Universidad de las Ciencias Informáticas
Facultad 4

Título:

"Administración de las Configuraciones y Definiciones de Seguridad del SIGEP"

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor: Eivys Hernández Suárez

Tutor: Luis Alberto Pimentel González

Ciudad de la Habana, mayo del 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 _____.

Eivys Hernández Suárez

_____
Firma del Autor

Luis Alberto Pimentel González

_____
Firma del Tutor

AGRADECIMIENTOS

Quiero agradecer a todas las personas que han hecho posible de una forma u otra que hoy pueda estar optando por el título de Ingeniero Informático. Le agradezco a mis compañeros de proyecto, tanto profesores como estudiantes, por siempre haber compartido las experiencias de trabajo y habernos hecho, los unos a los otros, cada vez mejores profesionalmente. Le agradezco a la UCI y sobre todo a la revolución por darme la posibilidad de estudiar una carrera que me gusta y sentirme útil contribuyendo al desarrollo informático de nuestra nación.

DEDICATORIA

Dedico este trabajo a toda mi familia, a mi papá que es mi guía y mi ejemplo, a mi mamá que sin ella nada fuera posible ya que es quien me cuida y me hace sentir segura, a mis hermanos que son mi esperanza y mi apoyo, a mis abuelos que me dan su amor incomparable, a mi novio que es quien logra iluminarme el corazón y me da la fuerza para seguir adelante. A mis tías, tíos, a mis primas y primos, a toda mi familia que son mi vida.

RESUMEN

El Sistema de Gestión Penitenciario (SIGEP) constituye la solución de software para la informatización de la gestión de los privados de libertad de la República Bolivariana de Venezuela.

La seguridad del SIGEP está constituida por un módulo de Seguridad que se rige por archivos de configuraciones donde se definen los recursos que requieren ser asegurados y quiénes pueden acceder a ellos. Estos datos son definidos manualmente, lo que trae consigo que consuma mucho tiempo el proceso de definición de estas configuraciones y definiciones de seguridad, sean poco adaptables a los cambios que pueda sufrir la aplicación y no puedan ser modificados mientras esté en funcionamiento el sistema.

Como propuesta de solución, se desarrolló un módulo que se encarga de simplificar el proceso de gestión de las configuraciones y definiciones de seguridad, haciéndolo más confiable y eficiente. Permite gestionar las configuraciones y definiciones de seguridad mediante interfaces visuales. Cambia los conceptos de administración de la seguridad haciendo posible que el administrador de seguridad, que forma parte del cliente, sea el encargado de gestionar estas configuraciones y definiciones. Independiza el proceso de desarrollo del software del de administración de la seguridad.

PALABRAS CLAVE

Administración, Seguridad, Configuraciones de Seguridad, Definiciones de Seguridad

TABLA DE CONTENIDOS

INTRODUCCIÓN

El Sistema de Gestión Penitenciario (SIGEP) constituye la solución de software para la informatización de la gestión de los privados de libertad de la República Bolivariana de Venezuela. Este sistema permitirá a los establecimientos penitenciarios (Internados Judiciales, Centros Penitenciarios, Centros de Tratamiento Comunitario) y otras sedes (Coordinaciones Regionales y Técnicas de Apoyo al Sistema Penitenciario), recopilar y controlar la información operativa que se genera en este tipo de centros. Esta solución controlará el tránsito de los privados de libertad por todos los componentes del sistema penitenciario, y en general manejará datos sobre la población penal y auditará los procesos legales para garantizar un cumplimiento justo de la sentencia.

La solución incluye además la gestión de los servicios médicos y alimenticios y el tratamiento ofrecido en estos establecimientos penitenciarios, así como el apoyo para la toma de decisiones estratégicas de la Dirección General de Custodia y Rehabilitación del Recluso.

Debido a la sensibilidad de los datos que se manejan en el SIGEP, se requieren mecanismos que restrinjan el acceso al sistema, de modo que sólo pueda acceder al mismo el personal autorizado, realizar las funcionalidades autorizadas, y desde la ubicación autorizada.

Para darle cumplimiento a estos requisitos de seguridad, el SIGEP implementó mecanismos que se encargan de controlar los usuarios, los roles y los recursos del sistema, así como la forma en que se accede a los mismos. Los mecanismos implementados para darle seguridad al SIGEP constan de un conjunto de configuraciones y definiciones, en los cuales se especifican parámetros que establecen el comportamiento de la seguridad de la aplicación.

En estas configuraciones y definiciones de seguridad se definen usuarios, roles y los recursos del sistema, así como la relación que existe entre ellos. Actualmente en el SIGEP estos aspectos se guardan en XML, archivos de texto, y en Base de Datos (BD), estos datos serán introducidos por el administrador del sistema de forma manual, y algunos son cargados sólo al inicio de la aplicación. Esto trae como desventajas que exista una alta probabilidad de cometer error debido al volumen de información que se manejará manualmente, un alto gasto de tiempo en el proceso, una baja adaptación a los cambios estructurales que pueda sufrir la aplicación ya que si esta cambia entonces también cambiarán muchas de estas configuraciones, y algunos de estos parámetros no pueden ser modificados mientras este en ejecución la aplicación.

Dada esta situación surge el siguiente problema:

¿Qué modificaciones realizar en el proceso de definición de las configuraciones y definiciones de seguridad del SIGEP para lograr mejoras en cuanto a confiabilidad y eficiencia?

Teniendo en cuenta el problema planteado se define como **objeto de estudio**: *Configuraciones y definiciones de seguridad del SIGEP.*

Por lo que se especifica el siguiente **campo de acción**: *Administración de configuraciones y definiciones de seguridad del SIGEP.*

Dadas estas condiciones se plantea lograr como **objetivo general**: *Desarrollar un módulo de administración de la seguridad que permita administrar de manera confiable y eficiente las configuraciones y definiciones que se establecen en la seguridad del SIGEP.*

Como **objetivos específicos**:

- Obtener automáticamente los elementos que requieren ser asegurados.
- Facilitar el proceso de administración de la seguridad mediante la creación de interfaces visuales, delegándole la responsabilidad de la administración de la seguridad al cliente.
- Permitir gestionar y aplicar los cambios en las configuraciones de seguridad sin necesidad de reiniciar la aplicación.
- Permitir salvar las configuraciones y definiciones de seguridad en una estructura confiable.

Según los objetivos planteados se define el siguiente conjunto de **tareas**:

- Identificar todas las configuraciones y definiciones de seguridad que requieren ser administradas.
- Diseñar e implementar mecanismos que permita obtener automáticamente los elementos de la aplicación que requieren ser asegurados.
- Diseñar funcionalidades que permitan administrar cada una de las configuraciones y definiciones identificadas mediante interfaces visuales.
- Diseñar mecanismos que permitan aplicar los cambios realizados en las configuraciones y definiciones de seguridad sin necesidad de reiniciar la aplicación.
- Diseñar y crear un modelo de datos donde se guarden las configuraciones y definiciones de seguridad que serán administradas.
- Implementar los mecanismos que administran las configuraciones y definiciones de seguridad.

- Programación de acceso a datos.
- Programación de la lógica de negocio.
- Programación de interfaz de usuario.

DESARROLLO

1. Descripción más detallada del problema

Los mecanismos de seguridad que posee el SIGEP constan de un conjunto de configuraciones y definiciones en los cuales se definen parámetros que determinan quién, a qué, cuándo y desde dónde se tiene acceso en el sistema.

Estos parámetros están basados en cuatro elementos fundamentales:

Usuario: Personal autorizado a acceder a un sistema informático e interactuar con él, presenta un conjunto de atributos con los cuales puede identificarse ante el sistema, los más comunes son nombre de usuario y contraseña.

Rol: Papel que juegan los usuarios dentro del sistema. Concepto que permite determinar los permisos que tiene el usuario sobre un recurso.

Recurso: Parte componente de un sistema de información. En este contexto se refiere a los elementos con los que el usuario puede interactuar y obtener información del sistema. Ejemplo: URL del sistema y métodos de negocio.

Permiso: Relación que se establece entre un rol y un recurso. Básicamente define quién puede acceder a qué.

Para definir y configurar los elementos de la seguridad se deben llevar a cabo las siguientes actividades:

- Definir los usuarios del sistema.
- Definir los roles del sistema.
- Identificar los recursos de la aplicación.
- Asignar a cada usuario los roles que posee en el sistema.
- Asignar a cada recurso los roles que pueden acceder a él.

¿Cómo se lleva a cabo cada una de estas actividades en el SIGEP?

Definir usuarios, roles y su relación

Para definir los usuarios, roles y su relación se utiliza una estructura en una BD. De los roles se almacena el nombre del rol así como una descripción donde se detallan los aspectos relevantes sobre

el mismo. De los usuarios se define el nombre de usuario, la contraseña, si está habilitado o no, las direcciones IP desde donde puede acceder y los roles que le pertenecen (Figura 1).

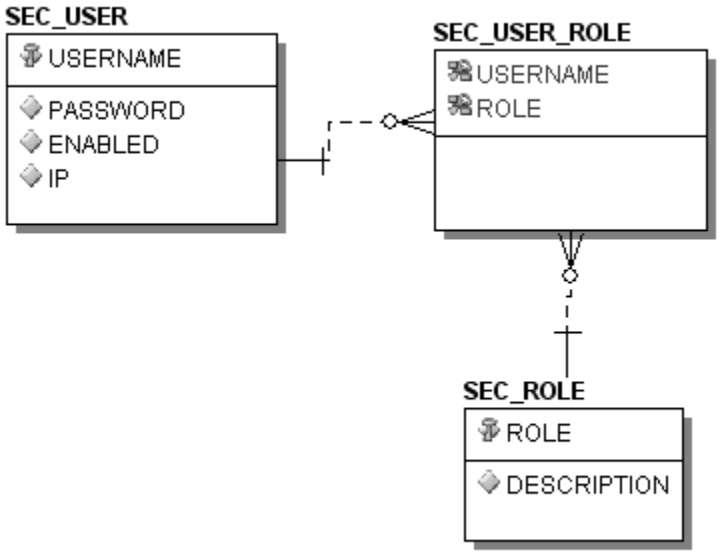


Figura 1: Modelo de datos (MD) que representa usuarios, roles y su relación.

El SIGEP no posee ninguna interfaz gráfica mediante la cual el administrador del sistema pueda introducir los datos de los usuarios y roles del sistema. Actualmente estos valores se introducen directamente en la BD utilizando clientes de BD. Esto trae como consecuencias que se requieran conocimientos técnicos de BD, se imposibilite realizar validaciones previas sobre los datos introducidos y no exista la posibilidad de auditar los cambios que se realicen sobre estos datos.

Como solución se propone elaborar una interfaz visual que permita administrar los usuarios, roles y su relación de manera sencilla y amigable, que permita verificar los datos antes de ser almacenados y que el sistema pueda auditar cada acción que se realice sobre los mismos.

Identificar recursos de la aplicación y definir permisos.

La seguridad del SIGEP necesita tener el control de un conjunto de recursos que existen en la aplicación, y de definir para cada uno de ellos los roles que tendrán acceso. La seguridad del SIGEP se enfoca sobre dos capas arquitectónicas fundamentales, la capa de presentación y la capa de la lógica de negocio, de las cuales se controlan las peticiones URL y las llamadas a los métodos de negocio que son exportados como servicios o son invocados desde la web, respectivamente.

¿Cómo se identifican las URL del SIGEP y se definen los permisos sobre ellas?

Actualmente las URL se definen en un archivo de texto (URL_resource.properties) y son introducidos por el programador de seguridad del SIGEP de forma manual.

El programador de la seguridad del SIGEP, para llevar a cabo el trabajo de identificar los recursos de la aplicación, debe chequear todos los archivos XML que controlan las configuraciones de la capa de presentación de cada módulo, en estos archivos los programadores de interfaz de usuario de cada módulo declaran las URL de sus respectivos módulos y le asigna un controlador a cada petición (Figura 2). El programador de seguridad obtiene de este fichero las URL de la aplicación y las trascribe al fichero URL_resource.properties.

```
<bean id="urlMapping"
    class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <props>
            <prop key="/controlpenal/datospersonales/eliminaridentidad.htm">
                datosPersonalesAjaxController
            </prop>
            <prop key="/controlpenal/datospersonales/salvarnacionalidad.htm">
                nacionalidadController
            </prop>
            <prop key="/controlpenal/datospersonales/nacionalidades.htm">
                nacionalidadTablaController
            </prop>
            <prop key="/controlpenal/datospersonales/otrosnombres.htm">
                otrosNombresController
            </prop>
            <prop key="/controlpenal/datospersonales/actmunicipios.htm">
                datosPersonalesAjaxController
            </prop>
            <prop key="/controlpenal/datospersonales/datospersonales.htm">
                datosPersonalesController
            </prop>
            <prop key="/controlpenal/datospersonales/devolveroficios.htm">
                oficioTablaController
            </prop>
            . . .
```

Figura 2: Declaración de las URL y controladores por parte de los programadores de interfaz de usuario.

Esta actividad es engorrosa, consume mucho tiempo y debe ser chequeada constantemente. Algún cambio en el nombre de las peticiones web que se realizan al servidor que no sea identificado por el programador de seguridad y no sea definida en el archivo URL_resource.properties puede representar un hueco en la seguridad de la aplicación.

Una vez identificadas y definidas todas las URL, el administrador de la seguridad se encargará de asignarle a cada una de ellas los roles que tienen acceso (Figura 3).

```
#   -----   Datos Personales  -----

/controlpenal/datospersonales/eliminaridentidad.htm=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL, ROL_DATOSPERSONALES
/controlpenal/datospersonales/salvarnacionalidad.htm=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL
/controlpenal/datospersonales/nacionalidades.htm=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL, ROL_DATOSPERSONALES
/controlpenal/datospersonales/otrosnombres.htm=ROL_ADMIN, ROL_DATOSPERSONALES
/controlpenal/datospersonales/actmunicipios.htm=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL, ROL_DATOSPERSONALES
/controlpenal/datospersonales/datospersonales.htm=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL, ROL_DATOSPERSONALES
/controlpenal/datospersonales/devolveroficios.htm=ROL_ADMIN, ROL_DATOSPERSONALES
/controlpenal/datospersonales/identidades.htm=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL, ROL_DATOSPERSONALES


#   -----    Ingresos   -----

/controlpenal/ingresos/ingreso.htm=ROL_ADMIN, ROL_INGRESO
/controlpenal/ingresos/cupos.htm=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL, ROL_INGRESO
/controlpenal/ingresos/bienvenido.htm=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL, ROL_INGRESO
/controlpenal/ingresos/jsonrpc.htm=ROL_ADMIN, ROL_INGRESO
/controlpenal/ingresos/expedientecreado.htm=ROL_ADMIN,  ROL_INGRESO
/controlpenal/ingresos/notificaciones.pdf=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL, ROL_INGRESO
/controlpenal/ingresos/concatNotificaciones.pdf=ROL_ADMIN,  ROL_INGRESO
/controlpenal/ingresos/expedientes.htm=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL
/controlpenal/ingresos/procesos.htm=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL
/controlpenal/ingresos/ejecuciones.htm=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL
/controlpenal/ingresos/tribunalesCausa.htm=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL
/controlpenal/ingresos/tiposRemitentesDecisionDocumento.htm=ROL_ADMIN, ROL_INGRESO
```

Figura 3: Definición de permisos a URL declarados en el fichero URL_resource.properties.

