Universidad de las Ciencias Informáticas

Título: “JASCOG 1.0”.
Herramienta para la generación de código JavaScript para la librería ExtJS.

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

Autor: Ricardo Sosa Marín

Tutor: Ing. Yudier Cervantes Puga

JULIO 2009

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Autor: Ricardo Sosa Marín Tutor: Ing. Yudier Cervantes Puga

AGRADECIMIENTOS

Quisiera agradecer, a la Revolución y al Comandante en Jefe Fidel Castro Ruz, por darme la oportunidad de ser uno de los protagonistas de la Universidad de las Ciencias Informáticas, lugar en el que pase cinco años formándome como profesional.

A todos mis amigos, en especial a Pepe, George, Borroto y Tomás, por su compañía durante todos estos años, por compartir los buenos y malos momentos, por ser verdaderamente amigos que quedan para toda la vida.

A mi novia, que me ha brindado su amor incondicionalmente y me ha ayudado a pasar tanto tiempo lejos de mi familia.

A mi tutor, por brindarme su ayuda y sus conocimientos en todo momento.

A mi familia por confiar tanto en mí y llenarme de orgullo en cada frase de aliento y demostración de cariño.

Y de manera muy especial a mis padres por su dedicación, confianza y amor. Que son los principales intérpretes de mi vida, esforzándose por convertirme en un ser cada día mejor, por guiarme siempre por el camino correcto, por alentarme, darme seguridad ante todos los acontecimientos de mi vida y estar a mi lado en cada momento.

RESUMEN

Entre los factores claves en la producción de software se encuentran las herramientas de generación de código. Con ellas se disminuye el tiempo de desarrollo y las probabilidades de cometer errores. El funcionamiento de estas herramientas se basa en la obtención de un modelo y la transformación del mismo en código fuente útil en el desarrollo de una aplicación. Generalmente se utilizan como modelos, un diseño de base de datos, un diagrama de clases o un modelo creado con la misma herramienta. El uso más difundido de los generadores es en la creación de capas como la de acceso a datos o las interfaces. Este trabajo está orientado a la creación de una herramienta para la generación de la interfaz gráfica de usuario para aplicaciones Web que usen la librería AJAX ExtJS.

PALABRAS CLAVE

JavaScript, ExtJS, herramienta, generador.

TABLA DE CONTENIDO

INTRODUCCIÓN

Las compañías productoras de software del mundo tienen un objetivo común, “desarrollar software de gran calidad en el menor tiempo posible”. Algunas no han logrado alcanzar esta meta, debido a que la relación “tiempo - cantidad de trabajo” está en constante contradicción. En muchas ocasiones las demandas de los clientes son superiores a la capacidad productiva de las empresas, esto conlleva a que un producto determinado no se termine en el tiempo acordado, o no tenga la calidad requerida. La solución a este conflicto se ha buscado en todas la direcciones, tanto en el creación de nuevas tecnologías para agilizar el desarrollo como en la preparación de los recursos humanos.

En la actualidad existen infinidades de herramientas que facilitan el trabajo en cualquiera de las capas o fases de un proyecto, es un factor clave escoger la herramienta adecuada para desarrollar, para minimizar la cantidad de errores que se pueden cometer y garantizar un menor tiempo de desarrollo, debido a que la mayoría de estas herramientas constan de una interfaz gráfica amigable y generan el código que necesario a partir de un diagrama muy fácil de hacer.

La capacidad de generar código es una de las funcionalidades más codiciadas en la actualidad, existen varias herramientas que cuentan con esta característica, entre ellas se pueden encontrar Rational Rose 2003 Enterprise Edition, Enterprise Architect, Visual Paradigm, Dreamweaver, Quanta Plus, etc.

La Universidad de las Ciencias Informáticas (UCI) tiene como principal objetivo convertir a Cuba en una de las potencias productoras de software del mundo, hasta la fecha cuenta con 7 años de experiencia en los cuales se han desarrollado un gran número de proyectos nacionales e internacionales. Uno de estos fue el recién concluido Sistema de Gestión para el Convenio Integral de Cooperación Cuba-Venezuela (CCV).

Para el desarrollo de la interfaz de usuario de CCV, se utilizó la librería ExtJS 2.0, la cual es una librería construida con JavaScript que proporciona una interfaz a las famosas librerías de Yahoo!, jQuery y Prototype, su potencia radica en la rica colección de componentes para el diseño de interfaces gráficas del lado del cliente haciendo uso extensivo de AJAX.

A pesar de que la ExtJS cuenta con una excelente documentación, el desarrollo de CCV se vio afectado por varios factores, uno de los más significativos fue el largo tiempo que se necesitaba para escribir, revisar, y analizar el código necesario para desarrollar la interfaz de usuario sin ayuda de alguna herramienta que facilite esta tarea. Actualmente se está desarrollando el Sistema Nacional Público para el Seguimiento de Inversiones y Sectores (SINAPSIS), en el cual se utilizará la ExtJS para desarrollar la interfaz de usuario y se desea que el tiempo de implementación sea reducido.

Por tanto, surge la necesidad de dar solución a la situación anteriormente expuesta; y el **problema** a tratar consiste en ¿Cómo reducir el tiempo de desarrollo de la interfaz de usuario del proyecto SINAPSIS u otras aplicaciones Web que utilicen la librería ExtJS?

Hipótesis

Si se desarrolla una herramienta capaz de generar el código para la interfaz de usuario, se logrará disminuir el tiempo de desarrollo del proyecto SINAPSIS u otras aplicaciones Web que utilicen la librería ExtJS.

Objeto de Estudio

Proceso de producción y modelado de herramientas generadoras de código.

Campo de Acción

Generador de código para la interfaz de usuario del proyecto SINAPSIS u otras aplicaciones Web que utilicen la librería ExtJS.

Objetivo general

Desarrollar una herramienta capaz de generar el código para la interfaz de usuario del Proyecto SINAPSIS u otras aplicaciones Web que utilicen la librería ExtJS.

Objetivos específicos

- Realizar estudio bibliográfico sobre la generación de código y la librería ExtJS.
- Realizar el levantamiento de los requisitos funcionales y no funcionales para la herramienta.
- Diseñar la herramienta de generación de código.
- Desarrollar la herramienta de generación de código.

Tareas para cumplir el objetivo

- Estudio bibliográfico profundo sobre la generación de código y la librería ExtJS.
- Levantamiento de los requisitos para la herramienta.
- Diseño de la aplicación para generar el código de la interfaz de usuario.
- Implementación de la herramienta.
- Documentación del proceso de desarrollo de software.
- Realización de prueba piloto

Métodos Científicos

Métodos Teóricos

- **Histórico – Lógico:** Para la realización del estudio del arte en el que se analizaron las ventajas y dificultades que se derivan del empleo del empleo de herramientas generadoras de código.
- **Analítico – Sintético:** Para procesar la documentación estudiada durante la realización del estudio del arte y establecer las particularidades del desarrollo de la herramienta.

Métodos empíricos

- **Experimentación:** Para detectar posibles errores de implementación en la herramienta que impidan la correcta ejecución de las funcionalidades descritas.
- **Medición:** Para comparar los resultados de las funcionalidades implementadas en la herramienta con estándares de calidad que permitan la óptima ejecución de la misma.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se explican los métodos empleados para solucionar el problema planteado. Se fundamenta el uso de cada una de las tecnologías empleadas y se muestra una visión general de cuáles son las principales herramientas para la generación de código existentes en la actualidad.

1.2 Acerca de Internet y la Web

1.2.1 ¿Qué es internet?

Internet es una gran red de redes, también llamada Súper Carretera de la Información. Es el resultado de la interconexión de miles de computadoras de todo el mundo.

Comenzó en los Estados Unidos de América en 1969, como un proyecto militar. La Agencia de Proyectos de Investigación Avanzados de Defensa (DARPA) desarrolló una red de computadoras llamada ARPANET, para no centralizar los datos, lo cual permitía que cada estación de la red pudiera comunicarse con cualquier otra por varios caminos diferentes, además presentaba una solución para cuando ocurrieran fallas técnicas que pudieran hacer que la red dejase de funcionar. (1)

Los sitios originales que se pusieron en la red eran bases militares, universidades y compañías con contratos del Departamento de Defensa. Conforme creció el tamaño de esta red experimental, lo mismo sucedió con las precauciones por la seguridad. Las mismas redes usadas por las compañías y las universidades para contratos militares se estaban volviendo cada vez más accesibles al público. (1)

Como resultado, en 1984, ARPANET se dividió en dos redes separadas pero interconectadas. El lado militar fue llamado MILNET. El lado educativo todavía era llamado técnicamente ARPANET, pero cada vez se hizo más conocida como Internet. (1)

En agosto de 1993, existían más de 14 000 redes conectadas con Internet y según las estadísticas de aquel momento, se incorporaban 1000 nuevas redes como promedio mensualmente. En enero de 1995, la cifra total había aumentado a 26 274 redes. Habían conectados 84 países con acceso directo a esta red y 48 millones de usuarios. En 1995, Internet alcanzaba a 148 de los 185 países miembros de las Naciones

Unidas (86%) en comparación con los 73 de los 159 países miembros (46%) en 1991. Las aplicaciones disponibles cubren la mayoría de los sectores de la vida de la sociedad. (1)

Se puede decir que el resultado final de lo que comenzó como un proyecto de investigación gubernamental y educativo ahora se ha convertido en uno de los medios de comunicación más importante de la actualidad. Gracias a internet, millones de personas tienen acceso fácil e inmediato a una cantidad extensa y diversa de información.

1.2.2 ¿Qué es la Web?

La Web es una idea que se construyó sobre la Internet, pero introduce una serie de ideas nuevas, heredando las existentes. Empezó a principios de 1990, en Suiza en el Centro de Estudios para la Investigación Nuclear (CERN) y la idea fue de Tim Berners Lee. (2)

Antes de que existiera la Web, la manera de obtener los datos por la Internet era caótica, había un sinfín de maneras posibles y con ello había que conocer múltiples programas y sistemas operativos. La Web introduce un concepto fundamental, la posibilidad de lectura universal, que consiste en que una vez que la información esté disponible, se pueda acceder a ella desde cualquier ordenador, desde cualquier país, por cualquier persona autorizada, usando un único y simple programa. Para que esto fuese posible, surgen los navegadores Web. (2)

1.2.3 ¿Qué es un navegador Web?

Los navegadores Web o en ingles “Web Browsers”, no son más que aplicaciones software que permiten al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores Web instalados alrededor del mundo a través de Internet. Cualquier navegador actual permite mostrar o ejecutar gráficos, secuencias de vídeo, sonido, animaciones y programas diversos además del texto y los hipervínculos. (3)

El primer navegador, desarrollado en el CERN a finales de 1990 y principios de 1991 por Tim Berners-Lee, era bastante sofisticado y gráfico, pero sólo funcionaba en estaciones NeXT. El navegador Mosaic, que funcionaba inicialmente en entornos UNIX sobre X11, fue el primero que se extendió debido a que pronto el Centro Nacional de Aplicaciones para Supercomputadoras (NCSA) preparó versiones para Windows y Macintosh. Sin embargo, poco más tarde entró en el mercado Netscape Navigator que

rápidamente superó en capacidades y velocidad a Mosaic. Este navegador tiene la ventaja de funcionar en la mayoría de los UNIX, así como en entornos Windows. (3)

Internet Explorer fue la apuesta tardía de Microsoft para entrar en el mercado y hoy en día ha conseguido desbancar al Netscape Navigator entre los usuarios de Windows. En los últimos años se ha vivido una auténtica explosión del número de navegadores, que ofrecen cada vez mayor integración con el entorno de ventanas en el que se ejecutan. Netscape Communications Corporation liberó el código fuente de su navegador, naciendo así el proyecto Mozilla. (3)

Actualmente el navegador más utilizado en el mundo es Internet Explorer en todas sus versiones ocupando un 65% del mercado, algunas empresas indican que esta ventaja se debe a que viene integrado con Windows, detrás de éste está el navegador de Mozilla Firefox usando un 32% del mercado, el cual se está popularizando cada vez más. Firefox es un competidor serio al producto de Microsoft. Existen también los navegadores Safari, Netscape Navigator, Opera y Chrome los cuales tienen un uso de menos del 2% en el mercado. (3)

1.2.4 Lenguajes de programación para la Web

Hoy en día existen diferentes lenguajes de programación para desarrollar en la Web, estos han ido surgiendo debido a las tendencias y necesidades de las plataformas. Desde los inicios de Internet, fueron surgiendo diferentes demandas por los usuarios y se dieron soluciones mediante lenguajes estáticos. A medida que paso el tiempo, las tecnologías fueron desarrollándose y surgieron nuevos problemas a solucionar. Esto dió lugar a desarrollar lenguajes de programación dinámicos para la Web, que permitieran interactuar con los usuarios y utilizaran sistemas de Bases de Datos.

