

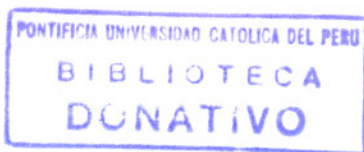


PONTIFICIA **UNIVERSIDAD CATÓLICA** DEL PERÚ

Esta obra ha sido publicada bajo la licencia Creative Commons  
Reconocimiento-No comercial-Compartir bajo la misma licencia 2.5 Perú.

Para ver una copia de dicha licencia, visite  
<http://creativecommons.org/licenses/by-nc-sa/2.5/pe/>





**PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ**

**ESCUELA DE GRADUADOS**



***"SISTEMA DE SOLICITUD DE SERVICIO PARA  
UNA INSTITUCIÓN EDUCATIVA"***

**TESIS PARA OPTAR EL TÍTULO DE  
MAGISTER EN INFORMÁTICA**

***PRESENTADA POR:***

**ING. VICTORIA LEÓN CHAN**

**LIMA-PERÚ**

**2002**

## DEDICATORIA

*A mi madre que siempre me ha apoyado e incentivado a ir cada vez más lejos  
A mi padre que estará siempre en mi corazón  
A mi hermano por estar siempre a mi lado*

## AGRADECIMIENTOS

*A la Pontificia Universidad Católica, mi alma mater.*

*Al Dr. Maynard Kong quien me asesoró en la presente tesis.*

*A Consuelo Moreno por sus palabras.*

*A mis compañeros de trabajo en la Dirección de Informática que me hacen sentir como en familia y que sin ellos no hubiera sido posible el desarrollo de la tesis.*

*A mi familia por estar ahí.*

*A los amigos por ser quienes son.*

## INDICE

<b>INTRODUCCION.....</b>	<b>1</b>
<b>CAPITULO I. METODOLOGIA DE TRABAJO.....</b>	<b>4</b>
1.1 Ciclo de vida de un proyecto. ....	
1.2 Análisis y diseño estructurado.....	5
1.3 Flujogramas del sistema.....	9
<b>CAPITULO II. PROCEDIMIENTOS PARA SOLICITAR SERVICIOS EN UNA INSTITUCION EDUCATIVA.....</b>	<b>10</b>
2.1 Oficina de Operaciones.....	
2.2 Oficina de Infraestructura.....	11
2.3 Facultades.....	12
2.4 Requerimientos. ....	14
<b>CAPITULO III. ESTRUCTURACION DE UNA SOLICITUD DE SERVICIO.....</b>	<b>16</b>
3.1 Concepto de metadata.....	17
3.2 Formulario de una solicitud de servicio.....	18
3.3 Datos que llena el solicitante en el formulario. ....	20
3.3.1 Concepto de XML.....	22
3.3.1.1 Características del XML. ....	
3.3.1.2 Estructura de un documento XML.....	23
3.3.2 Formulario electrónico generalizado para una solicitud de servicio. ....	24
<b>CAPITULO IV. DISEÑO DEL SISTEMA.....</b>	<b>28</b>
4.1. Planteamiento Funcional del sistema.....	
4.2. Planteamiento de la Base de Datos del sistema.....	38
4.3 Planteamiento de los Procesos del sistema.....	42
4.3.1. Conceptos sobre Aplicaciones.....	43
4.3.2. Arquitectura de las Aplicaciones.....	44
4.3.3. Aplicaciones del sistema.....	47
4.4. Planteamiento dinámico del sistema.....	49
4.4.1. Planteamiento de diagramas de estado para procesos del sistema. ....	
4.4.2. Planteamiento de diagramas de secuencia para procesos del sistema. ....	53
4.5 Control de acceso al Sistema.....	56



<b>CAPITULO V. FUNCIONES CRITICAS DESARROLLADAS.....</b>	<b>58</b>
5.1 Los APIs DOM y SAX. ....	59
5.2 Función para convertir un formulario electrónico en código HTML (presentación).	59
5.3 Función para convertir los datos de la solicitud en HTML a un documento XML.	61
5.4 Función para convertir un documento XML de una solicitud de servicio en código HTML (presentación). ....	63
<b>CAPITULO VI. CONCLUSIONES.....</b>	<b>66</b>

## **ANEXOS**

- ANEXO A. Código Fuente de aplicación: Recibir y atender varias solicitudes a la vez.
- ANEXO B. Código Fuente de función: Función para convertir un formulario Electrónico en código HTML
- ANEXO C. Código Fuente de función: Función para convertir los datos de la Solicitud en HTML a un documento XML
- ANEXO D. Código Fuente de función: Función para convertir un documento XML de una solicitud de servicio en código HTML
- ANEXO E: Manual de usuario para pedir un servicio
- ANEXO F: Manual de usuario para unidad de servicio
- ANEXO G: Ejemplos de formularios

## **Bibliografía**

## INTRODUCCION

Si bien es cierto que la actividad principal en una institución educativa es la enseñanza, existen actividades secundarias de índole administrativo que se realizan en sus ambientes.

Los alumnos además de asistir a clases y exámenes, dedican una parte de su tiempo a trámites documentarios en las facultades. Ejemplos de documentos: constancia de egresado, certificado de estudios, constancia de tercio superior, constancia de bachiller, cambio de especialidad, etc.

Asimismo, en toda institución educativa existen oficinas administrativas cuyas labores son de atender necesidades de diversa índole solicitado por otras oficinas. Necesidades como mantenimiento de muebles, telefonía, transporte, labores de pintura, mantenimiento de instalaciones eléctricas, mantenimiento de equipos, impresión y reproducción de documentos, etc.

La presente tesis propone un Sistema de Solicitudes de Servicio para una institución educativa de apoyo a las actividades previamente descritas. Esto permitirá que:

- 1) Una persona (alumno o personal administrativo) solicite un documento o servicio usando el sistema.
- 2) A través de este sistema, la oficina reciba la solicitud y haga un seguimiento del mismo.

Al analizar de manera general los procesos que se llevan a cabo para solicitar o tramitar, se aprecia que cada oficina (sea académica o administrativa) requiere “procesar” la información de manera distinta de acuerdo a su forma de trabajo.

Todo esto llevaría a considerar el desarrollo de varios sistemas de solicitudes de servicio. Uno para cada grupo de oficinas. Sin embargo, a lo largo de esta tesis se va a analizar estos mismos procesos pero desde otra óptica. El uso de nuevas herramientas informáticas hace más viable poder presentar una propuesta de un sistema de información que pueda ser utilizado en la mayoría de unidades administrativas y académicas. Y además que permita acceder a la información personalizada de acuerdo a la forma de trabajo de cada una de estas oficinas.

A continuación se indican los objetivos que se pretende alcanzar con la presente tesis:

- Proveer de un sistema de información que permita manipular los datos según lo requieran los procesos que se realizan en cada oficina administrativa y/o académica.
- Proveer de un sistema de información que almacene los datos de tal manera que facilite su acceso y uso en otros sistemas.
- Desarrollar un sistema de información que sirva como punto de partida para siguientes desarrollos de sistema para el apoyo a la gestión de cada oficina.

Y los objetivos del sistema de información son:

- Permita una comunicación directa, fluida y rápida entre el solicitante y la unidad prestadora de servicios.
- Reducir el tiempo que el alumno toma para tramitar un documento.
- Reducir el tiempo que las oficinas toman para solicitar un servicio.
- Agilizar la atención y seguimiento de las solicitudes..
- Eliminar en un 70% el uso de formularios en papel.

Para el estudio, desarrollo e implantación del sistema se tomó como modelo a la Pontificia Universidad Católica del Perú aprovechando que el sistema satisface las metas trazadas dentro de su plan estratégico y el uso de tecnología de punta en sus actividades. Sin embargo, este sistema podría ser implantado en otras entidades similares dado que el análisis y diseño también serían válidos.

La Tesis está organizada en 5 capítulos principales, seguidos de las conclusiones y anexos.

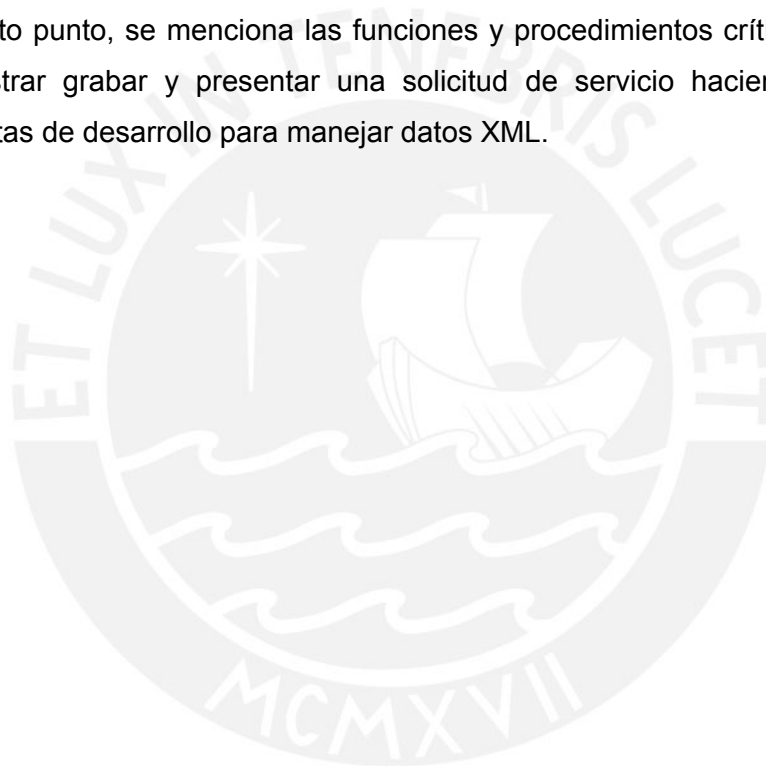
En el primer punto, se describe la metodología y conceptos utilizados de tal forma que el lector pueda entender las herramientas usadas para el análisis y diseño del sistema presentado en los siguientes puntos. En el segundo punto, se describe los procedimientos actuales que involucran tramitar un documento de un alumno; y solicitar y atender un servicio a una oficina. También se detallan las necesidades encontradas para ir acercándonos a posibles soluciones.



En el tercer punto, se explica el diseño propuesto para estructurar los datos contenidos en cada formulario de solicitud de servicio; aplicando los conceptos de metadata y XML. Esta etapa es crítica para, más adelante, definir el diseño del sistema de solicitud de servicio.

En el cuarto punto, se presenta el diseño propuesto que incluye: el modelo de datos que será la base del sistema y en el que se puede observar las relaciones que existen entre los datos; el modelo funcional, que ilustra las funciones que el sistema debe realizar y las aplicaciones propuestas.

En el quinto punto, se menciona las funciones y procedimientos críticos del sistema para registrar grabar y presentar una solicitud de servicio haciendo uso de las herramientas de desarrollo para manejar datos XML.



## CAPITULO I. METODOLOGIA DE TRABAJO

Se ha desarrollado la tesis tomando en cuenta las actividades involucradas en un proyecto:

Estudio de factibilidad.

Análisis.

Diseño.

Codificación

Pruebas.

Descripción de procedimientos.

Instalación.

Normalmente estas actividades se realizan en cascada. Es decir, antes de iniciar una actividad se debe terminar la anterior. Sin embargo, en ocasiones se ha requerido realizar actividades en paralelo e incluso volver a realizar actividades previas (retroalimentación). Esta manera de trabajo es útil por ejemplo, en el diseño se descubren detalles o errores no percibidos en la etapa del análisis. Este concepto que se aplica como metodología de trabajo se denomina ciclo de vida de un proyecto.

### 1.1 Ciclo de vida de un proyecto

Durante la primera etapa del estudio de factibilidad se han llevado reuniones con personal de las diversas áreas de la Universidad encargadas de la atención de solicitudes. Y en estas reuniones se discutieron los procesos, las necesidades, urgencias y deficiencias que tenían al recibir y atender los pedidos de la comunidad universitaria (alumnos, personal docente, personal administrativas, oficinas, etc.).

En algunos casos fue necesario diagramar los procesos para una mejor comprensión de las actividades que se desarrollan en cada ámbito.

En la siguiente etapa de análisis, se modeló el proceso propuesto con diagramas de flujo de datos, diagramas de entidad-relación u otras herramientas. El siguiente paso fue diseñar el sistema en base al análisis, es decir, hacer las especificaciones de cada proceso o actividad modelado en la etapa anterior. En este punto fue preferible desarrollar un prototipo. Esto obedece a las especificaciones muchas veces no son entendibles por personas ajenas al ámbito informático.

Para este fin, el prototipo será sólo una serie de pantallas con poco código de programa que ayudarán a dar una idea de cuál es el objetivo del sistema y cómo el usuario interactuaría con el sistema. En este proceso, el usuario o personal de la universidad dará sus apreciaciones y recomendaciones sobre su facilidad de uso, la presentación de pantallas, requerimientos, etc.

En la codificación se incluye la programación estructurada del sistema y la integración de los módulos que conforman el sistema de información. Las pruebas se realizan de dos maneras: pruebas de cada módulo y pruebas de todo el sistema. La descripción de procedimientos, documentación del funcionamiento de cada módulo, cuáles son sus *inputs* y cuál es el resultado obtenido. El resultado de todo esto es el manual para el usuario.

La instalación involucra la carga inicial de la información en la base de datos, demostración del sistema a los usuarios, entrenamiento y puesta en marcha del sistema. Adicionalmente existe otra etapa más que es el mantenimiento del sistema. En esta etapa es posible que se reciban sugerencias y comentarios que ayudarán a mejorar el sistema de información (retroalimentación).

## 1.2. Análisis y Diseño Estructurado

En la modelación del Sistema de Solicitudes de Servicio se utilizará el análisis y diseño estructurado. El análisis y diseño estructurado es una actividad basada en la construcción de modelos que reflejan el flujo y el contenido de la información.

Los modelos de análisis de sistemas son representaciones abstractas de lo que al final será una combinación de hardware, software y procedimientos realizados por personas. La construcción de modelos permite enfatizar ciertas propiedades críticas del sistema.

Las dos herramientas principales para modelar un sistema son: *diagrama de flujo de datos* y *diagrama entidad-relación*. Estas herramientas permiten simular a bajo costo un sistema complejo que se desee estudiar, enfocando las características importantes con la posibilidad de hacer cambios y correcciones a los requerimientos del usuario, y documentar de tal manera que los diseñadores y programadores puedan construir el sistema.

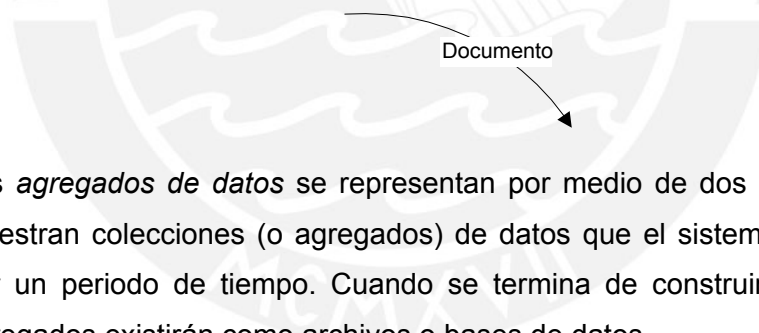
1. El *diagrama de flujo de datos* describe la transformación de entradas a salidas y proporciona una visión global de los componentes funcionales del sistema, es decir, permite resaltar las funciones que el sistema debe realizar.

Esta herramienta consiste en procesos, agregados de datos, flujos y terminadores:

- Los *procesos* se representan en el diagrama por medio de círculos o burbujas. Muestra una parte del sistema que transforma entradas en salidas. El proceso describe quién o qué lo está efectuando.



- Los *flujos* se muestran por medio de flechas curvas. Son las conexiones entre los procesos (funciones del sistema) y representan la información que dichos procesos requieren como entrada o la información que generan como salida. Describe el movimiento de bloques o paquetes de información de una parte del sistema a otra.



- Los *agregados de datos* se representan por medio de dos líneas paralelas. Muestran colecciones (o agregados) de datos que el sistema debe recordar por un periodo de tiempo. Cuando se termina de construir el sistema, los agregados existirán como archivos o bases de datos.

---

### REGISTRO DE FIRMA

---

- Los *terminadores* muestran las entidades externas con las que el sistema se comunica. Típicamente se trata de individuos o grupos de personas (por ejemplo, otro departamento o división dentro de la organización), sistemas de cómputo externos y organizaciones externas.

SOLICITANTE
-------------

2. El *diagrama entidad-relación* muestra la información al detalle de lo que hay en cada agregado de datos y la relación que existe entre ellos. Enfatiza en las relaciones entre los datos. Describe con alto nivel de abstracción la distribución de datos almacenados en un sistema.

Está conformado por dos componentes principales:

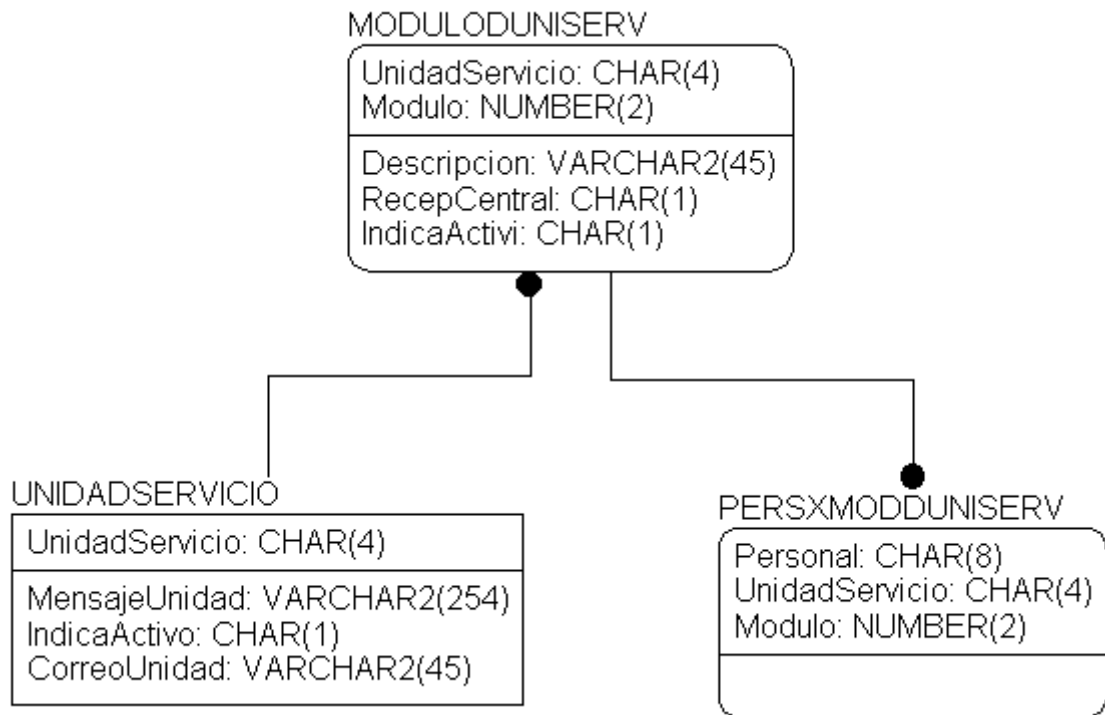
- *Entidades(Clases o Tipos de Objetos)*: se representan por medio de una caja rectangular en el diagrama. Esto representa una colección o conjunto de objetos (cosas) del mundo real cuyos miembros individuales juegan un papel necesario en el sistema que se construye; pueden identificarse de manera única por algún medio y cada uno puede describirse por uno o más datos.

TIPDATSERV

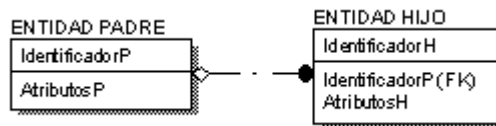
TipoDato: CHAR(2)
Descripcion: VARCHAR2(45)
Dato: VARCHAR2(24)
Abreviatura: VARCHAR2(15)
IndicaActivi: CHAR(1)
Longitud: NUMBER(3)
Precision: NUMBER(1)

- *Relaciones*: se representan por medio de líneas en el diagrama y que permiten conectar o asociar los diferentes tipos de entidades. Cada instancia de la relación representa una asociación entre cero o más ocurrencias de un objeto y cero o más ocurrencias del otro.
- A continuación un ejemplo completo de un *diagrama entidad-relación*.

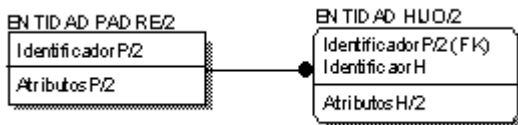




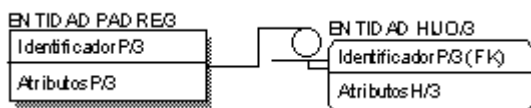
Descripción de los Tipos de Relaciones:



La Entidad Padre es un dato de referencia en la Entidad Hijo. Esto se traduce en que la Entidad Padre forma parte de los atributos de la Entidad Hijo.



La Entidad Padre forma parte del identificador de la Entidad Hijo. Esto se representa por la línea continua.



La Entidad Hijo es un subconjunto de la Entidad Padre, por consiguiente, el identificador de la Entidad Hijo es el mismo que la Entidad Padre.

### 1.3 Flujogramas del sistema.

Adicionalmente a los diagramas de flujo de datos, los flujogramas conforman una herramienta útil que muestran una visión integral del funcionamiento del sistema haciendo un seguimiento de los procesos de principio a fin.

Sirven también para separar responsabilidades por áreas funcionales (en caso de la Pontificia Universidad Católica del Perú: unidades organizacionales), y para ubicar claramente los procesos automatizados (soportados por aplicaciones específicas) así como los procesos manuales.



## CAPITULO II. PROCEDIMIENTOS PARA SOLICITAR SERVICIOS EN UNA INSTITUCION EDUCATIVA

Al presentar una solución a estas necesidades, se parte primeramente con el análisis de cómo son los procesos actuales para solicitar un servicio o un trámite en las oficinas respectivas. A continuación se presentarán los procesos que se realizan en algunas oficinas más representativas. Se comenzará con las unidades administrativas de la Universidad.

### Oficina de Operaciones

Esta oficina está dividida en cuatro secciones y cada una recibe pedidos de servicios varios:

- Sección de Abastecimientos (compras y almacenes): Encargada del abastecimiento adecuado y oportuno de bienes.
- Sección de Servicios: Encargada de la seguridad, limpieza, jardines, transporte, telefonía y mantenimiento de muebles.
- Control Patrimonial: Responsable del registro y control de los activos de la Universidad. Entre los pedidos que se hace a esta sección están: Préstamos de activos, Devolución de activos, salida de bienes del campus para eventos, etc.
- Imprenta: Encargada del diseño, diagramación, impresión y reproducción de documentos.

El procedimiento que se sigue para solicitar un servicio a esta oficina es la siguiente:

- Una oficina (unidad usuaria) llena un formulario en papel (3 copias).
- Es firmado y sellado por una autoridad de la unidad usuaria.
- La unidad usuaria entrega a la Oficina de Logística el documento.
- La Oficina de Logística verifica la validez del pedido en base al registro de firmas. Deriva la solicitud a la sección correspondiente para atender el pedido.

El registro de firmas es la relación de las personas de cada unidad usuaria que puede autorizar pedidos a la Oficina de Operaciones.

## 2.2 Oficina de Infraestructura

Paralelamente al análisis de los procesos en la Oficina de Operaciones, surgió en esta unidad una necesidad similar para realizar un seguimiento de sus solicitudes de trabajo.

La Oficina de Infraestructura está dividido a su vez en las siguientes secciones:

- Sección de Obras y Proyectos: Encargado del desarrollo de proyectos, ejecución de obras de construcción y remodelación de la infraestructura de la Universidad.
- Sección de Mantenimiento: Encargado del mantenimiento y reparación de la infraestructura de la Universidad. Labores como pintura, instalaciones eléctricas, instalaciones sanitarias, mantenimiento de equipos, etc.
- Sección de Gestión y Desarrollo: Encargado del archivo y actualización de la información relativa a la infraestructura de la Universidad como planos, estadísticas, expedientes técnicos, trámite de licencias; elaboración, planificación y ordenamiento del Campus. A esta sección llegan pedidos de señalización

El procedimiento normal que se sigue es el siguiente:

- La unidad usuaria llena un formulario en papel con los trabajos solicitados.
- Es firmado y sellado por una autoridad de la unidad usuaria.
- La unidad usuaria entrega el documento a la Secretaría de la Oficina de infraestructura.
- La Secretaría entrega el documento a la(s) área(s) encargadas.
- Cada área atiende el servicio indicado en la solicitud.

En el caso de una emergencia, el procedimiento difiere de esta manera:

- La unidad usuaria llama a la Oficina de Infraestructura.
- La Secretaría avisa a la(s) área(s) encargadas de la emergencia.
- El área atiende el servicio indicado en la solicitud.

- Posteriormente la Oficina de Infraestructura regulariza la solicitud.

### 2.3 Facultades

Las facultades atienden, en su mayoría, solicitudes o trámites de documentos de alumnos pertenecientes a la misma facultad. Ejemplos de solicitudes que gestiona un alumno:

- Constancia de egresado
- Solicitud de constancia.
- Carta de presentación
- Bachillerato automático y/o titulación
- Carta poder
- Permanencia / 4ta matrícula
- Reincorporación
- Cambio de especialidad

El proceso que sigue es el siguiente:

- El alumno va a la facultad y llena los datos en el respectivo formulario.
- La facultad recibe el formulario.
- La facultad atiende el pedido.
- Posteriormente el alumno regresa para recoger el documento o averiguar si su pedido fue aprobado.

El procedimiento que se sigue en cada facultad es similar a los procesos que se siguen en las unidades administrativas que se ha explicado en este mismo capítulo.

Sin embargo, la complejidad se presenta no en el proceso sino en las solicitudes en sí. Es decir, para cada solicitud debe llenarse un formulario. Y en cada formulario se llenan datos distintos. Para una mejor comprensión de lo expuesto, se ha seleccionado algunas solicitudes y la lista de datos que llenan:

#### Solicitud de Constancia:

- Código de alumno.
- Especialidad.



- Apellidos y nombre del alumno.
- Dirección.
- Teléfono.
- Semestre que cursa.
- Tipo de constancia (matrícula, egresado, certificado de estudios, tercio superior, bachiller, diploma, etc.)

#### Carta de presentación

- Código de alumno.
- Especialidad.
- Apellidos y nombre del alumno.
- Dirección.
- Teléfono.
- Nombre y cargo de la persona a la cual va dirigida la carta.
- Nombre de la Institución / Empresa a la cual va dirigida la carta.
- Motivo

#### Carta Poder

- Código del alumno que solicita.
- Especialidad del alumno que solicita.
- Apellidos y nombre del alumno que solicita.
- Código del alumno al que se le otorga el poder.
- Apellidos y nombre del alumno al que se le otorga el poder.
- Tipo de poder (matrícula, recojo de boleta de pago, recojo de trabajos y/o prácticas, etc.).

#### Solicitud de Reincorporación

- Código de alumno.
- Especialidad.
- Apellidos y nombre del alumno.
- Dirección.
- Teléfono.
- Último semestre en que se matriculó.
- Reincorporaciones anteriores.

## 2.4 Requerimientos

Resumiendo lo expuesto en este capítulo, se precisarán las necesidades encontradas agrupados según las entidades involucradas en los procedimientos descritos. Los cuales son:

### Unidad usuaria:

- Reducir el tiempo empleado en solicitar un servicio.
- Evitar, por desconocimiento u error, enviar una solicitud con un formato erróneo a la unidad de servicio que no atiende este servicio. Este caso ocurre especialmente cuando ocurren cambios en la estructura organizativa de la Universidad.

### Oficina de Logística y Servicios Generales:

- Agilizar la digitación de solicitudes en el sistema.
- Evitar los problemas de solicitudes escritos a mano, mala caligrafía, firmas y sellos ilegibles.
- Identificar la persona que está solicitando el servicio según el registro de firmas que contiene la relación de las personas autorizadas de cada unidad de la Universidad para solicitar servicios.
- Evitar servicios solicitados que no los atiende esta Oficina.
- Mejorar el control y seguimiento de las solicitudes de servicio.
- Reducir el tiempo en el procesamiento de las solicitudes.
- Evitar extravíos de documentos.
- Integración con los sistemas desarrollados en la Universidad.

### Alumno:

- Reducir el tiempo utilizado en tramitar un documento.

### Facultad:

- Evitar los problemas de solicitudes escritos a mano como mala ortografía, mala caligrafía.

- Validar los datos del alumno (Ej. Código, especialidad).
- Mejorar el control y seguimiento de las solicitudes de servicio.
- Evitar extravíos de documentos.
- Reducir tiempo en el procesamiento de las solicitudes.



### CAPITULO III. ESTRUCTURACION DE UNA SOLICITUD DE SERVICIO

Según las características de las personas que usarán el sistema, la orientación de la presente tesis será en el uso del Internet. Esto permitirá extender el sistema a cualquier parte del mundo. Solamente será necesario tener un navegador o browser y estar conectado a la red. A diferencia de los sistemas cliente-servidor, que deben ser instalados en la computadora de cada usuario, limitando la cantidad de usuarios sin mencionar el tiempo consumido por mantenimiento del software en los equipos de los usuarios.

Analizando los tres casos presentando en el capítulo anterior (Oficina de Logística, Ofician de Servicios Generales y Facultad), se podría decir que existen similitudes en cuanto al procedimiento para solicitar un servicio o tramitar documento. Por tanto es factible desarrollar un único sistema de información que cumpla con las necesidades de todos ellos. Sin embargo, la complejidad en el diseño del sistema se da en la diversidad de datos existentes en cada formulario y en la cantidad de servicios.

El dilema en esta etapa es si desarrollar un sistema general o desarrollar sistemas especializados. Un sistema especializado para una oficina brinda la gran posibilidad que el sistema se adapte en su totalidad a los procesos e información que se maneja de esta oficina pero con posibilidad de no ser enteramente utilizado en otras oficinas. No se desarrollaría un sistema sino dos o más sistemas y por tanto se emplearía más recursos en el desarrollo y en el mantenimiento.

Por otro lado, un sistema general puede ser utilizado en varias oficinas pero esto conlleva el riesgo de no poderse adaptar satisfactoriamente a los procesos y a la estructura de la información que se maneja en estas oficinas. Un punto a su favor es el menor costo en el uso de recursos pues sólo se desarrolla un único sistema y no varios.

El punto crítico para responder a la pregunta de desarrollar sistemas especializados o desarrollar un único sistema general, es cómo modelar los datos de las solicitudes de los servicios. Partiendo de una catalogación de las solicitudes de servicio se puede identificar cuáles son los que atiende cada oficina.

Pero esto no es suficiente. El sistema debe permitir registrar toda la información contenida en la solicitud. De acuerdo a la variedad de solicitudes que existe, identificar a una solicitud con una lista de atributos como: Nombre del solicitante, fecha de la solicitud, nombre del servicio, tipo de constancia, fecha inicio, fecha fin, ciclo, nombre de empresa, número de recibo de pago, proveedor sugerido, descripción del trabajo/servicio, etc.; no es lo óptimo.

### 3.1 Concepto de metadata

Antes de continuar, es conveniente entender qué es este concepto. Se sabe que la información es datos que tiene un significado para el que lo recibe. En términos de la computación, la información es convertida en data, es colocada en una computadora donde es guardada y procesada como data, y esta misma data es luego presentada de tal manera que es percibida como información para el lector. El término metadata (o metadato) ha sido usado en los últimos 15 años y se ha vuelto muy popular especialmente en el ambiente de la World Wide Web.

La metadata no tiene una definición exacta ni clara. Se dará varias definiciones<sup>1</sup> que ayudarán a entender mejor esta idea.

El prefijo *meta* se entiende en el contexto de la tecnología de la información “como descripción o definición de algo” y *metadata* es la descripción o definición de data.

*Metadata* es data estructurada de la data. Describe atributos y contenidos de un documento original o de un recurso electrónico. También es data que provee información sobre un recurso.

El propósito del uso de la *metadata* es de facilitar y mejorar el acceso a la información.

