Universidad de las Ciencias Informáticas

Facultad 7

Título: Diseño de una herramienta para la gestión de soluciones a problemas detectados durante el desarrollo de software

Trabajo de Diploma para optar por el Título de
Ingeniero en Ciencias Informáticas

Autora: Daimi Moreno Salas

Tutor: Ing. Lázaro González Rodríguez

La Habana, junio 2011
"Año 53 de la Revolución"

DATOS DE CONTACTO

Tutores:

Tutor principal: Lázaro González Rodríguez (lgonzalezr@uci. cu).

Graduado de Ingeniero en Ciencias Informáticas en el 2007. Se ha desempeñado como profesor del Departamento de Técnicas de Programación de la Facultad 7 y como analista en el Departamento de Software Médico Imagenológico del Centro de Informática Médica.

DEDICATORIA

A mi abuelo, mi mayor fuente de inspiración y a mi familia.

AGRADECIMIENTOS

A todos los que me ayudaron y contribuyeron con un granito de arena a que hoy pueda graduarme.

RESUMEN

El objetivo fundamental del presente trabajo, es el Diseño de una herramienta para la gestión de soluciones a problemas detectados durante el desarrollo de software. En el mundo son muchos los sitios y comunidades online que se dedican a dar soluciones a errores que se le presentan a los desarrolladores en su quehacer diario.

Estos utilizan técnica de corrección de defectos en dependencia de la experiencia que tengan en este campo, algunas de ellas son: Ensayo y error, Depuración manual y Búsqueda de información externa.

Para el diseño de la aplicación se realizó la investigación de los diferentes sistemas existentes a nivel nacional e internacional, analizando sus principales características. Con el objetivo de hacer una descripción de los mismos y ver las principales funcionalidades de utilidad para los desarrolladores; logrando determinar cuáles eran los requerimientos necesarios para el diseño de dicha aplicación.

Con el desarrollo y puesta en práctica de la aplicación para la gestión de soluciones a problemas detectados durante el desarrollo de software se le ofrecerá a los desarrolladores facilidades para la búsqueda de soluciones a los errores que le surgen, de manera rápida y eficiente logrando mejores resultados a más corto plazo.

Palabras claves:

Error, problema, solución, desarrollo, software.

TABLA DE CONTENIDOS

INTRODUCCIÓN

El surgimiento de las computadoras personales, la informática y las redes de comunicaciones durante la segunda mitad del siglo XX, han dotado a la humanidad de medios para la expansión del conocimiento, la cultura y la sabiduría. (1)

La informática puede definirse como la convergencia de los fundamentos de las ciencias de la computación, la programación y las metodologías para el desarrollo de software, la arquitectura de las computadoras, las redes de datos como Internet, la inteligencia artificial, así como determinados temas de electrónica (2). Estas tecnologías de la información y las comunicaciones juegan un importante papel en la actual revolución científico-técnica, siendo Internet hoy en día el medio por excelencia para la búsqueda de información. Todo ello ha cambiado la cultura del mundo, facilitándose la comunicación con el correo electrónico (e-mail), y la mensajería instantánea. Además baja el costo y mejora la calidad de los servicios en diferentes actividades tales como la salud, las industrias, las dependencias gubernamentales y otros servicios sociales.

La informática en Cuba se ha tratado de insertar en la sociedad, haciéndola accesible a todos mediante los Joven Club de Computación y Electrónica (JCCE); los que actualmente se pueden encontrar hasta en los más intrincados municipios del país.

Esta, además ha sido introducida en otros sectores, como el de la salud. Por ejemplo en las distintas Facultades de Medicina del país, se ha tomado en serio la necesidad de ampliar el conocimiento en la informática. Dado a que, el uso de las Tecnologías de la Informática, y las Comunicaciones (TIC) e Internet, además de incrementar la cultura general integral de los profesionales de la salud, los ayuda en su carrera profesional; desarrolla espacios de aprendizaje permanente e investigación en la gestión de la información y el conocimiento en la salud.

Adicionalmente, en la educación cubana se ha realizado un gran esfuerzo con el objetivo de introducir la enseñanza de la informática en todos los niveles de la educación: desde la primaria hasta la enseñanza superior, haciendo todo lo posible porque todos y cada uno de los estudiantes, tanto cubanos como extranjeros, tengan acceso a las TIC y los beneficios de éstas.

Al calor de la Batalla de Ideas surge la Universidad de las Ciencias Informáticas (UCI) a partir de una idea del Comandante en Jefe, con el propósito de constituir un centro de altos estudios vinculado a la

producción. Donde los estudiantes a partir del tercer año de la carrera, incluyendo algunos de los más aventajados de años inferiores, se vinculan a la producción en los centros de desarrollo de software; bajo la tutoría de gran parte de los profesores que también se vinculan a la esfera productiva. En estos centros, estudiantes y profesores trabajan en proyectos productivos reales, con el objetivo de desarrollar soluciones de software de primer nivel y de este modo contribuir con la economía del país.

El hecho de que la principal fuerza productiva de la universidad sean los propios estudiantes entraña un problema; y es que la experiencia acumulada por los estudiantes en los 5 años de su formación y durante su participación en proyectos productivos, se va junto a éstos al graduarse y marchar a sus localidades de origen. Sólo aquellos graduados que permanecen en la universidad pueden llegar a pasar sus conocimientos a las nuevas generaciones de estudiantes. De ahí que sea necesario lograr que el conocimiento, la experiencia y la información trasciendan de una generación a otra de estudiantes para así maximizar el potencial de éstos y obtener mejores resultados a más corto plazo.

En la actualidad el desarrollo de software se ha convertido en un proceso muy complejo. Los usuarios exigen software de gran calidad que implica desarrollar sistemas de decenas de miles de líneas de código, libre de errores, con requerimientos no funcionales complicados (3). Este es construido comúnmente por equipos de desarrollo que utilizan tecnologías de punta, algunas veces no del todo estable. Una etapa crucial en el desarrollo de aplicaciones de software es la fase de pruebas. La inyección de defectos por actividad humana es inevitable a pesar de los esfuerzos por incluir actividades de detección temprana de defectos durante el desarrollo. En la fase de pruebas, los defectos que se detectan se deben corregir e integrar en la versión que se entregará a los clientes.

La eliminación de defectos en este tipo de aplicaciones es una labor compleja y costosa en términos del esfuerzo que debe ser invertido (3). Cada desarrollador utiliza la técnica de corrección de defectos que más se ajuste a la experiencia que tenga en esta práctica. Algunas de las cuales se enumeran a continuación:

a) Ensayo y error: Esta técnica se basa en tomar una serie de variables o pará metros dentro de la aplicación y cambiar sus valores. Una vez los cambios son hechos se realizan pruebas sobre la aplicación. Si los cambios hechos no sirven, se vuelve a cambiar los valores y se realizan pruebas de nuevo. Esta técnica necesita ser efectuada por desarrolladores con experiencia, de

lo contrario puede resultar costosa, ya que la probabilidad de encontrar una solución por azar resulta muy baja.

b) Depuración manual: esta técnica consiste en hacer un seguimiento detallado del código fuente cuando se encuentra el software en ejecución. Específicamente se revisa instrucción por instrucción(o línea por línea) buscando el lugar exacto en donde se produce la falla. También se utiliza para obtener mayor información del contexto de ejecución de la aplicación, así como para validar métodos o servicios. Esta técnica es relegada en el caso de las aplicaciones distribuidas, debido a la alta dificultad de realizar seguimiento al código ejecutado en una máquina remota.

c) Búsqueda de información externa: se utiliza generalmente cuando las anteriores no son suficientes para solucionar el error. Dentro de esta técnica se realizan consultas en motores de búsqueda de Internet, en foros especializados o páginas de soporte, utilizando palabras claves o mensajes explícitos de error mostrados por la aplicación (excepciones). También se consulta otros desarrolladores más cercanos, como amigos o compañeros de trabajo, esperando que sirvan de ayuda para plantear una solución. Es decir, si el desarrollador no encuentra los datos de la solución específica, la información recopilada sirve de ayuda para encontrar una solución.

En el caso específico del desarrollo de software en la UCI, la práctica más común en muchos escenarios tiende a ser la búsqueda de información externa, dada la falta de experiencia de los estudiantes cuando se inician en las tareas productivas.

Además de la complejidad de las aplicaciones y los factores humanos, es frecuente encontrar que hay defectos similares que se repiten, y que al momento de corregirlos se evidencia que se olvidó la solución utilizada. Esto es llamado *pérdida del conocimiento asociado con la solución* y constituye un elemento clave de la **situación problémica** expuesta a continuación:

Actualmente en el Centro de Informática Médica existen múltiples sistemas en desarrollo, de los cuales varios se encuentran en las fases finales. Dada que la metodología más utilizada en los diferentes departamentos que conforman este centro es el Proceso Unificado de Rational[1] (RUP), los integrantes de los equipos de desarrollo e involucrados en las tareas de implementación, pruebas, despliegue y

[1]Rational Unified Process.

soporte de las aplicaciones se enfrentan a distintos tipos de problemas o errores que pueden traducirse en una considerable pérdida de tiempo en la mayoría de las ocasiones en que ocurren. Dichos errores pueden ser de desconocimiento a la hora de configurar una herramienta, relacionados con el entorno de desarrollo o con el sistema operativo; o errores en la implementación, tanto propia como de componentes de terceros que estén siendo utilizados.

Independientemente del problema que sea, se puede considerar que para darle solución es necesario encontrar alguien que se haya enfrentado a una situación similar y ayude a determinar el modo de resolverlo; o bien se debe encontrar una solución en Internet. La búsqueda de estas soluciones puede estar sujeta a fallas de conectividad o disponibilidad en el servicio de Internet, o simplemente a la inexistencia del personal capacitado para mitigar el error o con experiencia previa en una situación similar; por lo que surge el siguiente **Problema a resolver:** ¿Cómo viabilizar la reutilización de las soluciones aportadas, a errores recurrentes en los flujos de trabajo Implementación, Pruebas y Despliegue del proceso de desarrollo de software, en nuevos proyectos o desarrollos?

Se define como **Objeto de Estudio:** La gestión del conocimiento en equipos de desarrollo de software. Dentro del cual se enmarca como **Campo de Acción:** La gestión del conocimiento durante los flujos de trabajo Implementación, Pruebas y Despliegue del desarrollo de software en el Centro de Informática Médica.

Objetivo General: Diseñar una aplicación web que posibilite llevar trazas de los problemas o errores encontrados durante los flujos de trabajo Implementación, Pruebas y Despliegue del desarrollo de un software y gestionar las posibles soluciones a estos; independientemente del lenguaje de programación o entorno de desarrollo.

Para lograr el correcto cumplimiento del objetivo general se proponen las siguientes tareas investigativas:

1. Realizar un análisis crítico y valorativo de los sistemas informáticos existentes para la gestión de soluciones a problemas detectados durante el desarrollo de software, a nivel nacional e internacional, estableciendo similitudes con la investigación en curso.
2. Realizar un análisis crítico y valorativo de las tendencias, técnicas, tecnologías, plataformas, librerías, metodologías y herramientas usadas en la actualidad.

3. Analizar los procesos de negocio asociados a la gestión de la información relacionada con la gestión del conocimiento generado durante el desarrollo de software, logrando un modelo único como guía para la implementación del sistema.
4. Generar los artefactos correspondientes a los Flujos de Trabajo propuestos por la Metodología seleccionada, sirviendo de base a los desarrolladores.

El presente trabajo cuenta con 3 capítulos estructurados de la siguiente manera:

Capítulo 1: **Fundamentación teórica:** En el capítulo se hará mención de forma general a aspectos relacionados con el desarrollo de aplicaciones de software. Se hace una valoración de las aplicaciones de software similares existentes en el mundo. Además se analiza la metodología a utilizar para el análisis y diseño del sistema, se proponen las tecnologías y herramientas para el desarrollo de la aplicación.

Capítulo 2: Características del sistema: Para guiar el desarrollo del sistema se describe la herramienta propuesta, se detallan los conceptos más importantes logrando así el modelo de dominio, además de enumerar los requisitos funcionales y no funcionales, agrupándolos por Casos de Uso, así como las descripciones de los procesos que serán objeto a automatizar, hasta conformar el diagrama de casos de uso del sistema.

Capítulo 3: Análisis y Diseño del Sistema: En este capítulo se definen las clases de diseño del software, así como los diagramas de interacción correspondientes, para que se cumplan los requerimientos funcionales y no funcionales además del modelo de datos.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el capítulo se hará mención de forma general aspectos relacionados con el desarrollo de aplicaciones de software. Se hace una valoración del software similar ya existente en el mundo. Además se analiza la metodología a utilizar para el análisis y diseño del sistema, se proponen las tecnologías y herramientas para el desarrollo de la aplicación.

1.1. El apoyo al desarrollo de software

Son muchas las empresas en el mundo que se dedican a desarrollar herramientas para mejorar la experiencia de los desarrolladores a la hora de materializar un proyecto. Un buen entorno de desarrollo y herramientas no definen la calidad del producto final; pero pueden influir de modo determinante sobre el tiempo de desarrollo, pues el programador puede enfocarse más fácilmente en la tarea a cumplir.

No obstante, eso no es todo; es de vital importancia contar con el conocimiento, no solo el propio; sino también de las experiencias de otros desarrolladores que anteriormente hayan dado solución a errores comunes que a menudo plagan el trabajo.

1.1.1. En el Mundo

A nivel mundial es posible encontrar múltiples sitios y comunidades online destinadas a brindar ayuda a los desarrolladores de software y usuarios de las TIC en general a resolver los problemas a los que se enfrentan en el quehacer diario. La mayoría de los sitios existentes en la actualidad tienen un enfoque de foro, los que pueden estar enfocados a una temática en específico o abarcar múltiples temas de interés. Dada la importancia del desarrollo de software para la sociedad moderna, gran cantidad de los sitios mencionados se especializa en brindar ayuda a los desarrolladores, brindando la posibilidad de publicar preguntas, dudas o simplemente los errores emitidos por los entornos de desarrollo o aplicaciones utilizadas.

A continuación se presentan algunos de los sitios, o más bien comunidades que a lo largo de tiempo han alcanzado un notable prestigio en la red de redes; dado fundamentalmente por la cantidad de usuarios y la calidad de los materiales publicados.

- **Stackoverflow**: hasta cierto punto tiene un enfoque parecido a un foro, que abarca, disímiles temáticas, entre las que se incluyen el uso e instalación de diferentes aplicaciones y sistemas operativos, aunque la mayor afluencia de preguntas recae en los temas relacionados más directamente con el desarrollo de software con distintas herramientas, lenguajes, plataformas y tecnologías.

  Este sitio en particular cuenta con un sistema de preguntas y respuestas, donde el usuario coloca una pregunta utilizando etiquetas para su organización. Cada pregunta posee tres valores asociados: votos, respuestas y vistas. Una pregunta puede ser vista y leída (incluso recibir comentarios u opiniones) múltiples veces sin conseguir que alguien la responda adecuadamente. Una vez que el solicitante encuentra una solución válida entre los comentarios realizados puede marcarla como respuesta; y los usuarios más avanzados tienen la posibilidad de votar por las diferentes respuestas con el objetivo de facilitar la selección de la solución a utilizar (4).

  El problema fundamental de este sitio es la falta de organización derivada del sistema de etiquetación, donde cada usuario decide que etiquetas poner, en lugar de seleccionar de una lista predeterminada. Adicionalmente los cometarios se encuentran mezclados con las respuestas válidas, lo cual dificulta en ocasiones la lectura y comprensión de las respuestas.

- **CodeProject**: es una comunidad online de varios años de existencia donde tanto profesionales de la industria como aficionados a la programación publican sus soluciones a errores a los que se han enfrentado. Además publican artículos sobre algún proyecto personal o ejemplos de cómo hacer algo que consideren útil o interesante.

  Fue creado con el objetivo de brindar a los desarrolladores los recursos necesarios para su día a día de programación y ayudarles a mantenerse al día con las últimas tecnologías. Actualmente la comunidad de usuarios de este portal cuenta con más de 7,5 millones de usuarios, los cuales comparten con todo el mundo sus conocimientos en diversas áreas del desarrollo de software: C++, MFC, C#, ASP, ASP.NET y la plataforma .NET en general (5).

  Para este propósito existen diferentes secciones y servicios como foros, entrevistas, noticias, incluida una sección de preguntas y respuestas; actualmente en el sitio existen más de 30000 artículos publicados, en su mayoría de notable calidad y gran nivel técnico.

- **Experts Exchange**: es un sitio de ayuda en línea sobre temas de las tecnologías de la información y las comunicaciones que reúne expertos de todo el mundo; entre los que se encuentran consultores, profesionales de Microsoft y otros. Voluntarios entregan su tiempo para dar respuesta a los usuarios en un sistema de preguntas y respuestas muy similar a un foro; donde los miembros que dan respuesta a las preguntas obtienen puntos que le dan acceso a determinadas certificaciones (6) (7).

  Este sistema ha dado lugar a que en ocasiones los miembros aporten soluciones de media o baja calidad con el objetivo de obtener puntos, pues para mantener acceso a las funcionalidades más avanzadas del sitio es obligatorio obtener un determinado número de puntos mensualmente o efectuar un pago para ello. Desde el año 2007 se hizo obligatorio el pago para poder acceder a las soluciones.

Sin duda los sitios antes mencionados constituyen algunos de los puntos de encuentro fundamentales para los desarrolladores que buscan información para dar solución a las problemáticas a las que a diario se enfrentan. No obstante, es muy fácil encontrar navegando en Internet un sinnúmero de sitios dedicados a la capacitación sobre cómo utilizar una herramienta o tecnología determinada.

Aun cuando el número de ventajas que puede aportar el uso de esto sitios es notable, también hay una serie de limitaciones, como: la falta de información referente a pruebas, despliegue y soporte; pues se enfatiza más en la implementación; no se protege la confidencialidad de la información, por lo que no es posible hacer preguntas o publicar datos respecto a proyectos que contengan información sensible; además la mayoría de los proyectos de la universidad poseen características propias acorde a las necesidades de un sector u empresa específico; de modo que sólo pueden conocer en detalle el sistema aquellos que hayan estado involucrados en su desarrollo; motivo por el cual no se recomienda la utilización de estos sitios externos a menos que se trate de buscar soluciones a problemas de implementación que no incluyan información sensible.

Si bien los programadores hacen un amplio uso de la red de redes para buscar las respuestas que requieran, éstas no siempre son acertadas o han sido adaptadas a un entorno específico de trabajo (y no aportan esta información claramente) siendo inutilizables para muchos que terminan perdiendo el tiempo probando las soluciones encontradas.

De igual modo debe tenerse presente que en ocasiones los sistemas de seguimiento de errores (bug tracking) son utilizados incorrectamente con este fin, teniendo resultados muy pobres en consecuencia; pues estos sistemas no cuentan con buscadores especializados ni formas de organización que faciliten la búsqueda de información.

1.1.2. En Cuba

Hasta el momento no se han encontrado referencias relativas a algún sistema de este tipo; aunque no se descarta la posibilidad de la existencia de alguno; sobre todo en las universidades del país, especialmente donde se imparten las carreras de Ingeniería Informática o Licenciatura en Ciencias de la Computación u algún otro centro de estudios o empresas dedicadas al desarrollo de software.

1.1.3. En la Universidad de las Ciencias Informáticas

Se tiene conocimiento de la existencia de un banco de problemas en la Facultad #3; pero éste está destinado a proveer temas de tesis para los estudiantes de 5to año, no soluciones como es el caso en el que nos enmarcamos.

Hasta el momento no hay conocimiento de sistemas o soluciones enfocadas a brindar apoyo a los desarrolladores facilitando la reutilización de las experiencias previas para dar soluciones a errores o problemas durante el desarrollo.

1.2. Tecnologías, lenguajes y metodologías

1.2.1. UML 2.0 (Unified Modeling Language)

Por sus siglas en inglés, (Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. UML es un "lenguaje" para especificar y no un método o un proceso.

