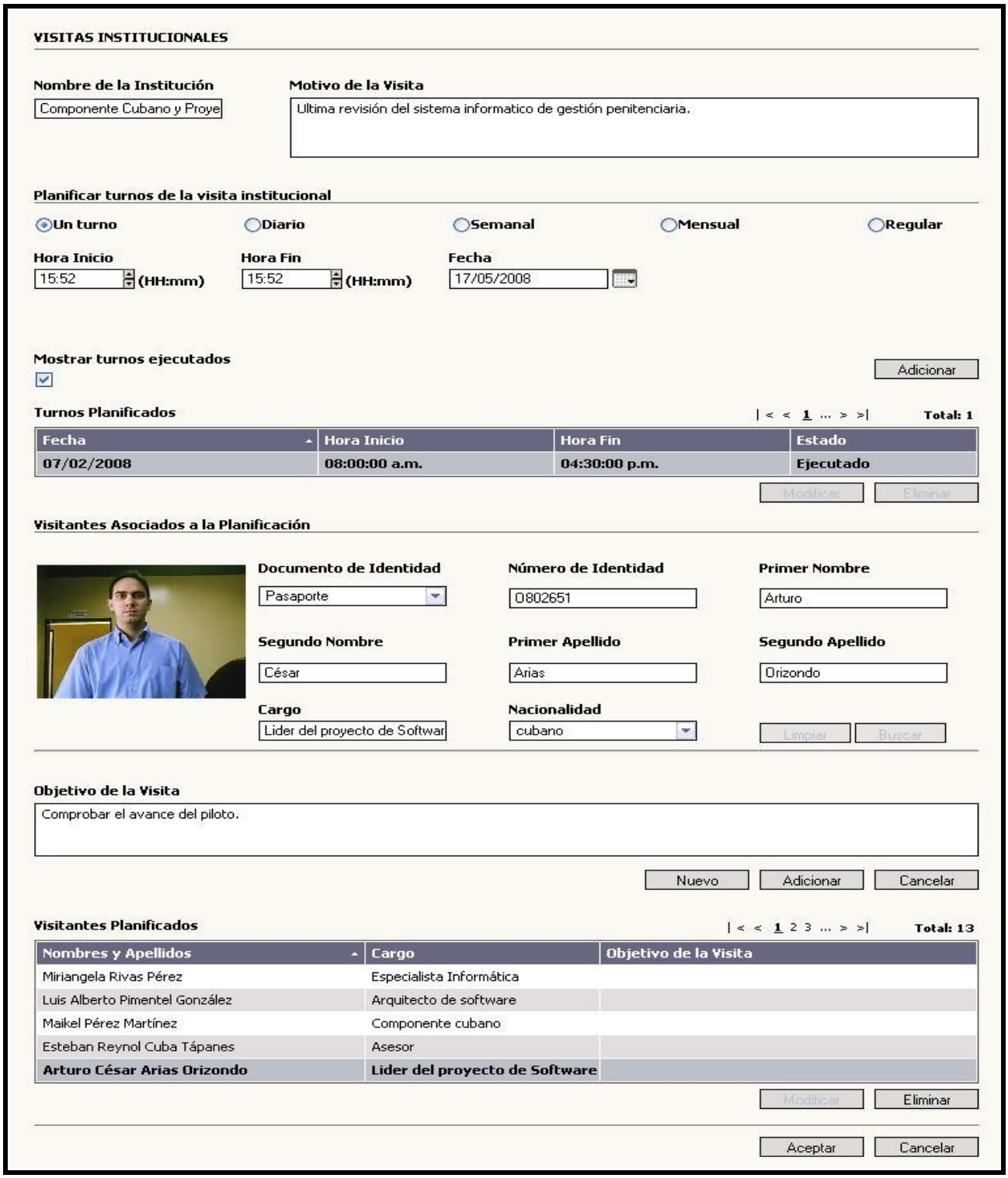

Figura 20 Adicionar/Modificar Visitas Planificadas No Permanentes

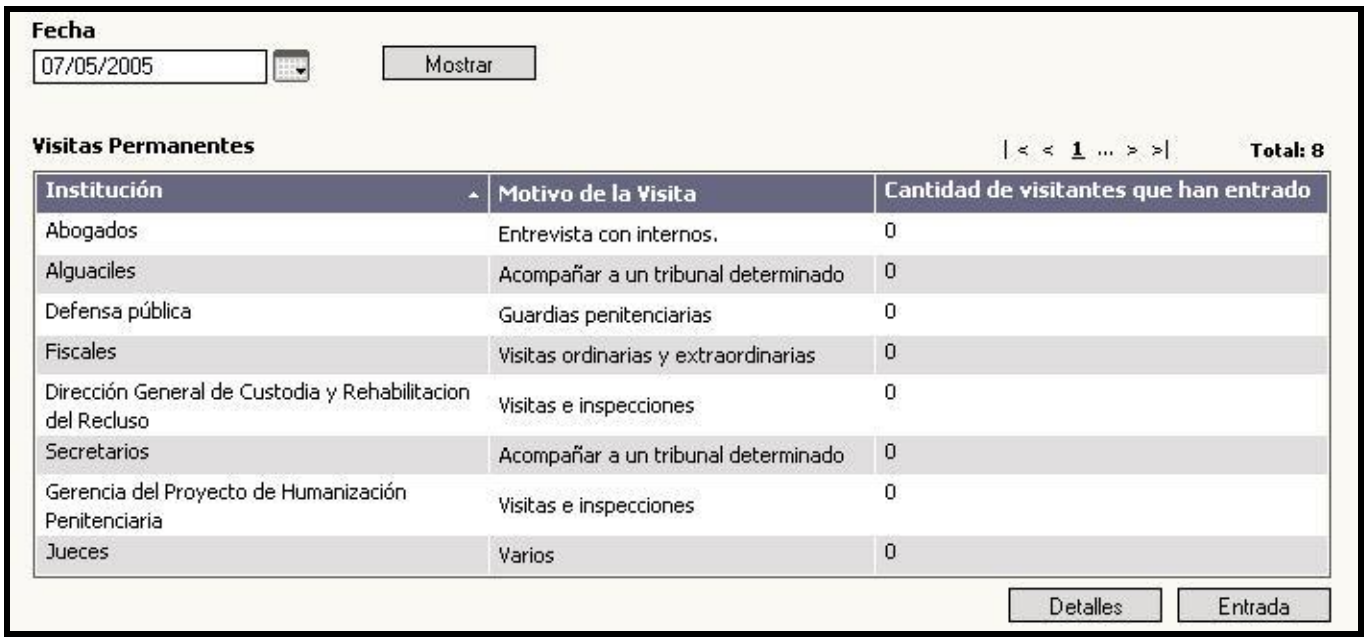

Figura 21 Mostrar Visitas por fecha

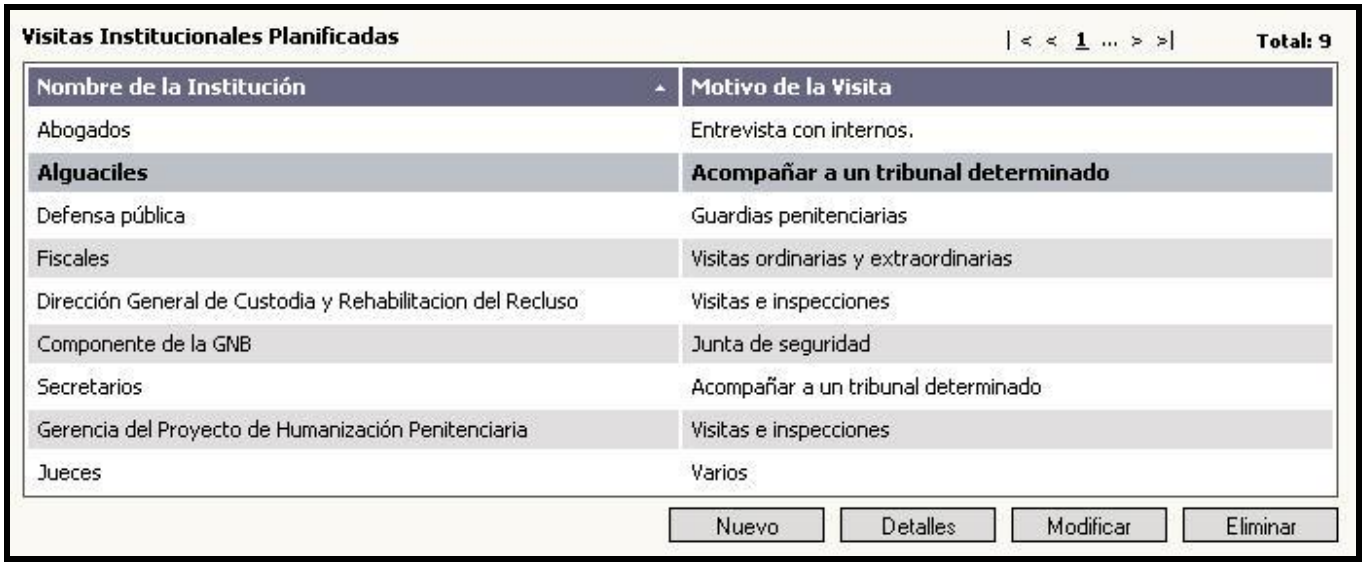