---

<sup>1</sup> Definición de la entidad World Wide Web Consortium: [www.w3.org](http://www.w3.org)



Un ejemplo típico es el catálogo de una biblioteca que contiene información (metadata) sobre publicaciones o libros (data).

Existen tres tipos de metadata:

- Descriptiva. Describe e identifica los recursos de la información. Ejemplo de uso. Búsqueda de recursos por la Web, atributos bibliográficos como título, autor, lenguaje.
- Estructural: Facilita la navegación y presentación de recursos electrónicos. Ejemplo de uso: tags estructurados como título de página, tabla de contenido, capítulos, índice.
- Administrativo: Facilita la administración de corto y largo plazo y el procesamiento digital de recursos. Ejemplo de uso: Datos técnicos de creación, control de calidad, administración de permisos, control de acceso a recursos.

Aplicando este concepto, se agrupará la información contenida en una solicitud en dos partes:

- 1) Los datos del formulario de la solicitud en blanco. En otras palabras, el "texto" existente en una hoja preimpresa sin llenar.
- 2) Y los datos que el solicitante llena en el formulario.

### 3.2 Formulario de una solicitud de servicio

El contenido de un formulario puede definirse como una lista de datos con características como título, tipo de dato y longitud del dato. En otras palabras, el formulario vendría a ser la metadata o la estructura de datos de una solicitud de servicio propiamente dicha. Por ejemplo, para una solicitud de carta de presentación de un alumno sería el siguiente:

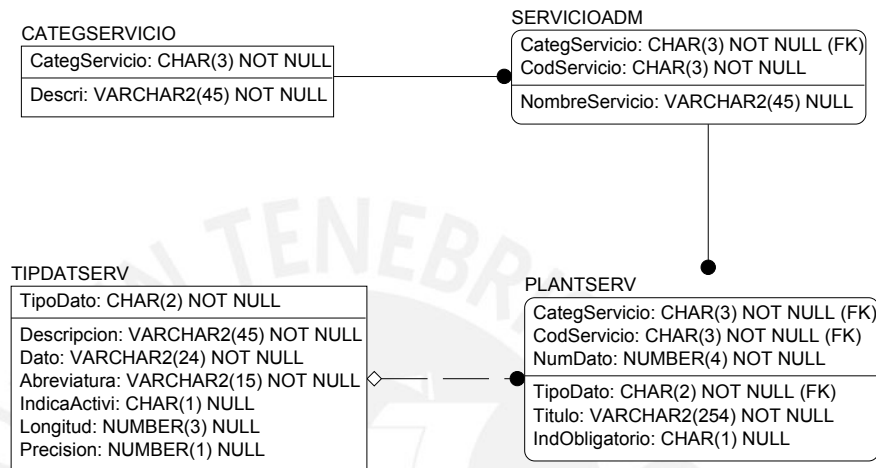
Posición	Nombre del dato	Tipo de dato	Longitud
1	Código de alumno.	Alfanumérico	8
2	Especialidad.	Alfanumérico	45
3	Apellidos y nombre del alumno.	Alfanumérico	75

4	Fecha de la solicitud	Fecha	
5	Dirección.	Alfanumérico	45
6	Teléfono.	Alfanumérico	7
7	Nombre y cargo de la persona a la cual va dirigida la carta.	Alfanumérico	45
8	Nombre de la Institución / Empresa a la cual va dirigida la carta.	Alfanumérico	45

Y para una solicitud de servicio de la Oficina de Logística sería:

Num	Título	Tipo de dato	Longitud
1	Nombre del solicitante.	Alfanumérico	45
2	Unidad Solicitante	Alfanumérico	45
3	Fecha	Fecha	
4	Número de la solicitud	Numérico	6
5	Teléfono / Anexo	Alfanumérico	8
6	Item 1: Servicio solicitado	Alfanumérico	45
7	Item 1: Proveedor sugerido	Alfanumérico	45
8	Item 1: Número del Activo	Numérico	8
6	Item 2: Servicio solicitado	Alfanumérico	45
7	Item 2: Proveedor sugerido	Alfanumérico	45
8	Item 2: Número del Activo	Numérico	8
6	Item 3: Servicio solicitado	Alfanumérico	45
7	Item 3: Proveedor sugerido	Alfanumérico	45
8	Item 3: Número del Activo	Numérico	8

Sobre la base de estos ejemplos, el formulario de la solicitud es modelado de esta manera:



Hasta el momento se ha planteado el formulario de la solicitud de servicio y esto sólo es la primera parte. Ahora se diseñará el modelo de datos para la información que el solicitante llena en el formulario.

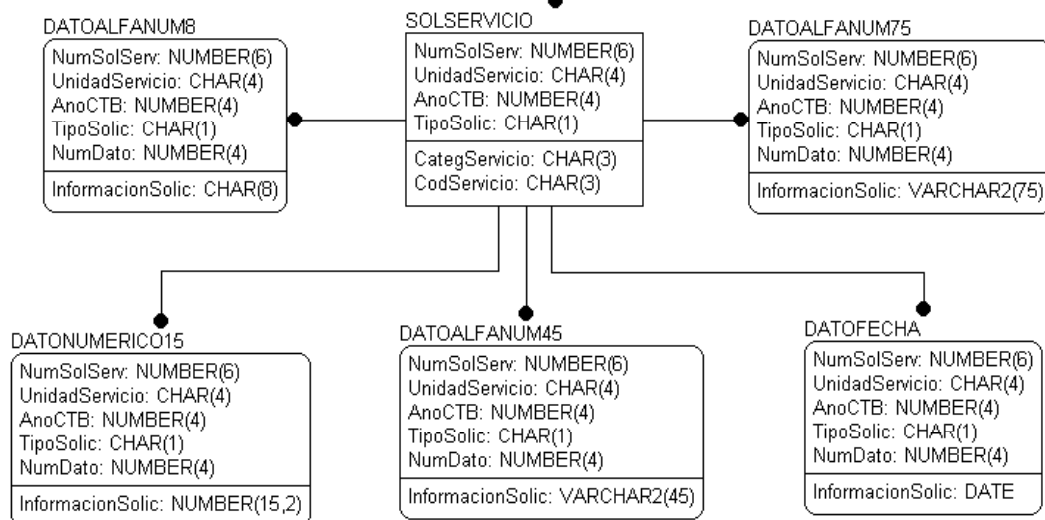
### 3.3 Datos que llena el solicitante en el formulario

Se presenta el siguiente ejemplo de una solicitud estructurada que llena un alumno.

NumDato	Título del dato	Información del solicitante
1	Código de alumno.	20024550
2	Especialidad.	Ingeniería Industrial
3	Apellidos y nombre del alumno.	Pérez Pérez, Juan
4	Fecha de la solicitud	20-05-2002
5	Dirección.	Av. Brasil 560
6	Teléfono.	222 – 3456

7	Nombre y cargo de la persona a la cual va dirigida la carta.	Ing. Guillermo Franco – Director
8	Nombre de la Institución / Empresa a la cual va dirigida la carta.	Fábrica de Calzado La Juvenil S.A.

En este ejemplo los datos que el alumno llena son alfanumérico de longitud 8, alfanumérico de longitud 45 y fecha. Si la información del ejemplo se almacena en tablas de acuerdo al tipo de dato y al formulario electrónico definido, se tendrá un modelo como el que mostramos a continuación:



Y para el ejemplo la información se almacenará de esta manera:

TABLA / ENTIDAD	NUMDATO	INFORMACIONESOLIC
DATOALFANUM8	1	20024550
DATOALFANUM8	6	222 – 3456
DATOALFANUM45	2	Ingeniería Industrial

DATOALFANUM45	5	Av. Brasil 560
DATOALFANUM45	7	Ing. Guillermo Franco – Director
DATOALFANUM45	8	Fábrica de Calzado La Juvenil S.A.
DATOALFANUM75	3	Pérez Pérez, Juan
DATOFecha	4	20-05-2002

El modelo diseñado ayuda en el registro de la información de una solicitud sin embargo, el desarrollo del sistema se hace más complejo y además no serviría en los casos que se modifique o actualice un formulario electrónico.

### 3.3.1 Concepto de XML

Dentro de las tendencias actuales en las tecnologías de información respecto al Internet y la metadata, se encuentra el XML.

XML (Extensible Markup Language) es un metalenguaje utilizado para definir otros lenguajes. El XML es utilizado para estructurar los datos contenidos en documentos y pueda ser usado o interpretado por otras aplicaciones. Un lenguaje de marcado (“Markup language”) es un mecanismo para identificar estructuras en un documento. Un ejemplo de un documento XML

```
<?xml version="1.0"?>
<cliente>
  <persona>
    <apellido>PEREZ</apellido>
    <nombre>JUAN</nombre>
  </persona>
  <ruc>10838374742</ruc>
</cliente>
```

#### 3.3.1.1 Características del XML

Las características del XML son:

- Estándar abierto para el intercambio de datos entre aplicaciones de manera simple y no propietario. Es portable.
- La estructura de datos es independiente del programa. Separa los datos de los procesos.

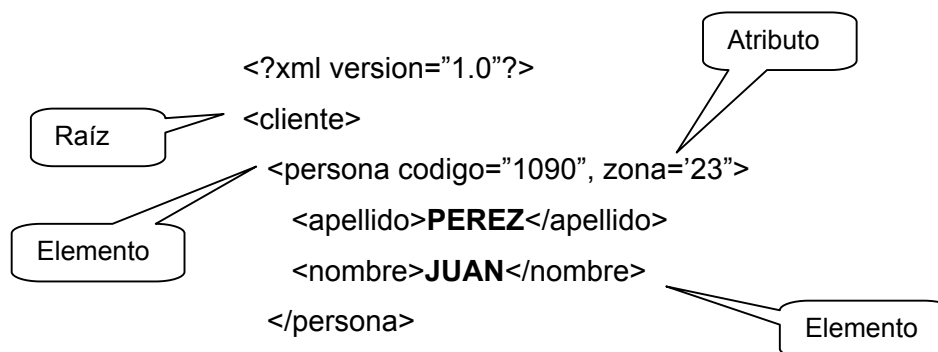
- Por ser un lenguaje extensible, uno puede crear sus propios tags para identificar los datos de acuerdo a las especificaciones del negocio. Ejemplo, los tags para estructurar información sobre libros incluirían <título>, <autor>, <año publicación>, <editorial>.
- Separa presentación del contenido. La presentación se personaliza en función al tipo de usuario y/o reglas de acceso.
- Mejora la búsqueda de contenido. Búsqueda por etiquetas, búsquedas dentro de etiquetas. No sólo busca por datos sino también por metadatos.
- Fácil de manipular.

La visión del XML, según sus creadores, debe ser fácil de usar en el Internet, debe soportar una gran variedad de aplicaciones, las aplicaciones que procesan documentos XML deben ser sencillos de desarrollar, los documentos XML deben ser entendibles por cualquier persona, fácil de crear.

### 3.3.1.2 Estructura de un documento XML

Un documento XML debe cumplir ciertas reglas para considerarlo bien formado (“well-formed”).

- La declaración XML debe estar obligatoriamente al inicio del documento.
- El documento debe contener un elemento conocido como raíz (“root”) que alberga a los demás elementos.
- Todos los elementos deben tener un tag inicial y un tag final: <nombretag>dato</nombretag>.
- Elementos vacíos debe tener un solo tag <nombretag/>.
- Los valores de los atributos deben estar entre comillas.





```
<ruc>10838374742</ruc>
</cliente>
```

Se recomienda usar los elementos para contener información útil para el usuario y atributos para contener o definir metadatos.

### 3.3.2 Formulario electrónico generalizado para una solicitud de servicio

De lo explicado en este capítulo se concluye que las solicitudes de servicio son documentos estructurados. Hasta el momento se ha explicado cómo definir un documento XML de la forma <nombre del dato>contenido</ nombre del dato>. Esta forma de representar la información está bien para una solicitud, pero al generalizar para solicitudes distintas como en la presente tesis hay que incluir la estructura del formulario en el documento XML.

De esta manera, la solicitud del ejemplo, transformado a documento XML será

```
<?xml version="1.0"?>
<INFOADIC>
<ITEM num="1", obligatorio="s", tipodato="varchar2", longitud="8">
<NOMBRE>codalumno</NOMBRE>
<TEXTO>20024550</TEXTO>
</ITEM>
<ITEM num="2", obligatorio="s", tipodato="varchar2", longitud="45">
<NOMBRE>especialidad</NOMBRE>
<TEXTO> Ingeniería Industrial </TEXTO>
</ITEM>
<ITEM num="3", obligatorio="s", tipodato="varchar2", longitud="75">
<NOMBRE>apellidoynombre</NOMBRE>
<TEXTO> Pérez Pérez, Juan </TEXTO>
</ITEM>
<ITEM num="4", obligatorio="s", tipodato="fecha">
<NOMBRE>fechasolicitud</NOMBRE>
<TEXTO> 20-05-2002</TEXTO>
</ITEM>
<ITEM num="5", obligatorio="s", tipodato="varchar2", longitud="45">
```

```

<NOMBRE>direccion</NOMBRE>
<TEXTO> Av. Brasil 560</TEXTO>
</ITEM>
<ITEM num="6", obligatorio="s", tipodato="char", longitud='8'>
<NOMBRE>telefono</NOMBRE>
<TEXTO> 222-3456</TEXTO>
</ITEM>
<ITEM num="7", obligatorio="s", tipodato="varchar2", longitud="45">
<NOMBRE>nombre y cargo de la persona</NOMBRE>
<TEXTO> Ing. Guillermo Franco – Director </TEXTO>
</ITEM>
<ITEM num="8", obligatorio="s", tipodato="varchar2", longitud="45">
<NOMBRE> Nombre de la Institución </NOMBRE>
<TEXTO> Fábrica de Calzado La Juvenil S.A.</TEXTO>
</ITEM>
</INFOADIC>

```

En general, dependiendo del tipo de dato se define cada elemento de la siguiente forma:

- Texto de longitud **nn**:

```

<ITEM num="numdato", obligatorio="s", tipodato="varchar2", longitud="nn">
<NOMBRE>título del dato en el formulario</NOMBRE>
<TEXTO>información que llena el solicitante</TEXTO>
</ITEM>

```

- Numérico de longitud **nn**, precisión **pp**:

```

<ITEM num="numdato", obligatorio="s", tipodato="varchar2", longitud="nn">
<NOMBRE>título del dato en el formulario</NOMBRE>
<TEXTO>información que llena el solicitante</TEXTO>
</ITEM>

```

- Numérico de longitud **nn** y precisión **pp**:

```
<ITEM num="numdato", obligatorio="n", tipodato="varchar2", longitud="nn">
<NOMBRE>título del dato en el formulario</NOMBRE>
<TEXTO>información que llena el solicitante</TEXTO>
</ITEM>
```

- Fecha:

```
<ITEM num="numdato", obligatorio="n", tipodato="fecha">
<NOMBRE>título del dato en el formulario</NOMBRE>
<FECHAINI>información que llena el solicitante</FECHAINI>
</ITEM>
```

- Rango de fechas:

```
<ITEM num="numdato", obligatorio="n", tipodato="rangoFecha">
<NOMBRE>título del dato en el formulario</NOMBRE>
<FECHAINI>información que llena el solicitante</FECHAINI>
<FECHAFIN>información que llena el solicitante</FECHAFIN>
</ITEM>
```

- Hora:

```
<ITEM num="numdato", obligatorio="n", tipodato="hora">
<NOMBRE>título del dato en el formulario</NOMBRE>
<HORAINI>información que llena el solicitante</HORAINI>
</ITEM>
```

- Rango de horas:

```
<ITEM num="numdato", obligatorio="n", tipodato="rangoHora">
<NOMBRE>título del dato en el formulario</NOMBRE>
<HORAINI>información que llena el solicitante</HORAINI>
```

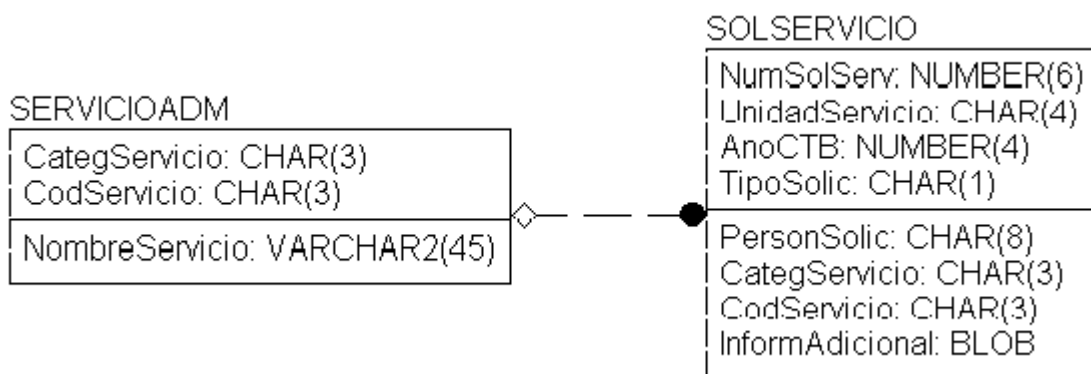
<HORAFIN>información que llena el solicitante</HORAFIN>  
</ITEM>

- Opción Si-No:

<ITEM num="numdato", obligatorio="n", tipodato="sino" >  
<NOMBRE>título del dato en el formulario</NOMBRE>  
<TEXTO>información que llena el solicitante</TEXTO>  
</ITEM>

Este planteamiento da facilidades en cuanto al almacenamiento de datos, portabilidad del documento XML entre aplicaciones y sistemas. Además permite que la solicitud de servicio sea independiente del formulario; es decir, permite modificar los metadatos en un formulario sin que existan inconsistencias en las solicitudes previamente almacenadas en la base de datos.

El modelo de datos es más simple, incluyendo el atributo **InformAdicional** (documento XML) a la entidad **Sol Servicio** (solicitud de servicio) será el siguiente:



## CAPITULO IV. DISEÑO DEL SISTEMA

En el presente capítulo se detalla el diseño del sistema partiendo primeramente de los diagramas de flujo de datos, el modelamiento de la base de datos y las aplicaciones propuestas.

### 4.1 Planteamiento Funcional del sistema

Como ya se mencionó, el *diagrama de flujo de datos* es una técnica gráfica que representa el flujo de la información y las transformaciones que se aplican a los datos al moverse desde la entrada hasta la salida de cada proceso.

A continuación se presentan los *diagramas de flujo* del sistema:



[Diagrama contexto]





[diagrama cero]



[diagrama 1.0]



[diagrama 2.0]



[diagrama 3.0]



[diagrama 4.0]



[diagrama 5.0]





[flujograma unidad solicitante]



[flujograma alumno]



#### 4.2. Planteamiento de la Base de Datos del sistema

Para hacer el planteamiento más ordenado y dar mayor facilidad para entenderlo se muestran 3 vistas del *diagrama entidad-relación*: en la primera vista se muestran todas las entidades planteadas con sus definiciones pero sin ningún atributo, con el fin de dar una visión integral del sistema; en la segunda se muestran todas las entidades, con sus respectivos atributos, que definen los tipos de servicios, formularios, autorizaciones de usuarios, etc.; y en la tercera todas las entidades del proceso de Pedido de Servicio.









#### 4.3. Planteamiento de los Procesos del sistema

El Sistema de Solicitud Servicio que planteado involucra una serie de procesos y funciones necesarios para su implantación. En su mayoría, tales procesos y funciones serán apoyadas con aplicaciones orientadas a automatizarlos, para una toma de decisiones rápida y lograr un funcionamiento integral que fomente la productividad y la calidad del sistema.

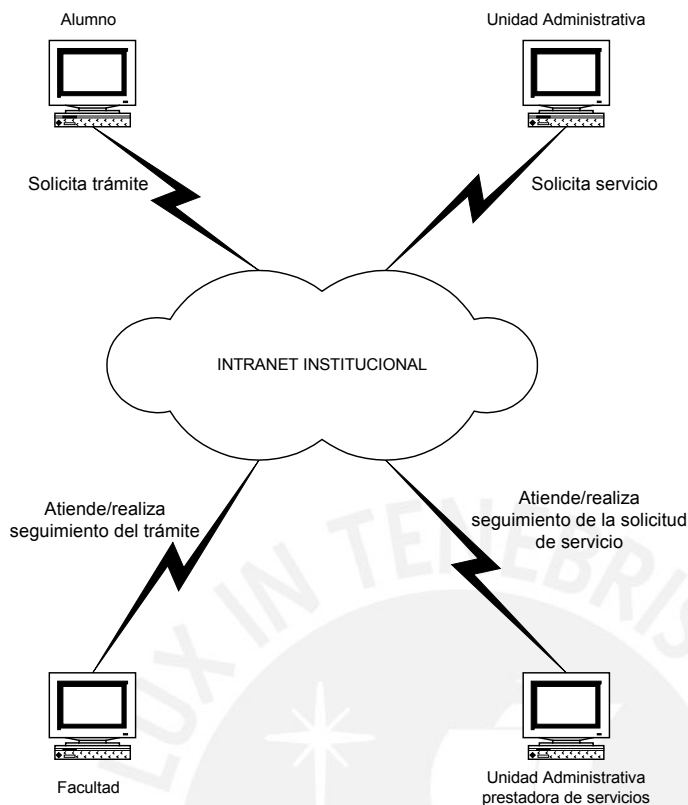
En el planteamiento de los procesos, se consideraron a los usuarios que usarán este Sistema. Se clasificaron a los usuarios de la siguiente manera:

1. Los que solicitan servicios.
2. Los que atienden servicios.

En el primer grupo se identifican a los alumnos y al personal de la Universidad (docentes y no docentes, jefes de unidades, secretarias, asistentes, etc.) que posiblemente tengan algo de conocimiento de la informática y del uso de la computadora. En este grupo pertenece un buen porcentaje de la comunidad universitaria. A este grupo vamos a denominarlos con el término *Solicitante* o *Unidad Solicitante*. En el segundo grupo se identifican a las oficinas y facultades prestadoras de servicios y al personal que labora en ellas. A este grupo lo vamos a llamar *Unidad de Servicio*.

Teniendo en cuenta estas consideraciones se vio conveniente desarrollar el sistema en el Intranet de la universidad. Esto proporciona como ventaja que cualquier persona pueda acceder al sistema de cualquier computadora conectada al Internet y un browser. A continuación se muestra un esquema de la forma cómo acceder al Sistema.





Las herramientas informáticas utilizadas para el desarrollo de las aplicaciones fueron: Erwin 3.1 para el modelado de la base de datos, Java Developer para la programación de las aplicaciones en Intranet, y Oracle 8i como manejador de base de datos y para la programación de procedimientos en ambiente servidor.

#### 4.3.1. Conceptos sobre Aplicaciones

Las aplicaciones automatizadas son módulos de software orientados a realizar una tarea específica.

Existen diferentes tipos de aplicaciones, las cuales pueden ser:

- Aplicaciones de actualización: dan mantenimiento a la información del sistema en línea.
- Aplicaciones de búsqueda: definen criterios necesarios para ubicar el(los) registro(s) de datos y no llevan a cabo procesamiento adicional sobre la información a mostrar.

- Aplicaciones de consulta: muestran información ad-hoc a las necesidades del usuario permitiendo el acceso a información relacionada con la entidad principal consultada.

Pueden involucrar procesamiento adicional sobre la información a mostrar.

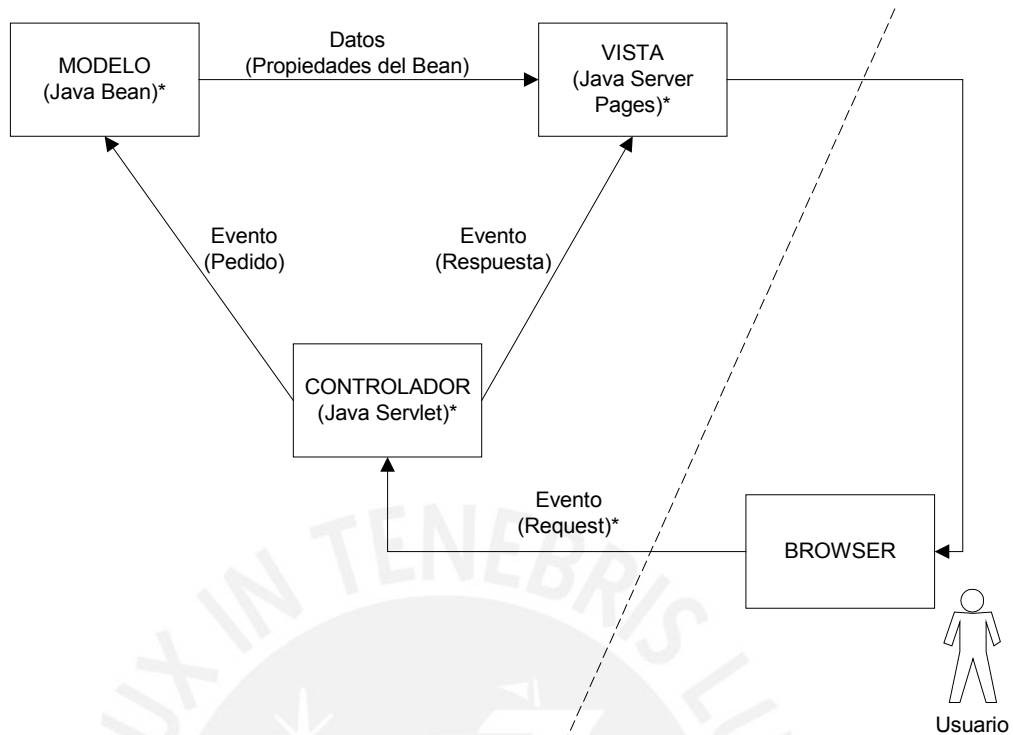
Cuando se desarrolla una aplicación es recomendable:

- Establecer el uso de estándares en la forma de trabajar al interior de ellas, por ejemplo definir la nomenclatura de variables, objetos a usar, funciones comunes, etc. De esta forma se reduce el tiempo de implementación, se facilita el mantenimiento y se pueda reutilizar la aplicación entre diferentes sistemas.
- El uso adecuado de las interfaces gráficas en el diseño de pantallas hace que la aplicación sea más natural, familiar, atractiva y finalmente fácil de utilizar para el usuario. Dan formas directas e intuitivas para realizar tareas.
- Modularidad para programar, esto además de hacer más ordenada la forma de trabajo facilita el mantenimiento de las aplicaciones.
- Deben mostrar sólo lo que se necesite, dar notoriedad a funciones importantes y hacer simples las acciones comunes; también se deben proveer opciones por defecto.

#### 4.3.2. Arquitectura de las Aplicaciones

Para la presente tesis se utilizará la arquitectura Modelo-Vista-Controlador, la cual es un modelo para el desarrollo de aplicaciones en Internet.

El modelo Modelo-Vista-Controlador (MVC) interactúa como se muestra en la figura. El modelo mantiene y almacena la data, la Vista recibe la data y genera una respuesta dinámica y el Controlador recibe las órdenes y decide si lo delega al Modelo o la Vista.



## El Modelo

El modelo representa la lógica del negocio en una aplicación y encapsula las reglas del negocio. Puede ser dividido en 2 componentes: Estado y Acción.

## Componente de Estado

El estado define los juegos posibles de valores del modelo e incluye los métodos para cambiar dichos valores. Los JavaBeans son la opción lógica para implementar este componente, ya que éstos son los suficientemente independientes y pueden ser accedidos desde cualquier aplicación. Al utilizar este componente provee de lo siguiente:

- Reutilización, permite a diferentes aplicaciones hacer uso de la misma “lógica del negocio”.
- Calidad, al colocar la lógica del negocio en un solo lugar, permite revisar y probar la lógica al detalle. Si la lógica estuviera en cada aplicación, aumentaría el costo de mantenimiento.

- Robustez, al encapsular la lógica del negocio, permite la reutilización y un mejor manejo de los errores.

### **Componente de Acción**

Las acciones definen los cambios posibles al estado en respuesta a un evento. Es aquí donde se definen las operaciones, muchas de las cuales son complejas, sobre los valores definidos en el componente de Estado.

### **La Vista**

La vista representa la lógica de presentación o “interfaz humana” de una aplicación. Este componente obtiene el estado actual del sistema a partir del modelo, y provee una interfaz al usuario. La vista es responsable de mostrar información específica que el usuario haya requerido en algún momento. En este caso se ha utilizado el JSP (Java Server Pages) como interfaz al usuario.

### **El Controlador**

El controlador es el responsable de recibir los eventos o las órdenes enviadas por el usuario, decidir qué función se deberá ejecutar y luego enviar una respuesta apropiada al usuario.

El controlador debe manejar las siguientes tareas:

- Seguridad, debe verificar la autenticación (la persona que ingresa al sistema es quien dice ser) y las autorizaciones (si es que tiene el permiso de ejecutar el evento).
- Identificación de eventos, debe identificar el evento que va a ser ejecutado.
- Procesar el evento, debe de analizar el pedido del usuario e invocar a la función apropiada.
- Manejo de errores, debe manejar cualquier error que se presente, utilizando una página de errores.

Con la utilización del lenguaje Java, los servlets son la selección ideal para el desarrollo del controlador.

Los beneficios de una arquitectura MVC incluye una separación muy clara entre el contenido dinámico y la presentación. El controlador proporciona un punto de entrada sencillo a la aplicación web, donde se puede centralizar convenientemente la lógica para autenticación de usuario, la validación de parámetros, las búsquedas de otras tareas similares. El programa controlador puede seleccionar dinámicamente diferentes páginas JSP de presentación, dependiendo de los parámetros de la petición, mostrando así diferentes vistas de los mismo datos. Las páginas JSP de presentación recuperan cualquier Bean que haya sido inicializado previamente por el programa controlador, y llaman a los métodos apropiados para extraer y presentar de forma clara el contenido dinámico.

En el anexo A se presenta un ejemplo de código fuente utilizando la arquitectura MVC.

#### 4.3.3. Aplicaciones del Sistema

Las aplicaciones se dividieron en tres procesos según los cuales son:

**Proceso de Solicitud de Servicio:** Comprende a todas las aplicaciones que usa el solicitante.

1) Pedido de servicio: Es el pedido del servicio propiamente dicha. Esta aplicación da inicio al proceso de Solicitud de Servicio. El usuario es llevado por una serie de “pantallas” de una manera sencilla hasta lograr su objetivo de enviar su pedido electrónicamente hasta la Unidad de Servicio.

2) Solicitudes por aprobar: Si el pedido proviene de una oficina de la universidad, se necesita de la aprobación o visto bueno de una autoridad de dicha oficina. Esta autoridad generalmente es un decano, un director, un jefe de área o sección, etc.

3) Actualizar solicitud por aprobar: Mientras una solicitud no ha sido aprobada por el Jefe de la Unidad, ésta puede ser modificada las veces que se desee por cualquier persona de la oficina (persona de apoyo y/o persona que autoriza).

4) Rechazar una solicitud por aprobar: Mientras una solicitud no ha sido aprobada por el Jefe de la Unidad, ésta puede ser rechazada. Es decir, borrada de la base de datos. Esta actividad lo puede realizar la persona de apoyo o la persona que autoriza.

5) Consulta de solicitudes enviadas: Una vez que el solicitante (sea alumno o unidad de la PUCP) envíe una solicitud, ésta ya no puede ser modificada bajo ninguna circunstancia. Lo único que se puede hacer es consultar su estado (por atender, en proceso, atendido, rechazado). El estado es actualizado por la unidad de servicio.

**Proceso de Recepción y Atención de Solicitudes de Servicio:** Aplicaciones que usa la unidad de servicio.

1) Solicitudes por Recibir: Lista de todas las solicitudes que han enviado los solicitantes y que aún no están siendo atendidos por la unidad de servicio.

2) y 3) Se han desarrollado dos aplicaciones para atender las solicitudes según:

Recibir y atender varias solicitudes a la vez (Aplicación “en batch” o en lote).

Recibir y atender una única solicitud a la vez (Aplicación de actualización: *Actualización del estado de la solicitud*).

