UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería
Departamento de Computación

"Análisis y diseño de un sistema parametrizable para el control de flujo de trabajo orientado a la administración de proyectos de software: Definición de procesos"

Trabajo de investigación presentado por
Jenniffer Suset Guzmán Verbena para optar
al grado académico de Licenciada en Ciencias de la Computación.

Guatemala
2006

“Análisis y diseño de un sistema parametrizable para el control de flujo de trabajo orientado a la administración de proyectos de software: Definición de procesos”

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería
Departamento de Computación

"Análisis y diseño de un sistema parametrizable para el control de flujo de trabajo orientado a la administración de proyectos de software: Definición de procesos"

Trabajo de investigación presentado por
Jenniffer Suset Guzmán Verbena para optar
al grado académico de Licenciada en Ciencias de la Computación.

Guatemala
2006

PREFACIO

Inicialmente, la idea de este trabajo fue investigar una metodología relativamente desconocida en Guatemala y aplicarla al desarrollo de software en el país. Al mismo tiempo, una compañera de trabajo realizaba una investigación respecto el control del flujo de trabajo de una empresa específica. Enseguida se identificó la posibilidad de unir ambos esfuerzos en un solo proyecto, siempre tomando en cuenta los objetivos originales de cada trabajo, pero incorporando nuevos objetivos y ofreciendo un nuevo enfoque: la definición y el diseño de un sistema totalmente parametrizable de control del flujo de trabajo de una empresa.

En las siguientes páginas, únicamente se detalla el análisis y diseño de la solución y se describen de forma general las especificaciones técnicas para su implantación. No obstante, se proveen todos los conceptos fundamentales y los lineamientos a seguir para el desarrollo y culminación exitosos de la solución propuesta.

MI MÁS SINCERO AGRADECIMIENTO A:

A Dios, por permitirme llegar a este punto tan importante de mi vida.

A mis padres y hermanos, por enseñarme el camino del bien y apoyarme incondicionalmente en todo momento.

A Jorge, por motivarme y darme fuerzas cuando más lo necesité. Por inspirarme a ser una mejor persona.

Al Ing. Julio Manuel Alvarez, por su constante ayuda y apoyo en la realización de este trabajo.

Jenniffer Guzmán

INDICE

LISTA DE CUADROS

LISTA DE ILUSTRACIONES

Ilustración Página

Marco teórico

Análisis

Diseño

RESUMEN

Este trabajo presenta el análisis y diseño de una herramienta parametrizable para el control de flujo de trabajo en una empresa, orientada a la administración de proyectos de software, denominada **SmartFlow**.

Esta herramienta permite la automatización del diseño y administración de procesos y tareas, de forma completamente parametrizable, proveyendo diversos tipos de análisis y reportes para evaluar la eficiencia y el funcionamiento del flujo de trabajo definido dentro del sistema. Además, permite automatizar el flujo de información dentro de una empresa, pues ya se sabe de antemano hacia donde debe dirigirse. Esta herramienta basa y promueve su seguridad en la especialización del trabajo, pues provee varios tipos de usuario, cada uno con distintos niveles de acceso dentro del sistema.

I. INTRODUCCIÓN

Es muy bien sabido que el desarrollo de software no es una tarea fácil: es un largo proceso que involucra múltiples tareas, así como la participación de muchas personas interactuando entre sí. En la mayoría de los casos, el proceso de desarrollo se lleva a cabo sin ningún plan de trabajo, lo que puede tener consecuencias graves para el desarrollo de un proyecto: atraso en el tiempo de entrega, mala utilización de recursos e incompatibilidad de módulos, por mencionar algunos ejemplos. Debido a esto, se identificó la necesidad de utilizar una disciplina para desarrollar los proyectos de software de una manera más eficiente; fue así como surgió el concepto de metodología. El objetivo de una metodología es hacer que el proceso al que se aplica la metodología sea más eficiente, obteniendo siempre los resultados requeridos, según las necesidades predefinidas.

El concepto de este trabajo es diseñar un sistema que permita al usuario final definir el flujo de trabajo de una empresa de desarrollo de software, facilitando la definición de procesos y tareas relacionadas. El sistema diseñado debe permitir una operación flexible y parametrizable. La definición de procesos, tareas y flujos debe ser útil para permitir llevar el control de los diferentes proyectos en la empresa.

El análisis y diseño del sistema fue dividido en dos módulos: el Módulo administrativo y el Módulo de usuario. Este trabajo tiene como finalidad diseñar el Módulo administrativo. Específicamente, este módulo permitirá al usuario final definir un flujo de trabajo de tal manera que se ajuste a las necesidades de su empresa, siempre siguiendo los lineamientos más destacados de tres metodologías a investigar. El módulo y diseño complementario de Manejo de proyectos o de usuarios, será desarrollado por Mariana Rendón en el trabajo titulado: “Análisis y diseño de un sistema parametrizable para el control de flujo de trabajo orientado a la administración de proyectos de software: Manejo de proyectos”.

La metodología a utilizar en este trabajo será la de cascada, sin embargo únicamente se llevarán a cabo las siguientes etapas:

- Investigación
    - Recopilación de información en textos e Internet.
        - Flujo de trabajo
        - Control de proyectos
    - Investigación de metodologías de desarrollo de software.
        - Ciclo de vida de software (Tradicional)
        - FDD (Feature-Driven Development)
        - XP (Extreme Programming)
    - Definición de requerimientos del módulo de administración.

- Análisis
    - Descripción general del módulo de administración.
    - Análisis de requerimientos del módulo de administración.
    - Diseño de diagramas de flujo de datos y entidad-relación.
    - Análisis de la estructura para la definición del flujo de trabajo.
    - Análisis de la estructura para la definición de procesos y subprocesos.

- Diseño
    - Diseño de diagramas UML (Casos de uso, clases y secuencia).
    - Diseño y descripción del modelo de base de datos.
    - Diseño y descripción de interfaces gráficas.

II. OBJETIVOS

A. Generales

1. Investigar la situación actual del desarrollo de software y las metodologías más utilizadas en la industria.

2. Evaluar tres diferentes metodologías utilizadas en el desarrollo de software y extraer sus características en común.

3. Diseñar una herramienta que permita la creación de un flujo de trabajo específico aplicable para cualquier empresa de desarrollo de software, tomando en cuenta las metodologías evaluadas.

4. Diseñar el módulo de administración que refleje los procesos reales de una empresa a través de la definición de un flujo de trabajo.

B. Específicos

1. Investigar a fondo tres metodologías de desarrollo de software: tradicional, FDD (Feature-Driven Development) y XP (Extreme Programming).

2. Permitir la definición de la estructura y funcionalidad de procesos y subprocesos en el flujo de trabajo dentro del diseño del sistema.

3. Diseñar la estructura para la definición del flujo de trabajo general.

4. Diseñar la estructura para la definición de procesos dentro de un flujo de trabajo.

III. MARCO TÉORICO

A. Análisis y diseño

1. Concepto de Análisis.

«Distinción y separación de las partes de un todo hasta llegar a conocer sus principios o elementos» (RAE-Diccionario de la Real Academia Española 2006).

«Estudio, mediante técnicas informáticas, de los límites, características y posibles soluciones de un problema al que se aplica un tratamiento por ordenador» (RAE-Diccionario de la Real Academia Española 2006).

Hablando específicamente del área de informática, el término análisis se refiere a un estudio exhaustivo del problema a resolver. Dicho estudio es la fase preliminar de cualquier proyecto de software y es sumamente importante, pues de esta fase depende el desarrollo y la evolución del proyecto.

En el análisis debe incluirse toda la información relevante del problema, desde su naturaleza hasta los posibles problemas que pueden surgir en su implantación. Deben contemplarse todos los requerimientos especificados por el cliente y enfocar la solución del problema a estos requerimientos, indicando qué es lo que se va a hacer para resolver el problema, sin entrar en detalles (Joyanes *et al.* 1998).

Desde los inicios del análisis de sistemas, han surgido numerosas técnicas y herramientas para hacer este proceso más eficiente. En este trabajo se describen dos enfoques a utilizar en el desarrollo del mismo: análisis estructurado y análisis orientado a objetos.

2. Concepto de diseño.

El diseño es el proceso mediante el cual se especifica cómo se realizará lo establecido en la fase de análisis, es decir, los pasos a seguir. Generalmente se incluyen varios tipos de diagramas según el enfoque, para simplificar la visión del problema.

La idea del diseño es realizar un modelo que incluya tanto la definición del problema así como la solución propuesta. Dicho modelo debe asegurar que la solución cumple con los requerimientos y expectativas del cliente, siempre tomando en cuenta la opinión de éste antes de avanzar a la siguiente etapa que sería la codificación (Joyanes *et al.* 1998).

Independientemente de la técnica que se utilice para construir el diseño de un sistema, éste debe representar siempre la mejor solución y debe especificar qué recursos se necesitan para llegar a ella. Al igual que con el análisis, se presentan dos enfoques para el diseño: estructurado y orientado a objetos.

3. Análisis y diseño estructurado. El objetivo del análisis estructurado es organizar las tareas asociadas con la determinación de requerimientos para comprender completa y exactamente una situación dada. Especifica lo que se requiere que haga el sistema o la aplicación y no establece como se cumplirán los requerimientos o la forma de implantarlos.

Además, permite que las personas observen los elementos lógicos separados de los componentes físicos. Después de esto, ya es posible desarrollar un modelo físico eficiente según la situación en donde será utilizado (Yourdon 1989).

El modelo de análisis estructurado debe:

- Describir las necesidades del cliente.
- Establecer las especificaciones internas.
- Definir un conjunto de requisitos que se puedan validar una vez que se ha construido el software.
- Obtener la aprobación del cliente.

Los siguientes diagramas se incluyen en el modelo de análisis estructurado:

a. Diccionario de datos. Se define como un catálogo de los elementos en un sistema: datos consumidos y producidos por software. Estos elementos se centran en los datos y en la forma en que están estructurados para satisfacer los requerimientos de los usuarios. Tienen características lógicas de los sitios donde se

almacenan los datos del sistema incluyendo nombre, descripción, alias, contenidos y organizaciones (Yourdon 1989).

El diccionario tiene dos tipos de descripciones para el flujo de datos dentro del sistema:

1) *Elementos de datos.* Son bloques básicos para todos los demás datos del sistema.

2) *Estructura de datos.* Grupo de datos elementales relacionados con otros y que en conjunto describen un componente del sistema (Yourdon 1989).

b. Diagrama entidad-relación. Representa las relaciones entre entidades de datos. Los atributos de cada entidad se pueden describir mediante la descripción de datos. Un diagrama de este tipo tiene cuatro componentes:

1) *Tipos de objetos.* Representan una colección de objetos en el mundo real cuyos miembros pueden ser identificados unívocamente y descritos por uno o más elementos de información.

2) *Relaciones.* Representan un conjunto de conexiones entre objetos.

3) *Indicadores de tipo de objeto asociativos.* Se refieren a relaciones para las cuales se desea almacenar información.

4) *Indicadores de super-tipo y sub-tipo.* Consisten de un tipo de objeto y una o más subcategorías, conectadas por medio de una relación (Date 1993, Korth *et al.* 1986).

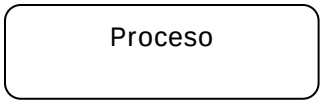
c. Diagrama de flujo de datos (DFD). Indica cómo se transforman los datos a medida que se avanza en el sistema; y representa las funciones que transforman el flujo de datos.

Este es uno de los diagramas más utilizados en el modelado de sistemas, especialmente para sistemas donde las funciones a realizar son más complejas que la información a manipular por el sistema (Yourdon 1989).

Los componentes de un DFD son:

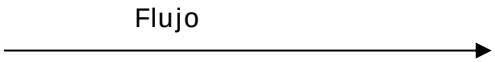
1) *Proceso.* También denominado función o transformación, representa una parte del sistema que convierte una entrada en una salida. Un proceso se representa con la siguiente notación:

Ilustración 1. Representación de un proceso en un diagrama DFD.



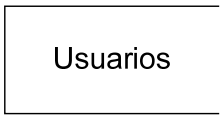
2) *Flujo.* Describe el movimiento de paquetes de información de una parte del sistema a otra. Se representa a través de una flecha dirigida junto con una breve descripción de la información que está fluyendo:

Ilustración 2. Representación del flujo en un diagrama DFD.



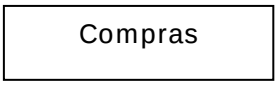
3) *Almacenamiento.* Se utiliza para modelar una colección de paquetes de información estáticos. Su representación gráfica es un cuadrado sin el borde derecho:

Ilustración 3. Representación del almacenamiento en un diagrama DFD.



4) *Terminador.* Se refiere a una entidad externa con la cual el sistema se comunica. Se representa a través de un rectángulo de la siguiente forma:

Ilustración 4. Representación del terminador en un diagrama DFD.



d. Diagrama de transición de estados (DTE). Indica cómo se comporta el sistema como consecuencia de sucesos externos. Los sistemas que se modelan utilizando estos diagramas son muy dinámicos y deben proveer respuestas rápidas para ser capaces de manejar su entorno (Yourdon 1989). Es importante preguntarse la interrogante: "qué sucede cuándo…".

Este diagrama consta de las siguientes partes:

1) *Estados del sistema.* «*Conjunto de circunstancias o atributos que caracterizan a una persona o cosa en determinado tiempo; forma de ser; condición*» (Webster's New World Dictionary). Un estado especifica la etapa en el que está el sistema en ese momento.

2) *Condiciones y acciones.* Las condiciones causan un cambio en un estado y las acciones se refieren a las acciones que el sistema toma cuando éste cambia de estado.

4. Análisis y diseño orientado a objetos.

En vista que el análisis estructurado presentaba algunas deficiencias, como por ejemplo: descomposición funcional, flujo de datos y modelado de datos, se desarrolló el Análisis Orientado a Objetos (AOO) aproximadamente en los años 80's.

El uso de la programación orientada a objetos (POO) ha modificado las técnicas de diseño para adaptarlas a los nuevos lenguajes y tiene como objetivo modelar el mundo real. Para esto, abstrae las características importantes del problema dejando los detalles para más adelante (Wang 2001).

El AOO ofrece un enfoque nuevo para el análisis de requisitos de sistemas de software. En lugar de considerar el software desde una perspectiva clásica de entrada/proceso/salida, como los métodos estructurados clásicos, se basa en modelar el sistema mediante los objetos que forman parte de él y las relaciones estáticas (herencia y composición) o dinámicas (uso) entre estos objetos. Este enfoque pretende conseguir modelos que se ajusten mejor al problema real, a partir del conocimiento del llamado dominio del problema. Desde este punto de vista, el AOO consigue una abstracción mayor que el análisis estructurado (Bray 2005).

En el AOO, se dice que un objeto es una representación o abstracción de una entidad que existe en el mundo real y que captura la estructura estática de un sistema. Además, para cada objeto debe existir información que defina su estructura y comportamiento: nombre, atributos y relaciones.

El AOO incluye entre sus características más importantes la definición de clases en el sistema, así como la especificación de los atributos y servicios de cada clase. Una clase se define como un conjunto de objetos con propiedades, atributos y comportamiento en común, con un conjunto de operaciones a realizar sobre los objetos. En este caso, dichas operaciones se refieren a los métodos o los servicios que la clase provee para sus instancias (Joyanes *et al.* 1998).

El diseño orientado a objetos (DOO) se enfoca en desarrollar un modelo orientado a objetos para implantar los requerimientos identificados en el análisis. Es decir, se redefinen los objetos a clases, se definen protocolos de mensajes para todos los objetos, se definen estructuras de datos y procedimientos y se trasladan a un lenguaje de POO.

El DOO puede dividirse en dos etapas:

a. Diseño de alto nivel. Se encarga de la descomposición del sistema en objetos grandes y complejos. No se toman en cuenta detalles.

b. Diseño de bajo nivel. Se especifican atributos y métodos para cada objeto definido en el alto nivel.

Este nuevo enfoque tiene varios beneficios: fácil mantenimiento debido al simple traslado de los requerimientos del mundo real a un modelo de objetos, reutilización de artefactos de análisis o diseño y aumento de la productividad, pues el traslado de dichos artefactos a un lenguaje de programación es relativamente fácil y rápido.

En cuanto a la notación para este tipo de análisis y diseño, la más común y más conocida es UML *(Unified Modelling Language),* la cual provee una gran diversidad de diagramas que representan las diferentes vistas y estados de un sistema (Wang 2001). Los fundamentos de este lenguaje serán detallados en el apéndice I.

B. Procesos

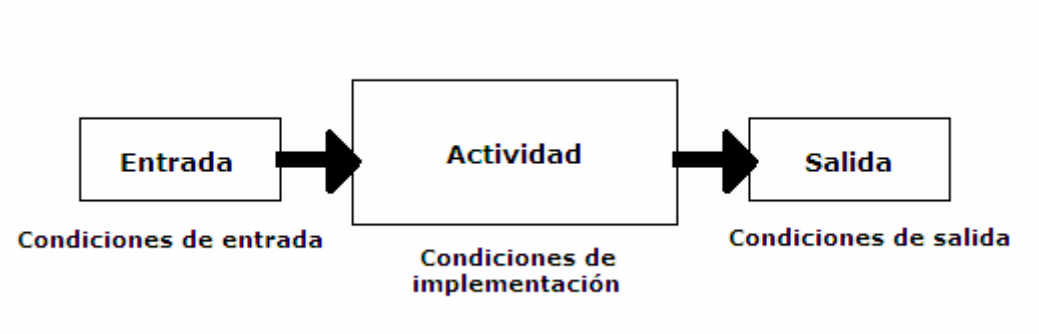
1. Concepto.

Proceso: «*Conjunto de las fases sucesivas de un fenómeno natural o de una operación artificial*» (RAE-Diccionario de la Real Academia Española 2006).

En términos generales, un proceso se define como un conjunto de actividades ordenadas y secuenciales que se realizan para cumplir un objetivo específico. En el ámbito del desarrollo de software, un proceso se refiere al conjunto de actividades que se realizan para obtener un producto de software, es decir, una aplicación funcional que será entregada a un cliente.

Cada actividad que se lleva a cabo en el proceso tiene una entrada y una salida las cuales deben de estar muy bien definidas. La entrada también se conoce como *condiciones de entrada* y se refiere a las condiciones que deben satisfacerse antes de iniciar alguna actividad. De manera similar se definen las *condiciones de salida,* las cuales deben cumplirse antes de que se pueda dar por concluida una actividad. Adicionalmente a estas condiciones, existen las *condiciones de implantación,* que son los pasos que explican qué es lo que debe hacerse y cómo debe hacerse (Whitten 1995). La Ilustración 5 muestra el modelo de una actividad que forma parte de un proceso.

Ilustración 5. Modelo de actividad.



2. Tipos de procesos

Se pueden definir tres tipos de procesos:

a. Procesos estratégicos. Son los que establecen las normas a seguir para los demás procesos, definen las estrategias y objetivos de una empresa. Este tipo de procesos está vinculado con la visión de la empresa.

b. Procesos fundamentales. Son los procesos relacionados con los clientes, pues de una u otra manera le ofrecen un valor agregado, e inciden directamente en su satisfacción. Estos procesos también pueden consumir muchos recursos y son los que se relacionan con la misión de una empresa.

c. Procesos de soporte. Sirven para dar apoyo a los procesos fundamentales. Son también los procesos necesarios para control, pero que no son estratégicos ni fundamentales; puede ser cualquier proceso relacionado con modelos de administración. Estos procesos no tienen relación con la visión ni con la misión de la empresa.

Independientemente del tipo de proceso a llevar a cabo, se tiene aún una clasificación más:

a. Procesos industriales. En donde existe un flujo de materiales.

b. Procesos de administración. En donde existe un flujo de información, tal es el caso de los procesos de desarrollo de software (Manual de Diseño de Procesos, UMH).

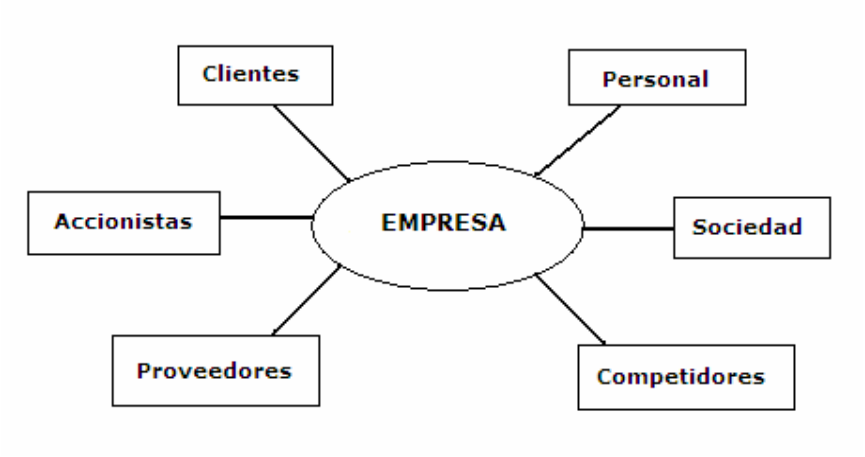
3. Administración y reingeniería de procesos

«*El enfoque basado en procesos consiste en la identificación y gestión sistemática de los procesos desarrollados en la organización y en particular las interacciones entre tales procesos*» (ISO 9000:2000).

El concepto de administración se refiere a la actividad mediante la cual se manejan todos los recursos de una empresa, así como sus procesos relacionados: cómo se planifican, cómo se organizan, cómo se coordinan y cómo se ejecutan. La administración de procesos tiene como fin modelar los sistemas de una empresa

como un conjunto de procesos relacionados con el fin de alcanzar un objetivo, el cual debe ser asegurar que dichos procesos se lleven a cabo de una manera eficiente y eficaz, satisfaciendo siempre las necesidades de todas las partes involucradas en una empresa. La Ilustración 6 muestra un diagrama con las entidades con las que interactúa una empresa.

Ilustración 6. Partes relacionadas a una empresa.



Para que la administración de procesos en una empresa sea exitosa deben tomarse en cuenta tres aspectos muy importantes:

- Se deben «*Identificar los procesos necesarios para el sistema de gestión de la calidad y su aplicación a través de la organización*» (Apartado 4.1.a, ISO 9000:2000).

- Se requiere «*Determinar la secuencia e interrelación de estos procesos*» (Apartado 4.1.b, ISO 9000:2000).

- Se matiza que «*La organización debe planificar y desarrollar los procesos necesarios para la realización del producto*» (Apartado 7.1, ISO 9000:2000).

Una característica que identifica a la administración de procesos es su enfoque en el resultado de los procesos y no específicamente en las tareas o actividades que se llevan a cabo. Para esto, debe existir una forma de asignar responsabilidades por cada proceso a ejecutar y cada responsable debe saber cómo su trabajo contribuirá al sistema global, enfocándose siempre en los resultados finales.

La reingeniería de procesos ha tenido una rápida expansión en las últimas décadas debido al impacto que ha logrado causar en muchas empresas. Se define como una revisión de los procesos estratégicos de una empresa y que tiene como fin obtener mejoras en aspectos críticos como costos, calidad, rapidez y servicio (Schuldt 1998).

La reingeniería se divide en cinco fases:

a. Organización e inicio. Debe tomarse la decisión de organizar un comité de reingeniería y deben asignarse equipos para el mejoramiento de procesos estratégicos específicos.

b. Identificación de procesos. Se debe realizar un análisis exhaustivo de todos los procesos que se llevan a cabo en la empresa, identificando cuáles son los procesos débiles y los fuertes. Debe evaluarse el actual servicio que los clientes reciben en todo sentido: velocidad de entrega de producto/servicio, calidad, acceso, puntualidad, consistencia y seguridad.

c. Desarrollo de la reingeniería. Está formado por tres etapas:

1) *Rediseño.* Deben identificarse todos los procesos en la empresa: integración de otros subprocesos, duración de cada uno, posibles errores y mejoras, variaciones de comportamiento y complejidades especiales. En esta etapa debe tomarse en cuenta el valor que cada cliente da a cada proceso, si es que el cliente forma parte de él y posteriormente redefinir todos los aspectos que deseen mejorarse en base al análisis previamente realizado y a los objetivos que se desean obtener.

2) *Revisión de procesos.* Debe analizarse el funcionamiento actual de cada proceso, así como la visión, misión y objetivos específicos de cada uno.

3) *Cambios en la organización.* Implantación de los cambios definidos en el rediseño; se definen las responsabilidades de trabajo para cada proceso y las relaciones entre cada una.

d. Seguimiento y control. Deben establecerse, entre otros, puntos de control e indicadores de rendimiento, eficiencia y satisfacción del cliente.

e. Mejoramiento continuo. Para obtener los resultados deseados en todos los procesos, debe darse capacitación a todas las personas involucradas, promoviendo un mejoramiento continuo y una evaluación periódica y permanente de todos los procesos.

4. Beneficios de la administración y reingeniería de procesos

a. Mayores beneficios económicos. Reducción de costos e incremento de rendimientos.

b. Mayor satisfacción al cliente. Mejora de calidad del producto/servicio, flexibilidad a sus necesidades.

c. Mayor satisfacción del personal. Mejor definición y asignación de tareas.

d. Mejor control de los procesos. Mejor flujo de información y disminución de tiempos de procesos.

Todo lo anteriormente expuesto tiene un enfoque totalmente orientado a un ambiente gerencial y administrativo; ahora bien, hablando específicamente de desarrollo de software, la aplicación puede considerarse bastante similar. Lo único que cambia es la naturaleza de los procesos mismos. A continuación se describen ocho pasos para definir un proceso de desarrollo de software.

5. Pasos para definir un proceso de desarrollo de software.

Antes de desarrollar un proyecto de software, es necesario contar con alguna técnica que permita controlar los diversos aspectos que conforman un proyecto: tiempo, costos y calidad (Whitten 1995). Con este fin, se definen a continuación los siguientes pasos:

a. Identificar modelo de desarrollo de software. Como primer paso, debe definirse el modelo de software que se utilizará para la definición del proceso. Más adelante, se presentan tres distintos modelos o metodologías a utilizar en el presente trabajo: ciclo de vida tradicional, y dos metodologías ágiles: *FDD (Feature-Driven Development) y XP (Extreme Programming)*. Como ya se mencionó, se obtendrán los aspectos más relevantes de cada una.

b. Identificar actividades. Posteriormente deben identificarse las actividades principales de la empresa que deben ser implantadas, junto con una descripción; todo esto con el fin de definir una visión global del proceso.

c. Identificar relaciones entre actividades. Para establecer cómo se relacionan las diferentes actividades, deben listarse las condiciones de entrada y de salida para cada actividad; con esto, será posible identificar la dependencia entre actividades.

d. Documentar información de cada actividad. Por cada actividad a realizar, debe incluirse cualquier información adicional que sea considerada como útil. A esta información se le dará un uso posterior. Con esto se completan las cuatro secciones que una actividad debe incluir: condiciones de entrada, condiciones de salida, descripción y notas o información adicional.

e. Documentar como construir el proceso. Ya con la información relevante de todas las actividades que componen un proceso se debe proceder a *construirlo.* Deben establecerse las reglas para su construcción. Es necesario especificar las actividades que serán obligatorias, las que pueden eliminarse y las que no, si algunas actividades pueden combinarse y si es posible agregar nuevas actividades complementarias.

f. Documentar cómo mejorar el proceso. Ya con las reglas definidas debe asegurarse la existencia de alguna forma escrita que permita mejorar el proceso continuamente. Aquí, deben tomarse en consideración dos casos: sugerencias para cambios en el proceso y cambios únicos e irrepetibles. En cualquiera de los casos, debe estar documentado cómo proceder en cualquiera de las situaciones.

g. Obtener retroalimentación del proceso. Hasta este momento, el proceso está completamente definido pero aun hacen falta dos aspectos importantes: todos los miembros involucrados de la empresa deben aprobar el nuevo proceso recién definido y cada uno de ellos debe ser capacitado para el correcto uso del mismo.

h. Utilizar y mejorar continuamente el proceso. En este punto, el proceso ya puede ser utilizado, tomando en cuenta que siempre puede cambiarse y mejorarse, es por esto que en el paso f debe dejarse por escrito el procedimiento a seguir. Es sumamente importante que todos los miembros de la empresa involucrados comprendan la noción de mejorar el proceso, antes de que este sea aprobado finalmente (Whitten 1995).

c. Flujo de trabajo

1. Concepto. El flujo de trabajo se encarga de estudiar los aspectos operacionales de una empresa; específicamente, todas las actividades realizadas desde su estructura, sincronización, asignación de tareas, seguimiento, etc. Un flujo de trabajo está compuesto de una serie de procesos, subprocesos y tareas, todos ordenados según reglas previamente establecidas. Dentro del flujo, existe información que está tanto relacionada a las actividades de la empresa así como a los individuos responsables de completar cada etapa del flujo.

El flujo es responsable tanto de definir el orden en que deben ser realizadas todas las fases para obtener un objetivo, así como de asegurar que cada individuo reciba la información necesaria para su respectiva tarea y que sea capaz de transmitirla a los responsables de la siguiente etapa.

2. Automatización del flujo de trabajo. Actualmente, el flujo de trabajo está tomando un nuevo rumbo: la automatización de procesos. La acción de convertir el flujo de trabajo en un proceso automático es cada vez más común y cada vez más hay usuarios que desean contar con una herramienta que les permita de una forma fácil, definir su propio flujo de trabajo (Hollingsworth 1995).

Antes de considerar un flujo de trabajo automatizado en una empresa, es necesario seguir ciertos pasos que aseguren su correcta implantación: evaluación y análisis de los métodos de trabajo actualmente utilizados en la empresa, creación de un modelo que se ajuste a las necesidades de la empresa, desarrollo e implantación del modelo de una forma fácil de utilizar y una vez más, la evaluación de la nueva forma de trabajo (Hollingsworth 1995).

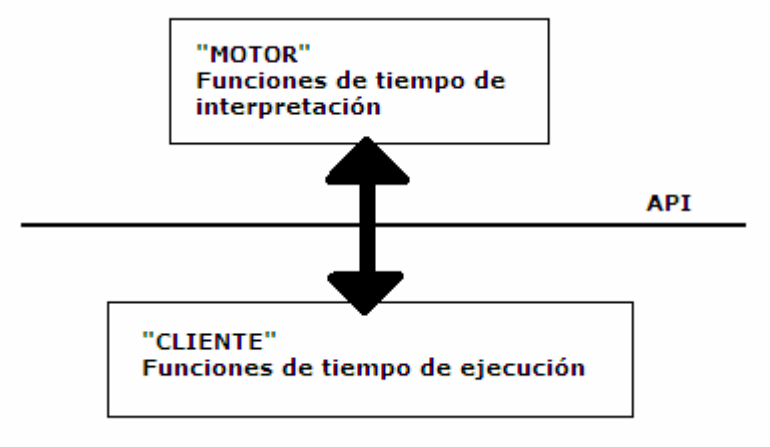
Un sistema para la administración del flujo de trabajo debe, en términos generales, proveer un mecanismo tanto para la automatización de procesos, así como para la administración de todas las actividades que se llevan a cabo en la empresa. Específicamente, debe proveer dos tipos de funciones:

a. Funciones de tiempo de interpretación. Definen y modelan el proceso del flujo de trabajo de acuerdo a las necesidades y a la estructura de la empresa. En estas funciones se evalúan los procesos actuales y se trasladan del mundo real a una representación formal computarizada; esto incluye la definición de un conjunto de reglas y operaciones que definen el progreso del proceso. Estas funciones conforman el "motor" o núcleo del sistema para el control de flujo de trabajo y este es el tema principal del presente trabajo.

b. Funciones de tiempo de ejecución. Estas funciones interpretan las definiciones de procesos previamente establecidas y son responsables de crear y controlar instancias operacionales de un proceso, es decir, crear instancias para nuevos proyectos, asignando recursos como lo pueden ser tiempo y personal, adicionalmente programando actividades. Dichas funciones forman parte de módulo de usuario o "cliente", a ser diseñado por Mariana Rendón en el trabajo titulado: "Análisis y diseño de un sistema parametrizable para el control de flujo de trabajo orientado a la administración de proyectos de software: Manejo de proyectos".

La interacción entre ambos tipos de funciones debe hacerse por medio de una interfaz (API) que permita la identificación de actividades y recursos, asignación de recursos, estructuras y referencias, y la definición de mecanismos de comunicación (Hollingsworth 1995). La Ilustración 7 muestra el esquema general del funcionamiento de un sistema para la administración del flujo de control.

Ilustración 7. Sistema para la administración de flujo de control.



3. Beneficios. El uso de una herramienta automatizada para definir el flujo de trabajo de una empresa tiene múltiples beneficios:

- Permite la especialización del trabajo, es decir, los administradores no tienen que preocuparse por el proceso de la asignación de tareas, pues esto ya se hace automáticamente según algún criterio definido, enfocándose de lleno en asuntos relacionados con el personal y los negocios exclusivamente.

- Con la automatización de los procesos se mejora la eficiencia, pues se eliminan procesos innecesarios, así como también se establece un estándar para los métodos de trabajo en la empresa.

- En todo momento se cuenta con documentación específica que indica cómo llevar a cabo determinada tarea, convirtiéndose en un entorno de trabajo pro-activo en lugar de reactivo.

- Permite el procesamiento en paralelo, es decir, varios procesos pueden ser realizados al mismo tiempo, así como la definición de avisos o mensajes a las personas involucradas en un proceso.

- Facilita a los administradores o supervisores obtener una vista en cualquier momento de un proyecto dentro del flujo y obtener de manera más rápida la ubicación de un proceso o subproceso específico dentro de ese flujo.

- Es una herramienta flexible pues tiene la capacidad de modificar el flujo en cualquier momento y cualquier proyecto dentro del flujo, de esta manera adaptándose a las necesidades cambiantes de la empresa y permitiendo evaluar la eficiencia de los cambios.

- Genera beneficios en cuanto a la reducción de tiempo de ejecución de procesos, reducción de costos, incremento de eficiencia y mejoramiento del servicio al cliente en general.

D. Metodologías

1. Concepto de metodología.

La Real Academia Española (2006) define una metodología como el «*Conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal*».

En un sentido más general, una metodología es un conjunto de procedimientos, técnicas y herramientas que se utilizan para llevar a cabo una tarea específica. En el desarrollo de software una metodología es un proceso disciplinado cuyo objetivo es hacer que el desarrollo sea mucho más predecible y eficiente. Para esto, una metodología puede estar compuesta por un conjunto de objetos: mejores prácticas, materiales de capacitación y herramientas de diagramación entre otros. Todas las metodologías pretenden lograr sus objetivos haciendo un fuerte énfasis en la planificación de fases descompuestas en sub-fases. Esta descomposición guía a los desarrolladores en la elección de las técnicas a utilizar para cada estado del proyecto, facilitando así la planificación, administración, control y evaluación de los proyectos. Una metodología es entonces, el camino a seguir para desarrollar aplicaciones sistemáticamente (Palacio 2005).

A continuación se describen las tres metodologías tomadas en cuenta para el desarrollo del presente trabajo: ciclo de vida de software tradicional y dos de las metodologías del desarrollo ágil más conocidas: XP y FDD.

2. Ciclo de vida. Se define el ciclo de vida del software como:

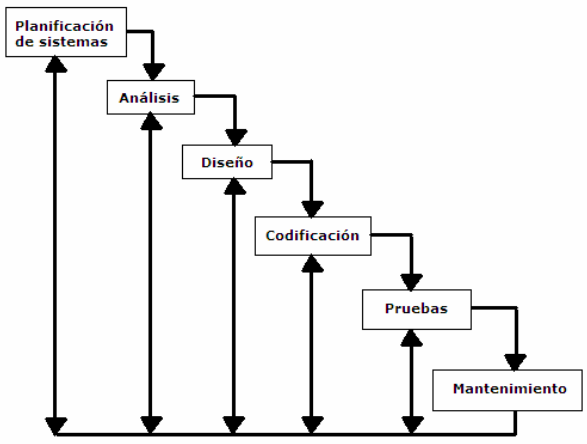
- «*Una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software*» (IEEE 1074).
- «*Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso*» (ISO 12207-1).

Existen diversas razones para contar con un ciclo de vida para los proyectos que se llevan a cabo en una empresa de desarrollo:

- Definir las actividades que se llevarán a cabo en el desarrollo de un proyecto.
- Introducir consistencia entre los diversos proyectos en la misma empresa.
- Proveer puntos de control para una mejor administración.

Con esto debe dejarse claro que un ciclo de vida definido en la empresa no será el encargado del proyecto; es decir, el ciclo de vida no tomará las decisiones importantes que el administrador debe tomar, no medirá alternativas ni negociará con los clientes. Su función se limitará a definir un estándar para la forma de trabajo de la empresa y ayudar a organizar todas las actividades relacionadas. El ciclo de vida tradicional o clásico también se denomina de cascada, debido a la forma en que se relacionan sus etapas, esto se muestra en la Ilustración 8.

Ilustración 8. Ciclo de Vida Tradicional



Este modelo divide el ciclo de vida de software en una serie de actividades sucesivas donde cada fase requiere información de entrada, procesos y resultados bien definidos.

El modelo se divide en las siguientes fases (Joyanes *et al*. 1998, Yourdon 1989):

a. Planificación del sistema. Esta etapa también se conoce como estudio de factibilidad, pues aquí debe determinarse si un proyecto es factible de realizar o no, determinando tiempos y costos aproximados. Deben identificarse las personas responsables de desarrollar una descripción inicial de lo que será el sistema, esto en la mayoría de casos requerirá de una serie de entrevistas para identificar quienes se verán afectados directamente por el sistema y quienes no. Es esencial también realizar una evaluación de las características y deficiencias del ambiente del sistema actual (si es que existe) y establecer las metas y objetivos del nuevo sistema.

Aunque no lo pareciera, esta etapa es una de las más importantes pues la falta de planeación de un sistema es la causa principal de retrasos en programación, incremento de costos, baja calidad y altos costos de mantenimiento en los desarrollos de productos de software. Usualmente se suele decir que no es factible hacer un plan inicial pues no se tiene la información necesaria sobre el proyecto, necesidades del cliente y restricciones del producto, pero un objetivo principal de esta etapa es identificar los objetivos, problemas y necesidades.

b. Análisis. Durante esta etapa se hace un estudio y una especificación estructurada del sistema. Esto involucra obtener los requerimientos del cliente, el modelado del ambiente del usuario con el uso de diagramas (entidad-relación, flujo de datos, transición) y la documentación de todo lo que se ha investigado y todo lo que se establecerá por hacer. Idealmente se debería incluir un conjunto de presupuestos y cálculos de costo-beneficio al final. Todo lo que aquí se defina debe ser comentado y aprobado con el cliente antes de continuar.

c. Diseño. La fase de diseño se encarga de establecer cómo se realizará todo lo que fue definido durante el análisis. Se debe construir una jerarquía apropiada de módulos e interfaces, de tal forma que se cumplan los requerimientos establecidos con anterioridad. Un aspecto muy importante es la transformación del diagrama

entidad-relación en un diseño de base de datos real. El diseño del software es realmente un proceso que se enfoca sobre cuatro atributos distintos del proyecto: la estructura de los datos, la arquitectura del software, el detalle de los procedimientos y la caracterización de la interfaz. En términos generales, el proceso de diseño traduce los requisitos en una representación del software que pueda ser establecida de manera que obtenga la calidad requerida antes de que empezar la codificación.

d. Codificación. Etapa también conocida como implantación, consiste en la traducción del diseño a un lenguaje de programación y la integración de los módulos previamente definidos en el diseño en un esquema general. Si el diseño se realiza de una manera detallada la codificación puede realizarse mecánicamente.

e. Prueba. Después de terminar la codificación se procede con la realización de una serie de pruebas sobre todos los módulos codificados. Estas pruebas deben centrarse en la lógica interna del software, asegurando que todas las sentencias sean probadas. Las funciones externas se deben validar realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.

f. Mantenimiento. En esta etapa se realizarán cambios, bien ya sea por errores que no se hayan detectado en la fase de prueba, por cambios en el entorno del sistema o por ampliaciones en los requerimientos del cliente.

A pesar de que este ciclo es el más antiguo y a veces el más utilizado, presenta ciertos problemas (Laplante 2004):

- Es difícil seguir la linealidad del ciclo; normalmente el cliente no especifica todos los requerimientos por lo que debe retrocederse a etapas anteriores.
- El cliente no ve una versión del producto hasta que se finaliza el ciclo.
- La mayoría de errores triviales se encuentran al inicio de la etapa de pruebas, mientras que los errores críticos se encuentran al final de dicha etapa.

3. Metodologías ágiles. El denominado desarrollo ágil surgió a mediados de los 90's como parte de una reacción contra los métodos convencionales, al

considerarlos burocráticos, lentos y contradictorios a las formas en las que realmente se realiza el trabajo (Fowler 2005).

Inicialmente, las metodologías ágiles fueron denominadas de *peso liviano* en contraste con las metodologías convencionales llamadas de *peso pesado*. En el año 2001, un grupo de personas influyentes adoptaron el nombre de *ágiles*, formando luego una organización no lucrativa que promueve el desarrollo ágil llamada la *Alianza Ágil (Agile Alliance).* De esta reunión surgió el denominado *Manifiesto Ágil (The Agile Manifesto),* el cual es un documento donde se especifican los principios básicos del desarrollo ágil (Highsmith 2001):

«*Estamos descubriendo mejores formas de desarrollar software, haciéndolo y ayudando a otros a hacerlo. A través de este trabajo, hemos llegado a valorar:*

> ***Individuos e interacciones*** *sobre procesos y herramientas*
> ***Software funcional*** *sobre documentación comprensiva*
> ***Colaboración de los clientes*** *sobre negociación por contrato*
> ***Responder al cambio*** *sobre seguir un plan*

Esto es, aunque existe valor en los objetos de la derecha, valoramos más los objetos de la izquierda» (The Agile Manifesto, ver Anexo C: Prinpicios detrás del Manifiesto Ágil).

La diferencia más grande entre ambos tipos de metodologías es que las ágiles son mucho menos orientadas a la documentación, pues generalmente son más orientadas a la codificación y a la comunicación verbal, estableciendo así una ruta que indique que la parte más importante de la documentación es el código fuente. Adicionalmente, se dice que las metodologías ágiles son adaptantes en lugar de predecibles; en otras palabras, se adaptan fácilmente a las realidades cambiantes de un sistema y con su uso, ya que es difícil describir exactamente qué va a pasar en el futuro, debido a que el cambio se considera parte inherente del desarrollo (Fowler 2005).

Las metodologías pertenecientes al movimiento ágil pretenden minimizar los riesgos de las metodologías tradicionales desarrollando un sistema en pequeñas partes denominadas iteraciones, las cuales tienen una duración de una a cuatro semanas. Cada iteración es una especie de proyecto en miniatura que incluye todas las etapas de desarrollo: planificación, análisis de requerimientos, diseño, codificación, pruebas y documentación entre otras. Uno de sus principales objetivos

es la “liberación” [1] de software funcional al final de cada iteración, enfatizando el software funcional como una medida del progreso del proyecto (Ambler 2006, Fowler 2005).

A continuación se describen dos de las metodologías ágiles más conocidas: *XP (Extreme Programming) y FDD (Feature-Driven Development).*

4. XP (Extreme Programming). Aunque no fue la primera metodología ágil desarrollada, definitivamente estableció la popularidad del desarrollo ágil. Fue creada en 1996 por Kent Beck y posteriormente se definió como:

- «un mecanismo para cambio social»
- «un estilo de desarrollo»
- «un camino al mejoramiento»
- «un intento por reconciliar la humanidad y productividad»
- «una disciplina de desarrollo de software»

La base del desarrollo XP es la importancia de las pruebas (testing), en donde cada desarrollador escribe pruebas y las ejecuta al terminar de escribir su código fuente. Las pruebas deben ser integradas continuamente en un proceso que genere una plataforma estable que permita proseguir con el desarrollo posterior. Sobre esta plataforma, XP construye un proceso evolutivo de diseño que se basa en obtener un sistema funcional en cada iteración. Una característica especial del desarrollo XP es que el diseño se centra en todo momento en la iteración actual, pues no se hace ningún diseño anticipando necesidades futuras (Jeffries 2001).

El objetivo principal de XP es minimizar el costo del cambio. Como ya se mencionó, en el desarrollo tradicional los requerimientos son tomados al inicio y generalmente se arreglan desde ese punto en adelante; esto quiere decir que el costo de cambiar los requerimientos en un punto más adelante en el desarrollo de un proyecto será muy alto. El objetivo se logra introduciendo valores básicos, prácticas y principios que a continuación se describen.

Prácticas base:

[1] Liberación de software (release) se refiere al punto en el que el código se considera de suficiente calidad como para ser presentado al usuario final y afirmar que ya no se necesita realizar trabajo adicional a este segmento de código.

- Retroalimentación (Feedback)
    - Desarrollo orientado a pruebas
    - Juego de planificación
    - Equipo de trabajo completo
    - Programación en paralelo (Jensen *et at.* 2003)
- Proceso continuo
    - Integración continua
    - Mejoramiento del diseño
    - Pequeñas “liberaciones” del sistema

- Comprensión compartida
    - Diseño simple
    - Metáfora del sistema
    - Propiedad colectiva del código
    - Estándares y convenciones de codificación

Estas prácticas se derivan de mejores prácticas generalmente aceptadas y son llevadas a sus extremos:

- *«La interacción entre desarrolladores y clientes es buena. Por lo tanto, un equipo XP debe contar con la presencia de un cliente, quien especifica y asigna prioridades al equipo y puede responder a las preguntas tan pronto como vayan surgiendo».*

- *«Si el aprendizaje es bueno, llevarlo a los extremos: reducir el tiempo de desarrollo y ciclos de regeneración, hacer pruebas tempranamente».*

- *«El código simple es más propenso a funcionar. Por lo tanto, los programadores extremos solo escriben código para satisfacer las necesidades reales del proyecto en el tiempo presente y van más allá para reducir complejidad y duplicaciones en su código».*

- *«Si el código simple es bueno, este se debe reescribir cuando se vuelve complejo».*

- *«Las revisiones de código son buenas. Por lo tanto los programadores XP trabajan en parejas compartiendo una pantalla y una teclado, a manera de que todo el código se revisa mientras se escribe».*

- *«Las pruebas al código son buenas. Por lo tanto en XP las pruebas son escritas antes de que el código sea escrito. El código se considera completo*

cuando este satisface las pruebas. El sistema es periódicamente probado utilizando todas las pruebas preexistentes para asegurar que funciona».

XP también cuenta con cinco valores que son parte integral de su funcionamiento (Marchesi 2005):

a. Comunicación. Se refiere a la transferencia de los requerimientos del sistema a los desarrolladores a través de comunicación verbal frecuente en lugar de documentación escrita.

b. Simplicidad. Establece que siempre debe iniciarse con la solución más simple y luego ir desarrollando soluciones mejores, siempre enfocándose en los esfuerzos de diseño y codificación en las necesidades presentes ya que si se hace para necesidades futuras, se corre el riesgo de gastar recursos en algo que puede no ser necesario.

c. Retroalimentación (Feedback). Se refiere a las diversas respuestas que se pueden obtener del sistema, del cliente y del equipo de desarrollo y las acciones a tomar en caso de respuestas no esperadas.

d. Coraje. Se refiere a la actitud que los programadores deben tener para revisar el sistema existente y modificarlo según se requiera.

e. Respeto. Tiene diferentes enfoques; respeto por los otros miembros del equipo y por uno mismo, buscando siempre la solución con el mejor diseño y la más alta calidad.