Figura 22 Mostrar Visitas Planificadas Permanentes

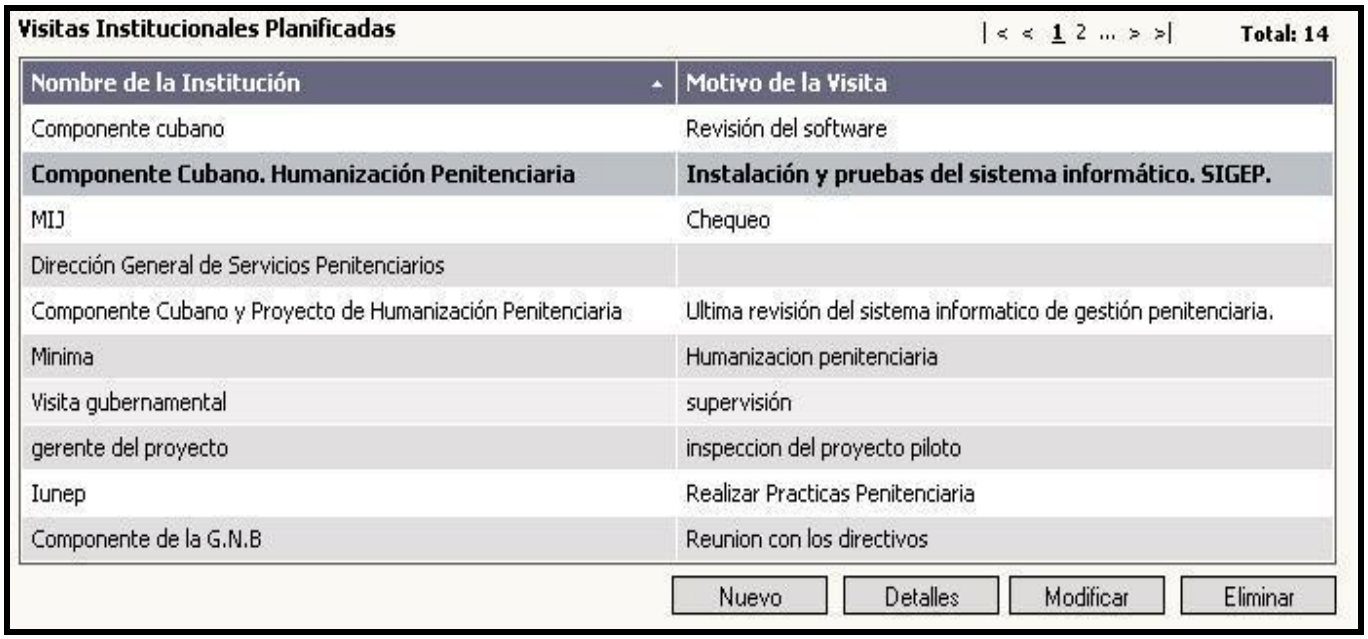

Figura 23 Mostrar Visitas Planificadas No Permanentes

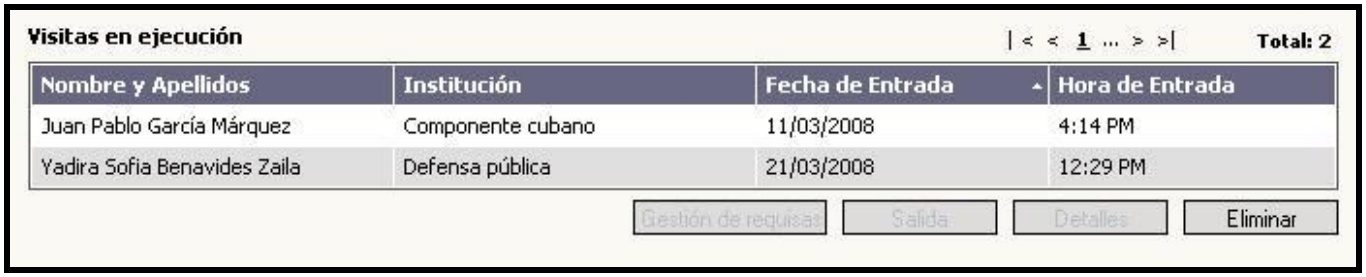