Entre los lenguajes de programación Web se pueden encontrar:

- **HTML:** Desde el surgimiento de Internet se han publicado sitios web gracias al lenguaje HTML. Es un lenguaje estático para el desarrollo de sitios web (acrónimo en inglés de HyperText Markup Language, en español Lenguaje de Marcas Hipertextuales). Desarrollado por el World Wide Web Consortium (W3C). (4)
- **JavaScript:** Este es un lenguaje interpretado, no requiere compilación. Fue creado por Brendan Eich en la empresa Netscape Communications. Utilizado principalmente en páginas Web. Es similar a Java, aunque no es un lenguaje orientado a objetos, el mismo no dispone de herencias.

La mayoría de los navegadores en sus últimas versiones interpretan código JavaScript. El código JavaScript puede ser integrado dentro de nuestras páginas Web. (4)

- **CSS:** Lenguaje de hojas de estilos en cascada, creado en 1995 por Hakon Wium Lie y Bert Bos para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML, es la mejor forma de separar los contenidos y su presentación, es imprescindible para crear páginas Web complejas. (5)
- **PHP:** Es un lenguaje de programación utilizado para la creación de sitios Web. PHP es un acrónimo recursivo que significa "Hypertext Pre-processor", inicialmente se llamó "Personal Home Page". Surgió en 1995, desarrollado por PHP Group. Este es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas Web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. (4)
- **ASP:** Es una tecnología del lado de servidor desarrollada por Microsoft para el desarrollo de sitios Web dinámicos. ASP significa en inglés "Active Server Pages", fue liberado por Microsoft en 1996. ASP no necesita ser compilado para ejecutarse y puede ser insertado junto con el código HTML. (4)
- **JSP:** Es un lenguaje para la creación de sitios Web dinámicos, acrónimo de Java Server Pages. Está orientado a desarrollar páginas Web en Java. JSP es un lenguaje multiplataforma creado para ejecutarse del lado del servidor, fue desarrollado por Sun Microsystems para la creación de aplicaciones Web potentes. Posee un motor de páginas basado en los servlets de Java. (4)
- **Python:** Es un lenguaje de programación creado en 1990 por Guido van Rossum, es el sucesor del lenguaje de programación ABC. Los usuarios consideran a Python como un lenguaje más limpio para programar. Permite la creación de diversos tipos de programas incluyendo los sitios web. Su código no necesita ser compilado, por lo que se llama que el código es interpretado. (4)
- **Ruby:** Es un lenguaje interpretado de muy alto nivel y orientado a objetos. Desarrollado en el 1993 por el programador japonés Yukihiro Matz Matsumoto. Su sintaxis está inspirada en Phyton, Perl. Es distribuido bajo licencia de software libre (Opensource). Ruby es un lenguaje dinámico para una programación orientada a objetos rápida y sencilla. (4)

A medida que la Web fue desarrollándose, fueron surgiendo nuevas demandas por los usuarios, no era suficiente con visualizar la información que estaba publicada en internet, sino que esta información tenía que mostrarse lo más rápido posible.

En las aplicaciones web tradicionales, las acciones del usuario en la página desencadenan llamadas al servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador del usuario. Esta técnica tradicional para crear aplicaciones Web funciona correctamente, pero no crea una buena sensación al usuario. Al realizar peticiones continuas al servidor, el usuario debe esperar a que se recargue la página con los cambios solicitados. Si la aplicación debe realizar peticiones continuas, la aplicación Web se convierte en algo más molesto que útil. Dada esta nueva problemática surge el término AJAX. (6)

1.2 5 ¿Qué es AJAX?

AJAX se acuñó por primera vez en el artículo "Ajax: A New Approach to Web Applications", en español "Ajax: Un nuevo enfoque para aplicaciones Web" publicado por Jesse James Garrett el 18 de Febrero de 2005. Hasta ese momento, no existía un término normalizado que hiciera referencia a un nuevo tipo de aplicación Web que estaba apareciendo. En realidad, AJAX es un acrónimo de Asynchronous JavaScript + XML, que se puede traducir como "JavaScript asíncrono + XML". El artículo define AJAX de la siguiente forma: (6)

"Ajax no es una tecnología en sí mismo. En realidad, se trata de la unión de varias tecnologías que se desarrollan de forma autónoma y que se unen de formas nuevas y sorprendentes." (6)

Tecnologías que forman AJAX

- **HTML y CSS** para el diseño que acompaña a la información.
- **DOM**, para la interacción y manipulación dinámica de la presentación.
- **XML**, **XSLT y JSON**, para el intercambio y la manipulación de información.
- **XMLHttpRequest**, para el intercambio asíncrono de información.
- **JavaScript**, para unir las demás tecnologías.

Gracias a AJAX, quedaron atrás los antiguos formularios cuyas páginas debían ser recargadas para el envío. Atrás quedaron también los interminables listados de información que nuestro navegador bajaba, información de la cual tal vez sólo se necesitaba una pequeña parte. AJAX hace que la información sea cargada de manera precisa, fácil y ágil.

Las aplicaciones AJAX se ejecutan en la máquina cliente, manipulando la página actual dentro de sus navegadores usando métodos de Document Object Model (DOM). Puede ser usado para multitud de tareas como actualizar o eliminar registros, expandir formularios Web, devolver peticiones simples de búsqueda, o editar árboles de categorías; sin tener la necesidad de recargar la página de HTML cada vez que se realiza un cambio. Generalmente sólo requiere enviar pequeñas peticiones al servidor, y se devuelven respuestas relativamente cortas. (6)

1.2.6 Librerías AJAX

Dado el desarrollo de las aplicaciones Web dinámicas con el uso de AJAX, han surgido un sin número de librerías, las cuales agrupan funcionalidades para hacer mucho más fácil este tipo de programación, entre ellas se pueden encontrar: (7)

Prototype	ATLAS	moo.fx
AHAH	Bajax	Rico
Dojo	MochiKit	script.aculo.us
AjaxAC	DynAPI	Tacos
JSAN	Qooxdoo	Yahoo! User Interface
Ajax.NET Professional	Taconite	ExtJS
AjaxRequest Library	jQuery	mootools
qForms JavaScript API	Zapatec AJAX Suite	zk

1.3 Acerca de los generadores de código

1.3.1 ¿Qué es un generador de código?

La generación de código data desde la existencia de los primeros compiladores. Hasta la aparición de los primeros generadores de código comerciales; la generación de código era exclusividad de programas compiladores especializados. En tiempos más recientes la generación de código, gracias al avance de la ingeniería del software, se ha llevado a un nivel diferente; donde se encuentran programas generadores de pantallas, reportes y consultas.

Un generador de código es una herramienta capaz de generar código de forma automática. Estas herramientas realizan el trabajo que a los programadores les tomaría mucho más tiempo lograr de forma manual. Con ellas se disminuye el tiempo de desarrollo y las posibilidades de cometer errores. El funcionamiento de estas herramientas se basa en la obtención de un modelo y la transformación del mismo en código fuente útil en el desarrollo de una aplicación. Generalmente se utilizan como modelos, un diseño de base de datos, un diagrama de clases o un modelo creado con la misma herramienta. (8)

1.3.2 Tipos de generadores de código

Según su interacción con el código generado se clasifican en:

- **Activos:** Son aquellos que permiten generar varias veces sobre el mismo código generado a partir de cambios en la entrada. Estos generadores definen espacios de código seguros donde el programador puede hacer los cambios que desee sin que éstos se pierdan en las sucesivas generaciones de código. (8)
- **Pasivos:** Generan el código una vez y no vuelven a tener interacción con él. Tienen la desventaja de que si se corrige un error en los mecanismos de generación o se cambia el diseño y se vuelve a generar se pierde lo que se codificó manualmente. (8)

Según la aproximación que usan para generar el código se clasifican en:

- **Estructural:** Generan bloques de código, desde modelos estáticos y relaciones entre objetos. Las primitivas de trabajo en estos modelos son clases, atributos, tipos y asociaciones. Algunas herramientas usan un motor de traducción y plantillas preexistente para especificar correspondencias con un código fuente en particular. La generación de código estructural es

incompleta pero ahorra esfuerzo de codificación manual y proporciona un marco de trabajo inicial consistente con los modelos. (9)

- **De Comportamiento:** Generan código completo a partir de modelos de máquinas de estados y la especificación de acciones en un lenguaje de alto nivel. Algunos métodos que modelan comportamiento con máquinas de estados, añaden código (como C++ o un lenguaje propietario) para representar las acciones que ocurren durante la transición de estado. Junto con modelos de estructuras de objetos y mecanismos de comunicación, esta técnica permite a las herramientas generar código para el modelo completo de la aplicación. Un beneficio de esta técnica es la capacidad para simular y verificar el comportamiento del sistema basado en modelos antes de que el código sea generado. Los lenguajes empleados incluyen C++, C e incluso ensamblador. (8)
- **Traductivos:** Se basan en que los modelos de aplicación y de arquitectura son independientes uno del otro. Un modelo de aplicación completo, con estructura de objetos, comportamiento y comunicaciones es creado usando el método de Análisis Orientado a Objetos. Un modelo de arquitectura (un conjunto de patrones llamados plantillas o arquetipos) es desarrollado con una herramienta que soporte esta aproximación. Entonces, un motor de traducción genera el código para la aplicación de acuerdo con las reglas de correspondencia en la arquitectura. Las aproximaciones traductivas ofrecen una reutilización significativa debido a que la aplicación y el modelo de arquitectura son independientes. (8)

1.3.3 Funcionamiento de los generadores de código

Los generadores implementan las siguientes fases:

- **Carga:** La fase de carga consiste en la lectura desde un repositorio (puede ser un fichero binario, XML, una base de datos, o un diagrama UML) del modelo a traducir y crear una representación de éste en memoria. La representación de este modelo en memoria, no tiene porqué ser completa, ni tampoco seguir la misma estructura del modelo. Al contrario, la carga y las estructuras pueden ser adaptadas para cargar sólo la información necesaria y disponerla del modo que sea más conveniente para la tarea de traducción a realizar. (9)
- **Inferencia:** Si se han definido una serie de mecanismos de inferencia, que completan la información de modelado, éstos se ejecutan. Las estructuras del modelo en memoria son completadas y extendidas. Es necesario llevar a cabo este proceso antes de proceder a la

generación propiamente dicha. El proceso es responsable de realizar pre cálculos útiles y de disponer adecuadamente la información para la fase posterior. (9)