La metodología XP incluye cuatro actividades básicas que son la base del proceso de desarrollo de software (Wells 1999):

a. Codificar. Se resalta la importancia del código, no importando su función y representación.

b. Probar. Debe asegurarse que una función produce los resultados esperados por medio de una serie de pruebas.

c. Escuchar. Se refiere a la actividad que debe realizarse por los desarrolladores en cuanto al lado administrativo del desarrollo del sistema, obteniendo las necesidades de los clientes y los consejos de los administradores.

d. Diseñar. Establece el valor de definir una buena estructura de diseño que organice la lógica del sistema, de tal forma se eviten dependencias en partes estratégicas del sistema.

5. FDD (Feature-Driven Development). FDD es una metodología de desarrollo ágil creada por Jeff Deluca con el fin de obtener un sistema funcional en tiempos relativamente cortos. La base de esta metodología es el concepto de *característica (feature)* el cual se define como una pieza de funcionalidad valorada por el cliente (Bauer 2005).

Dentro de esta metodología existen cinco procesos muy bien establecidos que definen las actividades a llevar a cabo:

a. Desarrollo de un modelo general. Involucra la creación de un modelo del dominio del negocio, haciendo énfasis en su forma más que en su contenido, es decir, el modelo no debe estar complemente detallado puesto que esto se hace en un proceso posterior.

b. Construir una lista de características. Pretende que se identifiquen todas las características necesarias para apoyar los requerimientos del proyecto. Lo más importante en este proceso es utilizar el lenguaje del dominio de negocios o un lenguaje entendible por el cliente de tal manera que él sea capaz de comprender y valorar las características escritas utilizando este lenguaje. FDD propone un estándar de sintaxis para la definición de las características de la siguiente forma:

<*Acción*> el/la/los <*Resultado*> para/por/a <*Objeto*>

Un ejemplo sería: *Calcular* el *Total* para *Ventas*

Una característica también se define en términos de duración (más de dos horas pero menos de dos semanas de trabajo). Si se estima que una característica tomará

más de dos semanas, esta debe ser dividida en características más pequeñas y trabajarlas independientemente. Es importante destacar que el uso de un lenguaje común entre desarrolladores y clientes no elimina el riesgo de cometer errores y advertir confusiones, pero si lo minimiza considerablemente.

c. Planificación. Se refiere al orden en que las características deben ser construidas junto con la asignación del personal responsable para cada una. Adicionalmente deben establecerse estrategias para obtener resultados que satisfagan las necesidades del cliente.

d. Diseño por características. Involucra el proceso de definir detalladamente cada característica. El diseño está dividido en tres etapas:

1) *Familiarización.* Los desarrolladores deben familiarizarse y habituarse con lo que diseñarán, deben formarse una idea general sin entrar a detalles específicos de diseño.

2) *Diseño.* Se refiere propiamente a la descripción del funcionamiento de cada característica y la forma en que será desarrollado.

3) *Inspección.* La revisión o inspección del diseño permite descubrir errores o defectos antes de empezar la codificación de cada característica.

e. Construcción por características. Se refiere no sólo a la codificación de cada característica, sino también a la acción de decidir cuando una característica está completamente terminada. Este proceso también está dividido en tres etapas:

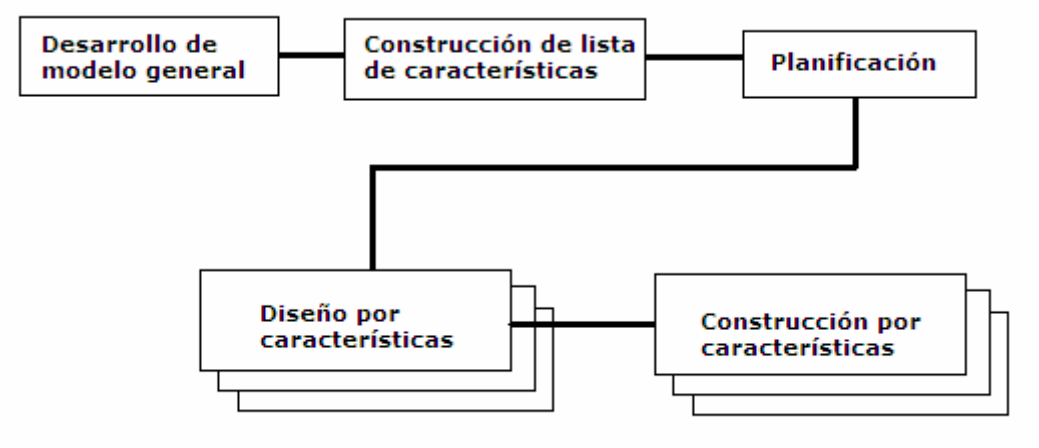
1) *Codificación.* Es el proceso en sí de trasladar el diseño a un lenguaje de programación.

2) *Inspección de la codificación.* Al igual que la inspección en el diseño, FDD recomienda una inspección al finalizar la codificación de una característica.

3) *Promoción para construir.* En esta etapa, debe definirse claramente cuándo una característica estará completamente terminada, esto quiere decir, cuando ya no haya más que hacer, siempre y cuando se cumpla con los requerimientos del cliente.

Cabe resaltar que los primeros tres procesos se deben realizar al inicio del proyecto mientras que los últimos dos, deben realizarse en cada iteración (Bauer 2005). La Ilustración 9 muestra los procesos recién descritos:

Ilustración 9. Procesos de la metodología FDD.



IV. ANÁLISIS

A. Descripción del problema

Idealmente, todas las empresas deberían implantar un conjunto de actividades que les ayuden a cumplir con sus objetivos en el mercado laboral y a la vez, sean la clave de un trabajo exitoso. En muchos casos, si dichas actividades existen, no están bien definidas en diversos aspectos: su estructura interna, su relación con otras actividades y su asignación al personal. Este conjunto de actividades se denomina flujo de trabajo en el mundo de los negocios y su papel es fundamental en el éxito de una empresa.

Actualmente existe un problema en muchas empresas y éste radica en el diseño de dicho flujo, incluyendo tanto la definición de procesos y subprocesos, así como la asignación de tareas a los distintos usuarios involucrados. Esta serie de actividades suele realizarse de forma manual, por lo que tiende a ser compleja, lenta e ineficiente. Además, en muchos casos no podrán evaluarse los resultados obtenidos al utilizar el flujo, ni obtener los resultados esperados, ya que es probable que en el camino se pierdan documentos relacionados o se modifique información importante.

Debido a esto es imprecindible contar con una herramienta que apoye de forma automatizada a las empresas en el proceso de definición de su propio flujo de trabajo. Esta herramienta debe tener como fin el promover ante todo la organización, eficiencia y calidad en el trabajo dentro de la empresa.

B. Descripción de la solución

1. Descripción del ambiente de la aplicación. Como solución al problema anteriormente descrito se propone el diseño de un sistema en línea denominado SmartFlow, cuyas características generales se detallan a continuación:

- SmartFlow permitirá la automatización completamente parametrizable del diseño y administración de procesos.

- SmartFlow proveerá diversos tipos de análisis y reportes para evaluar la eficiencia y el funcionamiento del flujo de trabajo en proyectos específicos.

- SmartFlow permitirá obtener una estructura organizacional derivada de los procesos de la empresa.

- SmartFlow automatizará el flujo de información, pues ya se sabe de antemano a donde debe dirigirse.

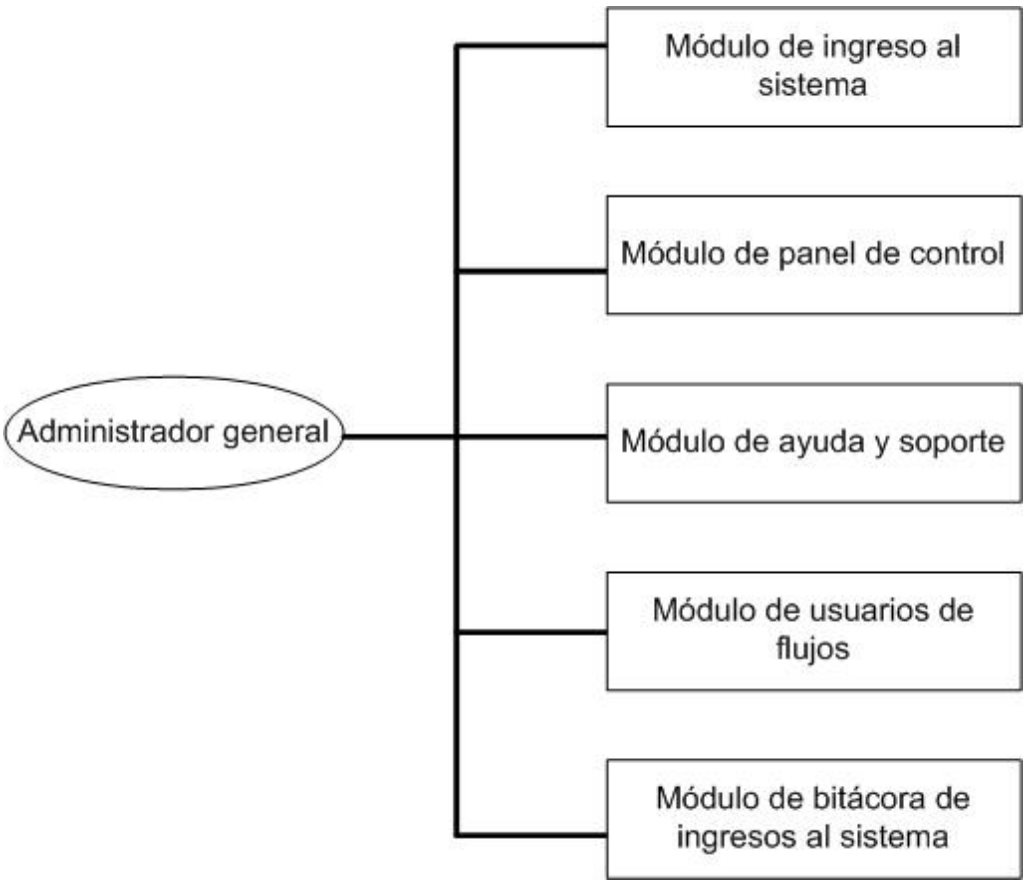
- SmartFlow proveerá distintos tipos de usuarios con diferentes niveles de acceso, garantizando completa seguridad en la especialización del trabajo.

- SmartFlow trabajará en línea, a través de un navegador, pudiéndose acceder desde cualquier lugar, en cualquier momento.

2. Usuarios de la aplicación. Dentro de SmartFlow existirán seis distintos tipos de usuarios, todos con funciones específicas. Enseguida se enumeran todos los usuarios, describiendo a grandes rasgos los módulos o funciones disponibles a los que tiene acceso cada tipo de usuario. Para un detalle completo de cada módulo, ver la siguiente sección: Descripción de módulos.

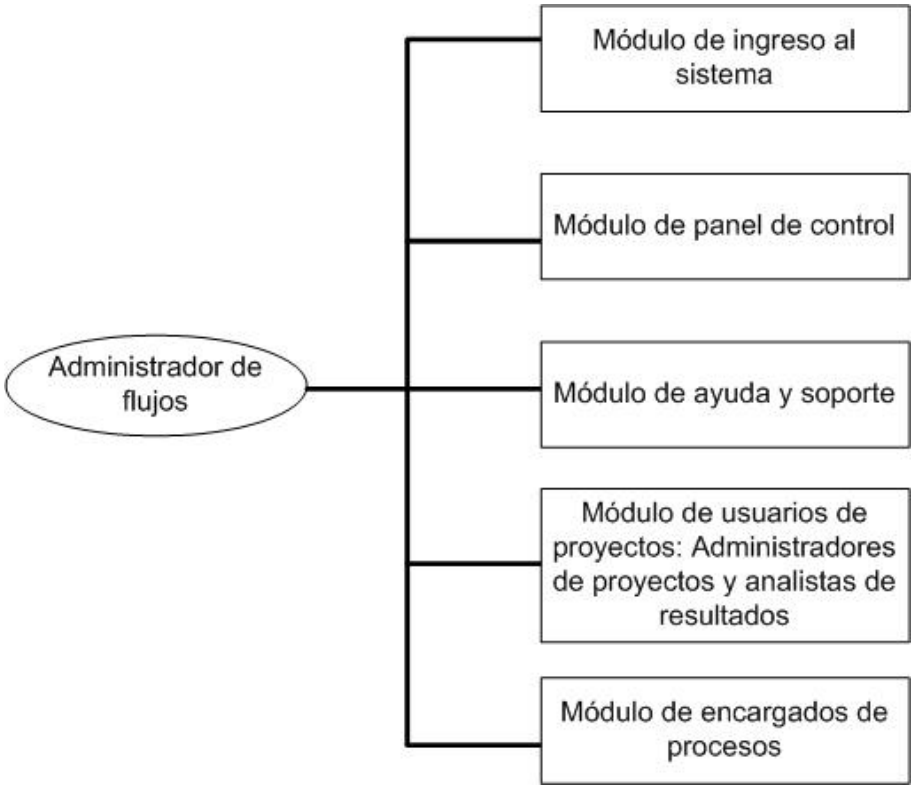
a. Administrador general. Es un único usuario dentro del sistema y es el encargado de crear a los usuarios que definirán los flujos de trabajo. Puede crear tantos administradores de flujos como se requiera. Este usuario tiene acceso a todos los módulos que se muestran en la ilustración 10.

Ilustración 10. Módulos de acceso para el administrador general.



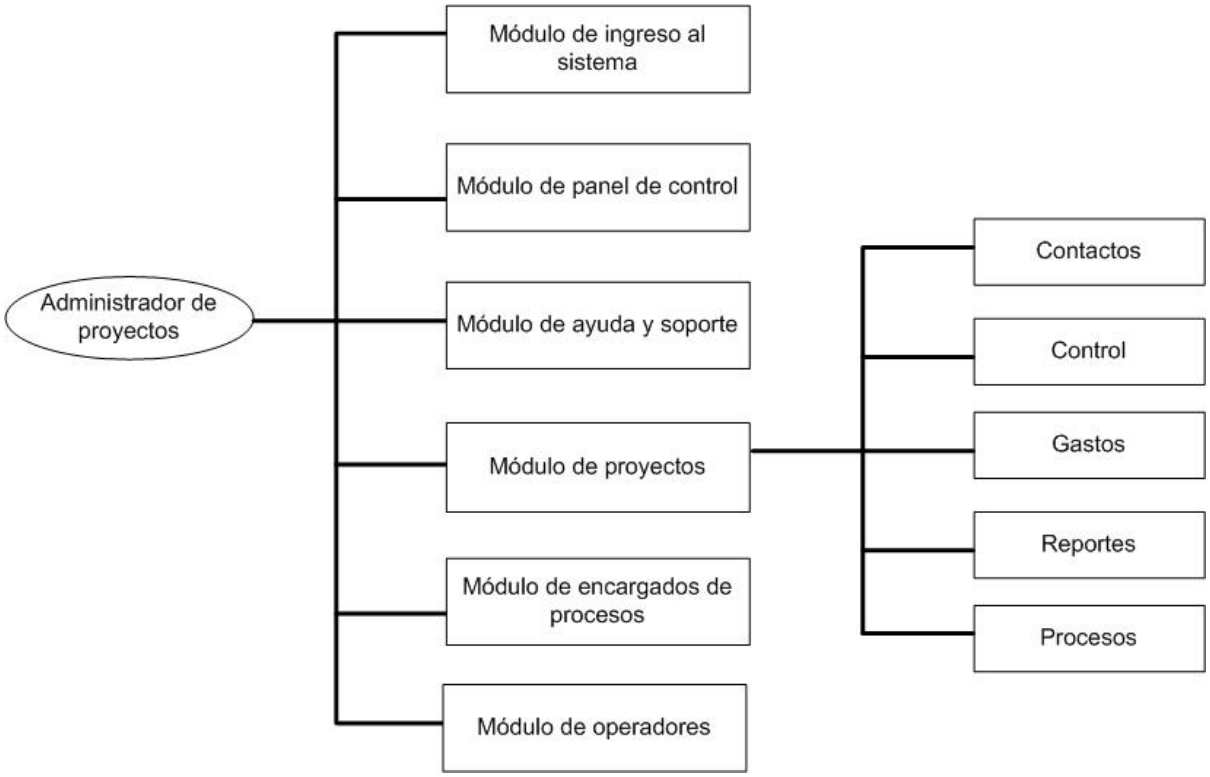
b. Administradores de flujos. Definen flujos de trabajo específicos para la empresa incluyendo: definición de procesos, orden y atributos, tareas relacionadas y la plantilla única para los proyectos que utilicen dichos flujos. Adicionalmente, los administradores de flujos son los responsables de crear los usuarios encargados de proyectos y los analistas de resultados, que son los que utilizarán directamente los flujos creados por ellos. Un administrador de flujos tiene acceso a los módulos que se muestran en la ilustración 11.

Ilustración 11. Módulos de acceso para el administrador de flujos.



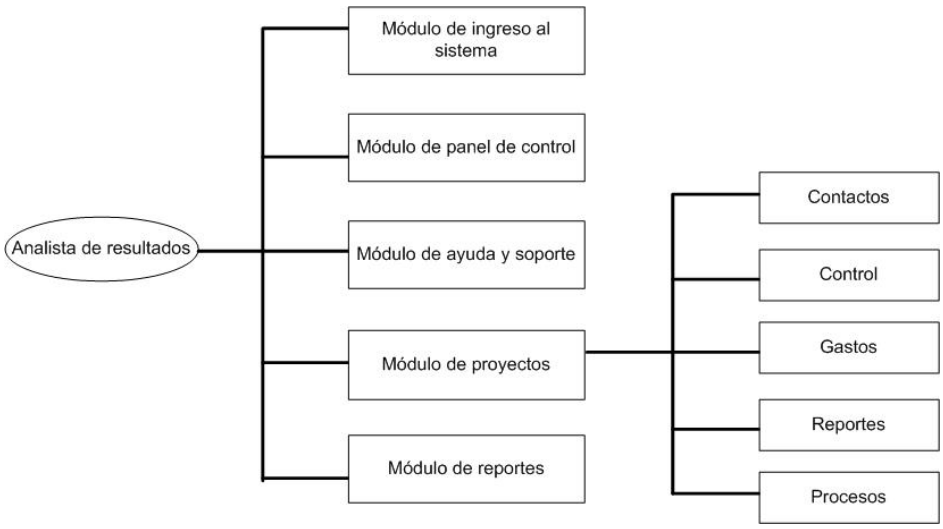
c. Administradores de proyectos. Son los encargados de definir proyectos específicos dentro del sistema con el fin de que estos pasen a través de los procesos definidos en el flujo, ejecutando las tareas correspondientes. Adicionalmente tienen a su cargo la definición de los encargados y operadores de todos los procesos en el flujo y la administración de toda la información relacionada únicamente con el o los proyectos que hayan creado. Esta información incluye: proyectos que se encuentren en determinados procesos, gastos reales y planificados por proyecto, análisis físico y financiero por proyecto, contactos o clientes relacionados al proyecto, las tareas que forman parte de los procesos y la estimación de fechas para la finalización de los procesos. A continuación se muestra la ilustración donde se indican los módulos a los que tiene acceso un administrador de proyectos.

Ilustración 12. Módulos de acceso para el administrador de proyectos.



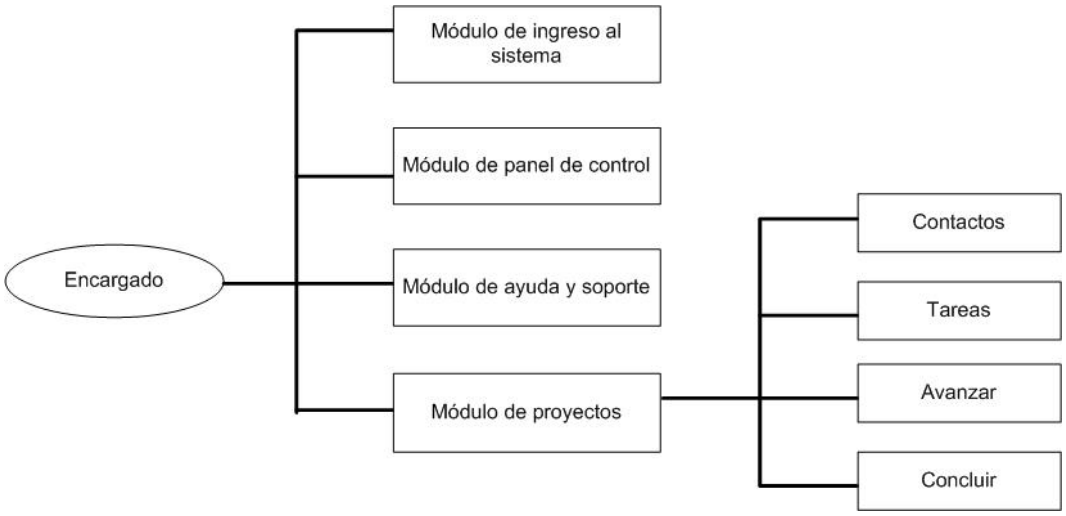
d. Analistas de resultados. Son aquellos usuarios cuya tarea es evaluar el rendimiento de la ejecución de los flujos previamente definidos en proyectos específicos, esto a través del análisis de reportes. Además, estos usuarios tienen la capacidad de evaluar y administrar todos los proyectos existentes en un momento dado, contrario a los administradores de proyectos que únicamente tienen acceso a los que ellos hayan creado. La ilustración 13 muestra sus módulos de acceso dentro del sistema.

Ilustración 13. Módulos de acceso para el analista de resultados.



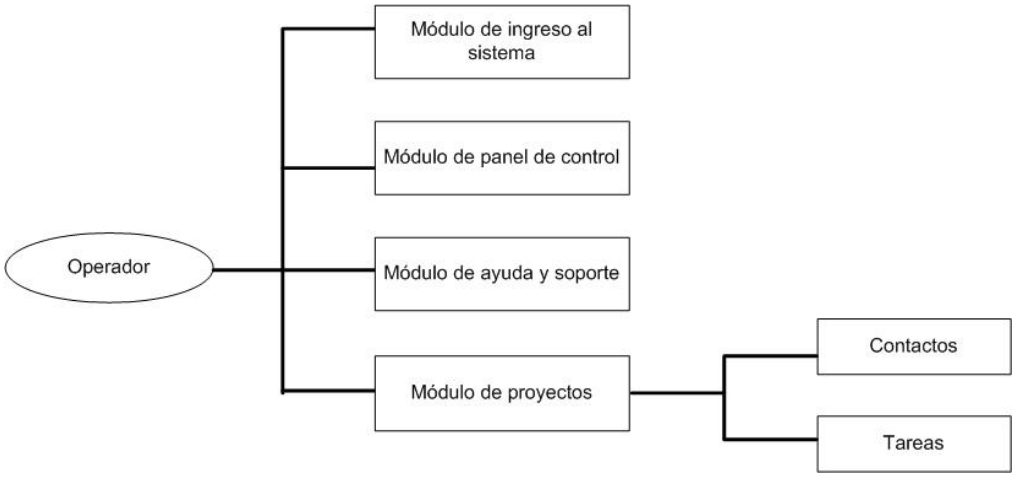
e. Encargados de proceso. Existe un único encargado por cada proceso en un proyecto y es el responsable de atender aspectos específicos del mismo: los diversos proyectos que estén en ese proceso en un momento dado, su desarrollo y los contactos relacionados. Adicionalmente, este tipo de usuario tiene a su cargo la asignación de tareas a los usuarios operadores; o a él mismo, pues él también puede tener asignadas varias tareas dentro de un proceso. Los módulos de acceso disponibles para un encargado se muestran en la ilustración 14.

Ilustración 14. Módulos de acceso para el encargado de proceso.



f. Operadores. Son los usuarios que deben realizar todas las tareas asignadas a ellos por el encargado de un proceso. Únicamente tienen acceso a sus tareas asignadas, a los contactos relacionados al proyecto en el cual trabajan y a los archivos relacionados a una tarea. Un operador puede tener asignadas al mismo tiempo varias tareas de proyectos distintos que se encuentren en procesos distintos. La ilustración 15 muestra los módulos a los cuales tiene acceso un usuario operador.

Ilustración 15. Módulos de acceso para el operador.



Como se puede observar en las ilustraciones anteriores, todos los usuarios descritos tienen acceso a tres módulos generales, cuyas ilustraciones se muestran a continuación:

Ilustración 16. Módulo de panel de control.

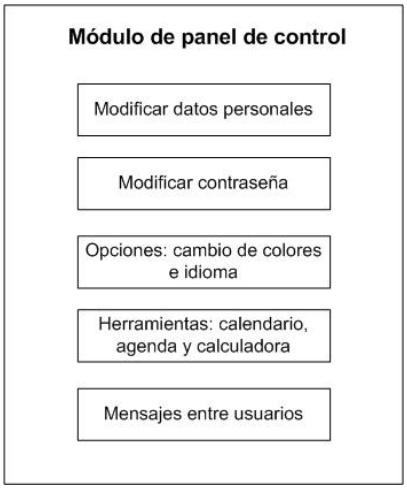
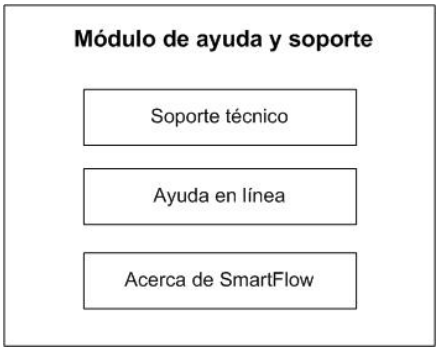


Ilustración 17. Módulo de ayuda y soporte.



- Módulo de ingreso al sistema: este módulo es descrito detalladamente en la sección: descripción de módulos.

Los usuarios detallados en este trabajo son únicamente el administrador general y el/los administrador(es) de flujos, cuyas funciones específicas son detalladas más adelante junto con la descripción de los respectivos módulos. Para la descripción de los cuatro usuarios restantes, referirse a la sección: Usuarios de la aplicación, en la parte de análisis del trabajo realizado por Mariana Rendón, titulado “Análisis y diseño de un sistema parametrizable para el control de flujo de trabajo orientado a la administración de proyectos de software: Manejo de proyectos”.

3. Descripción de clases. A continuación se presentan todas las clases a utilizar en el presente trabajo, con sus respectivos atributos y los métodos que ofrecen. Dichas clases son las que se proponen para el diseño del módulo administrativo de SmartFlow.

Por cada clase se muestra una descripción de la misma, así como la especificación de sus atributos y sus métodos. Además, para cada método se indican los parámetros que recibe y los valores que devuelve. Adicionalmente, se establece que todos los métodos son públicos y que todos los atributos dentro de las clases son privados (Ver marco teórico: Análisis y diseño orientado a objetos).

Para que el diseño del sistema se apegue al concepto de encapsulado de información relacionado a la programación orientado a objetos (Ver marco teórico: Análisis y diseño orientado a objetos), se definen dos métodos especiales para cada atributo:

- *Establecer atributo:* este método asigna un valor a un atributo específico.
    - Parámetros que recibe: nombre del atributo y valor del atributo.
    - Valor(es) que devuelve: ninguno.

- *Obtener atributo:* este método obtiene el valor asignado a un atributo específico.
    - Parámetros que recibe: nombre del atributo.
    - Valor(es) que devuelve: valor del atributo.

Siguiendo con la teoría de la programación orientada a objetos, es bien sabido que al hacer uso de un método dentro de una clase, los valores de los atributos que dicho método utilice se deben definir con el método *Establecer atributo* descrito anteriormente. Por ejemplo: Supóngase la existencia de la clase *Libros,* definida de la siguiente forma:

```
Clase Libros
 {
   Título: atributo privado
   Autor: atributo privado

   Método público establecerTítulo (valorTítulo)
   {
       //Método para establecer específicamente el título de un libro
        Título = valorTítulo
   }

   Método público obtenerTítulo()
  {
       //Método para obtener específicamente el título de un libro
        Devolver estaClase.valorTítulo
   }

   Método público establecerAutor (valorAutor)
   {
       //Método para establecer específicamente el autor de un libro
        Autor = valorAutor
   }

   Método público obtenerAutor()
   {
       //Método para obtener específicamente el autor de un libro
        Devolver estaClase.valorAutor
  }
```

```
    Método público mostrarLibro()
    {
        //Método para mostrar los atributos de un libro
         Desplegar "Título:" & estaClase.valorTítulo
         Desplegar "Autor:" & estaClase.valorAutor
    }
}
```

Para crear una nueva instancia (objeto) de la clase Libros, y utilizar el método mostrarLibro() de esta clase, debe realizarse lo siguiente:

```
Objeto Libro = nuevo Libros()  //creación de nueva instancia de la clase Libros
Libro.establecerTitulo = "El Alquimista"
Libro.establecerAutor = "Paulo Coelho"
mostrarLibro()
```

Con la llamada al método mostrarLibro se obtendrá el siguiente resultado:

Título: El Alquimista
Autor: Paulo Coelho

Todos los métodos de las clases que se definen a continuación utilizan la misma estructura de atributos y valores, mostrada en el ejemplo anterior.

a. FLUJOS. Maneja todos los aspectos relacionados con la administración de un flujo de trabajo dentro del sistema. Tiene relación directa con las clases definidas más adelante: PROCESOS, TAREAS, ESTADOS_TAREAS Y PLANTILLA.

1) Atributos

a) *Nombre.* Designación que se utilizará para identificar un flujo de trabajo dentro del sistema.

b) *Descripción.* Breve explicación de la función y objetivos del flujo de trabajo.

2) Métodos

a) *Agregar flujo de trabajo.* Permite agregar un nuevo flujo de trabajo al sistema.

- Parámetros que recibe: ninguno.
- Valores que devuelve: ninguno.

b) *Editar flujo de trabajo.* Modifica los atributos de un flujo de trabajo.

- Parámetros que recibe: ninguno.
- Valores que devuelve: ninguno.

c) *Mostrar listado de flujos de trabajo.* Método que obtiene un listado de todos los flujos de trabajo existentes en el sistema. Permite buscar un flujo por su nombre y ordenar el listado utilizando el mismo filtro.

- Parámetros que recibe: filtro, valor del filtro, orden y número de página. Este número se utiliza para la navegación entre páginas (Ver más adelante V. Diseño: diseño y descripción de interfaces gráficas como referencia). Esto aplica para todos los métodos similares que devuelven un conjunto de registros y lo muestran en forma de listado.
- Valores que devuelve: conjunto de registros de nombres de flujo.

d) *Contar flujos.* Provee un conteo de todos los flujos de trabajo que un administrador de flujos ha creado, siempre incluyendo algún criterio de búsqueda, en este caso es únicamente el nombre del flujo.

- Parámetros que recibe: filtro, valor del filtro.
- Valores que devuelve: número de flujos existentes.

e) *Mostrar flujo de trabajo*. Método que muestra los atributos propios de un flujo de trabajo.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

f) *Eliminar flujo de trabajo.* Elimina por completo un flujo de trabajo del sistema. Sin embargo, no se permite su eliminación si existen procesos definidos dentro de él.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

b. PROCESOS. Clase encargada de manejar todas las actividades que se relacionan con los procesos que forman parte de un flujo de trabajo.

1) Atributos

a) *Nombre*. Designación que se utilizará para identificar a un proceso dentro de un flujo de trabajo.

b) *Descripción*. Breve explicación de cual es la función y objetivos del proceso en el flujo de trabajo.

c) *Usuario*. Se refiere al usuario encargado o responsable del proceso.

d) *Flujo*. Se utiliza para indicar a qué flujo de trabajo pertenece un proceso.

2) Métodos

a) *Agregar proceso.* Agrega un nuevo proceso a un flujo de trabajo ya existente. Para esto, se requiere establecer valores para los atributos descritos anteriormente.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

b) *Editar proceso.* Permite la modificación de cualquier atributo de un proceso dentro de un flujo.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

c) *Mostrar listado de procesos.* Método que obtiene un listado de todos los procesos existentes en un flujo de trabajo específico. Permite buscar un proceso por su nombre y ordenar el listado por el mismo filtro.

- Parámetros que recibe: filtro, valor del filtro, orden y número de página.

- Valor(es) que devuelve: conjunto de registros de nombres de procesos.

d) *Contar procesos.* Método que provee un conteo de todos los procesos dentro de un flujo de trabajo, incluyendo algún criterio de búsqueda, en este caso es únicamente el nombre del proceso.

- Parámetros que recibe: filtro, valor del filtro.
- Valor(es) que devuelve: número de procesos.

e) *Mostrar proceso.* Muestra todos los atributos propios de un proceso dentro de flujo de trabajo.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

f) *Eliminar proceso.* Elimina por completo un proceso que forma parte de un flujo de trabajo del sistema. Sin embargo, no permite eliminarlo si existen tareas definidas dentro de él.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

g) *Obtener procesos.* Método que obtiene todos los procesos definidos dentro de un flujo.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: conjunto de registros de nombres de procesos.

h) *Establecer procesos antecesores.* Método que permite seleccionar qué procesos están definidos antes de un proceso dentro de un flujo de trabajo.

- Parámetros que recibe: conjunto de registros de nombres de procesos.
- Valor(es) que devuelve: ninguno.

i) *Establecer procesos sucesores.* Método que permite seleccionar qué procesos están definidos después de un proceso dentro de un flujo de trabajo.

- Parámetros que recibe: conjunto de registros de nombres de procesos.

- Valor(es) que devuelve: ninguno.

c. TAREAS. La definición de las tareas dentro del sistema se hará según la estructura de una característica o *feature* (Ver Marco teórico: Metodología ágil FDD). La definición de una tarea como *feature* incluye: una acción, un resultado y un objeto, a describir a continuación:

1) Atributos
   a) *Nombre.* Nombre propio de la tarea.
   b) *Acción.* Operación a realizar dentro de un proceso específico.
   c) *Resultado.* Indica lo que se desea obtener al realizar la acción antes mencionada dentro de un proceso.
   d) *Objeto*. Se refiere a la entidad sobre la cual se realizará la acción.
   e) *Proceso.* Se refiere al proceso específico al cual pertenece una tarea.
   f) *Flujo.* Se utiliza para indicar a qué flujo de trabajo pertenece un proceso y por lo tanto la tarea.
2) Métodos
   a) *Agregar tarea.* Método que agrega una nueva tarea a un proceso específico. Para esto, se requiere establecer los valores para los atributos descritos anteriormente.
      - Parámetros que recibe: ninguno.
      - Valor(es) que devuelve: ninguno.
   b) *Editar tarea.* Permite la modificación de cualquier atributo de una tarea dentro de un proceso.
      - Parámetros que recibe: ninguno.
      - Valor(es) que devuelve: ninguno.
   c) *Mostrar listado de tareas.* Obtiene un listado de todas las tareas existentes en un proceso específico. Permite buscar una tarea por su nombre y ordenar el listado utilizando el mismo filtro.
      - Parámetros que recibe: filtro, valor del filtro, orden y número de página.

- Valor(es) que devuelve: conjunto de registros de nombres de tareas.

d) *Contar tareas.* Método que provee un conteo de todas las tareas dentro de un proceso, incluyendo algún criterio de búsqueda, en este caso es únicamente el nombre de la tarea.

- Parámetros que recibe: filtro, valor del filtro.
- Valor(es) que devuelve: número de tareas.

e) *Mostrar tarea.* Muestra los atributos propios de una tarea dentro de un proceso específico.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

f) *Eliminar tarea.* Elimina por completo una tarea que forma parte de un proceso. Sin embargo, no se permite eliminarla si dicha tarea está asignada a algún usuario (Ver definición de la clase Tareas en el análisis del trabajo realizado por Mariana Rendón).

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

g) *Obtener tareas.* Método que obtiene todas las tareas definidas dentro de un proceso.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: conjunto de registros de nombres de tareas.

h) *Establecer tareas antecesoras.* Método que permite seleccionar qué tareas están definidas antes de una tarea especifica dentro de un proceso.

- Parámetros que recibe: conjunto de registros de nombres de tareas.
- Valor(es) que devuelve: ninguno.

i) *Establecer tareas sucesoras.* método que permite seleccionar qué tareas están definidas después de una tarea dentro de un proceso.

- Parámetros que recibe: conjunto de registros de nombres de tareas.

- Valor(es) que devuelve: ninguno.

d. USUARIOS. Esta clase permite a cualquier tipo de administrador realizar las operaciones relacionadas con los usuarios que él mismo defina dentro del sistema.

1) Atributos

a) *Usuario.* Identificador único que se asigna a cualquier persona que utilice el sistema. En este caso, se refiere únicamente a los usuarios administradores de flujos.

b) *Nombre.* Nombre propio que identifica al usuario del sistema.

c) *Apellido.* Nombre de la familia del usuario.

d) *Teléfono.* Número telefónico del usuario.

e) *Correo electrónico.* Dirección de correo electrónico.

f) *Contraseña.* Cadena de caracteres definida por el usuario con la cual se le concede acceso al sistema, únicamente él deberá conocerla.

g) *Tipo de Usuario.* Se refiere al tipo de usuario, y por lo tanto a las funciones a las que tiene acceso (Ver inciso 2. Usuarios de la aplicación).

h) *Activo*. Atributo que identifica si el usuario está activo dentro del sistema, es decir, si está disponible para realizar funciones dentro del mismo. Este atributo tan solo puede ser modificado por el administrador que lo haya creado.

2) Métodos

a) *Agregar usuario.* Método que agrega un nuevo usuario al sistema. Para esto, son requeridos todos los atributos descritos anteriormente.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

b) *Editar usuario*. Método que permite a un administrador modificar cualquier atributo de un usuario, incluyendo la contraseña.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

c) *Mostrar listado de usuarios.* Método que obtiene un listado de todos los usuarios existentes según el tipo de usuario que utilice este método. Si es el administrador general únicamente podrá ver a los administradores de flujos y si es un administrador de flujos, solo podrá consultar a los usuarios administradores de proyectos y analistas de resultados. Este método provee la capacidad de buscar un usuario por su apellido, nombre de usuario, tipo (según sea el caso) o estado, así como ordenar el listado utilizando los mismos criterios.

- Parámetros que recibe: filtro, valor del filtro, orden y número de página.
- Valor(es) que devuelve: conjunto de registros de usuarios que incluye: nombre y apellido, usuario, tipo y estado.

d) *Contar usuarios.* Provee un conteo de todos los usuarios, filtrando el resultado según el tipo del usuario que lo solicita.

- Parámetros que recibe: filtro, valor del filtro.
- Valor(es) que devuelve: número de usuarios.

e) *Mostrar usuario.* Método que muestra los atributos propios de un usuario específico dentro del sistema.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

f) *Eliminar usuario.* Permite eliminar por completo a un usuario del sistema.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

g) *Mostrar listado de nombres.* Este método únicamente obtiene los nombres propios de los usuarios activos dentro del sistema.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: conjunto de registros de nombres.

h) *Generar nueva contraseña.* Método que genera una nueva contraseña para un usuario en caso que él la haya olvidado.

- Parámetros que recibe: ninguno.

- Valor(es) que devuelve: nueva contraseña.

i) *Enviar nueva contraseña.* Este método permite enviar la nueva contraseña generada por el sistema a un usuario a través de su correo eletrónico.
   - Parámetros que recibe: ninguno.
   - Valor(es) que devuelve: ninguno.

j) *Almacenar nueva contraseña.* Almacena la nueva contraseña para un usuario dentro del sistema.
   - Parámetros que recibe: ninguno.
   - Valor(es) que devuelve: ninguno.

k) *Verificar contraseña actual.* Este método compara la contraseña almacenada en el sistema con la contraseña ingresada por el usuario.
   - Parámetros que recibe: contraseña actual.
   - Valor(es) que devuelve: verdadero o falso, según las contraseñas coincidan o no.

l) *Verificar nueva contraseña.* Método que verifica que dos contraseñas ingresadas por un usuario sean iguales.
   - Parámetros que recibe: contraseña nueva y verificación de contraseña.
   - Valor(es) que devuelve: verdadero o falso, según las contraseñas sean iguales o no.

m) *Establecer preferencias.* Sirve para almacenar las preferencias de un usuario dentro del sistema. Esto se refiere al tema (colores) y el idioma.
   - Parámetros que recibe: nuevo tema, nueva idioma.
   - Valor(es) que devuelve: ninguno.

n) *Identificar tipo.* Método que identifica y obtiene el tipo de un usuario.
   - Parámetros que recibe: ninguno.
   - Valor(es) que devuelve: tipo del usuario.

o) *Verificar si usuario está activo.* Este método se utiliza para verificar si un usuario está activo dentro del sistema.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: estado del usuario, activo o inactivo.

e. MENSAJES. Esta clase permite una comunicación fácil entre usuarios dentro del sistema, pues SmartFlow ofrece un mecanismo para enviar y recibir mensajes instantáneamente.

1) Atributos.

a) *Emisor.* Se refiere al usuario que genera el mensaje.

b) *Receptor.* Se refiere al usuario al que va dirigido el mensaje.

c) *Asunto*. De forma breve, indica el tópico del mensaje.

d) *Mensaje.* Es el texto a enviar como mensaje a un destinatario.

2) Métodos

a) *Mostrar listado de mensajes.* Obtiene un listado de todos los mensajes enviados a un usuario. Provee la capacidad de buscar un mensaje por la fecha en la que fue recibido, por su remitente o por el asunto, así como ordenar el listado utilizando estos mismos criterios.

- Parámetros que recibe: filtro, valor del filtro, orden y número de página.
- Valor(es) que devuelve: conjunto de registros de mensajes.

b) *Mostrar mensaje.* Muestra de manera individual el detalle de un mensaje recibido.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

c) *Contar mensajes.* Método que provee un conteo de todos los mensajes que un usuario ha recibido. Este método toma en cuenta algún criterio de búsqueda para el conteo, en este caso son los mismos descritos en el método *Mostrar listado de mensajes*.

- Parámetros que recibe: filtro, valor del filtro.

- Valor(es) que devuelve: número de mensajes.

d) *Eliminar mensaje.* Este método permite eliminar por completo un mensaje recibido.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

e) *Enviar mensaje.* Método que ofrece la funcionalidad de enviar un mensaje a cualquier usuario activo dentro del sistema.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

f. BITÁCORA. Esta clase se define para proveer al administrador general un mecanismo de control de los ingresos de todos los usuarios al sistema.

1) Atributos

a) *Usuario.* Se refiere al identificador único para cada persona que utilice el sistema.

b) *Tipo de Usuario.* Este tipo se refiere a todos los tipos de usuarios existentes en el sistema (Ver inciso 2. Usuarios de la aplicación).

c) *Fecha y Hora.* Indica la fecha y hora en las cuales un usuario ingresó al sistema.

2) Métodos

a) *Agregar ingreso.* Método que agrega un nuevo registro a la bitácora de usuarios. Esto se hace automáticamente en el momento que cualquier usuario ingrese al sistema.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

b) *Mostrar listado de ingresos.* Método que obtiene un listado de todos los ingresos de usuarios al sistema, únicamente desde seis meses atrás hasta el momento. Este método provee la funcionalidad adicional de filtrar los registros por el usuario, tipo y fecha de ingreso, así como la de ordenar el listado utilizando los mismos criterios definidos arriba. En caso que exista algún registro cuya fecha sea anterior a seis meses de la fecha actual, es

automáticamente eliminado del sistema y es almacenado en un archivo de copia de seguridad (*backup*).

- Parámetros que recibe: filtro, valor_filtro, orden, página.
- Valor(es) que devuelve: conjunto de registros de ingresos al sistema.

c) *Contar ingresos.* Método que obtiene un conteo de los registros almacenados en el listado de todos los registros de ingreso.

- Parámetros que recibe: filtro, valor_filtro.
- Valor(es) que devuelve: número de ingresos.

g. TICKETS. SmartFlow ofrece a todos los usuarios la opción de enviar alguna duda o comentario a los técnicos a través del mismo sistema utilizando un mecanismo denominado "control de tickets". Un ticket es un mensaje que se envía del usuario al soporte técnico y viceversa. La respuesta a estos tickets se hace a través de correo electrónico al respectivo usuario, así como a través de un mensaje dentro del sistema.

1) Atributos

a) *Asunto.* Especifica el tópico del mensaje a enviar.

b) *Descripción.* Texto a enviar como mensaje en el ticket.

c) *Usuario.* Indica el usuario que general el mensaje a enviar.

2) Métodos

a) *Mostrar listado de tickets.* Obtiene un listado de todos los tickets que son respuesta a algún ticket enviado por un usuario. Este método permite buscar un ticket por la fecha en la que fue enviado y por el asunto, así como ordenar el listado utilizando estos mismos criterios.

- Parámetros que recibe: filtro, valor del filtro, orden y número de página.
- Valor(es) que devuelve: conjunto de registros de tickets.

b) *Mostrar ticket.* Muestra de manera individual el detalle de un ticket recibido.

- Parámetros que recibe: ninguno.

- Valor(es) que devuelve: ninguno.

c) *Contar tickets.* Método que provee un conteo de todos los tickets que un usuario ha recibido. Este método toma en cuenta algún criterio de búsqueda, en este caso son los mismos descritos en el método *Mostrar listado de tickets*.

- Parámetros que recibe: filtro, valor del filtro.
- Valor(es) que devuelve: número de tickets.

d) *Eliminar ticket.* Este método permite eliminar por completo un ticket recibido.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

e) *Enviar nuevo ticket*. Método que ofrece la funcionalidad de enviar un ticket a un técnico dentro del sistema.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

h. PLANTILLA. Una plantilla es un conjunto de datos o características a almacenar de un proyecto. Todos los proyectos que sean creados dentro de un flujo utilizarán la plantilla definida para él; es decir, todos tendrán el mismo esquema y almacenarán las mismas características, únicamente que cada proyecto tendrá diferentes valores. Esta clase permite la definición de una plantilla para un flujo de trabajo.

1) Atributos

a) *Nombre.* Nombre que identifica al atributo.
b) *Descripción.* Breve descripción del atributo y de su información a almacenar.
c) *Longitud.* Tamaño que ocupará el valor para un atributo.

2) Métodos

a) C*onfigurar plantilla.* Se utiliza para definir la plantilla propia y única para los proyectos que utilizarán un flujo de trabajo determinado. Para cada atributo deben especificarse los tres atributos mencionados anteriormente. Esta configuración almacenará dentro

del sistema el número de atributos especificado por el administrador de flujos junto con sus descripciones.

- Parámetros que recibe: un conjunto de nombres, descripciones y longitudes de los atributos.
- Valor(es) que devuelve: ninguno.

i. ESTADOS_TAREAS. Una tarea siempre debe tener un estado asociado que indique en qué etapa se encuentra. SmartFlow permite personalizar los estados de las tareas dentro de un flujo de trabajo, por lo que se provee la siguiente clase donde se permite la asignación de un orden para cada estado, así como la definición de uno o varios estados como estado final o de terminación.