Figura 24 Mostrar Visitas en Ejecución

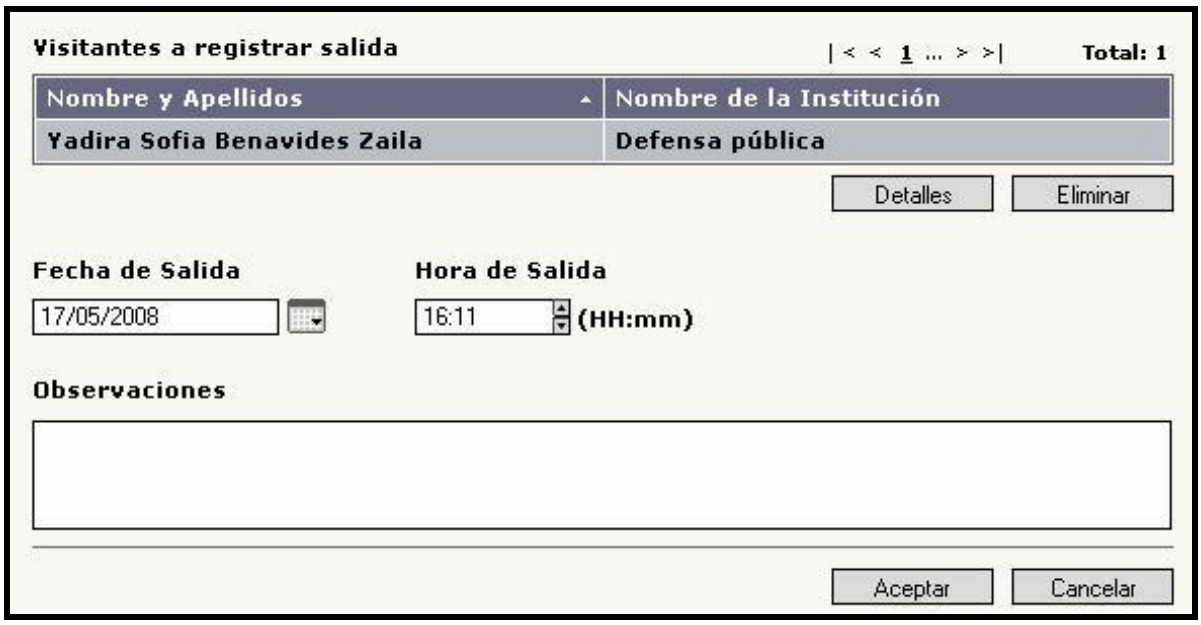

Figura 25 Registrar Salida a Visita en Ejecución

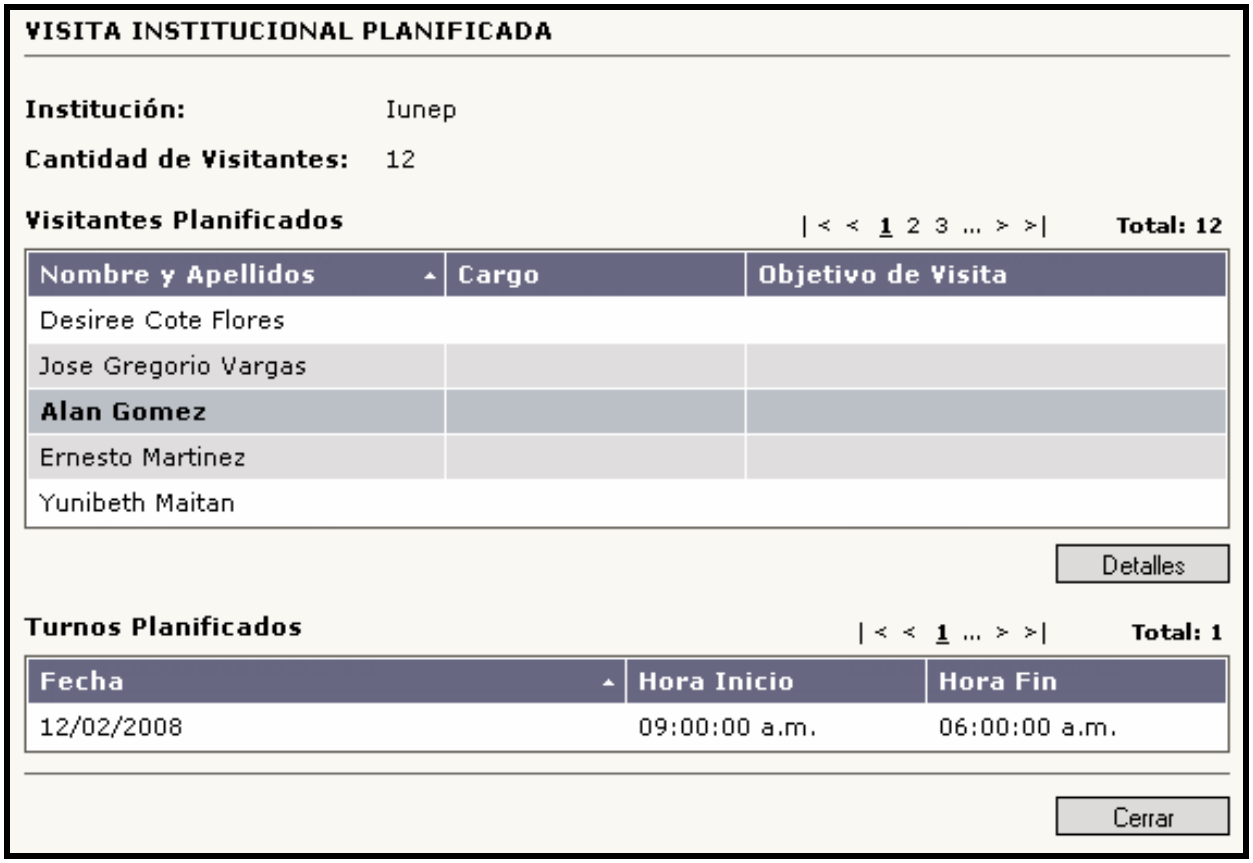

Figura 26 Detalles de Visita Planificada No Permanente