Actualmente el SIGEP cuenta con más de 700 URL y no se ha terminado de desarrollar la aplicación, esto implica que es muy trabajoso definir para cada recurso los roles que tienen acceso. El personal que se encargará de definir los permisos puede no conocer la funcionalidad específica que realiza cada URL del sistema y no cuenta con una descripción de la misma para poder definir correctamente los permisos.

Los permisos a URL identificados y definidos en el SIGEP son cargados por la seguridad al iniciarse la aplicación y no pueden ser modificados mientras esté funcionando la misma. Esto implica que si se necesita restringir el acceso a una URL determinada se requiere parar la aplicación y cargar nuevamente las definiciones del fichero URL_resource.properties.

Ante las condiciones actuales que existen para identificar las URL del SIGEP y definir los permisos de las mismas surge la necesidad de:

Encontrar una solución donde los recursos del sistema puedan ser identificados automáticamente y que se reduzca al mínimo probabilidad de cometer errores. Que el encargado de definir los permisos pueda obtener información de las URL, por ejemplo: subsistema, módulo al que pertenece y

descripción de la URL donde se especifique la funcionalidad que realiza. Encontrar un mecanismo mediante el cual, sin detener la aplicación, los permisos a las URL puedan ser modificados.

¿Cómo se identifican los métodos de negocio del SIGEP y definen los permisos sobre ellos?

Para identificar los métodos de negocio de la aplicación que requieren ser asegurados se deben desarrollar actividades muy parecidas a las realizadas con las URL del sistema.

El programador de seguridad debe identificar todos los métodos de negocio que se encuentran definidos en las interfaces de negocio y que posean entre sus anotaciones [1]la anotación llamada JSONExportedMethod, también se deben identificar todos los métodos que se encuentran en las interfaces de servicio. Una vez identificados todos los métodos deben ser transcritos en un fichero llamado methods_resource.properties y el administrador de la seguridad se encargará de asignarle a cada uno de ellos los roles que tienen acceso a él (Figura 4).

```
vnz.sigep.controlpenal.datospersonales.facade.DatosPersonalesFacade.actualizarDatosDeResidenciaIndividuo=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL
vnz.sigep.controlpenal.datospersonales.facade.DatosPersonalesFacade.actualizarDatosFamiliares=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL
vnz.sigep.controlpenal.datospersonales.facade.DatosPersonalesFacade.actualizarDatosPersonales=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL
vnz.sigep.controlpenal.datospersonales.facade.DatosPersonalesFacade.cantidadIndividuoDireccionPorIndividuo=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENA
vnz.sigep.controlpenal.datospersonales.facade.DatosPersonalesFacade.obtenerDireccionActivaIndividuo=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL
vnz.sigep.controlpenal.datospersonales.facade.DatosPersonalesFacade.eliminarGaleriaDeFotos=ROL_ADMIN
vnz.sigep.controlpenal.datospersonales.facade.DatosPersonalesFacade.obtenerSennasParticularesPorIndividuo=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL
vnz.sigep.controlpenal.datospersonales.facade.DatosPersonalesFacade.obtenerRangoDireccionesIndividuo=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL


vnz.sigep.controlpenal.common.service.ControlPenalService.obtenerCentroPorIndividuo=ROL_ADMIN, ROL_FUNCIONARIO_CUSTODIA
vnz.sigep.controlpenal.common.service.ControlPenalService.buscarEstanciaAbiertaPorIndividuo=ROL_ADMIN, ROL_FUNCIONARIO_SALA_SITUACIONAL
vnz.sigep.controlpenal.common.service.ControlPenalService.obtenerFechaCreacionEstancia=ROL_ADMIN, ROL_FUNCIONARIO_SALA_SITUACIONAL
vnz.sigep.controlpenal.common.service.ControlPenalService.obtenerEstanciaMasAntiguaPorExpediente=ROL_ADMIN, ROL_FUNCIONARIO_CUSTODIA
vnz.sigep.controlpenal.common.service.ControlPenalService.obtenerExpedientePorId=ROL_ADMIN, ROL_FUNCIONARIO_CUSTODIA, ROL_FUNCIONARIO_CONTROLPENA
vnz.sigep.controlpenal.common.service.ControlPenalService.obtenerExpedienteActivoPorIdIndividuo=ROL_ADMIN, ROL_FUNCIONARIO_CONTROLPENAL
```

Figura 4: Definición de permisos a métodos declarados en el fichero method_resource.properties.

Estas actividades son engorrosas, consumen mucho tiempo (existen alrededor de 1500 métodos en fachadas de negocio), deben ser chequeados constantemente pues algún cambio en el nombre de un método que no sea identificado por el programador de seguridad y no sea definido en el archivo methods_resource.properties puede representar un hueco en la seguridad de la aplicación. El personal que se encargará de definir los permisos puede no conocer la funcionalidad específica que realiza cada método de negocio y no cuenta con una descripción del mismo para definir correctamente los permisos. Además estos permisos a métodos definidos en el SIGEP son cargados por la seguridad al iniciarse por primera vez la aplicación y no pueden ser modificados mientras esté funcionando la misma. Esto implica que si se necesita restringir el acceso a métodos de negocio específicos, se

[1] Una Anotación Java es una forma de añadir metadatos al código fuente Java que están disponibles para la aplicación en tiempo de ejecución.

requiere parar la aplicación y cargar nuevamente las definiciones del fichero methods_resource.properties.

Ante las condiciones actuales que existen para identificar los métodos de negocio del SIGEP y definir los permisos de los mismos, surge la necesidad de:

Encontrar una solución donde los métodos de negocio del SIGEP puedan ser identificados automáticamente y se reduzca al mínimo la probabilidad de cometer errores. Que el encargado de definir los permisos pueda obtener información de los métodos de negocio, como subsistema en que se encuentra, módulo, clase y una descripción en la que se especifiquen las funcionalidades que este realiza. Implementar un mecanismo donde sin detener la aplicación se puedan modificar los permisos a los métodos de negocio.

2. Solución de software propuesta

La seguridad del SIGEP cuenta con un conjunto de configuraciones y definiciones que necesitan ser administradas. Este trabajo se centra en desarrollar un módulo de administración de la seguridad que permita administrar de manera sencilla, confiable y eficiente las configuraciones y definiciones que se establecen en la seguridad.

Se identificaron las actividades fundamentales que se realizan actualmente en el SIGEP para definir las configuraciones y definiciones de seguridad, así como los inconvenientes que presentan.

Para dar cumplimiento a los objetivos trazados se identificaron los siguientes casos de usos a implementar:

CU-1: Gestionar usuarios del sistema.

CU-1	Gestionar usuarios del sistema.
Actor	Administrador del Sistema.
Descripción	El sistema debe ser capaz de crear y modificar usuarios. De los mismos se debe especificar: nombre de usuario, contraseña, si está habilitado o no, dirección IP de la cual tiene acceso y roles que tiene en el sistema.
Precondiciones	El administrador debe estar autenticado en el sistema y contar con el rol que le da acceso a esta funcionalidad.
Poscondiciones	Los cambios son persistidos en la BD.

CU-2: Gestionar roles del sistema.

CU-2	Administrar roles del sistema.
Actor	Administrador del Sistema.
Descripción	El sistema debe ser capaz de crear, modificar y eliminar roles. Se debe especificar el nombre y la descripción del mismo.
Precondiciones	El administrador debe estar autenticado en el sistema y contar con el rol que le da acceso a esta funcionalidad.
Poscondiciones	Los cambios son persistidos en la BD.

CU-3: Identificar recursos requeridos por la seguridad.

CU-3	Identificar recursos requeridos por la seguridad
Actor	Administrador.
Descripción	El sistema debe ser capaz de identificar de manera automática todos los recursos que requieren ser manejados por la seguridad. De los mismos se debe conocer información referente al subsistema, módulo al que pertenece y una descripción.
Precondiciones	
Poscondiciones	El sistema debe almacenar todos los recursos identificados y los datos requeridos de los mismos.

CU-3: Identificar las URL del sistema.

CU-3	Identificar las URL del sistema.
Actor	Administrador.

Descripción	El sistema debe ser capaz de identificar de manera automática todas las URL. De las mismas se debe contar con información sobre el subsistema, módulo al que pertenece y una descripción.
Precondiciones	Se deben haber definido correctamente las URL en los ficheros XML de configuración de la capa de presentación de cada módulo.
Poscondiciones	El sistema debe almacenar todas las URL identificadas y los datos requeridos de las mismas.

CU-4: Identificar métodos requeridos por la seguridad.

CU-4	Identificar métodos requeridos por la seguridad.
Actor	Administrador.
Descripción	El sistema debe ser capaz de identificar de manera automática todos los métodos de negocio que son exportados como servicio o son invocados desde la web. Debe contar además con información sobre el subsistema, el módulo y la clase a la que pertenece el método, además de una descripción.
Precondiciones	Se debe haber definidos los métodos de negocio de forma correcta en las interfaces de negocio de los módulos.
Poscondiciones	El sistema debe almacenar todos los métodos identificados y los datos requeridos de los mismos.

CU-5 Gestionar permisos.

CU-5	Gestionar permisos.
Actor	Administrador del Sistema.
Descripción	El sistema debe ser capaz de adicionar y eliminar permisos en el sistema. Crear un permiso implica que exista una relación entre un recurso del sistema

	y un rol.
Precondiciones	El administrador debe estar autenticado en el sistema y contar con el rol que le da acceso a esta funcionalidad. El sistema debe identificar y mostrar los roles definidos en la seguridad. Se deben haber Identificado los recursos requeridos por la seguridad.
Poscondiciones	Los cambios realizados deben ser persistidos en la BD y cargados por la seguridad.

CU-6 Gestionar permisos a URL.

CU-6	Gestionar permisos a URL.
Actor	Administrador del Sistema.
Descripción	El sistema debe ser capaz de adicionar y eliminar permisos a URL. Crear un permiso a URL implica que exista una relación entre una URL y un rol.
Precondiciones	El administrador debe estar autenticado en el sistema y contar con el rol que le da acceso a esta funcionalidad. El sistema debe identificar y mostrar los roles definidos en la seguridad. Se debe haber ejecutado el caso de uso Identificar URL del Sistema.
Poscondiciones	Los cambios realizados deben ser persistidos en la BD y cargados por la seguridad.

CU-7 Gestionar permisos a métodos.

CU-7	Gestionar permisos a métodos.
Actor	Administrador del Sistema.
Descripción	El sistema debe ser capaz de adicionar y eliminar permisos. Crear un permiso

	implica que exista una relación entre un recurso del sistema y un rol.
Precondiciones	El administrador debe estar autenticado en el sistema y contar con el rol que le da acceso a esta funcionalidad. El sistema debe identificar y mostrar los roles definidos en la seguridad. Se debe haber ejecutado el caso de uso Identificar métodos requeridos por la seguridad
Poscondiciones	Los cambios realizados deben ser persistidos en la BD y cargados por la seguridad.

Diagrama de Casos de Uso del Sistema:

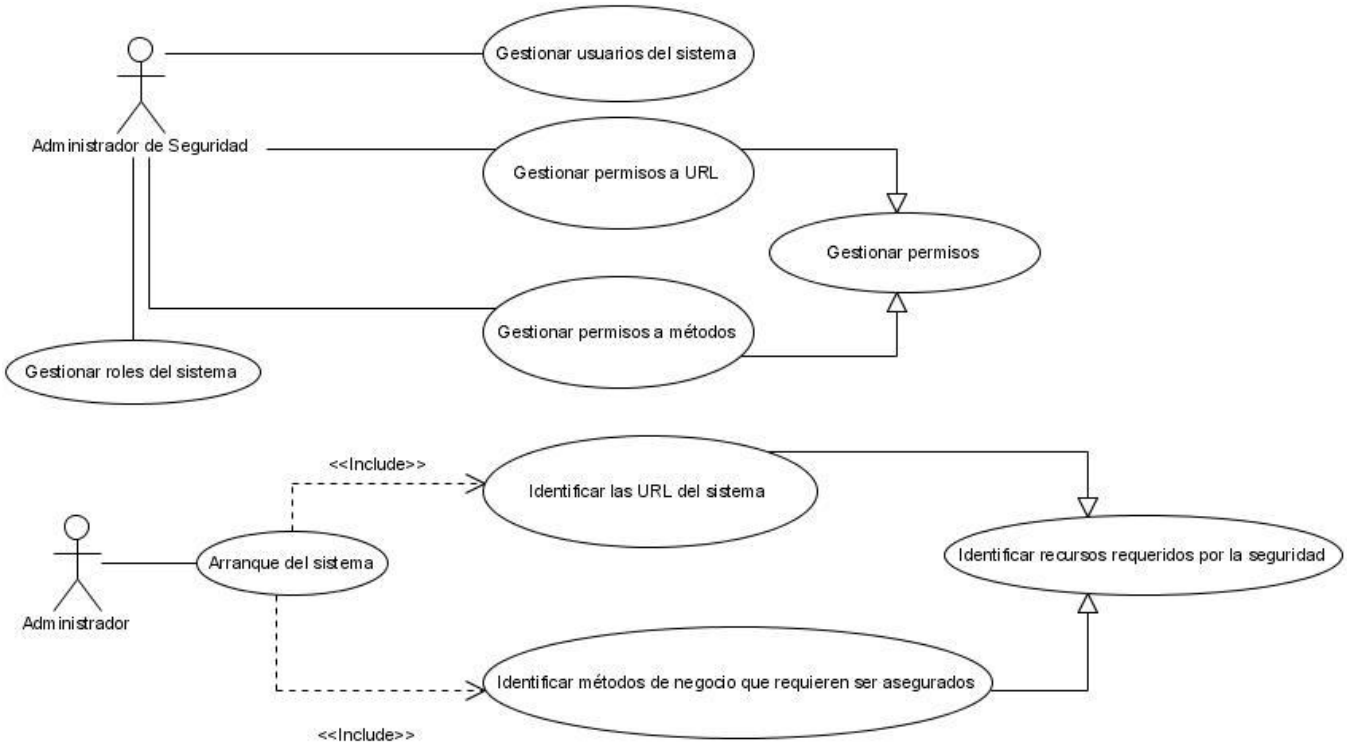


Figura 5: Diagrama de Casos de Uso del Sistema

3. Técnicas, métodos y herramientas utilizadas

Técnicas y métodos utilizados

Programación orientada a aspectos:

La Programación Orientada a Aspectos es un paradigma de programación relativamente reciente cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos. Permite insertar funcionalidades sobre otras ya implementadas sin necesidad de modificar el código original.