1) Atributos
   a) *Descripción.* Nombre que identifica el estado de una tarea.
   b) *Orden.* Se utiliza para establecer una jerarquía entre los estados definidos y determinar en qué punto se llega al estado final. Por definición, el orden del estado final debe ser mayor que el del resto.
   c) *Estado final*. Atributo que se utiliza para identificar si un estado es un estado de terminación. Pueden existir varios estados definidos como estados finales dentro del mismo flujo de trabajo.
2) Métodos
   a) *Agregar estado de tarea.* Método que agrega un nuevo estado de tarea a un flujo específico, siempre requiriendo todos los atributos descritos arriba.
      - Parámetros que recibe: ninguno.
      - Valor(es) que devuelve: ninguno.
   b) *Editar estado de tarea.* Este método modifica los estados de las tareas definidos para un flujo determinado.
      - Parámetros que recibe: ninguno.
      - Valor(es) que devuelve: ninguno.
   c) *Eliminar estado de tarea.* Permite eliminar por completo un estado de tarea definido para un flujo de trabajo. Sin embargo, no es

posible eliminar un estado de tarea que esté asociado a alguna tarea existente en un momento dado; primero deben eliminarse las tareas que se encuentren en ese estado y luego se procede con el estado.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

d) *Mostrar listado de estados de tareas.* Se proporciona un listado de todos los estados de tareas que forman parte de un flujo específico. Este listado puede ser ordenado por el nombre del estado o su orden.

- Parámetros que recibe: filtro, valor_filtro, orden, página.
- Valor(es) que devuelve: conjunto de registros de estados de tareas y su respectivo orden.

e) *Establecer como estado de terminación.* Permite establecer un estado específico como el último o de terminación, esto con el fin de avanzar al siguiente proceso en el flujo de trabajo. Cualquier tarea que se encuentre en este estado, puede considerarse como terminada y podrá procederse con las tareas de los procesos sucesores.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: ninguno.

j. TEMAS. Esta clase se encarga de manejar los temas o estilos disponibles dentro de SmartFlow para el uso del usuario.

1) Atributos

a) *Id_tema.* Atributo que identifica unívocamente a un tema dentro del sistema.

b) *Descripción.* Indica el nombre propio o descripción del tema.

2) Métodos

a) *Mostrar listado de temas.* Muestra un listado de los temas existentes dentro del sistema.

- Parámetros que recibe: ninguno.

- Valor(es) que devuelve: conjunto de registros de temas.

k. IDIOMAS. Esta clase se encarga de manejar los idiomas disponibles dentro de SmartFlow. Específicamente estos son: inglés y español.

1) Atributos

a) *Id_idioma.* Atributo que identifica unívocamente a un idioma dentro del sistema.

b) *Descripción.* Indica el nombre del idioma: inglés o español.

2) Métodos

a) *Mostrar listado de idiomas.* Muestra un listado de los idiomas existentes dentro del sistema.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: conjunto de registros de idiomas.

l. AYUDA EN LÍNEA. Esta clase se utiliza para administrar internamente los enunciados de ayuda en línea dentro del sistema. Cabe resaltar que esta ayuda está disponible dentro del sistema según el tipo del usuario que la acceda.

1) Atributos

a) *Descripción.* Se refiere al texto en sí a almacenar dentro del sistema.

b) *Idioma.* Especifica en qué idioma se encuentra un enunciado.

c) *Tipo de usuario.* Indica qué tipo de usuario tiene acceso a un enunciado específico de ayuda.

2) Métodos

a) *Obtener enunciado de ayuda.* Método que se utiliza para mostrar un conjunto de enunciados de ayuda para un usuario, según su tipo.

- Parámetros que recibe: ninguno.
- Valor(es) que devuelve: conjunto de registros de enunciados de ayuda.

4. Descripción de módulos.

Enseguida se muestra una descripción de cada módulo a manejar en el presente trabajo. Para cada módulo se muestra su propósito o función principal, el o los usuarios que lo utilizan, la especificación de clases y uno o varios diagramas de flujo de datos según sea el caso. En cada diagrama se muestra una leyenda en la parte inferior derecha que indica por medio de un color la clase que se utiliza dentro de ese módulo específico. Asimismo, cada proceso que se muestra en el diagrama tiene un color; esto quiere decir que cada proceso corresponde a un método definido dentro de la clase especificado a través de su color.

a. Módulo: Usuarios de flujos.

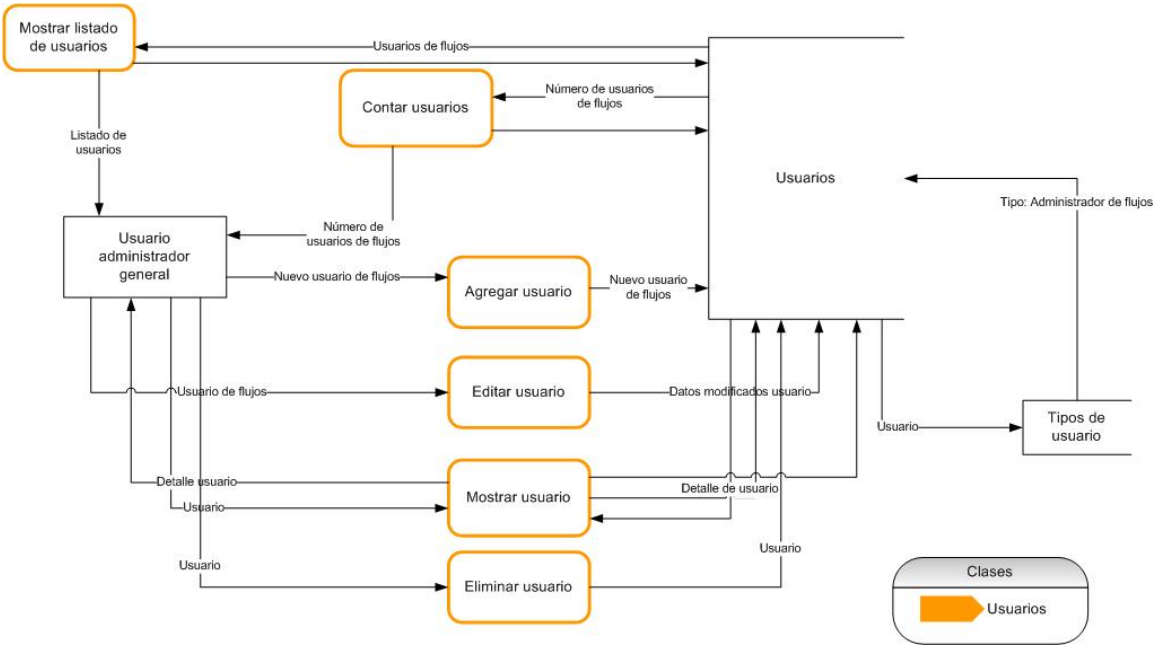
- *Propósito del módulo.* Este módulo pretende administrar únicamente a los usuarios que pueden crear flujos dentro del sistema, es decir Administradores de flujos.
- *Usuario que lo utiliza.* Administrador general.
- *Especificación de clases.* En este módulo se utilizan los siguientes métodos de la clase USUARIOS (Ver inciso 3.d.2):

    1) Agregar usuario
    2) Editar usuario
    3) Mostrar listado de usuarios
    4) Contar usuarios
    5) Mostrar usuario
    6) Eliminar usuario

    Específicamente, el tipo del usuario será siempre para este módulo *Administrador de flujos*.

- *Diagrama de flujo de datos:*

Ilustración 18. Diagrama DFD para el módulo de usuarios de flujos.



- *Relación con otros módulos:* Este módulo no se relaciona con ningún otro y es para el uso exclusivo del Administrador, para el manejo de Administradores de flujos.

b. Módulo: Bitácora de ingresos al sistema.

- *Propósito del módulo.* Este módulo sirve para llevar el control de los accesos de todos los usuarios al sistema.
- *Usuario que lo utiliza.* Administrador general.
- *Especificación de clases.* En este módulo se utilizará únicamente la clase BITÁCORA con todos sus métodos anteriormente especificados (Ver inciso 3.f.2).
- *Diagrama de flujo de datos.* el diagrama para esta clase se incluye dentro del diagrama de flujo de datos para el módulo de Ingreso al Sistema (Ver Ilustración 25).
- *Relación con otros módulos.* Este módulo no se relaciona con ningún otro y es para el uso exclusivo del Administrador General, únicamente para el control de ingresos al sistema.

c. Módulo: Usuarios de proyectos (Administradores de proyectos y analistas de resultados).

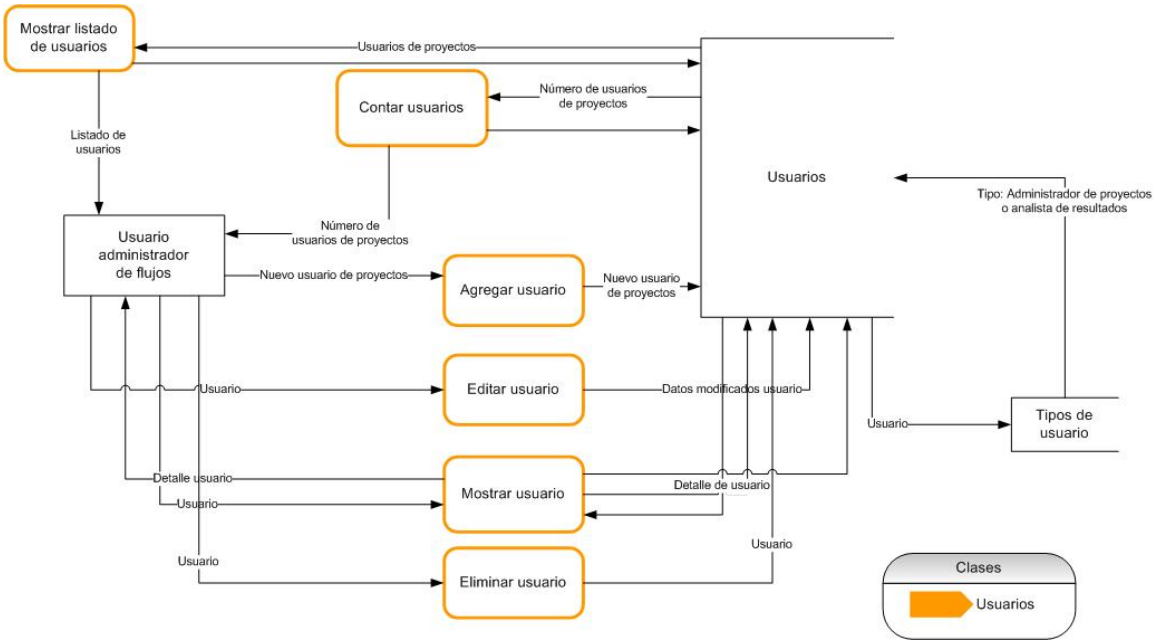
- *Propósito del módulo.* Este módulo se utiliza para administrar a los usuarios relacionados con proyectos, siendo estos los administradores de proyectos y los analistas de resultados.
- *Usuario que lo utiliza.* Administrador de flujos.
- *Especificación de clases.* En este módulo también se utiliza la clase USUARIOS con todos los siguientes métodos (Ver inciso 3.d.2):

    1) Agregar usuario
    2) Editar usuario
    3) Mostrar listado de usuarios
    4) Contar usuarios
    5) Mostrar usuario
    6) Eliminar usuario

En este módulo, existen dos tipos de usuarios: *Administrador de proyectos* y *analista de resultados*.

- *Diagrama de flujo de datos:*

Ilustración 19. Diagrama DFD para el módulo de usuarios de proyectos.



- *Relación con otros módulos.* Directamente, este módulo no se relaciona con ningún otro y es para el uso exclusivo de el/los administrador(es) de flujos con el fin de manejar a los distintos usuarios de proyectos que se pueden tener en el sistema para un flujo de trabajo determinado.

d. Módulo: Flujos.

- *Propósito del módulo.* Este módulo permite el diseño de un flujo de trabajo totalmente personalizado. Esto incluye la definición de múltiples procesos y tareas dentro de cada proceso. Complementariamente, permite establecer procesos y tareas en paralelo (Ver Marco teórico: Metodología XP), ya que cada proceso y cada tarea tienen relacionados varios procesos y tareas que son sus antecesores y sucesores. Un proceso antecesor es aquel cuyas tareas necesitan estar completamente terminadas antes de dar inicio al siguiente proceso, es decir, el proceso actual. Un proceso sucesor es aquel que sigue al proceso actual y requiere que las tareas de dicho proceso estén terminadas. Este módulo también permite la definición de los estados para sus tareas.
- *Usuarios que lo utilizan.* Administrador de flujos.
- *Especificación de clases.* este módulo utiliza cinco de las clases descritas anteriormente y éstas son:
    1) FLUJOS, con todos sus métodos (Ver inciso 3.a.)
    2) PROCESOS, con todos sus métodos (Ver inciso 3.b.)
    3) TAREAS, con todos sus métodos (Ver inciso 3.c.)
    4) ESTADOS_TAREAS, con todos sus métodos (Ver inciso 3.i.)
    5) PLANTILLA, con todos sus métodos (Ver inciso 3.h.)

- *Diagramas de flujo de datos.* A manera de mejor comprensión, a continuación se muestran los distintos diagramas para cada clase anteriormente descrita para el módulo de flujos.

Ilustración 20. Diagrama DFD para la clase FLUJOS en el módulo flujos.

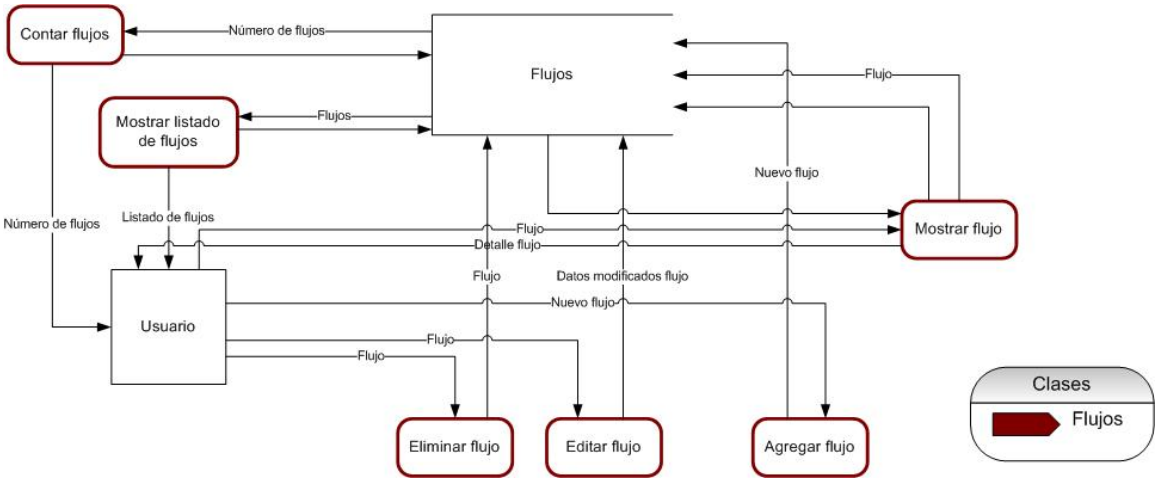


Ilustración 21. Diagrama DFD para la clase PROCESOS en el módulo de flujos.

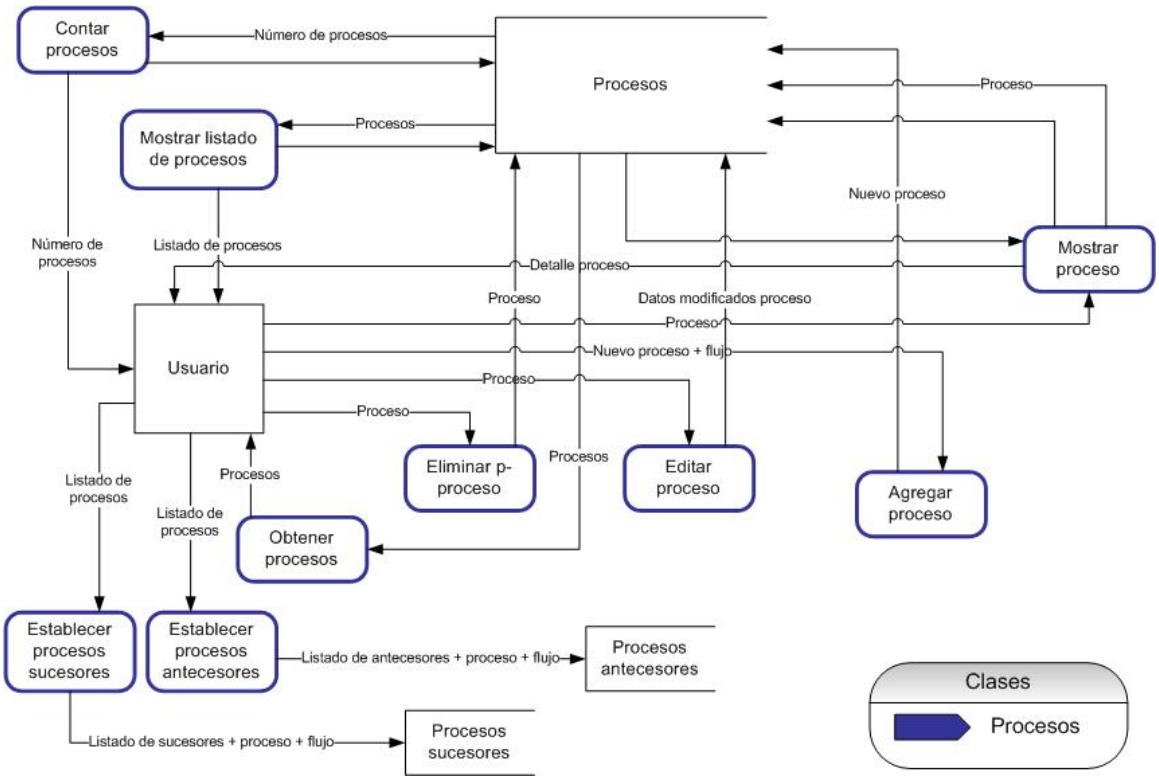


Ilustración 22. Diagrama DFD para la clase TAREAS en el módulo de flujos.

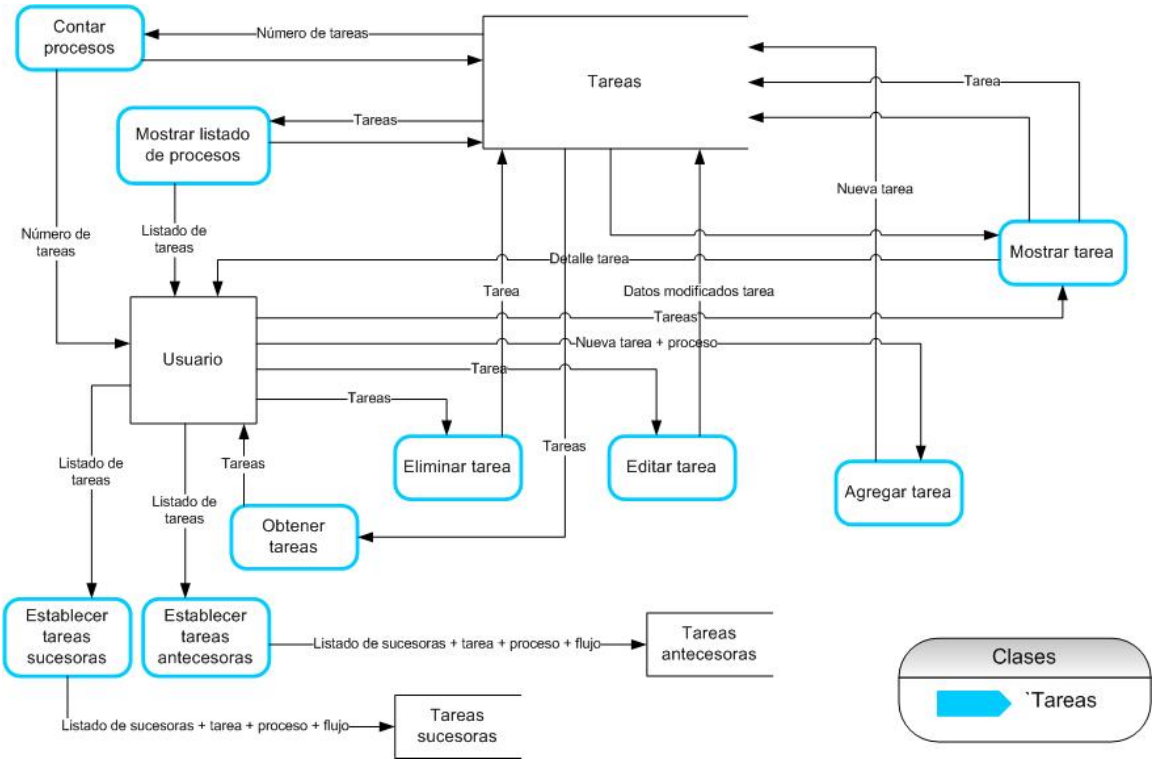


Ilustración 23. Diagrama DFD para la clase ESTADOS_TAREAS en el módulo de flujos.

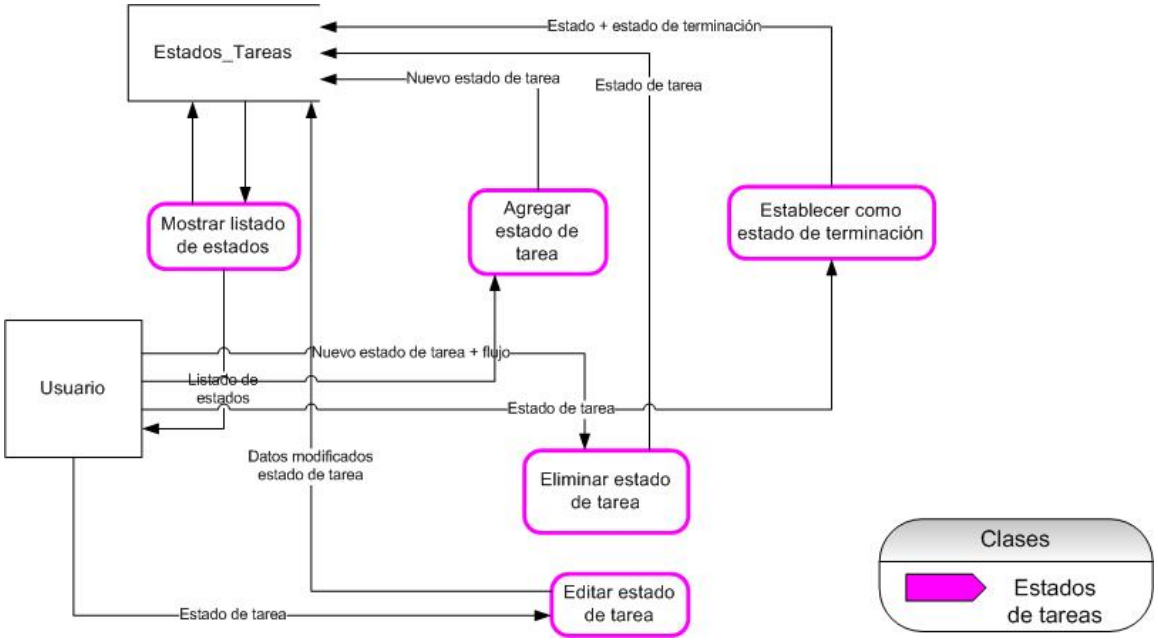


Ilustración 24. Diagrama DFD para la clase PLANTILLA en el módulo de flujos.

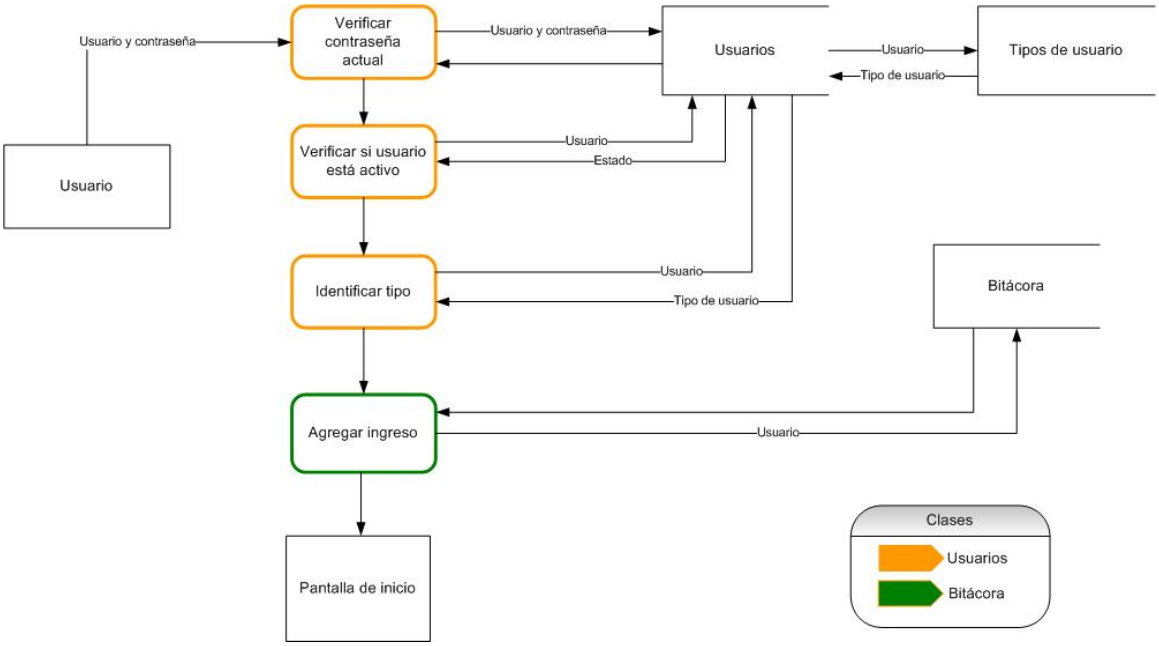
- *Relación con otros módulos.* El módulo de flujos no se relaciona directamente con otro, sin embargo lo definido en este será utilizado en el módulo de usuario.

e. Módulo: Ingreso al sistema.

- *Propósito del módulo.* Este módulo provee la interfaz para el ingreso de un usuario al sistema, siempre utilizando un nombre único de usuario y una contraseña personal como parámetros. Esto se hace a través de un mecanismo de autenticación detallado más adelante.
- *Usuarios que lo utilizan.* Administrador general, administrador(es) de flujos, administrador(es) de proyectos, encargado(s) y operador(es).
- *Especificación de clases.* Este módulo utiliza dos de las clases descritas anteriormente:
    1) Clase USUARIOS
        a. Identificar tipo de usuario.
        b. Verificar contraseña actual.
    2) Clase BITÁCORA DE INGRESOS
        a. Agregar ingreso

- *Diagrama de flujo de datos.*

Ilustración 25. Diagrama DFD para el método ingresar al sistema de la clase USUARIOS.



- *Relación con otros módulos.* este módulo está relacionado indirectamente con todos los demás módulos, pues de éste módulo depende si un usuario puede ingresar al sistema y a las funciones a las que tiene acceso según su tipo.

f. Módulo: Panel de control.

- *Propósito del módulo.* Este módulo ofrece al usuario distintas opciones a través de las cuales éste puede modificar y personalizar su ambiente de trabajo, así como utilizar las herramientas que el sistema provee y obtener una nueva contraseña en caso de que el usuario la haya olvidado.
- *Usuarios que lo utilizan.* Todos. Este módulo se encuentra disponible para todos los usuarios del sistema independientemente de su tipo.
- *Especificación de clases.* Este módulo utiliza dos de las clases mencionadas en el inciso 2: USUARIOS, específicamente los métodos Establecer preferencias, Modificar datos personales y Modificar contraseña. Además, se utiliza también la clase MENSAJES, con todos sus métodos.

Además, dentro de este módulo se ofrecen opciones adicionales que no son clases y se definen a continuación:

CALCULADORA

SmartFlow provee una calculadora simple integrada al sistema que puede ser utilizada en cualquier momento.

CALENDARIO

SmartFlow también provee un calendario, para que el usuario pueda consultar cualquier evento ajeno a las tareas asignadas en el sistema en cualquier momento.

- *Diagramas de flujo de datos.* A manera de mejor comprensión, se muestran los distintos diagramas para cada opción anteriormente mencionada.

Ilustración 26. Diagrama DFD para el método modificar datos de la clase USUARIOS en el módulo de panel de control.

Ilustración 27. Diagrama DFD para el método modificar contraseña de la clase USUARIOS en el módulo de panel de control.

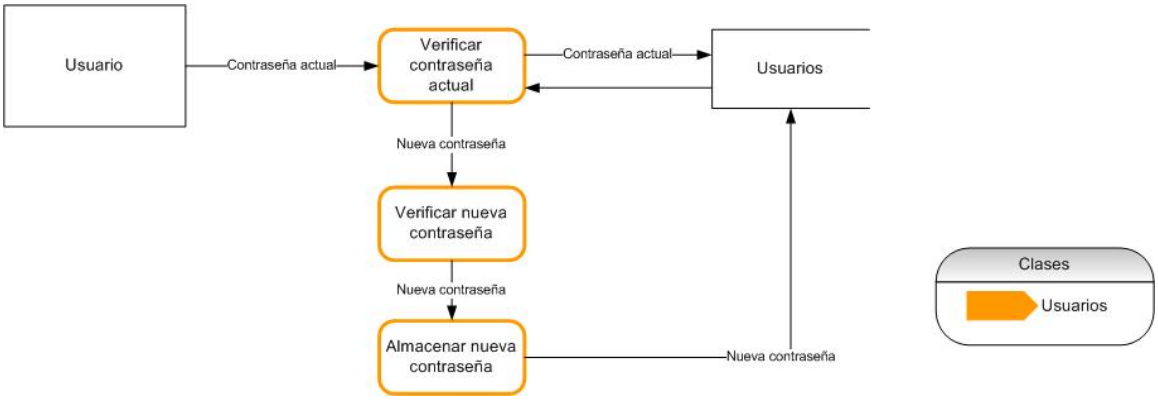


Ilustración 28. Diagrama DFD para el método establecer preferencias de la clase USUARIOS en el módulo de panel de control.

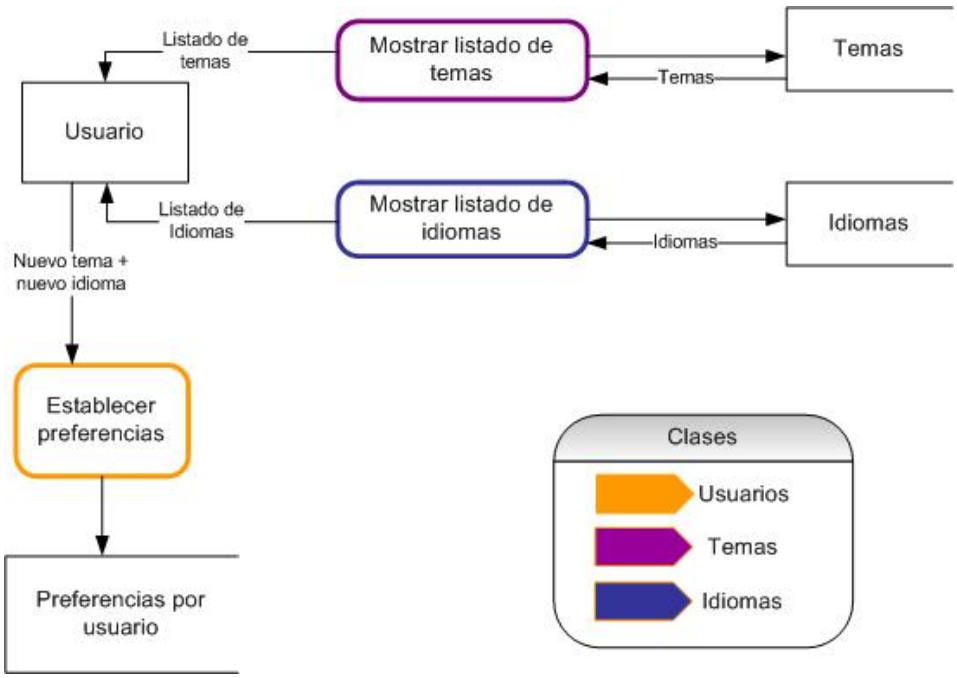


Ilustración 29. Diagrama DFD para la clase MENSAJES en el módulo de panel de control.

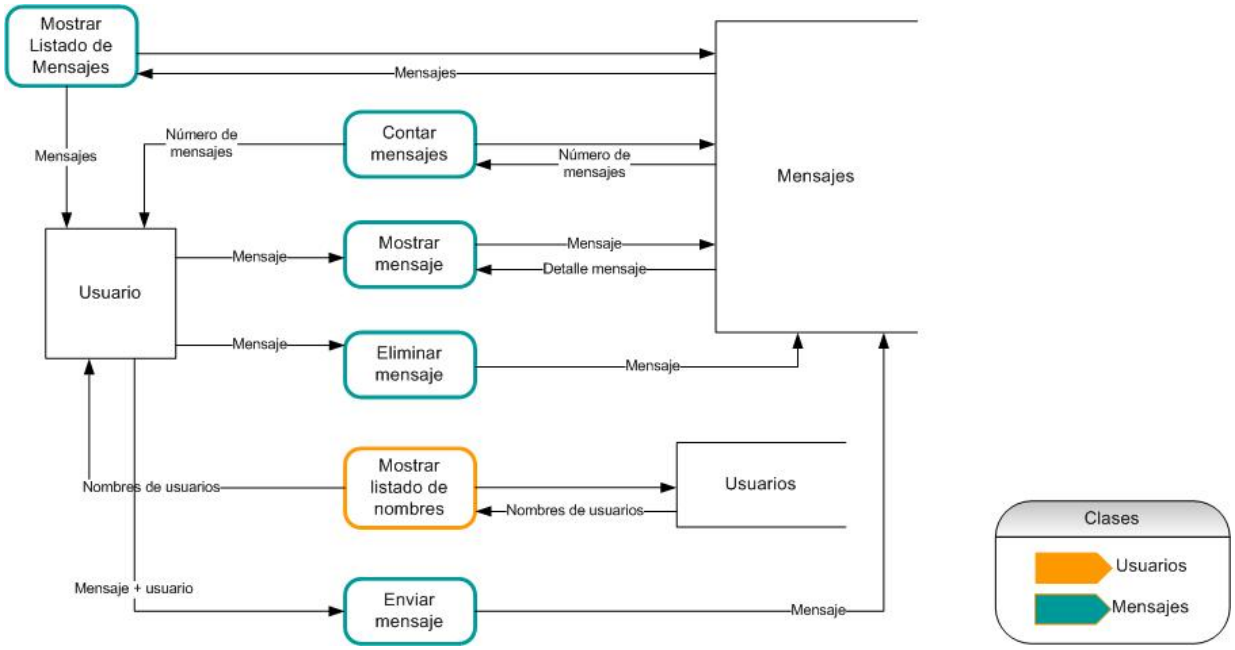
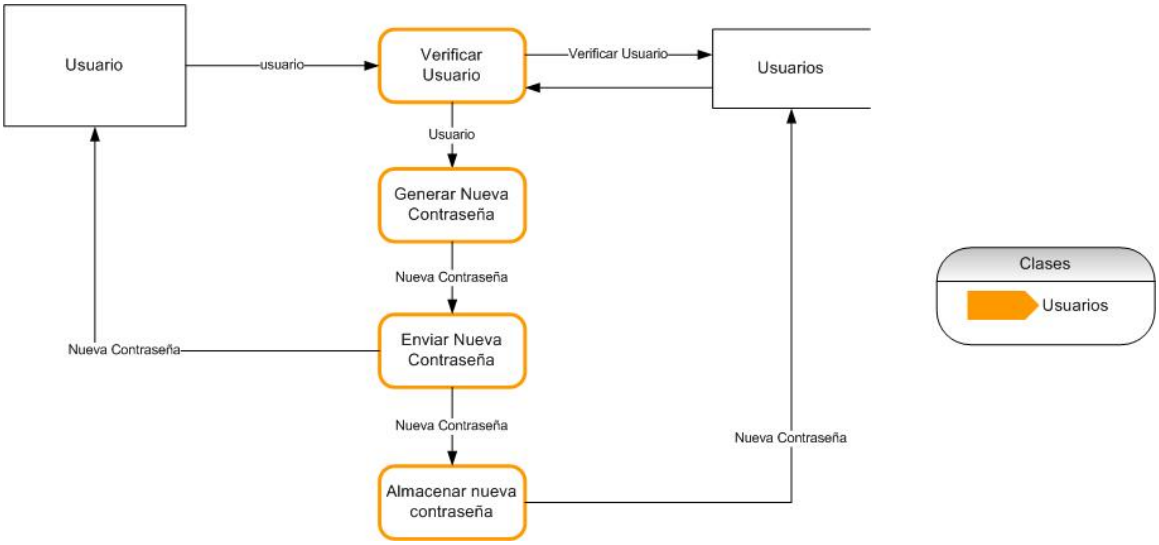


Ilustración 30. Diagrama DFD para recuperar contraseña de clase USUARIOS en el módulo de panel de control.



- *Relación con otros módulos.* No guarda relación alguna con otro módulo, pero éste se encuentra disponible en todas las páginas para cualquier usuario.

g. Módulo: Ayuda y soporte.

- *Propósito del módulo.* Se utiliza para proveer al usuario un mecanismo de ayuda dentro del sistema, ya sea de forma inmediata o sujeta a una respuesta por parte del equipo técnico de SmartFlow.
- *Usuarios que lo utilizan.* Este módulo está disponible para todos los usuarios del sistema independientemente de su tipo.
- *Especificación de clases.* Este módulo incluye dos clases: TICKETS (Ver inciso 2.g) y AYUDA EN LÍNEA (inciso 2.l). Además, se cuenta con sección puramente informativa donde se muestran las características más importantes de SmartFlow, y se denomina ACERCA DE SMARTFLOW. Cabe destacar que esta sección no es una clase dentro del sistema.

- *Diagramas de flujo de datos:*

Ilustración 31. Diagrama DFD para la clase TICKETS en el módulo de ayuda y soporte.

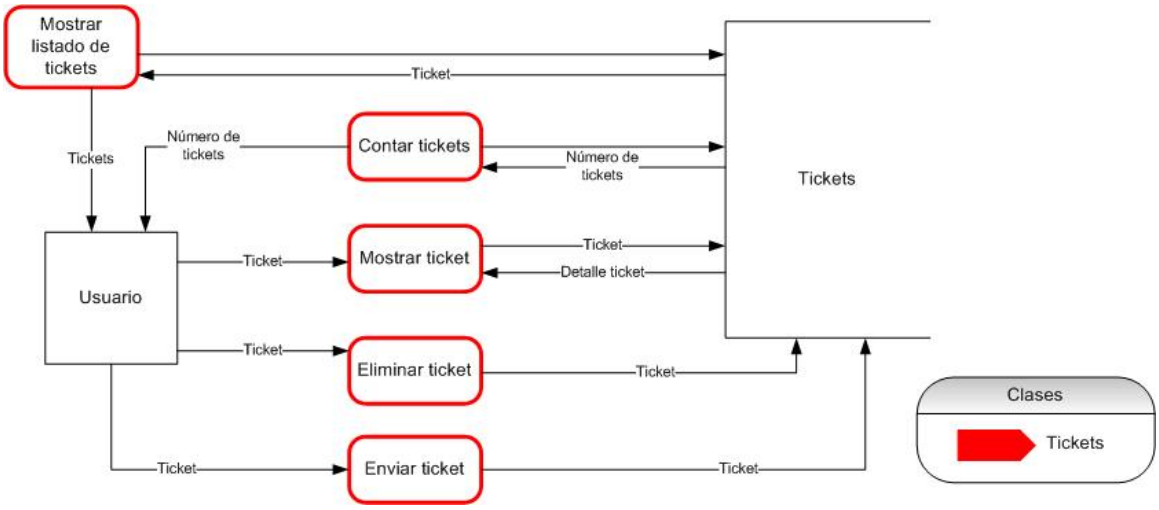
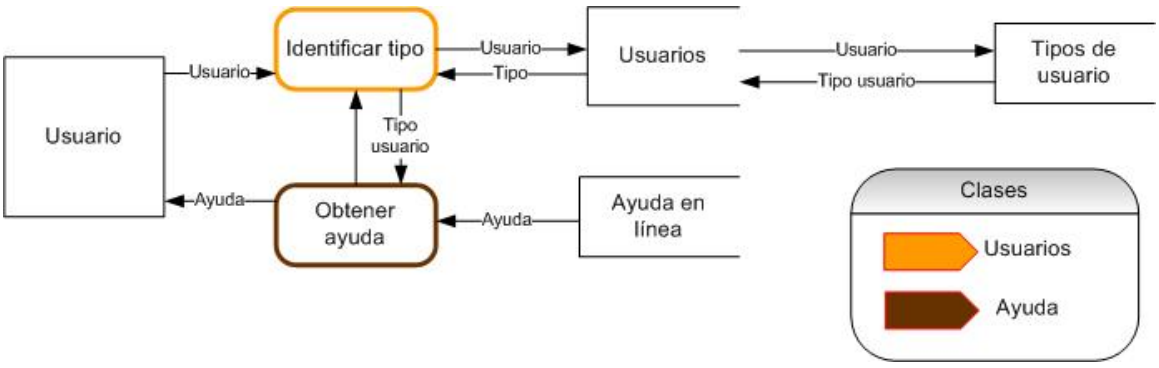


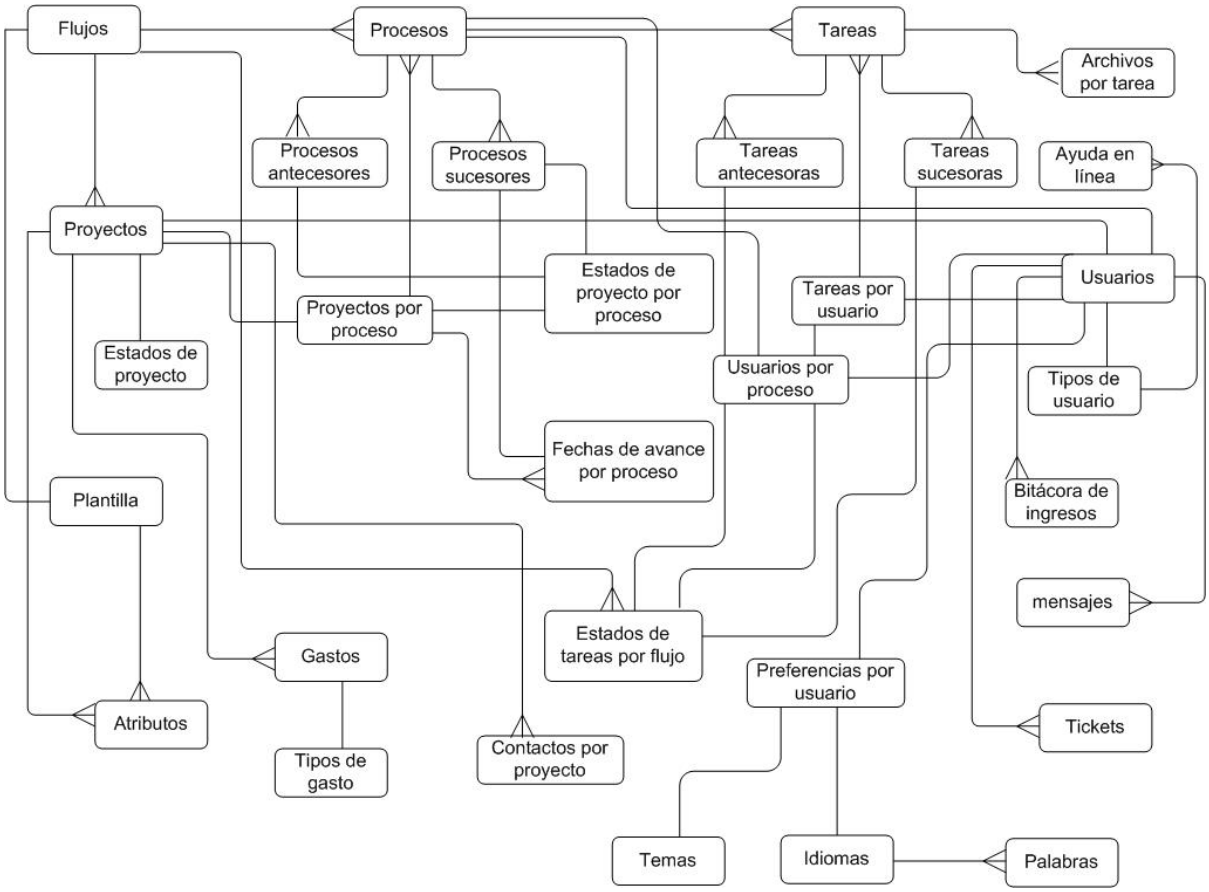
Ilustración 32. Diagrama DFD para la clase AYUDA EN LÍNEA en el módulo de ayuda y soporte.



- *Relación con otros módulos.* Al igual que el módulo de Panel de Control, éste módulo no se relaciona directamente con ningún otro pero está disponible siempre para todos los usuarios.

5. Diagrama entidad-relación

Ilustración 33. Diagrama entidad-relación para SmartFlow.



V. DISEÑO

A. DIAGRAMAS UML

Para la definición de UML, sus fundamentos y los diferentes tipos de diagrama que se utilizan, ver Apéndice A: UML.

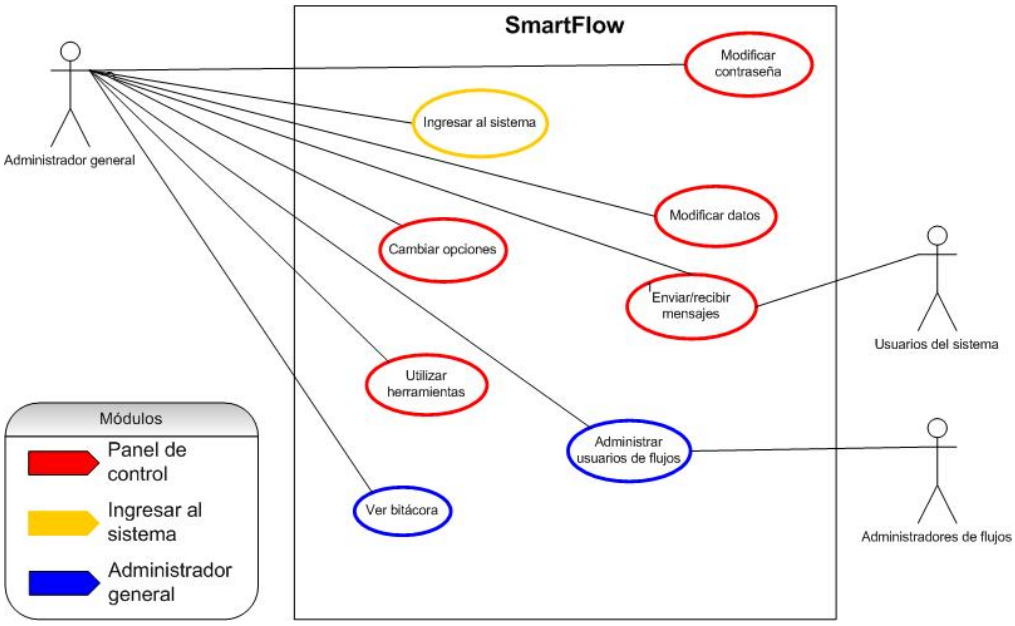
1. Diagramas de casos de uso. A continuación se presentan los diagramas de casos de uso para los dos usuarios que forman parte del módulo administrativo.

