UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Diseño e implementación de un sistema de administración de reportes de error (SIARE)

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A N

Alexander Ambriz Rivas

Carlos Alberto Ayala Rocha

DIRECTOR DE TESIS

M.C. Reynaldo Alanís Cantú

Ciudad Universitaria, Cd. Mx., 2003

Dedicatorias

A mis padres y abuela por el simple hecho de serlo. Por todas las oportunidades que me han brindado, fruto del amor, trabajo y sacrificio propio. Para ustedes mi completa admiración, saben que los amo y que son mi máximo orgullo. Sin ustedes esto no hubiese sido posible. ¡Este título es nuestro!

A todos aquellos que por alguna razón han sido parte directa o indirecta en la concepción, formación y continuo crecimiento en esta hermosa locura llamada vida.

A todos los que fueron, son y serán parte activa en la consolidación de mis metas.

Muy en especial a los imprescindibles que hacen que las cosas sucedan, siempre estando presentes de alguna u otra manera. Sin importar la distancia o el tiempo, si son días de bonanza o dificultad. ¡Siéntanse igualmente correspondidos!

Alejandro Cervantes (y familia), Enrique Basurto, José A. Espinosa, Luis Tejado, Mario Zarate, Raymundo Domínguez, Alexander Ambriz, Enrique Gaona, Omar Salinas e Ismael Flores.

Gracias a todos ustedes por trascender en mi vida.

Carlos Alberto Ayala Rocha

Dedico esta tesis a mi familia, amigos, compañeros de trabajo, profesores y a todos aquellos que de una u otra forma contribuyeron a alcanzar esta meta. Sin ustedes esto no hubiera sido posible.

Alexander Ambriz Rivas

Contenido

Figuras

Capítulo 1

Introducción

Los sistemas de software están presentes como un hecho en la vida moderna. En muchos casos el trabajo, bienestar, seguridad, entretenimiento e incluso la vida de muchas personas dependen de un sistema de software.

Los sistemas de software son creaciones complejas: realizan muchas funciones, están construidos para lograr muchos objetivos diferentes y, con frecuencia, conflictivos, comprenden muchos componentes, de los cuales muchos son en sí complejos y hechos a la medida, muchos participantes de disciplinas diferentes intervienen en el desarrollo de estos componentes, el proceso de desarrollo y el ciclo de vida del software a menudo abarcan muchos años y, por último, los sistemas complejos son difíciles de comprender por completo por una sola persona.

Los proyectos de desarrollo de software están sujetos a cambios constantes. Debido a que los requerimientos son complejos, necesitan ser actualizados cuando se descubren errores y cuando los desarrolladores tienen una mejor comprensión de la aplicación. Si el proyecto dura muchos años, la rotación de personal es alta y se requiere de entrenamiento constante. El tiempo entre los cambios tecnológicos con frecuencia es más corto que la duración del proyecto.

La complejidad del software se incrementa al mismo tiempo que cada vez más problemas críticos se resuelven con soluciones de software. Así, el desarrollo de software moderno tiene que enfrentar una complejidad creciente al tiempo que se encuentra bajo presión para crear soluciones baratas, rápidas (en cuanto al tiempo de desarrollo), y de buena calidad. La realización de soluciones de software complejas usualmente sólo puede llevarse a cabo por muchos desarrolladores agrupados en equipos. El uso de las técnicas de Ingeniería de Software es el camino al éxito en los proyectos complejos de desarrollo de software.

1.1 Ingeniería de Software

La Ingeniería de Software es la disciplina dentro de las Ciencias de la Computación que se encarga del estudio de las técnicas necesarias para la producción y mantenimiento de grandes sistemas de software. [KRU97]

La Ingeniería de Software es una actividad de modelado. Los ingenieros de software manejan la complejidad mediante el modelado, enfocándose sólo en los detalles relevantes e ignorando lo demás. En el curso del desarrollo, los ingenieros de software construyen muchos modelos diferentes del sistema y del dominio de aplicación.

La Ingeniería de Software es una actividad para la solución de problemas. Se usan los modelos para buscar una solución aceptable. Los ingenieros de software tienen recursos limitados, están restringidos por presupuestos y tiempos de entrega. Dada la falta de una teoría fundamental, con frecuencia tienen que apoyarse en métodos empíricos para evaluar los beneficios de alternativas diferentes.

La Ingeniería de Software es una actividad para la adquisición de conocimiento. En el modelado de los dominios de la aplicación y la solución, el ingeniero de software recopila datos, los organiza en información y los formaliza en conocimiento.

La Ingeniería de Software es una actividad dirigida por una fundamentación. Cuando se adquiere conocimiento y se toman decisiones acerca del sistema o sus dominios de aplicación, los ingenieros de software también necesitan captar el contexto en que se tomaron las decisiones y las razones que hay detrás de las mismas.

1.2 El proceso de ciclo de vida del software

El estándar IEEE para los procesos del ciclo de vida del software describe el conjunto de actividades y procesos obligatorios para el desarrollo y mantenimiento del software [IEEE Std. 1074-1995]. Su objetivo es establecer un marco común para el desarrollo de modelos de ciclo de vida.

Un proceso es un conjunto de actividades que se realiza para un propósito específico. El estándar IEEE lista un total de 17 procesos. Los procesos están agrupados en niveles de abstracción más elevados llamados grupos de procesos.

Grupo de procesos	Procesos
Modelado del ciclo de vida	Selección de un modelo de ciclo de vida
Administración del proyecto	Inicio del proyecto Supervisión y control del proyecto Administración de la calidad del software
Predesarrollo	Exploración de conceptos Asignación del sistema
Desarrollo	Requerimientos Diseño Implementación
Posdesarrollo	Instalación Operación y soporte Mantenimiento Retiro
Procesos integrales	Verificación y validación Administración de la configuración del software Desarrollo de la documentación Entrenamiento

1.2.1 El proyecto de desarrollo de software

Durante el modelado del ciclo de vida, el gerente del proyecto personaliza las actividades definidas en el IEEE 1074 para un proyecto específico (es decir, para una instancia del modelo del ciclo de vida).

Un proyecto es "un esfuerzo limitado por el tiempo, realizado para crear, corregir o dar mantenimiento a un producto o servicio nuevo o ya existente" [PMI96]; de tal forma que un proyecto puede ser el desarrollo de un sistema de software.

1.3 Calidad en desarrollo de software

La calidad está, en última instancia, definida por el cliente y representa qué tan cerca están los productos de satisfacer los requerimientos y expectativas del mismo.

El propósito de la administración de calidad es entender las expectativas del cliente en términos de calidad, y entonces poner en práctica un plan proactivo para satisfacer esas expectativas.

Dado que la calidad está definida por el cliente, podría parecer que es completamente subjetiva; de cualquier modo, hay mucho sobre ésta que puede hacerse objetivamente. Esto requiere , en primer lugar, romper el término genérico de calidad en áreas que definen las características de ésta; después, examinar cada una de las características individuales y determinar una o más métricas que pueden recolectarse para reflejar dichas características. Por ejemplo, una característica de calidad puede ser que la solución tenga la menor cantidad de errores. Esta característica puede medirse contando los errores de la solución.

La administración de calidad no es un evento, es un proceso y una forma de pensamiento. Un producto consistente de alta calidad no puede producirse a partir de un proceso malo. Existe la necesidad de un ciclo constante de medir la calidad, actualizar el proceso, medir otra vez, actualizar, y así consecutivamente.

Para hacer que la administración de calidad funcione es vital recolectar métricas; si éstas no se capturan será difícil mejorar los procesos a partir de una iniciativa de administración de calidad.

1.3.1 Métricas y mediciones

Las métricas de software pueden usarse para caracterizar cuantitativamente varios aspectos del proceso de software o los productos de software. Las métricas del proceso cuantifican atributos del proceso de software o el ambiente de desarrollo, mientras que las métricas del producto son medidas de los productos de software. Las métricas del producto permanecen independientes de los procesos usados para producir el producto. Como ejemplos de las métricas del proceso es posible citar productividad, calidad, métricas de recursos, tasa de inyección de defectos y eficiencia de remoción de defectos. Las métricas del producto incluyen tamaño, confiabilidad, calidad (que puede verse como métrica del proceso y del producto), complejidad del código y funcionalidad. [JAL02]

El uso de métricas necesariamente requiere de la realización de mediciones para obtener los datos. En cualquier programa de métricas es necesario entender claramente las metas del mismo, así como los modelos que se usan para hacer juicios basados en los datos.

En la práctica, algunas pocas métricas son suficientes para la mayor parte de las situaciones. La calendarización, el tamaño, el esfuerzo y los defectos son las mediciones básicas para los proyectos y forman un conjunto estable de métricas.

1.3.2 Procesos de administración de la calidad

El control de calidad se refiere a las actividades asociadas con la creación de productos durante el proyecto. Se usa para verificar que los productos sean de calidad aceptable, y que cumplan los criterios correctitud establecidos en el proceso de planeación de calidad. El control de calidad se lleva a cabo constantemente durante el proyecto y es responsabilidad de los miembros del equipo y del administrador del proyecto.

El aseguramiento de calidad no se refiere directamente a los productos específicos, se refiere a los procesos usados para crear los productos. En general, las actividades de aseguramiento de calidad se enfocan en los procesos usados para minimizar la introducción de defectos en el sistema.

1.4 Pruebas de software

La confiabilidad es una medida del éxito con que el comportamiento de un sistema se apega a su especificación. La confiabilidad del software es la probabilidad de que un sistema de software no causará la falla del sistema durante un tiempo determinado bajo condiciones especificadas. [IEEE Std. 982-1989]

Un defecto es una anomalía en el diseño o la codificación que puede causar un comportamiento inesperado en un componente del sistema. Un error es la manifestación de un defecto durante la ejecución del sistema. Una falla es una desviación entre la especificación de un componente y su comportamiento. Una falla es producida por uno o más errores. [BRU02g]

El objetivo de las pruebas es maximizar la cantidad de defectos descubiertos, lo cual permite que los desarrolladores los corrijan e incrementen la confiabilidad del sistema. Se definen las pruebas como el intento sistemático de localizar errores en forma planeada en el software implementado. De tal forma que las pruebas son un proceso de control de calidad.

1.5 Registro de errores y defectos

Una vez que se han ejecutado las pruebas y se han detectado las fallas, los desarrolladores cambian el componente para eliminar los defectos. Una corrección es un cambio a un componente con el propósito de reparar un defecto. [BRU02g]

Dado que los defectos tienen una relación directa con la calidad, el rastreo de éstos es crítico para asegurar la calidad. Un gran proyecto de desarrollo de software puede tener miles de defectos encontrados por distintas personas. Frecuentemente, la persona que corrige un defecto no es la misma que lo encuentra. Generalmente un líder de proyecto querrá remover la mayor parte de los defectos encontrados antes de la entrega final del producto; en tal escenario, reportar y cerrar los defectos no puede hacerse informalmente. El uso de mecanismos informales puede provocar que los defectos se encuentren y después se olviden, de modo que no se corrijan y se necesite un esfuerzo extra para encontrarlos de nuevo. [JAL02]

De ahí que, al menos, los errores y defectos deben registrarse y rastrearse hasta su corrección. Para este procedimiento es necesario contar con información como la manifestación del defecto, su localización, el nombre de la persona que encontró el error y de la persona que hizo la corrección. Una vez que cada error encontrado se registra (y posteriormente se cierra) es posible concentrar el análisis en qué tantos errores se han encontrado y qué porcentaje de los defectos sigue abierto. El seguimiento de los defectos está considerado dentro de las mejores prácticas para la administración de proyectos. [BRO96]

El solo hecho de registrar los errores y seguirlos hasta su cierre no es suficiente para apoyar otros análisis deseables. Para entender la eficiencia en la remoción de defectos es necesario conocer dónde se inyectan los mismos. Determinar la etapa de inyección requiere de un análisis del defecto. Con base en la naturaleza del mismo se puede hacer algunos juicios sobre dónde pudo haber sido introducido.

Es deseable entender, en ocasiones, la naturaleza de los defectos sin referencia a la etapa de inyección, sino en términos de sus categorías. Una clasificación puede ayudar a entender la distribución de los defectos a través de varias categorías; por esta razón es importante registrar el tipo de error. Finalmente, es importante registrar la severidad de un error, dado que si éste es severo, el líder de proyecto puede decidir qué debe corregirse cuanto antes. De igual manera, es posible decidir que los errores menores no necesitan corregirse para una entrega urgente.

Con esta información es posible llevar a cabo muchos análisis. Por ejemplo, se pueden dividir los defectos con respecto al tipo, severidad o localización; determinar la tasa de inyección semanal de

éstos; determinar su eficiencia de remoción y determinar su tasa de inyección en diferentes fases del proceso.

Estas actividades son llevadas a cabo por varios actores (líderes de proyecto, probadores y desarrolladores) quienes pueden encontrarse en localizaciones geográficas distintas; todos ellos deben ser capaces de acceder a la misma información. Esto sólo se puede lograr con el apoyo de herramientas de software.

1.6 Comunicación de proyectos

La Ingeniería de Software es una actividad de colaboración. El desarrollo de software reúne a participantes con diferentes conocimientos, como expertos en dominios, analistas, diseñadores, programadores, administradores y usuarios. Ningún participante puede comprender o controlar todos los aspectos del sistema que se esté desarrollando y, por lo tanto, todos los participantes dependen de los demás para su trabajo. Además, cualquier cambio al sistema o al dominio de la aplicación requiere que los participantes actualicen su comprensión del sistema. Estas dependencias hacen que sea crítico compartir información en forma precisa y a tiempo. [BRU02b]

Cuando se desarrolla un sistema, los desarrolladores se enfocan en la construcción de un sistema que se comporte de acuerdo con las especificaciones. Cuando los desarrolladores interactúan con otros participantes en el proyecto, se enfocan en comunicar información con precisión y en forma eficiente. La comunicación contribuye tanto al éxito del proyecto como un buen diseño o una implementación eficiente.

Un modo de comunicación se refiere a un tipo de intercambio de información que tiene objetivos y alcances definidos. Un modo de comunicación puede ser calendarizado, si consiste en acciones planeadas, o manejado por eventos, si sucede en forma aleatoria. El reporte de problemas es un ejemplo del modo de comunicación durante el cual un usuario reporta un error a los desarrolladores. Los reportes de problemas son manejados por eventos.

Un mecanismo de comunicación se refiere a una herramienta o procedimiento que puede usarse para transmitir y recibir información y apoyar a un modo de comunicación. Los mecanismos de comunicación son síncronos si requieren que el emisor y los receptores estén disponibles al mismo tiempo; si no es así, se dice que son asíncronos.

Sólo se pueden usar los mecanismos de comunicación asíncronos para apoyar los modos manejados por eventos.

Una vez que se ha reportado un problema y se han propuesto y evaluado soluciones, es necesario que se seleccione, comunique y ponga en práctica una solución única. La solución necesita documentarse y comunicarse a los participantes relevantes. La documentación de la resolución permite que los participantes hagan referencia a la decisión más adelante en el proyecto, en caso de malos entendidos. La comunicación efectiva de la decisión permite que los participantes permanezcan sincronizados.

Una base de problemas puede servir como mecanismo de comunicación para apoyar el seguimiento del problema y la solución del mismo.

Como mecanismos de comunicación asíncrona podemos citar el World Wide Web (WWW) y el correo electrónico.

Un navegador WWW despliega ante el usuario un documento que puede contener hipervínculos hacia otros documentos. Un localizador de recursos universal (URL por sus siglas en inglés) está asociado con cada vínculo y le describe al navegador la ubicación del documento de destino y su método de recuperación. La WWW ha tenido una popularidad creciente conforme se amplian los documentos de hipertexto para que contengan imágenes, animaciones y scripts ejecutables. En un proyecto de desarrollo de software, la WWW es ideal para la organización y para proporcionar acceso a la información generada.

El correo electrónico permite que un emisor transmita un mensaje de texto arbitrario a uno o más receptores. El correo electrónico en un proyecto de desarrollo se usa, con frecuencia, en vez de memorándum de oficina o llamadas telefónicas. Debido a su naturaleza asíncrona, el correo electrónico es ideal para apoyo de modos de comunicación manejados por evento, como peticiones de aclaraciones, peticiones de cambio y reporte de errores.

1.7 Enunciado del problema

El proceso de registro y rastreo de errores es una de las partes más importantes del proceso de desarrollo de software, puesto que los errores y defectos están estrechamente ligados con la calidad del producto de software.

En muchas ocasiones, los reportes de errores encontrados en un producto de software durante una revisión se manejan de forma manual, registrando en papel las observaciones de los probadores. Existen casos en que ni siquiera está definido un formato para reportar un error. Esta

falta de estandarización dificulta el seguimiento del reporte, se presta a malas interpretaciones y vuelve más compleja la administración del proyecto, pues es más difícil registrar qué correcciones ya se han realizado y cuáles están pendientes.

Para que el proceso de registro y rastreo de errores sea eficiente es necesario que se apoye en herramientas automatizadas. El uso de una herramienta tiene las siguientes ventajas:

- Obliga que se siga un proceso definido.
- Garantiza que se mantenga la consistencia de los datos respecto a los reportes de errores.
- Permite que la información esté disponible para todos los involucrados en el proyecto.
- Facilita el análisis de información para tomar decisiones tendientes a mejorar la calidad del producto de software, así como su proceso de desarrollo.

Esta tesis consiste en el diseño e implementación de un Sistema de Administración de Reportes de Error (SIARE) que permita a los probadores reportar los errores encontrados durante las pruebas, para que los líderes de proyecto puedan asignar el seguimiento de los errores a un desarrollador y, por último, permita a los probadores confirmar que se validó la corrección del error. Asimismo, generará estadísticas sobre los errores reportados para facilitar el análisis de la información a los líderes de proyecto, con el fin de mejorar la calidad del producto y del proceso de software.

Esta herramienta será una aplicación Web, con el fin de facilitar el acceso a la misma puesto que, como se ha mencionado, los actores involucrados en un proyecto de desarrollo de software pueden encontrarse geográficamente dispersos. Además, una aplicación Web puede ejecutarse aun en equipos con recursos limitados, pues sólo es indispensable contar con conexión a Internet o a una intranet. Este enfoque facilita también la instalación y administración de la aplicación, porque basta con instalarla en el servidor para que todos los usuarios tengan acceso a ella, y el uso de una base de datos centralizada simplifica las tareas de administración.

1.8 Metodología

Dado que esta tesis en sí es un proyecto de ingeniería de software, está estructurada de acuerdo a las actividades de desarrollo. El capítulo 1 presenta un panorama de la Ingeniería de Software y la definición del problema a resolver desde una perspectiva de alto nivel. El capítulo 2 presenta los objetivos del sistema y el modelo funcional del mismo, creado a partir de los requerimientos funcionales y no funcionales formalizados en forma de casos de uso. En el capítulo 3 se desarrolla el modelo de análisis a partir de los requerimientos mencionados en el capítulo 2. En este capítulo se especifican los modelos de objetos y el modelo dinámico del sistema. El capítulo 4 corresponde al diseño de alto nivel del sistema, donde se especifica la arquitectura de éste y la descomposición

en subsistemas. El capítulo 5 detalla el diseño del sistema a través de descripción del modelo de datos y la especificación de la interfaz para los componentes del sistema. Por último, el capítulo 6 presenta la evaluación del sistema y las conclusiones.

Como notación para describir los modelos del sistema se decidió usar UML. El lenguaje unificado de modelado o UML (Unified Modeling Language) es una notación que se produjo como resultado de la unificación de la técnica de modelado de objetos (OMT, por sus siglas en inglés [RUM91]), Booch [BOO94] e ingeniería de software orientada a objetos (OOSE, por sus siglas en inglés). [JAC92]

UML es un lenguaje de modelado, es decir, una notación (principalmente gráfica) de la que se valen los analistas para expresar los diseños.

El proceso de desarrollo elegido para el SIARE es el Proceso Unificado, un modelo de ciclo de vida propuesto por Booch, Jacobson y Rumbaugh. Este proceso especifica que un proyecto consta de varios ciclos y cada uno de ellos termina con la entrega de un producto al cliente. Cada ciclo consta de cuatro fases: concepción, elaboración, construcción y transición. La fase de concepción corresponde al capítulo 1, la fase de elaboración corresponde a los capítulos 2 y 3, y la construcción corresponde al capítulo 5.

Durante cada fase se generan distintos modelos de la aplicación. Todos los modelos están relacionados entre sí mediante dependencias de rastreabilidad. Un elemento del modelo puede rastrearse, al menos, hacia un elemento de un modelo asociado. Por ejemplo, todos los casos de uso tienen una relación de rastreabilidad con, al menos, una clase del modelo de análisis. La rastreabilidad nos permite comprender el efecto del cambio en un modelo sobre otros y, en particular, nos permite proporcionar rastreabilidad a los requerimientos. [BRU02h]

Capítulo 2

Requerimientos

El proceso de Ingeniería de Software se inicia con la recopilación de requerimientos. Durante esta fase el desarrollador recopila los requerimientos que el sistema a construir debe cumplir. Un requerimiento del sistema es una característica que éste debe tener o una restricción que debe cumplir. Durante la recopilación de requerimientos los objetivos principales del sistema a construir se formalizan al concentrarse en la interacción entre el usuario y el sistema. La información que se recopila a alto nivel es entendible por el cliente. Esta información consiste en requerimientos funcionales y no funcionales, los actores que interactúan con el nuevo sistema y las tareas principales del sistema formalizadas como casos de uso. [BRU02c]

Este capítulo presenta los datos que se obtuvieron durante la recopilación de requerimientos, organizados en las siguientes secciones:

- Objetivos
- Requerimientos funcionales
- Requerimientos no funcionales
- Modelo de casos de uso
- Reportes de casos de uso

Los objetivos describen las metas principales del SIARE. Los requerimientos funcionales y no funcionales presentan los requerimientos que el sistema debe cumplir a un alto nivel. El modelo de casos de uso y los reportes de éstos describen las principales interacciones entre el sistema y los usuarios.

2.1 Objetivos

El SIARE es una herramienta para controlar los reportes de errores encontrados en un sistema durante el proceso de pruebas. El objetivo principal es registrar y dar seguimiento a los errores hasta su cierre. El solo hecho de registrar y dar seguimiento a los errores hasta su cierre ayuda a aumentar la calidad de un producto de software. [JAL02]

Es necesario contar con un sistema automatizado que permita registrar los errores cuando son encontrados, asignar su seguimiento (por parte del líder de proyecto) a algún miembro del equipo para resolverlo, registrar que se ha validado la corrección y "cerrar" el error.

El SIARE permitirá a los probadores reportar en un formato estandarizado los errores que detecten; presentará al líder de proyecto, de forma centralizada, los reportes que hayan sido levantados y podrá asignarlos directamente a un desarrollador para que les dé seguimiento. De igual manera presentará al desarrollador los reportes que le han sido asignados para que registre la resolución de los mismos y, por último, permitirá a los probadores validar la resolución y cerrar el error.

El otro objetivo del SIARE es ayudar a la planeación y mejora de la calidad, por lo que generará estadísticas que permitan analizar los errores que se hayan registrado a lo largo del tiempo. Los parámetros a considerar en dichas estadísticas serán la etapa de inyección, severidad, tipo de error y tiempo de resolución del mismo; por lo tanto, estos parámetros se deberán considerar desde que se reporta un error, además de la descripción del mismo y su localización.

2.2 Requerimientos funcionales

Los requerimientos funcionales describen las interacciones entre el sistema y su ambiente, en forma independiente a su implementación. El ambiente incluye al usuario y cualquier otro sistema externo con el cual interactúe el sistema. [BRU02c]

Teniendo en mente los objetivos del SIARE, es necesario definir con mayor precisión la funcionalidad que el sistema provee. Para cubrir las necesidades de un sistema de control de errores, se han identificado los siguientes requerimientos funcionales:

- Solicitudes de inscripción al sistema
- Autorización de solicitudes
- Alta de probadores, desarrolladores y productos
- Reportar un error
- Asignación del reporte a un desarrollador para darle seguimiento
- Captura de la resolución de un error
- Cierre de un reporte
- Generación de estadísticas

2.2.1 Solicitudes de inscripción al sistema

El SIARE podrá ser usado por varios líderes de proyecto, por lo que el sistema solicitará los datos necesarios para identificar a un líder de proyecto antes de que tenga acceso a la funcionalidad del sistema.

2.2.2 Autorización de solicitudes

Una vez dada de alta una solicitud de inscripción al sistema, es necesario que un administrador valide las solicitudes y determine si se da acceso o no al sistema al líder de proyecto.

2.2.3 Alta de probadores, desarrolladores y productos

El líder de proyecto debe tener la posibilidad de dar de alta todos los elementos necesarios para que se puedan reportar los errores y darles seguimiento. Dichos elementos son probadores, desarrolladores y productos a probar.

2.2.4 Reportar un error

El sistema permitirá al probador reportar un error usando un formato estandarizado. El sistema automáticamente almacenará el reporte en una base de datos y notificará al líder de proyecto que se ha generado un nuevo reporte.

2.2.5 Asignación del reporte a un desarrollador para darle seguimiento

El sistema presentará al líder de proyecto de forma centralizada los reportes de error levantados y le permitirá asignarlos a un desarrollador para que les dé seguimiento. El sistema notificará automáticamente al desarrollador que se le ha asignado un reporte.

2.2.6 Captura de la resolución de un error

El sistema mostrará al desarrollador los reportes que le han sido asignados y le permitirá capturar la resolución de un reporte. Dicha resolución se almacenará automáticamente en la base de datos.

2.2.7 Cierre de un reporte

El sistema mostrará al probador los reportes que ha levantado con la resolución capturada por el desarrollador y le permitirá cerrar un error una vez que haya validado la corrección.

2.2.8 Generación de estadísticas

Para que el líder de proyecto pueda analizar los reportes levantados en el sistema, éste generará estadísticas con base en los parámetros de un reporte (tipo de error, severidad, etapa de inyección).

2.3 Requerimientos no funcionales

Los requerimientos no funcionales describen aspectos del sistema visibles por el usuario que no se relacionan en forma directa con el comportamiento funcional del sistema. [BRU02c]

2.3.1 Ambiente distribuido

El SIARE se ejecutará en un ambiente distribuido; esto es, la información de la base de datos y la lógica de la aplicación se mantendrán de forma centralizada en un servidor, pero los usuarios podrán acceder al sistema usando sus propios equipos. De esta manera muchos usuarios concurrentes podrán hacer uso del sistema. Los usuarios podrán acceder al sistema desde distintas localizaciones; la parte cliente de la aplicación deberá poder ejecutarse en equipos con pocos recursos (memoria, procesador).

2.3.2 Seguridad

Dado que el SIARE será utilizado por usuarios con diversos perfiles, es necesario garantizar que el acceso a la funcionalidad de éste y a los datos será controlado.

2.3.3 Economía

El SIARE deberá ser económico, desde el punto de vista del costo de las herramientas necesarias para su desarrollo y administración.

2.3.4 Portabilidad

El SIARE deberá ser portable, es decir, deberá poder instalarse en multitud de plataformas con un mínimo de cambios en el código.

2.4 Modelo de casos de uso

Un caso de uso es, en esencia, una interacción típica entre un usuario y un sistema de cómputo. El caso de uso contribuye con la especificación del sistema puesto que contiene una descripción textual del flujo de eventos iniciado por un actor que puede ser un usuario o una entidad externa al sistema. [FOW99a]

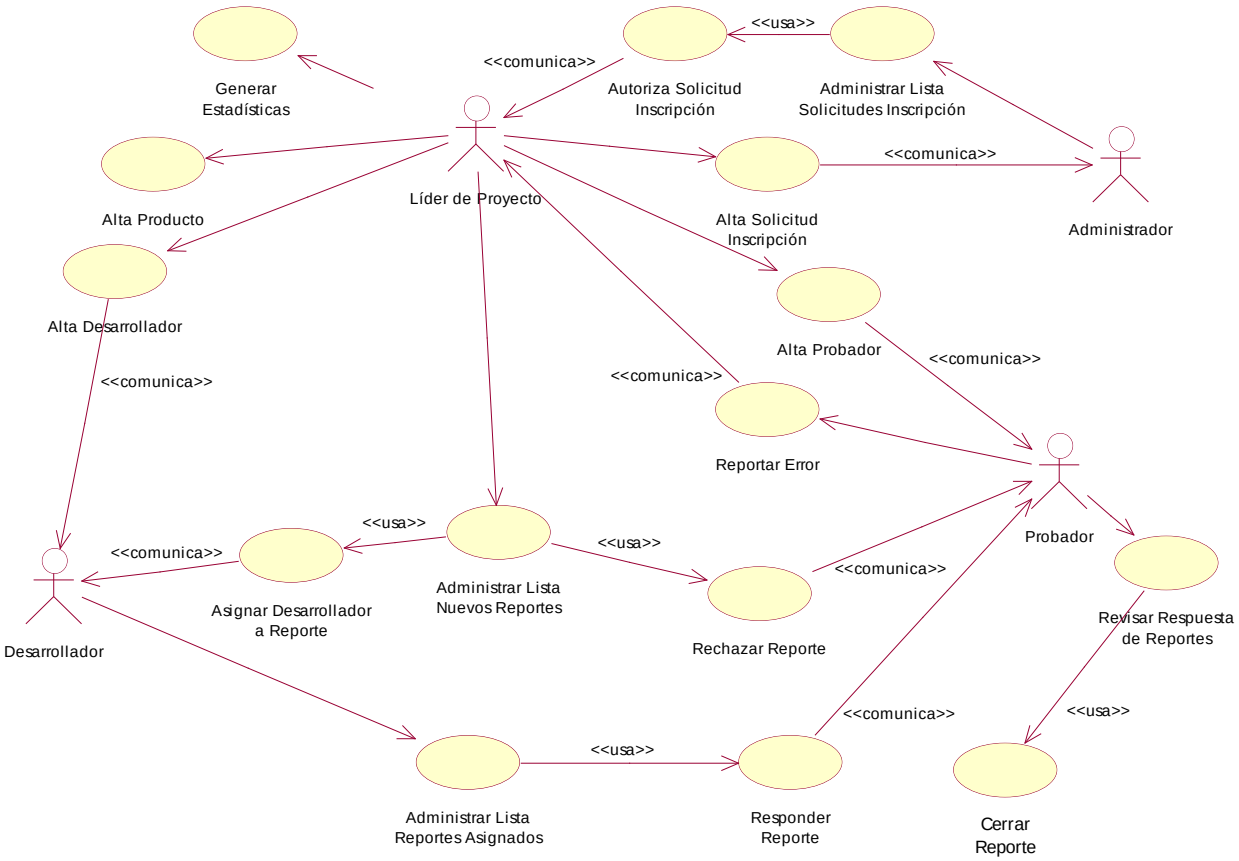


Figura 2-1 Modelo de casos de uso.

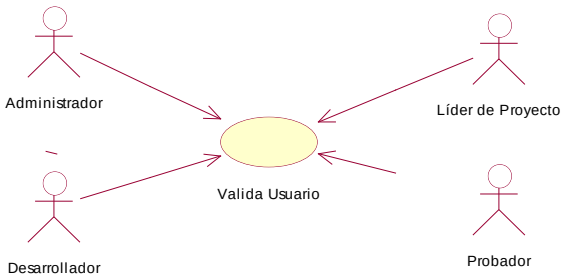


Figura 2-1 Modelo de casos de uso (continuación).

2.5 Reportes de casos de uso

2.5.1 Alta solicitud inscripción

Descripción:
Permite al líder de proyecto dar de alta su solicitud de inscripción al SIARE.
Precondiciones:
Ninguna.
Condición de éxito:
Que la solicitud de inscripción se guarde en la base de datos.
Condición de fracaso:
Que la solicitud de inscripción no se guarde en la base de datos.
Actores participantes:
Iniciado por líder de proyecto. Se comunica con administrador.
Flujo de eventos:
1. El líder de proyecto accede a la página principal del SIARE. 2. El líder de proyecto sigue una liga a la página de solicitud de alta en el sistema. 3. El sistema presenta un formulario para capturar nombre, datos de contacto (dirección, teléfono y correo electrónico), nombre de la organización donde labora, así como clave y password que usará el sistema para validar su acceso a éste. 4. El líder de proyecto captura los datos en el formulario y los envía mediante un botón. 5. El sistema da de alta los datos del formulario en la base de datos. 6. El sistema presenta una pantalla al líder de proyecto notificándole que su solicitud de inscripción se dio de alta exitosamente.

7. El sistema envía un correo electrónico al administrador notificándole que se recibió una nueva solicitud de inscripción.
Excepciones:
Si el sistema no logra comunicarse con la base de datos y dar de alta los datos de la solicitud de inscripción, envía un mensaje de error al líder de proyecto. Si ya existe un usuario del sistema con la misma clave, el sistema envía un mensaje de error al líder de proyecto.
Dependencias:
Ninguna.

2.5.2 Valida usuario

Descripción:
Valida la clave de acceso y password de los usuarios del SIARE para permitir su acceso al sistema, limitando la funcionalidad del mismo según el perfil de cada usuario.
Precondiciones:
Que el usuario (administrador, líder de proyecto, desarrollador o probador) esté dado de alta en la base de datos del sistema.
Condición de éxito:
Que el sistema permita al usuario registrado acceder a la funcionalidad definida según su perfil.
Condición de fracaso:
Que el sistema niegue el acceso a un usuario.
Actores participantes:
Administrador, líder de proyecto, desarrollador, probador.
Flujo de eventos:
Flujo normal 1:
1.1 El usuario accede a la página principal del SIARE. 1.2 El usuario sigue una liga a la página de acceso para usuarios registrados. 1.3 El sistema presenta un formulario con campos para capturar clave de usuario y password. 1.4 El usuario captura su clave de usuario y password; la información se envía mediante un botón. 1.5 El sistema compara la clave de usuario y password capturados contra las claves de usuarios y passwords almacenados en la base de datos. Encuentra una coincidencia y presenta al usuario un menú de opciones según su perfil.
Flujo normal 2:
2.1 El usuario accede a la página principal del SIARE. 2.2 El usuario sigue una liga a la página de acceso para usuarios registrados.

2.3 El sistema presenta un formulario con campos para capturar clave de usuario y password. 2.4 El usuario captura su clave de usuario y password; la información se envía mediante un botón. 2.5 El sistema compara la clave de usuario y password capturados contra las claves de usuarios y passwords almacenados en la base de datos. No encuentra una coincidencia y presenta al usuario una página con un mensaje notificándole la causa por la que se le niega el acceso al sistema y una liga para regresar a la página principal.
Excepciones:
Si el sistema no logra comunicarse con la base de datos para comparar la clave de usuario y el password capturados contra aquellos previamente registrados, presenta un mensaje de error al usuario.
Dependencias:
Ninguna.

2.5.3 Administrar lista solicitudes inscripción

Descripción:
Permite mostrar al administrador del sistema las solicitudes de registro dadas de alta por los líderes de proyecto, que no hayan sido autorizadas o rechazadas. Asimismo, permite seleccionar una solicitud y autorizarla o rechazarla, mediante el caso de uso *Autorizar solicitud inscripción*.
Precondiciones:
Que el administrador se haya validado como usuario del sistema (caso de uso *Valida usuario*).
Condición de éxito:
El sistema despliega las solicitudes de inscripción dadas de alta por los líderes de proyecto, permitiendo seleccionarlas para ver su detalle y autorizarlas o rechazarlas.
Condición de fracaso:
El sistema no despliega las solicitudes de inscripción registradas o no permite seleccionar una solicitud.
Actores participantes:
Administrador.
Flujo de eventos:
1. El administrador accede a una página Web que le presenta las solicitudes de inscripción al sistema dadas de alta por los líderes de proyecto. La lista contiene ligas que permiten ver el detalle de una solicitud y autorizarla mediante el caso de uso *Autoriza solicitud inscripción*.
Excepciones:
Si el sistema no encuentra solicitudes de inscripción despliega un mensaje informativo al administrador.

Si el sistema no logra comunicarse con la base de datos despliega un mensaje de error al administrador.
Dependencias:
Usa *Valida usuario* y *Autoriza solicitud inscripción*.

2.5.4 Autoriza solicitud inscripción

Descripción:
Permite al administrador del sistema aceptar o rechazar las solicitudes de inscripción dadas de alta por los líderes de proyecto.
Precondiciones:
Que la solicitud de inscripción esté dada de alta en la base de datos. Que se haya seleccionado la solicitud de la lista generada por el caso de uso *Administrar lista solicitudes inscripción*.
Condición de éxito:
Que se autorice la solicitud de inscripción y que se habilite al líder de proyecto como usuario del sistema.
Condición de fracaso:
Que no se autorice la solicitud de inscripción y no se habilite al líder de proyecto como usuario del sistema.
Actores participantes:
Iniciado por administrador. Se comunica con líder de proyecto.
Flujo de eventos:
Flujo normal 1:
1.1 El sistema presenta una página Web con los datos de la solicitud de inscripción dada de alta por el líder de proyecto. La página contiene un componente que permite autorizar la solicitud de inscripción, rechazarla o dejarla pendiente. 1.2 El administrador selecciona autorizar la solicitud y presiona un botón para enviar la información. 1.3 El sistema registra en la base de datos que la solicitud fue autorizada y que el líder de proyecto queda habilitado como usuario válido. 1.4 El sistema despliega un mensaje informando que se autorizó exitosamente la solicitud. 1.5 El sistema envía un correo electrónico al líder de proyecto informándole que su solicitud fue autorizada y que ya puede hacer uso del sistema.

Flujo normal 2:
2.1 El sistema presenta una página Web con los datos de la solicitud de inscripción dada de alta por el líder de proyecto. La página contiene un componente que permite autorizar la solicitud de inscripción, rechazarla o dejarla pendiente. 2.2 El administrador selecciona rechazar la solicitud y presiona un botón para enviar la información. 2.3 El sistema registra en la base de datos que la solicitud fue rechazada. 2.4 El sistema despliega un mensaje informando que se rechazó la solicitud. 2.5 El sistema envía un correo electrónico al líder de proyecto informándole que su solicitud fue rechazada.
Flujo normal 3:
3.1 El sistema presenta una página Web con los datos de la solicitud de inscripción dada de alta por el líder de proyecto. La página contiene un componente que permite autorizar la solicitud de inscripción, rechazarla o dejarla pendiente. 3.2 El administrador selecciona dejar pendiente la solicitud y presiona un botón para enviar la información. 3.3 El sistema mantiene el mismo estado de la solicitud (nueva). 3.4 El sistema despliega un mensaje informando que la solicitud quedó pendiente.
Excepciones:
Si el sistema no logra comunicarse con la base de datos despliega un mensaje de error para el administrador.
Dependencias:
Ninguna.

2.5.5 Alta probador

Descripción:
Permite al líder de proyecto dar de alta un nuevo probador que reporte errores encontrados en un producto.
Precondiciones:
Que el líder de proyecto se haya validado como usuario del sistema (caso de uso *Valida usuario*).
Condición de éxito:
Que se guarde el nuevo probador en la base de datos.
Condición de fracaso:
Que no se guarde el nuevo probador en la base de datos.

Actores participantes:
Iniciado por líder de proyecto. Se comunica con probador.
Flujo de eventos:
1. El líder de proyecto accede a una página Web que le presenta un formulario con campos para capturar el nombre del probador, sus datos de contacto (dirección, teléfono y correo electrónico), así como la clave de usuario y password que usará el probador para acceder al sistema. 2. El líder de proyecto llena los datos del formulario y los envía mediante un botón. 3. El sistema da de alta al nuevo probador en la base de datos. 4. El sistema despliega un mensaje informativo al líder de proyecto indicando que el probador se dio de alta exitosamente. 5. El sistema envía un correo electrónico al nuevo probador notificándole que fue dado de alta en el sistema.
Excepciones:
Si el sistema no logra comunicarse con la base de datos despliega un mensaje de error al líder de proyecto. Si ya existe un usuario con la misma clave que la capturada en el formulario, el sistema despliega un mensaje de error al líder de proyecto.
Dependencias:
Usa *Valida usuario.*

2.5.6 Alta desarrollador

Descripción:
Permite al líder de proyecto dar de alta un nuevo desarrollador que dé seguimiento a los reportes de error asociados a un producto.
Precondiciones:
Que el líder de proyecto se haya validado como usuario del sistema (caso de uso *Valida usuario*).
Condición de éxito:
Que se guarde el nuevo desarrollador en la base de datos.
Condición de fracaso:
Que no se guarde el nuevo desarrollador en la base de datos.
Actores participantes:
Iniciado por líder de proyecto.

Se comunica con desarrollador.
Flujo de eventos:
1. El líder de proyecto accede a una página Web que le presenta un formulario con campos para capturar el nombre del desarrollador, sus datos de contacto (dirección, teléfono y correo electrónico), así como la clave de usuario y password que usará el desarrollador para acceder al sistema. 2. El líder de proyecto llena los datos del formulario y los envía mediante un botón. 3. El sistema da de alta al nuevo desarrollador en la base de datos. 4. El sistema despliega un mensaje informativo al líder de proyecto indicando que se dio de alta exitosamente el desarrollador. 5. El sistema envía un correo electrónico al nuevo desarrollador notificándole que fue dado de alta en el sistema.
Excepciones:
Si el sistema no logra comunicarse con la base de datos despliega un mensaje de error al líder de proyecto. Si ya existe un usuario con la misma clave que la capturada en el formulario, el sistema despliega un mensaje de error al líder de proyecto.
Dependencias:
Usa *Valida usuario*.

2.5.7 Alta producto

Descripción:
Permite al líder de proyecto dar de alta un nuevo producto de software a probar.
Precondiciones:
Que el líder de proyecto se haya validado como usuario del sistema (caso de uso *Valida usuario*). Que el líder de proyecto haya dado de alta al menos un probador y un desarrollador que se asociarán al producto.
Condición de éxito:
Que el nuevo producto se guarde en la base de datos.
Condición de fracaso:
Que el nuevo producto no se guarde en la base de datos.
Actores participantes:
Líder de proyecto.

Flujo de eventos:
1. El líder de proyecto accede a una página Web que presenta un formulario con campos para capturar el nombre del producto de software a probar, una descripción del mismo y una clave para identificarlo. Asimismo, presenta una lista de los desarrolladores dados de alta de donde se seleccionan aquéllos que podrán dar seguimiento a los reportes de error asociados con el producto, y una lista de probadores dados de alta de donde se seleccionan aquéllos que podrán probar el producto). 2. El líder de proyecto captura los datos en el formulario y los envía mediante un botón. 3. El sistema da de alta el nuevo producto en la base de datos. 4. El sistema despliega un mensaje informativo al líder de proyecto indicando que se dio de alta exitosamente el producto.
Excepciones:
Si el sistema no logra comunicarse con la base de datos despliega un mensaje de error al líder de proyecto. Si no hay desarrolladores o probadores dados de alta que se puedan seleccionar para asociar al producto, se despliega un mensaje de error al líder de proyecto. Si ya existe un producto con la misma clave capturada en el formulario, el sistema envía un mensaje de error al líder de proyecto.
Dependencias:
Usa *Valida usuario*.

2.5.8 Reportar error

Descripción:
Permite al probador dar de alta un nuevo reporte de error asociado a un producto.
Precondiciones:
Que el probador se haya validado como usuario del sistema (caso de uso *Valida usuario*). Que el producto de software correspondiente al reporte esté dado de alta en la base de datos. Que en la base de datos estén dados de alta los catálogos de tipos de error y severidad de error.
Condición de éxito:
Que el reporte se guarde en la base de datos y se notifique al líder de proyecto.
Condición de fracaso:
Que el reporte no se guarde en la base de datos.
Actores participantes:
Iniciado por probador.

Se comunica con líder de proyecto.
Flujo de eventos:
1. El probador accede a una página Web que le presenta un formulario con una lista de los productos que puede probar, una lista con los tipos de error (documentación, sintaxis, paquete, asignación, interfaz, validaciones, datos, funciones, sistema y ambiente), una lista con los tipos de severidad (crítico, mayor, menor, cosmético) y campos para capturar el nombre del módulo del producto que se esté probando, la versión del producto y la descripción del error encontrado. 2. El probador captura los datos en el formulario y los envía mediante un botón. 3. El sistema da de alta el nuevo reporte en la base de datos, además de guardar los datos capturados mediante el formulario. El sistema guarda automáticamente la fecha y hora en que se da de alta el reporte y genera un número de reporte. 4. El sistema presenta un mensaje informativo al probador notificándole que se dio de alta el reporte. 5. El sistema envía un correo electrónico al líder de proyecto notificándole que se recibió un nuevo reporte.
Excepciones:
Si el sistema no logra comunicarse con la base de datos despliega un mensaje de error al probador.
Dependencias:
Usa *Valida usuario*.