Figura 27 Detalles de Visita en Ejecución

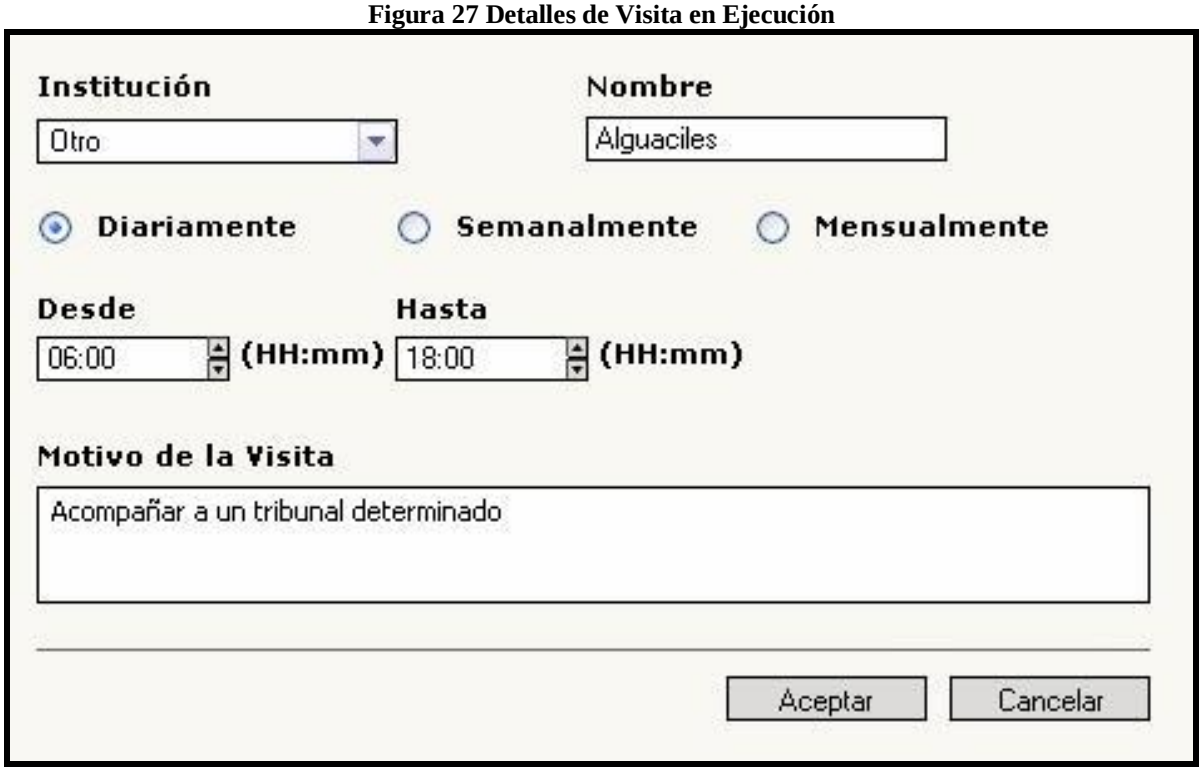

Figura 28 Adicionar/Modificar Visitas Planificadas Permanentes

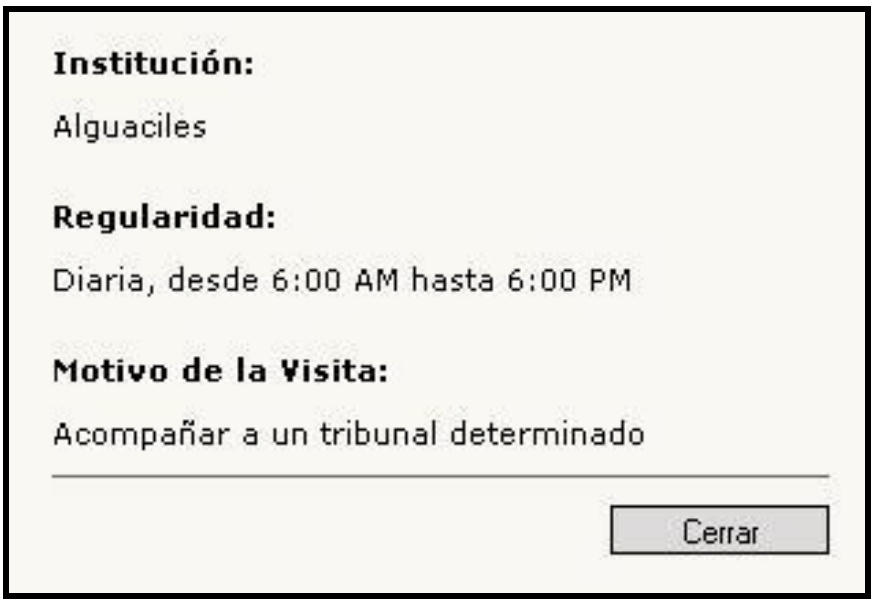