Se utilizo la programación orientada a aspectos en la definición de las transacciones sobre los métodos de negocio. La utiliza además el framework Acegi para interceptar todos los métodos a los cuáles se le aplica la seguridad.

Programación mediante Eventos:

La programación dirigida por eventos es un paradigma de programación en el que tanto la estructura como la ejecución de los programas van determinados por los sucesos que ocurran en el sistema o que ellos mismos provoquen.

Se utilizó la programación orientada a eventos en el módulo de administración para intercambiar información con el núcleo de la aplicación y con la seguridad; por ejemplo: conocer cuando el la capa de negocio de la aplicación ha sido totalmente cargada y para la administración notificarle a la seguridad que debe recargar las configuraciones.

Hilos de ejecución:

Los hilos se utilizan para independizar procesos de ejecución dentro de la programación.

Es utilizado por la administración durante el proceso de recarga de las configuraciones de seguridad. Mientras el módulo de seguridad está recargando las configuraciones definidas en la administración de permisos, la aplicación se encuentra en un estado inestable en el manejo de la seguridad, puesto a que el mecanismo de seguridad está reconfigurándose. Mientras esto sucede los clientes pueden continuar realizando peticiones con una alta concurrencia, debe asegurar que ninguna de estas peticiones sea tratada por la aplicación hasta más la seguridad no haya terminado de reconfigurarse y se encuentre en un estado estable nuevamente.

Como solución se implementó un mecanismo que se encarga de pausar los hilos de ejecución iniciados por cada petición del cliente y una vez alcanzado el estado estable de la seguridad todos los hilos de ejecución son reactivados para que sean ejecutados por la aplicación. Este proceso ha sido

probado en concurrencia y no afecta el tiempo de respuesta de la aplicación dado el poco tiempo que emplea la seguridad en recargarse en un servidor con altas prestaciones.

Ajuste de frameworks a las necesidades:

Los frameworks son básicamente librerías de clases que resuelven un problema concreto o brindan utilidades sobre áreas definidas. Es común de que se necesite variar el funcionamiento que propone algún framework por no ajustarse a las necesidades de una situación específica que pueda ocurrir.

En muchos casos fue necesario variar el comportamiento de algunas funcionalidades de Acegi para permitirle al módulo de administración realizar las funcionalidades previstas. Como técnica común ante este tipo de situación, se realizó el procedimiento de heredar de las clases que se querían variar e implementar nuevamente las funcionalidades necesarias. Este método evita tener que cambiar código dentro de los frameworks.

Ambiente de desarrollo

A continuación se enuncian los principales componentes del ambiente desarrollo acompañado de una breve descripción.

Entorno de desarrollo integrado:

Eclipse 3.3

Es una extensible plataforma de código abierto (open source) para desarrollar herramientas. Es administrado y dirigido por un consorcio de compañías de desarrollo de software con un interés comercial en promover Eclipse como plataforma compartida para herramientas de desarrollo de software.

Plugins

Un plugin o plug-in, es una aplicación que interactúa con otra para agregarle una funcionalidad específica y es ejecutada por la aplicación principal. En el caso particular de Eclipse no son más que un conjunto de clases que permiten hacerlo más extensible.

Web Tool Platform (WTP): La plataforma de herramientas web de Eclipse o Eclipse Web Tool Platform (WTP) provee varias API [2]para desarrollo de aplicaciones sobre la Web y JEE[3]. Estas

[2] Una 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.

incluyen editores gráficos de código fuente para una variedad de lenguajes, asistentes y aplicaciones, incorporadas para simplificar el desarrollo de servicios web, además de herramientas y API para soportar el despliegue, ejecución y prueba 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.

Hibernate Tools: Hibernate Tools es un conjunto de herramientas enteramente para trabajar con el framework Hibernate del cual se amplía en este capítulo, implementado como una suite integrada de plugins para Eclipse.

Subclipse: Subclipse es un plugin para Eclipse que adiciona integración para el control de versiones (Subversion, específicamente), permitiendo operaciones de sincronización, actualización, entre otras.

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.

Apache Tomcat 6.0

Apache Tomcat es un contenedor de Servlet usado en la implementación de referencia oficial para las tecnologías Java Servlet y JavaServer Pages. Es desarrollado en un ambiente participativo y abierto.

Subversion

Subversion es un software de sistema de control de versiones de código abierto y gratuito.

Oracle

Oracle es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés Relational Data Base Management System), fabricado por Oracle Corporation. Se considera uno de los

[3] Java Platform, Enterprise Edition o Java EE, 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.

sistemas de bases de datos más completos. Es un sistema gestor de base de datos robusto, tiene muchas características que nos garantizan la seguridad e integridad de los datos; que las transacciones se ejecuten de forma correcta, sin causar inconsistencias; ayuda a administrar y almacenar grandes volúmenes de datos; estabilidad, escalabilidad y es multiplataforma.

Herramientas de modelado

Los medios con los que siempre se ha realizado el intercambio de información de diseño e ideas usando la notación UML han sido populares: pizarras, cuadernos y trozos de papel, etcétera. Pero UML se utiliza mejor a través de una herramienta de modelado, la cual puede ser usada para capturar, guardar, rechazar, integrar automáticamente información, y diseño de documentación.

ER/Studio

ER/Studio es una herramienta de modelado para diseñar bases de datos. Ayuda a descubrir, documentar y reutilizar los datos activos. Con un soporte de ida y vuelta de bases de datos, los arquitectos de datos tienen el poder para analizar a donde las fuentes de datos existentes tan bien como el diseño e implementación de bases de datos de alta calidad.

Visual Paradigm Suite

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

A continuación se mencionan las principales herramientas presentes dentro de esta suite:

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

Plataforma JEE

JEE es una plataforma que habilita soluciones para desarrollo, uso efectivo y manejo de multicapas en aplicaciones centralizadas en el servidor. Define los estándares para desarrollar aplicaciones empresariales en Java. JEE simplifica el desarrollo de este tipo de aplicaciones, basándolas en componentes modulares y estandarizados, y proporcionando un conjunto de especificaciones que aseguran la portabilidad de las aplicaciones entre un amplio número de productos comerciales y de códigos abiertos existentes, capaces de soportar JEE. Esta plataforma cuenta con las siguientes características:

Portable; puede utilizarse en cualquier plataforma para la que haya disponible una Máquina Virtual Java (JVM).

Escalable; soporta el aumento de clientes, sin tener que reescribir todo el código de nuevo, tan solo añadiendo nuevos componentes JEE a una aplicación Web.

Altamente Soportada; prácticamente cualquier gran empresa de software tiene un contenedor de componentes (o servidor de aplicaciones) Web compatibles con JEE.

Segura; el entorno de seguridad de la plataforma JEE permite que se definan unas restricciones de seguridad en el momento de despliegue de la aplicación, aislando las aplicaciones de la complejidad de las implementaciones de seguridad, la plataforma JEE hace portables una gran complejidad de implementaciones de seguridad (Ciberaula 2006).

Frameworks para la Plataforma JEE

Un framework es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Puede incluir soporte de programas, bibliotecas, y un lenguaje de scripting, entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. Son diseñados con el intento de facilitar el desarrollo de software.

Spring

Spring Framework es un framework 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 (Wiley 2005).

Presenta un diseño que brinda una gran flexibilidad arquitectónica e interviene en todas las capas de una aplicación.

El mismo presenta varios módulos de los cuales los más importantes son:

- Contenedor de Inversión de Control.
- Un framework para la Programación Orientada a Aspectos (AOP).
- Una abstracción de acceso a datos.
- Simplificación de JDBC.
- Administración de transacciones.
- Framework Web MVC.
- Simplificación para trabajar con JNDI, JTA y otros APIs de JEE.

- Comunicación remota de peso ligero (Lightweight remoting).
- Soporte para Servicio de Mensajería de Java (Java Message Service, JMS).
- Soporte para Java Management Extensión (JMX).
- Soporte para comprensivas estrategias de pruebas para desarrolladores de aplicación.

Spring MVC

Spring proporciona un framework MVC (Model-View-Controller) Modelo-Vista-Controlador, construido sobre el núcleo de Spring. Este framework es altamente configurable vía interfaces y permite el uso de múltiples tecnologías para la capa vista, entre las que se encuentran JSP. De cualquier manera una capa modelo, realizada con Spring, puede ser fácilmente utilizada con una capa Web basada en cualquier otro framework MVC, como Struts, entre otros.

Hibernate

Hibernate es un software gratuito, de código abierto. Fue realizado por un equipo de desarrolladores de software Java de todo el mundo, capitaneado por Gavin King. Es una solución de Mapeo de los objetos de negocio con la información almacenada en bases de datos relacionales. Se encarga del mapeo de clases de Java a tablas de la base de datos y la generación de query's a la base de datos.

Hibernate facilita la migración de sistemas entre diferentes motores de bases de datos. Reduce aproximadamente el 95% de las tareas que un programador tenía que hacer para realizar funciones comunes de acceso a datos. Ofrece un framework fácil de usar para mapear un modelo orientado a objetos a una tradicional base de datos relacional. Hibernate soporta la mayoría de los sistemas de bases de datos SQL. Brinda facilidades para recuperación y actualización de datos, control de transacciones, repositorios de conexiones a bases de datos, consultas programáticas y declarativas, y un control de relaciones de entidades declarativas. (Mateos 2005).

Acegi

Acegi Security es un framework de código abierto altamente usado por la comunidad de Spring para proveer los servicios de seguridad a las aplicaciones basadas principalmente en Spring Framework. El diseño de Acegi le permite a las aplicaciones sobre java aplicar los cuatro requerimientos de seguridad más comunes de las aplicaciones empresariales: autenticación, seguridad sobre las peticiones web, seguridad sobre la capa de servicios y seguridad a las instancias de los objetos de dominio. Provee integración a la aplicación con la mayoría de los posibles requerimientos específicos para cada aplicación como el uso de Https, integración con LDAP, servicios para recordar usuarios e integración con las funcionalidades de Captcha.

JSON-RPC

JavaScript [4]Object Notation (JSON) es un formato de intercambio de datos de peso ligero. Este es de fácil lectura y escritura para los humanos y es fácil para ser parseados y generados por las maquinas. JSON es un formato de texto que es completamente independiente del lenguaje pero usa convenciones que son familiares a los programadores de los lenguajes de la familia de C, como son C, C++, C#, Java, JavaScript, Perl, Python, entre otros. Estas propiedades hacen a JSON un lenguaje de intercambio de datos ideal.

Arquitectura del SIGEP

La arquitectura definida para el desarrollo del SIGEP está basada en una arquitectura de tres capas lógicas fundamentales: Capa de Presentación, Capa de Lógica de Negocio, y Capa de Acceso a Datos (Figura 6). Dichas capas están bien delimitadas una de la otra, una capa superior interactúa con la inferior mediante interfaces, que definen las funcionalidades que la misma debe brindar.

En la arquitectura del SIGEP se define una fachada que representa la interacción entre las capas de Presentación y Lógica de Negocio, la utilización de esta fachada cumple con el patrón de diseño Facade[5].

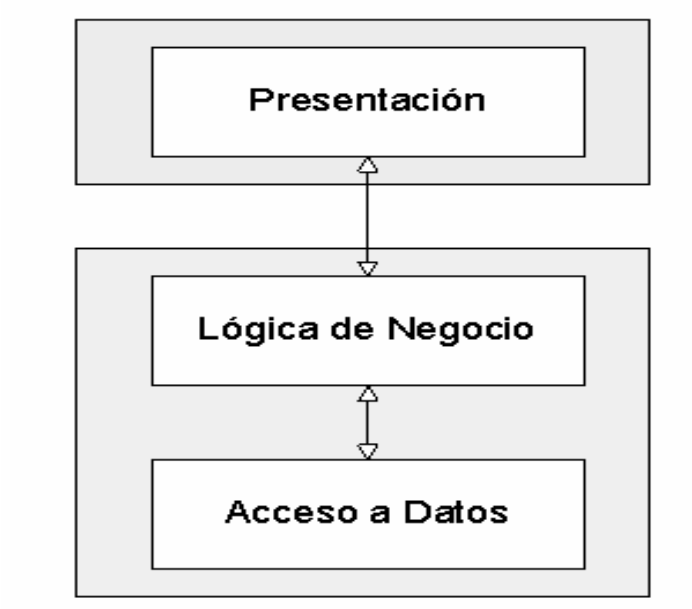


Figura 6: Capas Arquitectónicas del SIGEP.

[4] JavaScript: es un lenguaje de programación utilizado para crear pequeños programitas encargados de realizar acciones dentro del ámbito de una página web.

[5] Patrón Facade: El patrón de diseño Facade sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas.

Capa de Presentación:

Esta capa web es la responsable de tratar con las interacciones del usuario y obtener los datos que pueden ser mostrados en un formato determinado. Normalmente está compuesta por tres tipos de objetos:

- Controladores: Estos objetos son responsables de procesar las entradas del usuario en forma de peticiones HTTP, invocando las funcionalidades necesarias, expuestas por la capa de servicios de negocio y devolviendo un modelo requerido para ser mostrado.
- Modelo: Estos objetos contienen los datos resultantes de la ejecución de la lógica de negocio, los cuales deberían ser mostrados en la respuesta.
- Vistas: Estos objetos son responsables de mostrar el modelo en la respuesta de la petición. La forma de mostrar el modelo podrá ser de diferentes tipos de vistas, por ejemplo, archivos JSP, HTML, PDF, documentos de Excel, etcétera. Las vistas no son responsables de modificar los datos o incluso de obtener los datos; estas simplemente sirven para mostrar los dados del modelo que han sido suministrados por un controlador.

