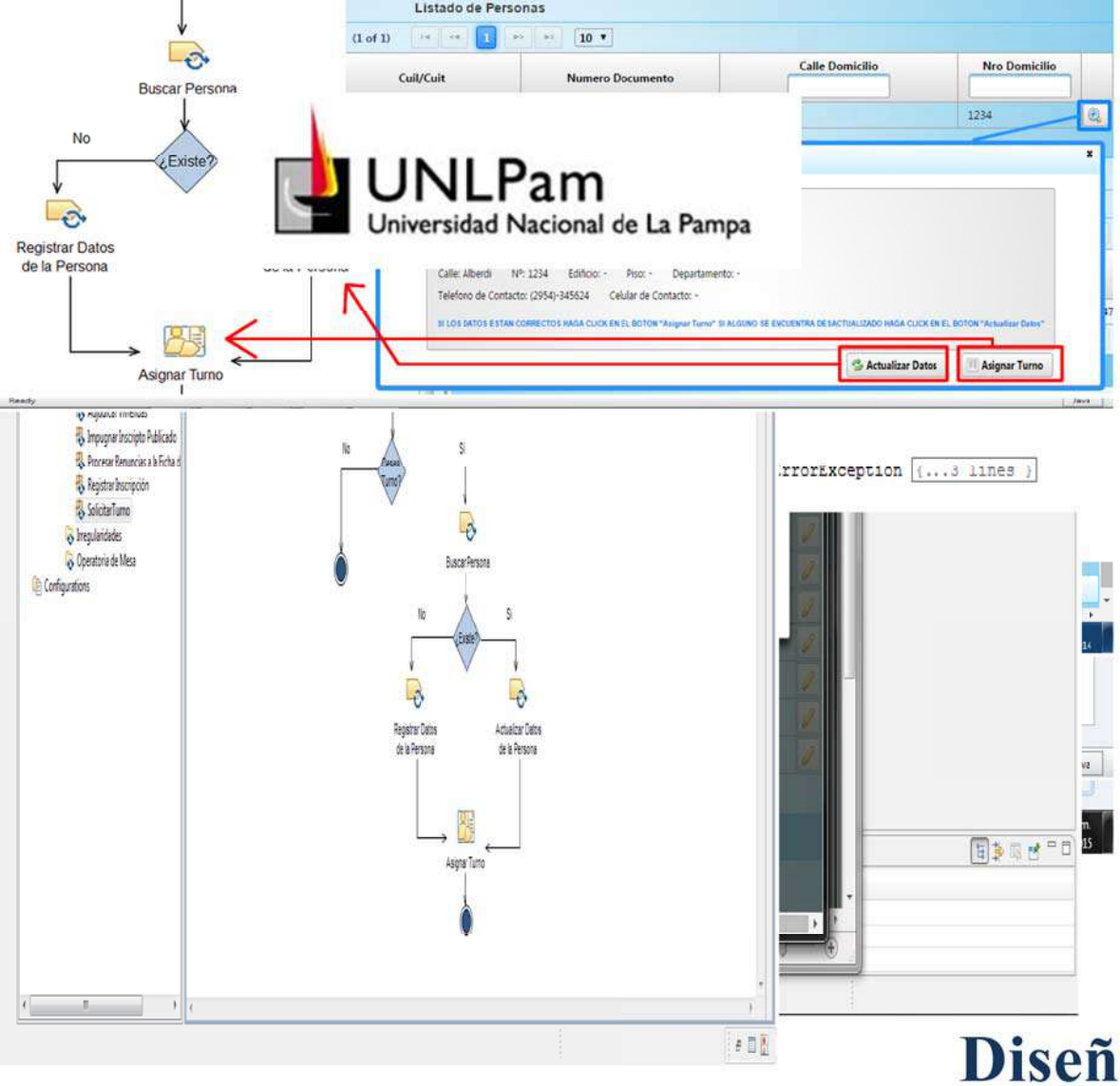

Diseño y Desarrollo de un Sistema Integrado de Gestión de Seguridad, Inscripciones y Adjudicación en el Instituto Provincial Autárquico de Vivienda (IPAV)

Carrera: Ingeniería en Sistemas

Alumno: Martínez, Alejandro

D.N.I.: 32.159.193

Tutor: Dr. Diván, Mario José

Prefacio

Este trabajo final se presenta como parte de los requisitos para lograr el grado Académico de Ingeniería en Sistemas de la Universidad Nacional de La Pampa. El mismo contiene un informe de la experiencia de una práctica en empresa realizada en el marco de una pasantía rentada en el Instituto Provincial Autárquico de la Vivienda (IPAV).

Si bien la práctica se realizó en el período comprendido entre el 26/11/2014 y el 06/03/2015, para una mayor comprensión del informe se incluyen la descripción de algunas actividades realizadas en el período de la pasantía, desde el 1/10/2013 al 06/03/2015.

Cabe destacar, que los resultados obtenidos son mostrados parcialmente a través de ejemplos ilustrativos, para resguardar la privacidad de la información del organismo IPAV.

La práctica en empresa fue realizada bajo la tutoría del Dr. Diván, Mario José por parte de la Facultad de Ingeniería y el CPN Roberto Vassia por parte del IPAV.

Agradecimientos

A todo el personal docente de la Universidad Nacional de La Pampa, por su tiempo compartido y por impulsar el desarrollo de mi formación profesional.

A todos los compañeros y amigos con los que he compartido horas de estudio y mates, dentro y fuera de las aulas.

A mis padres, que me brindaron su apoyo constante e incondicional en toda mi vida; en especial a mi madre que sin su ayuda hubiera sido difícil culminar mi carrera.

A mi esposa Vanesa, que estuvo dándome fuerzas, ayudándome y acompañándome en todo momento, y a mis hijos Micaela y Mateo que los amo con todo mi corazón.

Al Personal del IPAV, que me hizo sentir parte de ellos. A mis compañeros de trabajo, con quienes he tenido momentos gratos.

Y, en especial, al Dr. Mario Diván, por brindarme su experiencia y guiarme en el desarrollo de la tesis, aceptando ser mi tutor.

Índice

Lista de Figuras

Resumen

La Gerencia de Planificación y Adjudicación del IPAV tiene como funciones principales atender todo lo relacionado con la información de planes y programas de viviendas; supervisar la labor de adjudicación de viviendas, gestionar la cobranza y el seguimiento de los barrios construidos, entre otras.

Hasta el año 2013, dicha dependencia poseía un sistema de gestión que parcialmente y en forma aislada, brindaba apoyo a la administración de algunas de las funciones antes mencionadas. Si bien la Gerencia de Planificación y Adjudicación tiene los procesos formalizados, al momento de comenzar la pasantía, aún restaban automatizar mucho de ellos. Además, los sistemas no eran interoperables, no permitiendo la integración entre los distintos subsistemas, lo que hacía más difícil el control e intercambio de datos entre éstos de manera automatizada y consistente.

En este trabajo final de práctica en empresa se colaboró en el Diseño, Desarrollo e Implementación de un nuevo sistema para la gestión de la gerencia de Planificación y Adjudicación en el IPAV. Dicho sistema está orientado al servicio, tiene en cuenta un nuevo módulo troncal de seguridad y está totalmente integrado, conformando una arquitectura interoperable, segura, consistente y extensible.

El sistema desarrollado (que ya se encuentra en funcionamiento desde diciembre del 2014 en el IPAV) brinda la información para la gestión y monitoreo de inscripciones, de irregularidades, de impugnaciones, del sorteo y de las pre-adjudicaciones. Sirve de apoyo a la gestión desde la registración de las inscripciones hasta la emisión de los listados pertinentes vinculados con las adjudicaciones. Por otra parte, la célula de desarrollo actualmente se encuentra construyendo el módulo contable, que tendrá como principal funcionalidad la gestión de cobranza de las cuotas de viviendas, el inventario, las operatorias de asistencia financiera, las certificaciones de obras (tanto nexos como la construcción de viviendas), y será integrado al sistema en forma total.

Como resultado de este desarrollo no solo se obtuvo un producto de software, sino que significó un enriquecedor estudio e investigación, lo que se materializó en

experiencia profesional y conocimientos adquiridos que fueron plasmados en dos trabajos científicos que se publicaron en conferencias nacionales e internacionales [1, 2].

Estructura del Trabajo Final

Este informe se organiza de la siguiente forma:

En el capítulo 1 se introducen algunos conceptos básicos de las principales tecnologías utilizadas en el trabajo, para una mejor comprensión del informe. Luego en el capítulo 2 se describe el Equipo de Trabajo y las Herramientas utilizadas en el desarrollo de la práctica. Ya en el capítulo 3 se da comienzo al desarrollo del sistema con el relevamiento. En el capítulo 4 se detalla la etapa de Análisis y Diseño del sistema. Posteriormente en el capítulo 5 se explica cómo se implementaron los servicios web (WS) utilizados. El capítulo 6 especifica el testeo del sistema. En el capítulo 7 se detallan las tareas realizadas por el pasante. Por último, en el capítulo 8, la conclusión del informe.

1. Capítulo 1: Marco Conceptual

1.1 Introducción

Las actuales características de dinamismo y variabilidad de la industria software han precisado replantear los cimientos sobre los que se sustenta el desarrollo software convencional. Un reciente estudio realizado por Boehm en [3], sobre la tendencia en ingeniería del software, indica que el mercado actual está caracterizado por el rápido desarrollo de aplicaciones y la reducción de la vida de los productos. En este entorno, que se presenta inestable, la ventaja competitiva se encuentra en aumentar la productividad y satisfacer los cambios requeridos por el cliente en el menor tiempo posible para proporcionar un mayor valor al negocio. En esta dirección, las metodologías ágiles han irrumpido con fuerza en los últimos años como un intento de apoyo a esta demanda, y son muchas las organizaciones punteras con creciente interés en las mismas. Las metodologías ágiles apuntan a mejorar el rendimiento de los equipos de desarrolladores sin dejar de lado la calidad del producto, de una forma más “natural” que en los procesos tradicionales. Esto significa ciclos cortos, desarrollo interactivo, propensión al cambio, documentación dinámica, constante interacción y valorada comunicación entre todos los involucrados en el proyecto.

Uno de los primeros aspectos que se desarrolló en el marco del nuevo sistema de gestión, fue la implementación de un sistema de seguridad basado en la arquitectura orientada al servicio. La arquitectura orientada al servicio brinda aspectos de interoperabilidad y facilita la integración en aplicaciones distribuidas.

Debido a que el desarrollo realizado durante la práctica en la empresa IPAV se cumplimentó siguiendo una metodología ágil, se usó una arquitectura basada en servicios y para una mejor comprensión de este informe, es que en este capítulo se exponen los conceptos básicos de metodologías agiles, y las herramientas de apoyo usadas para tal fin.

1.2 Metodología Ágil

Las metodologías ágiles para la elaboración de software están basadas en el desarrollo iterativo e incremental del mismo, donde los requisitos y soluciones evolucionan mediante la colaboración de grupos auto-organizados y multidisciplinarios.

Las metodologías agiles dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Al estar usando procesos ágiles para la gestión de proyectos, se perciben distintos beneficios para el cliente:

- Flexibilidad en el proceso y las definiciones de los productos: Permite que el equipo de desarrollo se adapte a los cambios y se beneficie de ellos en favor del cliente.
- Realimentación continua con el cliente: De forma temprana el cliente recibe entregables, lo que permite ver los constantes avances, logrando así aportar lo necesario para que el equipo vaya construyendo en la dirección correcta. Inmediatamente, se reducen de forma drástica los errores y la posibilidad de costosas correcciones, respondiendo a los cambios en los requisitos de forma rápida y eficaz.
- Interacción constante: Importante a la hora de dar tranquilidad al cliente sobre los avances del producto que recibirá (debido a que el producto se va analizando a medida que avanza).
- Calidad mejorada: Esto significa que las prácticas de desarrollo ágil y sus constantes interacciones, proporcionan la funcionalidad suficiente como para satisfacer las expectativas del cliente con una alta calidad. La clave se encuentra en la continuidad, es decir, la calidad es integral al proceso, y no añadida.
- Proyectos no definidos claramente: Esto apunta a que los requisitos del cliente se van clarificando a medida que el proyecto va avanzando, lo que permite la fácil adaptación del desarrollo para cumplir los nuevos desafíos.

- Interacción y comunicación: La interacción entre los diferentes diseñadores y participantes es clave, es especialmente propicia para entornos orientados al trabajo en equipo.

1.3 SCRUM

Para el desarrollo del sistema del IPAV se utilizó la metodóloga SCRUM [4]. Este método define un conjunto de prácticas y roles (ver Figura 1) que se utilizaran para el desarrollo del proyecto. Los roles principales en Scrum son el *ScrumMaster* (Director de Proyecto), que hace un seguimiento de los procesos, facilitando la ampliación de la Metodología SCRUM y gestiona los cambios en el producto; el *ProductOwner* (Clientes –Propietario del Producto-), que representa a los *stakeholders* (interesados externos o internos); y el *Team* (Equipo de Trabajo), que incluye a los desarrolladores.

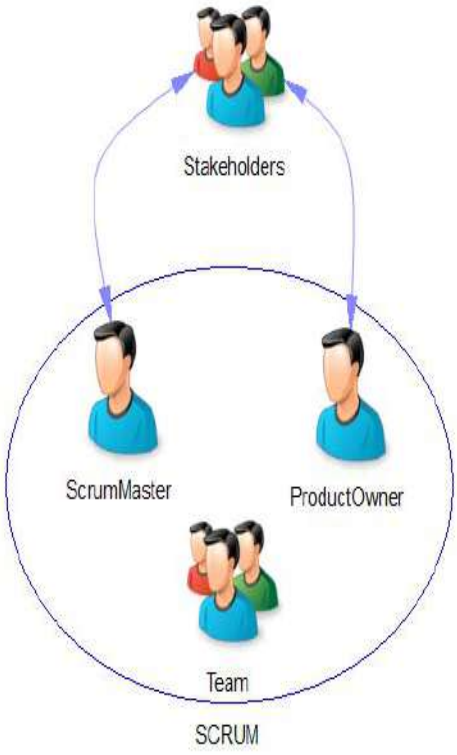

Figura : Relación entre los actores de SCRUM.

Durante cada sprint (Iteración), un periodo entre una y cuatro semanas (éste periodo es definida por el equipo), el equipo crea un incremento de software

potencialmente entregable (utilizable). El conjunto de características que forma parte de cada sprint viene del *Product Backlog*, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar. Los elementos del *Product Backlog*, que forman parte del sprint, se determinan durante la reunión de *Sprint Planning*. Durante esta reunión, el *ProductOwner* identifica los elementos del *Product Backlog* que quiere ver completados y los hace del conocimiento del equipo. Entonces, el equipo determina la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente sprint. Durante el sprint, nadie puede cambiar el *Sprint Backlog*, lo que significa que los requisitos están congelados durante el sprint.

Scrum permite la creación de equipos auto-organizados impulsando la co-localización de todos los miembros del equipo y la comunicación verbal entre todos los miembros y disciplinas involucrados en el proyecto.

Un principio clave de Scrum es el reconocimiento de que durante un proyecto los clientes pueden cambiar de idea sobre lo que quieren y necesitan (a menudo llamado *requirements churn*), y que los desafíos impredecibles no pueden ser fácilmente enfrentados de una forma predictiva y planificada. Por lo tanto, Scrum adopta una aproximación pragmática, aceptando que el problema no puede ser completamente entendido o definido, centrándose en maximizar la capacidad del equipo de entregar rápidamente y responder a requisitos emergentes.

Las características más marcadas que se logran notar en Scrum son: gestión regular de las expectativas del cliente, resultados anticipados, flexibilidad y adaptación, retorno de inversión, mitigación de riesgos, productividad y calidad, alineamiento entre cliente y equipo, y por último, equipo motivado. Cada uno de estos puntos mencionados hace que Scrum sea utilizado de manera regular en un conjunto de buenas prácticas para el trabajo en equipo y de esa manera obtener resultados posibles.

Una de las mayores ventajas de Scrum es que es muy fácil de aprender y requiere muy poco esfuerzo para comenzar a utilizarlo.