- **Generación:** La última fase es la de generación. En función del código destino a producir, se recorren secuencialmente, y de modo anidado, los elementos de modelo en sucesivas pasadas. Por ejemplo: para cada clase, para cada servicio y para cada argumento. Como resultado de esta fase se obtiene el código generado. (9)

1.3.4 Técnicas de generación de código

Se reconocen cuatro técnicas generales utilizadas por los generadores de códigos:

- **Clonación:** Si el código destino a producir contiene ficheros que permanecen constantes independientemente del modelo a traducir, la traducción puede llevarse a cabo por medio de clonación, o copia directa del fichero origen al directorio de generación. Las librerías de funciones genéricas o ficheros binarios representando imágenes o iconos constantes pueden ser tratados de este modo. (9)
- **Concatenación de cadenas:** La concatenación de cadenas es un modo sencillo de ir construyendo el código destino desde un lenguaje de programación clásico. El nombre proviene del proceso de ir concatenando cadenas de texto que contienen el código a producir. Finalmente, la cadena es volcada a un fichero. Se dispone de la potencia del lenguaje de programación para determinar que código debe ser generado, permitiendo cálculos, sustituciones y procesados complejos. (9)
- **Plantillas:** La generación mediante plantillas puede ser empleada cuando el código destino a producir tiene un patrón de repetición bien caracterizado, de modo que los ficheros generados son idénticos salvo los datos procedentes directamente del modelo a generar. En este caso puede definirse una plantilla genérica a partir de ejemplares del código objetivo. En esta plantilla, las dependencias del modelo son sustituidas por marcadores con nombre único. Aparejado a la plantilla, se puede definir un proceso de generación que, dado un elemento del modelo, la plantilla sea instanciada a código mediante un proceso de sustitución de cadenas: los marcadores son sustituidos por los datos del modelo correspondiente. (9)
- **Análisis de expresiones mediante gramáticas:** Existen ocasiones donde las estrategias previas de generación se vuelven insuficientes. En particular, un caso muy claro es el del tratamiento de expresiones que siguen una gramática dada. Aquí las técnicas de compilación clásicas son las

más adecuadas para producir el código necesario. La expresión puede ser convertida en una estructura con forma arbórea en memoria: Un analizador léxico, sintáctico y semántico realizan este trabajo. Después, un algoritmo puede recorrer dicho árbol que representa la expresión para analizarla y decidir el tipo de código a producir. (9)

1.3.5 Ventajas de la generación de código

- **Calidad:** La calidad del código generado está en correspondencia con la calidad de las plantillas y del proceso que se usa para la generación. A medida que son detectados errores y es mejorado el código de las plantillas la calidad del código generado aumenta. Se pueden imponer reglas de estilo al código generado para aumentar su homogeneidad y legibilidad. Se puede generar la documentación del código así como comentarios que faciliten su mantenimiento. (9)
- **Consistencia:** El código generado es extremadamente consistente. El nombre de las variables, métodos y clases es formado de la misma forma en el código. La relación entre el modelo y el código generado permite que también haya consistencia entre la documentación y el código, la cual es muy difícil de mantener en un proceso de desarrollo tradicional. (9)
- **Productividad:** Es fácil reconocer los beneficios de la generación de código respecto a la productividad. Se comienza con un diseño de entrada e instantáneamente se obtiene una implementación de salida que responde al diseño. Por otra parte estos beneficios son mucho más apreciados cuando se genera nuevamente el código debido a un cambio en el diseño y no se pierde el trabajo realizado. Por otra parte libera a los programadores del trabajo tedioso y repetitivo permitiéndoles enfocarse en los aspectos que sí requieren de su creatividad e inteligencia. (9)
- **Abstracción:** Muchos generadores construyen el código basados en modelos abstractos. Por ejemplo, se puede generar una capa de acceso a datos, a través de un XML que represente las tablas, los atributos y sus relaciones. Entre las ventajas que esto brinda está la portabilidad a distintas plataformas porque elevando el nivel de abstracción no se cae en especificaciones únicas de una plataforma, sino que permite centrarse en el modelado de los datos o del negocio. (9)

1.3.6 Desventajas de la generación de código

- **Esfuerzo extra en educación:** Se necesita educar a los desarrolladores en las ventajas que ofrece el generador y de qué forma deben emplearlo. (8)

- **Mantenimiento:** Cuando se usa un generador hay que darle mantenimiento constantemente. Si es desarrollado por terceros se tiene que estar al tanto de las versiones nuevas que salen y de los errores que se le van detectando. En cualquier caso hay que dedicarle esfuerzos a resolver los errores que pueda traer el generador y a agregarle las nuevas funcionalidades que van surgiendo en el mercado. Si es un generador poco usado se corre además el riesgo de que sus desarrolladores dejen de darle soporte y que por tanto en poco tiempo se vuelva obsoleto. (8)
- **Dominios reducidos:** La generación de código tradicionalmente se ha aplicado a dominios muy específicos y bien conocidos donde es posible anticiparse a la variabilidad de problemas que pueden encontrarse en ese dominio. Los dominios o áreas de aplicación demasiado grandes o fuera de ámbito no se benefician de la aplicación de técnicas de generación de código. (8)
- **Resistencia por parte de los desarrolladores:** Hay programadores convencionales que ven la generación de código como una amenaza para su trabajo. Ante lo cual, toman una postura defensiva o de rechazo. Otros afirman que el uso de generadores va contra las buenas prácticas de diseño y critican mucho la idea de copiar y pegar sobre plantillas. (8)

1.4 Herramientas de generación de código existente

Actualmente existe una gran diversidad de herramientas que generan código, estas se han desarrollado orientadas a cualquiera de las capas o fases de un proyecto.

1.4.1 Dreamweaver

Es el editor HTML visual más popular del mercado. Desde la aparición de su primera versión hasta la actual, han surgido novedosas herramientas para el diseño Web, tales como ColdFusion, ASP y PHP. Dreamweaver ha absorbido estas nuevas tecnologías permitiendo, tanto el trabajo con ellas por separado como en combinación, haciendo posible el paso del HTML estático a las aplicaciones Web dinámicas. (10)

1.4.2 Quanta Plus

Es una aplicación gratuita y de código abierto para GNU/Linux que nos permitirá desarrollar nuestra web de forma fácil y sencilla, gracias a su versatilidad y a su facilidad de uso. Permite trabajar tanto en modo visual como en modo editor de código, poniendo a nuestra disposición las herramientas necesarias para trabajar en cada uno de ellos. (11)

1.4.3 CodeCharge Studio

Es un potente entorno de desarrollo que proporciona a los programadores un efectivo, productivo y veloz medio para la creación de sitios Web interactivos y aplicaciones con amplio soporte para bases de datos y servidores Web. Cuenta con características de entorno de desarrollo integrado (IDE) y variadas funciones para la construcción de aplicaciones y la administración de la seguridad del sitio Web. CodeCharge Studio es capaz de generar automáticamente el código fuente de las aplicaciones Web para un mejor y más rápido desempeño, trabajando con lenguajes como Java Servlets, JSP, ASP.NET (C#), ASP, PHP, ColdFusion y Perl. (12)

1.4.4 TierDeveloper

Es una herramienta propietaria que permite la creación de la capa de acceso a datos para tecnología .NET. Soporta cuatro gestores de base de datos (SQL Server, Oracle, IBM Db2, Microsoft Access). Permite obtener un diagrama de objetos a partir de la estructura de la base de datos. Permite definir un diagrama propio de objeto y cómo se almacenará cada clase en la base de datos. Genera una clase por cada tabla de la base de datos y una clase para manejar la capa de acceso a datos. Genera una interfaz WEB para comprobar que funcionan las clases generadas. (13)

1.4.5 Enterprise Architect

Enterprise Architect es una herramienta comprensible de diseño y análisis UML, que cubre el desarrollo de software desde la captura de requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. Enterprise Architect es una herramienta de multi-usuarios, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Además, ofrece salida de documentación flexible y de alta calidad. (14)

1.4 6 Embarcadero ER/Studio

Es una herramienta para la arquitectura y el modelado de bases de datos que provee el descubrimiento, documentación y reutilización de los activos de información a través de modelos visuales. Direcciona las necesidades diarias de los administradores de bases de datos, desarrolladores y arquitectos de datos que construyen y mantienen aplicaciones de bases de datos grandes y complejas. ER/Studio soporta el muy popular SQL y Bases de Datos de escritorio, incluyendo Oracle, Sybase System, Microsoft SQL Server, IBM® DB/2 Universal Database, Open Systems, Informix, InterBase, Microsoft Access, Microsoft Visual FoxPro. (15)

1.5 Patrones de Diseño

Con el desarrollo del diseño orientado a objetos, han surgido ciertos tipos de problemas, los cuales se encuentran en repetidas ocasiones. Esta situación es la que ha obligado a los implicados en la tarea de diseñar aplicaciones, a buscar soluciones que resuelvan cada uno de esos problemas. Este par Problema – Solución, aplicado al diseño de aplicaciones software, es lo que se llama patrón de diseño. (16)

Los patrones de diseño proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifica y describe formas de solucionar problemas que ocurren de forma frecuente en el desarrollo de software. Por tanto está basado en la recopilación del conocimiento de los expertos en desarrollo de software. No debe verse los patrones de diseño como una teoría o una corriente. Es una experiencia real, probada y que funciona. Es Historia y nos ayuda a no cometer los mismos errores.

Algunos de los patrones de diseño utilizados actualmente para el desarrollo de software son:

1.5.1 Singleton

El patrón de diseño singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.

El patrón singleton se implementa creando en la clase un método que crea una instancia del objeto sólo si todavía no existe alguna. Para asegurar que la clase no puede ser instanciada nuevamente se regula la construcción de los objetos. En muchos lenguajes esto se logra restringiendo el alcance del constructor a través de atributos como protegido o privado. (17)

El patrón singleton provee una única instancia global gracias a que: (17)

- La propia clase es responsable de crear la única instancia.
- Permite el acceso global a dicha instancia mediante un método de clase.
- Declara el constructor de clase como privado para que no sea instanciable directamente.

1.5.2 GRASP

En Diseño orientado a objetos, GRASP son patrones generales de Software para asignación de responsabilidades, es el acrónimo de "General Responsibility Assignment Software Patterns", en español

“Patrones de Software de Asignación de Responsabilidades Generales”. Aunque se considera que más que patrones propiamente dichos, son una serie de "Buenas Prácticas" de aplicación recomendable en el diseño de software. (17)

1.5.2.1 Experto en información

El patrón Experto en información es el principio básico de asignación de responsabilidades. Nos indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce la información necesaria para crearlo. (17)

1.5.2.2 Creador

El patrón creador nos ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que: (17)

- Tiene la información necesaria para realizar la creación del objeto, o
- Usa directamente las instancias creadas del objeto, o
- Almacena o maneja varias instancias de la clase

1.5.2.3 Controlador

El patrón controlador sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. (17)

1.5.2.4 Alta cohesión

Nos dice que la información que almacena una clase debe de ser coherente y está en la mayor medida de lo posible relacionada con la clase. (17)

1.5.2.5 Bajo acoplamiento

Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. (17)

1.5.2.6 Polimorfismo

Siempre que se tenga que llevar a cabo una responsabilidad que dependa del tipo, se tiene que hacer uso del polimorfismo. (17)

1.5.2.7 Fabricación Pura

La fabricación pura se da en las clases que no representan un ente u objeto real del dominio del problema, si no que se ha creado intencionadamente para disminuir el acoplamiento, aumentar la cohesión y/o potenciar la reutilización del código. Es decir que es una clase "inventada" o que no existe en el problema como tal, pero que añadiéndola se logra mejorar estructuralmente el sistema. (17)

1.6 Acerca de las metodologías de desarrollo de software

El desarrollo de software es sin dudas una tarea difícil. Como resultado a este problema ha surgido una alternativa desde hace mucho: la Metodología. Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar inspirado por otras disciplinas de la ingeniería.