Capa de Servicios de Negocio:

En esta capa radican los objetos de negocio o entidades del dominio. Los objetos de negocio separan los datos y la lógica de negocio usando un modelo de objetos.

Por cada módulo se definirá una o más fachadas en caso de que se requiera que agrupen los métodos de negocio implementados en los manejadores o Managers que serán explicados más adelante. La fachada de un módulo se basa en el patrón de diseño Facade para permitir una clara división entre las capas arquitectónicas. Los Managers son las clases que se especializan en un conjunto de funcionalidades que representan el negocio sobre una o varias entidades. Los Managers son las únicas clases en la aplicación que tendrán lógica de negocio mientras que las fachadas se limitarán solamente a agrupar las funcionalidades de para ser expuestas a capas superiores.

Capa de Acceso a Datos:

La interacción del negocio con la capa de acceso a datos se realizará mediante el uso de interfaces. El término de Objeto de Acceso a Datos o en inglés, Data Access Object (DAO) es ampliamente usado en el desarrollo de software.

Los DAO encapsulan el manejo de acceso a datos de los objetos de dominios, proveen la persistencia de los objetos transitorios y las actualizaciones de los objetos existentes en la base de datos. Las implementaciones de los DAO estarán disponibles para los objetos (típicamente para los objetos de

negocio) haciendo uso de la inyección de dependencias con los objetos de negocio y las instancias de los DAO, configurada en el contenedor de inversión de control de Spring Framework.

Las interfaces de los DAO contienen básicamente los siguientes tipos de métodos:

- Métodos para descubrir**:** Estos localizan los objetos almacenados para ser usados por la capa de servicios de negocio.
- Métodos para persistir o salvar**:** Estos hacen persistentes a los objetos transitorios.
- Métodos para eliminar**:** Estos eliminan a los objetos guardados en el medio de almacenamiento (generalmente una base de datos).
- Métodos para conteos y otras funciones agregadas**:** Estos devuelven los resultados de operaciones que son más eficientes implementarlas usando funcionalidades de la base de datos (procedimientos almacenas, etcétera.) que iterar sobre los mismos objetos.

4. Descripción de la solución propuesta

Gestionar usuarios del sistema

El caso de uso Gestionar Usuarios del Sistema es el encargado de crear y modificar los usuarios del sistema. Cada usuario del sistema debe corresponder a un sólo miembro del penal al cual pertenece. Una vez creado el usuario del sistema, debe ser asignado a un miembro de la institución penal y una vez persistido el usuario y su relación con el personal no podrán ser eliminados. El usuario también contará con una conjunto de roles que definen las funcionalidades a las cuales tendrá acceso en el sistema.

Las clases que intervienen en la gestión de usuarios son la clases User, Role y Personal.

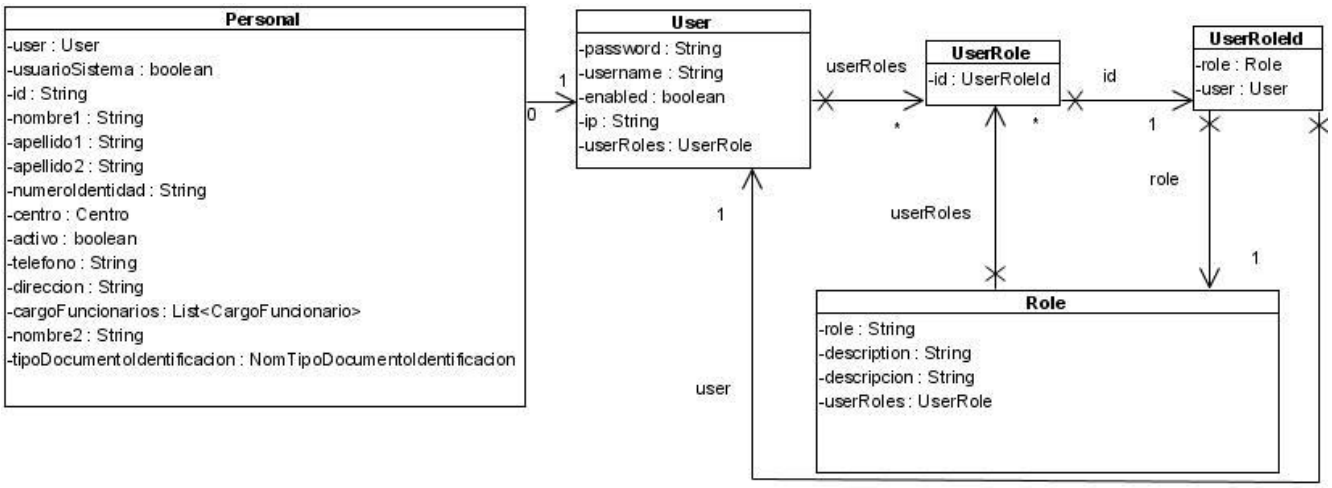


Figura 7: DC que muestra la relación entre las clases User, Role y Personal.

Descripción de los atributos de la clase *User*:

username: Nombre con el cual se va a identificar el usuario en el sistema.

password: Contraseña que debe introducir el usuario para que el sistema valide que es quien dice ser.

ip: Direcciones IP separadas por “coma” que definen desde qué IP o rangos de IP el usuario podrá acceder al sistema.

enabled: Atributo booleano que define si el usuario puede o no operar en el sistema. Si el valor es “true” el usuario puede operar el en sistema y acceder a las funcionalidades que tiene acceso, si el valor es “false” el usuario queda inhabilitado para operar en el sistema, incluso aunque cuente con permisos para hacerlo.

userRoles: Colección de roles con los que cuenta el usuario.

Diagrama de clases de la web, se muestran los principales componentes de la capa de presentación y sus relaciones, referentes a la gestión de los usuarios del sistema.

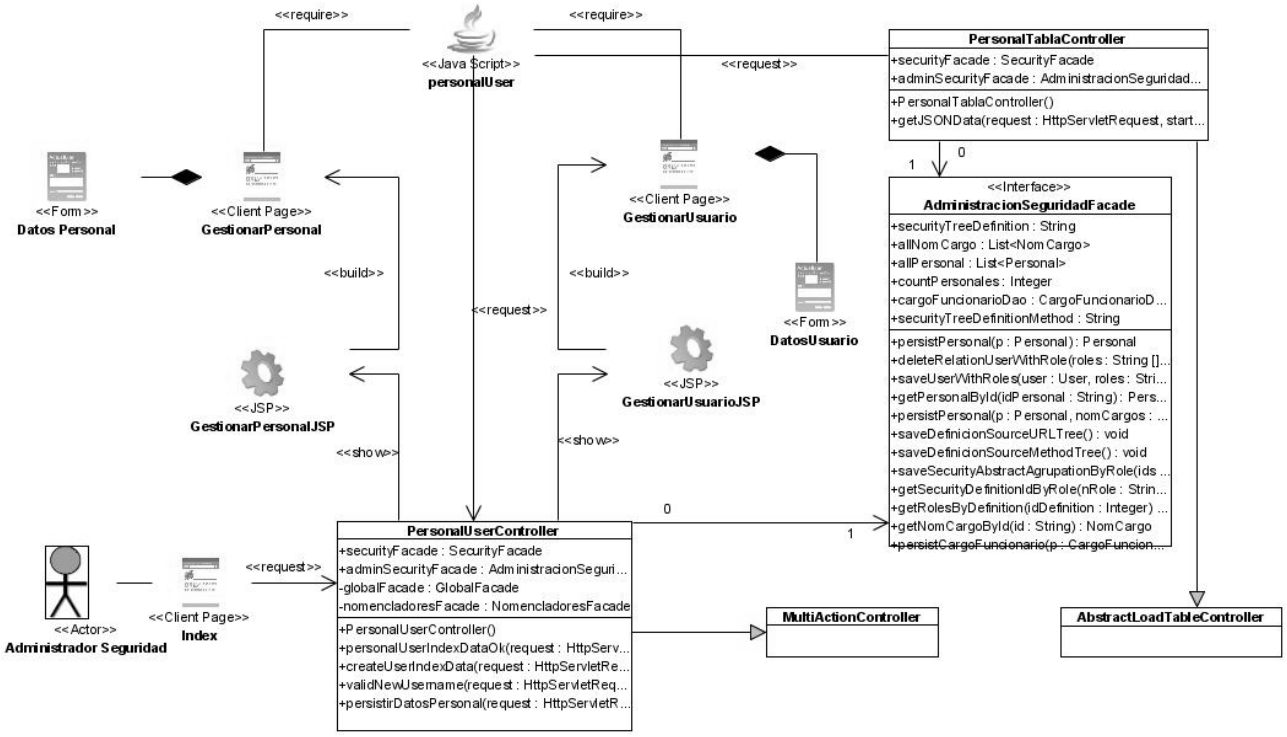


Figura 8: Diagrama de Clases de la Web (DCW) del caso uso Gestionar Usuarios del Sistema.

Los datos de los usuarios serán almacenados en la estructura de la BD definida por la seguridad del SIGEP (Figura 1), además, en este modelo también interviene la tabla Personal, que es la encargada de almacenar datos del personal del penal (Figura 9).

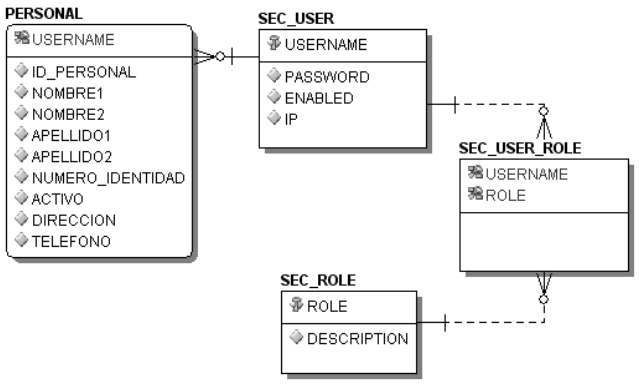


Figura 9: MD que representan los Usuarios, Roles y el Personal.

Gestionar roles del sistema

El caso de uso Gestionar Roles es el encargado de crear y modificar los roles del sistema. En este caso de uso sólo interviene la clase de dominio Role. Los roles podrán ser adicionados, modificados y eliminados.

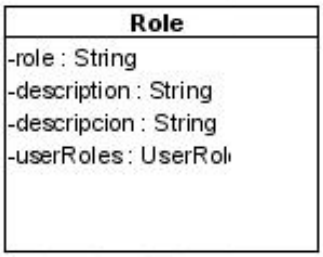


Figura 10: Clase Role.

Descripción de los atributos de la clase Role:

role: Define el nombre con el cual se va a identificar el rol.

description: Atributo donde se especifica el objetivo que debe cumplir el rol.

Los datos de los roles serán almacenados en la estructura de la BD definida por la seguridad del SIGEP (Figura 1).

El diseño de clases de la web que interviene en la implementación de este caso de uso se representará en la descripción de la implementación del caso de uso Gestionar Permisos, debido a que estas funcionalidades están muy relacionadas.

Identificar recursos requeridos por la seguridad:

La seguridad del SIGEP necesita tener el control de un conjunto de recursos que existen en la aplicación. Se centra en dos capas fundamentales: la capa de presentación y la capa de la lógica de negocio, de las cuales se controlan las peticiones URL y las llamadas a los métodos de negocio que son exportados como servicios o son invocados desde la web mediante JSON-RPC, respectivamente.

Se definieron dos casos de uso del sistema: Identificar URL del Sistema e Identificar Métodos Requeridos por la Seguridad que extienden del caso de uso Identificar Recursos Requeridos por la Seguridad. Aunque el proceso de identificación de los recursos es diferente en cada caso de uso, la información que se necesita obtener de ellos es similar.

Tanto de las URL del sistema como de los métodos se necesita controlar el nombre, una descripción y su origen, por ejemplo: subsistema y módulo al cual pertenece, en el caso de las URL pudiera

identificarse el caso de uso al cual pertenece la URL, y en caso del método la clase de la cual proviene.

Para representar los recursos de seguridad, tanto URL como métodos, se define un conjunto de clases relacionadas entre sí que se encargan de almacenar estos recursos y los datos adicionales necesarios.

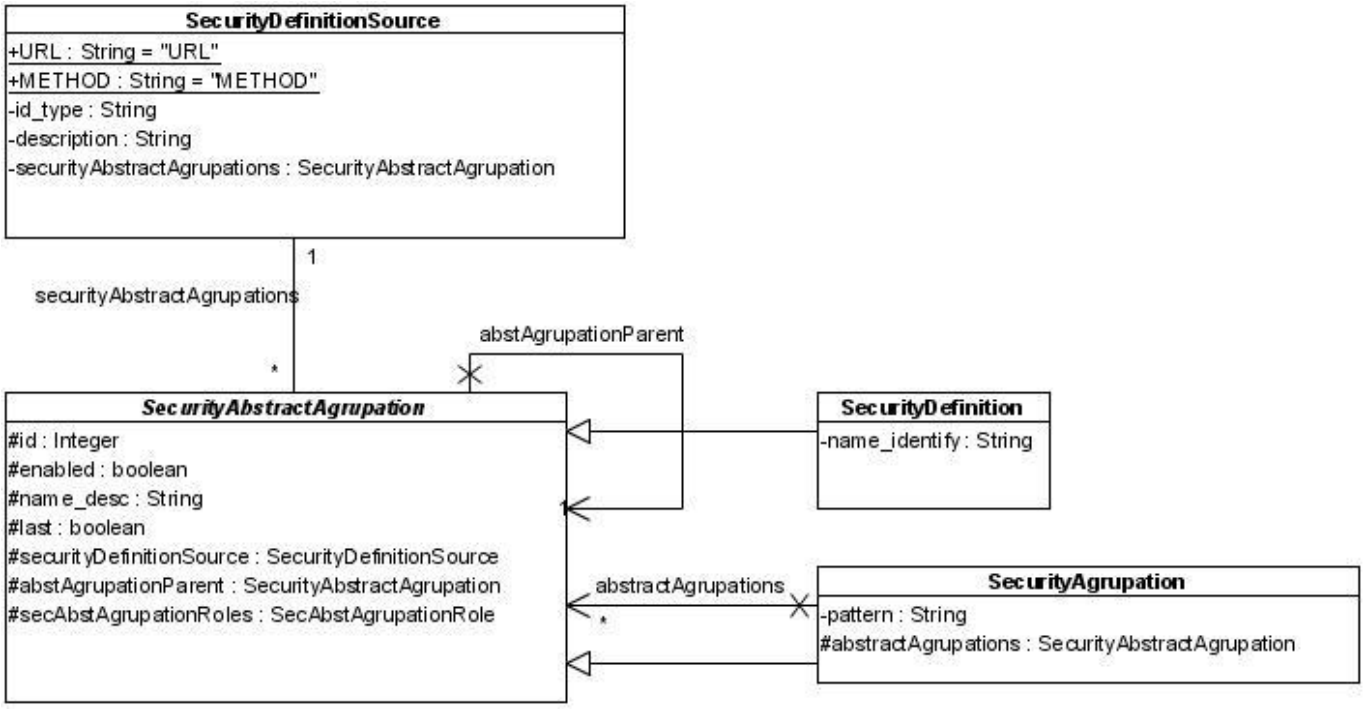


Figura 11: DC que modela los recursos de la aplicación.