2.5.9 Administrar lista nuevos reportes

Descripción:
Permite al líder de proyecto ver los nuevos reportes de error dados de alta por los probadores. Un reporte nuevo es aquél que no se haya rechazado o asignado a un desarrollador para que le dé seguimiento. Permite asignar un desarrollador al reporte o rechazar el reporte (casos de uso *Asignar desarrollador a reporte* y *Rechazar reporte*, respectivamente).
Precondiciones:
Que el líder de proyecto se haya validado como usuario del sistema (caso de uso *Valida usuario*). Que se haya dado de alta al menos un reporte de error en la base de datos, que no se haya rechazado o asignado a un desarrollador.
Condición de éxito:
El sistema despliega los reportes de error nuevos y permite seleccionar cualquier reporte para

rechazarlo (caso de uso *Rechazar reporte*) o asignarle un desarrollador (*Asignar desarrollador a reporte*).
Condición de fracaso:
El sistema no despliega los reportes de error nuevos o no permite seleccionar un reporte para rechazarlo o asignarle un desarrollador.
Actores participantes:
Líder de proyecto.
Flujo de eventos:
1. El líder de proyecto accede a una página Web que le presenta los reportes de error nuevos (aquéllos que no han sido rechazados o a los que se ha asignado un desarrollador para darle seguimiento). La lista contiene ligas que permiten ver el detalle de un reporte y rechazarlo (caso de uso *Rechazar reporte*) o asignarle un desarrollador para darle seguimiento (caso de uso *Asignar desarrollador a reporte*).
Excepciones:
Si el sistema no encuentra reportes de error despliega un mensaje informativo al líder de proyecto. Si el sistema no logra comunicarse con la base de datos despliega un mensaje de error al líder de proyecto.
Dependencias:
Usa *Valida usuario, Rechazar reporte* y *Asignar desarrollador a reporte*.

2.5.10 Rechazar reporte

Descripción:
Permite al líder de proyecto rechazar un reporte de error levantado por un probador. Al reporte rechazado ya no se le da seguimiento.
Precondiciones:
Que el reporte de error esté dado de alta en la base de datos, no se haya rechazado ni se haya asignado un desarrollador para darle seguimiento. Que se haya seleccionado el reporte de la lista generada por el caso de uso *Administrar lista nuevos reportes*.
Condición de éxito:
Que se marque en la base de datos que el reporte ha sido rechazado y se notifique vía correo electrónico al probador que se rechazó su reporte.

Condición de fracaso:
Que no se marque el reporte como rechazado o que no se notifique al probador el rechazo del reporte.
Actores participantes:
Iniciado por líder de proyecto. Se comunica con probador.
Flujo de eventos:
Flujo normal 1:
1.1 El sistema presenta una página Web que despliega los datos del reporte. También despliega un *"check box"* con la leyenda "¿Desea rechazar el reporte?" y un campo para capturar la causa por la que se rechaza el reporte. 1.2 El líder de proyecto selecciona el *"check box",* captura la causa de rechazo del reporte y envía dichos datos mediante un botón. 1.3 El sistema marca en la base de datos que el reporte fue rechazado. 1.4 El sistema envía un correo electrónico al probador notificándole que su reporte fue rechazado. 1.5 El sistema despliega una página informativa al líder de proyecto notificándole que el reporte se rechazó exitosamente.
Flujo normal 2:
2.1 El sistema presenta una página Web que despliega los datos del reporte. También despliega un "check box" con la leyenda "¿Desea rechazar el reporte?" y un campo para capturar la causa por la que se rechaza el reporte. 2.2 El líder de proyecto no selecciona el *"check box"* y presiona el botón. 2.3 El sistema regresa a la página con la lista de reportes de error nuevos.
Excepciones:
Si el sistema no logra comunicarse con la base de datos despliega un mensaje de error para el líder de proyecto.
Dependencias:
Ninguna.

2.5.11 Asignar desarrollador a reporte

Descripción:
Permite al líder de proyecto asignar un desarrollador para que dé seguimiento a un reporte de error.

Precondiciones:
Que el reporte de error esté dado de alta en la base de datos, no se haya rechazado ni se haya asignado un desarrollador para darle seguimiento. Que se haya seleccionado el reporte de la lista generada por el caso de uso *Administrar lista nuevos reportes.* Que esté, al menos, un desarrollador asignado al producto asociado al reporte.
Condición de éxito:
Que en la base de datos se registre qué desarrollador da seguimiento al reporte. Además, el sistema notifica por correo electrónico al desarrollador y al probador que la operación se llevó a cabo exitosamente.
Condición de fracaso:
Que no se registre en la base de datos el desarrollador que da seguimiento al reporte o no se notifique por correo electrónico al desarrollador y al probador.
Actores participantes:
Iniciado por líder de proyecto. Se comunica con probador y desarrollador.
Flujo de eventos:
1. El sistema presenta una página Web que despliega los datos del reporte. También presenta un formulario con una lista de los desarrolladores autorizados a dar seguimiento al reporte. 2. El líder de proyecto selecciona un desarrollador de la lista y envía ese dato mediante un botón. 3. El sistema almacena en la base de datos qué desarrollador da seguimiento al reporte. 4. El sistema envía un correo electrónico al desarrollador y al probador notificándoles la operación de asignación de recursos al reporte. 5. El sistema despliega una página informativa al líder de proyecto notificándole que la operación se llevó a cabo exitosamente.
Excepciones:
Si el sistema no logra comunicarse con la base de datos despliega un mensaje de error para el líder de proyecto.
Dependencias:
Ninguna.

2.5.12 Administrar lista reportes asignados

Descripción:
Muestra al desarrollador la lista de reportes de error que le han sido asignados y permite responderlos.

Precondiciones:
Que el desarrollador se haya validado como usuario del sistema (caso de uso *Valida usuario*). Que se haya asignado al menos un reporte al desarrollador.
Condición de éxito:
Que el sistema despliegue la lista de reportes asignados al desarrollador y permita seleccionar cualquier reporte mediante una liga para capturar la respuesta (caso de uso *Responder reporte*).
Condición de fracaso:
Que el sistema no despliegue los reportes asignados al desarrollador o no permita seleccionar un reporte para responderlo.
Actores participantes:
Desarrollador.
Flujo de eventos:
1. El desarrollador accede a una página Web que le presenta los reportes de error que le hayan sido asignados para responderlos. La lista no incluye aquellos reportes asignados al desarrollador que hayan sido cerrados por el probador. La lista contiene ligas que permiten ver el detalle de un reporte y responderlo (caso de uso *Responder reporte*).
Excepciones:
Si el sistema no encuentra reportes de error despliega un mensaje informativo al desarrollador. Si el sistema no logra comunicarse con la base de datos despliega un mensaje de error al desarrollador.
Dependencias:
Usa *Valida usuario* y *Responder reporte*.

2.5.13 Responder reporte

Descripción:
Permite al desarrollador responder a un reporte de error.
Precondiciones:
Que el desarrollador se haya validado como usuario del sistema (caso de uso *Valida usuario*). Que se haya seleccionado un reporte de error de la lista generada por el caso de uso *Administrar lista reportes asignados*. Que en la base de datos estén dados de alta el catálogo de etapas de inyección del error.
Condición de éxito:
Que se guarde en la base de datos la respuesta al reporte y se notifique al probador vía correo

electrónico que se ha recibido una respuesta.
Condición de fracaso:
Que no se guarde en la base de datos la respuesta al reporte o no se notifique al probador vía correo electrónico.
Actores participantes:
Iniciado por desarrollador. Se comunica con probador.
Flujo de eventos:
1. El sistema presenta una página Web que despliega los datos del reporte seleccionado. Presenta también un formulario con una lista de las etapas de inyección del error (recolección de requerimientos, revisión del plan de proyecto, diseño, revisión de código, pruebas unitarias, pruebas de integración, pruebas de aceptación) y con campos para capturar una descripción del diagnóstico y las acciones correctivas a tomar. Si previamente se ha capturado una respuesta al reporte, los campos del formulario despliegan los valores correspondientes. 2. El desarrollador captura los datos solicitados por el formulario y los envía mediante un botón. 3. El sistema da de alta la respuesta del reporte en la base de datos. 4. El sistema envía un correo electrónico al probador notificándole que se ha dado de alta o modificado la respuesta del reporte. 5. El sistema despliega una página con un mensaje para el desarrollador notificándole que la operación se llevó a cabo exitosamente.
Excepciones:
Si el sistema no logra comunicarse con la base de datos despliega un mensaje de error al desarrollador.
Dependencias:
Ninguna.

2.5.14 Revisar respuestas de reportes

Descripción:
Muestra al probador una lista de reportes que ha dado de alta, permite revisar la respuesta de un reporte y cerrarlo (mediante el caso de uso *Cerrar reporte*).
Precondiciones:
Que el probador se haya validado como usuario del sistema (caso de uso *Valida usuario*).
Condición de éxito:
Que se desplieguen los reportes dados de alta por el probador y que permita seleccionar un

reporte para ver su respuesta y cerrarlo.
Condición de fracaso:
Que no se desplieguen los reportes.
Actores participantes:
Iniciado por probador.
Flujo de eventos:
1. El probador accede a una página Web que le presenta los reportes de error que ha dado de alta y que se les ha asignado un desarrollador para darles seguimiento. La lista contiene ligas que permiten ver el detalle de un reporte con su respuesta. La liga también direcciona a una página que cierra el reporte mediante el caso de uso *Cerrar reporte*.
Excepciones:
Si el sistema no logra comunicarse con la base de datos despliega un mensaje de error al probador.
Dependencias:
Usa *Valida usuario* y *Cerrar reporte*.

2.5.15 Cerrar reporte

Descripción:
Permite al probador que dio de alta un reporte de error cerrarlo una vez que ha recibido una respuesta satisfactoria del desarrollador. Una vez cerrado, ya no se puede modificar la respuesta del reporte.
Precondiciones:
Que el probador se haya validado como usuario del sistema (caso de uso *Valida usuario*). Que haya seleccionado un reporte de la lista generada por el caso de uso *Revisar respuestas de reportes.*
Condición de éxito:
Que se marque en la base de datos que el reporte ha sido cerrado y se registre la fecha y hora en que ocurrió la operación.
Condición de fracaso:
Que no se cierre el reporte.
Actores participantes:
Iniciado por probador.

Flujo de eventos:
Flujo normal 1:
1.1 El sistema despliega una página Web con el detalle del reporte y su respuesta. Incluye además un formulario con un *"check box"* para indicar que se cierra el reporte y un botón para enviar los datos del formulario. 1.2 El probador selecciona el *"check box"* y envía los datos del formulario presionando el botón. 1.3 El sistema registra en la base de datos que el reporte está cerrado. Almacena la fecha y hora del cierre. 1.4 El sistema despliega una página Web notificando al probador que se cerró exitosamente el reporte.
Flujo normal 2:
2.1 El sistema despliega una página Web con el detalle del reporte y su respuesta. Incluye además un formulario con un *"check box"* para indicar que se cierra el reporte y un botón para enviar los datos del formulario. 2.2 El probador envía los datos del formulario sin haber seleccionado el *"check box".* 2.3 El sistema despliega una página Web notificando que no se cerró el reporte.
Excepciones:
Si el sistema no logra comunicarse con la base de datos despliega un mensaje de error al probador.
Dependencias:
Ninguna.

2.5.16 Generar estadísticas

Descripción:
Permite al líder de proyecto generar estadísticas para analizar el comportamiento de los reportes de error.
Precondiciones:
Que el líder de proyecto se haya validado como usuario del sistema (caso de uso *Valida usuario*).
Condición de éxito:
Que se genere el reporte estadístico solicitado por el líder de proyecto.
Condición de fracaso:
Que no se genere el reporte estadístico solicitado.
Actores participantes:
Iniciado por líder de proyecto.

Flujo de eventos:
1. El líder de proyecto accede a una página que le presenta la lista de estadísticas que el sistema puede generar. 2. El líder de proyecto selecciona un reporte estadístico de la lista. 3. El sistema solicita los parámetros necesarios para generar el reporte estadístico. 4. El líder de proyecto introduce los parámetros solicitados por el sistema. 5. El sistema genera el reporte estadístico.
Excepciones:
Si el sistema no logra comunicarse con la base de datos despliega un mensaje de error al líder de proyecto.
Dependencias:
Usa *Valida usuario*.

2.6 Especificación de estadísticas sobre los reportes de error

Dado que uno de los objetivos del SIARE es ayudar a mejorar el proceso de desarrollo de software mediante el análisis de los defectos y errores encontrados en un producto de software, es necesario que se consolide la información capturada sobre los reportes de errores en estadísticas que permitan determinar cuáles son los defectos y errores más comunes según cada uno de los parámetros que caracterizan a un reporte de error; de esta forma es posible poner en práctica acciones que prevengan esos defectos en el futuro.

A continuación se detallan las estadísticas que generará el SIARE.

2.6.1 Distribución por etapa de inyección

Muestra el número de defectos inyectados por cada etapa, así como el porcentaje del total de defectos inyectados por etapa. Puede generarse para todos los productos gestionados por el líder de proyecto, o para un producto en particular.

Ejemplo:

Distribución por etapa de inyección

Etapa	# de defectos inyectados	% del total
Recopilación de requerimientos	30	24
Análisis	20	16
Diseño de alto nivel	10	8
Diseño detallado	10	8
Codificación	40	32
Pruebas unitarias	10	8
Pruebas de integración	5	4
Pruebas de aceptación	0	0
Total	**125**	**100**

2.6.2 Distribución por severidad

Muestra el número de defectos clasificados según la severidad, así como el porcentaje del total por cada tipo de severidad. Puede generarse para todos los productos gestionados por el líder de proyecto, o para un producto en particular.

Ejemplo:

Distribución por severidad

Severidad	# de defectos	% del total
Crítico	25	20
Mayor	40	32
Menor	50	40
Cosmético	10	8
Total	**125**	**100**

2.6.3 Distribución por tipo

Muestra el número de defectos clasificados por tipo, así como el porcentaje del número total de defectos. Puede generarse para todos los productos gestionados por el líder de proyecto, o para un producto en particular.

Ejemplo:

Distribución por tipo

Tipo de defecto	# de defectos	% del total
Documentación	10	8
Sintaxis	20	16
Paquete (control de versiones)	5	4
Asignación	5	4
Interfaz	30	24
Validaciones	15	12
Datos	10	8
Funciones	20	16
Sistema	5	4
Ambiente	5	4
Total	**125**	**100**

2.6.4 Distribución por módulo

Muestra el número de defectos clasificados según el módulo del producto donde se encontraron. Requiere que se especifique el producto.

Ejemplo:

Distribución por módulo

Módulo	# de defectos	% del total
Facturación	25	20
Entregas	25	20
Caja	50	40
Atención a clientes	25	20
Total	**125**	**100**

2.6.5 Tipos de defecto por etapa de inyección

Muestra el número de defectos según su tipo agrupados por la etapa de su inyección. Puede generarse para un producto específico o para todos los productos gestionados por el líder de proyecto.

Ejemplo:

Tipos de defecto por etapa de inyección

Etapa: Recopilación de requerimientos

Tipo	**# de defectos**
Documentación	10
Validaciones	5
Total	15

Etapa: Análisis		
	Tipo	**# de defectos**
	Paquete	2
	Asignación	8
Total		10

Etapa: Diseño detallado		
	Tipo	**# de defectos**
	Interfaz	20
	Funciones	30
Total		50

Etapa: Codificación		
	Tipo	**# de defectos**
	Sintaxis	50
Total		50

TOTAL:	**125**

2.6.6 Tipos de defecto por severidad

Muestra el número de defectos según su tipo agrupados por severidad. Puede generarse para un producto específico o para todos los productos gestionados por el líder de proyecto.

Ejemplo:

Tipos de defecto por severidad

Severidad: Crítico		
	Tipo	**# de defectos**
	Sistema	5
Total		5

Severidad: Mayor		
	Tipo	**# de defectos**
	Ambiente	5
	Datos	10
Total		15

Severidad: Menor		
	Tipo	**# de defectos**
	Documentación	5
Total		5

TOTAL:	**25**

2.6.7 Estatus de los reportes por producto

Muestra la distribución de los reportes de error según su estatus para un producto especificado.

Ejemplo:

Estatus de los reportes por producto

```
Errores reportados: 20
Reportes nuevos: 5
Reportes rechazados: 2
Reportes asignados para seguimiento: 7
Reportes cerrados: 6
```

2.6.8 Distribución de reportes por tiempo para su cierre

Muestra la distribución de los reportes según el número de días transcurridos desde que se da de alta el reporte hasta que se cierra. Puede generarse para un producto específico o para todos los productos gestionados por el líder de proyecto.

Ejemplo:

Distribución de reportes por tiempo para su cierre

# de días	# de reportes cerrados
1	0
2	5
3	10
4	5
5	8
...	
Total	**30**

2.6.9 Reportes abiertos y cerrados por periodo

Muestra la distribución de reportes según su estatus (nuevo o cerrado) a lo largo de un periodo de tiempo especificado. Para cada día del periodo despliega cuántos reportes se han abierto y cuántos se han cerrado. Se genera para un producto específico.

Ejemplo:

Reportes abiertos y cerrados del 25/12/2002 al 27/12/2002

Día	# Reportes abiertos	# Reportes cerrados
25/12/2002	3	2
26/12/2002	1	3
27/12/2002	0	3
Total	**16**	**11**

Capítulo 3

Análisis

Después de la recopilación de requerimientos, el desarrollador analiza la información creada para asegurar que la descripción del sistema es correcta, completa, consistente y verificable. Esta fase se llama análisis. La principal tarea del análisis es examinar y especializar la información recolectada. Durante el análisis cualquier error originado en la recopilación de requerimientos debe encontrarse y corregirse. El análisis debe resultar en una clara descripción del sistema llamada modelo de análisis.

El modelo de análisis está compuesto por tres modelos individuales: el modelo funcional, representado por casos de uso; el modelo de objetos de análisis, representado por diagramas de clases; y el modelo dinámico, representado por gráficas de estado y diagramas de secuencia. [BRU02d]

3.1 Modelo de objetos de análisis

El modelo de objetos de análisis consiste en objetos de entidad, frontera y control. [JAC99] Los objetos de entidad representan la información persistente rastreada por el sistema. Los objetos de frontera representan la interacción entre los actores y el sistema. Los objetos de control representan las tareas realizadas por el usuario y soportadas por el sistema.

Modelar el sistema con objetos de entidad, frontera y control tiene varias ventajas: proporciona a los desarrolladores heurística simple para distinguir conceptos diferentes pero relacionados; el enfoque de tres tipos de objetos da como resultado objetos más pequeños y especializados; conduce a modelos que son más adaptables al cambio, es más probable que cambie la interfaz del sistema (representada por los objetos de frontera) que la funcionalidad básica (representada por los objetos de entidad y control). [BRU02d]

3.1.1 Identificación de objetos de entidad

Los objetos de entidad representan la información persistente rastreada por el sistema. Los datos persistentes sobreviven a una sola ejecución del sistema.

Usuario	Describe las características comunes a todos los usuarios del sistema. Contiene las propiedades de clave de usuario, password, nombre, dirección, teléfono y correo electrónico.
Líder de proyecto	Es un usuario que administra los reportes de error asociados a un producto de software. El líder de proyecto da de alta nuevos productos, probadores y desarrolladores. Asimismo, revisa los reportes de error levantados por los probadores y asigna a un desarrollador para darles seguimiento, o bien, rechaza los reportes. Contiene una propiedad correspondiente a la razón social de la organización donde trabaja.
Administrador	Es el usuario responsable de revisar las solicitudes de inscripción de nuevos líderes de proyecto y autorizar o rechazar las solicitudes.
Solicitud de inscripción	Es una solicitud con todos los datos necesarios para dar de alta un nuevo líder de proyecto en el sistema. Contiene una propiedad para indicar el estatus de la solicitud. Al guardar una nueva solicitud de inscripción, el sistema genera automáticamente un número de solicitud.
Probador	Es aquel usuario que prueba un producto y levanta reportes de error según los resultados de sus pruebas. Una vez que ha recibido una respuesta satisfactoria a un reporte de error lo cierra para que no se modifique la respuesta.
Desarrollador	Es aquel usuario que da seguimiento a un reporte de error y captura una respuesta para el mismo.
Producto	Es un producto de software que será validado por los probadores. Según los resultados de la prueba, los probadores levantarán reportes de error. Contiene propiedades correspondientes al nombre del producto, una descripción del mismo y una clave para identificarlo. Asimismo, contiene una propiedad correspondiente a la lista de desarrolladores que podrán dar seguimiento a los reportes de error asociados al producto, y una propiedad correspondiente a la lista de probadores que podrán validar el producto.
Reporte de error	Es un reporte relativo a un error encontrado en un producto validado por un probador. Contiene propiedades correspondientes al tipo de error, al tipo de severidad, una descripción del módulo del producto donde se encontró el error, la versión del producto y una descripción detallada del error. Al almacenarse el reporte de error, el sistema

	genera automáticamente un número de reporte y guarda la fecha y hora de alta del reporte. Si el líder de proyecto rechaza un reporte, la causa del rechazo se almacena en la propiedad de descripción *causa de rechazo*. El reporte contiene también las propiedades para almacenar la respuesta del reporte, capturada por el desarrollador. La respuesta consiste en la etapa de inyección del error y en una descripción del diagnóstico y las acciones correctivas a tomar. El sistema guarda la fecha y hora correspondiente a todo cambio de estatus del reporte.
Tipo severidad	Contiene las propiedades que describen el tipo de severidad de un error. Consta de una clave y una descripción del tipo de severidad. Los valores que puede tomar son *crítico, mayor, menor* y *cosmético*.
Tipo error	Contiene las propiedades que describen el tipo de error. Consta de una clave y una descripción del tipo de error. Los valores que puede tomar son *documentación, sintaxis, paquete (control de versiones), asignación, interfaz, validaciones, datos, funciones, sistema* y *ambiente*.
Etapa inyección error	Contiene las propiedades que describen la etapa de inyección de un error. Consta de una clave y una descripción del tipo de error. Los valores que puede tomar son *recopilación de requerimientos, análisis, diseño de alto nivel, diseño detallado, codificación, pruebas unitarias, pruebas de integración* y *pruebas de aceptación*.
Estatus solicitud	Contiene las propiedades que describen el estatus de una solicitud. Consta de una clave y una descripción del estatus de la solicitud. Los valores que puede tomar son *nueva, autorizada* y *rechazada.*
Estatus reporte	Contiene las propiedades que describen el estatus de un reporte. Consta de una clave y una descripción del estatus. Los valores que puede tomar son *nuevo, rechazado, asignado para seguimiento* y *cerrado.*
Lista solicitudes nuevas	Busca en la base de datos aquellas solicitudes de inscripción con estatus de *nueva*. Crea una lista de objetos *Solicitud de inscripción*.
Lista reportes nuevos	Busca en la base de datos los reportes de error con estatus de *nuevo*, correspondientes a los productos dados de alta por el líder de proyecto que invoca la búsqueda. Crea una lista de objetos *Reporte de error*.

Lista reportes levantados	Busca en la base de datos los reportes de error dados de alta por el probador que invoca la búsqueda. Crea una lista de objetos *Reporte de error.*
Lista reportes asignados	Busca en la base de datos los reportes de error (con estatus *asignado para seguimiento*) asignados al desarrollador que invoca la búsqueda. Crea una lista de objetos *Reporte de error*.
Distribución etapa inyección	Obtiene los datos necesarios para generar el reporte de distribución de defectos por etapa de inyección (sección 2.6.1).
Distribución severidad	Obtiene los datos necesarios para generar el reporte de distribución de defectos por severidad (sección 2.6.2).
Distribución tipo	Obtiene los datos necesarios para generar el reporte de distribución de defectos por tipo (sección 2.6.3).
Distribución módulo	Obtiene los datos necesarios para generar el reporte de distribución de defectos por módulo (sección 2.6.4).
Tipo x etapa	Obtiene los datos necesarios para generar el reporte de tipos de defecto por etapa de inyección (sección 2.6.5).
Tipo x severidad	Obtiene los datos necesarios para generar el reporte de tipos de defecto por severidad (sección 2.6.6).
Estatus x producto	Obtiene los datos necesarios para generar el reporte de estatus de los reportes por producto (sección 2.6.7).
Tiempo cierre	Obtiene los datos necesarios para generar el reporte de distribución de reportes por tiempo para su cierre (sección 2.6.8).
Estatus x periodo	Obtiene los datos necesarios para generar el reporte de reportes abiertos y cerrados por periodo (sección 2.6.9).

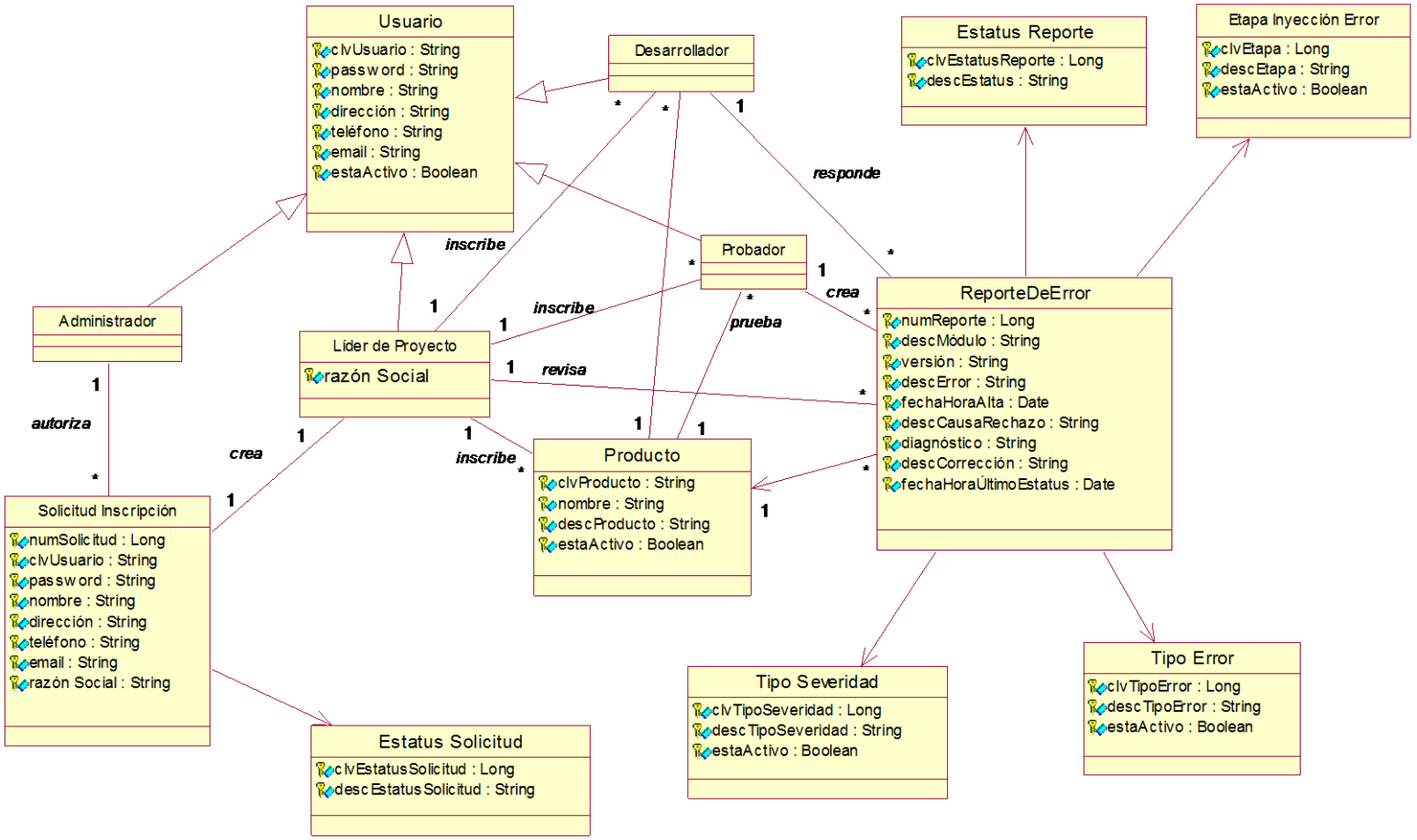


Figura 3-1 Diagrama de clases de entidad para el SIARE.

3.1.2 Identificación de objetos de frontera

Los objetos de frontera representan la interfaz del sistema con los actores. En cada caso de uso, cada actor interactúa con, al menos un objeto de frontera. Éste recopila la información del actor y la traduce hacia una forma neutral de interfaz que puede ser usada por los objetos de entidad y también por los de control. Los objetos de frontera modelan la interfaz de usuario a un nivel burdo.

WebAltaSolicitudInscripcion: Formulario utilizado por el líder de proyecto para dar de alta su solicitud de inscripción.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Clave de usuario	*Caja de captura de texto*	*Alfanumérico*	*20*
**Password*	*Caja de captura de texto*	*Alfanumérico*	*8*
**Confirmación password*	*Caja de captura de texto*	*Alfanumérico*	*8*
Nombre	*Caja de captura de texto*	*Alfanumérico*	*80*
Dirección	*Caja de captura de texto*	*Alfanumérico*	*120*
Teléfono	*Caja de captura de texto*	*Alfanumérico*	*20*
Email	*Caja de captura de texto*	*Alfanumérico*	*40*
Razón social	*Caja de captura de texto*	*Alfanumérico*	*80*

* En estos campos, al capturar la información deben desplegarse asteriscos para no mostrar el password que se esté capturando.

WebValidaUsuario: Formulario utilizado para validarse como usuario del sistema y tener acceso a la funcionalidad del mismo según el perfil del usuario.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Clave de usuario	*Caja de captura de texto*	*Alfanumérico*	*20*
**Password*	*Caja de captura de texto*	*Alfanumérico*	*8*

* En este campo, al capturar la información deben desplegarse asteriscos para no mostrar el password que se esté capturando.

WebMenuAdministrador: Página Web que permite acceder a la funcionalidad definida para el administrador.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Administración de solicitudes[1]	*Liga (link)*	*Ninguna*	*-*
Cerrar sesión[2]	*Liga (link)*	*Ninguna*	*-*

[1] Llama a WebAdministraciónSolicitudes.

[2] Regresa a página principal del sistema.

WebAdministracionSolicitudes: Despliega una lista de solicitudes de inscripción nuevas. Permite seleccionar una solicitud para autorizarla o rechazarla. La lista está formada por los siguientes campos:

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Número de solicitud[1]	*Liga (link)*	*Ninguna*	-
Nombre líder de proyecto	*Texto*	*Ninguna*	-
Razón social del líder de proyecto	*Texto*	*Ninguna*	-

[1] Llama a WebAutorizaSolicitud.

WebAutorizaSolicitud: Despliega el detalle de la solicitud de inscripción y permite autorizarla, rechazarla o dejarla pendiente .

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Número de solicitud	*Texto*	*Ninguna*	-
Clave de usuario	*Texto*	*Ninguna*	-
Nombre	*Texto*	*Ninguna*	-
Dirección	*Texto*	*Ninguna*	-
Teléfono	*Texto*	*Ninguna*	-
Email	*Texto*	*Ninguna*	-
Razón social	*Texto*	*Ninguna*	-
Cambia estatus solicitud[1]	*Lista de opciones (radio button)*	*Ninguna*	-

[1]Presenta las opciones para autorizar la solicitud, rechazarla o dejarla pendiente (sigue con estatus "nuevo").

WebMenuLiderProyecto: Página Web que permite acceder a la funcionalidad definida para el líder de proyecto.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Alta de desarrolladores[1]	*Liga (link)*	*Ninguna*	-
Alta de probadores[2]	*Liga (link)*	*Ninguna*	-
Alta de productos[3]	*Liga (link)*	*Ninguna*	-
Administración de reportes[4]	*Liga (link)*	*Ninguna*	-
Generación de estadísticas[5]	*Liga (link)*	*Ninguna*	-
Cerrar sesión[6]	*Liga (link)*	*Ninguna*	-

[1]Llama a WebAltaDesarrollador.

[2]Llama a WebAltaProbador.

[3]Llama a WebAltaProducto.

[4]Llama a WebAdministracionReportes.

[5]Llama a WebGenerarEstadisticas.

[6]Regresa a página principal del sistema.

WebAltaDesarrollador: Formulario que permite capturar los datos de un nuevo desarrollador.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Clave de usuario	*Caja de captura de texto*	*Alfanumérico*	*20*
**Password*	*Caja de captura de texto*	*Alfanumérico*	*8*

**Confirmación password*	*Caja de captura de texto*	*Alfanumérico*	*8*
Nombre	*Caja de captura de texto*	*Alfanumérico*	*80*
Dirección	*Caja de captura de texto*	*Alfanumérico*	*120*
Teléfono	*Caja de captura de texto*	*Alfanumérico*	*20*
Email	*Caja de captura de texto*	*Alfanumérico*	*40*
* En estos campos, al capturar la información deben desplegarse asteriscos para no mostrar el password que se esté capturando.			

WebAltaProbador: Formulario que permite capturar los datos de un nuevo probador.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Clave de usuario	*Caja de captura de texto*	*Alfanumérico*	*20*
**Password*	*Caja de captura de texto*	*Alfanumérico*	*8*
**Confirmación password*	*Caja de captura de texto*	*Alfanumérico*	*8*
Nombre	*Caja de captura de texto*	*Alfanumérico*	*80*
Dirección	*Caja de captura de texto*	*Alfanumérico*	*120*
Teléfono	*Caja de captura de texto*	*Alfanumérico*	*20*
Email	*Caja de captura de texto*	*Alfanumérico*	*40*
* En estos campos, al capturar la información deben desplegarse asteriscos para no mostrar el password que se esté capturando.			

WebAltaProducto: Formulario que permite capturar los datos de un nuevo producto.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Clave de producto	*Caja de captura de texto*	*Alfanumérico*	*10*
Nombre	*Caja de captura de texto*	*Alfanumérico*	*40*
Descripción	*Caja de captura de texto*	*Alfanumérico*	*120*
Lista desarrolladores	*Lista desplegable (combo box)*	*Ninguna*	*-*
Lista probadores	*Lista desplegable (combo box)*	*Ninguna*	*-*

WebAdministracionReportes: Despliega una lista de los reportes de error nuevos (aquellos que no han sido asignados a un desarrollador o se han rechazado). La lista consta de los siguientes campos:

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Número de reporte[1]	*Liga (link)*	*Ninguna*	*-*
Nombre de producto	*Texto*	*Ninguna*	*-*
Módulo	*Texto*	*Ninguna*	*-*
Versión	*Texto*	*Ninguna*	*-*
Tipo severidad	*Texto*	*Ninguna*	*-*
Fecha de alta	*Texto*	*Ninguna*	*-*
[1]Llama a WebDetalleReporte.			

WebDetalleReporte: Despliega la información del reporte de error.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Número de reporte	*Texto*	*Ninguna*	*-*

Nombre del probador	*Texto*	*Ninguna*	*-*
Nombre del producto	*Texto*	*Ninguna*	*-*
Módulo	*Texto*	*Ninguna*	*-*
Versión	*Texto*	*Ninguna*	*-*
Fecha de alta	*Texto*	*Ninguna*	*-*
Descripción del error	*Texto*	*Ninguna*	*-*
Tipo severidad	*Texto*	*Ninguna*	*-*
Tipo error	*Texto*	*Ninguna*	*-*
Asignar desarrollador a reporte[1]	*Liga (link)*	*Ninguna*	*-*
Rechazar reporte[2]	*Liga (link)*	*Ninguna*	*-*

[1]Llama a WebAsignarDesarrollador.

[2]Llama a WebRechazarReporte.

WebAsignarDesarrollador: Permite asignar al desarrollador que va a dar seguimiento al reporte.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Número de reporte	*Texto*	*Ninguna*	*-*
Lista de desarrolladores	*Lista desplegable (combo box)*	*Ninguna*	*-*

WebRechazarReporte: Permite rechazar un reporte.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Número de reporte	*Texto*	*Ninguna*	*-*
Causa de rechazo	*Caja de captura de texto*	*Alfanumérico*	*255*

WebGenerarEstadisticas: Permite al líder de proyecto generar las estadísticas especificadas en la sección 2.6. Presenta una lista de las estadísticas que se pueden generar, junto con componentes para capturar los parámetros necesarios para generarla (producto y periodo).

WebMenuProbador: Página Web que permite acceder a la funcionalidad definida para probador.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Reportar error[1]	*Liga (link)*	*Ninguna*	*-*
Revisar respuestas[2]	*Liga (link)*	*Ninguna*	*-*
Cerrar sesión[3]	*Liga (link)*	*Ninguna*	*-*

[1]Llama a WebReportarError.

[2]Llama a WebRevisarRespuestas.

[3]Regresa a página principal del sistema.

WebReportarError: Página Web que permite levantar un nuevo reporte de error.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Producto	*Lista desplegable (combo box)*	*Ninguna*	*-*
Módulo	*Caja de captura de texto*	*Alfanumérico*	*40*

Versión	*Caja de captura de texto*	*Alfanumérico*	*15*
Tipo de error	*Lista desplegable (combo box)*	*Ninguna*	*-*
Tipo de severidad	*Lista desplegable (combo box)*	*Ninguna*	*-*
Descripción error	*Caja de captura de texto*	*Alfanumérico*	*255*

WebRevisarRespuestas: Presenta al probador una lista con los reportes que ha dado de alta y no ha cerrado. La lista está formada por las siguientes columnas:

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Número de reporte[1]	*Liga (link)*	*Ninguna*	*-*
Nombre del producto	*Texto*	*Ninguna*	*-*
Módulo	*Texto*	*Ninguna*	*-*
Versión	*Texto*	*Ninguna*	*-*
Tipo severidad	*Texto*	*Ninguna*	*-*
Estatus	*Texto*	*Ninguna*	*-*
Fecha último estatus	*Texto*	*Ninguna*	*-*

[1]Llama a WebCerrarReporte.

WebCerrarReporte: Presenta al probador el detalle del reporte junto con su respuesta y le permite cerrarlo para que la respuesta ya no sufra modificaciones.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Número de reporte	*Texto*	*Ninguna*	*-*
Nombre del producto	*Texto*	*Ninguna*	*-*
Módulo	*Texto*	*Ninguna*	*-*
Versión	*Texto*	*Ninguna*	*-*
Fecha de alta	*Texto*	*Ninguna*	*-*
Descripción error	*Texto*	*Ninguna*	*-*
Tipo severidad	*Texto*	*Ninguna*	*-*
Tipo error	*Texto*	*Ninguna*	*-*
Estatus	*Texto*	*Ninguna*	*-*
Fecha último estatus	*Texto*	*Ninguna*	*-*
Diagnóstico[1]	*Texto*	*Ninguna*	*-*
Etapa de inyección[1]	*Texto*	*Ninguna*	*-*
Descripción corrección[1]	*Texto*	*Ninguna*	*-*
Cerrar reporte[2]	*"Check box"*	*Ninguna*	*-*

[1] Estos campos sólo se despliegan si el desarrollador ha capturado una respuesta.

[2] Al seleccionarlo permite cerrar el reporte.

WebMenuDesarrollador: Página Web que permite acceder a la funcionalidad definida para el desarrollador.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Administrar reportes asignados[1]	*Liga (link)*	*Ninguna*	*-*
Cerrar sesión[2]	*Liga (link)*	*Ninguna*	*-*

	Liga (link)	*Ninguna*	*-*

[1]Llama a WebAdministrarReportesAsignados.

[2]Regresa a página principal del sistema.

WebAdministrarReportesAsignados: Presenta al desarrollador una lista con los reportes a los que debe dar seguimiento y no han sido cerrados. La lista está formada por las siguientes columnas:

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Número de reporte[1]	*Liga (link)*	*Ninguna*	*-*
Nombre del producto	*Texto*	*Ninguna*	*-*
Módulo	*Texto*	*Ninguna*	*-*
Versión	*Texto*	*Ninguna*	*-*
Tipo severidad	*Texto*	*Ninguna*	*-*
Estatus	*Texto*	*Ninguna*	*-*
Fecha último estatus	*Texto*	*Ninguna*	*-*

[1]Llama a WebResponderReporte.

WebResponderReporte: Permite que el desarrollador capture o modifique la respuesta de un reporte que tiene asignado y no ha sido cerrado por el probador.

Nombre	***Tipo de componente***	***Validación***	***Número de posiciones***
Número de reporte	*Texto*	*Ninguna*	*-*
Nombre del producto	*Texto*	*Ninguna*	*-*
Módulo	*Texto*	*Ninguna*	*-*
Versión	*Texto*	*Ninguna*	*-*
Fecha de alta	*Texto*	*Ninguna*	*-*
Descripción error	*Texto*	*Ninguna*	*-*
Tipo severidad	*Texto*	*Ninguna*	*-*
Tipo error	*Texto*	*Ninguna*	*-*
Estatus	*Texto*	*Ninguna*	*-*
Fecha último estatus	*Texto*	*Ninguna*	*-*
**Diagnóstico*	*Caja de captura de texto*	*Alfanumérica*	*255*
**Etapa de inyección*	*Lista desplegable (combo box)*	*Ninguna*	*-*
**Descripción corrección*	*Caja de captura de texto*	*Alfanumérica*	*255*

*En caso de que ya se hubiera capturado una respuesta para el reporte, estos campos deben mostrar los valores previamente almacenados.

3.1.3 Identificación de objetos de control

Los objetos de control son responsables de la coordinación entre objetos de frontera y de entidad. Por lo general, los objetos de control no tienen una contraparte en el mundo real. A menudo hay

una relación cercana entre un caso de uso y un objeto de control. Un objeto de control se crea, por lo general, al inicio del caso de uso y deja de existir cuando termina; es responsable de la recopilación de información de los objetos de frontera y de enviarla a los objetos de entidad.

CtrlAltaSolicitudInscripcion	Este objeto se crea después de que el líder de proyecto envía los datos del formulario WebAltaSolicitudInscripcion. Recopila la información capturada y la envía a una nueva instancia de un objeto *Solicitud de inscripción*. Asigna un estatus de 'nuevo' a la solicitud de inscripción e invoca al método de alta. Con base en el resultado del alta crea un mensaje para el líder de proyecto. Si el alta fue exitosa, instancia un objeto *Correo* y crea un mensaje para notificar al administrador que se dio de alta una nueva solicitud e invoca al método *enviarCorreo*. Finalmente despliega en pantalla el mensaje para el líder de proyecto.
CtrlValidaUsuario	Este objeto se crea después de que cualquier usuario del sistema envía los datos capturados en el formulario WebValidaUsuario. Recopila la información capturada y envía la clave de usuario a una nueva instancia de un objeto *Usuario*. Busca si existe un usuario registrado en la base de datos con la misma clave. Si el usuario existe, compara el password capturado en el formulario con el password encontrado en la base de datos. Si son iguales, inicia una sesión del usuario y lo envía a la página con el menú correspondiente a su perfil. Si el usuario no existe en la base de datos o el password capturado es diferente al registrado en la base de datos, despliega en pantalla un mensaje para el usuario.
CtrlAutorizaSolicitud	Este objeto se crea después que el administrador envía los datos capturados desde el formulario de la página WebAutorizaSolicitud. Recopila la información capturada y la envía a una nueva instancia de un objeto *Solicitud de inscripción*. Actualiza el estatus de la solicitud. Si el estatus de la solicitud es *autorizada*, se instancia un nuevo objeto *Líder de proyecto*, se le asignan los valores de la solicitud de inscripción y se invoca el método de *alta*. Crea un mensaje según el resultado de la operación, instancia un objeto *Correo* para enviar el mensaje al líder de proyecto. Finalmente despliega en pantalla el mensaje para el administrador.

CtrlAltaDesarrollador	Este objeto se crea después de que el líder de proyecto envía los datos capturados desde el formulario de la página WebAltaDesarrollador. Recopila la información capturada, la envía a una nueva instancia de un objeto *Desarrollador* e invoca el método de *alta*. Con base en el resultado del alta crea un mensaje para el líder de proyecto. Si el alta fue exitosa, instancia un objeto *Correo* y crea un mensaje para notificar al desarrollador que se dio de alta e invoca al método *enviarCorreo*. Finalmente despliega en pantalla el mensaje para el líder de proyecto.
CtrlAltaProbador	Este objeto se crea después de que el líder de proyecto envía los datos capturados desde el formulario de la página WebAltaProbador. Recopila la información capturada, la envía a una nueva instancia de un objeto *Probador* e invoca el método de *alta*. Con base en el resultado del alta crea un mensaje para el líder de proyecto. Si el alta fue exitosa, instancia un objeto *Correo* y crea un mensaje para notificar al probador que se dio de alta e invoca al método *enviarCorreo*. Finalmente despliega en pantalla el mensaje para el líder de proyecto.
CtrlAltaProducto	Este objeto se crea después de que el líder de proyecto envía los datos capturados desde el formulario de la página WebAltaProducto. Recopila la información capturada, la envía a una nueva instancia de un objeto *Producto* e invoca el método de *alta*. Con base en el resultado del alta crea un mensaje para el líder de proyecto. Finalmente despliega en pantalla el mensaje para el líder de proyecto.
CtrlAsignarDesarrollador	Este objeto se crea después de que el líder de proyecto envía los datos capturados en el formulario de la página WebAsignarDesarrollador. Recopila la información capturada, instancia un objeto *Reporte de error*, busca el reporte por su número, le envía la clave de desarrollador y actualiza el reporte para almacenar la clave del desarrollador asignado. Instancia un objeto *Desarrollador*, le envía la clave de desarrollador, busca la información del mismo y utiliza la propiedad *email* para enviar la notificación al desarrollador, mediante el método *enviarCorreo* de la clase *Correo*. Finalmente despliega un mensaje para el líder de proyecto.