Hoy en día existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Un ejemplo de ellas son las propuestas tradicionales centradas específicamente en el control del proceso. Estas han demostrado ser efectivas y necesarias en un gran número de proyectos, principalmente aquellos proyectos de gran tamaño respecto a tiempo y recursos. Sin embargo la experiencia ha demostrado que las metodologías tradicionales no ofrecen una buena solución para proyectos donde el entorno es volátil y donde los requisitos no se conocen con exactitud, porque no están pensadas para trabajar con incertidumbre.

Aplicar metodologías tradicionales nos obliga a forzar a nuestro cliente a que tome la mayoría de las decisiones al principio. Luego el costo de cambio de una decisión tomada puede llegar a ser muy elevado si se aplican metodologías tradicionales.

Es por ello que varios problemas como los que a continuación se mencionan han sido detectados:

- Retrasos en la planificación
- Sistemas deteriorados
- Defectos
- Requisitos mal comprendidos
- Cambios de negocio
- Falsa riqueza
- Cambios de personal

Como respuesta a los problemas aplicando metodologías tradicionales surgieron otras metodologías que tratan de adaptarse a la realidad del desarrollo de software, estas se han categorizado en dos grandes grupos: las ágiles y las pesadas. Entre las metodologías ágiles están XP (Extreme Programming), FDD (Feature Driven Development), DSDM (Dynamic Systems Development Method), AUP (Agile Unified Process), Scrum, Crystal, Adaptive Software Development y otras. Así mismo se puede mencionar RUP (Rational Unified Process) como metodología pesada. (18) (19) (20)

Entre las metodologías más usadas actualmente están:

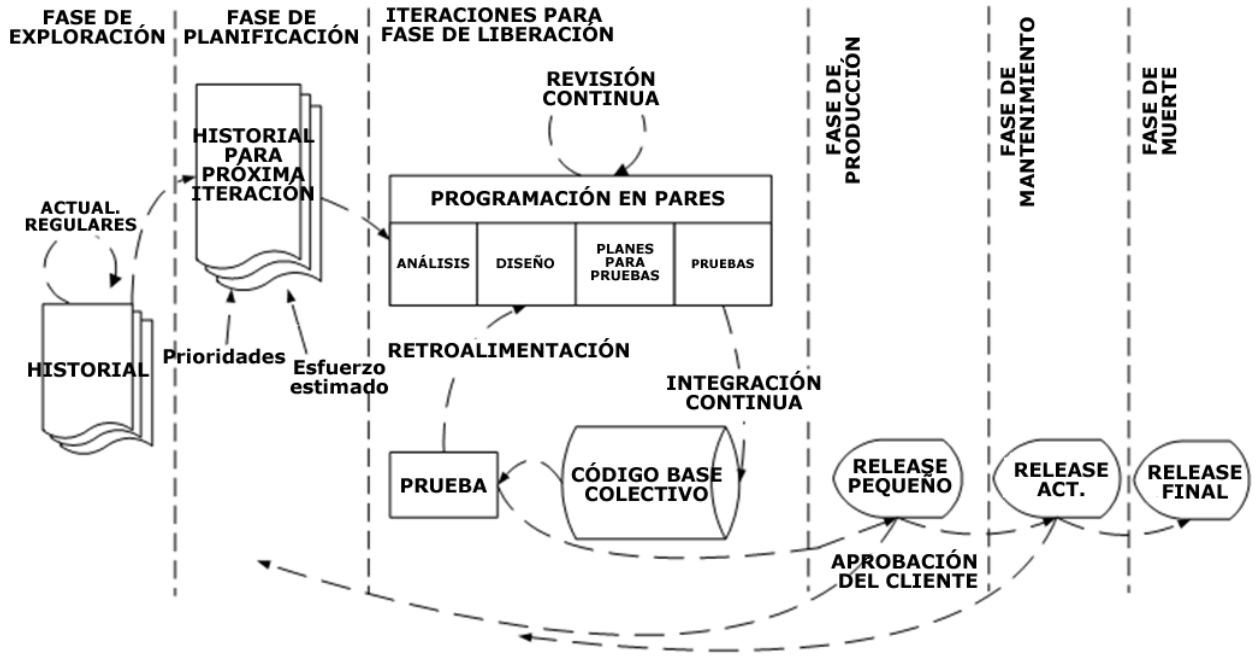
1.6.1 Metodología XP

La Programación Extrema (XP) surge ideada por Kent Beck, como proceso de creación de software diferente al convencional. En palabras de Beck: "XP es una metodología ligera, eficiente, con bajo riesgo, flexible, predecible y divertida para desarrollar software". XP

XP es una de las metodologías ágiles para el desarrollo de software más exitosas de la actualidad. Se utiliza en proyectos con pequeños equipos de desarrollo y con corto plazo de entrega. Se basa en la retroalimentación entre el cliente y el equipo de desarrollo, buena comunicación entre los participantes y simplicidad en las soluciones implementadas. Consiste en una programación rápida, cuya particularidad es que tiene como miembro del equipo al usuario final. Es adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (21)

En XP se sigue la idea de la programación en pares, dado las ventajas que ofrece respecto a la creación del código, pues se pueden evitar errores y malos diseños al controlar cada línea de código y decisión de diseño instantáneamente. La interacción entre ambos desarrolladores puede generar discusiones que lleven a mejores estructuras y algoritmos, aumentando la calidad del software. (22) (23)

Ciclo de vida del proceso XP



1.6.2 Metodología RUP

El proceso unificado de desarrollo (RUP) es una metodología para la ingeniería de software que va más allá del mero análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software. RUP es un proceso de desarrollo de software que junto al Lenguaje Unificado de Modelado (UML) constituyen la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. (24)

Principales elementos de RUP: (18) (25)

- **Trabajadores ("quién"):** Define habilidades y responsabilidades (rol) de un individuo o grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
- **Actividades ("cómo"):** Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.

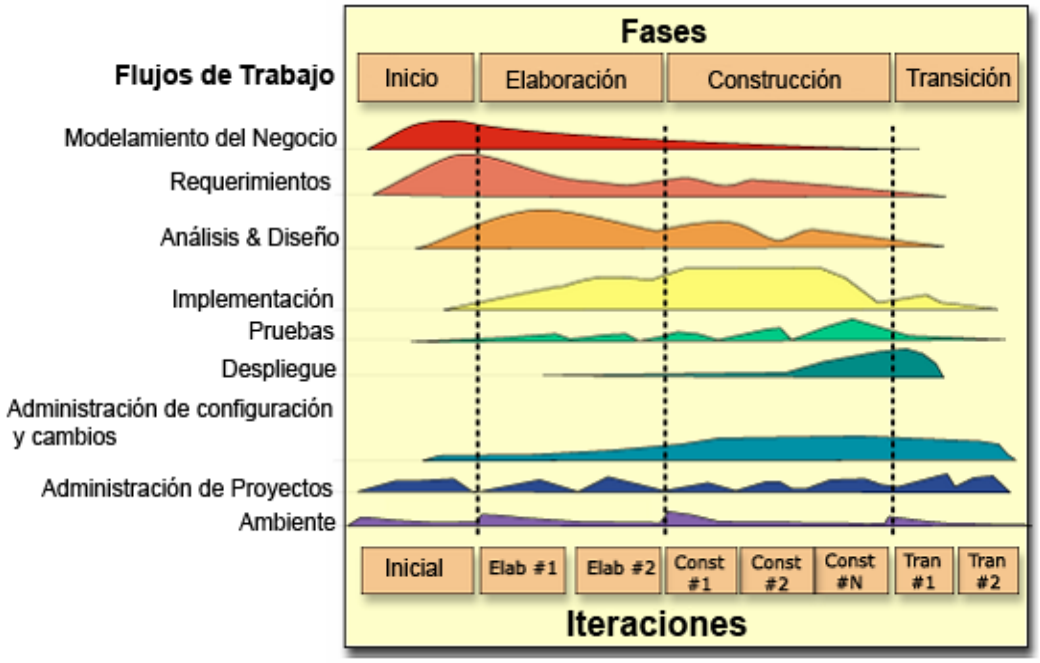
- **Artefactos ("qué"):** Productos tangibles del proyecto que son creados, modificados y usados por los trabajadores al realizar actividades.
- **Flujo de actividades ("cuándo"):** Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

RUP divide el proceso de desarrollo en una serie de ciclos que constituyen la vida de un sistema. El ciclo de vida de RUP se caracteriza por ser: (24) (18)

- **Centrado en los modelos:** Los diagramas son un vehículo de comunicación más expresivo que las descripciones en lenguaje natural. Se trata de minimizar el uso de descripciones y especificaciones textuales del sistema.
- **Guiado por los Casos de Uso:** Los Casos de Uso son el instrumento para validar la arquitectura del software y extraer los casos de prueba.
- **Centrado en la arquitectura:** Los modelos son proyecciones del análisis y el diseño constituye la arquitectura del producto a desarrollar.
- **Iterativo e incremental:** Durante el proceso de desarrollo se producen versiones incrementales (que se acercan al producto terminado) del producto en desarrollo.

Cada ciclo está compuesto por cuatro fases, y dentro de cada una, el trabajo se puede descomponer en iteraciones. Cada fase finaliza con un hito y cada uno de estos se determina por la disponibilidad de un conjunto de artefactos. Los hitos tienen muchos objetivos, entre ellos se encuentran: la toma de decisiones por parte de los directivos antes de que el trabajo pueda continuar en la siguiente fase, y además, permiten controlar el progreso del trabajo a la dirección del proyecto y a los mismos desarrolladores. (24) (25)

Flujos de trabajo y fases de RUP



Ventajas que aporta RUP (24)

- Permite desarrollar aplicaciones sacando el máximo provecho de las nuevas tecnologías, mejorando la calidad, le rendimiento, la reutilización, la seguridad y el mantenimiento del software mediante una gestión sistemática de los riesgos.
- Permite la producción de software que cumpla con las necesidades de los usuarios, a través de la especificación de los requisitos, con una agenda y costo predecible.
- Enriquece la productividad en equipo y proporciona prácticas óptimas de software a sus miembros.
- Permite llevar a cabo el proceso de desarrollo práctico, brindando amplias guías, plantillas y ejemplos para las actividades críticas.
- Proporciona guías explicitas para áreas tales como modelado de negocios, arquitectura Web, pruebas y calidad. También se proporciona guías para desarrollar en plataformas IBM WebSphere y Microsoft Web Solution para acelerar el desarrollo de los proyectos.

- Unifica el equipo de desarrollo de software y mejora la comunicación al brindar a cada miembro del mismo una base de conocimientos, un lenguaje de modelado y un punto de vista de cómo desarrollar software.
- Optimiza la productividad de cada miembro del equipo al poner al alcance la experiencia derivada de miles de proyectos y muchos líderes de la industria.
- No solo garantiza que los proyectos abordados serán ejecutados íntegramente sino que además evita desviaciones importantes respecto a los plazos.
- Permite una definición acertada del sistema en un inicio para hacer innecesarias las reconstrucciones parciales posteriores.

1.6.3 Selección de la metodología de desarrollo de software

Luego de realizar un estudio de las metodologías más utilizadas, se puede llegar a la conclusión de que RUP es la más indicada para guiar el desarrollo del sistema que se desea implementar. RUP constituye uno de los estándares internacionales que más aceptación ha tenido. Además varias herramientas CASE soportan dicha metodología, permitiendo generar código en distintos lenguajes de programación a partir de un diseño UML. RUP genera una amplia documentación necesaria para el desarrollo del software y muy útil para el caso en que se quiera estudiar el software e implementar nuevas versiones.