En estos diagramas se presenta de lado izquierdo el usuario específico; en el centro se observan las operaciones que él puede realizar dentro del sistema y de lado derecho aparecen los usuarios con los que interactúa y que él crea dentro del sistema.

Los casos de uso generales, es decir, los que están disponibles para todos los usuarios son: Modificar contraseña, modificar datos personales, ingresar al sistema, cambiar opciones y utilizar herramientas; tal y como fue descrito en el análisis. De estos módulos, el único que interactúa con otros usuarios es el de mensajes como se puede apreciar en las ilustraciones 34 y 35. Cada diagrama muestra en la parte inferior izquierda una leyenda indicando el módulo al cual pertenece el caso de uso marcado utilizando distintos colores.

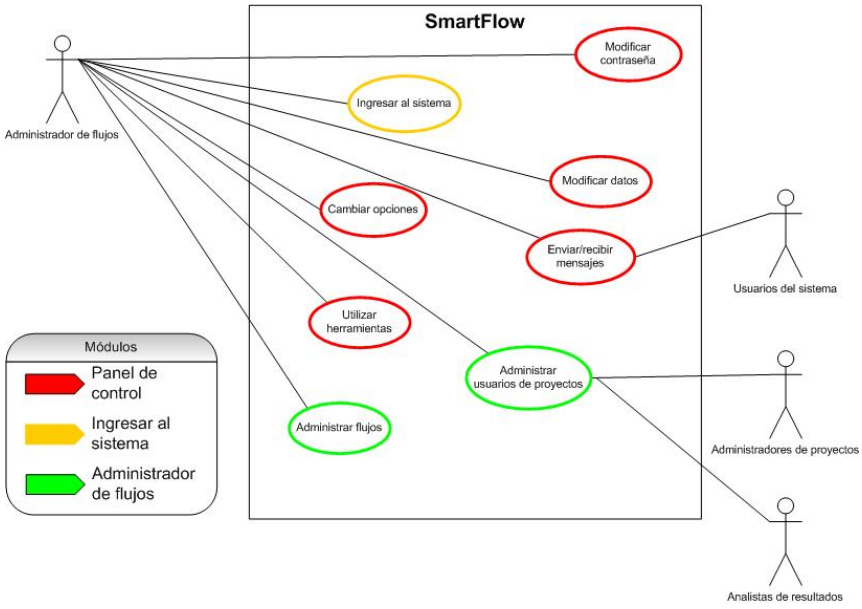
a. Administrador general. El siguiente diagrama muestra las operaciones específicas o casos de uso a los cuales el administrador general tiene acceso, siendo estos: la administración de usuario de flujos y la consulta de la bitácora de ingresos, así como los generales ya descritos anteriormente. (Ver Módulo: Administración de usuarios de flujos y módulo: Bitácora de ingresos al sistema en el análisis).

Ilustración 34. Diagrama de caso de uso para el administrador general.



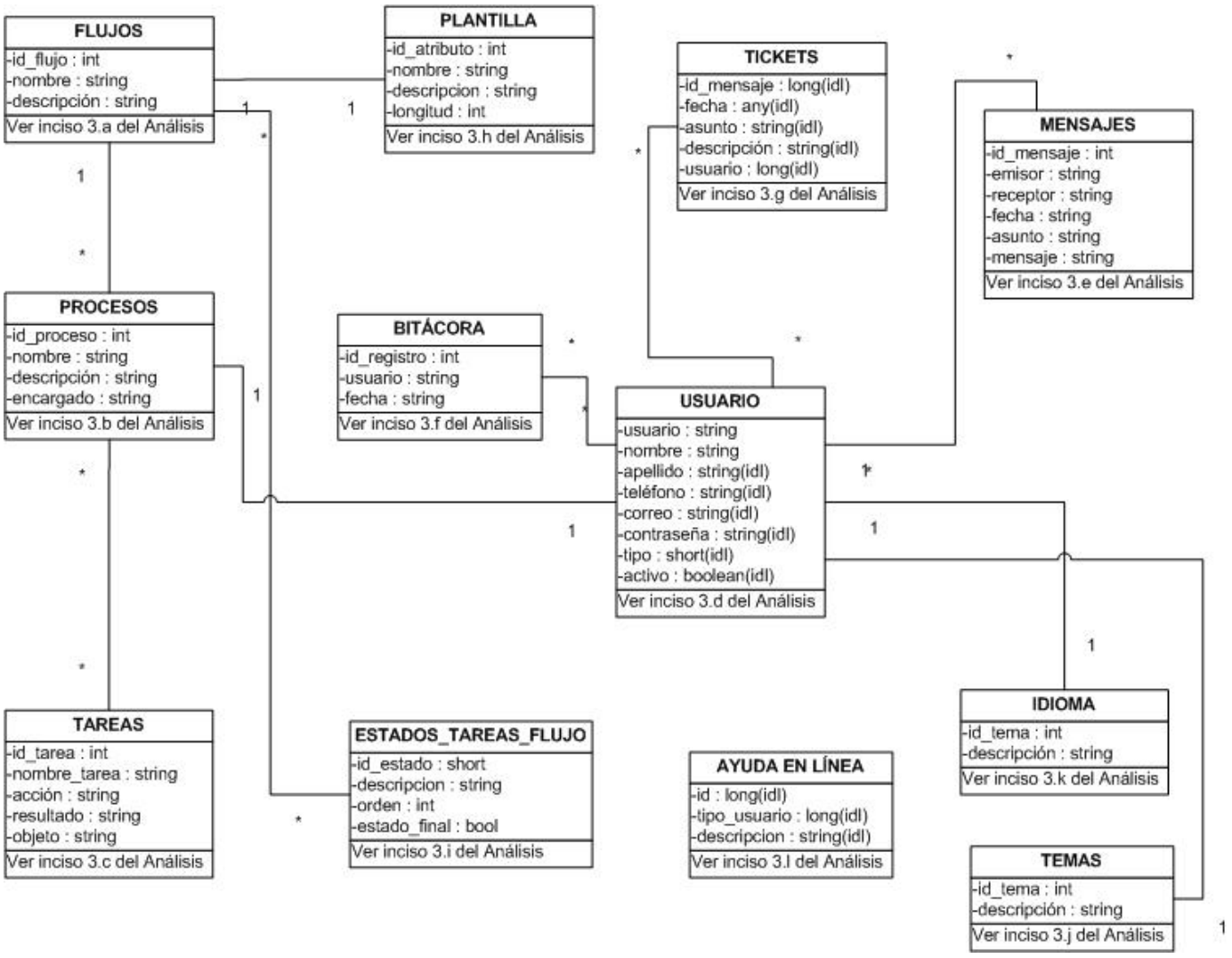
b. Administrador de flujos. Este usuario tiene acceso a los mismos casos de uso descritos anteriormente como generales; pero cuenta además con dos módulos adicionales que se utilizan para la administración de los usuarios de proyectos: Administradores de proyectos y analistas de resultados. Estos casos de uso pueden observarse en la siguiente ilustración y se puede obtener una referencia de ellos en el Módulo de usuarios de proyectos; y Módulo de flujos.

Ilustración 35. Diagrama de caso de uso para el administrador de flujos.



2. Diagrama de clases. Las clases utilizadas en este trabajo ya fueron definidas en la sección de análisis (Ver Descripción de clases). En la ilustración 36 se proporciona una representación gráfica de dichas clases a manera de observar las relaciones entre ellas y sus atributos. En la parte inferior de los recuadros se muestra una referencia al inciso donde se detallan los métodos u operaciones de cada clase, en la parte de análisis.

Ilustración 36. Diagrama de clases para el módulo administrativo.



3. Diagramas de secuencia. Estos diagramas muestran todos los acontecimientos que se llevan a cabo dentro del sistema cuando un usuario desea realizar alguna operación. En este tipo de diagramas interactúan usuarios, pantallas, objetos y entidades, dando lugar a un mapa secuencial de paso de mensajes. Todos estos objetos se identifican en los siguientes diagramas por rectángulos y líneas verticales que muestran su período de vida en el tiempo. Además, se utilizan flechas dirigidas para mostrar la dirección en la que se envían los mensajes. El orden en el que se muestra la secuencia de eventos, es el mismo orden en el que ocurrirán dentro del sistema.

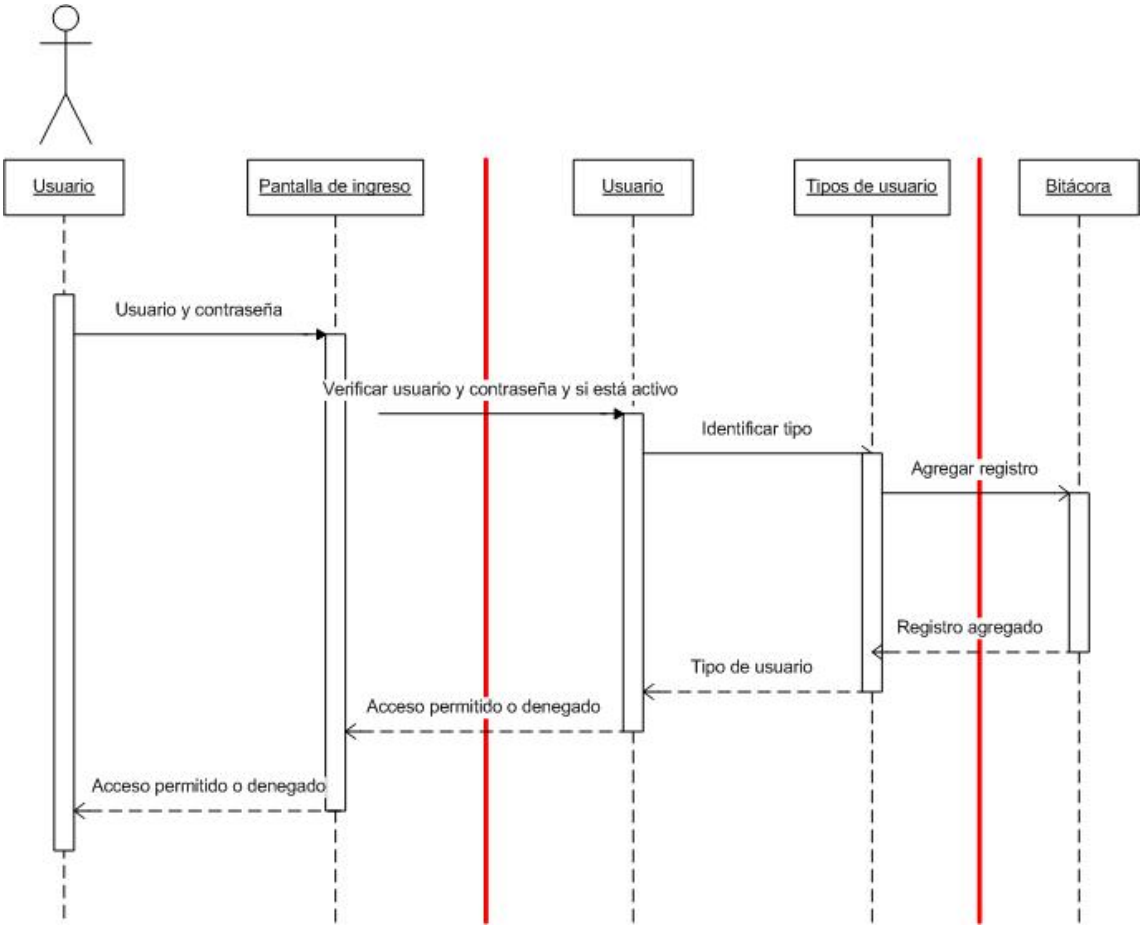
En todos los siguientes diagramas se muestra una línea vertical roja; esta línea se presenta entre una interfaz de usuario y una entidad o ente en la base de datos y representa a la clase que se encarga de la comunicación entre ambos objetos; es decir, se refiere a las funciones específicas para cada clase que ya fueron descritas en el análisis.

a. Panel de control. Como ya se ha mencionado, todos los usuarios tienen acceso al módulo panel de control. A continuación se presenta un diagrama de secuencia por cada sub-módulo que compone el panel de control.

1) Ingreso al sistema. El siguiente diagrama consta de cuatro objetos: el usuario que desea ingresar, la pantalla del sistema que le permite dicha operación, la clase encargada de proveer esta funcionalidad y de la tabla en la base de datos donde se almacena la información del usuario. El usuario le indica al sistema que desea ingresar proveyendo para esto su nombre de usuario y su contraseña.

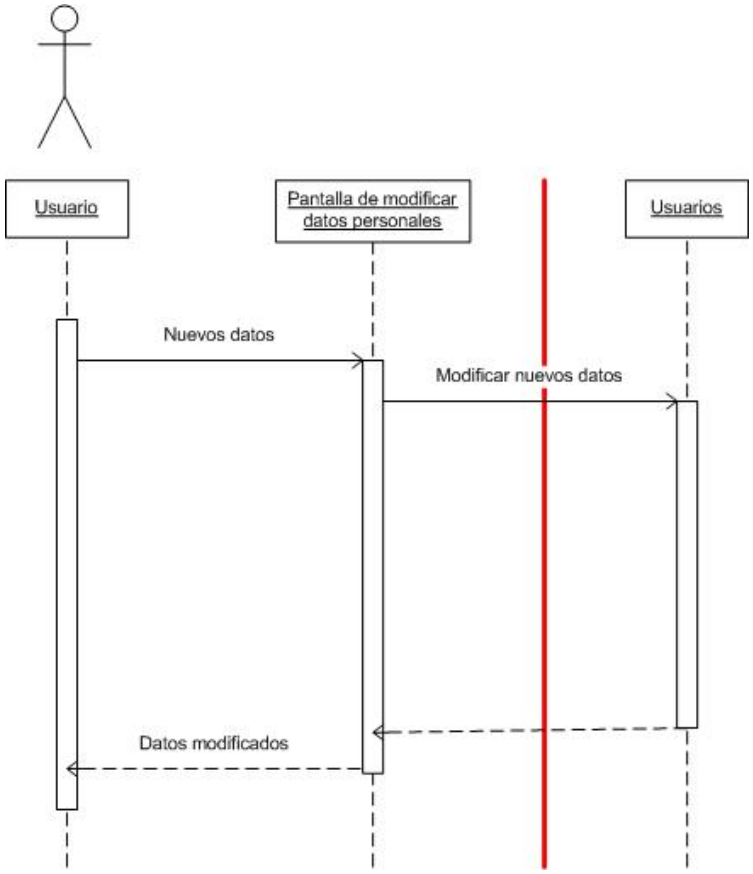
Esta interfaz se comunica con la clase USUARIOS, específicamente con la operación ingresar que a su vez se comunica directamente con la tabla de usuarios para verificar si la contraseña y el usuario proveídos son válidos y si el usuario se encuentra activo dentro del usuario. En cualquier caso, se produce un flujo de información desde la tabla hasta el usuario. La contraseña del usuario almacenada en la base de datos se encuentra cifrada para mayor seguridad, utilizando un mecanismo de cifrado de una sola vía, denominado MD5 (Ver Glosario). Además, se incluye un acceso a la tabla bitácora para dejar constancia del ingreso del usuario.

Ilustración 37. Diagrama de secuencia para el ingreso de usuarios.



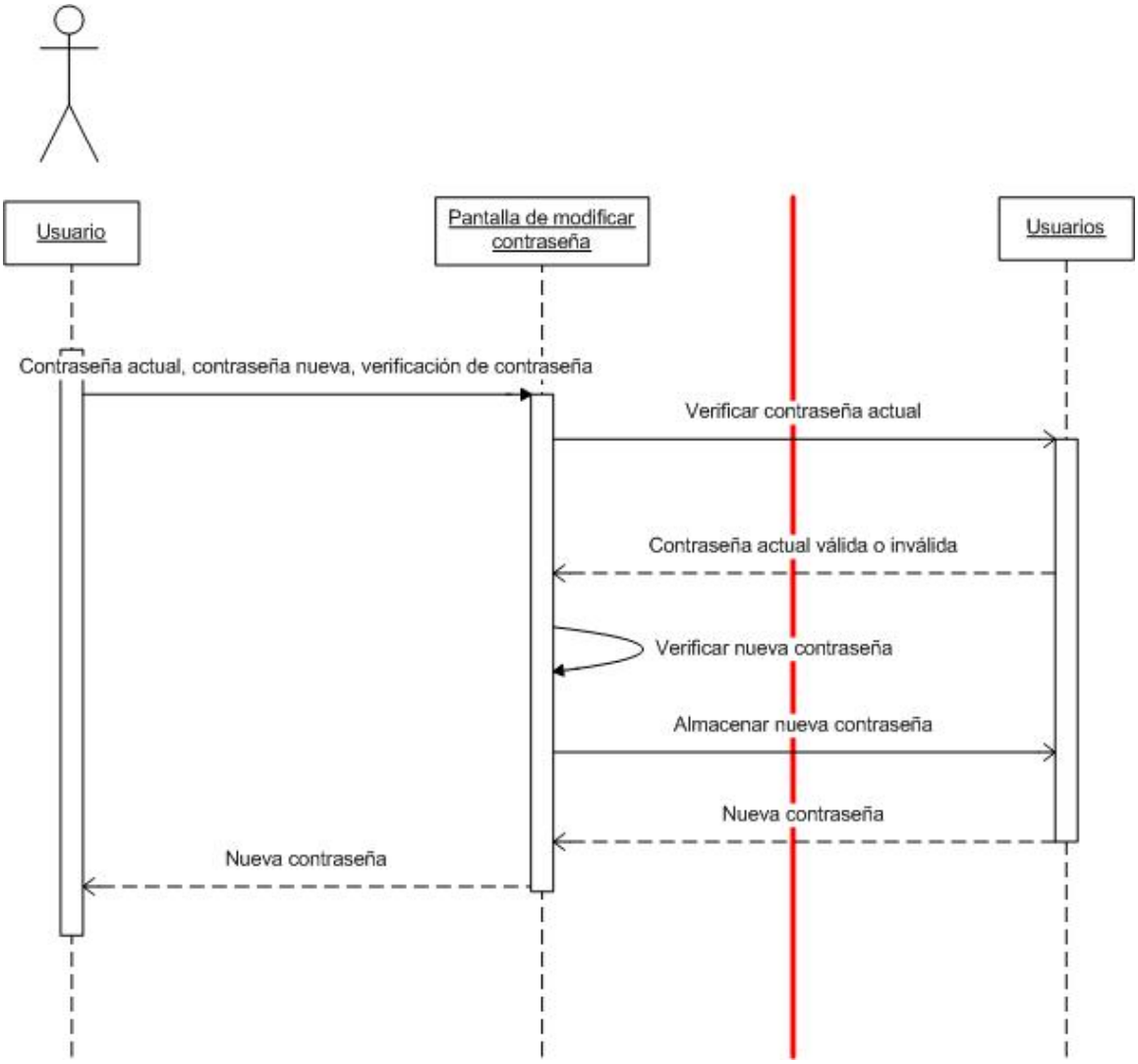
2) Modificar datos personales. La ilustración 38 muestra la secuencia de eventos en el caso que un usuario desee modificar sus datos personales dentro del sistema. Esto incluye la interacción con la clase usuarios y la operación específica de EditarUsuario. Esta clase se comunica con la tabla usuarios y con la interfaz proveída por el sistema, en donde el usuario especifica sus nuevos datos.

Ilustración 38. Diagrama de secuencia para la modificación de datos personales.



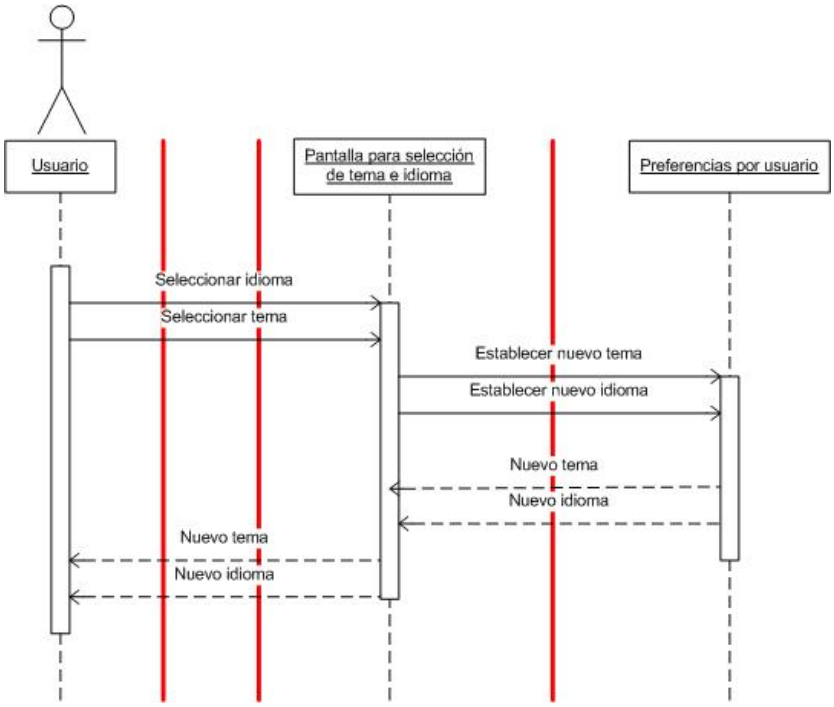
3) Modificar contraseña. El siguiente diagrama muestra los objetos que interactúan en el caso que un usuario desee modificar su contraseña. Se utiliza de nuevo la clase usuarios, y específicamente la operación almacenarNuevaContraseña para esta función. El usuario le indica al sistema que desea modificar su contraseña a través de la interfaz diseñada para esto. En esta, el usuario debe ingresar su contraseña actual, su nueva contraseña y una verificación de la misma. La contraseña actual se verifica en la base de datos en la tabla usuarios y la verificación de la nueva contraseña se hace al nivel de la interfaz anteriormente mencionada.

Ilustración 39. Diagrama de secuencia para la modificación de contraseña.



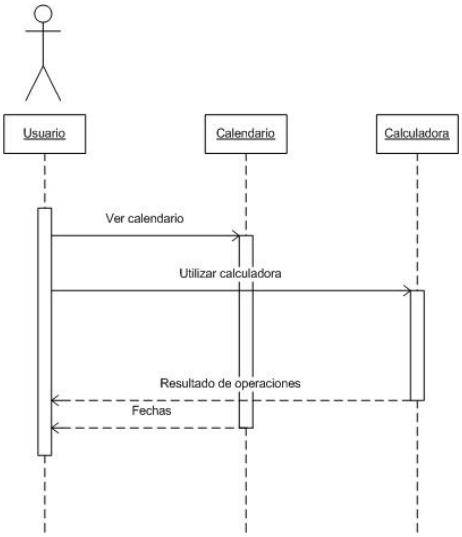
4) Establecer preferencias. La siguiente ilustración muestra los mensajes que se envían en el sistema cuando un usuario cambia sus preferencias dentro del mismo. Estas preferencias son: el tema o los colores del sistema y el idioma. Independientemente del orden en que realice alguna de estas operaciones, el usuario envía un mensaje al sistema indicando que la desea modificar. Inmediatamente el sistema se comunica con la clase usuarios y utiliza la operación establecerPreferencias para almacenar dichos cambios y automáticamente mostrarlos al usuario.

Ilustración 40. Diagrama de secuencia para establecer preferencias (tema e idioma).



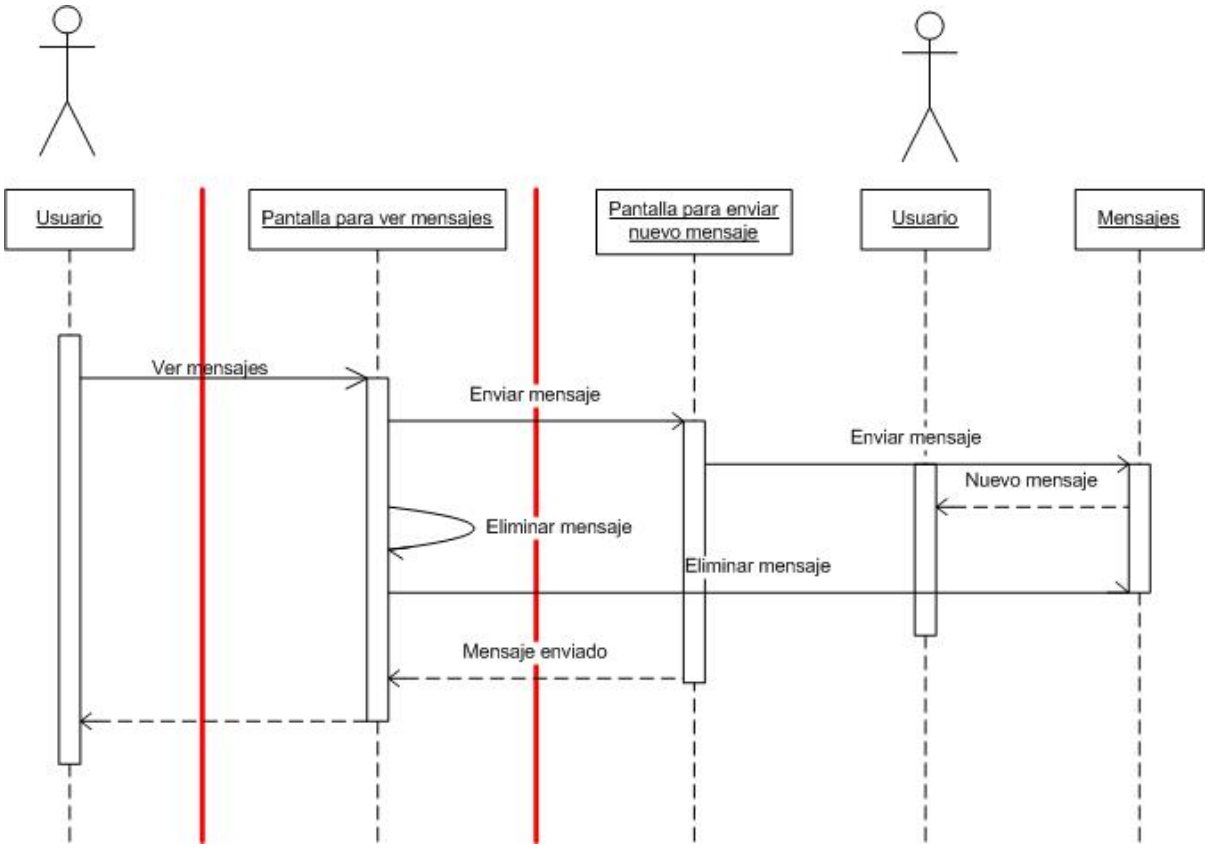
5) Uso de herramientas. El siguiente diagrama muestra la interacción de un usuario con las herramientas que SmartFlow provee y que están disponibles para los usuarios que se manejan en el presente trabajo: Administrador general y Administrador(es) de flujos.

Ilustración 41. Diagrama de secuencia para las herramientas del sistema (calendario y calculadora).



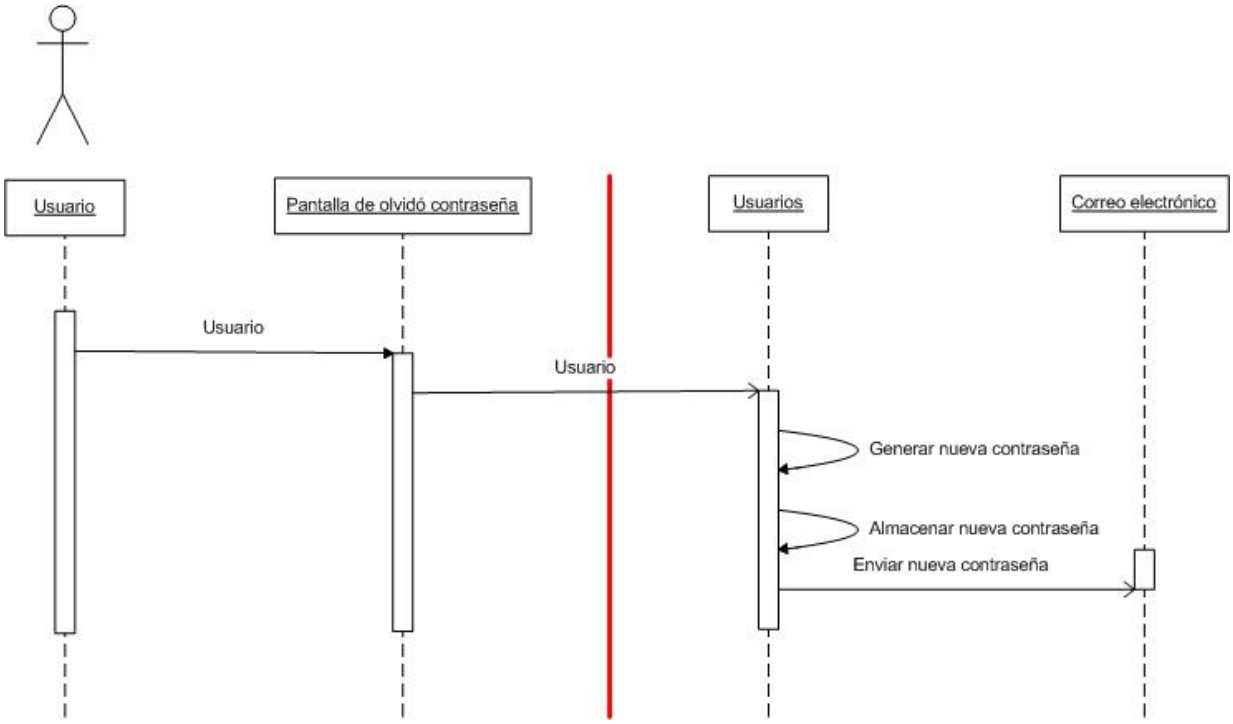
6) Mensajes. La ilustración 42 muestra la interacción entre dos usuarios en el sistema utilizando la herramienta de mensajes. Aquí, un usuario se comunica con otro, a través de la clase mensajes y sus respectivas funciones. Las operaciones disponibles dentro del sistema de mensajes son: ver listado de mensajes, enviar mensaje nuevo y eliminar mensaje.

Ilustración 42. Diagrama de secuencia para enviar y recibir mensajes.



7) Obtener nueva contraseña. En este diagrama se muestran los mensajes que se envían dentro del sistema cuando un usuario haya olvidado su contraseña y desee obtener una nueva. Por motivos de seguridad, SmartFlow no provee la misma contraseña al usuario, sino que existe una función específica que genera una nueva contraseña de manera totalmente aleatoria, la cual a través de la clase USUARIOS, se almacena en la base de datos como nueva contraseña para el usuario.

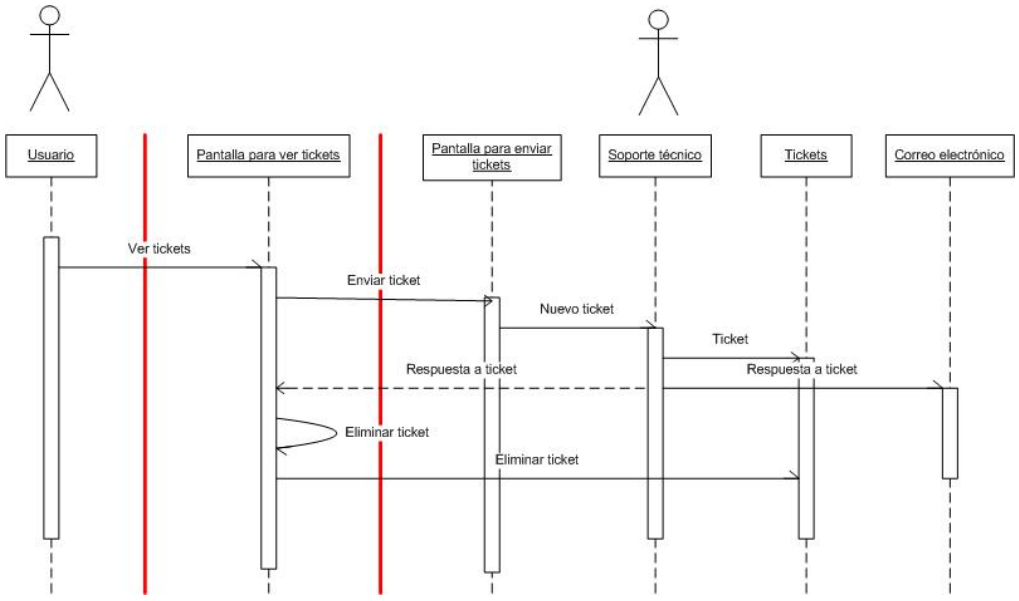
Ilustración 43. Diagrama de secuencia para obtener una nueva contraseña.



b. Ayuda y soporte técnico. Al igual que con el módulo de panel de control, todos los usuarios tienen acceso al módulo de ayuda y soporte técnico. A continuación se presenta un diagrama de secuencia por cada sub-módulo que lo compone.

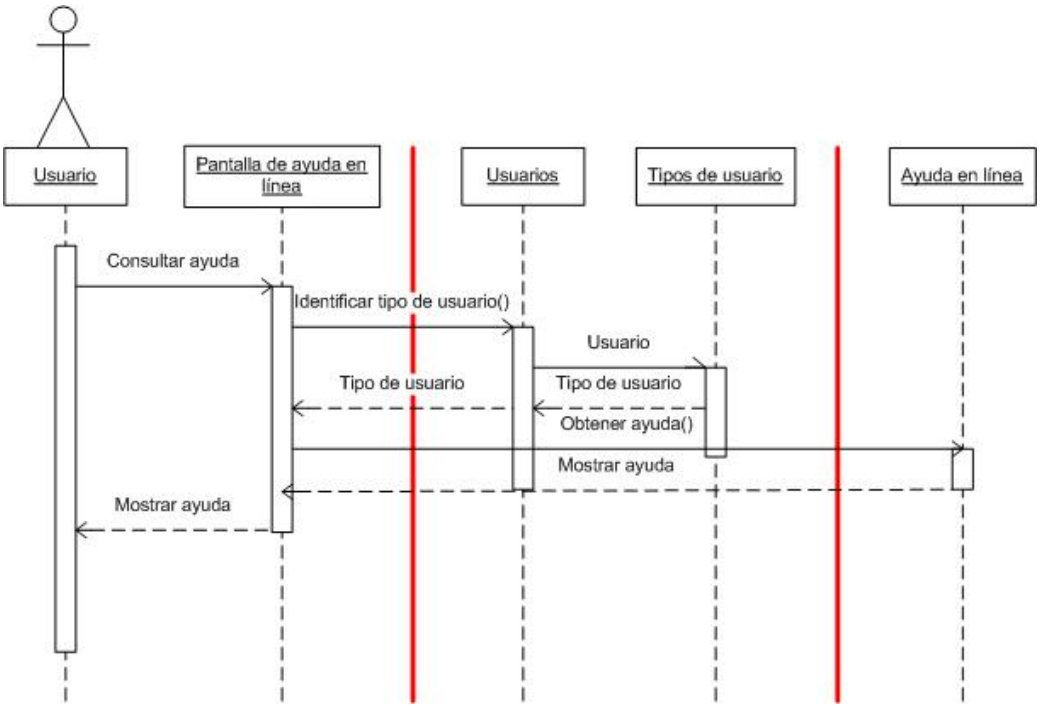
1) Sistema de tickets. El siguiente diagrama muestra el mecanismo que un usuario con alguna duda o problema técnico utilizaría para comunicarse con algún técnico. Esta duda sería resuelta dentro del mismo sistema utilizado mensajes y a través de correo electrónico. La única clase que interactúa en este diagrama es la de tickets, en donde se almacenan todas las dudas enviadas por los usuarios.

Ilustración 44. Diagrama de secuencia para uso de servicio técnico.



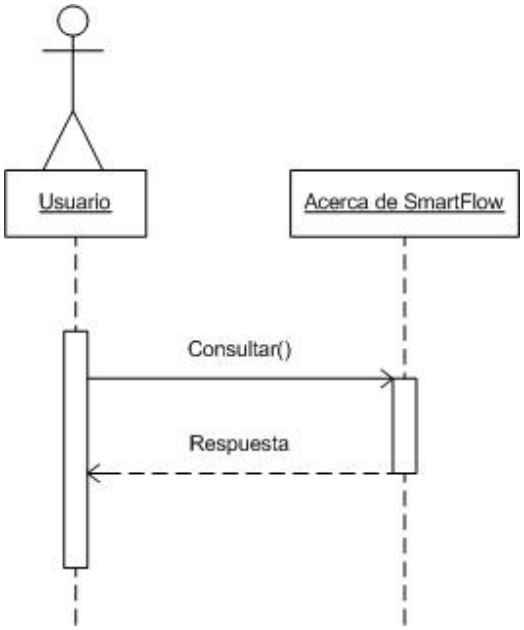
2) Ayuda en línea. La información disponible como ayuda dentro del sistema para los usuarios se trata de información dinámica (Ver clase ayuda en línea en ilustración 36). El usuario que acceda a este módulo podrá consultar cualquier tema relacionado con el funcionamiento del sistema según su tipo.

Ilustración 45. Diagrama de secuencia para uso de ayuda en línea.



3) Acerca de SmartFlow. La información que el sistema mostrará aquí es totalmente estática, no cambiará. El usuario únicamente accede a esta sección y obtiene como respuesta inmediata la información general de SmartFlow.

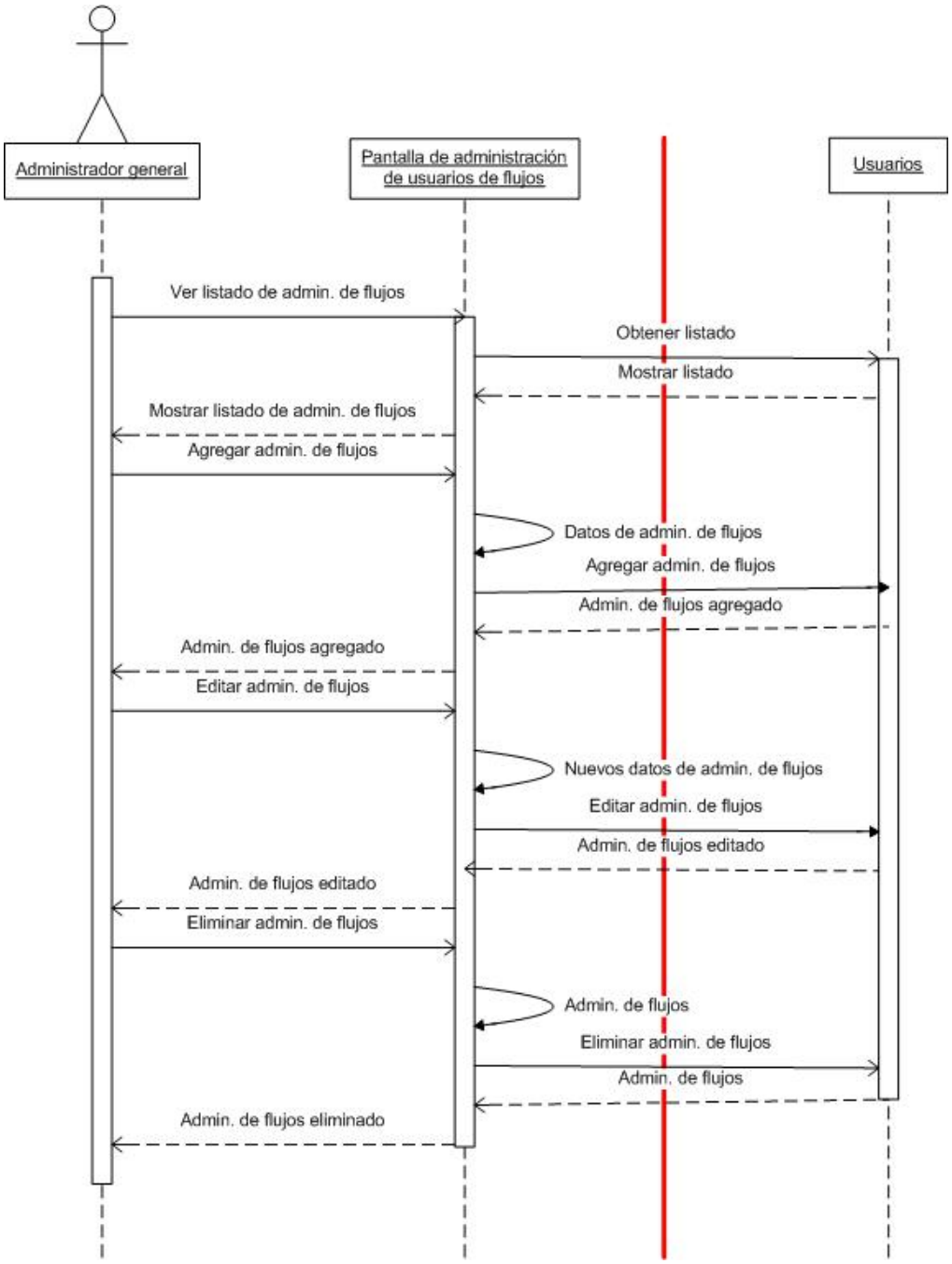
Ilustración 46. Diagrama de secuencia para uso de Acerca de SmartFlow.



c. Administrador General. A continuación se muestran los diagramas de secuencia para los módulos exclusivos del administrador general.

1) Usuarios de flujos. La siguiente ilustración presenta la interacción del Administrador general con el sistema para administrar los usuarios de flujos, para lo que se utiliza la clase USUARIOS.

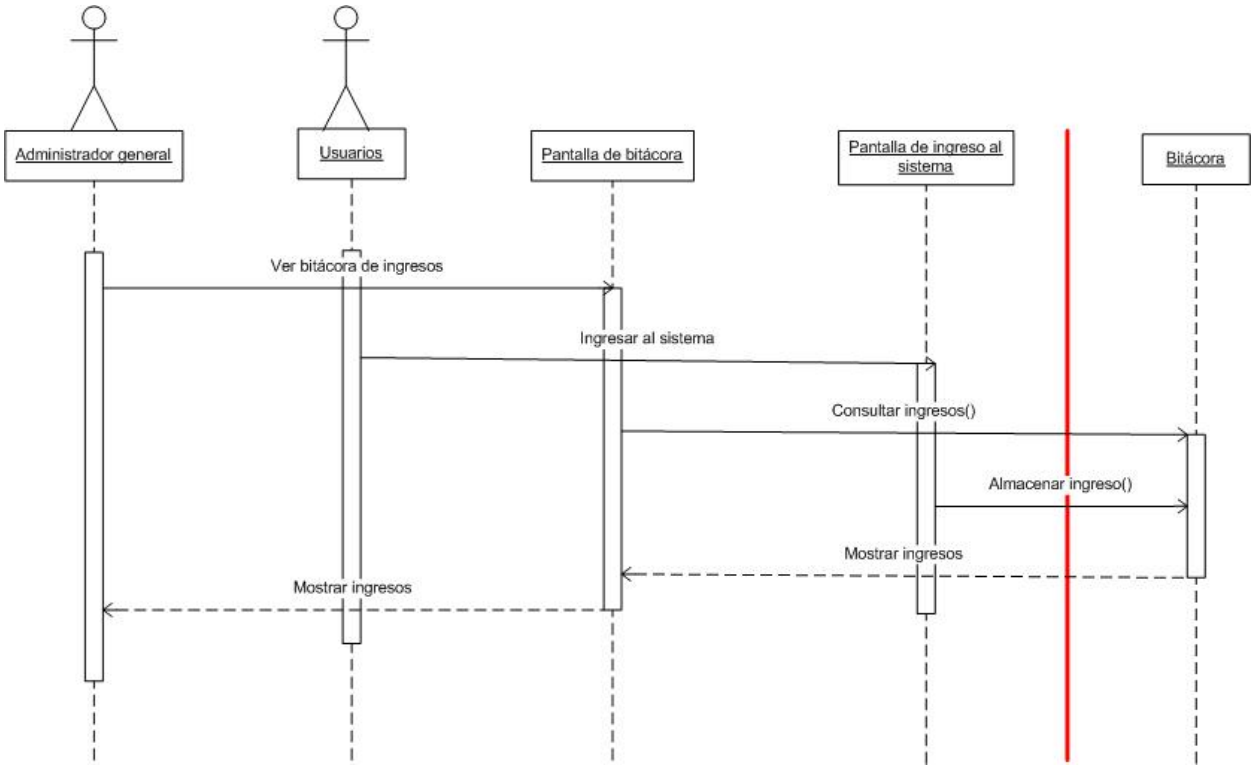
Ilustración 47. Diagrama de secuencia para módulo de usuarios de flujos.



2) Bitácora. A través de la bitácora el Administrador general puede consultar el ingreso de cualquier usuario al sistema. En este diagrama se muestra la participación de los usuarios en el sistema, utilizando para esto la pantalla de ingreso (Ver ilustración 37). La operación agregarRegistro de la clase BITÁCORA refleja

automáticamente dicho ingreso, el cual es notificado al Administrador a través de la pantalla de bitácora.