Descripción de las clases y sus atributos:

SecurityDefinitionSource: Se utiliza para definir qué tipo de recurso está siendo identificado (URL o método), sus atributos son:

id_type: Representa un tipo de recurso, los valores válidos son: “URL” y “METHOD”.

description: Su valor describe el tipo de recurso que está siendo identificado.

securityAbstractAgrupation: Colección de *SecurityAbstractAgrupation*.

SecurityAbstractAgrupation: Clase abstracta que contribuye a representar la información que se necesita obtener de elementos del sistema por ejemplo: subsistema, módulo, URL, clase, método, etc. De ella heredan dos clases *SecurityAgrupation* y *SecurityDefinition*, los atributos de esta clase son:

id: Identificador del objeto.

name_desc: Representa el nombre o la descripción, en caso de que la instancia sea un objeto del tipo *SecurityAgrupation* representa el nombre, en caso de que sea del tipo *SecurityDefinition* representa la descripción.

last: Es un atributo booleano que define si el valor del objeto es el último de la jerarquía.

enabled: Atributo booleano que define si a este elemento del sistema es posible acceder. Si el valor es “true” podrá acceder a él quien tenga acceso, si el valor es “false” nadie podrá acceder a él aunque cuente con los privilegios para hacerlo.

abstractAgrupationParent: Representa al padre de este elemento. Por ejemplo, si se está representando un módulo, el padre es el subsistema al que pertenece.

SecurityAgrupation: Representa al igual que se padre a un elemento del sistema, pero más específicamente a los elementos que puedan agrupar a otros elementos, por ejemplo: un subsistema, un módulo, un caso de uso, no representa ni las URL ni los métodos. Esta clase tiene como atributos todos lo que posee su clase padre *SecurityAbstractAgrupation* y los que se muestran a continuación:

pattern: Representa el patrón con el cual se puede identificar esta agrupación, si se están identificando las URL, el patrón estará dado por el patrón que pueden tener las URL de un módulo, subsistema o caso de uso específico. Si se están identificando los métodos entonces el patrón será la llave con la que se identifica un subsistema, un módulo, o una clase.

abstractAgrupation: Representa la colección de *SecurityAbstractAgrupations* que pueden existir dentro de un *SecurityAgrupation*, de esta forma se representan agrupaciones dentro de otras agrupaciones.

SecurityDefinition: Representa, al igual que su padre, una parte del sistema, específicamente los recursos que necesitan ser manejados por la seguridad como las URL y los métodos. Esta clase tiene como atributos todos lo que posee su clase padre *SecurityAbstractAgrupation* y el que se muestra a continuación:

name_identify: Contiene el nombre de la URL o el nombre del método que requiere ser manejado por la seguridad.

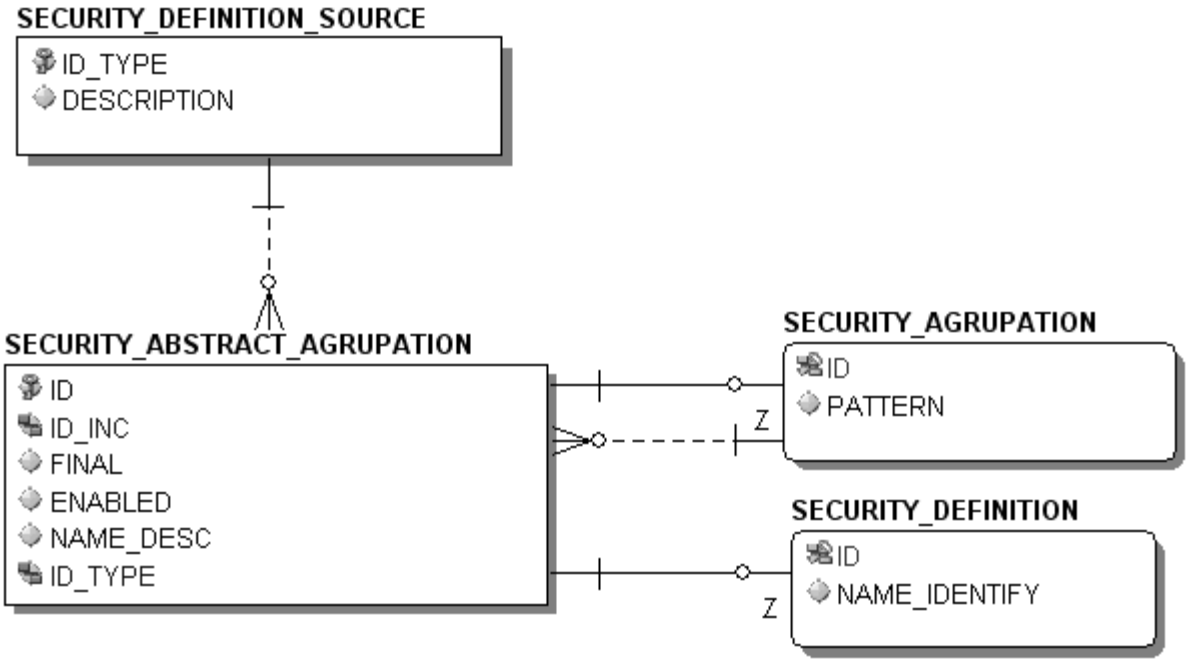


Figura 12: MD donde se almacenan los recursos de seguridad identificados.

Una vez definidas las clases que se encargan de encapsular los datos de los recursos que necesita controlar la seguridad, se expone cómo se Identifican las URL del sistema y los métodos requeridos por la seguridad.

Identificar las URL del sistema

La Seguridad del SIGEP necesita tener el control de todas las URL que existen en la aplicación. Se ha definido una estructura de clases que encapsula los datos de estos recursos; pero, ¿cómo se obtendrán de manera automática todas las URL y todos los datos adicionales de los cuales también se necesita tener control?

Como se mencionó anteriormente, las URL de cada módulo se definen por el programador de interfaz de usuario en los archivos XML que controlan las configuraciones de la capa de presentación de cada módulo. Sin embargo esta información que obtenemos no nos es suficiente, necesitamos una descripción de la función que realiza cada URL del sistema, además de conocer el subsistema, módulo e incluso poder identificar el caso de uso o funcionalidad a la cual responde una URL específica.

Para darle respuesta a este requerimiento de identificar las URL del sistema y obtener la información necesaria de las mismas, se desarrolló un conjunto de clases que ejecutan esta actividad.

Para tener control de las URL que se declaran en el fichero XML de configuración de la capa de presentación y para permitir definir estas URL agrupadas por casos de uso o por funcionalidades específicas que se realizan en un módulo, se definieron cuatro clases fundamentales: *Agrupation*, *MappingAgrupation*, *URLAgrupation* y *UrlMappingRegister*.

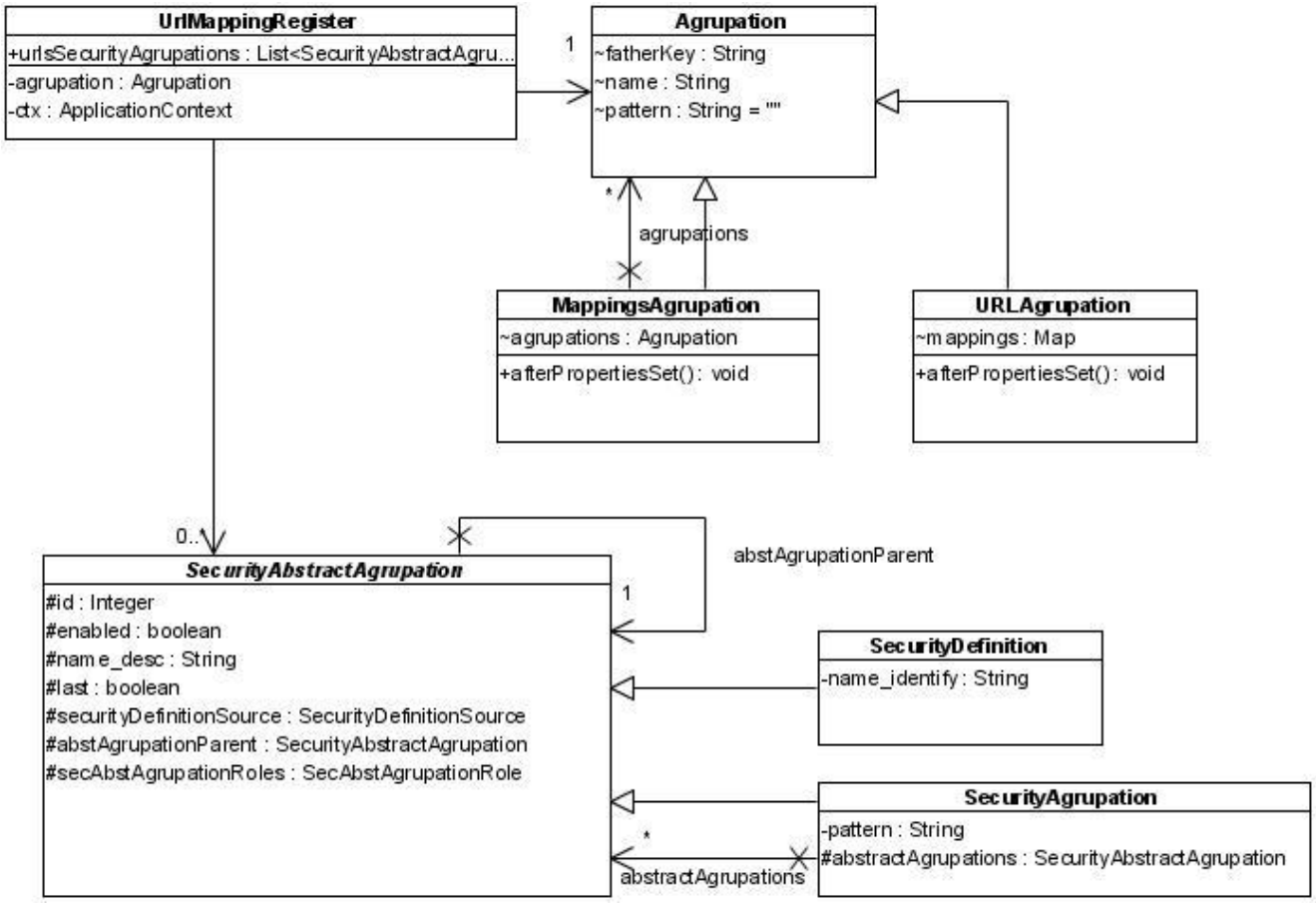


Figura 13: Clases encargadas de almacenar las agrupaciones de las URL.

Descripción de las Clases:

Agrupation: Clase abstracta que define una agrupación. De ella heredan dos clases *MappingsAgrupation* y *URLAgrupation*. Tiene como atributos:

name: Especifica el nombre de la agrupación.

pattern: Especifica el patrón con el que va a cumplir esta agrupación.

fatherKey: Especifica la llave de la agrupación que la contiene a él.

MappingsAgrupation: Clase que define una agrupación que contiene más agrupaciones, de esta forma se pueden definir dentro de un módulo agrupaciones que respondan a casos de uso o a funcionalidades comunes del módulo. El atributo *agrupations* define esta colección de agrupaciones.

URLAgrupation: Representa una agrupación. A diferencia de la clase *MappingAgrupation* esta clase agrupa sólo un conjunto de URL del sistema. Las URL se definen en el atributo *mappings* que es el tipo java.util.Map. Cada elemento del mapa define como llave la URL, separado por coma la descripción y como valor el nombre del controlador que le corresponde a esa URL.

Estas clases se definen en el XML de configuración de la capa de presentación en forma de árbol donde el nodo raíz se le inyecta a la clase *SimpleHandlerMapping* del framework Spring siendo inyectado al atributo *mappings* de esta clase.

```
<bean id="urlMapping"
    class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings" ref="securityModuleAgrupacion"></property>
</bean>
<bean name="securityModuleAgrupacion"
    class="vnz.sigep.administracion.seguridad.util.urlagrupation.MappingsAgrupation">
    <property name="fatherKey">
        <value>Administración</value>
    </property>
    <property name="name">
        <value>Seguridad</value>
    </property>
    <property name="pattern">
        <value>/seguridad</value>
    </property>
    <property name="agrupations">
        <list>
            <ref bean="asegurarUrls" />
            <ref bean="gestionarRolesPermisos" />
        </list>
    </property>
</bean>
<bean name="gestionarRolesPermisos"
    class="vnz.sigep.administracion.seguridad.util.urlagrupation.URLAgrupation">
    <property name="name">
        <value>Roles y Accesos</value>
    </property>
    <property name="pattern">
        <value></value>
    </property>
    <property name="mappings">
        <props>
            <prop
                key="/datosRoleSeleccionado.htm, Obtener Datos del Role Seleccionado">
                roleController
            </prop>
            <prop
                key="/createRoleIndex.htm, pagina de inicio para crear Roles">
                roleController
            </prop>
            <prop
                key="/rolesTabla.htm, Datos de la tabla de roles">
                roleTablaController
            </prop>
            <prop
                key="/persistirDatosRoles.htm, Datos de la tabla de roles">
                roleController
            </prop>
        </props>
     </property>
</bean>
```

Figura 14: Nueva forma de declarar las URL y los controladores por parte de los programadores de interfaz de usuario.