Además, RUP es una metodología que se encarga de: (25)

- Asegurar la producción de un software de alta calidad que reúna las necesidades de los usuarios finales dentro de un plan y un presupuesto predecible.
- Proveer un enfoque disciplinado para asignar tareas y responsabilidades dentro del desarrollo del sistema.
- Proveer un camino metódico, sistemático para desarrollar, diseñar y validar una arquitectura.
- Reducir en gran medida el riesgo que representa la construcción de sistemas complejos, porque evoluciona de forma incremental partiendo de sistemas más pequeños.

1.7 Acerca de los lenguajes de programación

Con la idea de facilitar las tareas que deben desempeñar los humanos, se han venido inventado diversas herramientas a lo largo de la historia, que permiten tener una mejor calidad de vida. Los ordenadores son uno más de los inventos del hombre, aunque se debe decir que las tecnologías para su fabricación y explotación han tenido un desarrollo sorprendente a partir de la segunda mitad del siglo XX. Esta herramienta por sí sola no es capaz de efectuar ninguna tarea, es tan sólo un conjunto de cables y circuitos que necesitan recibir instrucción por parte de los humanos para desempeñar alguna tarea. (26)

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo. En la actualidad existen una gran diversidad de lenguajes de programación, entre ellos están los lenguajes de bajo nivel, que son aquellos que se aproximan mas al lenguaje binario de las computadoras y los lenguajes de alto nivel que son aquellos que se aproximan mas al lenguaje natural humano. (26)

Entre los lenguajes de programación de alto nivel más usados en la actualidad están:

1.7.1 Lenguaje Java

Es un lenguaje de programación con el que se puede realizar cualquier tipo de programa. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Está desarrollado por la compañía Sun Microsystems con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más punteras. Sun describe a Java como "simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico". (27) (28)

Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente de la plataforma. Eso quiere decir que programa hecho en Java podrá funcionar en cualquier ordenador del mercado. Es una ventaja significativa para los desarrolladores de software, pues antes tenían que hacer un programa para cada sistema operativo, por ejemplo Windows, Linux, Apple, etc.

Esto lo consigue porque se ha creado una Máquina Virtual de Java para cada sistema que hace de puente entre el sistema operativo y el programa de Java y posibilita que este último se entienda perfectamente.

La versatilidad y eficiencia de la tecnología Java, la portabilidad de su plataforma y la seguridad que aporta, la han convertido en la tecnología ideal para su aplicación a redes. De portátiles a centros de datos, de consolas de juegos a súper equipos científicos, de teléfonos móviles a Internet, Java está en todas partes. (29)

Más de 4500 millones dispositivos utilizan la tecnología Java, como: (29)

- Más de 800 millones de equipos
- 2100 millones de teléfonos móviles y otros dispositivos de mano
- 3500 millones de tarjetas inteligentes
- Sintonizadores, impresoras, cámaras web, juegos, sistemas de navegación para automóviles, terminales de lotería, dispositivos médicos, cajeros de pago en aparcamientos.

1.7.2 Lenguaje C#

Pronunciado C Sharp, es actualmente uno de los lenguajes de programación más populares en informática y comunicaciones que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo. Es diseñado por Microsoft para su plataforma .NET. Sus principales creadores son Scott Wiltamuth y Anders Hejlsberg, éste último también conocido por haber sido el diseñador del lenguaje Turbo Pascal y la herramienta RAD Delphi. La sintaxis y estructuración de C# es muy parecida a la de C++ o Java, puesto que la intención de Microsoft es facilitar la migración de códigos escritos en estos lenguajes a C# y facilitar su aprendizaje a los desarrolladores habituados a ellos. (30) (31)

Principales características de C#: (30) (32)

- **Auto contenido:** Un programa en C# no necesita de ficheros adicionales al propio código fuente, como los ficheros de cabecera (.h) de C++.
- **Orientación a objetos:** Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos.

- **Gestión automática de memoria:** C# tiene a su disposición el recolector de basura del CLR. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos.
- **Sistema de tipos unificado:** A diferencia de C++, en C# los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada System.Object, por lo que dispondrán de los miembros definidos en ésta clase (es decir, serán "objetos").
- **Eficiente:** En principio, en C# el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros.
- **Compatibilidad:** C# mantiene una sintaxis muy similar a C++ o Java que permite, bajo ciertas condiciones, incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes.

1.7.3 Selección del lenguaje de programación

Luego de realizar un estudio de los lenguajes de programación más utilizados, se puede llegar a la conclusión de que Java es el más indicado para desarrollar el sistema que se desea implementar. Java posee una serie de ventajas significativas con respecto a los demás lenguajes de programación, el hecho de ser multiplataforma es muy importante, porque no es necesario desarrollar versiones del sistema para cada plataforma, es un lenguaje libre y existes entornos de desarrollo gratuitos de muy buena calidad para desarrollar con Java.

1.8 Acerca de los entornos de desarrollo para Java

Un entorno de desarrollo, o Integrated Development Environment (IDE) en inglés, es un programa compuesto por un conjunto de herramientas para el programador. Estos puedes dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios, es decir, consisten en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

En la actualidad existen varios entornos de desarrollo para Java, entre los más utilizados están:

1.8.1 NetBeans

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios mundialmente. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos. Hoy en día existen dos productos disponibles, NetBeans IDE y NetBeans Plataform. (33)

NetBeans IDE es un excelente entorno de desarrollo, una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Este ofrece las funciones de los IDE avanzados, como diseño de interfaz, creación automática de propiedades y clases, asistentes para la conexión con bases de datos. Puede obtener las herramientas que necesite para crear aplicaciones profesionales para el escritorio, la empresa, la web y equipos móviles con el lenguaje Java, C/C++, y Ruby. Muy fácil de instalar y de uso instantáneo y se ejecuta en varias plataformas incluyendo Windows, Linux y Mac OS X y Solaris. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. (33) (34)

Ambos productos son de código abierto y gratuito para el uso tanto comercial como no comercial. El código fuente está disponible para su reutilización de acuerdo con la Common Development and Distribution License (CDDL). (33)

1.8.2 Eclipse

Eclipse provee un conjunto de herramientas para administrar espacios de trabajo; construir, correr y depurar aplicaciones; compartir artefactos con un equipo y hacia versión de código. Es una plataforma que está diseñada para ser infinitamente extendida con cada vez más sofisticadas herramientas. Está construido sobre un mecanismo para el descubrimiento, integración y ejecución de módulos llamados plug-ins. Muchos plug-ins, comúnmente sin relación alguna, pueden ser instalados en una misma instancia de Eclipse, y convivir y cooperar sin problemas para ejecutar una cierta tarea. La clase de producto final incluye aplicaciones IDE, también denominados rich clients (clientes ricos), que se benefician del diseño de la plataforma de Eclipse y sus componentes. (35)

La plataforma Eclipse está habilitada para afrontar las siguientes necesidades:

- Soportar la construcción de gran variedad de herramientas de desarrollo.
- Soportar las herramientas proporcionadas por diferentes fabricantes de software independientes.

- Soportar herramientas que permitan manipular diferentes contenidos (HTML, Java, C, JSP, EJB, XML, y GIF).
- Facilitar una integración transparente entre las herramientas y tipos de contenidos sin tener en cuenta al proveedor.
- Proporcionar entornos de desarrollo gráfico (GUI) o no gráficos.
- Ejecutarse en una gran variedad de sistemas operativos, incluyendo Windows y Linux.

1.8.3 Selección del entorno de desarrollo

Luego de realizar un estudio de los entornos de desarrollo más utilizados para la programación en Java, se puede llegar a la conclusión de que NetBeans es el más indicado para desarrollar el sistema que se desea implementar.

Para desarrollar aplicaciones de escritorio en Eclipse hay que instalar una serie de plug-ins, los cuales no se configuran de forma sencilla, sin embargo NetBeans cuenta con una excelente interfaz muy fácil de usar para el desarrollo de este tipo de aplicaciones.

1.9 Acerca de las herramientas CASE

Se puede definir a una herramienta CASE (en español, ingeniería de software asistida por computadora, en inglés, Computer Arded Software Engineering) como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante los pasos del ciclo de vida de desarrollo de un software. (36)

Entre las principales ventajas que aporta la utilización de herramientas CASE, se tienen que permiten:

- El incremento en la velocidad de desarrollo de los sistemas.
- A los analistas tener más tiempo para el análisis y diseño además de minimizar el tiempo para codificar y probar.
- Automatizar el dibujo de diagramas.
- Ayudar en la documentación del sistema.
- Ayudar en la creación de relaciones en la base de datos.
- Generar estructuras de código.
- Aumentar la productividad. Esto se consigue a través de la automatización de determinadas tareas, como la generación de código y la reutilización de objetos o módulos.

Actualmente en el mercado existe una gran variedad de herramientas que automatizan una o varias de las actividades del ciclo de vida de los sistemas. Entre más amplio sea el espectro de actividades que automatiza, mayor efectividad se obtiene con su uso. Entre las más usadas están:

1.9.1 Rational Rose Enterprise Edition

Es una herramienta CASE basada en UML que permite crear los diagramas que se van generando durante el proceso de ingeniería en el desarrollo del software. Es completamente compatible con la metodología RUP, brinda muchas facilidades en la generación de la documentación del software que se están desarrollando, además posee un gran número de estereotipos predefinidos que viabiliza el proceso de modelación del software. Es capaz de generar el código fuente de las clases definidas en el flujo de trabajo de diseño, pero tiene la limitación de que aún hay varios lenguajes de programación que no soporta o que sólo lo hace a medias. (37)

1.9.2 Visual Paradigm

Visual Paradigm es una de las herramientas CASE del mercado, considerada muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Permite graficar los diferentes diagramas UML, revertir y generar código fuente para Java, C++, PHP, DotNet Exe/dll, XML, XML Schema, Python y Corba IDL. Visual Paradigm incluye los objetos más recientes de UML además de diagramas de casos de uso, diagramas de clase, diagramas de componentes, ofrece soporte para Rational Rose, integración con Microsoft Visio, además permite generar reportes y documentación en HTML/PDF. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto, además apoya los estándares más recientes de las notaciones de Java y de UML. (38)

Una de sus principales ventajas es que incorpora el soporte para trabajo en equipo, permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros. (38)

1.9.3 Selección de la herramienta CASE

Luego de realizar un estudio de las herramientas de modelado UML más utilizados para la programación en Java, se puede llegar a la conclusión de que Visual Paradigm es el más indicado para desarrollar el sistema que se desea implementar, además la Universidad de las Ciencias Informáticas cuenta con una

licencia para su uso, funciona perfectamente sobre la plataforma GNU Linux y se integra con el entorno de desarrollo NetBeans.

1.10 Acerca de los framework de pruebas de unidad para Java

Encontrar errores es una tarea desafiante para cualquier desarrollador, pero existen herramientas que proporcionan una manera sencilla, rápida y elegante para escribir pruebas y validarlas automáticamente. Entre las más usadas actualmente están:

1.10.1 JUnit

Es un framework que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente. (39)

1.10.2 TestNG

Es un framework para realizar pruebas para Java, está basado en JUnit, pero introduciendo nuevas funcionalidades que los hacen más poderosos y fáciles de usar, tales como: (39)

- Anotaciones JDK 5.
- Configuración flexible de pruebas.
- Soporte de pasaje de parámetros.
- Permite distribución de las pruebas en maquinas esclavas.
- Modelo de ejecución poderoso
- Soportado por herramientas y plugins importantes y variados como: (Eclipse, IDEA, Maven, etc.).
- Métodos dependientes para pruebas sobre servidores de aplicación.