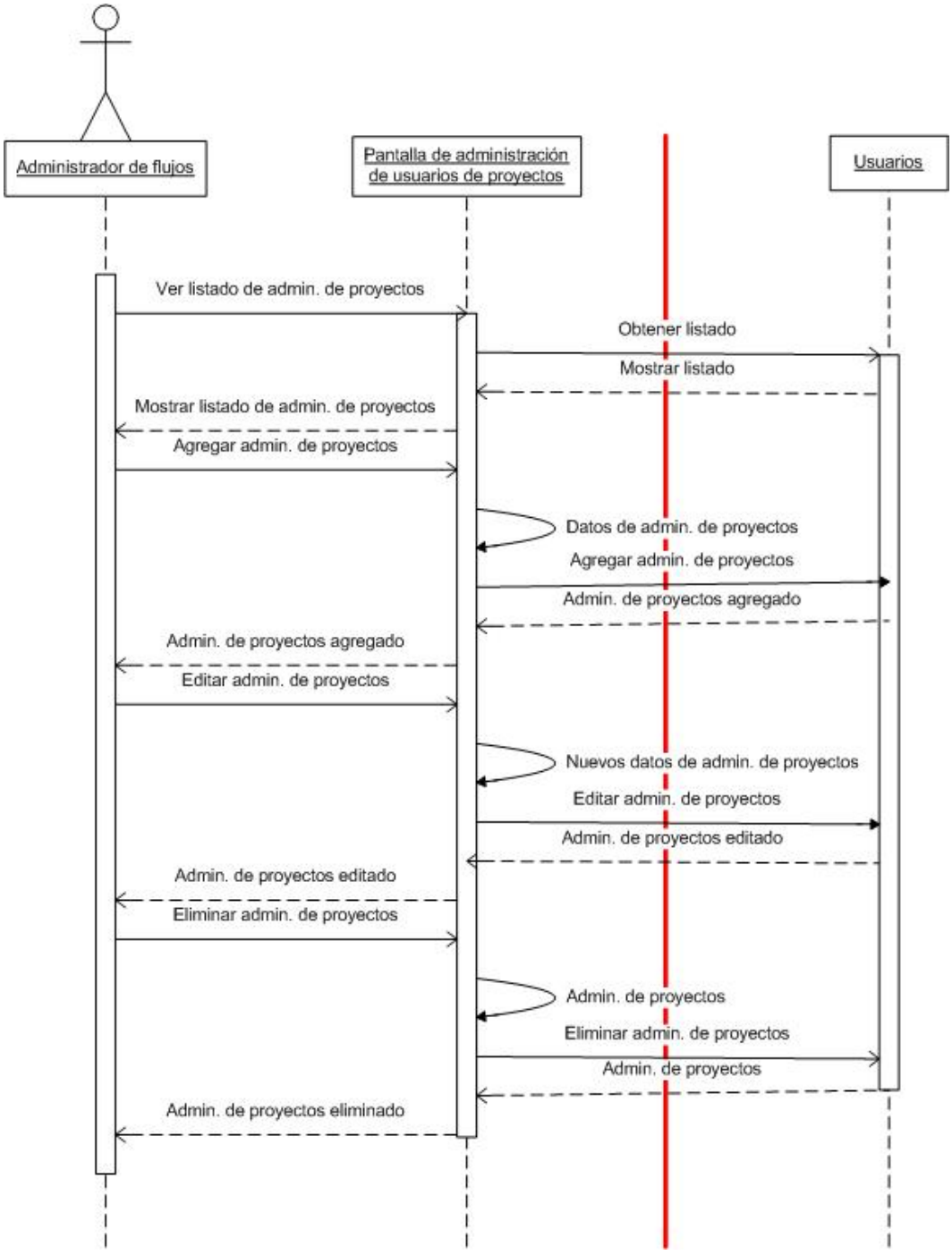
Ilustración 48. Diagrama de secuencia para módulo de bitácora de ingresos.



d. Administrador de flujos. A continuación se muestran los diagramas de secuencia para los módulos exclusivos de un Administrador de flujos.

1) Usuarios de proyectos. La siguiente ilustración muestra la interacción de un Administrador de flujos con el sistema para realizar operaciones sobre los usuarios de proyectos, es decir: Administradores de proyectos y analistas de resultados. Para esto, se requiere la clase usuarios junto con la tabla usuarios en la base de datos.

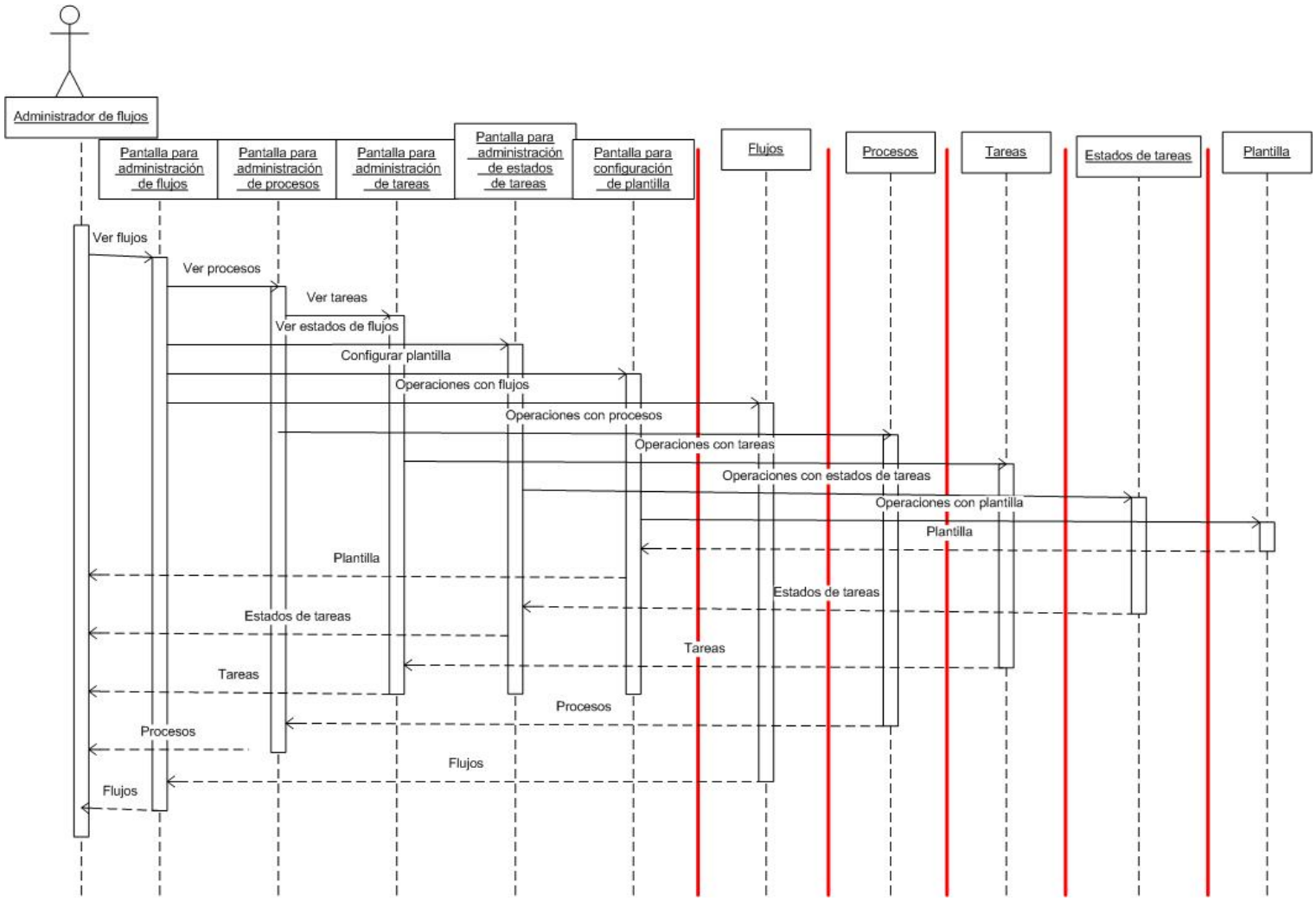
Ilustración 49. Diagrama de secuencia para módulo de usuarios de proyectos.



2) Flujos. La siguiente ilustración muestra los mensajes que se envían dentro del sistema para la administración de los flujos de trabajo. Esta ilustración muestra cuatro interfaces o pantallas y cuatro tablas, así como las clases representadas por

las líneas verticales. Las interfaces son para la administración de los flujos, procesos que conforman el flujo, tareas dentro de los procesos y los estados de tareas personalizados. Cada interfaz tiene relacionada su respectiva clase y utiliza las operaciones definidas en cada una (Ver definición de clases en el análisis) para mostrar al Administrador de flujos todos los flujos y objetos relacionados que él ha creado dentro del sistema.

Ilustración 50. Diagrama de secuencia para módulo de flujos.



B. DISEÑO Y DESCRIPCIÓN DE INTERFACES GRÁFICAS

Las ilustraciones que se muestran enseguida proporcionan una idea general del funcionamiento del Módulo administrativo. Estas ilustraciones son las interfaces

presentadas a un usuario por SmartFlow, la cuales están divididas por usuario. Adicionalmente se presentan al inicio las interfaces generales que están disponibles para todos los usuarios. Por cada interfaz se provee una explicación de la misma y de su utilización.

1. Generales

a. Panel de control

1) Ingreso al sistema. Esta interfaz le permite a un usuario ingresar al sistema, indicando su nombre de usuario y contraseña los cuales son verificados contra la base de datos. Específicamente, la contraseña es cifrada utilizando el mecanismo MD5 antes de enviarse al servidor para su verificación; de esta forma la contraseña nunca viaja en texto plano (plain-text) a través de la red y la comparación se hace entre ambas contraseñas cifradas. Si el usuario o la contraseña proporcionados por el usuario son inválidos, el sistema mostrará un mensaje de error.

Ilustración 51. Interfaz de ingreso al sistema.

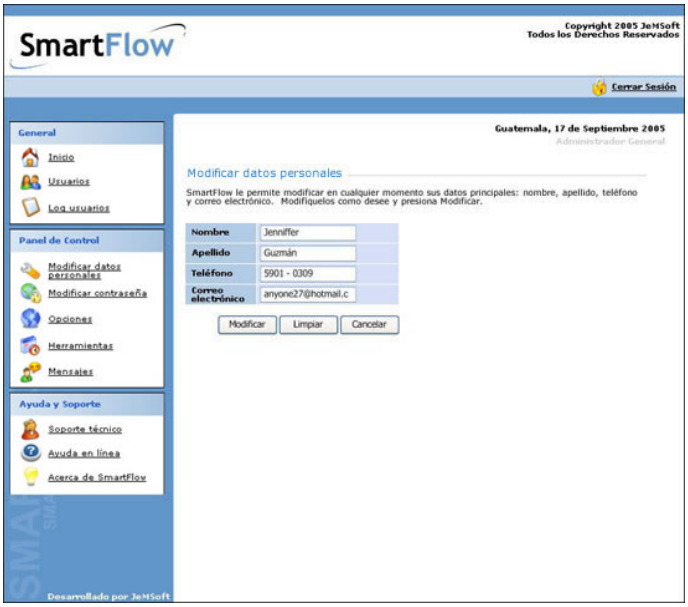
Todas las interfaces dentro del sistema cuentan con un menú de opciones del lado izquierdo: general, panel de control y ayuda, y soporte técnico. Las opciones dentro de general son las opciones disponibles para el usuario según su tipo, mientras que las de panel de control y las de ayuda y soporte pueden utilizarse por todos los usuarios. Además, en la parte superior derecha se muestra la fecha, así como la opción para cerrar la sesión.

En caso que el ingreso haya sido exitoso, SmartFlow muestra una pantalla de inicio donde se da una breve descripción del sistema y algunas instrucciones. Automáticamente se detecta el tipo de usuario que acaba de ingresar y se muestra en la parte superior derecha debajo de la fecha. Esta pantalla puede accederse a través de la opción inicio en el menú general del primer cuadro a la izquierda.

Ilustración 52. Interfaz de inicio del sistema.

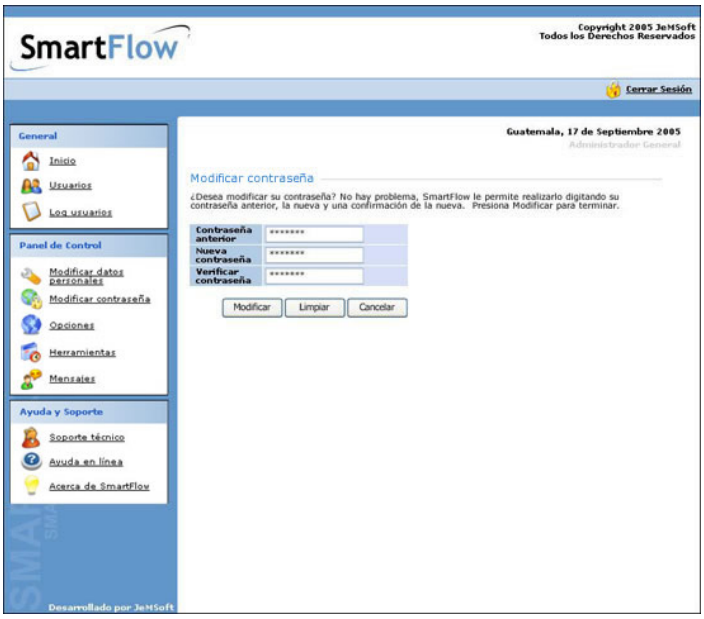
2) Modificar datos personales. Esta es una de las opciones disponibles para todos los usuarios. La Ilustración 53 muestra la interfaz donde aparecen los datos personales del usuario almacenados en el sistema y le permite modificarlos. Esta pantalla puede accederse a través de la opción modificar datos personales en el menú panel de control del segundo cuadro a la izquierda.

Ilustración 53. Interfaz para la modificación de datos personales.



3) Modificar contraseña. La siguiente interfaz permite a un usuario modificar su contraseña actual para el ingreso al sistema. Esto se logra ingresando en el primer campo de texto la contraseña actual, en el segundo la nueva contraseña, y en el tercero se requiere una verificación de la nueva contraseña. Esta pantalla puede accederse a través de la opción modificar contraseña en el menú panel de control del segundo cuadro a la izquierda.

Ilustración 54. Interfaz para la modificación de la contraseña

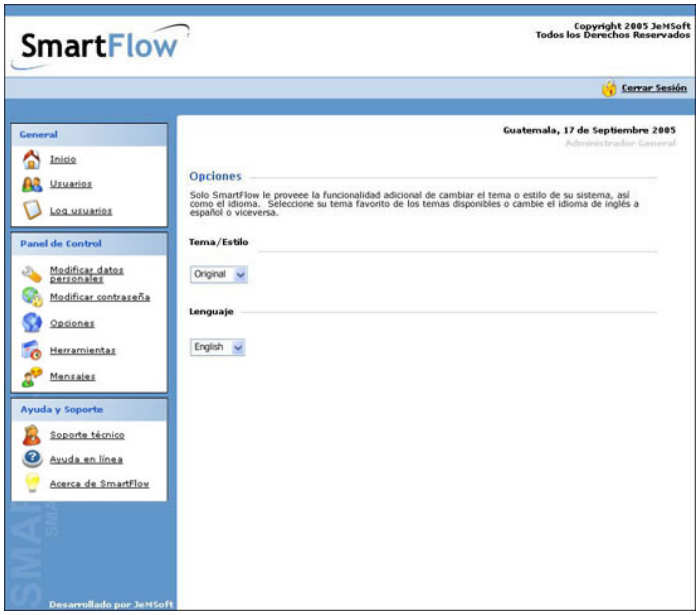


4) Olvidó contraseña. En caso que un usuario olvide su contraseña para ingresar al sistema, puede obtener una nueva a través de esta interfaz. Para esto se requiere que el usuario proporciona su nombre de usuario al sistema. Automáticamente se genera una nueva contraseña aleatoria para el usuario, la cual es enviada al usuario por medio de correo electrónico y es almacenada dentro del sistema. Esto interfaz puede accederse únicamente a través de la pantalla de Ingreso al sistema (Ver parte inferior de ilustración 51, debajo del botón entrar).

Ilustración 55. Interfaz para obtener una nueva contraseña.

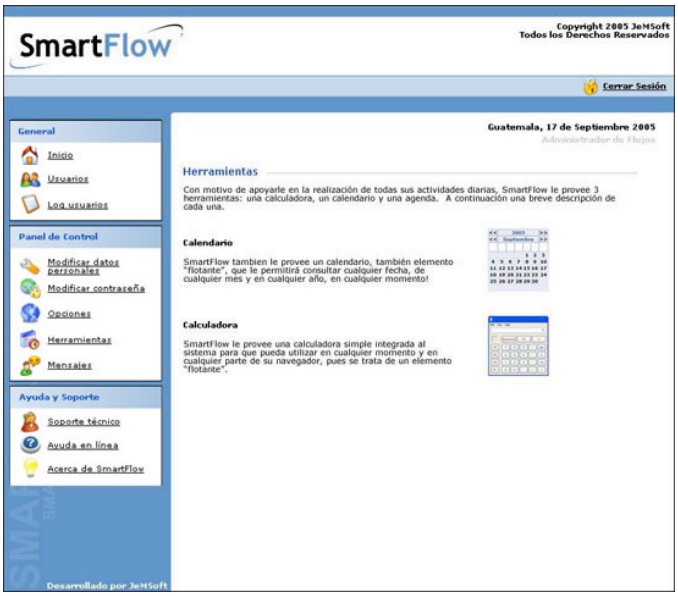
5) Establecer preferencias (opciones). Un usuario puede cambiar el aspecto de SmartFlow haciendo uso de esta interfaz. En el primer listado puede seleccionarse un tema para cambiar los colores del sistema (Ver Anexo A: Temas de SmartFlow), y en el segundo se puede cambiar el idioma, de inglés a español o viceversa. Al hacer cualquiera de estos cambios, automáticamente se ven reflejados al usuario y son almacenados como sus preferencias dentro del sistema. Esta pantalla puede accederse a través de la opción opciones en el menú panel de control del segundo cuadro a la izquierda.

Ilustración 56. Interfaz para cambiar opciones del sistema (tema e idioma).



6) Herramientas: La siguiente ilustración muestra la interfaz que SmartFlow proporciona como página principal de las herramientas disponibles. En ella se muestra únicamente una breve descripción de cada una y se provee un enlace a cada herramienta a través de su respectivo nombre, o bien la ilustración de la derecha. Esta ventana puede accederse a través de la opción herramientas en el menú panel de control del segundo cuadro a la izquierda.

Ilustración 57. Interfaz que muestra las herramientas disponibles.



La primera herramienta disponible es un calendario simple. Este calendario permite consultar cualquier fecha en cualquier momento, pues es un elemento independiente (Pop-up) de la ventana principal del sistema.

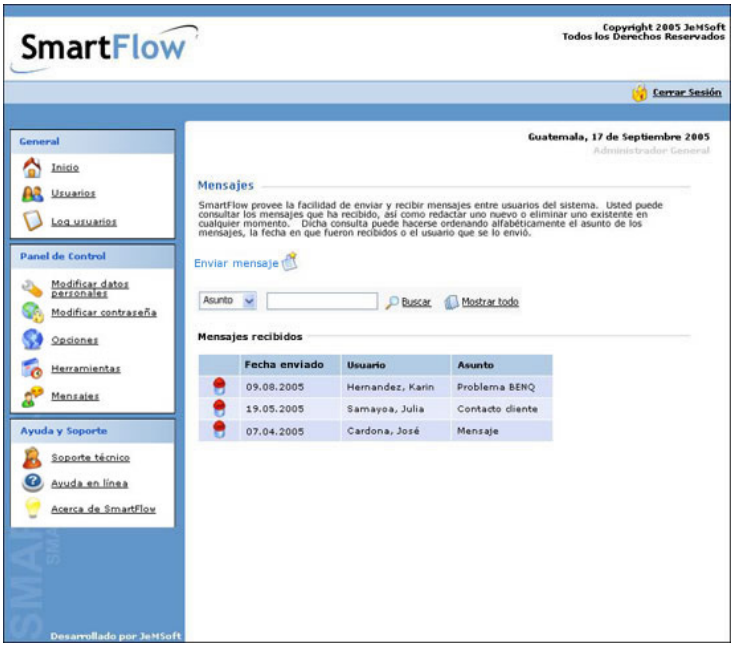
Ilustración 58. Calendario.

SmartFlow provee además una calculadora sencilla, y al igual que el calendario es un elemento independiente (Pop-up) que tiene la capacidad de estar disponible en cualquier momento.

Ilustración 59. Calculadora.

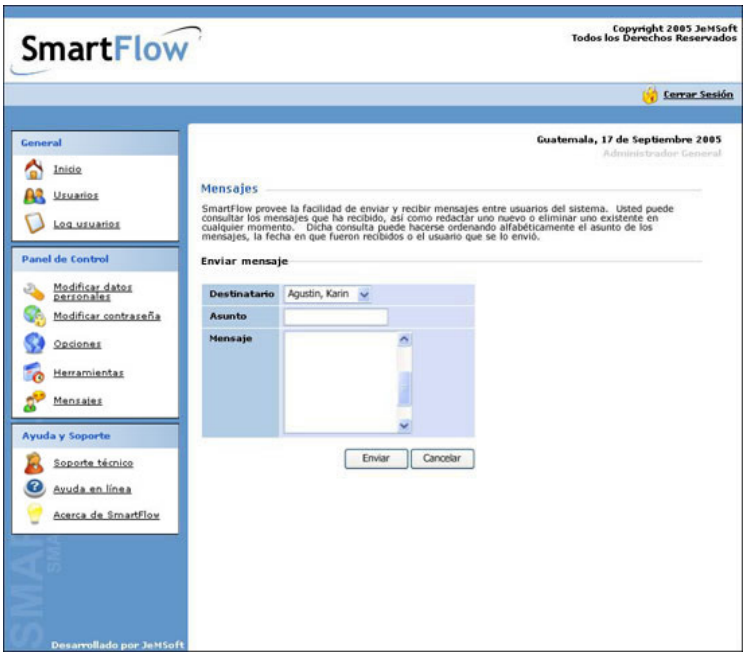
7) Mensajes. La siguiente ilustración muestra un listado de todos los mensajes que un usuario ha recibido. En la parte superior, se muestra un listado que permite establecer un criterio para buscar un mensaje específico. Estos criterios son: fecha en que fue recibido, usuario que lo envió y asunto del mensaje. Estos mismos criterios se utilizan para ordenar todos los mensajes mostrados, presionando sobre el encabezado de las columnas de la tabla. Además, el sistema le permite a un usuario eliminar un mensaje, mostrando siempre un mensaje de verificación de su deseo de eliminar el mensaje. Esta ventana puede accederse a través de la opción mensajes en el menú panel de control del segundo cuadro a la izquierda.

Ilustración 60. Interfaz que muestra los mensajes recibidos por un usuario



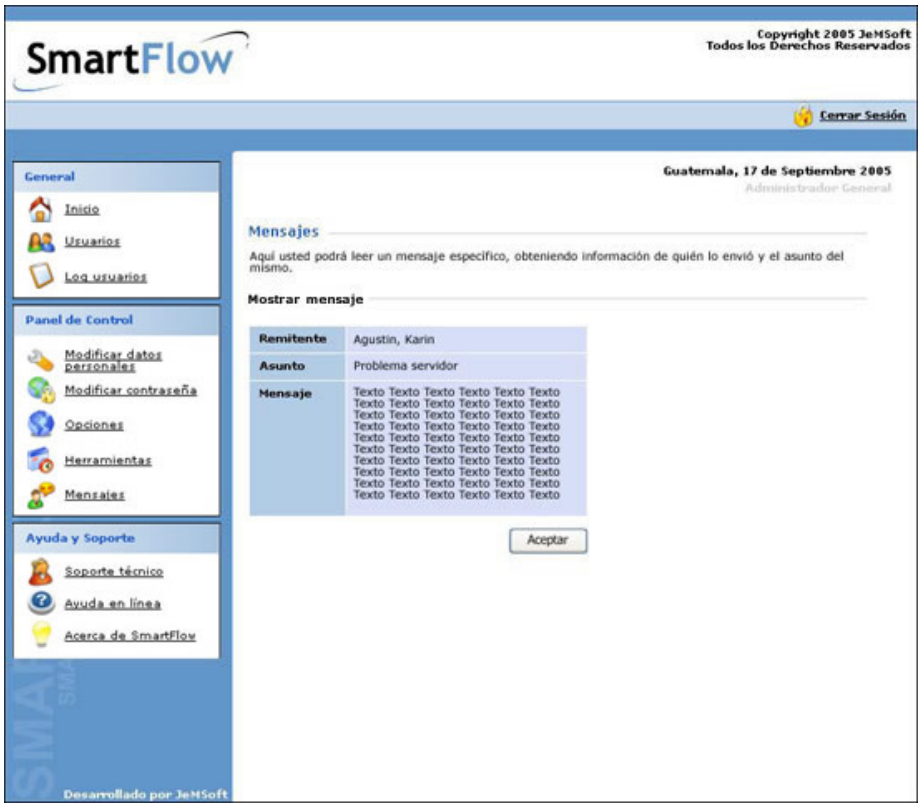
La ilustración 61 muestra la interfaz que un usuario utilizará si desea enviar un mensaje nuevo. En ésta se debe seleccionar el destinatario del listado de todos los usuarios existentes, indicar el asunto del mensaje y por último el texto del mensaje a enviar.

Ilustración 61. Interfaz para enviar un mensaje nuevo.



La siguiente ilustración muestra la interfaz que permite a un usuario leer un mensaje específico. Esto se hace seleccionando la fila del mensaje a consultar en la tabla de la Ilustración 60.

Ilustración 62. Interfaz para leer un nuevo mensaje.



b. Ayuda y soporte

1) *Soporte técnico.* La interfaz de soporte técnico le permite a un usuario enviar dudas o comentarios a los técnicos del sistema para recibir soporte. No es necesario indicar el destinatario, pues el sistema automáticamente envía el mensaje a una dirección de correo predeterminada. La respuesta a este mensaje se hace dentro del sistema a través de la administración de tickets y a través de correo electrónico. Esta interfaz puede accederse a través de la opción soporte técnico en el menú ayuda y soporte del tercer cuadro a la izquierda.

Ilustración 63. Interfaz para mostrar los tickets de respuesta que un usuario ha recibido.

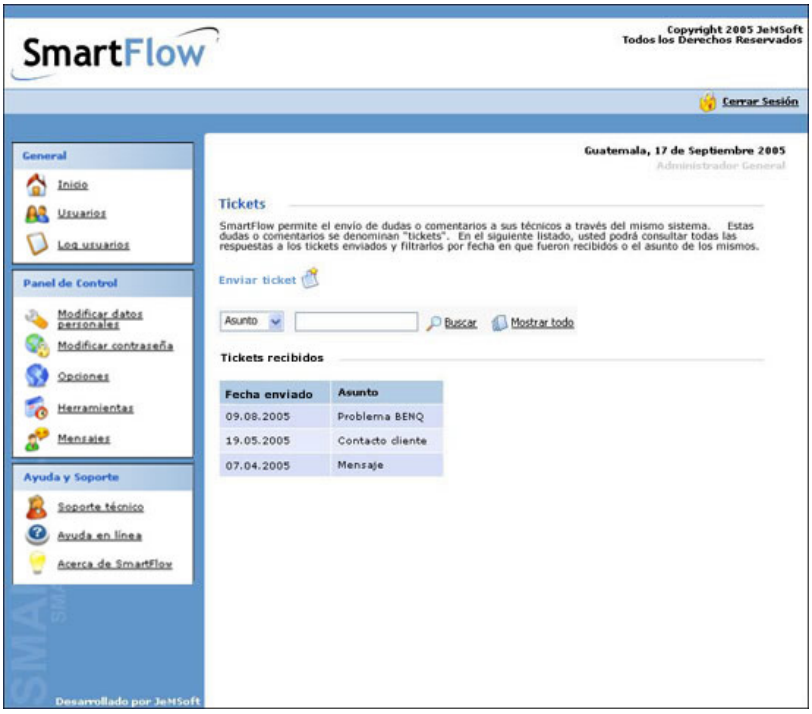


Ilustración 64. Interfaz para enviar tickets de comentarios y dudas al soporte técnico de SmartFlow.

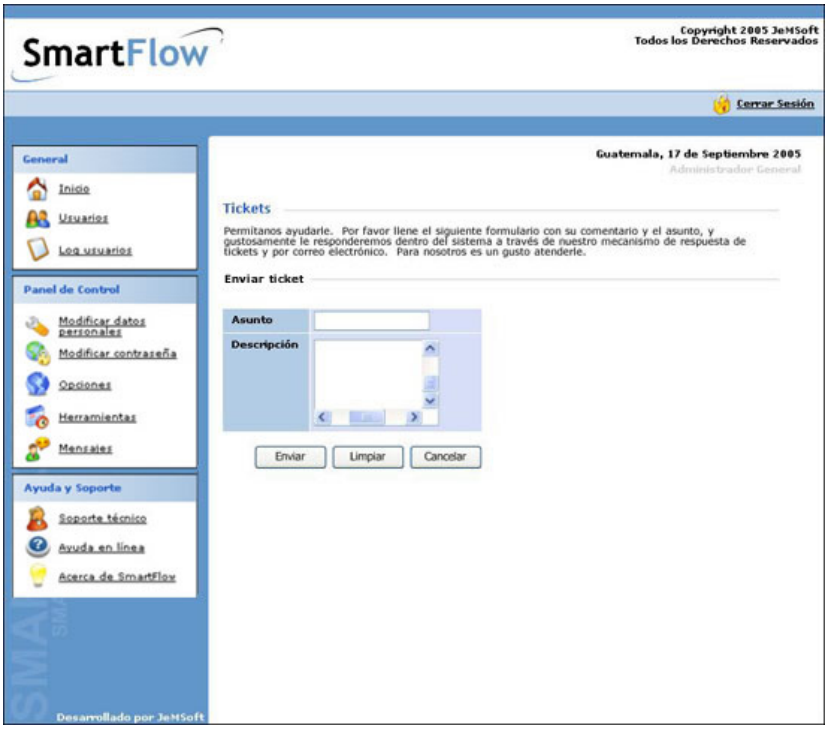
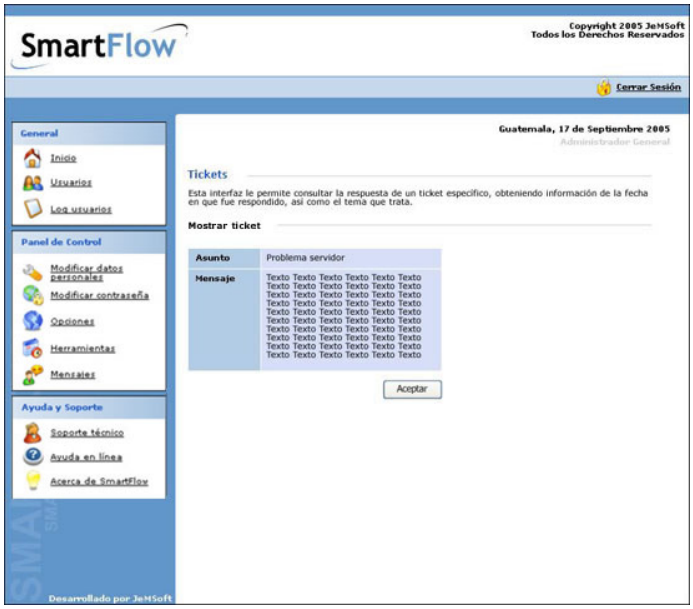
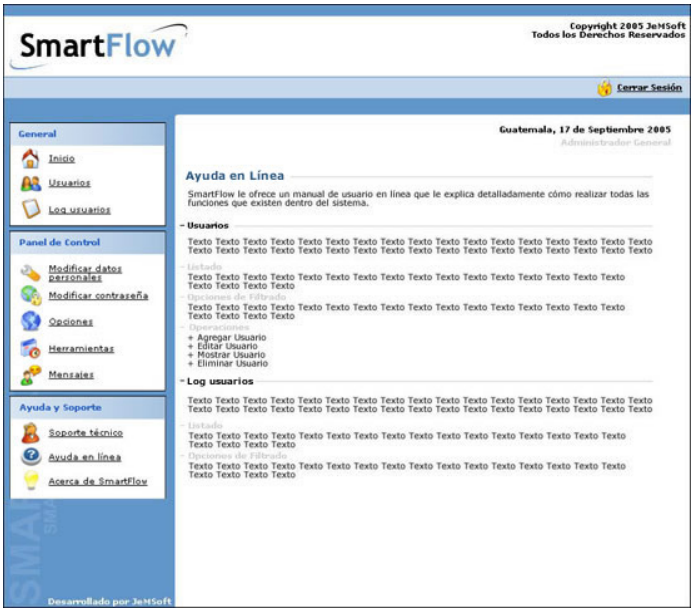


Ilustración 65. Interfaz para leer un ticket de respuesta del soporte técnico de SmartFlow.



2) Ayuda en línea. La siguiente Ilustración muestra la interfaz donde un usuario, según su tipo, puede consultar sobre el funcionamiento de las operaciones que él puede realizar dentro del sistema. Para esto, se ofrece una descripción detallada de cada operación. Esta interfaz puede accederse a través de la opción ayuda en línea en el menú ayuda y soporte del tercer cuadro a la izquierda.

Ilustración 66. Interfaz para consulta de ayuda en línea.



3) Acerca de. Esta ilustración muestra la interfaz que se le presenta a un usuario cuando éste desee consultar la información y características de SmartFlow. Puede accederse a través de la opción Acerca de SmartFlow en el menú ayuda y soporte del tercer cuadro a la izquierda.

Ilustración 67. Interfaz que muestra información general de SmartFlow.

2. Administrador general. Las interfaces a mostrar a continuación únicamente pueden ser vistas por el administrador general del sistema.

a. Usuarios de Flujos. La interfaz para la administración de usuarios de flujos puede accederse a través de la opción usuarios en el menú general del primer cuadro a la izquierda.

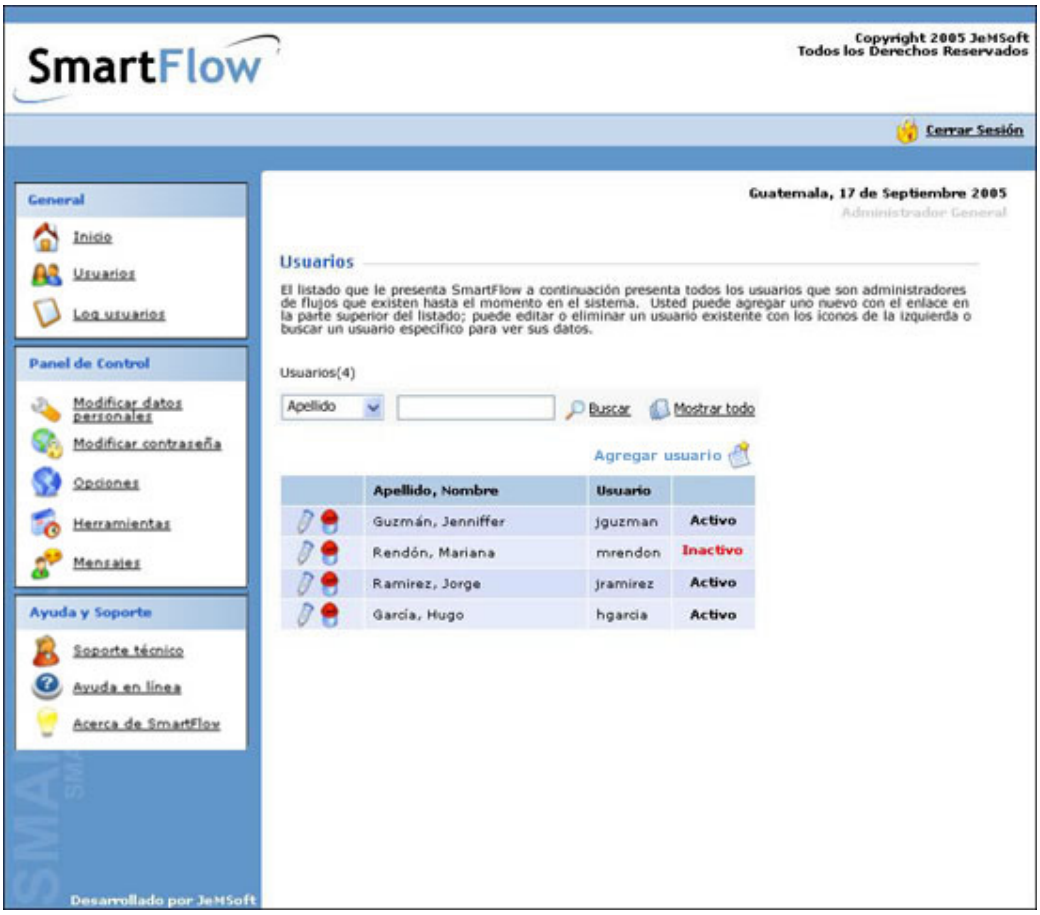
1)Mostrar listado de usuarios de flujos. Esta interfaz muestra un listado de todos los usuarios que son Administradores de flujos dentro del sistema.

Cada uno puede ser editado o eliminado, utilizando los iconos mostrados en la parte izquierda de cada usuario (Estos iconos se muestran en todas las siguientes

interfaces que presenten listados). La opción de editar un usuario se hace en una interfaz diferente a la que se muestra en la ilustración 68 (Ver ilustración 70). Ahora bien, al seleccionar el botón de Eliminar, se muestra un mensaje de verificación al usuario donde debe indicarse si está seguro de eliminar el registro seleccionado. Además, en esta pantalla se permite la búsqueda de un usuario específico y la ordenación de todos ya sea por apellido o por usuario.

En caso que el número de usuarios exceda diez, se mostrarán en la parte inferior las páginas donde se presentan los usuarios restantes de diez en diez. Esto aplica para todas las interfaces siguientes que presenten listados.

Ilustración 68. Interfaz que muestra listado de Administradores de flujos.

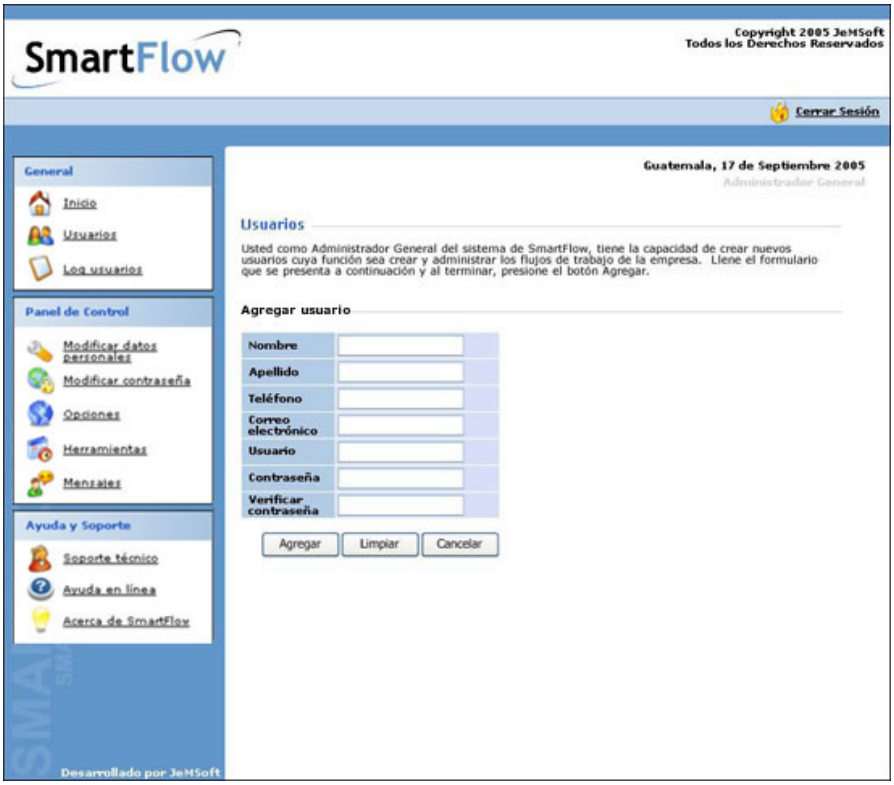


Para cada usuario que se muestra en la Ilustración 68, se muestra provee una columna adicional que permite cambiar el estado de un usuario. Si el usuario se encuentra actualmente activo dentro del sistema, esto se reflejará mostrando la

palabra activo en color negro; ahora bien, en el caso que un usuario esté inactivo, se mostrará la palabra inactivo en color rojo. El Administrador general del sistema puede establecer el estado de un usuario como activo o inactivo al simplemente seleccionar dicha columna en la fila del usuario a modificar y el color de la palabra cambiará automáticamente según el estado actual.

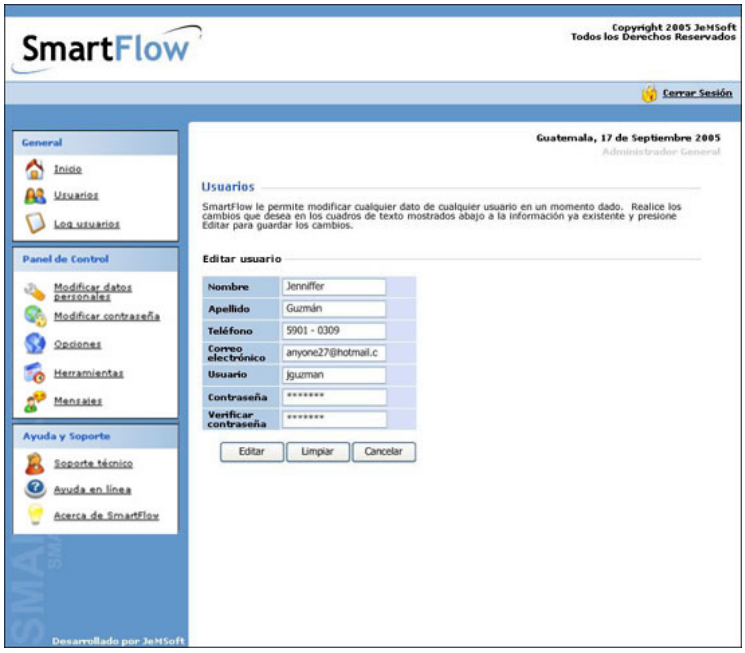
2) Agregar usuario de flujos. Esta opción puede accederse en la parte superior del listado de usuarios (Ver ilustración 68, agregar usuario). Para crear un nuevo usuario Administrador de flujos dentro del sistema, el Administrador general debe ingresar todos los campos que se muestran en la siguiente ilustración que corresponden a los datos del nuevo usuario.

Ilustración 69. Interfaz para agregar un nuevo administrador de flujos.



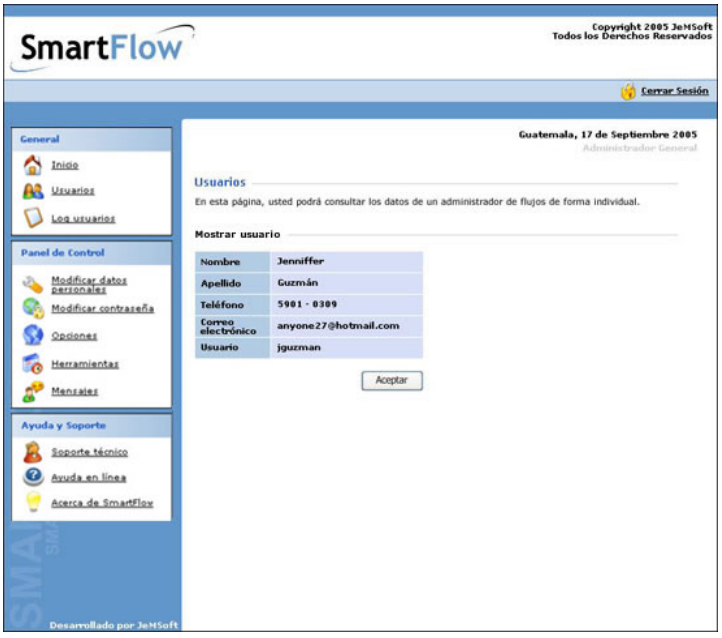
3) *Editar usuario de flujos.* Para editar la información de un Administrador de flujos, el Administrador general debe seleccionar el icono del lápiz que se encuentra en la fila del usuario a editar (Ver ilustración 68). Al hacer esto, se muestra la siguiente interfaz donde se muestra la información ya almacenada para ese usuario dentro del sistema y puede ser modificada.

Ilustración 70. Interfaz para editar un administrador de flujos existente.



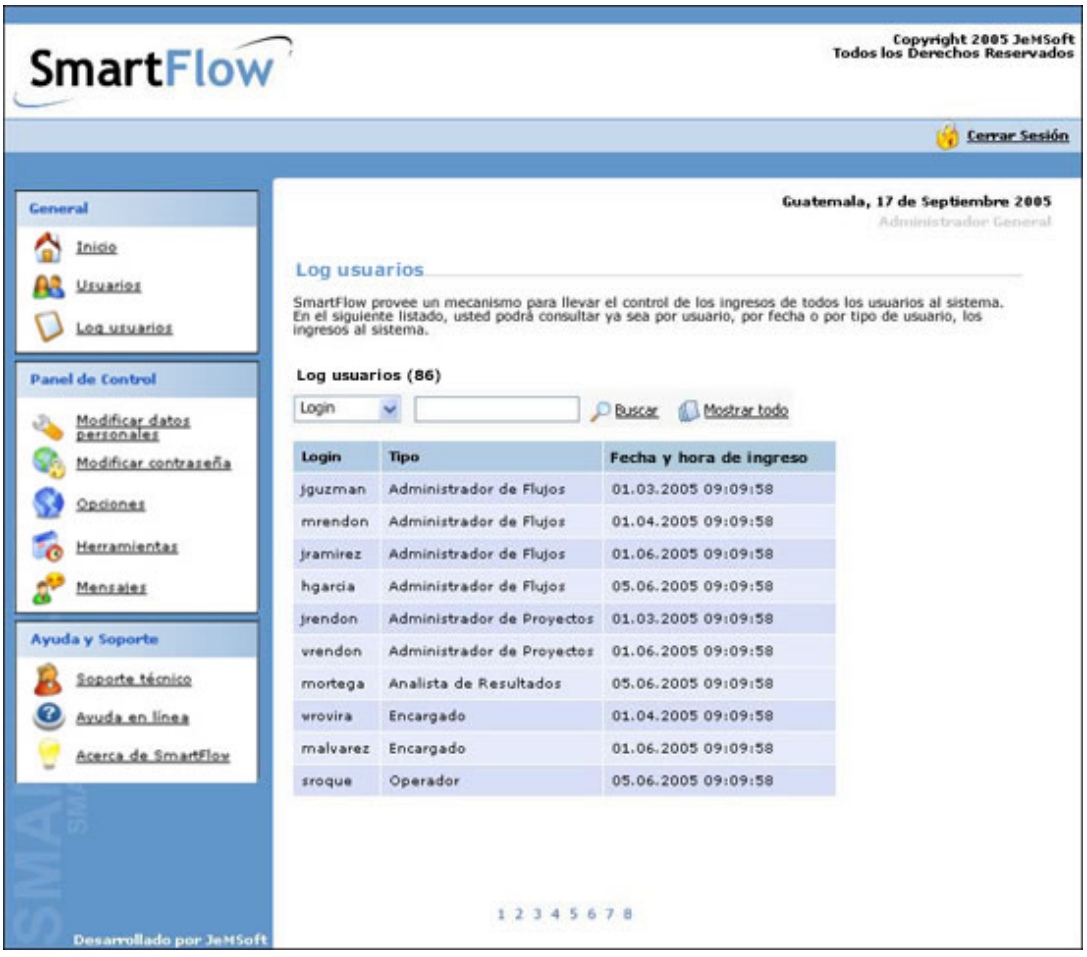
4) Mostrar usuario de flujos. Para consultar los datos de un usuario Administrador de flujos, el Administrador general debe seleccionar la fila del usuario que desee mostrar (Ver ilustración 68). Al hacer esto, se muestra la siguiente pantalla.