1.2.2. Metodología de desarrollo: RUP (Rational Unified Process)

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes. La versión se conoció en sus inicios como Proceso Unificado de Rational 5.0; de ahí las siglas con las que se identifica a este proceso de desarrollo. Es una de las metodologías más generales y más usadas de las que existen en la actualidad, pues está pensada para adaptarse a cualquier proyecto. Constituye además, una propuesta de proceso para el desarrollo de software orientado a objeto, utilizando UML, para describir todo el proceso, basándose en componentes. Este lenguaje es estándar, con él se puede visualizar, especificar, construir y documentar los artefactos de un sistema (8)(9)(10).

Las principales características de la metodología Proceso Unificado de Rational, que la hacen elegible para emprender grandes y complejos proyectos de desarrollo de software manteniendo altos índices de éxito en las empresas son las siguientes:

- **Dirigido por casos de uso**: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).
- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- **Iterativo e Incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Por ejemplo, una iteración de elaboración centra su atención en el análisis y diseño, aunque refina los requerimientos y obtiene un producto con un determinado nivel, pero que irá creciendo incrementalmente en cada iteración.

Existen nueve Flujos de trabajo de los cuales seis son ingenieriles y tres de apoyo, pero específicamente nos centraremos en:

- **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

1.2.3. Enterprise Architect 7.5 (EA)

Enterprise Architect 7.5 es una herramienta de diseño y análisis UML, cubriendo el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. EA es una herramienta multi-usuario, basada en Windows (aunque capaz de funcionar en el Sistema Operativo Linux), diseñada para ayudar a construir software robusto y fácil de mantener.

El Lenguaje Unificado de Modelado provee beneficios significativos para ayudar a construir modelos de sistemas de software rigurosos y donde es posible mantener la trazabilidad de manera consistente. Enterprise Architect soporta este proceso en un ambiente fácil de usar y rápido. Ofrece salida de documentación flexible y de alta calidad. Enterprise Architect provee trazabilidad completa desde el análisis de requerimientos hasta los artefactos de análisis y diseño, a través de la implementación y el despliegue.

Los equipos de Administradores de Proyectos y Calidad están equipados con la información que ellos necesitan para ayudarles a entregar proyectos en tiempo. EA soporta transformaciones de Arquitectura avanzada dirigida por Modelos (MDA) usando plantillas de transformaciones de desarrollo, fáciles de usar. Con transformaciones incorporadas para DDL, C#, Java, EJB y XSD, se puede rápidamente desarrollar

soluciones complejas desde los simples modelos independientes de plataforma. Entre sus características principales están:

- Intuitivo y de fácil uso.
- Extensiones personalizadas para modelado de procesos.
- Ingeniería de Código Directa e Inversa.
- Bajo costo de Licencias.
- Facilidad de Importación/Exportación XMl.

1.2.4. Microsoft Office Project

Microsoft Project (o MSP) es un Software de administración de proyectos desarrollado y vendido por Microsoft el cual esta creado para asistir a los administradores de proyectos. Aunque este software ha sido etiquetado como miembro de la familia Microsoft Office hasta el momento no ha sido incluido en ninguna de las ediciones de Office. Está disponible en dos versiones, Standard y Professional. La aplicación crea calendarización de rutas críticas, además de cadenas críticas y metodología de eventos en cadena disponibles como "add-ons[2]" de terceros. Los calendarios pueden ser "resource leveled", y las gráficas visualizadas en una Gráfica de Gantt. Adicionalmente, Project puede reconocer diferentes clases de usuarios, los cuales pueden contar con distintos niveles de acceso a proyectos, vistas y otros datos.

1.2.5. Microsoft Visual Studio 2010 SP1

Visual Studio es un completo conjunto de herramientas para la creación tanto de aplicaciones de escritorio como de aplicaciones web. Aparte de generar aplicaciones de escritorio de alto rendimiento, se pueden utilizar las eficaces herramientas de desarrollo basado en componentes y otras tecnologías de Visual Studio para simplificar el diseño, desarrollo e implementación en equipo de soluciones empresariales (11).

Es un entorno de desarrollo integrado (IDE), que habilita el uso compartido de herramientas y hace más sencilla la creación de soluciones en varios lenguajes como por ejemplo: Visual Basic, Visual C#, Visual C++

[2] Extensiones que añaden funcionalidades adicionales.

y Visual F#. Además incluye soporte para lenguajes específicos para las aplicaciones web como HTML y JavaScript y un buen editor de hojas de estilo CSS.

1.2.6. Lenguaje Visual C# 4.0

C# es un lenguaje de programación diseñado por Microsoft en 2001 como parte de su plataforma .NET. Combina el lenguaje de bajo nivel de C y la velocidad de la programación de alto nivel de Visual Basic.

C# es un lenguaje orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, es simple pero eficaz y está diseñado para escribir aplicaciones empresariales. Presenta considerables mejoras e innovaciones en áreas como seguridad de tipos, control de versiones, eventos y recolección de elementos no utilizados (liberación de memoria).

Ventajas:

- Compila a código intermedio independiente del lenguaje en que haya sido escrita la aplicación e independiente de la máquina donde vaya a ejecutarse.
- Realiza la recolección automática de basura.
- Posee capacidades de reflexión.
- Es flexible en cuanto al orden de definición de las clases y las funciones.
- Soporta definición de clases dentro de otras.
- Todos los valores son inicializados antes de ser usados (automáticamente por defecto, o manualmente desde constructores estáticos).

1.2.7. Plataforma .NET 4.0 (.NET Framework)

La plataforma .NET es un conjunto de tecnologías diseñadas para transformar Internet en una plataforma informática distribuida a escala completa. Proporciona nuevas formas de desarrollar aplicaciones a partir de colecciones de Servicios Web. La plataforma .NET soporta totalmente la infraestructura existente de Internet, incluyendo Hypertext Transfer Protocol (HTTP), Extensible MarkupLanguage (XML) y Simple Object Access Protocol (SOAP).

La plataforma .NET proporciona:

- Un modelo de programación coherente e independiente del lenguaje para todas las capas o niveles de una aplicación.
- Una interoperabilidad transparente entre tecnologías.
- Una fácil migración desde tecnologías existentes.
- Un completo soporte de tecnologías de Internet independientes de la plataforma y basadas en estándares, incluyendo HTTP, XML y SOAP.

Está diseñado para utilizar los servicios web con XML como mecanismo principal de comunicación entre aplicaciones. Posee avanzadas funciones en tiempo de ejecución lo que permite que cualquier aplicación pueda ser convertida en servicios web XML. Permite escribir programas en cualquiera de los lenguajes soportados por la plataforma, incluso utilizar simultáneamente varios lenguajes en un mismo programa. Entre los lenguajes que se han adherido a esta familia se encuentra el ya mencionado C#. Agrupa además al Visual Basic, C++, Pascal, Ruby y Python entre otros, conformando un grupo muy nutrido y en crecimiento.

Cuenta un componente de seguridad capaz de monitorear las acciones que pueden ser sensibles sobre el sistema operativo, controlando quién escribe y ejecuta el código y con qué propósitos lo hace.

1.2.8. ASP.NET MVC 3.0

Luego de varios años, cambios y avances en las tecnologías. Apareció en enero de 2002 con la versión 1.0 del Framework .NET, y es la tecnología sucesora de la tecnología Active Server Pages (ASP). Este está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML.

Esta se ha creado bajo varios principios como son facilidad de desarrollo el cual da la posibilidad de elegir el lenguaje de programación con el que se desee trabajar, por defecto lleva integrado C#, VB.Net y J#, pero se pueden usar otros lenguajes. Tiene gran independencia de la herramienta de desarrollo y cuenta con una biblioteca de clases que lleva incorporada.

El alto rendimiento y escalabilidad, una mejorada fiabilidad con la cual ASP.NET es capaz de detectar pérdidas de memoria, problemas con bloques y protección ante caídas. Entre otras cosas, es capaz de detectar aplicaciones web que pierden memoria, arrancando otro proceso limpio con una nueva instancia de la aplicación para cerrar la que pierde memoria liberando así la memoria perdida.

ASP.NET MVC constituye una plataforma de desarrollo web de Microsoft que combina la efectividad y sencillez de la arquitectura modelo-vista-controlador[3], las más modernas ideas y técnicas del desarrollo ágil y las mejores partes de la existente plataforma ASP.NET. Es una alternativa a la tradicional forma de desarrollar aplicaciones web con ASP.NET que provee considerables ventajas para proyectos de desarrollo de aplicaciones web (12).

En específico la versión 3.0 de ASP.NET MVC incorpora un potente motor de plantillas basado en los más utilizados en la actualidad para el desarrollo de aplicaciones web como Ruby on Rails, Django y Symfony. Adicionalmente brinda facilidades y soporte nativo para las librerías javascript jQuery, las cuales ofrecen grandes facilidades para el desarrollo de interfaces enriquecidas con AJAX[4] sin elevar en gran medida la complejidad.

1.2.9. Lucene.Net

Lucene.Net es una implementación del motor de búsquedas para Java Lucene; completamente escrita en C# para la plataforma .NET. Constituye una librería C# que encapsula un motor de búsquedas de alto rendimiento que con el objetivo de proveer potencia y usabilidad máximas utiliza en su API múltiples funcionalidades especiales de la plataforma(13).

Entre las características más destacables resalta la escalabilidad y alto rendimiento:

- Indexación de más de 95GB por hora en un hardware moderno.
- Bajos requerimientos de memoria RAM (HEAP de solamente 1MB).
- Indexado incremental tan rápido como el indexado secuencial.
- Tamaño del índice entre un 20% y 30% de la capacidad de todo el texto indexado.

[3] Model-View-Controller (MVC)
[4] Asyncronous Javascript and XML.

Adicionalmente el motor cuenta con una serie de algoritmos precisos y eficientes:

- Búsqueda con ranking (los mejores resultados primero).
- Varios tipos de consultas (por frases, comodines, por proximidad, rangos y otras).
- Búsquedas por campos (ejemplo: título, autor, contenido).
- Búsqueda por rangos de fechas.
- Ordenamiento por cualquier campo.
- Búsqueda en múltiples índices con resultados mixtos.
- Búsqueda y actualización simultáneas.

1.2.10. PostgreSQL 8.4

Es multiplataforma y además soporta procedimientos almacenados y triggers, los cuales pueden ser escritos usando lenguajes como PL/PgSQL, C++, Java PL/Java web, PL/Python y plPHP. El RSGBD Orientado a Objetos PostgreSQL, comenzó su desarrollo sobre los años 80. Es un sistema de gestión de bases de datos objeto relacional, basado en el proyecto POSTGRES.

El gestor se encuentra bajo una licencia BSD y es dirigido por una comunidad mundial de desarrolladores conocida como Grupo Mundial de Desarrollo de PostgreSQL, y está basado en contribuciones de proveedores comerciales, empresas a portantes y programadores de código abierto mayormente. (PGDG, por sus siglas en inglés). Entre sus principales características se encuentra su destreza para manejar altos niveles de concurrencia mediante un sistema llamado Acceso Concurrente Multiversión (MVCC, por sus siglas en inglés). Posee una amplia variedad de tipos nativos y permite al usuario crear sus propios tipos de datos.

1.3. Resultados esperados

Se espera el análisis y diseño de una aplicación para la gestión de soluciones a problemas detectados durante la fase de desarrollo de un software.

En dicha aplicación se podrán encontrar soluciones a problemas que surgen durante las diferentes fase de un proyecto o la instalación de una herramienta o aplicación, así como cualquier otro error que pudiese surgir, donde usted podrá votar por las soluciones, siendo la más correcta la que mayor número de votos obtenga; y en caso de que la respuesta no se encontrara, es posible dejar registrado el error para que otros

usuarios con conocimiento de una solución puedan publicar una respuesta y así contribuir al desarrollo del resto de los usuarios.

La investigación de los sitios que dan soluciones a problemas detectados en la rama de la informática, la permitió establecer las herramientas, metodología y lenguaje de programación para el modelado de la aplicación hasta lograr su análisis y diseño y más adelante su implementación. Además, la información recopilada sobre los antecedentes existentes en el mundo constituye la base para la toma de decisiones respecto a la manera de estructurar la aplicación.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

Para guiar el desarrollo del sistema se describe la herramienta propuesta, se detallan los conceptos más importantes definidos anteriormente, logrando así el modelo de dominio, además de enumerar los requisitos funcionales y no funcionales, agrupándolos por Casos de Uso, así como las descripciones de los procesos que serán objeto automatizar, hasta conformar el diagrama de casos de uso del sistema.

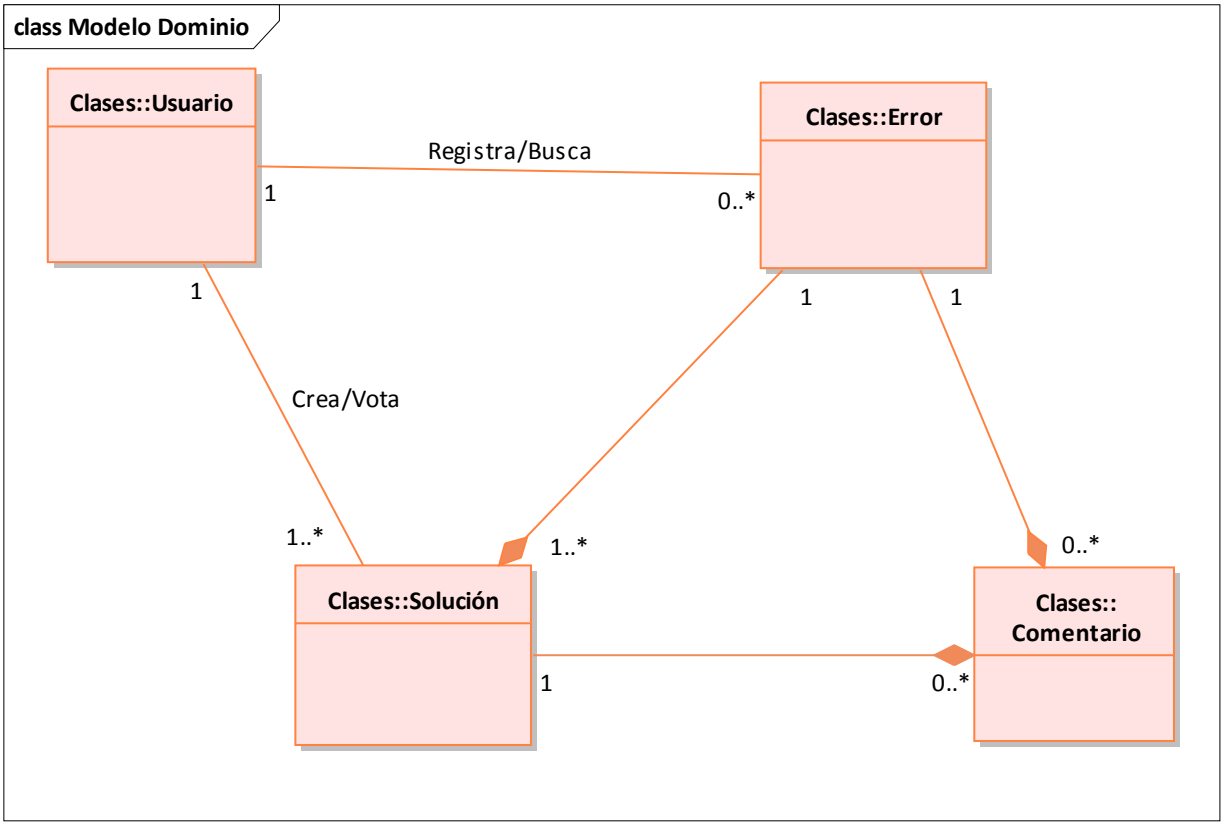
2.1. Breve descripción del sistema.

Se quiere realizar una aplicación donde los usuarios accedan para la búsqueda de la solución a un error que se le presentó en su trabajo ya sea durante la fase de implementación de un software, o durante la instalación de una herramienta o sistema. Si no se encuentra una solución existente, entonces se registrará el error para que algún otro usuario al acceder a la aplicación lo vea y si conoce la solución la registre, se podrán hacer comentarios tanto de los errores como de las soluciones para lograr tener mayor calidad en la solución propuesta, además se votará por la efectividad de dicha respuesta.

2.2. Modelo de Dominio.

Como parte del proceso de desarrollo de un software se realiza el modelado del negocio, el cual facilita comprender los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar nuestra aplicación; definiendo los procesos que serán objeto de automatización con la realización del sistema. Pero cuando el negocio (los procesos que abarca) no está bien definido o muy claro, se propone realizar un modelo de dominio, el cual es una representación de los conceptos clave de un área o problema. El modelo de dominio generalmente se utiliza para representar aspectos del mundo real o físico.

Teniendo en cuenta que para este sistema los procesos de negocio no están lo suficientemente bien definidos, se describe el negocio a través del modelo de dominio.



2.2.1. Conceptos Fundamentales.

Error:

Problemas o errores que se nos presentan, los cuales pueden ser errores de desconocimiento a la hora de configurar una herramienta, relacionados con el entorno de desarrollo o con el sistema operativo; o errores en la implementación. Estos impiden el avance del trabajo que se está realizando.

Solución:

Respuesta que se da al surgir un error o problema.

Comentario:

Opinión que se da para argumentar la información, ya sea de un error o problema o de una solución.

2.3. Especificación de los requisitos del software.

2.3.1. Requisitos Funcionales

RF01. Registrar un error o problema.
RF02. Registrar una solución a un error o problema.
RF03. Agregar comentarios sobre un error o problema.
RF04. Agregar comentarios sobre las soluciones propuestas.
RF05. Realizar búsquedas de errores o problemas.
RF06. Realizar búsquedas avanzadas de errores o problemas mediante el filtrado de la información.
RF07. Realizar búsquedas o filtrado de las soluciones.
RF08. Registrar una solución como válida
RF09. Realizar votos por la calidad de una solución.
RF10. Controlar el conteo de las veces que se ha visto un error.
RF11. Controlar el conteo de las veces que se ha visto una solución.
RF12. Mostrar los detalles de un error o problema.
RF13. Mostrar la solución o soluciones de un error o problema.
RF14. Mostrar los comentarios asociados a un error o problema.
RF15. Mostrar los comentarios asociados a una solución de un error o problema.
RF16. Listar los errores o problemas registrados.
RF17. Listar las soluciones asociadas a un error o problema.
RF18. Autenticar Usuario.
RF19. Editar un error o problema.
RF20. Editar una solución.
RF21. Editar una solución.

2.3.2. Requisitos No Funcionales

- Usabilidad
    - RNU01. La Aplicación debe ser fácil de usar.

- RNU02. Permitir que varios usuarios estén conectados a la vez.

- Eficiencia
  - RNE01. Las repuestas a las búsquedas deben ser rápidas (por debajo de 3 segundos).
- Seguridad
  - RNS01. Autenticarse para Acceder a la Aplicación.
- Interfaz de usuario
  - RNIU01. La aplicación debe contar con una interfaz segura y amigable.

2.4. Definición de los casos de uso del sistema.

2.4.1. Actores del sistema.

Usuario	Es la persona que accede a la aplicación, para registrar los errores o problemas, tratando de encontrar una solución a la cual puede comentar y votar por la solución después de probada.
Administrador	Es la persona que interactúa con la aplicación, para realizar la gestión de los usuarios y tareas de administración.

2.4.2. Casos de Uso.

CU1	Gestionar un error o problema.
Actor	Usuario
Descripción:	El usuario accede a la aplicación, para agregar o modificar un error o problema, y luego se guarda la acción realizada.

Referencia	RF.01; RF.10; RF.12; RF.14; RF.16; RF.20.

CU2	Gestionar Solución
Actor	Usuario
Descripción:	El usuario accede a la aplicación, para agregar o modificar una solución, y luego se guarda la acción realizada.
Referencia	RF.02; RF.07; RF.09; RF.11; RF.13; RF.15; RF.17;RF.21.

CU3	Comentar un error o problema.
Actor	Usuario
Descripción:	El usuario accede a la aplicación, para comentar un error o problema, es decir para brindar más información acerca de un error o problema.
Referencia	RF.03.

CU4	Comentar una solución.
Actor	Usuario
Descripción:	El usuario accede a la aplicación, para comentar una solución, es decir para

	brindar más información acerca de una solución.
Referencia	RF.04.