1.10.3 JTiger

Es un framework de código abierto que provee una solución robusta para realizar pruebas de unidad para Java. JTiger no requiere archivos externos de configuración para ejecutar las pruebas. (39)

1.10.4 Selección del framework de pruebas de unidad

Luego de realizar un estudio de los framework de pruebas de unidad para Java, se puede llegar a la conclusión de que JUnit es el más indicado para desarrollar el sistema que se desea implementar, es uno de los framework integrado con el entorno de desarrollo seleccionado NetBeans, lo cual facilita su uso.

1.11 Selección de otras herramientas y tecnologías a utilizar

1.11.1 XML

Lenguaje extensible de marcas (Extensible Markup Language) creado por World Wide WEB Consortium (W3C) para describir estructuras de datos a almacenar en un documento. La sencillez de su estructura y la facilidad de su uso lo han convertido en uno de los estándares más difundidos en Internet. Se dice que a partir de su creación se ha cambiado la forma en que se hacen aplicaciones. La inmensa mayoría de los lenguajes de programación han implementado interfaces para trabajar con este estándar. (40)

1.11.2 XSTREAM

Es una librería java muy fácil de usar, la cual permite transformar un objeto java a un documento XML y viceversa.

Características de XSTREAM: (41)

- **Fácil de usar:** Brinda una fachada de alto nivel que simplifica los casos de usos comunes.
- **No requiere mapeo:** La mayoría de los objetos pueden ser serializados sin la necesidad de especificar mapeos.
- **Alto rendimiento:** La velocidad y el tamaño de la memoria mínima son elementos cruciales en el diseño, haciéndolo ajustable para objetos grandes o sistemas con grandes cantidades de intercambio de mensajes.
- **XML limpio:** No hay información duplicada si esta no puede ser obtenida a través de reflexión. El resultado de esto es un XML que es más fácil de entender por las personas y más compacta que la serializarían Java.
- **Serializa campos internos:** Serializa campos internos incluyendo private y final, soporta clases no públicas e internas, las clases no necesitan tener un constructor por defecto.

- **Soporta referencias duplicadas:** Las referencias duplicadas del modelo de objetos serán mantenidas, soporta referencias circulares.
- **Integración con otros APIs XML:** XStream puede serializar directamente desde o hacia cualquier estructura árbol (no solo XML) mediante la implementación de interfaces.
- **Estrategias de conversión personalizables:** Las estrategias pueden ser registradas permitiendo la personalización de como son representados tipos particulares a XML.
- **Mensajes de error.** Cuando una excepción ocurre debido a un XML mal formado, se proveen diagnósticos detallados para poder aislar y arreglar el problema.

Conclusiones del capítulo

Después de haber estudiado de las principales herramientas y tecnologías existentes se decidió desarrollar una herramienta de generación de código, porque ninguna de las disponibles responde a las necesidades del desarrollo de aplicaciones Web que utilicen la librería AJAX ExtJS.

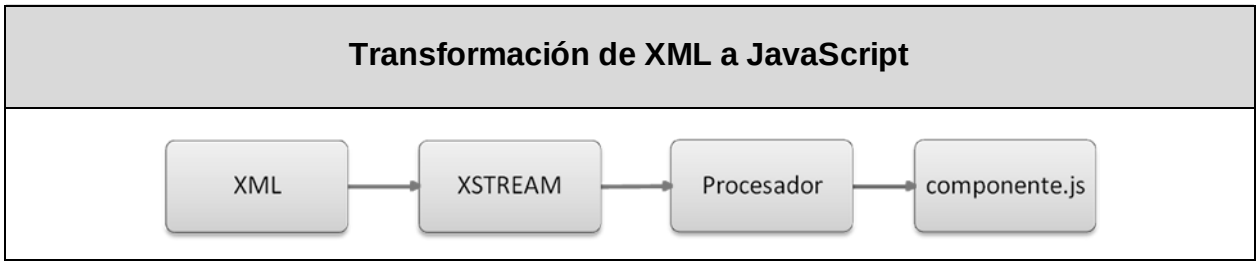
Inicialmente se hará un generador de JavaScript pasivo, aplicando las técnicas de clonación y concatenación de cadenas. Este generador tomará como entrada un modelo UML generado por el usuario y a través de él se obtendrá como salida el código JavaScript necesario para aplicaciones Web que utilicen la librería ExtJS.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Descripción general del sistema

2. 1.1 Funcionamiento del Generador

El uso de XSTREAM facilita el proceso de transformación de un modelo a código. Esta librería permite transformar un documento XML a un objeto Java y viceversa. De esta forma a partir de un modelo XML que contenga la información de un componente Ext para realizar las transformaciones se obtiene como resultado código JavaScript para crear el componente. A continuación se muestra un diagrama de su funcionamiento.



Resuelto el problema de la transformación de un modelo XML a código JavaScript falta definir cómo el generador llevará a cabo las transformaciones para obtener el resultado deseado.

2.1.2 Configurar el XML de un componente

Para representar los componentes Ext, se definen documentos XML que contienen sus propiedades iniciales y su sintaxis de programación, cada documento tiene un nombre único el cual identifica al componente.

El XML de un componente pudiera ser así:

```
<component>
    <name>Ext.form.TextField</name>
    <description>Basic text field. Can be used as a direct replacement for traditional text inputs, or as the base
    <syntax>
        <xtype>textfield</xtype>
        <start>new Ext.form.TextField({</start>
        <end>})</end>
    </syntax>
    <properties>
        <property>
            <realName>allowBlank</realName>
            <label>allowBlank</label>
            <description>False to validate that the defaultValue length > 0 (defaults to true)</description>
            <editor>
                <defaultValue>true</defaultValue>
                <valueType>boolean</valueType>
                <values>
                    <string>true</string>
                    <string>false</string>
                </values>
                <editable>false</editable>
            </editor>
        </property>
        <property>
            <realName>blankText</realName>
            <label>blankText</label>
            <description>Error text to display if the allow blank validation fails (defaults to 'This field is req
        </property>
        <property>
            <realName>emptyText</realName>
            <label>emptyText</label>
            <description>The default text to display in an empty field (defaults to null)</description>
        </property>
        <property>
            <realName>vtype</realName>
            <label>vtype</label>
            <description>A validation type name as defined in Ext.form.VTypes (defaults to null)</description>
            <editor>
                <values>
                    <string>alphaMask</string>
                    <string>alphanumMask</string>
                    <string>emailMask</string>
                </values>
                <editable>true</editable>
            </editor>
        </property>
    </properties>
</component>
```

- **<component>** comienza y finaliza el XML
- **<name>** nombre del componente

  El nombre del componente es utilizado para mostrarlo en el panel de propiedades.
- **<description>** descripción del componente

  La descripción del componente es utilizada para mostrarla en el panel de propiedades.
- **<syntax>** sintaxis del componente

  Codigo JavaScript con el que se declara el componente.
- **<xtype>** xtype del componente

  El xtype del componente es una propiedad de los componentes Ext que se utiliza para declarar el componente en caso que esté contenido dentro de otro componente.
- **<start>** comienzo de la declaración

  El comienzo de la declaración es utilizada para indicar al generador como debe iniciar la declaración del componente.
- **<end>** fin de la declaración

  El fin de la declaración es utilizada para indicar al generador como debe finalizar la declaración del componente.
- **<properties>** lista de propiedades del componente
- **<property>** propiedad del componente
- **<realName>** nombre real

  El nombre real de la propiedad es el que utiliza el generador para generar el código, este debe ser el nombre real de la propiedad.
- **<label>** titulo de la propiedad

  El titulo de la propiedad es utilizado para mostrarlo en el panel de propiedades.
- **<description>** descripción de la propiedad

  La descripción de la propiedad es utilizada para mostrarla en el panel de propiedades.
- **<editor>** editor de la propiedad

  El editor de la propiedad es el componente que utiliza el generador para modificar el valor de la propiedad.
- **<defaultValue>** valor por defecto

Es el valor que trae la propiedad por defecto en la Ext, si el valor de la propiedad es distinto del valor por defecto, esta propiedad se incluirá en el código generado.

- **<initialValue>** valor inicial

  Es el valor con el que iniciará el editor de la propiedad.
- **<valueType>** tipo del valor

  El tipo del valor es utilizado por el generador para validar el valor de la propiedad y para generar el editor, se admiten cinco tipos (text, number, boolean, code, item).
- **<values>** valores posibles

  Valores posibles que puede tomar la propiedad.
- **<editable>** editable

  Es utilizado por el generador para indicarle al editor si este va a ser editable o no, se admiten dos valores (true, false).

2.1.3 Configurar la paleta de componentes

Una vez configurados los componentes que contendrá el sistema, se debe configurar la paleta de componentes. La paleta de componentes es construida a partir de un documento XML, el cual es cargado con la librería XSTREAM. La estructura de este XML sería la siguiente:

```
<palette>
  <buttons>
    <button>
      <text>Form Panel</text>
      <classPath>Interface.Components.Ext.Panel.FormPanel</classPath>
      <icon>formPanel.png</icon>
      <group>Containers</group>
    </button>
    <button>
      <text>Grid Panel</text>
      <classPath>Interface.Components.Ext.Panel.GridPanel</classPath>
      <icon>gridPanel.png</icon>
      <group>Containers</group>
    </button>
    <button>
      <text>Text Field</text>
      <classPath>Interface.Components.Ext.Field.TextField</classPath>
      <icon>textField.png</icon>
      <group>Controls</group>
    </button>
    <button>
      <text>Number Field</text>
      <classPath>Interface.Components.Ext.Field.NumberField</classPath>
      <icon>numberfield.png</icon>
      <group>Controls</group>
    </button>
```

- **<palette>** comienza y finaliza el XML
- **<buttons>** lista de botones de la paleta de componentes
- **<button>** botón de la paleta de componentes
- **<text>** texto del botón
- **<classPath>** dirección de la clase

  La dirección de la clase es utilizada por el generador para saber que clase java es la que representa a este componente.
- **<icon>** icono del botón
- **<group>** grupo del componente

  El grupo es utilizado por el generador para agrupar los botones de la paleta de componentes.

2.2 Requisitos funcionales

A continuación se presentan los requisitos funcionales del sistema expresados en lenguaje natural. Los mismos serán identificados con las siglas RF- más el Número del requisito (Ej. RF-1.) y clasificados según su prioridad en Alta (Esencial), Media (Deseado) o Baja (Opcional).

RF-1 Seleccionar el espacio de trabajo.

El sistema permitirá seleccionar el espacio de trabajo mediante un explorador.
Prioridad: Media

RF-2 Validar espacio de trabajo.

El sistema validará que el directorio del espacio de trabajo exista.
Prioridad: Media

RF-3 Configurar espacio de trabajo.

El sistema verificará el espacio de trabajo, en caso de ser nuevo, este copiará la librería ExtJS.
Prioridad: Media

RF-4 Mostrar estructura del espacio de trabajo.

El sistema mostrará en forma de árbol la estructura del espacio de trabajo.
Prioridad: Media

RF-5 Mostrar paleta de componentes.

El sistema mostrara una paleta con los componentes Ext que se pueden crear.
Prioridad: Alta

RF-6 Crear proyecto.

El sistema permitirá crear nuevos proyectos en el espacio de trabajo.
Prioridad: Media

RF-7 Eliminar proyecto.

El sistema permitirá eliminar proyectos del espacio de trabajo.
Prioridad: Media

RF-8 Seleccionar proyecto principal.

El sistema permitirá seleccionar un proyecto del espacio de trabajo como proyecto principal, para indicar que sobre este se trabajará.
Prioridad: Media

RF-9 Crear archivo.