4) y 5) Una vez recibido las solicitudes, se deriva las mismas a la sección o persona responsable del proceso. Cada vez que la persona responsable termina su trabajo, se encarga luego de actualizar el estado de la solicitud: atendido, en proceso, rechazado o la envía a otra sección o persona responsable dentro de la unidad de servicio. Esto se realiza mediante estas dos aplicaciones:

Atención de varias solicitudes a la vez (Aplicación “en batch” o en lote).

Atención de una única solicitud a la vez (Aplicación de actualización: *Actualización del estado de la solicitud*).



6) Búsqueda de solicitudes: Esta aplicación es primordial para permitir que la unidad de servicio realice un mejor control y seguimiento a las solicitudes de servicio que está atendiendo. Muestra una lista de las solicitudes. Al seleccionar una solicitud permite ver el detalle del mismo.

7) Nueva solicitud de servicio: A pesar que el solicitante puede hacer su pedido desde el sistema, siempre cabe la posibilidad que éste haga su pedido por otros medios debido a la urgencia del mismo o porque no se dispone de una computadora conectada a Internet. La unidad de servicio puede con esta aplicación llenar la solicitud de servicio para luego seguir con el proceso normal en el Sistema.

8) Registro de firmas: Esta aplicación permite que la unidad de servicio lleve la relación de las personas autorizadas para solicitar servicios. Las autorizaciones se dan por oficina o parte de ella, fechas de inicio y fin, nivel de autorización (como persona de apoyo o como persona que autoriza).

9) Búsqueda del registro de firmas: Aplicación para consultar la relación de las personas autorizadas para solicitar servicios a nombre de su unidad.

10) Formulario para la solicitud de servicio: Esta aplicación permite que la misma unidad de servicio actualice el formulario según sus requerimientos. El formulario lo conforma una lista de datos y cada dato debe tener características como nombre, tipo de dato, longitud máximo y si es obligatorio o no.

11) Búsqueda de formularios: Aplicación que nos lista los servicios que provee la unidad de servicio y el detalle de los formularios.

#### 4.4. Planteamiento dinámico del sistema

##### 4.4.1. Planteamiento de diagramas de estado para procesos del sistema

El diagrama de estado es usado para mostrar como un elemento cambia de estado en el tiempo, las transiciones permitidas y las condiciones para la transición.



Un estado es la representación de un elemento que tiene una particular condición. Es una condición durante la vida de un objeto o una interacción durante por la cual se satisface una condición, se ejecuta alguna acción o se espera por algún evento.

Los diagramas de estado incluyen también flujos de estado (transiciones) y nodos de inicio y fin.

Un estado es dibujado como un rectángulo con las esquinas redondeadas:



El nodo de inicio es usado para indicar la posición de inicio



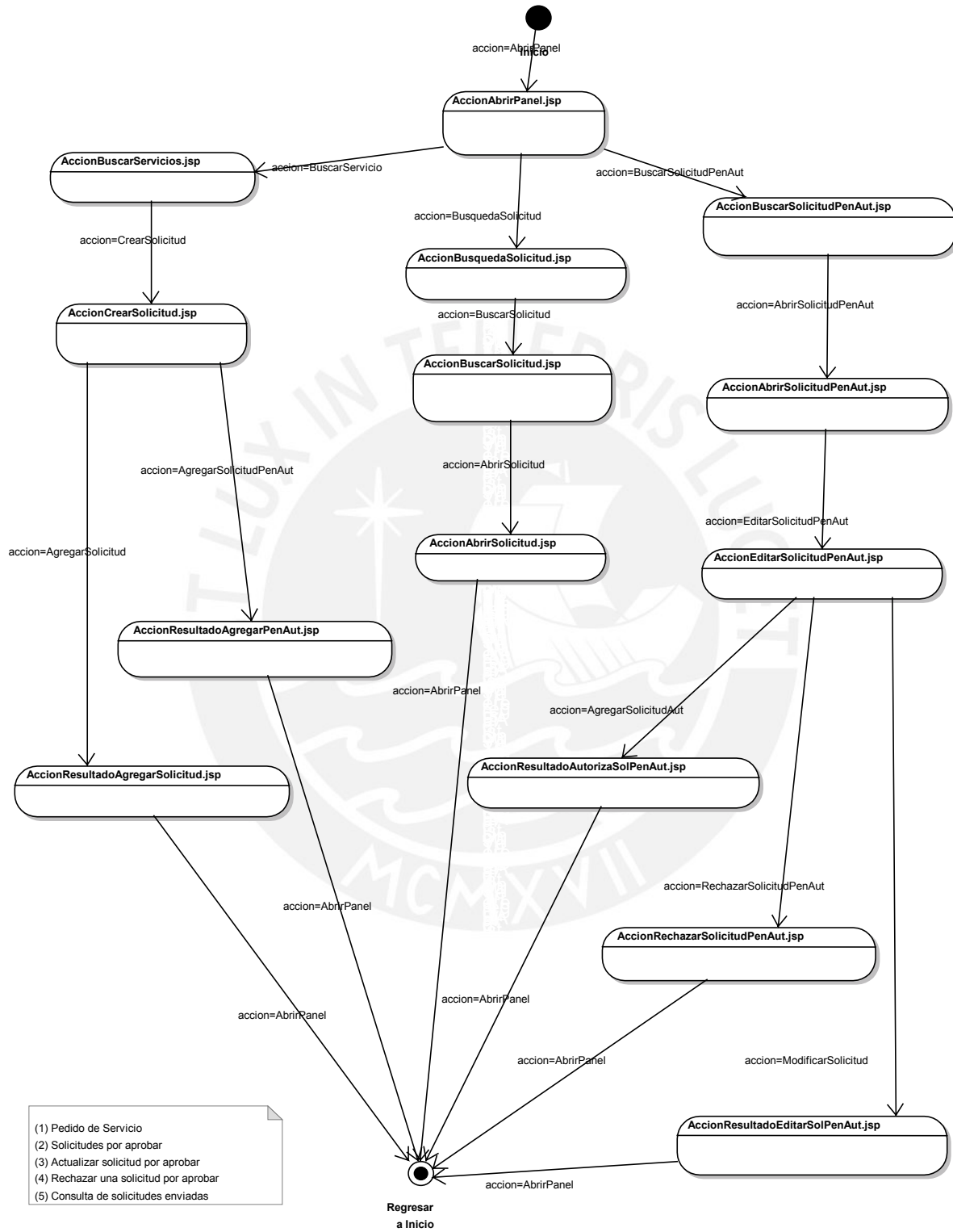
El nodo de fin es usado al final de las transiciones de estado.



El diagrama de estado nos va a servir para determinar qué acciones o eventos manejan una aplicación, cual es el alcance de la misma y como se relaciona con otras aplicaciones del sistema.

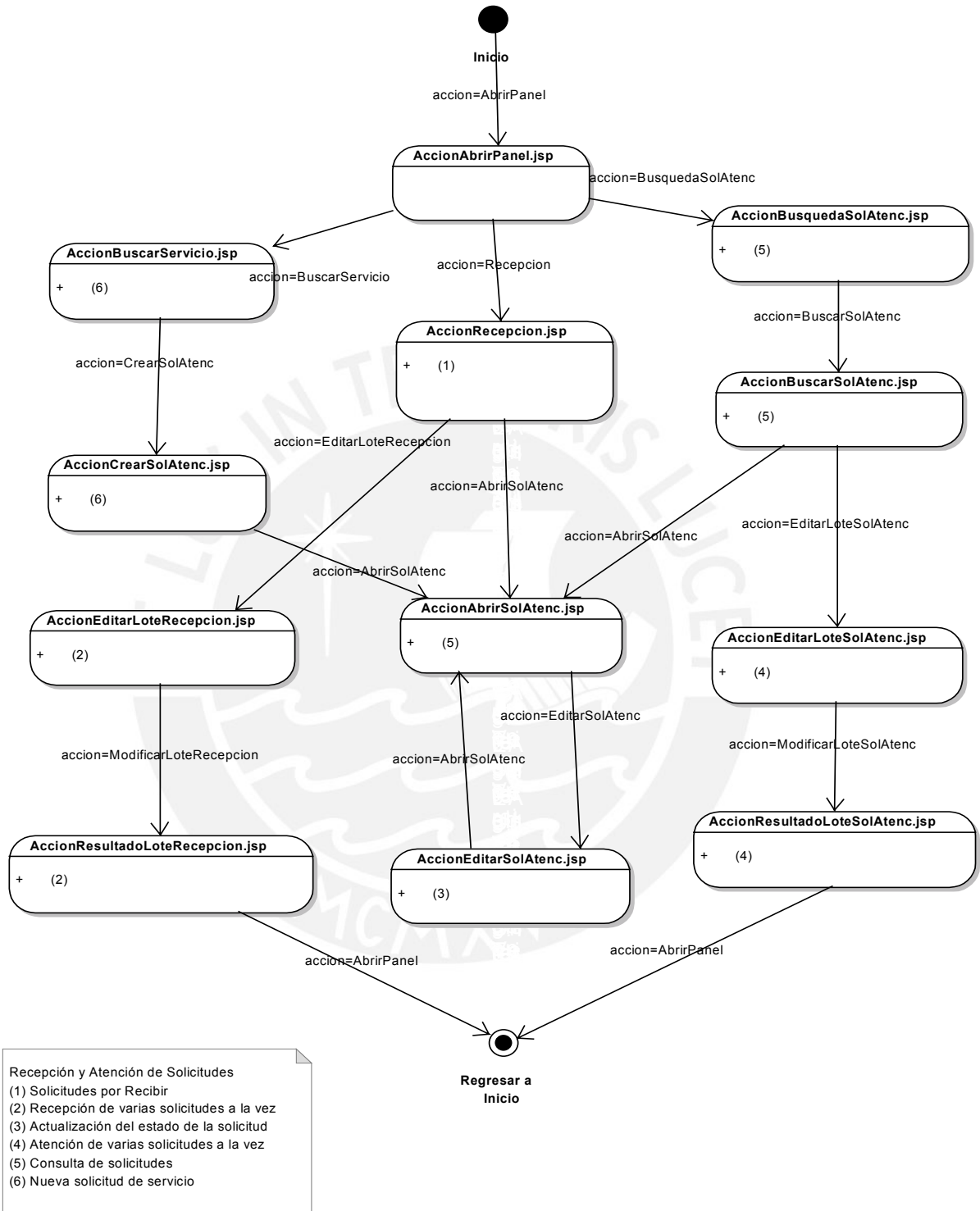
A continuación se muestra los diagramas de estados de los procesos de pedido de servicio y de la atención.

Diagrama de estado - Pedido de Servicio



- (1) Pedido de Servicio
- (2) Solicitudes por aprobar
- (3) Actualizar solicitud por aprobar
- (4) Rechazar una solicitud por aprobar
- (5) Consulta de solicitudes enviadas

Diagrama de Estado - Recepción y Atención de Solicitudes de Servicio



#### 4.4.2. Planteamiento de diagramas de secuencia para procesos del sistema

El diagrama de secuencia<sup>2</sup> es una representación estructurada del comportamiento de los procesos del sistema, como una serie de pasos a través del tiempo.

Es usado para representar el flujo de trabajo, intercambio de mensajes entre los elementos y como éstos se relacionan entre sí para lograr algún resultado.

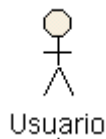
El diagrama de secuencia tiene 2 dimensiones:

- La dimensión vertical que representa el tiempo.
- La dimensión horizontal representa los diferentes objetos.

Cada objeto de la secuencia es dispuesto en una secuencia horizontal, con intercambio de mensajes entre los objetos.

El diagrama de secuencia puede contener los siguientes objetos:

**Actor:** representa al usuario que da inicio al flujo de eventos. Es la persona que interactúa con el sistema



**Boundary:** puede ser usado para representar la pantalla, el browser o el navegador del usuario.

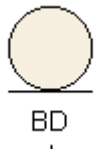


**Control:** representa al controlador del sistema. En este caso representa al servlet del sistema.



<sup>2</sup> Es parte de la metodología UML (Unified Modelling Language) orientado a objetos usado para modelar sistemas.

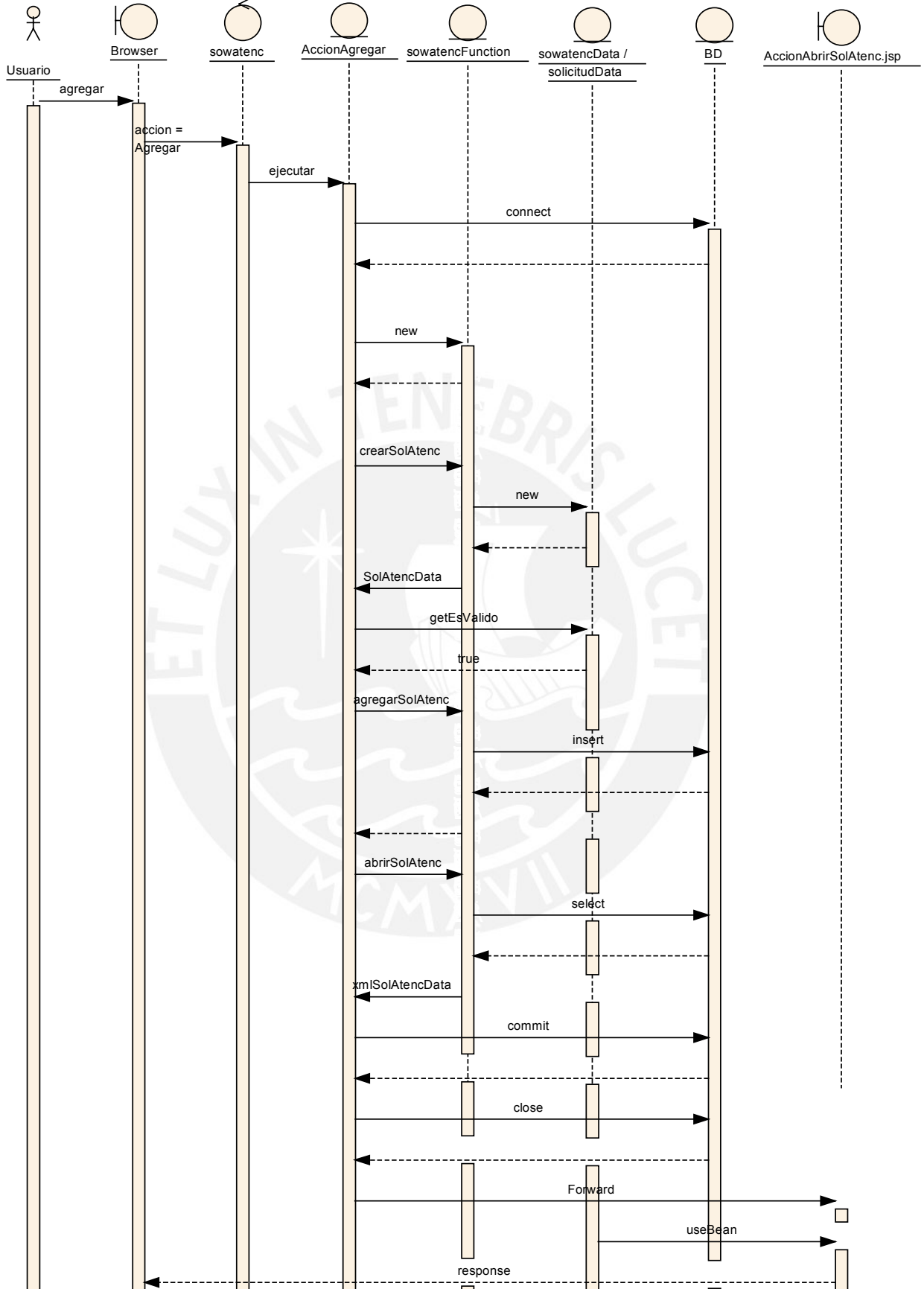
Entity: representa objetos persistentes y puede ser usado para representar la base de datos.



A continuación se muestra el diagrama de secuencia, a manera de ejemplo, del registro de una solicitud de servicio en el sistema:



Diagrama de Secuencia (accion=agregar)

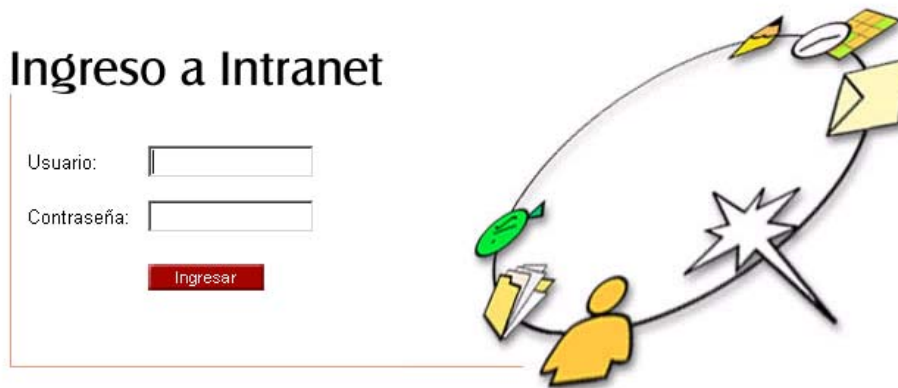


#### 4.5 Control de acceso al Sistema

Es importante para el sistema controlar de alguna manera el acceso a las aplicaciones dependiendo al papel que tendrá cada persona. Cada persona deberá tener un código de usuario y una contraseña para ingresar a la Intranet de la universidad.

##### **Primer Nivel:** Acceso al Intranet

Una vez que el usuario escribe su código (usuario) y contraseña en la siguiente pantalla, accederá al Intranet. Pudiendo luego “navegar” y acceder a información dependiendo de los siguientes niveles que se describen más adelante.



##### **Segundo Nivel:** Acceso a las aplicaciones

En este nivel se identificará al usuario y las aplicaciones que tendrá acceso dependiendo de las funciones y/o actividades que cumpla dentro de la organización.

##### **Tercer Nivel:** Seguridad para solicitar servicios

Según el perfil de cada usuario, el registro de firmas y el tipo de servicio se definen autorizaciones para solicitar servicios de la siguiente manera:

**Tipo de servicio:** Servicio para alumno, servicio para docente, servicio para personal administrativo, servicio para unidades de la universidad, etc.

**Perfil de usuario:** Al ingresar una persona a la Intranet, inmediatamente se identifica qué perfil(es) tiene; es decir, si es alumno, docente, no docente, etc.

**Registro de firmas:** Adicionalmente al perfil del usuario, existe el registro de firmas que es una relación de oficinas de la universidad y las personas que tienen la



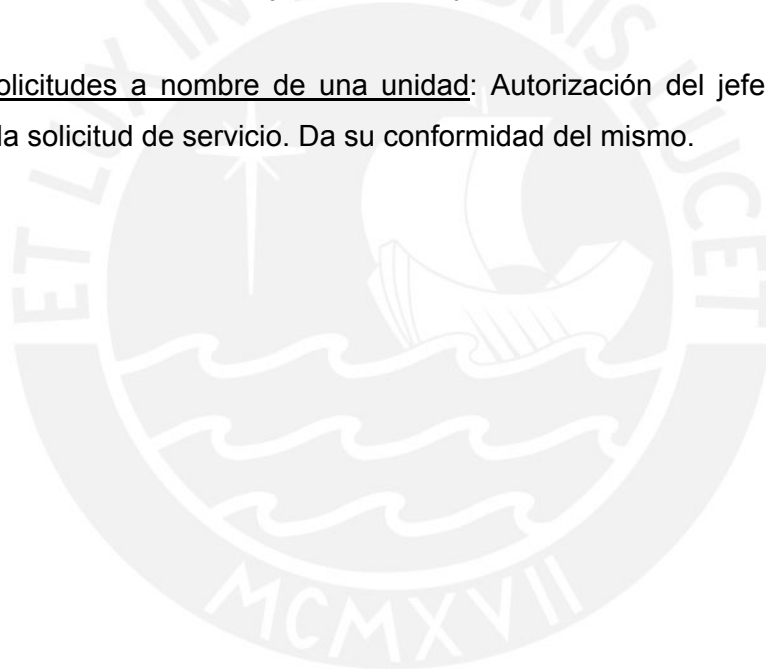
autorización (por el cargo que ocupan) para solicitar y autorizar (V°B° Jefe de Unidad) en cada oficina.

Combinando estas tres definiciones tendremos las siguientes actividades:

Solicitar un servicio o trámite a título personal: Servicios que una persona solicita un servicio para sí mismo (Ej. Trámites de un alumno). Una persona no podrá solicitar un trámite o servicio para otra persona. Asimismo, una persona podrá consultar solamente sus pedidos y no de otros.

Solicitar a nombre de una unidad: Servicios que una persona solicita un servicio para una oficina de la universidad (Ej. Reparación y/o mantenimiento de equipos).

Aprobar solicitudes a nombre de una unidad: Autorización del jefe de la unidad u oficina de la solicitud de servicio. Da su conformidad del mismo.



## CAPITULO V. FUNCIONES PARA EL MANEJO DE DOCUMENTOS XML

En el presente capítulo se detalla las funciones que se desarrollaron para procesar documentos XML. Debido a la concepción del sistema, éstas son las funciones más críticas. En la codificación de las funciones se utilizaron los parsers del XML para el lenguaje Java más utilizados en la actualidad. Estos son el DOM y el SAX. Si bien ambos cumplen el mismo objetivo de convertir un documento XML a información relevante, la manera de usarlos es diferente en ambos casos.

### 5.1 Los APIs DOM y SAX

Los APIs son herramientas que ayudan a utilizar un documento XML en un entorno de programación. Existen APIs en Java que cumplen con esta función. Al documento XML se le analizan todos sus datos, se verifica su estructura y recién es manipulado en la aplicación. Existen varios productos en el mercado pero los más comunes son el DOM y el SAX. Ambos cumplen con las recomendaciones del consorcio W3C (World Wide Web Consortium).

El SAX proporciona un marco basado en eventos para analizar datos XML. SAX lee detenidamente todo el documento y lo separa en datos identificables. Dependiendo del tipo de dato, se van activando los eventos. Los métodos o eventos definidos son el `startDocument()`, `startElement()`, `characters()`, `endElement()`, `endDocument()`. Cada evento es programado según las necesidades de cada sistema en particular.

Mientras que el SAX proporciona un medio para acceder a los datos de un documento XML, el DOM está diseñado para proporcionar un medio para manipularlos. Representa el documento en una estructura de tipo árbol. El DOM carga todo el documento en memoria, los datos son almacenados en nodos. Cada nodo representa parte de los datos extraídos del documento. El API del DOM permite acceder y manipular jerárquicamente la información del documento.

## 5.2 Función para convertir un formulario electrónico en código HTML (presentación)

Esta función es utilizada en los eventos donde el usuario llena una solicitud de servicio. A partir del formulario guardado en la base de datos, y a medida que se identifican los nombres de los datos y tipos de dato, se va armando las sentencias HTML respectivas. Estas sentencias se van almacenando en una variable que luego es invocada en el JSP de esta manera:

```
<%=CreaPlantillaSolServi.mostrarPlantillaServicio(formularioXML,solicitudData%>
```

Donde el parámetro formularioXML contiene las características de los datos del formulario para la solicitud de servicio y el parámetro solicitudData contiene los datos que llena el usuario. En el anexo B se muestra el código fuente de la aplicación. Como ejemplo mostraremos la representación en la pantalla del formulario para el traslado de línea telefónica.

Los tipos de datos definidos tenemos:

TIPODATO	DESCRIPCION	LONGITUD	PRECISION
A1	ALFANUMERICO LONG. MAX(10)	10	
A2	ALFANUMERICO LONG. MAX(20)	20	
A3	ALFANUMERICO LONG. MAX(30)	30	
A4	ALFANUMERICO LONG. MAX(40)	40	
A5	ALFANUMERICO LONG. MAX(50)	50	
A6	ALFANUMERICO LONG. MAX(100)	100	
A7	ALFANUMERICO LONG. MAX(256)	256	
N1	NUMERICO LONG. MAX(5)	5	
N2	NUMERICO LONG. MAX(10)	10	
N3	NUMERICO LONG. MAX(10,2)	10	2
FE	FECHA		
HR	HORA		
RF	RANGO DE FECHA		
RH	RANGO DE HORA		
SN	SELECCION SI/NO		
IM	ARCHIVO DE IMAGEN		
TX	ARCHIVO DE TEXTO		
TI	TITULO	20	0
TN	TITULO NEGRITA	20	0

En la base de datos se almacenaría así:

NUMDATO	TIPODATO	TITULO	INDOBLIGATORIO
1	A2	Número de línea directa y/o anexo	1
2	A5	Ubicación Inicial	1
3	A5	Ubicación Final	1
4	A7	Observaciones	1

El código HTML generado por la función sería el siguiente:

```

<table border='0' width='100%' >
<tr>
<td><input type='hidden' name='num' value='1' />
<input type='hidden' name='obligatorio' value='s' />
<input type='hidden' name='tipodato' value='A2' />
<input type='hidden' name='longitud' value='20' />
<input type='hidden' name='precision' value=' ' />
</td><td class='pucpEtiqoblig' width='30%' > Numero de linea directa y/o anexo</td>
<td><input type='hidden' name='nombre' value='Numero de linea directa y/o anexo' />
</td><td class='pucpCampo' width='70%'><input type='hidden' name='dialni' value=' ' />
<input type='hidden' name='mesIni' value=' ' />
<input type='hidden' name='anoIni' value=' ' />
<input type='hidden' name='diaFin' value=' ' />
<input type='hidden' name='mesFin' value=' ' />
<input type='hidden' name='anoFin' value=' ' />
<input type='hidden' name='horaIni' value=' ' />
<input type='hidden' name='minutoIni' value=' ' />
<input type='hidden' name='horaFin' value=' ' />
<input type='hidden' name='minutoFin' value=' ' />
<input type='hidden' name='datoIngresado' value=' ' />
<input type='text' class='pucpCampo' name='valorDato' value='' maxlength = '20' size = '21' /> </td>
</tr>
<tr>
<td><input type='hidden' name='num' value='2' />
<input type='hidden' name='obligatorio' value='s' />
<input type='hidden' name='tipodato' value='A5' />
<input type='hidden' name='longitud' value='50' />
<input type='hidden' name='precision' value=' ' />
</td><td class='pucpEtiqoblig' width='30%' > Ubicacion Inicial</td>
<td><input type='hidden' name='nombre' value='Ubicacion Inicial' />
</td><td class='pucpCampo' width='70%'><input type='hidden' name='dialni' value=' ' />
<input type='hidden' name='mesIni' value=' ' />
<input type='hidden' name='anoIni' value=' ' />
<input type='hidden' name='diaFin' value=' ' />
<input type='hidden' name='mesFin' value=' ' />
<input type='hidden' name='anoFin' value=' ' />
<input type='hidden' name='horaIni' value=' ' />
<input type='hidden' name='minutoIni' value=' ' />
<input type='hidden' name='horaFin' value=' ' />
<input type='hidden' name='minutoFin' value=' ' />
<input type='hidden' name='datoIngresado' value=' ' />
<input type='text' class='pucpCampo' name='valorDato' value='' maxlength = '50' size = '51' /> </td>
</tr>
<tr>
<td><input type='hidden' name='num' value='3' />
<input type='hidden' name='obligatorio' value='s' />
<input type='hidden' name='tipodato' value='A5' />
<input type='hidden' name='longitud' value='50' />
<input type='hidden' name='precision' value=' ' />
</td><td class='pucpEtiqoblig' width='30%' > Ubicacion Final</td>
<td><input type='hidden' name='nombre' value='Ubicacion Final' />
</td><td class='pucpCampo' width='70%'><input type='hidden' name='dialni' value=' ' />
<input type='hidden' name='mesIni' value=' ' />
<input type='hidden' name='anoIni' value=' ' />
<input type='hidden' name='diaFin' value=' ' />
<input type='hidden' name='mesFin' value=' ' />
<input type='hidden' name='anoFin' value=' ' />
<input type='hidden' name='horaIni' value=' ' />
<input type='hidden' name='minutoIni' value=' ' />
<input type='hidden' name='horaFin' value=' ' />
<input type='hidden' name='minutoFin' value=' ' />
<input type='hidden' name='datoIngresado' value=' ' />
<input type='text' class='pucpCampo' name='valorDato' value='' maxlength = '50' size = '51' /> </td>
</tr>
<tr>
<td><input type='hidden' name='num' value='4' />
<input type='hidden' name='obligatorio' value='s' />
<input type='hidden' name='tipodato' value='A7' />
<input type='hidden' name='longitud' value='256' />
<input type='hidden' name='precision' value=' ' />

```

```

</td><td class='pucpEtiqoblig' width='30%' > Observaciones</td>
<td><input type='hidden' name='nombre' value='Observaciones' />
</td><td class='pucpCampo' width='70%'><input type='hidden' name='dialni' value=' ' />
<input type='hidden' name='mesIni' value=' ' />
<input type='hidden' name='anoIni' value=' ' />
<input type='hidden' name='diaFin' value=' ' />
<input type='hidden' name='mesFin' value=' ' />
<input type='hidden' name='anoFin' value=' ' />
<input type='hidden' name='horaIni' value=' ' />
<input type='hidden' name='minutoIni' value=' ' />
<input type='hidden' name='horaFin' value=' ' />
<input type='hidden' name='minutoFin' value=' ' />
<input type='hidden' name='datoIngresado' value=' ' />
<textarea rows='4' name='valorDato' cols='40' maxlength='256' class='pucpCampo' ></textarea> </td>
</tr>
</table>
    
```

Y el formulario obtenido en la pantalla sería el siguiente:

<b>Numero de linea directa y/o anexo</b>	<input type="text"/>
<b>Ubicacion Inicial</b>	<input type="text"/>
<b>Ubicacion Final</b>	<input type="text"/>
<b>Observaciones</b>	<input type="text"/>

### 5.3 Función para convertir los datos de la solicitud en HTML a un documento XML

Esta función es utilizada en los eventos que involucran almacenar una solicitud de servicio en la base de datos. Los datos llenados por el usuario se guardan en una estructura de arreglo de datos (generados con la función anterior descrita) como se muestra a continuación.

```

<table border='0' width= '100%' >
<tr>
<td><input type='hidden' name='num' value='1' />
<input type='hidden' name='obligatorio' value='s' />
<input type='hidden' name='tipodato' value='A2' />
<input type='hidden' name='longitud' value='20' />
<input type='hidden' name='precision' value=' ' />
</td><td class='pucpEtiqoblig' width='30%' > Numero de linea directa y/o anexo</td>
<td><input type='hidden' name='nombre' value='Numero de linea directa y/o anexo' />
</td><td class='pucpCampo' width='70%'><input type='hidden' name='dialni' value=' ' />
<input type='hidden' name='mesIni' value=' ' />
<input type='hidden' name='anoIni' value=' ' />
<input type='hidden' name='diaFin' value=' ' />
<input type='hidden' name='mesFin' value=' ' />
<input type='hidden' name='anoFin' value=' ' />
<input type='hidden' name='horaIni' value=' ' />
<input type='hidden' name='minutoIni' value=' ' />
<input type='hidden' name='horaFin' value=' ' />
<input type='hidden' name='minutoFin' value=' ' />
    
```



```

<input type='hidden' name='datoIngresado' value=' ' />
<input type='text' class='pucpCampo' name='valorDato' value="4614590" maxlength = '20' size = '21' />
</td>
</tr>
<tr>
<td><input type='hidden' name='num' value='2' />
<input type='hidden' name='obligatorio' value='s' />
<input type='hidden' name='tipodato' value='A5' />
<input type='hidden' name='longitud' value='50' />
<input type='hidden' name='precision' value=' ' />
</td><td class='pucpEtiqoblig' width='30%' > Ubicacion Inicial</td>
<td><input type='hidden' name='nombre' value='Ubicacion Inicial' />
</td><td class='pucpCampo' width='70%'><input type='hidden' name='dialNi' value=' ' />
<input type='hidden' name='mesIni' value=' ' />
<input type='hidden' name='anoIni' value=' ' />
<input type='hidden' name='diaFin' value=' ' />
<input type='hidden' name='mesFin' value=' ' />
<input type='hidden' name='anoFin' value=' ' />
<input type='hidden' name='horalNi' value=' ' />
<input type='hidden' name='minutoIni' value=' ' />
<input type='hidden' name='horaFin' value=' ' />
<input type='hidden' name='minutoFin' value=' ' />
<input type='hidden' name='datoIngresado' value=' ' />
<input type='text' class='pucpCampo' name='valorDato' value="Sala de reuniones" maxlength = '50'
size = '51' /> </td>
</tr>
<tr>
<td><input type='hidden' name='num' value='3' />
<input type='hidden' name='obligatorio' value='s' />
<input type='hidden' name='tipodato' value='A5' />
<input type='hidden' name='longitud' value='50' />
<input type='hidden' name='precision' value=' ' />
</td><td class='pucpEtiqoblig' width='30%' > Ubicacion Final</td>
<td><input type='hidden' name='nombre' value='Ubicacion Final' />
</td><td class='pucpCampo' width='70%'><input type='hidden' name='dialNi' value=' ' />
<input type='hidden' name='mesIni' value=' ' />
<input type='hidden' name='anoIni' value=' ' />
<input type='hidden' name='diaFin' value=' ' />
<input type='hidden' name='mesFin' value=' ' />
<input type='hidden' name='anoFin' value=' ' />
<input type='hidden' name='horalNi' value=' ' />
<input type='hidden' name='minutoIni' value=' ' />
<input type='hidden' name='horaFin' value=' ' />
<input type='hidden' name='minutoFin' value=' ' />
<input type='hidden' name='datoIngresado' value=' ' />
<input type='text' class='pucpCampo' name='valorDato' value="Oficina del Director" maxlength = '50'
size = '51' /> </td>
</tr>
<tr>
<td><input type='hidden' name='num' value='4' />
<input type='hidden' name='obligatorio' value='s' />
<input type='hidden' name='tipodato' value='A7' />
<input type='hidden' name='longitud' value='256' />
<input type='hidden' name='precision' value=' ' />
</td><td class='pucpEtiqoblig' width='30%' > Observaciones</td>
<td><input type='hidden' name='nombre' value='Observaciones' />
</td><td class='pucpCampo' width='70%'><input type='hidden' name='dialNi' value=' ' />
<input type='hidden' name='mesIni' value=' ' />
<input type='hidden' name='anoIni' value=' ' />
<input type='hidden' name='diaFin' value=' ' />
<input type='hidden' name='mesFin' value=' ' />
<input type='hidden' name='anoFin' value=' ' />
<input type='hidden' name='horalNi' value=' ' />
<input type='hidden' name='minutoIni' value=' ' />
<input type='hidden' name='horaFin' value=' ' />
<input type='hidden' name='minutoFin' value=' ' />

```

```

<input type='hidden' name='datoIngresado' value=' ' />
<textarea rows='4' name='valorDato' value='Es urgente' cols='40' maxlength='256' class='pucpCampo'
></textarea> </td>
</tr>
</table>

```

A medida que leen los datos se va construyendo el documento XML y se van almacenando en una variable de tipo String. Al finalizar este proceso, la variable de tipo String es validada usando el parser DOM. Esto se realiza para identificar posibles errores al construir la estructura XML. En el anexo C se muestra el código fuente de la aplicación.