UrlMappingRegister: Esta clase tiene la función de identificar y registrar todas las URL de un subsistema, para ello es necesario crear una instancia de esta clase en cada subsistema, esta instancia se define en el contexto de negocio del subsistema y se le inyecta la propiedad *agrupation* que responde a la instancia de la clase *AbstractAgrupation* raíz del subsistema.

```
<bean class="vnz.sigep.administracion.seguridad.util.UrlMappingRegister">
    <property name="agrupation">
        <bean name="administracionModuleAgrupation"
            class="vnz.sigep.administracion.seguridad.util.urlagrupation.MappingsAgrupation">
            <property name="name">
                <value>Administración</value>
            </property>
            <property name="pattern">
                <value>/administracion</value>
            </property>
        </bean>
    </property>
</bean>
```

Figura 15: Declaración del bean del tipo UrlMappingRegister en el contexto de negocio.

Esta clase implementa la interfaz *org.springframework.beans.factory.InitializingBean* que le permite implementar el método *afterPropertiesSet()* que es ejecutado una vez llenados todos los atributos de esta clase. En este método se invoca al método *constructSecurityAbstractAgrupation()* que se encarga de construir el objeto *SecurityAbstractAgrupation* del subsistema. Es insertado en la colección estática *URLSecurityAgrupations*.

> *URLSecurityAgrupations*: Atributo estático que contiene una colección de *SecurityAbstractAgrupation*, cada objeto de esta colección responde a la agrupación de URL de cada subsistema. Una vez concluida la carga de la aplicación, todos los subsistemas, módulos, agrupaciones por casos de uso o funcionalidades quedarán guardadas en este atributo.

Identificar métodos requeridos por la seguridad

La Seguridad del SIGEP necesita tener el control de todos los métodos de negocio que son exportados como servicios o son invocados desde la web, para ello se creó una estructura de clases que encapsula los datos de estos recursos, pero, ¿cómo obtener de manera automática todos estos métodos de negocio y todos los datos adicionales de los cuales también se necesita tener control?

Es importante conocer cuáles son los métodos que necesitan ser identificados por la seguridad:

> *Métodos de Negocio exportados como servicio*: Todos los métodos que son declarados en las interfaces de servicio.

Métodos de Negocio invocados desde la web: Métodos que se encuentran la Fachada de cada módulo y que además entre sus anotaciones exista @JSONExportedMethod.

Para tener control de estos métodos y la información que se necesita conocer de ellos se define la clase *MethodMappingRegister*.

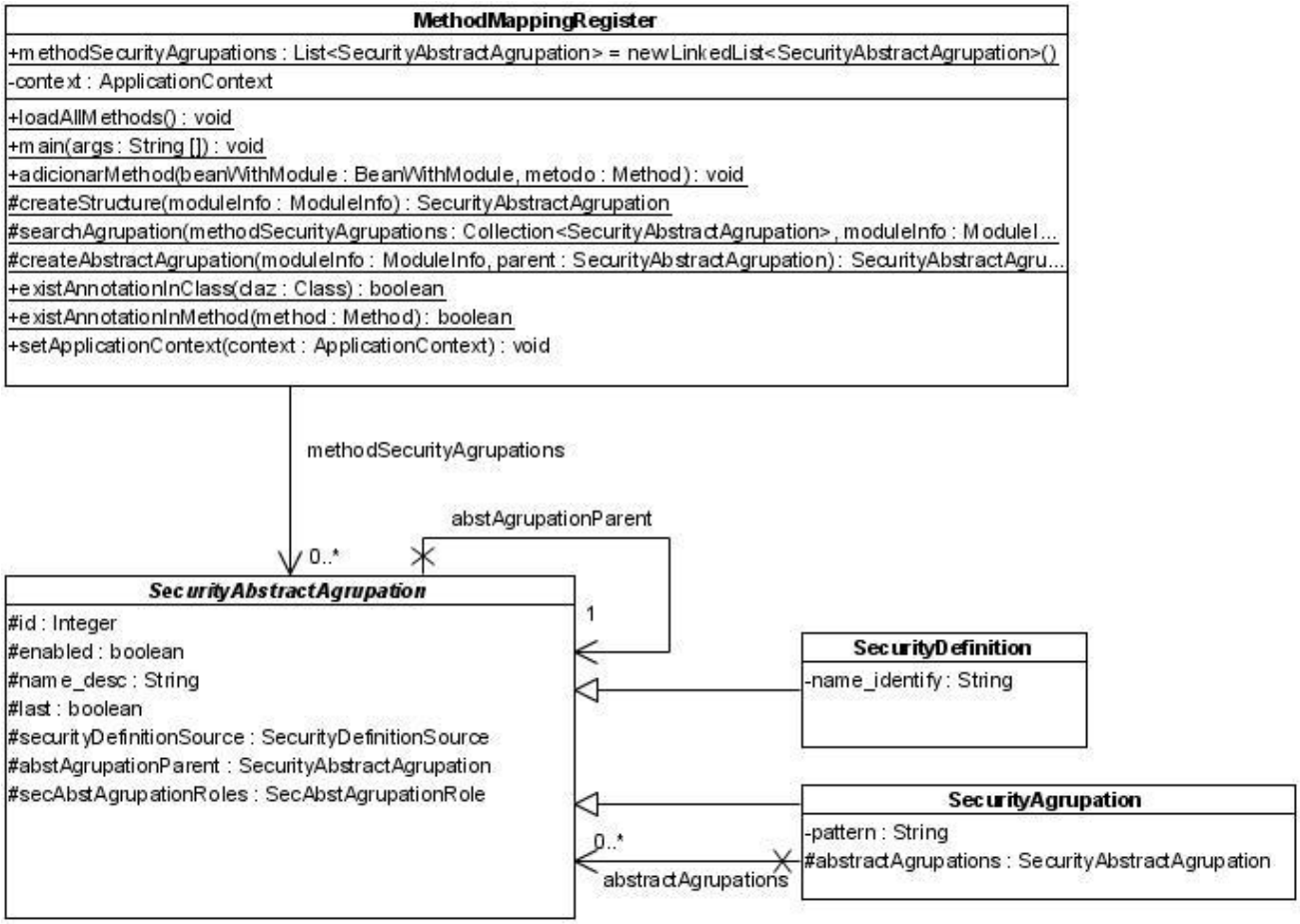


Figura 16: DC donde se muestra la relación entre las clases MethodMappingRegister y SecurityAbstractAgrupation.

MethodMappingRegister: Tiene la función de identificar y registrar todos los métodos de negocio que requieren ser controlados por la seguridad del SIGEP. Para ello implementa la interfaz *org.springframework.context.ApplicationContextAware.*

context: Este atributo es del tipo de datos *org.springframework.context.ApplicationContext*. A través de él se puede acceder a todos los contextos del SIGEP.

methodSecurityAgrupations: Atributo estático que contiene una colección de SecurityAbstractAgrupation, cada objeto de esta colección responde a la agrupación de los métodos identificados de cada subsistema.

El proceso de identificación de los métodos requeridos por la seguridad comienza cuando un evento lanzado por el sistema notifica que el negocio de la aplicación ha sido cargado y posteriormente se invoca al método *loadAllMethod()* definido en esta clase *MethodMappingRegister.* Él es el encargado de recorrer todos los contextos que existen en el SIGEP y buscar los métodos que requieren ser identificados. Una vez encontrados se crean los objetos del tipo *SecurityAbstractAgrupation* que responden a cada subsistema y serán insertados en la colección *methodSecurityAgrupations*.

Gestionar permisos

La Seguridad del SIGEP necesita tener control de los recursos que existen en la aplicación con los que puede interactuar el cliente, de modo que sólo pueden acceder a ellos el personal que este autorizado a hacerlo. Un permiso es la unión de un rol del sistema con un recurso, sólo podrán acceder a un recurso los usuarios que posean los roles que están asociados al mismo.

Para poder definir los permisos de la aplicación se necesita conocer los roles que existen en el sistema, además de los recursos que deben ser asegurados, como se mostró, se han identificado un conjunto de clases que definen a los roles y los recursos del sistema, ahora sólo queda definir los permisos que van a existir en la aplicación.

Diagrama de clases donde se muestra la relación que existe entre los recursos de la aplicación y los roles del sistema. Se muestran los permisos que se necesitan administrar y son representados por la clase *SecAbstractAgrupationRole*.

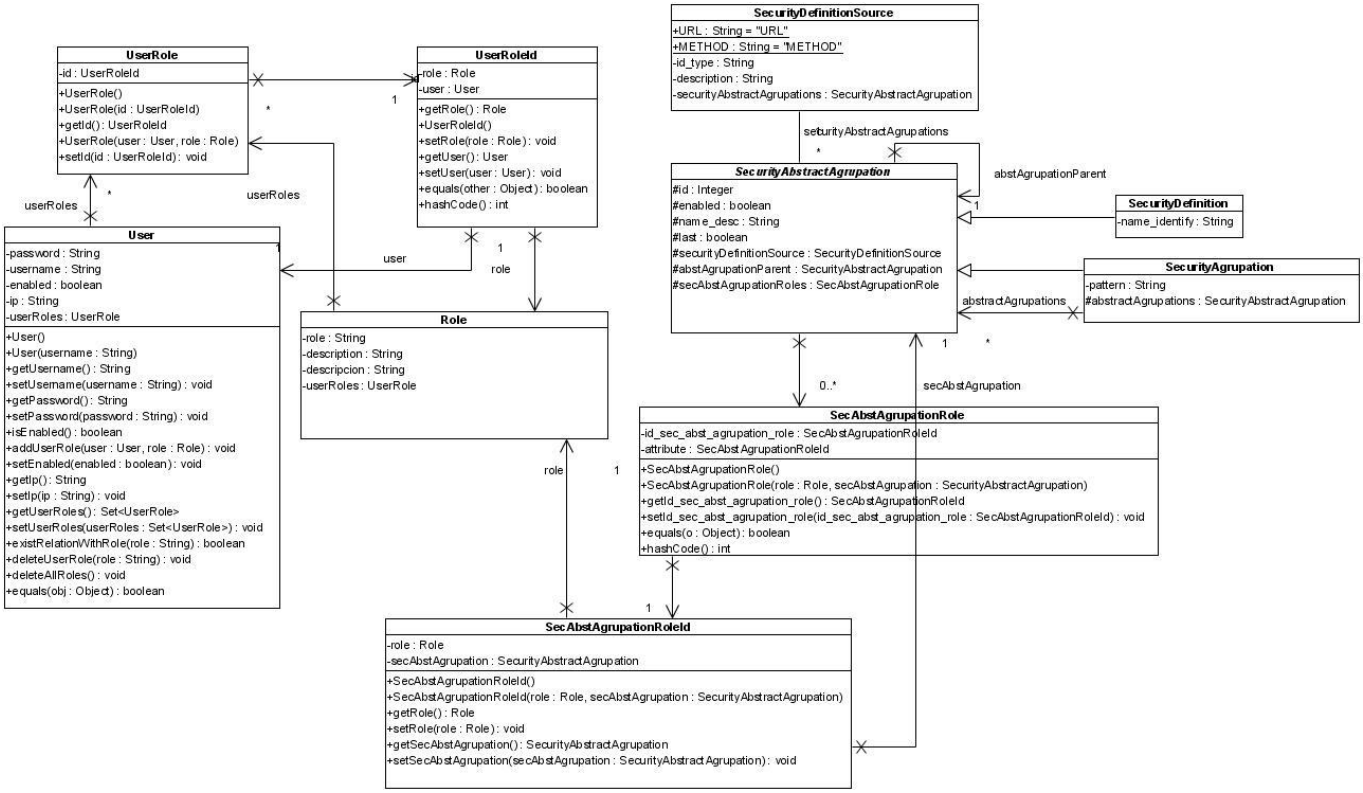


Figura 17: DC donde se muestra la relación entre las clases que modelan los recursos y los roles del sistema.

Como muestra el diagrama de clases, un permiso es la relación que existe entre un rol del sistema y un recurso de la aplicación. Si un recurso no tiene asignado un rol entonces nadie podrá acceder a este recurso, pero sin embargo, si este recurso no ha sido identificado por la seguridad y esta no tiene conocimientos del recurso entonces cualquiera tendrá acceso al él.

La primera vez que se inicia la aplicación no existe ningún permiso definido en el sistema, además de que la seguridad aún no conoce las URL ni los métodos que existen en la aplicación y se tiene acceso a todo en el sistema. El proceso de definir los permisos en la aplicación puede consumir un tiempo considerable y mientras no se hayan definido los permisos existe un hueco en la seguridad del SIGEP. Se hace necesario que, una vez que los permisos sean identificados por la administración, sean salvados en una estructura sólida como una BD y que sean cargados por la seguridad del SIGEP. Definir además un rol que tenga acceso a la administración de estas configuraciones de seguridad y un usuario que pueda administrar estas configuraciones para así poder definir los permisos.

Para ello se creó un conjunto de clases que se encargan de salvar las nuevos recursos que existen en el sistema una vez cargado el negocio de la aplicación.