CU5	Registrar una solución como válida.
Actor	Usuario
Descripción:	El usuario accede a la aplicación, prueba las soluciones que encuentra para un error o problema, y una vez probada las que funcionen la registra como válida.
Referencia	RF.08.

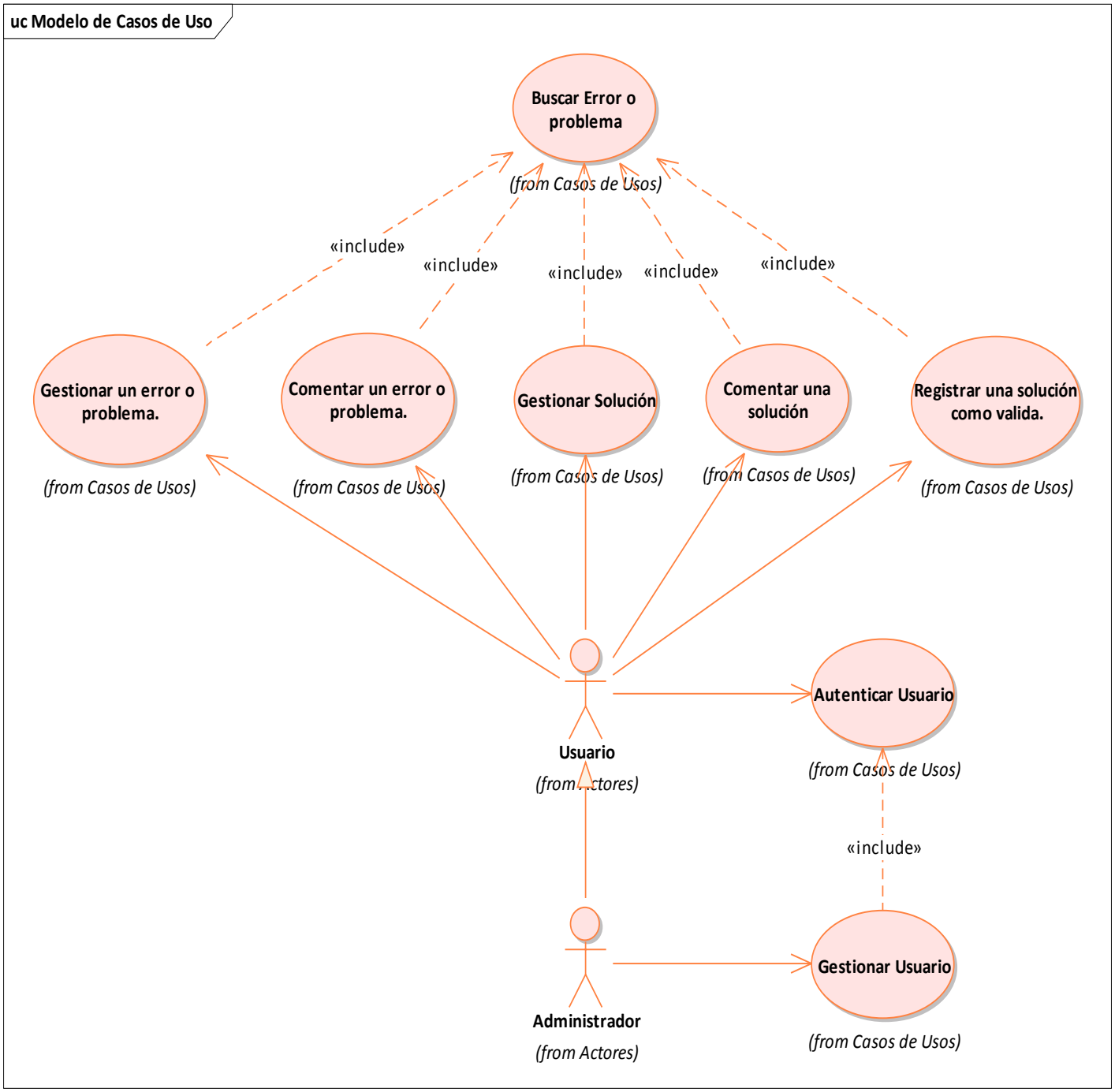
CU6	Buscar Error o problema
Actor	Usuario
Descripción:	El usuario accede a la aplicación, para buscar un error o problema, además se realizan búsquedas avanzadas mediante el filtrado de información para comentar una solución, es decir para brindar más información acerca de una solución.
Referencia	RF.05; RF.06.

CU7	Autenticar Usuario
Actor	Usuario

Descripción:	El usuario accede a la aplicación, puede o no autenticarse para realizar las funciones permitidas para él.
Referencia	RF.18.

CU8	Gestionar Usuario
Actor	Administrador
Descripción:	El administrador accede a la aplicación es obligatorio autenticarse, para luego poder agregar, eliminar, modificar la información de los usuarios.
Referencia	RF.19.

2.4.3. Modelo de Casos de Uso



2.4.4. Descripción extendida de los Casos de Uso

CU # 1. Gestionar un error o problema.	
Objetivo	*El Usuario gestiona toda la información de un error o problema, ya sea para agregarlo por no haber encontrado una solución, como para modificarlo.*
Actores	*Usuario: (Inicia) Adiciona, modifica,(Administrador) elimina, ve y valida la gestión de la información de un error o problema*
Resumen	*El usuario accede a la aplicación, para agregar, modificar, (Administrador) elimina un error o problema, y luego se guarda la acción realizada.*
Complejidad	*Alta.*
Prioridad	*Crítico.*
Referencias	*RF.01; RF.10; RF.12; RF.14; RF.16; RF.20*
Precondiciones	*Solo el administrador podrá modificar o eliminar un error o problema, y debe autenticarse antes.*
Postcondiciones	
Flujo de eventos	

1. *El caso de uso se inicia cuando el **Usuario** accede a la aplicación para registrar un error o problema.*
2. *La aplicación muestra las acciones que el **Usuario** puede realizar.*
3. *El **Usuario** procede a realizar las acciones que le son permitidas.*

a. *Adicionar. Ver evento1.*
b. *Modificar. Ver evento 2.*
c. *Eliminar. Ver evento 3*

4. *La aplicación guarda la nueva información.*
5. *Termina el caso de uso cuando la aplicación muestra mensaje.*

Evento 1 Adicionar

1. *El **Usuario** selecciona adicionar un nuevo error o problema.*
2. *La aplicación solicita los datos del nuevo error o problema.*
3. *El **Usuario** introduce la información referente al error o problema.*
4. *El **Usuario** adiciona el nuevo error o problema.*
5. *Termina el caso de uso cuando la aplicación agrega el nuevo error o problema a la lista.*

Evento 2 Modificar.

1. *El **Usuario** selecciona modificar un error o problema.*
2. *La aplicación solicita los datos a modificar.*
3. *El **Usuario** introduce los datos en la aplicación.*
4. *La aplicación modifica el error o problema.*
5. *La aplicación registra que usuario realizó la modificación del error o problema.*

Evento 3 Eliminar

1. *El **Administrador** selecciona eliminar un error o problema.*
2. *La aplicación solicita confirmación para eliminar.*
3. *Termina el caso de uso cuando se muestra el mensaje de que ya se eliminó el error o problema.*

Flujos alternos

a) *El **Usuario** selecciona modificar un error o problema determinado: Ejecuta los pasos del 3 al 6 y la aplicación actualiza la lista de errores o problema.* *b) El **Usuario** selecciona eliminar un error o problema determinado, la aplicación elimina el error o problema seleccionado.*		
Relaciones	**CU Incluidos**	*Buscar error o problema. Ver CU: Buscar error o problema*

CU # 2. Gestionar Solución	
Objetivo	*El usuario accede a la aplicación, para agregar o modificar una solución, y luego se guarda la acción realizada.*
Actores	*Usuario: (Inicia) Adiciona, modifica, (Administrador elimina), ve y valida la gestión de la información de una solución.*
Resumen	*El usuario accede a la aplicación, para agregar, modificar, (Administrador) elimina una solución, y luego se guarda la acción realizada.*
Complejidad	*Alta.*
Prioridad	*Crítico.*
Referencias	*RF.02; RF.07; RF.09; RF.11; RF.13; RF.15; RF.17;RF.21.*
Precondiciones	*Para modificar una solución el usuario debe autenticarse antes, y el administrador debe autenticarse antes de eliminar una solución.*
Flujo de eventos	

1. *El caso de uso se inicia cuando el **Usuario** accede a la aplicación para registrar una solución o cuando accede a los detalles de una solución.*
2. *La aplicación muestra las acciones que el **Usuario** puede realizar.*
3. *El **Usuario** procede realizar las acciones que le son permitidas.*
    a. *Adicionar. Ver evento1.*
    b. *Modificar. Ver evento 2.*
    c. *Eliminar. Ver evento 3*
4. *La aplicación guarda la nueva información.*
5. *Termina el caso de uso cuando la aplicación muestra mensaje.*

Evento 1 Adicionar

6. *El **Usuario** selecciona adicionar una solución.*
1. *La aplicación solicita los datos de la nueva solución.*
2. *El **Usuario** introduce la información referente a la solución.*
3. *El **Usuario** adiciona la nueva solución.*
4. *Termina el caso de uso cuando la aplicación agrega la solución a la lista.*

Evento 2 Modificar.

1. *El **Usuario** selecciona modificar una solución.*
2. *La aplicación solicita los datos a modificar.*
3. *El **Usuario** introduce los datos en la aplicación.*
4. *La aplicación modifica la solución.*
5. *La aplicación registra que usuario realizó la modificación de la solución.*

Evento 3 Eliminar

1. *El **Administrador** selecciona eliminar una solución.*

2. *La aplicación solicita confirmación para eliminar.*
3. *Termina el caso de uso cuando se muestra el mensaje de que ya se eliminó la solución.*

Flujos alternos

a) *El **Usuario** selecciona modificar una solución determinada: Ejecuta los pasos del 3 al 6 y la aplicación actualiza la lista de soluciones.*

*b) El **Administrador** selecciona eliminar una solución determinada, la aplicación elimina la solución seleccionada.*

CU # 3. Buscar Error o problema	
Objetivo	*El usuario accede a la aplicación, para buscar un error o problema, además se realizan búsquedas avanzadas mediante el filtrado de información para comentar una solución, es decir para brindar más información acerca de una solución.*
Actores	*Usuario: (Inicia) cuando Busca Error o problema*
Resumen	*El usuario accede a la aplicación, para buscar error o problema*
Complejidad	*Alta.*
Prioridad	*Crítico.*
Referencias	*RF.05; RF.06.*
Precondiciones	
Postcondiciones	

Flujo de eventos
1. *El caso de uso se inicia cuando el **Usuario** accede a la aplicación para buscar un error o problema que le ha surgido.* 2. *La aplicación muestra el resultado de la búsqueda.* 3. *El **Usuario** verifica que encontró lo que buscaba y selecciona las opciones disponibles.* a. *Ver soluciones. Ver evento1.* b. *Modificar. Ver caso de uso gestionar error o problema evento 2.* c. *Eliminar. Ver caso de uso gestionar error o problema evento 3.* d. *Comentar un error. Ver caso de uso comentar error o problema.* 4. *Termina el caso de uso cuando la aplicación realiza los eventos seleccionados.*
Evento 1 Ver soluciones
1. *El **Usuario** Selecciona ver soluciones.* 2. *La aplicación muestra los detalles del error y las soluciones que tiene asociadas.* 3. *El **Usuario** puede seleccionar:* a. *Ver detalles de una solución. Ver evento 1.1.* b. *Modificar una solución. Ver caso de uso gestionar solución evento 2.* c. *Eliminar una solución. Ver caso de uso gestionar solución evento 3.* d. *Comentar una solución. Ver caso de uso comentar solución.*
Evento1.1 Ver detalles de una solución
1. *El **Usuario** selecciona la opción ver detalles de una solución.* 2. *La aplicación muestra los detalles de una solución.*

CU # 4. Comentar un error o problema.	
Objetivo	*El usuario accede a la aplicación, para comentar un error o problema, es decir*

	para brindar más información acerca de un error o problema.
Actores	*Usuario: (Inicia) cuando decide comentar un error o problema.*
Resumen	*El usuario accede a la aplicación, para comentar un error o problema.*
Complejidad	*Alta.*
Prioridad	*Crítico.*
Referencias	RF.03.
Precondiciones	
Postcondiciones	

Flujo de eventos

1. *El caso de uso se inicia cuando el **Usuario** accede a la aplicación para comentar un error o problema del que conoce alguna información.*
2. *La aplicación muestra donde ingresar los datos del comentario del error o problema.*
3. *El **Usuario** introduce la información del comentario.*
4. *Termina el caso de uso cuando la aplicación muestra mensaje que se ha guardado la información introducida.*

CU # 5. Comentar una solución.	
Objetivo	*El usuario accede a la aplicación, para comentar una solución, es decir para brindar más información acerca de una solución.*

Actores	*Usuario: (Inicia) cuando decide comentar una solución.*
Resumen	*El usuario accede a la aplicación, para comentar una solución.*
Complejidad	*Alta.*
Prioridad	*Crítico.*
Referencias	RF.04.
Precondiciones	
Postcondiciones	

Flujo de eventos

1. *El caso de uso se inicia cuando el **Usuario** accede a la aplicación para comentar una solución de la que conoce información.*
2. *La aplicación muestra donde ingresar los datos del comentario de la solución.*
3. *El **Usuario** introduce la información del comentario.*
4. *Termina el caso de uso cuando la aplicación muestra mensaje que se ha guardado la información introducida.*

CU # 6. *Registrar una solución como válida.*	
Objetivo	*El usuario accede a la aplicación, prueba las soluciones que encuentra para un error o problema, y una vez probada las que funcionen la registra como válida.*
Actores	*Usuario: (Inicia) cuando decide registrar una solución como válida.*

Resumen	*El usuario accede a la aplicación, para registrar una solución como válida.*
Complejidad	*Alta.*
Prioridad	*Crítico.*
Referencias	RF.08.
Precondiciones	
Postcondiciones	
Flujo de eventos	

1. *El caso de uso se inicia cuando el **Usuario** accede a la aplicación, prueba las soluciones que encuentra para un error o problema, y una vez probada las que funcionen procede a registrarla como válida.*
2. *Termina el caso de uso cuando la aplicación muestra mensaje que se ha guardado la información introducida.*

CU # 7. Autenticar Usuario	
Objetivo	*El usuario accede a la aplicación, puede o no autenticarse para realizar las funciones permitidas para él.*
Actores	*Usuario: (Inicia) cuando decide autenticarse para realizar algunas acciones.*
Resumen	*El usuario accede a la aplicación, para autenticarse y así poder realizar algunas acciones.*
Complejidad	*Alta.*

Prioridad	*Crítico.*
Referencias	RF.18.
Precondiciones	
Postcondiciones	
Flujo de eventos	
1. *El caso de uso se inicia cuando el **Usuario** accede a la interfaz de autenticación para realizar las funciones que requieren de la misma, introduciendo los datos.* 2. *La aplicación comprueba los datos* 3. *Termina el caso de uso cuando la aplicación muestra la interfaz de la acción que se procedía a realizar.*	

CU # 8. *Gestionar Usuario*	
Objetivo	*El **Administrador** accede a la aplicación es obligatorio autenticarse, para luego poder agregar, eliminar, modificar la información de los usuarios.*
Actores	***Administrador**: (Inicia) cuando decide autenticarse para realizar algunas acciones.*
Resumen	*El **Administrador** accede a la aplicación, para gestionar la información de los usuarios*
Complejidad	*Alta.*

Prioridad	*Crítico.*	
Referencias	RF.19.	
Precondiciones	*El administrador debe verse autenticado antes de gestionar la información.*	
Flujo de eventos		
1. *El caso de uso se inicia cuando el **Administrador** se autentica para acceder a la aplicación, y gestionar la información de los usuarios.* 2. *La aplicación comprueba los datos introducidos.* 3. *Termina el caso de uso cuando la aplicación muestra las acciones que se pueden realizar.*		
Relaciones	**CU Incluidos**	*CU Autenticar Usuario Ver CU: Autenticar Usuario*

Durante el desarrollo del capítulo se ha podido lograr un mejor entendimiento del sistema y las restricciones que deben existir para satisfacer las necesidades del cliente. Se especificaron todos los requisitos funcionales y no funcionales del sistema, se identificaron los actores que intervienen. Así como todos los casos de uso, que mediante su descripción detallada se pudo obtener un mayor entendimiento de las funcionalidades recogidas en los requisitos.

CAPÍTULO 3. ANÁLISIS Y DISEÑO

En este capítulo se describe la arquitectura del sistema, se definen las clases de diseño del software, así como los diagramas de interacción correspondientes, para que se cumplan los requerimientos funcionales y no funcionales además del modelo de datos.

3.1. Arquitectura del sistema

El sistema propuesto constituye una aplicación WEB, que cuenta con una base de datos donde se almacenará la información. La comunicación entre los clientes y el servidor web será mediante el protocolo HTTP[5] o HTTPS[6] si fuese necesario asegurar la transmisión de la información entre los clientes y el servidor web. El enlace entre el servidor de bases de datos y el servidor web será utilizando en protocolo TCP/IP[7](ver Figura1).

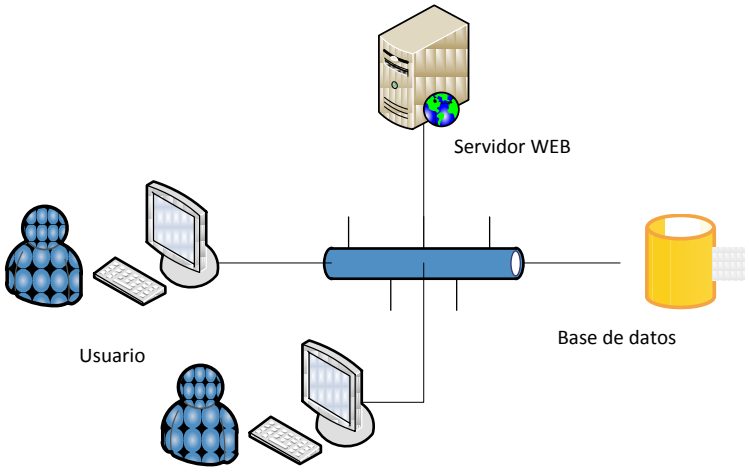


Figura 1

[5]HyperText Transfer Protocol: Protocolo de transferencia de hipertexto (utiliza el puerto TCP/IP 80)
[6]Secured HyperText Transfer Protocol: Protocolo seguro de transferencia de hipertexto (utiliza el puerto TCP/IP 443). Utiliza diferentes algoritmos de cifrado para asegurar la información.
[7] Transfer Control Protocol / Internet Protocol: Familia de protocolos que sustentan la mayoría de las comunicaciones y transmisión de información sobre Internet.

Siguiendo las tendencias actuales para la construcción de este tipo de aplicaciones, la arquitectura del sistema estará sustentada por el patrón de diseño Model-View-Controller (MVC)[8] (14), el cuál estipula la separación de los diferentes elementos que componen la presentación de una aplicación, mediante la asignación de roles específicos a tres componentes (ver Figura 2).

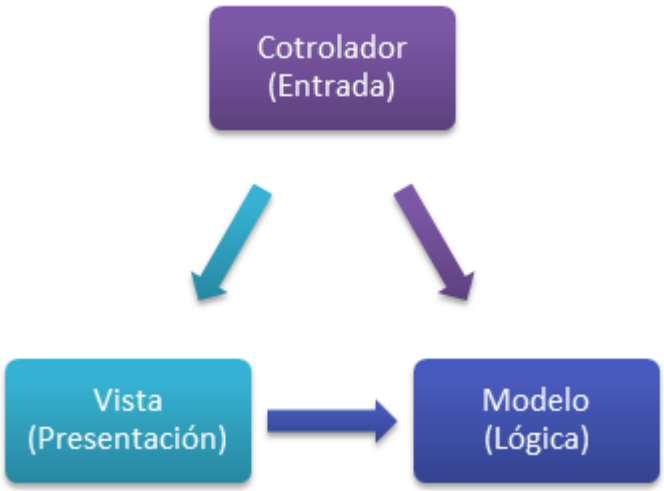


Figura 2

El Controlador es el responsable de manejar la entrada de los usuarios. Una vez que se han recibido datos de entrada el Controlador realizará las acciones u operaciones necesarias, lo cual puede incluir interactuar con el Modelo.

