Universidad del Bío-Bío

Facultad de Ciencias Empresariales
Departamento de Sistemas de Información

Profesor Guía: Manuel Crisosto

UNIVERSIDAD DEL BÍO-BÍO

"Sistema de apoyo administrativo para la canalización de ayudas rurales de la I. Municipalidad de Nueva Imperial"

Orlando Alarcón Gómez

Concepción - Invierno 2010

Índice general

Introducción

Los sistema informáticos, sin duda, ya están presentes en todos los ámbitos del día a día, cobrando un rol preponderante sobre todo en empresas y organizaciones en donde la información necesita ser controlada de manera rápida y eficiente, por ello se hacen indispensables en instituciones que manejen grandes cantidades de datos o que posean un masivo flujo de información.

Es así como la Ilustre Municipalidad de Nueva Imperial toma la decisión de automatizar el proceso de recepción de ayudas rurales, lo que implica la realización de un proyecto que de como resultado un sistema que se ajuste a sus necesidades.

El presente informe detalla los pasos realizados y las decisiones tomadas en el proyecto en cuestión, esta dividido 7 en capítulos que son los siguientes:

Capítulo 1 - **Descripciones preliminares:** En este capítulo se detallan todos los aspectos básicos para comprender el proyecto (como el entorno local), y el enfoque sobre la problemática.

Capítulo 2 - **Descripción del proyecto:** En este capítulo se establece la solución y las pautas que regirán el desarrollo del proyecto como la metodología a ocupar, además se acotan los aspectos más básicos del sistema.

Capítulo 3 - **Requerimientos:** En este capítulo se formalizan los requisitos que la I. Municipalidad de Nueva Imperial exige sobre el sistema.

Capítulo 4 - **Herramientas utilizadas:** En este capítulo se describen la herramientas más importantes escogidas para el diseño e implementación del sistema.

Capítulo 5 - **Estudio de Factibilidad:** En este capítulo se realiza un análisis para determinar la viabilidad del proyecto.

Capítulo 6 - **Diseño:** En este capítulo se presentan los diagramas más importantes necesarios para la implementación del sistema.

Capítulo 7 - **Pruebas, auditoría y respaldos:** En este capítulo se listan las pruebas realizadas sobre el sistema, y se analizan los aspectos de seguridad.

Capítulo 1

Descripciones preliminares

1.1. Introducción

El presente capítulo incorpora al lector al entorno local y a la organización misma en la cual se desarrolla éste proyecto, ofreciendo información relevante como la ubicación, composición y funciones que se desarrollan en la organización, por nombrar algunas. para luego detallar el problema encontrado y el conjunto de posibles soluciones exploradas para resolver el mismo.

1.2. Descripción del entorno

1.2.1. Breve historia de Nueva Imperial

Nueva Imperial fue fundada en 1551 por Pedro de Valdivia bajo el nombre de La Imperial y constituyó uno de los centros más importantes del Chile colonial; quedó totalmente deshabitada en 1599 por los ataques de indígenas araucanos, si bien retomó a comienzos del siglo XX su actual nombre en honor a la antigua ciudad, fue considerada una de las ciudades más importantes de la región durante muchos años, su característica primordial es la peculiar forma de pintar las viviendas mediante adornos, por lo que es llamada 'la ciudad acuarela'.[1]

1.2.2. Ubicación

Nueva Imperial, se localiza en territorios occidentales de la Depresión Intermedia y las últimas estribaciones montañosas del Sur de la cordillera de Nahuelbuta; a unos 35 kilómetros al Oeste de la ciudad de Temuco, Novena Región, Chile (Figura 1.1).

Figura 1.1: Ubicación de Nueva Imperial

1.2.3. Población

Según datos del Censo de 1992 se constató un saldo migratorio neto de 3.420 personas, siendo la región receptora principal la Metropolitana (51,5 %), siguiéndole la VIII (17,3 %) y la X (14,8 %). Según género las mujeres encabezan el porcentaje (51,9 %). En cuanto a los inmigrantes de la comuna de Nueva Imperial, tres son las comunas emisoras que presentan mayores porcentajes: la Metropolitana (50.9 %); la Octava (13.3 %); y la Décima (10.8 %). Según estimaciones realizadas por UFRO, INE, FII, PAESMI y CELADE sobre un Censo aplicado a comunidades indígenas en 1988, arrojan como resultado que la comuna tuvo una emigración de 2.639 personas entre 1982 y 1992, de las cuales el 59.2 % correspondieron a mujeres, lo que es coherente con los estudios realizados al respecto. Enfocado el estudio al área rural y urbana, esta última obtuvo una inmigración de 670 personas, mientras que 3.309 sujetos salieron de zonas rurales.[2]

Datos geográficos y censales (Proyección Estimada 2009) [3]

Densidad de Población por Km^2:	43,40
Población Comunal Estimada para el Año (por el INE):	31.790
Población Masculina Estimada para el Año (por el INE):	16.237
Población Femenina Estimada para el Año (por el INE):	15.553
Porcentaje de Población Rural:	39,19
Porcentaje de Población Urbana:	60,81
Superficie Comunal (km^2):	732,53
Porcentaje Población Comunal en Relación a la Población Regional:	3,36
Provincia a la que Pertenece la Comuna:	Cautín
Región a la que Pertenece la Comuna:	Región de la Araucanía

1.3. Descripción de la organización

1.3.1. Antecedentes generales de la municipalidad

Alcalde:	Manuel Adolfo Salas Trautmann
Partido Político:	PDC
Pacto:	Concertación Democrática
Dirección:	Arturo Prat Nro 65
Teléfono:	(45) 918696 - 918697 - 918704
Fax:	(45) 918695
Web:	www.nuevaimperial.cl
E-mail:	municipalidad@nuevaimperial.cl

[3]

1.3.2. Composición de la municipalidad

Entre los principales cargos que conforman la I. Municipalidad de Nueva Imperial se encuentran:

- **Alcalde** Manuel Adolfo Salas Trautmann.
- **Administrador municipal** Jaime Alberto Jofre Arevalo

concejo municipal compuesto por:

Nombre	**Partido político**	**Pacto**
José Turra Melillan	PS	Concertación Democrática
Juan Constanzo Matamala	PDC	Concertación Democrática
Rodrigo Ceferino Fabres Suarez	ILC	Concertación Democrática
Xenia Morales Araneda	RN	Alianza por Chile
Domingo Urbina Aravena	RN	Alianza por Chile
Adán Ancamil Quintremil	ILF	Concertación Progresista

1.3.3. Organigrama

A continuación se observa en la figura 1.2 la estructura interna de la I. Municipalidad de Nueva Imperial.

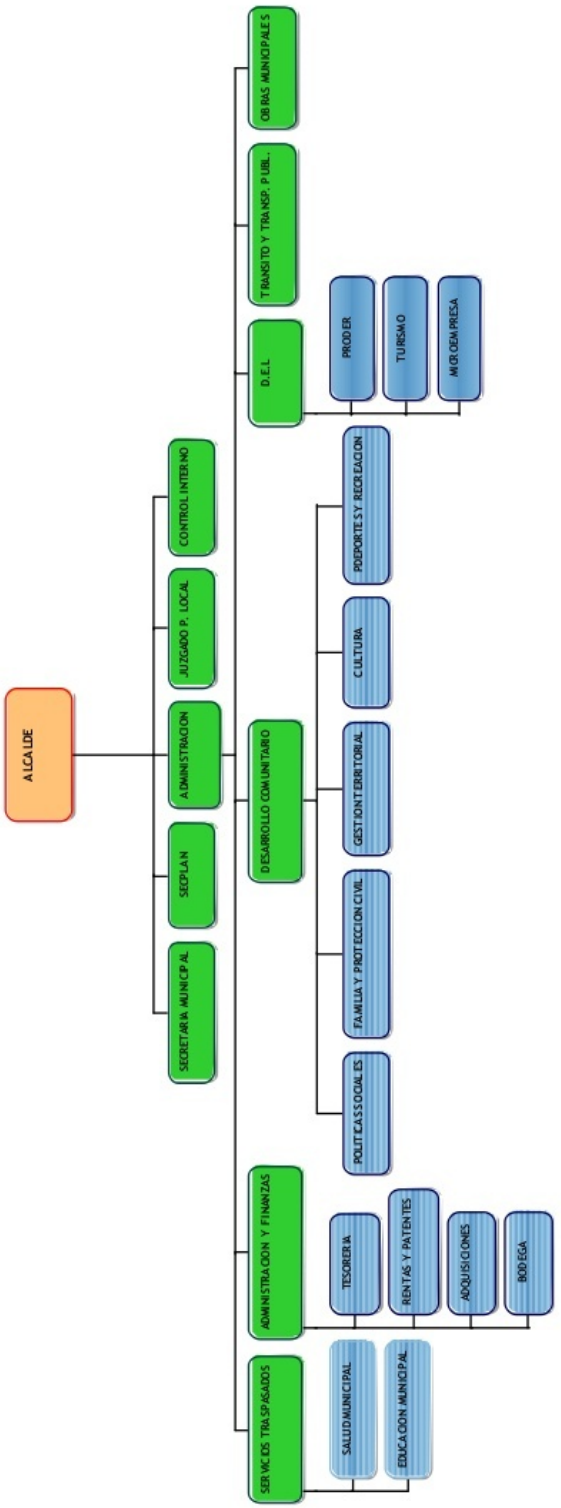

Figura 1.2: Organigrama de la I. Municipalidad de Nueva Imperial [2]

1.4. Área de coordinación (departamento de informática)

Tiene la función de elaborar y transformar la información que proporcionen las cifras numéricas, en datos útiles y oportunos para la toma de decisiones por parte del Administrador Municipal, de quien depende. Además tiene la función de planificar y asesorar a las unidades municipales en todo lo que respecta a la modernización y sistematización computacional. [4]

- Recopilar, procesar y entregar la información estadística comunal, a las unidades que lo requieran.
- Elaborar toda la información necesaria para la confección del Plan de Desarrollo Comunal, los programas y proyectos municipales.
- Interpretar y dar a conocer la información estadística que se le requiera.
- Asesorar y apoyar a las diferentes unidades que conforman la Secretaría Comunal de Planificación, en la aplicación de herramientas estadísticas y en la entrega de antecedentes de esta materia.
- Centralizar, recopilar y mantener datos estadísticos nacionales, regionales o comunales, necesarios para la Municipalidad, transformándolos en datos útiles para la toma de decisiones.
- Elaborar cuadros estadísticos con la información ingresada y procesada por la Unidad.
- Mantener actualizada la carta de situación comunal por sectores.
- Elaborar planes de sistematización de los procedimientos de las Distintas Unidades Municipales, en conjunto con las respectivas unidades.

1.5. Descripción área de estudio

El área de estudio comprende a **todos los departamentos** de la I. Municipalidad de Nueva Imperial. ya que el proceso de recepción de solicitudes los involucra de manera colaborativa.

1.6. Descripción situación actual

En la actualidad los municipios rurales cumplen un rol muy específico de entrega de ayudas municipales a la comunidad (ayuda social, entrega de animales, vivienda, terrenos, entre otros), el trámite comienza cuando el interesado se acerca a la municipalidad acudiendo ante alguna secretaria, le presenta la solicitud de ayuda y ésta lo deriva ante el encargado del área en cuestión, para ello se puede fijar una cita si se requiere (Figura 1.3). Ahora, una vez realizado todo ésto, es probable que según vaya avanzando la petición, el interesado sea derivado a distintos departamentos o encargados, así, se crea una secuencia o historial de la petición, este proceso no está automatizado por lo que no existe un seguimiento de las peticiones, el papeleo es completamente a mano lo que dificulta encontrar oportunamente la información sobre todo si la petición lleva un historial muy largo, ésto trae como consecuencia una falta de coordinación entre los encargados pudiéndose darse casos de colisiones de citas, además nada impide al solicitante volver a acudir a la municipalidad solicitando nuevamente la misma ayuda, debido a lo descrito anteriormente.

También cabe señalar que el alcalde suele requerir cierta información que se puede obtener de estas solicitudes como por ejemplo, ¿Qué sectores de la comuna han recibido efectivamente ayuda municipal? o

¿Qué departamentos o áreas son los mas solicitados por la comunidad?, entre otras preguntas, todo esto con el objetivo de idear estrategias políticas, ya que por ejemplo, sabiendo los sectores mas beneficiados, se puede crear una ruta de campaña o también viendo los sectores menos beneficiados se pueden idear nuevas formas distribución para acercar las ayudas a la comunidad.

Figura 1.3: Proceso de recepción de solicitudes

1.7. Descripción del problema

La problemática se resume en los siguiente puntos:

- El lento flujo de información.
- La poca oportuna consulta de datos.
- El prácticamente nulo seguimiento del historial.
- La incapacidad de determinar si un solicitante ya pidió anteriormente una solicitud similar.

Lo que conlleva a tener mas solicitudes canceladas o incompletas.

1.8. Alternativas de solución

El área de la administración de ayudas rurales es un área poco abordada y en la actualidad no existen programas que automaticen estas tareas. Es por ello que las alternativas de solución se ven reducidas al desarrollo a medida de las mismas (en vez de la compra de un paquete de software, por ejemplo). Por lo descrito, se expondrán a continuación 3 eventuales desarrollos como solución al problema planteado en el punto anterior.

1.8.1. Opción A

Descripción (Sistema de escritorio)

Esta solución plantea el desarrollo un sistema de escritorio, esto implica desarrollarlo bajo el sistema operativo Microsoft Windows XP, ya que es el sistema preponderante en los PC's de los encargados.

Ventajas

- El programa corre en forma nativa en el S.O. disminuyendo tiempos de respuesta.

Desventajas

- Herramientas de desarrollo muy específicas (framework .NET, Visual Studio, SQL Server)
- Desarrollado para plataforma específica.
- Costos por licencias de las herramientas de desarrollo.

1.8.2. Opción B

Descripción (Aplicación WEB)

Esta solución plantea el desarrollo de una aplicación WEB de acceso restringido, montado sobre la red local, con posibilidades a futuro de permitir accesos desde equipos ubicados fuera de la municipalidad, a través de Internet.

Ventajas

- Amplias opciones de implementación, ya que los lenguajes para el desarrollo dinámico de páginas WEB son variados (PHP, JSP, ASP, Ruby).
- Permite de manera muy sencilla el acceso desde el exterior, ya que el usuario sólo necesitaría un navegador Web.
- Sistema multiplataforma.

Desventajas

- Dependiente de Javascript [1]

[1] Aunque es cierto que la aplicación sea fuertemente dependiente de Javascript, todos los navegadores modernos vienen con esta característica habilitada, debido a la alta demanda que posee Javascript en los sitios sociales WEB 2.0 como facebook, twitter, por nombrar algunos.

1.8.3. Opción C

Descripción (Hojas de Cálculo)

Esta solución plantea el desarrollo de una mini aplicación basada en una planilla electrónica tipo, de Microsoft Excel, con programación en macros y Visual Basic 6 embebido.

Ventajas

- Rápida implementación.
- Archivos de poco tamaño.

Desventajas

- Informes poco personalizables.
- Escalabilidad limitada.
- Problemas para replicar la información a otros documentos.
- Difícil mantención.

Capítulo 2

Descripción del proyecto

2.1. Introducción

El presente capítulo trata sobre la descripción del proyecto en sí mismo, tomando como base la solución escogida de las propuestas en el capítulo anterior, elección que será abordada y justificada en breve. además se listan los objetivos tanto del proyecto como del sistema, la metodología utilizada, y el cronograma de actividades, estimando las principales tareas y fechas de los entregables (hitos).

2.2. Solución escogida

De las 3 posibles soluciones listadas en el capítulo anterior, se llego a la determinación de optar por la segunda (Opción B), esto por los motivos que se enumeran a continuación

1. La opción A si bien es una solución esperable a este tipo de sistemas, no se comporta de buena forma al momento de permitir accesos externos como lo son los accesos desde fuera de la red local a la Municipalidad, además:

   - El hecho de implementarlo en VB.NET implica mayores gastos de licencias.
   - Se limita su ejecución a 1 sola plataforma.
   - Para que lo ocupen otros usuarios, es Obligatorio realizar el proceso de instalación y posterior mantención, en cada equipo que ocupará el sistema, lo que provoca demoras extras.

2. La opción C es una buena solución pero para aplicaciones con menos exigencias y de menor envergadura, que no manejen tipos de acceso ni tanta cantidad de datos, puesto que se haría obligatorio compartir este tipo de archivos (.xls) lo que formaría un cuello de botella entre quienes deseen ocupar la planilla, o en su defecto, replicar la información en diferentes archivos o fuentes, haciendo prácticamente imposible una sincronización entre ellos.

3. La opción B en cambio se comporta bien en todos los casos esperables mencionados anteriormente lo que sumado a la facilidad técnica de utilización (sólo requiere un navegador WEB por parte del usuario cliente) y la facilidad de controlar los accesos en la red local, hacen de esta opción la mejor dentro de las listadas.

Como la opción elegida es una aplicación WEB, el lenguaje a ocupar será PHP5 esto debido a la gran comunidad existente, numerosos manuales/tutoriales y excelentes frameworks que facilitan aún más el desarrollo de aplicaciones (esto último es una de la grandes razones de optar por PHP5).

2.3. Sobre el proyecto

El proyecto nace de la necesidad de la I. Municipalidad de nueva Imperial, en mejorar el actual sistema de recepción y manejo de solicitudes de ayudas de la comunidad.

A continuación se presentan los objetivos del proyecto, basados en la solución elegida como propuesta, para la resolución del problema descrito en capítulo anterior.

2.3.1. Objetivos generales

"Desarrollar un sistema de apoyo administrativo para la canalización de ayudas rurales a la comunidad de la I. Municipalidad de Nueva Imperial, para agilizar las labores municipales dando un mayor control sobre el proceso reduciendo los tiempos de búsqueda de información"

2.3.2. Objetivos específicos

- Reformular o replantear, si es necesario, el sistema actual de canalización de ayudas.
- Explorar la posibilidad de integrarlos con la API de google maps o algún sistema de mapas similares.
- Investigar tecnologías Web avanzadas como ajax, uso de frameworks PHP para metodologías ágiles, utilización de ORM.
- Aprender e incorporar en la práctica los conceptos investigados.
- Implementar la solución elegida.

2.3.3. Metodología a utilizar

La metodología elegida para desarrollar este proyecto es el modelo iterativo incremental, es por ello que para comprender las ventajas, desventajas y cuando es conveniente su uso, se cita a continuación un capítulo del autor Ian Sommerville de su libro la ingeniera del software, quien explica estas características y detalla el modelo iterativo incremental.

El desarrollo incremental, implica producir y entregar el software en incrementos más que en un paquete único. Cada iteración del proceso produce un nuevo incremento del software. Las dos ventajas principales de adoptar un enfoque incremental para el desarrollo del software son:

- **Entrega acelerada de los servicios del cliente**. En los incrementos iniciales del sistema se pueden entregar las funcionalidades de alta prioridad para que los clientes puedan aprovechar el sistema desde el principio de su desarrollo. Los clientes puedan ver sus requerimientos en la práctica y especificar cambios a incorporar en entregas posteriores del sistema.
- **Compromiso del cliente con el sistema**. Los usuarios del sistema tienen que estar implicados en el proceso de desarrollo incremental debido a que tienen que proporcionar retroalimentación sobre los incrementos entregados al equipo de desarrollo. Su participación no sólo significa que es más probable que el sistema cumpla sus requerimientos, sino que también los usuarios finales del sistema tienen que hacer un compromiso con él y conseguir que éste llegue a funcionar.

En la figura 2.1 se ilustra un modelo de proceso general para el desarrollo incremental. Observe que las etapas iniciales de este proceso se centran en el diseño arquitectónico. Si no se considera la arquitectura al principio del proceso, es probable que la estructura general del sistema sea inestable y se degrade conforme se entreguen nuevos incrementos.

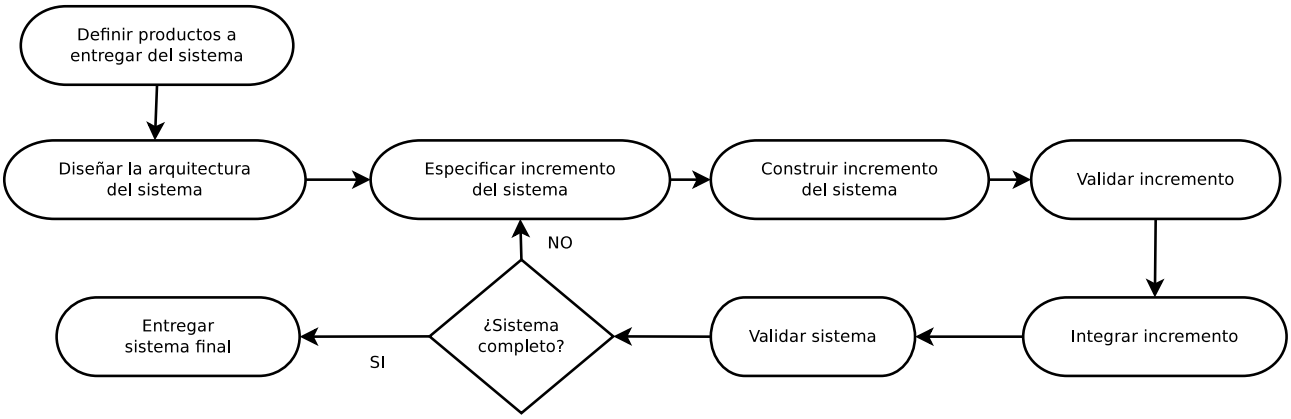

Figura 2.1: Proceso de desarrollo iterativo incremental

El desarrollo incremental del software es un enfoque mucho mejor para el desarrollo de la mayoría de los sistemas de negocio, comercio electrónico y personales porque refleja el modo fundamental al que todos nosotros tendemos al resolver problemas. Rara vez encontramos una solución completa a un problema por adelantado, peor nos movemos hacia una solución en una seria de pasos, dando marcha atrás cuando nos damos cuenta de que hemos cometido un error [18].

Sin embargo, puede haber verdaderos problemas con este enfoque, particularmente en las grandes compañías con procedimientos bastantes rígidos y en organizaciones donde el desarrollo del software normalmente se subcontrata con un contratista exterior. Los principales problemas con el desarrollo iterativo y la entrega incremental son:

- **Problemas de administración**. Las estructuras de administración del software para sistemas grandes se diseñan para tratar con modelos de proceso del software que generan entregas periódicas para evaluar el progreso. Los sistemas desarrollados incrementalmente cambian tan rápido que no es rentable producir una gran cantidad de documentación del sistema. Además. el desarrollo incremental muchas veces puede requerir el uso de tecnologías desconocidas para asegurar una entrega más rápida del software. A los administradores puede resultarles difícil utilizar el personal existente en los procesos de desarrollo incremental puesto que carecen de las habilidades requeridas.
- **Problemas contractuales**. El modelo contractual normal entre un cliente y un desarrollador de software se basa en la especificación del sistema. Cuando no existe tal especificación, puede ser difícil diseñar un contrato para el desarrollo del sistema. Los clientes pueden estar descontentos con un contrato que simplemente pague a los desarrolladores por el tiempo invertido en el proyecto, ya que puede conducir a que el sistema se desarrolle lentamente y se sobrepase el presupuesto; los desarrolladores probablemente no aceptarán un contrato con precio fijo debido a que no pueden controlar los cambios requeridos por los usuarios finales.

- **Problemas de validación**. En un proceso basado en la especificación, la verificación y la validación están pensadas para demostrar que el sistema cumple su especificación. Un equipo independiente de V & V puede empezar a trabajar tan pronto como esté disponible la especificación y puede preparar pruebas en paralelo con la implementación del sistema. Los procesos de desarrollo iterativo intentan minimizar la documentación y entrelazan la especificación y el desarrollo. Por lo tanto, la validación independiente de los sistemas desarrollados incrementalmente es difícil.
- **Problemas de mantenimiento**. Los cambios continuos tienden a corromper la estructura del cualquier sistema software. Esto significa que cualquiera, aparte de los desarrolladores originales, puede tener dificultades para entender el software. Una forma de reducir este problema es utilizar refactorización, donde se mejoran continuamente las estructuras del software durante el proceso de desarrollo. Además, si se utiliza tecnología especializada, como los entornos RAD puede convertirse en obsoleta. Por lo tanto, puede ser difícil encontrar personas que tengan los conocimientos requeridos para dar mantenimiento al sistema [18].

2.3.4. Cronograma de actividades

A continuación se observa en la figura 2.2 el cronograma de actividades planificadas según la metodología escogida.

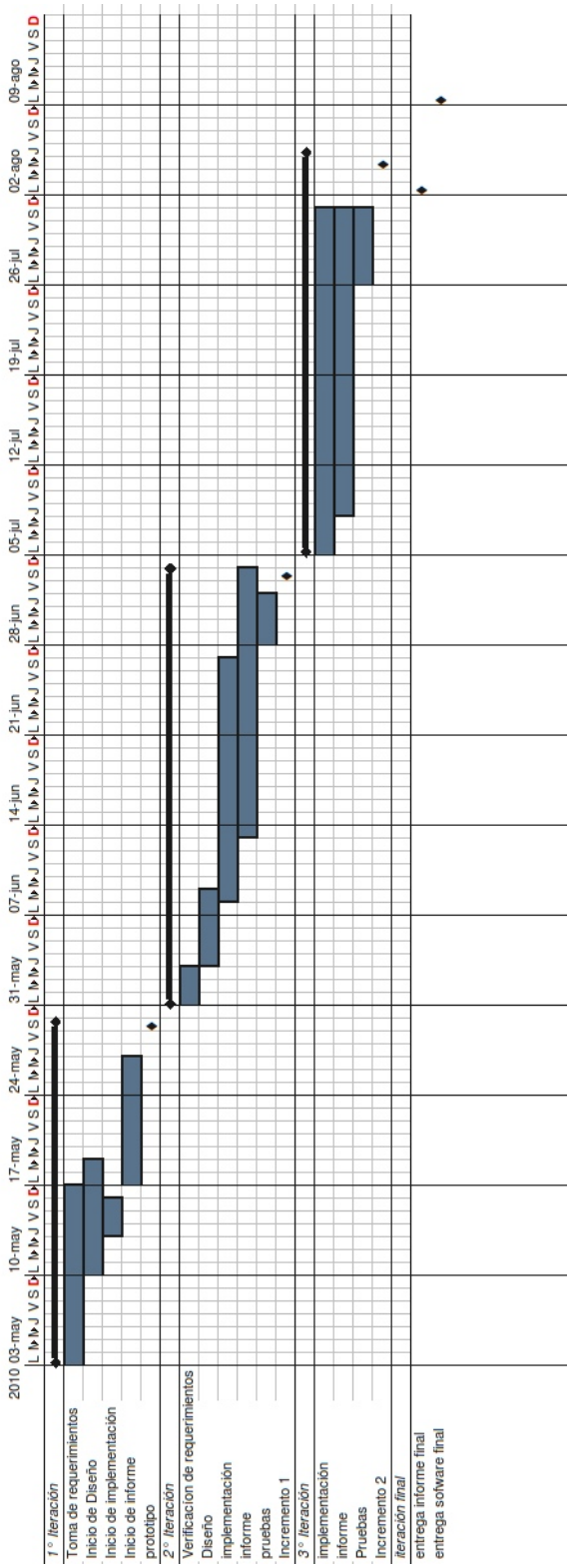

Figura 2.2: Cronograma de actividades

2.4. Sobre el sistema

Como nombre del sistema se escogió una palabra en mapudungun para estar afín al contexto de la municipalidad, debido a que como se describía anteriormente, la mayoría de los habitantes de Nueva Imperial son indígenas o descendientes de indígenas.

La palabra a escogida fue *Ramtun* que significa consultar o preguntar[5], ésta elección se llevó a cabo ya que su significado captura la esencia del sistema y además por que la acción "consultar" está involucrada, a grandes rasgos, en los objetivos del sistema.

2.4.1. Nombre del sistema

"Ramtun: Sistema de administración de ayudas de la I. Municipalidad de Nueva Imperial"

2.4.2. Objetivos generales

"Ofrecer una herramienta de consulta y de apoyo a la administración de ayudas, eficaz, de fácil aprendizaje y de nivel profesional"

2.4.3. Objetivos específicos

- Permitir un manejo de campañas políticas, al crear una forma de desplegar resúmenes de los sectores de la comuna que han solicitado más ayuda (o los sectores que efectivamente recibieron ésta ayuda).
- Tener un entorno amigable al usuario.
- Tener bajos tiempos de respuestas.
- Otorgar ayuda en la toma de decisiones de la municipalidad.
- Permitir escalabilidad futura.
- Permitir compartir información de manera expedita.

Capítulo 3

Requerimientos

3.1. Introducción

El presente capítulo se define como un documento de especificación de requerimientos del software (ERS), requerimientos que son acordados por el cliente (usuario), por lo que contiene en forma detallada todos los aspectos y funcionalidades esenciales para poder diseñar y desarrollar el sistema propuesto como solución.

Además se profundizara en los aspectos mismos de la relación que tendrán los usuarios con el software, y se detallaran los niveles de accesibilidad.

3.1.1. Propósito

El documento de especificación de requerimientos tiene por finalidad el describir de forma clara y lo más detalladamente posible las funciones que el cliente requiere para el sistema.