1.4 SPEM

Los procesos en el desarrollo de software pueden ser vistos como productos, ya que están constantemente cambiando y evolucionando. También, deben ser administrados y configurados para adaptarlos a las organizaciones y a las nuevas necesidades del entorno, agregando de esta forma la necesidad de un estándar unificado en esta área, esto debido a que cada una de estas técnicas y procesos definió sus propios estándares y terminologías usando incluso diferentes significados para la misma palabra.

Para especificar las actividades propuestas por un proceso de desarrollo particular, y de esta forma proveer una solución a la necesidad antes planteada, la OMG definió un Metamodelo para la Ingeniería de Procesos de Software (SPEM) [5].

SPEM describe un metamodelo genérico para la descripción de procesos de software concretos que está basado en las Facilidades de Meta-Objetos (MOF) y utiliza un Lenguaje Unificado de Modelado (UML) [6] como notación. Por tanto, se basa en los principios de orientación a objetos.

El metamodelo SPEM sirve como plantilla para la creación de modelos de procesos concretos, como podrían ser el “Proceso Unificado de Desarrollo de software de Rational” (RUP).

En el marco de la pasantía se utilizó la herramienta Eclipse Process Framework (EPF) Composer, el cual es un software para la definición y especificación de procesos SPEM versión 2.0.

1.5 Arquitectura Orientada a Servicios

La Arquitectura Orientada a Servicios (SOA, Service Oriented Architecture) es un paradigma de arquitectura para diseñar y desarrollar sistemas distribuidos. Las soluciones SOA han sido creadas para satisfacer los objetivos de negocio; estas incluyen facilidad y flexibilidad de integración con sistemas legados, alineación directa a los procesos de negocio reduciendo costos de implementación, innovación de servicios a clientes y una adaptación ágil ante cambios incluyendo reacción temprana ante la competitividad [7].

Este tipo de arquitectura permiten la creación de sistemas de información altamente escalables que reflejan el negocio de la organización. A su vez brinda una forma bien definida de exposición e invocación de servicios, lo cual facilita la interacción entre diferentes sistemas propios o de terceros.

2. Capítulo 2: Descripción del Proyecto

El proyecto de práctica en empresa consistió en el desarrollo de un nuevo sistema para la gestión de la gerencia de planificación y adjudicación orientado al servicio, en base a un nuevo sistema troncal de seguridad.
En este capítulo se describen los recursos utilizados tanto de hardware y software como humanos.

2.1 Equipo de trabajo

Se siguió la metodología Scrum, que recomienda que los grupos de trabajo sean de pocos integrantes. Por lo tanto, para el desarrollo del sistema del IPAV se conformó un equipo de trabajo integrado por 6 personas, incluyendo al Director de Proyecto.

Para el trabajo en equipo, se definió una célula de desarrollo distribuida geográficamente, que mediante el empleo de un software de control de versiones, en este caso GIT, mantenía el código fuente y los casos de prueba sincronizados. Las reuniones se sostenían normalmente con periodicidad semanal, con el objeto de monitorear el avance, cumplimentación de objetivos y re-priorización de funcionalidades si correspondía.

2.2 Recursos Físicos

Los recursos físicos que se utilizaron, proporcionados por el IPAV, fueron 6 (seis) PC´s (una para cada integrante del equipo) con las siguientes características:

Procesador: Doble núcleo CPU 3.30 GHz

Tipo de S.O.: Sistema Operativo de 64 bits

Memoria RAM: 4,00 GB

Disco Rígido: 500 GB

Se contaba con al menos 3 servidores virtuales montados sobre un servidor físico depositado en Aguas del Colorado SAPEM, a través de los cuales se mantienen los códigos fuentes sincronizados, el servidor de prueba y el de producción.

2.3 Herramientas de Software

A continuación se detallaran los entornos de desarrollo, programas y librerías más utilizados para desarrollar el sistema:

NetBeans v7.4: Es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java [8]. Existe además un número importante de módulos para extenderlo. Es un producto libre y gratuito sin restricciones de uso. Se puede descargar desde la página de NetBeans [9].

El mismo fue utilizado durante toda la práctica por su posibilidad de integrarse con servidores como Apache, extenderlo con sistemas de control de versiones como GIT y por su capacidad para poder incorporar múltiples librerías y frameworks, entre otras características.

Pentaho Report designer v5.1.0: Consiste en un motor de presentación capaz de generar informes programáticos sobre la base de un archivo de definición XML [10]. Los mismos son incluidos en el sistema para que el usuario, mediante diferentes comandos, pueda visualizarlos. El software tiene la capacidad de generar reportes en distintos formatos pdf, html, excel, etc. Se puede descargar desde la página de Pentaho [11].

En esta práctica se generaron diferentes reportes a visualizar por el usuario, como por ejemplo los comprobantes de inscripción, comprobantes de solicitud de turnos, el listado de actas con los que trabaja el IPAV, el listado de aspirantes a una vivienda por localidad, entre otros.

pgAdmin III: Es una herramienta de código abierto que soporta todas las características de PostgreSQL [12] y además facilita enormemente la administración de bases de datos. Permite escribir desde consultas SQL simples hasta bases de datos complejas. Se puede descargar desde la página de PostgreSQL [13]

Esta herramienta se utilizó para la creación de tablas, secuencias, triggers y funciones. Estas dos últimas ayudan a mantener la base de datos en forma íntegra. Además, se utilizó para definir los usuarios, roles y permisos para proporcionar seguridad a los datos.

Power Designer v16.1: Proporciona un único entorno de modelado que reúne a las técnicas y notaciones de procesos de negocios y modelado de requisitos, modelado de datos, modelado de arquitectura empresarial, y el modelado de aplicaciones UML.

En ésta práctica, se usó para desarrollar los diferentes diagramas de paquetes, diagramas de clases, mapas de navegación.

Eclipse Process Framework (EPF) Composer v1.5.1: Es una herramienta gratuita, de código abierto, utilizada para implementar, desplegar y mantener procesos para las organizaciones o proyectos individuales. Se puede descargar desde la página de EPF [14].

El mismo fue utilizado para especificar los procesos SPEM.

Herramientas de Microsoft office v2007: Es una recopilación de aplicaciones, las cuales sirven para diferentes funciones como crear, modificar, organizar, imprimir, etc., archivos y documentos.

Este conjunto de herramientas fue utilizado para generar documentación de soporte al equipo de trabajo, como por ejemplo: casos de prueba, documentar las entrevistas con el personal, glosario de términos, etc.

Servidor HTTP Apache: Apache es usado principalmente para publicar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web.

Apache Tomcat 7.0.41.0: Es un contenedor web con soporte de servlets y JSPs. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets [15]. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor HTTP Apache. Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

El IPAV proporcionó un servidor que permitió albergar los diferentes sistemas que se desarrollaron y que continúan desarrollando. En él se instaló Apache Tomcat para acceder a dichas aplicaciones.

PrimeFaces v4.0: PrimeFaces es una librería de componentes para JavaServer Faces (JSF) de código abierto que cuenta con un conjunto de componentes enriquecidos

para facilitar la creación de las aplicaciones web [16]. Se puede descargar desde la página de PrimeFaces [17]

Esta librería es utilizada mediante Netbeans para desarrollar la interfaces gráficas de la aplicación web que utilizarán los usuarios del Sistema del IPAV.

RESTFul Web services: REST (Representational State Transfer) es una técnica de arquitectura software para sistemas distribuidos. Se define un conjunto de reglas por el que se pueden diseñar servicios web (WS) que se centran en los recursos del sistema, incluyendo cómo se dirigen y se transfieren a través de HTTP para una amplia gama de clientes [18,19].

2.4 Utilización de Mantis como sistema para gestionar tareas

Mantis es una de las soluciones actuales más completas para la gestión de tareas dentro de un equipo de trabajo tales como gestionar los requerimientos, asignar tareas, testear soluciones, registrar versiones y seguimiento de la producción remotamente.

Mantis es una aplicación OpenSource (de Código Abierto) desarrollada en PHP y MySQL, fácil de instalar y muy flexible en su configuración [20]. Permite especificar un número indeterminado de estados para cada tarea (abierta, encaminada, testeada, devuelta, cerrada y reabierta) y tantos perfiles como sean necesarios (programador, tester, coordinador, visualizador, etc.). Otra característica es el envío de mails para reportar actividades, como nueva incidencia. También cuenta con la posibilidad de gestión de múltiples proyectos, con distintos tipos de permisos y perfiles para los usuarios.

También se puede configurar el flujo de trabajo, desde la propia herramienta, de forma que se pueden definir por ejemplo, que sólo los testers puedan abrir problemas, que sólo los coordinadores puedan analizarlos y sólo los programadores resolverlos.

En el presente caso, todos tienen acceso al total de la información, en el momento que se establece la tarea de prototipación o desarrollo a un integrante éste carga la incidencia y se la auto-asigna; de esta forma se evita que se carguen incidencias repetidas o que se omitan otras. Además, una vez finalizado el prototipo o desarrollo, puede enviarse la incidencia a otro integrante para realizar el testing,

agregando observaciones en el mismo de ser necesario; y luego reenviarlo para corrección o cerrar la incidencia en señal de tarea finalizada. Mantis también permite adjuntar archivos a las incidencias, ya sea códigos fuentes, documentos de textos e imágenes entre otras, lo cual facilita la explicación entre pares de errores u observaciones.

2.5 Utilización de GIT para el control de versiones y cambios

GIT es un controlador de versiones distribuido. Un controlador de versiones es un sistema que registra los cambios en un archivo o conjunto de archivos a través del tiempo para que se puedan recuperar versiones específicas de los mismos más adelante [21].

Se dice distribuido porque cuando los usuarios descargan la última versión de los archivos no solo están descargando los archivos; sino también realizando una copia fiel y exacta (copia de seguridad) del repositorio (carpeta donde se alojan estos archivos) de esta manera si dicho repositorio muere, cualquiera de los usuarios que descargaron estos archivos pueden restaurar el estado del mismo haciendo uso de su copia de seguridad (ver Figura 2). Por otra parte, esto permite que se puedan tener varios repositorios remotos para poder trabajar y colaborar con diferentes grupos de personas al mismo tiempo en un mismo proyecto, cosa que no es posible con los sistemas centralizados.

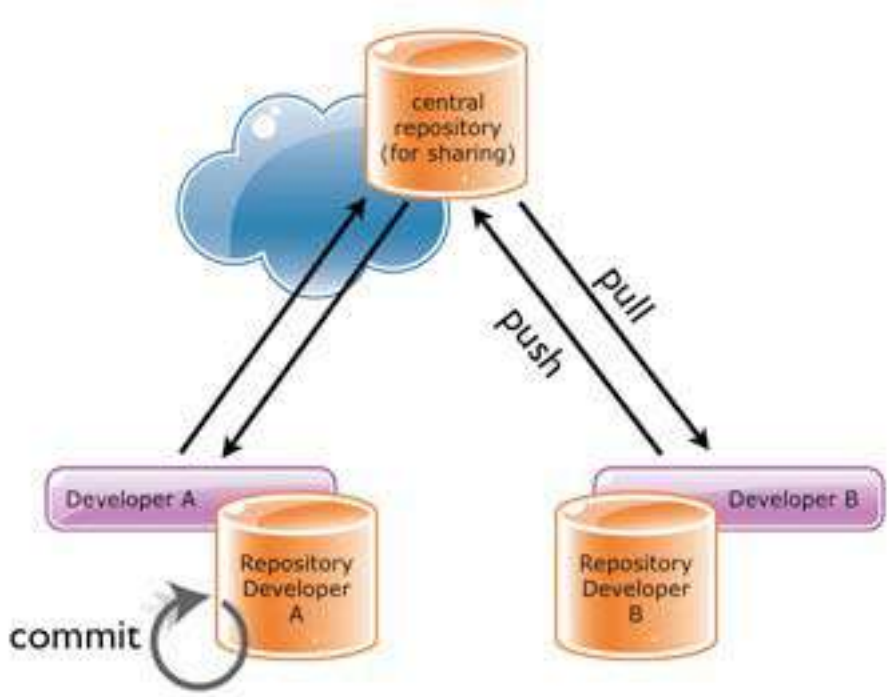

Figura : Arquitectura de un entorno GIT.

2.5.1 Características que destacan de GIT como control de versiones:

Instantáneas, no diferencias: GIT modela los datos como un conjunto de estados de un mini sistema de archivos. Cada vez que se confirma un cambio, o se guarda el estado de un proyecto en GIT, básicamente la herramienta hace una instantánea del aspecto de todos los archivos en ese momento, y almacena una referencia a esa instantánea. Para ser eficiente, si los archivos no se han modificado, GIT no almacena el archivo nuevamente, sólo un enlace al archivo anterior idéntico que ya tiene almacenado.

Casi todas las operaciones son locales: Sí ya se descargó un repositorio no es necesario estar conectado a una red o conexión a internet para trabajar sobre los archivos, todo lo necesario se encuentra en la computadora local.

Integridad de los datos: Cada componente en GIT es verificado mediante un checksum (suma de comprobación) antes de ser almacenado, y es identificado a partir de ese momento mediante dicha comprobación. De esta manera es imposible cambiar el contenido de cualquier archivo sin que GIT lo note.

GIT sólo añade información: Cuando se realizan operaciones en GIT, generalmente sólo se añade información a la base de datos de GIT. Esto asegura que el sistema no haga cambios que no se puedan deshacer, o que de algún modo borre información.

Trabaja con tres estados: GIT tiene tres estados principales en los que se pueden encontrar los archivos: committed (confirmado), modified (modificado), y staged (preparado). Confirmado significa que los datos están almacenados de manera segura en la base de datos local. Modificado significa que se han realizado modificaciones en el archivo pero todavía no se han confirmado en la base de datos. Preparado significa que se ha marcado un archivo modificado en su versión actual para que evolucione a su próxima confirmación.

3. Capítulo 3: Relevamiento

Al inicio del proyecto, se realizó un relevamiento de requerimientos del sistema [22] de seguridad e inscripciones, con un relevamiento inicial del módulo de contable, que tiene como principal funcionalidad la gestión de cobros de las viviendas adjudicadas. A partir de Enero de 2015, se siguió con el relevamiento detallado del módulo contable, que se incorporó al sistema de seguridad e inscripciones. Tanto el relevamiento del sistema de seguridad e inscripciones como del nuevo módulo contable consistieron de dos tareas, a saber: 1) entrevistas al personal y 2) análisis de la documentación.

Entrevista al personal: Se realizaron una serie de entrevistas con grabación de sonido, al personal del IPAV en distintas áreas de la gestión. Dichas entrevistas fueron realizadas tanto en las delegaciones de Santa Rosa como de General Pico por distintos integrantes del equipo de trabajo. Luego las grabaciones fueron escuchadas y pasadas en limpio para analizarlas y substraer de las mismas el conocimiento del dominio.

Análisis de la documentación: Se analizó la reglamentación vigente y documentación anexa, referida a la inscripción y adjudicación de viviendas y temas relacionados, con el fin de establecer claramente las reglas del negocio.

A partir del conocimiento adquirido a través de las entrevistas y de la documentación, se generaron especificaciones que describieran con claridad, sin ambigüedades, en forma consistente y compacta cuáles eran los requerimientos funcionales y no funcionales del sistema, como así también, qué datos se debería guardar y cuál es comportamiento del sistema, entre otros.

Con dicha especificación se realizó el modelado de los procesos en SPEM, los cuales fueron separados en distintos paquetes, que se detallarán en el capítulo 4.

3.1 Identificación de los requerimientos funcionales a desarrollar

En esta etapa se analizaron las funcionalidades a desarrollar a partir de los documentos de relevamiento. Los requerimientos funcionales describen las acciones que

desea el cliente obtener del sistema. En el siguiente listado se muestran algunas de las funcionalidades detectadas (el listado no es exhaustivo debido a restricciones en informar datos protegidos del organismo).