Este sería el documento XML resultante de procesar los datos en HTML del ejemplo.

```

<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<INFOADIC>
<ITEM num="1" obligatorio="s" tipodato="A2" longitud="20" precision="0">
<NOMBRE>Numero de linea directa y/o anexo</NOMBRE>
<VALORDATO>4614590</VALORDATO>
</ITEM>
<ITEM num="2" obligatorio="s" tipodato="A5" longitud="50" precision="0">
<NOMBRE>Ubicacion Inicial</NOMBRE>
<VALORDATO>Sala de reuniones</VALORDATO>
</ITEM>
<ITEM num="3" obligatorio="s" tipodato="A5" longitud="50" precision="0">
<NOMBRE>Ubicacion Final</NOMBRE>
<VALORDATO>Oficina del director</VALORDATO>
</ITEM>
<ITEM num="4" obligatorio="s" tipodato="A7" longitud="256" precision="0">
<NOMBRE>Observaciones</NOMBRE>
<VALORDATO>Es urgente</VALORDATO>
</ITEM>
</INFOADIC>

```

#### 5.4 Función para convertir un documento XML de una solicitud de servicio en código HTML (presentación)

Esta función es utilizada en los eventos que involucran la consulta de una solicitud de servicio. Se usó el parser SAX para leer los datos adicionales a la solicitud de servicio. Estos datos fueron guardados en base datos como un documento XML.

A partir del documento XML y a medida que se identifican los tags, atributos y datos, se van armando las sentencias HTML respectivas. Estas sentencias se van almacenando en una variable que luego es invocada en el JSP de esta manera:



```
<% InfoAdicSolServHtml.convierteXmlaHtml( datosSolicitudXML, indicaedicion); %>
<%=InfoAdicSolServHtml.formatoHTML()%>
```

Donde el parámetro *datosSolicitudXML* contiene los datos de la solicitud de servicio en la estructura XML almacenado en la Base de Datos y el parámetro *indicaedicion* es un booleano que sirve para obtener el código HTML sea para edición o para consulta de datos. En el anexo D se muestra el código fuente de la aplicación. El siguiente es un ejemplo de su uso:

Tomando el documento XML del ejemplo almacenado en la base de datos, se generaría el siguiente código HTML:

```
<table border='0' width= '100%' >
<tr><td>
</td>
<td class='pucpEtiqEdicion' width='30%' > Numero de linea directa y/o anexo</td>
<td></td>
<td width='70%' class='pucpCampo'>
4614590</td>
</tr>
<tr><td>
</td>
<td class='pucpEtiqEdicion' width='30%' > Ubicacion Inicial</td>
<td></td>
<td width='70%' class='pucpCampo'>
Sala de reuniones</td>
</tr>
<tr><td>
</td>
<td class='pucpEtiqEdicion' width='30%' > Ubicacion Final</td>
<td></td>
<td width='70%' class='pucpCampo'>
Oficina del director</td>
</tr>
<tr><td>
</td>
<td class='pucpEtiqEdicion' width='30%' > Observaciones</td>
<td></td>
<td width='70%' class='pucpCampo'>
Es urgente</td>
</tr>
</table>
```

Y el formulario que se mostraría en pantalla (browser) según si es para edición o sólo para consulta de datos:

<b>Numero de linea directa y/o anexo</b>	<input type="text"/>
<b>Ubicacion Inicial</b>	<input type="text"/>
<b>Ubicacion Final</b>	<input type="text"/>
<b>Observaciones</b>	<input type="text"/>

Numero de linea directa y/o anexo	4614590
Ubicacion Inicial	Sala de reuniones
Ubicacion Final	Oficina del director
Observaciones	Es urgente



## VI. CONCLUSIONES

Las conclusiones obtenidas durante el desarrollo de la presente tesis están agrupadas en varios rubros que se presentan a continuación:

### Arquitectura MVC

Esta arquitectura permite separar la lógica del negocio de una organización de la presentación de las aplicaciones. De esta manera se evita duplicar funciones y procedimientos y se pueda reutilizar las mismas funciones en varias aplicaciones. Si la lógica estuviera en cada aplicación, aumentaría el costo de mantenimiento.

Otro punto importante, es que presentar los sistemas desarrollados en diferentes plataformas como son Internet, tecnología WAP (teléfonos inalámbricos o celulares), PDAs (Personal Digital Assistant) y otras tendencias futuras. Esta arquitectura permitirá reutilizar código de programas con sólo cambiar la presentación o vista de las aplicaciones, dependiendo de la plataforma, con el consiguiente ahorro de tiempo y costo de programación.

El uso del lenguaje Java ofrece una herramienta más eficiente, fácil de utilizar, más poderosa y más portable que muchas otras tecnologías del tipo CGI.

Es eficiente porque se atiende las diferentes peticiones de los usuarios y las aplicaciones permanecen una sola vez en la memoria, con lo cual se evita recargar la memoria del servidor. Es portable por que las aplicaciones se pueden ejecutar en diferentes servidores web sin problemas.

El uso del Internet en la presente tesis, permite extender el sistema a cualquier parte del mundo. Solamente es necesario tener un navegador o browser y estar conectado a la red. Con esto se reduce el costo del mantenimiento e instalación de software en las computadoras de cada usuario como es en el caso de los sistemas desarrollados bajo la arquitectura cliente – servidor.

### XML

Este concepto facilita la manera de almacenar los datos contenidos cada formulario como un documento XML. Adicionalmente, en cada documento XML se ha guardado no sólo los datos que llena el usuario sino también la misma estructura de cada formulario en sí. Esto permite lo siguiente:

- Sólo se guarda en la base de datos la última versión del formulario. Es decir, si hubiera actualizaciones en el formato del formulario, éstas se verían reflejados en las futuras solicitudes de servicio que realice el solicitante. Más no en las solicitudes hechas con la anterior versión del formulario.
- El documento XML pueda ser enviado o utilizado en otros sistemas desarrollados en plataformas o servidores distintos al utilizado en esta tesis pudiendo ser reproducido en el browser la información contenida en el documento.

El XML no siempre será la solución a utilizar para almacenar la información pero en este caso ha ayudado para desarrollar un sistema genérico y a la vez personalizado a las necesidades de cada unidad de servicio. Asimismo ha simplificado el diseño de base de datos y a su vez ha reducido la codificación de los programas.

Existen otras herramientas que hacen uso de este lenguaje XML como XForms, Xlink, XSL, Voice XML, etc. La presente tesis sólo es un atisbo de lo que puede ofrecer el XML para futuros desarrollos de sistemas de información.

### **Sistema**

En cuanto al sistema desarrollado, permite reducir el tiempo que un solicitante (sea un alumno o una oficina) utiliza para realizar su pedido; hace un seguimiento del pedido tanto por el solicitante como por la unidad de servicio; minimiza el uso de papel y con ello reduciendo los costos de la universidad.

El solicitante no tiene que preocuparse a qué unidad de servicio debe dirigir sus pedidos ni con qué formulario en papel llenar el pedido. Sino simplemente busca en el sistema qué servicio necesita y lo solicita. El sistema automáticamente se encarga de enviar el pedido a la oficina correspondiente (unidad de servicio).

Usando herramientas relativamente nuevas permite desarrollar un sistema que puede ser utilizado por muchas oficinas y a la vez que pueda ser personalizado sin perder su característica de sistema genérico. Se ha evitado desarrollar varios sistemas.

La información almacenada de cada solicitud de servicio puede ser utilizada posteriormente como *input* para otros sistemas que se desarrollen de acuerdo a los requerimientos en cada oficina.

Se empleó los diagramas DFD junto con los diagramas de secuencia en el desarrollo de la tesis. A pesar que algunos autores no lo recomiendan, fue conveniente para este caso.

### **Proceso**

En cuanto al procedimiento que se sigue para solicitar un servicio, ésta ha variado:

#### Para el solicitante:

Ya no es necesario que el conserje lleve la solicitud a la unidad prestadora de servicio (Ahorro de tiempo).

Se elimina en un buen porcentaje el uso de formularios en papel (Ahorro en costos).

Es posible consultar el estado de su solicitud (Gestión administrativa)

#### Para la unidad de servicio

Se reduce el manejo de solicitudes enviadas en papel.

Mejora en la atención y seguimiento de las solicitudes.

A partir de la información almacenada en la base de datos se puede obtener estadísticas que apoyen la gestión administrativa.

### **Interacción con los usuarios**

Para el éxito de la presente tesis, constantemente se mantuvo comunicación con las oficinas (unidades de servicio) para recabar información de sus procesos y servicios que atienden y así poder desarrollar un sistema de acorde a sus necesidades.

Dado que el sistema será utilizado por ellos, se ha pretendido hacer un sistema de fácil uso, intuitivo, amigable, visualmente presentable.

La interacción con los usuarios no termina ahí. Una vez instalado el sistema, se debe tener en cuenta las recomendaciones de ellos, de modo que sirva como retroalimentación para mejorar los procesos y así mejorar la calidad del sistema. Se debe también hacer un seguimiento al sistema, de manera de poder detectar posibles errores y poder solucionarlos en el menor tiempo posible.

### **Implementación en otras instituciones**

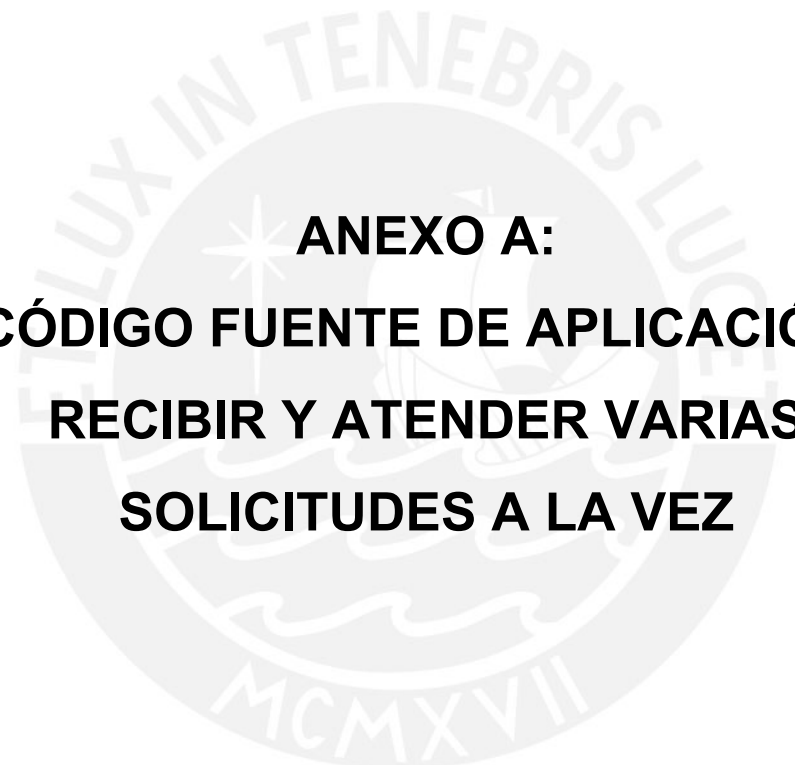
Este sistema puede ser implementado en otras instituciones. Los cambios que se realizarían son menores debido al modelo MVC planteado. Tan sólo se cambiarían los beans de funciones que obtienen y verifican los datos de los usuarios y las unidades solicitantes.

### **Consideraciones**

El sistema desarrollado presenta ciertas limitaciones. Al desarrollar un sistema genérico, siempre se busca satisfacer las necesidades de los casos generales. Por tanto, este sistema no va a poder satisfacer el 100% de los formularios en papel pero sí un alto porcentaje.

Se ha buscado cubrir casi el 100% de los requerimientos de cada unidad de servicio. Sin embargo, dada la naturaleza genérica de este sistema, puede haber una pequeña cantidad de necesidades especiales o únicas en cada unidad de servicio que son satisfechas parcialmente o en el peor de los casos, no satisfechas.

A pesar que el solicitante realice su pedido electrónicamente, siempre habrá casos que los pedidos lleguen por otros medios de comunicaciones (teléfono, papel, fax, etc.). Esto puede deberse a: emergencias, olvido de clave de acceso, urgencias, exalumnos sin clave de acceso, etc. Estas solicitudes serán registradas en el sistema por la misma unidad de servicio para su procesamiento.



**ANEXO A:  
CÓDIGO FUENTE DE APLICACIÓN:  
RECIBIR Y ATENDER VARIAS  
SOLICITUDES A LA VEZ**



Bean de acciones AccionModificarLoteRecepcion.java

```
// PUCP Copyright (c) 2001 PUCP DIRINFO
package pucp.solservi.sowatenc;

// Importacion de las clases a usar:
import pucp.solservi.beans.SolAtencBusquedaBeanData;
import pucp.solservi.beans.SolAtencCriteriosBeanData;
import pucp.solservi.beans.SowatencBeanFunction;
import pucp.solservi.beans.SowatencEdicionBeanData;
import pucp.solservi.beans.LibDatosBeanFunction;
import pucp.solservi.beans.SolserviBeanFunction;
import pucp.solservi.beans.SowatencListaBeanData;
import pucp.solservi.util.Xml;
import pucp.lib.PucpAccion;
import pucp.lib.PucpAplicacion;
import pucp.lib.PucpUsuarioVariables;
import pucp.lib.PucpConstant;
import pucp.lib.PucpSession;
import pucp.lib.PucpAplicacionVariables;
import java.util.HashMap;
import java.sql.Connection;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import pucp.solservi.beans.SowatencLoteBeanData;
import oracle.xml.parser.v2.XMLDocument;
import oracle.xml.parser.v2.XMLElement;
import org.w3c.dom.NodeList;
import org.w3c.dom.Element;
import pucp.solservi.util.FormatoDato;
import org.w3c.dom.Document;
import java.sql.SQLException;
// Exceptions:

import pucp.lib.exception.PucpException;
import pucp.lib.exception.PucpExceptionServicio;
import java.io.IOException;
import javax.servlet.ServletException;

/**
 * Actualiza el estado de las solicitudes recibidas
 * @author Victoria Leon
 */
public final class AccionModificarLoteRecepcion extends PucpAccion {

    public void ejecutar (ServletContext sc,
                        HttpServletRequest request,
                        HttpServletResponse response)
        throws Exception {

        super.ejecutar(sc, request, response);

        Connection connection = null;

        try {

            PucpSession.getVerificar(sc, request, response, 1);
            PucpAplicacion.getVerificarServicio(sc, request, response, 1, "sowatenc", PucpConstant.SER_INGRESAR);
            PucpAplicacion.getVerificarServicio(sc, request, response, 1, "sowatenc", PucpConstant.SER_MODIFICAR);

            // obtengo los valores la solicitudes seleccionadas para recibir y atender

            // total de solicitudes para procesar
            int ntotalSolicitudes = FormatoDato.numero(request.getParameter("totalSolicitudes"),0);

            // lista de solicitudes para procesar
            String cadenaSolServ = request.getParameter("rangoNumSolServ");

            boolean bError = false;
            int nintentosActualizacion = 0;
            boolean bintentoExitoso = false;
            XMLDocument xmlModificarLoteData = null;
```

```

boolean bExito = false;

// conexión a base de datos
connection = this.getConnection("bdatos","proyecto","contrasena");
// inicialización de bean de datos y bean de funciones
SowatencBeanFuncion sowatencFuncion = new SowatencBeanFuncion();
sowatencFuncion.setCon(connection);
SowatencLoteBeanData solAtencLoteData = sowatencFuncion.crearSolAtencLoteData(request,cadenaSolServ,
ntotalSolicitudes);
LibDatosBeanFuncion libDatosFuncion = new LibDatosBeanFuncion();

// verificamos los datos y si el usuario pertenece a este módulo de atención
if (solAtencLoteData.getEsValido())
{int nesUsuarioModuloUnidad =
sowatencFuncion.validaPersxModxUnidadServicio(solAtencLoteData.getCodUnidadServicio(),"01", spucUsuario);
if ((nesUsuarioModuloUnidad) != 2)
{ bError = true;
solAtencLoteData.setMsgEstado("Ocurrió un error al procesar los datos:");
solAtencLoteData.agregarMsgError("Ud. no tiene autorización para procesar estas solicitudes.");
}
else
bError=false;
}

// si todo ok, grabo
try {
while ((!bError) && (!bintentoExitoso)) {
bError = false; // inicializo cada vez que intento grabar.
// Intento varias veces grabar los datos

// lock a registros, devuelve documento XML con información de la base de datos
String stringXmlAbrirLoteData = sowatencFuncion.abrirLoteSolicitudAtencion(cadenaSolServ,
solAtencLoteData.getCodUnidadServicio(),
solAtencLoteData.getAnoactb(), solAtencLoteData.getTipoSolic(), true );

// modifico y verifico que todas las solicitudes se han actualizados
bExito = sowatencFuncion.modificarLoteSolServAtenc(solAtencLoteData,
Xml.parseString(stringXmlAbrirLoteData));
if (!bExito)
{ bError = true;}
else
{ bError = false;
bintentoExitoso = true; }
}
if (!bError) bintentoExitoso = true;
if (bError==false) bintentoExitoso = true;
else bintentoExitoso = false;

} catch (SQLException sqlExc) { // errores pk, fk, lock
connection.rollback();
nintentosActualizacion++;
if (nintentosActualizacion >= FormatoDato.MAX_INTENTOS_GRABAR)
{solAtencLoteData.setMsgEstado("Ocurrió un error al grabar los datos:");
solAtencLoteData.agregarMsgError("Ocurrió un error en BD");
solAtencLoteData.agregarMsgError(sqlExc.getLocalizedMessage());
solAtencLoteData.agregarMsgError(sqlExc.getMessage());
solAtencLoteData.agregarMsgError(sqlExc.getSQLState());
solAtencLoteData.agregarMsgError(sqlExc.toString());
throw sqlExc; //tiene el texto del error: ORA-00001:unique constraint.....
}
}
String urlDestino;

if (bError) // si hay error en el proceso
{ urlDestino="/pucp/solservi/sowatenc/jsp/AccionEditarLoteRecepcion.jsp";
XMLDocument xmlBuscarLoteData =
sowatencFuncion.buscarSolAtenc(solAtencLoteData.getCodUnidadServicio(),
solAtencLoteData.getAnoactb(), solAtencLoteData.getCodModuloAct(),
solAtencLoteData.getRangoNumsolservi() );
libDatosFuncion.setCon(connection); // se requiere el connection para buscar combos
String[][] comboModulo = libDatosFuncion.buscarModuloAtencion("A",
solAtencLoteData.getCodUnidadServicio(),"", "");
connection.rollback();
request.setAttribute("solAtencLoteData", solAtencLoteData);

```



```

    response.addDateHeader ("Expires", -1); //prevents caching at the proxy server
  %>
</head>
<body>
<form action="<%=response.encodeURL("sowatenc")%">" name="documento" method="post">
<%= FormatosComunes.menuSowatencPagina() %>
<br>

<!-- títulos de la pantalla --%>
<table class="pucpTablaTitulo">
<tr>
<%= FormatosComunes.tituloPagina("Recepción de solicitudes de servicio")%>
  <td><a href="javascript:document.documento.submit()">
  </a></td>
  <td><a href="javascript:SolicitudPorAtender()"></a></td>
</tr>
</table>
<p class="pucpInfReg">
<%=solAtencLoteData.getNombreUnidadServicio() %> - <%=solAtencLoteData.getDescriModuloAct()%>
</p>

<!-- muestro errores --%>
<table border="0" width="100%">
<%=PucpLenguaje.toHTMLErrores(solAtencLoteData) %>
</table>

<!-- opciones de proceso: Recibir-atender o rechazar pedidos --%>
<table class="pucpTablaSubTitulo" width="100%">
<tr><td width="214" class="pucpSubTitulo">Datos a asignar a las solicitudes</td></tr>
</table>
<table border="0" width="100%" >
<tr>
<td width="10%">
<% if (solAtencLoteData.getCodEstadoServ().equals("P"))|solAtencLoteData.getCodEstadoServ().equals("") { %>
  <input class="pucpCampo" type="radio" name="codEstadoServ" value="P" checked>
<% }else{ %>
  <input class="pucpCampo" type="radio" name="codEstadoServ" value="P">
<% } %> <font class="pucpEtiqOblig">Atender</font> </td>
<td width="15%" class="pucpEtiqOblig" width="124">y enviar a módulo:</td>
<td width="80%" class="pucpCampo">
<%=PucpLenguaje.toHTML(comboModulo,"codModuloSig","class="PucpCampo",solAtencLoteData.getCodModuloSi
g())%></td>
</tr>
<tr>
<td width="10%"> <% if (solAtencLoteData.getCodEstadoServ().equals("X")) { %>
  <input class="pucpCampo" type="radio" name="codEstadoServ" value="X" checked>
<% }else{ %>
  <input class="pucpCampo" type="radio" name="codEstadoServ" value="X">
<% } %> <font class="pucpEtiqOblig">Rechazar</font> </td>
<td width="15%"></td>
<td width="80%"></td>
</tr>
<tr>
<td width="15%" colspan="2" class="pucpEtiq">Información adicional</td>
<td width="80%"
class="pucpCampo"><%=FormatosComunes.datoPagina(1,solAtencLoteData.getObservacion(),"observacion",true)%
></td>
</tr>
</table>

<!-- lista de las solicitudes que se procesarán --%>
<% NodeList nodeLista = xmlBuscarLoteData.selectNodes("/ROWSET/ROW");
  int totalFilas = nodeLista.getLength(); %>

<table class="pucpTablaSubTitulo">
<tr>
<td class="pucpSubTitulo">Solicitudes que serán procesadas: <%= totalFilas%></td>
</tr>
</table>
<table border="0" cellPadding="1" cellSpacing="2" width="100%" >
<tr>
<td align="right" width="2%"></td>

```

```

<td class="pucpCeldaTitulo" width="14%">Número de solicitud</td>
<td class="pucpCeldaTitulo" width="13%">Fecha y hora</td>
<td class="pucpCeldaTitulo" width="35%">Persona/Unidad que solicitó</td>
<td class="pucpCeldaTitulo" width="23%">Servicio solicitado</td>
<td class="pucpCeldaTitulo" width="13%">Estado</td>
</tr>
<!-- acá comienzo a listar mi XML -->
<tr>
<% for (int i=0 ; i<totalFilas ; i++ ) { %>
  <% if ((i%2) == 0)
    { %> <tr class="pucpCelda">
  <% }else{%>
    <tr class="pucpCeldaGris">
  <% } %>
  <td class="pucpCampo"><%=i+1%></td>
  <td class="pucpCampo"><%=((XMLElement)nodeLista.item(i)).valueOf("DOCUMENTO").trim()%></td>
  <td class="pucpCampo"><%=((XMLElement)nodeLista.item(i)).valueOf("FECHAHORA")%></td>
  <td class="pucpCampo"><%=((XMLElement)nodeLista.item(i)).valueOf("SOLICITANTE")%></td>
  <td class="pucpCampo"><%=((XMLElement)nodeLista.item(i)).valueOf("SERVICIO")%> </td>
  <td class="pucpCampo"><%=((XMLElement)nodeLista.item(i)).valueOf("ESTADO")%> </td>
  </tr>
  <% } %>
</table>
<table class="pucpTablaTitulo">
<tr>
  <td width ="45%"></td>
  <td><a href="javascript:document.documento.submit()">
  </a></td>
  <td><a href="javascript:SolicitudPorAtender()"></a></td>
  <td width ="45%"></td>
</tr>
</table>
<!-- campos ocultos --%>
<input type="hidden" name="accion" value="ModificarLoteRecepcion"/>
<input type="hidden" name="orden" value="GrabarProcesar"/>
<%=FormatosComunes.datoPagina(2,solAtencLoteData.getAnoctb(),"anoctb",true)%>
<%=FormatosComunes.datoPagina(2,solAtencLoteData.getCodUnidadServicio(),"codUnidadServicio",true)%>
<%=FormatosComunes.datoPagina(2,solAtencLoteData.getTipoSolic(),"tipoSolic",true)%>
<%=FormatosComunes.datoPagina(2,String.valueOf(solAtencLoteData.getTotalSolicitudes()),"totalSolicitudes",true)
%>
<%=FormatosComunes.datoPagina(2,solAtencLoteData.getRangoNumsolserv(),"rangoNumSolServ",true)%>
<%=FormatosComunes.datoPagina(2,String.valueOf(solAtencLoteData.getTotalSolicitudes()),"totalSolicitudes",true)
%>
<%=FormatosComunes.datoPagina(2,"0001","codModuloAct",true)%>
</form>
<jsp:include page="/pucp/lib/jsp/pucppie.jsp" flush="true"/>
</body>
<head>
</head>
</html>

```





**ANEXO B:**  
**CÓDIGO FUENTE DE FUNCION:**

Función para convertir un formulario electrónico en código HTML

Bean de funciones CreaPlantillaSolServi.java

```
// PUCP Copyright (c) 2001 PUCP DIRINFO
package pucp.solservi.util;

// Importacion de las clases a usar:
import pucp.lib.PucpDBC;
import pucp.lib.PucpBeanFunction;
import pucp.solservi.util.FormatoDato;
import pucp.solservi.util.FormatosComunes;
import pucp.solservi.util.Fecha;
import java.lang.String;
import javax.servlet.http.HttpServletRequest;
import oracle.xml.parser.v2.XMLDocument;
import java.sql.Connection;
// Exceptions:
import pucp.lib.exception.PucpException;

/**
 * Funcion para armar en lenguaje HTML el formulario con datos obtenidos del
 * request (datos ingresados en linea por el usuario)
 * @author Victoria Leon Chan
 */
public final class CreaPlantillaSolServi extends PucpBeanFunction {

public final class CreaPlantillaSolServi extends PucpBeanFunction {

/**
 * Funcion principal
 *
 * @param String stringXmlDatosPlantilla --> es el formulario obtenido de la base de datos
 * como documento XML.
 * @param SowsolicBeanData -> Bean de datos que contiene la información ingresada por el usuario en el browser
 * @return Retorna la cadena StringBuffer con las sentencias HTML
 */

public static StringBuffer mostrarPlantillaServicio(String stringXmlDatosPlantilla
                                                    SowsolicBeanData solicitudData)
throws Exception {
return mostrarPlantillaServicio(stringXmlDatosPlantilla, solicitudData.getNum(),
solicitudData.getNombre(), solicitudData.getDialni(), solicitudData.getMesIni(), solicitudData.getAnolni(),
solicitudData.getDiaFin(), solicitudData.getMesFin(), solicitudData.getAnoFin(), solicitudData.getHoralni(),
solicitudData.getMinutoIni(), solicitudData.getHoraFin(), solicitudData.getMinutoFin(),
solicitudData.getValorDato(), solicitudData.getChkElimina());
}

/**
 * Funcion principal
 *
 * @param String stringXmlDatosPlantilla --> es el formulario obtenido de la base de datos
 * como documento XML.
 * @param String[] num --> identificador del item de datos. tag <item>
 * @param String[] nombre --> titulo del dato.
 * @param String[] dialni --> dato que ingresa el usuario
 * @param String[] mesIni --> dato que ingresa el usuario
 * @param String[] anolni --> dato que ingresa el usuario
 * @param String[] diaFin --> dato que ingresa el usuario
 * @param String[] mesFin --> dato que ingresa el usuario
 * @param String[] anoFin --> dato que ingresa el usuario
 * @param String[] horalnicio --> dato que ingresa el usuario
 * @param String[] minutoinicio --> dato que ingresa el usuario
 * @param String[] horaFin --> dato que ingresa el usuario
 * @param String[] minutoFin --> dato que ingresa el usuario
 * @param String[] valorDato --> dato que ingresa el usuario de tipo numerico/alfanumerico
 * @param String[]
 * @return Retorna la cadena StringBuffer con las sentencias HTML
 */

public static StringBuffer mostrarPlantillaServicio(String stringXmlDatosPlantilla,
                                                    String[] num, String[] nombre,
                                                    String[] dialni, String[] mesIni,
                                                    String[] anolni, String[] diaFin,
                                                    String[] mesFin, String[] anoFin,
```



```

        String[] horaInicio, String[] minutoInicio,
        String[] horaFin, String[] minutoFin,
        String[] valorDato, String[] chkElimina)

throws Exception {
try
{
    String select = null;
    PreparedStatement pstmt = null;
    ResultSet rset = null;
    boolean obligatorio;
    boolean editable;
    int nlongitud=0;
    int nprecision=0;
    String sFechaActual=null;
    String sHoraActual=null;
    String sFechaInicio;
    String sFechaFin;
    String sHoraInicio;
    String sHoraFin;
    String sValorDato;
    String sDatoIngresado;
    String indobligatorio=null;
    int nCuentaArch=0;
    XmlDocument xmlDatosPlantilla=null;

    StringBuffer cadena = new StringBuffer(9000);

    if (stringXmlDatosPlantilla==null || stringXmlDatosPlantilla.length()== 0)
    {return cadena;}
    else
    {
        xmlDatosPlantilla = Xml.parseString( stringXmlDatosPlantilla );
        sFechaActual=" ";
        sHoraActual=" ";

        cadena.append("<table border='0' width= '100%' >" + "\n");

        NodeList nf = xmlDatosPlantilla.selectNodes("/ROWSET/ROW"); // obtengo nodos del XML
        int total = nf.getLength();

        for (int i=0; i<total; i++)
        { // obtengo y guardo los datos de la plantilla

            String numdato= ((XMLElement)nf.item(i)).valueOf("NUMDATO").trim();
            String tipodato= ((XMLElement)nf.item(i)).valueOf("TIPODATO").trim();
            String titulo= ((XMLElement)nf.item(i)).valueOf("TITULO").trim();
            if (((XMLElement)nf.item(i)).valueOf("INDOBLIGATORIO").trim().equals("1"))
            { indobligatorio="s";}
            else
            { indobligatorio="n";}
            //String indobligatorio= ((XMLElement)nf.item(i)).valueOf("INDOBLIGATORIO").trim();
            String longitud= ((XMLElement)nf.item(i)).valueOf("LONGITUD").trim();
            String precision= ((XMLElement)nf.item(i)).valueOf("PRECISION").trim();

            if (longitud==null || longitud.equals(""))
            {longitud=" ";}
            else
            {nlongitud= Integer.parseInt(longitud);}

            if (precision==null || precision.equals(""))
            {precision=" ";}
            else
            {nprecision= Integer.parseInt(precision);}

            if (titulo==null || titulo.equals(""))
            {titulo=" ";}

            if (indobligatorio.equals("s"))
            {obligatorio=true;}
            else
            {obligatorio=false;}

            editable=true; //para formar los campos hidden