Ilustración 71. Interfaz para mostrar información de Administrador de flujos.



b. Log usuarios. El Administrador general puede consultar todos los ingresos de todos los usuarios al sistema a través de esta interfaz. Para cada usuario se muestra su tipo y la fecha y hora del ingreso. Además, a manera de control se presenta en la parte superior un conteo de todos los usuarios mostrados en ese momento. Adicionalmente, se proveen las opciones de ordenar el listado por: usuario, tipo y fecha de ingreso; así como la opción de buscar algún ingreso específico utilizando para esto los mismos criterios. Esta bitácora de usuarios elimina automáticamente los ingresos que se hayan realizado al sistema seis meses antes a la fecha actual. Es decir, en el listado se muestran únicamente los registros de hace seis meses. El resto de registros se elimina de la base de datos y es almacenado en un archivo. Esta interfaz puede accederse a través de la opción Log usuarios en el menú General del primer cuadro a la izquierda.

Ilustración 72. Interfaz que muestra bitácora de ingresos de usuarios.

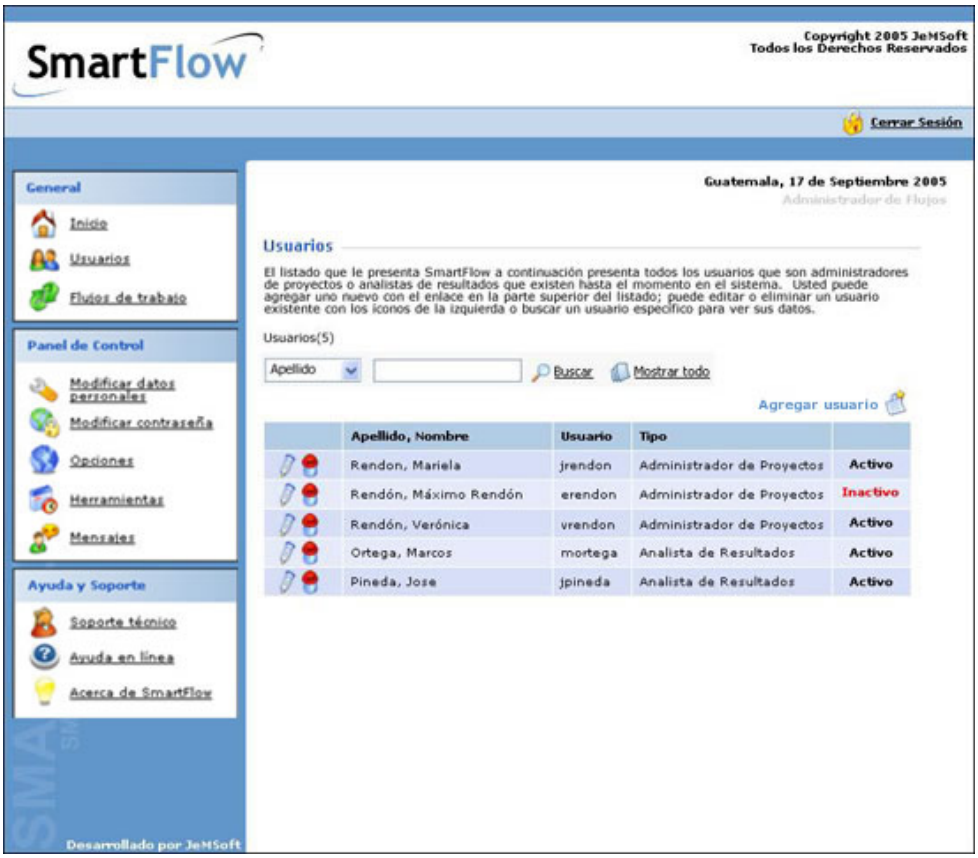


3. Administrador de flujos. Las siguientes interfaces únicamente pueden ser accedidas por los usuarios que sean Administradores de flujos dentro del sistema.

a. Usuarios de proyectos. La interfaz para la administración de usuarios de proyectos puede accederse a través de la opción usuarios en el menú general del primer cuadro a la izquierda. Los usuarios de proyectos son: Administrador de proyectos o Analista de resultados.

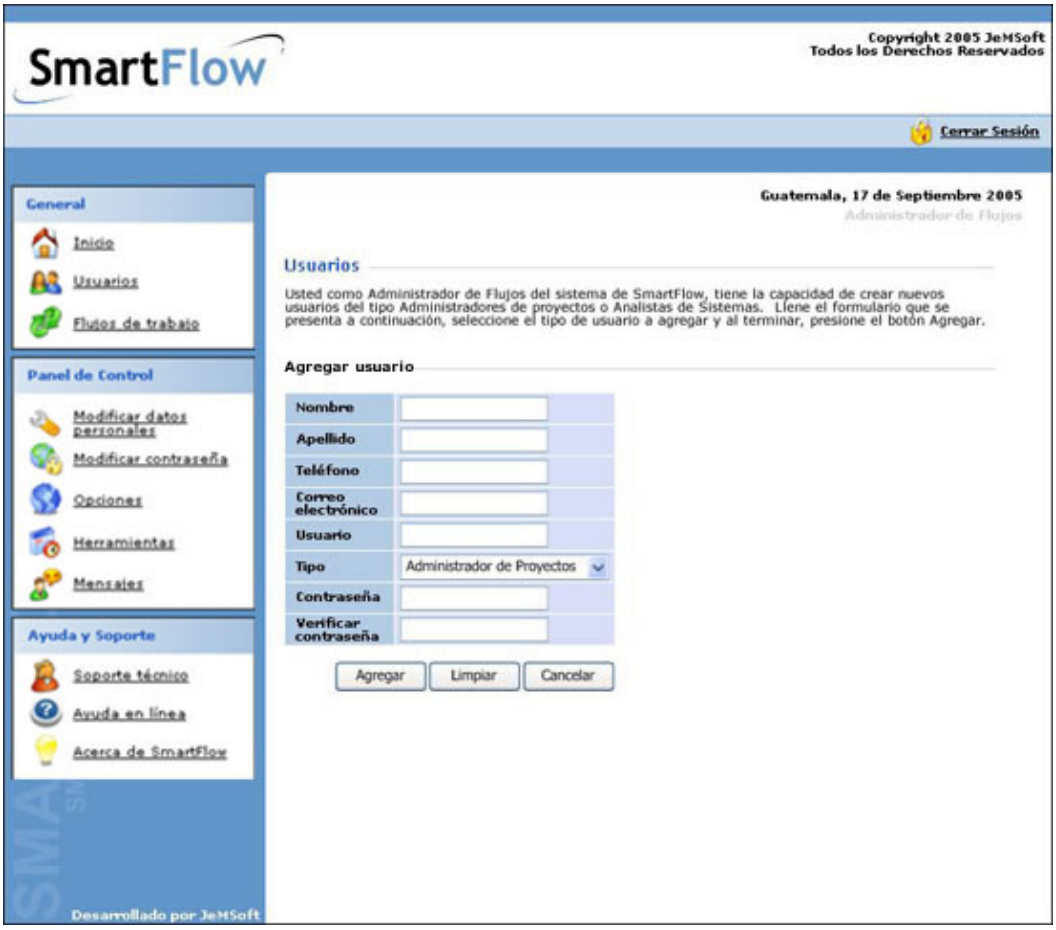
1) Mostrar listado de usuarios de proyectos. Esta pantalla muestra un listado de todos los usuarios de proyectos. Cada uno puede ser editado o eliminado. Además, se permite la búsqueda de un usuario específico y la ordenación de todos ya sea por apellido, usuario o tipo.

Ilustración 73. Interfaz que muestra un listado de todos los usuarios de proyectos existentes.



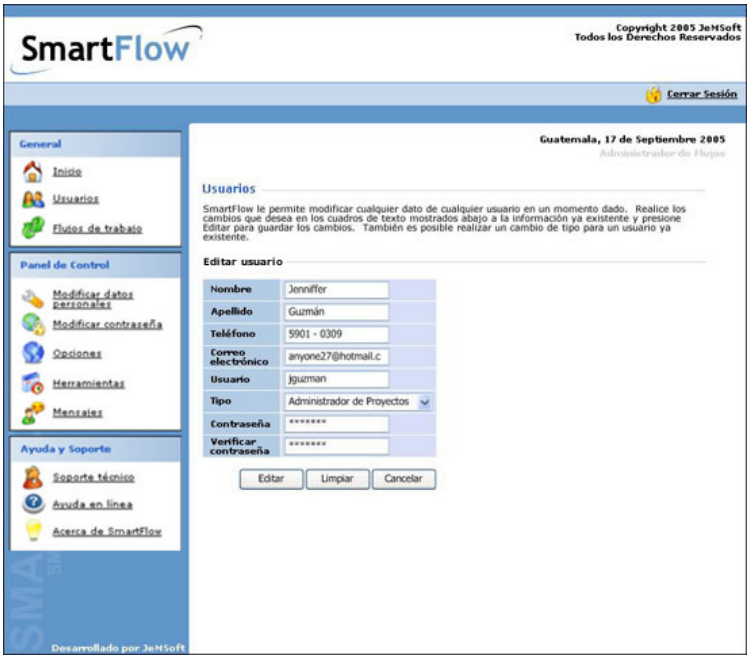
2) *Agregar usuario de proyectos.* Esta opción puede accederse en la parte superior del listado de usuarios (Ver ilustración 73). Para crear un nuevo usuario de proyectos dentro del sistema, el Administrador de flujos debe ingresar todos los campos que se muestran en la siguiente ilustración para el nuevo usuario y seleccionar su tipo (Administrador de proyectos o Analista de resultados) del listado proporcionado.

Ilustración 74. Interfaz para agregar un nuevo usuario de proyectos.



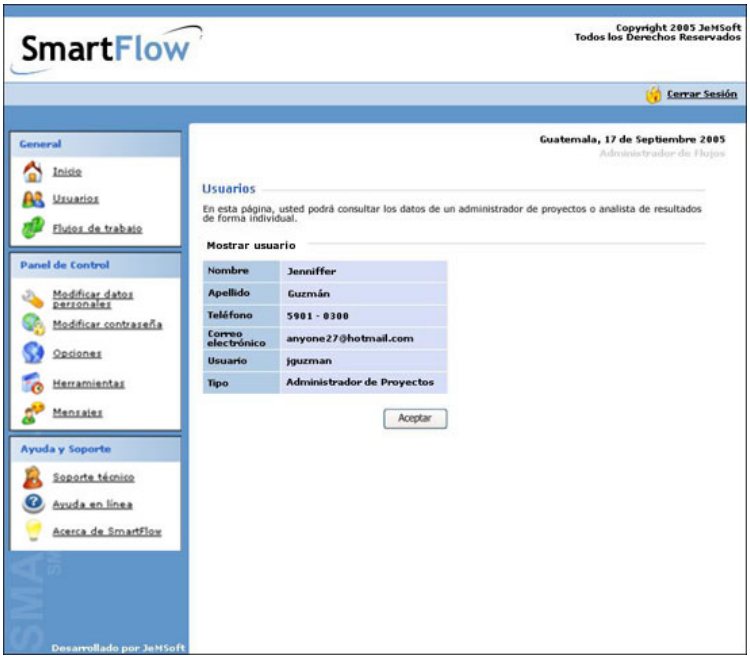
3) *Editar Usuario de Proyectos*. Para editar la información de un usuario de proyectos, el Administrador de flujos debe seleccionar el icono del lápiz que se encuentra en la fila del usuario a editar (Ver ilustración 73). Al hacer esto, se muestra la siguiente pantalla donde se muestra la información ya almacenada para ese usuario dentro del sistema y puede ser modificada. El tipo de un usuario de proyectos también puede ser editado.

Ilustración 75. Interfaz para editar datos de un usuario de proyectos.



4) Mostrar usuario de proyectos. Para consultar los datos de un usuario de proyectos, el Administrador de flujos debe seleccionar la fila del usuario que desee mostrar (Ver ilustración 73). Al hacer esto, se muestra la siguiente pantalla.

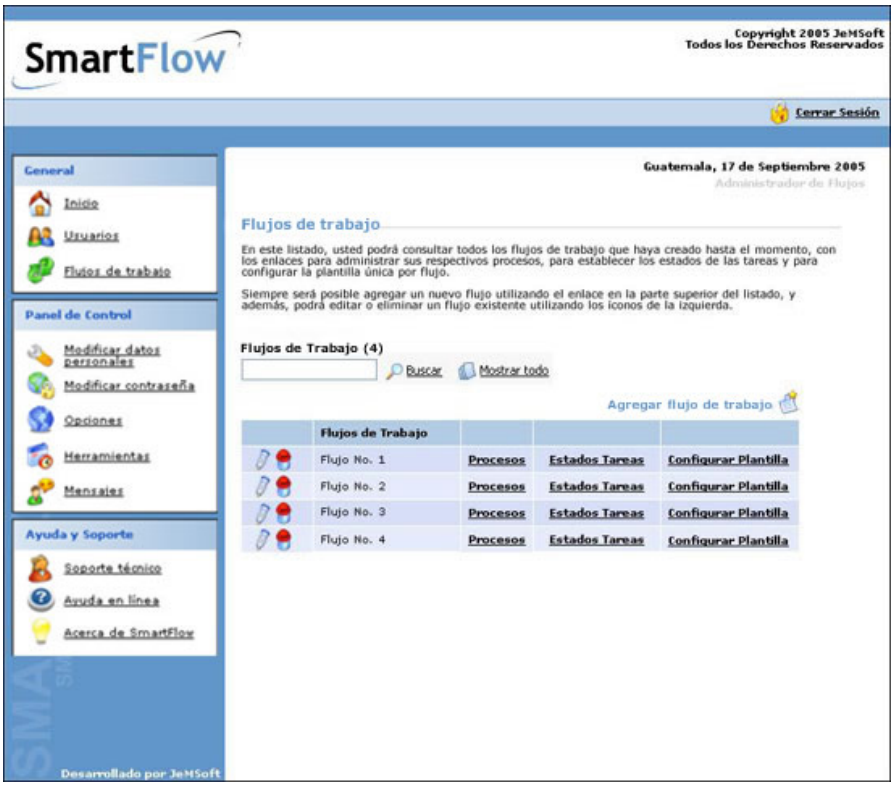
Ilustración 76. Interfaz para mostrar los datos de un usuario de proyectos.



b. Flujos de trabajo/procesos/tareas/estados de tareas. Esta pantalla podrá ser accedida a través de la opción Flujos de trabajo en el menú general en el primer cuadro de la izquierda.

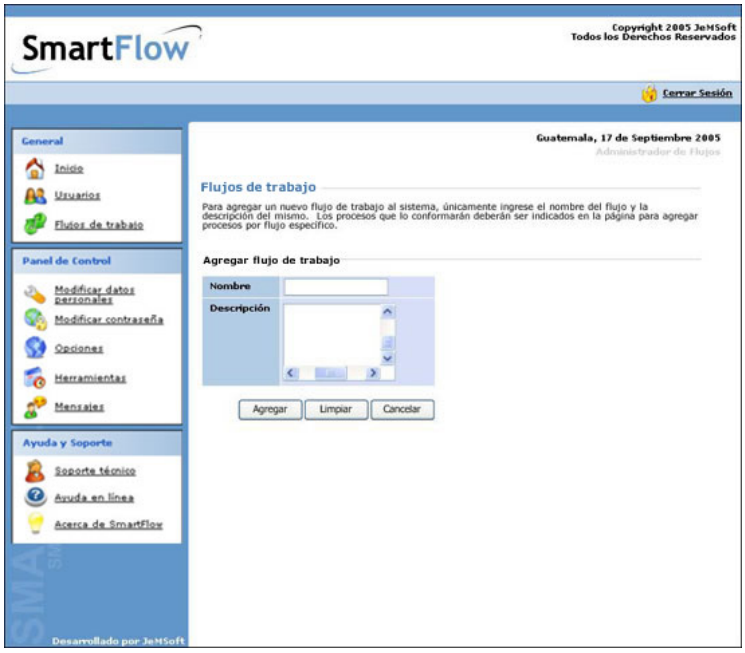
1. *Mostrar listado de flujos.* La siguiente ilustración muestra la pantalla donde un Administrador de flujos puede consultar todos los flujos de trabajo que él haya creado. Por cada flujo se presentan las opciones para editarlo o eliminarlo, la administración de los procesos dentro dicho flujo, la administración de los estados de las tareas y la configuración de la plantilla para los proyectos que utilicen dicho flujo. Este listado puede ser ordenado por el nombre del flujo y puede buscarse por el mismo criterio.

Ilustración 77. Interfaz que muestra listado de todos los flujos de trabajo.



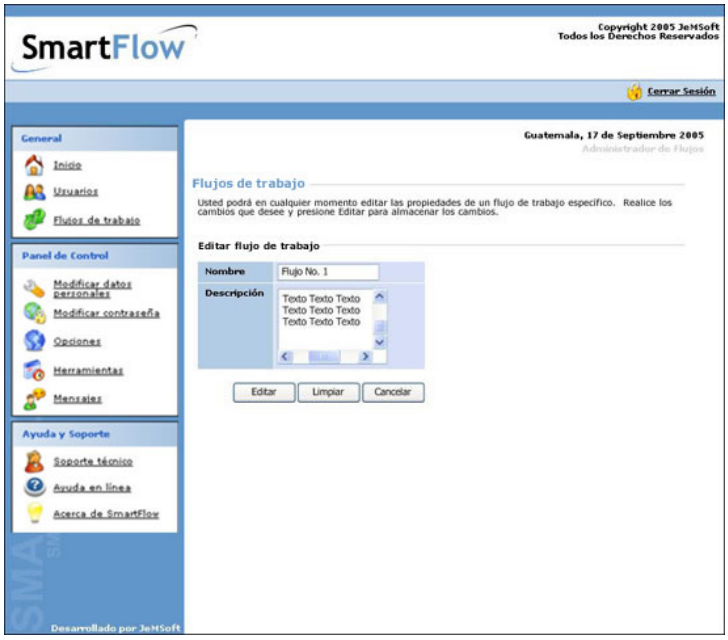
2. *Agregar flujo*. Esta opción puede accederse en la parte superior del listado de flujos (Ver ilustración 77, agregar flujo de trabajo). Para crear un flujo de trabajo dentro del sistema, el Administrador de flujos debe ingresar el nombre y la descripción del mismo para ser almacenados.

Ilustración 78. Interfaz para agregar nuevo flujo de trabajo.



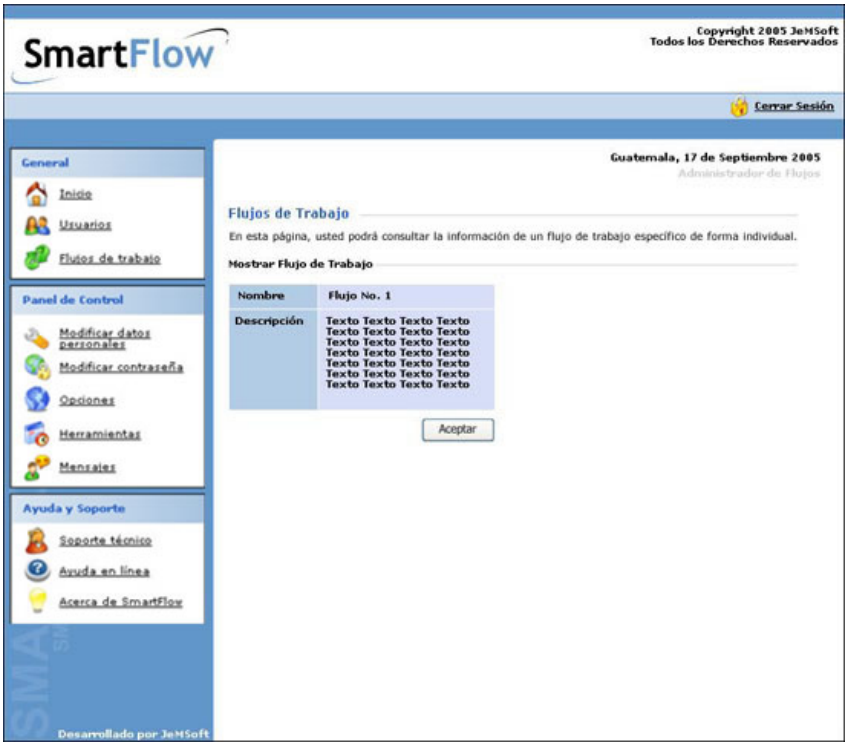
3. Editar flujo. Para editar la información de flujo de trabajo dentro del sistema, el Administrador de flujos debe seleccionar el icono del lápiz que se encuentra en la fila del flujo a editar (Ver ilustración 77). Al hacer esto, se muestra la siguiente pantalla donde se muestra la información ya almacenada para ese flujo dentro del sistema y puede ser modificada.

Ilustración 79. Interfaz para editar los datos de un flujo de trabajo.



4. *Mostrar flujo*. Para consultar los datos de flujo de trabajo específico, el Administrador de flujos debe seleccionar la fila del flujo que desee examinar (Ver ilustración 77). Al hacer esto, se muestra la siguiente pantalla.

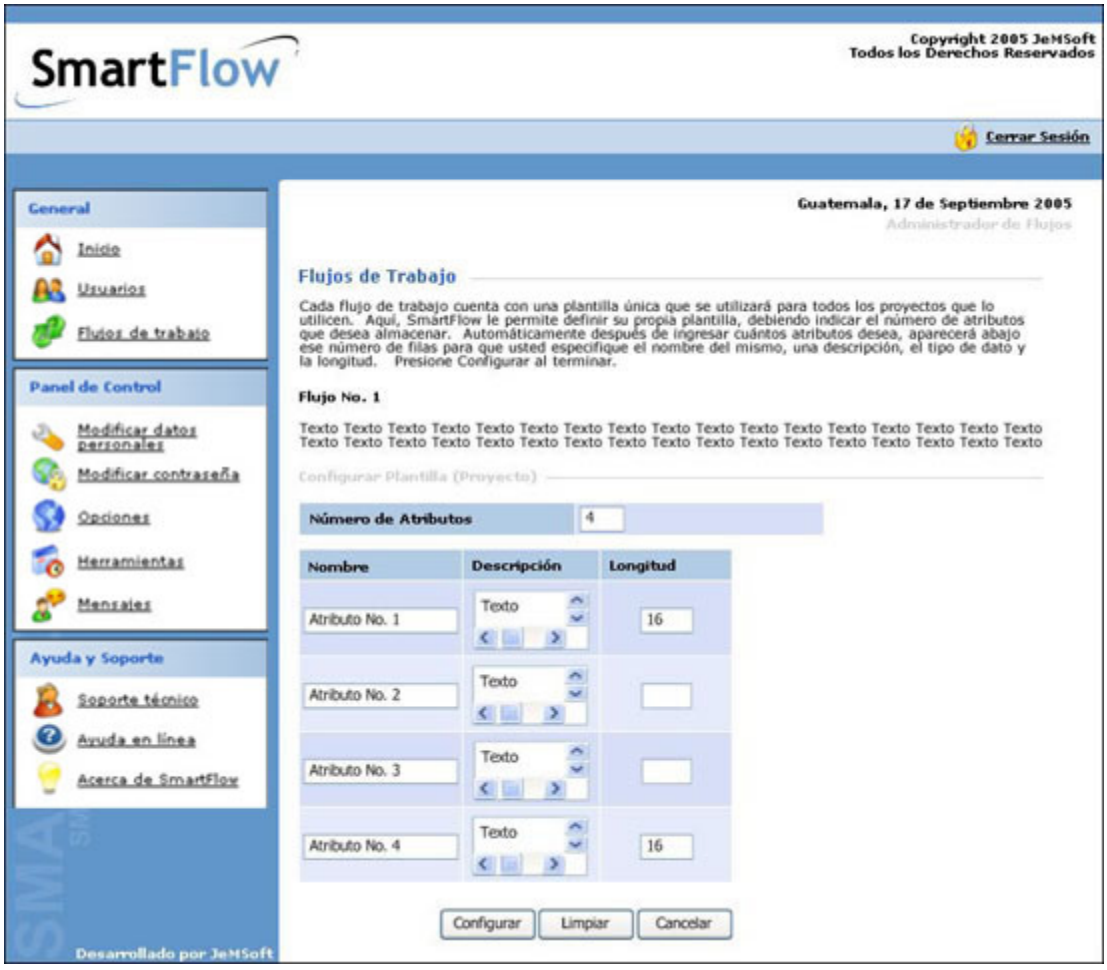
Ilustración 80. Interfaz para mostrar un flujo de trabajo.



5. *Configurar plantilla de proyectos*. Anteriormente se definió en que consiste la plantilla de proyectos dentro de un flujo (Ver Clase: Plantilla del análisis). La siguiente interfaz permite la creación y configuración de dicha plantilla de la siguiente forma: primero debe indicarse el número de atributos a almacenar para la plantilla y automáticamente después se mostrará una serie de cuadros de texto donde el Administrador de flujos debe especificar cada atributo, una descripción, el tipo y la longitud del mismo.

En todo momento se muestra en la parte superior la descripción almacenada en el sistema para el flujo con el que se está trabajando.

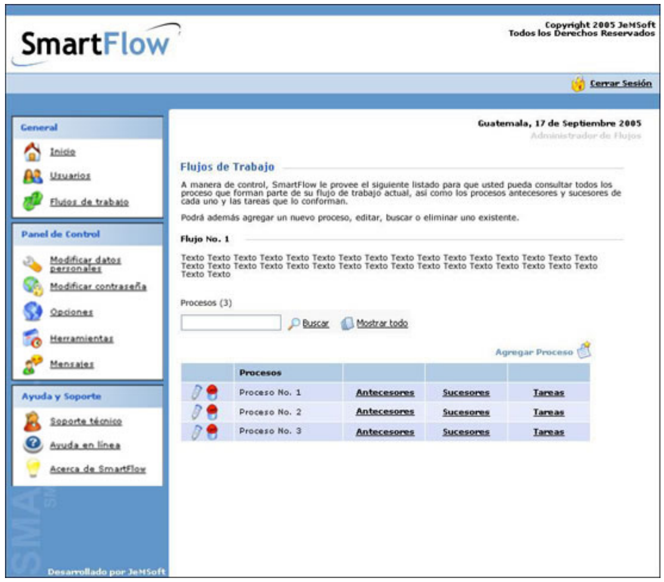
Ilustración 81. Interfaz que permite la configuración de la plantilla de proyectos.



6. *Mostrar listado de procesos.* Al presionar el vínculo procesos en el listado de flujos de la ilustración 77, se muestra la siguiente pantalla, donde se muestran los procesos que pertenecen a un flujo específico. Por cada proceso se presentan las opciones de editar o eliminar, así como la administración de sus procesos antecesores y sucesores, y las tareas que forman parte de él.

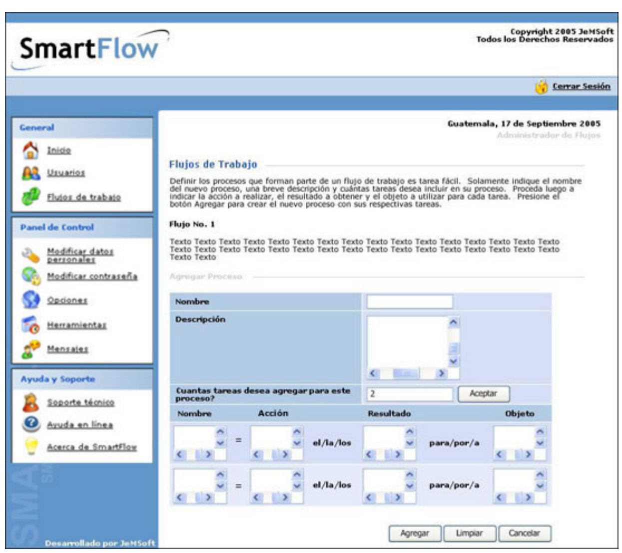
Este listado de procesos puede ser ordenado conforme el nombre de los procesos y es posible buscar un proceso específico utilizando el mismo criterio.

Ilustración 82. Interfaz que muestra listados de procesos dentro de un flujo.



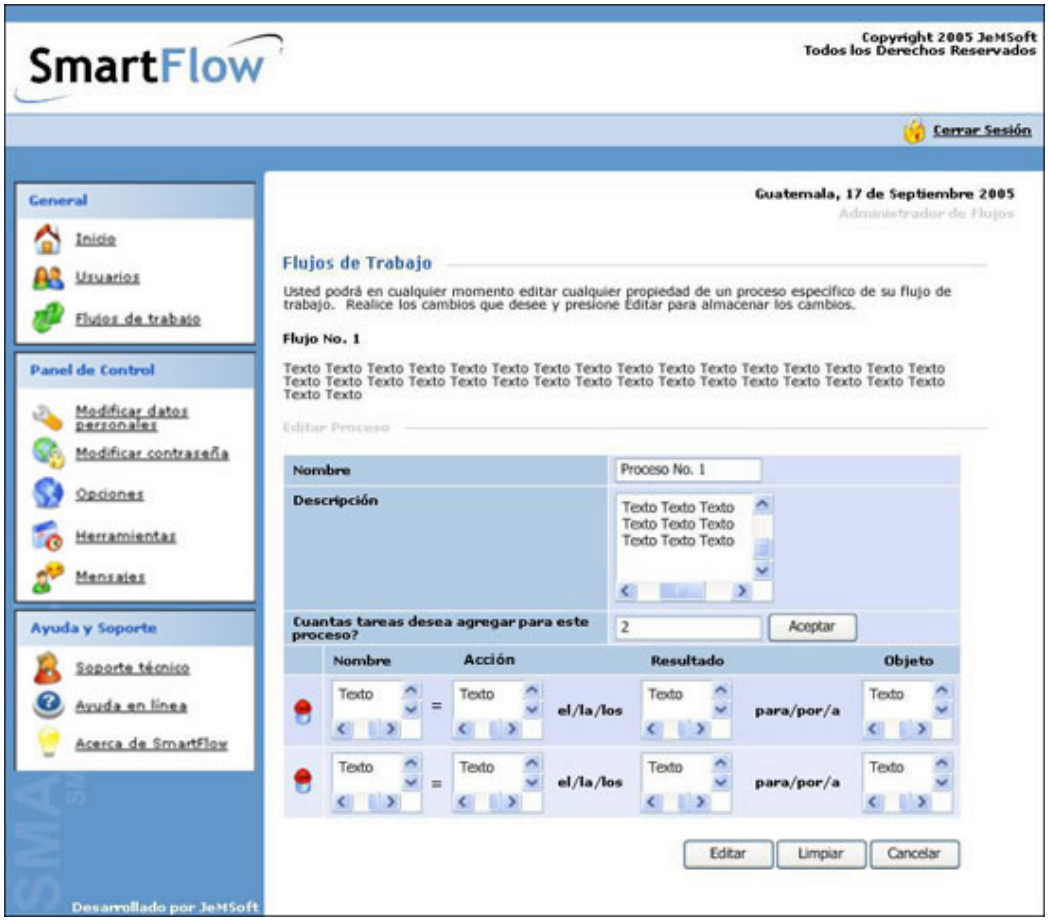
7. *Agregar proceso.* Esta opción puede accederse en la parte superior del listado de procesos (Ver ilustración 82, agregar proceso). Para crear un nuevo proceso dentro de un flujo de trabajo, el Administrador de flujos debe ingresar el nombre y la descripción del mismo, así como indicar el número de tareas para él. Automáticamente al presionar el botón aceptar se muestra ese número de campos para la definición de sus respectivas tareas (Ver definición de la clase tareas, en el análisis).

Ilustración 83. Interfaz que permite crear nuevo proceso junto con sus tareas relacionadas.



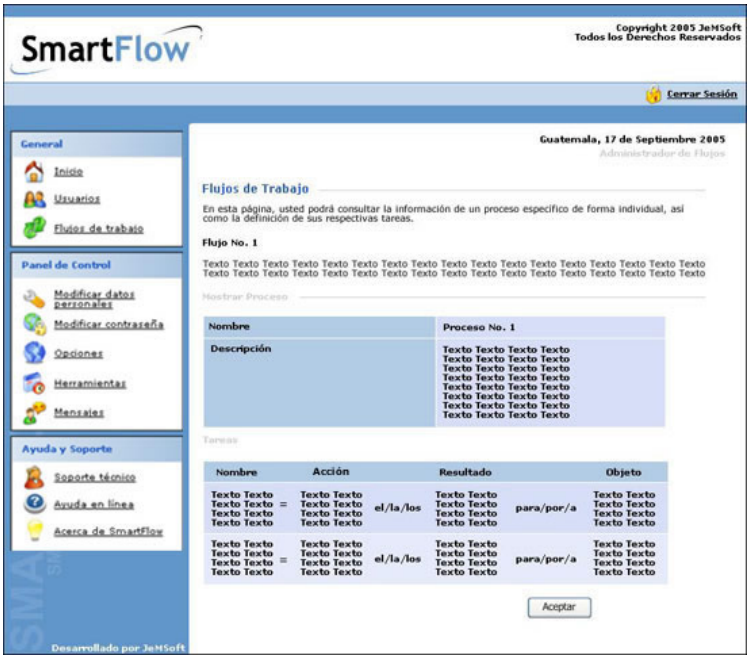
8. Editar proceso. Para editar la información de un proceso, el Administrador de flujos debe seleccionar el icono del lápiz en la fila del proceso a editar (Ver ilustración 82). Al hacer esto, se muestra la siguiente pantalla donde aparece la información ya almacenada para ese proceso, y la de sus tareas relacionadas, para ser modificada.

Ilustración 84. Interfaz para editar un proceso junto con sus tareas.



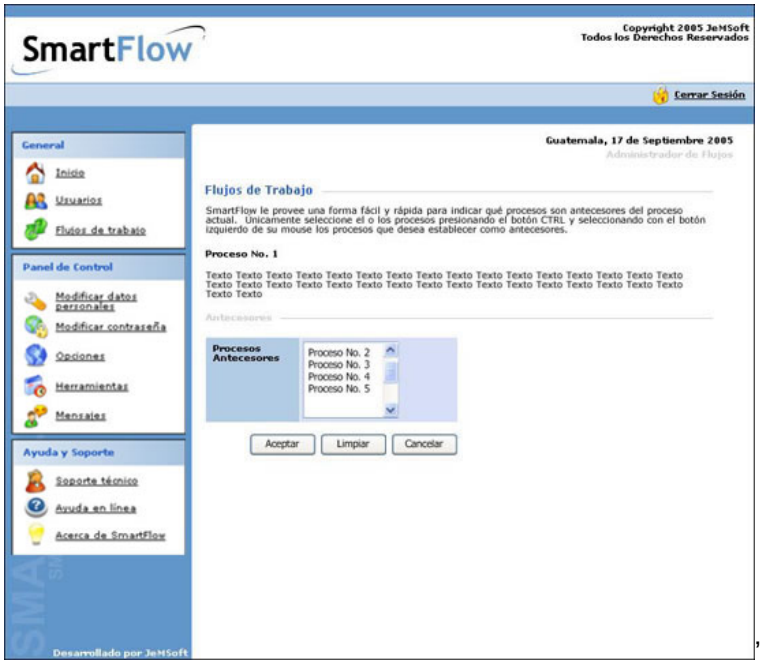
9. Mostrar proceso. La siguiente ilustración muestra la interfaz para que un Administrador de flujos consulte los datos de un proceso específico dentro de un flujo de trabajo. Para esto, se debe seleccionar la fila del proceso a examinar (Ver ilustración 82).

Ilustración 85. Interfaz que muestra proceso específico junto con sus tareas.



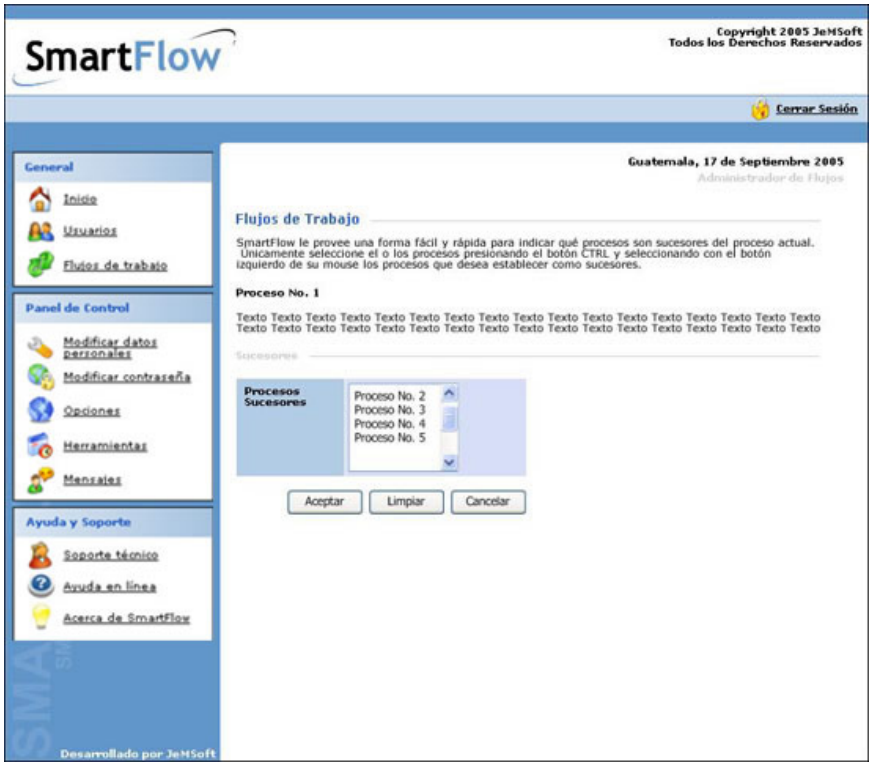
10. Establecer procesos antecesores. La siguiente interfaz le permite a un Administrador de flujos establecer los procesos antecesores del proceso seleccionado anteriormente en el listado de la ilustración 82. Si el proceso es el primero no tendrá antecesores, por lo que el listado de procesos debe permanecer vacío.

Ilustración 86. Interfaz para seleccionar procesos antecesores



11. *Establecer procesos sucesores.* La siguiente interfaz le permite a un Administrador de flujos establecer los procesos sucesores del proceso seleccionado en el listado de la ilustración 82. Si el proceso es el último en el flujo éste no tendrá sucesores, por lo que el listado de procesos debe permanecer vacío.

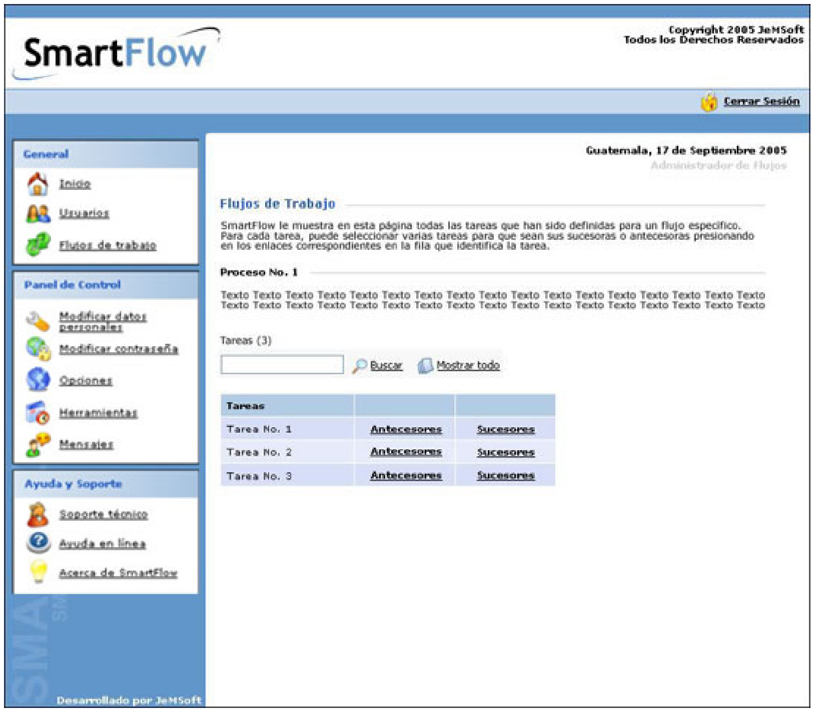
Ilustración 87. Interfaz para seleccionar procesos sucesores



12. *Mostrar listado de tareas*. La siguiente ilustración muestra la pantalla donde un Administrador de flujos puede consultar el listado de tareas en un proceso específico. En la parte superior del listado se muestra una descripción del proceso en el cual se está trabajando. Además, se provee la opción de ordenar el listado por el nombre de las tareas o buscar una tarea específica, siempre utilizando para esto el nombre de la misma.

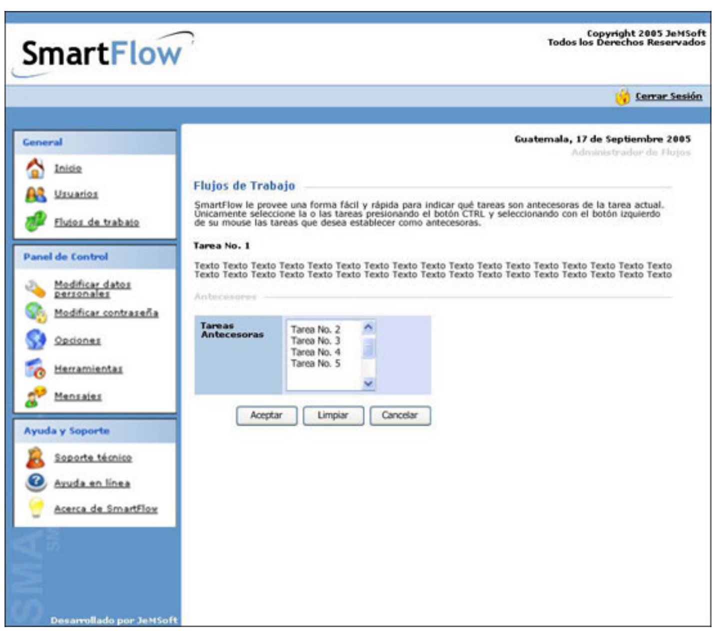
Para cada tarea se provee un enlace hacia la administración de sus tareas antecesoras y sucesores de forma similar a la de los procesos.

Ilustración 88. Interfaz que muestra todas las tareas dentro de un proceso.



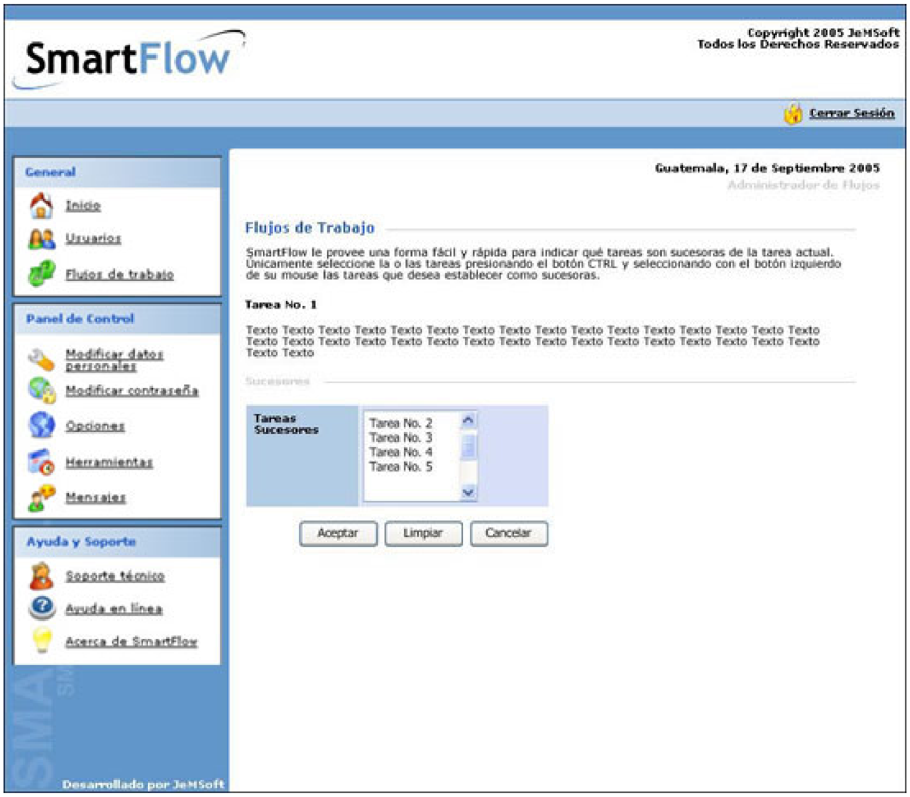
13. Establecer tareas antecesoras. La siguiente interfaz le permite a un Administrador de flujos establecer las tareas antecesoras de la tarea seleccionada en el listado de la Ilustración 88. Si la tarea es la primera dentro del proceso, ésta no tendrá antecesoras, por lo que el listado de tareas debe permanecer vacío.

Ilustración 89. Interfaz para seleccionar tareas antecesoras



14. Establecer tareas sucesoras. La siguiente interfaz le permite a un Administrador de flujos establecer las tareas sucesoras de una tarea. Si la tarea es la última en el proceso no tendrá sucesoras y el listado de tareas debe permanecer vacío.

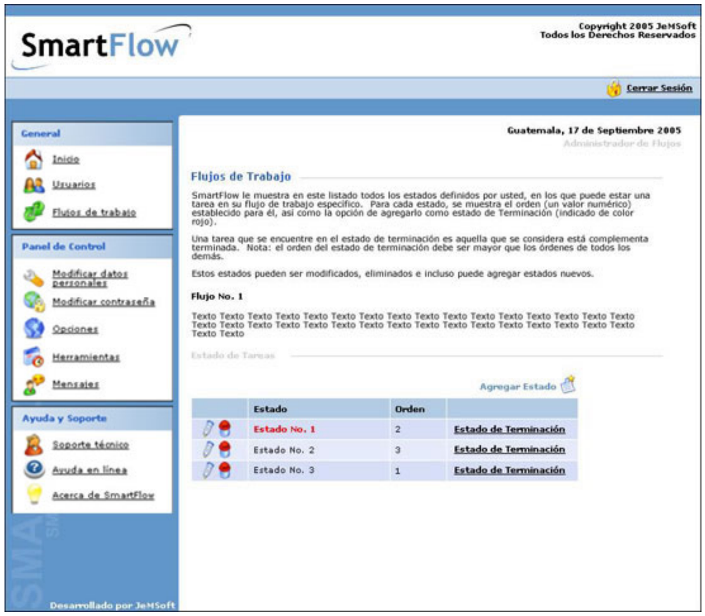
Ilustración 90. Interfaz para seleccionar tareas sucesoras



15. Mostrar listado de estados de tareas. En el análisis se describió la capacidad de SmartFlow para personalizar el estado de las tareas (Ver clase: Estados_Tareas). La siguiente ilustración muestra la interfaz que permite consultar todos los estados de tareas definidos en un flujo, donde cada estado puede ser modificado o eliminado.