Está dirigido tanto a los desarrolladores, diseñadores del sistema como a los mismos clientes (usuarios del sistema) ya que éste es un documento dinámico sujeto a diversas actualizaciones y/o correcciones futuras las cuales serán llevadas a cabo por cada uno de los grupos involucrados anteriormente nombrados, hasta alcanzar una versión final.

Sin embargo, para aspectos de éste proyecto se definirá sólo un breve ciclo de toma y levantamiento de requerimientos, esto para estar dentro de los plazos ya estipulados.

3.1.2. Alcance

Los requerimientos de este documento tienen directa relación con el sistema a desarrollar, por ende, también se relacionan al proceso mismo de recepción de solicitudes de ayudas, es decir sólo contendrá descripciones a las funcionalidades que se encuentren en el sistema y según se requiera, también es posible que contenga descripciones del nuevo procedimiento de recepción de solicitudes de ayudas.

3.1.3. Definiciones, siglas y abreviaciones

Definiciones

Cliente	Es la persona o entidad a la cual se le desarrollará la solución, puede coincidir o no con los usuarios del sistema.
Usuario	Persona o grupo de personas que utilizarán el sistema, existen diversos, cada uno con distinto tipo de acceso.
Solicitante	Son los miembros de la comuna los cuales se dirigirán ante el municipio a solicitar la ayuda correspondiente.

siglas

IEEE	Institute of Electrical and Electronics Engineers.
ERS	Especificación de requerimientos de software.

3.1.4. Referencias

Estándar IEEE 830[6]

3.1.5. Apreciación Global

Hasta ahora se ha hablado básicamente del ámbito del documento incluyendo la terminología necesaria para poder comprenderlo a cabalidad, las secciones siguientes describirán los factores generales que influyen y conciernen al sistema junto los requerimientos específicos que explicaran en detalle las funcionalidades exigidas por el cliente.

3.2. Descripción global

3.2.1. Perspectiva del producto

El sistema Ramtun surge para ofrecer una forma más rápida y más confiable de obtener información, permitiendo además hacer un seguimiento sencillo de las solicitudes de ayudas de la comunidad de nueva imperial, siendo una alternativa moderna al sistema manual actualmente ocupado en la municipalidad.

El sistema está pensado para ser autónomo, sin dejar de lado por ello, la posibilidad de agregar módulos o también anexar futuros sistemas a su base de datos.

3.2.2. Funciones del producto

A grandes rasgos las funcionalidades requeridas para el sistema Ramtun son las siguientes:

- Administración de usuarios y miembros de la comunidad.
- Administración de sectores, departamentos y áreas de solicitud.
- Administración de citas y solicitudes.
- Generación de reportes y despliegue visual de mapas.

Administración de usuarios y comunidad

Para tener un control sobre que funcionarios llevan a buen puerto las solicitudes de ayuda, es necesario un manejo tanto de la comunidad de nueva imperial como de los funcionarios mismos, para llevar a cabo esta tarea, las funciones básicas en esta sección son: ingresar usuario, modificar usuario, desplegar usuario, dar de baja usuario. y por el lado de los solicitantes las funciones serán: ingresar solicitante, modificar solicitante, desplegar solicitante.

Administración de sectores, departamentos y áreas

Se debe administrar los sectores de la comuna de nueva imperial para ello las funciones a considerar son: ingresar sector, modificar sector, desplegar sector. también debe estar la opción de administrar los departamentos de la municipalidad, cuyas funciones son: ingresar departamento, modificar departamento, desplegar departamento, por último se debe tener poder administrar los tipos de áreas a las cuales pudiera pertenecer una solicitud, las funciones son: ingresar área, modificar área.

Administración de citas y solicitudes

Esta tarea junto con las siguientes, son la base y razón de ser del sistema y hacen referencia a la posibilidad de, primero, administrar citas de los usuarios y segundo, generar las solicitudes con sus respectivos seguimientos.

Para la primera tarea se consideran las siguientes funcionalidades: ingresar cita, modificar cita, desplegar cita, borrar cita. y para la segunda se consideran: ingresar solicitud, modificar solicitud, desplegar solicitud, desplegar seguimiento

Generación de reportes y despliegue visual

Esta tarea corresponde la elaboración de informes, como también al despliegue gráfico de ciertos valores relevantes para las altas autoridades de la municipalidad, como es el caso de despliegue de mapas de los sectores de la comuna con la cantidad de ayuda recibida respectivamente. la funcionalidad requerida aquí es: generar reporte, desplegar mapa.

Cada una de estas funcionalidades es detallada ampliamente la sección siguiente.

3.2.3. Características del usuario

Administrador

El administrador tendrá todos los privilegios, poseerá el acceso a todas las secciones del sistema, como característica única, podrá gestionar quien puede acceder al sistema (también el tipo de acceso).

Alcalde

El alcalde posee accesos similares a un funcionario pero como característica única, podrá generar reportes.

Encargado

Es el usuario más restringido, sólo podrá administrar sus citas y generar solicitudes y sus respectivos seguimientos, también podrá ir ingresando a los miembros de la comunidad de Nueva imperial, si es que aun no están en la base de datos.

3.2.4. Restricciones

Es necesario un manejo de sesiones por cada usuario lo que implica que el acceso a este sistema es restringido a previa "suscripción" o registro de dicho usuario en el sistema (por parte del administrador).

Además debido a que la arquitectura es cliente/servido la implementación se ve obligada realizarse mediante un servidor local (para tener un mayor control sobre el respaldo de las base de datos y el sistema mismo)

3.2.5. Atención y dependencias

Como la solución es un sistema basado en web, se debe tener (y se asume) una red local bien configurada y administrada, además queda en evidencia que para poder ocupar eficientemente el sistema, el usuario debe estar más o menos familiarizado con el manejo básico del PC.

3.3. Requerimientos específicos

En esta sección se realiza una descripción en detalle de 5 tipos de requisitos observados para el sistema Ramtun, los cuales son:

- Requerimientos funcionales.
- Requerimientos no funcionales.
- Requerimientos de información.
- Requerimientos de técnicos.
- Requerimientos de seguridad y auditoría.

3.3.1. Requerimientos funcionales

A continuación se detallan las funcionalidades requeridas, están divididas y ordenadas de la misma manera que en la sección anterior (sección 3.2).

Administración de usuarios y comunidad

R 1.- El sistema deberá gestionar usuarios.

- **R 1.1.-** Deberá poder manejar usuarios con distintos niveles de privilegios.
- **R 1.2.-** Deberá permitir que sólo el administrador pueda realizar cambios de privilegios de los usuarios.
- **R 1.3.-** Deberá almacenar un foto del usuario.
- **R 1.4.-** Deberá poder ingresar un usuario.
- **R 1.5.-** Deberá poder modificar un usuario.
- **R 1.6.-** Deberá poder desplegar listado de usuarios.
- **R 1.7.-** Deberá poder dar de baja a un usuario.

R 2.- El sistema deberá restringir el acceso.

- **R 2.1.-** Deberá poder validar si un usuario puede o no entrar al sistema.
- **R 2.2.-** Deberá poder desplegar la información restringida según corresponda (por cada nivel de acceso)
- **R 2.3.-** Deberá poder manejar limites en los tiempos de vida de las sesiones.

R 3.- El sistema deberá gestionar los solicitantes.

- **R 3.1.-** Deberá poder ingresar datos relevantes de los solicitantes.
- **R 3.2.-** Deberá poder modificar solicitantes.
- **R 3.3.-** Deberá poder desplegar solicitantes.
- **R 3.4.-** Deberá poder permitir búsqueda mediante filtros.

Administración de sectores, departamentos y áreas

R 4.- El sistema deberá gestionar sectores de la comuna.

- **R 4.1.-** Deberá almacenar las coordenadas geográficas para ubicar o geolocalizar el sector en google maps.
- **R 4.2.-** Deberá existir la opción de ingresar sectores "al vuelo".
- **R 4.3.-** Deberá ingresar sector.
- **R 4.4.-** Deberá desplegar sectores
- **R 4.5.-** Deberá modificar sector.

R 5.- El sistema deberá gestionar departamentos del municipio

- **R 5.1.-** Deberá poder ingresar un departamento.
- **R 5.2.-** Deberá poder modificar departamentos.
- **R 5.3.-** Deberá poder desplegar listado de departamentos.

R 6.- El sistema deberá gestionar áreas (o categorías) de solicitudes

- **R 6.1.-** Las áreas de solicitud son conocidas y pueden ser: alimentación, ganado, vivienda, social, por nombrar algunas.
- **R 6.2.-** Deberá poder ingresar el área de la solicitud.
- **R 6.3.-** Deberá poder modificar el área de la solicitud.
- **R 6.4.-** Deberá poder desplegar un listado de áreas.

Administración de citas y solicitudes

R 7.- El sistema deberá permitir al usuario la planificación de citas.

- **R 7.1.-** Deberá poder desplegar un sistema de planificación (tipo agenda).
- **R 7.2.-** En caso de corresponder cita, deberá poder dar aviso en la pantalla de bienvenida las citas correspondientes.
- **R 7.3.-** Deberá poder ingresar una cita.
- **R 7.4.-** Deberá poder modificar una cita.
- **R 7.5.-** Deberá poder desplegar las citas por día respectivo.
- **R 7.6.-** Deberá poder eliminar una cita.

R 8.- El sistema deberá administrar todas las solicitudes elevadas por los solicitantes de la comunidad.

- **R 8.1.-** Deberá poder manejar distintos estados de solicitud de ayuda (pendiente, entregado, cancelado por nombrar algunos).
- **R 8.2.-** Deberá poder ingresar una solicitud.
- **R 8.3.-** Deberá poder modificar una solicitud.

R 8.4.- Deberá poder buscar una solicitud asociada a un solicitante, esto a través de la búsqueda de solicitantes. (es decir al momento de buscar un solicitante, permitir la opción de "listar solicitudes de este solicitante")

R 9.- El sistema deberá gestionar historiales de cada solicitud

R 9.1.- El sistema deberá desplegar un listado correspondiente al historial de las solicitudes realizadas por un determinado solicitante.

R 9.2.- El sistema deberá realizar un seguimiento en el historial de cada una de las solicitudes, desplegando cuándo y quién fue el último usuario en atender la respectiva solicitud.

Generación de reportes y despliegue visual de mapas

R 10.- El sistema deberá poder generar informes estadísticos relevantes extraídos de la información de las solicitudes y su correspondientes historiales.

R 10.1.- Deberá poder desplegar cuántas han sido las ayudas entregadas satisfactoriamente por cada sector de la comunidad.

R 10.2.- Deberá poder desplegar que funcionarios del municipio han llevado a entregar más ayudas satisfactoriamente.

R 10.3.- Deberá poder listar en orden que sectores de la comunidad han recibido mayor ayuda.

R 10.4.- Deberá poder mostrar por un periodo de tiempo cual ha sido del porcentaje de ayudas satisfactorias, confrontando el "grado de efectividad de entregas de ayuda" con el "porcentaje de ayudas entregadas aceptadas como mínimo mensual", incorporando así indicadores de gestión.

R 10.5.- Se deja abierta la posibilidad de incorporar más informes a futuro.

R 11.- El sistema deberá poder geolocalizar los sectores

R 11.1.- Deberá identificar cuánta ayuda a recibido cada sector desplegando dicha información en un mapa, para una mejor comprensión del usuario.

3.3.2. Requerimientos no funcionales

1. El sistema deberá tener un logo identificativo de la municipalidad.
2. El sistema deberá ser amigable y en gran medida de uso intuitivo.
3. El sistema deberá tener tiempos de respuesta en promedio inferiores a 7 segundos.
4. El lenguaje a ocupar para el desarrollo del proyecto sera PHP, junto a html y javascript.
5. El enfoque a ocupar en el proyecto se guiara mediante el patrón de diseño modelo, vista, controlador.
6. Para abaratar costos se ocupará como administrador de base de datos MySql apache 2
7. Para agilizar el desarrollo se optó por la integración de PHP a symfony 1,4 un framework MVC
8. Como capa de abstracción de BD se ocupará Doctrine 1.2
9. Como framework de javascript enriquecido se ocupará mootools 1.2.

3.3.3. Requerimientos de información

A continuación se detallan los datos que el sistema necesita registrar y administrar.

Sector

- nombre de sector
- coordenadas

Área

- nombre del área de solicitud

Departamento

- nombre del departamento
- descripción

Usuario

- nombre de usuario
- contraseña de acceso
- tipo de usuario
- nombres
- apellidos
- foto de perfil
- correo electrónico
- departamento

Solicitante

- Rut
- nombres
- apellidos
- sexo
- fecha nacimiento
- teléfono contacto
- correo electrónico
- sector
- dirección

Solicitud

- fecha de solicitud
- fecha de termino
- solicitante que la pide
- usuario que atiende
- descripción
- área de solicitud
- estado

Historial

- fecha de historial
- usuario que la recepciona
- observaciones
- solicitud asociada

Cita

- título de cita
- descripción de cita
- fecha de cita
- solicitante citado
- usuario propietario de la cita
- hora inicio
- hora término

3.3.4. Requerimientos técnicos

En esta sección se describen los requerimientos tanto de la parte cliente sistema, como de la parte servidora, detallando los requisitos mínimos para poder obtener una buena experiencia de usabilidad por parte del usuario final.

Requerimientos de desarrollo

Hardware

Estos son los requerimientos de hardware mínimos para poder realizar el desarrollo.

- Procesador Intel o compatible de 1.5 GHz
- 512 MB de memoria RAM
- Disco duro 50 GB
- Periféricos (Teclado, mouse, monitor)

Software

- Sistema operativo GNU/Linux (Ubuntu 10.04)
- Herramientas CASE: umbrello, dia, MySQL Workbench, gedit
- Servidor XAMPP

Requerimientos de puesta en marcha (Cliente)

Estos son los requerimientos de hardware mínimos para poder usar la parte cliente del sistema Ramtun con un mínimo usabilidad.

Hardware

- Procesador Intel o compatible de 1 GHz
- 256 MB de memoria RAM
- Disco duro 50 GB
- Periféricos (Teclado, mouse, monitor)

Software

- Windows o Sistema operativo GNU/Linux
- Ultima versión de navegador firefox (recomendado), chrome, Internet Explorer

Requerimientos de puesta en marcha (Servidor)

Hardware

- Procesador Intel o compatible de 1.8 GHz
- 512 MB de memoria RAM
- Disco duro 100 GB

Software

- Windows o Sistema operativo GNU/Linux
- Servidor WAMP o XAMPP (Según servidor elegido)

3.3.5. Requerimientos de seguridad y auditoría

Integridad de la base de datos

Como toda aplicación que maneja grandes cantidades de datos, es completamente necesario el constante respaldo de datos, se recomiendan realizar esta acción cada 15 días, pero esto queda a completa responsabilidad por parte de él o los encargados del sistema en el municipio (departamento de informática).

Control de acceso al sistema Ramtun y base de datos

El acceso al sistema estará restringido a sólo algunos usuarios, como se describía anteriormente sin embargo, existen 2 tipos de acceso al mismo.

Acceso de usuarios al sistema

Este tipo de acceso es el descrito en secciones anteriores, y hace referencia a los usuarios "físicos" del mismo, es decir, a las personas que usarán la aplicación su acceso esta permitido mediante el login del mismo, y las validaciones son por lenguaje de programación.

Acceso del sistema a la base de datos

Este tipo de acceso hace referencia al usuario de la base de datos; y es el usuario con el que el sistema se conecta al administrador de BD y realiza los cambios necesarios en ella, por seguridad sólo debe tener permisos para ejecutar acciones básicas sobre el conjunto de datos, como por ejemplo. select, delete, insert, update entre otras, y no permitir acciones como borrado del esquema (base de datos), agregar permisos o eliminar usuarios, esto se logra en conjunto con el administrador de base de datos y es transparente para el usuario final.

Control de acceso al servidor

El control de acceso a los sistemas es la primera forma de evitar la intrusión de individuos ajenos a la institución, pero esto debe ser llevado en conjunto con la restricción al acceso del servidor mismo, pues si un individuo "mal intencionado" accede físicamente al servidor, todos los esfuerzos de restricción por tipos de usuarios serian en vano, por ello se recomienda una análisis y posible reestructuración de las políticas de seguridad actuales.

Capítulo 4

Herramientas utilizadas

4.1. Introducción

El presente capítulo expondrá las herramientas, sistemas y utilidades más importantes utilizadas en el desarrollo e implementación del sistema. muchas de ellas se consideraron a partir de los requerimientos expuestos anteriormente.

Para cada una de ellas se detallará según convenga un poco de historia y una breve explicación de que es y para que sirve, si bien la lista podría parecer un poco extensa, el lector debe saber que no se quiso profundizar en todas las herramientas ocupadas, sino que solamente en las más relevantes involucradas con el diseño e implementación, dejando fuera algunas como las herramientas CASE ocupadas principalmente para el diagramado y creación de documentación anexa.

Las herramientas omitidas fueron: Gedit, Dia, Umbrello, entre otras.

4.2. GNU/Linux (Ubuntu 10.04)

Antes de hablar sobre el sistema operativo GNU/Linux es necesario introducir al lector al sistema Unix, ya que Linux es un Clon de este último [10]

4.2.1. Historia de Unix

En 1969, Ken Thompson, desarrollador de los laboratorios Bell (que forman parte de AT&T), inventó el sistema Unix.

K. Thompson había trabajado previamente en el sistema Multics que permitía, por ejemplo, ocultar el sistema al usuario e incluso al programador; el objetivo era poder desarrollar nuevas aplicaciones haciendo abstracción del sistema. Cuando Bells Labs se retiró del proyecto Multics, empezó a desarrollar su propio sistema en un equipo DEC PDP-7 salvado de la quema. Conservó ciertas ideas desarrolladas por Multics y añadió sus propias innovaciones: acababa de nacer Unix.

El mismo tiempo, Dennis Ritchie inventa un nuevo lenguaje de programación: el lenguaje C. En la línea de Unix, este lenguaje pretendía ser amigable, flexible y sin restricciones. Los laboratorios Bell, que conocían las herramientas desarrolladas, dieron apoyo a este dúo con una prima en 1971 para la elaboración de un sistema de automatización de escritorio de uso interno.

En realidad, los sistemas desarrollados hasta entonces estaban codificados en lenguaje ensamblador, próximo a la arquitectura de hardware, lo que les confería un alto rendimiento, pero obligaba también a reescribir el programa cuando el hardware obsoleto era reemplazado por equipos más recientes. Comprendiendo que los avances tecnológicos podían paliar una pérdida de velocidad y que las técnicas de compilación habían experimentado grandes progresos, D. Ritchie y K. Thompson dieron prioridad a la portabilidad del sistema. Al escribir la totalidad del sistema en Unix en lenguaje C, consiguieron hacer funcionar el sistema en muchos equipos de tipos diferentes desde 1974.

Así, un compilador de C desarrollado para cada equipo bastaba para hacer posible el uso del mismo entorno en diferentes equipos; esto no se había hecho nunca abría grandes posibilidades. El desarrollo de aplicaciones liberadas de la restricción del hardware disminuía considerablemente los costes de diseño para el usuario cuando era necesario renovar un equipo que se había quedado anticuado, Además, los programadores podían fabricarse cajas de herramientas de software y transferirlas de un equipo a otro sin tener que reinventarlo todo.

Bajo el impulso de AT&T Unix se desarrolló rápidamente. En 1980, este sistema estaba ya muy extendido en las universidades y en los centros de investigación.

La universidad de California en Berkeley (que utilizaba Unix desde hacía varios años), aportó diferentes mejoras a este sistema operativo, actualmente más conocido con el acrónimo de BSD (Berkeley Software Distribution). Los desarrollos "Berkeley"se publicaron paralelamente a los realizados por el grupo de trabajo de Unix de AT&T. Actualmente, la mayor parte de sistemas operativos de tipo Unix en el mercado consisten en una mezcla del código original de AT&T y de las "mejoras de Berkeley".

4.2.2. GNU

Debido a la imposibilidad de obtener y modificar el código fuente del controlador de la nueva impresora que llegó a su departamento, Richard Matthew Stallman, entonces joven investigador en el laboratorio de Inteligencia Artificial del MIT (Massachussets Institute of Technology), se concienció de los riesgos vinculados con software propietario; para oponerse a la comercialización del software y más particularmente a la falta de disponibilidad del código fuente, inició entonces un movimiento para el desarrollo de software libre de derechos 1984.

Este movimiento se tradujo en la creación de un proyecto que consistía en reescribir completamente un sistema operativo libre. el modelo seguido era Unix, y Richard Stallman llamó a su proyecto GNU, que significa GNU's Not Unix.

El proyecto GNU conoció rápidamente un gran éxito y muchas herramientas y aplicaciones de Unix se desarrollaron a partir de cero ("from scratch").

Sin embargo, el núcleo libre de este sistema, llamado "Hurd", no se desarrolló tan rápidamente. El proyecto GNU se limitó durante cierto tiempo a ser una gama de herramientas completa de Unix sin núcleo.

4.2.3. Linux

Utilizando el sistema operativo Minix, desarrollado por Andrew S. Tanenbaum con el objetivo de mostrar a sus estudiantes el funcionamiento de un sistema de tipo Unix, en un proyecto escolar sobre el modo protegido de los procesadores Intel 386, Linus Torvalds empezó a desarrollar su propio núcleo Unix para añadirle nuevas funcionalidades.

Linux (Linus's Unix) nació, pues, en 1991, gracias a un estudiante de la universidad de Helsinki. El éxito de Linux se basa en una idea ingeniosa de su creador, L. Torvalds: inscribir su proyecto bajo los términos de la licencia GPL y proponer a todos los programadores y otros hackers de Internet que le ayudaran.

El impulso de Linux proviene en gran parte del hueco que llenó en términos de núcleo en el proyecto GNU.

Características

Multitarea Está diseñado para ejecutar varios programas al mismo tiempo (PID - process ID, codificados en 32 bits). Utiliza un asignador para ejecutar varias acciones con un mismo procesador y también puede sacar partido de arquitecturas multiprocesador (hyperthreading, SMP - Symmetric Multi-Processors, NUMA - Non-Uniform Memory Access).

Multiusuario Este sistema permite el uso por parte de varia personas de los recursos que administra, Estas personas se distribuyen en grupos de usuario y es necesaria una autenticación para asegurar los derechos individuales. Los UID (User's ID) y GID (Group's ID) están codificados en 32 bits.

Multiplataforma Linux ha sido llevado a un gran número de arquitecturas de hardware, como Intel y sus clones (AMD, Cyrix), SPARC, MIPS, Dec Alpha, PowerPC, PalmPilot...

Encontramos también Linux en sistemas embebidos, como autómatas industriales o autorradios MP para el particular.

Sistema de archivos Linux soporta un gran número de sistemas de archivo, además de los de tipo Unix, incluyendo ISO9660 (CD-Rom), Windows 9x, NTFS y Macintosh. Soporta también los ACL (Access List) y las cuotas.

Administración de memoria En vez de efectuar un "swap"de procesos completos, el núcleo página la memoria virtual, lo que limita los accesos al disco cuando falta memoria física. Además, la memoria compartida con copia en escritura permite que varios procesos utilicen el mismo espacio de memoria; cuando uno de ellos intenta escribir en la memoria, se realiza una copia de página de memoria en otra parte. Este mecanismos optimiza el uso y la cantidad de memoria necesaria en el sistema. Además, Linux soporta hasta 64 GB de RAM con las funciones de hardware PAE (Physical Address Extension) de Intel.

Redes Como cualquier otro Unix, Linux posee una capa de red muy fiable y rápida. Soporta un amplio abanico de protocolos (TCP/IP) versiones 4 y 6, Netware, Appletalkm Lan Manager, IPX, IPsec ...) y dispositivos de red (Ethernet, TokenRing, FDDI, Wi-Fi ...)

Compatibilidad Linux es compatible con Unix System V (AT&T) y BSD a nivel de código fuente, soporta las bibliotecas Unix en formato COFF y ELF y los binarios SCO SVR3 y SVR4 con los módulos de emulación IBCS2.

4.2.4. Ubuntu

Ubuntu es una distribución de GNU/Linux patrocinada por la empresa Canonical Ltd. [11]

Las versiones de Ubuntu se suceden cada seis meses, el número de versión está formado por el año en que se comenzó a distribuir, un punto y un número de dos cifras correspondiente al mes de distribución. Cada versión del sistema operativo se actualiza hasta 18 meses después de su lanzamiento, excepto las versiones "Long Term Support" o LTS, que tiene soporte de tres años para la versión de escritorio y cinco años para la versión de servidor.

4.3. Apache

(Acrónimo de "a patchy server"). Servidor web de distribución libre y de código abierto, siendo el más popular del mundo desde abril de 1996, con una penetración actual del 50 % del total de servidores web del mundo (agosto de 2007) [9].

La principal competencia de Apache es el IIS (Microsoft Internet Information Services) de Microsoft.

Apache fue la primera alternativa viable para el servidor web de Netscape Communications, actualmente conocido como Sun Java System Web Server.

Apache es desarrollado y mantenido por una comunidad abierta de desarrolladores bajo el auspicio de la Apache Software Foundation.

La aplicación permite ejecutarse en múltiples sistemas operativos como Windows, Novell NetWare, Mac OS X y los sistemas basados en Unix.

Historia de Apache

La primera versión del servidor web Apache fue desarrollada por Robert McCool, quien desarrollaba el servidor web NCSA HTTPd (National Center for Supercomputing Applications). Cuando Robert dejó el NCSA a mediados de 1994, el desarrollo de httpd se detuvo.

Robert McCool buscó otros desarrolladores para que lo ayudaran, formando el Apache Group. Algunos miembros del grupo original fueron Brian Behlendorf, Roy T. Fielding, Rob Hartill, David Robinson, Cliff Skolnick, Randy Terbush, Robert S. Thau, Andrew Wilson, Eric Hagberg, Frank Peters y Nicolas Pioch.

La versión 2 del servidor Apache fue una reescritura sustancial de la mayor parte del código de Apache 1.x, enfocándose en una mayor modularización y el desarrollo de una capa de portabilidad, el Apache Portable Runtime.

Apache 2.x incluyó multitarea en UNIX, mejor soporte para plataformas no Unix (como Windows), una nueva API Apache y soporte para IPv6.

La versión 2 estable de Apache, fue lanzada el 6 de abril de 2002.

Características de Apache

- Soporte para los lenguajes perl, python, tcl y PHP.
- Módulos de autenticación: mod_access, mod_auth y mod_digest.
- Soporte para SSL y TLS.
- Permite la configuración de mensajes de errores personalizados y negociación de contenido.
- Permite autenticación de base de datos basada en SGBD.

Uso de Apache

Apache es principalmente usado para servir páginas web estáticas y dinámicas en la WEB. Apache es el servidor web del popular sistema XAMP, junto con MySQL y los lenguajes de programación PHP/Perl/Python. La "X"puede ser la inicial de cualquier sistema operativo, si es Windows: WAMP, si es el Linux: LAMP, etc.

4.4. MySQL

MySQl es el sistema de administración de bases de datos relacionales (RDBMS) de código abierto más extendida del mundo. Era desarrollada por MySQL AB, una empresa sueca, actualmente fue comprado por Oracle.

La primera versión de MySQL apareció en 1995. Esta primera versión fue creada para un uso personal a partir de MySQL.

En 2000, la versión 3.23 pasó a tener licencia GPL (General Public License).

En 2003, la versión 4, aparecida en 2001, se declaró estable. Esta versión aportó numerosas funcionalidades y mejoras: operador UNION, DELETE para varias tablas, nuevas opciones para la gestión de los privilegios, mejora del rendimiento, subconsultas, etc.

En 2005, la versión 5, aparecida en 2003, se declaró estable. Esta destacada versión introdujo numerosas características que faltaban en MySQL: rutinas, triggers, vistas.

A finales de 2007, se distribuyó la versión 5.1 como Release Candidate.
MySQL está disponible con dos licencias diferentes:

- Licencia GPL
- Licencia comercial.

Si se utiliza MySQL en un producto no comercial, se pude utilizar MySQL con su licencia GPL (versión MySQL Community Server). Si se utiliza MySQL en un producto comercial, (y para tener asistencia técnica para el programa) se debe adquirir una licencia comercial (versión MySQL Enterprise) [12].

4.5. PHP

PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje .ºpen Sourceïnterpretado de alto nivel, especialmente pensado para desarrollos web y el cual puede ser incrustado en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender. La meta de este lenguaje es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil, aunque se pueda hacer mucho más con PHP [7].

Historia de PHP

Rasmus Lerdorf, miembro del equipo de desarrollo de Apache, creó PHP (Personal Home Page) en 1994. Su única intención fue la de crear un pequeño sistema de control para verificar el número de personas que leían su currículum vitae en la Web [13].

En los meses siguientes a su creación, PHP se desarrolló en torno a un grupo de programadores que comprobaban el código y sus revisiones. Para dar más potencia al sistema, Rasmus creó funciones en lenguaje C para permitir conexión a bases de datos. Este fue el comienzo de la potencia real del lenguaje.

En 1995, apareció un conjunto de herramientas sobre PHP. Esta biblioteca se llamó "Herramientas para páginas personales" y contenían un analizador de código muy sencillo, un libro de visitas, un contador y algunas macros que facilitaban el trabajo de los diseñadores.