El sistema permitirá crear nuevos archivos, después de creado, el sistema mostrará una interfaz en la cual se podrá realizar el diseño y ver el código generado.
Prioridad: Alta

RF-10 Guardar archivo.

El sistema permitirá guardar archivos en el espacio de trabajo.
Prioridad: Alta

RF-11 Abrir archivo.

El sistema permitirá abrir archivos del espacio de trabajo.
Prioridad: Alta

RF-12 Eliminar archivo.

El sistema permitirá eliminar archivos del espacio de trabajo.
Prioridad: Alta

RF-13 Crear componente.

El sistema permitirá crear nuevos componentes en el archivo activo, una vez creado, el sistema mostrará sus propiedades.
Prioridad: Alta

RF-14 Seleccionar componente.

El sistema permitirá seleccionar un componente, una vez seleccionado, el sistema mostrará sus propiedades.
Prioridad: Alta

RF-15 Eliminar componente.

El sistema permitirá eliminar el componente seleccionado.
Prioridad: Alta

RF-16 Modificar propiedades.

El sistema permitirá modificar las propiedades del componente seleccionado.
Prioridad: Alta

RF-17 Generar código.

El sistema permitirá generar el código correspondiente al diseño del fichero activo.
Prioridad: Alta

RF-18 Probar el código generado.

El sistema permitirá probar el código generado del archivo activo en el navegador definido por defecto en el sistema operativo.
Prioridad: Media

2. 3 Requisitos no funcionales

Seguridad
RNF-1 Cargar configuración por sesión.

El sistema cargará su configuración en dependencia de la sesión en la que sea ejecutado.

Usabilidad
RNF-2 Usar teclado.

El sistema permitirá el uso del teclado para realizar operaciones.

RNF-3 Interfaz agradable.

El sistema poseerá una interfaz agradable al cliente.

RNF-4 Código organizado.

El sistema generará el código de forma organizada.

Portabilidad
RNF- 5 Multiplataforma

El sistema funcionará sobre cualquier plataforma.

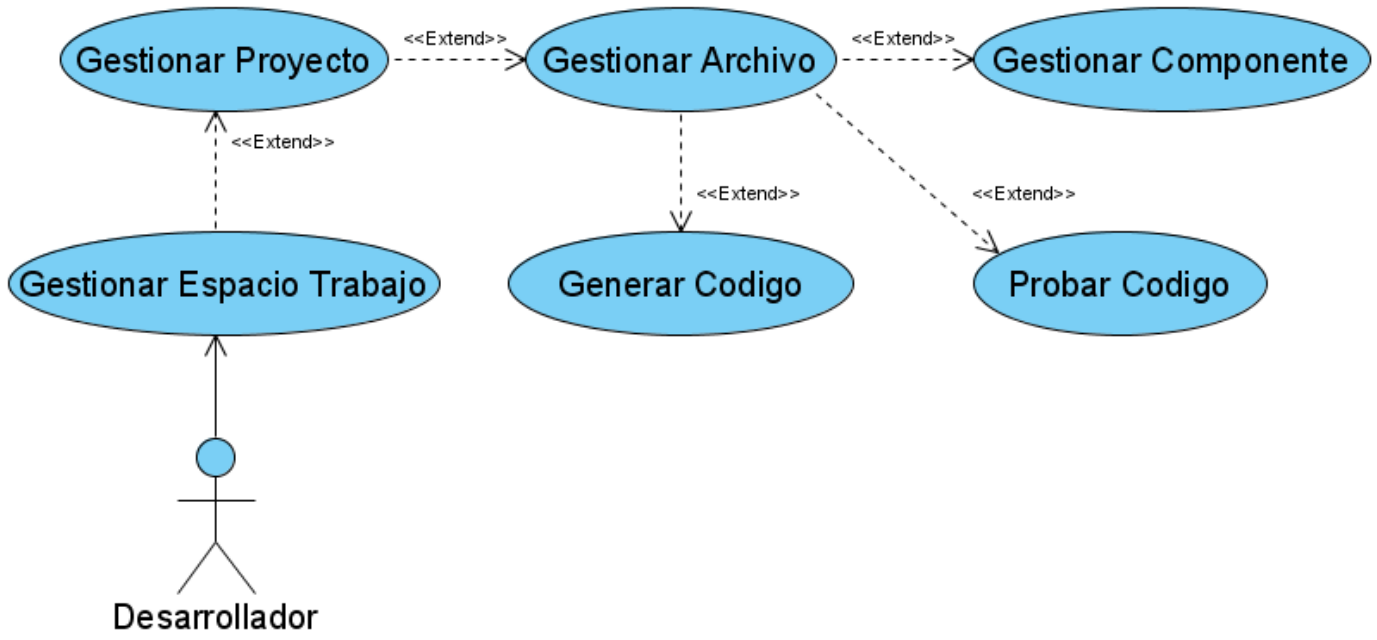
Software
RNF-6 Maquina Virtual de Java 1.6

Para que el sistema funcione, debe estar instalada la maquina virtual de Java 1.6.

2.4 Actor del Sistema

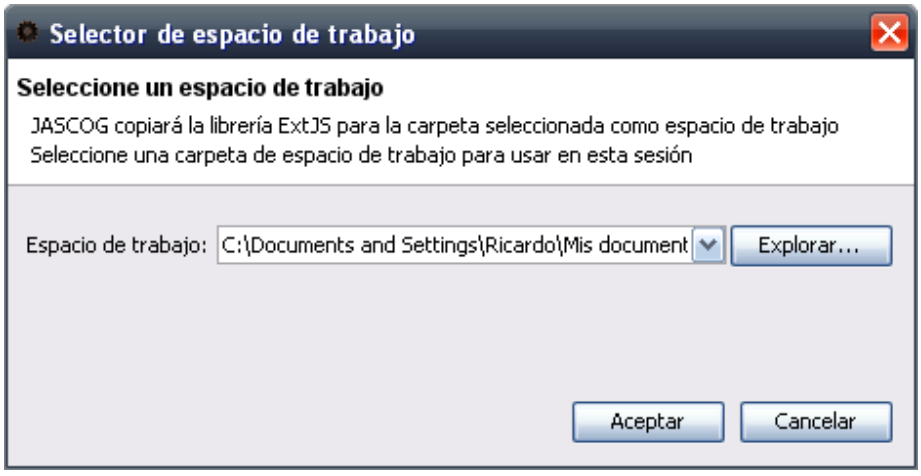
Actor	Descripción
Desarrollador	Es quien usará la herramienta de generación en el desarrollo de un software.

Diagrama de Casos de Uso del Sistema



2.5 Descripción de Casos de Uso del Sistema

<table>
<tr><td>Caso de Uso:</td><td colspan="2">Gestionar Espacio Trabajo</td></tr>
<tr><td>Actores:</td><td colspan="2">Desarrollador</td></tr>
<tr><td>Resumen:</td><td colspan="2">El caso de uso consiste en seleccionar la carpeta donde se van a crear los proyectos deseados y copiar la librería ExtJS para la carpeta seleccionada.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2">• El sistema debe estar instalado y ejecutándose correctamente.</td></tr>
<tr><td>Prioridad:</td><td colspan="2">Opcional</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="3">Sección “Seleccionar espacio de trabajo”</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. El caso de uso inicia cuando el actor Desarrollador ejecuta la aplicación.</td><td>2. El sistema carga la configuración personal y muestra una interfaz con una lista de los espacios de trabajo seleccionados anteriormente.</td></tr>
<tr><td colspan="2">3. El actor realiza una de las siguientes opciones:
a. Escribe la dirección del espacio de trabajo. Continúa con el paso 7 del flujo normal de eventos.
b. Selecciona la opción “Explorar”. Continúa con el paso 4 del flujo</td><td>4. El sistema muestra una interfaz con un explorador de carpetas para seleccionar una.</td></tr>
</table>

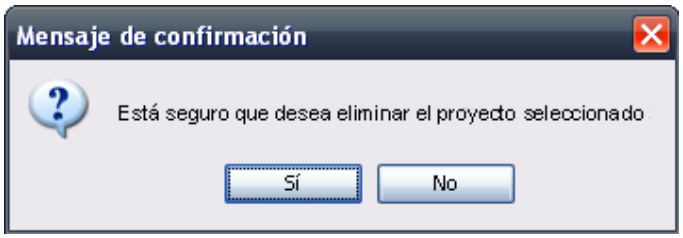
normal de eventos.	
5. El actor selecciona una carpeta y selecciona la opción “Abrir”.	6. El sistema obtiene la dirección de la carpeta seleccionada y la adiciona a la lista de espacios de trabajo.
7. El actor selecciona la opción “Aceptar”.	8. El sistema valida que la dirección seleccionada exista.
	9. El sistema guarda la configuración personal, carga los proyectos que tenga el espacio de trabajo, carga la paleta de componentes. Terminando así el caso de uso.
Prototipo de Interfaz	
	
Flujos alternos	
Flujo alterno al paso 3 “Operación cancelada”	
3. a El actor selecciona la opción “Cancelar”.	3. b El sistema cancela la operación y se cierra.

	Terminando así el caso de uso.
Flujo alterno al paso 5 "Datos incorrectos o campos vacíos"	
	8. a El sistema muestra mensaje de error. Terminando así el caso de uso.
Prototipo de Interfaz	
	
Pos-condiciones:	• El sistema queda con un nuevo espacio de trabajo creado. • El sistema queda con los mismos espacios de trabajo.

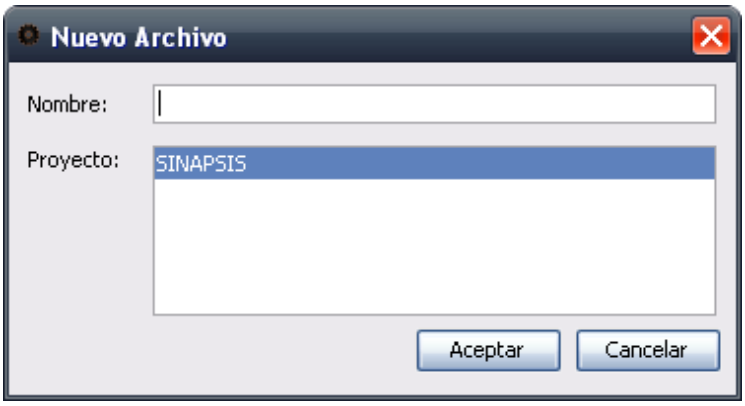
Caso de Uso:	**Gestionar Proyecto**
Actores:	Desarrollador
Resumen:	El caso de uso consiste en crear o eliminar un proyecto.
Precondiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • Debe estar seleccionado un espacio de trabajo.