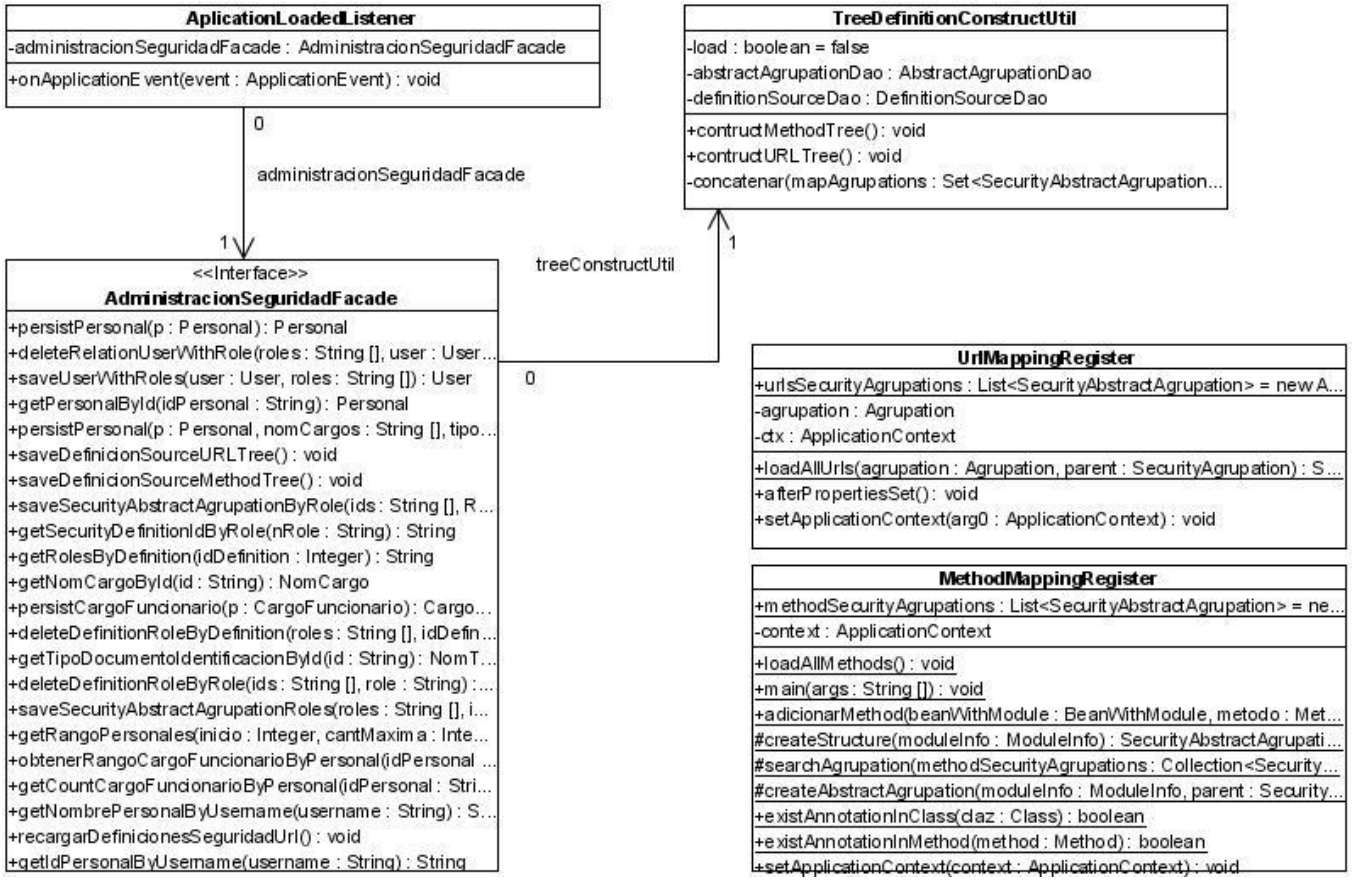


Figura 18: DC que muestra la relación entre las clases encargadas de salvar los recursos de la aplicación.

AplicationLoadedListener: Clase que extiende *de org.springframework.context.ApplicationListener* y se encarga de capturar el evento que es lanzado una vez cargado el negocio de la aplicación. Cuando el evento es capturado se ejecutan los métodos *saveDefinitionSourceURLTree()* y *saveDefinitionSourceMethodTree()* de la interfaz *AdministracionSeguridadFacade* .

TreeDefinitionContructUtils: Clase creada como utilidad, con el objetivo de cargar los valores de los recursos ya definidos en la BD y concatenar estos valores con los de los recursos identificados por la administración que se encuentran en las colecciones estáticas *URLSecurityAgrupation* y *methodSecurityAgrupation* de las clases *URLMappingRegister* y *MethodMappingRegister* respectivamente. Si existen nuevos valores de recursos que no están persistidos en la BD entonces estos recursos serán persistidos.

Diagrama de secuencia donde se muestran los mensajes enviados entre clases que hacen posible que se puedan obtener los recursos requeridos por la seguridad para poder gestionar los permisos a los mismos

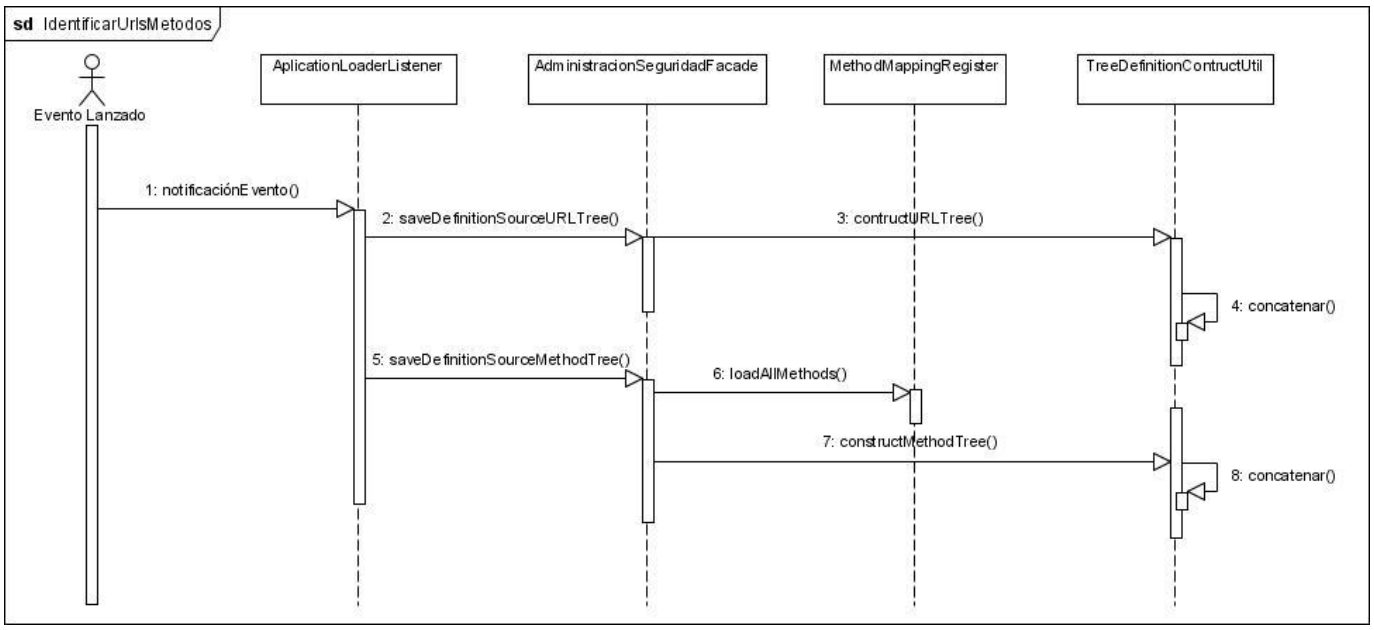


Figura 19: Diagrama de secuencia donde se muestran mensajes enviados entre las clases representadas.

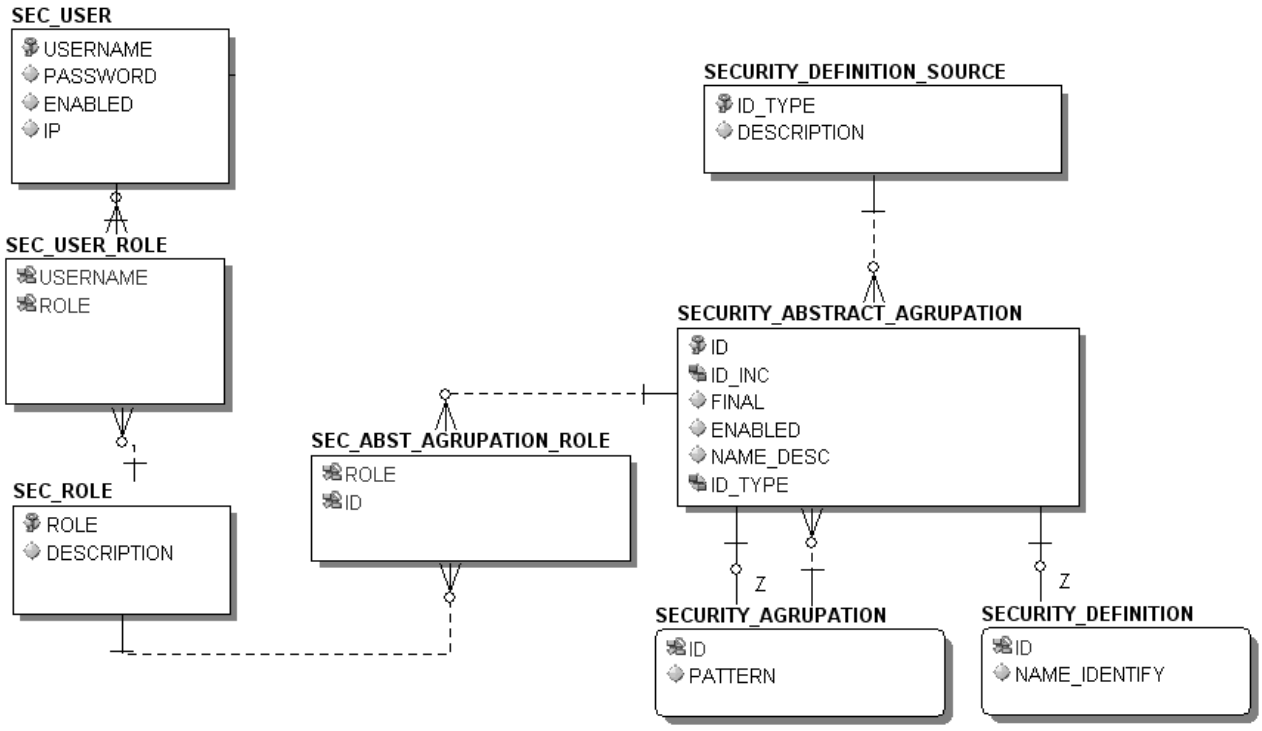


Figura 20: MD donde se representan los permisos.

Gestionar permisos a URL

Como vimos anteriormente gestionar permisos implica crear la relación que existe entre un recurso y un rol. Ya el sistema tiene el control de las URL que existen en la aplicación y de los roles del sistema, así como las clases que modelan la relación que existe entre ellos, sólo queda crear la interfaz visual que le permita al Administrador de la Seguridad relacionar las URL y los roles del sistema.

Diagrama de clases de la web, se muestran los principales componentes de la capa de presentación y las relaciones que existen entre ellos para lograr gestionar los permisos a las URL del sistema.

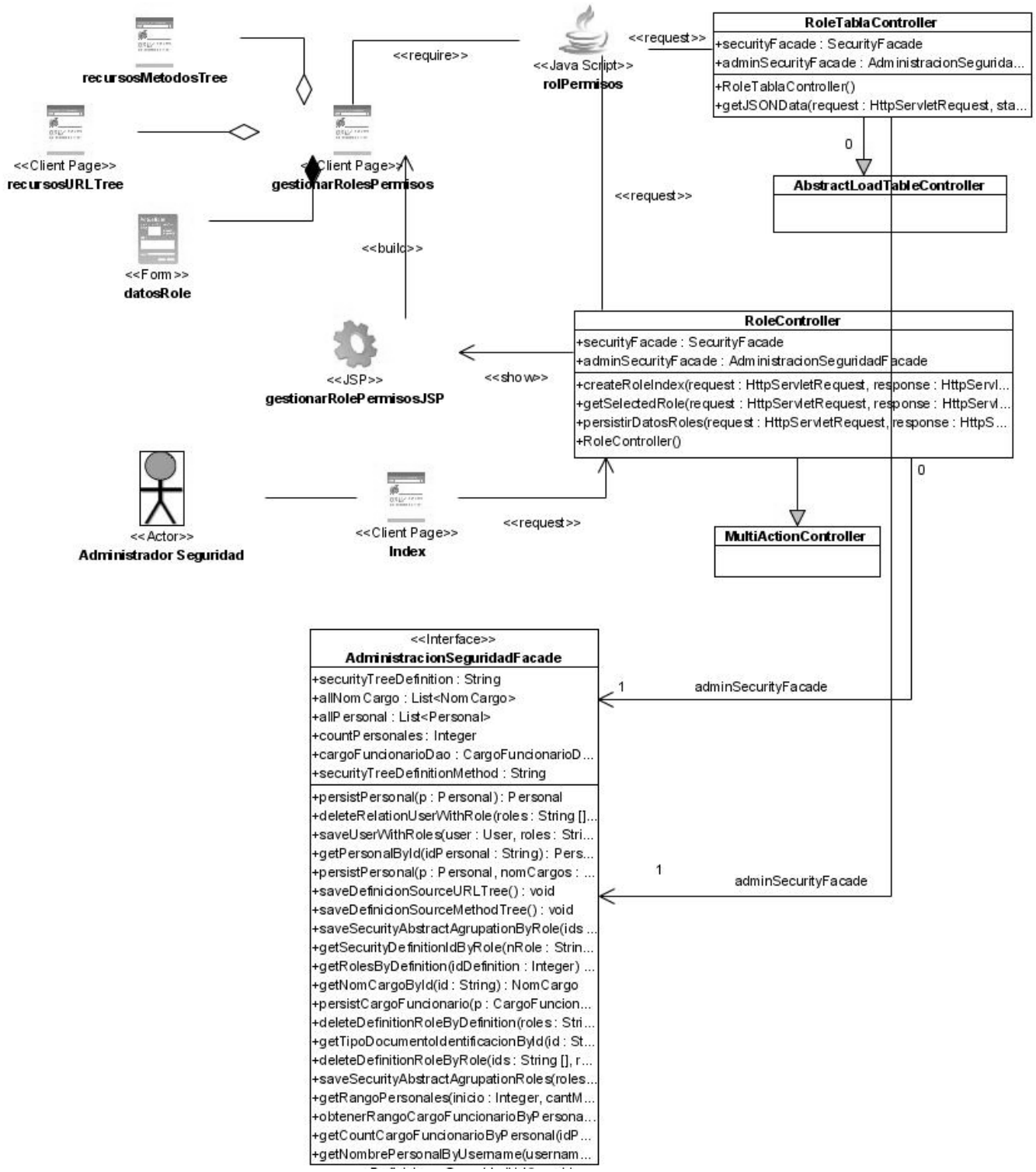


Figura 21: DCW referente a los casos de uso Gestionar Roles, Gestionar Permisos a URL y Gestionar Permisos a Métodos

Como se muestra en el diagrama la página cliente gestionarRolesPermisos incluye dos páginas cliente y un formulario:

Formulario datosRole: Se encarga de recoger los datos referentes a un rol, estos datos son el nombre del rol y la descripción.

Página cliente recursosURLTree: Representa el árbol de todas las URL del sistema, en el cual se identifican las URL a las cuales un rol tiene acceso.

Página cliente recursosMetodosTree: Representa el árbol de todos los métodos requeridos por la seguridad, en el cual se identifican los métodos a los cuales un rol tiene acceso.

Gestionar permisos a Métodos

El sistema ya posee el control de todos los métodos que necesitan ser controlados por la seguridad y de los roles del sistema, así como las clases que modelan la relación que existe entre ellos, sólo queda crear la interfaz visual que le permita al Administrador de la Seguridad relacionar estos métodos y los roles del sistema.