Figura 29Detalles de Visitas Planificadas Permanentes

DIAGRAMAS DE CLASES CON ESTEREOTIPOS WEB

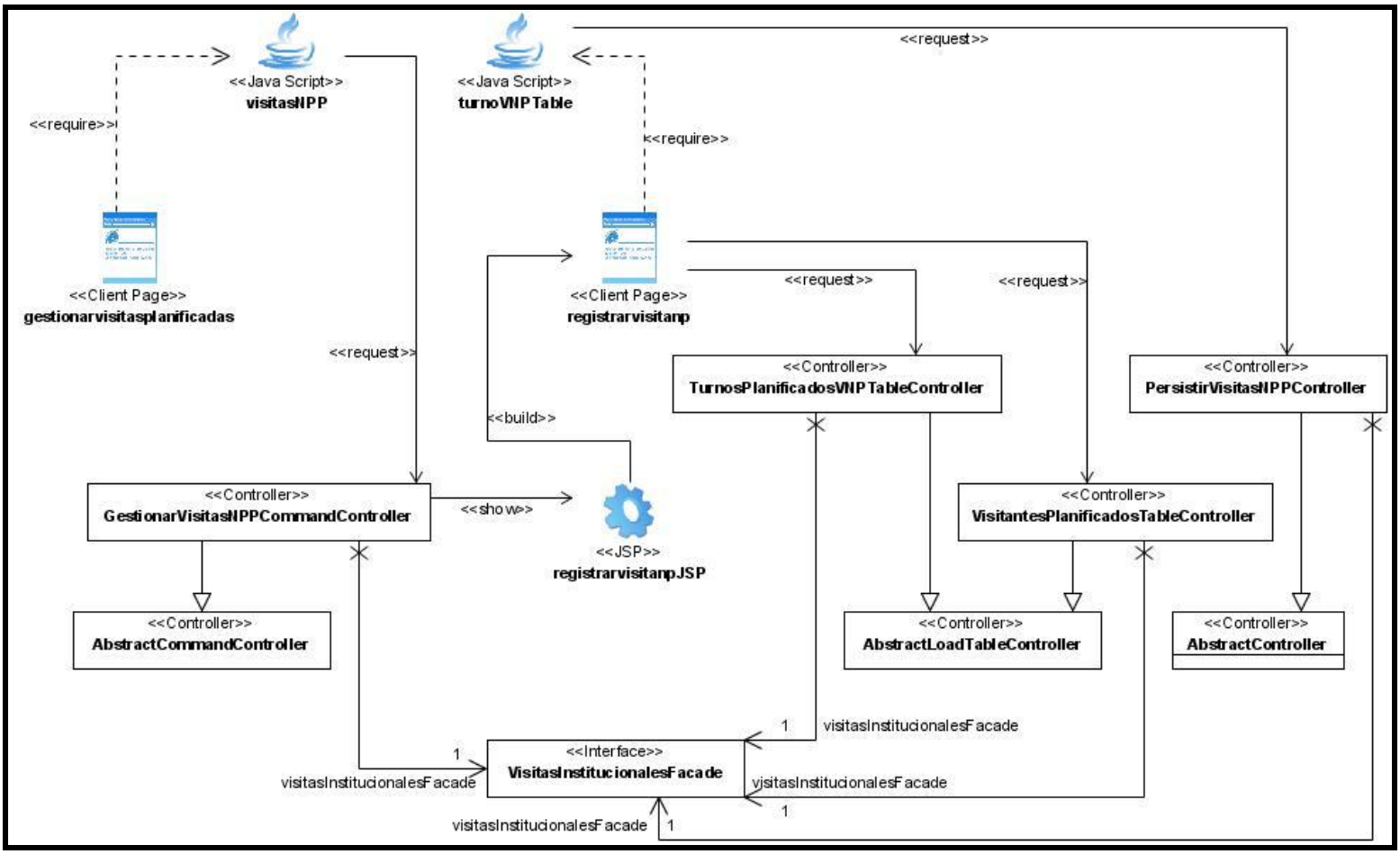

Figura 30 Diagrama de la Funcionalidad: Adicionar/Modificar Visitas Planificadas No Permanentes