A mediados de 1995, apareció una revisión pública llamada PHP/FI 2.0. Esta nueva versión contaba con un analizador sintáctico reescrito desde 0, además de unas herramientas escritas para el tratamiento de datos desde un formulario (de ahí el nombre Fi, Form Interpreter) y conectividad con MySQL (Gestor de bases de datos).

Hacia 1997, PHP/FI se estaba usando en más de 50.000 páginas en todo el mundo. En este período de tiempo, Zeev Suraski y Andi Gutmans decidieron crear una nueva versión de PHP/FI para solventar unos problemas con una aplicación de comercio electrónico que estaban desarrollando.

PHP 3.0 nació con suculentas innovaciones como la conectividad con varios gestores de bases de datos, protocolos y una API ampliada. La versión oficial de PHP 3.0 vio la luz en junio de 1998, donde se contemplaba ya la programación orientada a objetos.

En 1999 se realizó la primera revisión del motor Zend (Zend Engine), que aportaba modularidad, claridad y herramientas de optimización para páginas de gran escala. Zend viene de la unión de Zeev y Andi.

PHP 4.0 vio la luz en mayo de 2000, dividida en 3 partes: El motor Zend, la API de servidor y los módulos de funcionas. El motor Zend es el responsable de analizar el código PHP, definir la sintaxis y del lenguaje de programación. La API permite la comunicación con el servidor. Con esta API es posible utilizar PHP desde varios servidores. Los módulos contienen funciones para el manejo de cadenas, archivos XML o tratamiento de imágenes.

La orientación a objetos no está muy lograda en PHP 4.0. Los objetos tienen un tratamiento muy pobre e ilógico. La definición de las variables miembro (propiedades) y los métodos son siempre públicos, por lo que la encapsulación es nula. Todos los objetos se pasan por valor por defecto cuando deberían

pasarse por referencia. Todas estas propuestas realizadas por el equipo de desarrollo de PHP han desembocado en la creación del motor Zend 2.0. y su consecuencia PHP 5.

PHP 5 incorpora una verdadera orientación a objetos. Añadiendo las palabras reservadas public, protected y private a la definición de las propiedades y métodos de los objetos, se permite una verdadera encapsulación. Además del considerable avance con respecto a los objetos, PHP 5 incorpora un control de errores muy mejorado, al estilo de los lenguajes de programación más avanzados.

4.6. Doctrine

Doctrine es un mapeador de objeto relacional (ORM - Object Relational Mapper) para PHP 5.2.3+ Un ORM es una técnica usada en lenguajes de programación como capa de abstracción que permite negociar con una base de datos relacional cuando ocupamos tipos de datos incompatibles con ella (como son los objetos) lo que permite tener una "base de datos orientada a objetos virtual".

Una de las características claves que posee Doctrine es la opción de escribir consultas de base de datos en un dialecto SQL propietario orientado a objetos llamado DQL - Doctrine Query Language, inspirado en Hibernate HQL, este proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene flexibilidad sin necesidad de la duplicación innecesaria de código.

Se sitúa sobre PDO y se divide en dos capas principales, la capa de abstracción a la base de datos (DBAL - Database Abstraction Layer) y el ORM. como lo ilustra la figura 4.1.

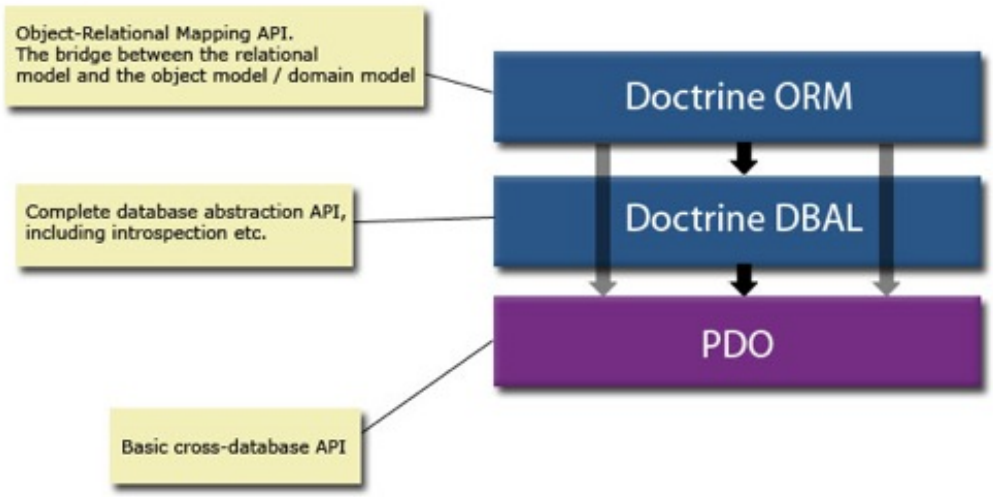

Figura 4.1: Interacción de las capas de Doctrine

4.7. Symfony

Antes de explicar sobre el framework symfony primero se extenderá sobre que es un framework. Un framework es una herramienta que simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un framework proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows. A continuación se muestran algunas de sus características [14].

Características de Symfony

Symfony se diseñó para que se ajustara a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares)
- Independiente del sistema gestor de bases de datos
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos
- Basado en la premisa de çonvenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web
- Preparado para aplicaciones empresariales y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros

4.8. Mootools

MooTools comenzó cuando Valerio Proietti lanzó un complemento para Prototype.js llamado moo.fx (que todavía está disponible en `http://moofx.mad4milk.net`) en octubre de 2005. Era una biblioteca de efectos ligera (3 KB!) que fue muy popular tanto por su facilidad de uso como por su pequeño tamaño.

No contento con la liberación de sólo un add-on para Prototype, Valerio comenzó a trabajar en su propio framework, MooTools, que significa "Mis herramientas orientadas a objeto" (My Object-Oriented Tools), y lo publicó en septiembre del 2006. La razón por la que comenzó esta tarea fue porque Prototype.js, lo que sumado numerosos accesos directos a la prototipos de Array, String, función, y así sucesivamente, de no ampliar las prototipo de elementos, y estaba cansado de escribir varias veces el prototipo de Elemento genérico.

En su núcleo, MooTools es un framework JavaScript que proporciona a los tres cosas esenciales que he mencionado antes sobre Prototype.js: proporciona accesos directos y clases base que hacen que hacer las cosas comunes fácil, se extiende objetos nativos para agregar funcionalidad a ellos, y, quizás lo más importante, la propia biblioteca se configura como una ilustración de cómo escribir Javascript bien, y, más concretamente, la forma de escribir JavaScript utilizando MooTools.

Estos conceptos no son necesariamente exclusivas de MooTools, de hecho, casi todos Javascript marcos (y hay un montón de ellos) a hacer estas cosas diversos grados de éxito. Lo que hace única es su código de MooTools estilo, su oferta bien redondeada, y su enfoque de base filosófica a su el desarrollo continuo. Las características principales, en las cuales se basa MooTools son:

- No duplicar código.
- Añadir funcionalidad que se adapte, en principio, con la propia filosofía de diseño de JavaScript.
- Si existe un buen estándar que funciona bien, pero aún no ha sido implementado, entonces implementar ese estándar.
- Extender objetos nativos (String, Function, Array, Element, Events y Number) como JavaScript fue diseñado para hacerlo.
- Escribir código de forma limpia y clara, para que sea comprensible por cualquier persona con los conocimientos necesarios para entenderlo.
- Cuidado de no exigir demasiado el navegador (memoria, ciclos de CPU, etc.).
- Abstraer tanto de distancia desde el navegador como sea posible.
- Siempre que sea posible, desde ya mismo sentir como si estuvieras escribiendo JavaScript.
- Hacerlo fácil, divertido, inspirador.
- Hacerlo modular.

4.9. GIT

Git es un sistema de control de versiones distribuido, libre y de código abierto, diseñado para manejar todo, desde pequeños a muy grandes proyectos con velocidad y eficiencia.

Cada clon Git es un repositorio con la historia completa del proyecto y total capacidad de seguimiento del mismo, sin depender del acceso de red o un servidor central, además la derivación en ramas y la mezcla de las mismas es fácil y rápida de hacer.

Git se utiliza para el control de versiones de archivos, al igual que herramientas tales como Mercurial, Bazaar, Subversion, CVS, Perforce, y Visual SourceSafe. [8]

Breve historia de GIT

Algunos de los objetivos del nuevo sistema fueron los siguientes [17]:

- Velocidad
- Diseño simple
- Fuerte soporte para desarrollo no lineal (miles de ramas en paralelo)
- Completamente distribuido
- Capacidad de manipular grandes proyectos eficientemente (como el Kernel de Linux)

Desde su nacimiento en el 2005, GIT ha evolucionado y madurado para ser fácil de usar y conservar estos cualidades iniciales. Es increíblemente rápido, es muy eficiente con grandes proyectos, y tiene un increíble sistema de ramificación para el desarrollo no lineal.

Capítulo 5

Estudio de factibilidad

5.1. Introducción

En los capítulos anteriores se han definido el problema a solucionar (junto con los motivos correspondientes), la descripción y definición de requerimientos del sistema. con esto ya se puede estimar la factibilidad del proyecto. Por lo que en éste capítulo se profundiza en los aspectos preliminares a la implementación, justificando y determinando a través de tres análisis si el proyecto es factible o no de llevar a cabo, esto mediante la medición de uso de recursos humanos, económicos y de tiempo.

los análisis expuestos son:

- Factibilidad técnica
- Factibilidad operativa
- Factibilidad económica

y se detallan a continuación:

5.2. Factibilidad técnica

La I. Municipalidad de Nueva Imperial ya cuenta con los equipo de los clientes (usuarios) que interactuarán con el sistema, esto implica tanto los componentes hardware como el software necesario (y sus respectivas licencias) para poder operarlos. Además posee una red integrada y una unidad informática propia por lo que la integración del sistema Ramtun es técnicamente posible.

5.3. Factibilidad operativa

Como el proyecto nace de la necesidad de la I. Municipalidad en mejorar su proceso actual de canalización de ayudas, y según los requerimientos tomados se puntualizan los siguientes términos del sistema relacionados con el nivel operativo del mismo:

- El sistema está pensado para ser intuitivo y de fácil manejo al usuario.
- El sistema reducirá los tiempos en los procesos relacionados a la canalización de ayudas rurales.
- El sistema permite recopilar la información relevante según los casos.
- El sistema permite acceso desde diferentes plataformas y equipos.

En definitiva las mejoras que otorga el sistema Ramtun, sumado a la facilidad de uso (por ende rápida curva de aprendizaje) y los planes de capacitación y puesta en marcha del sistema Ramtun (que se expondrán en el capítulo 7) prevén que el sistema es factible en términos operacionales.

5.4. Factibilidad económica

5.4.1. Estimación Personal

Análisis y diseño

En lo referente a las labores de analista se estima que las horas necesarias son alrededor de 65 horas totales que se serán distribuidas a lo largo del proyecto pues al estar usando el modelo iterativo incremental las tareas de análisis y toma de requerimientos son fuertes al principio pero van disminuyendo paulatinamente a medida que se avanza en el proyecto.

En los aspectos de diseño se estiman como total unas 25 horas a lo sumo, que al igual que en la fase de análisis son ampliamente ocupadas al inicio.

Implementación

En la que es codificación se estiman aproximadamente entre unas 13 a 14 semanas, es decir, poco más de 3 meses y medio, con un total de 190 horas de trabajo, reservando unas 13 horas de desarrollo por semana aproximadamente.

Costo

Sumando el análisis, diseño y desarrollo del sistema el total da 280 Horas de trabajo, luego, calculando la hora hombre a 0.5 UF y considerando la UF $21.143 se tiene:

$$costoH/H = 21143 * 0.4 = 8.457,2$$

lo que da como costo total aproximado de:

$$total = 8.457,2 * 280 = 2.368.016$$

5.4.2. Estimación según modelo cocomo

El modelo de estimación de costes de desarrollo de software ocupado para este análisis es el modelo COCOMO (COnstructive COst MOdel) desarrollado por Barry M. Boehm, este pertenece al grupo de los modelos algorítmicos que permiten establecer una relación matemática para estimar el esfuerzo y tiempo requerido para desarrollar un software.

COCOMO define tres modos de desarrollo o tipos de proyectos:

- **Orgánico**: proyectos relativamente sencillos, menores de 50 KDLC líneas de código, en los cuales se tiene experiencia de proyectos similares y se encuentran en entornos
- **Semi-libre**: proyectos intermedios en complejidad y tamaño (menores de 300 KDLC), donde la experiencia en este tipo de proyectos es variable, y las restricciones intermedias.
- **Rígido**: proyectos bastante complejos, en los que apenas se tiene experiencia y se engloban en un entorno de gran innovación técnica. Además se trabaja con unos requisitos muy restrictivos y de gran volatilidad. estables.

también existen diferentes submodelos definidos en COCOMO:

- **Modelo básico**: Se basa exclusivamente en el tamaño expresado en LDC.

- **Modelo intermedio**: Además del tamaño del programa incluye un conjunto de medidas subjetivas llamadas conductores de costes.
- **Modelo detallado**: Incluye todo lo del modelo intermedio además del impacto de cada conductor de coste en las distintas fases de desarrollo.

para cada uno de los modelos la función básica es la siguiente[18]:

$$E = a(Kldc)^b * m(X)$$

donde:

E es el *esfuerzo* medido en hombres/mes.

a y *b* son constantes que se definen en cada submodelo.

Kldc es la cantidad de lineas de código en miles.

m(X) es un factor que se calcula con 15 conductores de coste.

Para el caso del sistema Ramtun el modelo usado es el intermedio, debido a que es más preciso que el modelo básico, y lo suficiente para realizar una estimación adecuada para este proyecto, la tabla con las constantes según cada modo para el submodelo intermedio es la siguiente:

Modo/Constante	a	b	c	d
Orgánico	3.20	1.05	2.50	0.38
Semi-libre	3.00	1.12	2.50	0.35
Rígido	2.80	1.20	2.50	0.32

Para calcular el factor *kldc* es posible recurrir a una estimación de lineas de código o a través de puntos de función, para este análisis se optó por la primera opción.

Para el sistema Ramtun se estiman una 10.000 lineas de código aproximadamente, pero con la ayuda del framework (herramienta que automatiza varias tareas) se determinan que la cantidad real a codificar es menos de la mitad, es decir unas 4.200 lineas aproximadamente.

entonces se tiene:

$$kldc = 4.2$$

al ser inferior a las 50 *Kldc* se encasilla en el modo orgánico por lo cual las constantes a considerar en las formulas son las siguientes:

Constante	a	b	c	d
Orgánico	3.20	1.05	2.50	0.38

ahora corresponde encontrar el factor *m(X)* el cual se obtiene de la evaluación de 15 conductores de coste, lo que se explica en la siguiente tabla:

Valoración						
Conductores	**Muy Bajo**	**Bajo**	**Nom.**	**Alto**	**Muy Alto**	**Extr. Alto**
Fiabilidad requerida del SW	0.75	0.88	**1.00**	1.15	1.40	-
Tamaño base de datos	-	**0.94**	1.00	1.08	1.16	-
Complejidad del producto	0.70	0.85	**1.00**	1.15	1.30	1.65
Restric. tiempos de ejecución	-	-	1.00	**1.11**	1.30	1.66
Restric. almacenamiento principal	-	-	**1.00**	1.06	1.21	1.56
Volatilidad maq. virtual	-	**0.87**	1.00	1.15	1.30	-
Tiempos de respuesta del computador	-	0.87	1.00	**1.07**	1.15	-
Capacidad del analista	1.46	1.19	1.00	**0.86**	0.71	-
Experiencia en la aplicación	1.29	1.13	1.00	**0.91**	0.82	-
Capacidad de los desarrolladores	1.42	1.17	1.00	0.86	**0.70**	-
Experiencia en el S.O	1.21	1.10	1.00	**0.90**	-	-
Experiencia en el lenguaje de programación	1.14	1.07	**1.00**	0.95	-	-
Prácticas de programación modernas	1.24	1.10	1.00	**0.91**	0.82	-
Utilización de herramientas SW	1.24	1.10	**1.00**	0.91	0.83	-
Limitantes de planificación del proy.	1.23	**1.08**	1.00	1.04	1.10	-

por lo tanto:

$$m(X) = 1.00 * 0.94 * 1.00 * 1.11 * 1.00 * 0.87 * 1.07 * 0.86 * 0.91 * 0.70 * 0.90 * 1.00 * 0.91 * 1.00 * 1.08$$

$$m(X) = 0.470651453$$

luego reemplazando en la formula básica:

$$E = 3.2 * (4.2)^{1.05} * 0.470651453 = 6.8 hombres/mes$$

y en las formulas del modelo intermedio:

$$Tiempo = T = c * Ed$$

$$T = 2.5 * (6.8)^{0.38} = 5.18 meses$$

$$Personas = P = E/T$$

$$P = 6.8/5.18 = 1.31 personas$$

$$ldc/E = 4200/6.8 = 618 lineashombre/mes$$

5.4.3. Productividad

En la práctica los 1.31 programadores se aproximan a 1 lo que implica que para poder cumplir las expectativas el desarrollador deberá subir su carga a 808 lineas de código. Entonces la productividad queda como sigue: 1 programador desarrollando por poco mas de 5 meses, produciendo 808 lineas de código al mes, valores que se aproximan al cronograma estimado en capítulos anteriores. Además en el proyecto se estiman unas 300 H/H distribuidas en los 5,2 meses , dejando a cada mes con 58 horas para desarrollar las 808 lineas estimadas, lo que deja un margen de desarrollo de poco menos de 14 lineas de código cada 1 hora aproximadamente.

5.4.4. costo mano de obra

De lo anterior de calcula la hora/hombre como 0.4 UF considerando la UF $21.143.

$$costoH/H = 21143 * 0.4 = 8457,2$$

lo que multiplicado por las 300 horas da como resultado:

$$8457.2 * 300 = 2.537.160$$

5.4.5. Equipos y Software

Además en el desarrollo del sistema, se estima que se necesitaran de las siguientes herramientas software y componentes Hardware, cuyo costo de inversión se detallan a continuación:

Software	**Valor**
S.O. Linux	$0
PHP	$0
Symfony	$0
Apache	$0
MySQL	$0
Firefox	$0
Total:	**$0**

En el caso de los equipos se establecen 2 tipos el equipo cliente y el equipo para desarrollo, los que se pasan a detallar de forma aproximada.

Equipo cliente	**Valor**
Dual Core E5400 2.7	$48.400
Disco Duro Sata2 160Gb	$27.900
1GB RAM	$17.000
periféricos y otros	$40.000
Total:	**$133.300**

Equipo desarrollo	Valor
Core 2 Duo E7500 2.93	$90.000
Disco Duro Sata2 160Gb	$27.900
2GB RAM	$34.000
periféricos y otros	$40.000
Total:	**$191.900**

Cabe destacar que la municipalidad ya tiene los equipos necesarios, por lo que no se incurrirá en este gasto. Por lo tanto la única cifra real a considerar en el total, es la del desarrollo estimado con cocomo pues arrojo como resultado un valor levemente superior a la cifra dada por la estimación personal, esta diferencia se tomará como margen superior para evitar posible falta de recursos. La tabla siguiente gráfica el costo total:

Ítem	Valor
Costo desarrollo	$2.537.160
Costo equipos y software	$0
Costo Total del proyecto:	**$2.537.160**

Valores que están dentro de los margenes esperados, por lo tanto la realización de este proyecto en términos económicos es posible.

5.5. Conclusión del estudio de factibilidad

Se puede observar una diferencia en la estimación económica personal y la calculada con cocomo, que para este proyecto no fue muy significativa, se optó por el valor arrojado por cocomo para tener presente un margen mayor de recursos. Como apreciación final y debido a que todos los estudios elaborados (técnico, operativo, económico) dan al proyecto como factible de realizar, se puede concluir que la implementación del sistema Ramtun es completamente viable.

Capítulo 6

Diseño del sistema

6.1. Introducción

En este capítulo se procede a detallar varios niveles de diseños correspondiente al sistema software, desde su modelo arquitectónico, pasando por el modelo de datos y diagrama de clases, hasta la vista de las pantallas o diseño de las interfaces de usuario. cada uno de los modelos representa un nivel diferente del software y una etapa distinta de implementación de éste, aunque algunos de ellos se entrelazan entre sí.

Luego de ésta etapa, se poseerá una base sólida para el desarrollo e implementación del sistema Ramtun.

6.2. Arquitectura del sistema

Para comprender que es y el papel que desempeña la arquitectura en un sistema se cita un extracto del autor Ian Sommerville [18]:

El diseño arquitectónico es un proceso creativo en el que se intenta establecer una organización del sistema que satisfaga los requerimientos funcionales y no funcionales del propio sistema. Debido a que es un proceso creativo, las actividades dentro del proceso difieren radicalmente dependiendo del tipo de sistema a desarrollar, el conocimiento y la experiencia del arquitecto del sistema, y los requerimientos específicos del mismo. Es, por tanto, más útil pensar en el proceso de diseño arquitectónico desde una perspectiva de decisión en lugar de una perspectiva de actividades. Durante el proceso de diseño arquitectónico, los arquitectos del sistema tiene que tomar varias decisiones fundamentales que afectan profundamente al sistema y a su proceso de desarrollo. Basándose en su conocimiento y experiencia, los arquitectos del sistema tienen que responder a las siguientes cuestiones fundamentales:

- ¿Existe una arquitectura de aplicación genérica que pueda actuar como una plantilla para el sistema que se está diseñando?
- ¿Cómo se distribuirá el sistema entre varios procesadores?
- ¿Qué estilo o estilos arquitectónicos son apropiados para el sistema?
- ¿Cuál será la aproximación fundamental utilizada para estructurar el sistema?
- ¿Cómo se descompondrán en módulos las unidades estructurales del sistema?
- ¿Qué estrategia se usará para controlar el funcionamiento de las unidades del sistema?
- ¿Cómo se evaluará el diseño arquitectónico?
- ¿Cómo debería documentarse la arquitectura del sistema?

Para el sistema Ramtun, en el caso de la arquitectura física el modelo mas apropiado es el cliente servidor (figura 6.1) y la arquitectura lógica se optó por el modelo en capas, específicamente el "Modelo Vista Controlador" (MVC - Model View Controller) (figura 6.2) esto debido a que como se expuso previamente, la solución a implementar es basada en la WEB.

6.2.1. Arquitectura física

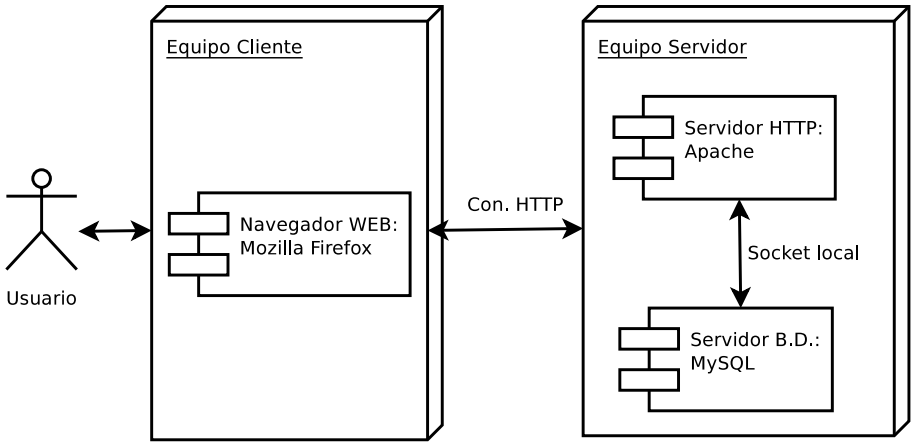

Figura 6.1: Arq. Cliente-servidor ocupada en el sistema Ramtun

6.2.2. Arquitectura lógica

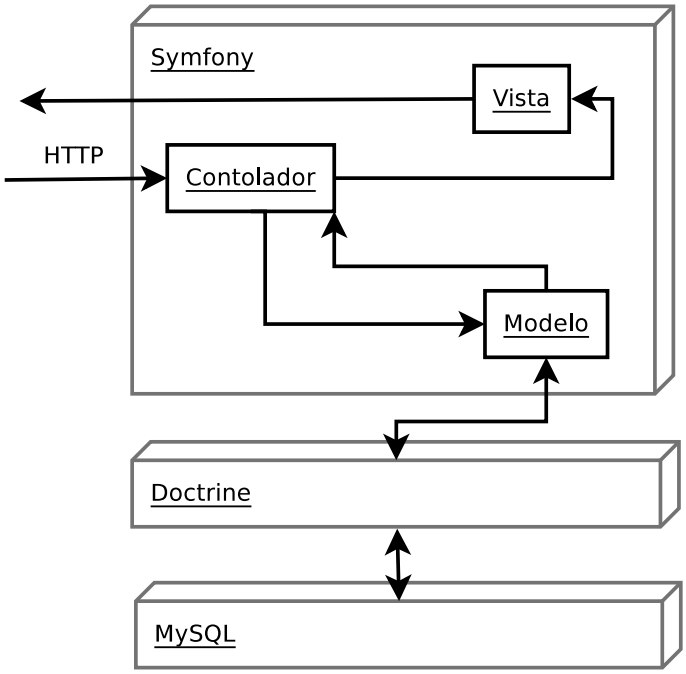

Figura 6.2: Arq. Modelo Vista Controlador ocupada en el sistema Ramtun

6.3. Modelo entidad relación

6.3.1. Modelo lógico

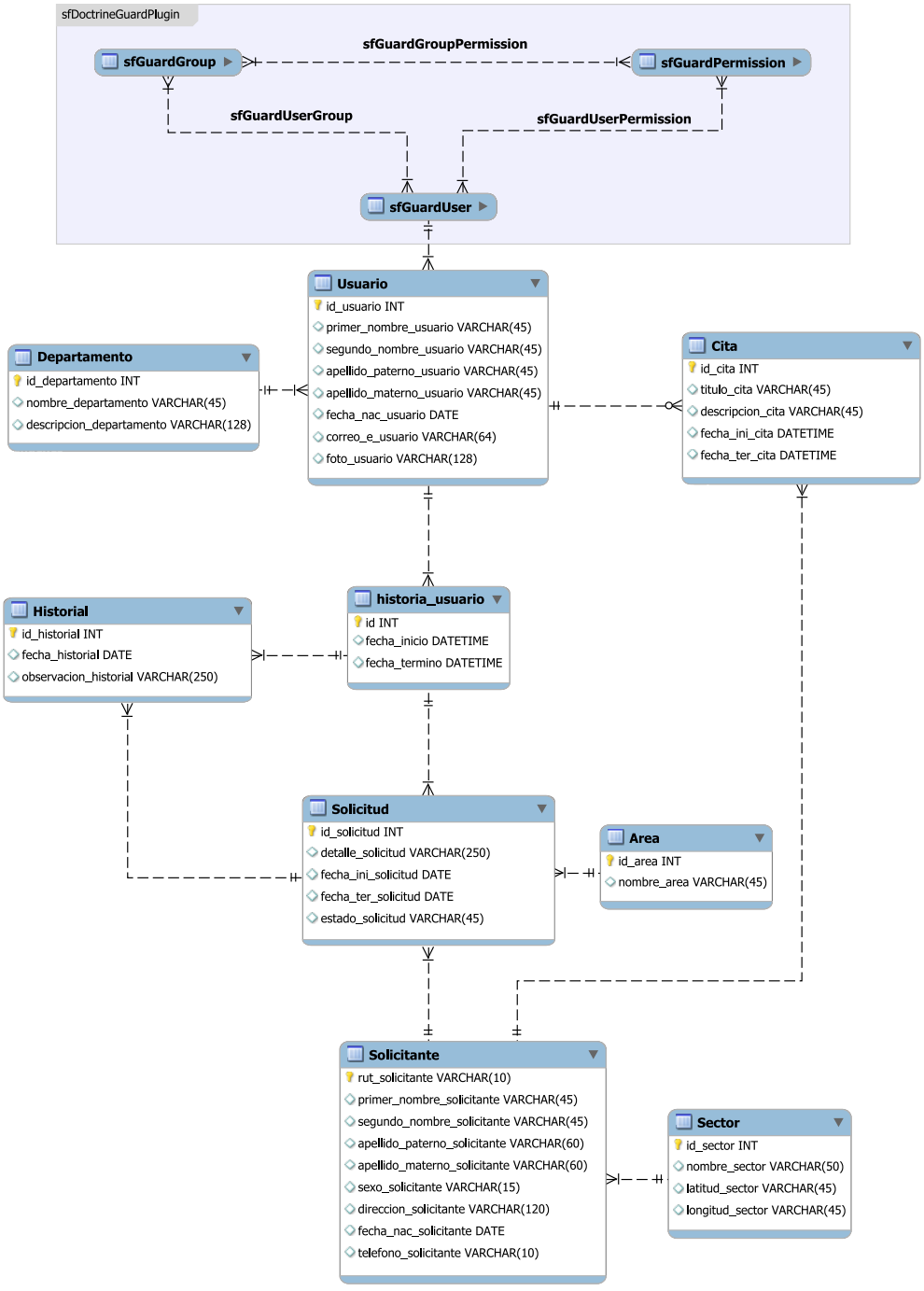

Figura 6.3: Modelo lógico de las base de datos del sistema Ramtun

6.3.2. Modelo físico

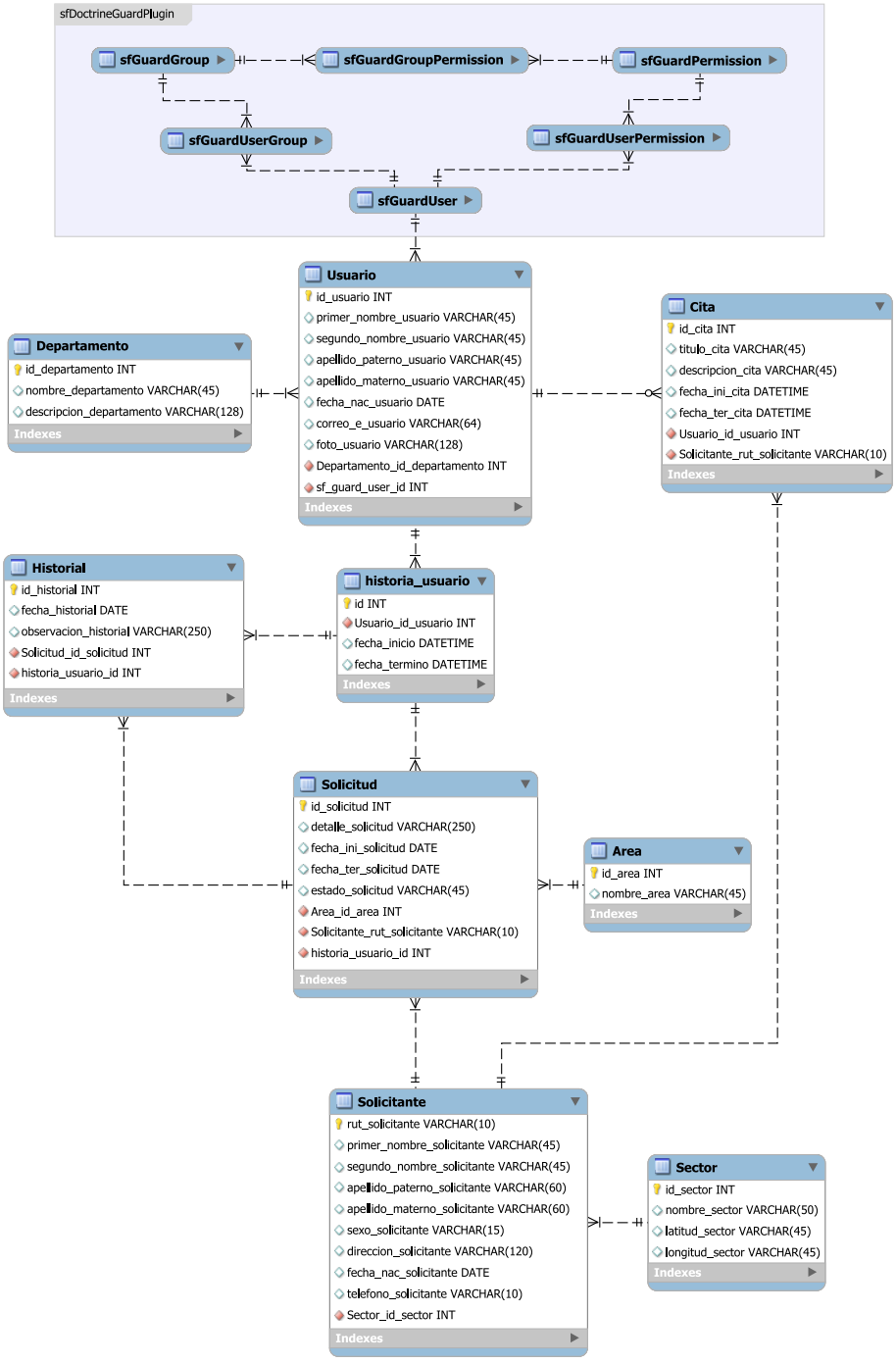

Figura 6.4: Modelo físico de las base de datos del sistema Ramtun

6.4. Casos de uso

6.4.1. Actores

A continuación se listan los actores que participan directamente con el sistema, describiendo brevemente su naturaleza y ordenados según el grado de privilegios con el que interactúan en el mismo.