CtrlRechazarReporte	Este objeto se crea después de que el líder de proyecto envía los datos capturados en el formulario de la página WebRechazarReporte. Recopila la información capturada, instancia un objeto *Reporte de error*, busca el reporte por su número, le asigna la causa de rechazo y actualiza el reporte para almacenar el cambio en la base de datos. Obtiene la clave del probador que dio de alta el reporte, busca el resto de las propiedades del probador y utiliza la propiedad *email* para enviar la notificación al probador mediante el método *enviarCorreo* de la clase *Correo*. Finalmente, despliega un mensaje para el líder de proyecto.
CtrlGenerarEstadisticas	Este objeto se crea después de que el líder de proyecto selecciona una estadística a generar y captura los parámetros solicitados por el sistema. Recopila la información capturada, con base en el tipo de estadística seleccionada instancia el objeto de entidad correspondiente, obtiene la información de la estadística y la despliega en pantalla.
CtrlReportarError	Este objeto se crea después de que el probador envía los datos capturados en el formulario de la página WebReportarError. Recopila la información capturada, instancia un objeto *Reporte de error*, le envía los valores capturados e invoca al método *alta*. A partir de la clave del producto que se esté probando, busca al líder de proyecto que gestiona el producto, busca su email y le envía la notificación de que se recibió un nuevo reporte mediante el método *enviarCorreo* de la clase *Correo*. Finalmente despliega un mensaje para informar al probador que su reporte se dio de alta.
CtrlCerrarReporte	Este objeto se crea después de que el probador envía los datos capturados en el formulario de la página WebCerrarReporte. Recopila la información capturada, instancia un objeto *Reporte de error*, busca el reporte por su número, le envía la clave de estatus correspondiente a *cerrado* e invoca al método *actualiza* para guardar el cambio. Finalmente despliega un mensaje para el probador.
CtrlResponderReporte	Este objeto se crea después de que el desarrollador envía los datos capturados en el formulario de la página WebResponderReporte. Recopila la información capturada, instancia un objeto *Reporte de error*, busca el reporte por su

	número, le envía los valores capturados e invoca el método *actualiza* para guardar la información en la base de datos. Busca al probador que dio de alta el reporte, y le envía notifica vía correo electrónico utilizando el método *enviarCorreo* de la clase *Correo*. Finalmente despliega un mensaje para el desarrollador.

3.2 Modelo dinámico

Para aumentar la comprensión del flujo de trabajo a través de numerosos casos de uso es conveniente usar diagramas de actividades. Cuando los casos de uso interactúan entre ellos, los diagramas de actividades son una herramienta que ayuda a representar y entender este comportamiento. [FOW99b]

Las actividades son estados que representan la ejecución de un conjunto de operaciones. La terminación de estas operaciones dispara una transición hacia otra actividad. Los diagramas de actividades pueden usarse para representar el flujo de control. [BRU02a]

El siguiente diagrama de actividades representa el flujo de un reporte de error, desde su creación hasta su cierre.

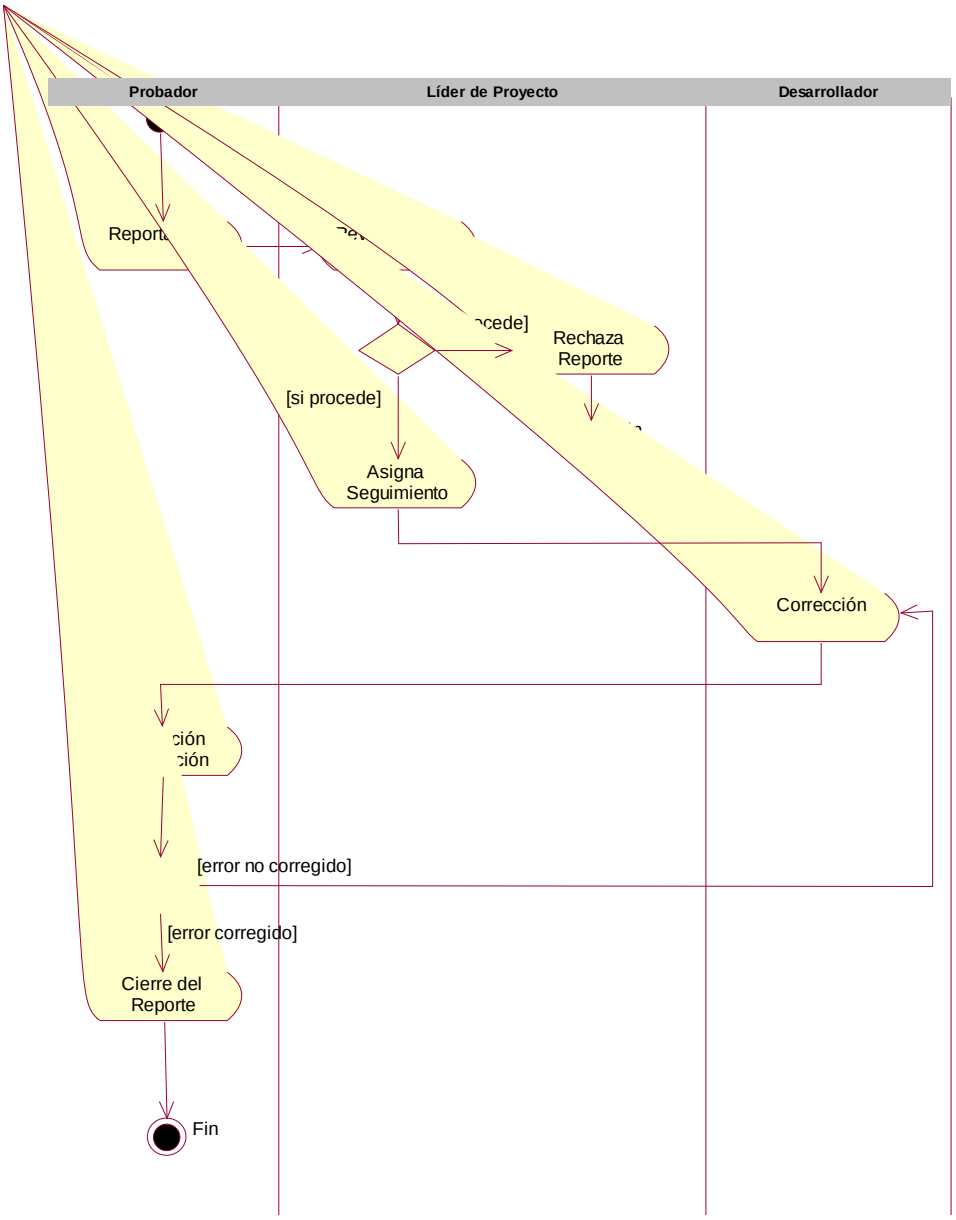


Figura 3-2 Diagrama de actividades para el flujo de un reporte de error.

Los diagramas de estados son una técnica para describir el comportamiento de un objeto a través de varios casos de uso Describen todos los estados posibles en los que puede entrar un objeto particular y la manera en que cambia el estado del objeto, como resultado de los eventos que llegan a él. En la mayor parte de las técnicas orientadas a objetos, los diagramas de estados se dibujan para una sola clase, mostrando el comportamiento de un solo objeto durante todo su ciclo de vida. [FOW99c]

A continuación se muestran los diagramas de estados para la solicitud de inscripción al sistema y para un reporte de error.

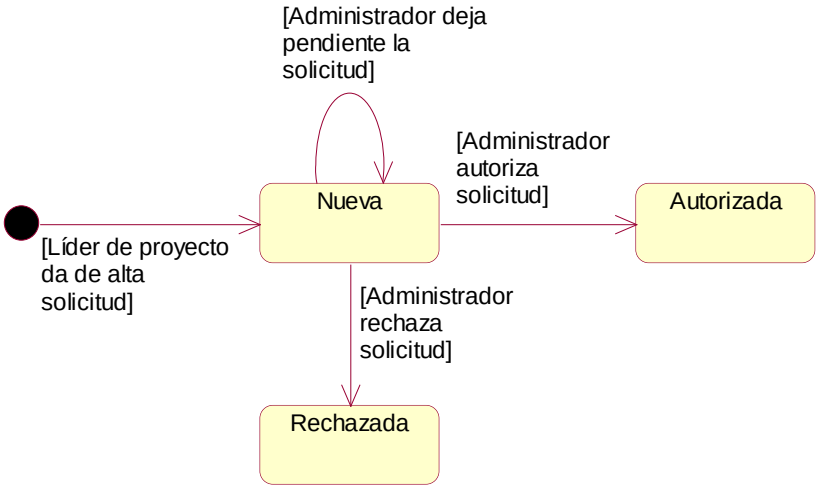


Figura 3-3 Diagrama de estados para el ciclo de vida de una solicitud.

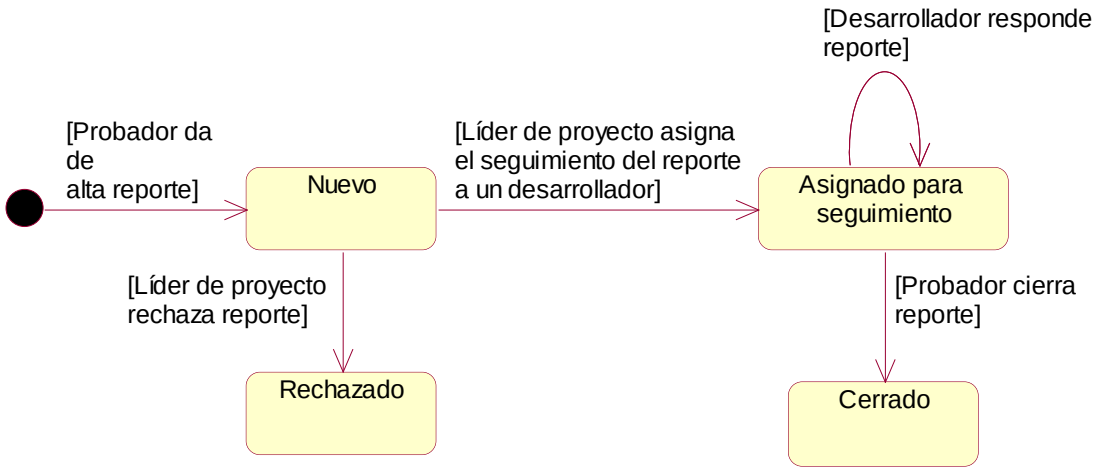


Figura 3-4 Diagrama de estados para el ciclo de vida de un reporte de error.

Un diagrama de secuencia une los casos de uso con los objetos. Muestra cómo se distribuye el comportamiento de un caso de uso entre sus objetos participantes. Los diagramas de secuencia no son, por lo general, un buen medio de comunicación con el usuario; sin embargo, representan un cambio de perspectiva y permiten que los desarrolladores encuentren objetos faltantes o áreas "grises" en la especificación del sistema.

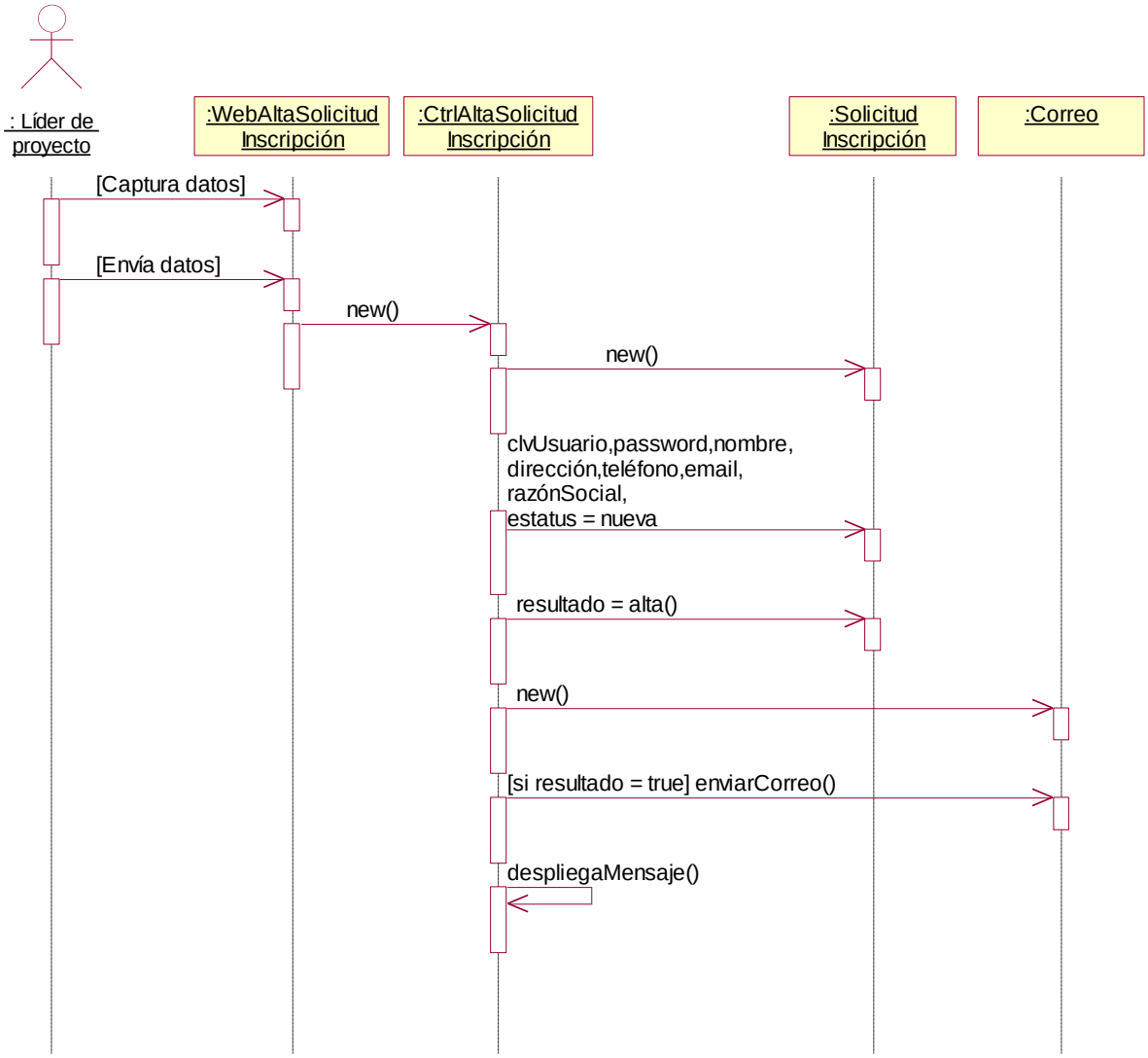


Figura 3-5 Diagrama de secuencia para el caso de uso *Alta solicitud inscripción*.

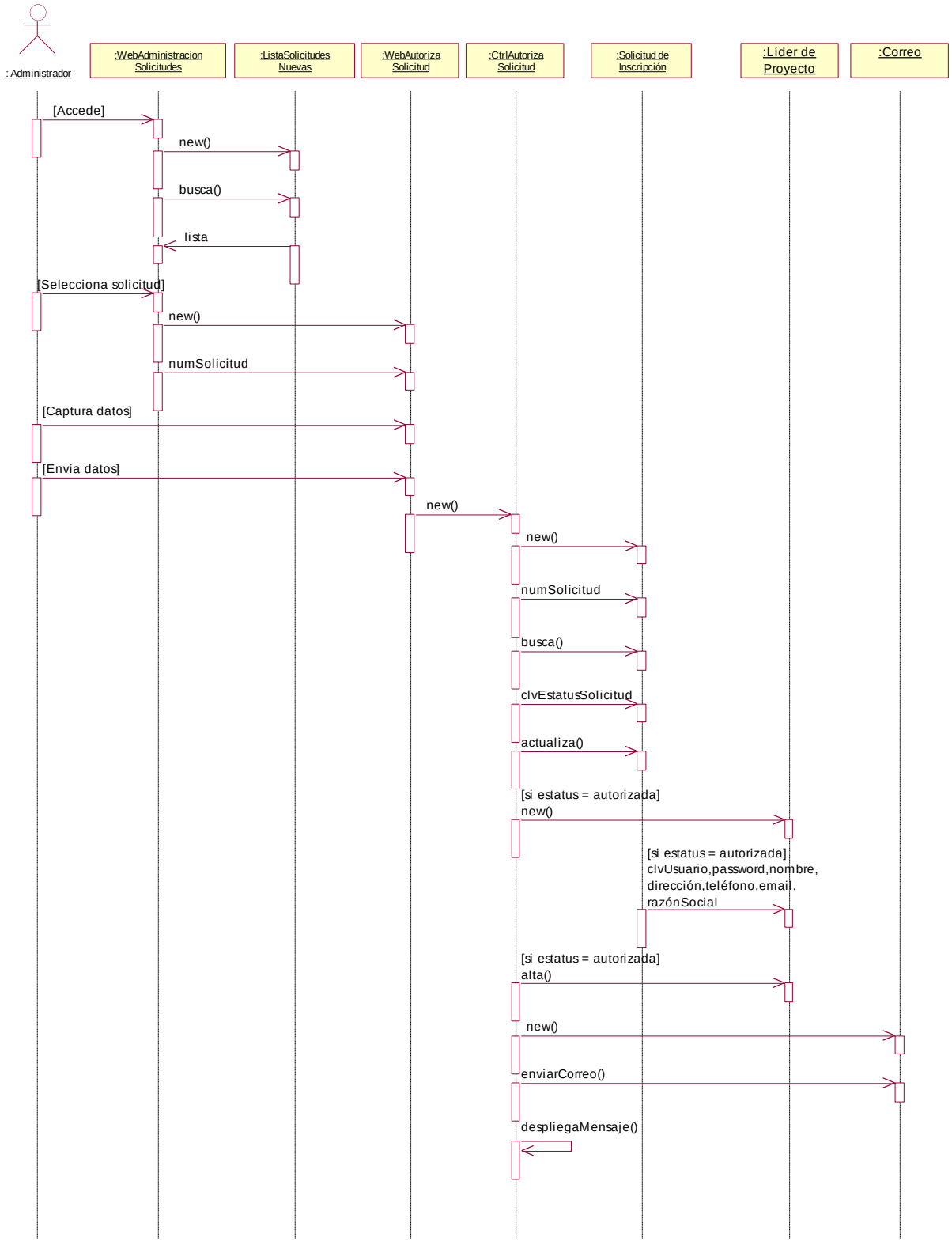


Figura 3-6 Diagrama de secuencia para el caso de uso *Autoriza solicitud inscripción.*

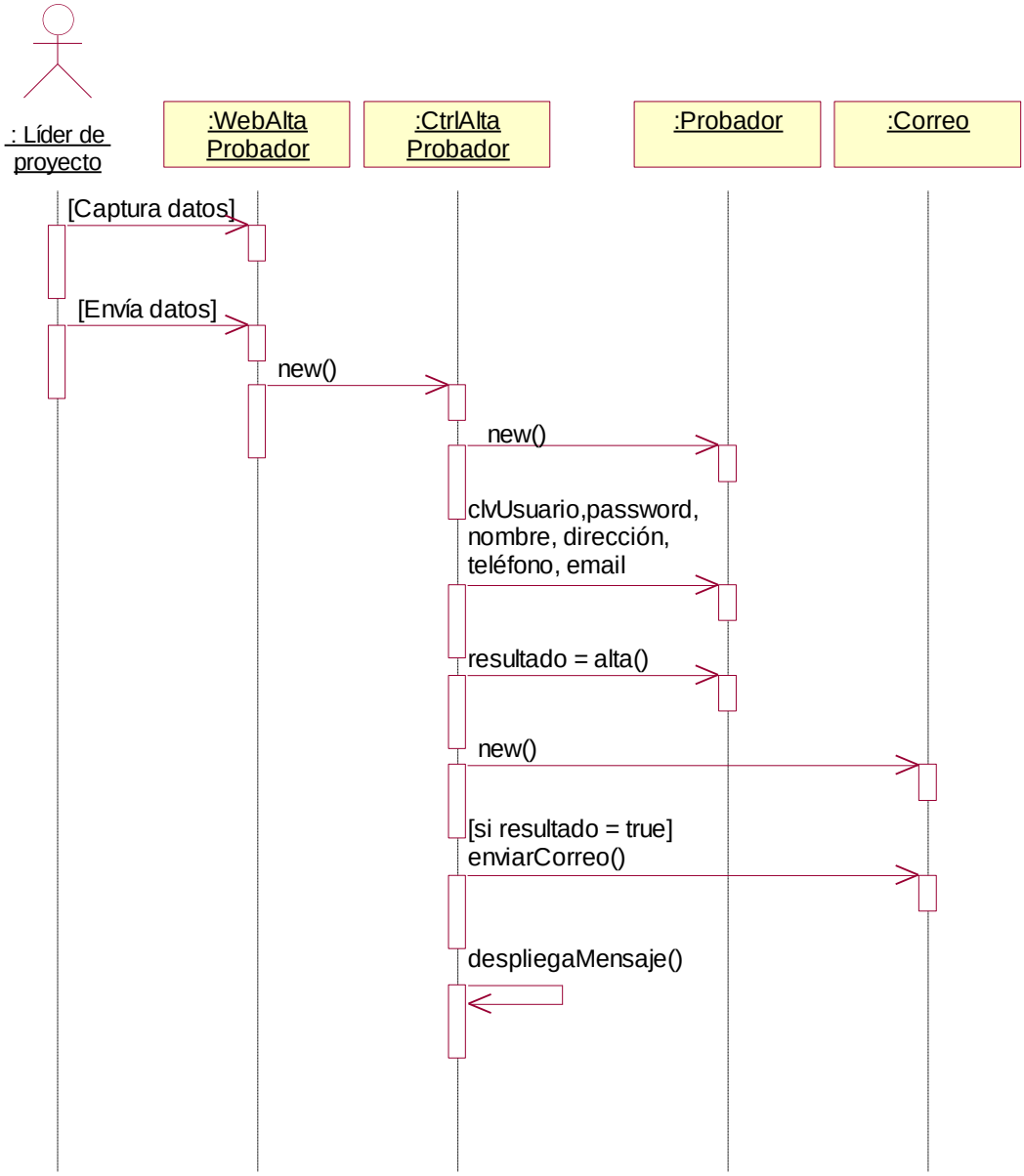


Figura 3-7 Diagrama de secuencia para el caso de uso *Alta probador*.

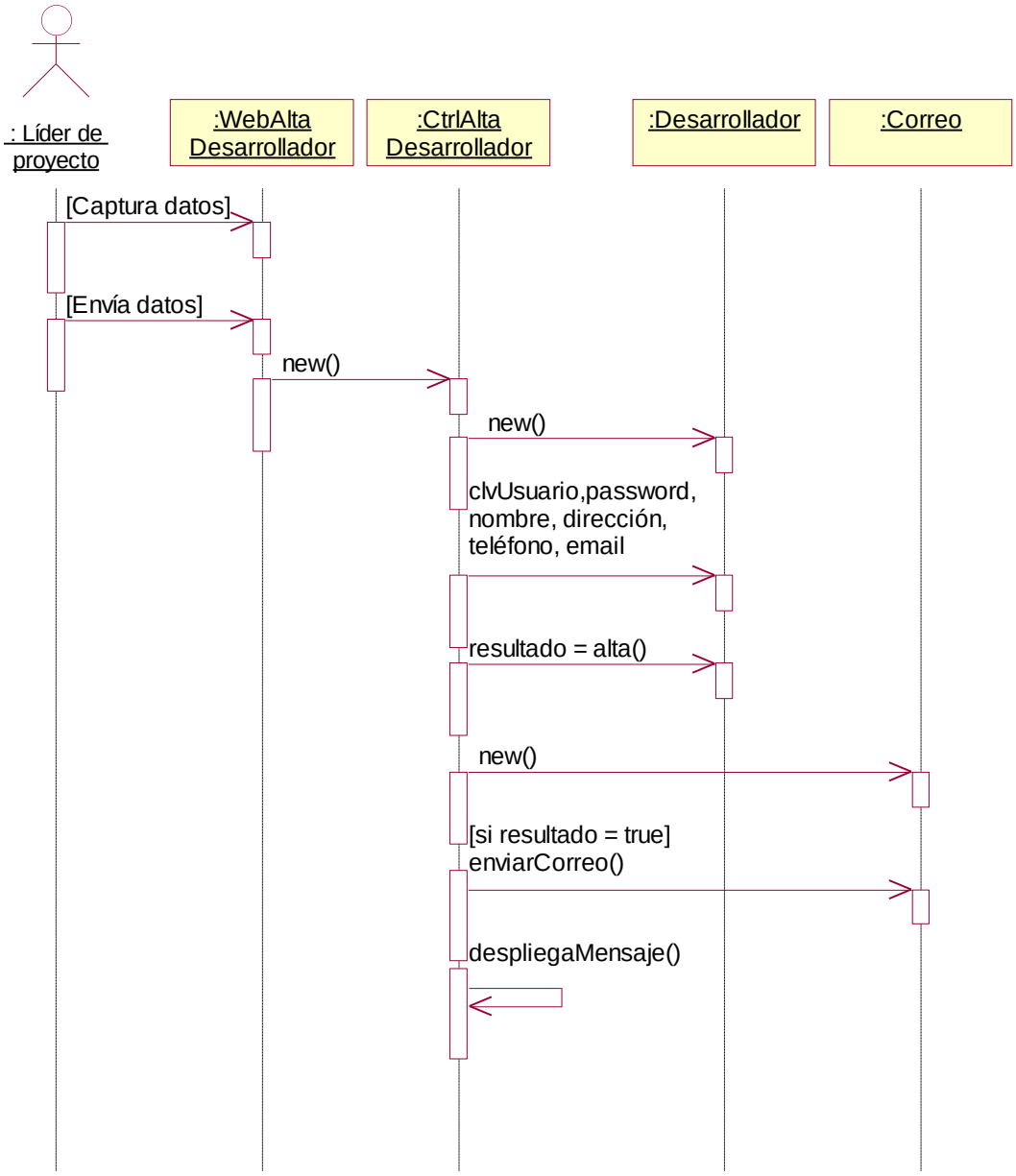


Figura 3-8 Diagrama de secuencia para el caso de uso *Alta desarrollador*.

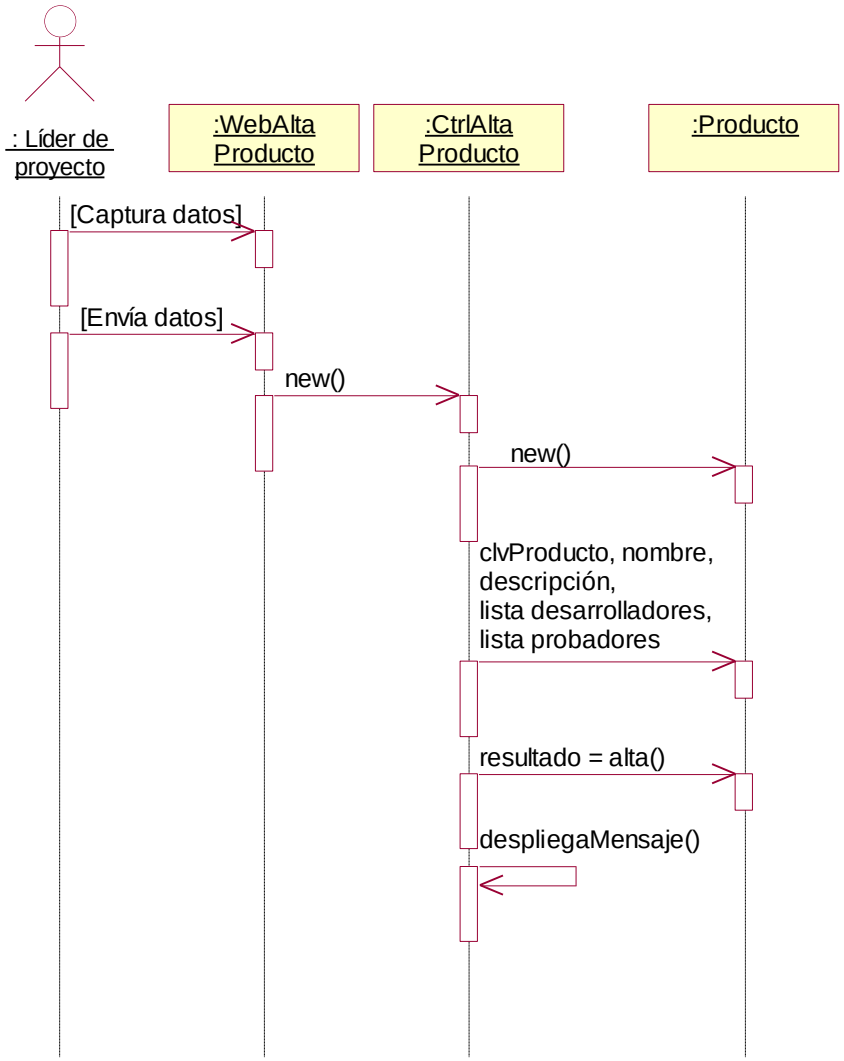


Figura 3-9 Diagrama de secuencia para el caso de uso Alta producto.

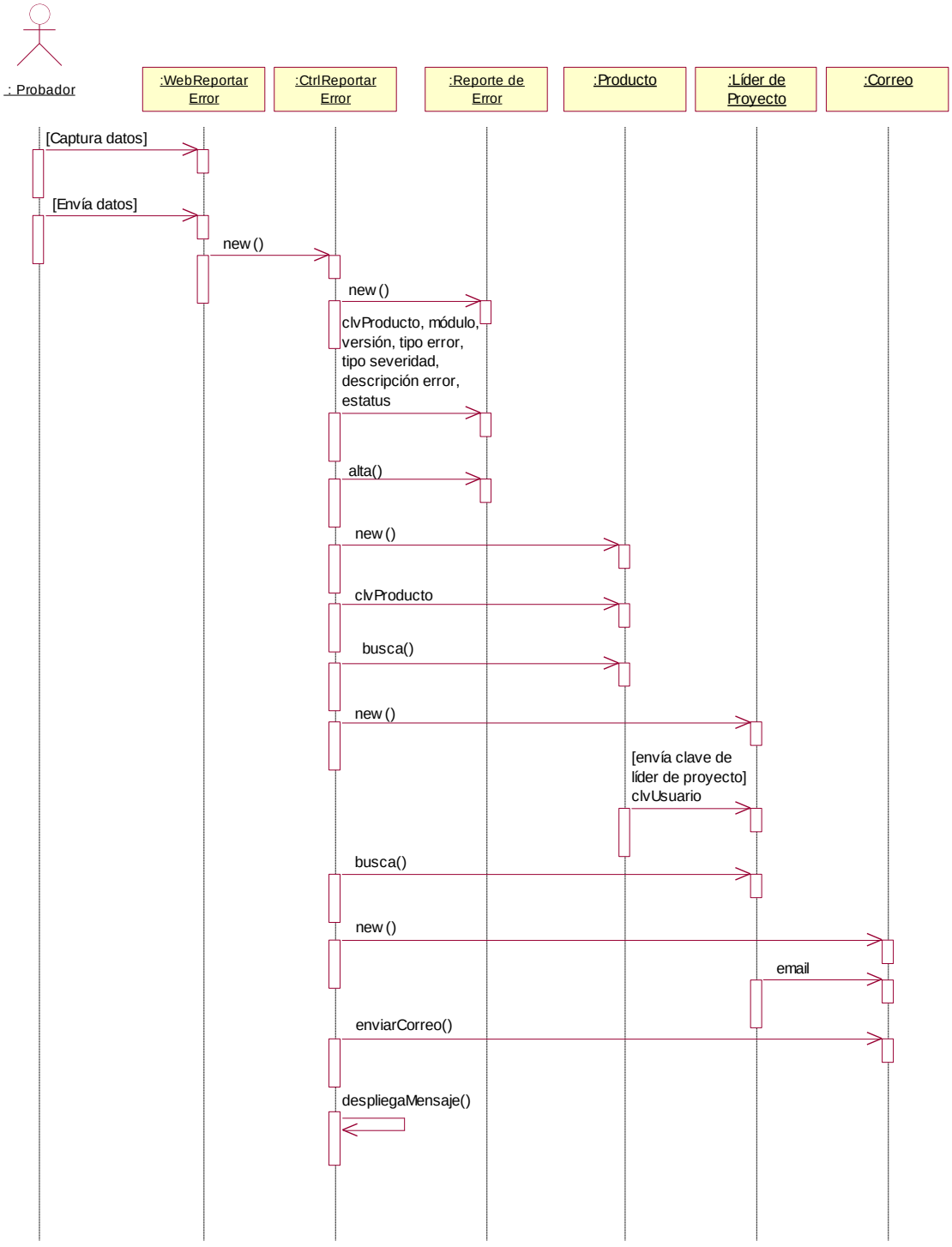


Figura 3-10 Diagrama de secuencia para el caso de uso *Reportar error*.

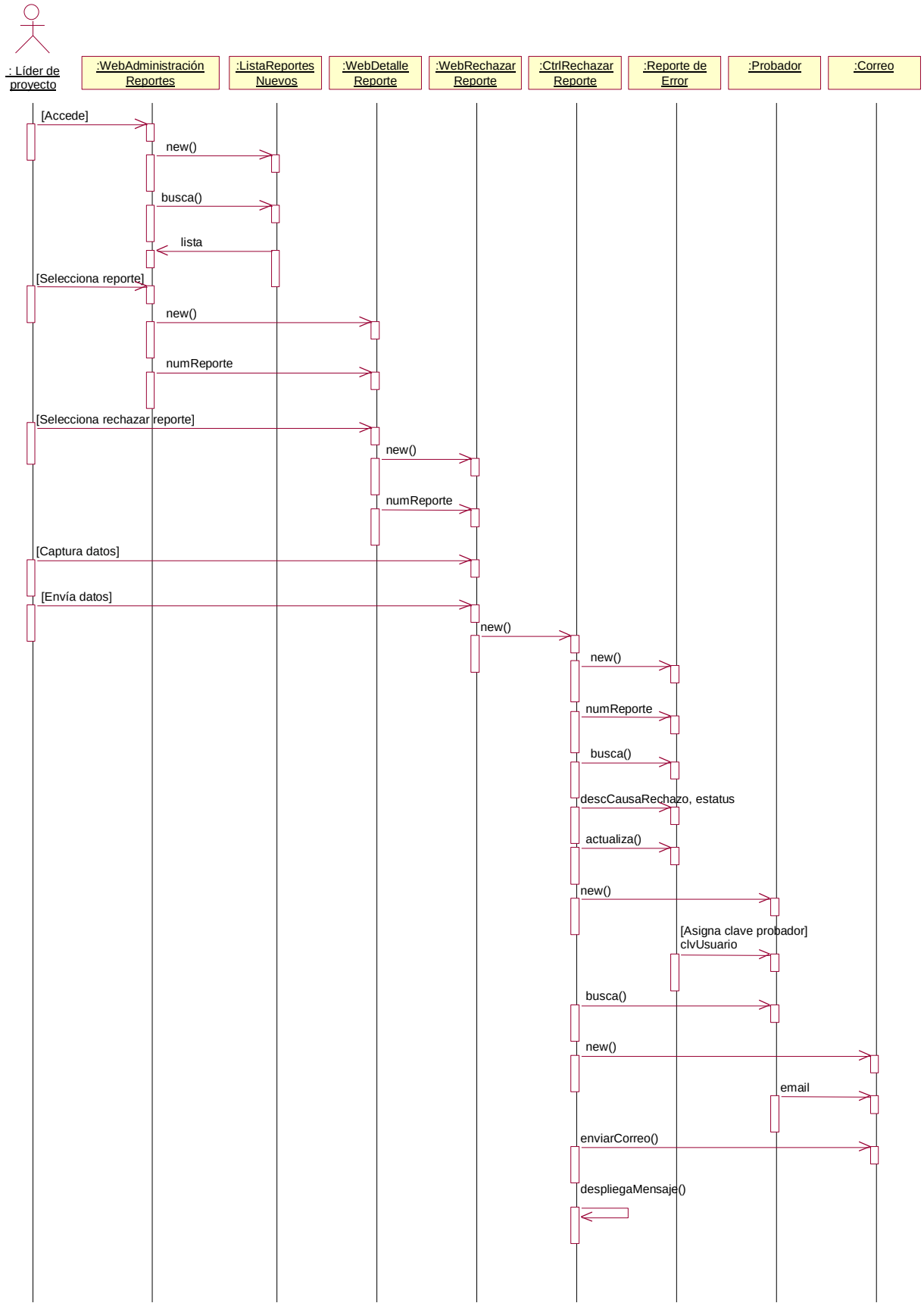


Figura 3-11 Diagrama de secuencia para el caso de uso *Rechazar reporte*.

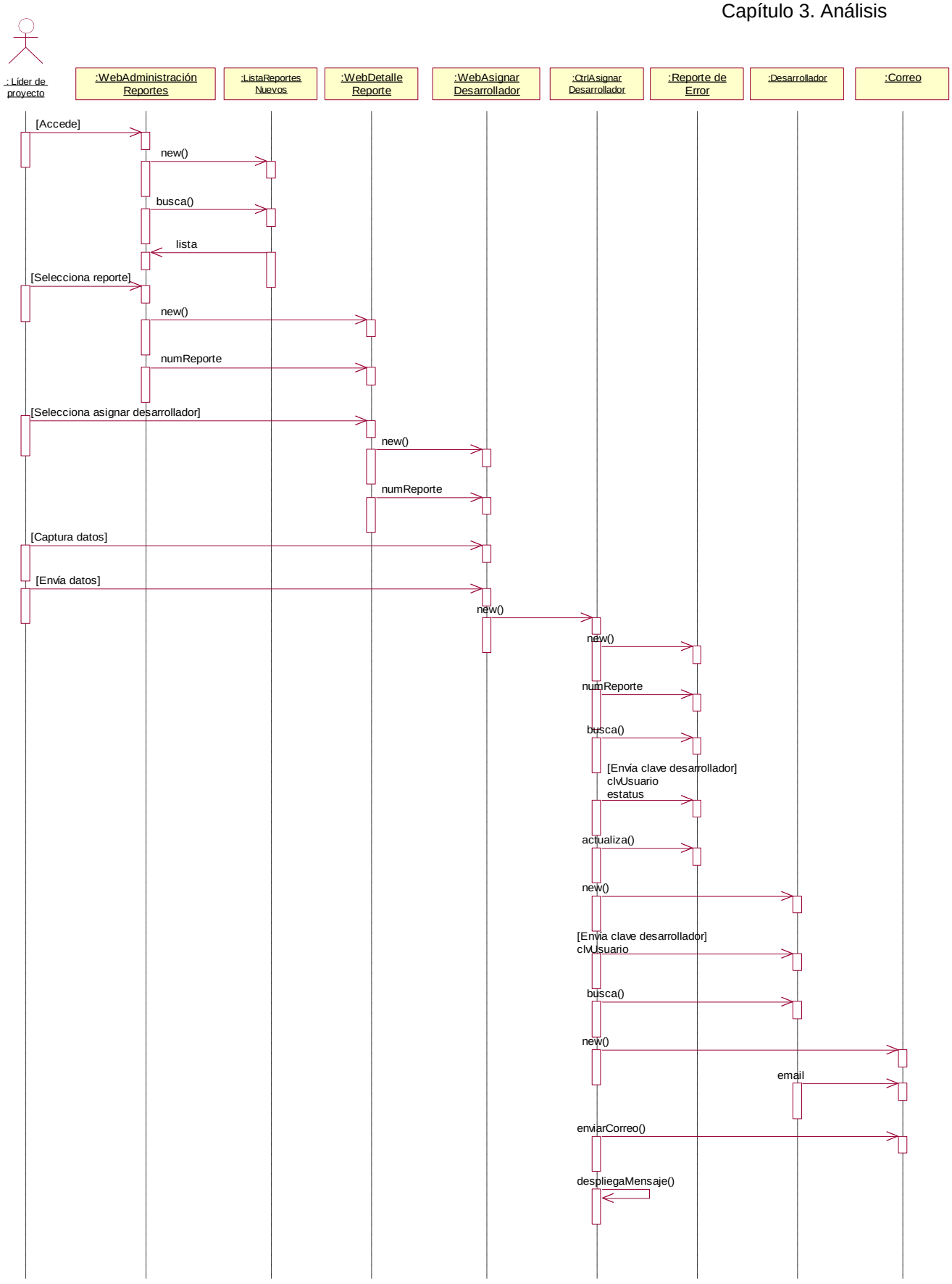


Figura 3-12 Diagrama de secuencia para el caso de uso *Asignar desarrollador*.

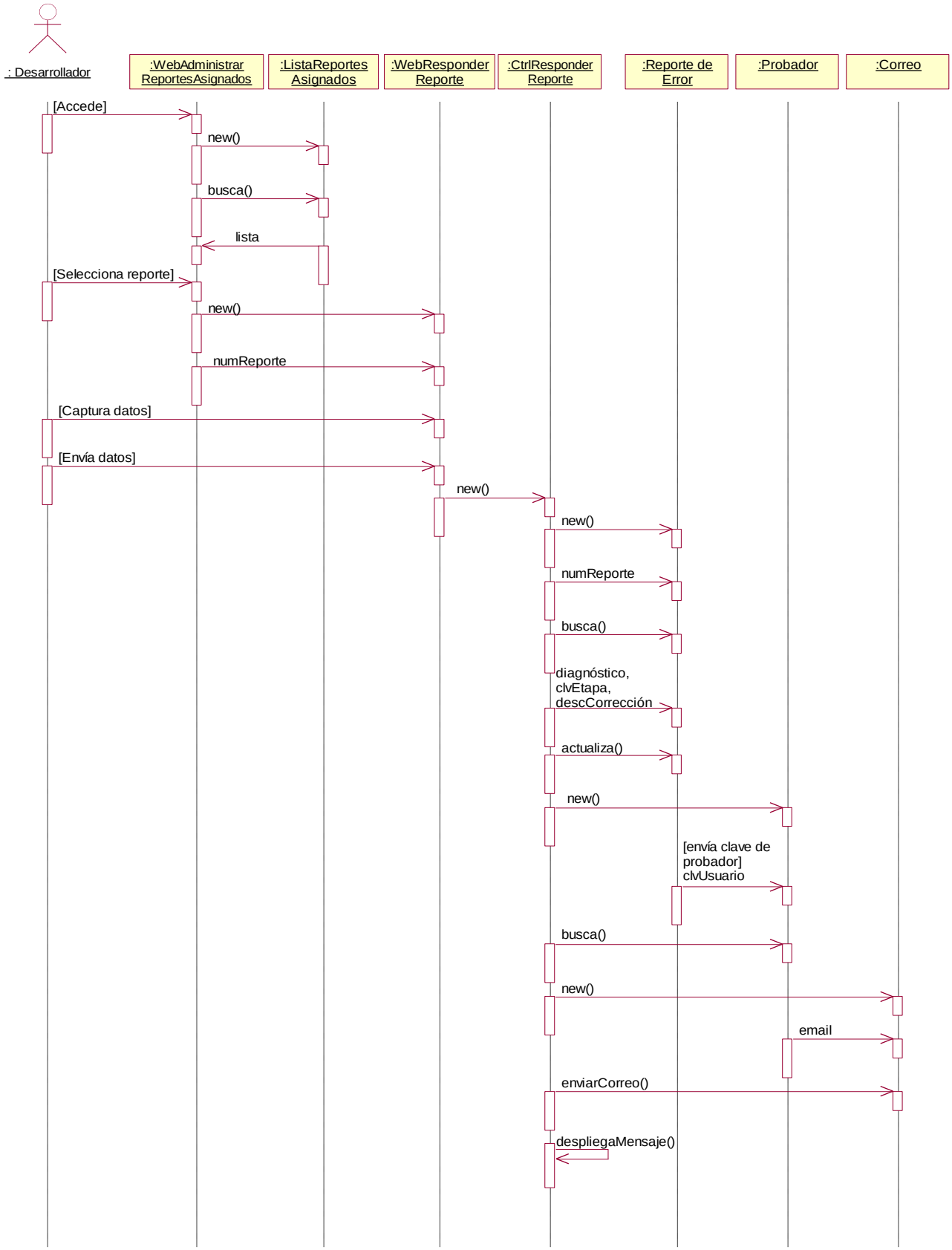


Figura 3-13 Diagrama de secuencia para el caso de uso *Responder reporte*.