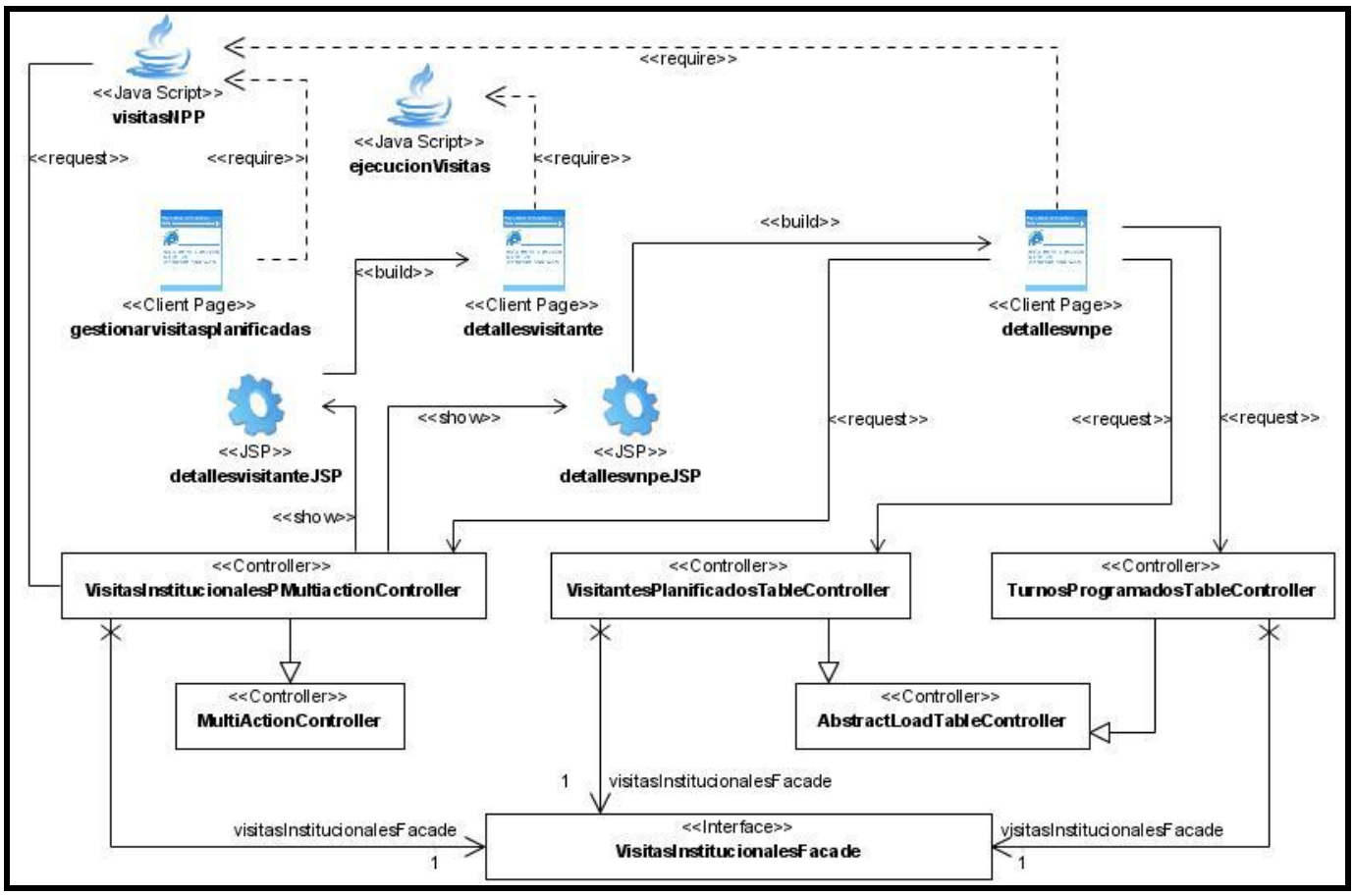

Figura 31 Diagrama de la Funcionalidad: Detalles de Visita Planificada No Permanente