- **Administrador** Es la máxima autoridad de los usuarios existentes, ya que posee todos los privilegios sobre el sistema.
- **Alcalde** Como su nombre lo indica se refiere al Alcalde mismo de la municipalidad. Este usuario es mas restringido que el administrador, como característica principal puede generar informes.
- **Encargado** Este usuario se refiere al encargado de recepcionar la solicitud, existe 1 persona por cada departamento, es mas restringido que el Alcalde.

6.4.2. Casos de uso general

A continuación se muestra en la figura 6.5 los el diagrama de casos de uso general donde se listan las funcionalidades de manera global.

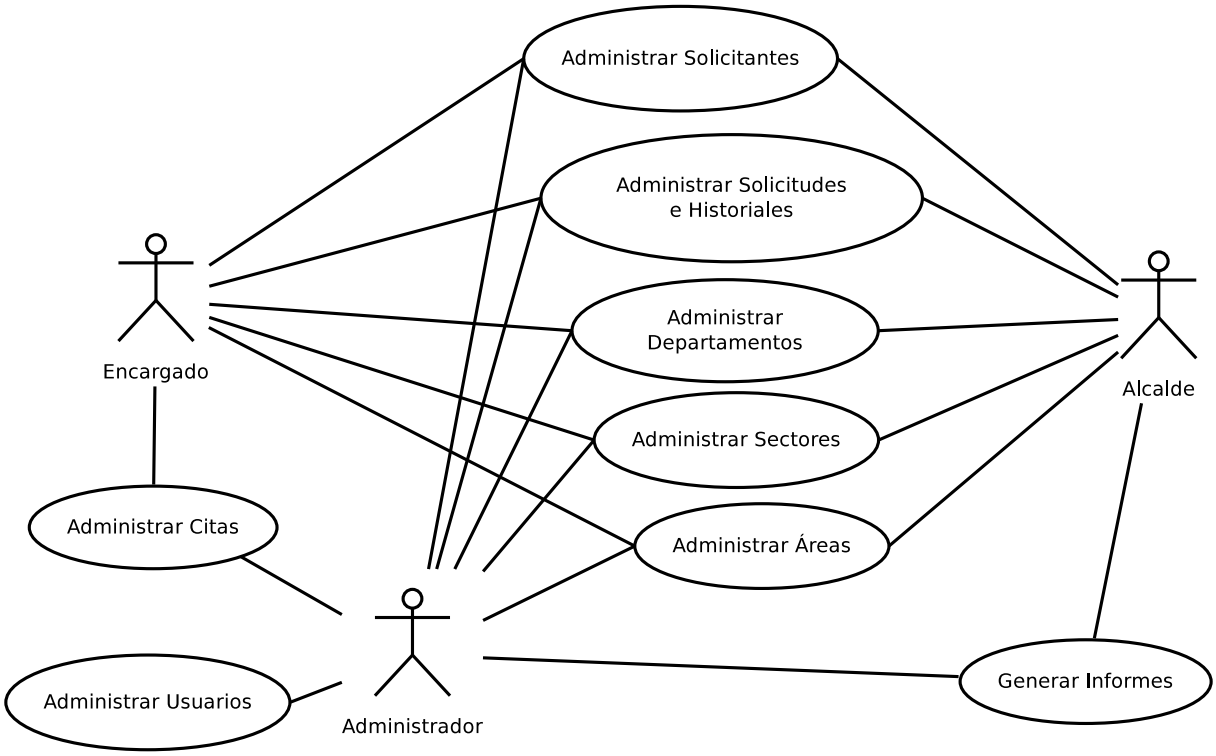

Figura 6.5: Diagrama de casos de uso general

6.4.3. Casos de uso específicos

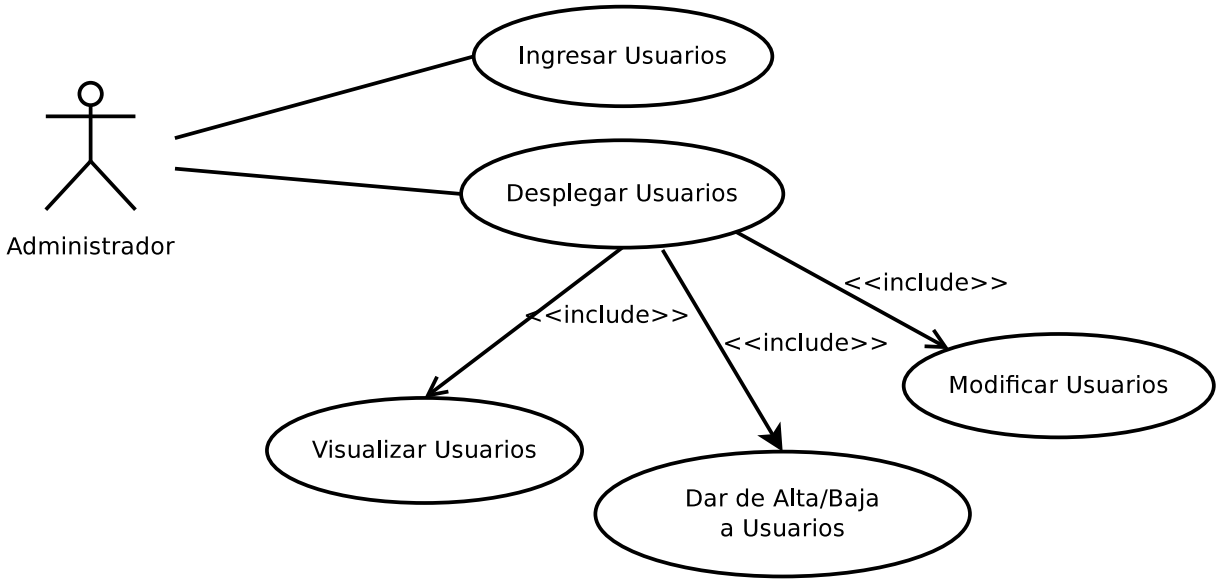

Figura 6.6: Administrar usuarios

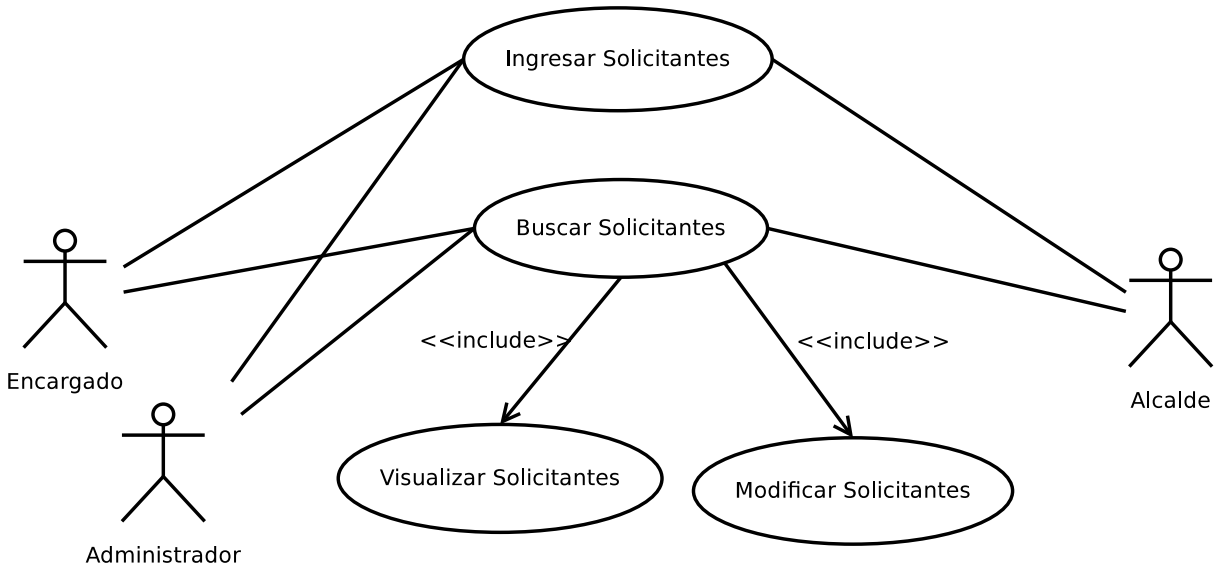

Figura 6.7: Administrar solicitantes

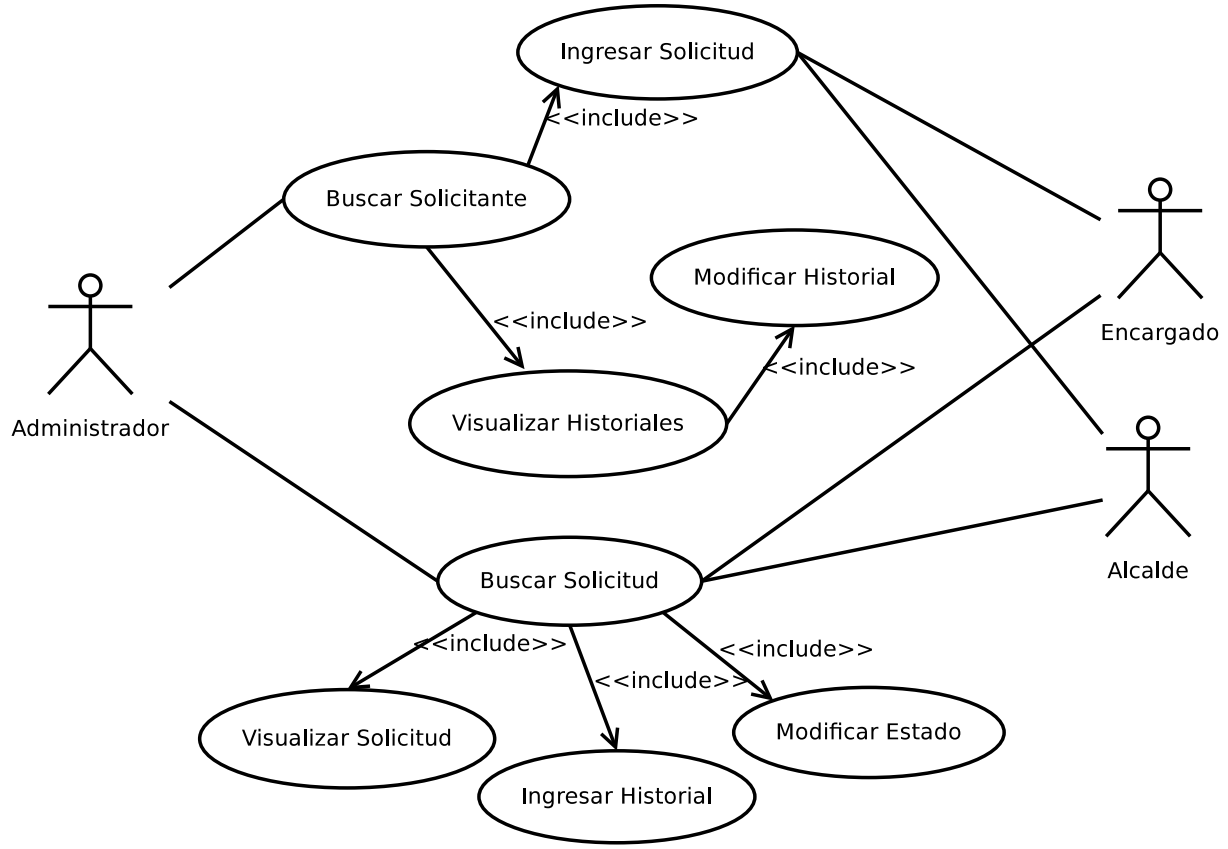

Figura 6.8: Administrar solicitud e historial

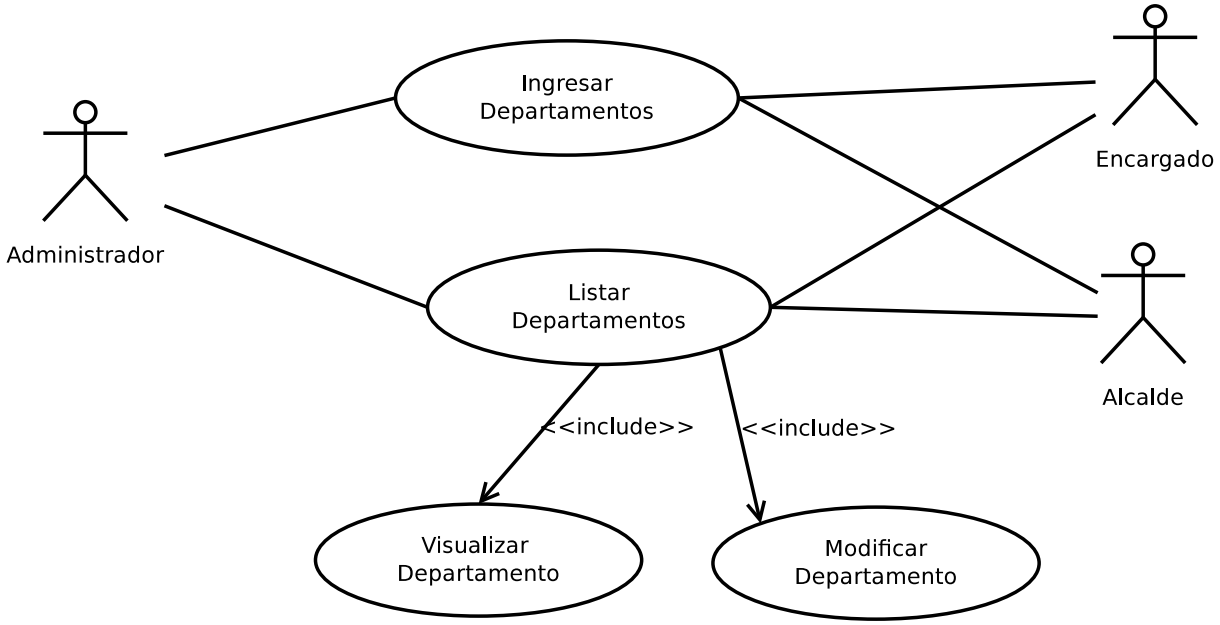

Figura 6.9: Administrar departamentos

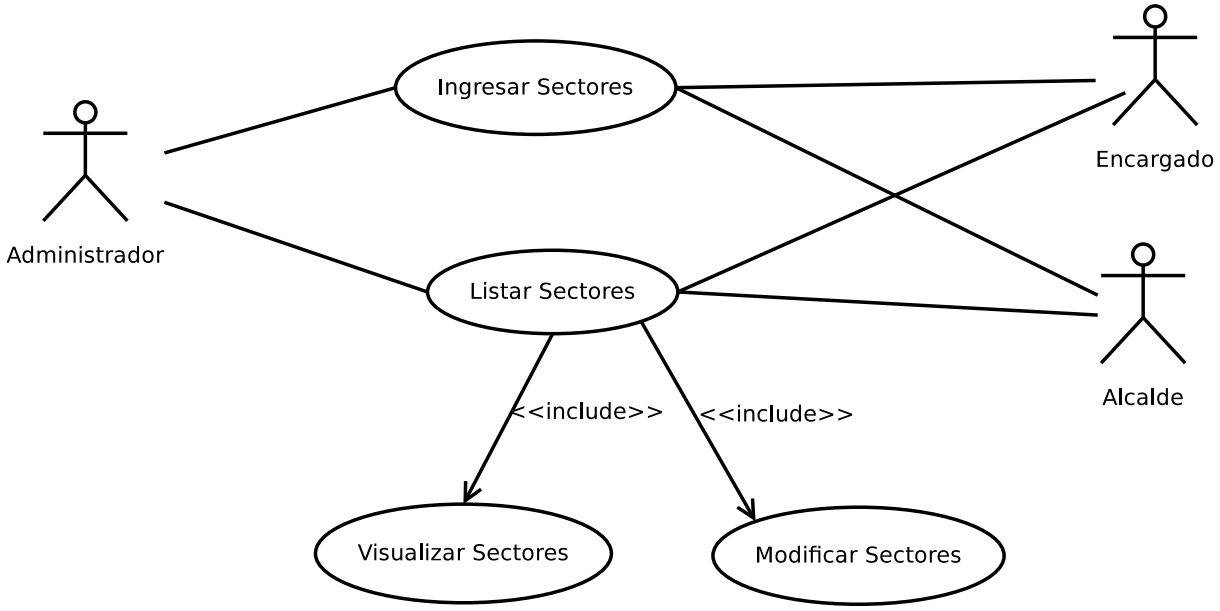

Figura 6.10: Administrar sectores

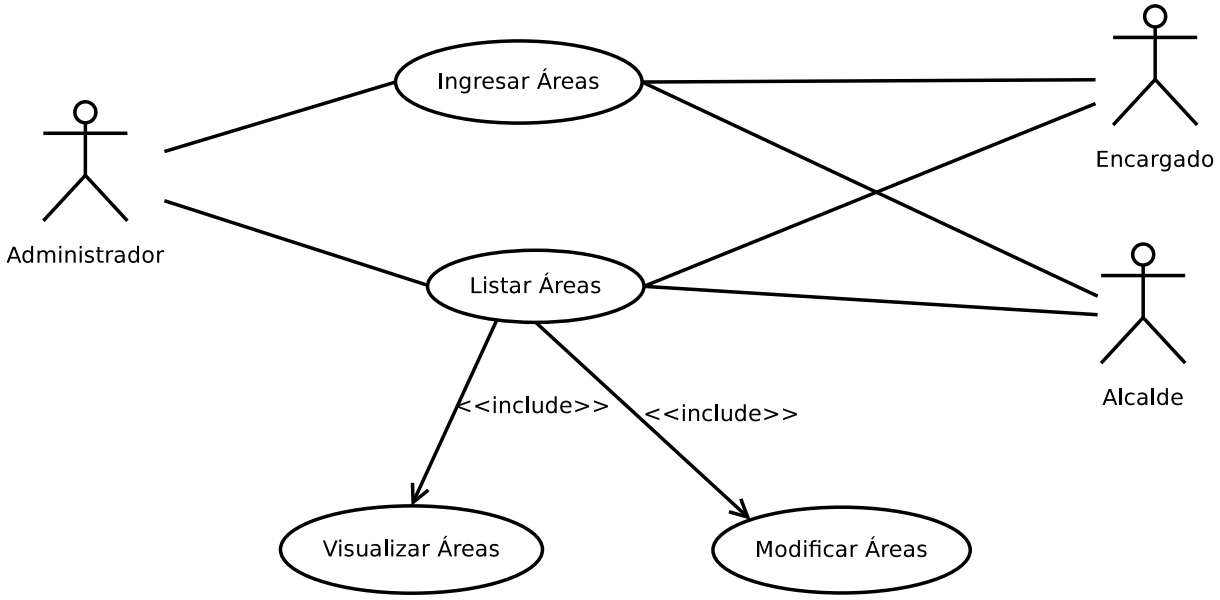

Figura 6.11: Administrar áreas

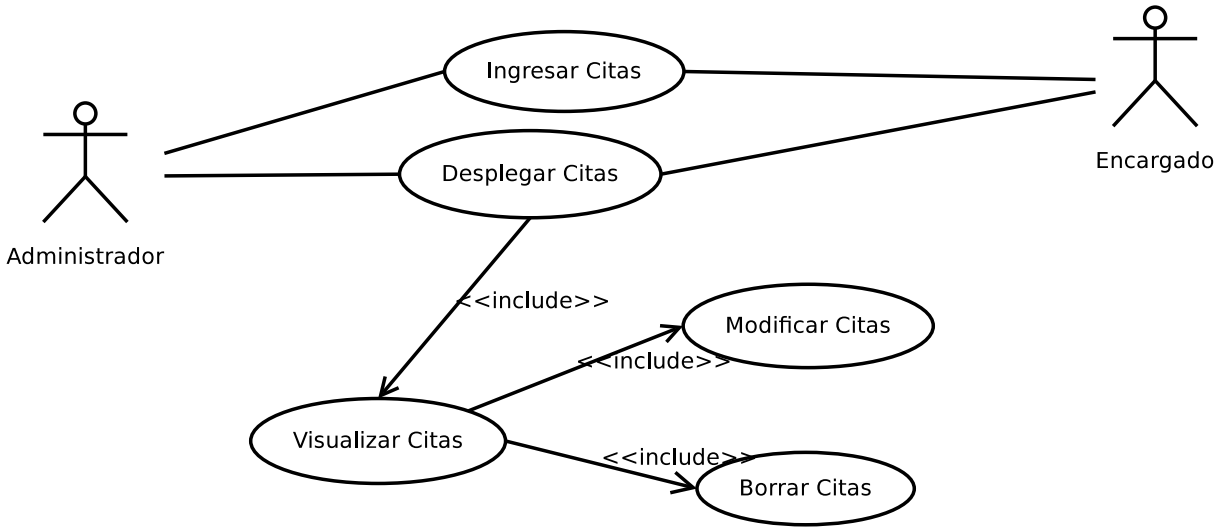

Figura 6.12: Administrar citas

6.4.4. Casos de uso extendidos

En esta sección se detallarán sólo los casos de uso más importantes para el sistema, el resto se puede consultar en el anexo B.

Nombre Caso de Uso: Ingreso al sistema	
Id:	CU00A_IS.
Actores:	Administrador, Encargado, Alcalde.
Resumen:	Este caso de uso describe el login o ingreso al sistema.
Pre-Condición:	
Flujo normal	
1- El usuario abre el navegador y coloca la URL del sistema. **3**- El usuario coloca su nombre de usuario y su contraseña luego selecciona entrar.	**2**- El sistema despliega la pantalla de acceso o login. **4**- El sistema procesa la autenticación con la información enviada y en caso afirmativo, entra al sistema desplegando la pantalla de bienvenida (Home).
Flujo Alternativo:	**4a**- En caso negativo el sistema vuelve a la pantalla de login y muestra un mensaje informando de lo ocurrido
Post-Condición:	

Nombre Caso de Uso: Ingresar Usuario	
Id:	CU01_IU.
Actores:	Administrador.
Resumen:	Este caso de uso describe el proceso de registro de un nuevo usuario al sistema.
Pre-Condición:	Para realizar esta función, el usuario Administrador debe estar previamente logeado al sistema **CU00A_IS**.

Flujo normal	
1- El administrador selecciona la categoría usuarios.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los usuarios.
3- El administrador selecciona la opción de ingresar un nuevo usuario.	
	4- El sistema despliega un formulario con los datos necesarios para el registro del usuario.
5- El administrador completa los datos solicitados y luego selecciona "Guardar".	
	6- El Sistema procesa los datos almacenando el usuario en la base de datos, luego despliega un mensaje embebido informando el resultado de la operación.

Flujo Alternativo:	**6a**- En caso de que la operación sea satisfactoria, los datos del nuevo usuario quedan ingresados al sistema, luego se informa que la operación fue llevada con éxito. **6b**- En caso de que la petición no sea satisfactoria el sistema vuelva al formulario de ingreso y se informa al administrador que la operación no fue realizada.
Post-Condición:	

Nombre Caso de Uso: Listar Usuarios	
Id:	CU02_LU.
Actores:	Administrador.
Resumen:	Este caso de uso describe el proceso de listar los usuarios, previamente registrados, del sistema.
Pre-Condición:	Para realizar esta función, el usuario Administrador debe estar previamente logeado al sistema **CU00A_IS**.
Flujo normal	
1- El administrador selecciona la categoría usuarios.	**2**- El sistema despliega una pantalla con las opciones posibles a realizar con los usuarios. Junto con el listado de usuarios ingresados al sistema.
Flujo Alternativo:	
Post-Condición:	

Nombre Caso de Uso: Visualizar Usuarios	
Id:	CU03_VU.
Actores:	Administrador.
Resumen:	Este caso de uso describe el proceso de ver los datos de los usuarios, previamente registrados, del sistema.
Pre-Condición:	Para realizar esta función, el usuario Administrador debe estar previamente logeado al sistema **CU00A_IS**, **CU02_LU**

Flujo normal	
1- El administrador selecciona la categoría usuarios.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los usuarios. Junto con el listado de usuarios ingresados al sistema.
3- El administrador identifica el usuario a visualizar dentro del listado, luego selecciona la opción "ver usuario".	
	4- El sistema procesa la petición y muestra una pantalla con los datos del usuario.

Flujo Alternativo:	
Post-Condición:	

Nombre Caso de Uso: Dar de Alta/Baja a un Usuario	
Id:	CU04_BU.
Actores:	Administrador.
Resumen:	Este caso de uso describe el proceso de dar de alta o baja a los usuarios previamente registrados en el sistema.
Pre-Condición:	Para realizar esta función, el usuario Administrador debe estar previamente logeado al sistema **CU00A_IS**, **CU02_LU**

Flujo normal	
1- El administrador selecciona la categoría usuarios.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los usuarios. Junto con el listado de usuarios ingresados al sistema.
3- El administrador identifica el usuario a dar de baja dentro del listado, luego selecciona la opción "dar de baja".	
	4- El sistema procesa la petición y muestra un mensaje embebido con el resultado de la operación.