Así, del relevamiento para el *módulo de inscripción y adjudicación*, se pudieron identificar los siguientes requerimientos funcionales:

- Consultar los turnos asignados.
- Administrar las impugnaciones.
- Dar de alta a un aspirante.
- Administrar los turnos de inscripción de aspirantes.
- Administrar la cancelación de una vivienda.
- Administrar el cambio de titularidad de un grupo familiar.
- Administrar las denuncias de vivienda.
- Administrar el comodato de una vivienda.
- Administrar la permuta de viviendas entre adjudicados.
- Administrar la renuncia a una vivienda.
- Visualizar adjudicaciones de viviendas.
- Emitir listados y certificados correspondientes.
- Otras funcionalidades.

Para el *módulo contable* se detectaron, entre otros, los siguientes requerimientos funcionales:

- Administrar cancelaciones de viviendas
- Administrar créditos hipotecarios
- Administrar certificados de obras
- Visualizar certificados de obras.
- Visualizar costos de vivienda.
- Emitir multas.
- Emitir actas de medición.
- Administrar acopios.
- Visualizar suministros.

- Visualizar inventario.
- Visualizar cobros.
- Administra cuotas.

3.2 Identificación de los requerimientos no funcionales

En esta etapa se analizaron los requerimientos no funcionales a partir de las entrevistas al cliente y las deficiencias observadas en el sistema anterior. A continuación se listan los requerimientos no funcionales que debe cumplir el sistema.

- **Disponibilidad**: Estar disponible 100% durante el horario hábil laboral del IPAV.
- **Flexibilidad**: El sistema debe ser construido sobre la base de un desarrollo evolutivo e incremental, de manera tal que nuevas funcionalidades y requerimientos relacionados puedan ser incorporados afectando el código existente de la menor manera posible; para ello deben incorporarse aspectos de reutilización de componentes.
- **Instalación**: El sistema debe ser fácil de instalar en todas las plataformas de hardware y software definidas por el área de Sistemas del IPAV.
- **Mantenibilidad**: Todo el sistema deberá estar complemente documentado, cada uno de los componentes de software que forman parte de la solución propuesta deberán estar debidamente documentados tanto en el código fuente como en los manuales de administración y de usuario.
- **Seguridad**: El acceso al Sistema debe estar restringido por el uso de claves asignadas a cada uno de los usuarios. Sólo podrán ingresar al Sistema las personas que estén registradas, estos usuarios serán clasificados en varios tipos de usuarios (o roles) con acceso a las opciones de trabajo definidas para cada rol.

4. Capítulo 4: Análisis y Diseño

4.1 Modelo de datos

Tomando como punto de partida la base de datos pre-existente y teniendo en cuenta los requerimientos relevados, se realizó el modelado de datos para el módulo de seguridad e inscripciones del sistema del IPAV. Al cual más tarde se extendió para incorporar el modelo de datos del módulo contable. Se utilizó la herramienta de diseño Power Designer para construir los diagramas de entidad relación (DER) [23].

Como el sistema de seguridad e inscripciones debe almacenar información compleja tomada de distintos dominios, a los efectos de mejorar la comprensión estática (por ejemplo, integridad referencial) y dinámica (por ejemplo, triggers) de los datos y sus relaciones, se decidió generar las siguientes vistas lógicas.

- El diagrama de Personas, donde se modelan los datos personales del individuo, sus distintas direcciones, grupo familiar, ficha de inscripción entre otras.
- El diagrama de Ranking, donde se muestra como se relaciona una ficha de inscripción y su posición en el ranking.
- El diagrama de Viviendas, muestra la información que hace única a la vivienda dentro de un proyecto de viviendas.
- El diagrama de Adjudicaciones, que está fuertemente ligado a los 3 diagramas anteriores, ya que modela las relaciones entre una ficha de inscripción y una vivienda según un ranking.
- El diagrama de Usuarios, tiene todo lo referente a los usuarios del sistema, permisos, roles, acciones disponibles, etc. El cual está muy ligado al sistema de seguridad.
- El diagrama de Certificaciones, muestra como lase
- certificaciones interactúan con todos los ítems relacionados al arranque, seguimiento y finalización de la obra.

- El diagrama de Liquidaciones, donde se muestra como se relacionan los diferentes tipos de liquidaciones, como por ejemplo, liquidaciones de cuotas, multas, cancelaciones y ajustes.
- El diagrama de Operatoria de Asistencia Financiera Individual (OAFI), donde se muestra como se relaciona una solicitud de crédito con la documentación necesaria, personas relacionadas a la obra, el cronograma de la misma, las cuotas del crédito y el desembolso del mismo.
- El diagrama de Proyectos, tiene todo lo referente a los datos de los proyectos de vivienda y los datos económicos de los mismos.
- El diagrama de Pagos, tiene lo referente a la administración de pagos y pedidos de fondos.

Una vez terminado el modelo de datos, éste fue analizado por dos integrantes del equipo de trabajo, que no habían formado parte del diseño, con el fin de depurar posibles errores.

A partir del diseño de los datos se creó la base de datos sobre PostgreSQL utilizando pgAdmin para su administración, también se crearon los triggers (con sus eventos asociados) y funciones correspondientes, para resguardar la integridad y persistencia de los datos.

4.2 Modelado de Procesos de Negocio

Los modelos de procesos de negocio permiten comprender los mecanismos principales del mismo. Estos actúan como base para la creación del sistema informático, facilitando la identificación de la estructura actual del negocio y su operación. Permiten identificar ideas para mejorar.

El modelo debe concentrarse en las tareas y mecanismos claves del negocio. Determinar con precisión las tareas principales e identificar qué debe plasmarse en el modelo.

A partir del relevamiento de los procesos, se realizó reingeniería para determinar con precisión las tareas principales, los roles, la documentación utilizada y las áreas

involucradas. Para esta tarea se utilizó la herramienta EPF Composer y el meta modelo SPEM versión 2.0.

Para el módulo de inscripciones y adjudicaciones se identificaron tres grupos de procesos (ver Figura 3):

- Inscripciones: procesos vinculados a la incorporación de aspirantes y adjudicación de viviendas.
- Irregularidades: procesos asociados con la registración y seguimiento de denuncias sobre beneficiarios de viviendas sociales que aún no han cancelado la vivienda.
- Operatorias de Mesa: procesos asociados con distintas opciones que poseen los beneficiarios sobre las viviendas, en la medida que no tengan irregularidad asociada, y que sean a la fecha adjudicatarios de una vivienda social.

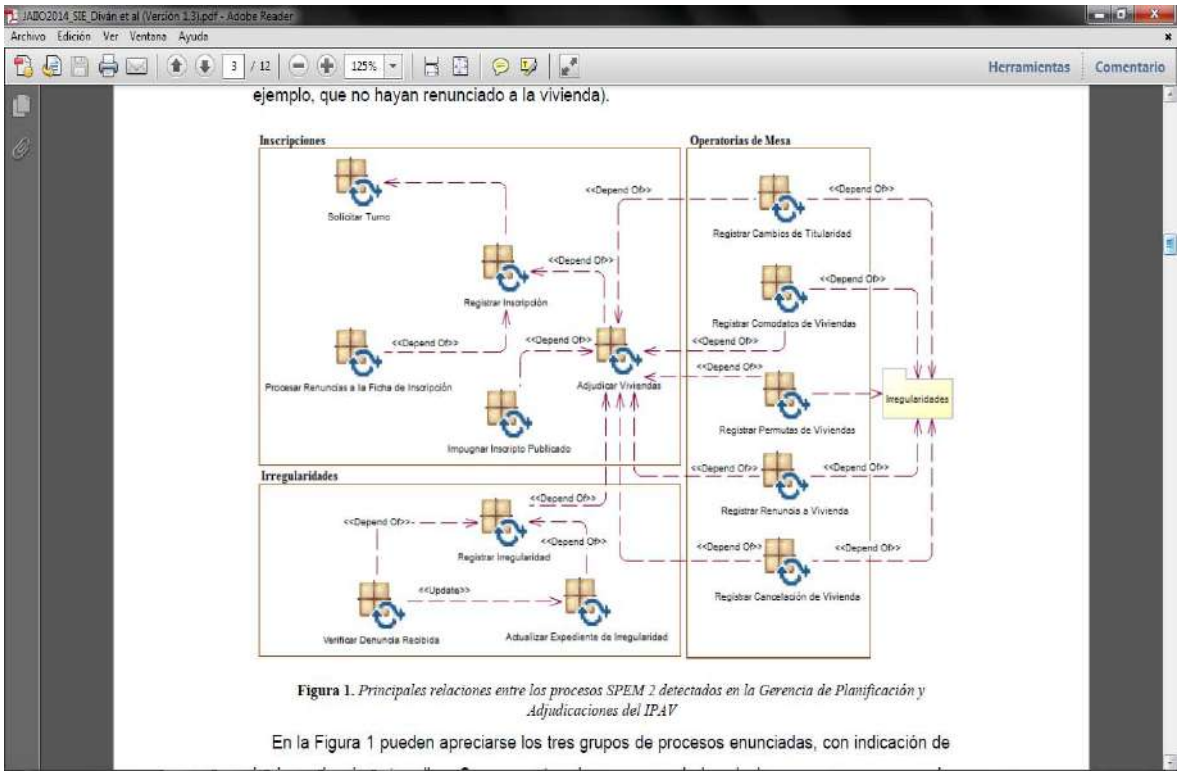

Figura : Principales relaciones entre los procesos detectados.

Para el módulo contable se identificaron los siguientes grupos de procesos:

- Certificaciones: procesos vinculados con las inspecciones, seguimiento de obras y elaboración de certificados de obras.
- OAFI: procesos vinculados con la solicitud, reprogramación y cancelación de los créditos hipotecarios.
- Recaudación: procesos vinculados con la recaudación de las cuotas de viviendas sociales y cancelación de las mismas.
- Inventario: procesos vinculados con la compra y entrega de suministros a los distintos departamentos del IPAV.

4.3 Identificación de los casos de uso

A partir del modelado de los procesos del IPAV, se identificaron posibles casos de uso y los actores asociados. Se diseñaron 44 casos de usos para el sistema de inscripción y adjudicación de viviendas y para el sistema contable se diseñaron 66. Los casos de uso se agruparon en distintos paquetes para mejorar su cohesividad, minimizar el grado de acoplamiento entre paquetes y promover una lectura más consistente de ellos. En la siguiente sección denominada “Construcción del diagrama de paquetes”, se muestra el diagrama de paquetes y una breve descripción de cada uno de ellos.

Para el modelado de los casos de uso se utilizó la herramienta Power Designer. A modo ilustrativo se muestra el diagrama de casos de uso del proceso “registrar inscripción” (ver Figura 4) y el caso de uso del proceso “renunciar a ficha de inscripción” (ver Figura 5).

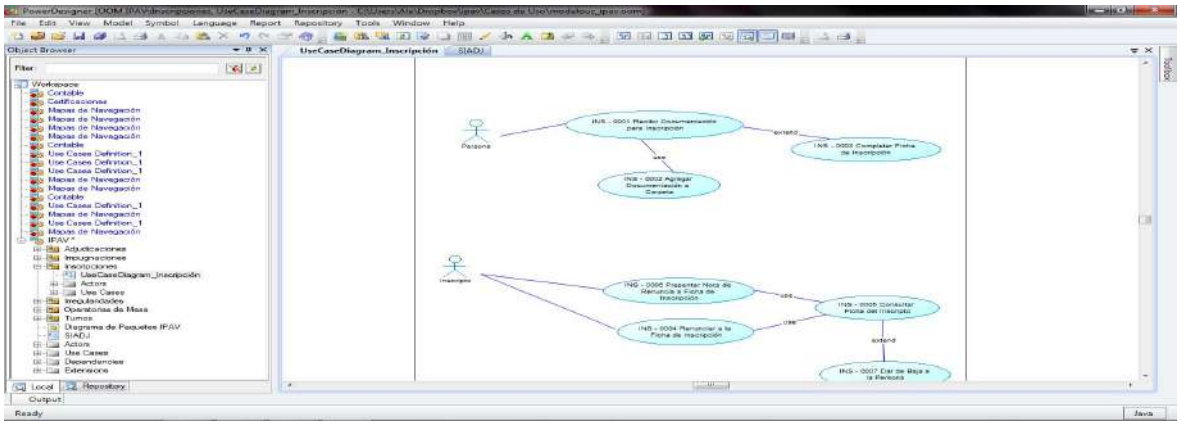

Figura : Diagrama de caso de uso; Registrar Inscripción.

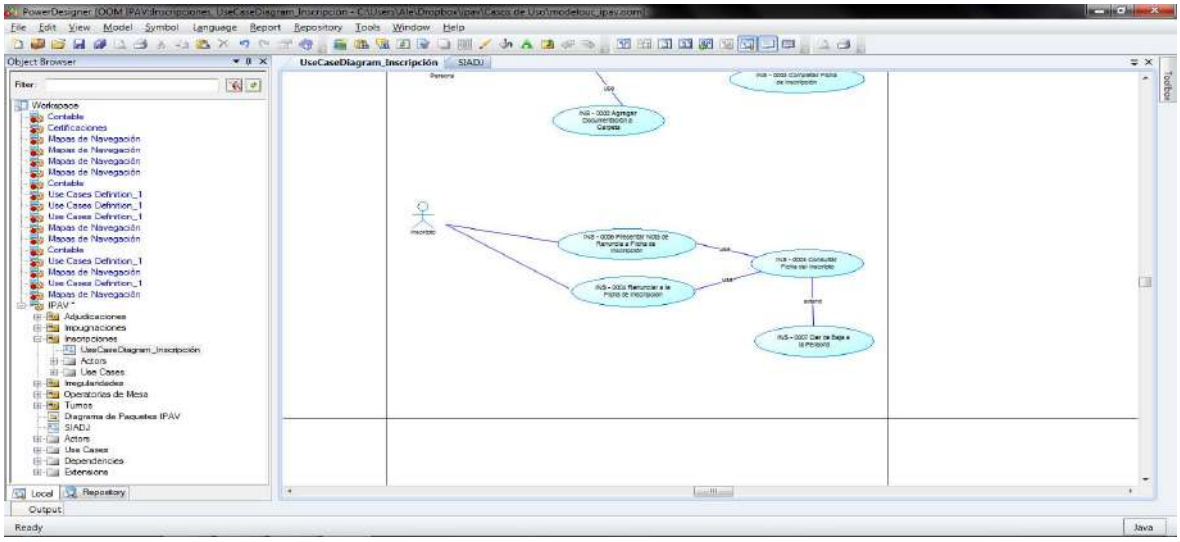

Figura : Diagrama de caso de uso; Renunciar Ficha de Inscripción.

Para la especificación del proceso de cada caso de uso se utilizó la planilla expuesta como Figura 6, donde se asigna a cada caso de uso un código, se describe su objetivo, sus requerimientos no funcionales que tengan influencia y de ser necesario, sus pre y post condiciones.

Actividad	**Código**	**Nombre**

Objetivo	
Pre-condiciones	
Especificación del proceso	
Post-condiciones	
Requerimientos no funcionales asociados	
Observaciones	

Figura : Planilla para la especificación de casos de uso.

En la Figura 7 y a título de ejemplo, se muestra la especificación del caso de uso INS – 0003 “Completar Ficha de Inscripción”.