El Modelo representa el núcleo de la lógica/negocio de la aplicación. Una vez que el Controlador obtiene datos del modelo o realiza alguna acción con el modelo se necesita construir un modelo de presentación que describa al Modelo en términos que la Vista pueda comprender.

La Vista es la representación visual del Modelo. Esta presenta los datos del modelo al usuario de modo que estos tengan un significado comprensible. En una aplicación web esto generalmente constituye el HTML que será mostrado en navegador del usuario.

[8]Modelo-Vista-Controlador

Con la puesta a punto de estos tres elementos, la capa de presentación queda limpiamente separada, de modo que cada componente puede ser desarrollado o probado de forma independiente, simplificando y facilitando en gran medida la implementación y el soporte de la aplicación.

3.2. Modelo de Diseño

El propósito del Modelo Diseño no es más que modelar el sistema y encontrar su forma para que soporte todos los requisitos, creando una entrada apropiada y un punto de partida para las actividades de implementación, capturando los requisitos o subsistemas individuales interfaces y clases, para entender mejor el modelo (15)(9), a continuación se muestran los modelos de las distintas clases diseño y se describen las mismas.

3.2.1. Diagramas de Clases del Diseño

Los diagramas de clases del diseño ilustran las relaciones entre las clases y su funcionamiento; con el objetivo de proporcionar más claridad a los desarrolladores que implementarán el sistema.

Ver Anexo 1 – Diagramas de Clases del Diseño.

3.2.2. Descripción de las Clases del Diseño

Nombre: (1) CP: Gestionar Error	
Tipo de clase Interfaz.	
Para cada responsabilidad:	
Descripción:	Se construye al acceder para adicionar, modificar o eliminar un error, e incluye los formularios.

Nombre: (1) SP: Gestionar Error	
Tipo de clase Controladora.	
Atributo	**Tipo**

Error	String
Para cada responsabilidad:	
Nombre:	Registrar Error
Descripción:	Permite adicionar o registrar un error
Nombre:	Modificar Error
Descripción:	Permite modificar un error
Nombre:	Eliminar Error
Descripción:	Permite eliminar un error

Nombre: (1) CP: Gestionar Solución	
Tipo de clase Interfaz.	
Para cada responsabilidad:	
Descripción:	Se construye al acceder para adicionar, modificar o eliminar una Solución, e incluye los diferentes formularios.

Nombre: (1) SP: Gestionar Solución	
Tipo de clase Controladora.	
Atributo	**Tipo**
Solución	String
Para cada responsabilidad:	
Nombre:	Registrar Solución

Descripción:	Permite adicionar o registrar una solución
Nombre:	Modificar Solución
Descripción:	Permite modificar una solución
Nombre:	Eliminar solución
Descripción:	Permite eliminar una solución

Nombre: (1) SP: Gestionar Usuario	
Tipo de clase Controladora.	
Atributo	**Tipo**
Usuario	String
Para cada responsabilidad:	
Nombre:	Registrar Usuario
Descripción:	Permite adicionar o registrar un Usuario
Nombre:	Modificar Usuario
Descripción:	Permite modificar un Usuario
Nombre:	Eliminar Usuario
Descripción:	Permite eliminar un Usuario

3.2.3. Diagramas de Interacción

Los diagramas de interacción pueden ser de secuencia o colaboración. Se utilizan para modelar los aspectos dinámicos del sistema. Su único objetivo es ilustrar la forma en que interactúan entre los diferentes elementos que participan en la realización de un caso de uso proporcionando al igual que las clases más claridad a los desarrolladores que implementarán el sistema. Ver

Anexo 2 – Diagramas de Interacción.

3.3. Modelo de Datos

El modelo de datos representa las entidades persistentes del sistema así como las relaciones existentes entre éstas.

3.3.1. Descripción de las tablas

Nombre: (Error)		
Tablas foráneas: (Las tablas que tributan información son Usuario, Categoría, SistemaOperativo, Tecnología, Lenguaje)		
Dependencias: (ComentarioError, Solución)		
Atributo	**Tipo**	**Descripción**
(Id)	(Int)	(Es un número para identificar dicho error)
Asunto	nvarchar	Acerca de que trata el error.
Detalles	nvarchar	Explicación de la información del error.
Fecha	DateTime	Fecha en que se realiza alguna operación con el error

Nombre: (Usuario)		
Tablas foráneas: -		
Dependencias: (Solución, Error, ComentarioError, ComentarioSolución)		
Atributo	**Tipo**	**Descripción**
(Id)	(int)	(Identificación del usuario)
Usr	nvarchar	Nombre con el que se va identificar en la

		aplicación.
Nombre	nvarchar	Nombre del usuario.
Apellido	nvarchar	Apellidos del usuario.
Correo	nvarchar	Dirección electrónica para contactar al usuario
Administrador	Bit	Es una especificación del usuario.

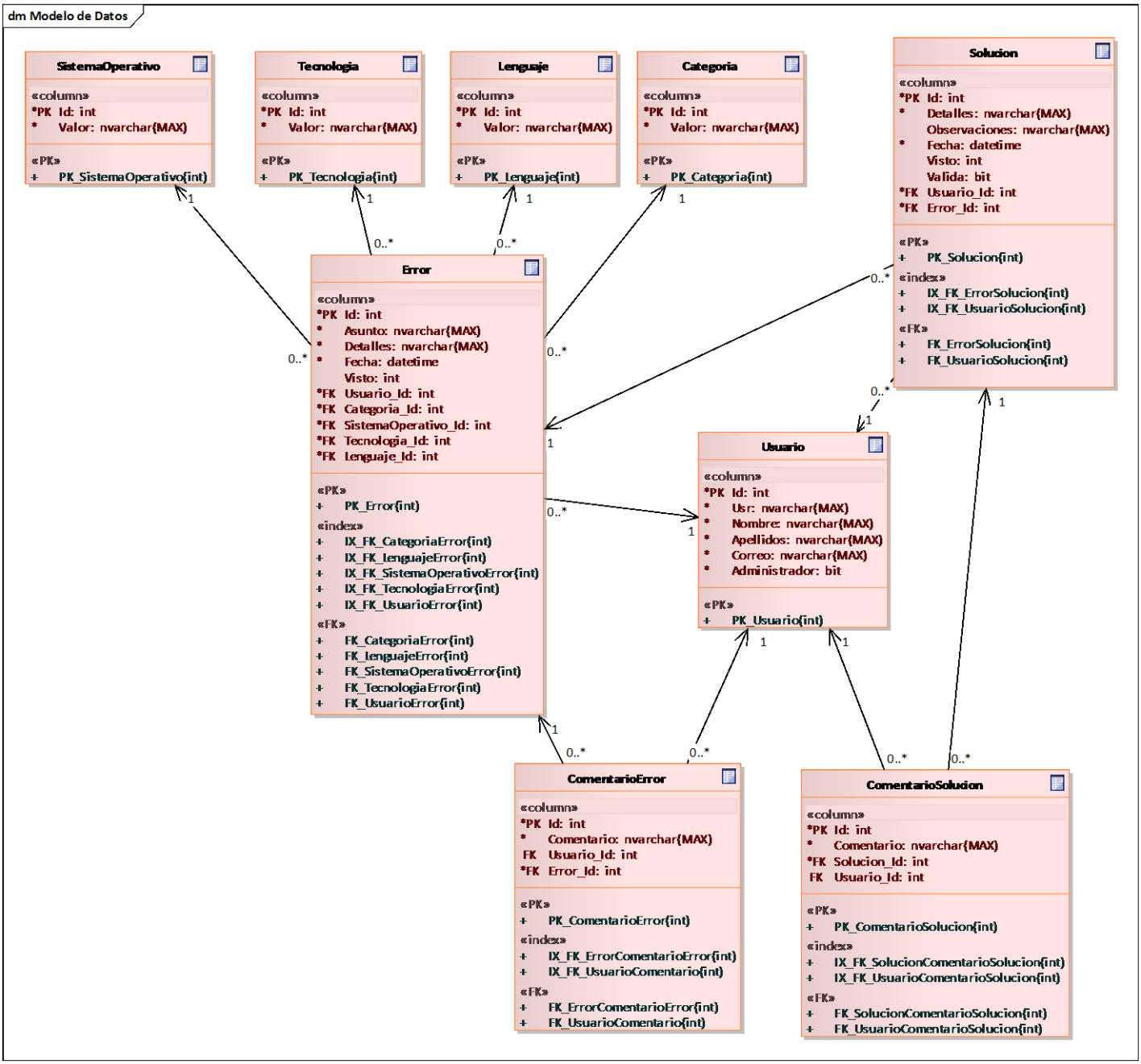
Nombre: (Solución)		
Tablas foráneas: Error, Usuario.		
Dependencias: (ComentarioSolución)		
Atributo	**Tipo**	**Descripción**
(Id)	(int)	(Identificación de la solución)
Detalles	nvarchar	Explicación de la información de la solución.
Observaciones	nvarchar	Información que se deba resaltar de la solución.
Fecha	DateTime	Fecha en que se realiza alguna operación con la solución

Nombre: (ComentarioSolución)		
Tablas foráneas: Solución, Usuario		
Dependencias: (-)		
Atributo	**Tipo**	**Descripción**
(Id)	(int)	(Identificación de ComentarioSolución)
Comentario	nvarchar	Información de la solución.

Nombre: (ComentarioError)		
Tablas foráneas: Error, Usuario		
Dependencias: (-)		
Atributo	**Tipo**	**Descripción**
(Id)	(int)	(Identificación de ComentarioError)
Comentario	nvarchar	Información del Error.

En el Capítulo se realizó la descripción de las clases de los diagramas de diseño, además de construir los diagramas de interacción, se hizo una breve descripción de la arquitectura del sistema, se describen las tablas del modelo de datos logrando el mismo como se muestra a continuación.

3.3.2. Modelo Entidad – Relación



CONCLUSIONES

Después del estudio realizado, la información recopilada sobre los antecedentes existentes en el mundo constituye la base para la toma de decisiones respecto a la manera de estructurar la solución, se proponen las herramientas y metodología para el desarrollo de la aplicación. Fueron elaborados y descritos brevemente algunos de los artefactos propuestos por RUP para el desarrollo de software, tal como: el modelo de dominio.

Además se ha logrado alcanzar un mejor entendimiento del sistema y las restricciones que deben existir para satisfacer las necesidades del cliente. Se especificaron todos los requisitos funcionales y no funcionales del sistema, se identificaron los actores que intervienen. La arquitectura del sistema quedó definida y se llevó a cabo la realización de los casos de uso, mediante los artefactos diagramas de interacción y de clases del diseño; se desarrolló además el modelo de datos.

El diseño de la aplicación, posee un conjunto de funcionalidades que dan respuesta a los requisitos planteados, y con ello se cumple con los objetivos propuestos para la presente investigación, sentando las bases para la posterior implementación del mismo, tal como plantea RUP.

RECOMENDACIONES

Se recomienda que la aplicación sea implementada, con el objetivo de facilitar el trabajo de los equipos de desarrollo del centro de informática médica para así disminuir la pérdida de tiempo en la búsqueda de soluciones a los errores que se le presentan.

Estimular a los desarrolladores para que utilicen la aplicación una vez implementada, y así acumular rápidamente la experiencia de estos para próximas generaciones de estudiantes.

Estudiar la posibilidad de poder adaptar la aplicación para que pueda ser utilizada en todos los centros de desarrollo de la universidad.

REFERENCIAS BIBLIOGRÁFICAS

1. **González, Lazaro y Durañona, Yanoksy.** *Cassandra Server.* 2007.

2. **Zozaya, C.** REFLEXIONES Y RECOMENDACIONES SOBRE EDUCACIÓN EN INFORMÁTICA. *Biblioteca Raúl Baillères Jr.* [En línea] 2004. http://biblioteca.itam.mx/estudios/estudio/letras39-40/texto11/sec_1.html.

3. **Casallas, R. O. R.** Gestión de conocimiento para la reutilización de la experiencia obtenida en la corrección de defectos de software. *Paradigma.* [En línea] 2008. http://paradigma.uniandes.edu.co/index.php?option=com_content&task=view&id=71&Itemid=1&lang=es&showall=1. 2011-0065.

4. **StackApps.** *StackOverflow.* [En línea] http://www.stackoverflow.com.

5. **The Code Project.** *The Code Project.* [En línea] http://www.codeproject.com.

6. **Experts-Exchange.** *Experts-Exchange.* [En línea] http://www.experts-exchange.com/.

7. —. Experts-Exchange's Help Page. *Experts-Exchange.* [En línea] http://www.experts-exchange.com/help.jsp#hi4.

8. **EVA-UCI.** Curso Ingeniería de Software I 2010-2011. *Entorno Virtual de Aprendizaje.* [En línea] 2010. http://eva.uci.cu/course/view.php?id=102.

9. **Pressman, R. S.** *Ingeniería del software. Un enfoque práctico.* s.l. : McGraw Hill Higher Education.

10. **Sommerville, I.** *Ingeniería de Software.* s.l. : Addison Wesley.

11. **Microsoft Corp.** *Visual Studio.* [En línea] 2011. http://www.microsoft.com/visualstudio/.

12. —. Asp.Net. *Asp.Net Mvc.* [En línea] http://www.asp.net/mvc.

13. **Apache Software Foundation.** Lucene.Net Features. *Lucene.Net.* [En línea] http://lucene.apache.org/java/docs/features.html.

14. **EVA-UCI.** Patrones de arquitectura. *Inter-Nos.* [En línea] 2010. mms://ucimedia.uci.cu/teleclases/1er_Semestre/3er/Ingenieria_de_Software_I/conf7.

15. —. Fase de Elaboración. Análisis - Diseño. *Inter-Nos.* [En línea] 2010. mms://ucimedia.uci.cu/teleclases/1er_Semestre/3er/Ingenieria_de_Software_I/conf5.

BIBLIOGRAFÍA

1. **González, Lazaro y Durañona, Yanoksy.** *Cassandra Server.* 2007.

2. **Zozaya, C.** REFLEXIONES Y RECOMENDACIONES SOBRE EDUCACIÓN EN INFORMÁTICA. *Biblioteca Raúl Baillères Jr.* [En línea] 2004. http://biblioteca.itam.mx/estudios/estudio/letras39-40/texto11/sec_1.html.

3. **Casallas, R. O. R.** Gestión de conocimiento para la reutilización de la experiencia obtenida en la corrección de defectos de software. *Paradigma.* [En línea] 2008. http://paradigma.uniandes.edu.co/index.php?option=com_content&task=view&id=71&Itemid=1&lang=es&showall=1. 2011-0065.

4. **StackApps.** *StackOverflow.* [En línea] http://www.stackoverflow.com.

5. **The Code Project.** *The Code Project.* [En línea] http://www.codeproject.com.

6. **Experts-Exchange.** *Experts-Exchange.* [En línea] http://www.experts-exchange.com/ranksDefined.jsp.

7. —. Experts-Exchange's Help Page. *Experts-Exchange.* [En línea] http://www.experts-exchange.com/help.jsp#hi4.

8. **EVA-UCI.** Curso Ingeniería de Software I 2010-2011. *Entorno Virtual de Aprendizaje.* [En línea] 2010. http://eva.uci.cu/course/view.php?id=102.

9. **Pressman, R. S.** *Ingeniería del software. Un enfoque práctico.* s.l. : McGraw Hill Higher Education.

10. **Sommerville, I.** *Ingeniería de Software.* s.l. : Addison Wesley.

11. **Microsoft Corp.** *Visual Studio.* [En línea] 2011. http://www.microsoft.com/visualstudio/.

12. —. Asp.Net. *Asp.Net Mvc.* [En línea] http://www.asp.net/mvc.

13. **Apache Software Foundation.** Lucene.Net Features. *Lucene.Net.* [En línea] http://lucene.apache.org/java/docs/features.html.

14. **EVA-UCI.** Patrones de arquitectura. *Inter-Nos.* [En línea] 2010. mms://ucimedia.uci.cu/teleclases/1er_Semestre/3er/Ingenieria_de_Software_I/conf7.

15. —. Fase de Elaboración. Análisis - Diseño. *Inter-Nos.* [En línea] 2010. mms://ucimedia.uci.cu/teleclases/1er_Semestre/3er/Ingenieria_de_Software_I/conf5.

16. **Tamayo, Karel y García, Jublar.** *Cassandra Clinic .* 2007.

17. Tech-Faq. [En línea] 2010. www.tech-faq.com/.../implementacion-de-software-a-traves-de-grupo-de-la-politica-de.htm.

18. **UsabilidadWeb.** Ingeniería de Software. *UsabilidadWeb.* [En línea] http://www.usabilidadweb.com.ar/ingenieria_software.php.

19. **Joven Club de Computación y Electrónica.** *Joven Club de Computación y Electrónica.* [En línea] http://www.jovenclub.cu.

20. **Experts-Exchange.** Experts-Exchange ranks and their definitions. *Experts-Exchange.* [En línea] http://www.experts-exchange.com/ranksDefined.jsp.

21. **Infomed.** *Portal de Salud de Cuba.* [En línea] http://www.sld.cu/.

22. **Definición.de.** Definición de Software. *Definición.de.* [En línea] http://definicion.de.

23. **EVA-UCI.** Fase de inicio. Levantamiento de requisitos. *Inter-Nos.* [En línea] 2010. mms://ucimedia.uci.cu/teleclases/1er_Semestre/3er/Ingenieria_de_Software_I/conf3.

24. **Batista, I. D. M. y Sacre, L. R. Q.** IMPORTANCIA DE LA INFORMÁTICA MÉDICA EN LA FORMACIÓN DEL MÉDICO GENERAL INTEGRAL. *Biblioteca Virtual en Salud de Cuba.* [En línea] http://bvs.sld.cu/revistas/san/vol5_3_01/san03301.pdf.

25. **Stack Apps.** Stack Overflow Internet Services. *Stack Overflow.* [En línea] http://stackapps.com/.

26. **R. R. Felipe, Rueda, Nicolás, López y Rubby, Casallas.** Administración de conocimiento en desarrollo de software para disminuir el esfuerzo de corregir defectos. *Universidad de los Andes.* [En línea] 2008. http://paradigma.uniandes.edu.co/.

27. **Fernández, Y. A.** Breve exposición de la informática en Cuba. *Revistas Electrónicas de la Universidad Complutense de Madrid.* [En línea] 2004. http://revistas.ucm.es/byd/11321873/articulos/RGID9494220195A.PDF.