Flujo Alternativo:	**4a**- En caso de que la operación sea satisfactoria, el usuario queda inhabilitado y se procede a registrar el evento en las tablas de historicidad, luego se informa al usuario que la operación fue llevada con éxito. **4b**- En caso de que la petición no sea satisfactoria se informa al usuario que la operación no fue realizada. **4c**- En caso de haber elegido dar de alta, el usuario vuelve a quedar habilitado en el sistema, y se procede a manejar e inicializar los datos de historicidad.
Post-Condición:	

Nombre Caso de Uso: Modificar Usuarios	
Id:	CU05_MU.
Actores:	Administrador.
Resumen:	Este caso de uso describe el proceso de modificar los usuarios, previamente registrados, del sistema.
Pre-Condición:	Para realizar esta función, el usuario Administrador debe estar previamente logeado al sistema **CU00A_IS**, **CU02_LU**

Flujo normal	
1- El administrador selecciona la categoría usuarios.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los usuarios. Junto con el listado de usuarios ingresados al sistema.
3- El administrador identifica el usuario a modificar dentro del listado, luego selecciona la opción "modificar".	
	4- El sistema procesa la petición y muestra un mensaje embebido con el resultado de la operación.

Flujo Alternativo:	**4a**- En caso de que la operación sea satisfactoria, los datos del usuario quedan actualizados, luego se informa que la operación fue llevada con éxito. **4b**- En caso de que la petición no sea satisfactoria se informa al usuario que la operación no fue realizada.
Post-Condición:	

Nombre Caso de Uso: Ingresar Solicitante	
Id:	CU06_IS.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de ingresar un nuevo solicitante al sistema.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**.

Flujo normal	
1- El usuario selecciona la categoría solicitantes.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los solicitantes. Junto con un breve listado de solicitantes previamente ingresados al sistema.
3- El usuario selecciona la opción "ingresar solicitante"	
	4- El sistema despliega la pantalla con un formulario con los datos necesarios para el ingreso del solicitante.
5- El usuario procede al llenado de los datos luego selecciona "guardar"	
	6- El sistema procesa la información y luego despliega un mensaje con el resultado de la operación.

Flujo Alternativo:	**6a**- En caso de que la operación sea satisfactoria, los datos del solicitante quedan ingresados al sistema, luego se informa que la operación fue llevada con éxito. **6b**- En caso de que la petición no sea satisfactoria el sistema vuelva al formulario de ingreso y se informa al usuario que la operación no fue realizada.
Post-Condición:	

Nombre Caso de Uso: Buscar Solicitante	
Id:	CU07_BS.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de buscar un solicitante.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**.

Flujo normal	
1- El usuario selecciona la categoría solicitantes.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los solicitantes. Junto con un breve listado de solicitantes previamente ingresados al sistema.
3- El usuario se posiciona en el formulario de búsqueda y llena los campos según el criterio que estime conveniente, luego selecciona buscar'	
	4- El sistema despliega una pantalla con los resultados, en caso de ser necesario distribuidos en diferentes hojas de resultados (resultados paginados).

Flujo Alternativo:	
Post-Condición:	

Nombre Caso de Uso: Visualizar Solicitante	
Id:	CU08_VS.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de ver los datos de un solicitante.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU07_BS**.

Flujo normal	
1- El usuario selecciona la categoría solicitantes.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los solicitantes. Junto con un breve listado de solicitantes previamente ingresados al sistema.
3- El usuario se posiciona sobre el solicitante correspondiente, luego selecciona "visualizar datos"	
	4- El sistema despliega una pantalla de visualización del perfil del solicitante.

Flujo Alternativo:	
Post-Condición:	

Nombre Caso de Uso: Modificar Solicitante	
Id:	CU09_MS.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de modificar los datos de un solicitante.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU07_BS**.

Flujo normal	
1- El usuario selecciona la categoría solicitantes.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los solicitantes. Junto con un breve listado de solicitantes previamente ingresados al sistema.
3- El usuario se posiciona sobre el solicitante correspondiente, luego selecciona "modificar datos"	
	4- El sistema despliega una pantalla con un formulario cargado con los datos actuales del solicitante.
5- El usuario procede a cambiar los datos que estime necesarios luego selecciona "guardar"	
	6- El sistema procesa los datos enviados y procede a desplegar en pantalla el resultado de la operación.

Flujo Alternativo:	**6a**- En caso de ser satisfactorio, los datos del solicitante son actualizados, luego el sistema informa que la operación se realizó satisfactoriamente. **6b**- En caso de no ser satisfactorio, el sistema informa que la operación no se llevó a cabo.
Post-Condición:	

Nombre Caso de Uso: Ingresar Solicitud	
Id:	CU10_IS.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de ingresar una nueva solicitud al sistema.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU07_BS**.

Flujo normal	
1- El usuario selecciona la categoría solicitantes.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los solicitantes. Junto con un breve listado de solicitantes previamente ingresados al sistema.
3- El usuario se posiciona sobre el solicitante correspondiente, luego selecciona "Ingresar solicitud"	
	4- El sistema despliega una pantalla con un formulario con los datos necesario para ingresar la nueva solicitud.
5- El usuario procede a llenar los datos que estime necesarios luego selecciona "guardar"	
	6- El sistema procesa los datos enviados y procede a desplegar en pantalla el resultado de la operación.

Flujo Alternativo:	**6a**- En caso de ser satisfactorio, la nueva solicitud es ingresada al sistema, luego el sistema informa que la operación se realizó satisfactoriamente. **6b**- En caso de no ser satisfactorio, el sistema informa que la operación no se llevó a cabo.
Post-Condición:	

Nombre Caso de Uso: Buscar Solicitud	
Id:	CU11_IS.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de buscar una solicitud.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**.

Flujo normal	
1- El usuario selecciona la categoría solicitud.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con las solicitudes. Junto con un breve listado de solicitudes previamente ingresados al sistema.
3- El usuario ve si la solicitud está en pantalla de no ser así se posiciona sobre el formulario de búsqueda, llenando con los datos que estime conveniente, luego selecciona "buscar"	
	4- El sistema procesa los datos enviados y procede a desplegar en pantalla las solicitudes que la búsqueda arrojo como resultado, de ser demasiadas los resultados se desplegarán paginados .

Flujo Alternativo:	
Post-Condición:	

Nombre Caso de Uso: Visualizar Solicitud	
Id:	CU12_VS.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de buscar un solicitud.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU11_IS**, **CU07_BS**.

Flujo normal	
1- El usuario selecciona la categoría solicitud.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con las solicitudes. Junto con un breve listado de solicitudes previamente ingresados al sistema.
3- El usuario se posiciona sobre la solicitud y luego selecciona "Visualizar solicitud"	
	4- El sistema despliega en pantalla los datos de la solicitud seleccionada.

Flujo Alternativo:	**1a**- El usuario selecciona Buscar solicitante, se posiciona sobre el solicitante elegido luego selecciona ver listado de solicitudes, para luego desplegar un listado con las solicitudes asociadas solamente al solicitante elegido.
Post-Condición:	

Nombre Caso de Uso: Modificar Solicitud	
Id:	CU13_MS.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de modificar una solicitud.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU11_IS**, **CU07_BS**.

Flujo normal	
1- El usuario selecciona la categoría solicitud.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con las solicitudes. Junto con un breve listado de solicitudes previamente ingresados al sistema.
3- El usuario se posiciona sobre la solicitud y luego selecciona "modificar solicitud"	
	4- El sistema despliega en pantalla un formulario cargado con los datos de la solicitud seleccionada.
5- El usuario modifica los datos que estime conveniente, para posteriormente seleccionar "Guardar".	
	6- El sistema procesa los datos y despliega en pantalla el resultado de la operación.

Flujo Alternativo:	**1a**- El usuario selecciona Buscar solicitante, se posiciona sobre el solicitante elegido luego selecciona ver listado de solicitudes, para luego desplegar un listado con las solicitudes asociadas solamente al solicitante elegido, de las cuales selecciona la solicitud a modificar. **6a**- En caso de ser satisfactorio, los datos de la solicitud son actualizado, luego el sistema despliega en pantalla que la operación se realizó con éxito. **6b**- En caso de no ser satisfactorio. el sistema despliega en pantalla que la operación no se llevo a cabo
Post-Condición:	

Nombre Caso de Uso: Ingresar Historial	
Id:	CU14_IH.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de ingresar un nuevo historial.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU11_IS**, **CU07_BS**.

Flujo normal

1- El usuario selecciona la categoría solicitud.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con las solicitudes. Junto con un breve listado de solicitudes previamente ingresados al sistema.
3- El usuario se posiciona sobre la solicitud y luego selecciona "Ingresar historial"	
	4- El sistema despliega en pantalla un formulario con los datos necesarios para el ingreso del historial.
5- El usuario procede a llenar los campos que estime conveniente, para posteriormente seleccionar "Guardar".	
	6- El sistema procesa los datos y despliega en pantalla el resultado de la operación.

Flujo Alternativo:	**1a**- El usuario selecciona Buscar solicitante, se posiciona sobre el solicitante elegido luego selecciona ver listado de solicitudes, luego el sistema despliega un listado con las solicitudes asociadas solamente al solicitante elegido, de las cuales se posiciona sobre la solicitud correspondiente luego selecciona "Ingresar Historial". **6a**- En caso de ser satisfactorio, el nuevo historial es ingresado, luego el sistema despliega en pantalla que la operación se realizó con éxito. **6b**- En caso de no ser satisfactorio. el sistema despliega en pantalla que la operación no se llevo a cabo
Post-Condición:	

Nombre Caso de Uso: Visualizar Historiales	
Id:	CU15_VH.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de ver los datos de historiales.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU11_IS**, **CU07_BS**.

Flujo normal	
1- El usuario selecciona la categoría solicitud.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con las solicitudes. Junto con un breve listado de solicitudes previamente ingresados al sistema.
3- El usuario se posiciona sobre la solicitud y luego selecciona "ver historiales asociados"	
	4- El sistema despliega en pantalla un listado con todos los datos de historiales asociados a la solicitud elegida.

Flujo Alternativo:	**1a**- El usuario selecciona Buscar solicitante, se posiciona sobre el solicitante elegido luego selecciona ver listado de solicitudes, para luego desplegar un listado con las solicitudes asociadas solamente al solicitante elegido, luego se posiciona sobre la solicitud elegido y selecciona "ver historiales asociados".
Post-Condición:	

Nombre Caso de Uso: Modificar Historial	
Id:	CU16_MH.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de modificar un historial.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU11_IS**, **CU07_BS**.

Flujo normal	
1- El usuario selecciona la categoría solicitud.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con las solicitudes. Junto con un breve listado de solicitudes previamente ingresados al sistema.
3- El usuario se posiciona sobre la solicitud y luego selecciona "ver historiales asociados"	
	4- El sistema despliega en pantalla un listado con todos los datos de historiales asociados a la solicitud elegida.
5- El usuario se posiciona sobre el historial y luego selecciona "modificar"	
	6- El sistema despliega en pantalla un formulario con los datos cargados del historial elegido.
7- El usuario modifica los datos que estime necesarios, a continuación selecciona "guardar".	
	8- El sistema procesa los datos y despliega en pantalla el resultado de la operación.

Flujo Alternativo:	**1a**- El usuario selecciona Buscar solicitante, se posiciona sobre el solicitante elegido luego selecciona ver listado de solicitudes, para luego desplegar un listado con las solicitudes asociadas solamente al solicitante elegido, se posiciona sobre la solicitud y selecciona ver historiales para finalmente seleccionar el historial a modificar. **6a**- En caso de ser satisfactorio, los datos del historial son actualizados, luego el sistema despliega en pantalla que la operación se realizó con éxito. **6b**- En caso de no ser satisfactorio. el sistema despliega en pantalla que la operación no se llevo a cabo.
Post-Condición:	

Nombre Caso de Uso: Ingresar Citas	
Id:	CU29_IC.
Actores:	Administrador, Encargado.
Resumen:	Este caso de uso describe el proceso de ingresar una nueva cita.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**.
Flujo normal	
1- El usuario selecciona la categoría citas.	**2**- El sistema despliega una pantalla tipo agenda con un calendario.
3- El usuario selecciona la opción "Ingresar cita"	**4**- El sistema despliega en pantalla un formulario con los datos necesario para ingresar una nueva cita.
5- El usuario procede a llenar los campos que estime conveniente luego selecciona "guardar"	**6**- El sistema procesa los datos y procede a desplegar en pantalla el resultado de la operación.
Flujo Alternativo:	**6a**- En caso de ser satisfactorio, la nueva cita es ingresada, luego el sistema despliega en pantalla que la operación se realizó con éxito. **6b**- En caso de no ser satisfactorio. el sistema despliega en pantalla que la operación no se llevo a cabo.
Post-Condición:	

Nombre Caso de Uso: Listar citas	
Id:	CU30_LC.
Actores:	Administrador, Encargado.
Resumen:	Este caso de uso describe el proceso de desplegar un listado de citas.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**.
Flujo normal	
1- El usuario selecciona la categoría citas. **3**- El usuario selecciona el día del cual quiere ver las citas.	**2**- El sistema despliega una pantalla tipo agenda con un calendario. **4**- El sistema despliega un segmento en pantalla con todas las citas para el día seleccionado, ordenadas cronológicamente.
Flujo Alternativo:	
Post-Condición:	

Nombre Caso de Uso: Visualizar Citas	
Id:	CU31_VC.
Actores:	Administrador, Encargado.
Resumen:	Este caso de uso describe el proceso de visualizar los datos de una cita.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU30_LC**.

Flujo normal	
1- El usuario selecciona la categoría citas.	
	2- El sistema despliega una pantalla tipo agenda con un calendario.
3- El usuario selecciona el día del cual quiere ver las citas.	
	4- El sistema despliega un segmento en pantalla con todas las citas para el día seleccionado, ordenadas cronológicamente.
5- El usuario selecciona la cita requerida.	
	6- El sistema despliega los datos de la cita.

Flujo Alternativo:	
Post-Condición:	

Nombre Caso de Uso: Modificar Citas	
Id:	CU32_MC.
Actores:	Administrador, Encargado.
Resumen:	Este caso de uso describe el proceso de modificar los datos de una cita.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU31_VC**.

Flujo normal	
1- El usuario selecciona la categoría citas.	
	2- El sistema despliega una pantalla tipo agenda con un calendario.
3- El usuario selecciona el día del cual quiere ver las citas.	
	4- El sistema despliega un segmento en pantalla con todas las citas para el día seleccionado, ordenadas cronológicamente.
5- El usuario selecciona la cita requerida.	
	6- El sistema despliega los datos de la cita.
7- El usuario selecciona "modificar".	
	8- El sistema despliega un formulario cargado con los datos de la cita seleccionada.
9- El usuario procede a cambiar los campos que estime conveniente luego selecciona "guardar".	
	10- El sistema procesa los datos y procede a desplegar en pantalla el resultado de la operación.

Flujo Alternativo:	**10a**- En caso de ser satisfactorio, los datos de la cita son actualizados, luego el sistema despliega en pantalla que la operación se realizó con éxito. **10b**- En caso de no ser satisfactorio. el sistema despliega en pantalla que la operación no se llevo a cabo.
Post-Condición:	

Nombre Caso de Uso: Borrar Citas	
Id:	CU33_BC.
Actores:	Administrador, Encargado.
Resumen:	Este caso de uso describe el proceso de borrar una cita.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU31_VC**.

Flujo normal	
1- El usuario selecciona la categoría citas.	
	2- El sistema despliega una pantalla tipo agenda con un calendario.
3- El usuario selecciona el día del cual quiere ver las citas.	
	4- El sistema despliega un segmento en pantalla con todas las citas para el día seleccionado, ordenadas cronológicamente.
5- El usuario selecciona la cita requerida.	
	6- El sistema despliega los datos de la cita.
7- El usuario selecciona "Borrar".	
	8- El sistema procesa la petición y procede a desplegar en pantalla el resultado de la operación.

Flujo Alternativo:	**8a**- En caso de ser satisfactorio, la cita es eliminada, luego el sistema despliega en pantalla que la operación se realizó con éxito. **8b**- En caso de no ser satisfactorio. el sistema despliega en pantalla que la operación no se llevo a cabo
Post-Condición:	

Nombre Caso de Uso: Generar Informes	
Id:	CU34_GI.
Actores:	Administrador, Alcalde.
Resumen:	Este caso de uso describe el proceso de desplegar informes.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**.

Flujo normal	
1- El usuario selecciona la categoría informes.	
	2- El sistema despliega un listado con los informes disponibles.
3- El usuario selecciona informe que desea consultar.	
	4- El sistema procesa la petición y procede a desplegar el informe en pantalla.
Flujo Alternativo:	**4a**- En caso de seleccionar desplegar georeferencias en vez de mostrar un informe se despliega un mapa de google maps.
Post-Condición:	

6.5. Diagrama de clases

En esta sección se expone el diagrama de clases el cual será presentado en 2 partes, primero el diagrama general y luego por cada clase listada, se incluirá un diagrama exclusivo de la respectiva clase, incluyendo los métodos que contenga. Cabe destacar que el diagrama de clases expuesto esta adaptado para integrase de mejor forma al plugin sfGuard (que facilita el manejo de autenticación), es por ello que se encuentran separados los datos de acceso (sfGuardUser) de los datos del perfil (Usuario).

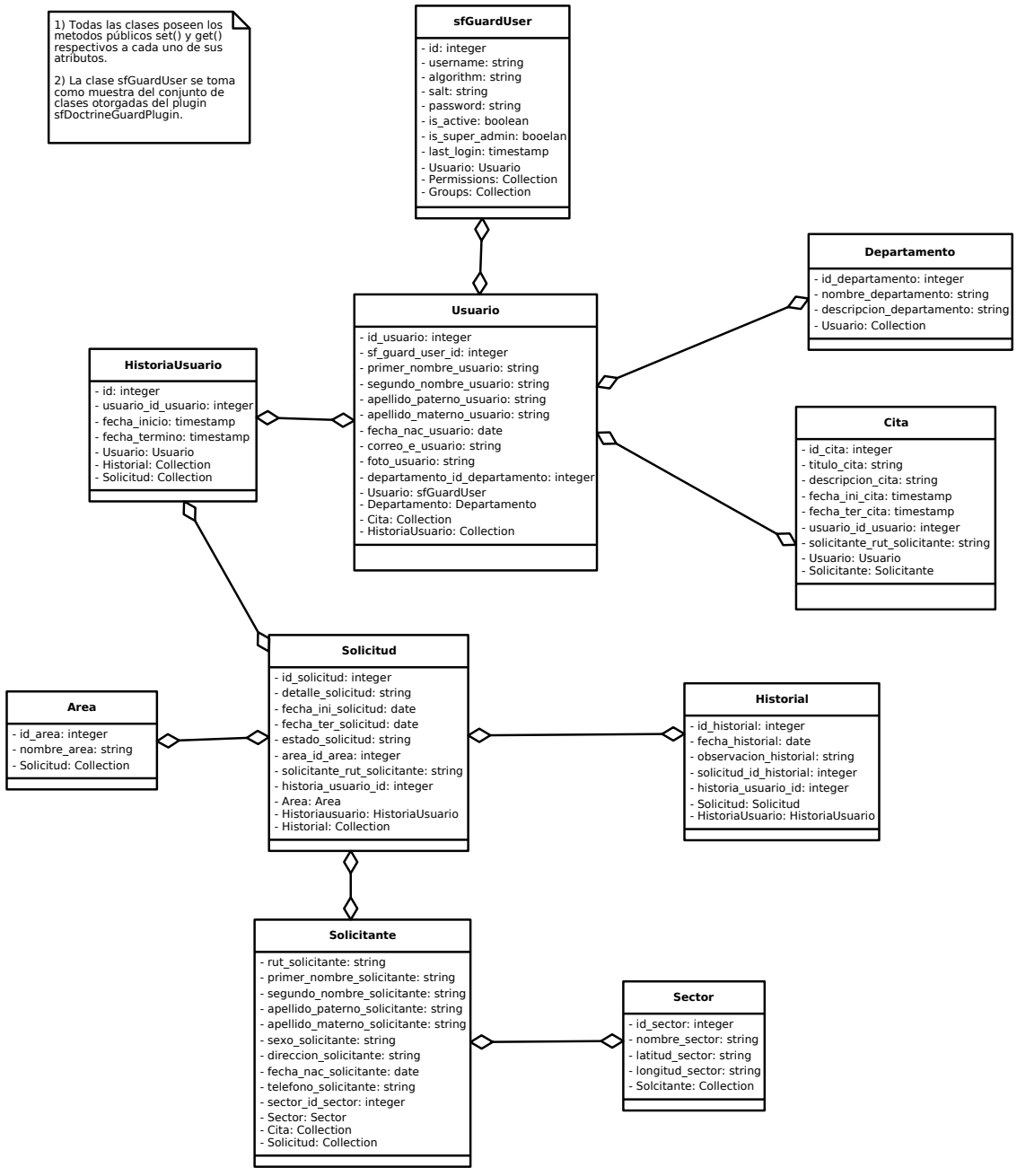

Figura 6.13: Diagrama de Clases

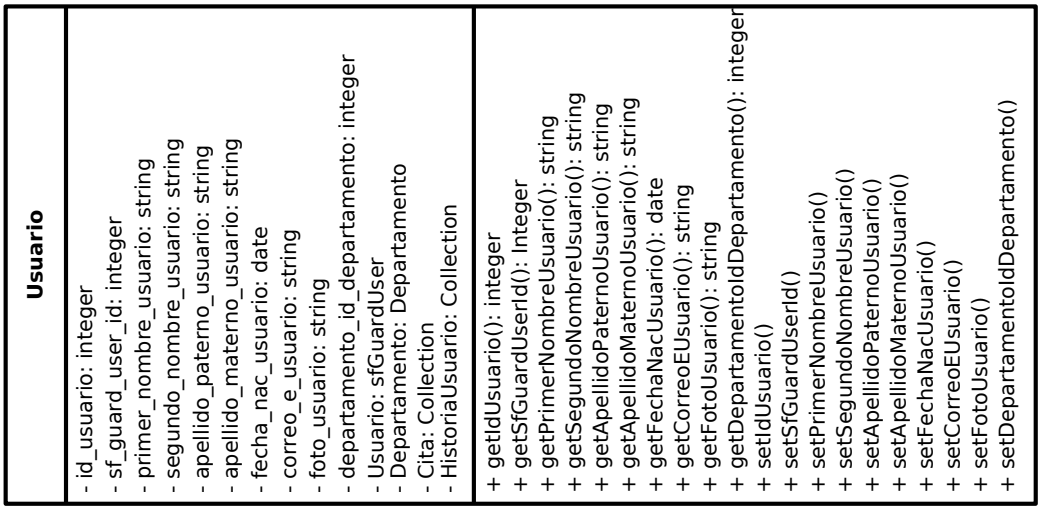

Figura 6.14: Clase Usuario

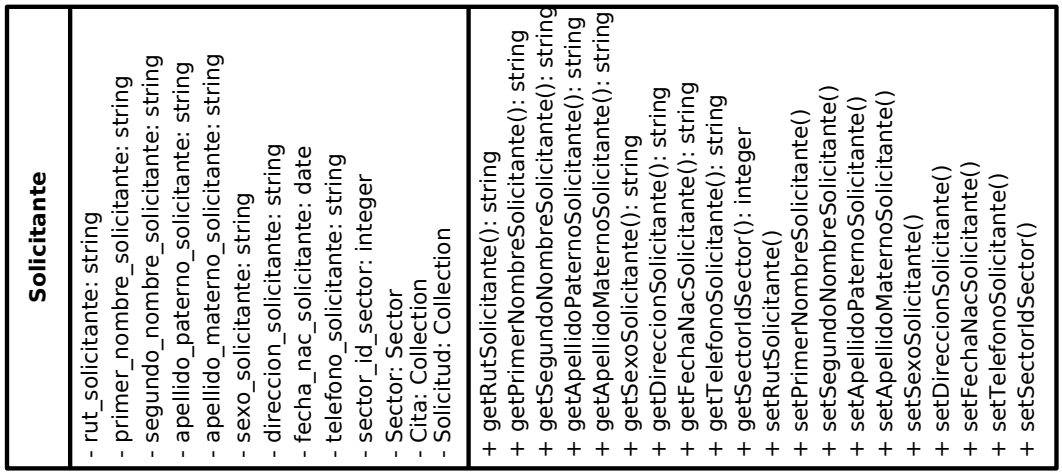

Figura 6.15: Clase Solicitante

Figura 6.16: Clase Solicitud

Figura 6.17: Clase Historial

Figura 6.18: Clase Cita

Figura 6.19: Clase HistoriaUsuario

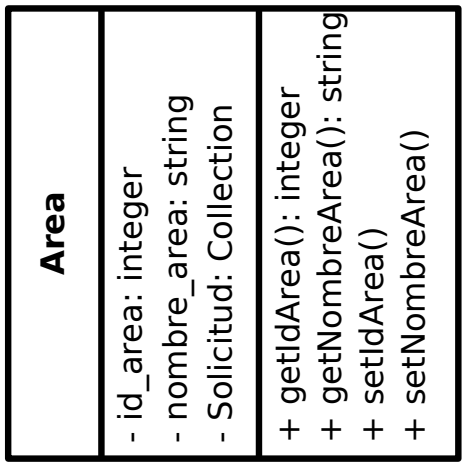

Figura 6.20: Clase Área

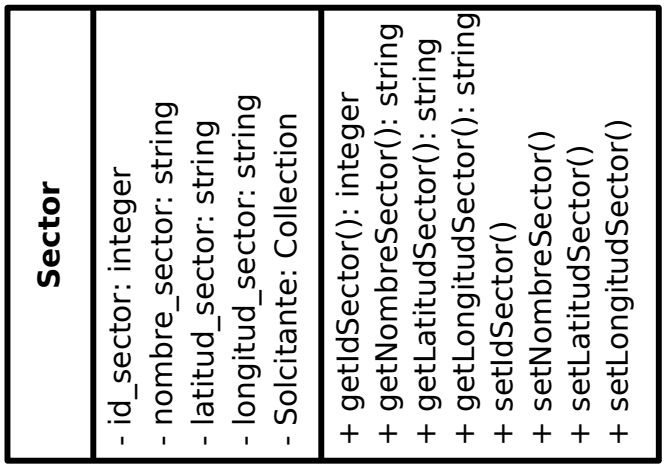

Figura 6.21: Clase Sector

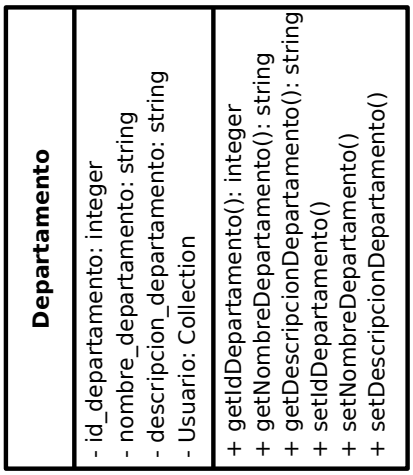

Figura 6.22: Clase Departamento

6.6. Diagramas de secuencia

En esta sección se listaran los diagramas de secuencia asociados a los casos de uso mas importantes además, para fines prácticos, se definirá un nuevo actor llamado **Usuario** el cual puede representar a cualquiera de los 3 actores definidos previamente (Administrador, Alcalde, Encargado).