Actividad	**Código**	**Nombre**
	INS – 0003	**Completar Ficha de Inscripción**
Objetivo	**Ingresar los datos correspondientes de la persona en la ficha de inscripción del sistema.**	
Pre-condiciones	**Que la persona haya presentado la documentación requerida.**	
Especificación de la actividad	**1. Responsable de Inscripción debe completar la ficha de inscripción con los datos de la persona.** • ***Carga los datos generales:*** **Residencia Actual, desde, País de Nacimiento, Provincia de Nacimiento, Localidad de Nacimiento.** • ***Carga los datos de Titular:*** **Apellido, Nombre, Tipo y Nº Documento, CUIL/CUIT, Fecha de Nacimiento, Estado Civil.** • ***Carga los datos vinculados al Domicilio de Contacto:*** **País, Provincia, Localidad, Calle, Nº, Edificio, Piso, Departamento, Teléfono, Celular de contacto, E-mail.** • ***Carga los datos vinculados a la Situación Laboral Actual:*** **Situación Laboral, Lugar de Trabajo, Tarea que realiza, Sueldo/Ingreso Mensual, Tipo de Trabajo, Situación de revista.** • ***Carga los datos del Grupo Familiar:*** **Apellido, Nombre, Relación de Parentesco, Sexo, Tipo Documento, N° Documento, Fecha de Nacimiento, País, Provincia, Localidad, Lugar de Trabajo, Tarea que realiza, Ingresos Mensuales.** • ***Carga los datos asociados a la Situación Habitacional:*** **Tipo de vivienda, Forma de Tenencia.**	

	• ***Carga lo datos vinculados a Adjudicaciones anteriores:*** **Adjudicaciones Anteriores, Plan , Causa por la que dejó.** • ***Carga los datos relacionados con Discapacidades:*** **Miembro familiar que posee discapacidad, Clasificación, Tipos de discapacidad.** **2. El Responsable de Inscripción confirma los datos ingresados en la ficha de inscripción.** **3. Le comunica a la persona el número con el que está inscripto en la ficha.** **4. La Persona recibe una copia de la ficha de inscripción y firma la misma.** **5. Fin.**
Post-condiciones	**No presenta.**
Requerimientos no funcionales asociados	
Observaciones	

Figura : Especificación del caso de uso Completar Ficha de Inscripción.

4.4 Construcción del diagrama de clases

En esta etapa del diseño se realizó la identificación de las clases que conforman el sistema, definiendo sus atributos, métodos y la visibilidad. Con el listado de las posibles clases, se confeccionó el diagrama de clases inicial con las distintas asociaciones entre ellas (ver Figura 8). Para esta tarea nuevamente se utilizó la herramienta Power Designer.

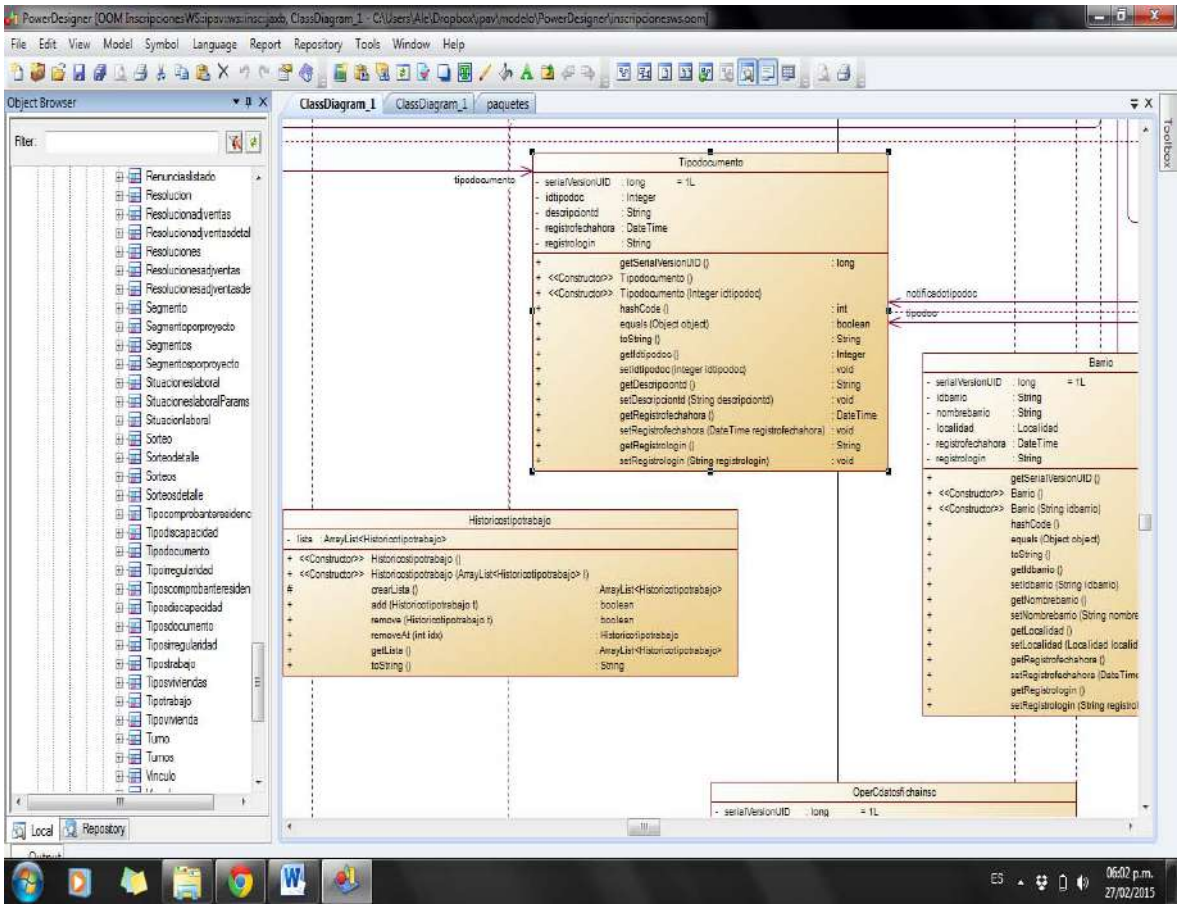

Figura : Parte del diseño del diagrama del clases.

4.5 Construcción del diagrama de paquetes

La construcción del diagrama de paquetes tiene como objetivo agrupar la lógica del sistema de forma que se maximice la articulación y se minimice el acoplamiento de los mismos, al mismo tiempo mejorar su mantenibilidad.

Para el sistema de inscripciones y adjudicaciones se agrupóa la lógica de la siguiente manera (ver Figura 9).

- Turnos: Se lleva a cabo la gestión de la solicitud de turnos para que la persona pueda luego realizar la inscripción correspondiente.
- Inscripciones: Se gestiona todo lo relacionado con el alta de la persona en el sistema de inscripciones y en caso de ya estar inscripto, actualizar la información asociada a dicha persona.
- Adjudicaciones: Se lleva a cabo la gestión de todo lo asociado con el cálculo de puntaje de los grupos familiares para acceder a una vivienda, el sorteo de las mismas, la pre adjudicación de las viviendas a adjudicar.
- Operatoria de Mesa: En este paquete se concentran todas las operatorias de mesa, asociadas con la cancelación, comodato, renuncia y demás de las viviendas adjudicadas.
- Impugnaciones: Se llevan a cabo la recepción y verificación de las denuncias recibidas sobre aquellas personas pre - adjudicadas de viviendas.
- Irregularidades: Se gestiona todo lo referente con las denuncias de las personas cuya vivienda fue adjudicada y no viven en la misma. Se lleva a cabo verificaciones de vivienda para corroborar la denuncia recibida.

Figura : Diagrama de paquetes.

4.6 Mapas de Navegación

El Diagrama de mapa de navegación representa una idea general de la navegación entre páginas de la aplicación web. El mapa de navegación se debe crear uno por subsistema o bien, si es pequeña la aplicación, un solo mapa de navegación para el sistema completo. El mapa es de gran ayuda para identificar con rapidez la ubicación de las diferentes funcionalidades dentro del sistema, las características de cada subsistema y para tener una idea de cómo ordenar el sistema en la web.

Para el desarrollo del mapa de navegación se utilizó la herramienta Power Designer. Con ella se realizó un diagrama preliminar del mapa de navegación y luego se fue actualizando a medida que se iba desarrollando la aplicación web. Como ejemplo, en la Figura 10 se puede ver el mapa de navegación correspondiente a tipos de documentos.

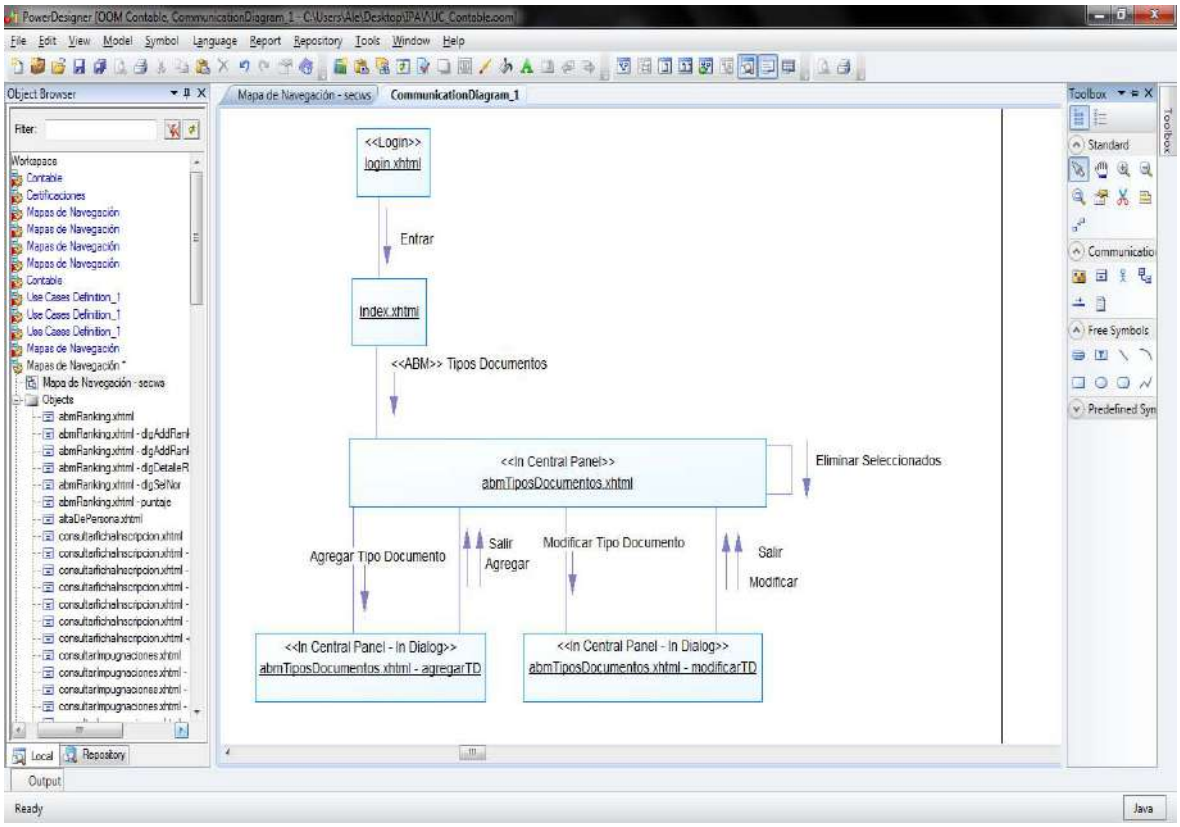

Figura : Mapa de navegación del ABM de Tipos de Documentos.

4.7 Prototipación de interfaces

En esta etapa se desarrollaron los prototipos de las pantallas destinadas al sistema de seguridad e inscripciones y adjudicaciones del IPAV. Para cumplir con ésta tarea se utilizó la herramienta de desarrollo NetBeans, junto con la librería PrimeFaces.

Los prototipos se desarrollaron teniendo en cuenta el modelo de datos, la especificación de los procesos realizados en EPF Composer, el mapa de navegación y los casos de usos identificados. El modelo de datos nos aporta la definición del dominio de valores asociado con cada dato, por lo que es posible seleccionar los componentes y delinear las reglas de verificación asociadas. Los procesos formalizados nos permiten situar la relación del prototipo en relación de otras tareas/actividades/procesos del sistema de información. El mapa de navegación junto con el modelo de procesos, permite visualizar el grado de acoplamiento entre los prototipos modelados y finalmente, los casos de usos permiten aproximar la complejidad de cada prototipo.

Estas pantallas son sólo de muestra, junto con el diagrama de casos de uso y la especificación de procesos ayudan a que el equipo de trabajo pueda comunicarle al cliente aspecto y funcionalidades que tendrá el sistema terminado. Por otro lado, el cliente puede identificar cualquier discrepancia del sistema solicitado y realizar las correcciones pertinentes.

A continuación, y a modo de ejemplo, se expone como Figura 11 el prototipo de interfaces para registrar una ficha de inscripción. Esta pantalla surge del caso de uso INS – 0003 “Completar Ficha de Inscripción” (ver Figura 4). Dentro de la especificación de este caso de uso (ver Figura 7) se describe la carga los datos generales, la carga del grupo familiar, la carga de datos asociados a la Situación Habitacional, entre otros, los cuales se pueden apreciar en la interface expuesta.

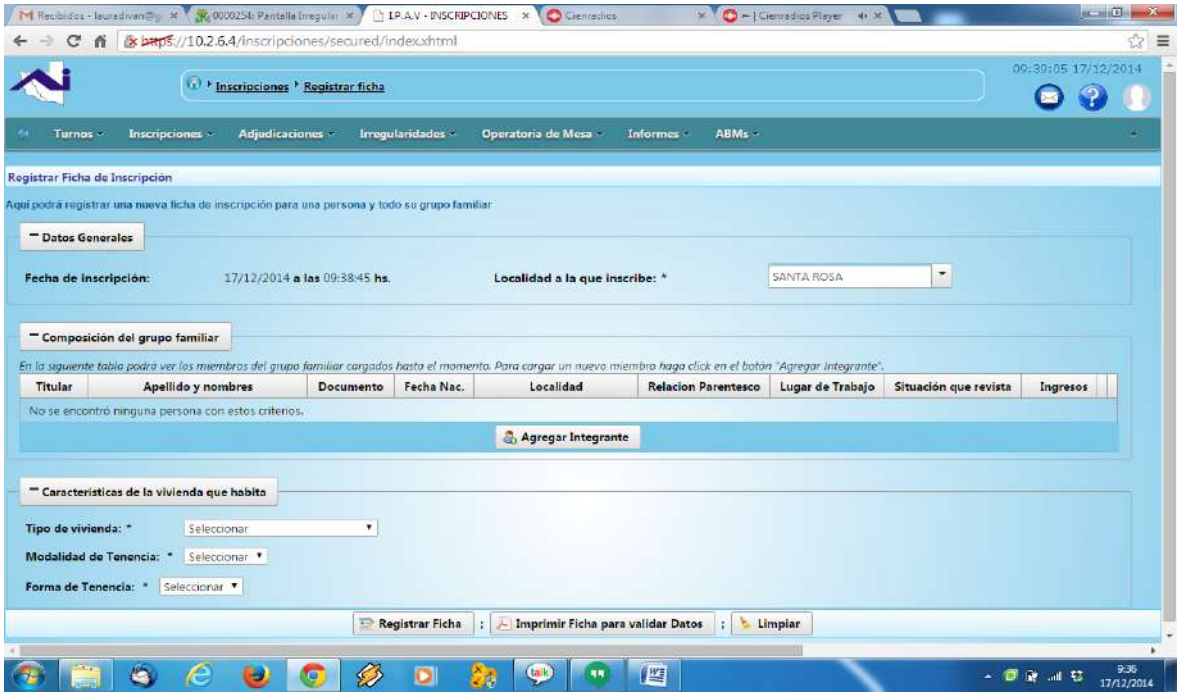

Figura : Prototipo de interface para Registrar Ficha de Inscripción.

Este caso de uso proviene del proceso SPEM modelado Registrar inscripción (ver Figura 12), el cual tiene tareas por ejemplo, Carga de datos generales en la inscripción y cargar grupo familiar.