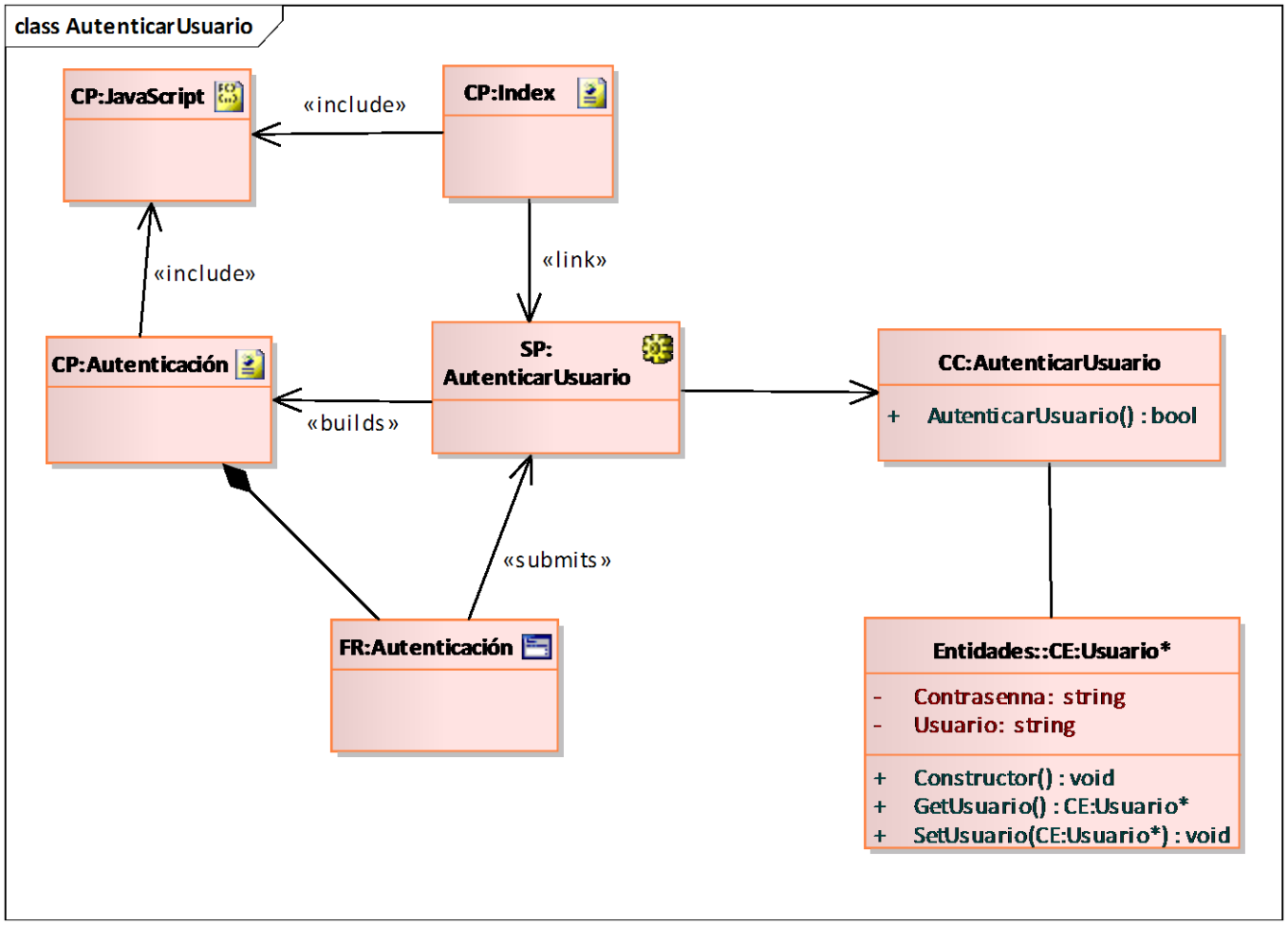
28. **González, C. D.** Desarrollo de Sitios Web de Calidad, Usables, Seguros, Válidos y Accesibles. *Desarrollo de software. Aplicaciones basadas en web.* [En línea] http://www.usabilidadweb.com.ar/ingenieria_software.php.

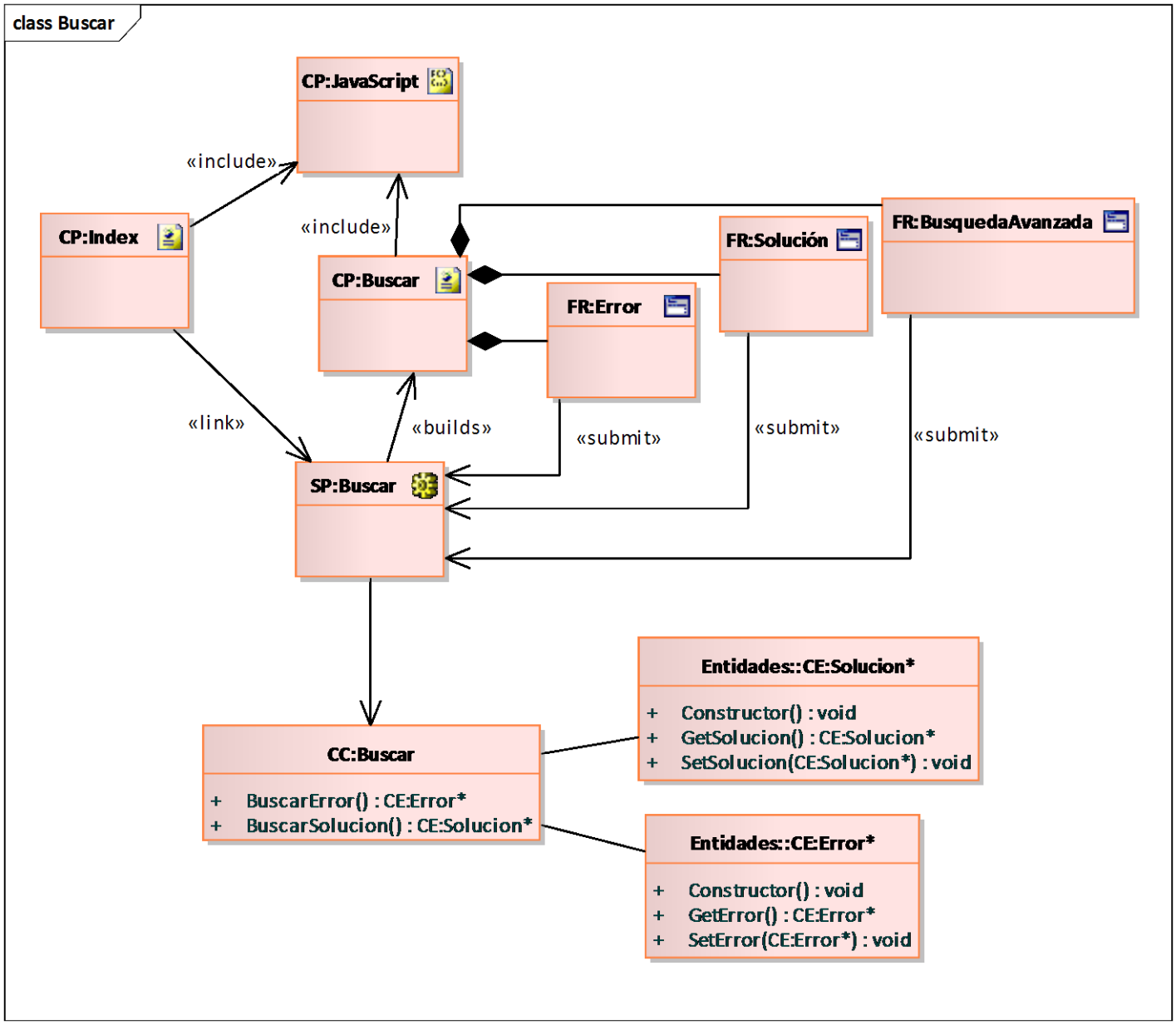
29. **Moron, R. G.** Desarrollo de la Medicina en los Siglos XX-XXI. *Dpto HC. Universidad Carlos III de Madrid.* [En línea] http://www.uc3m.es/uc3m/dpto/HC/SIGLOS/r,galan.doc.

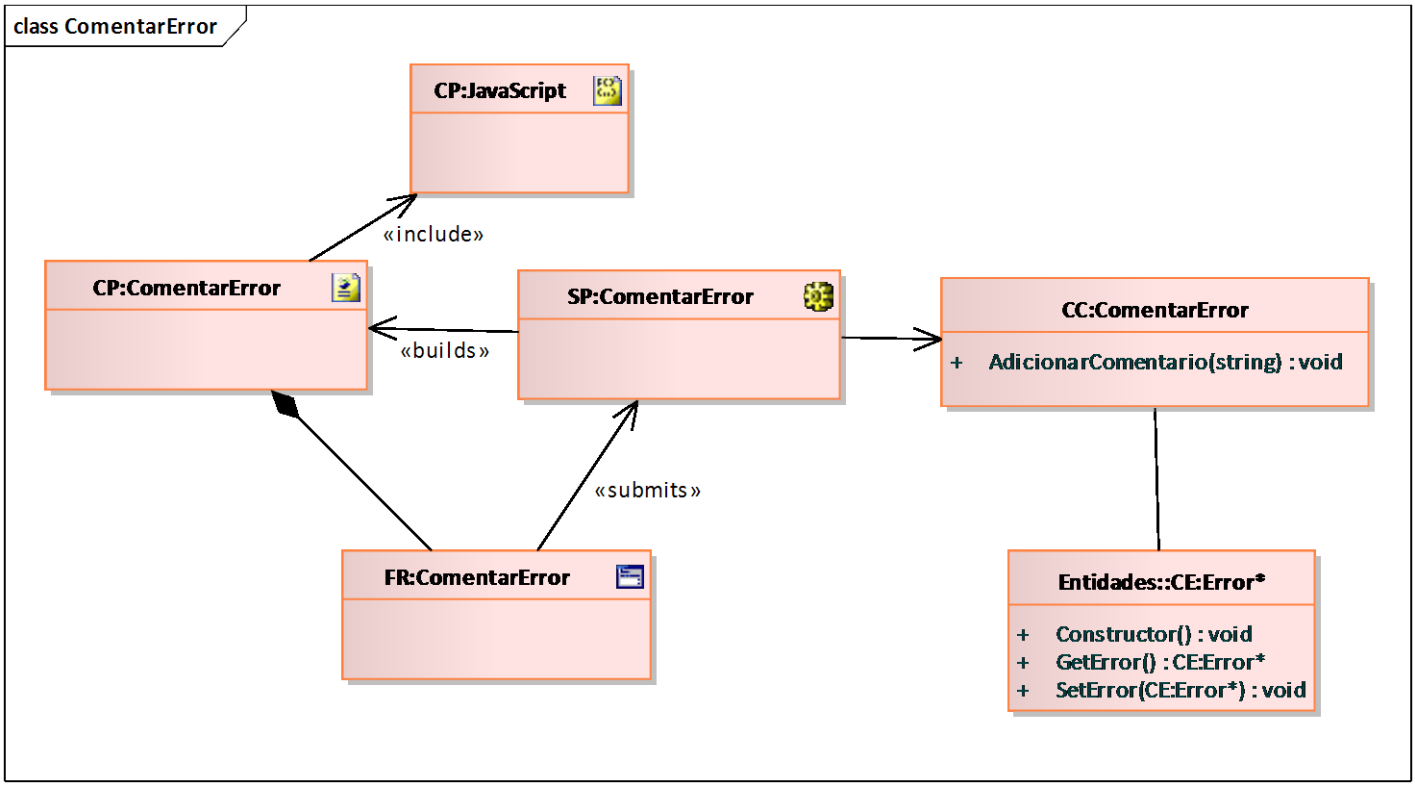
30. **Vergara, K.** Concepto y tipos de software: programas, definición. [En línea] http://www.bloginformático.com.

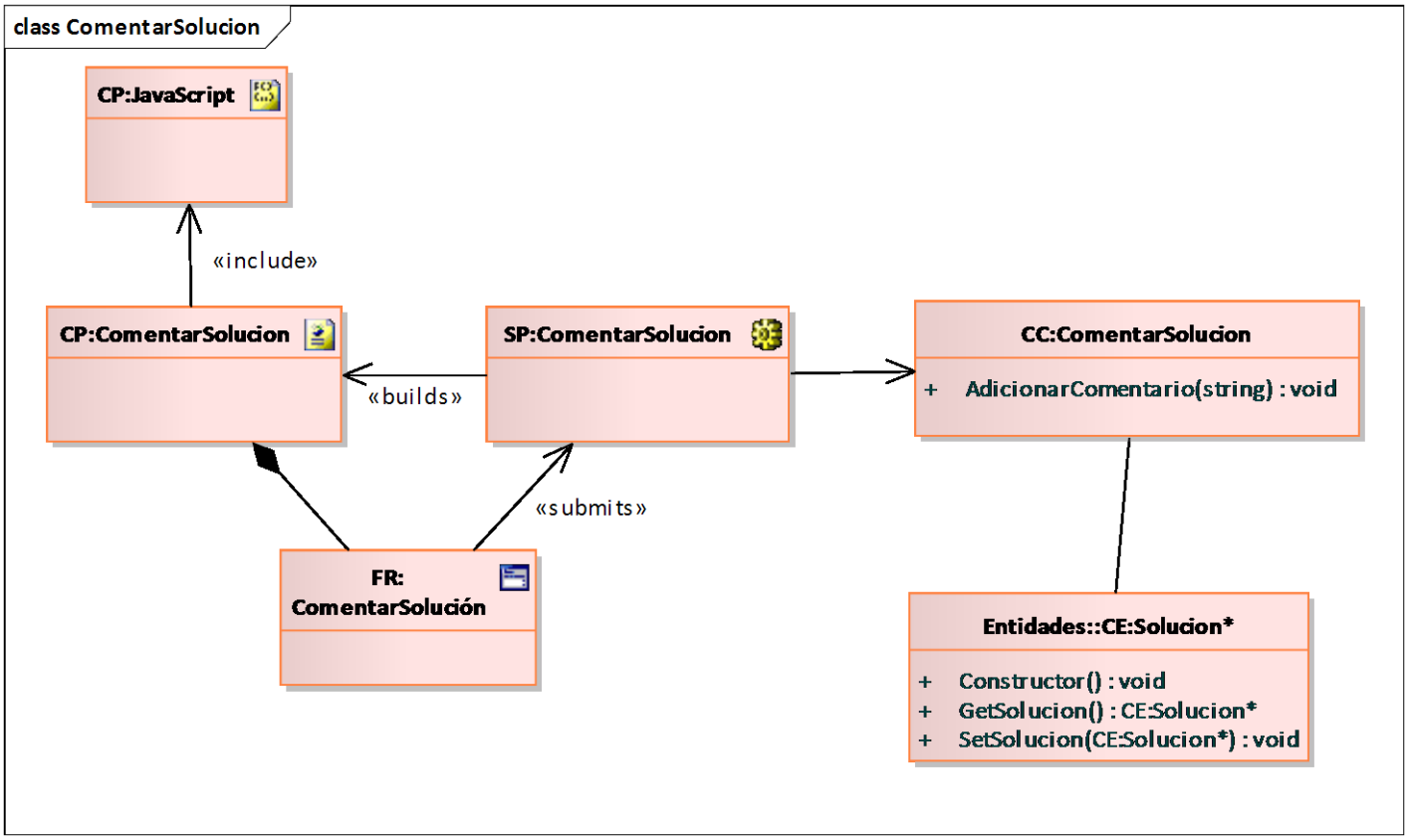
ANEXOS

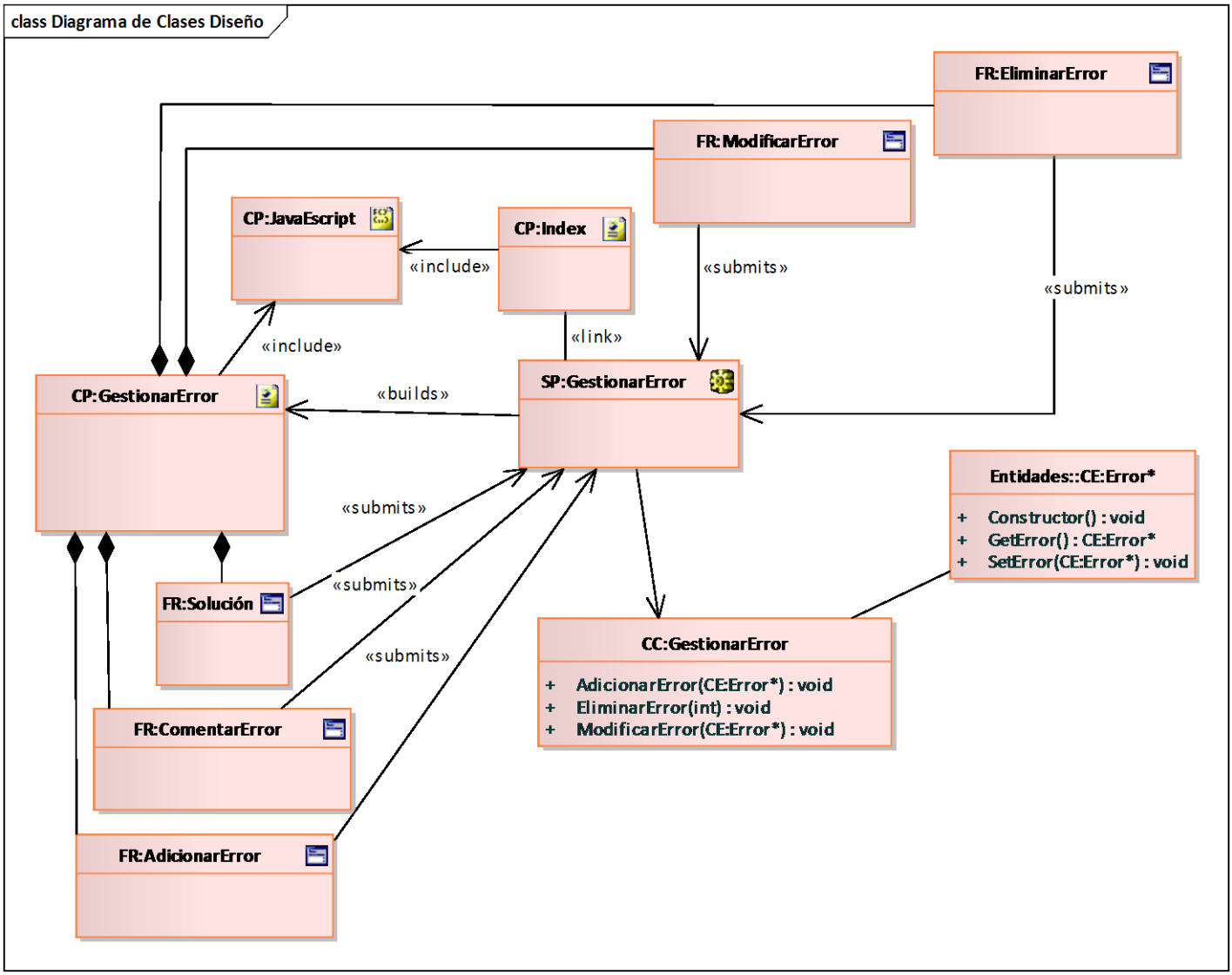
Anexo 1 – Diagramas de Clases del Diseño