6.6.1. Usuario

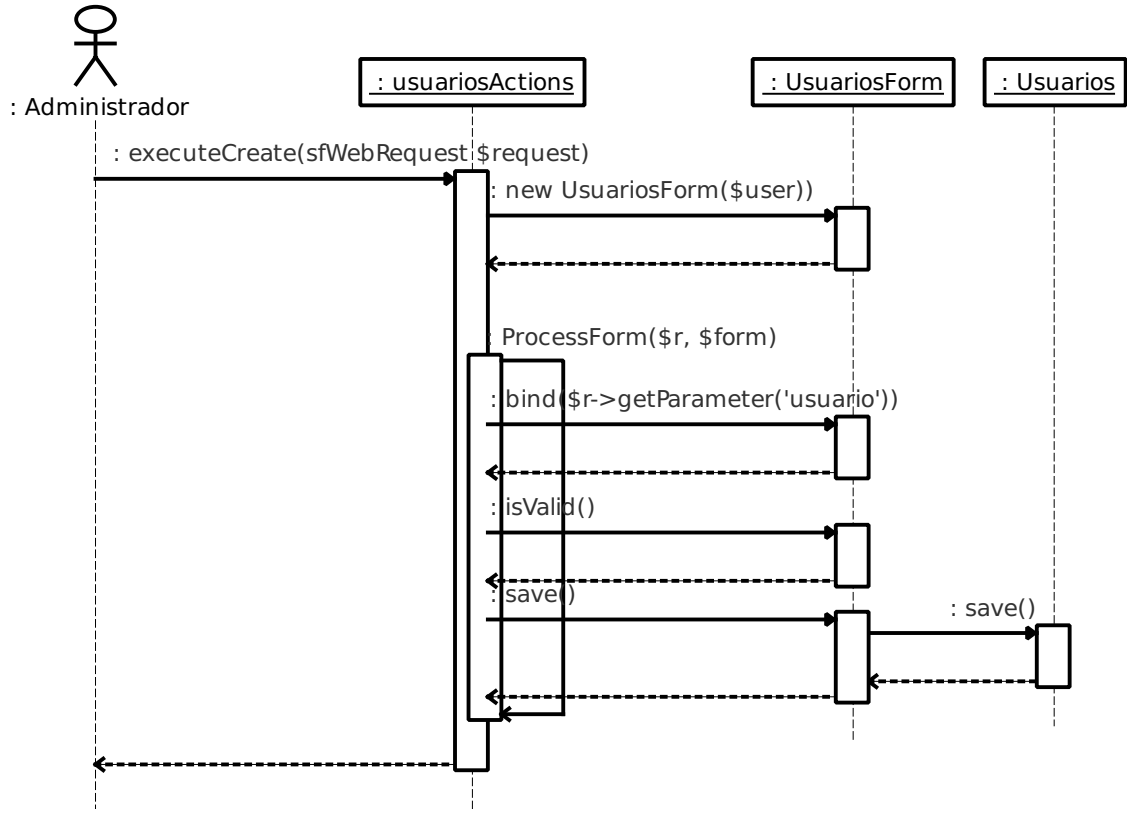

Figura 6.23: Ingresar Usuarios

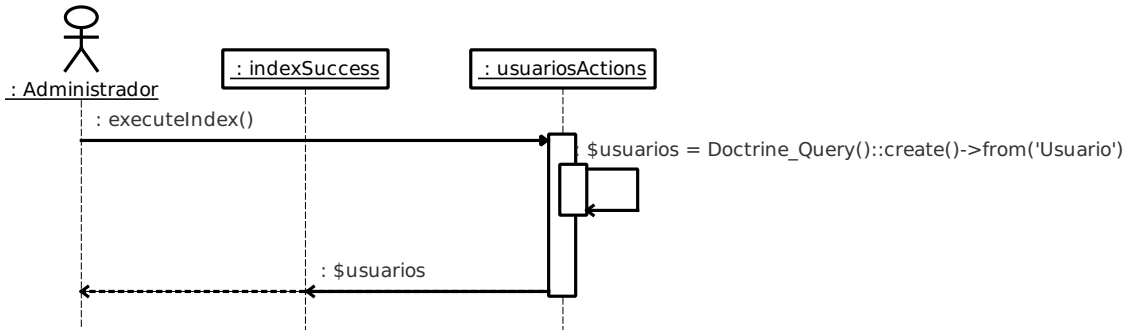

Figura 6.24: Listar Usuarios

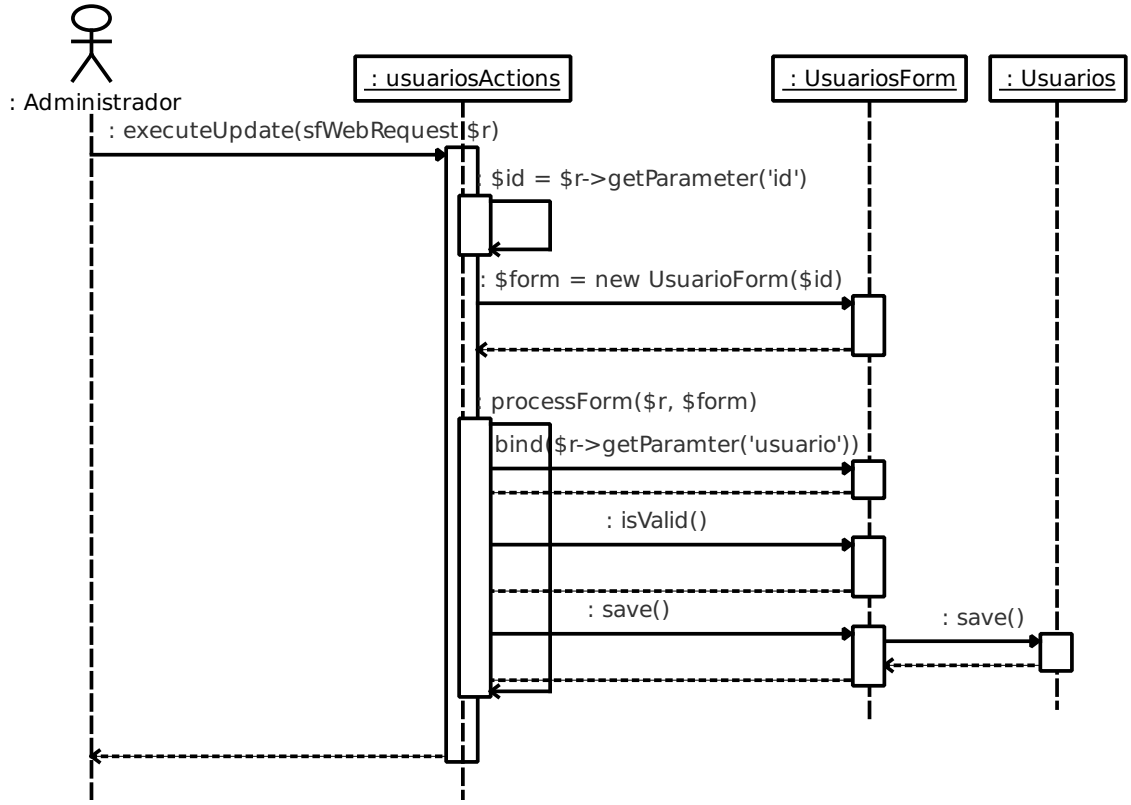

Figura 6.25: Modificar Usuarios

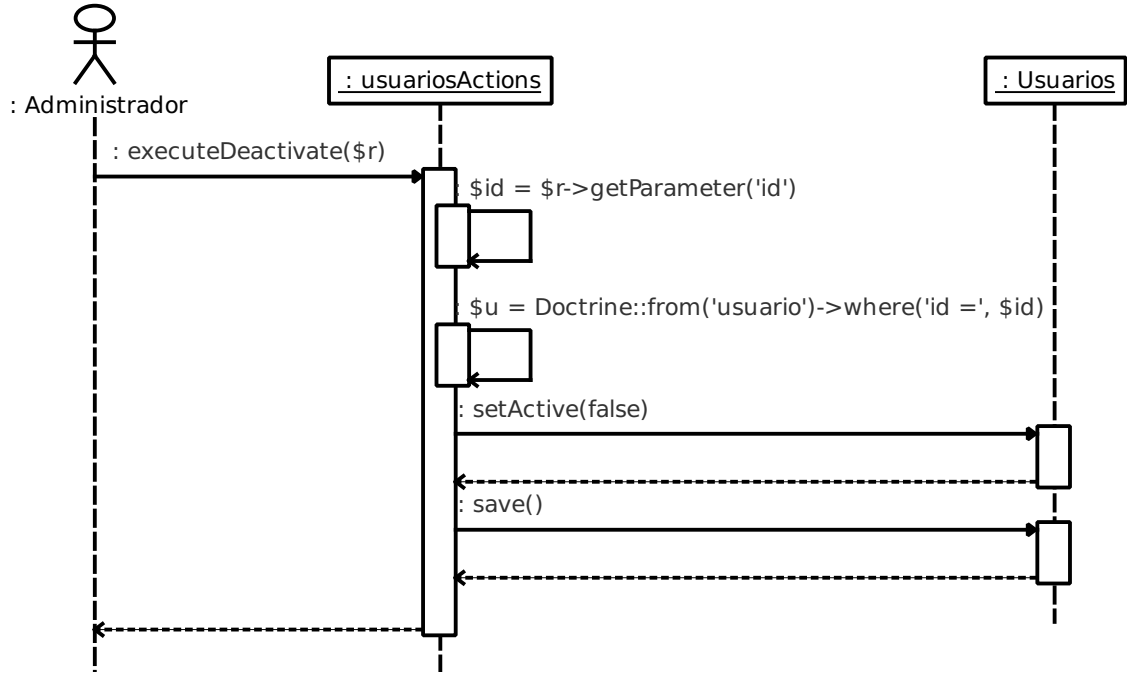

Figura 6.26: Dar de Baja Usuarios

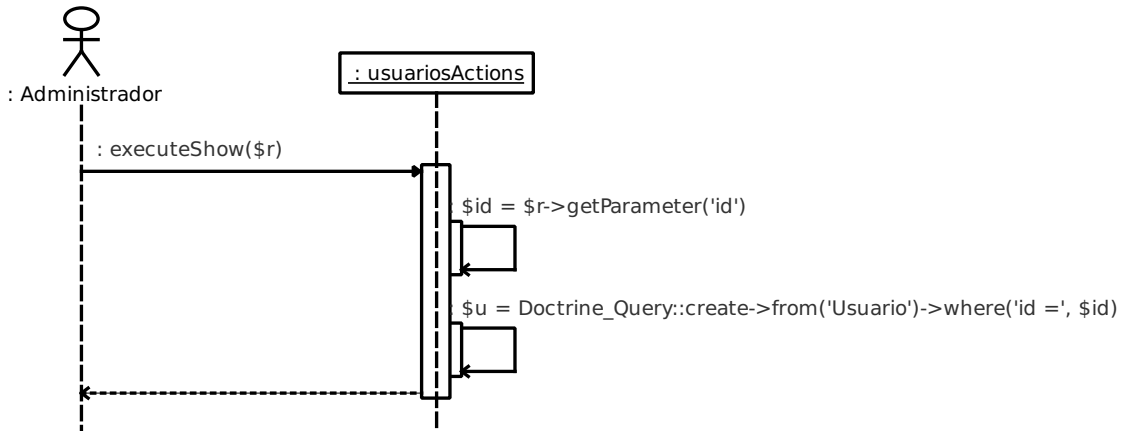

Figura 6.27: Ver Usuarios

6.6.2. Solicitante

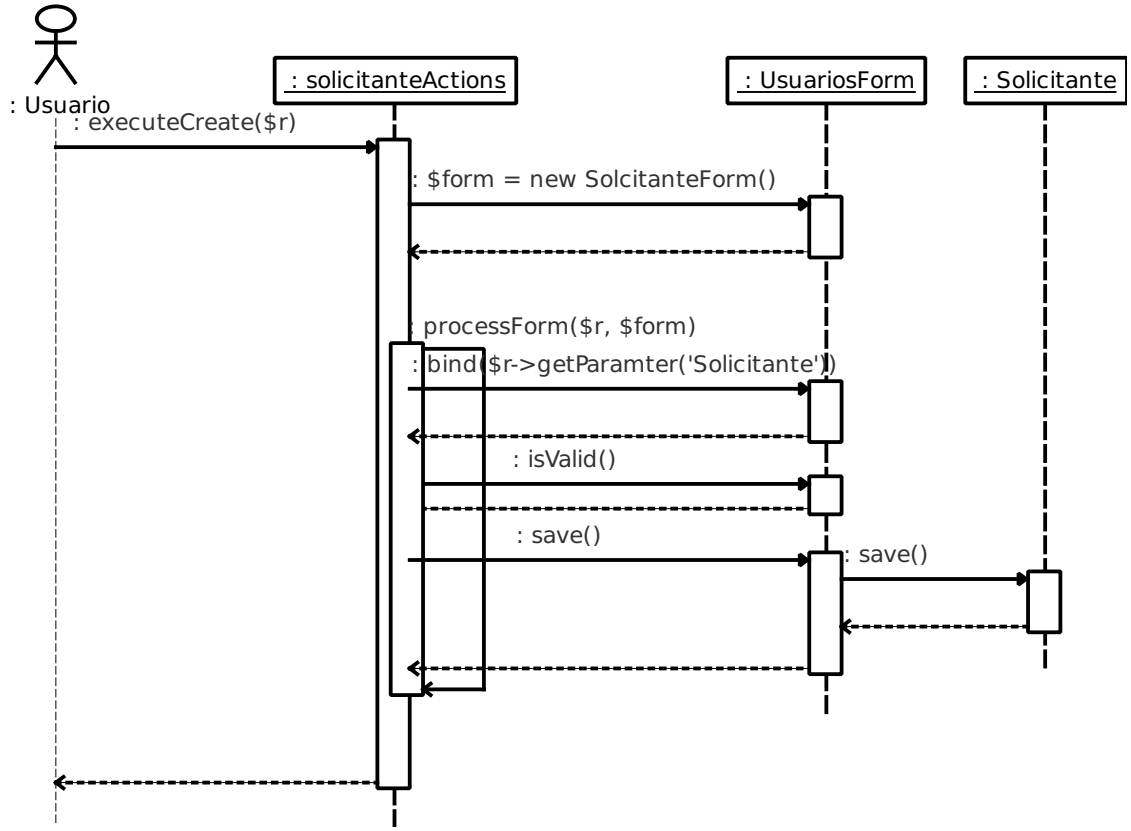

Figura 6.28: Ingresar Solicitante

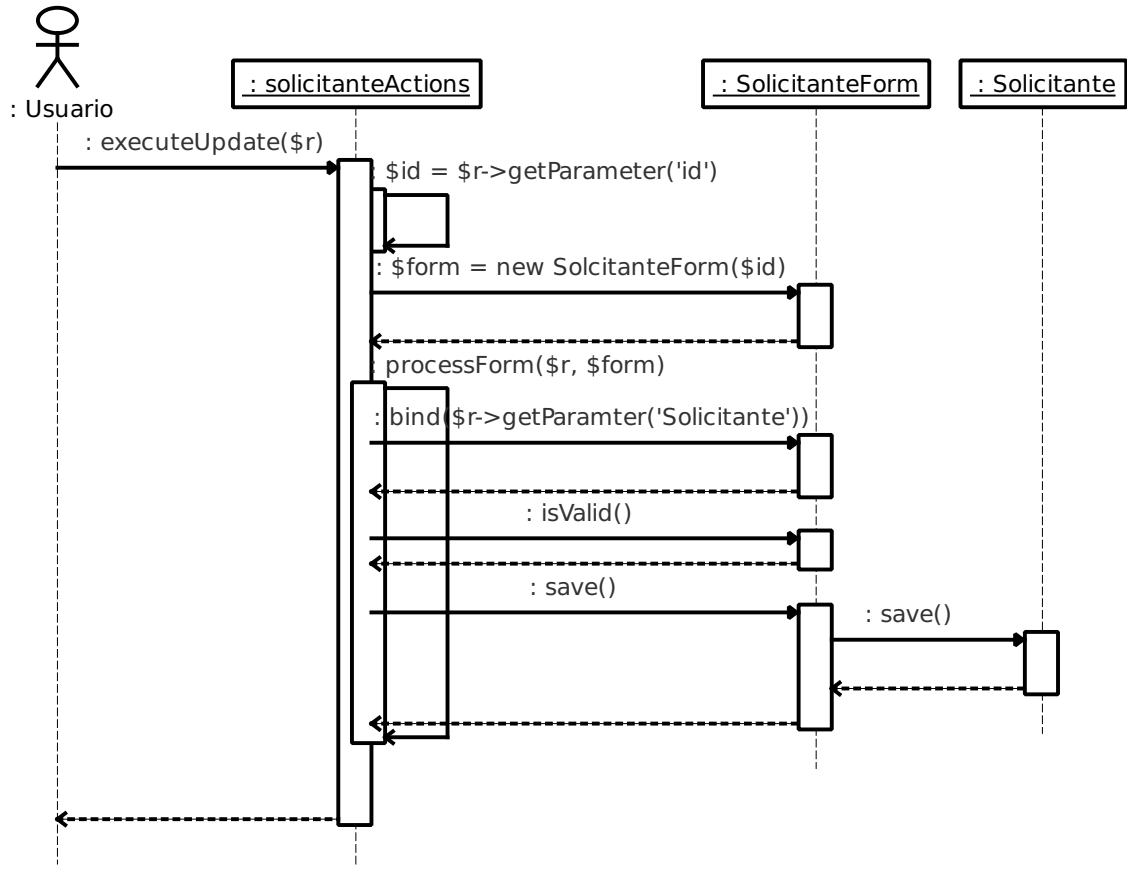

Figura 6.29: Modificar Solicitante

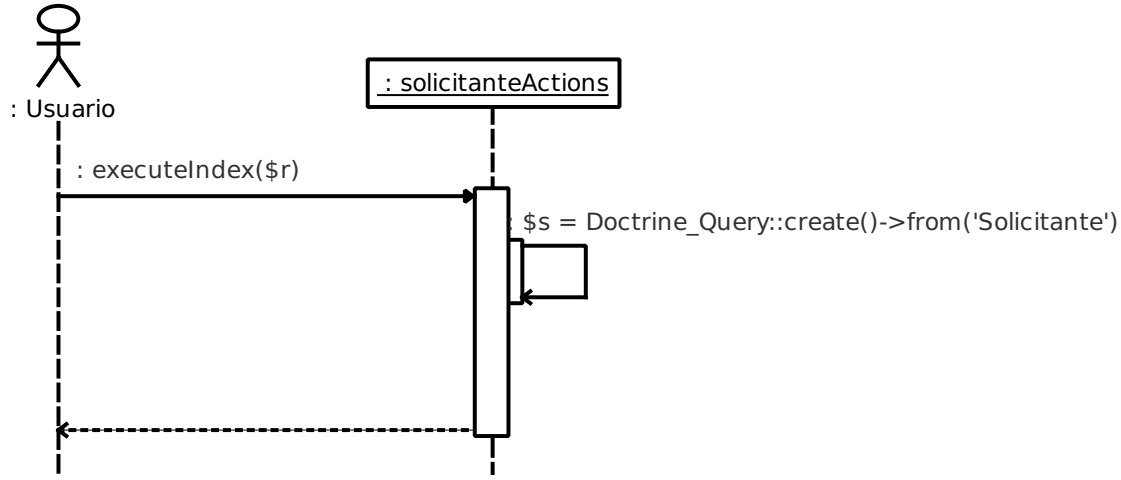

Figura 6.30: Listar Solicitante

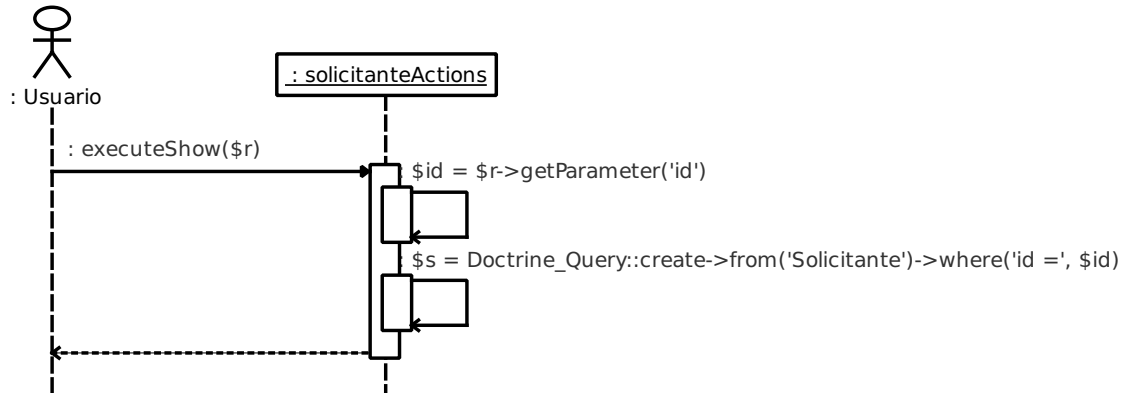

Figura 6.31: Ver Solicitante

6.6.3. Solicitud

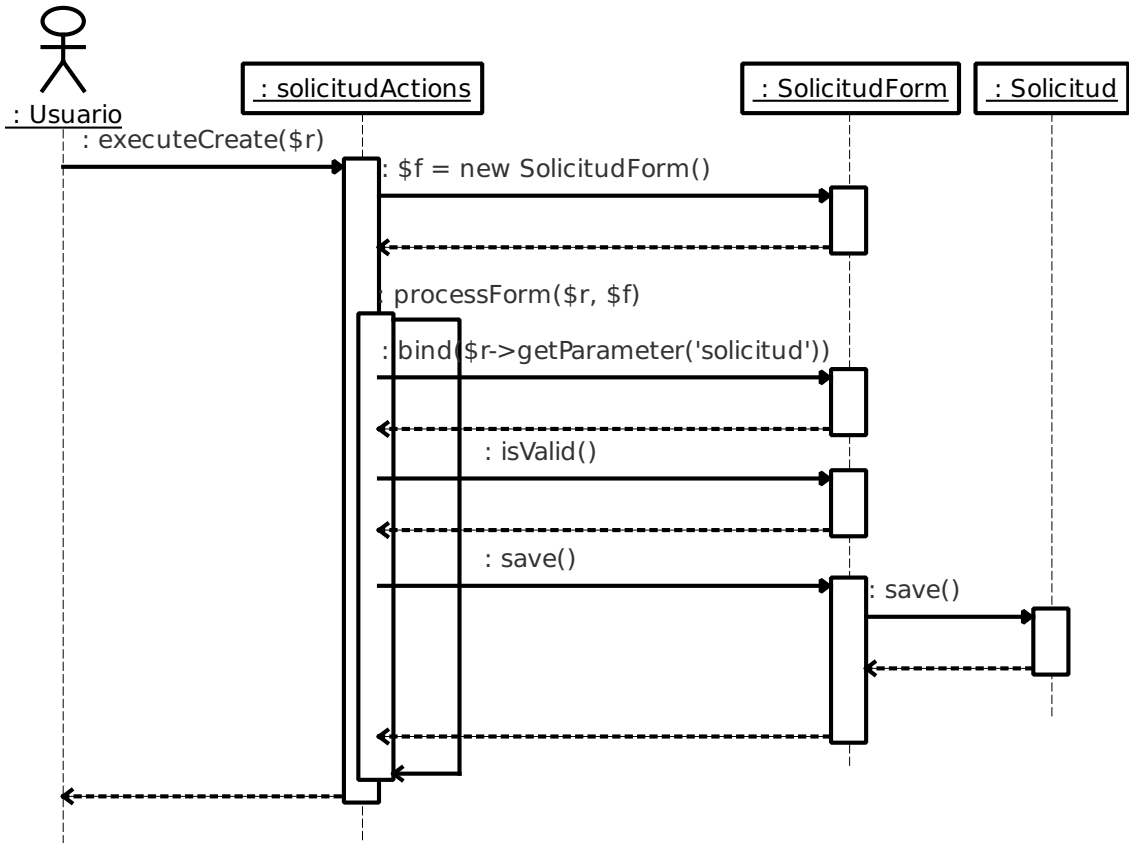

Figura 6.32: Ingresar Solicitud

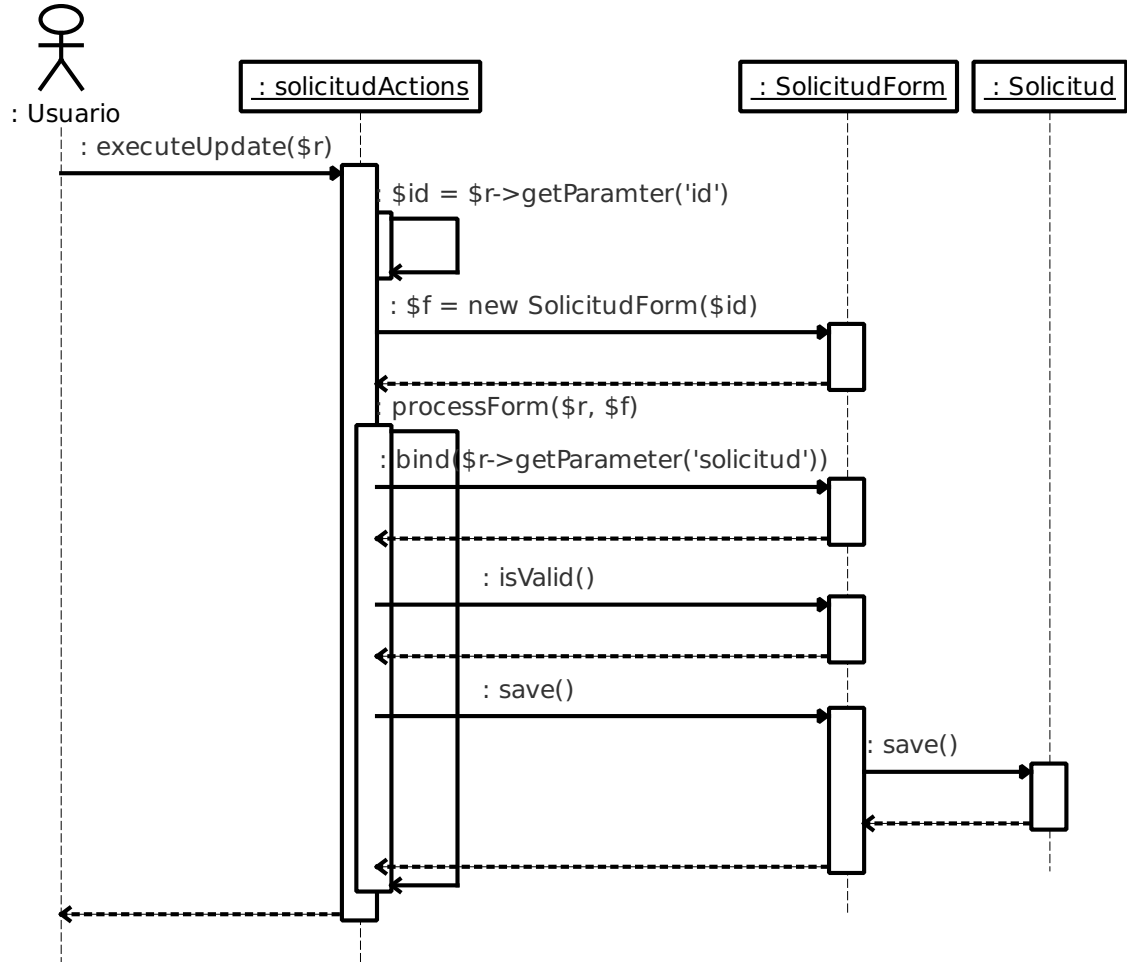

Figura 6.33: Modificar Solicitud

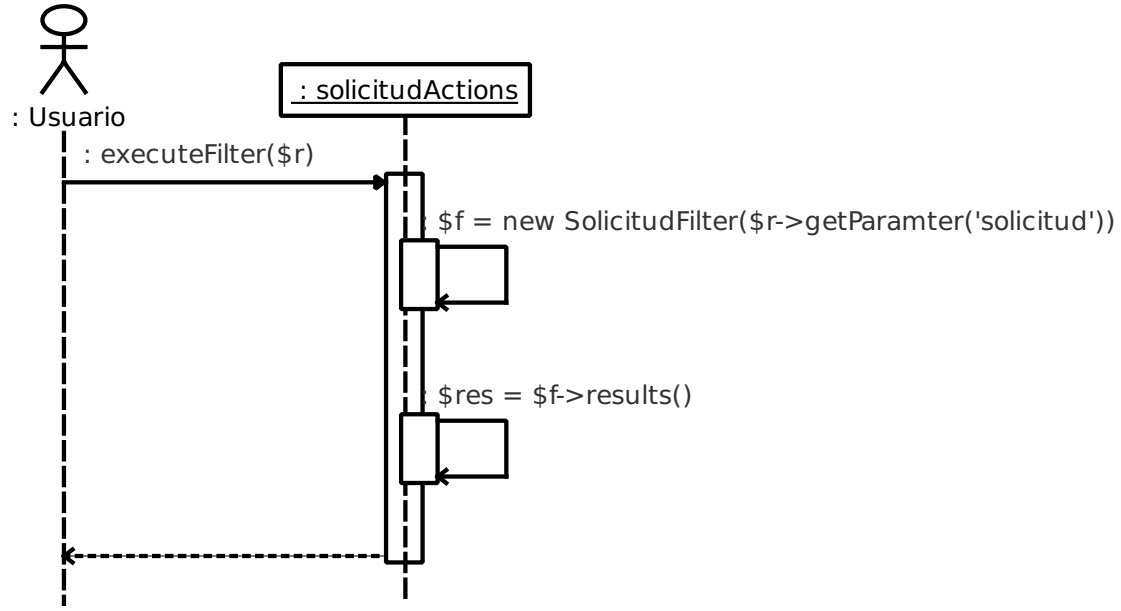

Figura 6.34: Buscar Solicitud

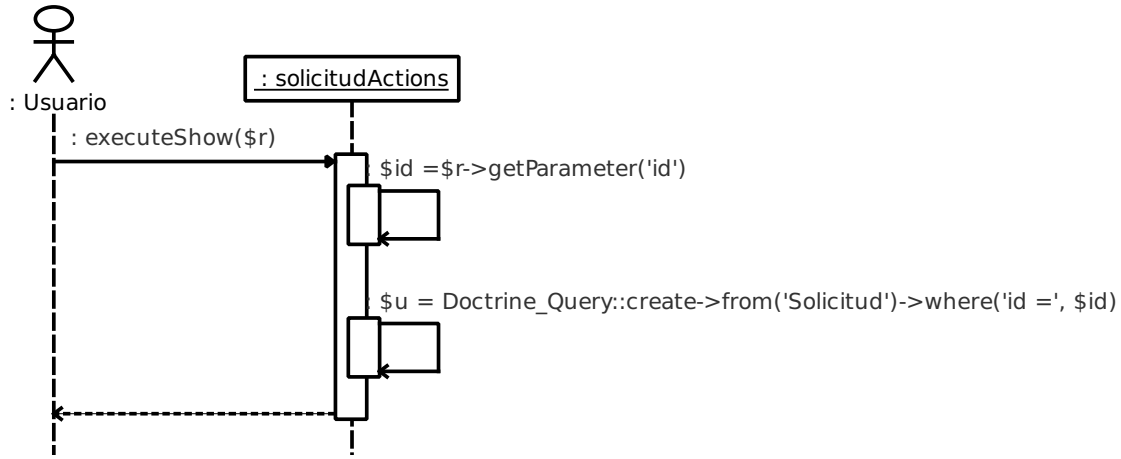

Figura 6.35: Ver Solicitud

6.6.4. Historial

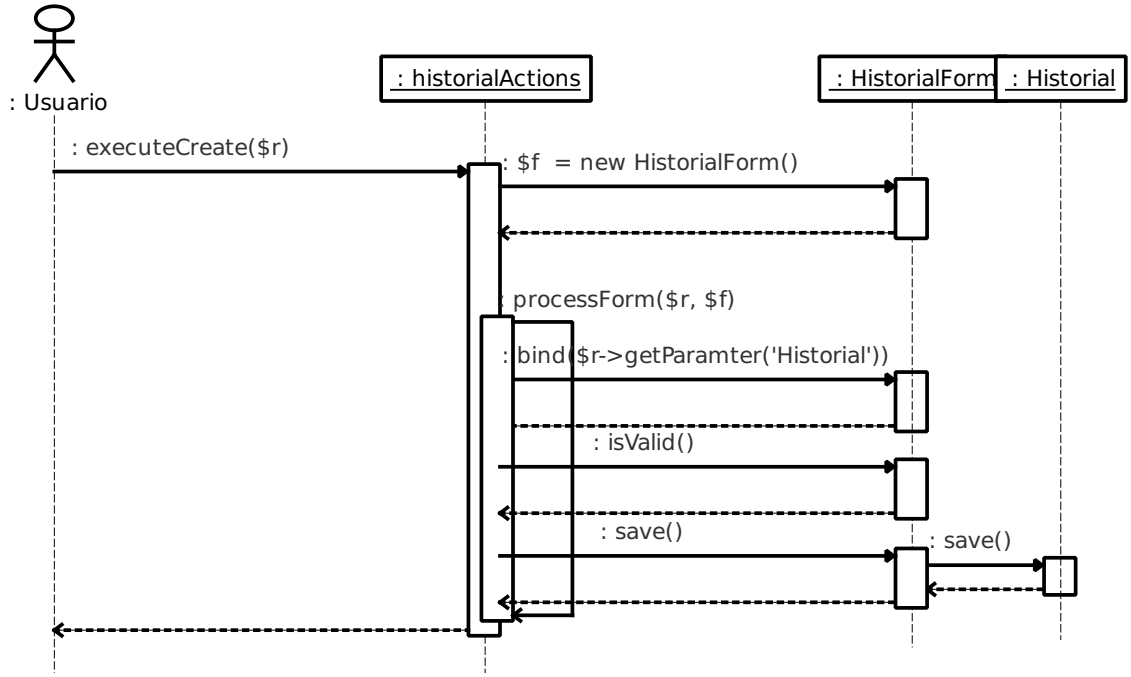

Figura 6.36: Ingresar Historial

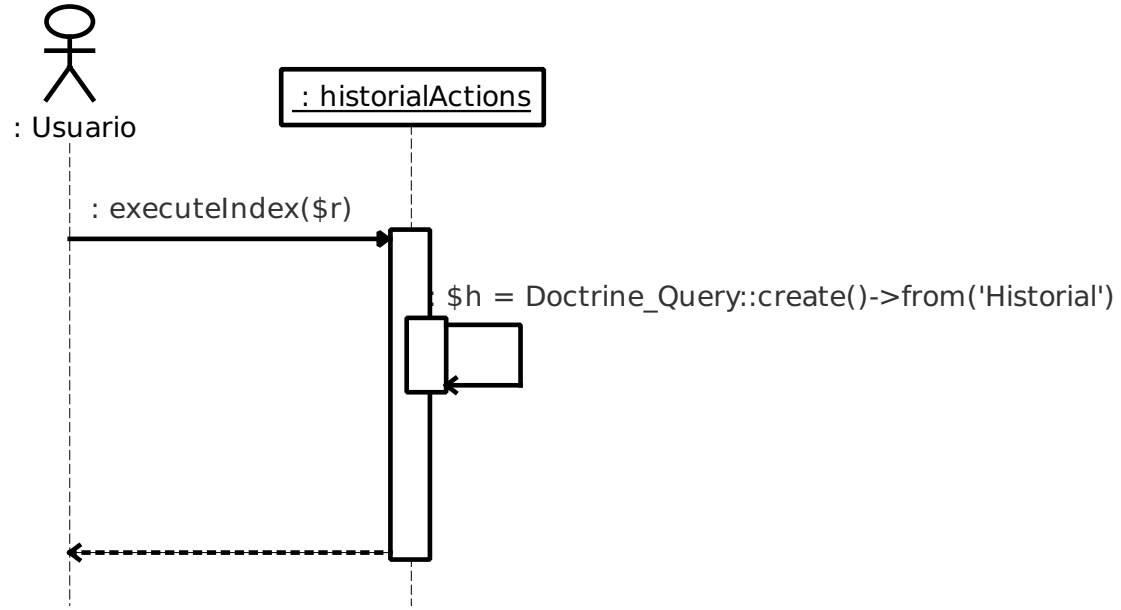

Figura 6.37: Visualizar Historiales

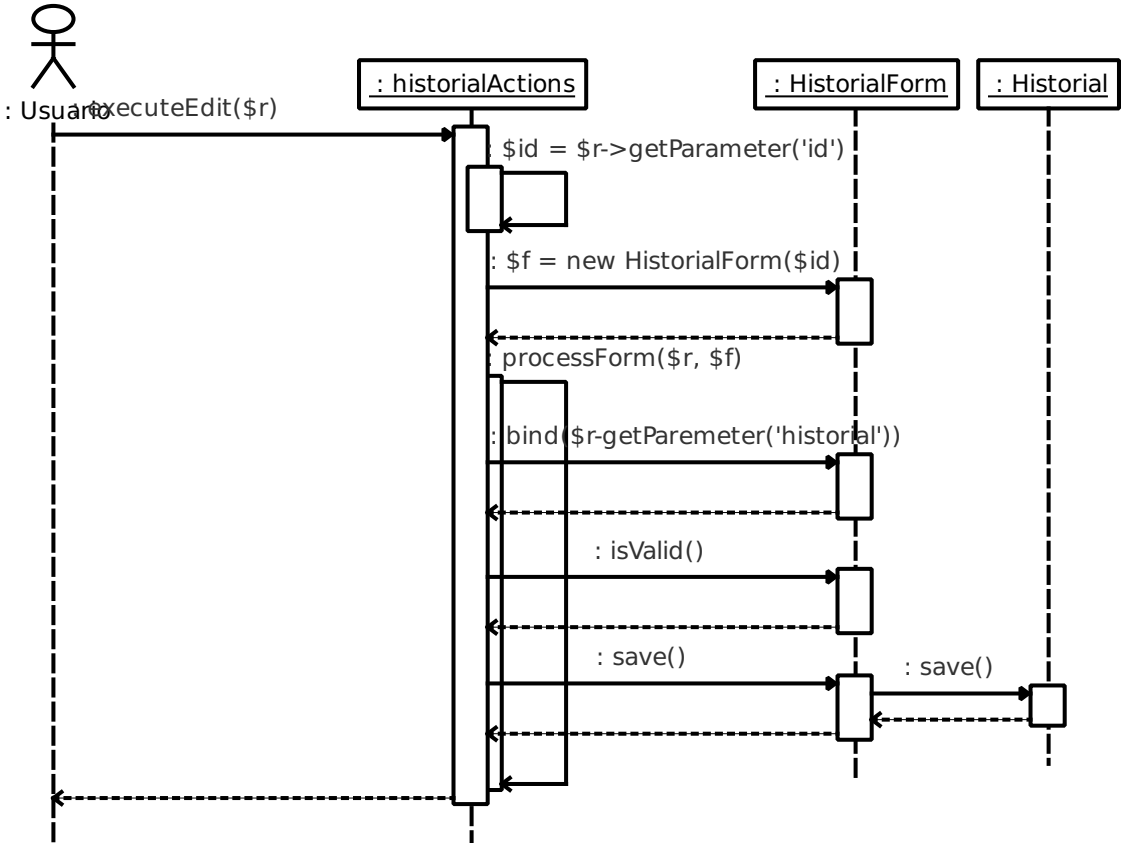

Figura 6.38: Modificar Historial

6.6.5. Cita

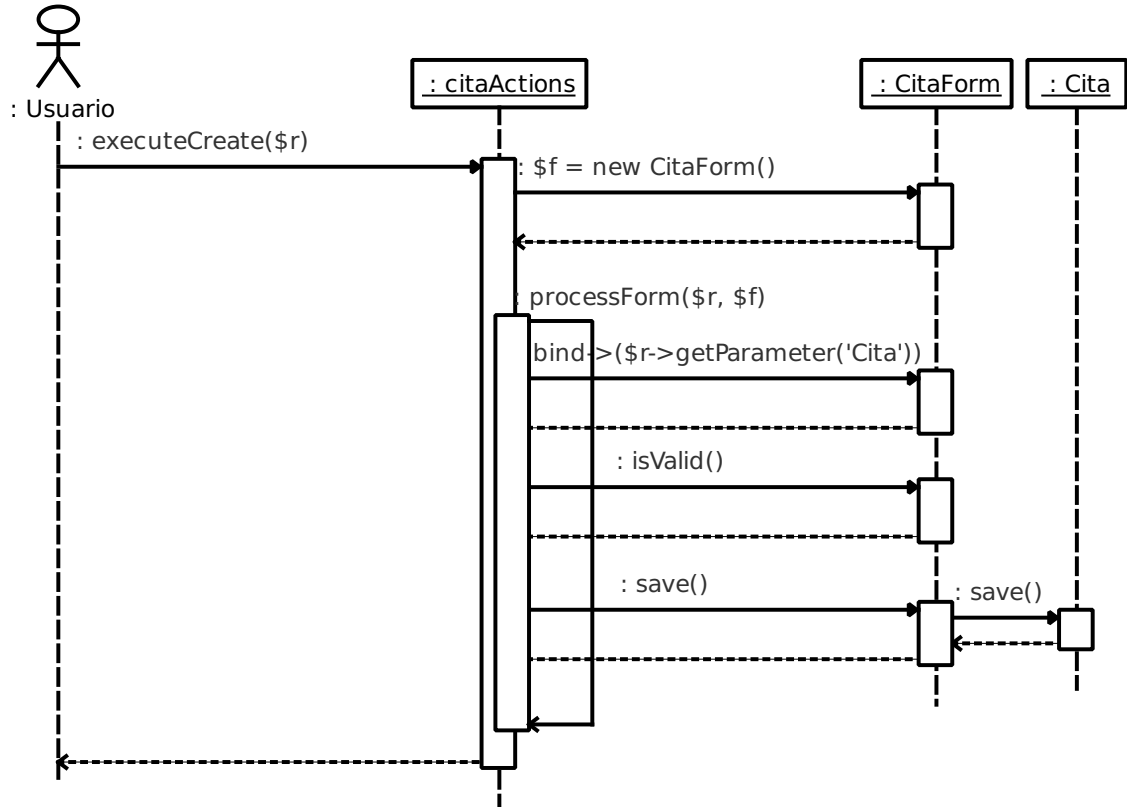

Figura 6.39: Ingresar Cita

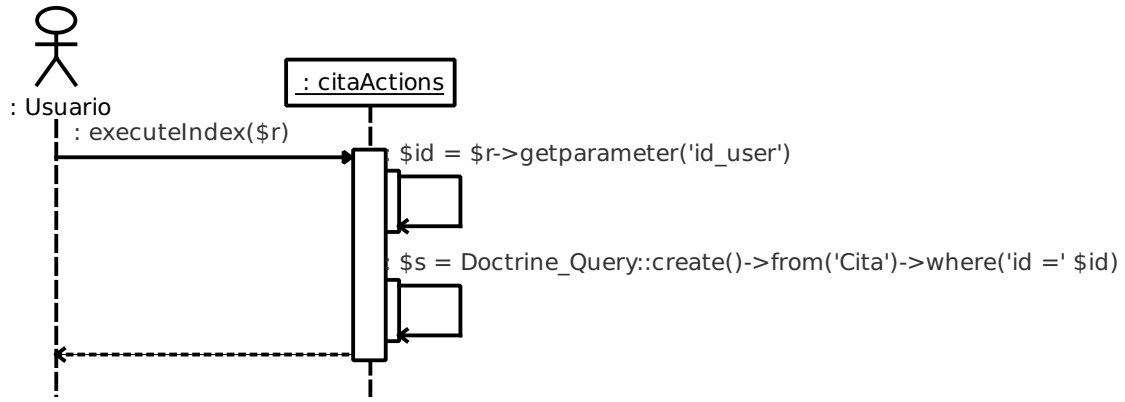

Figura 6.40: Desplegar Cita