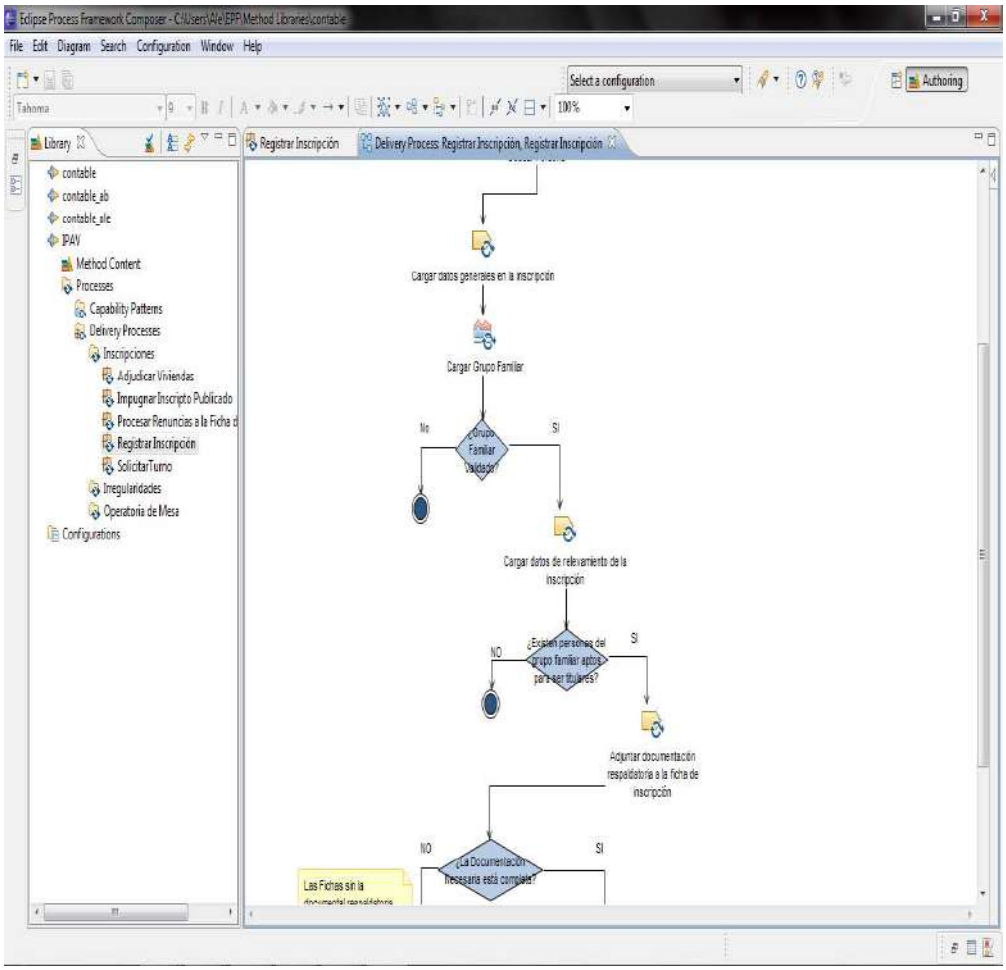

Figura : Porción del proceso Registrar Inscripción.

En el mapa de navegación que se muestra en la Figura 13 se puede apreciar como la pantalla utilizada de ejemplo se acopla al funcionamiento del sistema.

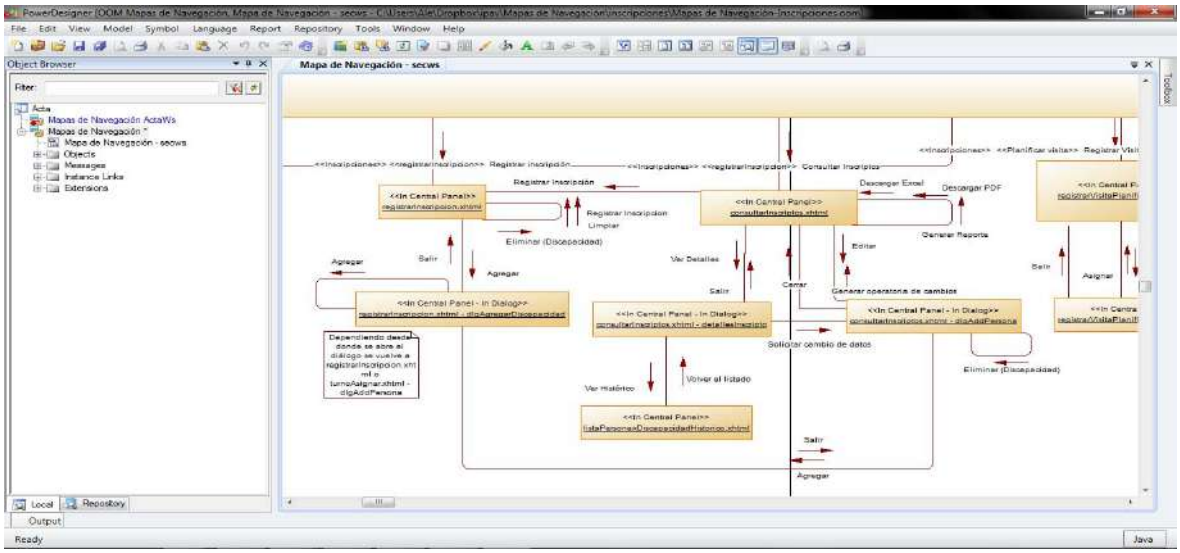

Figura : Porción del mapa de navegación de Registrar Inscripción.

5. Capítulo 5: Implementación

5.1 Programación de la capa lógica

La programación de la capa lógica consistió en desarrollar los servicios principales de la aplicación, estableciendo el nexo entre la interface de usuario y la base de datos.

Como se utilizó una Arquitectura Orientada a Servicios [24], tomando como estándar RESTFul, se necesitó de varios pasos para la creación de los servicios web. Lo primero que se necesitó hacer fue crear los objetos java JAXB, que se usan como parámetros de los WS. Para ello, dichos objetos se deben poder representar en XML, es decir deben ser serializables.

Para esto se utilizó la librería Google JSON (GSON) que permite transformar objetos en XML y viceversa, además de mapearlos naturalmente como representación JSON [25]. A continuación, se muestra el desarrollo de una clase, en este caso la clase "Tipodocumento" (ver Figura 14).

```
@XmlRootElement(name="Tipodocumento")
@XmlType(propOrder={"idtipodoc","descripciontd",
    "registrofechahora","registrologin"})
@XmlAccessorType(XmlAccessType.NONE)
public class Tipodocumento implements Serializable {
            ...
    @XmlElement
    public Integer getIdtipodoc() {...3 lines }

    /**...3 lines */
    public void setIdtipodoc(Integer idtipodoc) {...3 lines }

    /**...3 lines */
    @XmlElement
    public String getDescripciontd() {...3 lines }

    /**...3 lines */
    public void setDescripciontd(String descripciontd) {...3 lines }

    /**...3 lines */
    @XmlElement
    @XmlJavaTypeAdapter(DateTimeAdapter.class)
    public DateTime getRegistrofechahora() {...3 lines }
            ...
```

Figura : Código fuente, se muestra la utilización de la librería JAXB.

A continuación se explica la función de cada componente JAXB utilizado:

- @XmlRootElement: Define cual va a ser el elemento raíz del XML. Con el atributo opcional “Name” se le puede asignar un nombre al elemento.
- @XmlType: Asigna un esquema de representación de los datos en XML. Con el elemento PropOrder, se define una lista de nombres de atributos java en la clase. Este es el orden de los elementos del esquema XML cuando dichos atributos se serializan.
- @XmlAccesorType: Controla si los atributos java son serializados por defecto. Con el elemento XmlAccessType.None se está estableciendo que los atributos no se mapearan a XML a menos que estén especificados con algunas de las anotaciones JAXB
- @XmlElement: Mapea un atributo Java a un elemento XML derivando el nombre de la propiedad.

También se creó la clase “Tiposdocumento” que es un contenedor de objetos Tipodocumento y la clase SesionTipodocumento la cual es utilizada como parámetro dentro de la declaración de recursos RESTFul (ver Figura 15). Estas clases también fueron mapeadas utilizando la librería JAXB.

Figura : Diagrama de clase, se muestra la relación de Tipo Documento.

El segundo paso necesario para la creación de los servicios web fue declarar las interfaces de las funciones que daban lugar a los servicios e implementar las mismas. En la Figura 16 se puede apreciar la declaración de la interface “ITipodocumento” para el WS. En otras palabras, la interface define el contrato de los servicios que deberá proveer la clase que la implemente. Tales servicios, vienen derivados justamente del modelo de procesos y de los casos de uso.

```
/** Interface para la gestión de Tipos de Documentos ...6 lines */
public interface ITiposdocumento {
    /** Permite la consulta por patrón de contención los ...11 lines */
    String consultar(String idsesion, String patron);

    /** Consulta por identificador de instancia ...9 lines */
    String consultarByID(String idsesion, Integer idtipodocumento);

    /** Permite incorporar nuevos tipos de documentos ...8 lines */
    String registrar(String idsesion, Tiposdocumento lista);

    /** Permite la modificación de la descripción de tipos ...8 lines */
    String modificar(String idsesion, Tiposdocumento lista);

    /** Permite dar de baja los tipos de documentos indicados, ...8 lines */
    String borrar(String idsesion, Tiposdocumento lista);
```

Figura : Interface de los métodos del servicio web Tipo Documento.

Luego esta interface fue implementada en la clase "ITipoDocumentoImpl", donde se desarrolló las funcionalidades propias de cada método (ver Figura 17).

```
public class ITipoDocumentoImpl extends BaseInscWSImpl implements ITiposdocumento{
                ...
    public String consultar(@Context ServletContext pcontexto, @Context HttpServletRequest prequest,
            String idsesion, String patron){...7 lines }

    @Override
    public String consultar(String idsesion, String patron) {...83 lines }
                ...
    public String registrar(@Context ServletContext pcontexto, @Context HttpServletRequest prequest,
            String idsesion, Tiposdocumento lista){...7 lines }

    @Override
    public String registrar(String idsesion, Tiposdocumento lista) {...90 lines }
                ...
```

Figura : Implementación de los métodos del servicio web Tipo Documento.

El tercer paso necesario para la creación de los servicios web es la declaración de los recursos (ver Figura 18). Estos son accedidos por componentes o aplicaciones mediante su URI (Identificador Uniforme de Recurso). Los WS son accedidos por medio de los recursos.

```
// ////////////////////////////////////////////////////////////////////////////
// ITiposdocumento
// ////////////////////////////////////////////////////////////////////////////

    @Path("/consultarTipoDocumento")
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String consultarTipoDocumento(@QueryParam("idsesion") String idsesion,
            @QueryParam("patron") String patron) {...5 lines }

    @Path("/registrarTiposDocumento")
    @POST
    @Produces(MediaType.APPLICATION_XML)
    @Consumes(MediaType.APPLICATION_XML)
    public String registrarTiposDocumento(SesionTipoDocumento xml) {...10 lines }
```

Figura : Implementación de los recursos de Tipo documento.

5.2 Programación de las UIs asociadas con los prototipos

Se programaron las interfaces de usuarios, teniendo en cuenta los prototipos de pantalla desarrollados en la etapa de diseño y haciendo uso de los WS desarrollados con anterioridad. Para establecer el orden en que se programaron las interfaces de usuario se tuvo en cuenta, como criterios más importantes, el diagrama de paquetes, los diagramas de procesos y los modelos de datos.

Para el desarrollo de las interfaces de usuario se trabajó con JavaServer Faces (JSF) [26], incorporando la librería de componentes PrimeFaces. JSF es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones. Esta incluye:

- Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- Un conjunto por defecto de componentes para la interfaz de usuario.
- Bibliotecas de etiquetas personalizadas para JavaServer Pages (JSP) [27] que permiten expresar una interfaz JSF dentro de una página JSP.
- Un modelo de eventos en el lado del servidor.
- Un administrador de estados.

- Un conjunto de Beans administrados.

A manera ilustrativa se muestran dos fragmentos de la pantalla relacionada con Tipo documento. En la Figura 19 se muestra la pantalla principal donde se puede visualizar los Tipos de Documentos. En la misma se puede filtrar los datos por id y descripción, se puede Agregar un Tipo Documento, seleccionar uno o más para eliminar o editar (icono en forma de lápiz).

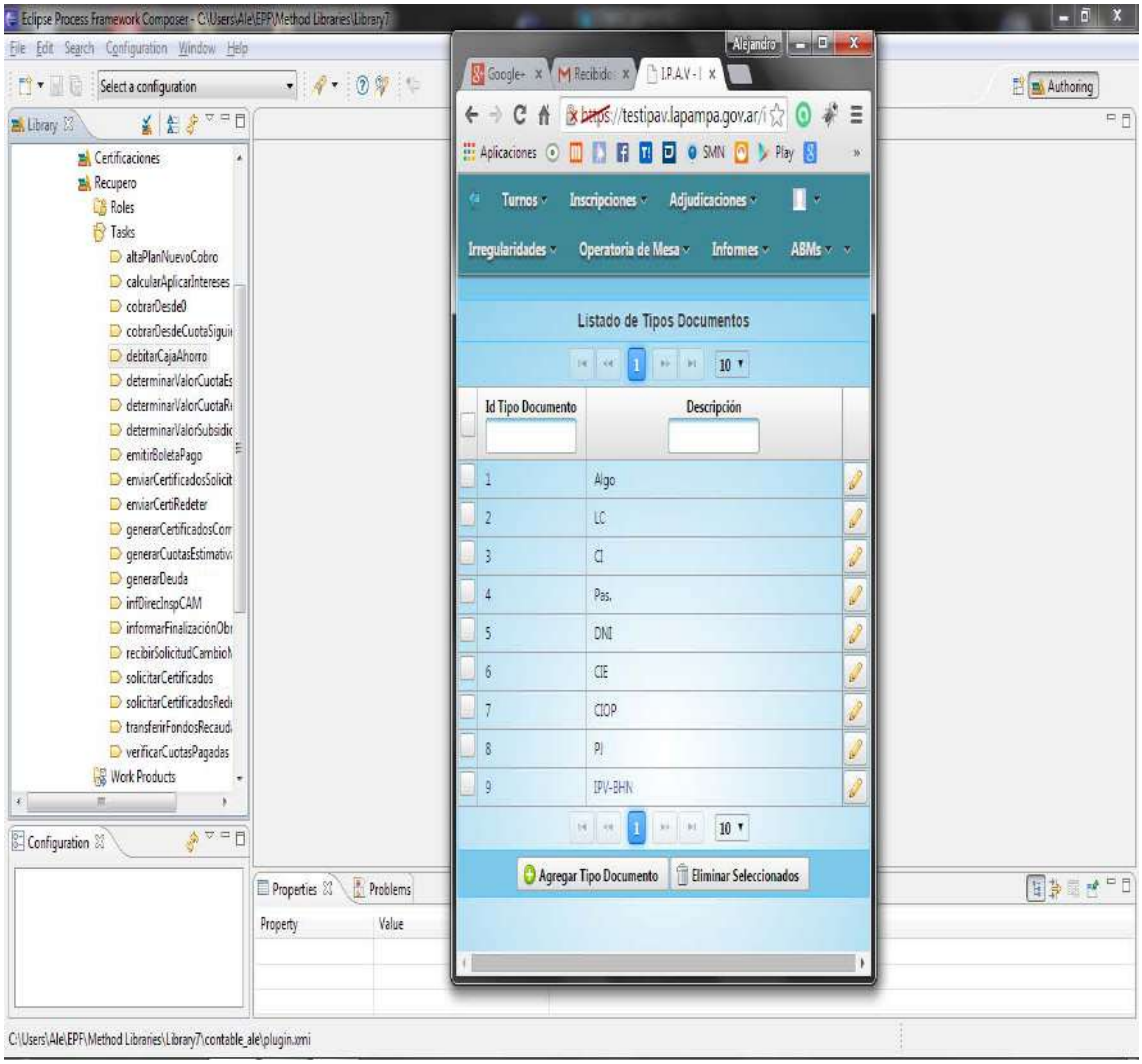

Figura : Pantalla de Tipo Documento.

En la Figura 20 se puede ver el diálogo que surge cuando se quiere Agregar un Tipo Documento. En él se puede ingresar un id y una descripción válida (que cumpla con requisitos como por ejemplo no ser null).