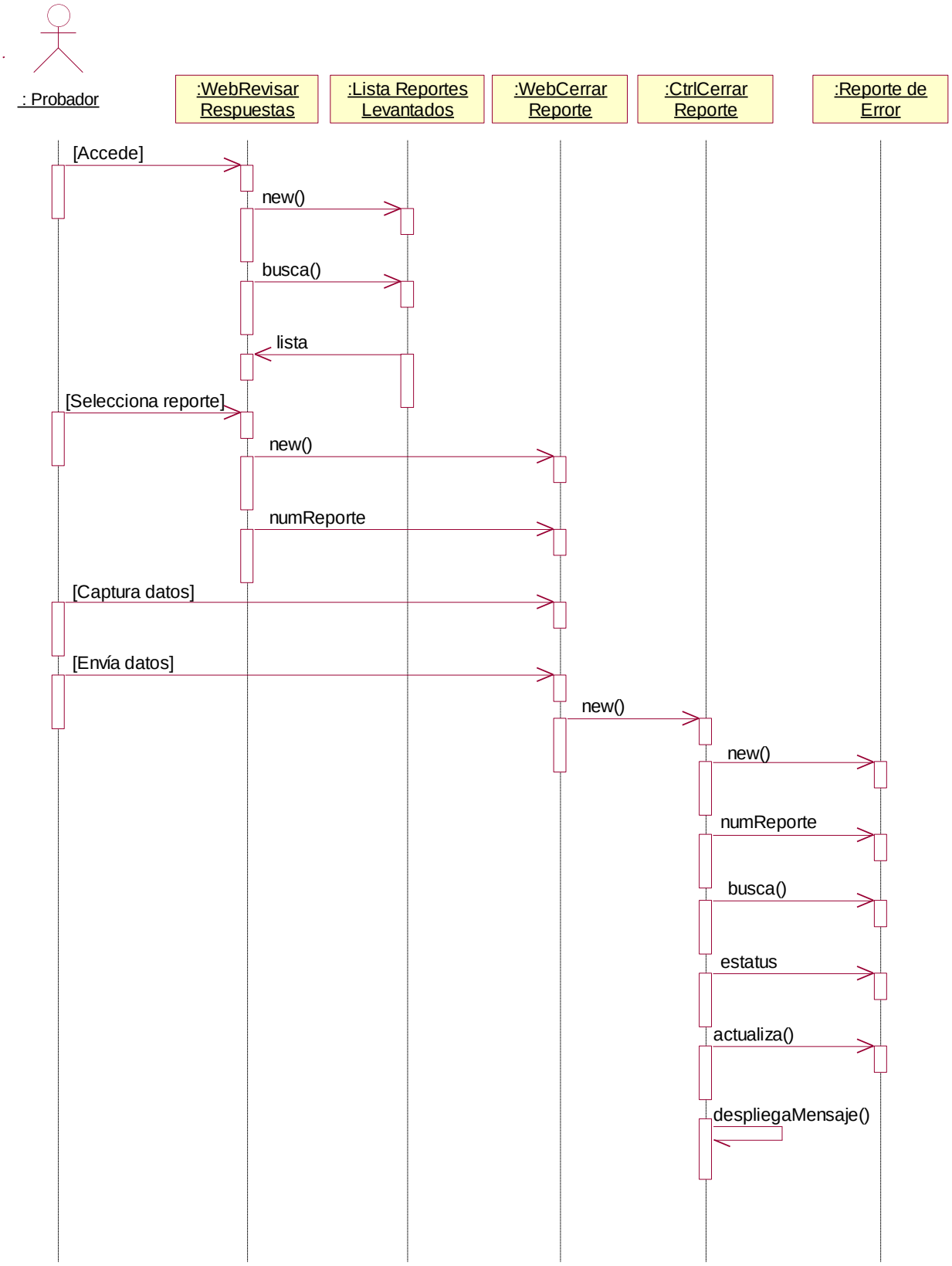


Figura 3-14 Diagrama de secuencia para el caso de uso *Cerrar reporte*.

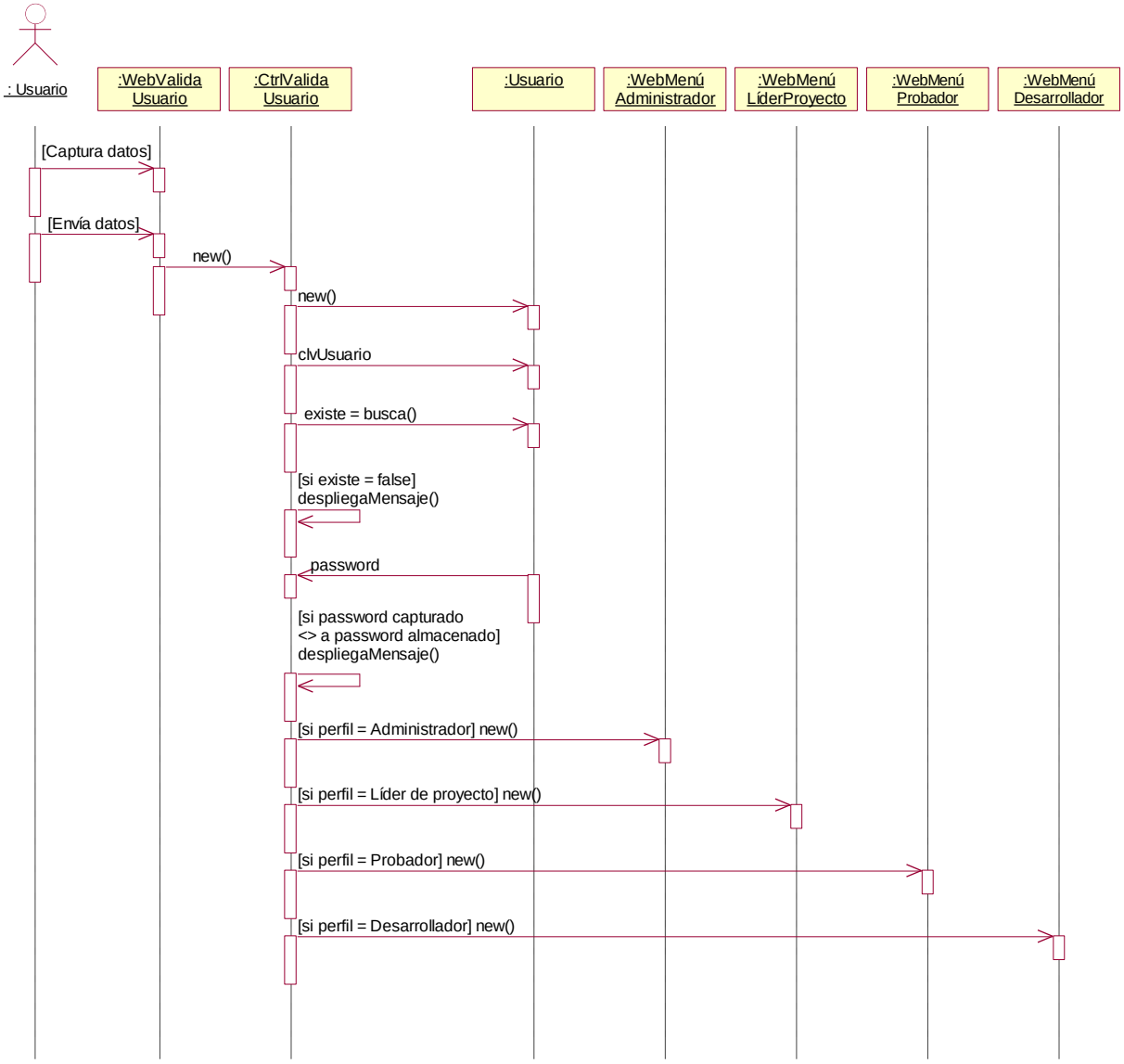


Figura 3-15 Diagrama de secuencia para el caso de uso *Valida usuario*.

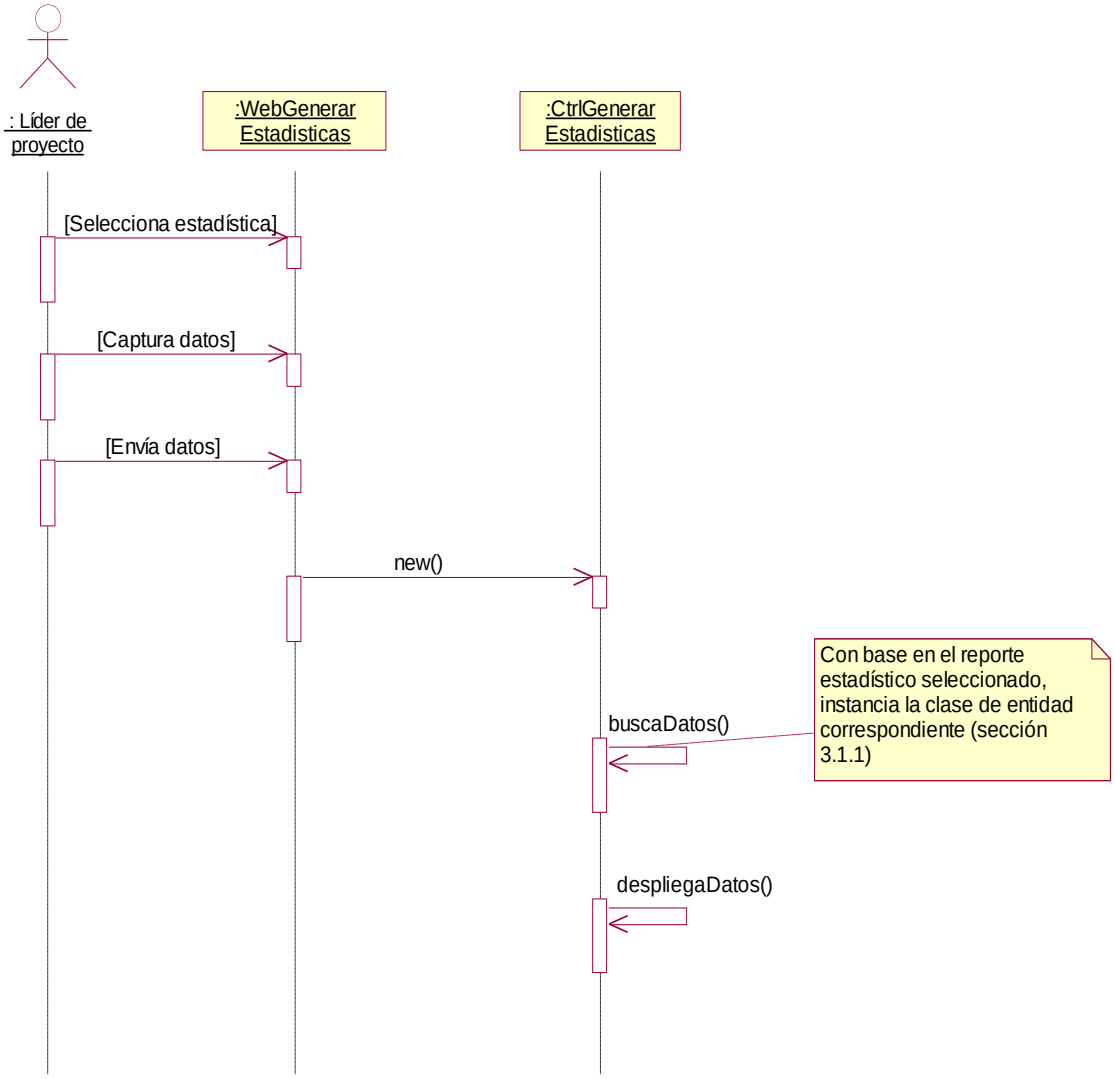


Figura 3-16 Diagrama de secuencia para el caso de uso *Generar estadísticas*.

Capítulo 4

Diseño de alto nivel

El diseño de sistemas es la transformación del modelo de análisis en un modelo de diseño del sistema. Durante el diseño del sistema los desarrolladores definen los objetivos de diseño del proyecto y descomponen el sistema en subsistemas más pequeños. También seleccionan estrategias para la construcción del sistema, como la plataforma de hardware y software en la que se ejecutará el sistema, la estrategia de almacenamiento de datos persistentes, el flujo de control global y la política de control de acceso.

En este capítulo se detallan las decisiones de diseño del sistema, que incluyen:

- Identificación de objetivos de diseño
- Almacenamiento de datos persistentes
- Arquitectura del sistema
- Diseño del flujo de control
- Seguridad
- Manejo de excepciones
- Componentes del sistema
- Descomposición en subsistemas

4.1 Identificación de objetivos de diseño

La definición de los objetivos de diseño es el primer paso en el diseño del sistema. Identifica las cualidades en las que debe enfocarse el sistema. Es necesario especificarlos de manera explícita para que cada una de las decisiones de diseño importantes pueda tomarse en forma consistente siguiendo el mismo conjunto de criterios. [BRU02e]

- El SIARE deberá ser de fácil acceso para los usuarios finales.
- El SIARE deberá poder ejecutarse en equipos con recursos limitados (memoria RAM y procesador).
- El SIARE deberá garantizar la integridad de los datos alimentados por los usuarios finales.
- El SIARE deberá ser seguro, es decir, cada usuario tendrá acceso únicamente a la funcionalidad y los datos correspondientes a su perfil.

- El SIARE deberá ser económico, desde el punto de vista del costo de las herramientas necesarias para su desarrollo y administración.
- El SIARE deberá estar modularizado de forma que los cambios en la lógica de la aplicación no afecten a la interfaz gráfica de usuario (GUI) y viceversa.
- El SIARE deberá ser portable, es decir, deberá poder instalarse en multitud de plataformas con un mínimo de cambios en el código.

4.2 Almacenamiento de datos persistentes

Los datos persistentes sobreviven a una sola ejecución del sistema. Por ejemplo, un autor puede guardar su trabajo en un archivo cuando usa un procesador de palabras; luego puede volver a abrir el archivo días o semanas después, no es necesario que se esté ejecutando el procesador de palabras para que exista el archivo.

El lugar y la manera en que se guardan los datos tienen un impacto en la descomposición del sistema. La selección de un sistema de administración de base de datos específico también puede tener implicaciones en la estrategia de control general y la administración de la concurrencia.

En general, primero es necesario identificar cuáles objetos necesitan ser persistentes. La persistencia de objetos se infiere en forma directa del dominio de la aplicación. Luego es necesario decidir la manera en que deben guardarse estos objetos. La decisión sobre la administración del almacenamiento es más compleja y, por lo general, está dictada por requerimientos no funcionales: ¿deben recuperarse rápido los objetos?, ¿se necesitan consultas complejas?, ¿ocupan mucho espacio los objetos? Los sistemas de administración de base de datos proporcionan mecanismos para el control de la concurrencia y consultas eficientes sobre conjuntos de datos grandes. [BRU02e]

En la actualidad hay tres opciones realistas para la administración del almacenamiento:

- **Archivos planos.** Los archivos son la abstracción de almacenamiento proporcionada por los sistemas operativos. La aplicación guarda sus datos como una secuencia de bytes y define la manera y el momento en que deben recuperarse los datos. La abstracción de archivo es relativamente de bajo nivel y permite que la aplicación realice una variedad de optimizaciones de tamaño y velocidad. Sin embargo, los archivos requieren que la aplicación se encargue de varios asuntos, como el acceso concurrente y la pérdida de datos en caso de falla del sistema.
- **Base de datos relacional.** Una base de datos relacional proporciona una abstracción de datos que es más elevada que la de los archivos planos. Los datos se guardan en tablas que se

apegan a un tipo predefinido llamado esquema. Cada columna de la tabla representa un atributo. Cada renglón representa un concepto de datos como un tuplo de valores del atributo. Varios tuplos de diferentes tablas se usan para representar los atributos de un objeto individual. Las bases de datos relacionales ya se han usado bastante tiempo y son una tecnología madura.

- **Base de datos orientada a objetos.** Una base de datos orientada a objetos proporciona servicios similares a los de una base de datos relacional. A diferencia de una base de datos relacional, guarda los datos como objetos y asociaciones. Además de proporcionar un nivel de abstracción más alto (reduciendo, por tanto, la necesidad de traducir entre objetos y entidades de almacenamiento), las bases de datos orientadas a objetos proporcionan a los desarrolladores herencia y tipos de datos abstractos. Las bases de datos orientadas a objetos son, por lo general, más lentas que las bases de datos relacionales para consultas típicas.

Para el SIARE se decidió usar una base de datos relacional por las siguientes razones:

- El sistema generará accesos concurrentes sobre los datos.
- El sistema realizará consultas complejas sobre los datos.
- El sistema actuará sobre un conjunto grande de datos.
- El sistema deberá diseñarse para ser portable entre varias plataformas.
- Múltiples programas del sistema accederán a los mismos datos.
- El sistema requiere que se garantice la integridad de los datos; una base de datos relacional provee los servicios necesarios para garantizar dicha integridad de forma transparente para la construcción del sistema.

4.3 Arquitectura del sistema

Conforme se incrementa la complejidad de los sistemas se hace crítica la especificación de la descomposición del sistema. Es difícil modificar o corregir una descomposición débil una vez que ha comenzado el desarrollo conforme tienen que cambiarse más interfaces del subsistema. En reconocimiento de la importancia de este problema ha surgido el concepto de arquitectura de software, la cual incluye la descomposición del sistema, el flujo de control global, las políticas de manejo de errores y los protocolos de comunicación entre subsistemas. [SHA96]

4.3.1 Arquitectura cliente/servidor

En la arquitectura cliente/servidor, un subsistema, el servidor, proporciona servicios a instancias de los demás subsistemas llamados clientes, que son responsables de la interacción con el usuario. La petición de un servicio se realiza, por lo general, mediante un mecanismo de llamada a procedimiento remoto. El flujo de control en los clientes y el servidor es independiente, a excepción de la sincronización para administrar las peticiones o para recibir los resultados. [BRU02e]

Un sistema de información con una base de datos central es un ejemplo de la arquitectura cliente/servidor. Los clientes son responsables de la recepción de entrada del usuario, la realización de revisiones de rango y la iniciación de transacciones de base de datos una vez que se han recolectado todos los datos necesarios. El servidor es responsable de la realización de las transacciones y de garantizar la integridad de los datos.

Las arquitecturas cliente/servidor son muy adecuadas para los sistemas distribuidos que manejan grandes cantidades de datos.

4.3.1.1 Aplicaciones Web

Una aplicación Web es cualquier aplicación o sistema de aplicaciones que utiliza el protocolo de transferencia de hipertexto o HTTP como principal protocolo de transporte. HTTP es el protocolo fundamental del que se sirve la Web, por lo que su definición abarca las aplicaciones que se utilizan en ésta. [MAR00a]

La Web se diseñó originalmente como un medio para suministrar páginas estáticas a los usuarios de Internet. Cuando un navegador Web envía una consulta HTTP a un servidor Web, este último extrae el archivo de consulta de su sistema de archivos y lo devuelve al navegador a través de la conexión HTTP.

Sin embargo, lo que devuelve el servidor Web no tiene por qué ser siempre una página estática (archivo) almacenada en el servidor.

Entre los principales mecanismos para la creación de páginas Web dinámicas es posible citar:

CGI (Common Gateway Interface): Es una especificación para la transferencia de información entre un servidor Web y un programa CGI, que no es más que un programa que puede aceptar parámetros de una consulta HTTP y devolver los datos como si se tratara de una página

almacenada. CGI presenta un alto costo porque cada vez que se invoca un CGI, se inicia un nuevo proceso.

Scripting: El código ejecutable está contenido en la página HTML. El servidor Web está configurado de forma que cuando se accede a un URL que hace referencia a un script, se carga una determinada biblioteca dinámica de enlace (DLL) en el espacio del proceso del servidor y la secuencia de invocación estándar invoca la DLL como si formara parte del código del servidor. Puesto que el programa de aplicación (DLL) se ejecuta en el mismo espacio de proceso que el servidor resulta muy eficiente. Los lenguajes de scripting más utilizados para la creación de páginas Web dinámicas son Perl, ASP y PHP.

Servlets: Un servlet es un mecanismo que permite invocar un programa Java. El marco de trabajo de Java del lado del servidor para aplicaciones Web es el que permite que éstas interactúen con clientes externos y con otras aplicaciones Web. Cuando se realiza una consulta a un URL, la clase Java asociada (servlet) se ejecuta. El servlet se ejecuta en una JVM (Java Virtual Machine) que reside en el mismo espacio de proceso que el servidor, por lo que no se produce ningún costo de intercambio de proceso.

Las aplicaciones Web se construyen habitualmente siguiendo el modelo de tres capas. Este modelo surgió ante la necesidad de separar la lógica de la interfaz gráfica de usuario y la base de datos remota. De acuerdo con este modelo, tres procesos separados y perfectamente definidos o, lo que es lo mismo, tres módulos se ejecutan en plataformas distintas:

1.- La interfaz gráfica de usuario, es decir, el navegador que se ejecuta en la computadora del usuario.
2.- El programa o programas de aplicación que se ejecutan en el servidor Web y que se encargan de procesar los datos (el nivel lógico).
3.- Un sistema de base de datos que almacena los datos que requiere la capa 2 del modelo.

El uso masivo que alcanzaron los navegadores Web junto con el uso de las tecnologías para creación dinámica de páginas hicieron posibles y prácticas las aplicaciones en tres capas. Este modelo presenta una serie de ventajas frente a los modelos tradicionales de una o dos capas, entre las que cabe destacar:

- Los navegadores Web están presentes en máquinas basadas en todas las plataformas, lo cual facilita el acceso a las aplicaciones Web.
- Las aplicaciones pueden compartir un mismo aspecto y finalidad.

- La estructura modular facilita la modificación o sustitución de alguno de los módulos sin que ello afecte a los demás.

El protocolo HTTP es un protocolo no orientado a conexión (*connectionless*), es decir, una vez que el servidor responde a una petición del cliente se pierde la conexión entre ambos. Esta característica hace que, en principio, no se puedan mantener datos persistentes entre llamadas a distintas páginas Web. Conforme las aplicaciones Web se volvieron más complejas, se requirió que se mantuvieran datos persistentes entre solicitudes a páginas que forman parte de una misma aplicación Web; de tal forma que se idearon dos métodos para mantener datos persistentes entre solicitudes a distintas páginas:

1) **Cookies**: Una "cookie" es un archivo que se guarda en la máquina cliente, dentro del archivo se almacenan en forma de texto los valores de las variables que se desea hacer persistentes.
2) **Variables de sesión**: Los valores de las variables persistentes se almacenan en la memoria del servidor Web. Todas las variables de sesión se serializan después de que la solicitud termina.

Se decidió usar para el SIARE una aplicación Web por las siguientes razones:

- Facilitar el acceso a la aplicación a todos los usuarios, dada la ubicuidad de los navegadores Web.
- Minimizar la necesidad de instalación de software en las máquinas cliente.
- Permitir que el sistema pueda usarse aun en máquinas con recursos limitados.
- Facilitar la distribución de los cambios en la aplicación (dado que todas las actualizaciones al sistema sólo necesitan hacerse en el servidor).
- Existe gran variedad de opciones en cuanto a tecnologías disponibles para la creación de aplicaciones Web.

4.3.2 Diseño del flujo de control

El flujo de control es el ordenamiento de las acciones en un sistema. En los sistemas orientados a objetos las acciones a ordenar incluyen la decisión de cuáles operaciones deben ejecutarse y en qué orden. Estas decisiones se basan en eventos externos provocados por un actor o en el paso del tiempo. [BRU02e]

El flujo de control es un problema de diseño. Durante el análisis, el flujo de control no importa debido a que se asume tan sólo que todos los objetos se están ejecutando en forma simultánea, ejecutando operaciones en el momento en que necesitan hacerlo. Durante el diseño del sistema es necesario tomar en cuenta que no todos los objetos tienen el lujo de estar ejecutándose en su propio procesador.

Un servidor Web funciona mediante un control por *hilos*, que son la variación concurrente del control manejado por procedimientos: el sistema puede crear una cantidad arbitraria de hilos y cada una responde a un evento diferente. Si un hilo necesita datos adicionales, espera entrada de un actor específico. De esta forma, un servidor Web soporta concurrentemente varios usuarios accediendo al sistema.

A su vez, una aplicación Web vista desde el lado cliente, se controla mediante eventos, es decir, un ciclo principal espera un evento externo. Cada vez que se tiene disponible un evento se le despacha al objeto adecuado con base en la información asociada con el evento. Este tipo de flujo de control tiene la ventaja de conducir hacia una estructura más simple y centralizar toda la entrada en el ciclo principal.

4.3.3 Seguridad

4.3.3.1 Control de acceso

En los sistemas multiusuarios, actores diferentes tienen acceso a funcionalidades y datos diferentes. Durante el análisis se modelan estas distinciones asociando diferentes casos de uso a actores diferentes. Durante el diseño del sistema se modela el acceso examinando el modelo de objetos, determinando cuáles objetos están compartidos entre actores y definiendo la manera en que los actores pueden controlar el acceso. Dependiendo de los requerimientos de seguridad del sistema, también se define la manera en que los actores se autentifican ante el sistema (es decir, cómo prueban los actores quiénes son ante el sistema). [BRU02e]

Es posible modelar el acceso a las clases con una matriz de acceso. Los renglones de la matriz representan a los actores del sistema. Las columnas representan a las clases cuyo acceso controlamos. A una entrada *(clase, actor)* de la matriz de acceso se le llama derecho de acceso y enlista las operaciones que puede ejecutar el *actor* en instancias de la *clase*.

Podemos representar la matriz de acceso usando uno de tres diferentes enfoques:

- Una **tabla de acceso global** representa en forma explícita a cada celda de la matriz como un tuplo *(actor, clase, operación)*. La determinación de si un actor tiene acceso a un objeto específico requiere que se busque el tuplo correspondiente; si no se encuentra, se niega el acceso.
- Una **lista de control de acceso** asocia una lista de pares *(actor, operación)* con cada *clase* a acceder. Cada vez que se accede a un objeto se revisa la lista de acceso para buscar el actor y la operación correspondientes.
- Una **capacidad** asocia un par *(clase, operación)* con un *actor*. Una capacidad proporciona a un actor la obtención de acceso a un objeto de la clase descrita en la capacidad. La negación de la capacidad es equivalente a la negación del acceso.

La representación de la matriz de acceso también es un asunto de desempeño. Las tablas de acceso globales se usan rara vez ya que requieren mucho espacio. Las listas de control de acceso hacen que sea rápido responder la pregunta "¿quién tiene acceso a este objeto?", mientras que las listas de capacidad hacen que sea rápido responder a la pregunta "¿a cuáles objetos tiene acceso este actor?"

Para el SIARE se decidió usar una lista de control de acceso para validar qué actores pueden acceder a cada clase.

La clase que valida el acceso al sistema verifica que el usuario exista en la base de datos y que el password introducido corresponda con el almacenado en la base de datos. Si el usuario existe se obtiene también su perfil (administrador, líder de proyecto, probador o desarrollador) y se registra el mismo en la sesión del usuario. Cada una de las pantallas validará que el perfil almacenado en la sesión corresponda al perfil autorizado para acceder a la pantalla.

4.3.3.2 Criptografía

En un ambiente en que se comparten recursos entre varios usuarios, la autenticación, por lo general, no es suficiente. En el caso de una red, por ejemplo, es relativamente fácil para un intruso encontrar herramientas para husmear el tráfico de red, incluyendo paquetes generados por otros usuarios. Los protocolos TCP/IP no fueron diseñados pensando en la seguridad: un intruso puede falsificar paquetes que aparezcan como si vinieran de usuarios legítimos. [BRU02e]

El reciente desarrollo de las tecnologías de criptografía está cambiando la situación en la que actualmente se encuentra la seguridad. Utilizadas correctamente, estas tecnologías pueden garantizar la seguridad en las comunicaciones vía Internet. Los protocolos criptográficos pueden aplicarse tanto al transporte de los mensajes como a su formato. [MAR00b]

Secure Sockets Layer

SSL (Secure Sockets Layer o capa de sockets segura) es un estándar definido por Netscape Communications para proteger las conexiones HTTP. Desde que Netscape lo implementara en su navegador, SSL se ha convertido en el estándar *de facto* para garantizar la seguridad en las conexiones HTTP. Hoy en día, prácticamente todos los navegadores y servidores HTTP soportan SSL. SSL es un protocolo seguro en la capa de sesión, por lo que puede proporcionar seguridad de punto a punto.

De forma general, la seguridad en la mensajería incluye cuatro aspectos importantes:

- Confidencialidad. El contenido del mensaje no puede ser controlado o copiado por una entidad no autorizada.
- Integridad. El contenido del mensaje no puede ser alterado por una entidad no autorizada.
- Autenticación. Nadie puede hacerse pasar por la persona a la que va dirigida el mensaje.
- Total aceptación. El remitente del mensaje no puede negar el hecho de haber enviado el mensaje ni el contenido que ha incluido en el mismo.

SSL ofrece los tres primeros tipos de protección: confidencialidad, integridad y autenticación. La confidencialidad en los mensajes se garantiza mediante el uso de un sistema simétrico de encripción como:

- DES (Data Encryption Standard o estándar de encripción de datos), que incluye una clave de 56 bits, y
- RC-4 o Código de Ron, que desarrolló el criptógrafo Ron Rivest. La longitud de su clave varía, pero suele incluir entre 40 y 128 bits.

La integridad se garantiza utilizando un código de autenticación de mensajes (MAC) basado en una función segura de dispersión como MD5 (Message Digest 5 o compendio 5 de mensaje) y SHA 1 (Secure Hash Algorithm 1 o algoritmo seguro de dispersión 1).

La autenticación del cliente es optativa en una conexión SSL, pero la del servidor es obligatoria. Para utilizar SSL, un sitio Web tiene que adquirir un certificado digital de servidor a una autoridad certificadora (CA), una organización independiente que emite certificados digitales. Un certificado

digital garantiza que la clave pública que contiene pertenece al propietario de la misma, lo que permite al destinatario de un mensaje firmado digitalmente comprobar la identidad de quien firma el mensaje. El formato de certificado digital que utiliza SSL es X.509, definido por la ITU-T (Unión Internacional de Telecomunicaciones - Sector Estándares de Telecomunicaciones).

Implementar aplicaciones Web basadas en SSL

Puesto que la mayoría de navegadores y servidores Web soportan SSL, puede decirse que ya están implementados los mecanismos básicos para construir una aplicación Web basada en SSL.
La ventaja de los servidores Web con soporte para SSL es que puede usarse la misma aplicación tanto para la conexión HTTP (puerto 80) como para la conexión HTTPS (puerto 443). Para activar SSL sólo es necesario volver a configurar el servidor Web. En este proceso de reconfiguración debe obtenerse e instalarse un certificado de servidor.

Dadas sus características, se decidió implementar SSL para la construcción del SIARE.

4.3.4 Manejo de excepciones

En general, una excepción es un evento inesperado o error que sucede durante la ejecución del sistema. [BRU02e]

Las excepciones son causadas por una de tres fuentes diferentes:

- Un *error de usuario*. El usuario, por error o de manera deliberada, introduce datos que están fuera de los límites. El SIARE contará con validaciones en los campos de captura libre y con catálogos predefinidos de entrada de datos para minimizar la posibilidad de errores generados por el usuario.
- Una *falla de hardware*. El hardware envejece y falla. La falla de un vínculo de red, por ejemplo, puede desconectar en forma momentánea a dos nodos del sistema. Una falla de disco duro puede dar lugar a la pérdida permanente de datos.
- Un *error de software*. Un error puede ocurrir debido a que el sistema o alguno de sus componentes contiene un error de diseño. Aunque es difícil la escritura de software libre de errores, los subsistemas individuales pueden anticipar errores de otros subsistemas y protegerse contra ellos. Los subsistemas del SIARE que interactúan con componentes externos (base de datos relacional y servicio de correo electrónico) validarán la disponibilidad de los componentes externos para que, en caso de que no estén funcionando, no se envíen transacciones a ellos y se despliegue un mensaje informativo al usuario.

Dado que el lenguaje PHP no implementa directamente un manejo de excepciones, cada clase tendrá una propiedad que indicará si la última operación realizada fue exitosa o errónea, y en cualquier momento se podrá obtener el valor de esa propiedad para determinar si ocurrió una excepción o no.

4.4 Componentes del sistema

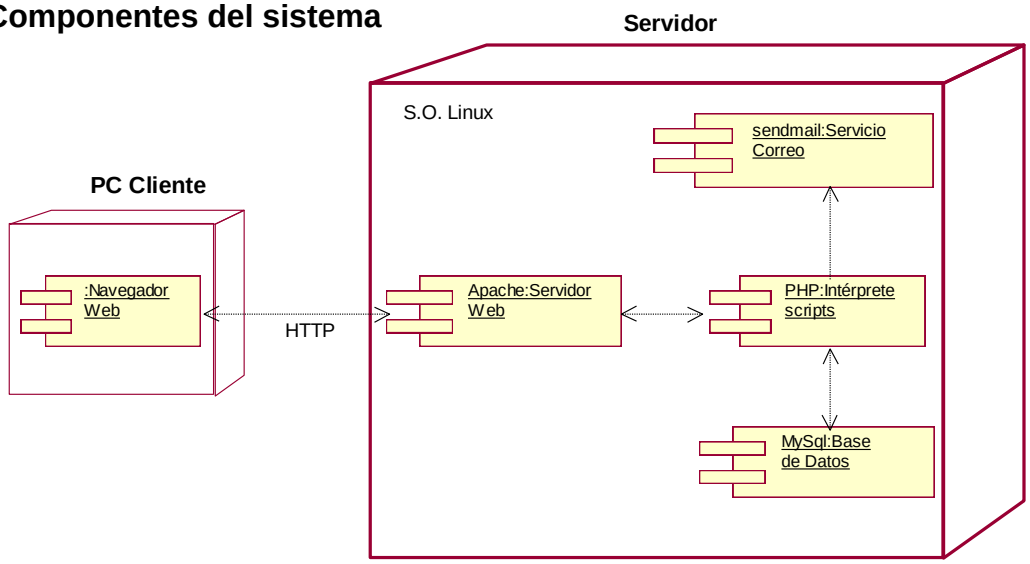


Figura 4-1 Diagrama de despliegue y de componentes para el SIARE.

La selección de los componentes del SIARE se basó en dos criterios:

- Costo de los componentes.
- Facilidad para instalar los componentes en distintas plataformas de hardware.

Linux	Es un sistema operativo semejante a UNIX que fue diseñado para proveer a los usuarios de computadoras personales un sistema operativo de bajo costo comparable a los sistemas tradicionales UNIX. Linux tiene reputación de eficiente y rápido desempeño. Linux es un sistema operativo completo que incluye una interfaz gráfica de usuario, TCP/IP y otros componentes frecuentemente encontrados en los sistemas UNIX. Dado que Linux cumple con la Interfaz Estándar de Sistemas Operativos Portables, los desarrolladores pueden escribir aplicaciones portables a otros sistemas operativos. Existen versiones de Linux para la mayor parte de las arquitecturas de microprocesadores como Intel, Power PC, Sparc y Alpha.

MySql	Es un sistema de gestión de bases de datos relacional (RDBMS) que usa el lenguaje de consultas estructurado (SQL), el lenguaje más popular para agregar, acceder y procesar datos en una base de datos. Dado que se ofrece bajo una licencia de código abierto, cualquiera puede obtener una copia de MySql y ajustarlo a sus necesidades de acuerdo a la licencia de uso público. MySql destaca por su velocidad, confiabilidad y flexibilidad. Tiene un soporte completo a *multi-threading*; provee interfaces de programas de aplicación (APIs) para C, C++, Eiffel, Java, Perl, PHP, Python y Tcl. MySql corre actualmente en plataformas Linux, UNIX y Windows. La versión que se utilizó en el SIARE es la 3.23.53. [MYSQL]
Sendmail	Es la implementación más popular basada en UNIX del protocolo simple de transferencia de correo (SMTP). Cuando un servidor de sendmail recibe un correo electrónico, intenta enviarlo inmediatamente al destinatario, y si el destinatario no está presente encola los mensajes para enviarlos posteriormente.
PHP	En la programación Web, PHP es un lenguaje de scripting y un intérprete que está disponible gratuitamente, usado principalmente en servidores Web basados en Linux. Los scripts de PHP están contenidos dentro de una página Web junto con el código HTML. Antes de que la página se envíe al usuario que la solicitó, el servidor Web llama al intérprete de PHP y ejecuta las operaciones invocadas en el script de éste. Se puede usar PHP para generar páginas HTML dinámicas, dado que el contenido variará según los resultados de la interpretación del script. PHP es gratis y se ofrece bajo una licencia de código abierto. La versión que se utilizó en el sistema es la 4.1.2.[PHP]
Apache	Es un servidor Web disponible gratuitamente que se ofrece bajo la licencia de código abierto. Corre en la mayor parte de los sistemas operativos basados en UNIX (como Linux, Solaris, Digital UNIX y AIX), así como Windows. De acuerdo a la encuesta de NetCraft de febrero de 2001, 60% de los sitios Web de todo Internet usan Apache. Apache cumple con la especificación más reciente de HTTP (1.1). HTTP (protocolo de transferencia de hipertexto) es un conjunto de reglas para intercambiar archivos (texto, imágenes, sonido, video, etc.) en Internet, se apoya en el conjunto de protocolos TCP/IP. Los conceptos esenciales de HTTP incluyen la idea de que los archivos pueden contener referencias a otros archivos, cuya selección implicará solicitudes de transferencias adicionales. Todo servidor Web contiene, además de archivos, un proceso diseñado para esperar las solicitudes HTTP y para manejarlas.

	La versión de Apache usada en el SIARE es la 1.3.23. [APACHE]
Navegador Web	Un navegador es un programa que provee una forma de ver e interactuar con la información en el World Wide Web.

4.5 Descomposición en subsistemas

Encontrar subsistemas durante el diseño del sistema tiene muchas similitudes con encontrar objetos durante el análisis: es una actividad volátil manejada por heurística. La descomposición en subsistemas se revisa en forma constante siempre que se tratan temas nuevos: los subsistemas se combinan en un subsistema, un subsistema complejo se divide en partes y se añaden algunos subsistemas para que se encarguen de nueva funcionalidad. [BRU02e]

La descomposición en subsistemas reduce la complejidad del dominio de solución minimizando las dependencias entre clases. El objetivo de la descomposición es reducir en forma efectiva el acoplamiento entre subsistemas.

Los subsistemas que se identifican durante la descomposición inicial en subsistemas resultan, con frecuencia, del agrupamiento de varias clases relacionadas desde el punto de vista funcional.

Subsistema administrador	Agrupa los objetos de frontera y de control que proveen la funcionalidad para administrar las solicitudes de inscripción al sistema y para autorizar una solicitud de inscripción.
Subsistema líder de proyecto	Agrupa los objetos de frontera y de control que proveen la funcionalidad para dar de alta una nueva solicitud de inscripción, dar mantenimiento a los catálogos de productos, probadores y desarrolladores, administración de reportes de error, asignación de un desarrollador para dar seguimiento a un reporte y rechazo de un reporte.
Subsistema probador	Agrupa los objetos de frontera y de control que proveen la funcionalidad para reportar un error, revisar las respuestas para un reporte y para cerrar un reporte.
Subsistema desarrollador	Agrupa los objetos de frontera y de control que proveen la funcionalidad para administrar los reportes a los que se debe dar seguimiento y para responder un reporte.

Subsistema seguridad	Agrupa los objetos de frontera y de control que permiten validar el acceso de un usuario al sistema.
Subsistema solicitud	Agrupa los objetos de entidad relacionados con una solicitud de inscripción al sistema.
Subsistema actores	Agrupa los objetos de entidad correspondientes a los actores involucrados en el sistema.
Subsistema reportes	Agrupa los objetos de entidad relacionados con los reportes de errores.
Subsistema base de datos	Agrupa las clases encargadas de la comunicación entre el intérprete de scripts PHP y el administrador de bases relacionales MySql. Incluye clases que mantienen los parámetros de conexión con la base de datos (ParametrosConexion), administrar las conexiones de una base de datos (Conexion), ejecutar operaciones sobre la base de datos (Query), mantener los resultados de una consulta (Resultset), y paginar los resultados de una consulta (Pagina y Lista).
Subsistema mensajería	Agrupa las clases encargadas de comunicar el intérprete de scripts PHP y el servicio de correo sendmail, que implementa del SMTP, usado para el envío de correo electrónico.
Subsistema Estadísticas	Agrupa las clases de entidad usadas para generar los reportes estadísticos definidos en la sección 2.6.
Subsistema utilerías	Agrupa las siguientes clases: Date: Manejo de fechas. FormateoTexto: Formatea cadenas de caracteres. Link: Facilita la creación de hipervínculos (links) en HTML. Redireccion: Permite controlar el flujo al navegar entre páginas Web. Serializacion: Transforma un objeto en un arreglo de bytes y viceversa. Status: Estructura de datos utilizada para indicar el resultado de una transacción. Url: Modela un URL (Uniform Resource Locator) formado por una dirección y una lista de parámetros. Vector: Modela una estructura de datos de lista.

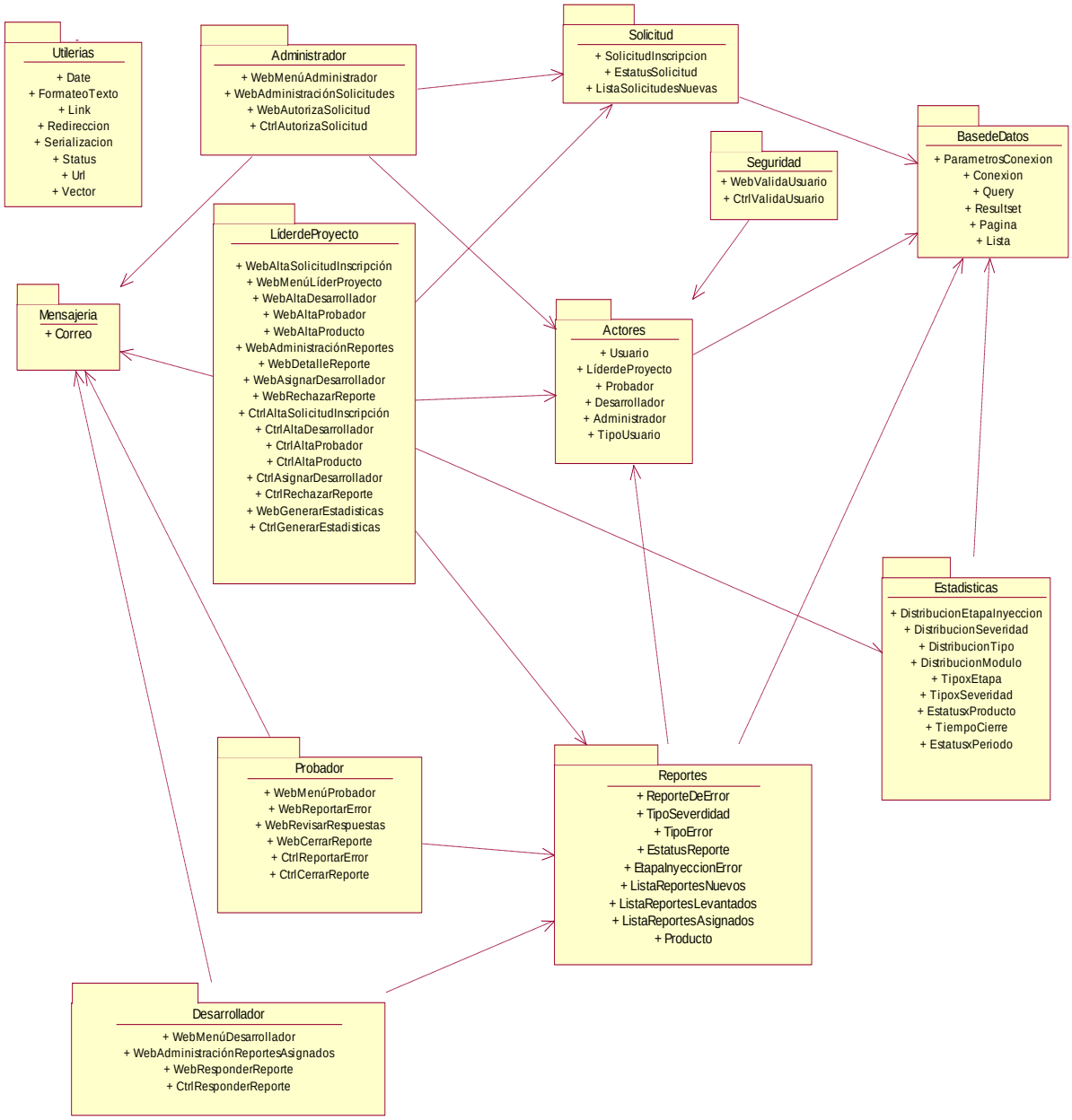


Figura 4-2 Diagrama de paquetes para los subsistemas del SIARE.

Capítulo 5

Diseño detallado

Durante el diseño del sistema se describe éste desde el punto de vista de su arquitectura, como su descomposición en subsistemas, su flujo de control global y la administración de la persistencia. Durante el diseño del sistema también se define la plataforma de hardware y software sobre la que se construirá. Durante el diseño detallado se cierra el "hueco" entre la arquitectura de software y los componentes a construir.

El diseño detallado incluye las siguientes actividades:

- Diseño del modelo de datos.
- Diseño de objetos.

5.1 Diseño del modelo de datos

El almacenamiento de datos es considerado por algunos como la parte medular de los sistemas de información. En primer lugar, los datos tienen que estar disponibles cuando el usuario quiere usarlos. Segundo, los datos deben ser precisos y consistentes (deben poseer integridad). Además, los objetivos de diseño de la base de datos incluyen el almacenamiento eficiente de los datos, así como su eficiente actualización y recuperación. Por último, es necesario que la recuperación de la información tenga un propósito. La información obtenida de los datos almacenados debe estar en un formato útil para la administración, planeación, control o toma de decisiones. [KEN97]

Tal como se especificó en el punto 4.2, el SIARE usará una base de datos relacional como almacenamiento de datos persistentes.

Una estructura de datos relacional consiste en tablas de dos dimensiones llamadas entidades. Los renglones de la tabla representan los registros y las columnas contienen atributos. Las relaciones son asociaciones entre entidades.