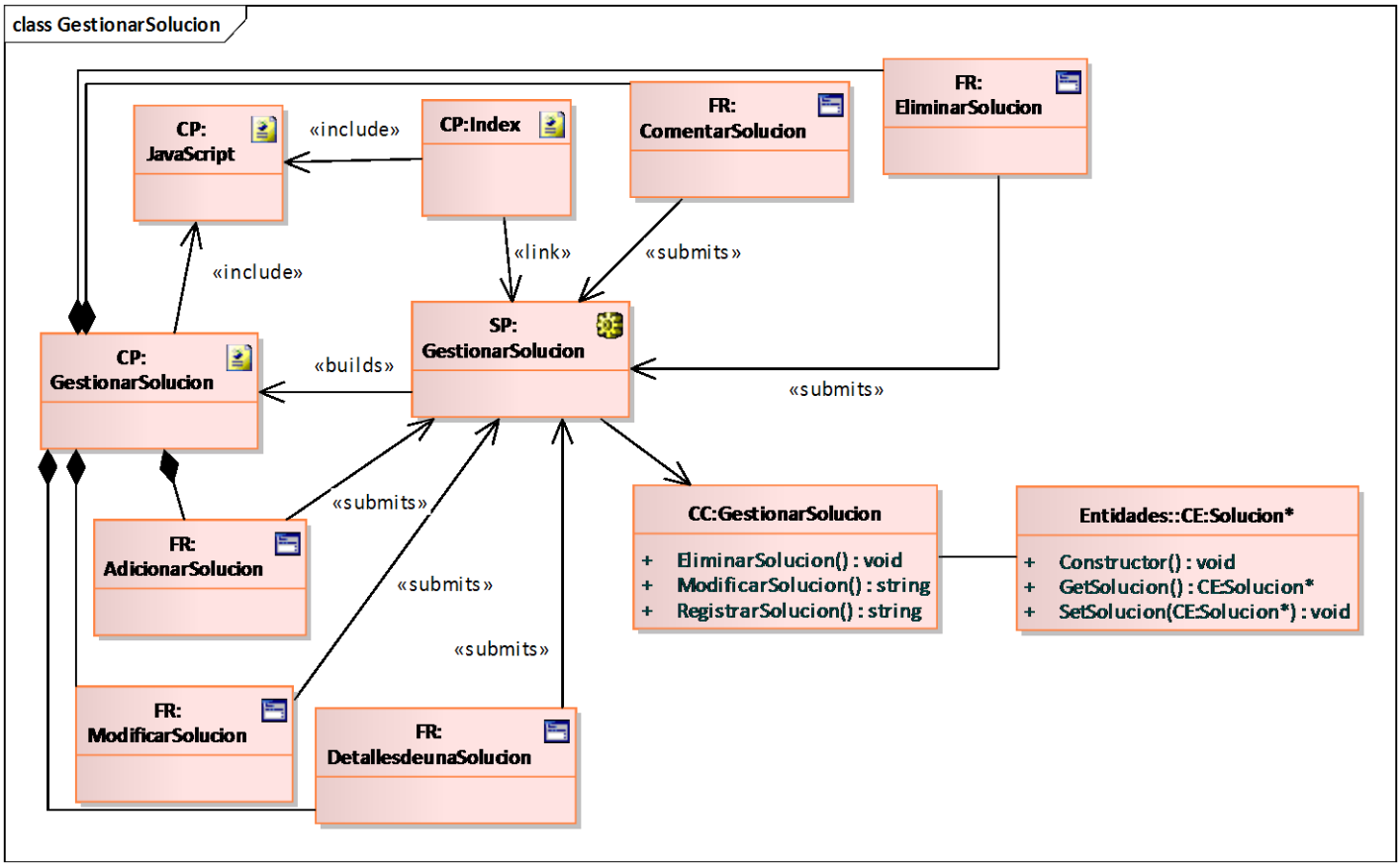


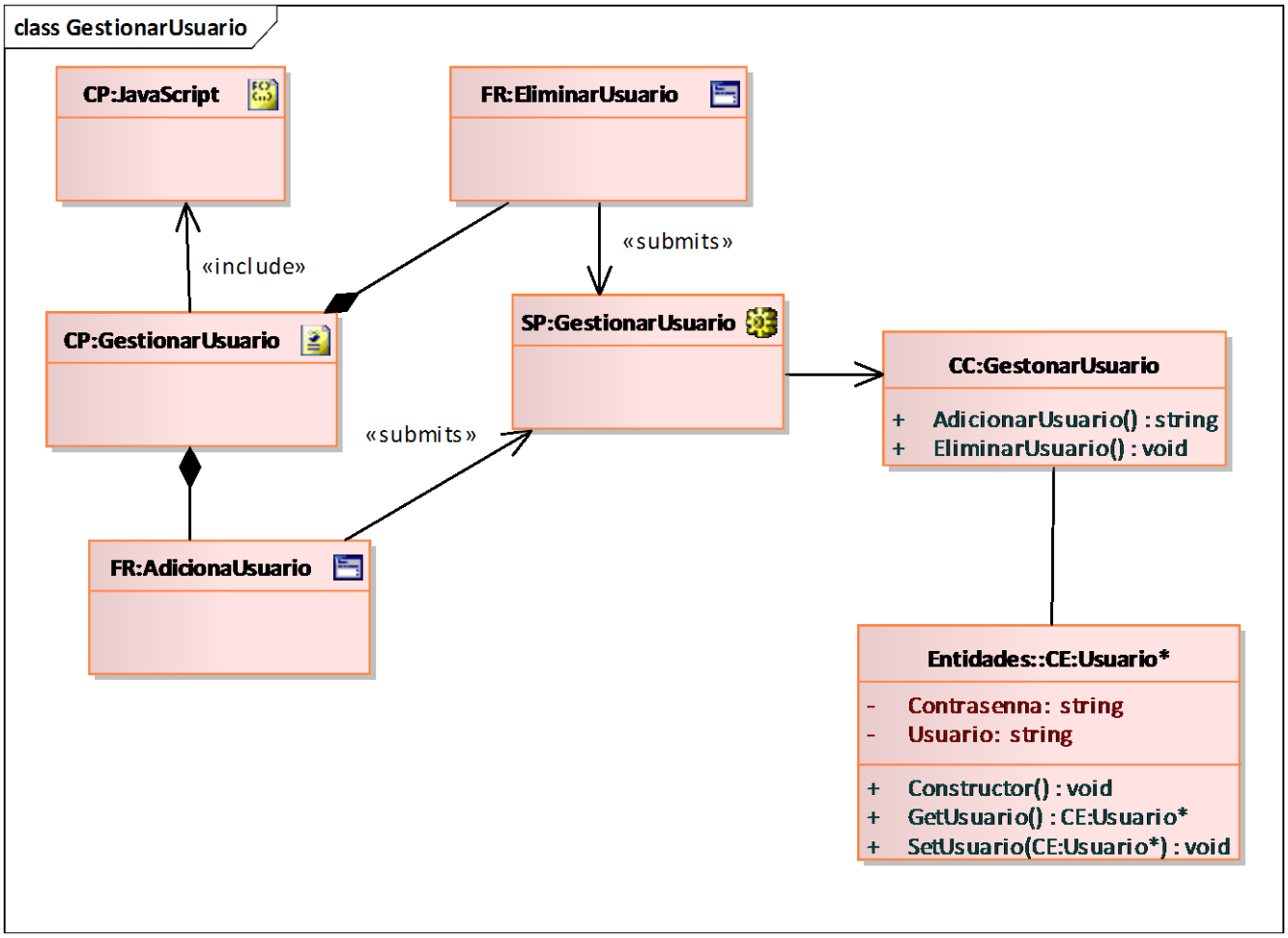

class Buscar
CP:JavaScript
«include»
«include»
CP:Index
CP:Buscar
FR:Solución
FR:BusquedaAvanzada
FR:Error
«link»
«builds»
«submit»
«submit»
«submit»
SP:Buscar
Entidades::CE:Solucion*
+ Constructor() : void
+ GetSolucion() : CE:Solucion*
+ SetSolucion(CE:Solucion*) : void
CC:Buscar
+ BuscarError() : CE:Error*
+ BuscarSolucion() : CE:Solucion*
Entidades::CE:Error*
+ Constructor() : void
+ GetError() : CE:Error*
+ SetError(CE:Error*) : void

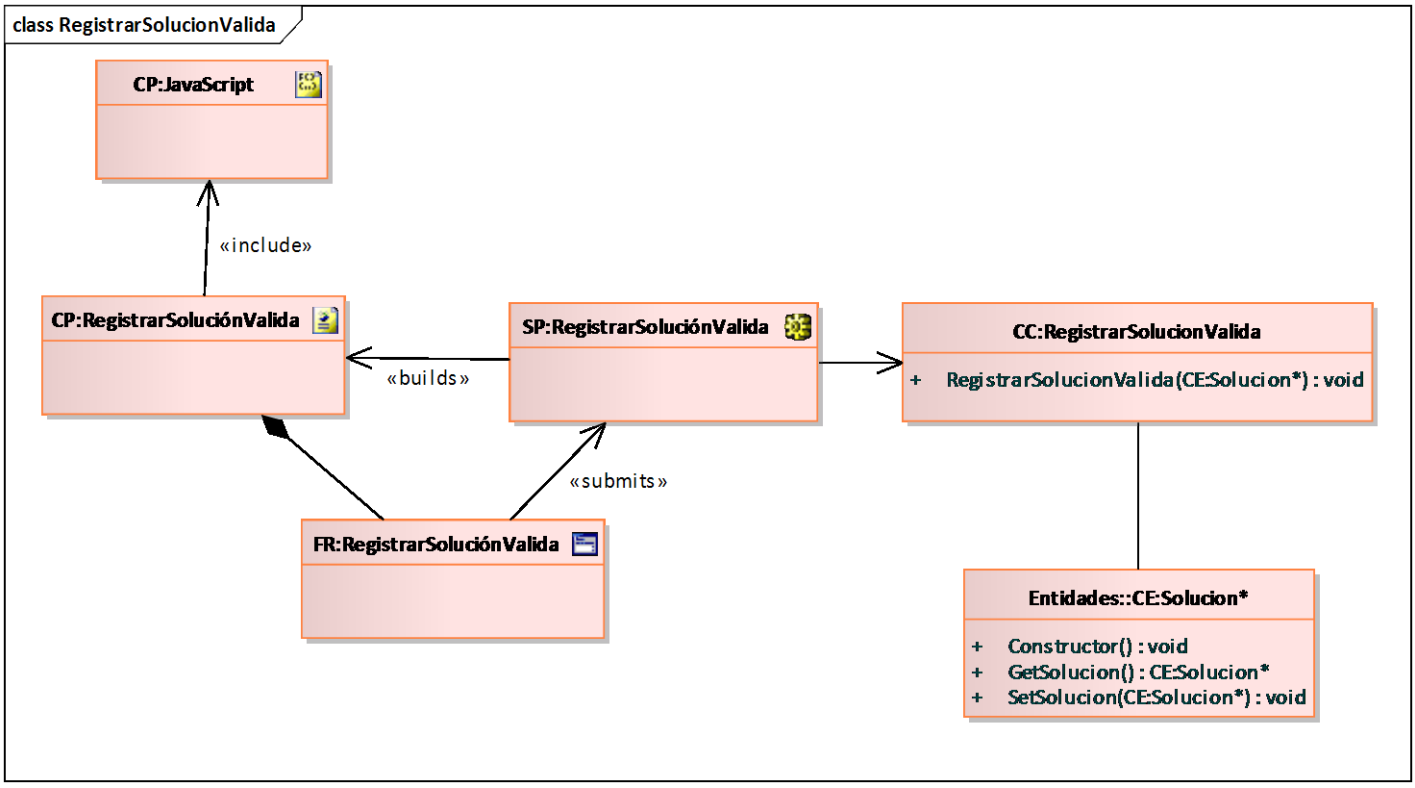

class ComentarError
CP:JavaScript
«include»
CP:ComentarError
«builds»
SP:ComentarError
CC:ComentarError
+ AdicionarComentario(string) : void
«submits»
FR:ComentarError
Entidades::CE:Error*
+ Constructor() : void
+ GetError() : CE:Error*
+ SetError(CE:Error*) : void


class ComentarSolucion
CP:JavaScript
«include»
CP:ComentarSolucion
«builds»
SP:ComentarSolucion
CC:ComentarSolucion
+ AdicionarComentario(string) : void
«submits»
FR: ComentarSolución
Entidades::CE:Solucion*
+ Constructor() : void
+ GetSolucion() : CE:Solucion*
+ SetSolucion(CE:Solucion*) : void

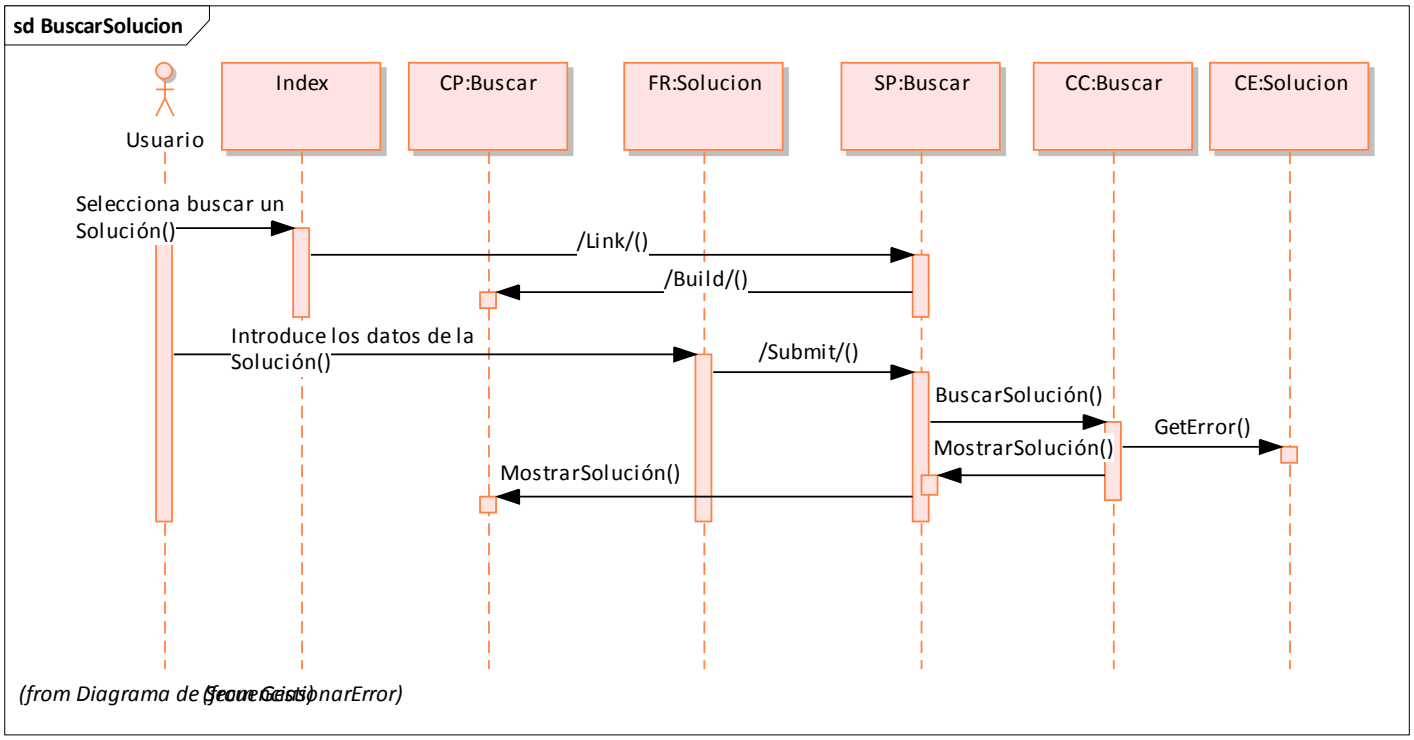

class Diagrama de Clases Diseño
FR:EliminarError
FR:ModificarError
CP:JavaEscript
CP:Index
«include»
«submits»
«submits»
«include»
«link»
CP:GestionarError
«builds»
SP:GestionarError
Entidades::CE:Error*
+ Constructor() : void
+ GetError() : CE:Error*
+ SetError(CE:Error*) : void
«submits»
«submits»
«submits»
FR:Solución
CC:GestionarError
+ AdicionarError(CE:Error*) : void
+ EliminarError(int) : void
+ ModificarError(CE:Error*) : void
FR:ComentarError
FR:AdicionarError

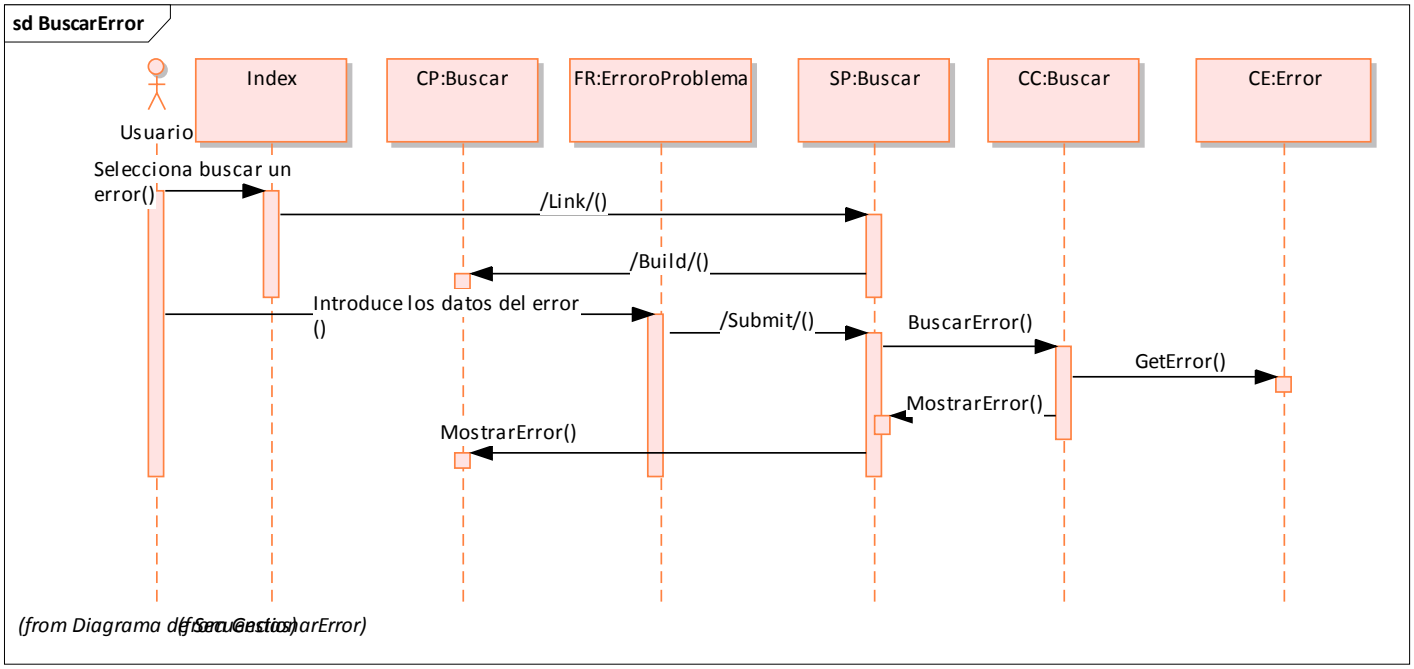

class GestionarSolucion
CP: JavaScript
«include»
CP:Index
FR: ComentarSolucion
FR: EliminarSolucion
«include»
«link»
«submits»
CP: GestionarSolucion
«builds»
SP: GestionarSolucion
«submits»
«submits»
FR: AdicionarSolucion
CC:GestionarSolucion
+ EliminarSolucion() : void
+ ModificarSolucion() : string
+ RegistrarSolucion() : string
Entidades::CE:Solucion*
+ Constructor() : void
+ GetSolucion() : CE:Solucion*
+ SetSolucion(CE:Solucion*) : void
«submits»
«submits»
FR: ModificarSolucion
FR: DetallesdeunaSolucion

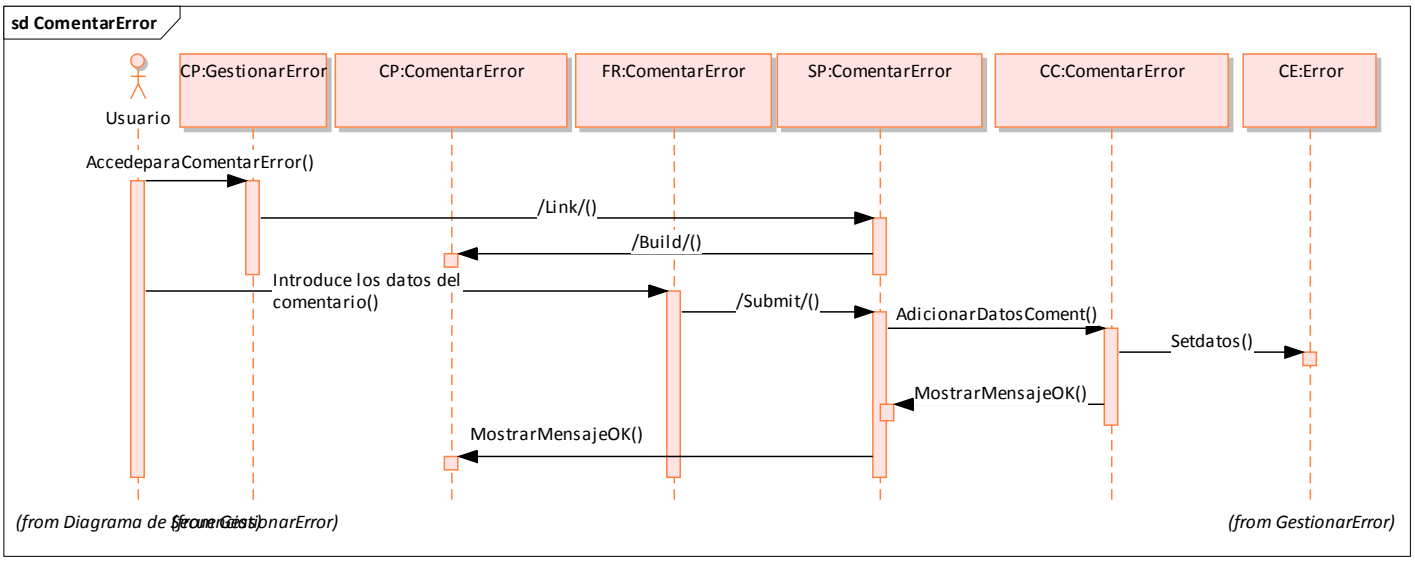

class GestionarUsuario
CP:JavaScript
FR:EliminarUsuario
«include»
«submits»
CP:GestionarUsuario
SP:GestionarUsuario
CC:GestonarUsuario
+ AdicionarUsuario() : string
+ EliminarUsuario() : void
«submits»
FR:AdicionaUsuario
Entidades::CE:Usuario*
- Contrasenna: string
- Usuario: string
+ Constructor() : void
+ GetUsuario() : CE:Usuario*
+ SetUsuario(CE:Usuario*) : void