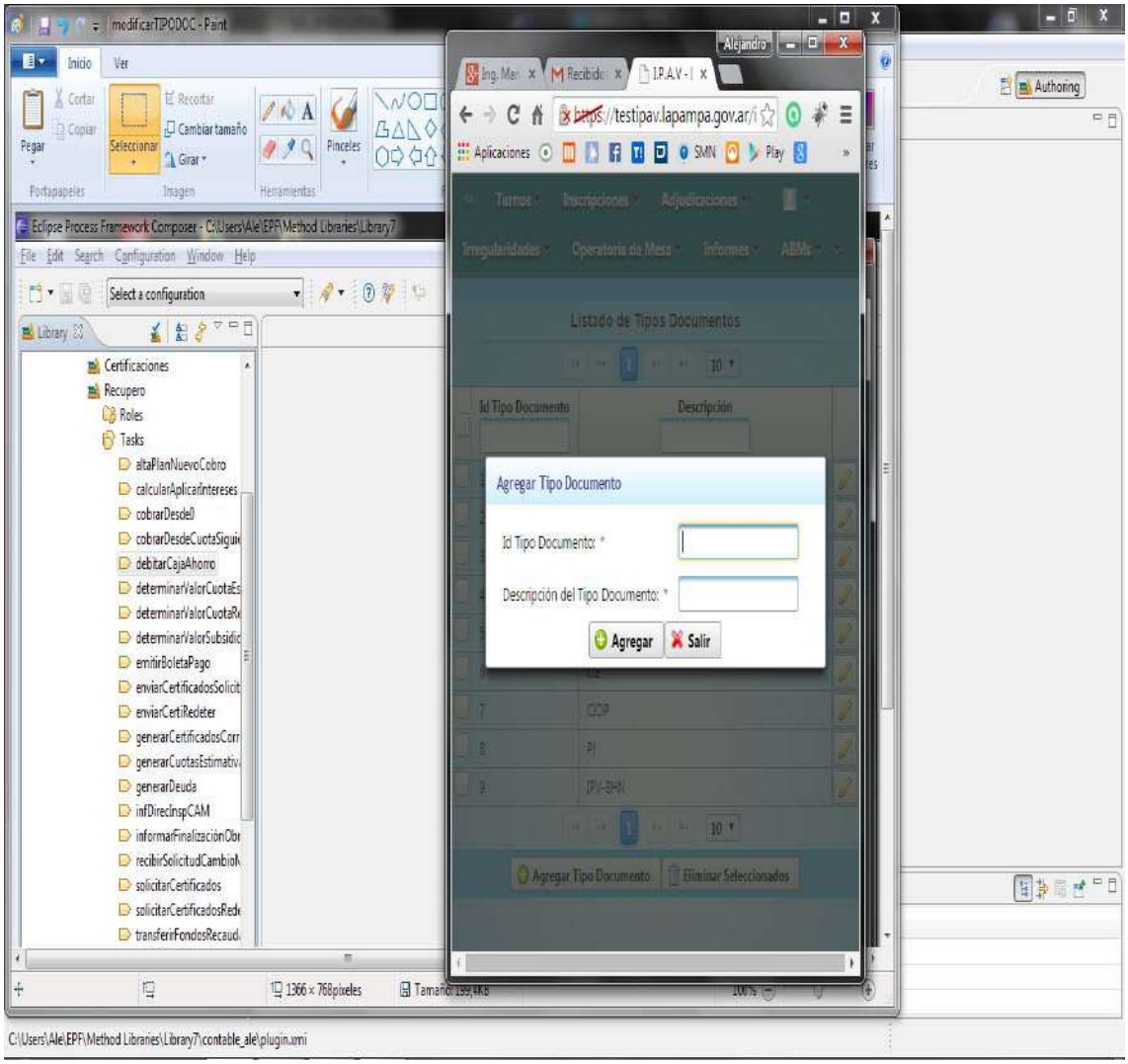

Figura : En tipo documento, el diálogo Agregar Tipo Documento

6. Capítulo 6: Pruebas del Sistema

6.1 Pruebas a nivel de Clases

Luego de programar las clases java JAXB y los contenedores, el equipo de trabajo realizó las pruebas unitarias de las mismas. El objetivo de éstas fue detectar algún error de programación en las clases antes de ser usadas en los servicios web. Se evaluó la visibilidad de los atributos, los métodos de acceso a los mismos, el constructor de las clases y que se realice la transformación a XML de las clases de forma exitosa.

Los resultados obtenidos de las pruebas fueron documentados en planillas específicas para esta función (ver Figura 21). Estas eran remitidas al encargado de programar la clase o modulo testeado para la corrección del mismo. Las planillas pueden tener 2 resultados posibles, “el esperado” (ver Figura 22) el cual indica que la prueba pasó correctamente y el “no esperado” (ver Figura 23) el cual indica que el desarrollador tendrá que volver a revisar la programación del mismo.

Nombre de Caso de Prueba:	
Objetivo:	
Pre – requisitos de ejecución:	
Tipo de Requerimiento:	
Éxito:	**No éxito:**

Descripción del caso de prueba:	
Resultado obtenido:	
Observaciones:	
Ejecutor del test case:	
Responsable del test case:	

Figura : Planilla utilizada para describir los casos de prueba.

Nombre de Caso de Prueba: Transformar la clase ActaFinObra a XML (#TC-CONT-AFO-0001)
Objetivo: Verificar que la trasformación a xml de la clase ActaFinObra sea correcta.
Pre – requisitos de ejecución: No aplica
Tipo de Requerimiento: Funcional

Éxito: *XML contienen todos los campos cargados*	**No éxito:** *Cualquier otra respuesta*
Descripción del caso de prueba:	
1. Crear una instancia de la clase. 2. Cargarle todos los campos. 3. Transformar la clase a xml. 4. Capturar Salida.	
Resultado obtenido:	**El esperado**
Observaciones:	**Ninguna**
Ejecutor del test case	Integrante del equipo que realizó la prueba
Responsable del test case	Integrante del

	equipo que desarrolló el JAXB

Figura : Planilla de Caso de prueba con el resultado “el esperado”

Nombre de Caso de Prueba: Remover elemento nulo a la lista ActasFinObra (#TC-CONT-AsFO-0006)	
Objetivo: Remover elemento nulo a la lista de la clase ActasFinObra, utilizando el método remove de la clase.	
Pre – requisitos de ejecución: Que se encuentre un elemento en la lista	
Tipo de Requerimiento: Funcional	
Éxito: *La función devolvió el valor boolean false*	**No éxito:** *Cualquier otra respuesta*
Descripción del caso de prueba:	
1- Utilizar el método “remove” de la clase ActasFinObra enviando la instancia de ActaFinObra. 2- Capturar Salida.	

Resultado obtenido:	**No esperado**
Observaciones:	**La función remove devuelve true sin importar la situación.**
Ejecutor del test case	Integrante del equipo que realizó la prueba
Responsable del test case	Integrante del equipo que desarrolló el JAXB

Figura : Planilla de Caso de prueba con el resultado “no esperado”.

A continuación, se hará una breve explicación de los elementos de la planilla.

- Nombre del caso de prueba: Describe brevemente que es lo que se está testeando y bajo qué condición. A esto se le agrega un código para poder identificarlo.
- Objetivo: Es el propósito a alcanzar en el caso de prueba, indicando que es lo que se va a testear.
- Tipo de requerimiento: Se indica si lo que se va a testear es un requerimiento funcional o no funcional.
- Pre – requisitos de ejecución: Es la condición que se debe cumplir antes de ejecutar en la prueba.
- Éxito: Resultado esperado del caso de prueba.
- No éxito: Cualquier otro resultado que se obtiene que difiere al de éxito.
- Descripción del caso de prueba: Son los pasos a seguir para ejecutar el caso de prueba que llevan al resultado esperado o no. Debe tener el mayor detalle posible para que luego se pueda repetir la prueba.
- Resultado obtenido: Es un comentario que indica si la respuesta obtenida fue un éxito. Los valores posibles son "el esperado" y "no esperado".
- Observaciones: Es un campo en el que se debe indicar cuales son las dificultades obtenidas al momento de ejecutar el caso.
- Ejecutor del test case: Integrante del equipo que realizó la prueba.
- Responsable del test case: Integrante del equipo que desarrolló lo que se está testeando.

6.2 Pruebas a nivel de Servicios Web

Se ejecutaron diferentes tipos de pruebas sobre los servicios web. Estos fueron sometidos ante diferentes situaciones y bajo distintas condiciones, a los efectos de evaluar el accionar de los mismos. El objetivo de las pruebas es detectar errores en forma temprana (previo a la utilización de los WS), a los efectos de determinar las circunstancias en las que se produce la anomalía o comportamiento incorrecto.

Para realizar las pruebas de los servicios web fue creado un cliente que utilizó los recursos RESTFul (ver Figura 24). Además, dichas pruebas se realizaron utilizando el framework JUnit.

JUnit [28] permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera.

```
//*****************************************************************************
//    ITipoDocumentoImpl
//*****************************************************************************

/** Permite la consulta por patrón de contención los ...13 lines */
public String consultarTipoDocumento(String patron, String idsesion) throws ClientErrorException {...11 lines }

/** Permite la consulta de un tipo de documento a partir del idtipodoc ...12 lines */
public String consultarTipoDocumentoByID(Integer idtipodoc, String idsesion) throws ClientErrorException {...11 lines }

/** Permite la registracion de Tipos de Documentos ...11 lines */
public String registrarTiposDocumento(SesionTipoDocumento requestEntity) throws ClientErrorException {...3 lines }

/** Permite la modificacion de Tipos de documentos ...11 lines */
public String modificarTiposDocumento(SesionTipoDocumento requestEntity) throws ClientErrorException {...3 lines }

/** Permite el borrado de Tipos de documentos ...11 lines */
public String borrarTiposDocumento(SesionTipoDocumento requestEntity) throws ClientErrorException {...3 lines }
```

Figura : Cliente del servicio web Tipo Documento.

Para documentar el testing de los WS se utilizó la planilla de casos de pruebas (ver Figura 21) antes mencionada. En la Figura 25 se observa el caso de prueba para el

WS consultarTipoDocumento. Al igual que en las pruebas a nivel de clase, estas eran remitidas al encargado de programar el WS testeado para la corrección del mismo.

Nombre de Caso de Prueba: Consultar Tipos documento por patrón con parámetros idsesion null (#TC-INSCWS-TDOC-0002)	
Objetivo: Probar la obtención de tipos de documentos a partir de un patrón de búsqueda usando el webservice /consultarTipoDocumento, pasando parámetros idsesion null	
Pre – requisitos de ejecución: Haber iniciado sesión con un usuario válido en Inscripciones, que posea permisos WS_INSCRIPCIONES_TDOCUMENTO_CONSULTAR	
Tipo de Requerimiento: Funcional	
Éxito: *Respuesta bajo el esquema XML MensajeIPAV con código "**AS-0002**"*	**No éxito:** *Cualquier otra respuesta*
Descripción del caso de prueba:	

Consultar webservice con los siguientes parámetros: java.lang.String patrón = Válido java.lang.String idsesion = Null Capturar respuesta	
Resultado obtenido:	Esperado
Observaciones:	Ninguna
Ejecutor del test case	Integrante del equipo que realizó la prueba
Responsable del test case	Integrant

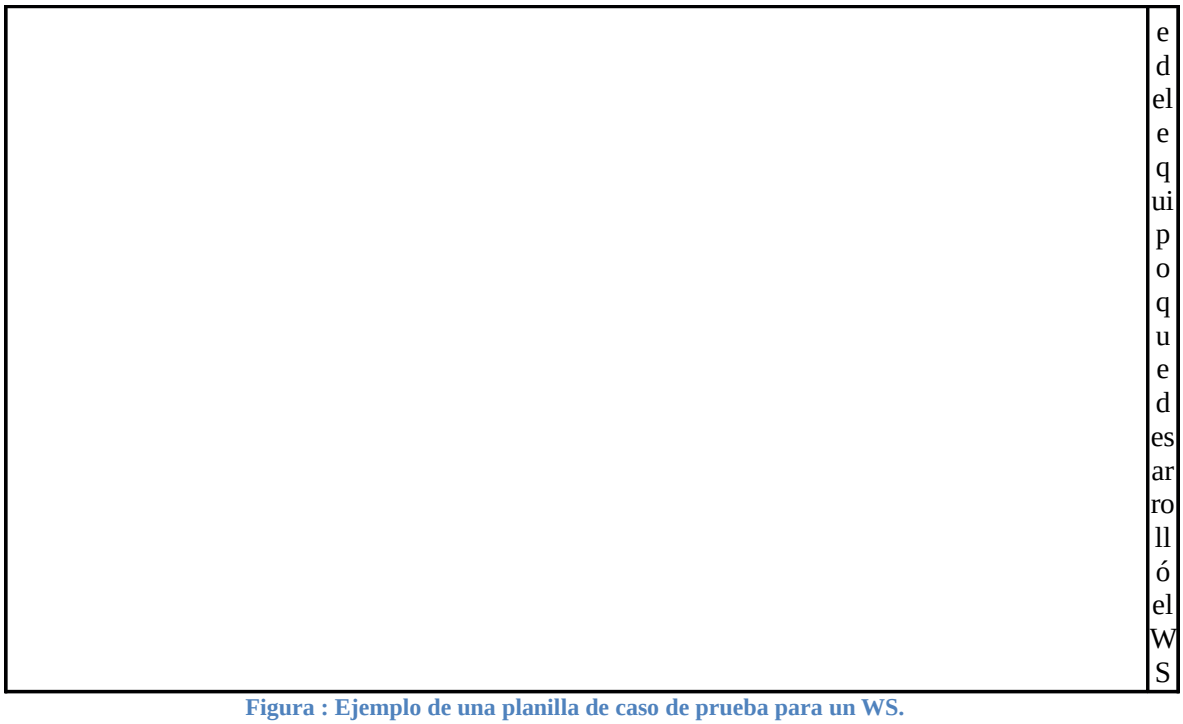

Figura : Ejemplo de una planilla de caso de prueba para un WS.

6.3 Pruebas a nivel de interface de usuario

Se ejecutaron diferentes tipos de pruebas sobre las pantallas del sistema. Estas fueron evaluadas de forma individual, teniendo en cuenta la funcionalidad esperada de las mismas. El objetivo de las pruebas fue detectar errores, previa integración de las mismas al sistema.

Para realizar las pruebas a la interface de usuario y su funcionalidad se evaluaron todas las opciones posibles que la pantalla proporcionaba. Luego los resultados obtenidos se documentaron en la planilla antes mencionada (ver Figura 21).

6.4 Prueba Integral del Sistema

La prueba integral del sistema tiene como objetivo evaluar el funcionamiento del módulo desarrollado (previo a la liberación del entregable del software) en base al circuito administrativo documentado. Se tiene en cuenta los procesos SPEM evidenciados y los casos de usos especificados para la evaluación de las pantallas, verificando la integración y acoplamiento de la misma en el circuito.

En este apartado se mostrará cómo se fueron verificando las pantallas de solicitar turno integradas al sistema, haciendo uso del proceso SPEM Solicitar Turno (ver Figura 26) y los casos de usos relacionados (ver Figura 27).