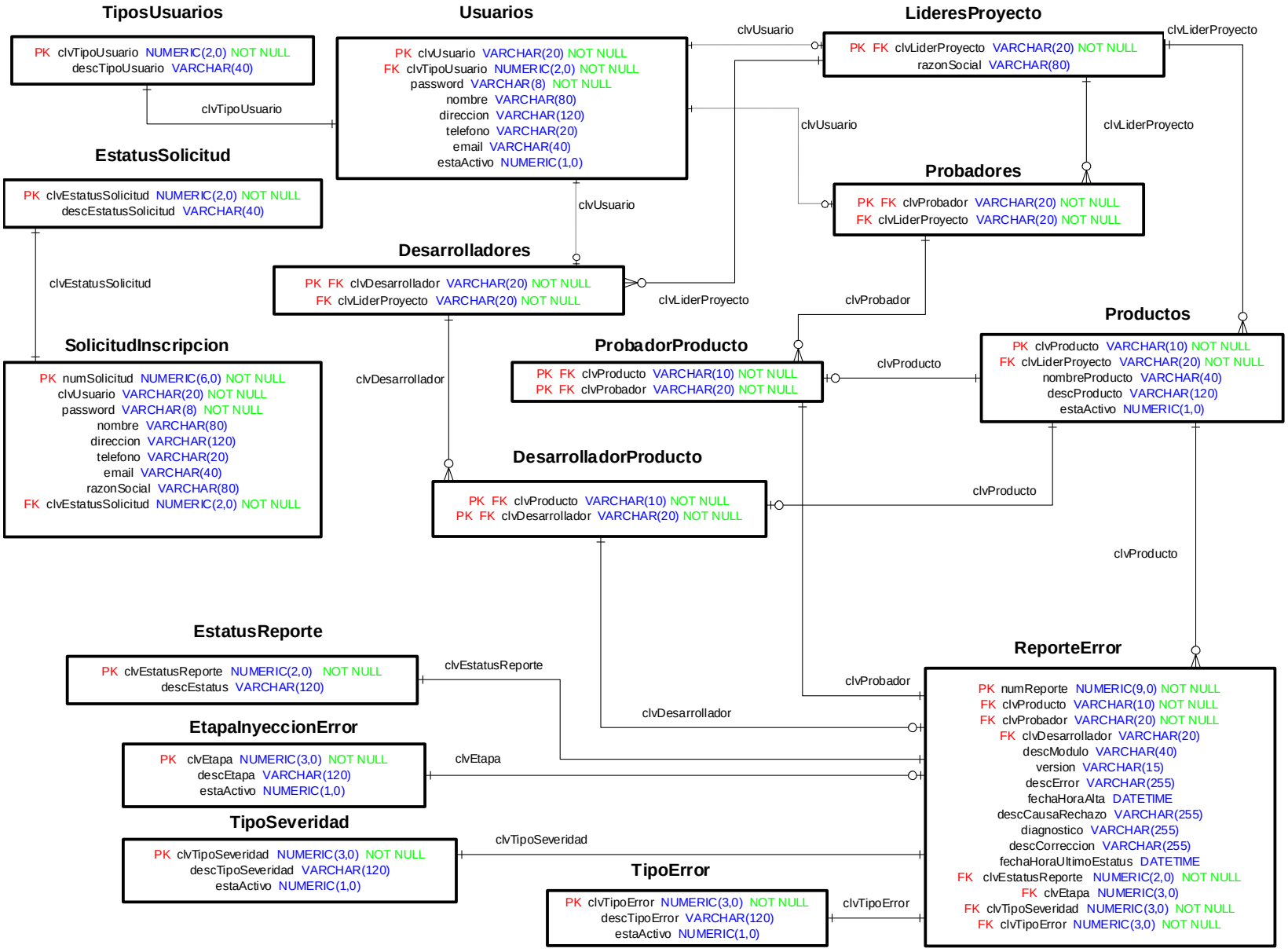


Figura 5-1 Diagrama entidad - relación para la base de datos del SIARE.

5.1.1 Diccionario de datos

TiposUsuarios

Catálogo de tipos de usuarios válidos en el sistema.

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
clvTipoUsuario	Numérico	2 enteros, 0 decimales	No	Clave para el tipo de usuario
descTipoUsuario	Alfanumérico	40	Sí	Descripción del tipo de usuario

Llave(s) primaria(s): *clvTipoUsuario.*

Llave(s) foránea(s): Ninguna.

Valores:

clvTipoUsuario	descTipoUsuario
1	Administrador
2	Líder de proyecto
3	Probador
4	Desarrollador

EstatusSolicitud

Catálogo de estatus para una solicitud de inscripción al sistema.

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
clvEstatusSolicitud	Numérico	2 enteros, 0 decimales	No	Clave para el estatus de la solicitud
descEstatusSolicitud	Alfanumérico	40	Sí	Descripción del estatus de la solicitud

Llave(s) primaria(s): *clvEstatusSolicitud.*

Llave(s) foránea(s): Ninguna.

Valores:

clvEstatusSolicitud	descEstatusSolicitud
1	Nueva
2	Autorizada
3	Rechazada

SolicitudInscripcion

Almacena las solicitudes de inscripción al sistema, dadas de alta por el líder de proyecto.

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
numSolicitud	Numérico	6 enteros, 0 decimales	No	Número consecutivo que identifica las solicitudes
clvUsuario	Alfanumérico	20	No	Clave de usuario introducida por el líder de proyecto al dar de alta la solicitud
password	Alfanumérico	8	No	Password seleccionado por el líder de proyecto
nombre	Alfanumérico	80	Sí	Nombre del líder de proyecto
direccion	Alfanumérico	120	Sí	Dirección del líder de proyecto
telefono	Alfanumérico	20	Sí	Teléfono del líder de proyecto
email	Alfanumérico	40	Sí	Email del líder de proyecto
razonSocial	Alfanumérico	80	Sí	Razón social de la organización donde trabaja el líder de proyecto
clvEstatusSolicitud	Numérico	2 enteros, 0 decimales	No	Clave del estatus de la solicitud

Llave(s) primaria(s): *numSolicitud.*

Llave(s) foránea(s):

- *clvEstatusSolicitud* hace referencia a *clvEstatusSolicitud* de la tabla *EstatusSolicitud.*

Nota: Esta tabla se relaciona únicamente con el catálogo de estatus de solicitud. Cuando se autoriza una solicitud de inscripción se copian los datos a la tabla de Usuarios. De esta forma se logra que para las solicitudes rechazadas no se genere un registro.

Usuarios

Almacena la información de todos los usuarios del sistema (administradores, líderes de proyecto, probadores y desarrolladores).

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
clvUsuario	Alfanumérico	20	No	Clave de usuario
clvTipoUsuario	Numérico	2 enteros, 0 decimales	No	Tipo de usuario
password	Alfanumérico	8	No	Password del usuario
nombre	Alfanumérico	80	Sí	Nombre del usuario
direccion	Alfanumérico	120	Sí	Dirección del usuario
telefono	Alfanumérico	20	Sí	Teléfono del usuario
email	Alfanumérico	40	Sí	Email del usuario
estaActivo	Numérico	1 enteros, 0 decimales	Sí	Indica si usuario está activo (1) o inactivo (0)

Llave(s) primaria(s): *clvUsuario.*

Llave(s) foránea(s):

- *clvTipoUsuario* hace referencia a *clvTipoUsuario* de la tabla *TiposUsuarios.*

LideresProyecto

Almacena la información específica de los líderes de proyecto.

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
clvLiderProyecto	Alfanumérico	20	No	Clave de usuario introducida por el líder de proyecto
razonSocial	Alfanumérico	80	Sí	Razón social de la organización donde trabaja el líder de proyecto

Llave(s) primaria(s): *clvLiderProyecto.*

Llave(s) foránea(s):

- *clvLiderProyecto* hace referencia a *clvUsuario* de la tabla *Usuarios.*

Probadores

Almacena la información específica de los probadores.

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
clvProbador	Alfanumérico	20	No	Clave de usuario del probador
clvLiderProyecto	Alfanumérico	20	Sí	Clave del líder de proyecto que dio de alta al probador

Llave(s) primaria(s): *clvProbador.*

Llave(s) foránea(s):

- *clvProbador* hace referencia a *clvUsuario* de la tabla *Usuarios.*
- *clvLiderProyecto* hace referencia a *clvLiderProyecto* de la tabla *LideresProyecto.*

Desarrolladores

Almacena la información específica de los desarrolladores.

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
clvDesarrollador	Alfanumérico	20	No	Clave de usuario del desarrollador
clvLiderProyecto	Alfanumérico	20	No	Clave del líder de proyecto que dio de alta al desarrollador

Llave(s) primaria(s): *clvDesarrollador.*

Llave(s) foránea(s):

- *clvDesarrollador* hace referencia a *clvUsuario* de la tabla *Usuarios.*
- *clvLiderProyecto* hace referencia a *clvLiderProyecto* de la tabla *LideresProyecto.*

Productos

Almacena la información de un producto.

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
clvProducto	Alfanumérico	10	No	Clave del producto
clvLiderProyecto	Alfanumérico	20	No	Clave del líder de proyecto que dio de alta el producto
nombreProducto	Alfanumérico	40	Sí	Nombre del producto
descProducto	Alfanumérico	120	Sí	Descripción del producto
estaActivo	Numérico	1 enteros, 0 decimales	Sí	Indica si el producto está activo (1) o inactivo (0)

Llave(s) primaria(s): *clvProducto.*

Llave(s) foránea(s):

- *clvLiderProyecto* hace referencia a *clvLiderProyecto* de la tabla *LideresProyecto.*

ProbadorProducto

Almacena la relación entre un producto y los probadores que pueden validarlo.

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
clvProducto	Alfanumérico	10	No	Clave del producto
clvProbador	Alfanumérico	20	No	Clave del probador que puede validar el producto

Llave(s) primaria(s): *clvProducto, clvProbador.*

Llave(s) foránea(s):

- *clvProducto* hace referencia a *clvProducto* de la tabla *Productos.*
- *clvProbador* hace referencia a *clvProbador* de la tabla *Probadores.*

DesarrolladorProducto

Almacena la relación entre un producto y los desarrolladores que pueden corregirlo.

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
clvProducto	Alfanumérico	10	No	Clave del producto
clvDesarrollador	Alfanumérico	20	No	Clave del desarrollador que puede corregir el producto

Llave(s) primaria(s): *clvProducto, clvDesarrollador.*

Llave(s) foránea(s):

- *clvProducto* hace referencia a *clvProducto* de la tabla *Productos.*
- *clvDesarrollador* hace referencia a *clvDesarrollador* de la tabla *Desarrolladores.*

EstatusReporte

Catálogo con los estatus posibles para un reporte de error.

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
clvEstatusReporte	Numérico	2 enteros, 0 decimales	No	Clave del estatus del reporte
descEstatus	Alfanumérico	120	Sí	Descripción del estatus del reporte

Llave(s) primaria(s): *clvEstatusReporte.*

Llave(s) foránea(s): Ninguna.

Valores:

clvEstatusReporte	descEstatus
1	Nuevo
2	Rechazado
3	Asignado para seguimiento
4	Cerrado

EtapaInyeccionError

Catálogo con las etapas de inyección de error posibles.

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
clvEtapa	Numérico	3 enteros, 0 decimales	No	Clave de la etapa de inyección del error
descEtapa	Alfanumérico	120	Sí	Descripción de la etapa de inyección del error
estaActivo	Numérico	1 enteros, 0 decimales	Sí	Indica si la etapa de inyección esta activa (1) o inactiva (0)

Llave(s) primaria(s): *clvEtapa.*

Llave(s) foránea(s): Ninguna.

Valores:

clvEtapa	descEtapa
1	Recopilación de requerimientos
2	Análisis
3	Diseño de alto nivel
4	Diseño detallado
5	Codificación
6	Pruebas unitarias
7	Pruebas de integración
8	Pruebas de aceptación

TipoSeveridad

Catálogo con los tipos de severidad para un error.

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
clvTipoSeveridad	Numérico	3 enteros, 0 decimales	No	Clave del tipo de severidad del error
descTipoSeveridad	Alfanumérico	120	Sí	Descripción del tipo de severidad del error
estaActivo	Numérico	1 enteros, 0 decimales	Sí	Indica si el tipo de severidad del error está activo (1) o inactivo (0)

Llave(s) primaria(s): *clvTipoSeveridad.*

Llave(s) foránea(s): Ninguna.

Valores:

clvTipoSeveridad	descTipoSeveridad
1	Crítico
2	Mayor
3	Menor
4	Cosmético

TipoError

Catálogo con los tipos de error.

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
clvTipoError	Numérico	3 enteros, 0 decimales	No	Clave del tipo de error
descTipoError	Alfanumérico	120	Sí	Descripción del tipo de error
estaActivo	Numérico	1 enteros, 0 decimales	Sí	Indica si el tipo de error está activo (1) o inactivo (0)

Llave(s) primaria(s): *clvTipoError.*

Llave(s) foránea(s): Ninguna.

Valores:

clvTipoError	descTipoError
10	Documentación
20	Sintaxis
30	Paquete (control de versiones)
40	Asignación
50	Interfaz
60	Validaciones
70	Datos
80	Funciones
90	Sistema
100	Ambiente

ReporteError

Almacena los reportes de error.

Detalle de campos:

Nombre	Tipo	Longitud	Nulo	Descripción
numReporte	Numérico	9 enteros, 0 decimales	No	Número consecutivo del reporte
clvProducto	Alfanumérico	10	No	Producto del que se reporta el error
clvProbador	Alfanumérico	20	No	Probador que reporta el error
clvDesarrollador	Alfanumérico	20	Sí	Desarrollador asignado a dar seguimiento al reporte
descModulo	Alfanumérico	40	Sí	Módulo del producto donde se encontró el error
version	Alfanumérico	15	Sí	Versión del producto donde se encontró el error
descError	Alfanumérico	255	Sí	Descripción del error encontrado
fechaHoraAlta	DateTime	-	Sí	Fecha y hora de alta del reporte
descCausaRechazo	Alfanumérico	255	Sí	Causa del rechazo de un reporte (sólo en caso que estatus reporte = rechazado)
diagnostico	Alfanumérico	255	Sí	Diagnóstico del error
descCorreccion	Alfanumérico	255	Sí	Descripción de la corrección del error
fechaHoraUltimoEstatus	DateTime	-	Sí	Fecha y hora del último cambio de estatus
clvEstatusReporte	Numérico	2 enteros, 0 decimales	No	Clave del estatus del reporte
clvEtapa	Numérico	3 enteros, 0 decimales	Sí	Clave de la etapa de inyección del error
clvTipoSeveridad	Numérico	3 enteros, 0 decimales	No	Clave del tipo de severidad del error
clvTipoError	Numérico	3 enteros, 0 decimales	No	Clave del tipo de error

Llave(s) primaria(s): *numReporte.*

Llave(s) foránea(s):

- *clvProducto* hace referencia a *clvProducto* de la tabla *Productos.*

- *clvProbador* hace referencia a *clvProbador* de la tabla *ProbadorProducto.*
- *clvDesarrollador* hace referencia a *clvDesarrollador* de la tabla *DesarrolladorProducto.*
- *clvEstatusReporte* hace referencia a *clvEstatusReporte* de la tabla *EstatusReporte.*
- *clvEtapa* hace referencia a *clvEtapa* de la tabla *EtapaInyeccionError.*
- *clvTipoSeveridad* hace referencia a *clvTipoSeveridad* de la tabla *TipoSeveridad.*
- *clvTipoError* hace referencia a *clvTipoError* de la tabla *TipoError.*

5.2 Diseño de objetos

Durante el diseño de objetos se refinan los modelos de análisis y de diseño del sistema. Asimismo, se especifican los servicios de subsistemas (identificados durante el diseño del sistema) desde el punto de vista de interfaces de clase, incluyendo operaciones, argumentos, firmas de tipo y excepciones. Durante esta actividad también se encuentran operaciones faltantes y objetos necesarios para transferir datos entre subsistemas. El resultado de la especificación de servicios es una especificación de interfaz completa para cada subsistema. A la especificación de servicios de subsistema con frecuencia se le llama API (interfaz de programación de aplicaciones, por sus siglas en inglés). [BRU02f]

A continuación se resume la especificación de objetos de entidad (agrupados en los paquetes *Solicitud, Actores* y *Reportes*) así como de los objetos de infraestructura (reunidos en los paquetes *BasedeDatos, Mensajeria, Utilerias*) empleados en el SIARE. Se omitieron los objetos de frontera y de control (agrupados en los paquetes *Administrador, LiderdeProyecto, Probador, Desarrollador* y *Seguridad*) ya que sus características han sido especificadas en la identificación de objetos de frontera y control (secciones 3.1.2 y 3.1.3), así como en los diagramas de secuencia (sección 3.2). La especificación completa de API puede encontrarse en el anexo A.

Nota: Debido a restricciones del lenguaje PHP utilizado para implementar el SIARE, en los nombres de paquetes, clases, métodos, parámetros y constantes se omitieron los acentos. Los nombres de paquetes y clases inician siempre con mayúsculas. Los nombres de métodos y parámetros inician siempre con minúsculas. En caso de que cualquier nombre esté formado por más de una palabra, se escribe sin espacios y con la letra inicial de cada palabra en mayúscula, para facilitar la lectura. Los nombres de constantes se escriben siempre con mayúsculas.

5.2.1 Resumen de paquetes

Utilerias

Este paquete agrupa las clases de utilerías usadas en todo el sistema.

Sumario de clases:

Status	Modela la estructura de datos usada por todas las transacciones para indicar si fueron exitosas o fallidas.
Date	Modela el manejo de fechas.
Url	Modela la estructura de un URL (Uniform Resource Locator). El URL es una cadena que permite localizar cualquier recurso disponible en internet, tiene el siguiente formato: *<protocolo>://<dirección>/<recurso>* En caso de que el recurso sea una página Web, es posible además especificar un conjunto de parámetros con los que se desea llamar a la página: *<protocolo>://<dirección>/<recurso>?parámetro1=valor1&parámetro2=valor2&...*
Link	Modela la creación de una liga (link) en HTML. Para generar la liga necesita de un URL, por lo que esta clase extiende a la clase Url. Además necesita de un texto que aparecerá en la página Web. Crea una cadena con el siguiente formato: <A HREF=*url*>*texto*</A>
Redireccion	Permite llamar automáticamente a una página Web desde otra, usando instrucciones HTML. Para llamar a otra página requiere de un URL, por lo que esta clase extiende a la clase Url. Necesita además del tiempo en milisegundos que esperará para llamar a la otra página. Crea una cadena con el siguiente formato: <meta http-equiv='Refresh' content=*tiempo*;URL=*url*>
FormateoTexto	Agrupa funciones para dar formato a cadenas de caracteres.
Vector	Modela una estructura de datos de lista, donde se puede agregar cualquier tipo de objeto.
Serializador	Contiene métodos que permiten convertir un objeto a un arreglo de bytes y viceversa. Esto resulta útil cuando se desea almacenar y recuperar un objeto como variable de sesión.

BaseDatos

Este paquete agrupa las clases relacionadas con la conexión entre PHP y el administrador de bases de datos relacional MySql.

Sumario de clases:

ParametrosConexion	Contiene los valores de los parámetros por omisión para la conexión entre PHP y MySql. Para conectar PHP con MySql es necesario contar con un nombre de servidor donde se ubica la base de datos a la que se desea conectar, nombre de la base de datos, usuario de la base de datos y password del usuario.
Conexion	Administra una conexión entre PHP y MySql. Lleva a cabo las operaciones de conexión y desconexión con la base de datos. Toma por omisión los parámetros definidos en la clase *ParametrosConexion*. Usa la clase *Status* para indicar si las operaciones de conexión y desconexión se efectuaron correctamente o generaron un error.
Query	Permite ejecutar operaciones de altas, bajas, cambios, consultas y actualizaciones sobre una base de datos MySql. Recibe la operación en forma de una sentencia SQL. Además permite bloquear y desbloquear tablas para escritura (este mecanismo sirve para que durante una transacción, una tabla no pueda ser afectada por otra transacción hasta que se desbloquee ésta). Contiene además un método que permite generar un número consecutivo a partir del valor más grande que encuentre en un campo de una tabla especificada. Usa la clase *Status* para indicar el éxito o fracaso de una operación. Devuelve los resultados de una consulta mediante la clase *Resultset*. Para indicar sobre qué base de datos ejecuta la operación, recibe una instancia de la clase *Conexion*.
Resultset	Modela la estructura de datos que almacena los resultados de una operación sobre la base de datos. Los resultados se almacenan en un arreglo en que el primer índice hace referencia al registro y el segundo al campo que se desea obtener. Cada vez que se obtiene un registro, avanza el primer índice. Almacena también el número de registros afectados por la operación.
Pagina	Modela una estructura de datos que permite organizar los registros obtenidos como resultado de una consulta en forma de páginas, de tamaño parametrizable. Al organizar los registros así es posible desplegarlos página por página en el front-end, evitando formar páginas

	HTML muy extensas.
Lista	Permite ejecutar una consulta sobre la base de datos y paginar los resultados para que, cuando se desplieguen en el front-end, no se forme una página muy extensa de HTML, sino se formen páginas de tamaño parametrizable de registros. Esta clase ejecuta la consulta que recibe en el constructor y llena objetos tipo *Pagina* con los registros obtenidos. Mediante un apuntador se le indica si se desea obtener la siguiente página de registros, o retroceder.

Mensajeria

Este paquete agrupa las clases relacionadas con el envío de mensajes.

Sumario de clases:

Correo	Permite el envío de correos electrónicos.

Solicitud

Este paquete agrupa las clases relacionadas con el manejo de las solicitudes de inscripción al SIARE.

Sumario de clases:

SolicitudInscripcion	Modela la entidad correspondiente a una solicitud de inscripción al SIARE. Sus propiedades corresponden a los campos de la tabla *SolicitudInscripcion* de la base de datos. Contiene métodos que permiten actualizar, buscar y dar de alta una solicitud.
EstatusSolicitud	Modela los distintos estatus válidos para una solicitud de inscripción al sistema.
ListaSolicitudesNuevas	Ejecuta una consulta sobre la tabla *SolicitudInscripcion* de la base de datos para buscar todas las solicitudes nuevas.

Actores

Este paquete agrupa las clases relacionadas a los actores del sistema.

Sumario de clases:

TipoUsuario	Modela los tipos de usuario válidos en el sistema.
Usuario	Modela la entidad correspondiente a las características comunes a todos los usuarios del sistema.
LiderdeProyecto	Modela la entidad correspondiente a las características específicas del líder de proyecto.
Probador	Modela la entidad correspondiente a las características específicas del probador.
Desarrollador	Modela la entidad correspondiente a las características específicas del desarrollador.
Administrador	Modela la entidad correspondiente a las características específicas del administrador.

Reportes

Este paquete agrupa las clases relacionadas con los reportes de errores.

Sumario de clases:

EstatusReporte	Modela las propiedades que describen el estatus de un reporte.
TipoError	Modela las propiedades que describen el tipo de error.
TipoSeveridad	Modela las propiedades que describen el tipo de severidad de un error.
EtapaInyeccionError	Modela las propiedades que describen la etapa de inyección de un error.
Producto	Modela las propiedades de un producto de software a probar.
ReporteDeError	Modela las propiedades de un reporte de error.
ListaReportesNuevos	Ejecuta una consulta sobre la tabla *ReporteError* de la base de datos para buscar todos los reportes nuevos para los productos dados de alta por un líder de proyecto.
ListaReportesLevantados	Ejecuta una consulta sobre la tabla *ReporteError* de la base de datos para buscar todos los reportes levantados por un probador, que se les haya asignado un desarrollador para darles seguimiento.
ListaReportesAsignados	Ejecuta una consulta sobre la tabla *ReporteError* de la base de datos para buscar todos los reportes asignados a un desarrollador, que no hayan sido cerrados.

Estadisticas

Este paquete agrupa las clases relacionadas con la generación de estadísticas sobre los reportes de errores.

Sumario de clases:

DistribucionEtapaInyeccion	Obtiene los datos necesarios para generar el reporte de distribución de defectos por etapa de inyección (sección 2.6.1).
DistribucionSeveridad	Obtiene los datos necesarios para generar el reporte de distribución de defectos por severidad (sección 2.6.2).
DistribucionTipo	Obtiene los datos necesarios para generar el reporte de distribución de defectos por tipo (sección 2.6.3).
DistribucionModulo	Obtiene los datos necesarios para generar el reporte de distribución de defectos por módulo (sección 2.6.4).
TipoxEtapa	Obtiene los datos necesarios para generar el reporte de tipos de defecto por etapa de inyección (sección 2.6.5).
TipoxSeveridad	Obtiene los datos necesarios para generar el reporte de tipos de defecto por severidad (sección 2.6.6).
EstatusxProducto	Obtiene los datos necesarios para generar el reporte de estatus de los reportes por producto (sección 2.6.7).
TiempoCierre	Obtiene los datos necesarios para generar el reporte de distribución de reportes por tiempo para su cierre (sección 2.6.8).
EstatusxPeriodo	Obtiene los datos necesarios para generar el reporte de reportes abiertos y cerrados por periodo (sección 2.6.9).

Capítulo 6

Resultados y conclusiones

En los capítulos anteriores se analizaron los requerimientos del SIARE, así como los modelos generados para manejar la complejidad del sistema y para construir el mismo. En este capítulo se analizan los resultados obtenidos de la construcción del sistema enumerando nuevos requerimientos, que han sido identificados durante las pruebas, y que resultaría conveniente incluir en futuras versiones.

6.1 Análisis de Resultados

El fenómeno del software puede ser analizado desde los puntos de vista del producto y del proceso. Dado que esta tesis consiste en un proyecto de desarrollo de software, es necesario analizar los resultados del mismo tanto desde el punto de vista del producto final como del proceso de desarrollo utilizado para llegar a él.

6.1.1 Evaluación del producto

A partir de los modelos descritos en los capítulos 2 a 5, se procedió a codificar la aplicación. Se puede acceder al SIARE desde la dirección http://www.seminario-sqa.com/tesis/index.php.

La evaluación del producto tiene como objetivo responder a la pregunta ¿funciona el sistema?, es decir, se pretende determinar si se resuelven los requerimientos necesarios para que cumpla sus objetivos.

Tal como se mencionó en el punto 1.4, las pruebas de software son un mecanismo de control de calidad que permite localizar, en forma sistemática, errores en un producto con objeto de eliminarlos antes de que éste sea puesto a disposición de los usuarios finales. A continuación, se describe el proceso utilizado para probar el SIARE.

Para determinar la estrategia de pruebas es necesario suponer una descomposición jerárquica de los componentes del sistema, donde cada uno de éstos pertenece a capas ordenadas con

respecto a la asociación "Llama" [BRU02g]. Los componentes que forman el SIARE están organizados lógicamente en tres capas: objetos de entidad, de frontera y de control. Además, existen componentes de infraestructura como los objetos encargados de las operaciones sobre la base de datos y del envío de correo electrónico.

La estrategia elegida para probar el sistema fue la de **abajo hacia arriba**, donde se prueba cada componente de la capa inferior primero de manera individual y luego integrándolo con componentes de la siguiente capa superior. [BRU02g]

La **prueba unitaria** se enfoca en los bloques de construcción del sistema de software; esto es, los objetos y los subsistemas. Hay tres motivaciones para utilizar el enfoque de los componentes. Primero, la prueba unitaria reduce la complejidad de las actividades de prueba generales permitiendo enfocarse en unidades más pequeñas del sistema. Segundo, facilita resaltar y corregir defectos, ya que están involucrados pocos componentes. Tercero, permite el paralelismo en las actividades de prueba, cada componente puede probarse en forma independiente de los demás.

En primer lugar se procedió a probar los componentes relacionados con las operaciones sobre la base de datos y el envío de correo electrónico (estos componentes se encuentran en la capa más interna del sistema). Dado que estos componentes son utilizados en todos los objetos de entidad y en buena parte de los objetos de control, era necesario garantizar su correcto funcionamiento. Cualquier falla provocaría a su vez otras en los componentes que los invocan, dificultando detectar el origen. Posteriormente se probaron los objetos de entidad, después los de control, para finalizar con los objetos de frontera, esto es, la interfaz con el usuario final.

La ejecución de pruebas sobre componentes solos requiere que el componente a probar se aísle del resto del sistema. Los manejadores de prueba simulan la parte del sistema que llama al componente a probar. Un manejador de prueba envía al componente las entradas de prueba identificadas en el análisis del caso de prueba y muestra los resultados. Aunque la programación de los manejadores de prueba implique una inversión de tiempo, ésta es recompensada con un ahorro durante la ejecución de las pruebas integrales del sistema.

La ventaja de las pruebas de abajo hacia arriba es que pueden localizarse con más facilidad los defectos de interfaz: cuando los desarrolladores sustituyen un manejador de prueba por un componente de nivel superior, tienen un modelo claro de la manera en que funciona el componente de nivel inferior y las suposiciones incrustadas en su interfaz. Si el componente de nivel superior viola las suposiciones de nivel inferior, es más probable que los desarrolladores lo encuentren con rapidez. La desventaja de la prueba de abajo hacia arriba es que prueba al último los subsistemas más importantes, a saber, los componentes de la interfaz de usuario. Primero, los defectos que se

encuentran en la capa superior a menudo pueden conducir a cambios en las interfaces de los subsistemas de las capas inferiores. Segundo, las pruebas de la capa superior pueden derivarse del modelo de requerimientos y, por tanto, son más efectivas para la localización de defectos que son visibles ante el usuario.

Considerando lo anterior se realizaron las pruebas del sistema con objeto de asegurar que cumple los requerimientos funcionales y no funcionales.

La prueba funcional es una técnica de caja negra: los casos de prueba se derivan del modelo de casos de uso. A partir de cada flujo de un caso de uso se genera un caso de prueba, que es una especificación de los eventos con que se ejercita un componente con el propósito de causar fallas y detectar defectos. El caso de prueba consiste en precondiciones, flujo de eventos y poscondiciones.

La especificación completa de los casos de prueba del SIARE puede encontrase en el anexo B.

Una vez ejecutados exitosamente todos los casos de prueba anteriormente citados sobre la versión final del SIARE, es posible afirmar que satisface todos los requerimientos funcionales establecidos en la sección 2.2.

Para que un sistema de software cumpla con sus objetivos debe satisfacer además los requerimientos no funcionales. Para probar que se cumplieran los requerimientos no funcionales establecidos en la sección 2.3 y detallados en los objetivos de diseño de la sección 4.1 se ejecutaron las siguientes pruebas:

1. El requerimiento no funcional más importante del SIARE es que se pueda ejecutar en un ambiente distribuido. La información y la lógica se mantienen de forma centralizada en un servidor Web, mientras que los clientes pueden acceder desde diversas localizaciones dispersas, aun en equipos con recursos limitados. Una vez que la versión final del sistema fue instalada en un servidor Web accesible vía internet, fue posible comprobar que se cumple este requerimiento. Los usuarios pudieron acceder al sistema, siempre que contaran con una conexión a internet, ya sea dentro de una red de área local o mediante una conexión *dial-up*. Se puede acceder al sistema usando cualquier equipo con navegador Web que soporte HTML 1.0, y que esté habilitado para ejecutar Javascript. El sistema se probó usando Microsoft Internet Explorer 5.50 (uno de los navegadores ampliamente empleado).

2. En cuanto a la seguridad, la prueba consistió en intentar acceder a módulos del SIARE usando un perfil de usuario distinto al especificado en los casos de uso (secciones 2.4 y 2.5). Por

ejemplo, se intentó acceder a la funcionalidad de un líder de proyecto usando el perfil de desarollador, de probador y de administrador. Todos los módulos del sistema validan el perfil del usuario, si no es el correspondiente, redirecciona al usuario al menú principal.

Con respecto a la encripción, dadas ciertas restricciones en la administración del sitio Web impuestas por el Proveedor de Servicios de Internet (ISP), no se habilitó la encripción usando SSL. Esto no afecta la funcionalidad del SIARE. El habilitar SSL no implica ningún cambio en el sistema.

3. La economía en la construcción del SIARE fue otro de los objetivos principales de diseño. Las herramientas usadas para construir el sistema son gratuitas. La codificación se llevó a cabo en equipos con sistema operativo Windows Me y XP, con las versiones correspondientes del servidor Web Apache, el administrador de base de datos relacionales MySql y el intérprete de PHP. La versión final del sistema se instaló en un servidor proporcionado por un ISP, a un costo de aproximadamente $50 USD anuales. Además de resultar económico, permite a los desarrolladores concentrarse en el análisis, diseño y construcción del sistema, pues facilita en gran medida las tareas de administración e instalación.

4. Finalmente, se comprobó la portabilidad del código fuente al instalar el sistema, que originalmente se programó en un ambiente Windows, en el servidor con Sistema Operativo Linux. Únicamente fue necesario modificar los parámetros de conexión a la base de datos y al servicio de correo electrónico.

Se omitieron las pruebas de estrés y desempeño porque no se contaba con las herramientas necesarias para ejecutarlas. Sin embargo, se observó que los tiempos de respuesta para todas las operaciones realizadas por el sistema están por debajo de 3 segundos, lo cual se puede considerar aceptable para los usuarios finales.

Después de la ejecución satisfactoria de las pruebas funcionales y no funcionales es posible afirmar que el SIARE cumple con los objetivos para los que fue analizado y diseñado. Sin embargo, no hay mejor prueba para un sistema que el ser usado por los usuarios finales en el ambiente destino. Por ese motivo se solicitó a personas dedicadas al desarrollo de software que probaran el SIARE en su ambiente destino. A esto se le conoce como **prueba beta**. Dado que el sistema es accesible mediante internet, fue posible que varios usuarios ejecutaran las pruebas sin necesidad de descargar o instalar algún archivo en sus equipos cliente. Los usuarios refirieron que, una vez familiarizados con el proceso de reportar y rastrear los errores, el uso del sistema fue intuitivo y satisfizo sus necesidades de controlar los errores detectados facilitando su análisis para

determinar cómo sería posible mejorar sus procesos de desarrollo de software, mediante una mejor comprensión de la naturaleza de los errores.

6.1.2 Evaluación del proceso

Aunque para el usuario final podría parecer que el único producto importante generado durante un proyecto de desarrollo de software es el sistema funcionando, desde el punto de vista de la Ingeniería de Software, el proceso utilizado es tan importante como el mismo producto. Un buen proceso de desarrollo de software hace más predecible el resultado final permitiendo estimar adecuadamente los recursos necesarios para llevar a cabo el proyecto. También permite predecir la calidad del producto final. Por otro lado, un mal proceso de desarrollo afectará negativamente la calidad del producto final.

El proceso de desarrollo de software usado para crear el SIARE fue el Proceso Unificado. A continuación, se mencionan los resultados observados durante el desarrollo del sistema aplicando éste.

Una de las fortalezas del Proceso Unificado es que facilita una definición clara y precisa del sistema a construir, permite resolver el problema y definir la solución adecuada. Mediante el análisis guiado por casos de uso fue posible enfocarse en las necesidades de los usuarios finales. Los casos de uso proporcionaron una visión completa del sistema, comprensible para todos los involucrados en el desarrollo. El modelo de casos de uso mostró claramente los actores involucrados en el sistema y la funcionalidad a la que cada uno puede acceder. También, el modelo de casos de uso muestra claramente el alcance del sistema, minimizando así el riesgo asociado a los cambios no controlados.

El uso del proceso unificado en el desarrollo del SIARE permitió mejorar la rastreabilidad de todos los modelos. Cada característica del sistema final puede rastrearse a través de los modelos de diseño y análisis hasta llegar al modelo de casos de uso. Los casos de prueba se derivan directamente de los casos de uso. De ser necesarios cambios en el sistema, es posible ubicar inmediatamente que modelos y qué clases se verán afectadas.

El Proceso Unificado usa el Lenguaje de Modelado Unificado (UML) para representar los diversos modelos de la aplicación. Usar una notación única para expresar los diseños facilitó la comunicación entre todos los participantes en el desarrollo del SIARE.

Otra de las principales características del Proceso Unificado es que permite un desarrollo iterativo. Los modelos del sistema se fueron refinando sucesivamente. En las primeras iteraciones se puso énfasis en la recopilación de requerimientos y al análisis; posteriormente al diseño y a la implementación.

La fase de concepción en la cual se plasman los objetivos del sistema y los requerimientos de alto nivel, corresponde al capítulo 1. Los capítulos 2, 3, 4 y parte del 5 corresponden a fase de elaboración en donde se analiza el dominio del problema y se establece una base arquitectónica. Finalmente, la fase construcción que integra todos los componentes, la codificación del sistema y las pruebas corresponden al capítulo 5.

El proceso de desarrollo iterativo permite desarrollar y liberar un sistema gradualmente. Cada entrega cumple un subconjunto de los requerimientos del proyecto. Esta forma de trabajo disminuye los riegos asociados a liberar un sistema de golpe. Cuando se pretende liberar un sistema completo de una sola vez se corre el riesgo de que el producto final no cumpla con los requerimientos de los usuarios, además de que, al integrar todos los componentes, resulta probable que se encuentren muchos errores en las interfaces de los subsistemas.

Una de las "mejores prácticas" recomendadas por el Proceso Unificado es la verificación constante de la calidad de los productos generados. Cada vez que se concluía una fase, se verificaban los productos creados en la misma. De igual manera, al final de codificar un componente del SIARE se probaba, tal como se señaló en la sección 6.1.1. Gracias a estas prácticas, la integración de los componentes del sistema fue rápida y sencilla.

Finalmente, otra de las mejores prácticas propuestas por el Proceso Unificado es el control de cambios, el cual consiste en controlar, seguir y monitorear los cambios en los diversos artefactos (modelos, código y documentos). Durante todo el desarrollo del SIARE se utilizó un sistema de control de versiones donde se almacenaron todos los artefactos generados, de forma que es posible conservar un historial de versiones, y determinar fácilmente los cambios que se realizaron durante el desarrollo.

6.2 Mejoras futuras

Los sistemas de software son entes en evolución constante. Sin importar que un sistema cumpla con los objetivos originales para los que fue construido, conforme se usa un sistema, los usuarios detectan nuevas características que sería deseable incorporar al sistema. El SIARE no es la

excepción, durante las pruebas del sistema se detectaron las siguientes características que convendría incorporar en versiones futuras.

1. Es necesario mejorar la administración de todos los catálogos del sistema (líderes de proyecto, probadores, desarrolladores, productos y parámetros de los reportes tales como etapas de inyección de error, severidad de los errores y tipos de errores). Esta funcionalidad no se incluyó en la primera versión para permitir que los usuarios se concentraran en comprender el flujo de los reportes de error. El haber incluido el mantenimiento de todos los catálogos podría provocar confusión en usuarios no familiarizados con el proceso completo para reportar y dar seguimiento a los errores.

2. Es deseable automatizar tareas de administración del sistema, concretamente el respaldo y recuperación de la base de datos. Una opción para realizar el respaldo de la base de datos podría ser generando un archivo XML que contenga los registros almacenados en la base de datos. La recuperación podría efectuarse leyendo el archivo XML e insertando los registros en una base de datos vacía. Este mecanismo permitiría además exportar los datos a otros administradores de bases de datos relacionales que cumplan con el estándar ANSI SQL (Oracle, Sql Server, etcétera.).

3. La generación de estadísticas sobre los reportes de error usando el lenguaje PHP resultó compleja, y requirió de un mayor tiempo de desarrollo comparado con el desarrollo de otros módulos del sistema. Existen en el mercado gran variedad de herramientas para análisis de datos. El uso de estas herramientas reduce drásticamente el tiempo necesario para crear reportes a partir de la información almacenada en una base de datos. Además, facilitan la generación de gráficas con los datos especificados. Una de las características más destacables de este tipo de herramientas es que permiten que usuarios sin experiencia en programación definan los reportes que les interese generar. Dado que estas herramientas pueden acceder directamente a la base de datos de donde van a extraer la información, su uso no implicaría cambios en el SIARE. La mayor parte de estas herramientas requieren de una licencia comercial para su uso, por lo que sería necesario analizar la relación costo/beneficio de cada opción para determinar la más factible. Como ejemplos de este tipo de herramientas es posible citar Crystal Reports [CRYSTAL] y AnaliZe.IT [ANALIZE].

4. Con respecto a la seguridad, además de la encripción usando SSL, es conveniente que se autentifique a los clientes que acceden a la aplicación. Esto se puede hacer mediante autentificación SSL del cliente basada en certificados. Dado que los clientes usan un navegador estándar para acceder a la aplicación, no es necesaria programación adicional. Sólo es necesario instalar el certificado en el navegador.

La autentificación del cliente con certificados sería particularmente importante si se quisiera dar un uso comercial al SIARE, en particular si se ofreciera como un servicio rentado vía internet (siguiendo el modelo de Proveedor de Servicio de Aplicaciones o ASP).

6.3 Conclusiones

- El SIARE cumple con el objetivo de registrar los errores descubiertos en un sistema de software, y darles seguimiento hasta su resolución. Dado que la calidad de un producto de software está directamente asociada a los defectos presentes en el mismo, es de suma importancia garantizar un producto libre de errores antes de ser liberado a los usuarios finales. Un proceso de registro y rastreo de errores manual no garantiza que se dé seguimiento a éstos hasta su resolución. Los sistemas de software modernos pueden tener miles de defectos, encontrados por algunas personas y corregidos por otras. De ahí la necesidad de apoyarse en herramientas que proporcionen un formato estándar para reportar los errores y para registrar la resolución de los mismos. Asimismo debe permitir a los líderes de proyecto controlar el flujo de los reportes de error. El SIARE cumple con las características anteriormente citadas, por lo que se puede concluir que su utilización puede ayudar efectivamente a mejorar la calidad de un producto de software, minimizando la cantidad de errores.

- El SIARE ayuda también a mejorar los procesos de desarrollo de software de los equipos que hacen uso de él. La clasificación de los errores según su tipo, severidad, etapa de inyección y tiempo para su resolución, permite a los líderes de proyecto analizar la naturaleza de los errores y defectos que los provocan. De esta manera es posible determinar las causas que originan los defectos, para poder prevenirlos posteriormente. Las estadísticas generadas por el SIARE ayudan a los líderes de proyecto a comprender la naturaleza de los defectos y a prevenir su inyección.

- El uso del SIARE mejora la comunicación entre los miembros de los equipos de desarrollo de software, puesto que al ser accesible vía Web, puede usarse en cualquier equipo capaz de conectarse a internet, aun cuando tenga recursos limitados. Una comunicación fluida es un factor crítico de éxito en un proyecto de desarrollo de software. Además, ya que el SIARE está basado en Web, desplegarlo resulta fácil, puesto que los usuarios no necesitan instalar programa alguno para utilizar la aplicación, de igual manera, las actualizaciones se llevan a cabo de forma transparente para los usuarios finales.

- Las herramientas de software libre usadas (MySql, PHP y Apache) contribuyeron a que se cumplieran dos de los objetivos más importantes del sistema. Por un lado permitieron que el

sistema resultara muy económico, debido al ahorro en la compra de licencias. Por otro lado, dado que estas herramientas pueden instalarse en gran variedad de plataformas con distintos sistemas operativos, el sistema resultó portable, puesto que no es necesario efectuar cambios en el código fuente para poder ejecutarlo sobre otro sistema operativo.

- Durante el desarrollo del sistema fue posible observar que una de las fases más críticas del proceso de desarrollo de software es la recopilación de requerimientos. No importa que un sistema funcione bien aparentemente, puesto que si no cumple con los requerimientos de los usuarios finales resulta inútil. Es necesario identificar la mayor parte de los requerimientos desde el principio del proyecto. De esta forma se evita el trabajo asociado a las modificaciones que se deben realizar cuando se descubre que el requerimiento es incorrecto. El desarrollo dirigido por casos de uso facilitó la recopilación, organización y documentación de los requerimientos del SIARE.

- El Proceso Unificado y el Lenguaje de Modelado Unificado (UML), usados en el desarrollo del SIARE, permitieron mejorar la comunicación a lo largo del proyecto. Asimismo, mejoran la rastreabilidad de los diversos artefactos que conforman el sistema, desde el modelo de casos de uso hasta los casos de prueba. Dado que UML es un estándar *de facto* en la industria del desarrollo de software, sería fácil para otros desarrolladores comprender los modelos del SIARE, y de ser necesario, modificarlo o extenderlo de acuerdo a nuevas necesidades.

Anexo A

Interfaces de clases

Utilerias

Status

Esta clase modela la estructura de datos usada por todas las transacciones para indicar si fueron exitosas o fallidas. Consta de tres propiedades:

`boolean` *bandera* - indica si la transacción fue exitosa o fallida.

`int` *codigo* - código numérico del estado de la transacción (0 si es exitosa, < 0 en caso de error).

`String` *mensaje* - texto informativo.

Constructores:

Status*(*`boolean` *bandera,* `int` *codigo,* `String` *mensaje)*

Crea una nueva instancia de la clase *Status* usando los valores especificados para cada propiedad.

Métodos:

`void` ***setBandera****(*`boolean` *bandera)*

Establece el valor de la propiedad *bandera*.

`void` ***setCodigo****(*`int` *codigo)*

Establece el valor de la propiedad *codigo*.

`void` ***setMensaje****(*`String` *mensaje)*

Establece el valor de la propiedad *mensaje*.

`boolean` ***getBandera****()*

Regresa el valor de la propiedad *bandera*.

`int` ***getCodigo****()*

Regresa el valor de la propiedad *codigo*.

`String` ***getMensaje()***

Regresa el valor de la propiedad *mensaje*.

Utilerias

Date

Esta clase modela el manejo de fechas.

Constructores:

Date()

Crea una nueva instancia de la clase *Date*.

Métodos:

`void` ***setYear(***`int` *year)*

Establece el valor de la propiedad *year* (año).

`void` ***setMonth(***`int` *month)*

Establece el valor de la propiedad *month* (mes).

`void` ***setDay(***`int` *day)*

Establece el valor de la propiedad *day* (día).

`void` ***setHour(***`int` *hour)*

Establece el valor de la propiedad *hour* (hora).