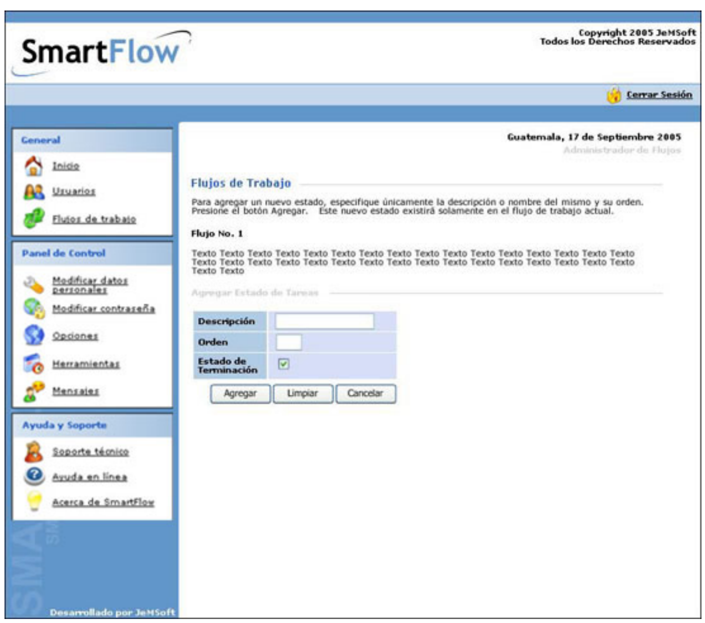
Además, cada estado tiene un orden específico y puede ser designado como un estado de terminación en el enlace que se muestra en la última columna (estado de terminación). Al hacer esto, automáticamente el nombre del estado cambia a color rojo, indicando que se trata del estado de terminación para las tareas dentro del flujo.

Ilustración 91. Interfaz que muestra listado de los estados de tareas



16. Agregar estado de tarea. Para agregar un nuevo estado de tareas al flujo, debe seleccionarse el enlace agregar estado en la parte superior del listado de estados mostrado en la ilustración 91. Esa acción mostrará al Administrador de tareas la siguiente interfaz donde debe proporcionar una descripción o nombre del estado, el orden dentro del flujo e indicar si se trata de un estado de terminación.

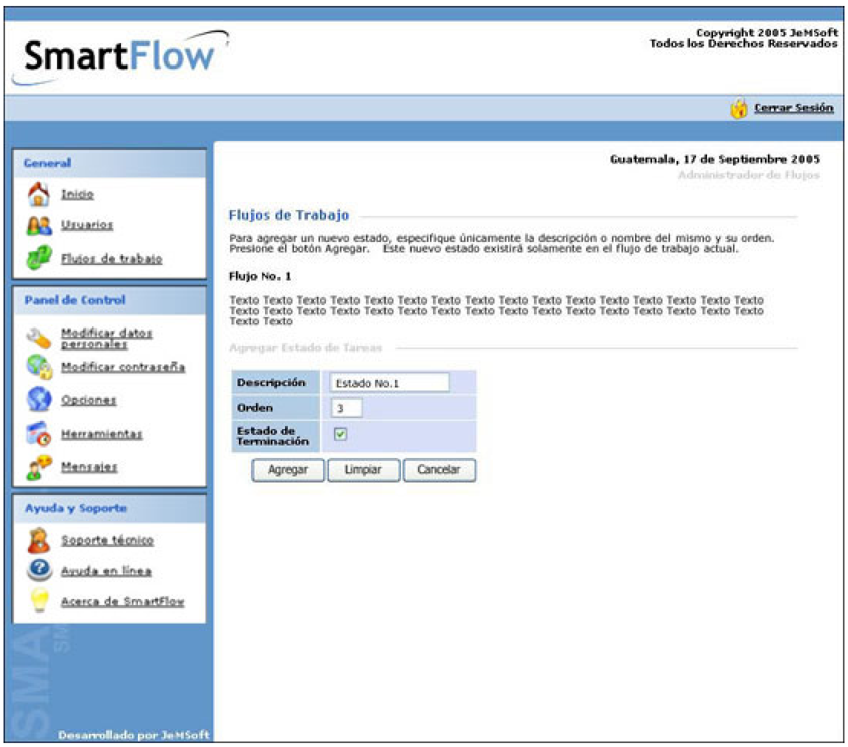
Ilustración 92. Interfaz para agregar un nuevo estado de tarea a un flujo.



17. Editar estado de tarea. Para editar un estado de tarea existente dentro de un flujo de trabajo, el Administrador de flujos debe seleccionar el icono del lápiz que se encuentra en la fila del estado a editar (Ver ilustración 91).

Al hacer esto, se muestra la siguiente pantalla donde se muestra la información ya almacenada para ese estado para ser modificada.

Ilustración 93. Interfaz para editar un estado de tarea existente para un flujo.



C. ESPECIFICACIONES TÉCNICAS (HERRAMIENTAS Y PLATAFORMAS)

Este trabajo se diseñó para ser desarrollado utilizando las siguientes herramientas: PHP, MySQL y Apache (Ver Anexo C para más detalles de cada una).

PHP es un lenguaje de programación interpretado de scripts, que se utiliza en el desarrollo de páginas web. PHP se destaca por su capacidad de incluirse en código HTML y de ser ejecutado en un servidor. Al ejecutar el código del lado del servidor, se garantiza que el cliente únicamente reciba el resultado solicitado, siendo el código totalmente transparente para él. Una de las ventajas de PHP es su gran versatilidad pues permite realizar funciones complejas. Además, PHP es una herramienta open-source, por lo que está disponible para el uso gratuito de cualquier persona. Adicionalmente, PHP puede utilizarse en cualquiera de los principales sistemas operativos más conocidos, soporta la mayoría de los servidores web, así como interfaces hacia una gran variedad de bases de datos.

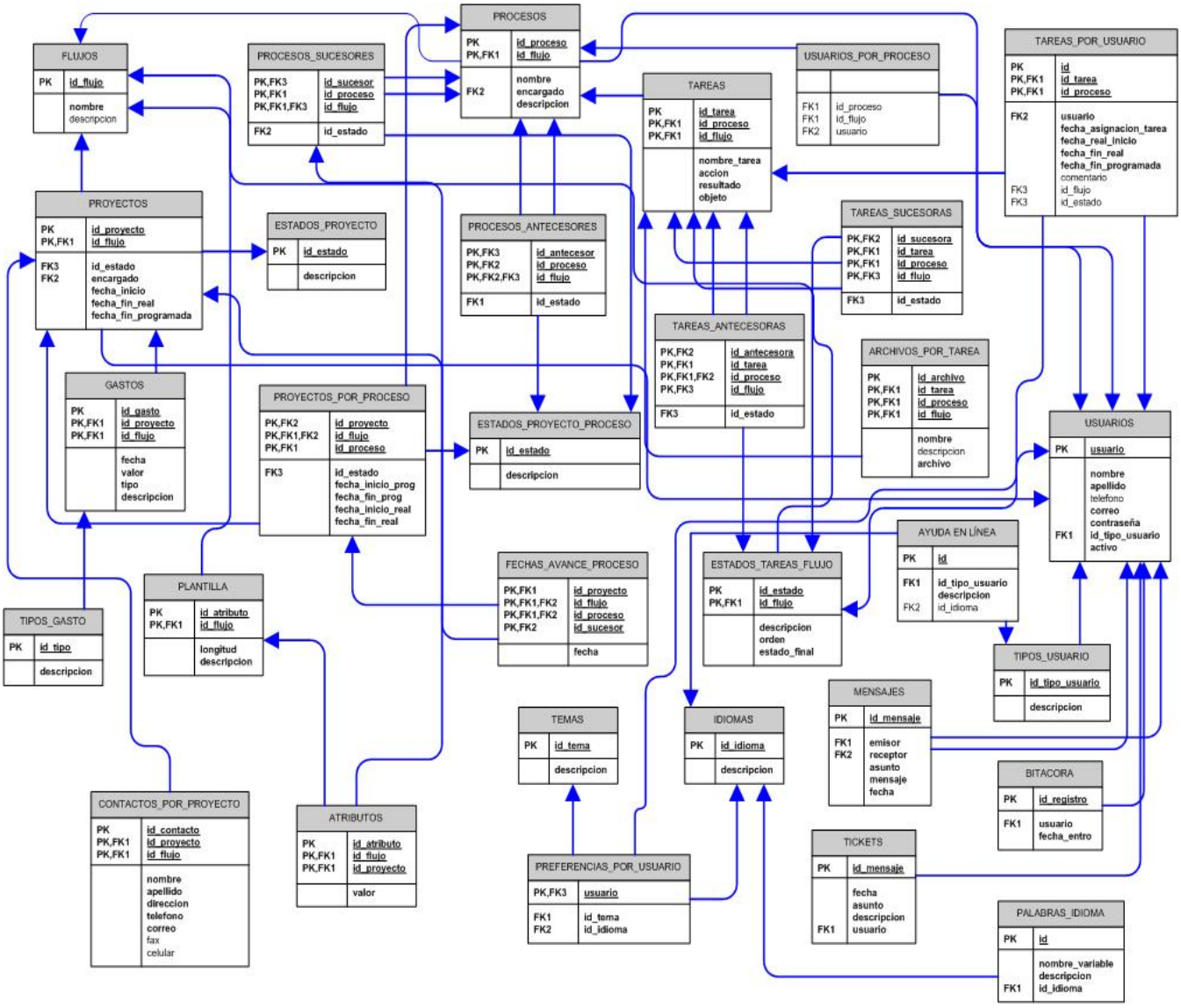
MySQL es un popular manejador de bases de datos relacionales también open-source y se caracteriza por su alto rendimiento, alta confiabilidad y facilidad de uso. MySQL también ofrece la ventaja de poder ser utilizado en diferentes sistemas operativos y provee interfaces para varios lenguajes de programación entre los que se incluye PHP.

Por último, Apache es un servidor web open-source, poderoso debido a su flexibilidad y extensibilidad; además permite modificar el código fuente completo sin ninguna restricción. Al igual que PHP y MySQL, Apache es multiplataforma y se puede ejecutar en varios sistemas operativos, con la ventaja adicional de que siempre están desarrollando nuevas librerías que se le pueden agregar.

D. DISEÑO DEL MODELO DE LA BASE DE DATOS

La ilustración que se muestra a continuación refleja el modelo diseñado para la base de datos del sistema SmartFlow. En este diagrama se incluyen las tablas relacionadas tanto al módulo administrativo como al módulo de usuario.

Ilustración 94. Diagrama de la base de datos.



El diagrama anterior muestra 31 tablas con sus respectivas relaciones. A continuación se detallan los campos de cada tabla junto con sus tipos (Ver Apéndice C: MySQL, para consultar los tipos de datos que soporta).

1. FLUJOS. Esta tabla almacena todos los flujos dentro del sistema. Los campos a almacenar de cada flujo son:

Cuadro 1. Campos de la tabla FLUJOS.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_flujo	Identificador único y correlativo para cada flujo.	INT	☑	☐
Nombre	Nombre propio que identifica al flujo.	VARCHAR(30)	☐	☐
Descripción	Breve descripción del flujo de trabajo, función, objetivos, etc.	VARCHAR(70)	☐	☐

2. PROYECTOS. Esta tabla almacena todos los proyectos dentro del sistema. Cada proyecto está relacionado a un flujo y está a cargo de un Administrador de proyectos. Esta tabla está relacionada con las tablas FLUJOS, ESTADOS_PROYECTO y USUARIOS.

Cuadro 2. Campos de la tabla PROYECTOS.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_proyecto	Identificador único y correlativo para cada proyecto.	INT	☑	☐
Id_flujo	Indica el flujo al que pertenece el proyecto.	INT	☑	☐
Id_estado	Especifica el estado del proyecto.	SMALLINT	☐	☑
Encargado	Se refiere al nombre de usuario del Administrador de proyectos que lo creó.	VARCHAR (20)	☐	☑
Fecha_inicio	Fecha en la que el proyecto fue creado.	DATETIME	☐	☐
Fecha_fin_real	Fecha en la que realmente el proyecto fue terminado.	DATETIME	☐	☐
Fecha_fin_programada	Fecha estimada de finalización del proyecto.	DATETIME	☐	☐

3. GASTOS. Esta tabla almacena los gastos que puede tener un proyecto. Está relacionada con la tabla PROYECTOS.

Cuadro 3. Campos de la tabla GASTOS.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_gasto	Identificador único y correlativo para cada registro de gasto.	INT	☑	☐
Id_proyecto	Especifica a qué proyecto pertenece un gasto.	INT	☑	☑
Id_flujo	Indica el flujo al que pertenece el proyecto, y por lo tanto el gasto.	INT	☑	☑
Fecha	Fecha en la que el gasto fue programado o realizado.	DATE	☐	☑
Valor	Cantidad monetaria del gasto.	DOUBLE	☐	☐
Tipo	Tipo del gasto, puede ser programado o real.	SMALLINT	☐	☐
Descripción	Descripción del gasto.	VARCHAR (70)	☐	☐

4. TIPOS_GASTO. Dentro del sistema, existen únicamente dos tipos de gastos: gastos planificados y gastos realizados. Esta tabla almacena estos tipos y está relacionada con la tabla GASTOS.

Cuadro 4. Campos de la tabla TIPOS_GASTO.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_tipo	Identificador único y correlativo para el tipo de gasto.	SMALLINT	☑	☐
Descripción	Descripción del gasto: “Gasto planificado” o “Gasto realizado”	VARCHAR(70)	☐	☐

5. CONTACTOS_POR_PROYECTO. Dentro del sistema, se pueden guardar varios contactos relacionados a un proyecto. Esta tabla almacena todos los contactos de todos los proyectos y se relaciona con la tabla PROYECTOS.

Cuadro 5. Campos de la tabla CONTACTOS_POR_PROYECTO

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_contacto	Identificador único y correlativo para cada contacto.	INT	☑	☐
Id_proyecto	Especifica a qué proyecto pertenece un contacto.	INT	☑	☑
Id_flujo	Indica el flujo al que pertenece el proyecto, y por lo tanto el contacto.	INT	☑	☑
Nombre	Nombre propio de la persona que es contacto del proyecto.	VARCHAR (30)	☐	☐
Apellido	Apellido de la persona contacto.	VARCHAR (30)	☐	☐
Dirección	Dirección para localizar al contacto	VARCHAR (50)	☐	☐
Teléfono	Número de teléfono.	VARCHAR (10)	☐	☐
Correo	Dirección de correo electrónico	VARCHAR (30)	☐	☐
Fax	Número de fax.	VARCHAR (10)	☐	☐
Celular	Número de teléfono celular.	VARCHAR (10)	☐	☐

6. PLANTILLA. Esta tabla almacena los atributos definidos para las plantillas para todos los flujos. Se relaciona con la tabla FLUJOS y TIPOS_ATRIBUTO.

Cuadro 6. Campos de la tabla PLANTILLA.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_atributo	Identificador único y correlativo para el atributo.	INT	☑	☐
Id_flujo	Indica el flujo al que pertenece la plantilla.	INT	☑	☑
Descripción	Descripción o nombre del atributo.	VARCHAR(20)	☐	☐
Longitud	Longitud del atributo.	INT	☐	☐

7. ATRIBUTOS. Esta tabla almacena los valores reales para cada atributo dentro de un proyecto. Se relaciona con las tablas PLANTILLA y PROYECTOS.

Cuadro 7. Campos de la tabla ATRIBUTOS.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_atributo	Identificador único y correlativo para el atributo.	INT	☑	☑
Id_flujo	Indica el flujo al que pertenece el atributo.	INT	☑	☑
Id_proyecto	Indica el proyecto al que pertence el atributo.	INT	☑	☑
Valor	Dato o valor real del atributo.	VARCHAR(30)	☐	☐

8. PROYECTOS_POR_PROCESO. Esta tabla almacena datos relacionados para todos los proyectos y los procesos en los que se encuentran. Se relaciona con las tablas PROCESOS, PROYECTOS y ESTADOS_PROYECTO_PROCESO.

Cuadro 8. Campos de la tabla PROYECTOS_POR_PROCESO.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea

Id_proyecto	Identifica a un proyecto dentro del sistema.	INT	☑	☑
Id_flujo	Indica el flujo al que pertenece el proyecto.	INT	☑	☑
Id_proceso	Indica el proceso en el cual se encuentra el proyecto.	INT	☑	☑
Id_estado	Se refiere al estado del proyecto dentro de un proceso.	SMALLINT	☐	☑
Fecha_inicio_prog	Fecha estimada para que el proyecto ingrese a un proceso.	DATE	☐	☐
Fecha_fin_prog	Fecha estimada de finalización de un proceso para un proyecto.	DATE	☐	☐
Fecha_inicio_real	Fecha que realmente el proyecto ingresó a un proceso.	DATE	☐	☐
Fecha_fin_real	Fecha real de finalización del proceso para un proyecto.	DATE	☐	☐

9. ESTADOS_PROYECTO. Esta tabla almacena los estados predefinidos en los que puede estar un proyecto. Estos estados son: iniciado, en proceso y terminado.

Cuadro 9. Campos de la tabla ESTADOS_PROYECTO.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_estado	Identificador único y correlativo para el estado de un proyecto.	SMALLINT	☑	☐
Descripción	Descripción o nombre del	VARCHAR(20)	☐	☐

	estado: iniciado, en proceso y terminado.			

10. PROCESOS. Esta tabla almacena todos los procesos que existen dentro del sistema, por cada flujo. Se relaciona con la tabla FLUJOS Y USUARIOS.

Cuadro 10. Campos de la tabla PROCESOS.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_proceso	Identificador único y correlativo para un proceso.	INT	☑	☐
Id_flujo	Indica el flujo al que pertenece el proceso.	INT	☑	☑
Nombre	Nombre propio del proceso.	VARCHAR(30)	☐	☐
Encargado	Usuario encargado del proceso.	VARCHAR(20)	☐	☑
Descripción	Breve descripción de la función del proceso.	VARCHAR(70)	☐	☐

11. PROCESOS_ANTECESORES. En esta tabla se almacenan todos los procesos antecesores para los procesos que lo requieren. Esto se especifica por flujo de trabajo y la tabla se relaciona con las tablas PROCESOS y ESTADOS_PROCESO_PROYECTO.

Cuadro 11. Campos de la tabla PROCESOS_ANTECESORES.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_antecesor	Identifica un proceso antecesor para un proceso.	INT	☑	☑
Id_proceso	Identifica el proceso para el cual se especifican sus antecesores.	INT	☑	☑
Id_flujo	Indica el flujo al que	INT	☑	☑

	pertenece el proceso.			
Id_estado	Estado del proceso antecesor.	INT	☐	☑

12. PROCESOS SUCESORES. En esta tabla se almacenan todos los procesos sucesores para los procesos que lo requieren. Esto se especifica por flujo de trabajo y la tabla se relaciona con las tablas PROCESOS y ESTADOS_PROCESO_PROYECTO.

Cuadro 12. Campos de la tabla PROCESOS_SUCESORES.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_sucesor	Identifica un proceso sucesor para un proceso.	INT	☑	☑
Id_proceso	Identifica el proceso para el cual se especifican sus sucesores.	INT	☑	☑
Id_flujo	Indica el flujo al que pertenece el proceso.	INT	☑	☑
Id_estado	Estado del proceso sucesor.	INT	☐	☑

13. ESTADOS_PROYECTO_PROCESO. Esta tabla almacena los estados predefinidos en los que puede estar un proceso. Estos estados son: iniciado, en proceso y terminado.

Cuadro 13. Campos de la tabla ESTADOS_PROYECTO_PROCESO.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_estado	Identificador único y correlativo para el estado de un proyecto.	SMALLINT	☑	☐
Descripción	Descripción o nombre del estado: iniciado, en proceso y terminado.	VARCHAR(20)	☐	☐

14. TAREAS. En esta tabla se almacenan todas las tareas por proceso, por lo que se relaciona con la tabla PROCESOS.

Cuadro 14. Campos de la tabla TAREAS.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_tarea	Identificador único y correlativo de una tarea.	INT	☑	☐
Id_proceso	Identifica a qué proceso pertenece la tarea.	INT	☑	☑
Id_flujo	Identifica al flujo al que pertenece el proceso y por lo tanto, la tarea.	INT	☑	☑
Nombre_tarea	Nombre de la tarea	VARCHAR(20)	☐	☐
Acción	Especifica la operación a realizar dentro de la tarea.	VARCHAR(20)	☐	☐
Resultado	Indica el resultado esperado de la acción.	VARCHAR(20)	☐	☐
Objeto	Se refiere al objeto sobre el cual se realizará la acción.	VARCHAR(20)	☐	☐

15. TAREAS ANTECESORAS. En esta tabla se almacenan todas las tareas antecesoras y su estado para las tareas que lo requieren. Esto se especifica por proceso, por lo que se relaciona con las tablas TAREAS y ESTADOS_TAREA_FLUJO.

Cuadro 15. Campos de la tabla TAREAS_ANTECESORAS.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_antecesora	Identifica una tarea antecesora para una tarea.	INT	☑	☑
Id_tarea	Identifica la tarea para la cual se especifican sus antecesoras.	INT	☑	☑
Id_proceso	Identifica el proceso al cual	INT	☑	☑

	pertenecen las tareas.			
Id_flujo	Indica el flujo al que pertenece el proceso y por lo tanto las tareas.	INT	☑	☑
Id_estado	Estado personalizado de la tarea antecesora.	INT	☐	☑

16. TAREAS SUCESORAS. En esta tabla se almacenan todas las tareas sucesoras y su estado para las tareas que lo requieren. Esto se especifica por proceso, por lo que se relaciona con las tablas TAREAS y ESTADOS_TAREA_FLUJO.

Cuadro 16. Campos de la tabla TAREAS_SUCESORAS.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_sucesora	Identifica una tarea sucesora para una tarea.	INT	☑	☑
Id_tarea	Identifica la tarea para la cual se especifican sus sucesoras.	INT	☑	☑
Id_proceso	Identifica el proceso al cual pertenecen las tareas.	INT	☑	☑
Id_flujo	Indica el flujo al que pertenece el proceso y por lo tanto las tareas.	INT	☑	☑
Id_estado	Estado personalizado de la tarea sucesora.	INT	☐	☑

17. USUARIOS_POR_PROCESO. Esta tabla únicamente guarda las referencias de todos los usuarios que pertenecen a algún proceso dentro de un flujo de trabajo. Se relaciona con las tablas USUARIOS y PROCESOS.

Cuadro 17. Campos de la tabla USUARIOS_POR_PROCESO.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea

Id_proceso	Identifica el proceso al cual pertenece un usuario.	INT	☐	☑
Id_flujo	Indica el flujo al que pertenece el proceso y por lo tanto el usuario.	INT	☐	☑
usuario	Usuario que pertenece al proceso.	INT	☐	☑

18. TAREAS_POR_USUARIO. En esta tabla se almacenan las referencias por usuario, de las tareas que él debe realizar. Las tareas se identifican por el proceso al que pertenecen. Esta tabla se relaciona con las tablas USUARIOS y TAREAS.

Cuadro 18. Campos de la tabla TAREAS_POR_USUARIO.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id	Identificador único para los registros de esta tabla.	INT	☑	☑
Id_tarea	Identifica la tarea asignada a un usuario.	INT	☑	☑
Id_proceso	Identifica el proceso al cual pertenece la tarea anterior.	INT	☑	☑
Usuario	Usuario al cual se le asignan tareas.	VARCHAR(20)	☐	☑
Fecha_asignación	Fecha en que el encargado asignó la tarea al usuario operador.	DATETIME	☐	☐
Fecha_real_inicio	Fecha en que realmente el operador inicia su tarea.	DATETIME	☐	☐
Fecha_fin_real	Fecha real en que operador termina su tarea.	DATETIME	☐	☐
Fecha_fin_prog	Fecha estimada para la	DATETIME	☐	☐

	finalización de una tarea para un usuario.			
Comentario	Información adicional para la tarea.	VARCHAR(70)	☐	☐
Id_flujo	Flujo de trabajo al que pertenece un proceso, y por lo tanto, la tarea.	INT	☐	☑
Id_estado	Estado personalizado en el cual se encuentra la tarea que un usuario realiza.	SMALLINT	☐	☑

19. USUARIOS. Esta tabla almacena la información de todos los usuarios existentes en el sistema y se relaciona con la tabla TIPOS_USUARIO.

Cuadro 19. Campos de la tabla USUARIOS.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Usuario	Identificador único para una persona que utilice el sistema.	VARCHAR(20)	☑	☐
Nombre	Nombre propio del usuario.	VARCHAR(30)	☐	☐
Apellido	Apellido del usuario.	VARCHAR(30)	☐	☐
Teléfono	Número telefónico del usuario.	VARCHAR(10)	☐	☐
Correo	Dirección de correo electrónico del usuario.	VARCHAR(30)	☐	☐
Contraseña	Contraseña de acceso al sistema.	VARCHAR(20)	☐	☐
Id_tipo_usuario	Tipo de usuario.	SMALLINT	☐	☑
Activo	Identifica el estado del usuario.	CHAR	☐	☐

20. TIPOS_USUARIO. SmartFlow cuenta con seis usuarios: Administrador general, Administrador de flujos, Administrador de proyectos, Analista de resultados, encargado y operador. Dichos tipos de usuario son almacenados en esta tabla.

Cuadro 20. Campos de la tabla TIPOS_USUARIO.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_tipo_usuario	Identificador único y correlativo de un tipo de usuario.	SMALLINT	☑	☐
descripción	Es cualquier de los nombres: Administrador general, Administrador de flujos, Administrador de proyectos, Analista de resultados, encargado u operador.	VARCHAR(30)	☐	☐

21. ARCHIVOS_POR_TAREA. Esta tabla almacena todos los archivos relacionados con una tarea dentro del sistema. Cada tarea se identifica por el proceso al que pertenece y este a su vez, al flujo de trabajo. Esta tabla se relaciona con la tabla TAREAS.

Cuadro 21. Campos de la tabla ARCHIVOS_POR_TAREA.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_archivo	Identificador único y correlativo para un archivo.	INT	☑	☐
Id_tarea	Especifica la tarea a la cual pertenece el archivo.	INT	☑	☑
Id_proceso	Indica al proceso al que pertenece la tarea.	INT	☑	☑
Id_flujo	Se refiere al flujo de trabajo al que pertenece el proceso del cual forma parte la tarea.	INT	☑	☑
Nombre	Nombre del archivo.	VARCHAR(20)	☐	☐
Descripción	Breve descripción del contenido del archivo.	VARCHAR(50)	☐	☐

archivo	Representación binaria del archivo.	LONGBLOB	☐	☐

22. FECHAS_AVANCE_PROCESO. En esta tabla se guardan las fechas en las que un proyecto es avanzado a algún proceso sucesor. Se relaciona con las tablas PROYECTOS_POR_PROCESO y PROCESOS_SUCESORES.

Cuadro 22. Campos de la tabla FECHAS_AVANCE_PROCESO.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_proyecto	Identifica al proyecto que será avanzado al siguiente proceso.	INT	☑	☑
Id_flujo	Identifica el flujo al que pertenece el proyecto.	INT	☑	☑
Id_proceso	Proceso actual del proyecto.	INT	☑	☑
Id_sucesor	Proceso sucesor del proceso actual.	INT	☑	☑
fecha	Fecha en que se avanzó al proceso sucesor.	DATE	☐	☐

23. ESTADOS_TAREAS_FLUJO. Almacena todos los estados personalizadas para las tareas dentro de un flujo.

Cuadro 23. Campos de la tabla ESTADOS_TAREA_FLUJO.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_estado	Identificador único y correlativo del estado.	INT	☑	☐
Id_flujo	Identifica el flujo al que pertenece un estado.	INT	☑	☑
Descripción	Nombre o descripción del estado.	VARCHAR(30)	☐	☐

Orden	Orden del estado.	INT	☐	☐
Estado_final	Indica si un estado es estado final o de terminación	CHAR	☐	☐

24. BITÁCORA. En esta tabla se almacenan todas las entradas de todos los usuarios al sistema. Se relaciona únicamente con la tabla USUARIOS.

Cuadro 24. Campos de la tabla BITÁCORA.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_registro	Identificador único y correlativo del registro o entrada al sistema.	INT	☑	☐
Usuario	Usuario que entra al sistema.	VARCHAR(20)	☐	☑
Fecha_hora	Fecha y hora en que un usuario entró al sistema.	DATETIME	☐	☐

25. MENSAJES. Esta tabla almacena todos los mensajes que se envían y se reciben por todos los usuarios dentro del sistema. Se relaciona con la tabla USUARIOS.

Cuadro 25. Campos de la tabla MENSAJES.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_mensaje	Identificador único y correlativo de un mensaje	INT	☑	☐
Emisor	Usuario que envía un mensaje.	VARCHAR(20)	☐	☑
Receptor	Usuario que recibe un mensaje.	VARCHAR(20)	☐	☑
Asunto	Asunto o tema del mensaje.	VARCHAR(50)	☐	☐

Mensaje	Texto del mensaje a enviar.	VARCHAR(200)	☐	☐
Fecha	Fecha y hora en que un usuario entró al sistema.	DATETIME	☐	☐

26. SERVICIO_TÉCNICO. En esta tabla se almacenan todas las dudas o comentarios enviadas por los usuarios del sistema, por lo que está relacionada con la tabla USUARIOS.

Cuadro 26. Campos de la tabla SERVICIO_TECNICO.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_mensaje	Identificador único y correlativo de un mensaje	INT	☑	☐
Fecha	Fecha y hora en que un usuario envió su duda o comentario.	DATETIME	☐	☐
Asunto	Asunto o tema del mensaje.	VARCHAR(50)	☐	☐
Descripción	Texto del mensaje a enviar.	VARCHAR(200)	☐	☐
Usuario	Usuario que solicita el servicio técnico.	VARCHAR(20)	☐	☑

27. AYUDA_EN_LÍNEA. El contenido de esta tabla es información predefinida que se mostrará a un usuario según su tipo; debido a esto, esta tabla está relacionada con la tabla TIPOS_USUARIO.

Cuadro 27. Campos de la tabla AYUDA_EN_LÍNEA.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id	Identificador único y correlativo de un registro de ayuda.	INT	☑	☐

Id_tipo_usuario	Especifica el tipo de usuario que podrá ver un enunciado específico.	INT	☐	☑
Descripción	Enunciado de ayuda.	TEXT	☐	☐
Id_idioma	Indica el idioma del enunciado de ayuda.	SMALLINT	☐	☑

28. PREFERENCIAS_POR_USUARIO. Por cada usuario, existirá un registro en esta tabla y se almacenará su configuración de tema e idioma del sistema. Debido a esto, esta tabla está relacionada con las tablas USUARIOS, TEMAS, e IDIOMAS.

Cuadro 28. Campos de la tabla PREFERENCIAS_POR_USUARIO.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Usuario	Especifica el usuario que almacenará sus preferencias.	INT	☑	☑
Id_tema	Indica el tema actual.	INT	☐	☑
Id_idioma	Indica el idioma actual.	SMALLINT	☐	☑

29. IDIOMAS. Esta tabla almacenará la información de los idiomas manejados por SmartFlow. Por el momento, únicamente dos valores predeterminados: inglés y español.

Cuadro 29. Campos de la tabla IDIOMAS.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_idioma	Identificador único y correlativo del idioma.	SMALLINT	☑	☐
Descripción	Inglés o español.	INT	☐	☐

30. PALABRAS_IDIOMA. En esta tabla se almacenan las palabras que forman parte del contenido de todo el sistema, especificando el idioma en el que se encuentra cada palabra, por lo que se relaciona con la tabla IDIOMAS.

Cuadro 30. Campos de la tabla PALABRAS_IDIOMA.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id	Identificador único y correlativo de.	INT	☑	☐
Nombre_variable	Nombre de la variable que almacena la palabra.	VARCHAR(20)	☐	☐
Descripción	Enunciado que forma parte del contenido, en inglés o español.	VARCHAR(100)	☐	☐
Id_idioma	Indica el idioma en que está determinada palabra.	SMALLINT	☐	☑

31. TEMAS. SmartFlow provee a un usuario tres distintos temas ya predefinidos para cambiar la apariencia del sistema. Esta tabla almacena la información de los tres temas existentes.

Cuadro 31. Campos de la tabla TEMAS.

Nombre del campo	Descripción	Tipo	Llave primaria	Llave foránea
Id_tema	Identificador único y correlativo del tema.	INT	☑	☐
Descripción	Nombre o descripción del tema.	VARCHAR(30)	☐	☐

VI. DISCUSIÓN

El diseño del sistema en general es capaz de mejorar el proceso actual de desarrollo de software, impulsando una nueva forma de trabajo, en donde la empresa misma que utiliza el sistema, adquiere el compromiso de definir su propio flujo de trabajo y obtiene la libertad de cambiarlo en cualquier momento, si es que este aun no se adecua a sus necesidades.

Sin embargo, es importante resaltar que el diseño del sistema no puede considerarse responsable de una mala utilización o de una definición errónea de un flujo de trabajo. SmartFlow únicamente es una herramienta de apoyo que se utiliza para la definición de flujos de trabajo y la implementación de dichos flujos para proyectos específicos dentro de una empresa.

Debido a que el sistema debe ser totalmente parametrizable por un usuario, durante las fases de análisis y diseño surgieron diversas inquietudes y sugerencias que en ciertas ocasiones llevaron a la reingeniería; tal fue el caso del diseño del módulo para la configuración de la plantilla. Además, surgieron funcionalidades adicionales necesarias a diseñar dentro del sistema pero que por razones de tiempo, limitación del alcance del proyecto y otras limitaciones no fueron incluidas, pero que no obstante son presentadas más adelante como recomendaciones a personas interesadas en desarrollar el proyecto.

VII. CONCLUSIONES

1. El sistema diseñado cumple con la funcionalidad descrita en los objetivos propuestos en el protocolo del presente trabajo.

2. El módulo diseñado permite a un usuario final definir los procesos y subprocesos que forman parte del flujo de trabajo de su empresa; detallando el orden y definición de todas las actividades necesarias que se llevan a cabo en la empresa como parte del desarrollo de un proyecto.

3. El módulo diseñado permite definir los parámetros y condiciones asociadas para cada proceso y subproceso específicos; asegurando siempre que dichos elementos reflejen la realidad del trabajo que se realiza físicamente en la empresa.

4. El módulo provee la capacidad de redefinir en cualquier momento, ya sea parcial o completamente un proceso específico o el flujo general. Esta capacidad se considera esencial ya que debido a los cambios a los que está sujeta una empresa, el flujo definido puede que no se adapte a las necesidades nuevas de la empresa.

5. El producto de la definición del flujo de trabajo en el módulo administrativo se utiliza en el módulo de usuario, en donde es posible alimentar la información necesaria para inferir sobre el desempeño del flujo de trabajo y el rendimiento y control de los proyectos que lo utilizan.

VIII. RECOMENDACIONES

Para la implementación de SmartFlow, se recomienda lo siguiente:

- Servidor con Sistema Operativo Linux, específicamente Fedora Core 4.
- Lenguaje de programación PHP.
- Manejador de base de datos MySQL.
- Servidor Web Apache

(Ver Sección C: Especificaciones Técnicas en el diseño, para las razones de su selección).

Se recomienda utilizar un servidor de base Linux por las siguientes razones:

- La seguridad que éste provee.
- El costo es totalmente gratuito.
- Debido a que se trata de un sistema operativo de fuente abierta, el soporte disponible en la red es extenso.
- Provee estabilidad.
- Pone a la disposición del usuario los medios necesarios para que él se encargue de la seguridad del sistema.
- Ahora bien, específicamente Fedora Core 4 provee un sistema operativo completo de propósito general para el desarrollo de aplicaciones open-source, proveyendo regularmente paquetes de actualizaciones para el sistema.

A continuación se presentan varias recomendaciones adicionales con el fin de expandir el funcionamiento general ofrecido por el diseño de SmartFlow:

1. Diseño y desarrollo de un módulo que permita realizar gráficamente todos los elementos ya definidos en un flujo de trabajo, así como lo pueden ser sus procesos y tareas relacionadas.

2. Diseñar e implementar un módulo para permitir las revisiones de las tareas asignadas a un usuario dentro del mismo sistema.

3. Desarrollar un módulo de flujos “inteligente”, que sea capaz de indicarle a un usuario cuando la definición de un flujo de trabajo o sus partes relacionadas es incorrecta.

4. Implementar un foro de discusión para que todos los usuarios del sistema participen en él compartiendo ideas y comentarios.

5. Desarrollar un módulo que proporcione a cada usuario un espacio propio dentro del sistema, una especie de diario privado al cual únicamente tiene acceso ese usuario.

6. Desarrollar un mecanismo que permita a un usuario obtener una nueva contraseña de una forma más segura. Se propone la utilización de una pregunta y una respuesta personales, las cuales serían solicitadas al usuario la primera vez que este ingrese al sistema. De esta forma, cuando el usuario solicite una nueva clave, debe responder a su respectiva pregunta ya almacenada en el sistema y si la respuesta proporcionada coincide con la almacenada, el sistema procede a generar una nueva contraseña para este usuario.

7. Para desarrollar las funciones del Administrador general, es decir: Administración de usuarios de flujos y bitácora de ingresos al sistema, se recomienda el trabajo de un programador y se espera que éste finalice las funciones requeridas en dos semanas y media.

8. Para desarrollar las funciones del Administrador de flujos, específicamente el Módulo de flujos y de usuarios de proyectos, se recomienda el trabajo de tres desarrolladores, cuyos resultados finales deben esperarse un mes después de haber iniciado el trabajo en dichos módulos.

IX. BIBLIOGRAFÍA

A. REFERENCIAS BIBLIOGRÁFICAS

Ambler, S. 2006. *Examining the Agile Manifesto*. Ambysoft.

Bauer, M. 2005. *Successful Web Development Methodologies.* http://www.sitepoint.com/article/successful-development

Beedle, M., et al. 2001. *Manifesto for Agile Software Development.* http://www.agilemanifesto.org

Bray, M. 2005. *Object-Oriented Analysis.* Carnegie Mellon University. http://www.sei.cmu.edu/str/descriptions/ooanalysis_body.html

Date, J.C. 1993. *Introducción a los Sistemas de Bases de Datos*, Vol. 1, 5ta. edición. Addison-Wesley Iberoamericana, Argentina.

Evaluación y Análisis de los Procesos de Desarrollo del Software. 2003. http://www.synspace.com/ES/Assessments/spa.html

E-workflow - the workflow portal. 2000-2003. http://www.e-workflow.org/

Fahey, T. 1994. *Diccionario de Internet.* Traducción de H. Acuña. Prentice-Hall Hispanoamericana, S.A. México. 220 págs.

Fowler, M. 2005. *The New Methodology.* http://www.martinfowler.com/articles/newMethodology.html #id109070

Haynes, M. 1996. *Administración de Proyectos.* Editorial Iberoamericana. Mexico. 85 págs.

Highsmith, J. 2001. *History: The Agile Manifesto.* Agile Alliance

Hollingsworth, D. 1995. *The Workflow Reference Model*. The Workflow Management Coalition Specification.

Jeffries, R. 2001. *What is Extreme Programming?* http://www.xprogramming.com/xpmag/whatisxp.htm

Jensen, R. y W. Jensen. 2003. *A Pair Programming Experience.* CrossTalk. http://www.stsc.hill.af.mil/crosstalk/2003/03/jensen.html

Joyanes, L. e I. Zahonero. 1998. *Estructura de Datos: Algoritmos, abstracción y objetos.* McGraw-Hill Interamericana, España. 857 pp.

Korth, H. y A. Silberschatz. 1986. *Database Systems Concepts*, McGraw-Hill, New York.

Laplante, P. y N. Penn. 2004. *The Demise of the Waterfall Model Is Imminent and Other Urban Myths.* ACM Queue vol. 1, no.10

Manual de Diseño de Procesos. Universidad Miguel Hernández. 11 pp.

Marchesi, Michele. *The New XP*. http://www.agilexp.org/downloads/TheNewXP.pdf

Navarro, E. *Gestión y Reingeniería de Procesos.* Improven Consultores. http://www.improven-consultores.com/paginas/documentos_gratuitos/gestion_reingenieria.php

Palacio, J. 2005. *Gestión y modelos para la eficiencia en empresas de desarrollo de software*. http://www.baquia.com/imprimir.php?id=9651

Pressman, R. 1993. *Ingeniería del Software, un enfoque práctico*. Mc-Graw Hill, México.

Schuldt, J. 1998. *Reingeniería de Procesos*. http://www.geocities.com/WallStreet/Exchange/9158/reingen.htm

Wang, J. 2001. Object-Oriented Analysis Methodology. http://www.umsl.edu/~sauter/analysis/488_f01_papers/wang.htm

Wells, D. 1999. *The Rules and Practices of Extreme Programming.* http://www.extremeprogramming.org/rules.html

Whitten, N. 1995. *Managing Software Development Projects*. John Wiley & Sons, Inc. 2nd.Edition. Canada. 384 págs.

Workflow Research. 2004. http://www.workflow-research.de/

Yourdon, E. 1989. *Modern Structured Analysis.* Yourdon Press, Prentice-Hall, Inc. United States of America. 672 págs.

B. REFERENCIAS TÉCNICAS EN LÍNEA

http://www.mysql.com

http://www.php.net

http://www.apache.org

http://fedora.redhat.com/

http://www.uml.org/

C. DICCIONARIOS DE TÉRMINOS TÉCNICOS

http://www.webopedia.com

http://www.wikipedia.com

X. APÉNDICE 1

A. UML

1. Historia y definición. UML (*Unified Modelling Language*, por sus siglas en inglés) es un lenguaje que se utiliza para modelar y construir elementos que forman parte de un sistema orientado a objetos. Actualmente, UML es una de las herramientas más utilizadas y esto se debe a que incorpora varios métodos y notaciones ya conocidos y los unifica, constituyéndose así como un estándar en el ámbito del análisis y desarrollo orientado a objetos.

La notación UML se deriva y unifica las tres metodologías de análisis y diseño orientado a objetos más extendidas:

- Metodología de Grady Booch para la descripción de conjuntos de objetos y sus relaciones.
- Técnica de modelado, orientada a objetos de James Rumbaugh.
- Aproximación de Ivar Jacobson, mediante la metodología de casos de uso.

El desarrollo de UML comenzó a finales de 1994 cuando Grady Booch y Jim Rumbaugh de Rational Software Corporation empezaron a unificar sus métodos. A finales de 1995, Ivar Jacobson se incorpó a ellos, aportando su propio método. De estas metodologías, la de Booch y Rumbaugh pueden ser descritas como orientadas a objetos, pues se enfocan hacia el modelado de los objetos que componen el sistema, su relación y colaboración. Por otra parte, la metodología de Jacobson es más orientada al usuario, pues su método se basa en los escenarios de uso.

En 1997 UML 1.1 fue aprobada por la OMG (*Object Management Group*), convirtiéndose en la notación estándar para el análisis y el diseño orientado a objetos. La especificación de UML por la OMG declara:

«El Lenguaje Unificado de Modelado (UML - Unified Modelling Language) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema de software intensivo. UML ofrece una forma estándar de escribir los planos de un sistema, incluyendo aspectos como los procesos de negocios y las funciones del sistema, así como aspectos concretos como instrucciones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables de software» (OMG 1997-2005).

2. Características. UML establece su propia notación y, por lo tanto, define su propio conjunto de diagramas específicos para modelar diferentes aspectos de un sistema. UML permite el modelado de las partes estáticas así como las dinámicas, esto a través de diagramas que representan la arquitectura del sistema. Cabe mencionar que UML es únicamente un lenguaje de especificación y no un método o procedimiento, pues únicamente se utiliza para definir objetos y artefactos y modelar el comportamiento de los mismos dentro del sistema. UML no es en sí, ni tampoco especifica alguna metodología específica.

UML incluye en su definición el concepto de modelo y se refiere a una abstracción de un sistema del mundo real tomando en cuenta sus propósitos y los aspectos importantes de su entorno. Un modelo es una descripción completa de un sistema desde una perspectiva en particular.

3. Tipos de diagramas. Un diagrama es una representación gráfica de un conjunto de elementos. Debe representar los elementos semánticos de un modelo y el significado no debe verse afectado por la forma de representación.

UML define nueve tipos de diagramas, los cuales se detallan a continuación:

a. Diagrama de casos de uso. Este diagrama modela la funcionalidad de un sistema, incluyendo la relación con las personas que interactúa. Se modela un comportamiento del sistema a través de casos de uso o situaciones específicas, que son las operaciones que se pueden llevar a cabo dentro del sistema, incluyendo siempre la participación de los actores. En este diagrama se

enfatiza qué es lo que hace el sistema y no tanto cómo lo hace. Está relacionado con un conjunto de escenarios o situaciones específicas en donde una persona interactúa con el sistema. Este diagrama tiene tres componentes principales:

1) *Actores.* Se refiere a los usuarios del sistema, ya sean personas o sistemas de computadoras.
2) *Casos de uso.* Se refiere a operaciones o tareas que un actor realiza entro del sistema. Un caso de uso es una pieza individual de trabajo para un actor y describe la funcionalidad que el sistema debe tener.
3) *Relaciones.* Son los elementos de unión entre actores y casos de uso. UML provee tres diferentes tipos de relaciones:
    a) *Comunica.* Relación entre un actor y un caso de uso. Indica la participación de un actor en el caso de uso determinado.
    b) *Usa.* Relación entre dos casos de uso, y define la inclusión del comportamiento de un caso de uso en otro.
    c) *Extiende.* También relaciona dos casos de uso, y se refiere a la situación en donde un caso de uso es una especialización de otro.

Un diagrama de caso de uso se utiliza generalmente para determinar nuevos requerimientos mientras el sistema se analiza con el fin de comunicarse con los clientes debido a su notación simple, y así generar casos de prueba, es decir, crear una colección de escenarios, situaciones y posteriormente las respectivas pruebas que pueden realizarse sobre esta colección.

b. Diagrama de clases. Modela la estructura estática del sistema y da un panorama del sistema mostrando sus clases y las relaciones entre ellas.
Como ya se mencionó, los diagramas de clases son estáticos pues únicamente muestran qué es lo que interactúa, pero no muestran qué sucede cuando esta interacción ocurre.