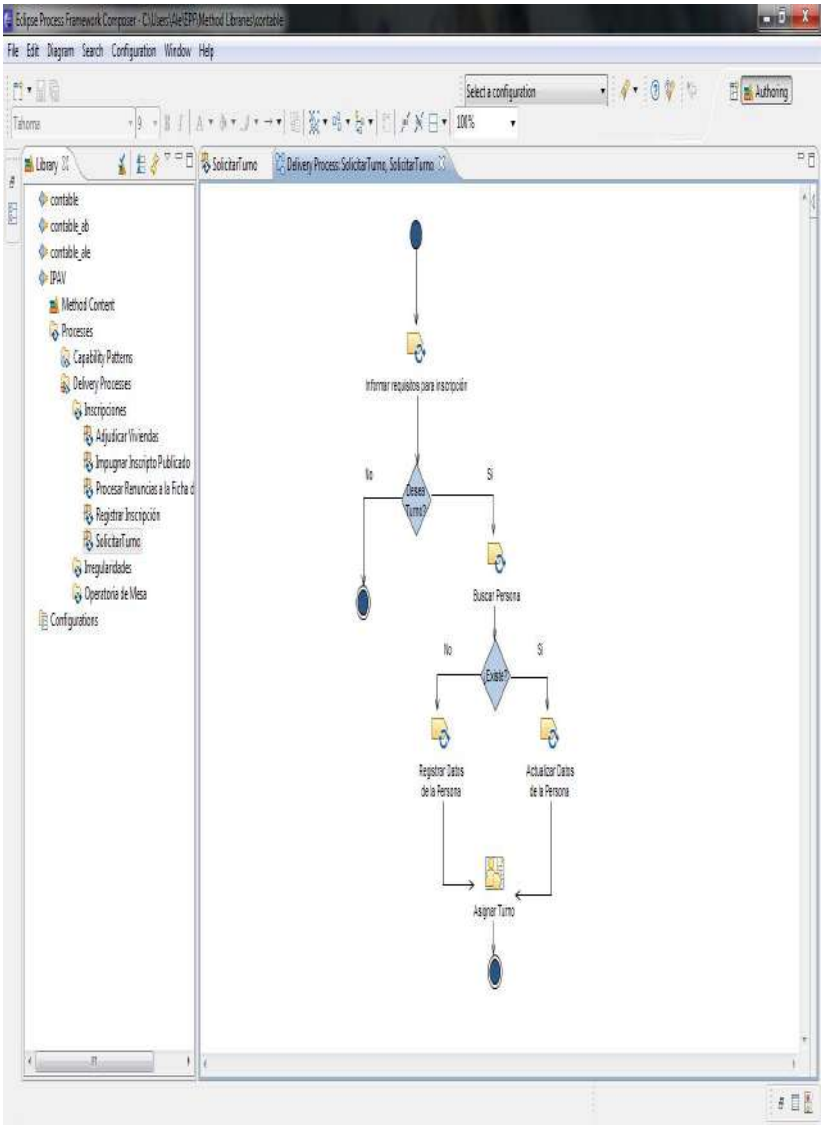

Figura : Modelo de actividad del proceso SPEM Solicitar Turno

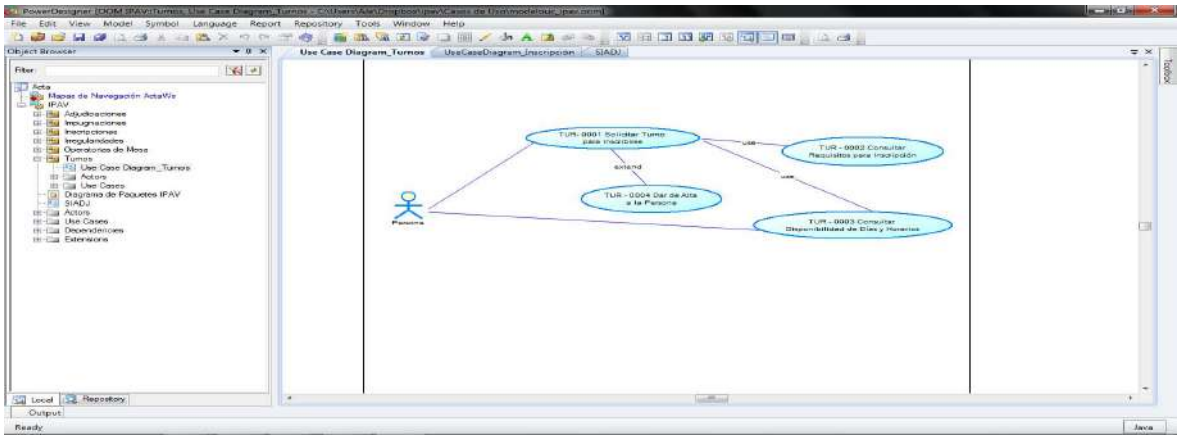

Figura : Modelo de casos de uso del proceso Solicitar Turno

En la Figura 28 se puede observar como concuerda la pantalla solicitar turno con el proceso SPEM documentado. Las tareas Buscar Persona y Registrar Datos de la Persona están desarrolladas en la pantalla con el botón Buscar y Nuevo Aspirante respectivamente. A su vez estas funcionalidades están especificadas en los casos de usos TUR – 0001 “Solicitar Turno para Inscribirse” (ver Figura 29).

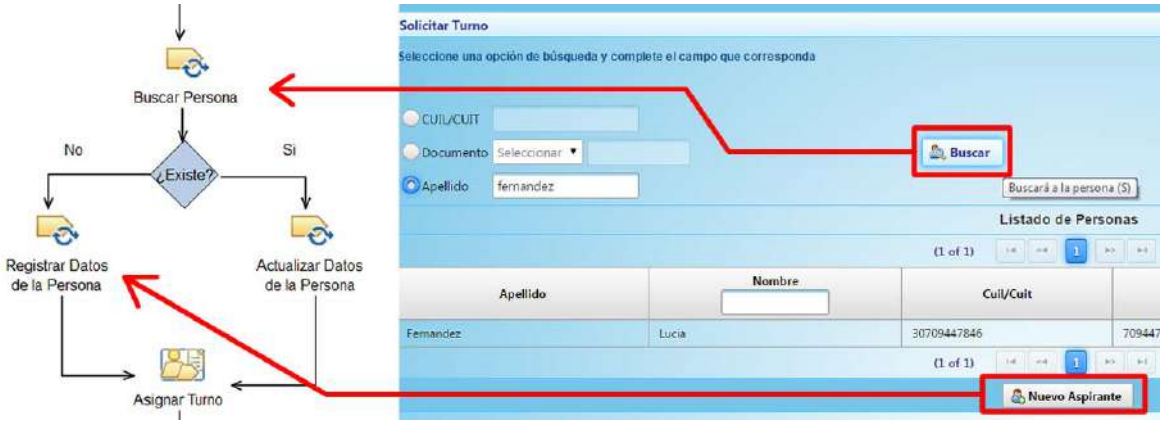

Figura : Comparacion del proceso con la pantalla Solicitar Turno 1

En la Figura 30, las funcionalidades de Actualizar Datos y Asignar Turnos son visibles una vez que se selecciona una persona del listado. También en este caso se puede apreciar que las tareas de Actualizar Datos de la persona y Asignar Turnos, se condicen con las pantallas desarrolladas.

Actividad	**Código**	**Nombre**
	TUR – 0001	**Solicitar turno para inscribirse**

Objetivo	**Informar a la persona la disponibilidad de turnos para inscribirse**
Pre-condiciones	**No aplica.**
Especificación de la actividad	1. **La persona se presenta en Mesa de Entradas del IPAV para solicitar un turno** 2. **Encargada de Turnos le informa de los requisitos para inscripción. Usa caso de uso TUR – 0002 Consultar Requisitos para Inscripción.** 3. **La Persona le informa los datos personales para que la Encargada de Turnos los busque en el Sistema.** 4. **Encargada de Turnos, busca a la persona a través del CUIT/CUIL ó DNI ó Apellido y Nombre.** 5. **Se encontró a la persona?** **a. Sí:** **a.i. Se debe consultar la disponibilidad de los turnos. Usa caso de uso TUR – 0003 Consultar Disponibilidad de Días y Horarios.** **a.ii. Continuar en paso 6.** **b. No:** **b.i. Se debe dar de alta a la persona en el sistema. Se extiende al caso de uso TUR – 0004 Dar de Alta a la Persona.** **b.ii. Volver al paso 4.** 6. **Una vez que se tiene toda la información acerca de los turnos, Encargada de Turnos, le informa del mismo a la persona.** 7. **Fin.**
Post-condiciones	**No aplica.**
Requerimientos no funcionales asociados	
Observaciones	

Figura : Especificación del caso de uso Solicitar Turno para Inscribirse

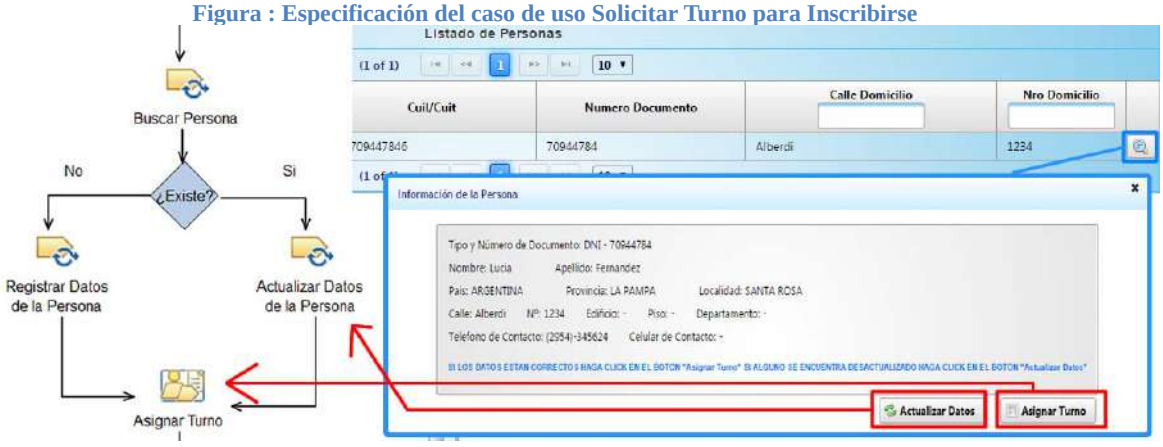

Figura : Comparación del proceso con la pantalla Solicitar Turno 2

7. Capítulo 7: Bitácora de Actividades

Para el desarrollo del sistema para el IPAV se realizaron distintos tipos de tareas, a continuación se explicaran brevemente en qué consiste cada una.

- **Estudio de los documentos y entrevistas**: Se analiza las entrevistas realizadas al personal junto con la reglamentación vigente y documentación anexa, para identificar posibles procesos, tareas a realizar, roles a desempeñar, documentación necesaria y áreas intervinientes dentro del sistema.
- **Modelado de procesos**: Se modela el proceso de negocio especificando las tareas intervinientes, los roles y la documentación. Para la tarea se realiza una breve descripción aclarando quién la realiza (rol) y qué documentación tiene como entrada o salida. Además, se realiza un diagrama de actividades mostrando la secuencia de las tareas.
- **Verificación de los procesos modelados**: Se realiza un seguimiento del proceso modelado, por otro integrante del grupo, verificando que este sea consistente con la información del relevamiento, y de ser necesario informar posibles correcciones.
- **Identificación y especificación de casos de uso**: Se listan los posibles casos de usos y los actores asociados que disparan el proceso. Se modela las relaciones entre ellos, se detalla el caso de uso, utilizando la planilla.
- **Programación de prototipos de interface gráfica**: Se crean los prototipos de interfaces de usuario, es decir, se modelan las distintas pantallas por las que eventualmente el usuario podría interactuar con el sistema. Se delimitan las condiciones que deben cumplirse para que el programa: a) muestre una interface en particular, b) se defina el intercambio de datos entre interface graficas (asociado con el diagrama de casos de uso), y c) se esboza el formato de la información a exponer.

- **Funcionalidades de interface gráfica**: Se desarrollan las distintas funcionalidades de la pantalla, de esta manera cuando el usuario ejecuta un comando de la misma éste funciona de acuerdo a lo esperado.

- **Tareas de testing sobre clases JAXB**: Se ejecutan diferentes tipos de pruebas unitarias sobre las clases que luego serán usadas como parámetros de los WS.

A continuación se detallarán las actividades realizadas por el autor durante la práctica en empresa, por lapsos temporales (semanales), con 20 horas por semana, desde el 27 de noviembre del 2014 hasta el 6 de marzo del 2015. Cabe destacar que la escritura de la presente tesina se realizó en paralelo, insumiendo 10 horas semanales adicionales.

7.1 [27-nov-2014; 28-nov-2014]

7.1.1 Programación de prototipos y funcionalidades de la interface gráfica

Durante esta semana realicé el prototipo de la interface gráfica y programé la funcionalidad de la siguiente operación dentro del sistema.

- Pre-adjudicados: Permite generar la lista de pre-adjudicados (inscripto que está en condiciones de recibir una vivienda) teniendo en cuenta la publicación de los inscriptos para determinada localidad. Además da lugar a que un pre-adjudicado renuncie a su puesto por algún motivo.

Esta pantalla con su funcionalidad desarrollada fue asignada mediante mantis a otro integrante del grupo de trabajo para que realice el testing de la misma. Luego, corregí los errores detectados.

7.2 [01-dic-2014; 05-dic-2014]

7.2.1 Programación de prototipos y funcionalidades de la interface gráfica

Durante esta semana realicé el prototipo de la interface gráfica y programé la funcionalidad de la siguiente operación dentro del sistema.

- Adjudicar Viviendas: Permite asignar una vivienda de forma definitiva a un pre-adjudicado (inscripto que está en condiciones de recibir una vivienda). Luego de que la vivienda fue sorteada entre un listado de pre-adjudicados.

Esta pantalla con su funcionalidad desarrollada fue asignada mediante mantis a otro integrante del grupo de trabajo para que realice el testing de la misma. Luego, corregí los errores detectados.

7.3 [09-dic-2014; 12-dic-2014]

7.3.1 Programación de prototipos y funcionalidades de la interface gráfica

Durante esta semana realicé el prototipo de las interfaces gráficas y programé las funcionalidades de las siguientes operaciones dentro del sistema.

- Consultar Adjudicados: Permite visualizar el listado de adjudicados teniendo en cuenta los filtros. Por cada adjudicado el sistema permite imprimir el acta de tenencia precaria, y modificar la modalidad de entrega de la casa.
- Operatorias de Cambio de Permuta (OCP): La pantalla permite visualizar el listado de permutas, los cuales se pueden filtrar por diferentes campos. Además, ofrece la posibilidad de visualizar en detalle la OCP, donde se muestra la información de las viviendas que se permutan, los datos de los respectivos grupos familiares y datos sobre ubicación de las viviendas. Dentro de la pantalla principal, se puede agregar una nueva OCP, anular una operatoria o modificar una operatoria ya existente.

Estas pantallas con sus funcionalidades desarrolladas fueron asignadas mediante mantis a otro integrante del grupo de trabajo para que realice el testing de las mismas. Luego, corregí los errores detectados.

7.4 [15-dic-2014; 19-dic-2014]

7.4.1 Programación de prototipos y funcionalidades de la interface gráfica

Durante esta semana realicé el prototipo de la interface gráfica y programé la funcionalidad de la siguiente operación dentro del sistema.

- Irregularidades: Permite visualizar el listado de Irregularidades (denuncias a viviendas que no cumplen con las normas del IPAV) teniendo en cuenta los distintos filtros, agregar una nueva irregularidad, anular una irregularidad o modificar una ya existente. Además, permite visualizar en detalle una irregularidad de interés.

Esta pantalla con su funcionalidad desarrollada fue asignada mediante mantis a otro integrante del grupo de trabajo para que realice el testing de la misma. Luego, corregí los errores detectados.

7.5 [22-dic-2014; 23-dic-2014 y 29-dic-2014; 30-dic-2014]

7.5.1 Depuración y mejora de código

En estos 4 días, finalizando el año, realice tareas de depuración y mejora del código generado. Agregando comentarios para su mejor documentación y optimizando las funciones.

7.6 [05-ene-2015; 09-ene-2015]

7.6.1 Programación de prototipos y funcionalidades de la interface gráfica

Durante esta semana realicé el prototipo de la interface gráfica y programé la funcionalidad de la siguiente operación dentro del sistema.

- Acta Constatación de Irregularidad: Permite visualizar el listado de Acta Constatación de Irregularidad (acta que se genera cuando personal del IPAV va hasta la vivienda a verificar la irregularidad) teniendo en cuenta los distintos filtros, agregar una nueva acta, borrar un acta o modificar una ya existente. Además, permite visualizar en detalle un acta de interés, donde

muestra un encabezado, un cuerpo donde poner lo que se observó y la fecha en que se hizo la visita.

Esta pantalla con su funcionalidad desarrollada fue asignada mediante mantis a otro integrante del grupo de trabajo para que realice el testing de la misma. Luego, corregí los errores detectados.

7.7 [12-ene-2015; 16-ene-2015]

7.7.1 Estudio de los documentos y entrevistas

Durante esta semana analicé las entrevistas y reglamentación vigente sobre el siguiente tema.