```

```

cadena.append("<tr> \n <td>");
cadena.append(FormatosComunes.datoPagina(2,numdato,"num",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2,indobligatorio,"obligatorio",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2,tipodato,"tipodato",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2,longitud,"longitud",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2,precision,"precision",besEditable) + "\n");
cadena.append("</td>");
if (tipodato.equals("TI") || tipodato.equals("TN"))
{cadena.append(FormatosComunes.tituloHTML(titulo,30,besEditable,tipodato)+ "\n");}
else
{cadena.append(FormatosComunes.textoHTML(titulo,30,besEditable,bobligatorio)+ "\n");}
cadena.append("<td>");
cadena.append(FormatosComunes.datoPagina(2,titulo,"nombre",besEditable) + "\n");
cadena.append("</td>");

if (num==null)
{
  sFechaInidato=sFechaActual;
  sFechaFindato=sFechaActual;
  sHoraInidato=sHoraActual;
  sHoraFindato=sHoraActual;
  sValorDatodato="";
  sDatoIngresadodato="";
}
else
{
  if (i>=num.length)
  {sFechaInidato=sFechaActual;
  sFechaFindato=sFechaActual;
  sHoraInidato=sHoraActual;
  sHoraFindato=sHoraActual;
  sValorDatodato="";
  }
  else {
  if (dialNi[i]==null || dialNi[i].trim().equals(""))
  {dialNi[i]=" ";}
  if (mesNi[i]==null || mesNi[i].trim().equals(""))
  {mesNi[i]=" ";}
  if (anoNi[i]==null || anoNi[i].trim().equals(""))
  {anoNi[i]=" ";}
  if (diaFin[i]==null || diaFin[i].trim().equals(""))
  {diaFin[i]=" ";}
  if (mesFin[i]==null || mesFin[i].trim().equals(""))
  {mesFin[i]=" ";}
  if (anoFin[i]==null || anoFin[i].trim().equals(""))
  {anoFin[i]=" ";}
  if (horalNicio[i]==null || horaNicio[i].trim().equals(""))
  {horalNicio[i]=" ";}
  if (minutoInicio[i]==null || minutoInicio[i].trim().equals(""))
  {minutoInicio[i]=" ";}
  if (horaFin[i]==null || horaFin[i].trim().equals(""))
  {horaFin[i]=" ";}
  if (minutoFin[i]==null || minutoFin[i].trim().equals(""))
  {minutoFin[i]=" ";}

  if ( ( dialNi[i]==null || dialNi[i].trim().equals("")) &&
  (mesNi[i]==null || mesNi[i].trim().equals("")) &&
  (anoNi[i]==null || anoNi[i].trim().equals("")) )
  { sFechaInidato=sFechaActual;
  }
  else
  { sFechaInidato = dialNi[i]+ "-" + mesNi[i]+ "-" + anoNi[i];}

  if ( ( diaFin[i]==null || diaFin[i].trim().equals("")) &&
  (mesFin[i]==null || mesFin[i].trim().equals("")) &&
  (anoFin[i]==null || anoFin[i].trim().equals("")) )
  { sFechaFindato=sFechaActual;
  }
  else
  { sFechaFindato = diaFin[i]+ "-" + mesFin[i]+ "-" + anoFin[i];}

  if ( ( horaNicio[i]==null || horaNicio[i].trim().equals("")) &&
  (minutoInicio[i]==null || minutoInicio[i].trim().equals("")) )

```

```

    { sHoraInidato=sHoraActual;
    }
  else
  { sHoraInidato = horaInicio[i]+ ":" + minutoInicio[i]; }

  if ( (horaFin[i]==null || horaFin[i].trim().equals("")) &&
    (minutoFin[i]==null || minutoFin[i].trim().equals("")) )
  { sHoraFindato=sHoraActual;
  }
  else
  { sHoraFindato = horaFin[i]+ ":" + minutoFin[i]; }

  if (valorDato[i]==null || valorDato[i].trim().equals(""))
  { sValorDatodato=" "; }/CAMBIADO
  else
  { sValorDatodato = valorDato[i]; }
}
}

if (tipodato.equals("HR")) // hora
{ cadena.append("<td class='pucpCampo'> " +
FormatosComunes.horaHTML("inicio",sHoraInidato,besEditable) + "</td> \n");
// campos hidden
cadena.append("<td>");
cadena.append(FormatosComunes.datoPagina(2," ","dialNi",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","mesIni",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","anoIni",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","diaFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","mesFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","anoFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","horaFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","minutoFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","datoIngresado",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","valorDato",besEditable) + "\n");
cadena.append("</td> </tr> \n");
}

if (tipodato.equals("FE")) // fecha
{
// campo fecha
cadena.append("<td class='pucpCampo'> " +
FormatosComunes.fechaHTML("inicio",sFechaInidato,besEditable) + "</td> \n");
// campos hidden
cadena.append("<td>");
cadena.append(FormatosComunes.datoPagina(2," ","diaFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","mesFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","anoFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","horaIni",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","minutoIni",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","horaFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","minutoFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","datoIngresado",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","valorDato",besEditable) + "\n");
cadena.append("</td> </tr> \n");
}

if (tipodato.equals("RF")) // rango de fechas
{
cadena.append("<td width='70%' class='pucpCampo'> del " +
FormatosComunes.fechaHTML("inicio",sFechaInidato,besEditable) + "\n");
cadena.append(" al " + FormatosComunes.fechaHTML("fin",sFechaFindato,besEditable) + " \n");
// campos hidden
cadena.append(FormatosComunes.datoPagina(2," ","horaIni",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","minutoIni",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","horaFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","minutoFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","datoIngresado",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","valorDato",besEditable) + "\n");
cadena.append("</td> </tr> \n");
}

if (tipodato.equals("RH")) // rango de horas
{ // campos hidden
cadena.append("<td class='pucpCampo'> ");
cadena.append(FormatosComunes.datoPagina(2," ","dialNi",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","mesIni",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","anoIni",besEditable) + "\n");
}

```

```

cadena.append(FormatosComunes.datoPagina(2," ","diaFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","mesFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","anoFin",besEditable) + "\n");
// tag hora y minutos iniciales
cadena.append(" del " + FormatosComunes.horaHTML("inicio",sHoraInidato,besEditable) + "\n");
// tag hora y minutos finales
cadena.append(" al " + FormatosComunes.horaHTML("fin",sHoraFindato,besEditable) + "\n");
// campo hidden
cadena.append(FormatosComunes.datoPagina(2," ","datoIngresado",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","valorDato",besEditable) + "\n");
cadena.append("</td> </tr> \n");
}
if (tipodato.equals("SN")) // check box (si - no)
{
// campos hidden
cadena.append("<td class='pucpCampo'>");
cadena.append(FormatosComunes.datoPagina(2," ","dialNi",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","mesIni",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","anolNi",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","diaFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","mesFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","anoFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","horalNi",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","minutolNi",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","horaFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","minutoFin",besEditable) + "\n");

// check box tag ----- sDatoIngresadodato
if (sValorDatodato.equals("si"))
sValorDatodato="si";
else
sValorDatodato="no";
cadena.append(FormatosComunes.datoPagina(2,sValorDatodato,"valorDato",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(4,sValorDatodato,"datolIngresado",besEditable) + "\n");
cadena.append("</td> </tr> \n");
}
}
if (tipodato.equals("TI") || tipodato.equals("TN")) // titulo
{
cadena.append("<td> </td>");
cadena.append("<td>");
// campos hidden todas las variables son nulas
cadena.append(FormatosComunes.datoPagina(2," ","dialNi",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","mesIni",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","anolNi",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","diaFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","mesFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","anoFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","horalNi",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","minutolNi",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","horaFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","minutoFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","datoIngresado",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","valorDato",besEditable) + "\n");
cadena.append("</td> </tr> \n");
}
}
if (tipodato.equals("TX") || tipodato.equals("IM")) // archivos de texto o imagenes
{
// campos hidden todas las variables son nulas
cadena.append("<td class='pucpCampo'>");
cadena.append(FormatosComunes.datoPagina(2," ","dialNi",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","mesIni",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","anolNi",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","diaFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","mesFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","anoFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","horalNi",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","minutolNi",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","horaFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","minutoFin",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2," ","datoIngresado",besEditable) + "\n");
cadena.append(FormatosComunes.datoPagina(2,sValorDatodato,"valorDato",besEditable) + "\n");
// campo archivo
if (!(sValorDatodato==null) && !(sValorDatodato.trim()==""))
{ cadena.append("<font class='pucpCampo'>" + sValorDatodato + "</font> \n");
}
}
}

```

```

    }
    cadena.append("<font class='pucpCampo'>" + FormatosComunes.archivoHTML(sValorDatodato,
"archivo"+String.valueOf(nCuentaArch), besEditable) + "</font> \n");

    if (!(sValorDatodato==null) && !(sValorDatodato.trim()=="") && (indobligatorio.equals("s")))
    {
    cadena.append("<font class='pucpCampo'>eliminar</font> \n");
    if (chkElimina!=null)
    { int k=0;
    while (k < chkElimina.length)
    { if (chkElimina[k].equalsIgnoreCase(String.valueOf(nCuentaArch)))
    {cadena.append("<font class='pucpCampo'><input type='checkbox' class='pucpCampo'
name='chkElimina' value='"+String.valueOf(nCuentaArch)+"' CHECKED /></font> \n");}
    k=k+1;
    }
    }
    else
    {cadena.append("<font class='pucpCampo'><input type='checkbox' class='pucpCampo'
name='chkElimina' value='"+String.valueOf(nCuentaArch)+"'></font> \n");}
    }
    cadena.append("</td> </tr>" + "\n");

    nCuentaArch=nCuentaArch+1;
  }
  // caso de otros tipos de datos (alfanuméricos, numéricos
  // campos hidden
  else
  { cadena.append("<td class='pucpCampo' width='70%'>");
  cadena.append(FormatosComunes.datoPagina(2," ","diaIni",besEditable) + "\n");
  cadena.append(FormatosComunes.datoPagina(2," ","mesIni",besEditable) + "\n");
  cadena.append(FormatosComunes.datoPagina(2," ","anoIni",besEditable) + "\n");
  cadena.append(FormatosComunes.datoPagina(2," ","diaFin",besEditable) + "\n");
  cadena.append(FormatosComunes.datoPagina(2," ","mesFin",besEditable) + "\n");
  cadena.append(FormatosComunes.datoPagina(2," ","anoFin",besEditable) + "\n");
  cadena.append(FormatosComunes.datoPagina(2," ","horalni",besEditable) + "\n");
  cadena.append(FormatosComunes.datoPagina(2," ","minutolni",besEditable) + "\n");
  cadena.append(FormatosComunes.datoPagina(2," ","horaFin",besEditable) + "\n");
  cadena.append(FormatosComunes.datoPagina(2," ","minutoFin",besEditable) + "\n");
  cadena.append(FormatosComunes.datoPagina(2," ","datolngresado",besEditable) + "\n");
  if (nlongitud <= 50)

  cadena.append(FormatosComunes.datoPagina(1,sValorDatodato,"valorDato",nlongitud+1,nlongitud,0,besEditable) +
"</td> \n </tr> \n");
    else

  cadena.append(FormatosComunes.datoPagina(3,sValorDatodato,"valorDato",40,nlongitud,4,besEditable) + "</td> \n
</tr> \n");

  }
  }
  cadena.append("</table>");
  }
  return cadena;
} catch (Exception exc){
  throw exc;
}
}
}
}

```

#### Bean de funciones para presentación de datos en HTML.

```

// Copyright (c) 2001 Direccion de Informatica PUCP
package pucp.solservi.util;

```

```

import pucp.lib.componentes.PucpListaVectorFecha;
import pucp.lib.util.PucpLenguaje;

```

```

/**
 * Clase que contiene los formatos para titulos y otros
 * <P> Clase que contiene los formatos para titulos y otros, utilizados en
 * <P> las páginas del proyecto.
 * @author Victoria Leon
 */

```



```

public class FormatosComunes
{
/**
 * Formato en HTML del título de la página
 * <P> Formato en HTML del título de la página
 * @param String Título de la página
 * @return String título de la página en lenguaje HTML
 * @author Victoria Leon
 */
public static String tituloPagina (String ptitulo)
{
return "<td class='pucpTitulo' width='100%'> \n" +
ptitulo + "</td> \n";
}

/**
 * Formato en HTML del subtítulo de la página
 * <P> Formato en HTML del subtítulo de la página
 * @param String subTítulo de la página
 * @return String subTítulo de la página en lenguaje HTML
 * @author Victoria Leon
 */
public static String tituloAdicPagina (String psubTitulo)
{
return "<p class='pucpInfReg'> " + psubTitulo + "</p> \n";
}

/**
 * Formato en HTML del subtítulo
 * <P> Formato en HTML del subtítulo de la página
 * @param String subtítulo de página
 * @return String subtítulo página en lenguaje HTML
 * @author Victoria Leon
 */

public static String subTituloPagina (String psubtitulo)
{
return "<p><table class='pucpTablaSubTitulo'> \n" +
" <tr> \n"+
" <td class='pucpSubTitulo'>" + psubtitulo + "</td> \n" +
" </tr>\n"+
"</table></p> \n";
}

/**
 * Formato en HTML de los links al pie de la página para app Sowatenc
 * <P> Formato en HTML de los links al pie de la página para app Sowatenc
 * @param String sesión
 * @return String de los links de la página en lenguaje HTML
 * @author Victoria Leon
 */

public static String linkSowatencPagina ()
{
return "<br> \n" +
"<p class='pucpLinkMenu' align=center> \n" +
"<a href='javascript:CrearSolicitud();'>Creación de solicitud</a> | \n" +
"<a href='javascript:SolicitudPorAtender();'>Recepción de solicitudes</a> | \n" +
"<a href='javascript:BusquedaSolicitud();'>Atención de solicitudes</a> \n" +
"</p> \n" ;
}

/**
 * Formato en HTML de los tags de input según condición
 * <P> Formato en HTML de los tags de input según condición
 * @param int tipo de dato - 1:text
 * @param String valor del dato
 * @param String nombre del tag
 * @param boolean indicaeditable '1' editable '0' no editable
 * @return String del campo en HTML
 * @author Victoria Leon

```

```

*/

public static String datoPagina (int ptipoDato, String pvalor,
    String pnombreTag, boolean indicaeditable)
{ if (ptipoDato == 1)
    return datoPagina(ptipoDato, pvalor, pnombreTag, 30, 45, 4, indicaeditable);
  else
    return datoPagina(ptipoDato, pvalor, pnombreTag, 45, 254, 4, indicaeditable); }

/**
 * Formato en HTML de los tags de input según condición
 * <P> Formato en HTML de los tags de input según condición
 * @param int tipo de dato - 1:text
 * @param String valor del dato
 * @param String nombre del tag
 * @param int tamaño del tag
 * @param int numero de filas en el text area.
 * @param int longitud max del valor del dato
 * @param boolean indicaeditable '1' editable '0' no editable
 * @return String del campo en HTML
 * @author Victoria Leon
 */

public static String datoPagina (int ptipoDato, String pvalor,
    String pnombreTag, int ptamanoTag, int plongmax, int pnnumFilaTextArea, boolean indicaeditable)
{
  String datoHtml = null;
  // int size = plongitud + 1;
  switch (ptipoDato)
  {
    case 1: // input tag
      if (indicaeditable)
        datoHtml = "<input type='text' class='pucpCampo' name='"+
            pnombreTag+"' value='"+pvalor+"' maxlength = '"+
            plongmax+" size = '"+ptamanoTag+" /> ";
        else
          datoHtml = pvalor;
          break;
      case 2: // hidden tag
        if (indicaeditable)
          datoHtml = "<input type='hidden' name='"+pnombreTag+"' value='"+pvalor+"' /> ";
        else // no devuelvo nada si indicaeditable=false
          datoHtml = "";
          break;
      case 3: // textarea
        if (indicaeditable)
          datoHtml = "<textarea rows='"+pnnumFilaTextArea+"' name='"+pnombreTag+"' cols='"+ptamanoTag+"'
            maxlength='"+plongmax+"' class='pucpCampo' >"+pvalor+"</textarea> ";
          else
            datoHtml = pvalor;
            break;
          case 4: // checkbox
            if (indicaeditable)
              { datoHtml = "<input type='checkbox' class='pucpCampo' name='"+pnombreTag+"' ";
                  if (pvalor.equalsIgnoreCase("si"))
                    datoHtml += "CHECKED";
                  datoHtml += " /> "; }
              else
                datoHtml = pvalor;
                break;
            default: datoHtml = pvalor;
            break;
          }
    }
  return datoHtml;
}

/**
 * Formato en HTML de los tags de input de fecha de inicio/fin
 * <P> Formato en HTML de los tags de input de fecha de inicio
 * @param String indicador ini - fin
 * @param String valor del dato

```



```

* @param boolean indicaEditable '1' editable '0' no editable
* @return String del campo en HTML
* @author Victoria Leon
*/
/*
public static String fechaHTML (String pindFecha, String pvalor, boolean indicaEditable )
{ String datoHTML ="";
  if (indicaEditable)
  {
    String dia = "";
    String mes = "";
    String ano = "";
    if (pindFecha.equals("inicio"))
    { dia = "diaIni";
      mes = "mesIni";
      ano = "anoIni";
    }
    if (pindFecha.equals("fin"))
    { dia = "diaFin";
      mes = "mesFin";
      ano = "anoFin";
    }
  }

  PucpListaVectorFecha comboFecha = new PucpListaVectorFecha();
  comboFecha.borrar();
  comboFecha.poblarDia(); //puebla el combo con datos del tipo dia(01..31)
  comboFecha.insertar(" ", "Dia", 0);
  if (!FormatoDato.esCadenaVacía(pvalor.trim()))
  datoHTML = PucpLenguaje.toHTML(comboFecha.toArrayString(), dia, "class='PucpCampo'", pvalor.substring(0, 2));
  else
  datoHTML = PucpLenguaje.toHTML(comboFecha.toArrayString(), dia, "class='PucpCampo'", "");

  comboFecha.borrar();
  comboFecha.poblarMes(); //puebla el combo con datos del tipo mes(1..12)
  comboFecha.insertar(" ", "Mes", 0);

  if (!FormatoDato.esCadenaVacía(pvalor.trim()))
  datoHTML +=
  PucpLenguaje.toHTML(comboFecha.toArrayString(), mes, "class='PucpCampo'", pvalor.substring(3, 5));
  else
  datoHTML += PucpLenguaje.toHTML(comboFecha.toArrayString(), mes, "class='PucpCampo'", "");

  comboFecha.borrar();
  comboFecha.poblarAnio(2001, 2010); //puebla el combo con datos del tipo ano
  comboFecha.insertar(" ", "Año", 0);
  if (!FormatoDato.esCadenaVacía(pvalor.trim()))
  datoHTML +=
  PucpLenguaje.toHTML(comboFecha.toArrayString(), ano, "class='PucpCampo'", pvalor.substring(6, 10));
  else
  datoHTML += PucpLenguaje.toHTML(comboFecha.toArrayString(), ano, "class='PucpCampo'", "");
  }
  else
  datoHTML = pvalor;
  return datoHTML;
}
*/

public static String fechaHTML (String pindFecha, String pvalor, boolean indicaEditable )
{ String datoHTML ="";
  if (indicaEditable)
  {
    String dia = "";
    String mes = "";
    String ano = "";
    if (pindFecha.equals("inicio"))
    { dia = "diaIni";
      mes = "mesIni";
      ano = "anoIni";
    }
    if (pindFecha.equals("fin"))
    { dia = "diaFin";
      mes = "mesFin";
      ano = "anoFin";
    }
  }
}

```

```

PucpListaVectorFecha comboFecha = new PucpListaVectorFecha();
comboFecha.borrar();
comboFecha.poblarDia(); //puebla el combo con datos del tipo dia(01..31)
comboFecha.insertar(" ","Dia",0);
if (!FormatoDato.esCadenaVacía(pvalor.trim()))
    datoHTML = PucpLenguaje.toHTML(comboFecha.toArrayString(),dia,"class='PucpCampo'",pvalor.substring(0,2));
else
    datoHTML = PucpLenguaje.toHTML(comboFecha.toArrayString(),dia,"class='PucpCampo'", "");

comboFecha.borrar();
comboFecha.poblarMes(); //puebla el combo con datos del tipo mes(1..12)
comboFecha.insertar(" ","Mes",0);

if (!FormatoDato.esCadenaVacía(pvalor.trim()))
    datoHTML +=
PucpLenguaje.toHTML(comboFecha.toArrayString(),mes,"class='PucpCampo'",pvalor.substring(3,5));
else
    datoHTML += PucpLenguaje.toHTML(comboFecha.toArrayString(),mes,"class='PucpCampo'", "");

comboFecha.borrar();
comboFecha.poblarAnio(2001,2010); //puebla el combo con datos del tipo ano
comboFecha.insertar(" ","Año",0);
if (!FormatoDato.esCadenaVacía(pvalor.trim()))
    datoHTML +=
PucpLenguaje.toHTML(comboFecha.toArrayString(),ano,"class='PucpCampo'",pvalor.substring(6,10));
else
    datoHTML += PucpLenguaje.toHTML(comboFecha.toArrayString(),ano,"class='PucpCampo'", "");
}
else
{
    if (pvalor.length() == 10)
    {
        String dia=null;
        String mes=null;
        String ano=null;

        dia=pvalor.substring(0,2);
        if (pvalor.substring(3,5).equals("01"))
            mes="ENE";
        if (pvalor.substring(3,5).equals("02"))
            mes="FEB";
        if (pvalor.substring(3,5).equals("03"))
            mes="MAR";
        if (pvalor.substring(3,5).equals("04"))
            mes="ABR";
        if (pvalor.substring(3,5).equals("05"))
            mes="MAY";
        if (pvalor.substring(3,5).equals("06"))
            mes="JUN";
        if (pvalor.substring(3,5).equals("07"))
            mes="JUL";
        if (pvalor.substring(3,5).equals("08"))
            mes="AGO";
        if (pvalor.substring(3,5).equals("09"))
            mes="SET";
        if (pvalor.substring(3,5).equals("10"))
            mes="OCT";
        if (pvalor.substring(3,5).equals("11"))
            mes="NOV";
        if (pvalor.substring(3,5).equals("12"))
            mes="DIC";
        ano=pvalor.substring(6,10);

        datoHTML = dia + "-" + mes + "-" + ano;

    }
    else
    {
        datoHTML=pvalor;
    }
}
}
return datoHTML;
}

```

```

/**
 * Formato en HTML de los tags de input de hora de inicio/fin
 * <P> Formato en HTML de los tags de input de hora de inicio
 * @param String indicador ini - fin
 * @param String valor del dato
 * @param boolean indicaeditable '1' editable '0' no editable
 * @return String del campo en HTML
 * @author Victoria Leon
 */

public static String horaHTML (String pindHora, String pvalor, boolean indicaeditable )
{ String datoHTML = "";
  if (indicaeditable)
  {
    String hora = "";
    String minuto = "";

    if (pindHora.equals("inicio"))
    { hora = "horaIni";
      minuto = "minutoIni";
    }
    if (pindHora.equals("fin"))
    { hora = "horaFin";
      minuto = "minutoFin";
    }

    PucpListaVectorFecha comboFecha = new PucpListaVectorFecha();
    comboFecha.borrar();
    comboFecha.poblarHora(); //puebla el combo con datos del tipo dia(01..31)
    comboFecha.insertar(" ", "--", 0);
    if (!FormatoDato.esCadenaVacía(pvalor.trim()))
      datoHTML =
PucpLenguaje.toHTML(comboFecha.toArrayString(), hora, "class='PucpCampo'", pvalor.substring(0,2));
    else
      datoHTML = PucpLenguaje.toHTML(comboFecha.toArrayString(), hora, "class='PucpCampo'", "");

    comboFecha.borrar();
    comboFecha.poblarMinuto(); //puebla el combo con datos del tipo dia(01..31)
    comboFecha.insertar(" ", "--", 0);
    if (!FormatoDato.esCadenaVacía(pvalor.trim()))
      datoHTML +=
PucpLenguaje.toHTML(comboFecha.toArrayString(), minuto, "class='PucpCampo'", pvalor.substring(3,5));
    else
      datoHTML += PucpLenguaje.toHTML(comboFecha.toArrayString(), minuto, "class='PucpCampo'", "");
    }
    else
      datoHTML = pvalor;

return datoHTML;
}

/**
 * Formato en HTML del texto según campo editable o no, obligatorio o no
 * <P> Formato en HTML del texto según campo editable o no, obligatorio o no
 * @param String texto
 * @param int ancho del tag en porcentaje
 * @param boolean indicaeditable '1' editable '0' no editable
 * @param boolean indica campo obligatorio
 * @return String del campo en HTML
 * @author Victoria Leon
 */

public static String textoHTML (String ptexto, int pancho, boolean pindicaeditable, boolean pindicaobligatorio)
{ String datoHtml="";
  if (pindicaeditable)
  { if (pindicaobligatorio)
    { datoHtml = "<td class='pucpEtiqoblig' width='"+ pancho + "%' > " + ptexto + "</td>";
    }
    else
      datoHtml = "<td class='pucpEtiq' width='"+ pancho + "%' > " + ptexto + "</td>";
    }
  else
    datoHtml = "<td class='pucpEtiqEdicion' width='"+ pancho + "%' > " + ptexto + "</td>";

return datoHtml;
}

```

```

}

public static String tituloHTML (String ptexto, int pancho, boolean pindicaeditable, String ptipoTitulo)
{ String tituloHtml="";
  if (pindicaeditable)
  { if (ptipoTitulo.equals("TN"))
    //tituloHtml = FormatosComunes.textoHTML(ptexto, pancho, pindicaeditable,true);
    tituloHtml = "<td class='pucpEtiqoblig' width='"+ pancho + "%' colspan='3' > " + ptexto + "</td>";
  else
    //tituloHtml = FormatosComunes.textoHTML(ptexto, pancho, pindicaeditable,false);
    tituloHtml = "<td class='pucpEtiq' width='"+ pancho + "%' colspan='3' > " + ptexto + "</td>";
  }
  else
    tituloHtml = "<td class='pucpInfReg' width='"+ pancho + "%' colspan='3' > " + ptexto + "</td>";//25

  return tituloHtml;
}

/**
 * Formato en HTML de los tags de input para los datos tipo archivo
 * <P> Formato en HTML de los tags de input para los datos tipo archivo
 * @param String valor del dato
 * @param String nombre del tag
 * @param boolean indicaeditable '1' editable '0' no editable
 * @return String del campo en HTML
 * @author Karina Alcócer
 */

public static String archivoHTML (String pvalor, String pnombreTag, boolean indicaeditable)
{
  String datoHtml = null;
  if (indicaeditable)
    datoHtml = "<input type='file' class='pucpCampo' name='"+pnombreTag+"' value=''+pvalor+"' size = '"+40.3+"'
  /> ";
  else datoHtml = pvalor;
  return datoHtml;
}

public static String linkSowsolicPagina ()
{
  return "<br> \n" +
    "<p class='pucpLinkMenu' align=center> \n" +
    "<a href='javascript:BusquedaSolServicio();'>Búsqueda de solicitudes</a> \n" +
    "</p> \n";
}

public static String menuSowsolicPagina ()
{
  return "<table class='pucptablamenu' align='right'>" +
    "<tr>" +
    "<td class='pucpceldamenu'><a class='pucpRefMenu' href='\"javascript:BuscarServicios('1');\">Nueva
solicitud</a></td>" +
    "<td class='pucpceldamenu'><a class='pucpRefMenu' href='javascript:BuscarSolPenAut();'>Solicitudes por
enviar</a></td>" +
    "<td class='pucpceldamenu'><a class='pucpRefMenu' href='javascript:BusquedaSolServicio();'>Solicitudes
enviadas</a></td>" +
    "</tr> \n" +
    "</table> \n";
}

/*Cuando es una b'squeda de una persona que solicitó a título personal*/
public static String menuSowsolicPaginaTitPers ()
{
  return "<table class='pucptablamenu' align='right'>" +
    "<tr>" +
    "<td class='pucpceldamenu'><a class='pucpRefMenu' href='\"javascript:BuscarServicios('0');\">Nueva
solicitud</a></td>" +
    "<td class='pucpceldamenu'><a class='pucpRefMenu' href='javascript:BusquedaSolServicio();'>Solicitudes
enviadas</a></td>" +
    "</tr> \n" +
    "</table> \n";
}
}
}