Un diagrama de clase UML no sólo incluye las relaciones entre clases, asociaciones, generalizaciones y los contenidos de los atributos, sino que también incluye elementos adicionales como interfaces, plantillas, relaciones e instancias como objetos y enlaces. El elemento fundamental de un diagrama de clase es naturalmente una clase, que representa un conjunto de objetos con estructura, comportamiento y relaciones similares. En UML existen tres tipos de relaciones:

1) *Asociación.* Es un enlace entre instancias de dos clases. Existe una asociación entre dos clases si una instancia de una clase debe saber acerca de otra para realizar su trabajo.
2) *Agregación.* Se refiere a una relación en donde una clase pertenece a una colección. Se define también como una asociación especializada en donde todo se relaciona con sus partes pero el "todo" es más importante que las partes.
3) *Generalización.* Es otro nombre que se le pueda dar a la herencia de una clase a otra. Se refiere a una relación entre dos clases en donde una clase es una versión especializada de la otra.

c. Diagrama de objetos. Un diagrama de objeto es una instancia de un diagrama de clase, mostrando una vista del estado en el cual se encuentra el sistema en un momento determinado. Como los diagramas de clase pueden contener objetos, un diagrama de clase con objetos y sin clases es simplemente un diagrama de objeto. Los diagramas de objetos se pueden utilizar para probar la exactitud y precisión de un diagrama de clases.

d. Diagrama de estados. Un objeto tiene comportamiento y un estado. El estado del objeto depende de su actividad o condición actual. El diagrama de estados muestra el comportamiento y los posibles estados de los objetos en respuesta a estímulos externos. Además, modela el flujo de control dinámico de estado a estado dentro del sistema y muestra las transiciones que causan un cambio de estado.

e. Diagrama de secuencia. Es un diagrama de interacción que muestra cómo se realizan las operaciones en el sistema, especificando los mensajes qué se envían y cuándo deben enviarse, proveyendo un mapa secuencial de los mensajes que son pasados entre los objetos a través del tiempo. Este diagrama es extremadamente útil cuando se desean modelar funciones que dependen del tiempo (aplicaciones de tiempo real, por ejemplo). Está compuesto de dos dimensiones: el tiempo y los objetos por un lado, y los actores que participan en un evento determinado por el otro.

f. Diagrama de colaboración. Es similar al diagrama de secuencia, únicamente que la información la muestra utilizando otro enfoque. Este diagrama se enfoca más en los papeles de los objetos en lugar de los tiempos en los cuales se envían los mensajes y describe las relaciones entre objetos en términos de mensajes secuenciales.

g. Diagrama de actividades. Este diagrama ilustra la naturaleza dinámica de un sistema a través del modelado del flujo de control de una actividad a otra dentro del sistema. Es básicamente un diagrama de flujo que se enfoca en el flujo de actividades que están involucradas a un proceso específico. Una actividad representa una operación en alguna clase en el sistema que resulta en un cambio de estado para ese sistema. Es muy similar a un diagrama de estados, pero permite la concurrencia de actividades, así como su control secuencial. Los principales componentes de un diagrama de actividades son: estados de actividad, estados de acción, transiciones y objetos.

h. Diagrama de componentes. Describe la organización de los componentes físicos en un sistema y muestran además las relaciones, dependencias, comunicación, localización y condiciones entre los componentes. Hay diferentes tipos de componentes, por ejemplo: ejecutables, código fuente y APIs. En este diagrama deben incluirse librerías, archivos, ejecutables y

documentos que formen parte del sistema y la relación entre ellos. Se dice que un diagrama de componente es un análogo físico a un diagrama de clases.

i. Diagrama de despliegue. En este diagrama se indica la situación física de los componentes lógicos desarrollados. Es decir, se sitúa el software en el hardware que lo contiene y cada hardware se representa como un nodo. Con base a esto, muestra la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria.

B. BASES DE DATOS

1. Introducción. Un sistema manejador de bases de datos (DBMS – Database Management System, por sus siglas en inglés) es un conjunto de datos interrelacionados y el conjunto de programas necesarios para acceder a estos datos, cuyo propósito es almacenar y extraer estos datos de forma eficiente. Este conjunto de datos o elementos es lo que se conoce como Base de Datos.

El objetivo de una base de datos es almacenar la información de modo que:

- No haya redundancia ni inconsistencia en los datos.
- La información sea fácil de recuperar.
- Varios usuarios puedan consultar la misma información al mismo tiempo.
- Se garantice la seguridad; es decir, que únicamente tengan acceso a la información los usuarios determinados para esto.
- Se cumplan ciertas restricciones de consistencia.

2. Abstracción de datos. Un sistema DBMS debe además proveer a sus usuarios una vista abstracta de la información almacenada en él. Esto quiere decir, ocultar ciertos detalles de la forma en que la información está almacenada. A pesar de esto, la información aun debe poder accederse de forma eficiente y rápida. El concepto de eficiencia lleva implícito el diseño de estructuras de datos complejas, lo cual debe ocultarse al usuario del sistema. Debido a esto, se definieron tres niveles de abstracción a los cuales una base de datos puede referirse.

- Nivel físico. Se refiere a la definición de las estructuras de datos detalladas que especifican cómo se almacena la información.

- Nivel conceptual. Únicamente describe cuál es la información almacenada en la base de datos y como se relaciona entre sí.

- Nivel de vista. Se utiliza para describir una parte de la base de datos que interese.

3. Modelos de datos. Para describir la estructura de una base de datos, es necesario definir una colección de herramientas conceptuales para describir los datos a almacenar, las relaciones entre ellos, la semántica de los datos y las restricciones de consistencia. Esta colección se denomina Modelo de datos y existen tres tipos: Lógicos basados en objetos, Lógicos basados en registros y de datos físicos. Para efectos del presente trabajo, el único modelo que aplica es el Lógico basado en Objetos el cual se explica a continuación.

Un modelo lógico orientado a objetos se define para describir la información a nivel conceptual y de vista. Provee una estructura flexible y la capacidad de especificar restricciones de datos explícitamente. El modelo de este tipo más conocido es el Modelo entidad-relación, el cual se basa en una percepción del mundo real. Este modelo se basa en la definición de objetos llamados entidades que son objetos que existen y se distinguen de otros por medio de sus atributos. Además, en

este modelo se definen relaciones entre entidades, las cuales se refieren a asociaciones entre las mismas y pueden tener cierta cardinalidad y opcionalidad. La cardinalidad expresa el número de entidades a las cuales una entidad puede relacionarse. La opcionalidad define si existe una relación o no entre 2 entidades.

En el momento que se traslada el modelo entidad-relación a un DBMS, éste se convierte en una base de datos en donde las entidades son denominadas tablas y se requiere de una abstracción a nivel físico. Aquí, ya aparece el concepto de llave primaria, el cual se define como un conjunto de atributos que identifican de forma única a cada entidad dentro de un conjunto de entidades. Además de la llave primaria, existe la llave foránea o secundaria, la cual se refiere al conjunto de atributos que no identifican de forma única a un registro dentro de una tabla pero que se utiliza para hacer referencia a otra tabla. Esta llave siempre será la llave primaria de otra tabla.

C. PHP, MySQL y APACHE

1. PHP. PHP es un lenguaje diseñado para aplicaciones que califican como cliente servidor dentro de la rama del Internet. Difiere de un CGI en el hecho de que no es necesario estar escribiendo comandos y rutinas que devuelvan código en HTML. PHP es capaz de generar el código por sí mismo e inclusive provee comandos para la generación de tablas para reportes obtenidos de bases de datos remotas.

Lo que distingue a PHP de herramientas comunes que se utilizan del lado de la aplicación cliente como lo es el Javascript, es el hecho de que el código se ejecuta del lado del servidor. Esto incrementa la eficiencia y la seguridad de cualquier sistema. La eficiencia es incrementada ya que todos los procesos, y en especial los relacionados con bases de datos, se ejecutan por el servidor y lo único que es

transmitido a través de la red es el código HTML ya generado, para evitar así el tráfico innecesario de información.

La seguridad y confidencialidad de todos los aspectos del sistema obtienen un nuevo nivel con PHP. Debido a que PHP lo único que devuelve es código en HTML todos los programas escritos en el propio lenguaje nunca son vistos por el usuario final. El usuario sólo tiene acceso a ver el código HTML ya generado por PHP. En cuanto a la seguridad, un sistema que utiliza PHP le permite conectividad remota a cualquier tipo de base de datos. Lo que esto significa es que la página Web solo hará una petición al servidor y este se encargará de tramitar ya de forma local las transacciones necesarias con el servidor de bases de datos y devolverá únicamente la información deseada ya en formato HTML. Una excelente forma de aislar la información confidencial pero al mismo tiempo muy efectiva para compartir la información pública.

Como ya se discutió, PHP es un poderos lenguaje e intérprete de comandos. Sin importar cómo lo utilice siempre podrá tener acceso a los archivos en el servidor, ejecutar comandos y abrir conexiones de red. Estas propiedades convierten cualquier proceso que se corra en el servidor vulnerable por definición. PHP está diseñado para contrarrestar este efecto y ser más seguro que los lenguajes utilizados tradicionalmente para la ejecución de CGI's en un servidor. Si se utiliza una configuración correcta y las opciones de seguridad disponibles adecuadas, se podrá tener la combinación de accesibilidad y seguridad que se necesita.

Las funciones de PHP se describen a continuación:

a. Creación de Imágenes GIF. PHP no está limitado a generar resultados en formato HTML. También puede ser utilizado para generar imágenes tipo GIF. Inclusive se pueden generar secuencias de imágenes. Esta propiedad puede ser muy útil para generar imágenes que apoyen resultados que de otra forma tan sólo podrían ser representados con texto, es decir gráficas. Por otro lado esta

característica se puede utilizar para poder tomar una imagen cualquiera y modificarla del lado del servidor y luego devolver la imagen ya modificada a la página Web. Esta utilidad a pesar de ser muy innovadora se ha probado ya muy estable. La gama de funciones que presenta PHP para la manipulación de imágenes es muy completa y robusta.

b. Autenticación de HTTP. Al ser configurado como un módulo en su servidor PHP es posible tener control de autenticación a través del protocolo HTTP común de Internet. Es posible utilizar una función nativa de PHP para enviar un mensaje de "Requerir Autorización" al navegador cliente causando de esta forma que el navegador despliegue una ventana que solicite el ingreso de un nombre de usuario y su contraseña. Una vez que el usuario ingrese su información, la página original que solicitó la autenticación será llamada de nuevo con las variables de ambiente necesarias igualadas a los valores ingresados para lograr un acceso final a los contenidos de la página.

c. Cookies. PHP transparentemente soporta los famosos Cookies. Los cookies son un mecanismo utilizado para almacenar data en el navegador remoto y por ende poder identificar los usuarios que estén retornando a la página. Cualquier cookie que le sea enviado será transformado en una variable de PHP y así se podrá manejar como cualquier otro tipo de dato nativo de PHP.

d. Grabación de archivos. PHP es capaz de recibir archivos y grabarlos en su servidor. Esta característica le permite a sus usuarios grabar o subir archivos tanto de texto como binaros. En conjunto con sus funciones de autenticación y manipulación de archivos se obtiene un control completo sobre quienes tienen autorización de subir o grabar archivos y que se debe de hacer con este archivo una vez que es recibido. PHP también recibe grabaciones de formato PUT. Este formato es utilizado por algunas aplicaciones que no son necesariamente navegadores. Esta función expande la funcionalidad de su sistema al remover la limitación del navegador como cliente exclusivo.

e. Uso de archivos remotos. La mayoría de funciones de PHP que reciben un nombre de archivo como parámetro son totalmente compatibles con nombres de archivo especificados a través de un URL de HTTP o FTP. Por ejemplo, esto se puede utilizar para abrir un archivo en un servidor remoto, analizar lo devuelto para encontrar la información deseada y luego utilizar esta información para una búsqueda en una base de datos o simplemente presentar esta información en un formato equivalente al del resto de su página Web. También es posible escribir a un sitio FTP siempre y cuando se conecte como un usuario con los privilegios necesarios para realizar esta operación y el archivo no exista en el directorio destino. Para conectarse como un usuario no anónimo es necesario especificar un nombre de usuario y posiblemente una contraseña dentro del URL.

f. Manejo de conexiones. Internamente en PHP se mantiene un estatus de conexión. Los tres estados posibles son: normal, abortado y tiempo vencido. Cuando un script de PHP está corriendo normalmente, el estado se identifica como normal. Si el cliente remoto se desconecta se enciende la bandera de abortado. Esto ocurre usualmente cuando el usuario presiona el botón de Stop en su navegador. Si se excede el tiempo límite definido sin una respuesta remota se enciende la bandera de tiempo vencido. Se puede decidir si el corte en la conexión remota aborta el script de PHP. Muchas veces es conveniente que el programa acabe ejecución total sin importar que el usuario final reciba el resultado.

g. Conexiones a bases de datos persistentes. Las conexiones persistentes son vínculos de SQL que no se cierran cuando acaba la ejecución de su script. Cuando se solicita una conexión de este tipo, PHP verifica que no exista una conexión persistente exactamente igual. Si ya existe, la usa. La conexión es idéntica si se conecta al mismo servidor y utiliza el mismo nombre de usuario y contraseña.
Las personas que no están familiarizadas con el funcionamiento de los navegadores y cómo estos distribuyen su carga pueden confundir las conexiones persistentes por algo que no son. En particular no le dan la habilidad de abrir “sesiones de usuarios”

en el mismo vínculo de SQL, no dan la habilidad de crear una transacción eficientemente. De hecho para aclarar el tema estas conexiones no agregan funcionalidad sobre las conexiones no persistentes. Únicamente hacen que la comunicación sea más eficiente. Esto reduce la carga en el servidor, la base de datos y el navegador de forma que todas las transacciones se ejecuten rápidamente y se puedan manejar más usuarios de una forma simultánea.

2. MySQL. MySQL es un sistema manejador de bases de datos relacional. Una base de datos relacional almacena los datos en tablas separadas y no en un área de almacenamiento común. Esto incrementa la velocidad y flexibilidad del manejador. Los vínculos entre las tablas son definidos por relaciones, lo que hace posible la combinación de datos de diferentes tablas en cuanto esto sea necesario. El lenguaje básico de MySQL es el SQL (Structured Query Language), el lenguaje más común y robusto utilizado para accesar y manejar bases de datos.

MySQL consiste de un servidor multi-hilos que soporta varios programas clientes y librerías, herramientas administrativas e interfases de programación. MySQL puede aprovechar múltiples procesadores sin ningún problema.

A continuación se presentan los requerimientos de espacio de almacenamiento para distintos tipos de datos utilizados por MySQL:

Cuadro 32. Requerimientos de espacio de almacenamiento para columnas de tipo numérico.

Tipo de la columna	Espacio requerido
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT	4 bytes
INTEGER	4 bytes
BIGINT	8 bytes

FLOAT(X)	4 si X <= 24 o 8 si 25 <= X <= 53
FLOAT	4 bytes
DOUBLE	8 bytes
DOUBLE PRECISION	8 bytes
REAL	8 bytes
DECIMAL(M,D)	M+2 bytes si D > 0, M+1 bytes si D = 0 (D+2, si M < D)
NUMERIC(M,D)	M+2 bytes si D > 0, M+1 bytes si D = 0 (D+2, si M < D)

Cuadro 33. Requerimientos de espacio de almacenamiento para columnas de tipo texto.

Tipo de la columna	Espacio requerido
CHAR(M)	M bytes, 1 <= M <= 255
VARCHAR(M)	L+1 bytes, donde L <= M y 1 <= M <= 255
TINYBLOB, TINYTEXT	L+1 bytes, donde L < 2^8
BLOB, TEXT	L+2 bytes, donde L < 2^16
MEDIUMBLOB, MEDIUMTEXT	L+3 bytes, donde L < 2^24
LONGBLOB, LONGTEXT	L+4 bytes, donde L < 2^32
ENUM('value1','value2',...)	1 o 2 bytes, dependiendo del numero de valores de enumeración (65535 valores como máximo)
SET('value1','value2',...)	1, 2, 3, 4 u 8 bytes, dependiendo del número de miembros del set (64 miembros máximos)

Cuadro 34. Requerimientos de espacio de almacenamiento para columnas de tipo fecha.

Tipo de la columna	Espacio requerido

DATE	3 bytes
DATETIME	8 bytes
TIMESTAMP	4 bytes
TIME	3 bytes
YEAR	1 byte

Cuadro 35. Conversión entre tipos estándar de datos y tipos MySQL.

Tipos estándar (Oracle, MsSQL Server, etc.)	Tipo MySQL
BINARY(NUM)	CHAR(NUM) BINARY
CHAR VARYING(NUM)	VARCHAR(NUM)
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT1	TINYINT
INT2	SMALLINT
INT3	MEDIUMINT
INT4	INT
INT8	BIGINT
LONG VARBINARY	MEDIUMBLOB
LONG VARCHAR	MEDIUMTEXT
MIDDLEINT	MEDIUMINT
VARBINARY(NUM)	VARCHAR(NUM) BINARY

3. Apache. Apache es un servidor HTTP de fuente abierta para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1 (RFC 2616) y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que originalmente Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en inglés, a patchy server (un servidor parcheado).

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation.

Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, sin embargo fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

En la actualidad (octubre de 2005), Apache es el servidor HTTP más usado, siendo el servidor HTTP del 68% de los sitios Web en el mundo y creciendo aún su cuota de mercado (estadísticas históricas y de uso diario proporcionadas por Netcraft.)

Versión 2.x

El núcleo 2.x de Apache tiene varias mejoras clave sobre el núcleo de Apache 1.x. Estas mejoras incluyen threads de UNIX, mejor soporte para plataformas no Unix (como Windows), un nuevo API, y soporte de IPv6

Módulos

El servidor de base puede ser extendido con la inclusión de módulos entre los cuales se encuentran:

mod_perl - Páginas dinámicas en Perl.

mod_php - Páginas dinámicas en PHP.

mod_python - Páginas dinámicas en Python.

mod_jk - Conector para enlazar con el servidor Jakarta Tomcat de páginas dinámicas en Java (servlets y JSP).

mod_ssl - Comunicaciones Seguras.

mod_rewrite - reescritura de direcciones servidas.

XI. APÉNDICE 2

A. EJEMPLO DE UTILIZACIÓN

A continuación se muestra un ejemplo que explica la creación de un flujo de trabajo junto con sus procesos y tareas relacionadas. Debido al enfoque del presente trabajo, es decir el desarrollo de software, se presenta un ejemplo de forma detallada donde se define un flujo de trabajo para la creación de aplicaciones de escritorio utilizando la herramienta SmartFlow.

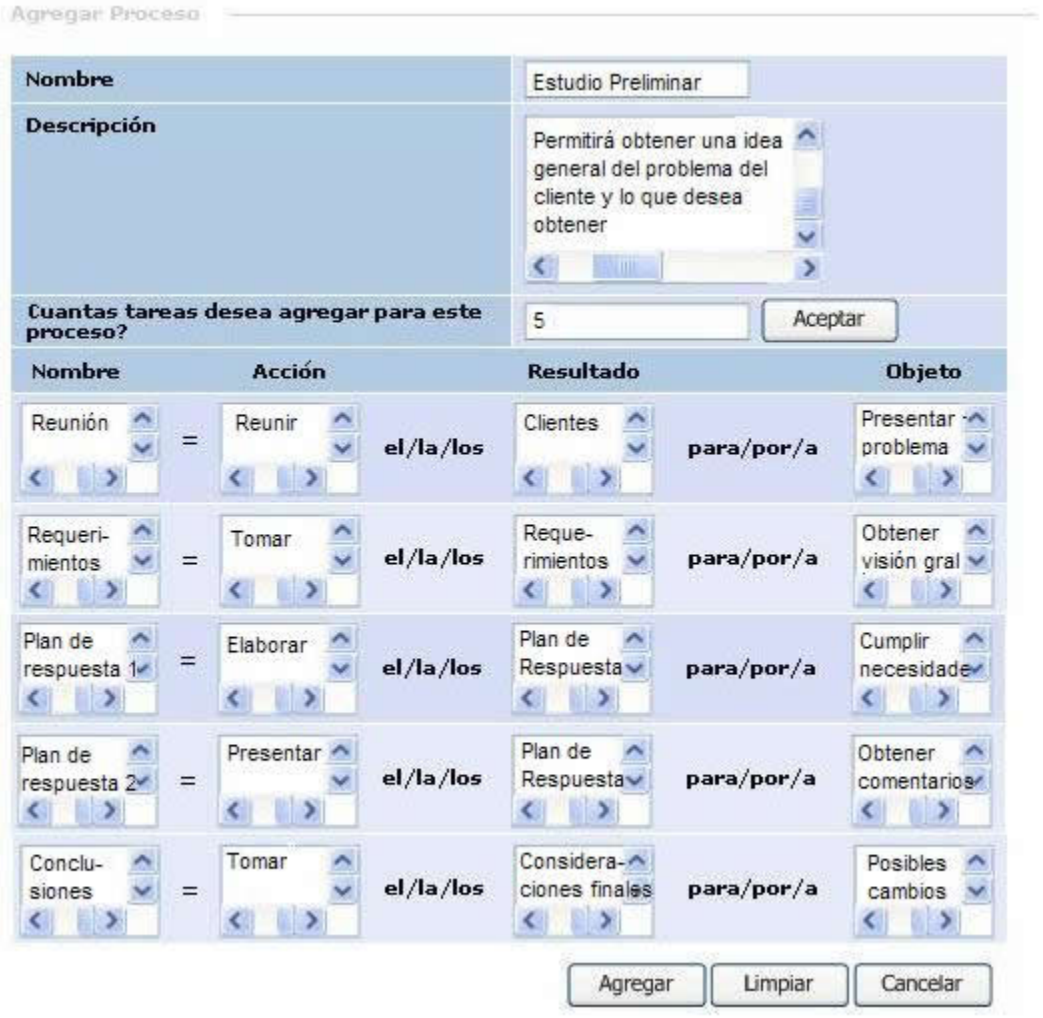
1. Creación del flujo de trabajo. Para crear un nuevo flujo de trabajo dentro del sistema, deberá especificarse el nombre propio para el mismo, así como una breve descripción.

Ilustración 95. Agregar flujo de trabajo (ver ilustración 78)

1. Definición de procesos y tareas. Para crear nuevos procesos y las tareas dentro de ellos, deberá utilizarse la interfaz Agregar Proceso(s), ver Ilustración 83.

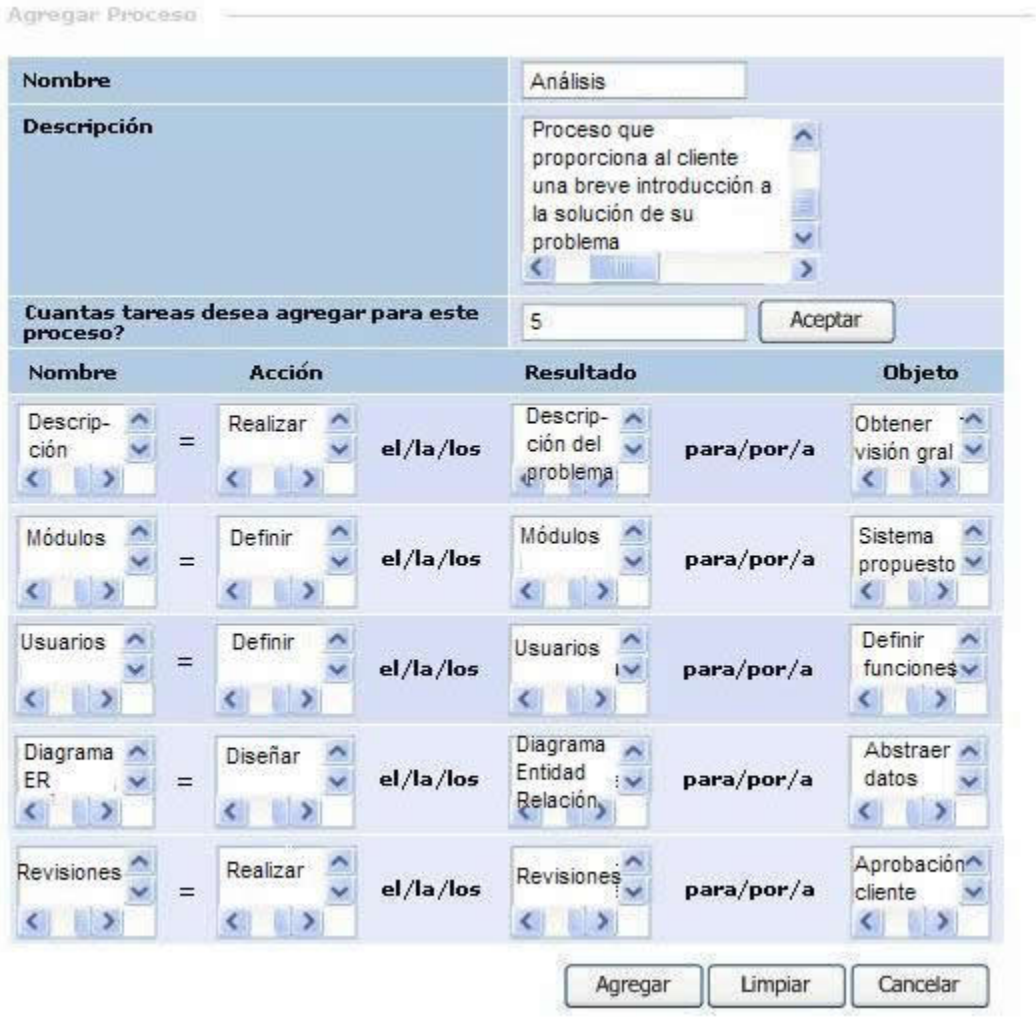
a. Proceso 1: Estudio preliminar

Ilustración 96. Agregar proceso de trabajo: Estudio preliminar, con sus respectivas tareas



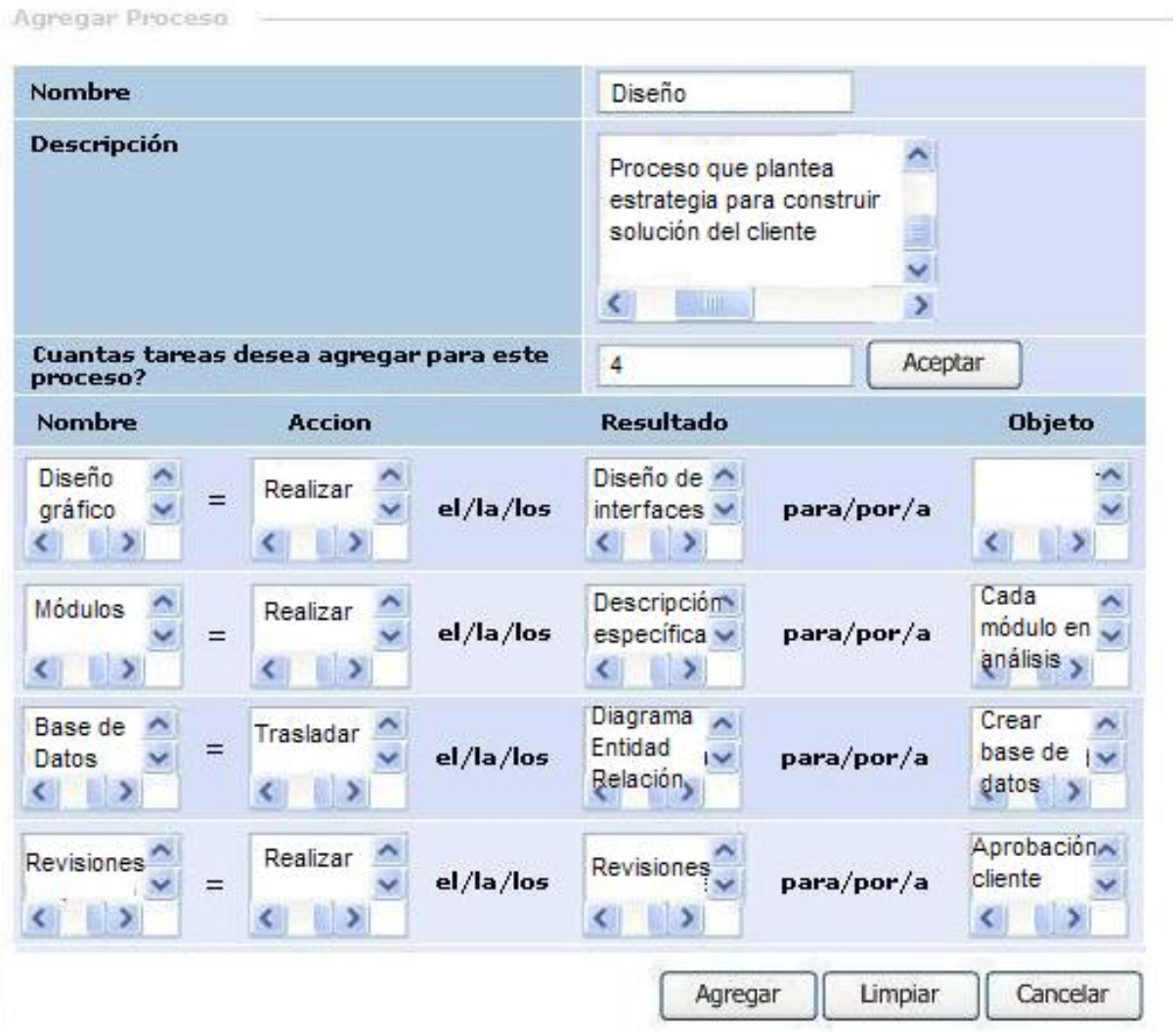
b. Proceso 2: Análisis

Ilustración 97. Agregar proceso de trabajo: Análisis, con sus respectivas tareas



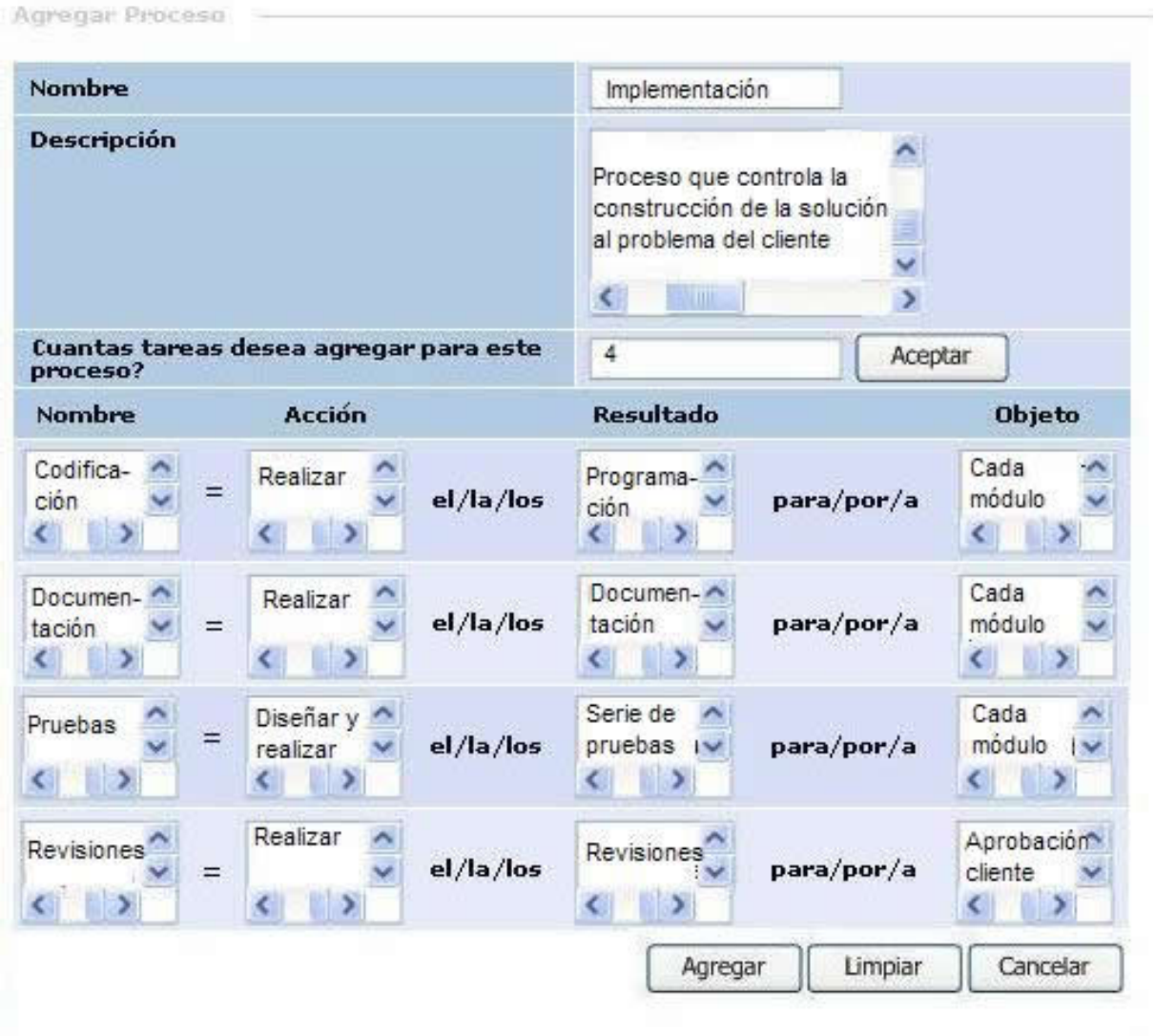
c. Proceso 3: Diseño

Ilustración 98. Agregar proceso de trabajo:
Diseño, con sus respectivas tareas



d. Proceso 4: Implantación

Ilustración 99. Agregar proceso de trabajo:
Implantación, con sus respectivas tareas



2. Establecer procesos antecesores y sucesores. Para que la información dentro del flujo de trabajo siga un orden, es necesario establecer los procesos antecesores y sucesores para todos los procesos definidos anteriormente.

a. Proceso 1

Ilustración 100. Establecer procesos antecesores para el proceso 1

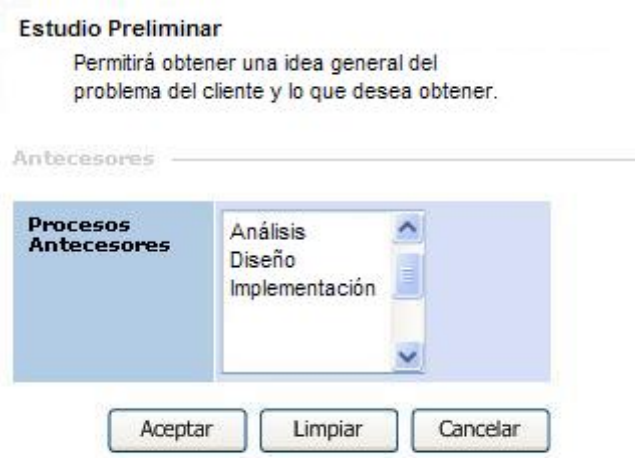
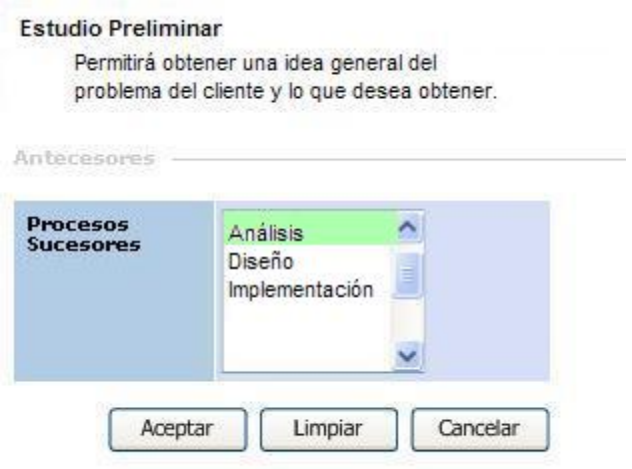


Ilustración 101. Establecer procesos sucesores para el proceso 1



b. Proceso 2

Ilustración 102. Establecer procesos antecesores para el proceso 2

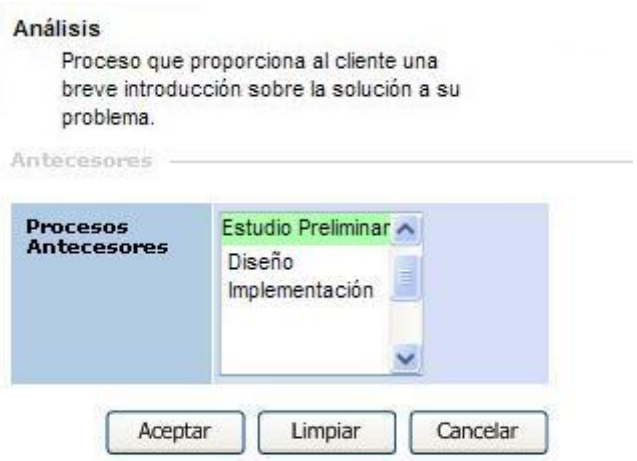
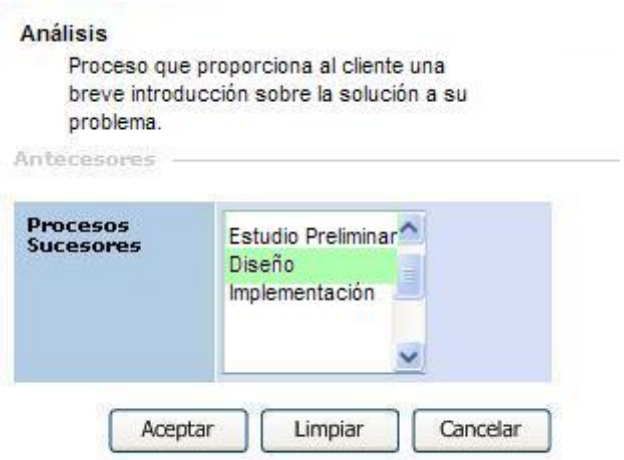


Ilustración 103. Establecer procesos sucesores para el proceso 2



De manera similar definimos los procesos antecesores y sucesores para los procesos 3 y 4: Diseño e Implementación, respectivamente de la siguiente forma.

c. Proceso 3: Diseño

Antecesores: Análisis

Sucesores: Implementación

d. Proceso 4: Implementación

Antecesores: Diseño

Sucesores: Ninguno

Asimismo, debe realizarse un proceso similar para establecer las tareas antecesoras y sucesoras (Ver ilustraciones 102 y 103).

3. Configuración de plantilla. La plantilla se define como el conjunto de características a almacenar de un proyecto de desarrollo de software, es decir, un estándar predefinido para un tipo específico de proyecto. A continuación se muestra un ejemplo detallando las características que se almacenarían para una aplicación de escritorio.

Ilustración 104. Configuración de plantilla para aplicación de escritorio

B. TEMAS DISPONIBLES EN SMARTFLOW

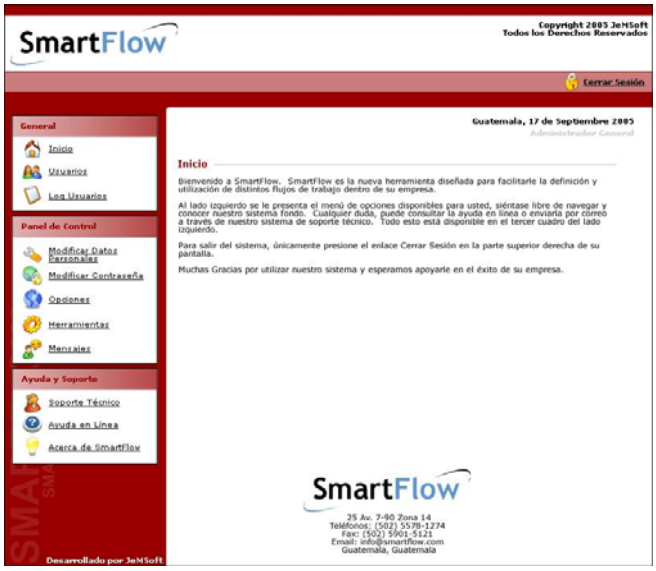
A continuación se presentan los temas disponibles en SmartFlow para modificar la apariencia del mismo. Los tres temas son: Zafiro, Rubí y Esmeralda.

1. Zafiro (Original)

Ilustración 95. Tema de SmartFlow: Zafiro.

2. Rubí

Ilustración 96. Tema de SmartFlow: Rubí.



3. Esmeralda

Ilustración 97. Tema de SmartFlow: Esmeralda

C. PRINCIPIOS DETRÁS DEL MANIFIESTO ÁGIL

A continuación se presentan los principios básicos que forman parte fundamental de la estructura del Manifiesto ágil, documento que especifica los lineamientos a seguir para el desarrollo de software ágil:

> «We follow these principles:
>
> Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
>
> Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
>
> Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
>
> Business people and developers must work together daily throughout the project.
>
> Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
>
> The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
>
> Working software is the primary measure of progress.
>
> Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
>
> Continuous attention to technical excellence and good design enhances agility.
>
> Simplicity--the art of maximizing the amount of work not done--is essential.
>
> The best architectures, requirements, and designs emerge from self-organizing teams.
>
> At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly».

XII. GLOSARIO

1. Abstracción

Es la propiedad que permite la representación de las características más importantes de un objeto, dejando por un lado los detalles.

2. API (Application Program Interface)

Se traduce como Interfaz de Programación de Aplicaciones y se refiere a un conjunto de rutinas de comunicación para la transferencia de datos y comandos entre un programa y otro. Determina como una aplicación debe acceder a un servicio.

3. ARPANet (Advanced Research Projects Agency Network)

Red precursora de la actual Internet, desarrollada en los 60s por el Departamento de la Defensa de Estados Unidos.

4. Autenticación

Proceso mediante el cual se verifica la identidad de un usuario en la red.

5. Automatización

Se refiere a la ejecución automática de cualquier tipo de tarea.

6. Backup

Se refiere a una o más copias de seguridad de información importante. Es la práctica de realizar archivos de respaldo, así como equipo y procedimientos a utilizar en caso de emergencias o fallos en un sistema computacional.

7. Cifrado

Proceso de codificación que se aplica a datos confidenciales a manera de esconder u ocultar su verdadero contenido. Esto se hace con el fin de que cuando los datos viajen a través de una red, estos no puedan ser leídos al ser interceptados por un tercero. Generalmente hace uso de operaciones matemáticas para proteger la información.

8. Cliente

Computadora, persona o programa que solicita y accede a los servicios ofrecidos por otra computadora o programa llamado servidor.

9. Contraseña (Password)

Cadena de caracteres alfanuméricos que permite a un usuario el acceso a un servicio. Debe ser personal y secreta.

10. Daemon

Se refiere a un proceso que ejecuta una función específica pero no interviene en la ejecución de otros procesos. Además, trabaja independientemente de la interacción del usuario, por lo que su funcionamiento es totalmente transparente a él.

11. Encapsulación

Es la propiedad de un modelo objeto que permite asegurar que su contenido está oculto al mundo exterior.

12. En línea (Online)

Se refiere a la condición de estar conectado a una red, por lo tanto, se tienen disponibles los servicios de la misma para su uso.

13. FDD (Feature-Driven Development)

Metodología de desarrollo ágil creada por Jeff Deluca con el fin de obtener un sistema funcional en tiempos relativamente cortos. La base de esta metodología es el concepto de *característica (feature)* el cual se define como una pieza de funcionalidad valorada por el cliente.

14. Fuente abierta (Open-source)

También se conoce este término como código abierto y se refiere a cualquier software cuyo código fuente está disponible para que cualquier persona lo modifique y lo adapte a sus necesidades.

15. Herramientas CASE

CASE es un acrónimo en inglés de Computer Aided Software Engineering y se refiere a los instrumentos utilizados para brindar soporte en todas las actividades relacionadas con el desarrollo de software.

16. Hipertexto

Documento que puede contener imágenes, textos, sonidos y elementos multimedia relacionados entre sí por medio de enlaces, de modo que al señalar uno, se pasa a otro.

17. Hosting

Servicio que permite ofrecen algunas empresas donde brindan espacio en su servidor para alojar páginas o sitios web de terceras personas.

18. HTML (HyperText Markup Language)

Lenguaje utilizado en la creación de documentos de hipertexto. Es el estándar para la definición de documentos usados en la red WWW.

19. HTTP (HyperText Transport Protocol)

Protocolo de transporte de hipertexto que se utiliza en la Red WWW. Define el estándar de cómo deben enviarse y formatearse los mensajes entre servidores web y navegadores.

20. HTTPD

HTTP daemon. Programa que corre de fondo en un servidor Web y espera peticiones de navegadores, usualmente son externas al servidor. Responde las peticiones utilizando el protocolo HTTP.

21. IEEE (Institute of Electrical and Electronic Engineering)

Organización profesional no lucrativa que se dedica a crear estándares en las áreas de electrónica e informática.

22. Interfaz

Frontera o límite que se establece entre 2 sistemas que interactúan y se comunican entre sí, haciendo uso de la misma para este fin. Existen 3 tipos de interfaces: de usuario, de software o de hardware.

23. Internet

La denominada “red de redes” creada de la unión de muchas redes TCP/IP a nivel internacional y cuyos antecedentes están en la ARPANet.

24. ISO (International Organization for Standarization)

Organización no gubernamental cuyo objetivo es elaborar normas internacionales de calidad en diferentes áreas incluyendo la informática.

25. Linux

Sistema operativo completo, robusto y construido sobre el esquema de UNIX. Es el más popular de los sistemas operativos disponibles libremente; su desarrollo fue iniciado por Linus Torvlads.

26. Login

Se refiere a la acción de conectarse a un sistema mediante una identificación de usuario y contraseña.

27. MD5

Algoritmo de cifrado creado en 1991 que se utiliza para crear firmas digitales. Se trata de una función irreversible (one-way hash function) que toma un mensaje y lo convierte en una cadena de dígitos no comprensibles.

28. Navegador (Browser)

Programa que se utiliza tanto para navegar o recorrer la red WWW así como para visualizar documentos de tipo multimedia o de texto. El protocolo más conocido que utiliza es el HTTP.

29. OMG (Object Management Group)

Consorcio creado en 1989 responsable de la creación, desarrollo y revisión de la arquitectura CORBA. No produce software, únicamente especificaciones.

30. One-way hash functions

Algoritmo que convierte texto en una cadena de dígitos con motivos de seguridad. Con estas funciones es casi imposible derivar el texto original del texto cifrado.

31. Plantilla (Template)

Archivo o documento de forma predeterminada que se utiliza como referencia o patrón para dar formato a otros objetos.

32. Pop-up

Ventana emergente o flotante que se muestra sobre la ventana actual en el navegador y es independiente de ésta.

33. Protocolo

Conjunto de reglas y normas que establecen procedimientos para llevar a cabo una acción. En informática, un protocolo generalmente define estándares para establecer comunicación entre 2 sistemas.

34. Script

Segmento de código escrito en un lenguaje interpretado, no compilado y ejecuta una función específica.

35. Servidor Web

Se refiere a una computadora o programa que recibe peticiones de clientes y les proporciona páginas Web a través del navegador. Cada servidor tiene una dirección de IP y un nombre de dominio, el cual interpreta el protocolo HTTP.

36. Sistema Operativo

Conjunto de programas que forman parte de la capa intermedia que existe entre un usuario y una computadora. Permite la comunicación en ambas vías, administra datos y controla la ejecución de programas dentro de una computadora.

37. TCP/IP (Transmission Control Protocol/Internet Protocol)

Sistema de protocolos en los que se basa una buena parte Internet. El primero se encarga de dividir la información en paquetes desde el origen, para luego recomponerla en el destino. El segundo la dirige adecuadamente a través de la red.

38. Testing

Es el proceso de verificación formal que se hace para probar cualquier posible error o fallo en una aplicación.

39. Texto plano (Plain-text)

Es el mensaje o archivo original antes de ser cifrado y después de ser descifrado.

40. URL

Sistema unificado de identificación de recursos en la red. Las direcciones se componen de protocolo, FQDN y dirección local del documento dentro del servidor.

41. WWW (World Wide Web)

Sistema de servidores de Internet que se utiliza para transmitir y recibir información basado en hipertexto, creado a principio de los 90s por Tim Berners Lee.

42. XP (Extreme Programming)

Metodología ágil creada en 1996 por Kent Beck y define como base de su estructura la importancia de las pruebas (testing), en donde cada desarrollador escribe pruebas y las ejecuta al terminar de escribir su código fuente.