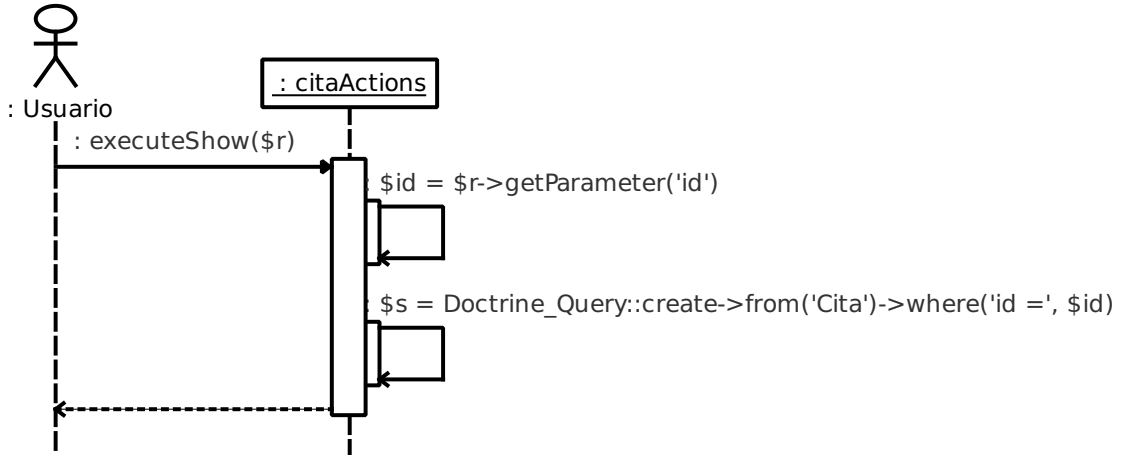

Figura 6.41: Ver Cita

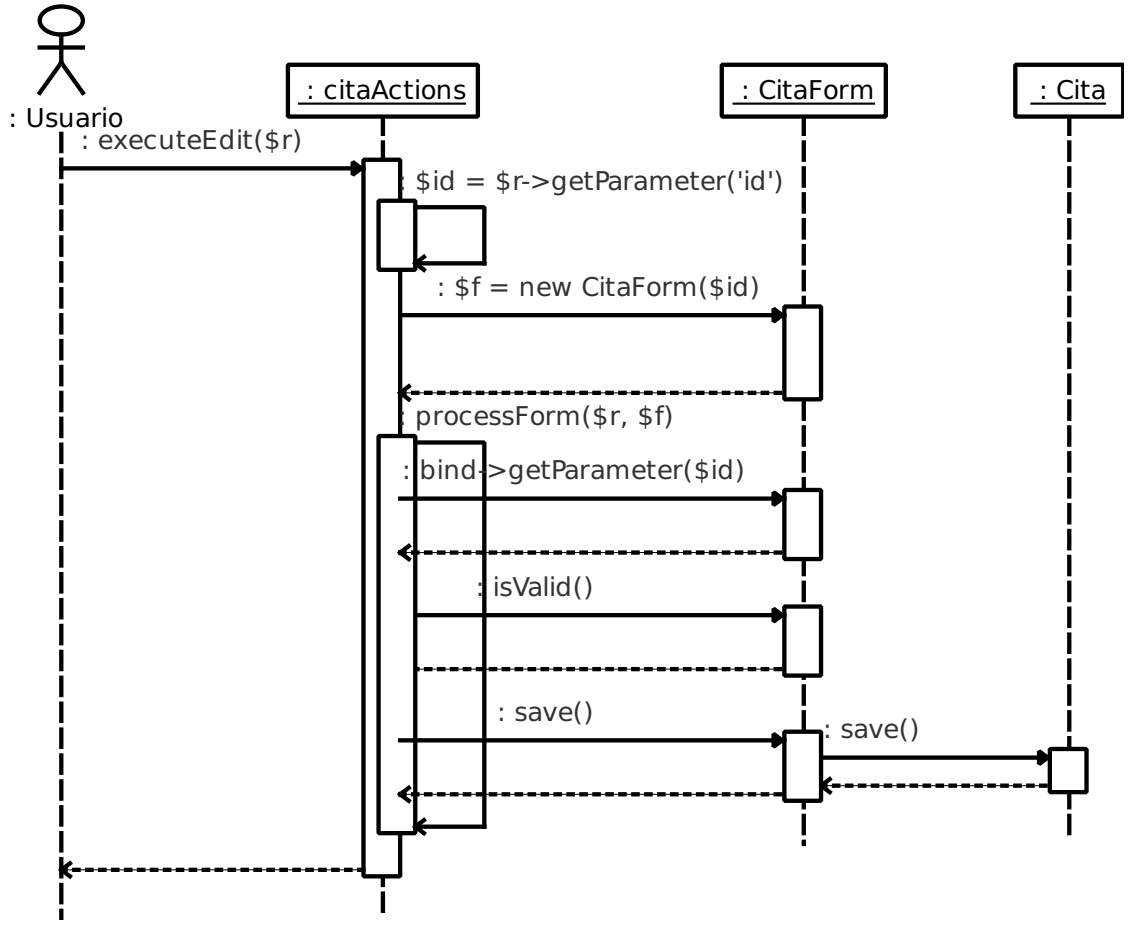

Figura 6.42: Modificar Cita

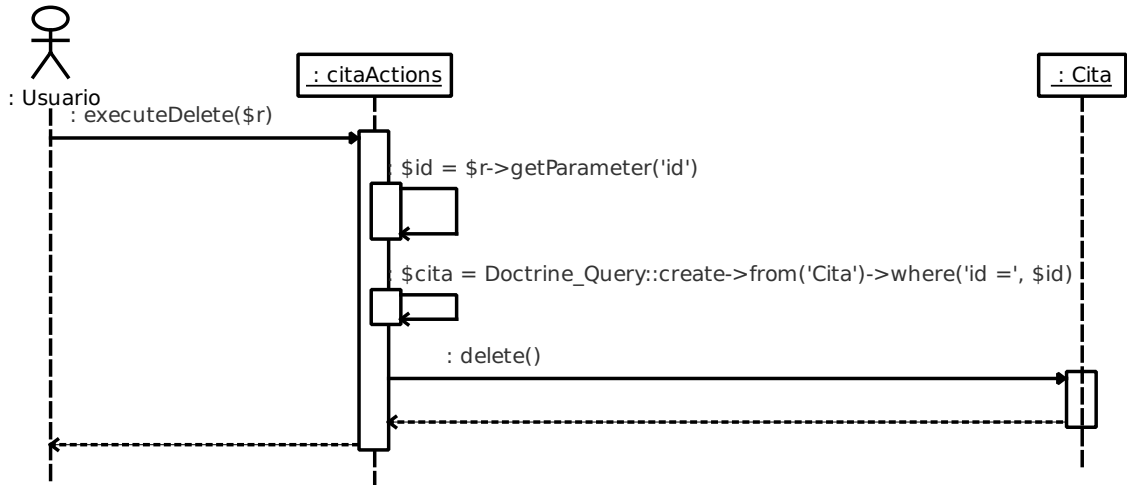

Figura 6.43: Borrar Cita

6.7. Pantallas del sistema

Figura 6.44: Login

Figura 6.45: Home

Figura 6.46: Ingresar Usuarios

Figura 6.47: Listar Usuarios

Figura 6.48: Modificar Perfil Usuarios

Figura 6.49: Modificar Acceso Usuarios

Figura 6.50: Dar de baja Usuarios

Figura 6.51: Ver Usuarios

Figura 6.52: Nuevo Solicitante

Figura 6.53: Listar Solicitantes

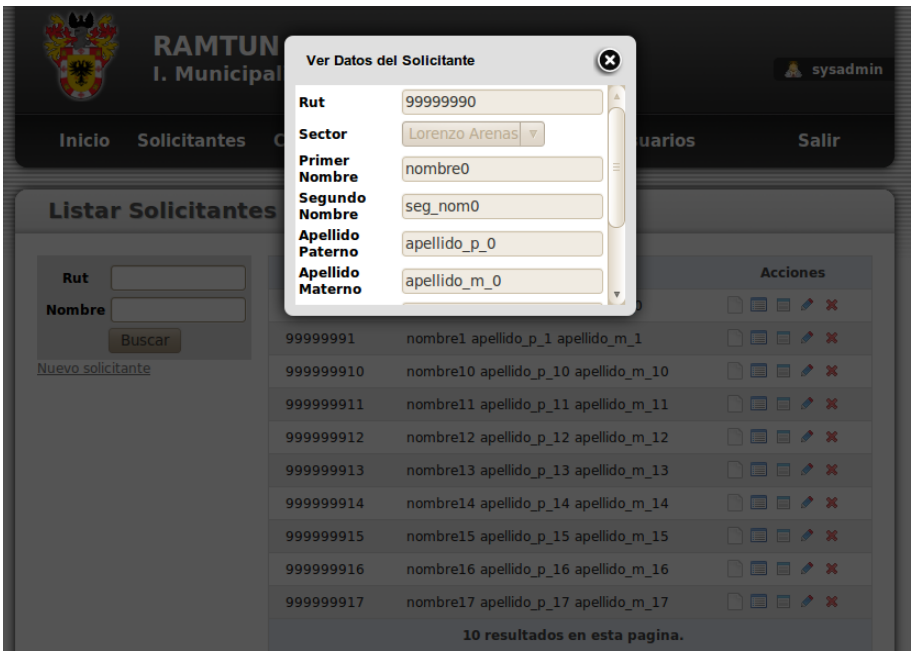

Figura 6.54: Ver Solicitante

Figura 6.55: Editar Solicitantes

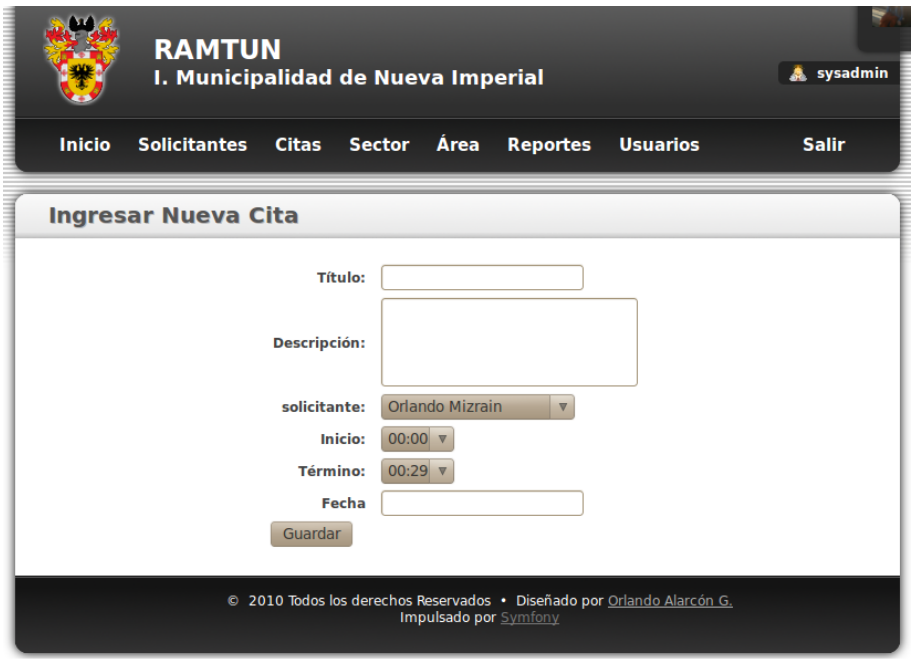

Figura 6.56: Nueva Cita

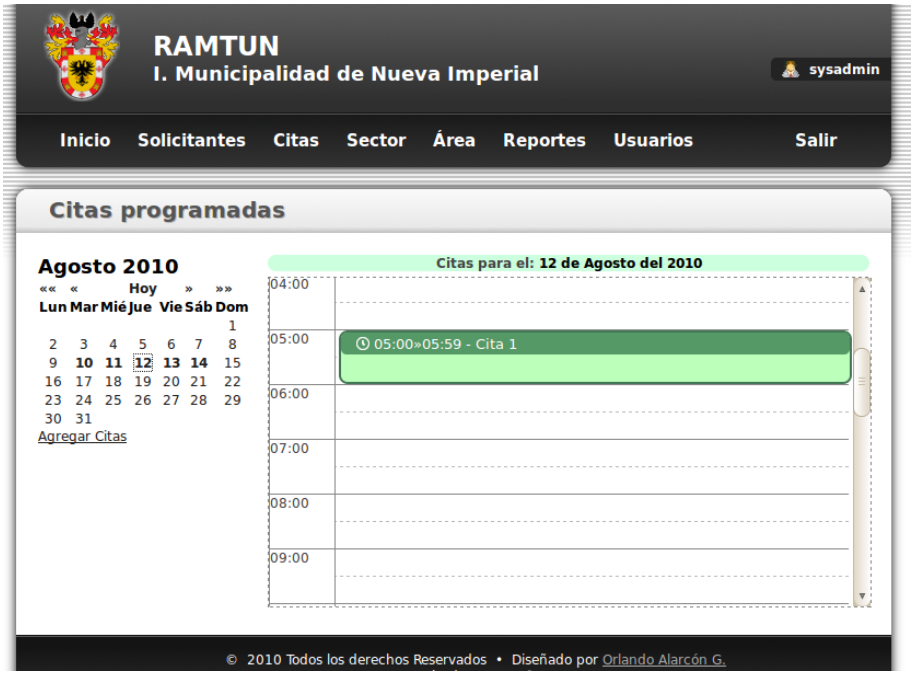

Figura 6.57: Listar Cita

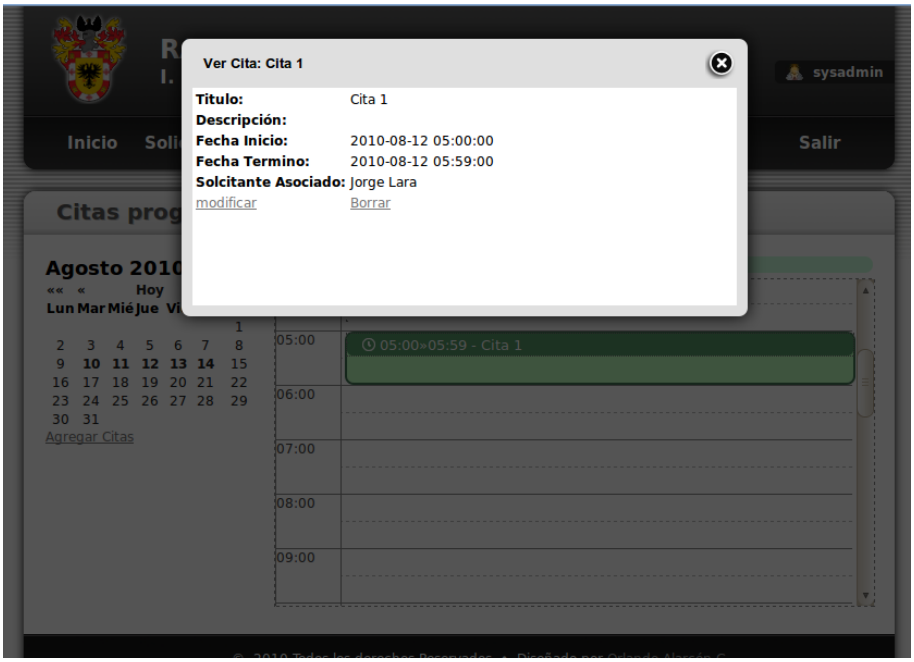

Figura 6.58: Ver Cita

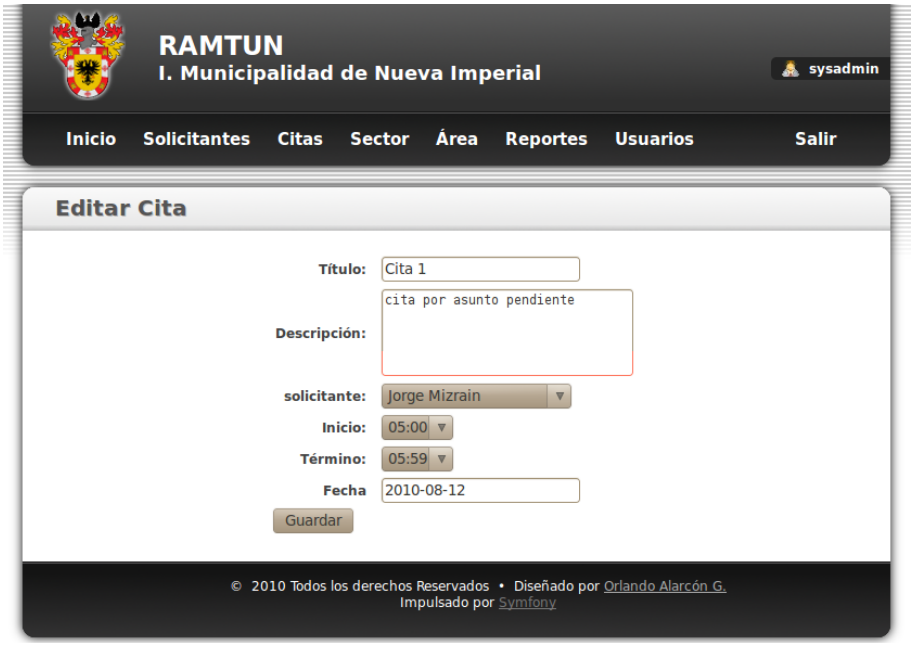

Figura 6.59: Editar Cita

Figura 6.60: Nueva Área

Figura 6.61: Listar Áreas

Figura 6.62: Editar Áreas

Capítulo 7

Pruebas, auditoría y respaldos

7.1. Introducción

El presente capítulo detalla la fase de ejecución de tests o pruebas que se realizaron al sistema antes de su puesta en marcha, para ello se describirá el plan de pruebas, las herramientas que se ocuparon y los tipos de prueba que se ejecutaron, también se incluye una breve explicación de cada tipo de prueba, esto con el objetivo de apoyar la comprensión del lector.