Para gestionar los permisos a los métodos del sistema el Administrador de Seguridad debe acceder a la página cliente *gestionarRolesPermisos* en la cual podrá especificar los métodos a los que tiene acceso un rol especificado, esto lo realiza en el árbol de métodos representado en la pagina cliente *recursosMetodosTree* (Figura 21).

De esta forma la administración de la seguridad gestiona los permisos a los métodos de negocio que requieren seguridad.

CONCLUSIONES

Se identificaron las configuraciones y definiciones de seguridad que requieren ser administradas.

Se diseñaron e implementaron mecanismos que permiten obtener automáticamente los elementos de la aplicación que requieren ser asegurados.

Se diseñaron mecanismos que permiten gestionar las configuraciones y definiciones de seguridad mediante interfaces visuales y aplicar los cambios realizados en las configuraciones y definiciones de seguridad sin necesidad de reiniciar la aplicación.

Se creó un modelo de datos para almacenar los valores de las configuraciones y definiciones de seguridad que requieren ser administradas.

Se implementaron los mecanismos diseñados para administrar las configuraciones y definiciones de seguridad, se programó el acceso a los datos, lógica de negocio y la interfaz visual.

De esta forma se hace más fácil, confiable y eficiente el proceso de definir la seguridad de un sistema, Se cambian los conceptos de administración de la seguridad, independizando el proceso de desarrollo del software del de definición de la seguridad de un sistema.

RECOMENDACIONES

Se recomienda que se analice el uso de este módulo de administración de las configuraciones y definiciones de seguridad en otros proyectos de la universidad que presenten arquitecturas similares a la del SIGEP.

BIBLIOGRAFÍA

Alex, B. (2006). Acegi Reference Documentation.

BAUER, C. and G. KING (2005). Hibernate in Action.

Craig Walls, R. B. (2008). Spring In Action Second Edition.

Larman, C. UML y Patrones.

Orizondo, A. C. A. (2006). Proyecto Técnico de Asesoría Especializada, Colaboración Médica Odontológica, Comunicación Institucional y Solución Tecnológica para apoyar la modernización del Sistema Penitenciario de la República Bolivariana de Venezuela. Venezuela.

Pimentel, L. A. and J. E. Martínez (2008). Documento de Arquitectura de Software. Cuba.

Pimentel, L. A. and I. P. Rivero (2007). ArBaWeb: ARQUITECTURA BASE SOBRE LA WEB. La Habana, Universidad de las Ciencias Informáticas.

SETH LADD, D. D., STEVEN DEVIJVER, COLIN YATES (2006). Expert Spring MVC and Web Flow.

Wiley, J. (2005). Professional Java Development with the Spring Framework.

Ciberaula. (2006). "Java (J2EE)." Retrieved 15/03/2007, 2007, from http://www.ciberaula.com/curso/java2/que_es/.

Mateos, I. M. (2005). Evaluación del Estado del Arte de J2EE y de los Servidores de Aplicaciones JBoss y Geronimo. Facultad de Ingeniería., Universidad de Deusto.: 119.

GLOSARIO DE TÉRMINOS

Subsistema: Un subsistema se refiere a un conjunto de módulos que por razones de similitud o de perseguir objetivos comunes, son agrupados.

Módulo: Encapsula un conjunto de funciones que debe realizar el sistema, las cuales son agrupadas por tener características muy similares y se definen en la etapa de diseño.

Interface: Es un conjunto de constantes y métodos a los que no se les da implementación, también se puede ver como una clase que solo cuenta con métodos abstractos.

Clase: Es una abstracción sobre un conjunto de objetos que tienen en común estructura y comportamiento.

Atributos: Son las características o propiedades que definen a un objeto

Método: Son operaciones que pueden modificar el estado de un objeto o simplemente obtener datos sobre el mismo.

Objeto: Es el resultado de la instanciación de una clase.

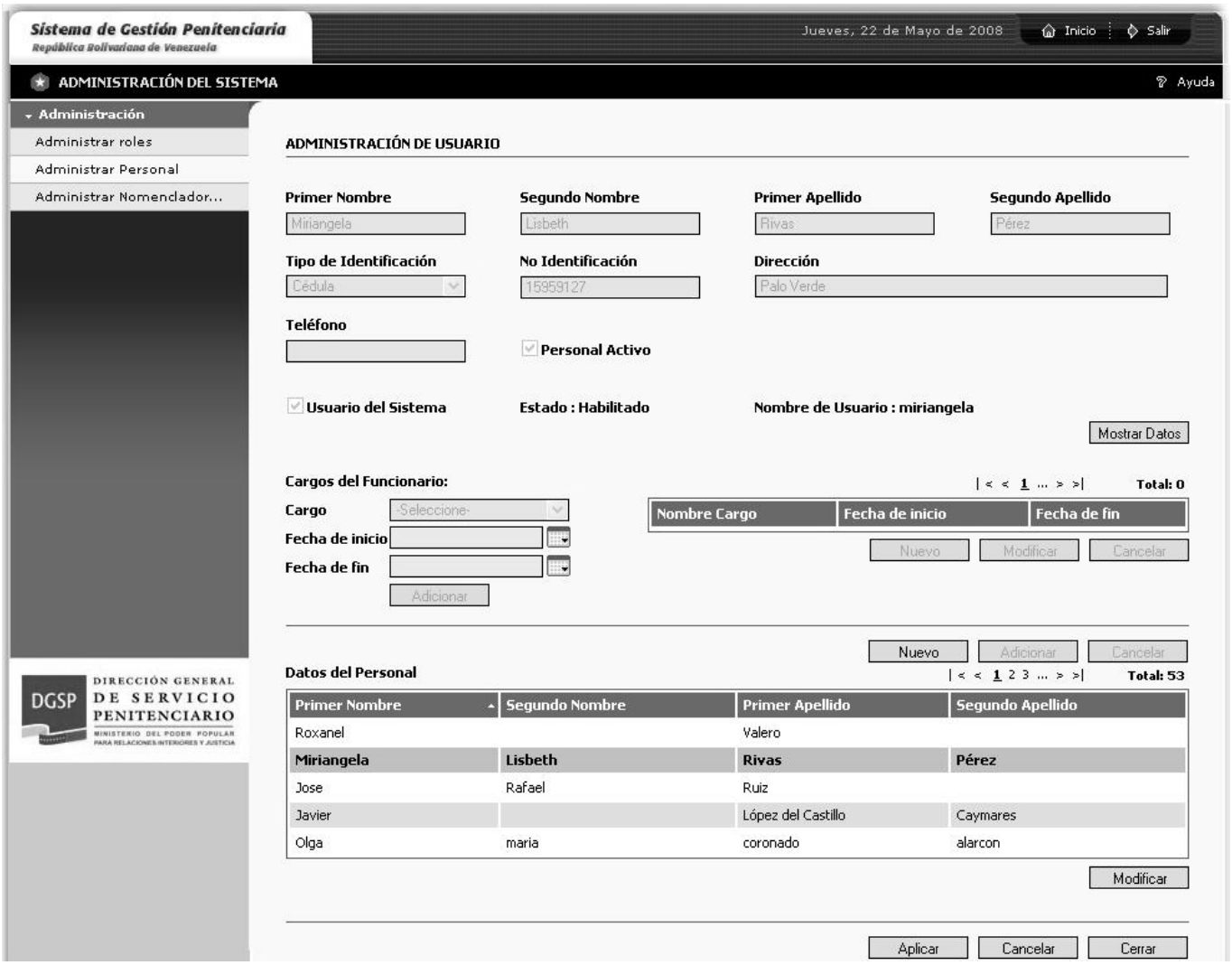
Caso de uso: Fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores.

Framework: Conjunto de APIs y herramientas destinadas a la construcción de un determinado tipo de aplicaciones de manera generalista.

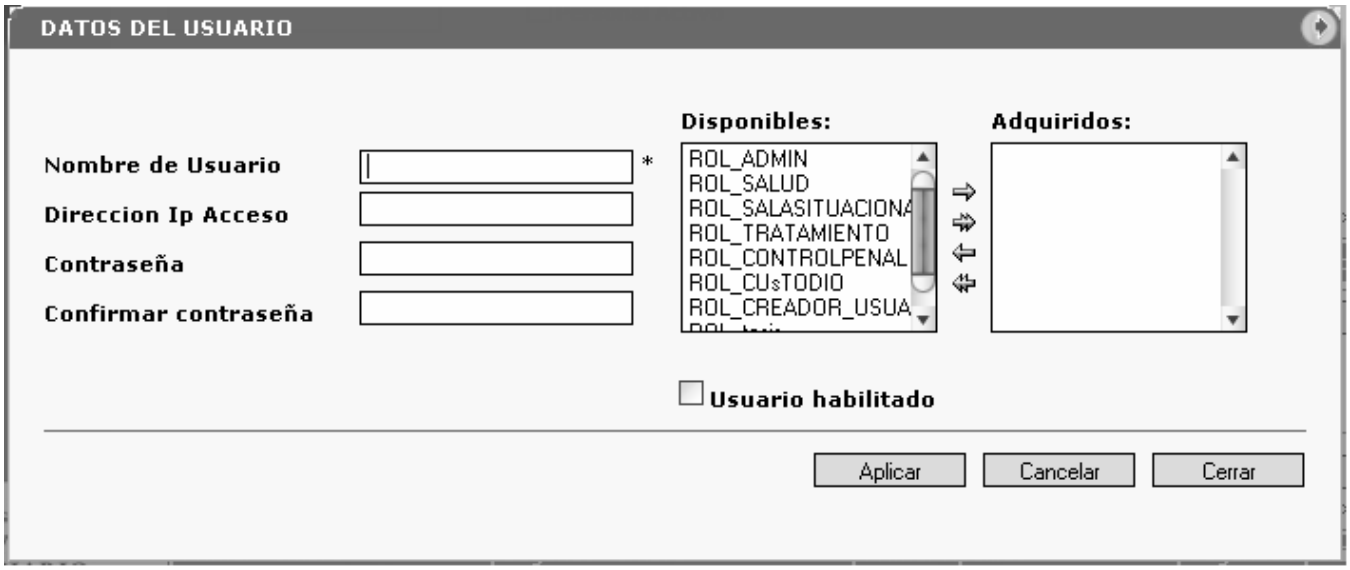
Servlet: Es una clase Java que ofrece funciones suplementarias al servidor.

Software: Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.

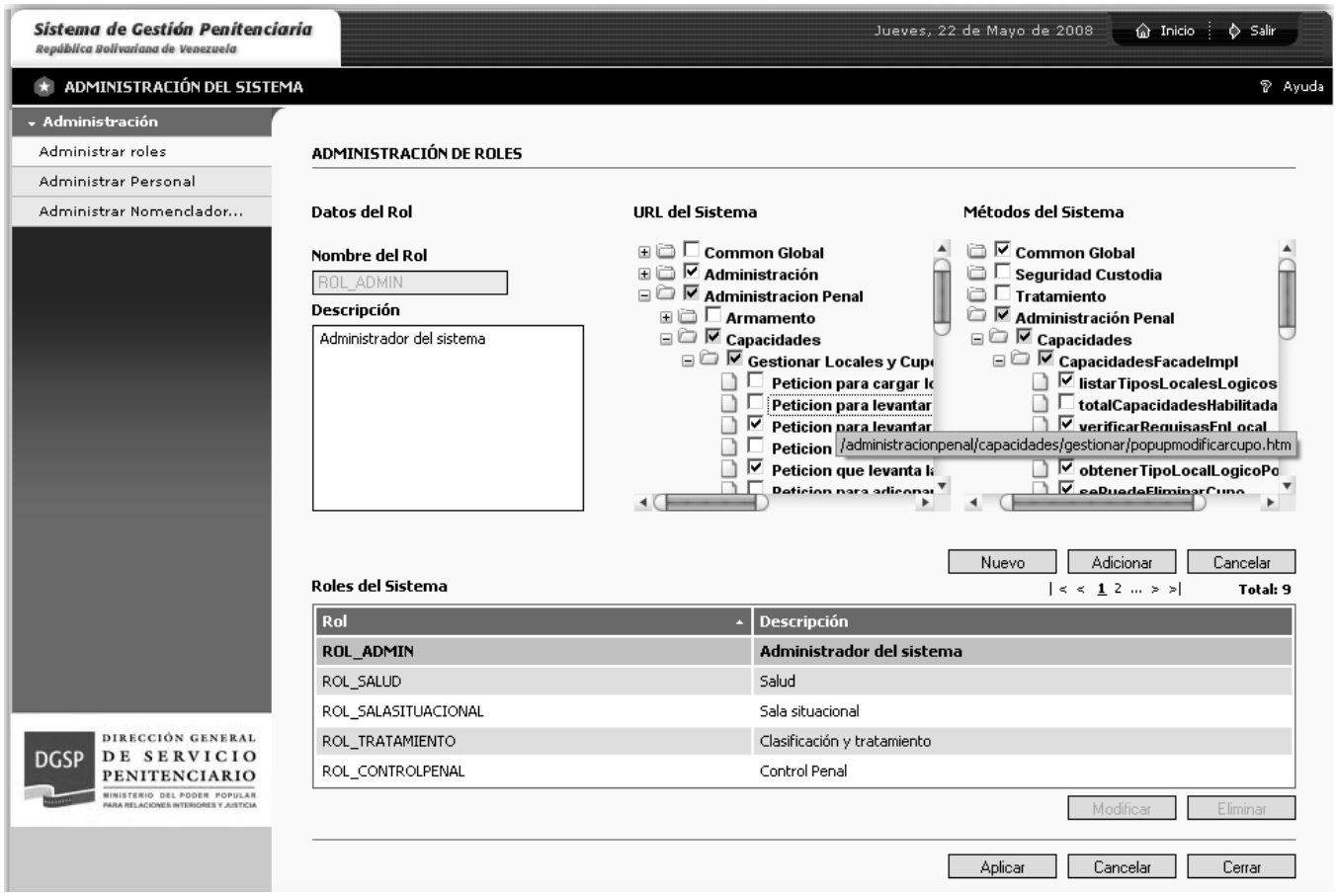
ANEXOS



Anexo 1: Interfaz encargada de gestionar los datos del personal del penal al cual se le asigna un usuario del sistema.



Anexo 2: Interfaz visual encargada de gestionar los usuarios del sistema.



Anexo 3: Interfaz visual encargada de gestionar los Roles del sistema, los permisos a URL y a Métodos.

1