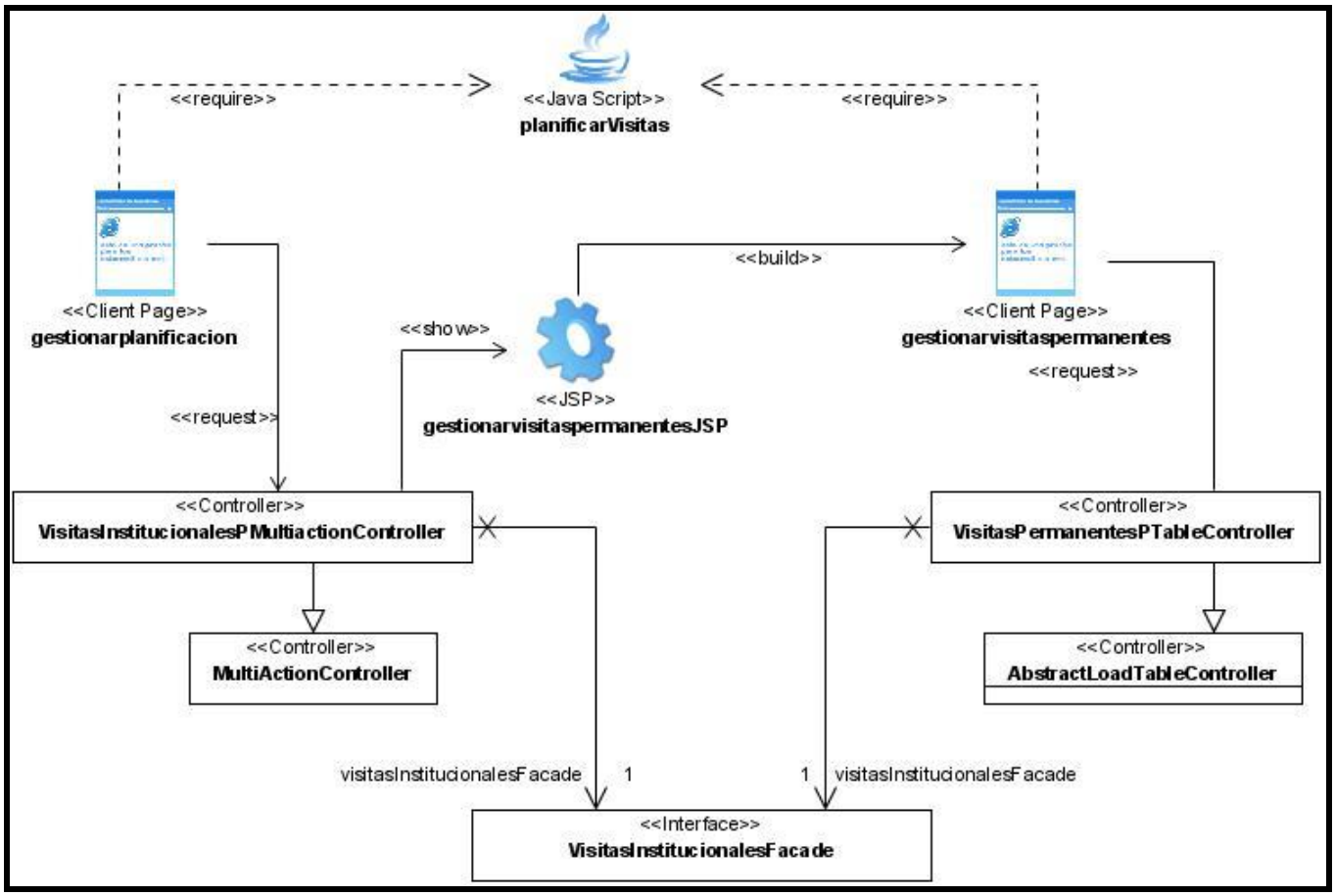

Figura 32 Diagrama de la funcionalidad: Mostrar Visitas Planificadas Permanentes

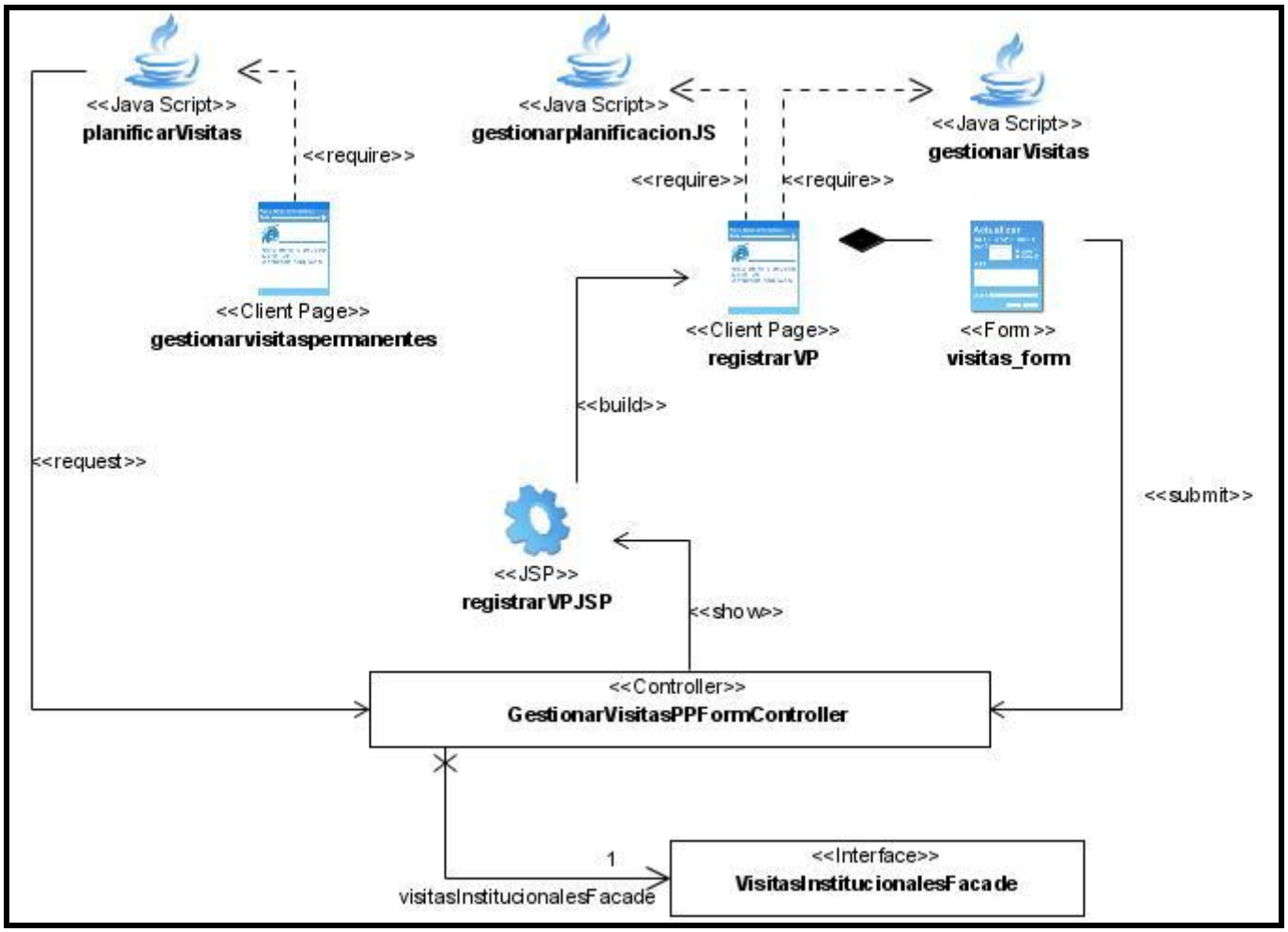

Figura 33 Diagrama de la funcionalidad: Adicionar/Modificar visitas Planificadas Permanentes

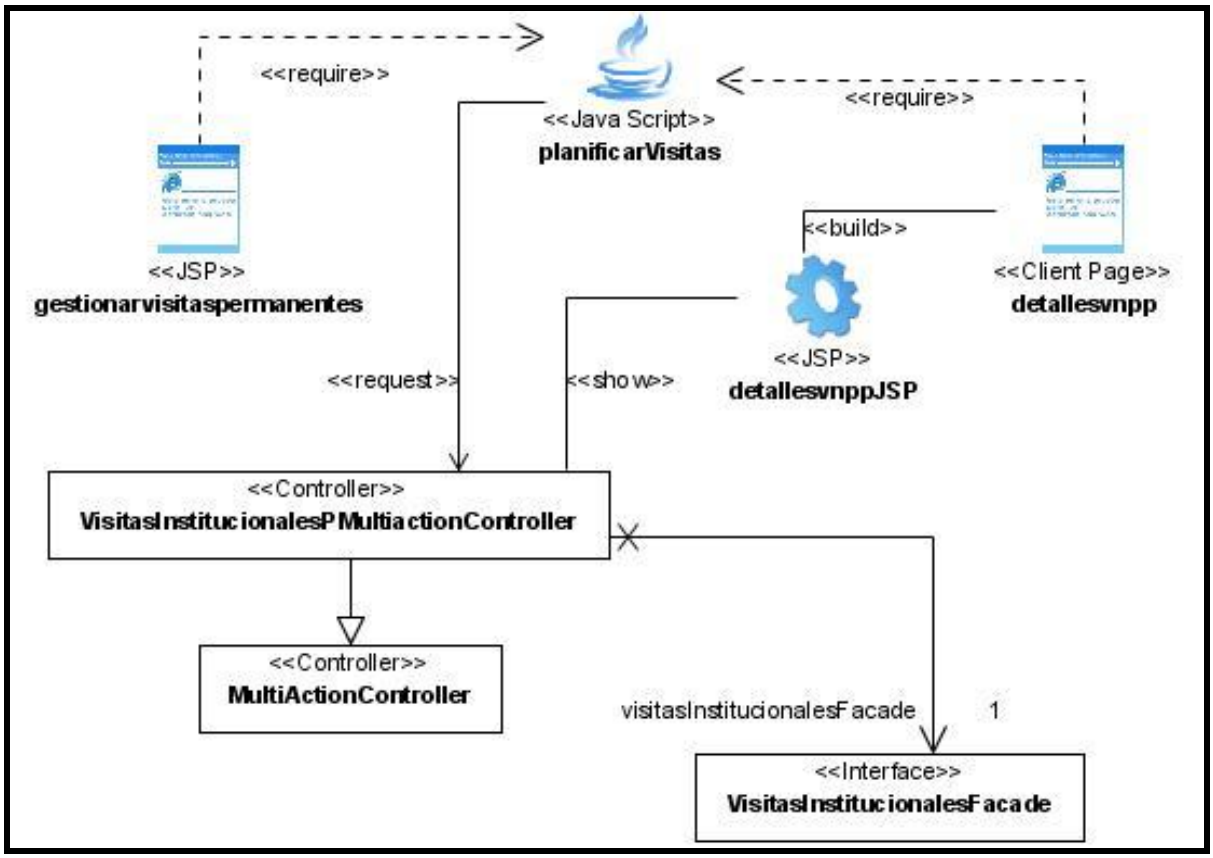

Figura 34 Diagrama de la funcionalidad: Detalles de Visitas Planificadas Permanentes

GLOSARIO

API: Del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Capa de presentación: Generalmente se identifica como la capa Web. Esta capa debe ser tan fina como sea posible. Además, debe permitir diferentes capas de presentación, tales como una capa Web y/o fachadas de servicios Web remotos, sobre una simple y bien diseñada capa de negocio.

JSP: API de la plataforma J2EE para la creación de páginas web dinámicas mediante el uso de librerías de tags.

XML: Formato estándar para el intercambio de datos basado en archivos de texto plano con una estructura de tags.

JDBC: API de la plataforma Java para el acceso a sistemas gestores de base de datos.

Widget: Pequeña aplicación o programa, usualmente presentado en archivos o ficheros.

Inversión de Control: También conocido como inyección de dependencia, se refiere a la forma en que un objeto usa otro objeto.