```



**ANEXO C:**  
**CÓDIGO FUENTE DE FUNCION:**

Función para convertir los datos de la solicitud en HTML a un documento XML

Bean de función InfoAdicSolServXML.java

```
// PUCP Copyright (c) 2001 PUCP DIRINFO
package pucp.solservi.util;

import pucp.solservi.beans.SowsolicBeanData;
import pucp.solservi.util.FormatoDato;
import java.util.FormatoDato;
import java.util.FormatosComunes;
import pucp.lib.util.PucpMultipartRequest;

/**
 * <P> Función que convierte los datos del usuario en documento XML
 * @author Victoria Leon
 */

public class InfoAdicSolServXML
{
/**
 *
 * <P> Función que convierte los datos del usuario en documento XML
 * @param SowsolicBeanData solicitudData --> Bean de datos. Contiene los datos del usuario
 * @return String --> documento XML
 * @author Victoria Leon
 */
public static String generarInfoAdicXML(SowsolicBeanData solicitudData)
throws {
    String[] num = solicitudData.getNum();
    String[] obligatorio = solicitudData.getObligatorio();
    String[] tipodato = solicitudData.getTipodato();
    String[] longitud = solicitudData.getLongitud();
    String[] precision = solicitudData.getPrecision();
    String[] nombre = solicitudData.getNombre();
    String[] dialni = solicitudData.getDialni();
    String[] meslni = solicitudData.getMeslni();
    String[] anolni = solicitudData.getAnolni();
    String[] diaFin = solicitudData.getDiaFin();
    String[] mesFin = solicitudData.getMesFin();
    String[] anoFin = solicitudData.getAnoFin();
    String[] horalni = solicitudData.getHoralni();
    String[] minutolni = solicitudData.getMinutolni();
    String[] horaFin = solicitudData.getHoraFin();
    String[] minutoFin = solicitudData.getMinutoFin();
    String[] valorDato = solicitudData.getValorDato();
    return generarInfoAdicXML(num, obligatorio, tipodato, longitud, precision, nombre, dialni, meslni, anolni,
    diaFin, mesFin, anoFin, horalni, minutolni, horaFin, minutoFin, valorDato );
}

/**
 *
 * <P> Función que convierte los datos del usuario en documento XML
 * * @param String[] num --> identificador del item de datos. tag <item>
 * @param String[] nombre --> titulo del dato.
 * @param String[] dialni --> dato que ingresa el usuario
 * @param String[] meslni --> dato que ingresa el usuario
 * @param String[] anolni --> dato que ingresa el usuario
 * @param String[] diaFin --> dato que ingresa el usuario
 * @param String[] mesFin --> dato que ingresa el usuario
 * @param String[] anoFin --> dato que ingresa el usuario
 * @param String[] horalnicio --> dato que ingresa el usuario
 * @param String[] minutolnicio --> dato que ingresa el usuario
 * @param String[] horaFin --> dato que ingresa el usuario
 * @param String[] minutoFin --> dato que ingresa el usuario
 * @param String[] valorDato --> dato que ingresa el usuario de tipo numerico/alfanumerico
 * @return String --> documento XML
 * @author Victoria Leon
 */
public static String generarInfoAdicXML(String[] num, String[] obligatorio,
    String[] tipodato, String[] longitud, String[] precision,
    String[] nombre, String[] dialni, String[] meslni, String[] anolni,
    String[] diaFin, String[] mesFin, String[] anoFin,
    String[] horalnicio, String[] minutolnicio,
```



```

String[] horaFin, String[] minutoFin, String[] valorDato)
throws {
    int i=0;
    String elemento=null;
    String numValor;
    String obligatorioValor;
    String tipodatoValor;
    String longitudValor;
    String precisionValor;
    String nombreValor;
    String dialNiValor;
    String mesNiValor;
    String anolNiValor;
    String diaFinValor;
    String mesFinValor;
    String anoFinValor;
    String horalnicioValor;
    String minutoInicioValor;
    String horaFinValor;
    String minutoFinValor;
    String valorDatoValor;

    StringBuffer resultbuffer = new StringBuffer();

    // cabecera del XML
    resultbuffer.append("<?xml version='1.0' encoding='ISO-8859-1' standalone='yes'?>\n");

    // raiz del XML
    resultbuffer.append("<INFOADIC>\n");

    while (i < num.length)
    {
        numValor = num[i];
        obligatorioValor = obligatorio[i];
        tipodatoValor = tipodato[i];
        longitudValor = longitud[i];
        if ((longitudValor == null || longitudValor.trim().equals(""))
        {longitudValor="0";}
        precisionValor = precision[i];
        if ((precisionValor == null || precisionValor.trim().equals(""))
        {precisionValor="0";}

        nombreValor = nombre[i];

        dialNiValor = dialNi[i];
        mesNiValor = mesNi[i];
        anoNiValor = anoNi[i];
        diaFinValor = diaFin[i];
        mesFinValor = mesFin[i];
        anoFinValor = anoFin[i];
        horalnicioValor = horalnicio[i];
        minutoInicioValor = minutoInicio[i];
        horaFinValor = horaFin[i];
        minutoFinValor = minutoFin[i];
        valorDatoValor = FormatoDato.convierteCadena(valorDato[i]);

        i = i+1;

        // Se van formando los atributos del nodo en formato XML
        elemento= FormaAtributos(numValor, obligatorioValor, tipodatoValor, longitudValor, precisionValor);
        resultbuffer.append("<ITEM " + elemento);

        elemento=null;

        // Se va formando el título del dato en formato XML
        elemento= FormaElementoNombre(nombreValor);
        if (elemento!=null)
        { resultbuffer.append(elemento);}

        elemento=null;

        // Se va formando el contenido del nodo en formato XML
        elemento=FormaContenidoNodo(tipodatoValor, dialNiValor, mesNiValor, anoNiValor,
        diaFinValor, mesFinValor,anoFinValor,

```



```

        horainicioValor, minutoInicioValor,
        horaFinValor, minutoFinValor, valorDatoValor);
    resultbuffer.append(elemento);
}

resultbuffer.append("</INFOADIC>");
String informAdic=resultbuffer.toString();

return informAdic;
}

/**
 *
 * <P> Función que convierte las características del dato en documento XML
 * @param String num --> Número del dato
 * @param String obligatorio --> si dato es obligatorio o no
 * @param String tipodato --> alfanumérico, numerico, fecha, hora, archivo etc.
 * @param String longi --> longitud máxima del dato
 * @param String precis --> precision
 * @return String --> características del dato en atributos XML
 * @author Victoria Leon
 */

private static String FormaAtributo(String num, String obligatorio,
    String tipodato, String longi,
    String precis)
throws Exception{

    String atributo=null;

    if (!(num == null || num.trim().equals(""))
    { atributo= " num=\""+num+"\""; }

//---- valor obligatorio ( s -- n)

    if (FormatoDato.esCadenaVacía(atributo)
    { atributo= " obligatorio=\""+obligatorio+"\""; }
    else
    { atributo = atributo + " obligatorio=\""+obligatorio+"\""; }

//---- Tipo de dato -- alfanumerico, numerico, fecha, hora, etc.

    if (!(tipodato == null || tipodato.trim().equals(""))
    {
        if (FormatoDato.esCadenaVacía(elemento))
        { atributo= " tipodato=\""+tipodato+"\""; }
        else
        { atributo= atributo + " tipodato=\""+tipodato+"\""; }
    }

//---- longitud máxima que puede tener este dato

    if (!(longi == null || longi.trim().equals(""))
    {if (FormatoDato.esCadenaVacía(atributo))
    { atributo= " longitud=\""+longi+"\""; }
    else
    { atributo= atributo + " longitud=\""+longi+"\""; }
    }

//---- Precisión máxima que puede tener el dato si éste es de tipo numérico

    if (!(precis == null || precis.trim().equals(""))
    {if (FormatoDato.esCadenaVacía(atributo))
    { atributo= " precision=\""+precis+"\""; }
    else
    { atributo= atributo + " precision=\""+precis+"\""; }
    }

    atributo=atributo+">\n";

return atributo;

}

```

```

/**
 *
 * <P> Función que convierte el título del dato en formato XML
 * @param String nombre --> Título del dato
 * @author Victoria Leon
 */

private static String FormaElementoNombre(String nombre )
throws Exception{

String elemento=null;

if (!(nombre == null || nombre.trim().equals(""))
{ elemento= "<NOMBRE>" + nombre + "</NOMBRE>\n"; }

return elemento;
}

/**
 *
 * <P> Función que convierte los datos en nodo XML
 * @param String[] tipoDato --> Tipo de dato
 * @param String[] diaIni --> dato que ingresa el usuario
 * @param String[] mesIni --> dato que ingresa el usuario
 * @param String[] anoIni --> dato que ingresa el usuario
 * @param String[] diaFin --> dato que ingresa el usuario
 * @param String[] mesFin --> dato que ingresa el usuario
 * @param String[] anoFin --> dato que ingresa el usuario
 * @param String[] horalnicio --> dato que ingresa el usuario
 * @param String[] minutolnicio --> dato que ingresa el usuario
 * @param String[] horaFin --> dato que ingresa el usuario
 * @param String[] minutoFin --> dato que ingresa el usuario
 * @param String[] valorDato --> dato que ingresa el usuario de tipo numerico/alfanumerico
 * @return String --> documento XML
 * @author Victoria Leon
 */
private static String FormaContenidoNodo( String tipoDato,
String diaIniValor, String mesIniValor, String anoIniValor,
String diaFinValor, String mesFinValor, String anoFinValor,
String horalnicioValor, String minutolnicioValor,
String horaFinValor, String minutoFinValor,
String valorDatoValor)
throws Exception {

String elemento=null;

//----- Tipo de dato Fecha
if (tipoDato.equals("FE"))
{
if ((diaIniValor == null || diaIniValor.trim().equals("")) &&
(mesIniValor == null || mesIniValor.trim().equals("")) &&
(anoIniValor == null || anoIniValor.trim().equals(""))
{
if (FormatoDato.esCadenaVacía(elemento))
{elemento= "<FECHAINI></FECHAINI>\n";}
else
{elemento= elemento + "<FECHAINI></FECHAINI>\n";}
}
else
{ if (FormatoDato.esCadenaVacía(elemento))
{elemento= "<FECHAINI>" + diaIniValor + "-" + mesIniValor + "-" + anoIniValor+"</FECHAINI>\n";}
else
{elemento= elemento + "<FECHAINI>" + diaIniValor + "-" + mesIniValor + "-" + anoIniValor+"</FECHAINI>\n";}
}

elemento = elemento + "</ITEM>\n";
return elemento;
}

//----- Tipo de dato Rango de fecha

```

```

if (tipoDato.equals("RF"))
{
  if ((diaIniValor == null || diaIniValor.trim().equals("")) &&
      (mesIniValor == null || mesIniValor.trim().equals("")) &&
      (anoIniValor == null || anoIniValor.trim().equals("")))
  {
    if (FormatoDato.esCadenaVacía(elemento))
    {elemento= "<FECHAINI></FECHAINI>\n";}
    else
    {elemento= elemento + "<FECHAINI></FECHAINI>\n";}
  }
  else
  { if (FormatoDato.esCadenaVacía(elemento))
    {elemento= "<FECHAINI>" + diaIniValor + "-" + mesIniValor + "-" + anoIniValor+"</FECHAINI>\n";}
    else
    {elemento= elemento + "<FECHAINI>" + diaIniValor + "-" + mesIniValor + "-" + anoIniValor+"</FECHAINI>\n";}
  }

  if ((diaFinValor == null || diaFinValor.trim().equals("")) &&
      (mesFinValor == null || mesFinValor.trim().equals("")) &&
      (anoFinValor == null || anoFinValor.trim().equals("")))
  {
    if (FormatoDato.esCadenaVacía(elemento))
    {elemento= "<FECHAFIN></FECHAFIN>\n";}
    else
    {elemento= elemento + "<FECHAFIN></FECHAFIN>\n"; }
  }
  else
  {
    if (FormatoDato.esCadenaVacía(elemento))
    {elemento= "<FECHAFIN>" + diaFinValor + "-" + mesFinValor + "-" + anoFinValor+"</FECHAFIN>\n";}
    else
    {elemento= elemento + "<FECHAFIN>" + diaFinValor + "-" + mesFinValor + "-" + anoFinValor+"</FECHAFIN>\n"; }
  }

  elemento = elemento + "</ITEM>\n";
  return elemento;
}

//----- Tipo de dato Hora

if (tipoDato.equals("HR"))
{
  if ((horalInicioValor == null || horalInicioValor.trim().equals("")) &&
      (minutoInicioValor == null || minutoInicioValor.trim().equals("")))
  { if (FormatoDato.esCadenaVacía(elemento))
    { elemento= "<HORAINI></HORAINI>\n"; }
    else
    { elemento= elemento + "<HORAINI></HORAINI>\n";}
  }
  else
  {
    if (FormatoDato.esCadenaVacía(elemento))
    { elemento= "<HORAINI>" + horalInicioValor + ":" + minutoInicioValor + "</HORAINI>\n"; }
    else
    { elemento= elemento + "<HORAINI>" + horalInicioValor + ":" + minutoInicioValor + "</HORAINI>\n";}
  }

  elemento = elemento + "</ITEM>\n";
  return elemento;
}

//----- Tipo de dato rango de hora
if (tipoDato.equals("RH"))
{
  if ((horalInicioValor == null || horalInicioValor.trim().equals("")) &&
      (minutoInicioValor == null || minutoInicioValor.trim().equals("")))
  { if (FormatoDato.esCadenaVacía(elemento))
    { elemento= "<HORAINI></HORAINI>\n"; }
    else
    { elemento= elemento + "<HORAINI></HORAINI>\n";}
  }
  else
  {
  }
}

```

```

if (FormatoDato.esCadenaVacía(elemento))
{ elemento= "<HORAINI>" + horaInicioValor + ":" + minutoInicioValor + "</HORAINI>\n"; }
else
{ elemento= elemento + "<HORAINI>" + horaInicioValor + ":" + minutoInicioValor + "</HORAINI>\n"; }
}

if ((horaFinValor == null || horaFinValor.trim().equals("")) &&
(minutoFinValor == null || minutoFinValor.trim().equals("")))
{
if (FormatoDato.esCadenaVacía(elemento))
{ elemento= "<HORAFIN></HORAFIN>\n"; }
else
{ elemento= elemento + "<HORAFIN></HORAFIN>\n"; }
}
else
{
if (FormatoDato.esCadenaVacía(elemento))
{ elemento= "<HORAFIN>" + horaFinValor + ":" + minutoFinValor + "</HORAFIN>\n"; }
else
{ elemento= elemento + "<HORAFIN>" + horaFinValor + ":" + minutoFinValor + "</HORAFIN>\n"; }
}
elemento = elemento + "</ITEM>\n";
return elemento;
}

//----- Otros tipo de datos
if (valorDatoValor == null || valorDatoValor.trim().equals(""))
{
if (FormatoDato.esCadenaVacía(elemento))
{ elemento= "<VALORDATO></VALORDATO>\n"; }
else
{ elemento= elemento + "<VALORDATO></VALORDATO>\n"; }
}
else
{
if (FormatoDato.esCadenaVacía(elemento))
{ elemento= "<VALORDATO>" + valorDatoValor + "</VALORDATO>\n"; }
else
{ elemento= elemento + "<VALORDATO>" + valorDatoValor + "</VALORDATO>\n"; }
}
elemento = elemento + "</ITEM>\n";
return elemento;
}

```

## ANEXO D: CÓDIGO FUENTE DE FUNCION:

Función para convertir un documento XML de una solicitud de servicio en código  
HTML

Bean de función InfoAdicSolServHtml.java

```
// PUCP Copyright (c) 2001 PUCP DIRINFO
package pucp.solservi.util;

import pucp.lib.PucpDBC;
import pucp.lib.PucpBeanFunction;

import org.xml.sax.Attributes;
import org.xml.sax.AttributeList;
import org.xml.sax.helpers.AttributeListImpl;
import org.xml.sax.helpers.AttributesImpl;

import org.xml.sax.SAXParseException;
import oracle.xml.parser.v2.SAXParser;
import java.io.ByteArrayInputStream;
import org.xml.sax.helpers.DefaultHandler;

import java.net.URL;
import java.net.URLConnection;
import pucp.solservi.util.FormatoDato;
import pucp.solservi.util.FormatosComunes;

/**
 * Parser en SAX utilizado para convertir a formato HTML
 * los datos adicionales del servicio solicitado (formulario con datos del usuario)
 * Reemplaza métodos propios del parser SAX
 * @author Victoria Leon
 */

public final class InfoAdicSolServHtml extends DefaultHandler
{

    static StringBuffer displayStrings = new StringBuffer("");
    static int nlongitud = 0;
    static String nombre = "";
    static String tipoDato = "";
    static String valorDato="";
    static boolean besFilaDato=false;
    static boolean besEditable = false;
    static boolean bobligatorio = false;

    static String sNumdato = "";
    static int numdato=0;
    static int contador=-1;

    /**
     * Devuelve en formato HTML la información adicional del servicio
     * (formulario con datos de la solicitud de servicio del usuario)
     * @return String de la información adicional del servicio en HTML
     * @author Victoria Leon
     */

    public static StringBuffer formatoHTML()
    { return displayStrings;
    }

    /**
     * Devuelve en formato HTML la información adicional del servicio
     * <P> Devuelve en formato HTML la información adicional del servicio
     * @param String pinformAdicional --> datos adicionales en formato XML del servicio solicitado
     * @param boolean pbesEditable --> indica si creamos HTML para edicion o para consulta de datos
     * editable '1' editable '0' no editable
     * @return String formato en HTML de los datos adicionales del servicio solicitado
     * @author Victoria Leon
     */

    public static void convierteXmlaHtml(String pinformAdicional, boolean pbesEditable)
    {
    try {
    displayStrings.setLength(0);

```



```

numdato=0;
contador=-1;
// si no hay datos adicionales, displayString es vacío
if (!FormatoDato.esCadenaVacía(pinformAdicional))
{
    besEditable = pbesEditable;
    // IndentingParserSAX SAXHandler = new IndentingParserSAX();
    InfoAdicSolServHtml saxHandler = new InfoAdicSolServHtml();
    SAXParser parser = new SAXParser();
    parser.setContentHandler(saxHandler);
    parser.setErrorHandler(saxHandler);

    // Open an input stream on the string
    ByteArrayInputStream theStream = new ByteArrayInputStream( pinformAdicional.getBytes() );

    parser.parse(theStream);
}
}
catch (Exception e) {
e.printStackTrace(System.err);
}
}

/**
 * Inicio del documento XML
 * <P> Inicio del documento XML. Acá inicio la estructura en formato HTML
 * @author Victoria Leon
 */

public void startDocument()
{
displayStrings.append("<table border='0' width= '100%' >" + "\n");
}

/**
 * Inicio del elemento
 * <P> Inicio del elemento. Acá muestro nombre valor de los atributos y nombre del elemento
 * @param String uri
 * @param String localName
 * @param String rawName
 * @param Attributes attributes
 * @author Victoria Leon
 */

public void startElement(String uri, String localName, String rawName,
Attributes attributes)
{
    if (rawName.equals("ITEM"))
        besFilaDato = true;

    if (attributes != null) {
        int numberAttributes = attributes.getLength();
        if (numberAttributes>0) {displayStrings.append("<tr><td>");}
        for (int loopIndex = 0; loopIndex < numberAttributes; loopIndex++) {
            if (attributes.getQName(loopIndex)=="num" || attributes.getQName(loopIndex)=="obligatorio" ||
                attributes.getQName(loopIndex)=="tipodato" || attributes.getQName(loopIndex)=="longitud" ||
                attributes.getQName(loopIndex)=="precision" )
            {

displayStrings.append(FormatosComunes.datoPagina(2,attributes.getValue(loopIndex),attributes.getQName(loopIndex),besEditable) + "\n");
            }

            if (attributes.getQName(loopIndex).equals("obligatorio") &&
!FormatoDato.esCadenaVacía(attributes.getValue(loopIndex)))
                if (attributes.getValue(loopIndex).equalsIgnoreCase("s"))
                    bobligatorio = true;
                else
                    bobligatorio = false;
        }
    }
}

```



```

if (attributes.getQName(loopIndex).equals("longitud") &&
!FormatoDato.esCadenaVacía(attributes.getValue(loopIndex)))
    longitud = FormatoDato.numero(attributes.getValue(loopIndex),0);

if (attributes.getQName(loopIndex).equals("tipodato") &&
!FormatoDato.esCadenaVacía(attributes.getValue(loopIndex)))
    tipoDato = attributes.getValue(loopIndex).trim();

}
if (numberAttributes>0) {displayStrings.append("</td>\n");}
}
}

/**
 * Muestro valor del elemento
 * <P> Muestro valor del elemento
 * @param char characters[]
 * @param int length
 * @author Victoria Leon
 */
public void characters(char characters[], int start, int length)
{
String characterData = (new String(characters, start, length)).trim();

// guardo valor del dato en variable valorDato
if(characterData.indexOf("\n") < 0 && characterData.length() > 0) {
    valorDato = characterData;
}
else
{ valorDato = ""; }
}

/**
 * No muestro nada. Función vacía
 * <P> No muestro nada. Función vacía
 * @param char characters[]
 * @param int start
 * @param int length
 * @author Victoria Leon
 */
public void ignorableWhitespace(char characters[], int start, int length)
{

}

/**
 * Fin del elemento
 * <P> Fin del elemento. Acá armo los tags HTML dependiendo de los atributos de cada
 * <P> dato adicional (tipodato, obligatorio, longitud) y si son tags HTML para consulta o para edición
 * @param String uri
 * @param String localName
 * @param String rawName
 * @param Attributes attributes
 * @author Victoria Leon
 */

public void endElement(String uri, String localName, String rawName)
{
if (rawName.equals("ITEM")) besFilaDato = false;
if (besFilaDato) {
if (rawName.equals("NOMBRE"))

{ // textoHTML
if (tipoDato.equals("T1") || tipoDato.equals("TN"))
{displayStrings.append(FormatosComunes.tituloHTML(valorDato,30,besEditable,tipoDato)+ "\n");}//nuevo
else
{displayStrings.append(FormatosComunes.textoHTML(valorDato,30,besEditable,bobligatorio)+ "\n");}

// campos hidden dependiendo del booleano
displayStrings.append("<td>"+FormatosComunes.datoPagina(2,valorDato,"nombre",besEditable) + "</td> \n");

```

```

// IMPORTANTE limpiar variable valorDato espacio en blanco
valorDato = " ";

}
else

{
if (tipoDato.equals("HR")) // hora
{ displayStrings.append("<td width='70%' class='pucpCampo'> " +
FormatosComunes.horaHTML("inicio",valorDato,besEditable) + "\n ");
// campos hidden dependiendo del booleano
displayStrings.append(FormatosComunes.datoPagina(2,"","diaIni",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","mesIni",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","anoIni",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","diaFin",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","mesFin",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","anoFin",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","horaFin",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","minutoFin",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","datoIngresado",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","valorDato",besEditable) + "\n");
displayStrings.append("</td> \n </tr> \n");
// IMPORTANTE limpiar variable valorDato espacio en blanco
valorDato = " ";
return; }

if (tipoDato.equals("FE")) // fecha
{
// campo fecha
displayStrings.append("<td width='70%' class='pucpCampo'> " +
FormatosComunes.fechaHTML("inicio",valorDato,besEditable) + "\n");
// campos hidden dependiendo del booleano
displayStrings.append(FormatosComunes.datoPagina(2,"","diaFin",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","mesFin",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","anoFin",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","horaIni",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","minutoIni",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","horaFin",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","minutoFin",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","datoIngresado",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","valorDato",besEditable) + "\n");
displayStrings.append("</td> \n </tr> \n");
// IMPORTANTE limpiar variable valorDato espacio en blanco
valorDato = " ";

return; }

if (tipoDato.equals("RF")) // rango de fechas
{
if (rawName.equals("FECHAINI")) // fecha inicial
{
displayStrings.append("<td width='70%' class='pucpCampo'> del " +
FormatosComunes.fechaHTML("inicio",valorDato,besEditable) + "\n");

}
if (rawName.equals("FECHAFIN"))
{
displayStrings.append(" al " + FormatosComunes.fechaHTML("fin",valorDato,besEditable) + "\n");

// campos hidden dependiendo del booleano
displayStrings.append(FormatosComunes.datoPagina(2,"","horaIni",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","minutoIni",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","horaFin",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","minutoFin",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","datoIngresado",besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2,"","valorDato",besEditable) + "\n");
displayStrings.append("</td> \n </tr> \n");
}
// IMPORTANTE limpiar variable valorDato espacio en blanco
valorDato = " ";

return; }

if (tipoDato.equals("RH"))
{ if (rawName.equals("HORAINI"))

```

```

// campos hidden dependiendo del booleano
displayStrings.append("<td width=70% class='pucpCampo'>");
displayStrings.append(FormatosComunes.datoPagina(2, "", "diaIni", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "mesIni", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "anoIni", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "diaFin", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "mesFin", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "anoFin", besEditable) + "\n");
// tag hora y minutos iniciales
displayStrings.append(" del " + FormatosComunes.horaHTML("inicio", valorDato, besEditable) + "\n");
}
if (rawName.equals("HORAFIN"))
{
// tag hora y minutos finales
displayStrings.append(" al " + FormatosComunes.horaHTML("fin", valorDato, besEditable) + "\n ");
displayStrings.append(FormatosComunes.datoPagina(2, "", "datoIngresado", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "valorDato", besEditable) + "\n");
displayStrings.append("</td> \n </tr> \n");
}
// IMPORTANTE limpiar variable valorDato espacio en blanco
valorDato = " ";

return; }
if (tipoDato.equals("SN"))
{
// campos hidden
displayStrings.append("<td width=70% class='pucpCampo'>");
displayStrings.append(FormatosComunes.datoPagina(2, "", "diaIni", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "mesIni", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "anoIni", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "diaFin", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "mesFin", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "anoFin", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "horalni", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "minutolni", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "horaFin", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "minutoFin", besEditable) + "\n");
// check box tag
String sValorDatodato="";
if (valorDato.equals("si"))
sValorDatodato="si";
else
sValorDatodato="no";
displayStrings.append(FormatosComunes.datoPagina(4, sValorDatodato, "datoIngresado", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "valorDato", besEditable) + "\n");
displayStrings.append("</td> \n </tr> \n");
// IMPORTANTE limpiar variable valorDato espacio en blanco
valorDato = " ";

return; }
if (tipoDato.equals("TX") || tipoDato.equals("IM")) // titulo
{contador=contador+1;
displayStrings.append("<td>");
displayStrings.append(FormatosComunes.datoPagina(2, "", "diaIni", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "mesIni", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "anoIni", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "diaFin", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "mesFin", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "anoFin", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "horalni", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "minutolni", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "horaFin", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "minutoFin", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, "", "datoIngresado", besEditable) + "\n");
displayStrings.append(FormatosComunes.datoPagina(2, valorDato, "valorDato", besEditable) + "\n");
if (!(besEditable==true) && !(valorDato==null) && !(valorDato.trim().equals("")))
{
numdato=numdato+1;
sNumdato=String.valueOf(numdato);
displayStrings.append("<a class=pucpLinkCelda
href='\"+javascript:ContenidoDocumento(\""+sNumdato+"\"+\">"+valorDato+"</a> \n");
}
else
{if (besEditable==true)

```

```

    { if ((valorDato.trim()!=null) && !(valorDato.trim().equals("")))
      { numdato=numdato+1;
        sNumdato=String.valueOf(numdato);
        displayStrings.append("<a class=pucpLinkCelda
href=\\javascript:ContenidoDocumento(\""+sNumdato+"\\")>"+valorDato+"</a>\\n");
      }

      displayStrings.append("<font class='pucpCampo'>" + FormatosComunes.archivoHTML("",
"archivo"+String.valueOf(contador), besEditable) + "</font> \\n");
      if ((valorDato.trim()!=null) && !(valorDato.trim().equals(""))) && (obligatorio==false)
      {
        displayStrings.append("<font class='pucpCampo'>eliminar</font>");
        displayStrings.append("<font class='pucpCampo'><input type='checkbox' class='pucpCampo'
name='chkElimina' value='"+String.valueOf(contador)+"'/></font>\\n");
      }
      displayStrings.append("</td> \\n </tr> \\n");
    }
  }
  // IMPORTANTE limpiar variable valorDato espacio en blanco
  valorDato = " ";

  return;
}

if (tipoDato.equals("TI") || tipoDato.equals("TN")) // titulo
{
  displayStrings.append("<td> </td> \\n");
  // campos hidden todas las variables son nulas
  displayStrings.append("<td>");
  displayStrings.append(FormatosComunes.datoPagina(2, " ", "diaIni", besEditable) + "\\n");
  displayStrings.append(FormatosComunes.datoPagina(2, " ", "mesIni", besEditable) + "\\n");
  displayStrings.append(FormatosComunes.datoPagina(2, " ", "anoIni", besEditable) + "\\n");
  displayStrings.append(FormatosComunes.datoPagina(2, " ", "diaFin", besEditable) + "\\n");
  displayStrings.append(FormatosComunes.datoPagina(2, " ", "mesFin", besEditable) + "\\n");
  displayStrings.append(FormatosComunes.datoPagina(2, " ", "anoFin", besEditable) + "\\n");
  displayStrings.append(FormatosComunes.datoPagina(2, " ", "horaIni", besEditable) + "\\n");
  displayStrings.append(FormatosComunes.datoPagina(2, " ", "minutoIni", besEditable) + "\\n");
  displayStrings.append(FormatosComunes.datoPagina(2, " ", "horaFin", besEditable) + "\\n");
  displayStrings.append(FormatosComunes.datoPagina(2, " ", "minutoFin", besEditable) + "\\n");
  displayStrings.append(FormatosComunes.datoPagina(2, " ", "datoIngresado", besEditable) + "\\n");
  displayStrings.append(FormatosComunes.datoPagina(2, " ", "valorDato", besEditable) + "\\n");
  displayStrings.append("</td> \\n </tr> \\n");
  // IMPORTANTE limpiar variable valorDato espacio en blanco
  valorDato = " ";

  return;
}

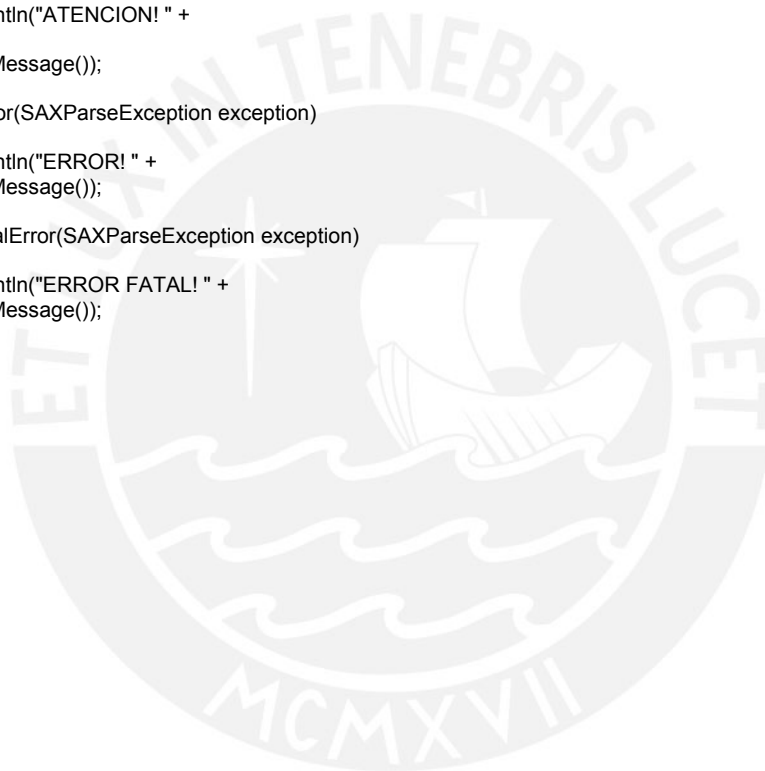
// caso de otros tipos de datos (alfanuméricos, numéricos)
// campos hidden
displayStrings.append("<td width='70%' class='pucpCampo'>");
displayStrings.append(FormatosComunes.datoPagina(2, " ", "diaIni", besEditable) + "\\n");
displayStrings.append(FormatosComunes.datoPagina(2, " ", "mesIni", besEditable) + "\\n");
displayStrings.append(FormatosComunes.datoPagina(2, " ", "anoIni", besEditable) + "\\n");
displayStrings.append(FormatosComunes.datoPagina(2, " ", "diaFin", besEditable) + "\\n");
displayStrings.append(FormatosComunes.datoPagina(2, " ", "mesFin", besEditable) + "\\n");
displayStrings.append(FormatosComunes.datoPagina(2, " ", "anoFin", besEditable) + "\\n");
displayStrings.append(FormatosComunes.datoPagina(2, " ", "horaIni", besEditable) + "\\n");
displayStrings.append(FormatosComunes.datoPagina(2, " ", "minutoIni", besEditable) + "\\n");
displayStrings.append(FormatosComunes.datoPagina(2, " ", "horaFin", besEditable) + "\\n");
displayStrings.append(FormatosComunes.datoPagina(2, " ", "minutoFin", besEditable) + "\\n");
displayStrings.append(FormatosComunes.datoPagina(2, " ", "datoIngresado", besEditable) + "\\n");
if (nlongitud <= 50)

displayStrings.append(FormatosComunes.datoPagina(1, valorDato, "valorDato", nlongitud+1, nlongitud, 0, besEditable) +
"</td> \\n </tr> \\n");
else
  displayStrings.append(FormatosComunes.datoPagina(3, valorDato, "valorDato", 40, nlongitud, 4, besEditable) + "</td>
\\n </tr> \\n");
// IMPORTANTE limpiar variable valorDato espacio en blanco
valorDato = " ";

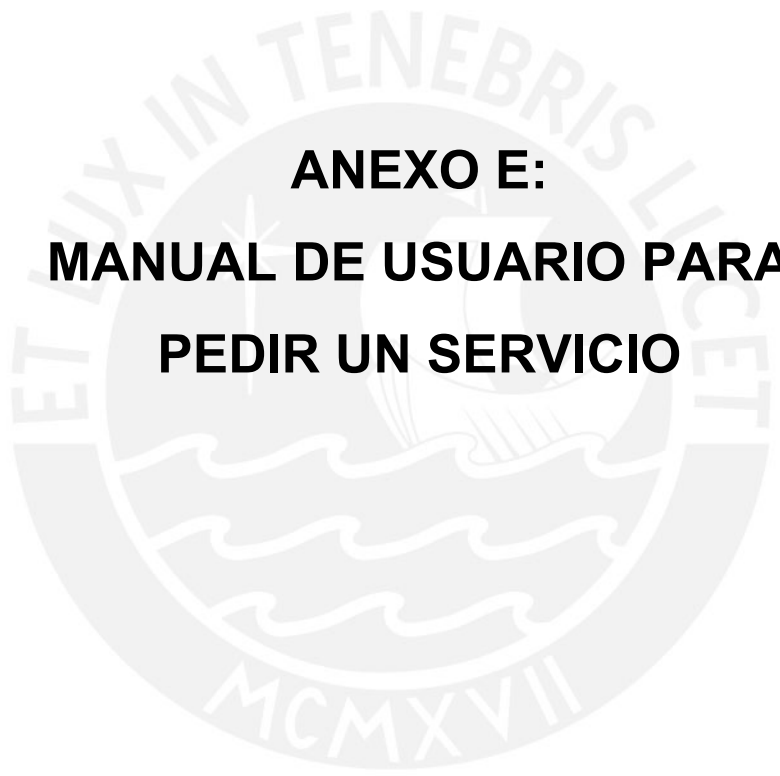
return;