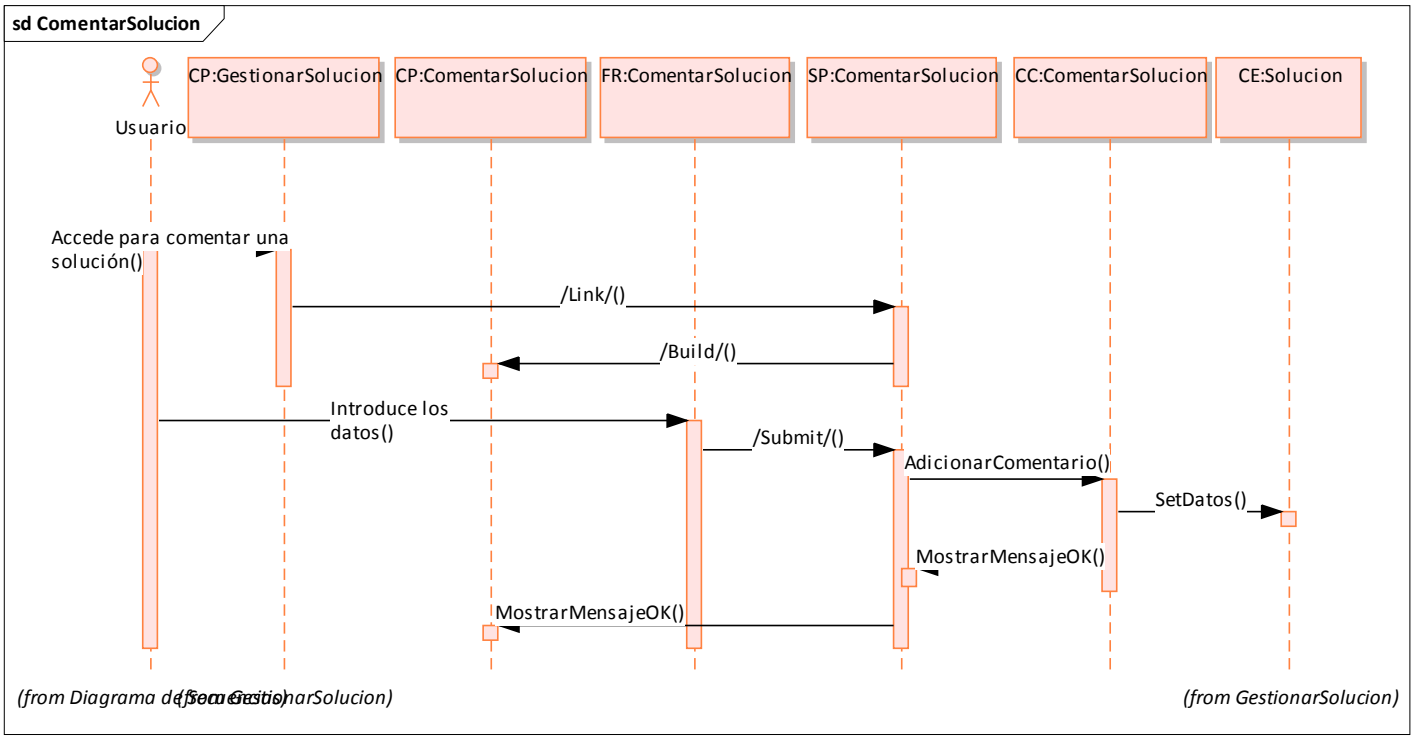

class RegistrarSolucionValida
CP:JavaScript
«include»
CP:RegistrarSoluciónValida
«builds»
SP:RegistrarSoluciónValida
CC:RegistrarSolucionValida
+ RegistrarSolucionValida(CE:Solucion*) : void
«submits»
FR:RegistrarSoluciónValida
Entidades::CE:Solucion*
+ Constructor() : void
+ GetSolucion() : CE:Solucion*
+ SetSolucion(CE:Solucion*) : void

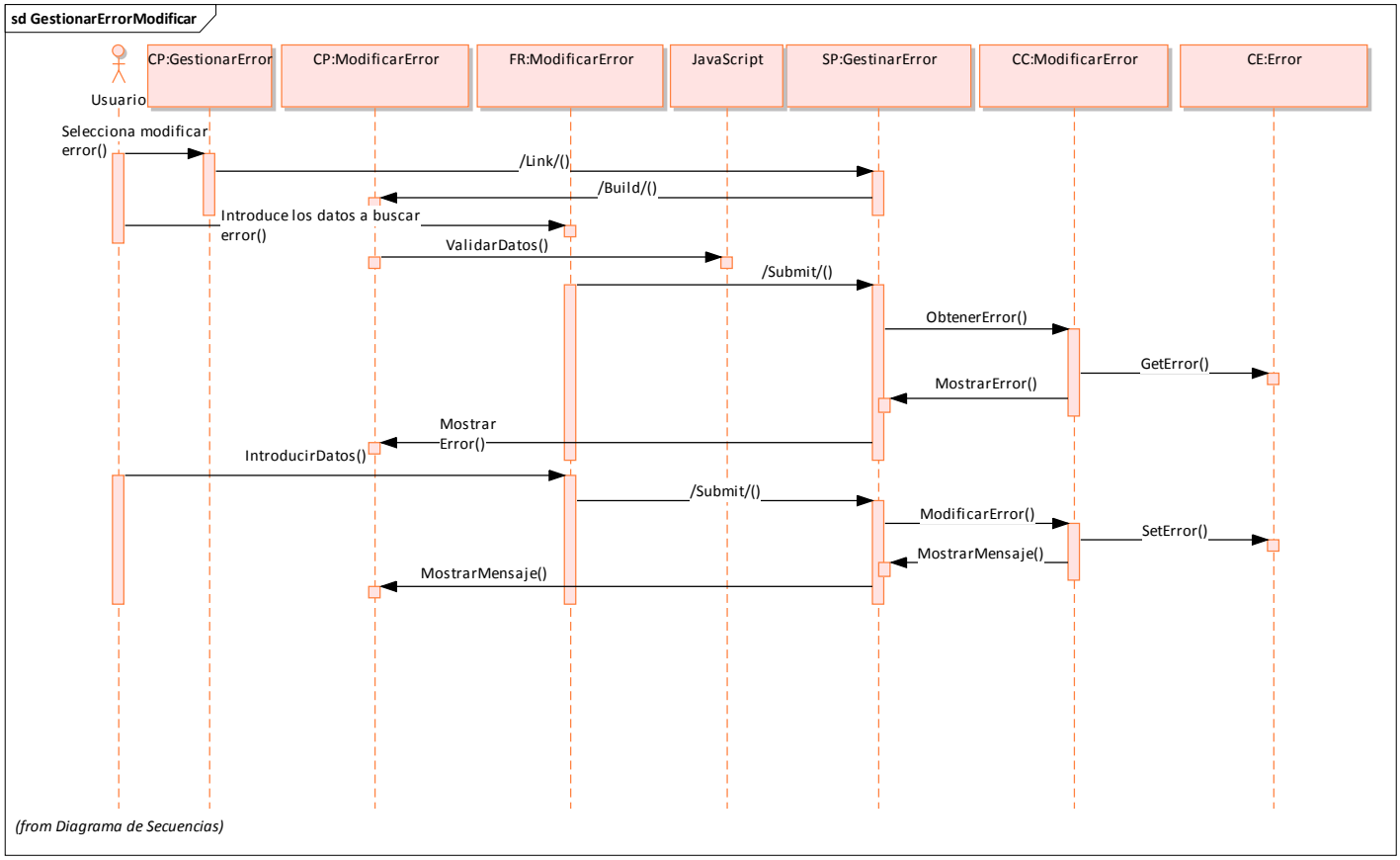
Anexo 2 – Diagramas de Interacción

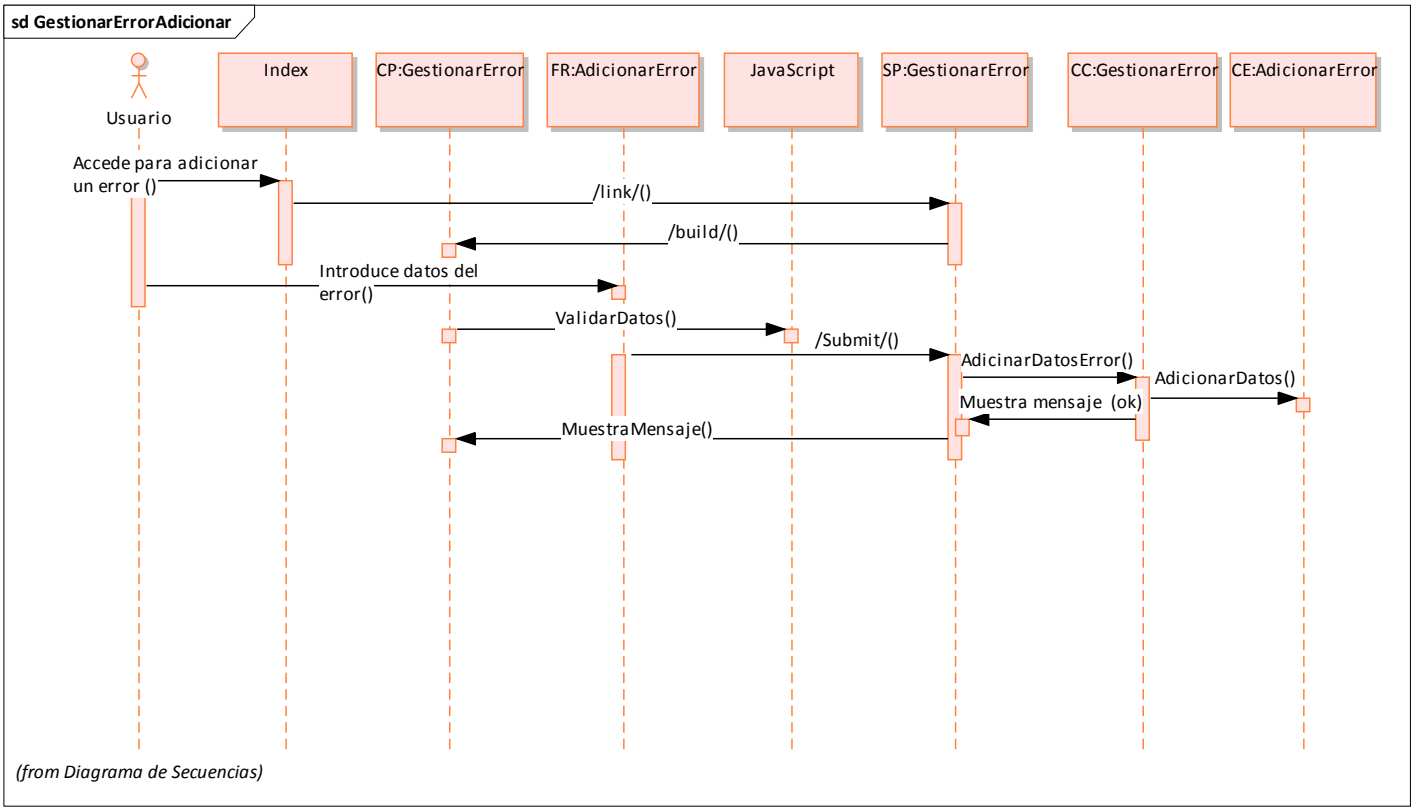


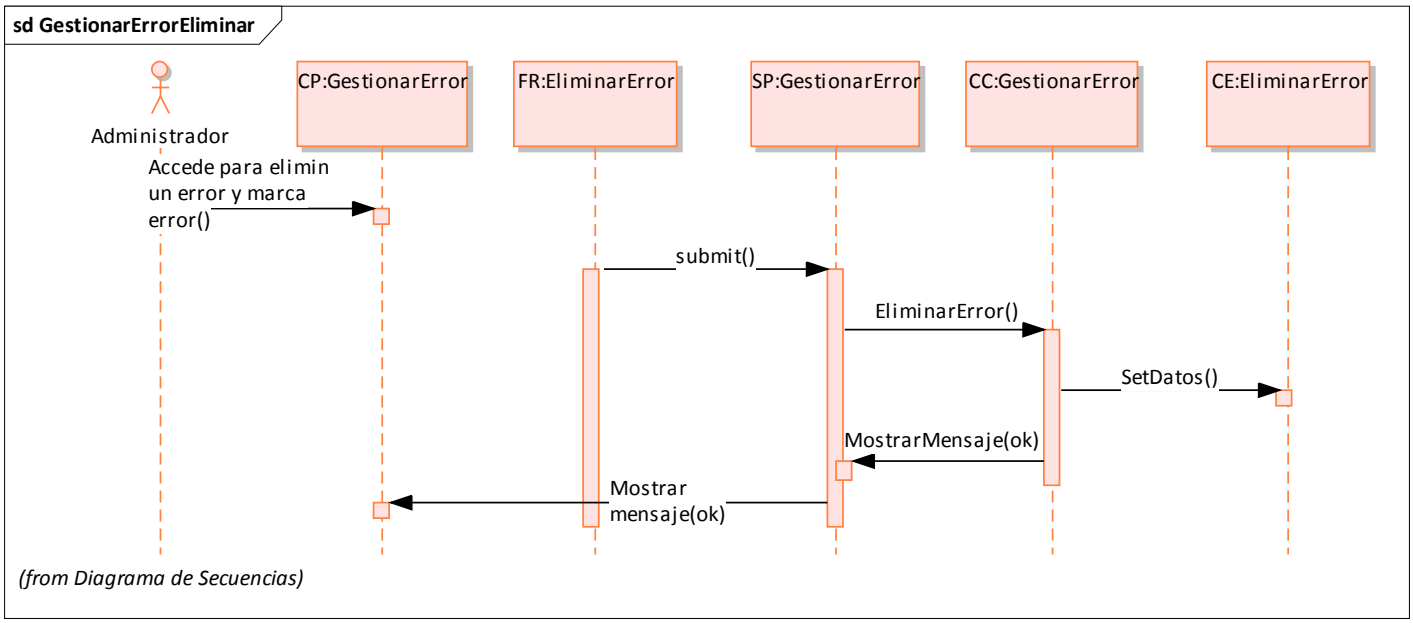

sd BuscarError
Usuario
Index
CP:Buscar
FR:ErroroProblema
SP:Buscar
CC:Buscar
CE:Error
Selecciona buscar un error()
/Link/()
/Build/()
Introduce los datos del error ()
/Submit/()
BuscarError()
GetError()
MostrarError()
MostrarError()
(from Diagrama de Secuencia)

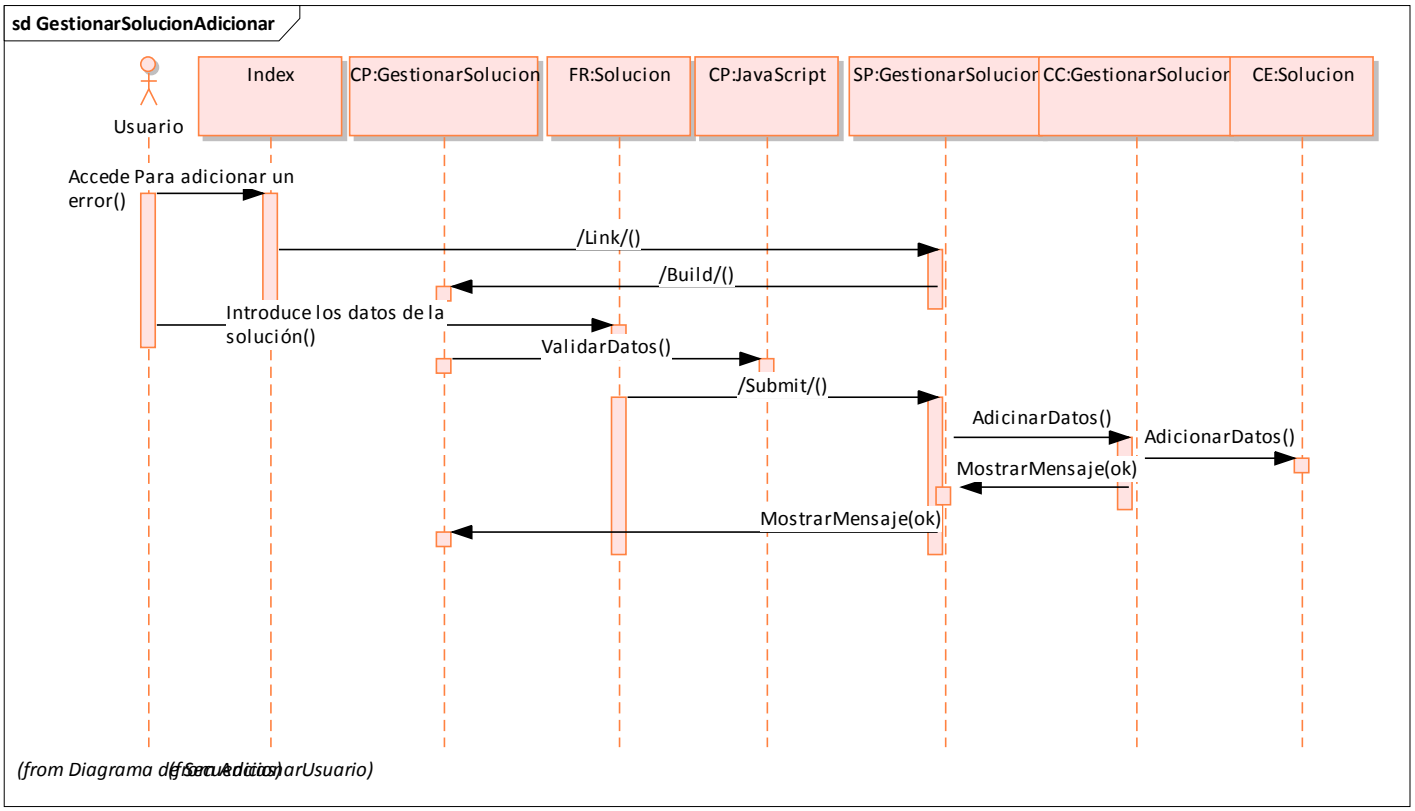

sd ComentarError
Usuario
CP:GestionarError
CP:ComentarError
FR:ComentarError
SP:ComentarError
CC:ComentarError
CE:Error
AccedeparaComentarError()
/Link/()
/Build/()
Introduce los datos del comentario()
/Submit/()
AdicionarDatosComent()
Setdatos()
MostrarMensajeOK()
MostrarMensajeOK()
(from Diagrama de Secuencia)
(from GestionarError)