`void` ***setMin(***`int` *min)*

Establece el valor de la propiedad *min* (minuto).

`void` ***setSec(***`int` *sec)*

Establece el valor de la propiedad *sec* (segundo).

`int` ***getYear()***

Regresa el valor de la propiedad *year*.

`int` ***getMonth()***

Regresa el valor de la propiedad *month*.

`int` ***getDay()***

Regresa el valor de la propiedad *day*.

`void` ***getHour()***

Regresa el valor de la propiedad *hour* (hora).

`void` ***getMin()***

Regresa el valor de la propiedad *min* (minuto).

`void` ***getSec()***

Regresa el valor de la propiedad *sec* (segundo).

`boolean` ***setDate(****`int` year, `int` month, `int` day, `int` hour, `int` min, `int` sec)*

Establece el valor de las propiedades *year, month, day, hour, min* y *sec*. En caso de que los valores representen una fecha válida regresa *true* en caso contrario regresa *false*.

`boolean` ***setDateMySql(****`String` dateMySql)*

Establece el valor de las propiedades *year, month, day, hour, min, sec* a partir de una cadena con formato YYYY-MM-DD HH:MM:SS (formato de fecha utilizado por MySql). En caso de que la cadena represente una fecha válida regresa *true,* en caso contrario regresa *false*.

`String` ***getDateMySql()***

Regresa una cadena con formato YYYY-MM-DD HH:MM:SS (formato de fecha utilizado por MySql) que representa la fecha almacenada en una instancia de la clase *Date*.

`boolean` ***esValida()***

Verifica que la fecha representada por las propiedades *year, month, day, hour, min* y *sec* sea válida.

`long` ***getTimeStamp()***

Regresa el *timestamp* (número de segundos transcurridos desde el 01/01/1970 00:00 hasta la fecha representada por las propiedades *year, month, day, hour, min* y *sec*).

`void` ***setTimeStamp(****`long` timestamp)*

Establece los valores de las propiedades *year, month, day, hour, min, sec* a partir del parámetro *timestamp* (número de segundos transcurridos desde el 01/01/1970 00:00 hasta una fecha especificada).

`void` ***addDays****(*`int` *numeroDias)*

Suma el número de días especificado por el parámetro *numeroDias* a la fecha representada por las propiedades *year, month* y *day*.

`long` ***diferenciaFechas****(*`Date` *fecha)*

Regresa la diferencia en días naturales entre la fecha representada por las propiedades *year, month* y *day* y la fecha almacenada en el parámetro *fecha*.

`void` ***setFechaActual****()*

Establece los valores de las propiedades *year, month* y *day* a partir de la fecha actual del servidor donde se ejecuta el script.

Utilerias

Url

Modela la estructura de un URL (Uniform Resource Locator). El URL es una cadena que permite localizar cualquier recurso disponible en internet, tiene el siguiente formato:
<protocolo>://<dirección>/<recurso>

En caso de que el recurso sea una página Web, es posible además especificar un conjunto de parámetros con los que se desea llamar a la página:
<protocolo>://<dirección>/<recurso>?parámetro1=valor1&parámetro2=valor2&...

Constructores:

Url*(*`String` *direccion)*

Crea una nueva instancia de la clase *Url*, inicializando la propiedad *direccion*.

Métodos:

`void` ***setDireccion****(*`String` *direccion)*

Establece el valor de la propiedad *direccion*.

`void` ***agregaParametro****(*`String` *nombre,* `String` *valor)*

Agrega un parámetro con que se llamará a una página Web.

`String` ***armaCadena()***

Crea una cadena que representa el URL formado por la dirección especificada y los parámetros que se hayan agregado.

Utilerias

Link

Esta clase modela la creación de una liga (link) en HTML. Para generar la liga necesita de un URL, por lo que esta clase extiende a la clase *Url*. Además necesita de un texto que aparecerá en la página Web. Crea una cadena con el siguiente formato:

`<A HREF=`*url*`>`*texto*`</A>`

Requiere de: *Url*.

Extiende a: *Url*.

Constructores:

Link *(*`String` *texto,* `String` *direccion)*

Crea una nueva instancia de la clase *Link*, inicializando el texto que aparecerá en la página Web y la dirección a la que apuntará el link.

Métodos:

`void` ***setTexto*** *(*`String` *texto)*

Establece el valor de la propiedad texto.

`void` ***creaLink()***

Genera el código HTML necesario para crear una liga (link).

Utilerias

Redireccion

Esta clase permite llamar automáticamente a una página Web desde otra, usando instrucciones HTML. Para llamar a otra página requiere de un URL, por lo que esta clase extiende a la clase *Url*. Necesita además del tiempo en milisegundos que esperará para llamar a la otra página. Crea una cadena con el siguiente formato:

`<meta http-equiv='Refresh' content=`*tiempo*`;URL=`*url*`>`

Requiere de: *Url*.

Extiende a: *Url*.

Constructores:

Redireccion *(`long` tiempo,`String` direccion)*

Crea una nueva instancia de la clase *Redireccion*, inicializando la dirección y el tiempo de espera.

Métodos:

`void` ***setTiempo*** *(`long` tiempo)*

Establece el valor de la propiedad tiempo.

`void` ***llamarUrl****()*

Genera el código HTML necesario para llamar automáticamente a otra página.

Utilerias

FormateoTexto

Esta clase agrupa funciones para dar formato a cadenas de caracteres.

Requiere de: *Status*.

Constructores:

FormateoTexto*()*

Crea una nueva instancia de la clase *FormateoTexto*.

Métodos:

`String` ***getCadenaTruncada*** *(`String` cadena, `int` longitud)*

Regresa una cadena de texto que representa el parámetro *cadena* truncado a *longitud*.

`Status` ***validaCadena*** *(`String` cadena, `int` longitud)*

Verifica que el parámetro *cadena* corresponda a una cadena de texto, de longitud mayor a cero y menor o igual a *longitud*. Regresa un objeto *Status*, con la propiedad bandera = *true* en caso de éxito, en caso de fracaso la propiedad bandera del objeto *Status* tiene el valor *false*.

`Status` ***validaNumero*** *(`double` numero, `int` longitud)*

Verifica que el parámetro *numero* corresponda a un número, de longitud mayor a cero y menor o igual a *longitud*. Regresa un objeto *Status*, con la propiedad bandera = *true* en caso de éxito, en caso de fracaso la propiedad bandera del objeto *Status* tiene el valor *false*.

`String` ***insertaSaltosLinea*** *(`String` cadena, `int` longitud)*

Regresa una cadena que corresponde al parámetro *cadena*, con saltos de línea y retornos de carácter (\r\n) insertados cada cierto número de caracteres, este número corresponde al parámetro *longitud*.

Utilerias

Vector

Esta clase modela una estructura de datos de lista, donde se puede agregar cualquier objeto.

Constructores:

Vector*()*

Crea una nueva instancia de la clase *Vector*, e inicializa el número de elementos que contiene a cero.

Métodos:

`void` ***addElement****(`Object` elemento)*

Agrega un elemento al *vector* e incrementa el contador de elementos.

`Object` ***elementAt****(`int` indice)*

Regresa el objeto almacenado en la posición apuntada por el parámetro *indice*. Regresa nulo si no hay un objeto almacenado en la posición apuntada por el parámetro *indice*.

`int` ***size****()*

Regresa el número de elementos almacenados en la instancia de la clase *Vector*.

`boolean` ***setElementAt****(`Object` elemento, `int` posicion)*

Almacena el objeto *elemento* en la posición apuntada por el parámetro *posicion*. Regresa *true* en caso de éxito, *false* en cualquier otro caso.

`boolean` ***removeElementAt****(`int` posicion)*

Elimina el objeto almacenado en la posición especificada por el parámetro *posicion* y decrementa el contador de elementos. Regresa *true* en caso de éxito, *false* en cualquier otro caso.

Utilerias

Serializador

Contiene métodos que permiten convertir un objeto a un arreglo de bytes y viceversa. Esto resulta útil cuando se desea almacenar y recuperar un objeto como variable de sesión.

Constructores:

Serializador()

Crea una nueva instancia de la clase *Serializador*.

Métodos:

`byte[]` ***objToByteArr(***`Object` *objeto)*

Regresa un arreglo de bytes que representa a *objeto*.

`Object` ***byteArrToObj(***`byte[]` *byteArr,*`Object` *objeto)*

Transforma el arreglo de bytes *byteArr* a un objeto. Para que al deserializar el arreglo de bytes se genere un objeto del mismo tipo del que se serializó con el método *objToByteArr*, es necesario que el parámetro *objeto* sea una instancia de la clase a la que se desea convertir el arreglo de bytes.

BaseDatos

ParametrosConexion

Contiene los valores de los parámetros por omisión para la conexión entre PHP y MySql.
Para conectar PHP con MySql es necesario contar con un nombre de servidor donde se ubica la base de datos a la que se desea conectar, nombre de la base de datos, usuario de la base de datos y password del usuario.

Constantes:

`String` *SERVIDOR*

Nombre del servidor de base de datos.

`String` *BD_MYSQL*

Nombre de la base de datos.

`String` *USUARIO_MYSQL*

Usuario de la base de datos.

`String` *PASSWORD_MYSQL*

Password del usuario que desea conectarse.

BaseDatos

Conexion

Administra una conexión entre PHP y MySql. Lleva a cabo las operaciones de conexión y desconexión con la base de datos. Toma por omisión los parámetros definidos en la clase *ParametrosConexion*. Usa la clase *Status* para indicar si las operaciones de conexión y desconexión se efectuaron correctamente o generaron un error.

Requiere de: *ParametrosConexion, Status.*

Constructores:

Conexion *(`String` servidor, `String` bd, `String` usuario, `String` password)*

Establece los parámetros que se usarán para conectarse a la base de datos. En caso que no se especifiquen los valores explícitamente, se usan los valores por omisión definidos en la clase *ParametrosConexion*.

Métodos:

`void` ***conectar****()*

Efectúa una conexión con la base de datos especificada por los parámetros de conexión. Establece el valor del status de la operación, con los siguientes valores:

0 - La conexión se efectúo exitosamente.

-1 - Usuario/password no válido.

-2 - No se encontró la base de datos.

El status de la operación se puede consultar en cualquier momento con el método *getStatus().*

`int` ***getLink****()*

Regresa el identificador de la conexión con MySql.

`void` ***desconectar****()*

Elimina la conexión con la base de datos. Establece el valor del *Status* de la operación, con los siguientes valores:

0 - Se cerró la conexión exitosamente.

-3 - Error al cerrar la conexión.

`void` ***setStatus****(`boolean` bandera, `int` codigo, `String` mensaje)*

Usado por los métodos *conectar()* y *desconectar()* para establecer el valor de *Status* mantenido por la instancia de la clase *Conexion*.

`Status` ***getStatus()***

Regresa el valor de *Status* mantenido por la instancia de la clase *Conexion*.

BaseDatos

Query

Esta clase permite ejecutar operaciones de altas, bajas, consultas y actualizaciones sobre una base de datos MySql. Recibe la operación en forma de una sentencia SQL. Permite además bloquear y desbloquear tablas para escritura (este mecanismo sirve para que durante una transacción una tabla no pueda ser afectada por otra transacción hasta que se desbloquee ésta). Contiene además un método que permite generar un número consecutivo a partir del valor más grande que encuentre en un campo de una tabla especificada. Usa la clase *Status* para indicar el éxito o fracaso de una operación. Devuelve los resultados de una consulta mediante la clase *Resultset*. Para indicar sobre qué base de datos ejecuta la operación, recibe una instancia de la clase *Conexion*.

Requiere de: *Status, Resultset, Conexion.*

Constructores:

Query *(*`String` *query,* `Conexion` *conexion)*

Establece que operación se va a ejecutar mediante el parámetro *query* que representa la operación como una sentencia SQL. También establece sobre qué base de datos se ejecutará la operación mediante el parámetro *conexion*.

Métodos:

`void` ***setConexion****(*`Conexion` *conexion)*

Establece sobre qué base de datos se ejecutará la operación mediante el parámetro *conexion*.

`void` ***setQuery****(*`String` *query)*

Establece qué operación se va a ejecutar mediante el parámetro *query* que representa la operación como una sentencia SQL.

`void` ***ejecutaSQL****()*

Ejecuta la sentencia SQL especificada por la propiedad *query*. Establece el valor de status de la operación de la siguiente manera:

0 - Operación exitosa

-4 - Error en la operación

El *Status* de la operación se puede consultar en cualquier momento con el método *getStatus().*

En caso de que la operación sea exitosa, llena la propiedad *Resultset*, que se puede leer en cualquier momento con el método *getResultset().* El *resultset* mantiene los registros generados por la sentencia y el número de registros afectados por la misma.

`void` ***setStatus(***`boolean` *bandera,* `int` *codigo,* `String` *mensaje)*
Usado por los métodos *ejecutaSQL(), bloqueaTabla()* y *getSecuencia()* para establecer el valor de la propiedad *Status* mantenida por una instancia de la clase *Query*.

`void` ***setResultset(***`String[][]` *resultado,* `int` *numeroRegistros)*
Usado por el método *ejecutaSQL()* para guardar el resultado de la sentencia en la propiedad *Resultset*.

`Status` ***getStatus()***
Regresa el valor de *Status* mantenido por la instancia de la clase *Query*.

`Resultset` ***getResultset()***
Regresa el valor de la propiedad *Resultset*.

`void` ***bloqueaTabla(***`String` *nombreTabla)*
Bloquea la(s) tabla(s) especificada(s) por *nombreTabla*. Mientras la tabla esté bloqueada, sólo podrá ser accedida para escritura por operaciones que utilicen la misma conexión que bloqueó la(s) tabla(s). De esta manera se asegura la consistencia en las transacciones. Establece el valor de *Status* de la operación de la siguiente manera:
0 - Operación exitosa
-4 - Error en la operación

`void` ***desbloqueaTablas()***
Desbloquea las tablas bloqueadas para escritura con el método *bloqueaTabla*. Establece el valor de *Status* de la operación de la siguiente manera:
0 - Operación exitosa
-4 - Error en la operación

`long` ***getSecuencia(***`String` *nombreCampo,* `String` *nombreTabla,* `String` *condicion)*
Este método genera un número consecutivo a partir del registro con mayor valor del campo especificado por el parámetro *nombreCampo*, de la tabla especificada por el parámetro

nombreTabla. Adicionalmente se puede agregar una condición al query que genera el consecutivo, especificada por *condicion*. Crea una sentencia SQL de la siguiente forma:

```
select ifnull(max(nombreCampo),0)+1 as seq
from nombreTabla
where condicion
```

Regresa el valor de `seq` en caso de éxito.
Regresa -1 en caso de error. Establece el valor de *Status* de la operación de la siguiente manera:
0 - Operación exitosa
-4 - Error en la operación

BaseDatos

Resultset

Esta clase modela la estructura de datos que almacena los resultados de una operación sobre la base de datos. Los resultados se almacenan en un arreglo bidimensional, en que el primer índice hace referencia al registro y el segundo al campo que se desea obtener. Cada vez que se obtiene un registro, se incrementa el primer índice. Almacena también el número de registros afectados por la operación.

Constructores:

Resultset*(`String[][]` resultado, `int` numeroRegistros)*
Inicializa el conjunto de registros y el número de registros mantenido por una instancia de la clase *Resultset*.

Métodos:

`void` ***setResultado****(`String[][]` resultado)*
Establece el valor del conjunto de registros.

`String[][]` ***getResultado****()*
Regresa el valor del conjunto de registros.

`String[]` ***getRegistro****()*
Regresa un registro almacenado en el conjunto de registros y hace avanzar el apuntador correspondiente al siguiente registro.

`void` ***liberaResultado()***

Libera la memoria asociada a un conjunto de registros.

`void` ***setNumRegistrosAfectados***(`int` *numRegistros)*

Establece el número de registros afectados por una operación.

`int` ***getNumRegistrosAfectados()***

Regresa el número de registros afectados por una operación.

BaseDatos

Pagina

Esta clase modela una estructura de datos que permite organizar los registros obtenidos como resultado de una consulta en forma de páginas, de tamaño parametrizable. Organizando los registros de esta forma es posible desplegarlos página por página en el front-end, evitando formar páginas HTML muy extensas.

Constructores:

Pagina()

Inicializa el contador de registros almacenados en la página a cero.

Métodos:

`void` ***agregaRegistro***(`String[]` *registro)*

Agrega un registro a la instancia de la clase *Pagina* y suma uno al contador de registros.

`String[]` ***getRegistro()***

Regresa un registro almacenado en la *Pagina* y hace avanzar el apuntador correspondiente al siguiente registro.

`void` ***reiniciaApuntador()***

Regresa el apuntador al primer registro almacenado en una instancia de la clase *Pagina*.

`int` ***getNumRegistros()***

Regresa el número de registros almacenados en una instancia de la clase *Pagina*.

BaseDatos

Lista

Esta clase permite ejecutar una consulta sobre la base de datos y paginar los resultados para que, cuando se desplieguen en el front-end, no se forme una página HTML muy extensa, sino se formen páginas de tamaño parametrizable de registros. Esta clase ejecuta la consulta que recibe en el constructor y llena objetos *Pagina* con los registros obtenidos. Mediante un apuntador se le indica si se desea obtener la siguiente página de registros o si se desea retroceder a una página anterior.

Requiere de: *Query, Pagina.*

Constructores:

Lista(`String` *consulta,* `Conexion` *conexion,* `int` *size,* `int` *indice)*

Este constructor recibe la consulta que se desea ejecutar en forma de sentencia SQL, la conexión a la base de datos donde se desea ejecutar la consulta, el tamaño de las páginas que se van a crear y el índice que apunta a la primer página que se desea obtener.

Métodos:

`void` ***setConsulta***(`String` *consulta)*

Establece el valor de la propiedad *consulta*.

`void` ***setConexion***(`Conexion` *conexion)*

Establece el valor de la propiedad *conexion*.

`void` ***setSize***(`int` *size)*

Establece el valor de la propiedad *size* (tamaño máximo de las páginas a formar).

`void` ***setIndicePagina***(`int` *indice)*

Establece el valor de la propiedad *indice* (apuntador a la primer página que se desea obtener).

`Resultset` ***obtieneDatos***()

Ejecuta la consulta especificada por la propiedad *consulta*. Regresa un *Resultset* con los resultados de la consulta.

`Pagina` ***dividePagina***(`int` *indice)*

Este método ejecuta el método *obtieneDatos()* y divide los resultados obtenidos en páginas de tamaño *size*. Devuelve la página indicada por el parámetro *indice*.

`Pagina` ***getPaginaActual()***

Ejecuta el método *dividePagina* usando como parámetro la propiedad *indicePagina*.

`Pagina` ***getPaginaSiguiente()***

Suma 1 a la propiedad *indicePagina* y ejecuta el método *dividePagina* usando el nuevo valor de *indicePagina* como parámetro.

`Pagina` ***getPaginaAnterior()***

Resta 1 a la propiedad *indicePagina* y ejecuta el método *dividePagina* usando el nuevo valor de *indicePagina* como parámetro.

`int` ***getIndicePagina()***

Regresa el valor de la propiedad *indicePagina*.

`int` ***getNumMaximoPaginas()***

Regresa el número de páginas de tamaño *size* en que se dividen los resultados de la consulta.

`int` ***getNumRegistros()***

Regresa el número de registros generados por la consulta.

Mensajeria

Correo

Esta clase permite el envío de correos electrónicos.

Requiere de: *Status*.

Constructores:

Correo()

Crea una nueva instancia de la clase *Correo*.

Métodos:

`Status` ***enviaMensaje(***`String` *email,* `String` *subject,* `String` *mensaje)*

Envía un correo electrónico a la dirección especificada por *email*, con el asunto *subject* y el mensaje especificado por el parámetro *mensaje*. Regresa un objeto status, con la propiedad bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

Solicitud

SolicitudInscripcion

Esta clase modela la entidad correspondiente a una solicitud de inscripción al SIARE. Sus propiedades corresponden a los campos de la tabla *SolicitudInscripcion* de la base de datos. Contiene métodos que permiten dar de alta una solicitud, actualizar una solicitud y buscar una solicitud.

Requiere de: *Query, EstatusSolicitud, Status*.

Constructores:

SolicitudInscripcion*()*

Crea una nueva instancia de la clase *SolicitudInscripcion*. Crea una nueva instancia de la clase *EstatusSolicitud*, usada como propiedad del objeto tipo *SolicitudInscripcion* recién instanciado.

Métodos:

`void` ***setConexion****(*`Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`void` ***setNumSolicitud****(*`long` *numSolicitud)*

Establece el valor de la propiedad numSolicitud.

`void` ***setClvUsuario****(*`String` *clvUsuario)*

Establece el valor de la propiedad clvUsuario.

`void` ***setPassword****(*`String` *password)*

Establece el valor de la propiedad password.

`void` ***setNombre****(*`String` *nombre)*

Establece el valor de la propiedad nombre.

`void` ***setDireccion****(*`String` *direccion)*

Establece el valor de la propiedad dirección.

`void` ***setTelefono****(*`String` *telefono)*

Establece el valor de la propiedad teléfono.

`void` ***setEmail(***`String` *email)*

Establece el valor de la propiedad email.

`void` ***setRazonSocial(***`String` *razonSocial)*

Establece el valor de la propiedad razón social.

`void` ***setEstatusSolicitud(***`EstatusSolicitud` *estatus)*

Establece el valor de la propiedad estatus solicitud.

`long` ***getNumSolicitud()***

Regresa el valor de la propiedad numSolicitud.

`String` ***getClvUsuario()***

Regresa el valor de la propiedad clvUsuario.

`String` ***getPassword()***

Regresa el valor de la propiedad password.

`String` ***getNombre()***

Regresa el valor de la propiedad nombre.

`String` ***getDireccion()***

Regresa el valor de la propiedad dirección.

`String` ***getTelefono()***

Regresa el valor de la propiedad teléfono.

`String` ***getEmail()***

Regresa el valor de la propiedad email.

`String` ***getRazonSocial()***

Regresa el valor de la propiedad razón social.

`EstatusSolicitud` ***getEstatusSolicitud()***

Regresa el valor de la propiedad estatus solicitud.

`Status` ***alta()***

Inserta un nuevo registro en la tabla *SolicitudInscripcion* de la base de datos, donde el valor de cada campo corresponde al valor de la propiedad en el objeto. En el caso de la propiedad estatus solicitud sólo almacena el valor de la clave. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en otro caso.

`Status` ***actualiza()***

Actualiza los valores de un registro en la tabla *SolicitudInscripcion* de la base de datos, donde el valor de cada campo corresponde al valor de la propiedad en el objeto. En el caso de la propiedad estatus solicitud sólo actualiza el valor de la clave. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en otro caso.

`Status` ***busca()***

Consulta la tabla *SolicitudInscripcion* de la base de datos, usando como llave la propiedad numSolicitud. Llena las propiedades del objeto con los valores de los campos correspondientes. En el caso de la propiedad estatus solicitud, busca también se descripción. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en otro caso.

`boolean` ***existe(***`String` *clvUsuario)*

Consulta la tabla *SolicitudInscripcion* de la base de datos, usando como llave la propiedad clvUsuario. Regresa *true* en caso de encontrar algún registro, *false* en cualquier otro caso.

Solicitud

EstatusSolicitud

Esta clase modela los distintos estatus válidos para una solicitud de inscripción al SIARE.

Requiere de: *Query, Status*.

Constantes:

`int` *NUEVA = 1*

Constante para solicitudes nuevas.

`int` *AUTORIZADA = 2*

Constante para solicitudes autorizadas.

`int` *RECHAZADA = 3*

Constante para solicitudes rechazadas.

Constructores:

EstatusSolicitud*()*

Crea una nueva instancia de la clase *EstatusSolicitud*.

Métodos:

`void` ***setConexion****(*`Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`void` ***setClave****(*`int` *clave)*

Establece el valor de la propiedad clvEstatusSolicitud.

`void` ***setDescripcion****(*`String` *descripcion)*

Establece el valor de la propiedad descEstatusSolicitud.

`int` ***getClave****()*

Regresa el valor de la propiedad clvEstatusSolicitud.

`String` ***getDescripcion****()*

Regresa el valor de la propiedad descEstatusSolicitud.

`Status` ***busca****()*

Consulta la tabla *EstatusSolicitud* de la base de datos usando como llave la propiedad clvEstatusSolicitud. Busca la descripción correspondiente. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

Solicitud

ListaSolicitudesNuevas

Esta clase ejecuta una consulta sobre la tabla *SolicitudInscripcion* de la base de datos para buscar todas las solicitudes con estatus de nueva.

Requiere de: *Vector, Query, SolicitudInscripcion*, *EstatusSolicitud*.

Métodos:

`void` ***setConexion****(*`Conexion` *conexion)*

Establece el valor de la propiedad conexión. Esta propiedad es necesaria para indicar sobre qué base de datos se realizarán las transacciones.

`Vector` ***buscaLista****()*

Ejecuta una consulta sobre la tabla *SolicitudInscripcion* de la base de datos para obtener todas las solicitudes con estatus de nueva. Por cada registro que se obtiene, se instancia un objeto *SolicitudInscripcion*, se le pasa el número de solicitud obtenido, se buscan todas las propiedades de la misma, y se agrega a un objeto tipo *Vector* que regresará este método.

Actores

TipoUsuario

Esta clase modela los tipos de usuario válidos en el SIARE.

Requiere de: *Query, Status*.

Constantes:

`int` *ADMINISTRADOR = 1*

Constante para administradores.

`int` *LIDERPROYECTO = 2*

Constante para líderes de proyecto.

`int` *PROBADOR = 3*

Constante para probadores.

`int` *DESARROLLADOR = 4*

Constante para desarrolladores.

Constructores:

TipoUsuario*()*

Crea una nueva instancia de la clase *TipoUsuario*.

Métodos:

`void` ***setConexion****(*`Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`void` ***setClave****(*`int` *clave)*

Establece el valor de la propiedad clvTipoUsuario.

`void` ***setDescripcion****(*`String` *descripcion)*

Establece el valor de la propiedad descTipoUsuario.

`int` ***getClave****()*

Regresa el valor de la propiedad clvTipoUsuario.

`String` ***getDescripcion****()*

Regresa el valor de la propiedad descTipoUsuario.

`Status` ***busca****()*

Consulta la tabla *TiposUsuarios* de la base de datos usando como llave la propiedad clvTipoUsuario. Busca la descripción correspondiente. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

Actores

Usuario

Esta clase modela las características comunes a todos los usuarios del sistema.

Requiere de: *Query, TipoUsuario, Status*.

Constructores:

Usuario*()*

Crea una nueva instancia de la clase usuario.

Métodos:

`void` ***setConexion****(*`Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`void` ***setClave***`(String` *clave)*

Establece el valor de la propiedad clave.

`void` ***setPassword***`(String` *password)*

Establece el valor de la propiedad password.

`void` ***setNombre***`(String` *nombre)*

Establece el valor de la propiedad nombre.

`void` ***setDireccion***`(String` *direccion)*

Establece el valor de la propiedad dirección.

`void` ***setTelefono***`(String` *telefono)*

Establece el valor de la propiedad teléfono.

`void` ***setEmail***`(String` *email)*

Establece el valor de la propiedad email.

`void` ***setTipoUsuario***`(TipoUsuario` *tipoUsuario)*

Establece el valor de la propiedad tipo usuario.

`void` ***setEstaActivo***`(boolean` *estaActivo)*

Establece el valor de la propiedad está activo.

`String` ***getClave***`()`

Regresa el valor de la propiedad clave.

`String` ***getPassword***`()`

Regresa el valor de la propiedad password.

`String` ***getNombre***`()`

Regresa el valor de la propiedad nombre.

`String` ***getDireccion***`()`

Regresa el valor de la propiedad dirección.

`String` ***getTelefono()***

Regresa el valor de la propiedad teléfono.

`String` ***getEmail()***

Regresa el valor de la propiedad email.

`TipoUsuario` ***getTipoUsuario()***

Regresa el valor de la propiedad tipo usuario.

`boolean` ***getEstaActivo()***

Regresa el valor de la propiedad está activo.

`Status` ***altaUsuario()***

Almacena en la tabla *Usuarios* de la base de datos la información correspondiente a cada propiedad de una instancia de la clase *Usuario*. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

`Status` ***buscaUsuario()***

Consulta la tabla *Usuarios* de la base de datos usando como llave primaria la propiedad clave. Busca el resto de las propiedades y las almacena en la misma instancia. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

Actores

LiderdeProyecto

Esta clase modela la entidad correspondiente a las características específicas del líder de proyecto.

Requiere de: *Usuario, Status*.

Extiende a: *Usuario*.

Constructores:

LiderdeProyecto()

Crea una nueva instancia de la clase *LiderdeProyecto*.

Métodos:

`void` ***setRazonSocial****(*`String` *razonSocial)*

Establece el valor de la propiedad razón social.

`String` ***getRazonSocial****()*

Regresa el valor de la propiedad razón social.

`Status` ***altaLider****()*

Almacena en las tablas *Usuarios* y *LideresProyecto* de la base de datos la información correspondiente a cada propiedad de una instancia de la clase *LiderdeProyecto*. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

`Status` ***buscaLider****()*

Consulta las tablas *Usuarios* y *LideresProyecto* de la base de datos usando como llave primaria la propiedad clave. Busca el resto de las propiedades y las almacena en la misma instancia. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

`Vector` ***buscaListaProductos****()*

Consulta la tabla *Productos* usando como llave primaria la clave del líder de proyecto. Para cada registro encontrado instancia un objeto *Producto*, busca el resto de sus propiedades y lo agrega a un *Vector* que se regresará como respuesta.

Actores

Probador

Esta clase modela la entidad correspondiente a las características específicas del probador.

Requiere de: *Usuario, Vector, Producto, LiderdeProyecto, Status*.

Extiende a: *Usuario*.

Constructores:

Probador*()*

Crea una nueva instancia de la clase probador.

Métodos:

`void` ***setLiderProyecto****(*`LiderdeProyecto` *liderProyecto)*

Establece el valor de la propiedad líder de proyecto.

`LiderdeProyecto` ***getLiderProyecto()***

Regresa el valor de la propiedad líder de proyecto.

`Status` ***altaProbador()***

Almacena en las tablas *Usuarios* y *Probadores* de la base de datos la información correspondiente a cada propiedad de una instancia de la clase *Probador*. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

`Status` ***buscaProbador()***

Consulta las tablas *Usuarios* y *Probadores* de la base de datos usando como llave primaria la propiedad clave. Busca el resto de las propiedades y las almacena en la misma instancia. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

`Vector` ***buscaListaProbadores(***`LiderdeProyecto` *liderProyecto)*

Consulta la tabla *Probadores* usando como llave primaria la clave del líder de proyecto. Para cada registro encontrado instancia un objeto *Probador*, busca el resto de sus propiedades y lo agrega a un *Vector* que se regresará como respuesta.

`Vector` ***buscaListaProductos()***

Consulta la tabla *ProbadorProducto* usando como llave primaria la clave del probador. Para cada registro encontrado instancia un objeto *Producto*, busca el resto de sus propiedades y lo agrega a un *Vector* que se regresará como respuesta.

Actores

Desarrollador

Esta clase modela la entidad correspondiente a las características específicas del desarrollador.

Requiere de: *Usuario, Vector, Producto, LiderdeProyecto, Status*.

Extiende a: *Usuario*.

Constructores:

Desarrollador()

Crea una nueva instancia de la clase *Desarrollador*.

Métodos:

`void` ***setLiderProyecto(***`LiderdeProyecto` *liderProyecto)*

Establece el valor de la propiedad líder de proyecto.

`LiderdeProyecto` ***getLiderProyecto()***

Regresa el valor de la propiedad líder de proyecto.

`Status` ***altaDesarrollador()***

Almacena en las tablas *Usuarios* y *Desarrolladores* de la base de datos la información correspondiente a cada propiedad de una instancia de la clase *Desarrollador*. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

`Status` ***buscaDesarrollador()***

Consulta las tablas *Usuarios* y *Desarrolladores* de la base de datos usando como llave primaria la propiedad clave. Busca el resto de las propiedades y las almacena en la misma instancia. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

`Vector` ***buscaListaDesarrolladores(***`LiderdeProyecto` *liderProyecto)*

Consulta la tabla *Desarrolladores* usando como llave primaria la clave del líder de proyecto. Para cada registro encontrado instancia un objeto *Desarrollador*, busca el resto de sus propiedades y lo agrega a un *Vector* que se regresará como respuesta.

Actores

Administrador

Esta clase modela la entidad correspondiente a las características específicas del administrador.

Requiere de: *Usuario*.

Extiende a: *Usuario*.

Constructores:

Administrador()

Crea una nueva instancia de la clase administrador.

Reportes

EstatusReporte

Modela las propiedades que describen el estatus de un reporte.

Requiere de: *Query, Status*.

Constantes:

`int` *NUEVO = 1*

Constante para reportes nuevos.

`int` *RECHAZADO = 2*

Constante para reportes rechazados.

`int` *ASIGNADO = 3*

Constante para reportes a los que se ha asignado un desarrollador para darle seguimiento.

`int` *CERRADO = 4*

Constante para reportes cerrados.

Constructores:

***EstatusReporte** ()*

Crea una nueva instancia de la clase *EstatusReporte*.

Métodos:

`void` ***setConexion**(`Conexion` conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`void` ***setClave**(`int` clave)*

Establece el valor de la propiedad clvEstatusReporte.

`void` ***setDescripcion**(`String` descripcion)*

Establece el valor de la propiedad descEstatusReporte.

`int` ***getClave**()*

Regresa el valor de la propiedad clvEstatusReporte.

`String` ***getDescripcion****()*

Regresa el valor de la propiedad descEstatusReporte.

`Status` ***busca****()*

Consulta la tabla *EstatusReporte* de la base de datos usando como llave la propiedad clvEstatusReporte. Busca la descripción correspondiente. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

Reportes

TipoError

Modela las propiedades que describen el tipo de error.

Requiere de: *Query, Vector, Status*.

Constructores:

TipoError *()*

Crea una nueva instancia de la clase *TipoError*.

Métodos:

`void` ***setConexion****(*`Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`void` ***setClave****(*`int` *clave)*

Establece el valor de la propiedad clvTipoError.

`void` ***setDescripcion****(*`String` *descripcion)*

Establece el valor de la propiedad descTipoError.

`void` ***setEstaActivo****(*`boolean` *estaActivo)*

Establece el valor de la propiedad está activo.

`int` ***getClave****()*

Regresa el valor de la propiedad clvTipoError.

`String` ***getDescripcion****()*

Regresa el valor de la propiedad descTipoError.

`boolean` ***getEstaActivo****()*

Regresa el valor de la propiedad está activo.

`Status` ***busca****()*

Consulta la tabla *TipoError* de la base de datos usando como llave la propiedad clvTipoError. Busca la descripción correspondiente. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

`Vector` ***buscaLista****()*

Consulta la tabla *TipoError* de la base de datos buscando todos los registros existentes en la misma. Para cada registro instancia un nuevo objeto *TipoError*, busca el resto de sus propiedades y lo agrega a un *Vector* que se regresará como respuesta.

Reportes

TipoSeveridad

Modela las propiedades que describen el tipo de severidad.

Requiere de: *Query, Vector, Status*.

Constructores:

TipoSeveridad *()*

Crea una nueva instancia de la clase *TipoSeveridad*.

Métodos:

`void` ***setConexion****(*`Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`void` ***setClave****(*`int` *clave)*

Establece el valor de la propiedad clvTipoSeveridad.

`void` ***setDescripcion(***`String` *descripcion)*

Establece el valor de la propiedad descTipoSeveridad.

`void` ***setEstaActivo(***`boolean` *estaActivo)*

Establece el valor de la propiedad está activo.

`int` ***getClave()***

Regresa el valor de la propiedad clvTipoSeveridad.

`String` ***getDescripcion()***

Regresa el valor de la propiedad descTipoSeveridad.

`boolean` ***getEstaActivo()***

Regresa el valor de la propiedad está activo.

`Status` ***busca()***

Consulta la tabla *TipoSeveridad* de la base de datos usando como llave la propiedad clvTipoSeveridad. Busca la descripción correspondiente. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

`Vector` ***buscaLista()***

Consulta la tabla *TipoSeveridad* de la base de datos buscando todos los registros existentes en la misma. Para cada registro instancia un nuevo objeto *TipoSeveridad*, busca el resto de sus propiedades y lo agrega a un *Vector* que se regresará como respuesta.

Reportes

EtapaInyeccionError

Modela las propiedades que describen la etapa de inyección de un error.

Requiere de: *Query, Vector, Status*.

Constructores:

EtapaInyeccionError *()*

Crea una nueva instancia de la clase EtapaInyeccionError.

Métodos:

`void` ***setConexion****(*`Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`void` ***setClave****(*`int` *clave)*

Establece el valor de la propiedad clvEtapaInyeccionError.

`void` ***setDescripcion****(*`String` *descripcion)*

Establece el valor de la propiedad descEtapaInyeccionError.

`void` ***setEstaActivo****(*`boolean` *estaActivo)*

Establece el valor de la propiedad está activo.

`int` ***getClave****()*

Regresa el valor de la propiedad clvEtapaInyeccionError.

`String` ***getDescripcion****()*

Regresa el valor de la propiedad descEtapaInyeccionError.

`boolean` ***getEstaActivo****()*

Regresa el valor de la propiedad está activo.

`Status` ***busca****()*

Consulta la tabla *EtapaInyeccionError* de la base de datos usando como llave la propiedad clvEtapaInyeccionError. Busca la descripción correspondiente. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

`Vector` ***buscaLista****()*

Consulta la tabla *EtapaInyeccionError* de la base de datos buscando todos los registros existentes en la misma. Para cada registro instancia un nuevo objeto *EtapaInyeccionError*, busca el resto de sus propiedades y lo agrega a un *Vector* que se regresará como respuesta.

Reportes

Producto

Modela las propiedades de un producto de software a probar.

Requiere de: *Query, Vector, LiderdeProyecto, Probador, Desarrollador, Status*.

Constructores:

***Producto** ()*

Crea una nueva instancia de la clase *Producto*.

Métodos:

`void` ***setConexion***`(Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`void` ***setClave***`(String` *clave)*

Establece el valor de la propiedad clave.

`void` ***setLiderProyecto***`(LiderdeProyecto` *lider)*

Establece el valor de la propiedad líder de proyecto.

`void` ***setNombre***`(String` *nombre)*

Establece el valor de la propiedad nombre.

`void` ***setDescripcion***`(String` *descripcion)*

Establece el valor de la propiedad descripción.

`void` ***setEstaActivo***`(boolean` *estaActivo)*

Establece el valor de la propiedad está activo.

`void` ***setListaProbadores***`(Vector` *listaProbadores)*

Establece el valor de la propiedad lista de probadores.

`void` ***setListaDesarrolladores***`(Vector` *listaDesarrolladores)*

Establece el valor de la propiedad lista de desarrolladores.

`String` ***getClave()***

Regresa el valor de la propiedad clave.

`LiderdeProyecto` ***getLiderProyecto()***

Regresa el valor de la propiedad líder de proyecto.

`String` ***getNombre()***

Regresa el valor de la propiedad nombre.

`String` ***getDescripcion()***

Regresa el valor de la propiedad descripción.

`boolean` ***getEstaActivo()***

Regresa el valor de la propiedad está activo.

`Vector` ***getListaProbadores()***

Regresa el valor de la propiedad lista de probadores.

`Vector` ***getListaDesarrolladores()***

Regresa el valor de la propiedad lista de desarrolladores.

`Status` ***alta()***

Almacena en las tablas *Productos, DesarrolladorProducto* y *ProbadorProducto* de la base de datos la información correspondiente a cada propiedad de una instancia de la clase *Producto*. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

`Status` ***busca()***

Consulta las tablas *Productos, DesarrolladorProducto* y *ProbadorProducto* de la base de datos usando como llave primaria la propiedad clave. Busca el resto de las propiedades y las almacena en la misma instancia. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

Reportes

ReporteDeError

Modela las propiedades de un reporte de error.

Requiere de: *Query, Producto, Probador, Desarrollador, EstatusReporte, EtapaInyeccionError, TipoSeveridad, TipoError, Date, Status*.

Constructores:

ReporteDeError ()

Crea una nueva instancia de la clase *ReportedeError*.

Métodos:

`void` ***setConexion***`(Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`void` ***setNumReporte***`(long` *numReporte)*

Establece el valor de la propiedad numReporte.

`void` ***setProducto***`(Producto` *producto)*

Establece el valor de la propiedad producto.

`void` ***setProbador***`(Probador` *probador)*

Establece el valor de la propiedad probador.

`void` ***setDesarrollador***`(Desarrollador` *desarrollador)*

Establece el valor de la propiedad desarrollador.

`void` ***setDescModulo***`(String` *descModulo)*

Establece el valor de la propiedad descripción de módulo.

`void` ***setVersion***`(String` *version)*

Establece el valor de la propiedad versión.

`void` ***setDescError***`(String` *descError)*

Establece el valor de la propiedad descripción de error.

`void` ***setFechaHoraAlta***`(Date` *fechaHoraAlta)*

Establece el valor de la propiedad fecha y hora de alta.

`void` ***setDescCausaRechazo(***`String` *descCausaRechazo)*

Establece el valor de la propiedad descripción de la causa de rechazo.

`void` ***setDiagnostico(***`String` *diagnostico)*

Establece el valor de la propiedad diagnóstico.

`void` ***setDescCorreccion(***`String` *descCorreccion)*

Establece el valor de la propiedad descripción de corrección.

`void` ***setFechaHoraUltimoEstatus(***`Date` *fechaHoraUltimoEstatus)*

Establece el valor de la propiedad fecha y hora de último estatus.

`void` ***setEstatus(***`EstatusReporte` *estatus)*

Establece el valor de la propiedad estatus.

`void` ***setEtapa(***`EtapaInyeccionError` *etapa)*

Establece el valor de la propiedad etapa.

`void` ***setTipoSeveridad(***`TipoSeveridad` *tipoSeveridad)*

Establece el valor de la propiedad tipo de severidad.

`void` ***setTipoError(***`TipoError` *tipoError)*

Establece el valor de la propiedad tipo de error.

`void` ***getNumReporte(***`long` *numReporte)*

Regresa el valor de la propiedad numReporte.

`void` ***getProducto(***`Producto` *producto)*

Regresa el valor de la propiedad producto.

`Probador` ***getProbador()***

Regresa el valor de la propiedad probador.

`Desarrollador` ***getDesarrollador()***

Regresa el valor de la propiedad desarrollador.

`String` ***getDescModulo()***

Regresa el valor de la propiedad descripción de módulo.

`String` ***getVersion()***

Regresa el valor de la propiedad versión.

`String` ***getDescError()***

Regresa el valor de la propiedad descripción de error.

`Date` ***getFechaHoraAlta()***

Regresa el valor de la propiedad fecha y hora de alta.

`String` ***getDescCausaRechazo()***

Regresa el valor de la propiedad descripción de la causa de rechazo.

`String` ***getDiagnostico()***

Regresa el valor de la propiedad diagnóstico.

`String` ***getDescCorreccion()***

Regresa el valor de la propiedad descripción de corrección.

`Date` ***getFechaHoraUltimoEstatus()***

Regresa el valor de la propiedad fecha y hora de último estatus.

`EstatusReporte` ***getEstatus()***

Regresa el valor de la propiedad estatus.

`EtapaInyeccionError` ***getEtapa()***

Regresa el valor de la propiedad etapa.

`TipoSeveridad` ***getTipoSeveridad()***

Regresa el valor de la propiedad tipo de severidad.

`TipoError` ***getTipoError()***

Regresa el valor de la propiedad tipo de error.

`Status` ***alta()***

Almacena en la tabla *ReporteDeError* de la base de datos la información correspondiente a cada propiedad de una instancia de la clase *ReporteDeError*. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

`Status` ***busca()***

Consulta la tabla *ReporteDeError* de la base de datos usando como llave primaria la propiedad numReporte. Busca el resto de las propiedades y las almacena en la misma instancia. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en cualquier otro caso.

`Status` ***actualiza()***

Actualiza los valores de un registro en la tabla *ReporteError* de la base de datos, donde el valor de cada campo corresponde al valor de la propiedad en el objeto. En el caso de la propiedad estatus solicitud sólo actualiza el valor de la clave. Regresa un objeto *Status* con bandera = *true* en caso de éxito, bandera = *false* en otro caso.

Reportes

ListaReportesNuevos

Esta clase ejecuta una consulta sobre la tabla *ReporteError* de la base de datos para buscar todos los reportes con estatus de nuevo para los productos dados de alta por un líder de proyecto.

Requiere de: *Vector, Query, ReporteDeError*, *LiderdeProyecto, EstatusReporte*.

Métodos:

`void` ***setConexion(***`Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`Vector` ***buscaLista(***`LiderdeProyecto` *liderProyecto)*

Ejecuta una consulta sobre la tabla *ReporteError* de la base de datos para obtener todos los reportes con estatus de nuevo correspondientes a los productos dados de alta por el líder de proyecto especificado. Por cada registro que se obtiene, se instancia un objeto *ReporteDeError*, se le pasa el número de reporte obtenido, se buscan todas las propiedades del mismo, y se agrega a un objeto tipo *Vector* que regresará este método.

Reportes

ListaReportesLevantados

Esta clase ejecuta una consulta sobre la tabla *ReporteError* de la base de datos para buscar todos los reportes levantados por un probador, que se les haya asignado un desarrollador para darles seguimiento.