7.2. Plan de pruebas

Para ejecutar el plan de pruebas se ocupo la herramienta lime, integrada al framework symfony 1.4, esta herramienta esta especializada en hacer pruebas unitarias y combinándola con el framework, se pueden llegar a ejecutar pruebas funcionales simulando la interacción del navegador, esto es gracias a la clase sfBrowser de symfony, de esta forma se omite el paso por el servidor web. Los modelos de las pruebas se verán en la sección siguiente y si el lector desea puede encontrar la ficha del plan de pruebas en el anexo B.

Los tipos de pruebas ejecutadas fueron:

7.2.1. Pruebas unitarias

una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Luego, con las Pruebas de Integración, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión.

La idea es escribir casos de prueba para cada función no trivial o método en el módulo de forma que cada caso sea independiente del resto.

7.2.2. Pruebas funcionales

Una prueba funcional es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. La pruebas funcionales se hacen mediante el diseño de modelos de prueba que buscan evaluar cada una de las opciones con las que cuenta el paquete informático.

7.3. Pruebas unitarias

Todas las pruebas listadas a continuación fueron realizadas con el procedimiento tipo de *valor obtenido vs valor esperado* para ello se pasan valores al limite de la validación, es decir valores que sabemos fallarán(o pasarán con éxito), como acotación los valores inválidos son todos aquellos que no cumplen con los requisitos de la validación, es decir, si un campo es obligatorio y se omite, pasará a considerase un valor inválido, de la misma manera si un campo se envía con un formato inesperado (como el caso de las fechas y los mail) también se considerarán inválidos.

7.3.1. Validaciones

Validar RUT	
Entrada	**Salida**
16152818-8	true
11155356-k	false

Validar correo electrónico	
Entrada	**Salida**
correo@server.cl	true
correo@.server	false

Validar clave repetida	
Entrada	**Salida**
campos con valores iguales	true
campos con distintos valores	false

Validar ingreso clave	
Entrada	**Salida**
clave correcta	true
clave incorrecta	false

7.4. Pruebas funcionales

Ingresar Usuario	
Entrada	**Salida**
Datos válidos	true
Datos inválidos	false

Modificar Usuario	
Entrada	**Salida**
Datos válidos	true
Datos inválidos	false

Dar de Baja Usuario	
Entrada	**Salida**
selección de la opción	true (cambio de estado)

Autenticación de Usuario	
Entrada	**Salida**
Datos de acceso correctos	true
Datos de acceso incorrectos	false

Ingresar Solicitantes	
Entrada	**Salida**
clave correcta	true
clave incorrecta	false

Modificar Solicitantes	
Entrada	**Salida**
clave correcta	true
clave incorrecta	false

Ingresar Cita	
Entrada	**Salida**
Datos válidos	true
Datos inválidos	false

Modificar Cita	
Entrada	**Salida**
Datos válidos	true
Datos inválidos	false

Borrar Cita	
Entrada	**Salida**
Selección de la acción (borrar)	true (acción ejecutada)

7.5. Respaldo y auditoría

Como la cantidad de datos a manipular es elevada, es crucial contar con políticas de seguridad que permitan garantizar la integridad de los datos. algunas propuestas para ello son las siguientes:

7.5.1. Control de accesos físico del servidor

Para evitar cualquier pormenor con el hardware del servidor y para prevenir algún problema tipo denegación de servicio, por algún descuido, es altamente recomendable que el acceso físico a la estancia del servidor sea, en lo posible, restringido solamente a los encargados y a mantención, Con esto además se evita la posibilidad de acceder al sistema por "fuerza bruta"(acceso directo y no a través de la red) y de esta forma, mantener la privacidad de los datos sensibles en la base de datos.

7.5.2. Respaldos periódicos

Es recomendable que la base de datos sea respaldada en medios extraíbles o en sistemas tipo backups, por lo menos cada 15 días, esto con el fin de minimizar en lo posible la pérdida de datos en casos de catástrofes, corrupción de archivos o cualquier eventual problema que pudiera surgir con la integridad de la base de datos.

Los puntos anteriores son solo recomendaciones y deben tomarse como formas básicas de minimizar riesgos, puesto que estas decisiones se dejan a exclusiva responsabilidad del departamento de informática de la I. Municipalidad de Nueva Imperial.

Conclusión

El trabajo expuesto anteriormente deja en evidencia, tanto la importancia de los sistemas informáticos para las organizaciones, y cómo alivianan el trabajo de éstas, cómo lo vitales que son los aspectos previos a la implementación de un sistema, como el levantamiento de requerimientos y el diseño, para que un proyecto llegue a buen término.

Por otro lado, también se aprecia el buen resultado de ocupar herramientas de desarrollo modernas, como son los frameworks que ofrecen un conjunto de herramientas y librerías que incorporan muchas de las mejores prácticas y patrones de diseño que se recomiendan ocupar actualmente en los desarrollos, otorgando un ritmo de trabajo ágil, liberando al desarrollador de las tareas más rutinarias y tediosas, permitiendo enfocarse en los aspectos únicos y relevantes de cada proyecto. En el caso de este proyecto el framework symfony 1.4 cumplió con todas las expectativas esperadas.

Por último y a título personal, el autor del proyecto y de este informe pudo adquirir experiencia valiosa en el desarrollo de sistemas aplicables a la realidad, cumpliendo con los objetivos establecidos en el proyecto, sobre todo en la investigación, aprendizaje y aplicación de conceptos y tecnologías nuevas incorporadas en el desarrollo, por otra parte el sistema cumplió a cabalidad con lo que esperaba la I. Municipalidad de Nueva Imperial, además la relación entablada con la organización es una importante puerta de entrada al mundo laboral lo que eventualmente podría traducirse en proyectos a futuro, y como última acotación se apreció de primera fuente el verdadero trabajo que realiza un ingeniero informático.

Bibliografía

Enlaces

[1] I. Municipalidad de Nueva Imperial, "*home*" [en linea]. http://www.nuevaimperial.cl/index.php?id=36 [consulta: 10 Junio 2010].

[2] I. Municipalidad de Nueva Imperial, "*Caracterización Comunal* " [en linea]. http://www.nuevaimperial.cl/index.php?id=9 [consulta: 10 Junio 2010].

[3] Sistema Nacional de Información Municipal, "*ficha comunal*" [en linea]. http://www.sinim.cl/ficha_comunal/fcomunal.php?id_muni=09111&ano=2008&periodo=A [consulta: 10 Junio 2010].

[4] I. Municipalidad de Nueva Imperial, "*Reglamento Interno Municipal*" [en linea]. http://www.nuevaimperial.cl/uploads/media/Reglametno_interno_municipal.pdf [consulta: 10 Junio 2010].

[5] Portal ser indígena, "*Diccionario español - mapudungun*" [en linea]. http://diccionarios.serindigena.org/diccionarios/espa_mapu.htm [consulta: 17 Junio 2010].

[6] IEEE std 830-1998, "IEEE Recommended Practice for Software Requeriements Specifications"

[7] Documentación Oficial de PHP [En Linea]. http://docs.php.net/manual/es/index.php [consulta: 01 Julio 2010].

[8] Página oficial de GIT [En Linea]. http://git-scm.com/ [consulta: 01 Julio 2010].

[9] Definición de apache - ¿Que es Apache? [En Linea]. http://www.alegsa.com.ar/Dic/apache.php [consulta: 01 Julio 2010].

Libros

[10] PONS, Nicolas. *Linux: principios básicos del uso del sistema*. 2ª Ed. Barcelona, Ediciones ENI, 2009. 320p.

[11] CARDONA, Xavier. *Sistemas Operativos Monopuesto*. 109p.

[12] HEURTEL, Olivier. *PHP y MySQL Domine el desarrollo de un sitio Web dinamico e interactivo*. Ediciones ENI, 2009. 624p.

[13] CABEZAS, Luis M. *Manual Imprescindible de PHP5*. Ediciones Anaya, 2004.

[14] POTENCIER Fabien, FRANÇOIS Zaninotto. *Symfony 1.2, la guía definitiva*. Editorial Apress.

[15] WAGE Jonathan H, VESTERINEN Konsta. *Doctrine Orm for PHP*. Sensio Sa, 2009. 522p.

[16] NEWTON Aaron. *Mootools Essentials: The Official MooTools Reference for JavaScript and Ajax Development*. Apress, 2008. 276p.

[17] CHACON Scott. *Pro Git: Everything you need to know about the Git distributed source control tool*. Apress, 2009. 300p.

[18] SOMMERVILLE, Ian. *Ingeniería del software*. 7ª Ed., Pearson Educación, 2005. 687p.

Anexo A

Diccionario de datos

Tabla sfGuardUser: se encarga de almacenar la información de autentificación y acceso de usuarios al sistema. esta tabla pertenece al plugin sfDoctrineGuard desarrollado para el framework Symfony 1.4 que es el ocupado en el sistema Ramtun.

sfGuardUser				
Atributo	**Tipo de Dato**	**Validación**	**Restricción**	**Descripción**
id	Integer	no núlo	Es clave primaria, solo acepta números enteros	Almacena el id único de cada usuario
username	Varchar(128)	no núlo	Posee un máximo de 128 caracteres	Almacena el nombre usuario con el cual accede al sistema
algorithm	Varchar(128)	no núlo	Posee un máximo de 128 caracteres	Almacena el tipo de algorithmo con el que se encripta la contraseña del usuario
salt	Varchar(128)	no núlo	Posee un máximo de 128 caracteres	Almacena valor de control y seguridad para la encriptación de la clave
password	Varchar(128)	no núlo	Posee un máximo de 128 caracteres	Almacena la clave enriptada del usuario
created_at	Datetime	-	Acepta valores en formato fecha	Almacena la fecha de la creación del usuario
last_login	Datetime	-	Acepta valores en formato fecha	Almacena la fecha de ultimo acceso del usuario
is_active	Boolean	-	Acepta valores booleanos (0 - 1)	Almacena si el usuario esta activo
is_super_admin	Boolean	-	Acepta valores booleanos (0 - 1)	Almacena si el usuario es administrador o super usuario

Tabla Departamento: se encarga de almacenar la información asociada a los departemento existentes en la I. Municipalidad de Nueva Imperial.

Departamento				
Atributo	**Tipo de Dato**	**Validación**	**Restricción**	**Descripción**
id_departamento	Integer	no nulo	Es clave primaria, solo acepta números enteros	Almacena el identificador unico del departamento
nombre_departamento	Varchar(45)	-	Posee un máximo de 45 caracteres	Almacena el nombre del departamento
descripcion_departamento	Varchar(128)	-	Posee un máximo de 128 caracteres	Almacena una breve descripción del departamento

Tabla Sector: se encarga de almacenar la información asociada a los sectores poblacionales de la comuna de Nueva Imperial.

Sector				
Atributo	**Tipo de Dato**	**Validación**	**Restricción**	**Descripción**
id_sector	Integer	no nulo	Es clave primaria, solo acepta números enteros	Almacena el identificador único del sector
nombre_sector	Varchar(50)	-	Posee un máximo de 50 caracteres	Almacena el nombre del sector
latitud_sector	Varchar(45)	-	Posee un máximo de 45 caracteres	Almacena la latitud de asociada a la ubicacion geografica referencial del sector
longitud_sector	Varchar(45)	-	Posee un máximo de 45 caracteres	Almacena la longitud de asociada a la ubicacion geografica referencial del sector

Tabla Área: se encarga de almacenar la información asociada a los tipos de solicitudes o áreas, las cuales estan relacionadas con la solicitud de ayudas.

Área				
Atributo	**Tipo de Dato**	**Validación**	**Restricción**	**Descripción**
id_area	Integer	no nulo	Es clave primaria, solo acepta números enteros	Almacena el identificador único del área
nombre_area	Varchar(45)	-	Posee un máximo de 45 caracteres	Almacena el nombre del área

Tabla Cita: se encarga de almacenar la información asociada a las citas programadas por cada usuario.

Cita				
Atributo	**Tipo de Dato**	**Validación**	**Restricción**	**Descripción**
id_cita	Integer	no núlo	Es clave primaria, solo acepta números enteros	Almacena el id único de cada cita
titulo_cita	Varchar(45)	-	Posee un máximo de 45 caracteres	Almacena el titulo de la cita
descripcion_cita	Varchar(45)	-	Posee un máximo de 45 caracteres	Almacena la descripción de la cita
fecha_ini_cita	Datetime	-	Acepta solo valores en formato fecha	Almacena la fecha/hora de inicio de la cita
fecha_ter_cita	Datetime	-	Acepta solo valores en formato fecha	Almacena la fecha/hora de termino de la cita
Usuario_id_usuario	Integer	-	Es clave foranea, solo acepta números enteros	Alamacena el identificador del usuario propietario de la cita
Solicitante_rut_solicitante	Varchar(10)	-	Es clave foranea, solo acepta Ruts validos	Alamacena el RUT del solicitante asociado de la cita

Tabla Usuario: se encarga de almacenar la información personal asociada a los usuarios registrados en el sistema.

Usuario				
Atributo	**Tipo de Dato**	**Validación**	**Restricción**	**Descripción**
id_usuario	Integer	no núlo	Es clave primaria	Almacena el id único de cada usuario
primer_nombre_usuario	Varchar(45)	-	Posee un máximo de 45 caracteres	Almacena el primer nombre del usuario
segundo_nombre_usuario	Varchar(45)	-	Posee un máximo de 45 caracteres	Almacena el segundo nombre del usuario
apellido_paterno_usuario	Varchar(45)	-	Posee un máximo de 45 caracteres	Almacena el apellido paterno del usuario
apellido_materno_usuario	Varchar(45)	-	Posee un máximo de 45 caracteres	Almacena el apellido materno del usuario
fecha_nac_usuario	Date	-	Acepta solo valores en formato fecha	Almacena la fecha de nacimiento del usuario
correo_e_usuario	Varchar(64)	-	Posee un máximo de 64 caracteres	Almacena la dirección de correo electrónico del usuario
foto_usuario	Varchar(128)	-	Posee un máximo de 128 caracteres	Almacena la url de la foto del usuario
Departamento_id_departamento	Integer	-	Es clave foranea, solo acepta números enteros	Almacena el identificador del departamento al cual pertenece el usuario
sf_guard_user_id	Integer	-	Es clave foranea, solo acepta números enteros	Almacena el identificador del perfil correspondiente en las tablas del plugin sfGuardPlugin

Tabla Solicitante: se encarga de almacenar la información personal asociada a los solicitantes registrados en el sistema.

Solicitante				
Atributo	**Tipo de Dato**	**Validación**	**Restricción**	**Descripción**
rut_solicitante	Varchar(10)	no núlo	Es clave primaria	Almacena el rut único de cada solicitante
primer_nombre_solicitante	Varchar(45)	-	Posee un máximo de 45 caracteres	Almacena el primer nombre del solicitante
segundo_nombre_solicitante	Varchar(45)	-	Posee un máximo de 45 caracteres	Almacena el segundo nombre del solicitante
apellido_paterno_solicitante	Varchar(60)	-	Posee un máximo de 60 caracteres	Almacena el apellido paterno del solicitante
apellido_materno_solicitante	Varchar(60)	-	Posee un máximo de 60 caracteres	Almacena el apellido materno del solicitante
sexo_solicitante	Varchar(15)	-	Posee un máximo de 15 caracteres	Almacena el sexo del solicitante
direccion_solicitante	Varchar(60)	-	Posee un máximo de 60 caracteres	Almacena la dirección del solicitante
fecha_nac_solicitante	Date	-	Acepta solo valores en formato fecha	Almacena la fecha de nacimiento del solicitante
telefono_solicitante	Varchar(10)	-	Posee un máximo de 10 caracteres	Almacena el número telefónico del solicitante
Sector_id_sector	Integer	-	Es clave foranea, solo acepta números enteros	Almacena el identificador del sector poblacional al cual pertenece el usuario

Tabla Solicitud: se encarga de almacenar la información asociada a las solicitudes de ayudas registradas en el sistema.

Solicitud				
Atributo	**Tipo de Dato**	**Validación**	**Restricción**	**Descripción**
id_solicitud	Integer	-	Es clave primaria, solo acepta números enteros	Almacena el identificador único de la solicitud
detalle_solicitud	Varchar(250)	-	Posee un máximo de 250 caracteres	Almacena el detalle de la solicitud
fecha_ini_solicitud	Date	-	Acepta solo valores en formato fecha	Almacena la fecha de creación de la solicitud
fecha_ter_solicitud	Date	-	Acepta solo valores en formato fecha	Almacena la fecha de termino de la solicitud
estado_solicitud	Varchar(45)	-	Posee un máximo de 45 caracteres	Almacena el estado de la solicitud
Area_id_area	Integer	-	Acepta solo valores enteros	Almacena el id del area asociada
Solicitante_rut_solicitante	Varchar(10)	-	Es clave foranea, posee un máximo de 10 caracteres	Almacena el RUT del solicitante asociado
Historia_usuario_id	Integer	-	Es clave foranea, acepta solo valores enteros	Almacena el id de la hisoricidad del usuario

Tabla Historial: se encarga de almacenar la información de historial o seguimiento asociado a las solicitudes de ayudas registradas en el sistema.

Historial				
Atributo	**Tipo de Dato**	**Validación**	**Restricción**	**Descripción**
id_historial	Integer	-	Es clave primaria, acepta solo valores enteros	Almacena el identificador único del historial
fecha_historial	Date	-	Acepta solo valores en formato fecha	Almacena el detalle de la solicitud
observacion_historial	Varchar(250)	-	Posee un máximo de 250 caracteres	Almacena las nuevas observaciones correspondientes a la solicitud asociada
Solicitud_id_solicitud	Integer	-	Es clave foranea, acepta solo valores enteros	Almacena el id de la solicitud asociada
Historia_usuario_id	Integer	-	Es clave foranea, acepta solo valores enteros	Almacena el id de la hisoricidad del usuario

Tabla Historia Usuario: se encarga de almacenar la información necesaria para permitir historicidad de usuarios registrados en el sistema.

Historia Usuario				
Atributo	**Tipo de Dato**	**Validación**	**Restricción**	**Descripción**
id	Integer	-	Es clave primaria, acepta solo valores enteros	Almacena un identificador unico para permitir historicidad
Usuario_id_usuario	Integer	-	Es clave primaria, acepta solo valores enteros	Almacena el identificador único del usuario
fecha_inicio	Datetime	-	Acepta solo valores en formato fecha hora	Almacena la fecha de registro del usuario
fecha_termino	Datetime	-	Acepta solo valores en formato fecha hora	Almacena la fecha de dada de baja del usuario

Anexo B

Documentos anexos

Nombre Caso de Uso: Ingresar Departamentos	
Id:	CU17_ID.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de ingresar un nuevo departamento.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**.

Flujo normal	
1- El usuario selecciona la categoría departamento.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los departamentos. junto con un breve listado de departamentos previamente ingresados al sistema.
3- El usuario selecciona la opción "Ingresar departamento"	
	4- El sistema despliega en pantalla un formulario con los datos necesario para ingresar un nuevo departamento.
5- El usuario procede a llenar los campos que estime conveniente luego selecciona guardar	
	6- El sistema procesa los datos y procede a desplegar en pantalla el resultado de la operación.

Flujo Alternativo:	**6a**- En caso de ser satisfactorio, el nuevo departamento es ingresado, luego el sistema despliega en pantalla que la operación se realizó con éxito. **6b**- En caso de no ser satisfactorio. el sistema despliega en pantalla que la operación no se llevo a cabo
Post-Condición:	

Nombre Caso de Uso: Listar Departamentos	
Id:	CU18_LD.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de desplegar un listado de departamentos.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**.
Flujo normal	
1- El usuario selecciona la categoría departamento.	**2**- El sistema despliega una pantalla con las opciones posibles a realizar con los departamentos. junto con un breve listado de departamentos previamente ingresados al sistema.
Flujo Alternativo:	
Post-Condición:	

Nombre Caso de Uso: Visualizar Departamentos	
Id:	CU19_VD.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de visualizar un departamento.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU18_LD**.

Flujo normal	
1- El usuario selecciona la categoría departamento.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los departamentos. junto con un breve listado de departamentos previamente ingresados al sistema.
3- El usuario se posiciona sobre el departamento y selecciona la opción "Ver datos de departamento"	
	4- El sistema despliega en pantalla los datos del departamento.

Flujo Alternativo:	
Post-Condición:	

Nombre Caso de Uso: Modificar Departamentos	
Id:	CU20_MD.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de modificar los datos de un departamento.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU18_LD**.

Flujo normal	
1- El usuario selecciona la categoría departamento.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los departamentos. junto con un breve listado de departamentos previamente ingresados al sistema.
3- El usuario se posiciona sobre el departamento y selecciona la opción "Modificar departamento"	
	4- El sistema despliega en pantalla un formulario con los datos cargados del departamento seleccionado.
5- El usuario procede a cambiar los campos que estime conveniente luego selecciona "guardar"	
	6- El sistema procesa los datos y procede a desplegar en pantalla el resultado de la operación.
Flujo Alternativo:	**6a**- En caso de ser satisfactorio, los datos del departamento son actualizados, luego el sistema despliega en pantalla que la operación se realizó con éxito. **6b**- En caso de no ser satisfactorio. el sistema despliega en pantalla que la operación no se llevo a cabo
Post-Condición:	

Nombre Caso de Uso: Ingresar Sectores	
Id:	CU21_IS.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de ingresar un nuevo sector.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**.

Flujo normal	
1- El usuario selecciona la categoría sectores.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los sectores. junto con un breve listado de sectores previamente ingresados al sistema.
3- El usuario selecciona la opción "Ingresar sector"	
	4- El sistema despliega en pantalla un formulario con los datos necesario para ingresar un nuevo sector.
5- El usuario procede a llenar los campos que estime conveniente luego selecciona "guardar"	
	6- El sistema procesa los datos y procede a desplegar en pantalla el resultado de la operación.
Flujo Alternativo:	**6a**- En caso de ser satisfactorio, el nuevo sector es ingresado, luego el sistema despliega en pantalla que la operación se realizó con éxito. **6b**- En caso de no ser satisfactorio. el sistema despliega en pantalla que la operación no se llevo a cabo
Post-Condición:	

Nombre Caso de Uso: Listar Sectores	
Id:	CU22_LS.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de desplegar un listado de sectores.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**.

Flujo normal	
1- El usuario selecciona la categoría sectores.	**2**- El sistema despliega una pantalla con las opciones posibles a realizar con los sectores. junto con un breve listado de sectores previamente ingresados al sistema.
Flujo Alternativo:	
Post-Condición:	

Nombre Caso de Uso: Visualizar Sectores	
Id:	CU23_VS.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de visualizar un sector.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU22_LS**.

Flujo normal	
1- El usuario selecciona la categoría sectores.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los sectores. junto con un breve listado de sectores previamente ingresados al sistema.
3- El usuario se posiciona sobre el sector y selecciona la opción "Ver datos de sector"	
	4- El sistema despliega en pantalla los datos del sector.
Flujo Alternativo:	
Post-Condición:	

Nombre Caso de Uso: Modificar Sectores	
Id:	CU24_MS.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de modificar los datos de un sector.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU22_LS**.

Flujo normal	
1- El usuario selecciona la categoría sectores.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con los sectores. Junto con un breve listado de sectores previamente ingresados al sistema.
3- El usuario se posiciona sobre el sector y selecciona la opción "Modificar sector"	
	4- El sistema despliega en pantalla un formulario con los datos cargados del sector seleccionado.
5- El usuario procede a cambiar los campos que estime conveniente luego selecciona "guardar"	
	6- El sistema procesa los datos y procede a desplegar en pantalla el resultado de la operación.

Flujo Alternativo:	**6a**- En caso de ser satisfactorio, los datos del sector son actualizados, luego el sistema despliega en pantalla que la operación se realizó con éxito. **6b**- En caso de no ser satisfactorio. el sistema despliega en pantalla que la operación no se llevo a cabo
Post-Condición:	

Nombre Caso de Uso: Ingresar Áreas	
Id:	CU25_IA.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de ingresar una nueva área.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**.

Flujo normal	
1- El usuario selecciona la categoría áreas.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con las áreas. junto con un breve listado de áreas previamente ingresados al sistema.
3- El usuario selecciona la opción "Ingresar área"	
	4- El sistema despliega en pantalla un formulario con los datos necesario para ingresar una nueva área.
5- El usuario procede a llenar los campos que estime conveniente luego selecciona "guardar"	
	6- El sistema procesa los datos y procede a desplegar en pantalla el resultado de la operación.

Flujo Alternativo:	**6a**- En caso de ser satisfactorio, la nueva área es ingresada, luego el sistema despliega en pantalla que la operación se realizó con éxito. **6b**- En caso de no ser satisfactorio. el sistema despliega en pantalla que la operación no se llevo a cabo
Post-Condición:	

Nombre Caso de Uso: Listar Áreas	
Id:	CU26_LA.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de desplegar un listado de áreas.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**.
Flujo normal	
1- El usuario selecciona la categoría áreas.	**2**- El sistema despliega una pantalla con las opciones posibles a realizar con las áreas. Junto con un breve listado de áreas previamente ingresadas al sistema.
Flujo Alternativo:	
Post-Condición:	

Nombre Caso de Uso: Visualizar Áreas	
Id:	CU27_VA.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de visualizar los datos de un área.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU26_LA**.

Flujo normal	
1- El usuario selecciona la categoría áreas.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con las áreas. junto con un breve listado de áreas previamente ingresados al sistema.
3- El usuario se posiciona sobre el área y selecciona la opción "Ver datos de área"	
	4- El sistema despliega en pantalla los datos del área.

Flujo Alternativo:	
Post-Condición:	

Nombre Caso de Uso: Modificar Áreas	
Id:	CU28_MA.
Actores:	Administrador, Alcalde, Encargado.
Resumen:	Este caso de uso describe el proceso de modificar los datos de un área.
Pre-Condición:	Para realizar esta función, el usuario debe estar previamente logeado al sistema **CU00A_IS**, **CU26_LA**.

Flujo normal	
1- El usuario selecciona la categoría áreas.	
	2- El sistema despliega una pantalla con las opciones posibles a realizar con las áreas. Junto con un breve listado de áreas previamente ingresados al sistema.
3- El usuario se posiciona sobre el sector y selecciona la opción "Modificar sector"	
	4- El sistema despliega en pantalla un formulario con los datos cargados del área seleccionada.
5- El usuario procede a cambiar los campos que estime conveniente luego selecciona "guardar"	
	6- El sistema procesa los datos y procede a desplegar en pantalla el resultado de la operación.

Flujo Alternativo:	**6a**- En caso de ser satisfactorio, los datos del área son actualizados, luego el sistema despliega en pantalla que la operación se realizó con éxito. **6b**- En caso de no ser satisfactorio. el sistema despliega en pantalla que la operación no se llevo a cabo
Post-Condición:	

Plan de Pruebas	
Código:	P01_ramtun.
Alcance:	Pruebas Unitarias y funcionales a los modulos Usuario, Solicitud, Solicitante.
Items a probar:	Acciones y funciones privadas de los controladores de los modulos Usuario, Solicitud, Solicitante.
Estrategia:	Pruebas de caja negra y caja Blanca.
Manejo de riesgos:	Criterio de correccion oportuna, basado en los resultados esperados permitiendo control centralizado del manejo de pruebas, otorgando de esta forma una base solida en conjunto con una correcta implementacion de captura de excepciones.
Responsable:	Orlando Alarcón Gómez

Anexo C

Manual de Usuario