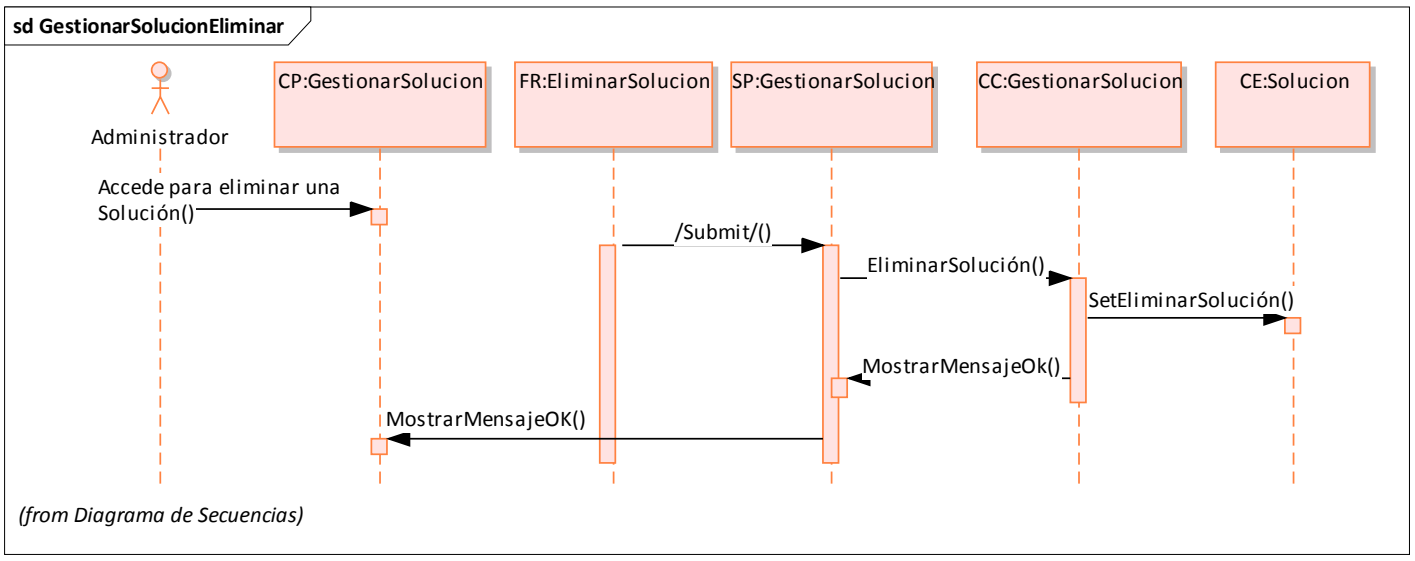

sd ComentarSolucion
Usuario
CP:GestionarSolucion
CP:ComentarSolucion
FR:ComentarSolucion
SP:ComentarSolucion
CC:ComentarSolucion
CE:Solucion
Accede para comentar una solución()
/Link/()
/Build/()
Introduce los datos()
/Submit/()
AdicionarComentario()
SetDatos()
MostrarMensajeOK()
MostrarMensajeOK()
(from Diagrama de Secuencia)
(from GestionarSolucion)
(from GestionarSolucion)

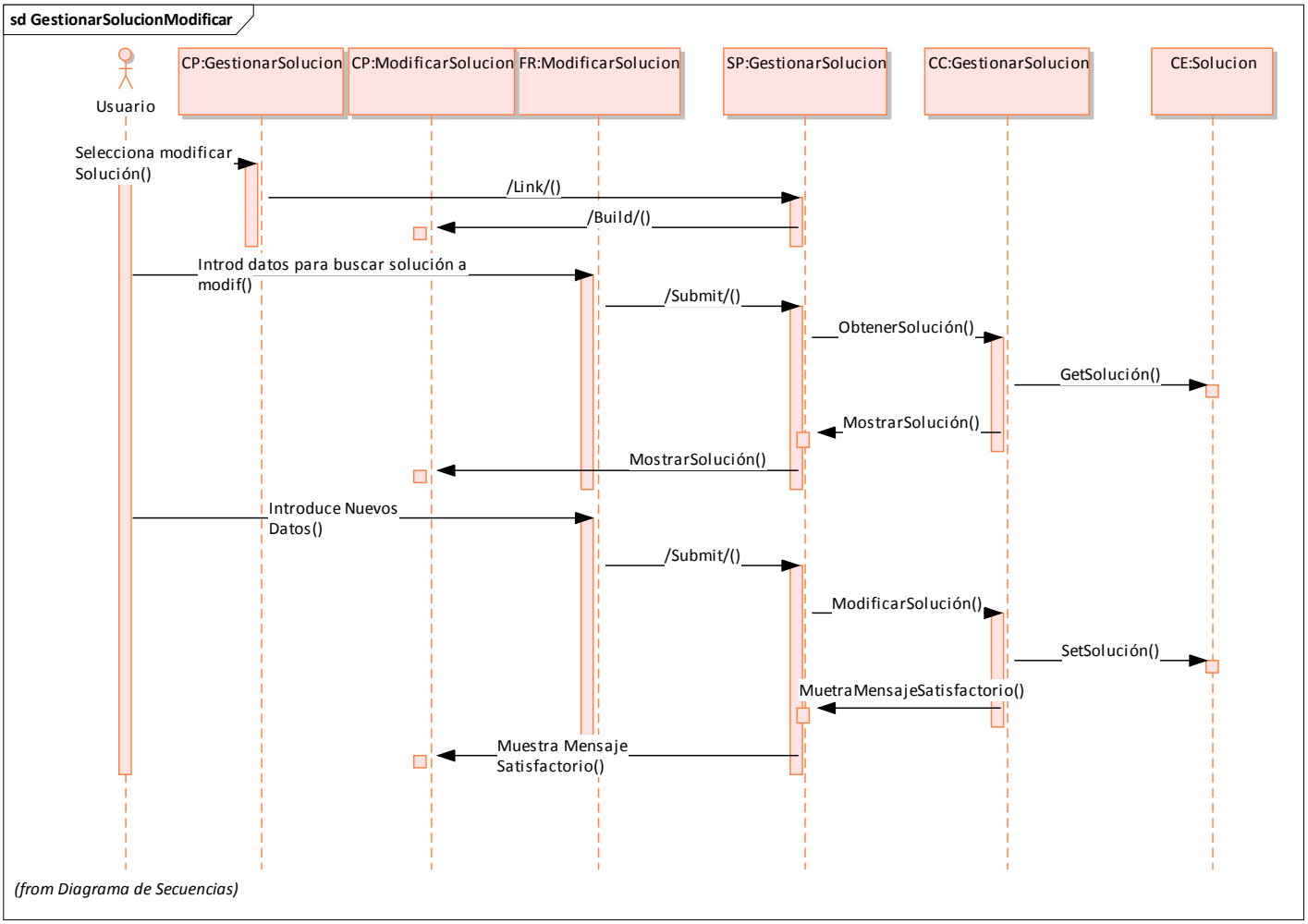

sd GestionarErrorModificar
Usuario
CP:GestionarError
CP:ModificarError
FR:ModificarError
JavaScript
SP:GestinarError
CC:ModificarError
CE:Error
Selecciona modificar error()
/Link/()
/Build/()
Introduce los datos a buscar error()
ValidarDatos()
/Submit/()
ObtenerError()
GetError()
MostrarError()
Mostrar Error()
IntroducirDatos()
/Submit/()
ModificarError()
SetError()
MostrarMensaje()
MostrarMensaje()
(from Diagrama de Secuencias)

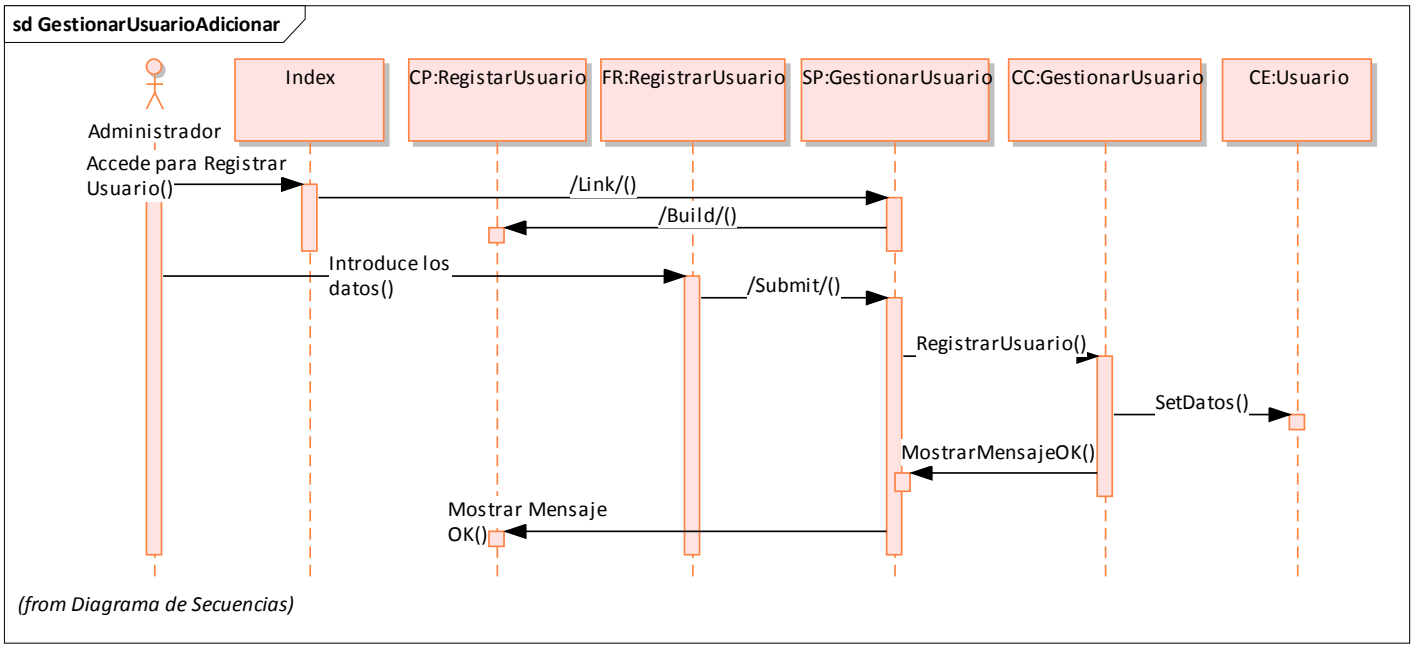

sd GestionarErrorAdicionar
Usuario
Index
CP:GestionarError
FR:AdicionarError
JavaScript
SP:GestionarError
CC:GestionarError
CE:AdicionarError
Accede para adicionar un error ()
/link/()
/build/()
Introduce datos del error()
ValidarDatos()
/Submit/()
AdicinarDatosError()
AdicionarDatos()
Muestra mensaje (ok)
MuestraMensaje()
(from Diagrama de Secuencias)

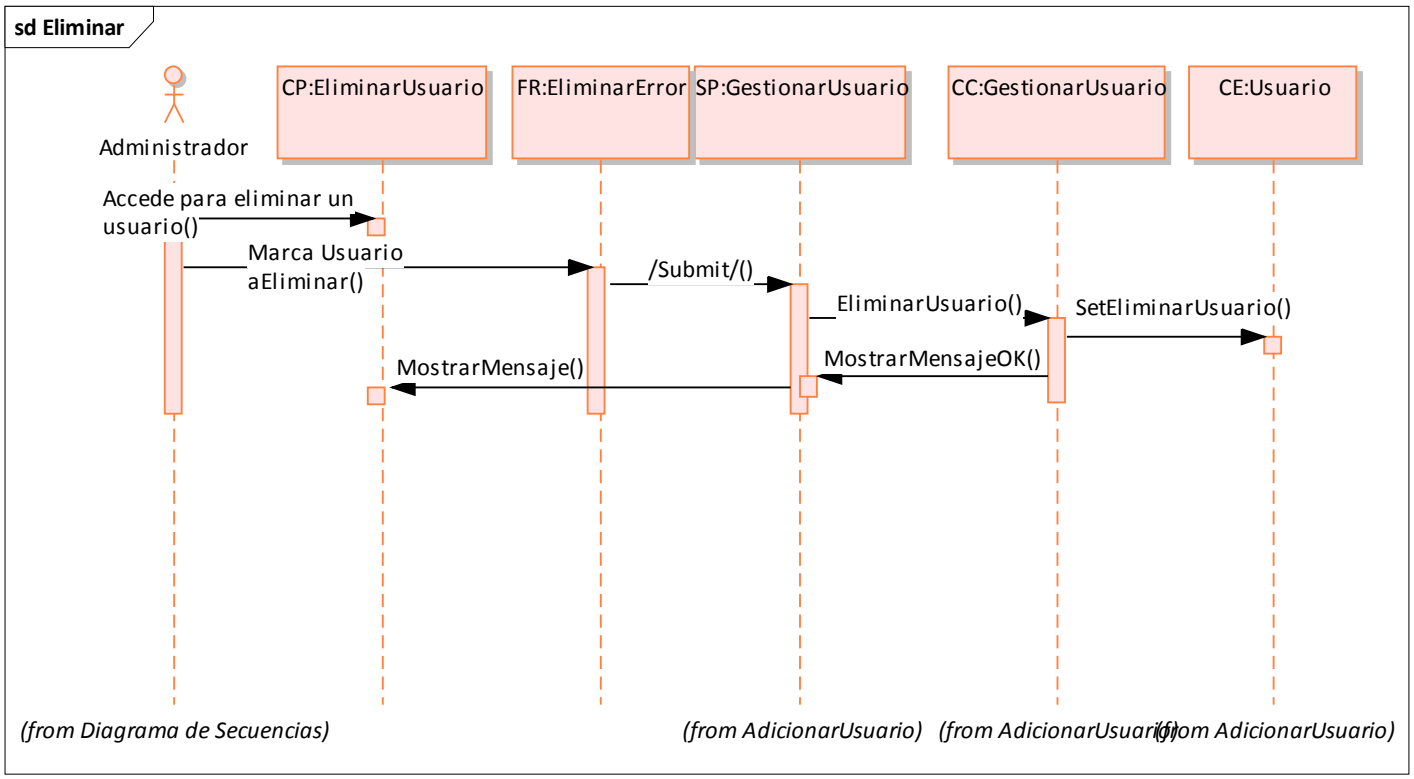

sd GestionarErrorEliminar
Administrador
CP:GestionarError
FR:EliminarError
SP:GestionarError
CC:GestionarError
CE:EliminarError
Accede para elimin un error y marca error()
submit()
EliminarError()
SetDatos()
MostrarMensaje(ok)
Mostrar mensaje(ok)
(from Diagrama de Secuencias)

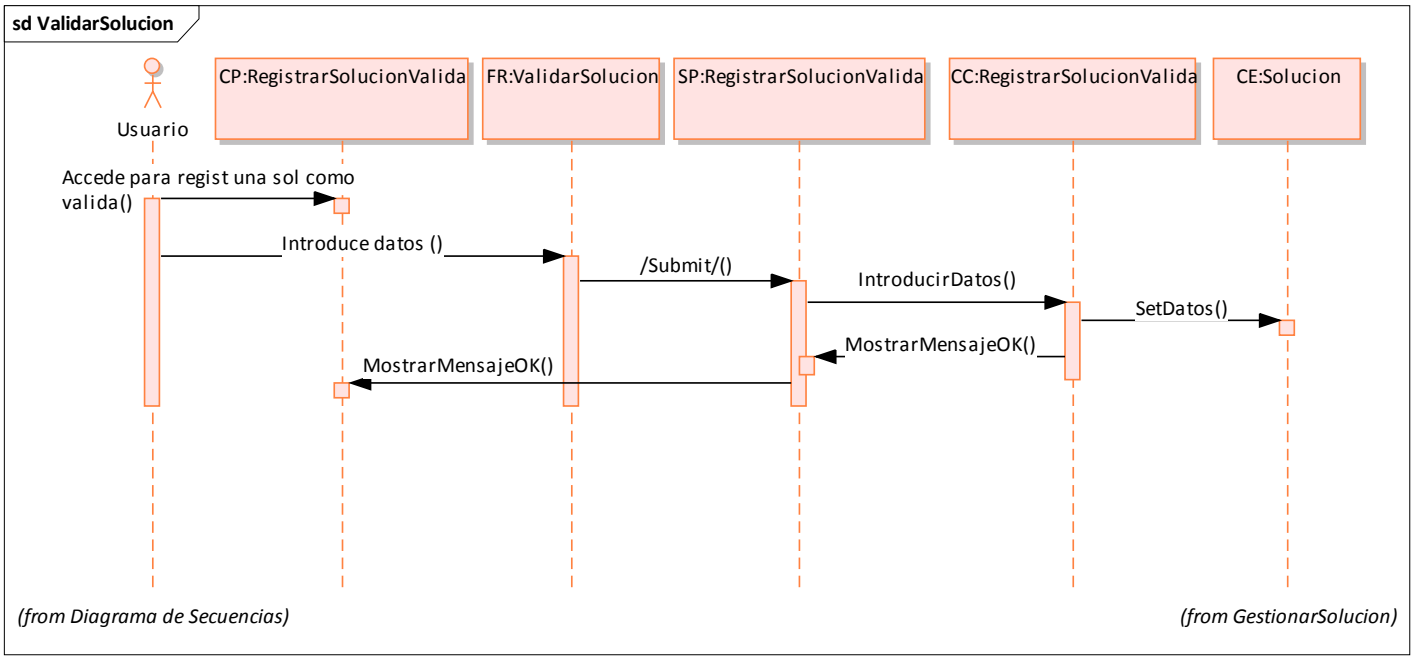

sd GestionarSolucionAdicionar
Usuario
Index
CP:GestionarSolucion
FR:Solucion
CP:JavaScript
SP:GestionarSolucion
CC:GestionarSolucion
CE:Solucion
Accede Para adicionar un error()
/Link/()
/Build/()
Introduce los datos de la solución()
ValidarDatos()
/Submit/()
AdicinarDatos()
AdicionarDatos()
MostrarMensaje(ok)
MostrarMensaje(ok)
(from Diagrama de Secuencia)


sd GestionarSolucionEliminar
Administrador
CP:GestionarSolucion
FR:EliminarSolucion
SP:GestionarSolucion
CC:GestionarSolucion
CE:Solucion
Accede para eliminar una Solución()
/Submit/()
EliminarSolución()
SetEliminarSolución()
MostrarMensajeOk()
MostrarMensajeOK()
(from Diagrama de Secuencias)


sd GestionarSolucionModificar
Usuario
CP:GestionarSolucion
CP:ModificarSolucion
FR:ModificarSolucion
SP:GestionarSolucion
CC:GestionarSolucion
CE:Solucion
Selecciona modificar Solución()
/Link/()
/Build/()
Introd datos para buscar solución a modif()
/Submit/()
ObtenerSolución()
GetSolución()
MostrarSolución()
MostrarSolución()
Introduce Nuevos Datos()
/Submit/()
ModificarSolución()
SetSolución()
MuetraMensajeSatisfactorio()
Muestra Mensaje Satisfactorio()
(from Diagrama de Secuencias)


sd GestionarUsuarioAdicionar
Administrador
Index
CP:RegistarUsuario
FR:RegistrarUsuario
SP:GestionarUsuario
CC:GestionarUsuario
CE:Usuario
Accede para Registrar Usuario()
/Link/()
/Build/()
Introduce los datos()
/Submit/()
RegistrarUsuario()
SetDatos()
MostrarMensajeOK()
Mostrar Mensaje OK()
(from Diagrama de Secuencias)


sd Eliminar
Administrador
CP:EliminarUsuario
FR:EliminarError
SP:GestionarUsuario
CC:GestionarUsuario
CE:Usuario
Accede para eliminar un usuario()
Marca Usuario aEliminar()
/Submit/()
EliminarUsuario()
SetEliminarUsuario()
MostrarMensajeOK()
MostrarMensaje()
(from Diagrama de Secuencias)
(from AdicionarUsuario)
(from AdicionarUsuario)
(from AdicionarUsuario)


sd ValidarSolucion
Usuario
CP:RegistrarSolucionValida
FR:ValidarSolucion
SP:RegistrarSolucionValida
CC:RegistrarSolucionValida
CE:Solucion
Accede para regist una sol como valida()
Introduce datos ()
/Submit/()
IntroducirDatos()
SetDatos()
MostrarMensajeOK()
MostrarMensajeOK()
(from Diagrama de Secuencias)
(from GestionarSolucion)

GLOSARIO DE TÉRMINOS