Requiere de: *Vector, Query, ReporteDeError*, *Probador, EstatusReporte*.

Métodos:

`void` ***setConexion***`(Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`Vector` ***buscaLista***`(Probador` *probador)*

Ejecuta una consulta sobre la tabla *ReporteError* de la base de datos para obtener todos los reportes con estatus de asignado para seguimiento, dados de alta por el probador especificado. Por cada registro que se obtiene, se instancia un objeto *ReporteDeError*, se le pasa el número de reporte obtenido, se buscan todas las propiedades del mismo, y se agrega a un objeto tipo *Vector* que regresará este método.

Reportes

ListaReportesAsignados

Esta clase ejecuta una consulta sobre la tabla *ReporteError* de la base de datos para buscar todos los reportes asignados a un desarrollador, que no hayan sido cerrados.

Requiere de: *Vector, Query, ReporteDeError*, *Desarrollador, EstatusReporte*.

Métodos:

`void` ***setConexion***`(Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`Vector` ***buscaLista***`(Desarrollador` *desarrollador)*

Ejecuta una consulta sobre la tabla *ReporteError* de la base de datos para obtener todos los reportes con estatus de asignado para seguimiento, para el desarrollador especificado. Por cada registro que se obtiene, se instancia un objeto *ReporteDeError*, se le pasa el número de reporte

obtenido, se buscan todas las propiedades del mismo, y se agrega a un objeto tipo *Vector* que regresará este método.

Estadisticas

DistribucionEtapaInyeccion

Esta clase obtiene los datos necesarios para generar el reporte de distribución de defectos por etapa de inyección (sección 2.6.1).

Requiere de: *Query, LiderdeProyecto, Producto*.

Métodos:

`void` ***setConexion***`(Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`String` ***generaReporte***`(LiderdeProyecto` *liderProyecto*`, Producto` *producto)*

Consulta la tabla *ReporteError* para obtener el número de defectos inyectados por cada etapa, así como el porcentaje del total de defectos inyectados por etapa. Si el parámetro *producto* es nulo, busca la información para todos los productos gestionados por el líder de proyecto especificado por el parámetro *liderProyecto*. En caso de que se especifique un valor distinto de nulo para el parámetro *producto*, se buscan sólo los datos asociados al producto especificado. Regresa una cadena en formato HTML con los datos encontrados.

Estadisticas

DistribucionSeveridad

Esta clase obtiene los datos necesarios para generar el reporte de distribución de defectos por severidad (sección 2.6.2).

Requiere de: *Query, LiderdeProyecto, Producto*.

Métodos:

`void` ***setConexion***`(Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`String` ***generaReporte(*** `LiderdeProyecto` *liderProyecto,* `Producto` *producto)*

Consulta la tabla *ReporteError* para obtener el número de defectos, clasificados según la severidad, así como el porcentaje del total por cada tipo de severidad. Si el parámetro *producto* es nulo, busca la información para todos los productos gestionados por el líder de proyecto especificado por el parámetro *liderProyecto*. En caso de que se especifique un valor distinto de nulo para el parámetro *producto*, se buscan sólo los datos asociados al producto especificado. Regresa una cadena en formato HTML con los datos encontrados.

Estadisticas

DistribucionTipo

Obtiene los datos necesarios para generar el reporte de distribución de defectos por tipo (sección 2.6.3).

Requiere de: *Query, LiderdeProyecto, Producto*.

Métodos:

`void` ***setConexion(*** `Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`String` ***generaReporte(*** `LiderdeProyecto` *liderProyecto,* `Producto` *producto)*

Consulta la tabla *ReporteError* para obtener el número de defectos, clasificado por tipo, así como el porcentaje del número total de defectos. Si el parámetro *producto* es nulo, busca la información para todos los productos gestionados por el líder de proyecto especificado por el parámetro *liderProyecto*. En caso de que se especifique un valor distinto de nulo para el parámetro *producto*, se buscan sólo los datos asociados al producto especificado. Regresa una cadena en formato HTML con los datos encontrados.

Estadisticas

DistribucionModulo

Obtiene los datos necesarios para generar el reporte de distribución de defectos por módulo (sección 2.6.4).

Requiere de: *Query, Producto*.

Métodos:

`void` ***setConexion***`(Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`String` ***generaReporte***`(Producto` *producto)*

Consulta la tabla *ReporteError* para obtener el número de defectos, clasificados según el módulo del producto donde se encontraron. Busca los datos asociados al producto especificado por el parámetro *producto*. Regresa una cadena en formato HTML con los datos encontrados.

Estadisticas

TipoxEtapa

Obtiene los datos necesarios para generar el reporte de tipos de defecto por etapa de inyección (sección 2.6.5).

Requiere de: *Query, LiderdeProyecto, Producto*.

Métodos:

`void` ***setConexion***`(Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`String` ***generaReporte***`(LiderdeProyecto` *liderProyecto*`, Producto` *producto)*

Consulta la tabla *ReporteError* para obtener el número de defectos según su tipo, agrupados por la etapa de su inyección. Si el parámetro *producto* es nulo, busca la información para todos los productos gestionados por el líder de proyecto especificado por el parámetro *liderProyecto*. En caso de que se especifique un valor distinto de nulo para el parámetro *producto*, se buscan sólo los datos asociados al producto especificado. Regresa una cadena en formato HTML con los datos encontrados.

Estadisticas

TipoxSeveridad

Obtiene los datos necesarios para generar el reporte de tipos de defecto por severidad (sección 2.6.6).

Requiere de: *Query, LiderdeProyecto, Producto*.

Métodos:

`void` ***setConexion****(*`Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`String` ***generaReporte****(*`LiderdeProyecto` *liderProyecto*, `Producto` *producto)*

Consulta la tabla *ReporteError* para obtener el número de defectos según su tipo, agrupados por severidad. Si el parámetro *producto* es nulo, busca la información para todos los productos gestionados por el líder de proyecto especificado por el parámetro *liderProyecto*. En caso de que se especifique un valor distinto de nulo para el parámetro *producto*, se buscan sólo los datos asociados al producto especificado. Regresa una cadena en formato HTML con los datos encontrados.

Estadisticas

EstatusxProducto

Obtiene los datos necesarios para generar el reporte de estatus de los reportes por producto (sección 2.6.7).

Requiere de: *Query, Producto*.

Métodos:

`void` ***setConexion****(*`Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`String` ***generaReporte****(*`Producto` *producto)*

Consulta la tabla *ReporteError* para obtener la distribución de los reportes de error según su estatus. Busca los datos asociados al producto especificado por el parámetro *producto*. Regresa una cadena en formato HTML con los datos encontrados.

Estadisticas

TiempoCierre

Esta clase obtiene los datos necesarios para generar el reporte de distribución de reportes por tiempo para su cierre (sección 2.6.8).

Requiere de: *Query, LiderdeProyecto, Producto*.

Métodos:

`void` ***setConexion****(*`Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`String` ***generaReporte****(*`LiderdeProyecto` *liderProyecto*, `Producto` *producto)*

Consulta la tabla *ReporteError* para obtener la distribución de los reportes según el número de días transcurridos desde que se da de alta el reporte hasta que se cierra. Si el parámetro *producto* es nulo, busca la información para todos los productos gestionados por el líder de proyecto especificado por el parámetro *liderProyecto*. En caso de que se especifique un valor distinto de nulo para el parámetro *producto*, se buscan sólo los datos asociados al producto especificado. Regresa una cadena en formato HTML con los datos encontrados.

Estadisticas

EstatusxPeriodo

Esta clase obtiene los datos necesarios para generar el reporte de reportes abiertos y cerrados por periodo (sección 2.6.9).

Requiere de: *Date, Query, LiderdeProyecto, Producto*.

Métodos:

`void` ***setConexion****(*`Conexion` *conexion)*

Establece el valor de la propiedad conexión. Necesario para indicar sobre qué base de datos se realizarán las transacciones.

`String` ***generaReporte****(*`Producto` *producto,* `Date` *fechaInicio,* `Date` *fechaFin)*

Consulta la tabla *ReporteError* para obtener la distribución de reportes según su estatus (nuevo o cerrado), a lo largo de un periodo de tiempo especificado. Para cada día del periodo, despliega cuántos reportes se han abierto y cuántos se han cerrado. Busca sólo los datos asociados al producto especificado. Regresa una cadena en formato HTML con los datos encontrados.

Anexo B

Casos de Prueba

Nombre del caso de prueba	Condición inicial	Flujo de eventos	Condición final
Alta solicitud inscripción - Alta exitosa	1. El líder de proyecto se encuentra en la página principal del SIARE (http://www.seminario-sqa.com/tesis/index.php). 2. El líder de proyecto cuenta con una dirección de correo electrónico válida.	1. El líder de proyecto da clic en la liga "Alta Solicitud Inscripción". 2. El líder de proyecto llena todos los datos solicitados por el formulario. El nombre de usuario no se ha registrado previamente en el sistema; la contraseña y su confirmación son idénticas. 3. El líder de proyecto da clic en el botón "Registrarse".	1. El sistema despliega un mensaje indicando que la solicitud fue dada de alta exitosamente, incluyendo el número de solicitud generado. 2. El sistema envía un correo electrónico al administrador del sistema (admin@seminario-sqa.com) indicándole que hay una nueva solicitud de inscripción.
Alta solicitud inscripción - Datos incompletos	1. El líder de proyecto se encuentra en la página principal del SIARE (http://www.seminario-sqa.com/tesis/index.php). 2. El líder de proyecto cuenta con una dirección de correo electrónico válida.	1. El líder de proyecto da clic en la liga "Alta Solicitud Inscripción". 2. El líder de proyecto omite cualquiera de los datos requeridos por el formulario. 3. El líder de proyecto da clic en el botón "Registrarse".	1. El sistema despliega un mensaje indicando el dato faltante.
Alta solicitud inscripción - Correo inválido	1. El líder de proyecto se encuentra en la página principal del SIARE (http://www.seminario-sqa.com/tesis/index.php). 2. La dirección de correo electrónico del líder de	1. El líder de proyecto da clic en la liga "Alta Solicitud Inscripción". 2. El líder de proyecto llena todos los datos solicitados por el formulario. El nombre de usuario	1. El sistema despliega un mensaje indicando que el correo electrónico introducido no es válido.

	proyecto no es válida (*usuario@servidor*).	no se ha registrado previamente en el sistema; la contraseña y su confirmación son idénticas. 3. El líder de proyecto da clic en el botón "Registrarse".	
Alta solicitud inscripción - Confirmación de contraseña incorrecta	1. El líder de proyecto se encuentra en la página principal del SIARE (http://www.seminario-sqa.com/tesis/index.php). 2. El líder de proyecto cuenta con una dirección de correo electrónico válida.	1. El líder de proyecto da clic en la liga "Alta Solicitud Inscripción". 2. El líder de proyecto llena todos los datos solicitados por el formulario. El nombre de usuario no se ha registrado previamente en el sistema; la contraseña y su confirmación son distintas. 3. El líder de proyecto da clic en el botón "Registrarse".	1. El sistema despliega un mensaje indicando que la contraseña y su confirmación no son idénticas.
Alta solicitud Inscripción - Usuario ya registrado	1. El líder de proyecto se encuentra en la página principal del SIARE (http://www.seminario-sqa.com/tesis/index.php). 2. El líder de proyecto cuenta con una dirección de correo electrónico válida.	1. El líder de proyecto da clic en la liga "Alta Solicitud Inscripción". 2. El líder de proyecto llena todos los datos solicitados por el formulario. El nombre de usuario ya se encuentra registrado en el sistema; la contraseña y su confirmación son idénticas. 3. El líder de proyecto da clic en el botón "Registrarse".	1. El sistema despliega un mensaje indicando que el nombre de usuario ya se ha registrado.
Valida usuario - Validar usuario Administrador	1. El administrador se encuentra en la página principal del SIARE (http://www.seminario-sqa.com/tesis/index.php). 2. El administrador cuenta con un nombre de usuario y contraseña registrados en el sistema (admin, admin).	1. El administrador da clic en la liga "Valida Usuario". 2. El administrador introduce el nombre de usuario y la contraseña. 3. El administrador da clic en el botón "Valida".	1. El sistema despliega el menú para el administrador (http://www.seminario-sqa.com/tesis/Administrador/WebMenuAdministrador.php).
Valida usuario - Validar usuario Líder de proyecto	1. El líder de proyecto se encuentra en la página principal del SIARE (http://www.seminario-sqa.com/tesis/index.php).	1. El líder de proyecto da clic en la liga "Valida Usuario". 2. El líder de proyecto introduce el nombre de	1. El sistema despliega el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLider

	2. El líder de proyecto cuenta con un nombre de usuario y contraseña registrados en el sistema.	usuario y la contraseña. 3. El líder de proyecto da clic en el botón "Valida".	Proyecto.php).
Valida usuario - Validar usuario Probador	1. El probador se encuentra en la página principal del SIARE (http://www.seminario-sqa.com/tesis/index.php). 2. El probador cuenta con un nombre de usuario y contraseña registrados en el sistema.	1. El probador da clic en la liga "Valida Usuario". 2. El probador introduce el nombre de usuario y la contraseña. 3. El probador da clic en el botón "Valida".	1. El sistema despliega el menú para el probador (http://www.seminario-sqa.com/tesis/Probador/WebMenuProbador.php).
Valida usuario - Validar usuario Desarrollador	1. El desarrollador se encuentra en la página principal del SIARE (http://www.seminario-sqa.com/tesis/index.php). 2. El desarrollador cuenta con un nombre de usuario y contraseña registrados en el sistema.	1 El desarrollador da clic en la liga "Valida Usuario". 2 El desarrollador introduce el nombre de usuario y la contraseña. 3 El desarrollador da clic en el botón "Valida".	1. El sistema despliega el menú para el desarrollador (http://www.seminario-sqa.com/tesis/Desarrollador/WebMenuDesarrollador.php).
Valida usuario - Validar usuario inexistente	1. Un usuario se encuentra en la página principal del SIARE (http://www.seminario-sqa.com/tesis/index.php). 2. El usuario no se ha registrado en el sistema	1. El usuario da clic en la liga "Valida Usuario". 2. El usuario introduce un nombre de usuario y contraseña. 3. El usuario da clic en el botón "Valida".	1. El sistema despliega un mensaje informando que no existe el usuario.
Valida usuario - Contraseña incorrecta	1. Un usuario se encuentra en la página principal del SIARE (http://www.seminario-sqa.com/tesis/index.php). 2. El usuario cuenta con un nombre de usuario y contraseña registrados en el sistema.	1. El usuario da clic en la liga "Valida Usuario". 2. El usuario introduce un nombre de usuario y contraseña. La contraseña capturada no es igual a la almacenada en la base de datos. 3. El usuario da clic en el botón "Valida".	1. El sistema despliega un mensaje informando que la contraseña es incorrecta.
Autoriza solicitud inscripción - Autorizar solicitud	1. El administrador se encuentra en el menú para el administrador (http://www.seminario-sqa.com/tesis/Administrador/WebMenuAdministrador.php). 2. Se ha dado de alta al menos una solicitud de	1. El administrador da clic en la liga "Administración de Solicitudes". 2. El administrador da clic en el número de solicitud que desea autorizar. 3. El sistema despliega el detalle de la solicitud y	1. El sistema despliega un mensaje informando que la solicitud fue autorizada. 2. El sistema envía un correo electrónico a la dirección capturada por el líder de proyecto en la solicitud, indicando que la solicitud fue

	inscripción al sistema.	muestra un formulario que permite seleccionar si se autoriza, se rechaza o se deja pendiente la solicitud. 4. El administrador selecciona la opción de autorizar 5. El administrador da clic el botón "Cambia".	autorizada. 3. La solicitud desaparece de la lista desplegada usando la opción "Administración de Solicitudes".
Autoriza solicitud inscripción - Rechazar solicitud	1. El administrador se encuentra en el menú para el administrador (http://www.seminario-sqa.com/tesis/Administrador/WebMenuAdministrador.php). 2. Se ha dado de alta al menos una solicitud de inscripción al sistema.	1. El administrador da clic en la liga "Administración de Solicitudes". 2. El administrador da clic en el número de solicitud que desea rechazar. 3. El sistema despliega el detalle de la solicitud y muestra un formulario que permite seleccionar si se autoriza, se rechaza o se deja pendiente la solicitud. 4. El administrador selecciona la opción rechazar. 5. El administrador da clic el botón "Cambia".	1. El sistema despliega un mensaje informando que la solicitud fue rechazada. 2. El sistema envía un correo electrónico a la dirección capturada por el líder de proyecto en la solicitud, indicando que la solicitud fue rechazada. 3. La solicitud desaparece de la lista desplegada usando la opción "Administración de Solicitudes".
Autoriza solicitud inscripción - Dejar solicitud pendiente	1. El administrador se encuentra en el menú para el administrador (http://www.seminario-sqa.com/tesis/Administrador/WebMenuAdministrador.php). 2. Se ha dado de alta al menos una solicitud de inscripción al sistema.	1. El administrador da clic en la liga "Administración de Solicitudes". 2. El administrador da clic en el número de solicitud que deseado. 3. El sistema despliega el detalle de la solicitud y muestra un formulario que permite seleccionar si se autoriza, se rechaza o se deja pendiente la solicitud. 4. El administrador selecciona la opción pendiente. 5. El administrador presiona el botón "Cambia".	1. El sistema despliega un mensaje informando que la operación fue exitosa. 2. La solicitud sigue apareciendo en la lista desplegada usando la opción "Administración de Solicitudes".
Alta probador - Alta exitosa	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. El probador a dar de alta cuenta con una	1. El líder de proyecto da clic en la liga "Alta de Probadores". 2. El líder de proyecto llena todos los datos solicitados por el formulario. El nombre de usuario no se ha registrado previamente en el sistema; la contraseña y su confirmación son idénticas.	1. El sistema despliega un mensaje indicando que el probador fue dado de alta exitosamente. 2. El sistema envía un correo electrónico a la dirección capturada en el formulario, para indicar al probador que fue dado de alta en

	dirección de correo electrónico válida.	3. El líder de proyecto da clic en el botón "Registrarse".	el sistema.
Alta probador - Datos incompletos	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. El probador a dar de alta cuenta con una dirección de correo electrónico válida.	1. El líder de proyecto da clic en la liga "Alta de Probadores". 2. El líder de proyecto omite cualquiera de los datos requeridos por el formulario. 3. El líder de proyecto da clic en el botón "Registrarse".	1. El sistema despliega un mensaje indicando el dato faltante.
Alta probador - Correo inválido	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. La dirección de correo electrónico del probador a dar de alta no es válida (*usuario@servidor*).	1. El líder de proyecto da clic en la liga "Alta de Probadores". 2. El líder de proyecto llena todos los datos solicitados por el formulario. El nombre de usuario no se ha registrado previamente en el sistema; la contraseña y su confirmación son idénticas. 3. El líder de proyecto da clic en el botón "Registrarse".	1. El sistema despliega un mensaje indicando que el correo electrónico introducido no es válido.
Alta probador - Confirmación de contraseña incorrecta	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. El probador a dar de alta cuenta con una dirección de correo electrónico válida.	1. El líder de proyecto da clic en la liga "Alta de Probadores". 2. El líder de proyecto llena todos los datos solicitados por el formulario. El nombre de usuario no se ha registrado previamente en el sistema; la contraseña y su confirmación son distintas. 3. El líder de proyecto da clic en el botón "Registrarse".	1. El sistema despliega un mensaje indicando que la contraseña y su confirmación no son idénticas.

Alta probador - Usuario ya registrado	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. El probador a dar de alta cuenta con una dirección de correo electrónico válida.	1. El líder de proyecto da clic en la liga "Alta de Probadores". 2. El líder de proyecto llena todos los datos solicitados por el formulario. El nombre de usuario ya se encuentra registrado en el sistema; la contraseña y su confirmación son idénticas. 3. El líder de proyecto da clic en el botón "Registrarse".	1. El sistema despliega un mensaje indicando que el nombre de usuario ya se ha registrado.
Alta desarrollador - Alta exitosa	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. El desarrollador a dar de alta cuenta con una dirección de correo electrónico válida.	1. El líder de proyecto da clic en la liga "Alta de Desarrolladores". 2. El líder de proyecto llena todos los datos solicitados por el formulario. El nombre de usuario no se ha registrado previamente en el sistema; la contraseña y su confirmación son idénticas. 3. El líder de proyecto da clic en el botón "Registrarse".	1. El sistema despliega un mensaje indicando que el desarrollador fue dado de alta exitosamente. 2. El sistema envía un correo electrónico a la dirección capturada en el formulario, para indicar al desarrollador que fue dado de alta en el sistema.
Alta desarrollador - Datos incompletos	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. El desarrollador a dar de alta cuenta con una dirección de correo electrónico válida.	1. El líder de proyecto da clic en la liga "Alta de Desarrolladores". 2. El líder de proyecto omite cualquiera de los datos requeridos por el formulario. 3. El líder de proyecto da clic en el botón "Registrarse".	1. El sistema despliega un mensaje indicando el dato faltante.
Alta desarrollador - Correo inválido	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. La dirección de correo electrónico del desarrollador a dar de alta no es válida (*usuario@servidor*).	1. El líder de proyecto da clic en la liga "Alta de Desarrolladores". 2. El líder de proyecto llena todos los datos solicitados por el formulario. El nombre de usuario no se ha registrado previamente en el sistema; la contraseña y su confirmación son idénticas. 3. El líder de proyecto da clic en el botón "Registrarse".	1. El sistema despliega un mensaje indicando que el correo electrónico introducido no es válido.

Alta desarrollador - Confirmación de contraseña incorrecta	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. El desarrollador a dar de alta cuenta con una dirección de correo electrónico válida.	1. El líder de proyecto da clic en la liga "Alta de Desarrolladores". 2. El líder de proyecto llena todos los datos solicitados por el formulario. El nombre de usuario no se ha registrado previamente en el sistema; la contraseña y su confirmación son distintas. 3. El líder de proyecto da clic en el botón "Registrarse".	1. El sistema despliega un mensaje indicando que la contraseña y su confirmación no son idénticas.
Alta desarrollador - Usuario ya registrado	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. El desarrollador a dar de alta cuenta con una dirección de correo electrónico válida.	1. El líder de proyecto da clic en la liga "Alta de Desarrolladores". 2. El líder de proyecto llena todos los datos solicitados por el formulario. El nombre de usuario ya se encuentra registrado en el sistema; la contraseña y su confirmación son idénticas. 3. El líder de proyecto da clic en el botón "Registrarse".	1. El sistema despliega un mensaje indicando que el nombre de usuario ya se ha registrado.
Alta de Producto - Alta exitosa	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. Existen dados de alta al menos un probador y un desarrollador.	1. El líder de proyecto da clic en la liga "Alta de Productos". 2. El líder de proyecto llena todos los datos solicitados por el formulario. El nombre del producto no se encuentra registrado en el sistema. Para indicar a los probadores y desarrolladores asociados al producto, se selecciona el "*check box*" correspondiente en la lista de probadores y en la lista de desarrolladores. 3. El líder de proyecto da clic en el botón "Alta".	1. El sistema despliega un mensaje indicando que el producto fue dado de alta exitosamente.
Alta de Producto - Datos incompletos	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php).	1. El líder de proyecto da clic en la liga "Alta de Productos". 2. El líder de proyecto omite cualquiera de los datos requeridos por el formulario. 3. El líder de proyecto da clic en el botón "Alta".	1. El sistema despliega un mensaje indicando el dato faltante.

	2. Existen dados de alta al menos un probador y un desarrollador.		
Alta de Producto - Producto existente	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. Existen dados de alta al menos un probador y un desarrollador.	1. El líder de proyecto da clic en la liga "Alta de Productos". 2. El líder de proyecto llena todos los datos solicitados por el formulario. El nombre del producto se encuentra registrado en el sistema. Para indicar a los probadores y desarrolladores asociados al producto, se selecciona el "*check box*" correspondiente en la lista de probadores y en la lista de desarrolladores. 3. El líder de proyecto da clic en el botón "Alta".	1. El sistema despliega un mensaje indicando que ya existe un producto con la misma clave.
Reportar Error - Reporte exitoso	1. El probador se encuentra en el menú para el probador (http://www.seminario-sqa.com/tesis/Probador/WebMenuProbador.php). 2. El probador está asociado al menos un producto.	1. El probador da clic en la liga "Reportar Error". 2. El probador llena todos los datos solicitados por el formulario. 3. El probador da clic en el botón "Registrar".	1. El sistema despliega un mensaje informando que se dio de alta el reporte, incluyendo el número de reporte generado. 2. El sistema envía un correo electrónico al líder de proyecto informándole que hay un nuevo reporte de error.
Reportar Error - Datos incompletos	1. El probador se encuentra en el menú para el probador (http://www.seminario-sqa.com/tesis/Probador/WebMenuProbador.php). 2. El probador está asociado al menos un producto.	1. El probador da clic en la liga "Reportar Error". 2. El probador omite cualquiera de los datos solicitados por el formulario. 3. El probador da clic en el botón "Registrar".	1. El sistema despliega un mensaje indicando el dato faltante.
Asignar desarrollador a reporte	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. Se ha dado de alta al menos un reporte de error.	1. El líder de proyecto da clic en la liga "Administración de Reportes". 2. El líder de proyecto selecciona de la lista el reporte que quiere asignar, dando clic en el número de reporte deseado. 3. El sistema despliega el detalle del reporte. 4. El líder de proyecto da clic en la liga "Asignar	1. El sistema despliega un mensaje informando que el reporte fue asignado exitosamente al desarrollador. 2. El sistema envía un correo electrónico al desarrollador informándole que se le asignó el seguimiento a un nuevo reporte. 3. El Reporte desaparece de la lista generada

		desarrollador a Reporte" 5. El sistema presenta un formulario para seleccionar el desarrollador que dará seguimiento al reporte. El líder de proyecto selecciona al desarrollador. 6. El líder de proyecto da clic en el botón "Asignar".	en la pantalla de "Administración de Reportes".
Rechazar reporte - Rechazo exitoso	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. Se ha dado de alta al menos un reporte de error.	1. El líder de proyecto da clic en la liga "Administración de Reportes". 2. El líder de proyecto selecciona de la lista el reporte que quiere rechazar, dando clic en el número de reporte deseado. 3. El sistema despliega el detalle del reporte. 4. El líder de proyecto da clic en la liga "Rechazar Reporte". 5. El sistema presenta un formulario para capturar la causa del rechazo. El líder de proyecto captura la causa. 6. El líder de proyecto da clic en el botón "Aceptar".	1. El sistema despliega un mensaje informando que el reporte fue rechazado exitosamente. 2. El sistema envía un correo electrónico al probador que reportó el error, notificándole que el reporte fue rechazado. 3. El Reporte desaparece de la lista generada en la pantalla de "Administración de Reportes".
Rechazar reporte - Rechazo fallido	1. El líder de proyecto se encuentra en el menú para el líder de proyecto (http://www.seminario-sqa.com/tesis/LiderProyecto/WebMenuLiderProyecto.php). 2. Se ha dado de alta al menos un reporte de error.	1. El líder de proyecto da clic en la liga "Administración de Reportes". 2. El líder de proyecto selecciona de la lista el reporte que quiere rechazar, dando clic en el número de reporte deseado. 3. El sistema despliega el detalle del reporte. 4. El líder de proyecto da clic en la liga "Rechazar Reporte". 5. El sistema presenta un formulario para capturar la causa del rechazo. El líder de proyecto no captura la causa. 6. El líder de proyecto da clic en el botón "Aceptar".	1. El sistema despliega un mensaje informando que no se capturó la causa del rechazo.

Responder reporte - Respuesta exitosa	1. El desarrollador se encuentra en el menú para el desarrollador (http://www.seminario-sqa.com/tesis/Desarrollador/WebMenuDesarrollador.php). 2. Se ha asignado al menos un reporte al desarrollador.	1. El desarrollador da clic en la liga "Administrar Reportes Asignados". 2. El desarrollador selecciona de la lista el reporte que quiere responder, dando clic en el número de reporte deseado. 3. El sistema despliega el detalle del reporte y un formulario para capturar el diagnóstico, etapa de inyección y corrección. 4. El desarrollador captura todos los datos requeridos en el formulario. 5. El desarrollador da clic en el botón "Aceptar".	1. El sistema despliega un mensaje indicando que la respuesta al reporte se guardó exitosamente. 2. El sistema envía un correo electrónico al probador que dio de alta el reporte, notificándole que se capturó una respuesta al reporte.
Responder reporte - Error al guardar respuesta	1. El desarrollador se encuentra en el menú para el desarrollador (http://www.seminario-sqa.com/tesis/Desarrollador/WebMenuDesarrollador.php). 2. Se ha asignado al menos un reporte al desarrollador.	1. El desarrollador da clic en la liga "Administrar Reportes Asignados". 2. El desarrollador selecciona de la lista el reporte que quiere responder, dando clic en el número de reporte deseado. 3. El sistema despliega el detalle del reporte y un formulario para capturar el diagnóstico, etapa de inyección y corrección. 4. El desarrollador omite cualquiera de los datos requeridos por el formulario. 5. El desarrollador da clic en el botón "Aceptar".	1. El sistema despliega un mensaje indicando el dato faltante.
Cerrar reporte - Cierre exitoso	1. El probador se encuentra en el menú para el probador (http://www.seminario-sqa.com/tesis/Probador/WebMenuProbador.php). 2. El probador ha dado de alta al menos un reporte de error.	1. El probador da clic en la liga "Revisar Respuestas". 2. El probador selecciona de la lista el reporte a cerrar dando clic en el número de reporte deseado. 3. El sistema despliega el detalle del reporte, y un formulario con un "*check box*" para cerrar el reporte. 4. El probador activa el "*check box*".	1. El sistema despliega un mensaje informando que el reporte se ha cerrado. 2. El reporte deja de aparecer en la lista generada en la pantalla "Revisar Respuestas". 3. El reporte deja de aparecer en la lista generada en la pantalla "Administrar Reportes Asignados" del desarrollador.

		5. El probador da clic en el botón "Aceptar".	
Cerrar reporte - Cierre pendiente	1. El probador se encuentra en el menú para el probador (http://www.seminario-sqa.com/tesis/Probador/WebMenuProbador.php). 2. El probador ha dado de alta al menos un reporte de error.	1. El probador da clic en la liga "Revisar Respuestas". 2. El probador selecciona de la lista el reporte a cerrar dando clic en el número de reporte deseado. 3. El sistema despliega el detalle del reporte, y un formulario con un "*check box*" para cerrar el reporte. 4. El probador no activa el "*check box*". 5. El probador da clic en el botón "Aceptar".	1. El reporte continúa apareciendo en la lista generada en la pantalla "Revisar Respuestas". 2. El reporte continúa apareciendo en la lista generada en la pantalla "Administrar Reportes Asignados" del desarrollador.

Anexo C

Manual de instalación

C.1 Instalación en los equipos cliente

El SIARE es una aplicación Web, por lo tanto los clientes de la aplicación no necesitan instalar ningún sofware para acceder al sistema. Sólo es necesario contar con un explorador de internet compatible con HTML 1.0 y Javascript. El SIARE se probó con el explorador Microsoft Internet Explorer 5.50. Los usuarios pueden acceder al sistema desde la dirección http://www.seminario-sqa.com/tesis/index.php.

C.2 Instalación en el servidor

El SIARE puede instalarse en cualquier servidor Web habilitado para ejecutar scripts de PHP, con el manejador de base de datos MySql y el servicio de correo Sendmail. Como precondición para instalar el SIARE, el servidor destino debe contar con el siguiente software utilitario:

- Servidor Web capaz de ejecutar scripts de PHP. Se recomienda Apache versión 1.3.23.
- Intérprete de scripts PHP versión 4.1.2 o superior.
- Manejador de base de datos MySql versión 3.23.53 o superior.
- Servicio de correo Sendmail.

C.2.1 Creación de la base de datos

El primer paso para crear la base de datos es configurar un usuario en MySql, que se utilizará en todas las transacciones que ejecute el SIARE. El siguiente comando crea un usuario en MySql y le otorga todos los privilegios necesarios.

```
mysql>insert into user (host, user, password) values
('localhost','TESIS','TESIS');
mysql>grant all privileges on *.* to TESIS@localhost identified by
'TESIS'
```

A continuación, se ejecutan los scripts de creación de ambiente, creación de tablas y llenado de catálogos. Es necesario que se ejecuten en el orden especificado.

```
>type creacionAmbiente.sql|mysql -u TESIS -p TESIS
>type creacionTablas.sql|mysql -u TESIS -p TESIS
>type llenadoCatalogos.sql|mysql -u TESIS -p TESIS
```

C.2.2 Creación de la estructura de directorios y archivos

Una vez configurada la base de datos, se procede a crear la estructura de directorios y copiar los scripts dentro del directorio de documentos del servidor Web, hasta llegar a la siguiente estructura:

```
 El volumen de la unidad C es HPNOTEBOOK
 El número de serie del volumen es 1719-1AFC

Directorio de C:\sambar44\docs\tesis

.              <DIR>        30/12/02  6:54p .
..             <DIR>        30/12/02  6:54p ..
LIDERP~1       <DIR>        30/12/02  7:33p LiderProyecto
ADMINI~1       <DIR>        30/12/02  7:32p Administrador
PROBADOR       <DIR>        30/12/02  7:33p Probador
PATH     PHP           306  28/02/03  7:39p Path.php
DESARR~1       <DIR>        30/12/02  7:33p Desarrollador
SEGURI~1       <DIR>        30/12/02  7:33p Seguridad
SOLICI~1       <DIR>        30/12/02  7:33p Solicitud
ACTORES        <DIR>        30/12/02  7:34p Actores
REPORTES       <DIR>        30/12/02  7:34p Reportes
BASEDA~1       <DIR>        30/12/02  7:34p BaseDatos
ESTADI~1       <DIR>        30/12/02  7:34p Estadisticas
UTILER~1       <DIR>        30/12/02  7:35p Utilerias
MENSAJ~1       <DIR>        30/12/02  7:36p Mensajeria
IMAGENES       <DIR>        16/01/03  8:27p imagenes
INDEX    PHP         8,315  28/02/03  7:39p index.php
DIRECT~1 TXT             0  23/04/03 10:14p directorio.txt
         3 archivos          8,621 bytes
```

```
Directorio de C:\sambar44\docs\tesis\Actores

.              <DIR>        30/12/02  7:34p .
..             <DIR>        30/12/02  7:34p ..
TIPOUS~1 PHP         1,249  28/02/03  7:39p TipoUsuario.php
USUARIO  PHP         4,014  28/02/03  7:39p Usuario.php
ADMINI~1 PHP           409  28/02/03  7:39p Administrador.php
LIDERD~1 PHP         2,501  28/02/03  7:39p LiderdeProyecto.php
PROBADOR PHP         3,761  28/02/03  7:39p Probador.php
DESARR~1 PHP         3,993  28/02/03  7:39p Desarrollador.php
         6 archivos         15,927 bytes

Directorio de C:\sambar44\docs\tesis\Administrador

.              <DIR>        30/12/02  7:32p .
..             <DIR>        30/12/02  7:32p ..
CTRLAU~1 PHP           974  28/02/03  7:39p CtrlAutorizaSolicitud.php
WEBMEN~1 PHP         6,673  28/02/03  7:39p WebMenuAdministrador.php
TXAUTO~1 PHP         3,545  28/02/03  7:39p TxAutorizaSolicitud.php
WEBADM~1   PHP                             8,152      28/02/03      7:39p
WebAdministracionSolicitudes.php
WEBAUT~1 PHP        12,774  28/02/03  7:39p WebAutorizaSolicitud.php
         5 archivos         32,118 bytes

Directorio de C:\sambar44\docs\tesis\BaseDatos

.              <DIR>        30/12/02  7:34p .
..             <DIR>        30/12/02  7:34p ..
LISTA    PHP         4,696  28/02/03  7:39p Lista.php
CONEXION PHP         2,933  28/02/03  7:39p Conexion.php
PAGINA   PHP         1,501  28/02/03  7:39p Pagina.php
QUERY    PHP         5,650  28/02/03  7:39p Query.php
RESULT~1 PHP         2,310  28/02/03  7:39p Resultset.php
PARAME~1 PHP           258  28/02/03  7:39p ParametrosConexion.php
         6 archivos         17,348 bytes

Directorio de C:\sambar44\docs\tesis\Desarrollador
```

```
.              <DIR>        30/12/02  7:33p .
..             <DIR>        30/12/02  7:33p ..
CTRLRE~1 PHP         1,026  28/02/03  7:39p CtrlResponderReporte.php
TXRESP~1 PHP         1,389  28/02/03  7:39p TxResponderReporte.php
WEBMEN~1 PHP         6,839  28/02/03  7:39p WebMenuDesarrollador.php
WEBADM~1   PHP                              8,845      28/02/03      7:39p
WebAdministrarReportesAsignados.php
WEBRES~1 PHP        11,391  28/02/03  7:39p WebResponderReporte.php
         5 archivos         29,490 bytes

Directorio de C:\sambar44\docs\tesis\Estadisticas

.              <DIR>        30/12/02  7:34p .
..             <DIR>        30/12/02  7:34p ..
DISTRI~1   PHP                              3,718      28/02/03      7:39p
DistribucionEtapaInyeccion.php
ESTATU~2 PHP         4,089  28/02/03  7:39p EstatusxPeriodo.php
DISTRI~2 PHP         3,797  28/02/03  7:39p DistribucionSeveridad.php
DISTRI~3 PHP         3,723  28/02/03  7:39p DistribucionTipo.php
DISTRI~4 PHP         3,058  28/02/03  7:39p DistribucionModulo.php
ESTATU~1 PHP         2,791  28/02/03  7:39p EstatusxProducto.php
TIPOXE~1 PHP         4,633  28/02/03  7:39p TipoxEtapa.php
TIPOXS~1 PHP         4,756  28/02/03  7:39p TipoxSeveridad.php
TIEMPO~1 PHP         3,169  28/02/03  7:39p TiempoCierre.php
         9 archivos         33,734 bytes

Directorio de C:\sambar44\docs\tesis\imagenes

.              <DIR>        16/01/03  8:27p .
..             <DIR>        16/01/03  8:27p ..
ATTRIBCR JPG         8,226  28/02/03  7:39p attribcr.jpg
BACKGR01 GIF           216  28/02/03  7:39p backgr01.gif
BG-EARTH JPG        16,546  28/02/03  7:39p bg-earth.jpg
DACCESS  JPG         4,933  28/02/03  7:39p DAccess.jpg
DELETE   GIF           122  28/02/03  7:39p delete.gif
ED-ITEM  GIF           179  28/02/03  7:39p ed-item.gif
HELPL    GIF           627  28/02/03  7:39p helpl.gif
HOME     GIF           688  28/02/03  7:39p home.gif
```

```
POBREG   GIF           153  28/02/03  7:39p pobreg.gif
POBTRANS GIF            43  28/02/03  7:39p pobtrans.gif
RED      GIF            42  28/02/03  7:39p red.gif
SHELP    GIF           145  28/02/03  7:39p shelp.gif
SPACE    GIF           882  28/02/03  7:39p space.gif
UIWIZRLL GIF            88  28/02/03  7:39p uiwizrll.gif
UIWIZRLR GIF            89  28/02/03  7:39p uiwizrlr.gif
UIWIZRUL GIF           851  28/02/03  7:39p uiwizrul.gif
UIWIZRUR GIF           852  28/02/03  7:39p uiwizrur.gif
WSD      GIF        10,279  28/02/03  7:39p wsd.gif
WWCBAN   JPG         3,093  28/02/03  7:39p wwcban.jpg
W_128X52 JPG         2,726  28/02/03  7:39p w_128x52.jpg
        20 archivos         50,780 bytes

Directorio de C:\sambar44\docs\tesis\LiderProyecto

.              <DIR>        30/12/02  7:33p .
..             <DIR>        30/12/02  7:33p ..
CTRLAL~1   PHP                               776      28/02/03      7:39p
CtrlAltaSolicitudInscripcion.php
CTRLAL~2 PHP         1,057  28/02/03  7:39p CtrlAltaProbador.php
TXALTA~2 PHP         2,850  28/02/03  7:39p TxAltaProbador.php
TXALTA~3 PHP         1,463  28/02/03  7:39p TxAltaProducto.php
CTRLAL~3 PHP         1,084  28/02/03  7:39p CtrlAltaDesarrollador.php
CTRLAL~4 PHP         1,533  28/02/03  7:39p CtrlAltaProducto.php
TXALTA~4   PHP                           3,620      28/02/03      7:39p
TxAltaSolicitudInscripcion.php
TXRECH~1 PHP         1,447  28/02/03  7:39p TxRechazarReporte.php
CTRLRE~1 PHP           905  28/02/03  7:39p CtrlRechazarReporte.php
TXASIG~1 PHP         1,834  28/02/03  7:39p TxAsignarDesarrollador.php
TXALTA~1 PHP         2,985  28/02/03  7:39p TxAltaDesarrollador.php
CTRLAS~1 PHP           996  28/02/03  7:39p CtrlAsignarDesarrollador.php
WEBMEN~1 PHP         7,793  28/02/03  7:39p WebMenuLiderProyecto.php
WEBALT~1   PHP                          10,734      28/02/03      7:39p
WebAltaSolicitudInscripcion.php
WEBALT~2 PHP        11,680  28/02/03  7:39p WebAltaProbador.php
WEBALT~3 PHP        11,937  28/02/03  7:39p WebAltaDesarrollador.php
WEBGEN~1 PHP         7,400  28/02/03  7:39p WebGenerarEstadisticas.php
```

```
WEBALT~4 PHP        10,985  28/02/03  7:39p WebAltaProducto.php
WEBDET~1 PHP         9,436  28/02/03  7:39p WebDetalleReporte.php
WEBADM~1 PHP         8,374  28/02/03  7:39p WebAdministracionReportes.php
CTRLGE~1 PHP         3,287  28/02/03  7:39p CtrlGenerarEstadisticas.php
WEBREC~1 PHP         6,972  28/02/03  7:39p WebRechazarReporte.php
WEBASI~1 PHP         8,270  28/02/03  7:39p WebAsignarDesarrollador.php
WEBPAR~1 PHP         9,819  28/02/03  7:39p WebParametrosEstadisticas.php
        24 archivos        127,237 bytes

Directorio de C:\sambar44\docs\tesis\Mensajeria

.              <DIR>        30/12/02  7:36p .
..             <DIR>        30/12/02  7:36p ..
CORREO   PHP         2,163  28/02/03  7:39p Correo.php
         1 archivos          2,163 bytes

Directorio de C:\sambar44\docs\tesis\Probador

.              <DIR>        30/12/02  7:33p .
..             <DIR>        30/12/02  7:33p ..
TXREPO~1 PHP         3,599  28/02/03  7:39p TxReportarError.php
CTRLCE~1 PHP         1,056  28/02/03  7:39p CtrlCerrarReporte.php
TXCERR~1 PHP           919  28/02/03  7:39p TxCerrarReporte.php
WEBREP~2 PHP        11,315  28/02/03  7:39p WebReportarError.php
WEBMEN~1 PHP         6,858  28/02/03  7:39p WebMenuProbador.php
CTRLRE~2 PHP         1,356  28/02/03  7:39p CtrlReportarError.php
WEBCER~1 PHP         9,765  28/02/03  7:39p WebCerrarReporte.php
WEBREV~1 PHP         8,590  28/02/03  7:39p WebRevisarRespuestas.php
         8 archivos         43,458 bytes

Directorio de C:\sambar44\docs\tesis\Reportes

.              <DIR>        30/12/02  7:34p .
..             <DIR>        30/12/02  7:34p ..
ESTATU~1 PHP         1,230  28/02/03  7:39p EstatusReporte.php
TIPOER~1 PHP         2,347  28/02/03  7:39p TipoError.php
TIPOSE~1 PHP         2,479  28/02/03  7:39p TipoSeveridad.php
ETAPAI~1 PHP         2,419  28/02/03  7:39p EtapaInyeccionError.php
```

```
PRODUCTO PHP         7,564  28/02/03  7:39p Producto.php
REPORT~1 PHP         9,676  28/02/03  7:39p ReporteDeError.php
LISTAR~1 PHP         1,509  28/02/03  7:39p ListaReportesNuevos.php
LISTAR~2 PHP         1,465  28/02/03  7:39p ListaReportesLevantados.php
LISTAR~3 PHP         1,484  28/02/03  7:39p ListaReportesAsignados.php
         9 archivos         30,173 bytes

Directorio de C:\sambar44\docs\tesis\Seguridad

.              <DIR>        30/12/02  7:33p .
..             <DIR>        30/12/02  7:33p ..
CTRLVA~1 PHP         1,507  28/02/03  7:39p CtrlValidaUsuario.php
TXVALI~1 PHP         1,910  28/02/03  7:39p TxValidaUsuario.php
WEBVAL~1 PHP         8,224  28/02/03  7:39p WebValidaUsuario.php
         3 archivos         11,641 bytes

Directorio de C:\sambar44\docs\tesis\Solicitud

.              <DIR>        30/12/02  7:33p .
..             <DIR>        30/12/02  7:33p ..
ESTATU~1 PHP         1,231  28/02/03  7:39p EstatusSolicitud.php
SOLICI~1 PHP         5,583  28/02/03  7:39p SolicitudInscripcion.php
LISTAS~1 PHP         1,300  28/02/03  7:39p ListaSolicitudesNuevas.php
         3 archivos          8,114 bytes

Directorio de C:\sambar44\docs\tesis\Utilerias

.              <DIR>        30/12/02  7:35p .
..             <DIR>        30/12/02  7:35p ..
STATUS   PHP         1,173  28/02/03  7:39p Status.php
DATE     PHP         5,789  28/02/03  7:39p Date.php
URL      PHP         1,625  28/02/03  7:39p Url.php
LINK     PHP         1,226  28/02/03  7:39p Link.php
REDIRE~1 PHP         1,400  28/02/03  7:39p Redireccion.php
FORMAT~1 PHP         9,276  28/02/03  7:39p FormateoTexto.php
VECTOR   PHP         2,088  28/02/03  7:39p Vector.php
SERIAL~2 PHP         1,392  28/02/03  7:39p Serializador.php
         8 archivos         23,969 bytes
```

```
Número total de archivos en la lista:
       110 archivos        434,773 bytes
        41 directorios        4,967,56 MB libres
```

C.2.3 Configuración de parámetros del sistema

Finalmente, se deben configurar los siguientes parámetros:

a) En el script *Path.php*, la ruta del directorio donde se encuentran los scripts.

```
<?
   /* define la ruta donde se buscan los archivos
    * dicha ruta se referenciará por la variable
    * $DOCUMENT_ROOT a la que se debe concatenar
    * el resto de la ruta donde se encuentra en archivo
    */
   $DOCUMENT_ROOT = "c:\\sambar44\\docs\\tesis";
   //$SERVER_NAME = "localhost";
?>
```

b) En el script *ParametrosConexion.php*, los parámetros de conexión a la base de datos.

```
<?php
 /* Header
  * AAR
  * 16/X/2000
  * Define las constantes a utilizarse en la conexion a mysql
  *
  */
  define("SERVIDOR","localhost");
  define("USUARIO_MYSQL","TESIS");
  define("PASSWORD_MYSQL","TESIS");
  define("BD_MYSQL","TESIS");
?>
```

c) En el script *Correo.php*, la dirección y el puerto del servicio de Sendmail, así como la dirección de correo electrónico que usará el sistema para enviar todos los mensajes.