- OAFI: La asistencia financiera individual se otorga a propietarios de terrenos o viviendas ubicados en zonas urbanas o suburbanas con dotación de servicios básicos provenientes de red u otro sistema, para la construcción de soluciones habitacionales básicas. Estudié y analicé el reglamento de la operatoria de asistencia financiera individual.

7.8 [19-ene-2015; 23-ene-2015]

7.8.1 Estudio de los documentos y entrevistas

Durante esta semana analicé las entrevistas y reglamentación vigente sobre el siguiente tema.

- OAFI: La asistencia financiera individual se otorga a propietarios de terrenos o viviendas ubicados en zonas urbanas o suburbanas con dotación de servicios básicos provenientes de red u otro sistema, para la construcción de soluciones habitacionales básicas. Estudié y analicé las entrevistas realizadas al personal encargado de OAFI.

7.8.2 Verificación de los procesos modelados

Durante esta semana analicé los siguientes procesos modelados por otros integrantes del grupo de trabajo. Estos procesos estaban modelados con la herramienta

EFP Composer, mi tarea consistía en realizar una verificación de lo modelado en función a las entrevistas realizadas y la documentación relevada. Luego de revisar todo el proceso y de existir alguna posible inconsistencia le informaba las observaciones al compañero encargado de modelar dicho proceso.

- Cancelar OAFI: Proceso por el cual el titular del crédito solicita la cancelación del mismo, se analiza la posibilidad de pago, se autoriza el mismo y se acciona el levantamiento de la hipoteca.
- Paralizar Obra: Proceso por el cual el beneficiario podrá solicitar la paralización de la obra, siempre y cuando hayan acreditado de manera fehaciente la imposibilidad de continuar con la misma.
- Des-paralizar Obra: Proceso por el cual el beneficiario, una vez paralizada la obra y antes que se cumpla un año desde que se paralizo, podrá solicitar continuar con la misma.
- Registrar Acta de Medición: Proceso por el cual los inspectores cargan en el sistema los avances o no de obra, previa confirmación del beneficiario.
- Solicitar Desembolso: Proceso por el cual se solicita, previa inspección de la obra, el desembolso de la asistencia financiera.
- Reprogramar Obra: Proceso por el cual, si la obra con atraso superó el plazo de ejecución previsto en el cronograma, se solicita la reprogramación de la misma presentando un nuevo cronograma, sin que este exceda el límite máximo de tiempo. Para luego de verificar su factibilidad sea autorizado por el IPAV.
- Solicitar OAFI: Proceso por el cual el aspirante solicita la Operatoria de Asistencia Financiera Individual presentando los requisitos necesarios. Luego se analiza su factibilidad, se arma el expediente del mismo con toda la documentación presentada, y de ser aprobado, se solicita el desembolso de la asistencia financiera.

7.9 [26-ene-2015; 30-ene-2015]

7.9.1 Identificación y especificación de casos de uso

Durante esta semana analicé los procesos relacionados a OAFI, verificados la semana anterior, del cual obtuve un listado de los posibles casos de uso y de sus actores. Realicé junto a un compañero los diagramas de casos de uso identificando sus relaciones y especificando las actividades que se deben llevar a cabo.

Dentro de los procesos relacionados a OAFI se identificaron 32 casos de uso.

7.9.2 Verificación de los procesos modelados

Durante esta semana analicé el siguiente proceso modelado por otro integrante del grupo de trabajo. Estos procesos estaban modelados con la herramienta EFP Composer, mi tarea consistía en realizar una verificación de lo modelado en función a las entrevistas realizadas y la documentación relevada. Luego de revisar todo el proceso y de existir alguna posible inconsistencia le informaba las observaciones al compañero encargado de modelar dicho proceso.

- Solicitar Escrituración: Proceso por el cual el beneficiario solicita la baja de la hipoteca por parte del IPAV, previa cancelación de la deuda. En el mismo se realiza la nueva escritura de la obra finalizada.

7.10 [02-feb-2015; 06-feb-2015]

7.10.1 Estudio de los documentos y entrevistas

Durante esta semana analicé las entrevistas y reglamentación vigente sobre el siguiente tema.

- Certificaciones: Es el área donde se emiten los certificados de pago, también se realiza la confección del certificado de redeterminación de precios de las obras, los anticipos a las empresas y las órdenes de pago de los OAFI.

7.10.2 Modelado de procesos

En esta semana modelé el siguiente proceso con la herramienta EPF Composer. Me base en las entrevistas de certificaciones analizadas en la misma semana. Mi tarea

consistió, en base a las entrevistas, en definir las tareas, roles y documentación que pertenecen al proceso y modelar como se relacionan. En este caso otro integrante del grupo revisó mi trabajo informándome las inconsistencias hasta que el proceso quedó modelado adecuadamente.

- Reprogramar plazos de obras públicas: Proceso por el cual la empresa constructora solicita la reprogramación de la obra presentando los motivos por el cual se atrasó la construcción de la misma. Se analizó los motivos del atraso y la factibilidad de la reprogramación.

7.11 [09-feb-2015; 13-feb-2015]

7.11.1 Modelado de procesos en SPEM

En esta semana modelé el siguiente proceso con la herramienta EPF Composer. Me base en las entrevistas de certificaciones analizadas en la semana anterior. Mi tarea consistió, en base a las entrevistas, en definir las tareas, roles y documentación que pertenecen al proceso y modelar como se relacionan. En este caso otro integrante del grupo revisó mi trabajo informándome las inconsistencias hasta que el proceso quedó modelado adecuadamente.

- Registrar multa: Proceso por el cual se lleva a cabo la registración y el cálculo de las multas aplicadas a las empresas constructoras que presenten incumplimientos en sus plazos de obra.

7.11.2 Identificación y especificación de casos de uso

Durante esta semana analicé los procesos relacionados a Certificaciones que me tocaron modelar, de los cuales obtuve un listado de los posibles casos de uso y de sus actores. Realicé los diagramas de casos de uso identificando sus relaciones y especificando las actividades que se deben llevar a cabo. Luego integré los casos de usos que yo especifiqué con los especificados por mis compañeros.

Dentro de los procesos relacionados a certificaciones que me tocaron modelar identifiqué 7 casos de uso.

7.12 [18-feb-2015; 20-feb-2015]

7.12.1 Estudio de los documentos y entrevistas

Durante esta semana analicé la entrevista y reglamentación vigente sobre el siguiente tema.

- Recaudación: Es el área donde se emiten y se calculan las cuotas de las viviendas sociales, también se realizan las cancelaciones de viviendas o los créditos OAFI.

7.12.2 Modelado de procesos en SPEM

En esta semana modelé los siguientes procesos con la herramienta EPF Composer. Me basé en las entrevistas de recaudación analizadas en la semana. Mi tarea consistió, en base a las entrevistas, en definir las tareas, roles y documentación que pertenecen al proceso y modelar como se relacionan. En este caso otro integrante del grupo revisó mi trabajo informándome las inconsistencias hasta que el proceso quedó modelado adecuadamente.

- Recupero de Cuota: Proceso por el cual se registra el cobro de las cuotas, teniendo en cuenta lo informado por el organismo recaudador y donde se genera el listado de cuotas a cobrar para el mes siguiente.
- Refinanciar plan de cuotas: Proceso por el cual se lleva a cabo el refinanciamiento de la deuda debido a un cambio en la modalidad de cobro o un ajuste en el interés de la misma.

7.13 [23-feb-2015; 27-feb-2015]

7.13.1 Estudio de los documentos y entrevistas

Durante esta semana analicé la entrevista y reglamentación vigente sobre el siguiente tema.

- Recaudación: Es el área donde se emiten y se calculan las cuotas de las viviendas sociales, también se realizan las cancelaciones de viviendas o los créditos OAFI.

7.13.2 Modelado de procesos en SPEM

En esta semana modelé los siguientes procesos con la herramienta EPF Composer. Me basé en las entrevistas de recaudación analizadas en la semana. Mi tarea consistió, en base a las entrevistas, en definir las tareas, roles y documentación que pertenecen al proceso y modelar como se relacionan. En este caso otro integrante del grupo revisó mi trabajo informándome las inconsistencias hasta que el proceso quedó modelado adecuadamente.

- Generar Plan de Cuota Inicial: Proceso por el cual se calcula el valor de la vivienda, teniendo en cuenta los certificados de obra, y donde se emiten las mismas para su cobro.

Debido a que se modelaron los diagramas de procesos en forma simultánea se utilizaron diferentes archivos para no sobrescribir ningún modelo. Por esta razón, como tarea especial, me asignaron la integración de los diagramas desarrollados por el equipo de trabajo en un solo archivo.

7.14 [02-mar-2015; 06-mar-2015]

7.14.1 Tareas de Testing sobre clases JAXB

Durante esta semana realicé y documenté las pruebas unitarias de varias clases Java que van a ser utilizadas como parámetros en los WS del módulo contable. Estas clases fueron creadas por otro integrante del equipo de trabajo. Mi tarea consiste en evaluar la visibilidad de los atributos, los métodos de acceso a los mismos, los constructores de las clases y verificar la transformación a XML de las clases en forma correcta. La documentación generada, en caso de tener algún error, fue remitida al creador de la clase.

8. Capítulo 8: Conclusiones

En este informe se han presentado las actividades que desarrollé en la práctica en la empresa IPAV, donde me desempeñé como pasante desde Octubre de 2013. La principal actividad realizada consistió en colaborar en el desarrollo y puesta en marcha de un sistema de Inscripciones y Adjudicación para la citada empresa y posteriormente el diseño y desarrollo del Sistema Contable, el cual está en proceso de inicio a la fecha. Esto permitió cumplir satisfactoriamente los objetivos planteados en el trabajo final de grado y llevar gradualmente las funcionalidades del IPAV hacia un sistema de información integrado, interoperable, seguro y extensible.

Uno de los primeros aspectos que se desarrolló en el marco del nuevo sistema de gestión, fue la implementación de un sistema de seguridad basado en la arquitectura orientada al servicio, que brindó la posibilidad de tener un acceso controlado a los datos, aplicar en línea políticas de seguridad, accediendo a los recursos en forma uniforme para todas las aplicaciones actuales y futuras del organismo. De esta manera se promueve la interoperabilidad de los servicios a terceros, y permite aplicar las políticas de seguridad en forma instantánea y basada en los procesos SPEM.

Usar el sistema integrado de seguridad planteado como servicio para todas las aplicaciones, permitió el desarrollo de una arquitectura interoperable y segura, que facilitó la implementación del sistema de Inscripciones y Adjudicación y la posterior integración de nuevos sistemas que se desarrollen para el instituto en el futuro, por ejemplo, el módulo de gestión de recupero, de certificaciones, de inventario y de operatoria de asistencia financiera.

Luego de recorrer todas las etapas desde el inicio del proyecto, he adquirido una enriquecedora experiencia en el diseño y desarrollo de sistemas de mediana complejidad, en entornos distribuidos, basado en servicios y con fuertes restricciones de seguridad, incorporando interoperabilidad y extensibilidad a los mismos. Además, esta práctica contribuyó a ampliar y profundizar los conocimientos adquiridos en la carrera de Ingeniería en Sistemas, a la vez que completó mi formación, brindándome experiencia laboral en el desarrollo de la profesión y trabajo en equipo.

Como trabajo a futuro se prevé seguir participando en el desarrollo del nuevo sistema de inventarios, certificaciones, gestión de recupero, y operatorias de asistencia financiera del IPAV, contribuyendo a su armado a través del modelado de procesos, actualización de requerimientos, prototipación, generación de mapas de navegación, desarrollo de los diferentes niveles de testing, documentación, implementación e implantación.

Referencias

[1] Hacia una Arquitectura Orientada a Servicios en el Instituto Provincial Autárquico de Vivienda de La Pampa – Mario José Diván, María Laura Sánchez Reynoso, Marcos Alejandro Fredes, Ana Oddone, Bruno Sebastián Cavallo and Alejandro Maximiliano Martínez. En Anales del 8º Simposio Argentino de Informática en el Estado, 43 JAIIO. ISSN: 1851-2526.

[2] A Data Monitoring Strategy based in Snapshots for the Score Calculation in the Housing Distribution – Mario José Diván, María Laura Sánchez Reynoso, Marcos Alejandro Fredes, Ana Oddone, Bruno Sebastián Cavallo and Alejandro Maximiliano Martínez. En XL Conferencia Latinoamericana en Informática (CLEI 2014).

[3] Boehm, B.: A View of 20th and 21st Century Software Engineering. In: 28th international conference on Software engineering, pp. 12--29. Shanghai, China (2006).

[4] Rubin, K (2012) “Essential Scrum: A Practical Guide to the Most Popular Agile Process”. Addison-Wesley.

[5] Object Management Group (2008) “Software & Systems Process Engineering Meta-Model Specification”.

[6] Rumbaugh, J, Jacobson, I & Booch, G (2000) “El lenguaje unificado de Modelado. Manual de Referencia”. Addison-Wesley.

[7] Erl, T (2007) SOA: Principles of Service Design. Prentice Hall.

[8] Kurniawan, B (2014) “Java 7: A Comprehnsive Tutorial”. BrainySoftware.

[9] https://netbeans.org/downloads/7.4/ [Último Acceso: 20/04/15 17:30].

[10] Gorman, W (2009) “Pentaho Reporting 3.5 for Java Developers”. Packt Publishing.

[11] http://community.pentaho.com/projects/reporting/ [Último Acceso: 20/04/15 17:30].

[12] Postgresql Global Development Group (2011) “PostgreSQL 9.0 Official Documentation - Volume II. Server Administration”. Fultus Corporation.

[13] http://www.postgresql.org/ftp/pgadmin3/release/v1.20.0/win32/ [Último Acceso: 20/04/15 17:30].

[14] https://eclipse.org/epf/downloads/tool/tool_downloads.php [Último Acceso: 20/04/15 17:30].

[15] Khare, T. (2012) “Apache Tomcat 7 Essentials”. Packt Publishing.

[16] Civici, C (2014) “Primefaces User’s Guide 5.0”. PrimeTek Informatics. Primera Edición.

[17] http://www.primefaces.org/downloads [Último Acceso: 20/04/15 17:30].

[18] IBM (2014) “RESTFul Web Services: The basics”. Fuente:https://www.ibm.com/developerworks/webservices/library/ws-restful/ [Último Acceso: 10/01/15 16:00].

[19] Richardson, L & Ruby, S. (2007) “RESTful Web Services. Web services for the real world”. O’Reilly Media.

[20] Kretsis, A., Kokkinos, P., Christodoulopoulos, K. & Varvarigos, E. (2013) “Mantis: Optical network planning and operation tool”. In proceedings of 15th International Conference on Transparent Optical Networks (ICTON). IEEE. Pp. 1-4.

[21] Scott Chacon (2009) “Pro Git”. Apress. Primera Edición. Fuente: http://git-scm.com/book/es/v1 [Último Acceso: 22/02/15 15:40].

[22] Software Requirements. Karl. E.Wiegers. Microsoft Press, 1999. pp 153-162

[23] Bai, Y (2011) “Practical Database Programming with Java”. IEEE Press & Wiley

[24] Erl, T, Roy, S, Thomas, P & Tost, a (2014) SOA with Java: RealizingSerive-Orientation with Java Technologies. Prentice-Hall/Pearson PTR.

[25] Hao, J., Lo, A., Ozyer, T. & Alhajj, R (2005) “XML views based approach for Web services”. In proceeding of International Conference on Information Reuse and Integration (IRI). IEEE. Pp. 458-463.

[26] Burns, E. & Schalk, C. (2010) “JavaServer Faces 2.0, The Complete Reference”. McGraw Hill Professional.

[27] Geary, D (2002) “Core JSTL: Mastering the JSP Standard Tag Library”. Prentice-Hall.

[28] Cheng-hui, H. & Huo Yan, C. (2005) “A semi-automatic generator for unit testing code files based on JUnit”. In proceedings of IEEE International Conference on Systems, Man and Cybernetics. Vol.1 pp. 140-145.