```

```
    }  
  }  
  
}  
  
/**  
 * Fin del documento  
 * <P> Fin del documento  
 * @author Victoria Leon  
 */  
  
public void endDocument()  
{  
displayStrings.append("</table>");  
  
}  
  
public void warning(SAXParseException exception)  
{  
System.err.println("ATENCIÓN! " +  
  
exception.getMessage());  
}  
public void error(SAXParseException exception)  
{  
System.err.println("ERROR! " +  
exception.getMessage());  
}  
public void fatalError(SAXParseException exception)  
{  
System.err.println("ERROR FATAL! " +  
exception.getMessage());  
}  
}  
  
}
```








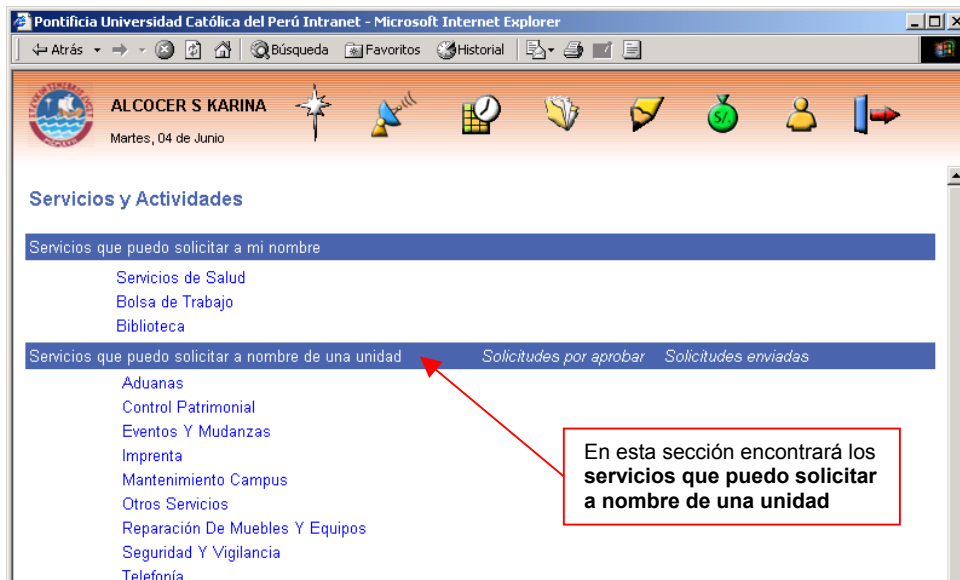
**ANEXO E:  
MANUAL DE USUARIO PARA  
PEDIR UN SERVICIO**

## ¿Cómo accedo a los servicios que puedo solicitar a nombre de mi unidad desde la Intranet?

Al ingresar a la Intranet dar clic en el ícono **Mi Universidad**  , seguidamente verá la siguiente página:



A continuación se mostrará la siguiente página:





## ¿Cómo puedo solicitar un servicio?

Ubíquese en la sección **Servicios que puedo solicitar a nombre de una unidad** y elija la categoría de servicio a solicitar, seguidamente se mostrará una siguiente página en donde deberá elegir el servicio deseado. Por ejemplo:

Servicios que puedo solicitar a nombre de una unidad      Solicitudes por aprobar      Solicitudes enviadas

Aduanas  
Control Patrimonial  
Eventos Y Mudanzas

Haga clic en la categoría deseada, por ejemplo: **Eventos y Mudanzas**

Pontificia Universidad Católica del Perú Intranet - Microsoft Internet Explorer

ALCÓCER S KARINA      Miércoles, 05 de Junio

**Solicitud de Servicio**

Busca el servicio por:  o por su nombre:

Haga clic en el nombre del servicio que va a consultar:

Resultado de búsqueda: 2 ocurrencias

Nombre del Servicio	Unidad que atiende el Servicio	Categoría del Servicio
1 Apoyo A Eventos	OFICINA DE OPERACIONES	EVENTOS Y MUDANZAS
2 Apoyo Para Mudanza	OFICINA DE OPERACIONES	EVENTOS Y MUDANZAS

Haga clic en el servicio deseado, por ejemplo: **Apoyo a Eventos**

Luego, complete la información de la solicitud en la siguiente página:

Pontificia Universidad Católica del Perú Intranet - Microsoft Internet Explorer

ALCÓCER S KARINA      Miércoles, 05 de Junio

**Solicitud de Servicio**

Apoyo A Eventos (OFICINA DE OPERACIONES)

Llene a continuación los datos requeridos:

Datos generales de la solicitud

Solicitado por:  *Unidad que solicita*

Para ser autorizado por:  *Persona que autoriza*

Para ser coordinado por:

Teléfono/Anexo Coord.:

Correo Electr. Coord.:

Registrado por: ALCÓCER SUCHERO, KARINA YVONNE - kalcoce@pucp.edu.pe *Persona que llena la solicitud*

Detalle de la solicitud

Especificación del servicio 1:

Cantidad 1:

Para terminar presione **Grabar**

**En caso la persona de apoyo registre la solicitud, ¿inmediatamente después de grabar los datos, la solicitud es enviada a la unidad que brinda el servicio?**

No, la solicitud no es enviada a la unidad de servicio hasta que la persona que autoriza la haya revisado y autorizado.

Paralelamente a la grabación de los datos realizada por la persona de apoyo, le llegará a la persona que autoriza un correo electrónico con los datos de la solicitud por aprobar.

**En caso la persona que autoriza registre la solicitud, ¿inmediatamente después de grabar los datos, la solicitud es enviada a la unidad que brinda el servicio?**

Sí, automáticamente se asigna un número a la solicitud y es enviada a la unidad de servicio.

Paralelamente le llegará a la unidad de servicio, a la persona de apoyo y a la persona que coordina, un correo electrónico con los datos de la solicitud de servicio por atender.



## ¿Cómo apruebo una solicitud de servicio pendiente de autorización?

Debe seguir los siguientes pasos:

Servicios que puedo solicitar a nombre de una unidad      *Solicitudes por aprobar*      Solicitudes enviadas

Aduanas  
Control Patrimonial  
Eventos Y Mudanzas

Haga clic en la opción **Solicitudes por aprobar**

**Solicitud de Servicio** Regresar

Las siguientes solicitudes deben ser aprobadas por el jefe de la unidad para ser enviadas y atendidas por la unidad de servicio. De ser necesario puede corregir su solicitud.

Haga clic en el documento para autorizar o consultar

Resultado de búsqueda 4 ocurrencias

Fecha y hora registro	Solicitado por	Servicio por solicitar	Autorizado por
03-JUN-2002 13:44	81.01.00.00.0 ADM. DIRECCION DE INFORMATICA	AUTORIZACIÓN DE INGRESO/SALIDA AL CAMPUS	LEON CHAN, VICTORIA (00002837)
03-JUN-2002 13:55	81.01.00.00.0 ADM. DIRECCION DE INFORMATICA	AUTORIZACIÓN DE INGRESO/SALIDA AL CAMPUS	LEON CHAN, VICTORIA (00002837)
03-JUN-2002 19:00	81.01.00.00.0 ADM. DIRECCION DE INFORMATICA	APOYO A EVENTOS	LEON CHAN, VICTORIA (00002837)
05-JUN-2002 16:18	81.01.00.00.0 ADM. DIRECCION DE INFORMATICA	APOYO A EVENTOS	LEON CHAN, VICTORIA (00002837)

Seleccione la solicitud que desea aprobar dando clic sobre el documento.

**Solicitud de Servicio** Nueva solicitud   Solicitudes por enviar   Solicitudes enviadas

**SOLICITUD POR APROBAR** VB Jefe   Editar   Rechazar   Regresar

Servicio: APOYO A EVENTOS - OFICINA DE OPERACIONES  
Estado: Pendiente la autorización de LEON CHAN, VICTORIA (00002837)

**Datos generales de la solicitud**

Solicitado por: 81.01.00.00.0000 ADM. DIRECCION DE INFORMATICA  
Para ser autorizado por: LEON CHAN, VICTORIA (00002837)  
Correo Electr. Soli.: vleon@pucp.edu.pe  
Para ser coordinado por: María López  
Teléfono/Anexo Coord.: 256  
Correo Elect. Coord.:  
Registrado por: ALCÓCER SUCHERO, KARINA YVONNE (19931603)  
Correo Elect. Reg.: kalcoce@pucp.edu.pe

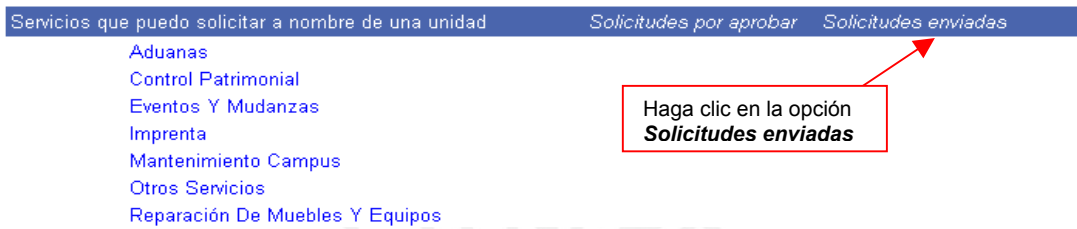
**Detalle de la solicitud**

Si todo está correcto presione el botón **VBJefe**.

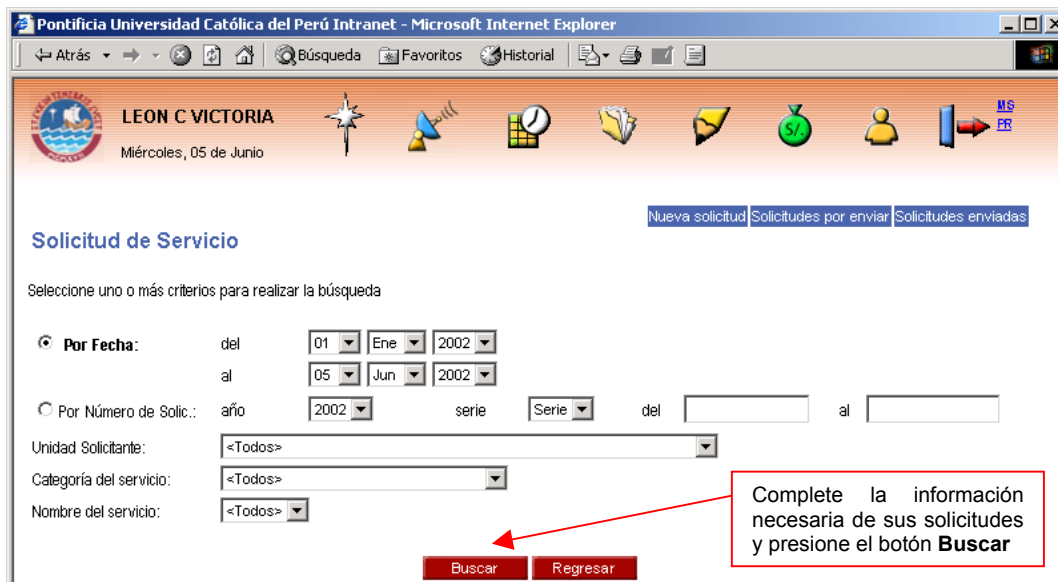
Al aprobar la solicitud de servicio automáticamente se envía un correo electrónico con los datos de la solicitud a la unidad de servicio, a la persona de apoyo y a la persona que coordina.

### ¿Cómo puedo consultar las solicitudes de servicio que han sido enviadas a la Unidad de Servicio correspondiente?

Debe seguir los siguientes pasos:



A continuación aparecerá la siguiente página de búsqueda:



Pontificia Universidad Católica del Perú Intranet - Microsoft Internet Explorer

LEON C VICTORIA  
Miércoles, 05 de Junio

Nueva solicitud | Solicitudes por enviar | Solicitudes enviadas

**Solicitud de Servicio**

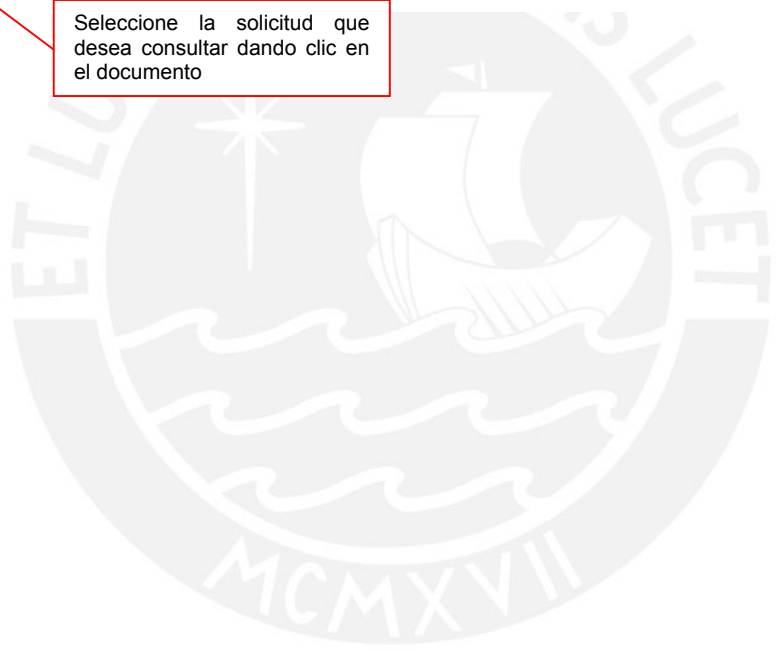
Rango Fecha: del 01-01-2002 al 05-06-2002  
 Categoría de Servicio: Todos  
 Solicitado por: Todos


Si desea consultar el detalle, haga clic en el número de la solicitud.

Resultado de búsqueda 78 ocurrencias

Número de solicitud	Unidad de servicio que atiende	Fecha y hora	Solicitado por	Estado	Servicio solicitado
1 <a href="#">2002-0604-000051</a>	OFICINA DE OPERACIONES	03-JUN-2002 09:59	81.01.00.00.0000 ADM. DIRECCION DE INFORMATICA	POR ATENDER	AUTORIZACIÓN DE INGRESO/SALIDA AL CAMPUS
2 <a href="#">2002-0604-000052</a>	OFICINA DE OPERACIONES	03-JUN-2002 10:02	81.01.00.00.0000 ADM. DIRECCION DE INFORMATICA	POR ATENDER	AUTORIZACIÓN DE INGRESO/SALIDA AL CAMPUS


Seleccione la solicitud que desea consultar dando clic en el documento





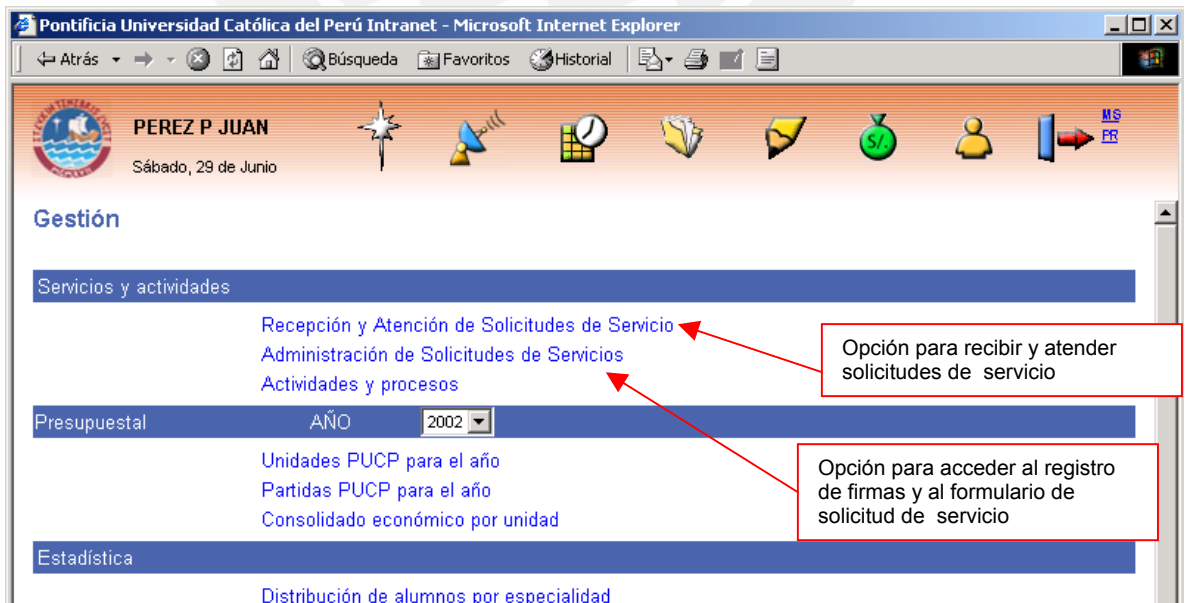
**ANEXO F:  
MANUAL DE USUARIO PARA  
LA UNIDAD DE SERVICIO**

## ¿Cómo accedo para atender las solicitudes de servicio que llegaron a mi oficina?

Al ingresar a la Intranet haga clic en el ícono **Mi Universidad**  , luego verá la siguiente página:

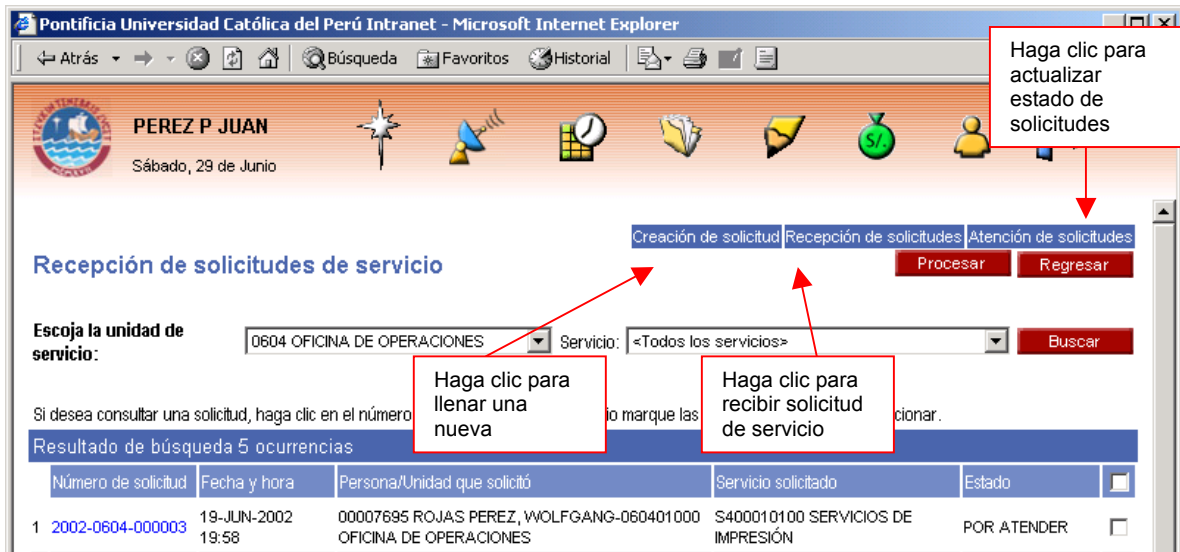


Al presionar el ícono **Gestión** se mostrará la siguiente página:





Haga clic en la opción **Recepción y Atención de Solicitudes de Servicio**. Se mostrará la siguiente pagina:



Recepción de solicitudes de servicio

Escoja la unidad de servicio: 0604 OFICINA DE OPERACIONES Servicio: <Todos los servicios> Buscar

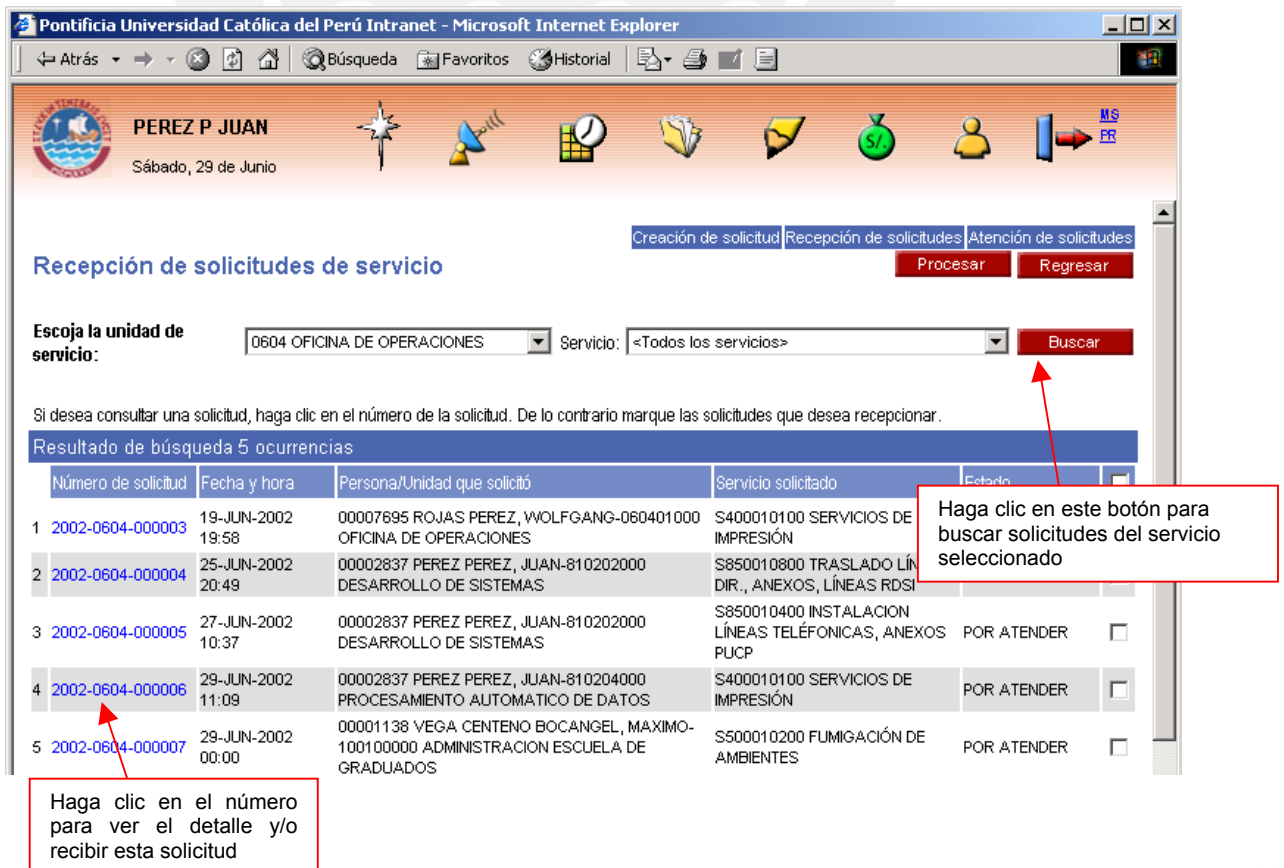
Si desea consultar una solicitud, haga clic en el número de solicitud o marque las solicitudes que desea recibir.

Resultado de búsqueda 5 ocurrencias

Número de solicitud	Fecha y hora	Persona/Unidad que solicitó	Servicio solicitado	Estado
1 2002-0604-000003	19-JUN-2002 19:58	00007695 ROJAS PEREZ, WOLFGANG-060401000 OFICINA DE OPERACIONES	S400010100 SERVICIOS DE IMPRESIÓN	POR ATENDER

### ¿Cómo veo qué solicitudes de servicio llegaron a mi oficina?

Antes que pueda atender una solicitud, ésta debe ser recibida por la Oficina o Unidad de Servicio. Todas las solicitudes llegan primero a la recepción o secretaría con el estado *Por atender*.



Recepción de solicitudes de servicio

Escoja la unidad de servicio: 0604 OFICINA DE OPERACIONES Servicio: <Todos los servicios> Buscar

Si desea consultar una solicitud, haga clic en el número de la solicitud. De lo contrario marque las solicitudes que desea recepcionar.

Resultado de búsqueda 5 ocurrencias

Número de solicitud	Fecha y hora	Persona/Unidad que solicitó	Servicio solicitado	Estado
1 2002-0604-000003	19-JUN-2002 19:58	00007695 ROJAS PEREZ, WOLFGANG-060401000 OFICINA DE OPERACIONES	S400010100 SERVICIOS DE IMPRESIÓN	
2 2002-0604-000004	25-JUN-2002 20:49	00002837 PEREZ PEREZ, JUAN-810202000 DESARROLLO DE SISTEMAS	S850010800 TRASLADO LÍNEA DIR., ANEXOS, LÍNEAS RDSI	
3 2002-0604-000005	27-JUN-2002 10:37	00002837 PEREZ PEREZ, JUAN-810202000 DESARROLLO DE SISTEMAS	S850010400 INSTALACION LÍNEAS TELEFÓNICAS, ANEXOS PUCP	POR ATENDER
4 2002-0604-000006	29-JUN-2002 11:09	00002837 PEREZ PEREZ, JUAN-810204000 PROCESAMIENTO AUTOMÁTICO DE DATOS	S400010100 SERVICIOS DE IMPRESIÓN	POR ATENDER
5 2002-0604-000007	29-JUN-2002 00:00	00001138 VEGA CENTENO BOCANGEL, MAXIMO-100100000 ADMINISTRACION ESCUELA DE GRADUADOS	S500010200 FUMIGACIÓN DE AMBIENTES	POR ATENDER

**¿Puedo recibir varias solicitudes a la vez?**

Sí. Vaya a la página de Recepción de **Solicitudes de Servicio**. Seleccione las solicitudes que desee y haga clic en el botón procesar.

Si desea consultar una solicitud, haga clic en el número de la solicitud. De lo contrario marque las solicitudes que desea recepcionar.

Resultado de búsqueda 5 ocurrencias

Número de solicitud	Fecha y hora	Persona/Unidad que solicitó	Servicio solicitado	Estado	<input type="checkbox"/>
1 2002-0604-000003	19-JUN-2002 19:58	00007695 ROJAS PEREZ, WOLFGANG-060401000 OFICINA DE OPERACIONES	S400010100 SERVICIOS DE IMPRESIÓN	POR ATENDER	<input type="checkbox"/>
2 2002-0604-000004	25-JUN-2002 20:49	00002837 PEREZ PEREZ, JUAN-810202000 DESARROLLO DE SISTEMAS	S850010800 TRASLADO LÍNEAS DIR., ANEXOS, LÍNEAS RDSI	POR ATENDER	<input checked="" type="checkbox"/>
3 2002-0604-000005	27-JUN-2002 10:37	00002837 PEREZ PEREZ, JUAN-810202000 DESARROLLO DE SISTEMAS	S850010400 INSTALACION LÍNEAS TELÉFONICAS, ANEXOS PUCP	POR ATENDER	<input checked="" type="checkbox"/>
4 2002-0604-000006	29-JUN-2002 11:09	00002837 PEREZ PEREZ, JUAN-810204000 PROCESAMIENTO AUTOMATICO DE DATOS	S400010100 SERVICIOS DE IMPRESIÓN	POR ATENDER	<input type="checkbox"/>
5 2002-0604-000007	29-JUN-2002 00:00	00001138 VEGA CENTENO BOCANGEL, MAXIMO-100100000 ADMINISTRACION ESCUELA DE GRADUADOS	S500010200 FUMIGACIÓN DE AMBIENTES	POR ATENDER	<input type="checkbox"/>

Seleccione las solicitudes

Seguidamente se mostrará la siguiente página. Seleccione el módulo de atención al que le enviará las solicitudes para atender y presione el botón grabar

Pontificia Universidad Católica del Perú Intranet - Microsoft Internet Explorer

PEREZ P JUAN  
Sábado, 29 de Junio

[Creación de solicitud](#) | [Recepción de solicitudes](#) | [Atención de solicitudes](#)

**Recepción de solicitudes de servicio**

0604 OFICINA DE OPERACIONES - RECEPCION

Datos a asignar a las solicitudes

**Atender y enviar a módulo:** SERV. TELEFONIA

**Rechazar**

Información adicional

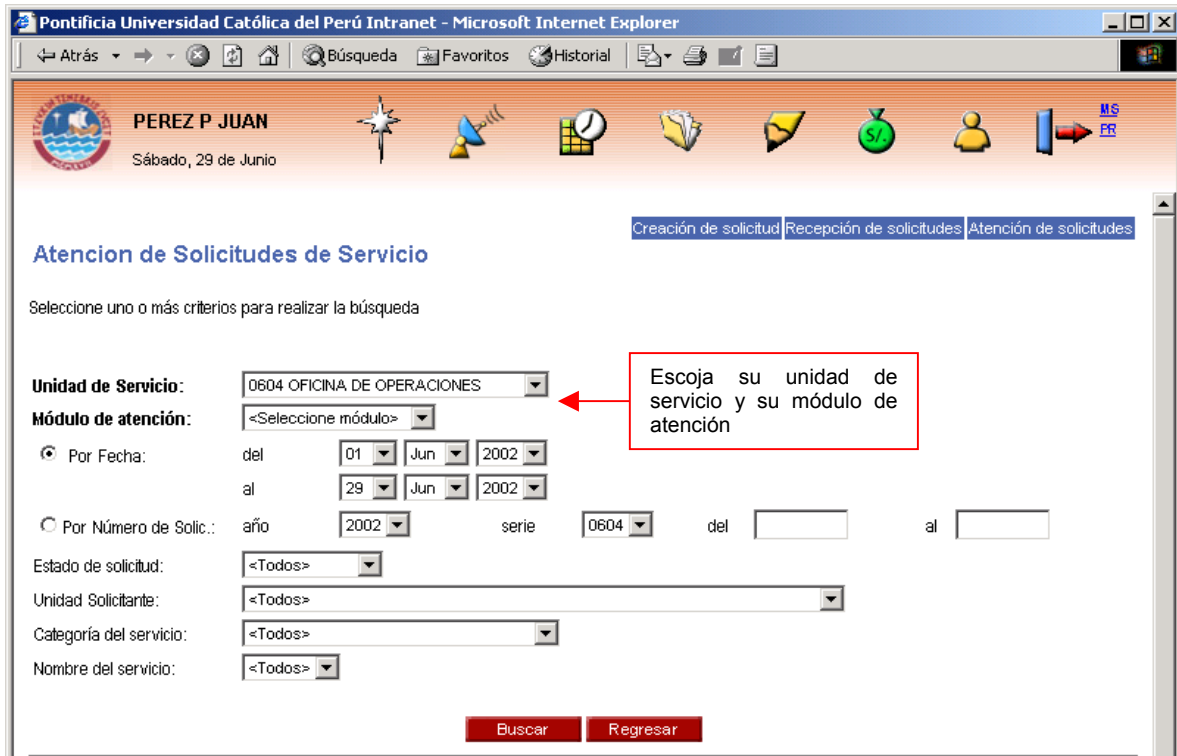
Solicitudes que serán procesadas: 2

Número de solicitud	Fecha y hora	Persona/Unidad que solicitó	Servicio solicitado	Estado
1 2002-0604-000004	25-JUN-2002 20:49	00002837 PEREZ PEREZ, JUAN-810202000 DESARROLLO DE SISTEMAS	S850010800 TRASLADO LÍNEAS DIR., ANEXOS, LÍNEAS RDSI	POR ATENDER
2 2002-0604-000005	27-JUN-2002 10:37	00002837 PEREZ PEREZ, JUAN-810202000 DESARROLLO DE SISTEMAS	S850010400 INSTALACION LÍNEAS TELÉFONICAS, ANEXOS PUCP	POR ATENDER

Solicitudes que van a hacer recibidas

### ¿Cómo consulto las solicitudes de servicio que hay en mi módulo de atención?

Haga clic en la opción del menú **Atención de solicitudes**. Se mostrará la siguiente página de criterios de búsqueda. Complete los datos necesarios y haga clic en el botón **Buscar**.