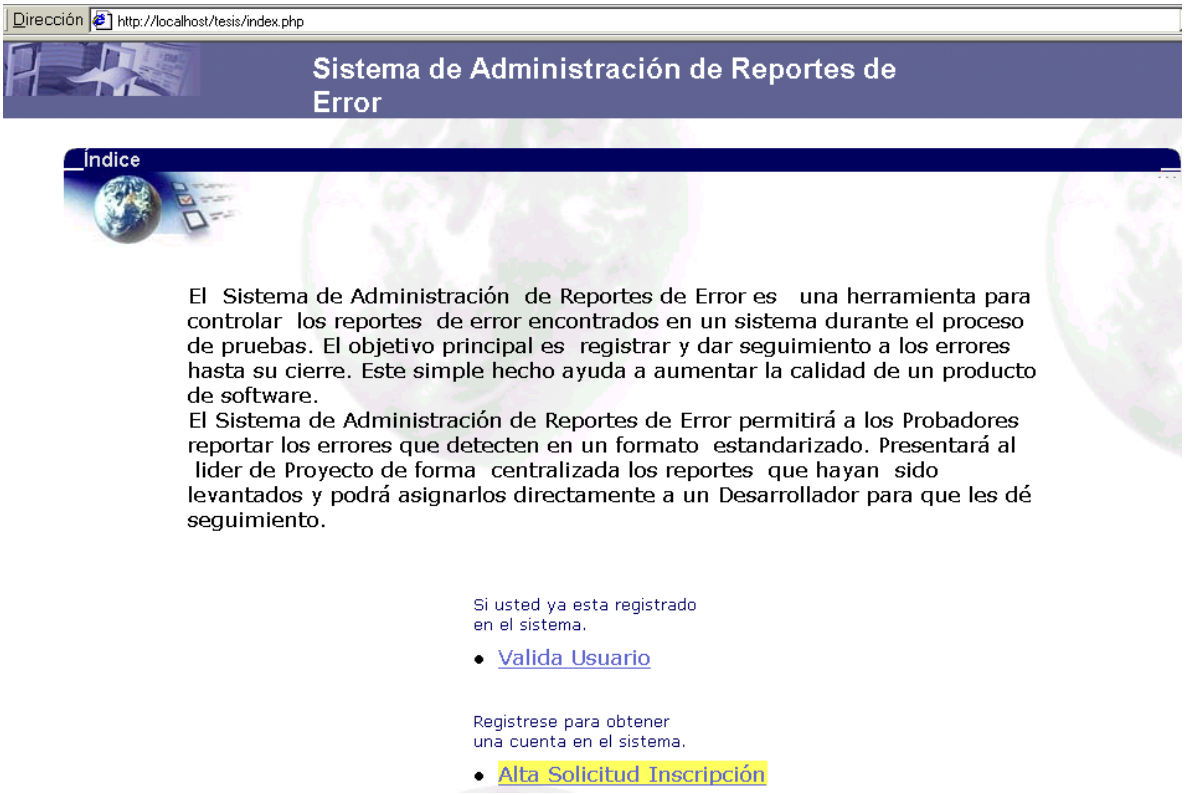
```
define ("SERVIDORSOCKET","localhost");
define ("PUERTO",25);
define ("SERVIDORMAIL","foobar.com");
define ("USUARIO","admin");
define ("DELAY",50000);
```

Anexo D

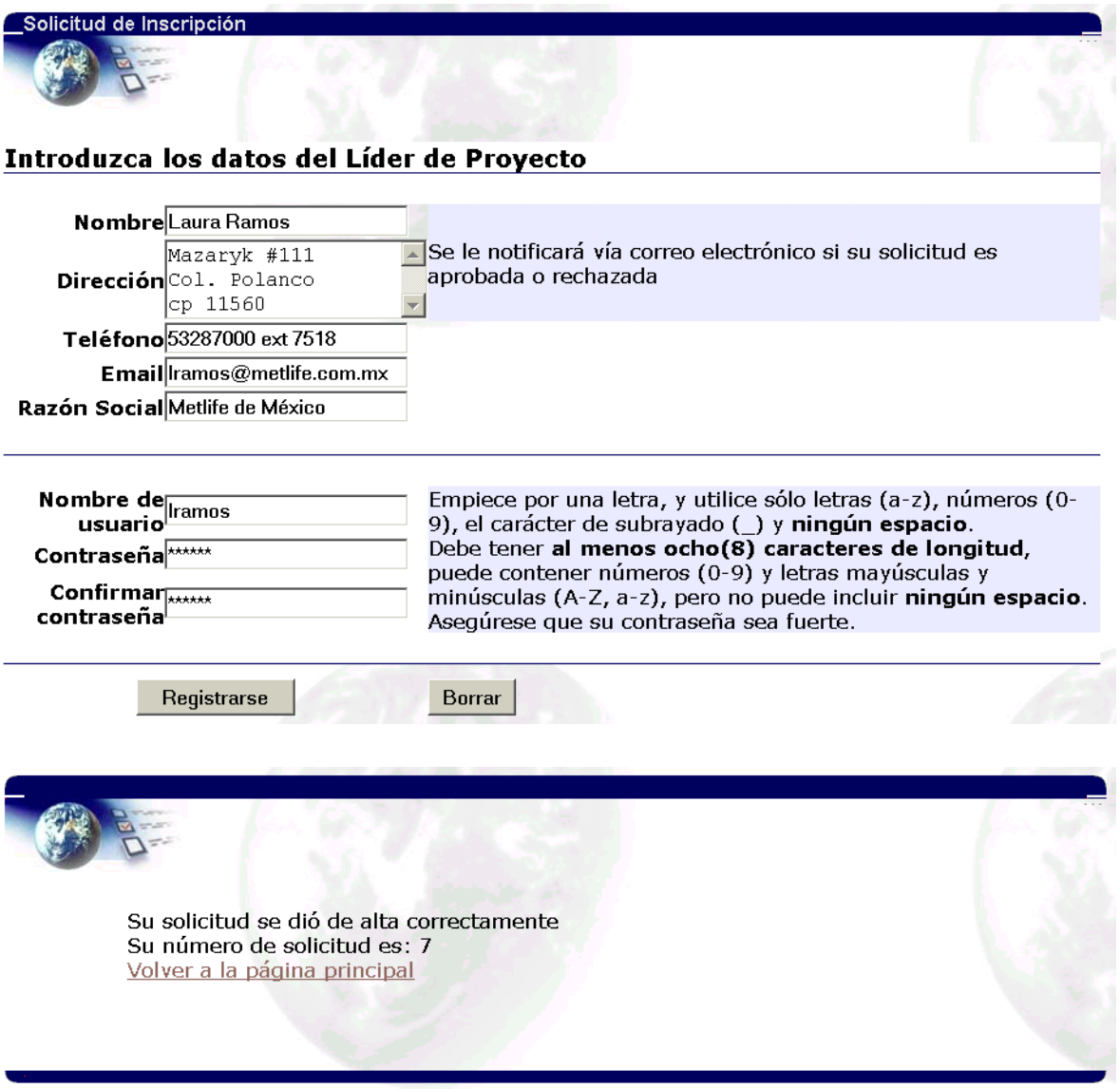
Manual de usuario

D.1 Inscripción al SIARE

Para inscribirse en el SIARE, el líder de proyecto debe acceder a la página principal y seguir la liga "Alta Solicitud Inscripción".



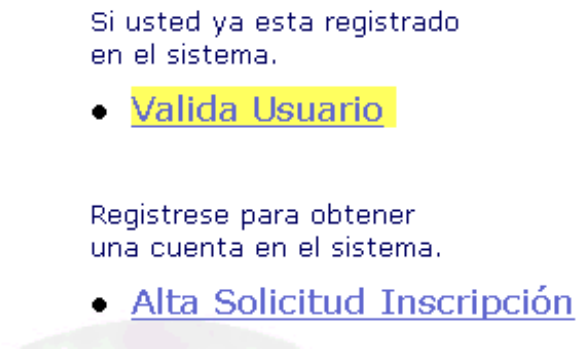
A continuación, el líder de proyecto llena el formulario de inscripción y presiona el botón "Registrarse".



Una vez dada de alta la solicitud de inscripción, el líder de proyecto debe esperar a que el administrador del SIARE la autorice para poder empezar a utilizar el sistema.

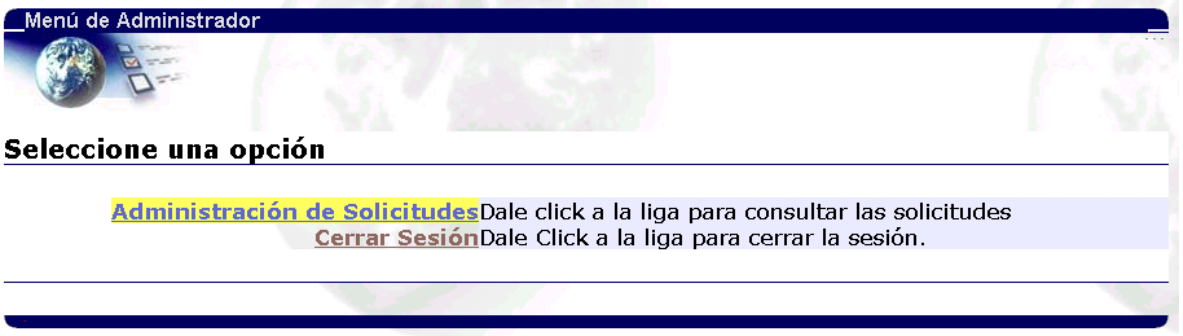
D.2 Autorización de solicitud de inscripción

Para autorizar una solicitud de inscripción el usuario debe tener el perfil de administrador del sistema. En primer lugar necesita acceder a la página principal del SIARE y seguir la liga "Valida Usuario".



El administrador captura su clave de usuario y password en el formulario, y presiona el botón "Valida".

Si el SIARE comprueba que los datos del usuario son correctos, despliega el menú para los administradores. Ahí el usuario selecciona la opción "Administración de Solicitudes".



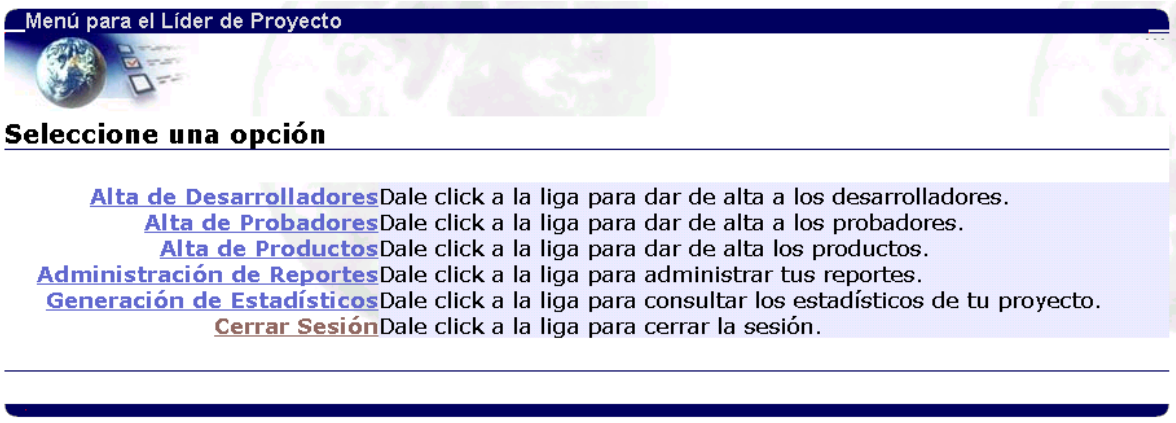
El SIARE despliega una lista de las solicitudes dadas de alta. Al presionar el número de solicitud, el sistema presenta los datos de la misma, y da la opción de autorizarla o rechazarla.

El administrador selecciona la opción deseada y presiona el botón "Cambia". El SIARE notifica al líder de proyecto vía correo electrónico sobre el cambio en el estatus de su solicitud.

D.3 Alta de probadores, desarrolladores y productos

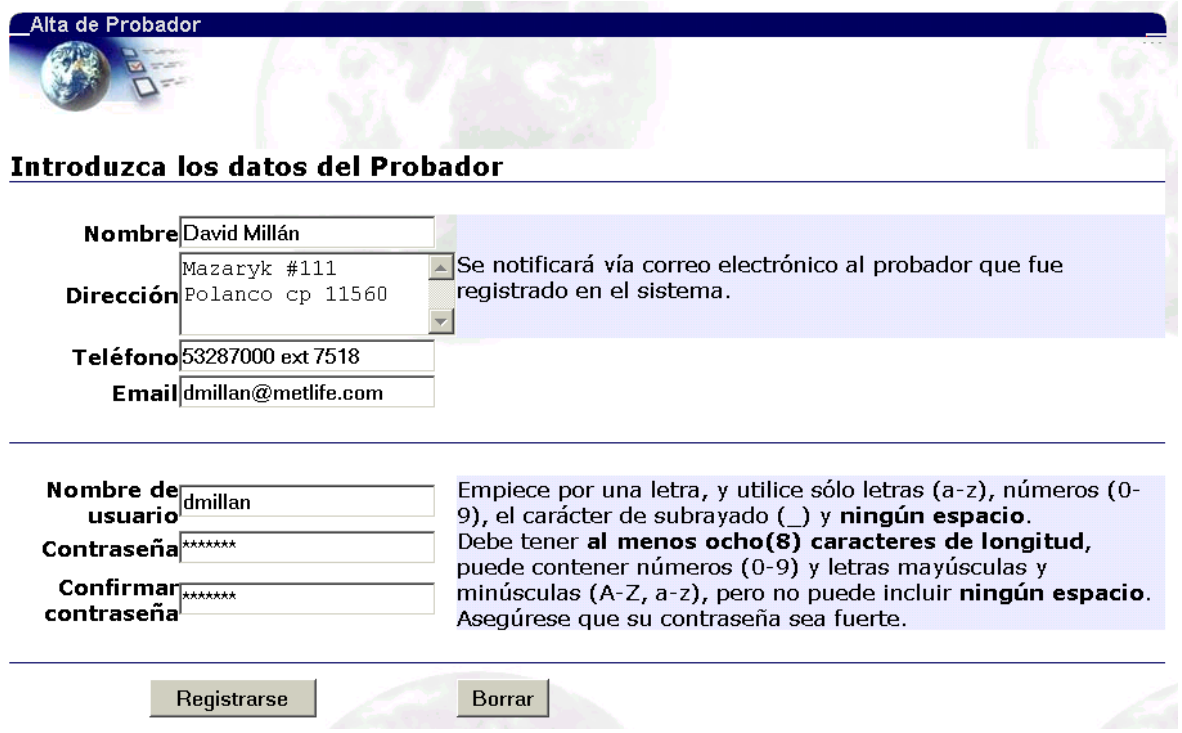
Estas operaciones se llevan a cabo por líderes de proyecto ya autorizados a utilizar el SIARE.

El líder de proyecto debe validarse como usuario del sistema (siguiendo el mecanismo anteriormente descrito para el administrador). Si el SIARE comprueba que los datos del usuario son correctos, despliega el menú para líderes de proyecto.

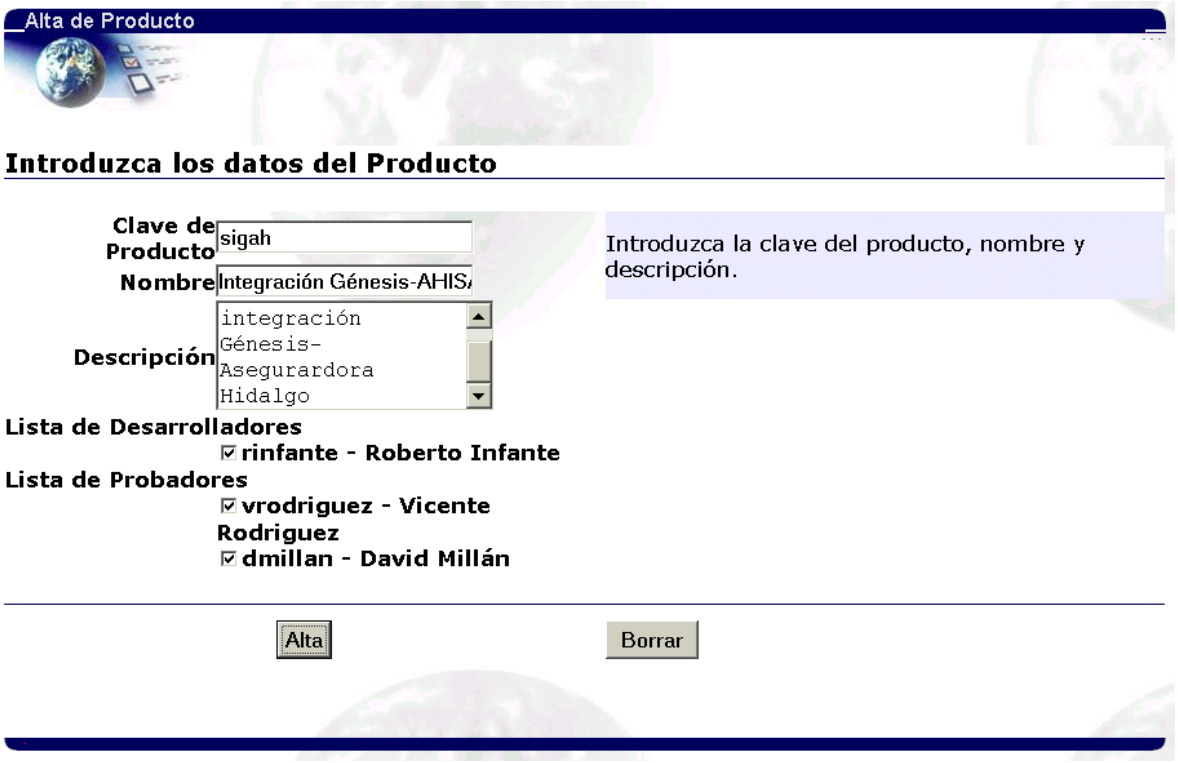


La opción "Alta de Desarrolladores" permite inscribir al SIARE nuevos desarrolladores, encargados de efectuar correcciones en los productos. El sistema despliega un formulario para capturar los datos del desarrollador. Al terminar presione el botón "Registrarse". El SIARE registra en la base de datos al nuevo desarrollador y le notifica vía correo electrónico que ya puede utilizar el sistema.

La opción "Alta de Probadores" permite inscribir al SIARE nuevos probadores que validarán los productos. El sistema despliega un formulario para capturar los datos del probador. Al terminar presione el botón "Registrarse". El SIARE registra en la base de datos al nuevo probador y le notifica vía correo electrónico que ya puede utilizar el sistema.

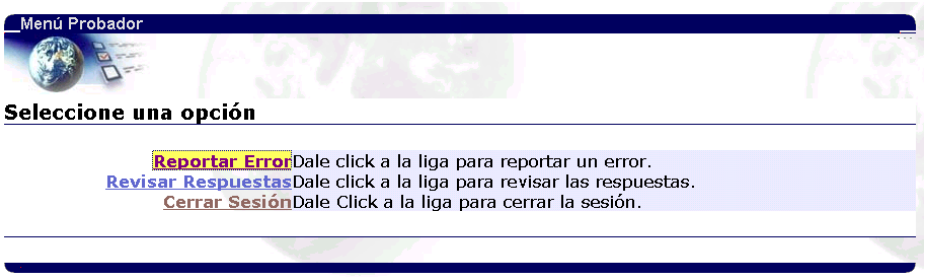


La opción "Alta de Productos" permite registrar nuevos productos a probar. El SIARE presenta un formulario para capturar los datos del producto. Requiere que ya existan desarrolladores y probadores dados de alta en el sistema. Al terminar presione el botón "Alta".



D.4 Alta de reportes

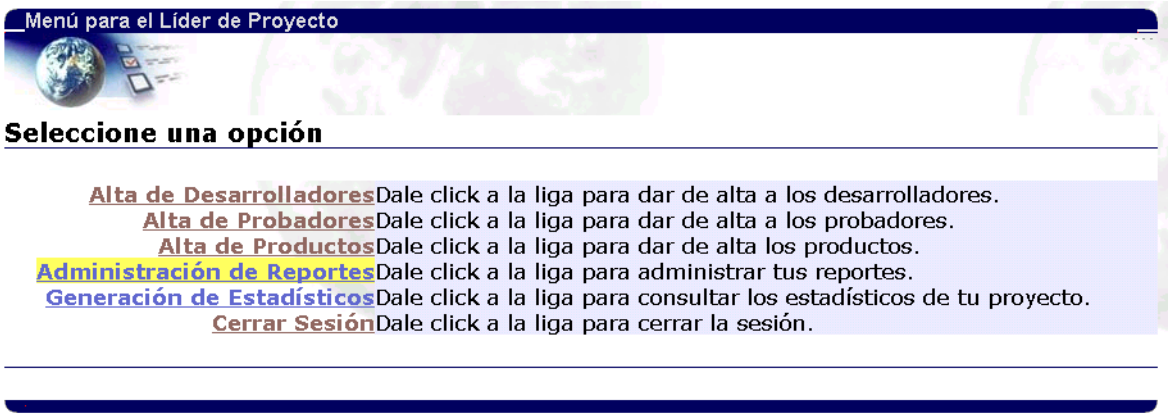
El alta de reportes se lleva a cabo por usuarios con perfil de probador. Requiere que el usuario se valide en el SIARE. Si el sistema comprueba que los datos del usuario son correctos, despliega el menú para probadores.



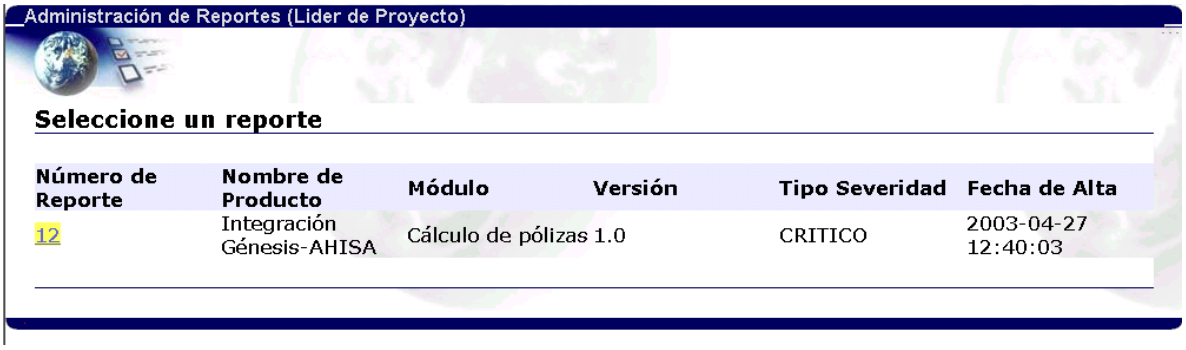
El SIARE despliega un formulario para capturar el reporte del error. Al terminar presione el botón "Registrar". El sistema asigna automáticamente un número de reporte.

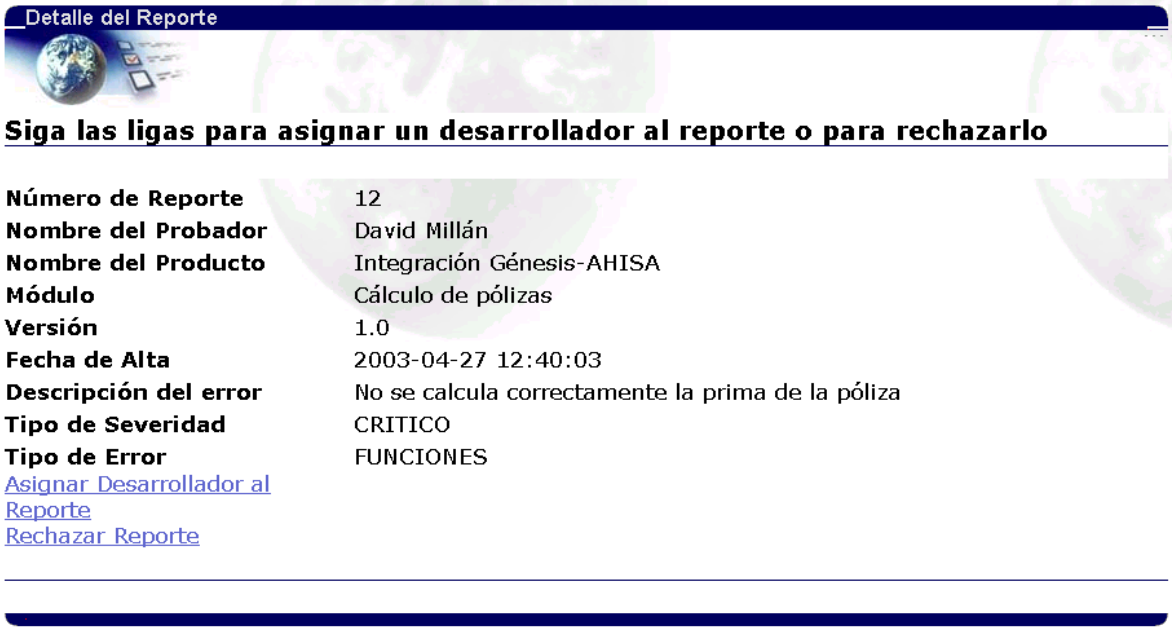
D.5 Administración de reportes

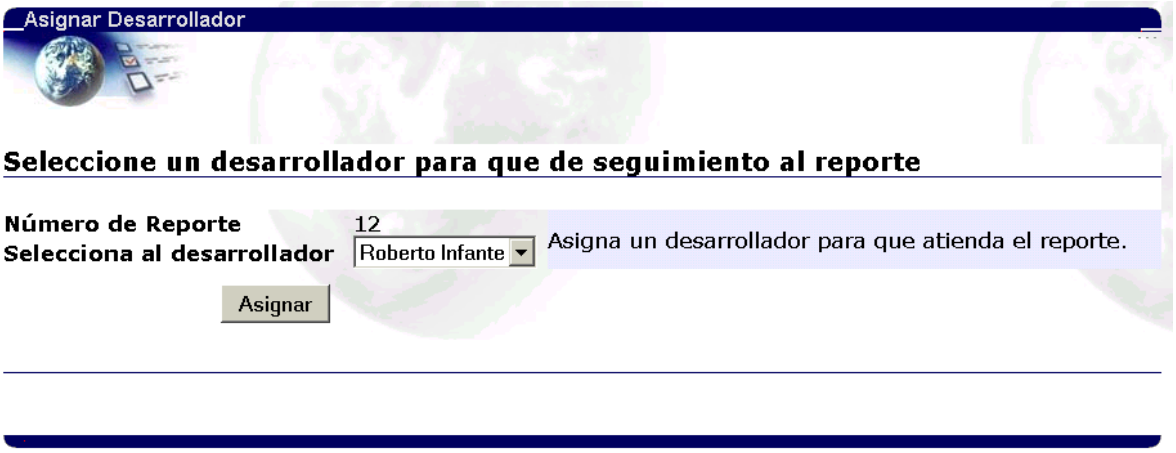
Esta opción permite al líder de proyecto revisar los reportes dados de alta por los probadores y asignarlos a un desarrollador para que les de seguimiento, o bien, permite rechazar un reporte. Requiere haberse validado en el SIARE como líder de proyecto y seguir la opción "Administración de Reportes" del menú.

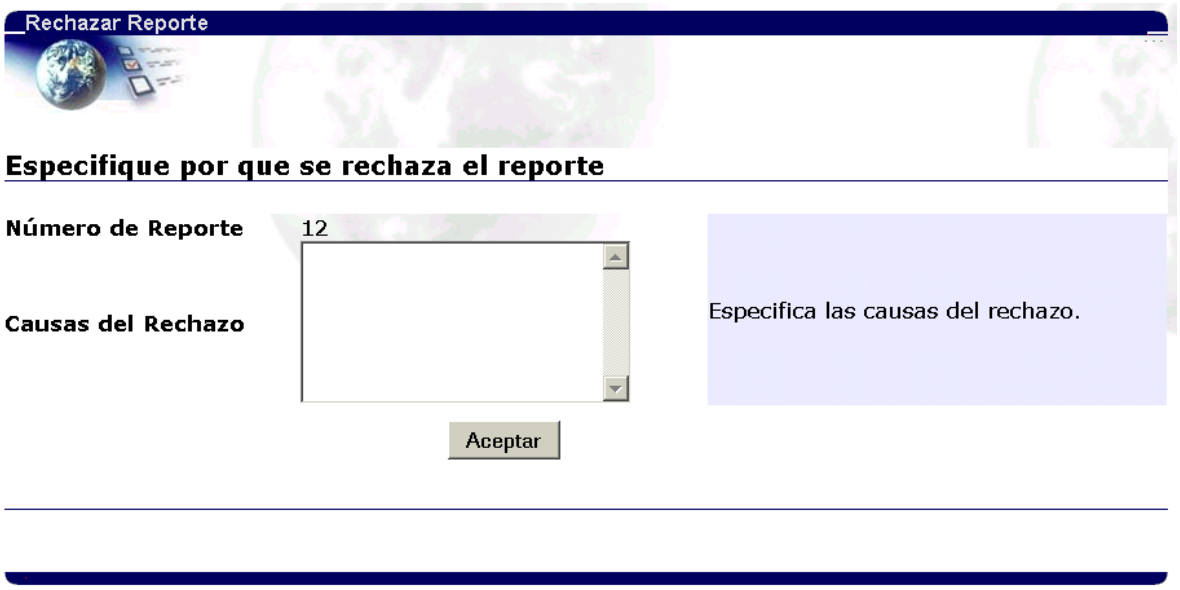


El SIARE despliega una lista de los reportes nuevos. Al presionar el número de reporte, se despliega el detalle del mismo. Se muestran también ligas a formularios que permiten asignar el reporte a un desarrollador o rechazarlo.


Administración de Reportes (Lider de Proyecto)
Seleccione un reporte
Número de Reporte
Nombre de Producto
Módulo
Versión
Tipo Severidad
Fecha de Alta
12
Integración Génesis-AHISA
Cálculo de pólizas
1.0
CRITICO
2003-04-27 12:40:03

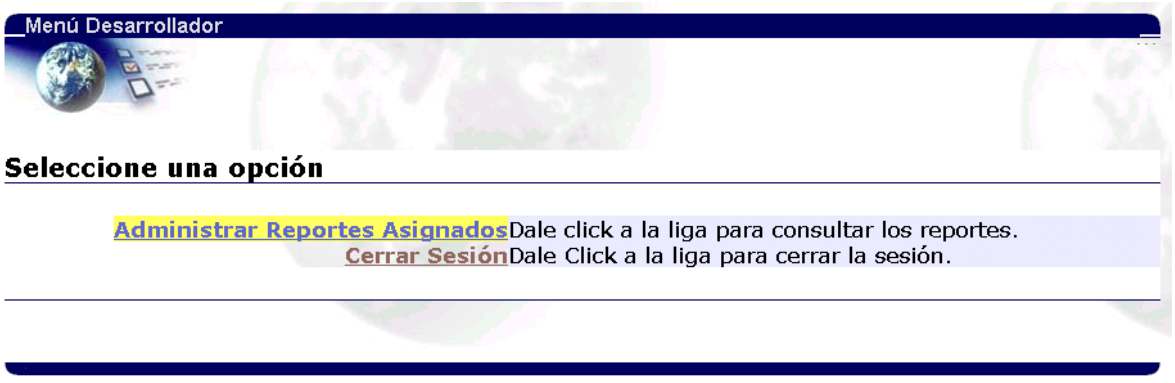

Detalle del Reporte
Siga las ligas para asignar un desarrollador al reporte o para rechazarlo
Número de Reporte 12
Nombre del Probador David Millán
Nombre del Producto Integración Génesis-AHISA
Módulo Cálculo de pólizas
Versión 1.0
Fecha de Alta 2003-04-27 12:40:03
Descripción del error No se calcula correctamente la prima de la póliza
Tipo de Severidad CRITICO
Tipo de Error FUNCIONES
Asignar Desarrollador al Reporte
Rechazar Reporte


Asignar Desarrollador
Seleccione un desarrollador para que de seguimiento al reporte
Número de Reporte 12
Selecciona al desarrollador
Roberto Infante
Asigna un desarrollador para que atienda el reporte.
Asignar



D.6 Responder reportes

Esta funcionalidad corresponde a los desarrolladores que tengan reportes asignados. Requiere que el usuario se valide en el SIARE. A partir del menú para desarrolladores, entre a la opción "Administrar Reportes Asignados". El sistema despliega una lista de los reportes asignados al desarrollador. Al presionar el número de reporte se muestra el detalle del mismo, así como un formulario para capturar la respuesta.



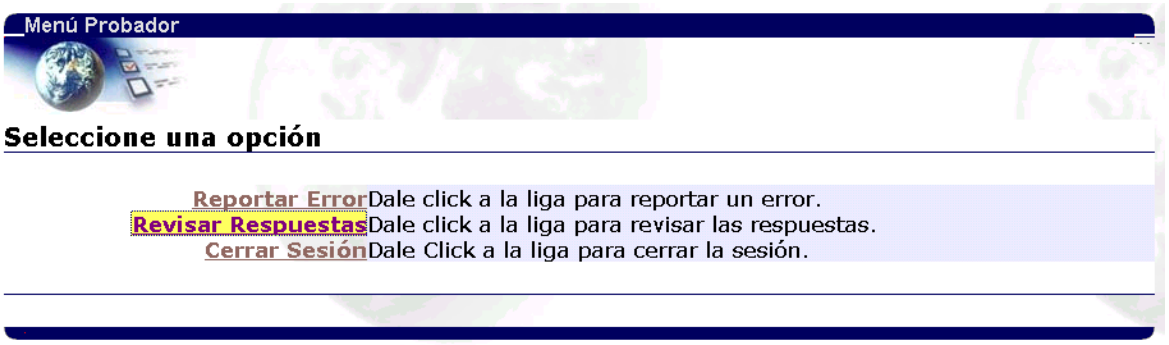
Capture la respuesta al Reporte

Número de Reporte	12
Nombre del Probador	David Millán
Nombre del Producto	Integración Génesis-AHISA
Módulo	Cálculo de pólizas
Versión	1.0
Fecha de Alta	2003-04-27 12:40:03
Descripción del error	No se calcula correctamente la prima de la póliza
Tipo de Severidad	CRITICO
Tipo de Error	FUNCIONES
Estatus	ASIGNADO PARA SEGUIMIENTO
Fecha Ultimo Estatus	2003-04-27 13:00:05
Diagnóstico	No se utilizó el algoritmo correcto para calcular la prima
Etapa de Inyección	DISEÑO DETALLADO
Descripción Corrección	Se corrigió el código para usar el algoritmo correcto

Aceptar Borrar

D.7 Revisar respuestas

Esta funcionalidad está disponible para probadores que hayan reportado errores. Requiere que el usuario se haya validado en el SIARE. El sistema despliega el menú para probadores, y se selecciona la opción "Revisar Respuestas". Entonces se muestra una lista de los reportes dados de alta por el probador. Al presionar el número de reporte, el SIARE despliega el detalle del reporte, la respuesta del desarrollador y presenta un formulario que permite cerrar el reporte.

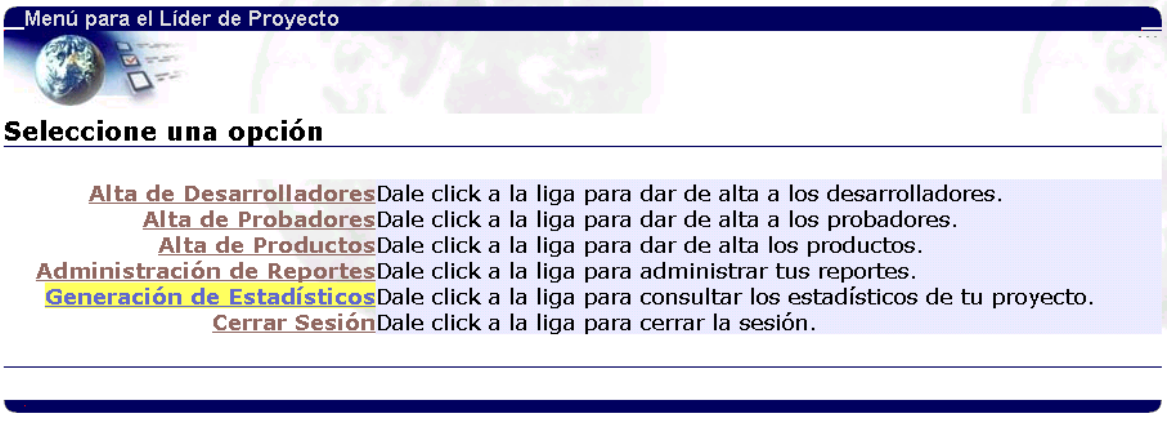


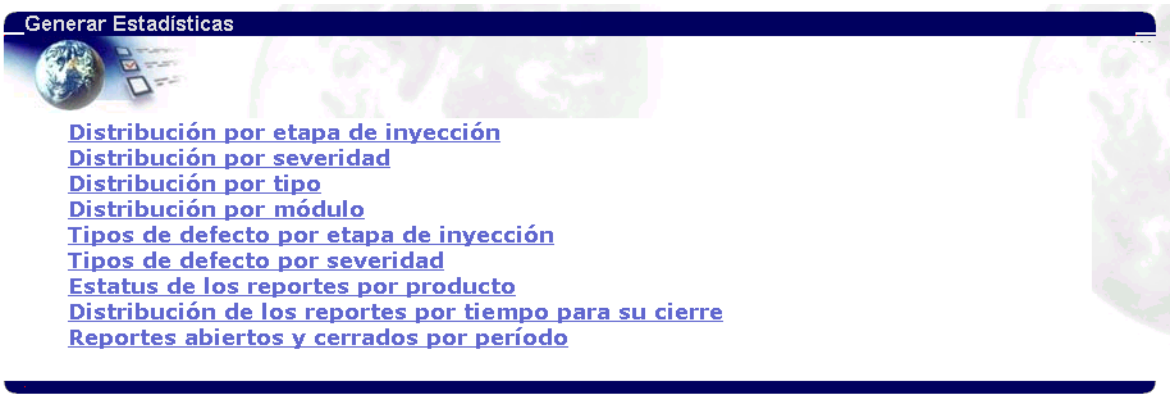
Seleccione un Reporte

Número de Reporte	Nombre del Producto	Módulo	Versión	Tipo de Severidad	Estatus	Fecha Ultimo Estatus
12	Integración Génesis-AHISA	Cálculo de pólizas	1.0	CRITICO	ASIGNADO PARA SEGUIMIENTO	2003-04-27 13:00:05

D.8 Generar estadísticas

Esta funcionalidad está disponible para el líder de proyecto. Una vez que se ha validado como usuario del SIARE, seleccione la opción "Generación de Estadísticos". El sistema despliega una lista de estadísticos disponibles. Al seleccionar alguno, el SIARE genera el reporte correspondiente.




Generar Estadísticas
Distribución por etapa de inyección
Distribución por severidad
Distribución por tipo
Distribución por módulo
Tipos de defecto por etapa de inyección
Tipos de defecto por severidad
Estatus de los reportes por producto
Distribución de los reportes por tiempo para su cierre
Reportes abiertos y cerrados por período

Bibliografía

[ANALIZE] Langaard Software Home Page
http://www.langaard.50megs.com

[APACHE] Apache Web Server Home Page
http://httpd.apache.org

[BOO94] Booch G. *Object-Oriented Analysis and Design with Applications*. 2da edición. Benjamin/Cummings. Redwood. 1994.

[BRO96] Brown N. *Industrial-strenght management strategies*. IEEE Software. 1996.

[BRU02a] Bruegge B., Dutoit A. *Ingeniería de software orientado a objetos*. 1ª edición. Pearson Educación. México. 2002. Págs. 23-61.

[BRU02b] Bruegge B., Dutoit A. *Ingeniería de software orientado a objetos*. 1ª edición. Pearson Educación. México. 2002. Págs. 63-94.

[BRU02c] Bruegge B., Dutoit A. *Ingeniería de software orientado a objetos*. 1ª edición. Pearson Educación. México. 2002. Págs. 97-129.

[BRU02d] Bruegge B., Dutoit A. *Ingeniería de software orientado a objetos*. 1ª edición. Pearson Educación. México. 2002. Págs. 131-165.

[BRU02e] Bruegge B., Dutoit A. *Ingeniería de software orientado a objetos*. 1ª edición. Pearson Educación. México. 2002. Págs. 167-229.

[BRU02f] Bruegge B., Dutoit A. *Ingeniería de software orientado a objetos*. 1ª edición. Pearson Educación. México. 2002. Págs. 231-281.

[BRU02g] Bruegge B., Dutoit A. *Ingeniería de software orientado a objetos*. 1ª edición. Pearson Educación. México. 2002. Págs. 327-369.

[BRU02h] Bruegge B., Dutoit A. *Ingeniería de software orientado a objetos*. 1ª edición. Pearson Educación. México. 2002. Págs. 457-493.

[CRYSTAL] Crystal Decisions Home Page
http://www.crystaldecisions.com

[FOW99a] Fowler M. *UML gota a gota*. 1ª edición. Addison-Wesley. México. 1999. Págs. 49-59.

[FOW99b] Fowler M. *UML gota a gota*. 1ª edición. Addison-Wesley. México. 1999. Págs. 147-160.

[FOW99c] Fowler M. *UML gota a gota*. 1ª edición. Addison-Wesley. México. 1999. Págs. 137-145.

[IEEE Std. 982-1989] *IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software*. IEEE Standards Board. 1989.

[IEEE Std. 1074-1995] *IEEE Standard for Developing Software Life Cycle Processes*. IEEE Computer Society. New York. 1995.

[JAC92] Jacobson I., Christerson M., Jonsson P., Overgaard G. *Object-Oriented Software Engineering - A Use Case Driven Approach.* Addison-Wesley. Reading, MA. 1992.

[JAC99] Jacobson I., Booch G., Rumbaugh J. *The Unified Software Development Process*. Addison-Wesley. Reading, MA. 1999.

[JAL02] Jalote P. *Software Project Management in Practice*. 1ª edición. Addison-Wesley. Boston. 2002. Págs. 109-125.

[KRU97] Kruse R., Tondo C., Leung P. *Data Structures and Program Design in C*. 2da edición. Prentice Hall. New Jersey. 1997. Pág. 35.

[KEN97] Kendall K., Kendall J. *Análisis y Diseño de Sistemas*. 3ª edición. Prentice Hall. México. 1997. Pág. 585.

[MAR00a] Maruyama H., Tamura K., Uramoto N. *Creación de Sitios Web con XML y Java*. 1ª edición. Prentice Hall. Madrid. 2000. Págs. 7-9.

[MAR00b] Maruyama H., Tamura K., Uramoto N. *Creación de Sitios Web con XML y Java*. 1ª edición. Prentice Hall. Madrid. 2000. Págs. 246-254.

[MYSQL] MySQL Home Page
http://www.mysql.com

[PHP] PHP Home Page
http://www.php.net

[PMI96] Project Management Institute. *A Guide to the Project Management Body of Knowledge*. EUA. 1996.

[RUM91] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W. *Object-Oriented Modeling and Design*. Prentice Hall. New Jersey. 1991.

[SHA96] Shaw M., Garlan D. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall. New Jersey. 1996.