Pontificia Universidad Católica del Perú Intranet - Microsoft Internet Explorer

Perez P Juan  
Sábado, 29 de Junio

Creación de solicitud | Recepción de solicitudes | **Atención de solicitudes**

#### Atencion de Solicitudes de Servicio

Seleccione uno o más criterios para realizar la búsqueda

**Unidad de Servicio:** 0604 OFICINA DE OPERACIONES

**Módulo de atención:** <Seleccione módulo>

Por Fecha: del 01 Jun 2002 al 29 Jun 2002

Por Número de Solic.: año 2002 serie 0604 del [ ] al [ ]

Estado de solicitud: <Todos>

Unidad Solicitante: <Todos>

Categoría del servicio: <Todos>

Nombre del servicio: <Todos>

**Buscar** **Regresar**

Aparecerá una página con el resultado de su búsqueda similar a ésta:



Pontificia Universidad Católica del Perú Intranet - Microsoft Internet Explorer

Perez P Juan  
Sábado, 29 de Junio

Creación de solicitud | Recepción de solicitudes | **Atención de solicitudes**

**0604 OFICINA DE OPERACIONES - SERV.TELEFONIA**

Rango Fecha: del 01-06-2002 al 29-06-2002 Estado: [ ]

Unidad Solicitante: [ ] Categoría Serv.: [ ]

Si desea consultar el detalle, haga clic en el número de la solicitud. De lo contrario marque las solicitudes que desea atender.

Resultado de búsqueda 2 ocurrencias

Número de solicitud	Fecha y hora	Persona/Unidad que solicitó	Servicio solicitado	Estado	
1 2002-0604-000004	25-JUN-2002 20:49	00002837 PEREZ PEREZ, JUAN-810202000 DESARROLLO DE SISTEMAS	S850010800 TRASLADO LÍNEAS DIR., ANEXOS, LÍNEAS RDSI	EN PROCESO	<input type="checkbox"/>
2 2002-0604-000006	29-JUN-2002 11:09	00002837 PEREZ PEREZ, JUAN-810204000 PROCESAMIENTO AUTOMATICO DE DATOS	S400010100 SERVICIOS DE IMPRESIÓN	EN PROCESO	<input type="checkbox"/>

**Procesar** **Regresar**

### ¿Cómo atiendo una sola solicitud?

Realice el mismo procedimiento para consultar solicitudes de servicio (ver pregunta anterior). Haga clic en el número de la solicitud y se mostrará la información de la solicitud.



Pontificia Universidad Católica del Perú Intranet - Microsoft Internet Explorer

PEREZ P JUAN  
Sábado, 29 de Junio

Creación de solicitud | Recepción de solicitudes | **Atención de solicitudes**

[Editar](#) [Regresar](#)

#### 0604 OFICINA DE OPERACIONES - SERV.TELEFONIA

**Datos generales**

Número de solicitud: 2002-0604-000004  
 Fecha y hora: 25-JUN-2002 20:49  
 Unidad que solicita: 81.02.02.00.0000 DESARROLLO DE SISTEMAS  
 Persona que autoriza: PEREZ PEREZ, JUAN (00002837)  
 Correo Electr. Solic.: vleon@pucp.edu.pe  
 Persona de coordinación: Sr. Juan Ponce  
 Teléfono/Anexo Coord.: 123  
 Correo Elect. Coord.:

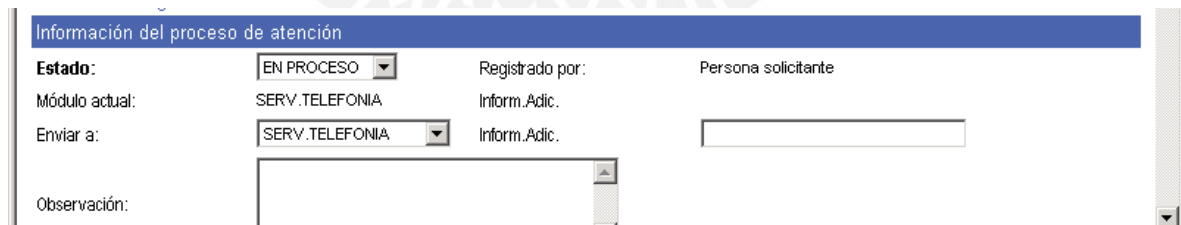
**Información del proceso de atención**

Estado: EN PROCESO Registrado por: Persona solicitante  
 Módulo actual: SERV.TELEFONIA Inform.Adic.  
 Observación:

**Detalle de la solicitud**

Servicio: S850010800-TRASLADO LÍNEAS DIR., ANEXOS, LÍNEAS RDSI  
 Numero de línea directa v/o anexo: 4614590

Presione el botón **editar** para actualizar el estado de la solicitud. Al finalizar presione el botón **grabar**.

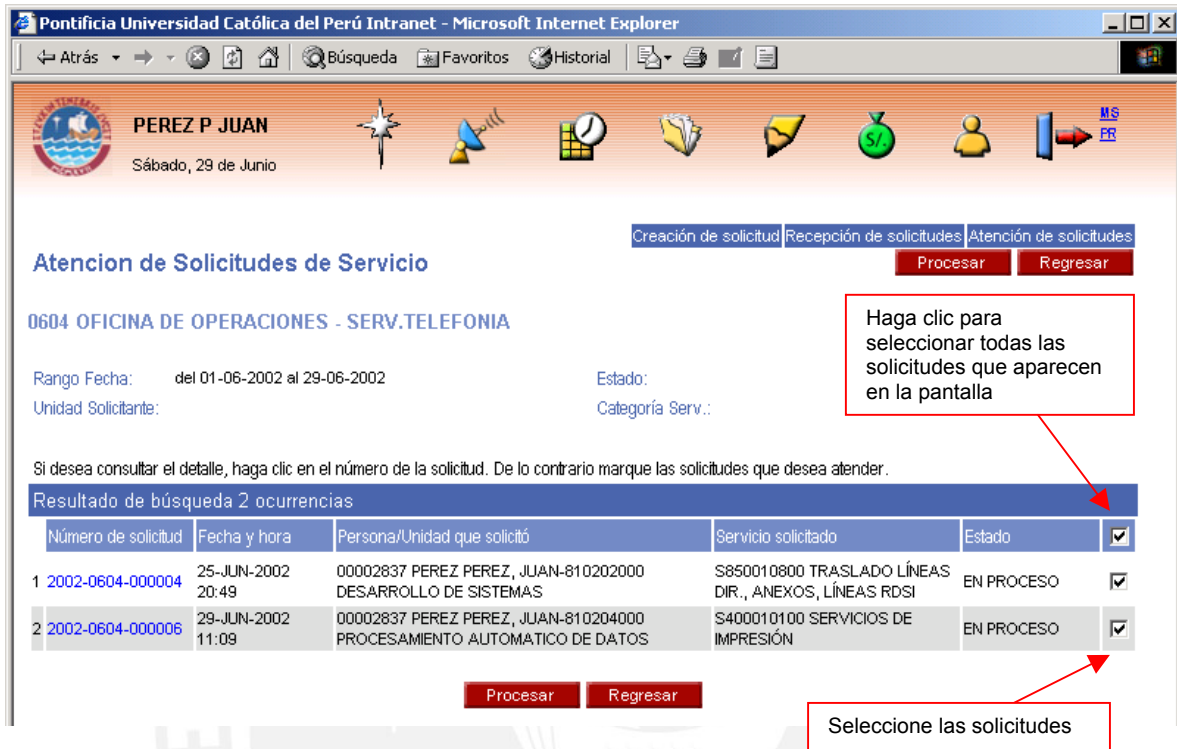


**Información del proceso de atención**

Estado:  Registrado por: Persona solicitante  
 Módulo actual: SERV.TELEFONIA Inform.Adic.  
 Enviar a:  Inform.Adic.   
 Observación:

### ¿Cómo atiendo varias solicitudes a la vez?

Realice el mismo procedimiento para consultar solicitudes de servicio. Seleccione las solicitudes que desea atender y luego presione el botón **Procesar**.



**Atencion de Solicitudes de Servicio**

0604 OFICINA DE OPERACIONES - SERV.TELEFONIA

Rango Fecha: del 01-06-2002 al 29-06-2002 Estado:  
Unidad Solicitante: Categoría Serv.:

Si desea consultar el detalle, haga clic en el número de la solicitud. De lo contrario marque las solicitudes que desea atender.

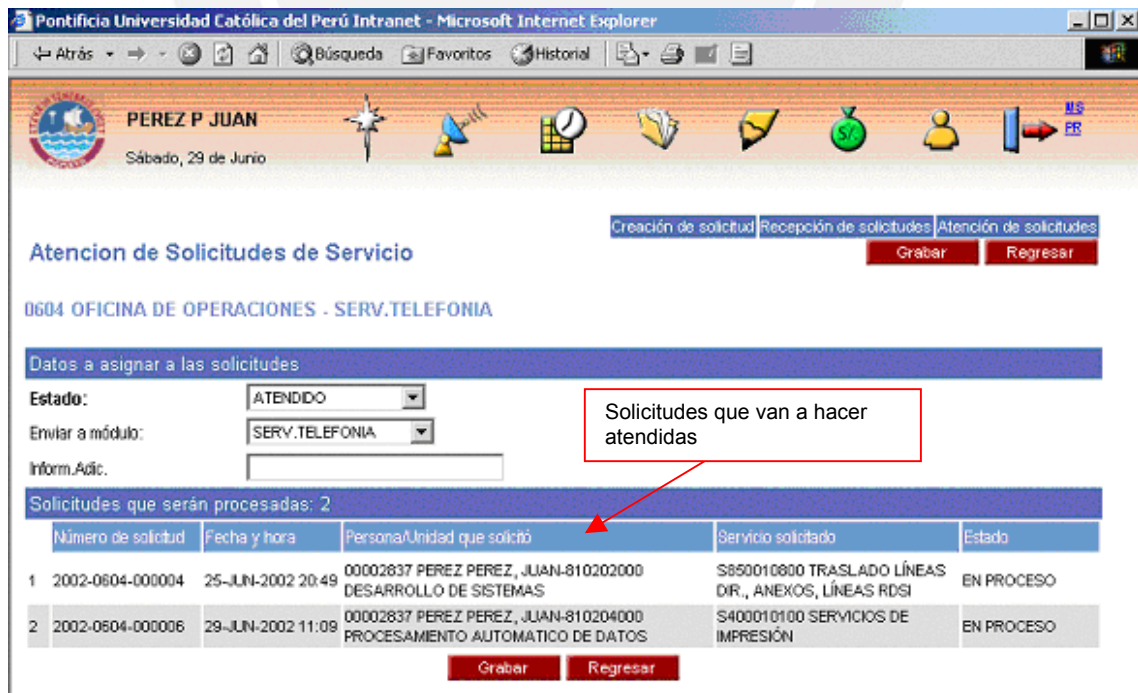
Resultado de búsqueda 2 ocurrencias

Número de solicitud	Fecha y hora	Persona/Unidad que solicitó	Servicio solicitado	Estado	<input type="checkbox"/>
1 2002-0604-000004	25-JUN-2002 20:49	00002837 PEREZ PEREZ, JUAN-810202000 DESARROLLO DE SISTEMAS	S850010800 TRASLADO LÍNEAS DIR., ANEXOS, LÍNEAS RDSI	EN PROCESO	<input checked="" type="checkbox"/>
2 2002-0604-000006	29-JUN-2002 11:09	00002837 PEREZ PEREZ, JUAN-810204000 PROCESAMIENTO AUTOMATICO DE DATOS	S400010100 SERVICIOS DE IMPRESIÓN	EN PROCESO	<input checked="" type="checkbox"/>

Procesar Regresar

Seleccione las solicitudes

Actualice el estado de las solicitudes y presione el botón **Grabar**.



**Atencion de Solicitudes de Servicio**

0604 OFICINA DE OPERACIONES - SERV.TELEFONIA

Datos a asignar a las solicitudes

Estado:   
Enviar a módulo:   
Inform.Adic.:

Solicitudes que serán procesadas: 2

Número de solicitud	Fecha y hora	Persona/Unidad que solicitó	Servicio solicitado	Estado
1 2002-0604-000004	25-JUN-2002 20:49	00002837 PEREZ PEREZ, JUAN-810202000 DESARROLLO DE SISTEMAS	S850010800 TRASLADO LÍNEAS DIR., ANEXOS, LÍNEAS RDSI	EN PROCESO
2 2002-0604-000006	29-JUN-2002 11:09	00002837 PEREZ PEREZ, JUAN-810204000 PROCESAMIENTO AUTOMATICO DE DATOS	S400010100 SERVICIOS DE IMPRESIÓN	EN PROCESO

Grabar Regresar

Solicitudes que van a hacer atendidas



### Si el solicitante ha enviado su solicitud por otros medios, cómo lo incluyo en el Sistema?

Presione la opción del menú **Creación de solicitud**, seleccione el servicio que va a atender al solicitante.



Creación de solicitud | Recepción de solicitudes | Atención de solicitudes

**Nueva solicitud para atender**

Unidad de Servicio: 0604 OFICINA DE OPERACIONES

Busca el servicio por: Mantenimiento Campus o por su nombre:  **Buscar**

Haga clic en el nombre del servicio que se atenderá:

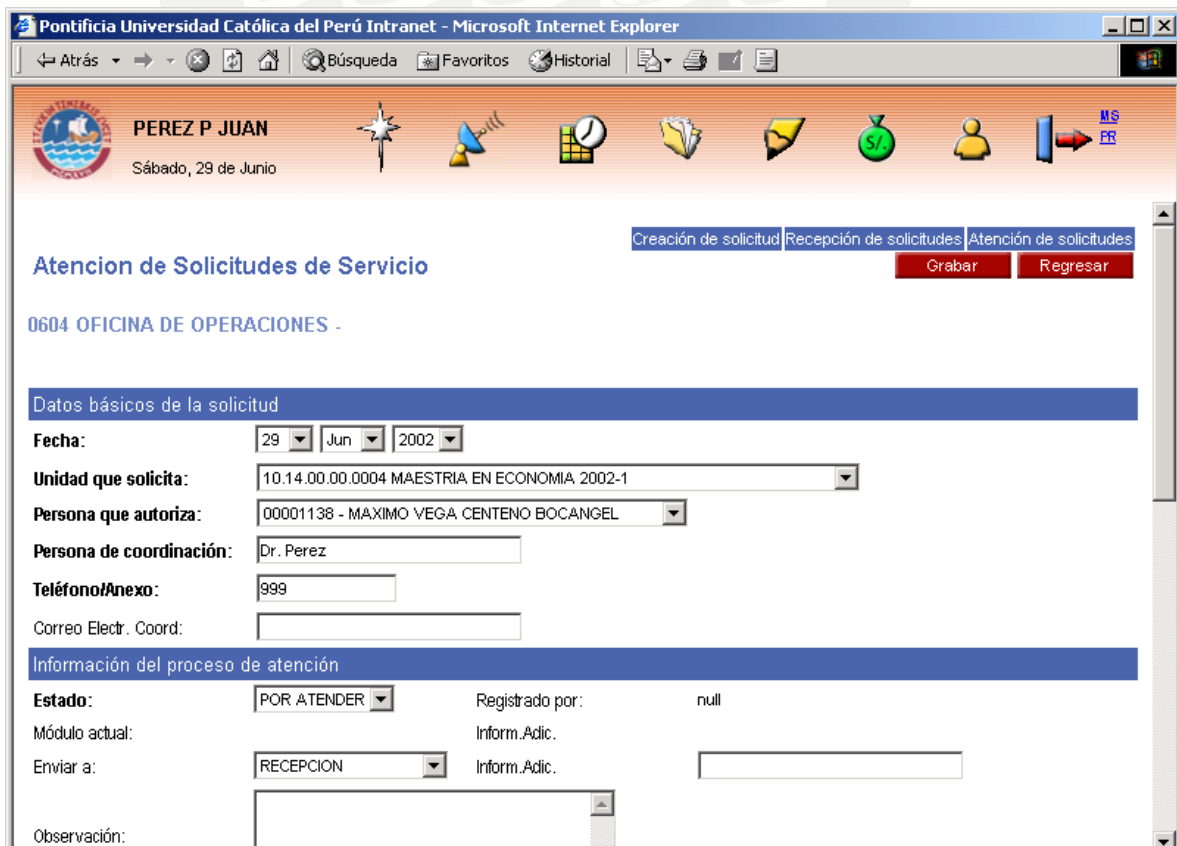
Resultado de búsqueda: 4 ocurrencias

Nombre del Servicio	Unidad que atiende el Servicio	Categoría del Servicio
1 Eliminación De Desechos	OFICINA DE OPERACIONES	MANTENIMIENTO CAMPUS
2 Fumigación De Ambientes	OFICINA DE OPERACIONES	MANTENIMIENTO CAMPUS
3 Jardines Y Plantas	OFICINA DE OPERACIONES	MANTENIMIENTO CAMPUS
4 Lavado Y Mantenim. De Alfombras Y Cortinas	OFICINA DE OPERACIONES	MANTENIMIENTO CAMPUS

Haga clic en un servicio

Haga clic en este botón para buscar relación de servicios

Complete la información de la solicitud y presione el botón **Grabar**.



Creación de solicitud | Recepción de solicitudes | Atención de solicitudes

**Atencion de Solicitudes de Servicio** **Grabar** **Regresar**

0604 OFICINA DE OPERACIONES -

Datos básicos de la solicitud

Fecha: 29 Jun 2002

Unidad que solicita: 10.14.00.00.0004 MAESTRIA EN ECONOMIA 2002-1

Persona que autoriza: 00001138 - MAXIMO VEGA CENTENO BOCANGEL

Persona de coordinación: Dr. Perez

Teléfono/Anexo: 999

Correo Electr. Coord:

Información del proceso de atención

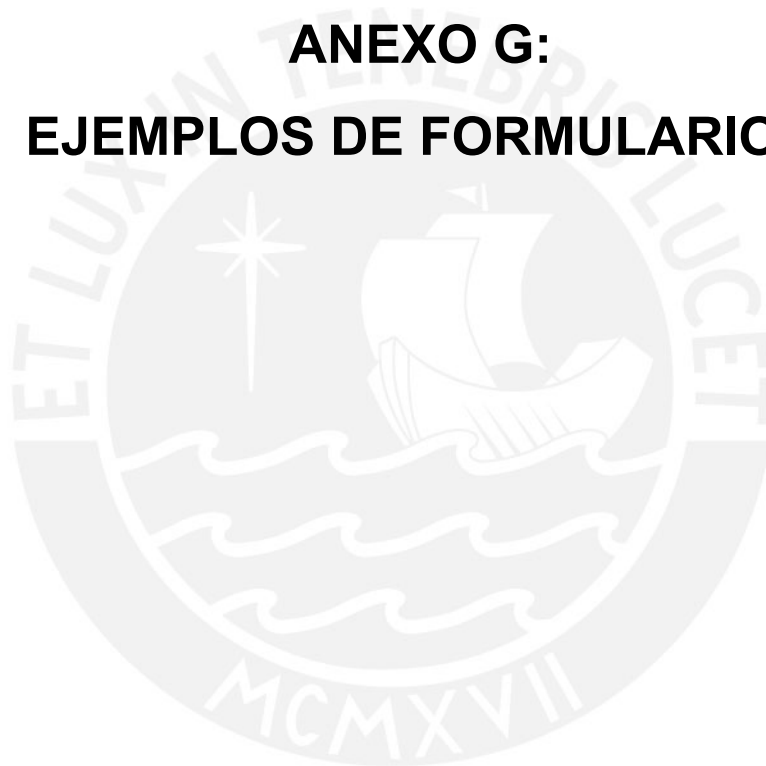
Estado: POR ATENDER Registrado por: null

Módulo actual: Inform.Adic.

Enviar a: RECEPCION Inform.Adic.

Observación:

## ANEXO G: EJEMPLOS DE FORMULARIOS





**Fumigación de ambientes.**

**Especificacion del servicio**

**Ambientes**

**Fecha requerida** Dia  Mes  Año

Horario disponible del  --  al  --

**Justificacion**

Observaciones

**Servicio de movilidad**

**Fecha** Dia  Mes  Año

**Hora Salida**  --

**Hora Regreso**  --

**Lugar de Salida**

**Lugar de Destino**

**Cantidad de Personas**

Justificacion

Observaciones

**Persona Responsable del Traslado**

**Traslado de líneas telefónicas**

<b>Numero de línea directa y/o anexo</b>	<input type="text"/>
<b>Ubicacion Inicial</b>	<input type="text"/>
<b>Ubicacion Final</b>	<input type="text"/>
<b>Observaciones</b>	<input type="text"/>

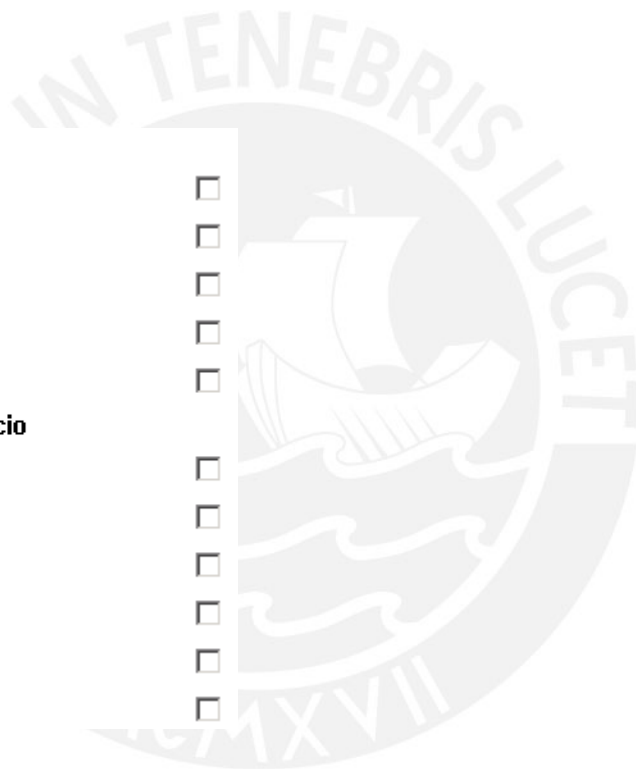
**Imprenta:**

**Tipo de Servicio**

- Diagramacion/Diseño:
- Impresion
- Fotocopiado/Duplicacion
- Encuadernado/ Anillado
- Otros Servicios

**Infomacio para el servicio**

- Reimpresion
- Diseño
- Tira
- Retira
- Numero de originales
- Numero de copias



**Carta de presentación (alumno)**

**I. CARTA DIRIGIDA A**

**Título Persona**

- A) Doctor(a)
- B) Licenciado(a)
- C) Señor(a)
- D) Señorita

**Nombre Persona**

**Cargo**

**Institución/Empresa**

**II. SOLICITA PARA: (marque lo que corresponda con una X)**

- A) Prácticas
- B) Hacer uso de biblioteca
- C) Laborar
- D) Postular a universidad del extranjero
- E) Beca de estudio
- F) Obtener información para la elaboración de trabajos en:

1. Curso

**Oficina de Informaciones**

Vacantes

**Costo**

Financiamiento

Banco/cuenta

Contacto

Teléfonos

Correo electrónico

Página Web

**MEDIOS DE DIFUSIÓN DEL EVENTO**

**Envío de información escrita (folletería, afiches, etc.)**

Adjunta archivo: (attach) a:  
informa@pucp.edu.pe

Examinar...



**ANEXO H:  
GLOSARIO DE TERMINOS**

**Unidad de Servicio**

Es la oficina o unidad de la universidad que atiende un servicio.

**Unidad de Solicitante**

Es la oficina o unidad de la universidad que tiene un requerimiento o necesidad. Por ejemplo: reparación de una fotocopiadora.

**Browser**

Programa que se utiliza para desplegar una página web e interactuar en ella.

**CGI**

Common Gateway Interface. Es un programa que se ejecuta en tiempo real en un Web Server en respuesta a una solicitud de un Browser

**Clase**

Una entidad lógica que encapsula datos y comportamiento. Una clase es un modelo para un objeto.

**MVC**

Arquitectura Modelo-Vista-Controlador utilizada en el desarrollo de aplicaciones internet. Los programas son desarrollados por separado y se unen en tiempo de ejecución.

**Servlet**

Programa escrito en Java que se ejecuta en un servidor y que recibe y responde las peticiones de los usuarios.

**XML**

Extensible Markup Language. Utilizado para estructurar los datos contenidos en documentos.

## Bibliografía

- Edward Yourdon, “*Análisis Estructurado Moderno*”: Primera edición, Prentice-Hall Hispanoamericana, 1993.
- Roger S. Pressman, “*Ingeniería del Software*”: Tercera edición, McGraw-Hill, 1993.
- James A. Senn, “*Análisis y Diseño de Sistemas de Información*”: Segunda edición, McGraw-Hill, 1992.
- Paul McFedries, “*Creando una Página Web con HTML ¡Fácil!*”: Segunda edición, Prentice-Hall, 1997.
- Hans Bergen, “*Java Server Pages*”: Primera Edición, O’Reilly, 2001.
- Paul Whitehead, “*JavaServer Pages*”: Hungry Minds, Inc., 2001.
- Deitel y Deitel, “*Cómo programar en Java*”: Primera Edición, Prentice-Hall, 1998.
- Nirva Morisseau-Leroy, “*Oracle 8i Programación de componentes Java*”: Primera edición, McGraw –Hill, 2001.
- Marty Hall, “*Servlets and JavaServer Pages*”: Prentice-Hall, 2000.
- Steve Muench, “*Building Oracle XML Applications*”: First edition, O’Reilly, 2000.
- Steve Holzner, “*Inside XML*”: First edition, New Riders Publishing, 2000.
- World Wide Web Consortium, <http://www.w3.org>



**INDICE**

<b>INTRODUCCION.....</b>	<b>1</b>
<b>CAPITULO I. METODOLOGIA DE TRABAJO.....</b>	<b>4</b>
1.1 Ciclo de vida de un proyecto. ....	
1.2 Análisis y diseño estructurado.....	5
1.3 Flujogramas del sistema.....	9
<b>CAPITULO II. PROCEDIMIENTOS PARA SOLICITAR SERVICIOS EN UNA INSTITUCION EDUCATIVA.....</b>	<b>10</b>
2.1 Oficina de Operaciones.....	
2.2 Oficina de Infraestructura.....	11
2.3 Facultades.....	12
2.4 Requerimientos. ....	14
<b>CAPITULO III. ESTRUCTURACION DE UNA SOLICITUD DE SERVICIO.....</b>	<b>16</b>
3.1 Concepto de metadata.....	17
3.2 Formulario de una solicitud de servicio.....	18
3.3 Datos que llena el solicitante en el formulario. ....	20
3.3.1 Concepto de XML.....	22
3.3.1.1 Características del XML. ....	
3.3.1.2 Estructura de un documento XML.....	23
3.3.2 Formulario electrónico generalizado para una solicitud de servicio. ....	24
<b>CAPITULO IV. DISEÑO DEL SISTEMA.....</b>	<b>28</b>
4.1. Planteamiento Funcional del sistema.....	
4.2. Planteamiento de la Base de Datos del sistema.....	38
4.3 Planteamiento de los Procesos del sistema.....	42
4.3.1. Conceptos sobre Aplicaciones.....	43
4.3.2. Arquitectura de las Aplicaciones.....	44
4.3.3. Aplicaciones del sistema.....	47
4.4. Planteamiento dinámico del sistema.....	49
4.4.1. Planteamiento de diagramas de estado para procesos del sistema. ....	
4.4.2. Planteamiento de diagramas de secuencia para procesos del sistema. ....	53
4.5 Control de acceso al Sistema.....	56

<b>CAPITULO V. FUNCIONES CRITICAS DESARROLLADAS</b> .....	58
5.1 Los APIs DOM y SAX. ....	
5.2 Función para convertir un formulario electrónico en código HTML (presentación).	59
5.3 Función para convertir los datos de la solicitud en HTML a un documento XML.	61
5.4 Función para convertir un documento XML de una solicitud de servicio en código HTML (presentación). ....	63
<b>CAPITULO VI. CONCLUSIONES</b> .....	66

## ANEXOS

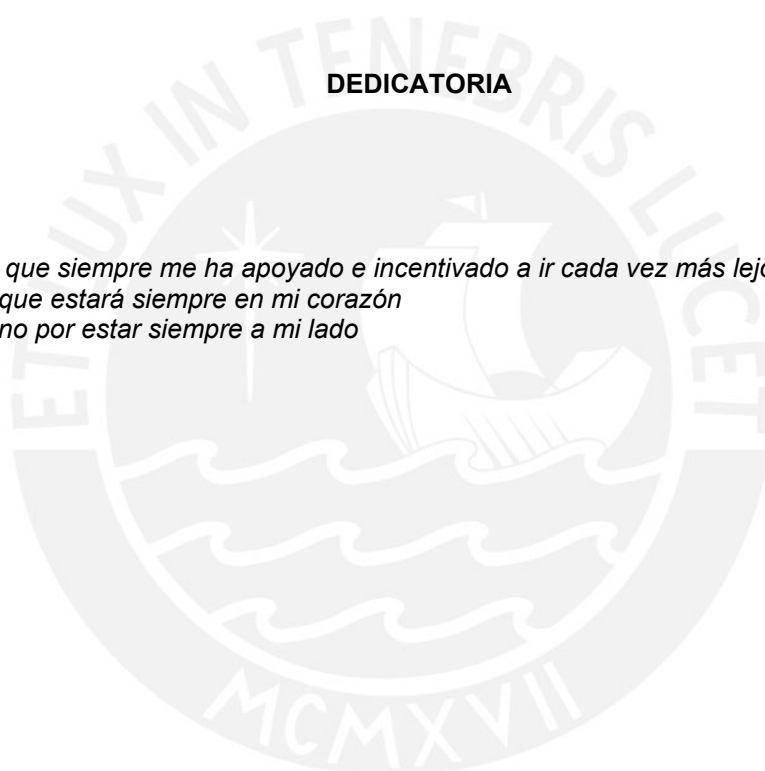
- ANEXO A. Código Fuente de aplicación: Recibir y atender varias solicitudes a la vez.  
 ANEXO B. Código Fuente de función: Función para convertir un formulario Electrónico en código HTML  
 ANEXO C. Código Fuente de función: Función para convertir los datos de la Solicitud en HTML a un documento XML  
 ANEXO D. Código Fuente de función: Función para convertir un documento XML de una solicitud de servicio en código HTML  
 ANEXO E: Manual de usuario para pedir un servicio  
 ANEXO F: Manual de usuario para unidad de servicio  
 ANEXO G: Ejemplos de formularios

## Bibliografía



## DEDICATORIA

*A mi madre que siempre me ha apoyado e incentivado a ir cada vez más lejos  
A mi padre que estará siempre en mi corazón  
A mi hermano por estar siempre a mi lado*



## AGRADECIMIENTOS

*A la Pontificia Universidad Católica, mi alma mater.*

*Al Dr. Maynard Kong quien me asesoró en la presente tesis.*

*A Consuelo Moreno por sus palabras.*

*A mis compañeros de trabajo en la Dirección de Informática que me hacen sentir como en familia y que sin ellos no hubiera sido posible el desarrollo de la tesis.*

*A mi familia por estar ahí.*

*A los amigos por ser quienes son.*