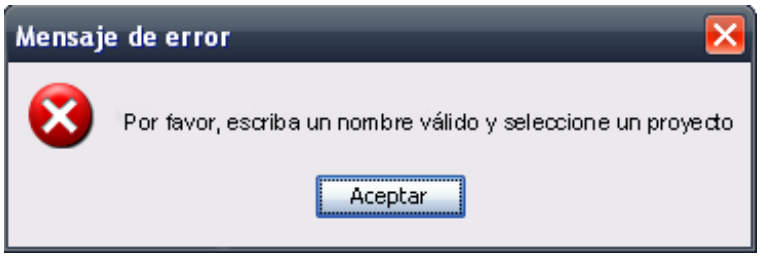
Prioridad:	Opcional
Flujo Normal de Eventos	
Sección "Crear proyecto"	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el actor Desarrollador presiona la combinación de teclas "Ctrl+Mayúsculas+N", selecciona el menú "Archivo" y dentro de este la opción "Nuevo proyecto" o hace clic derecho sobre el espacio de trabajo y selecciona la opción "Nuevo proyecto".	2. El sistema muestra la interfaz para crear un proyecto, solicitando el nombre.
3. El actor introduce el nombre del proyecto y selecciona la opción "Aceptar".	4. El sistema valida que el nombre del proyecto sea correcto.
	5. El sistema crea el proyecto. Terminando así el caso de uso.
Prototipo de Interfaz	
	

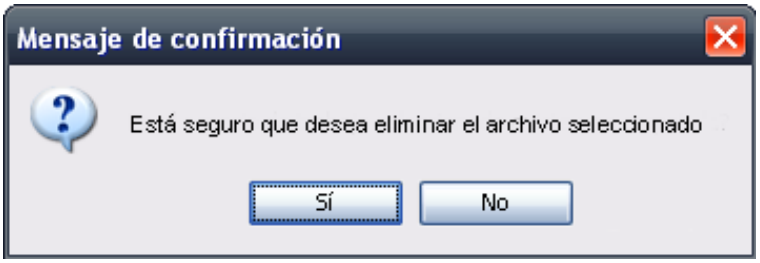
<table>
<tr><th colspan="2">Flujos alternos</th></tr>
<tr><th colspan="2">Flujo alterno al paso 3 “Operación cancelada”</th></tr>
<tr><td>3. a El actor selecciona la opción “Cancelar”.</td><td>3. b El sistema cancela la operación. Terminando así el caso de uso.</td></tr>
<tr><th colspan="2">Flujo alterno al paso 4 “Nombre incorrecto”</th></tr>
<tr><td></td><td>4. a El sistema muestra un mensaje de error. Terminando así el caso de uso.</td></tr>
<tr><th colspan="2">Prototipo de Interfaz</th></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><th colspan="2">Flujo Normal de Eventos</th></tr>
<tr><th colspan="2">Sección “Eliminar proyecto”</th></tr>
<tr><th>Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td>1. El caso de uso inicia cuando el actor Desarrollador selecciona un proyecto y presiona la tecla “Suprimir” o hace clic derecho sobre el proyecto y selecciona la opción “Eliminar”.</td><td>2. El sistema muestra un mensaje de confirmación.</td></tr>
</table>

3. El actor selecciona la opción “Si”.	4. El sistema elimina el proyecto. Terminando así el caso de uso.
Prototipo de Interfaz	
	
Flujos alternos	
Flujo alterno al paso 3 “Operación cancelada”	
3. a El actor selecciona la opción “No”.	3. b El sistema cancela la operación. Terminando así el caso de uso.
Pos-condiciones:	• El sistema queda con un nuevo proyecto creado. • El sistema queda sin un proyecto de los existentes.

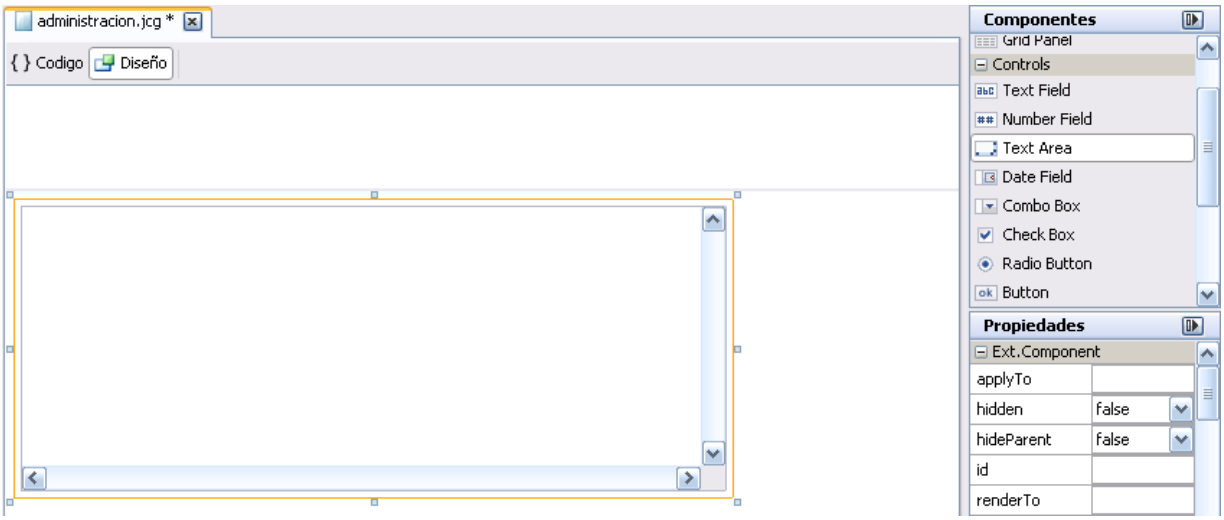
Caso de Uso:	**Gestionar Archivo**
Actores:	Desarrollador
Resumen:	El caso de uso consiste en crear o eliminar un archivo.
Precondiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • Debe estar seleccionado un espacio de trabajo. • Debe estar creado un proyecto.
Prioridad:	Crítico

<table>
<tr><th colspan="2">Flujo Normal de Eventos</th></tr>
<tr><th colspan="2">Sección “Crear archivo”</th></tr>
<tr><th>Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td>1. El caso de uso inicia cuando el actor Desarrollador presiona la combinación de teclas “Ctrl+N“, selecciona el menú “Archivo” y dentro de este la opción “Nuevo archivo” o hace clic derecho sobre un proyecto y selecciona la opción “Nuevo Archivo”.</td><td>2. El sistema muestra la interfaz para crear un archivo, solicitando el nombre y el proyecto donde va a ser creado.</td></tr>
<tr><td>3. El actor introduce el nombre del archivo, selecciona el proyecto y selecciona la opción “Aceptar”.</td><td>4. El sistema valida que el nombre del archivo sea correcto y que exista un proyecto seleccionado.</td></tr>
<tr><td></td><td>5. El sistema crea el archivo. Terminando así el caso de uso.</td></tr>
<tr><th colspan="2">Prototipo de Interfaz</th></tr>
<tr><td colspan="2"> 
</td></tr>
</table>

<table>
<tr><th colspan="2">Flujos alternos</th></tr>
<tr><th colspan="2">Flujo alterno al paso 3 “Operación cancelada”</th></tr>
<tr><td>3. a El actor selecciona la opción “Cancelar”.</td><td>3. b El sistema cancela la operación.
Terminando así el caso de uso.</td></tr>
<tr><th colspan="2">Flujo alterno al paso 4 “Nombre incorrecto”</th></tr>
<tr><td></td><td>4. a El sistema muestra mensaje de error.
Terminando así el caso de uso.</td></tr>
<tr><th colspan="2">Prototipo de Interfaz</th></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><th colspan="2">Flujo Normal de Eventos</th></tr>
<tr><th colspan="2">Sección “Guardar archivo”</th></tr>
<tr><th>Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td>1. El caso de uso inicia cuando el actor Desarrollador presiona la combinación de teclas “Ctrl+S“o selecciona el menú “Archivo” y dentro de este la opción “Guardar”.</td><td>2. El sistema guarda el archivo.
Terminando así el caso de uso.</td></tr>
<tr><th colspan="2">Flujo Normal de Eventos</th></tr>
<tr><th colspan="2">Sección “Eliminar archivo”</th></tr>
</table>

<table>
<tr><th colspan="2">Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td colspan="2">1. El caso de uso inicia cuando el actor Desarrollador selecciona un archivo y presiona la tecla “Suprimir” o hace clic derecho sobre el archivo y selecciona la opción “eliminar”.</td><td>2. El sistema muestra un mensaje de confirmación.</td></tr>
<tr><td colspan="2">3. El actor selecciona la opción “Si”.</td><td>4. El sistema elimina el archivo. Terminando así el caso de uso.</td></tr>
<tr><td colspan="3">Prototipo de Interfaz</td></tr>
<tr><td colspan="3"> 
</td></tr>
<tr><td colspan="3">Flujos alternos</td></tr>
<tr><td colspan="3">Flujo alterno al paso 3 “Operación cancelada”</td></tr>
<tr><td colspan="2">3. a El actor selecciona la opción “No”.</td><td>3. b El sistema cancela la operación. Terminando así el caso de uso.</td></tr>
<tr><td>Pos-condiciones:</td><td colspan="2">• El sistema queda con un nuevo archivo creado.
• El sistema queda con un archivo modificado y guardado.
• El sistema queda sin un archivo de los existentes.</td></tr>
</table>

<table>
<tr><td>Caso de Uso:</td><td colspan="2">Gestionar Componente</td></tr>
<tr><td>Actores:</td><td colspan="2">Desarrollador</td></tr>
<tr><td>Resumen:</td><td colspan="2">El caso de uso consiste en crear, modificar o eliminar un componente.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2">• El sistema debe estar instalado y ejecutándose correctamente.
• Debe estar seleccionado un espacio de trabajo.
• Debe estar creado un proyecto.
• Debe estar creado un archivo.</td></tr>
<tr><td>Prioridad:</td><td colspan="2">Crítico</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="3">Sección “Crear componente”</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. El caso de uso inicia cuando el actor selecciona un componente de la paleta de componentes y hace clic sobre el área de diseño.</td><td>2. El sistema crea el componente, lo adiciona en el área de diseño, lo adiciona en el inspector de objetos, lo selecciona y muestra sus propiedades. Terminando así el caso de uso.</td></tr>
<tr><td colspan="3">Prototipo de Interfaz</td></tr>
</table>

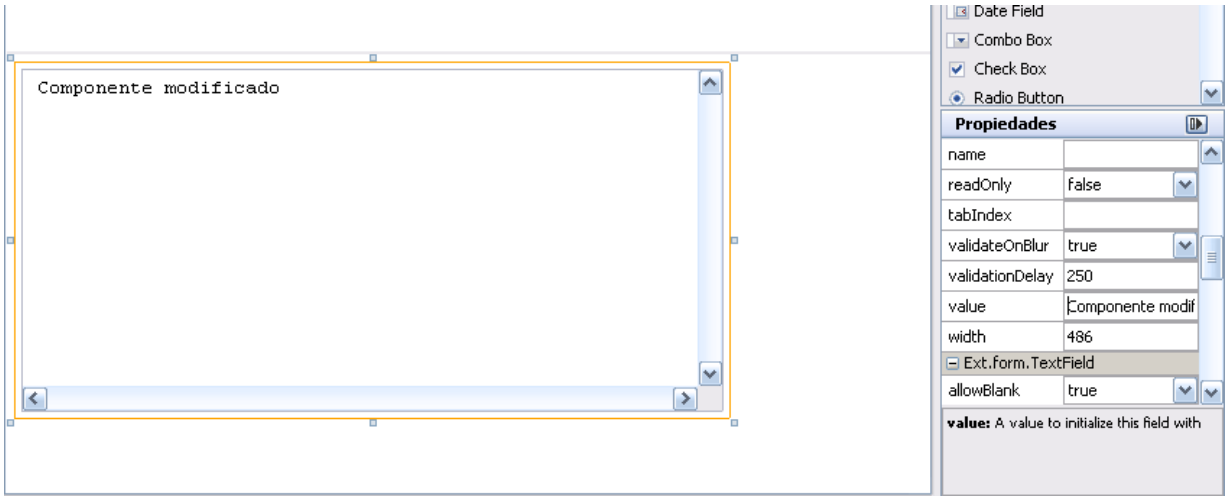


Flujo Normal de Eventos

Sección "Modificar componente"

Acción del Actor	Respuesta del Sistema
1. El caso de uso inicia cuando el actor Desarrollador hace clic sobre un componente creado.	2. El sistema selecciona el componente y muestra sus propiedades.
3. El actor modifica sus propiedades.	4. El sistema modifica el componente. Terminando así el caso de uso.

Prototipo de Interfaz



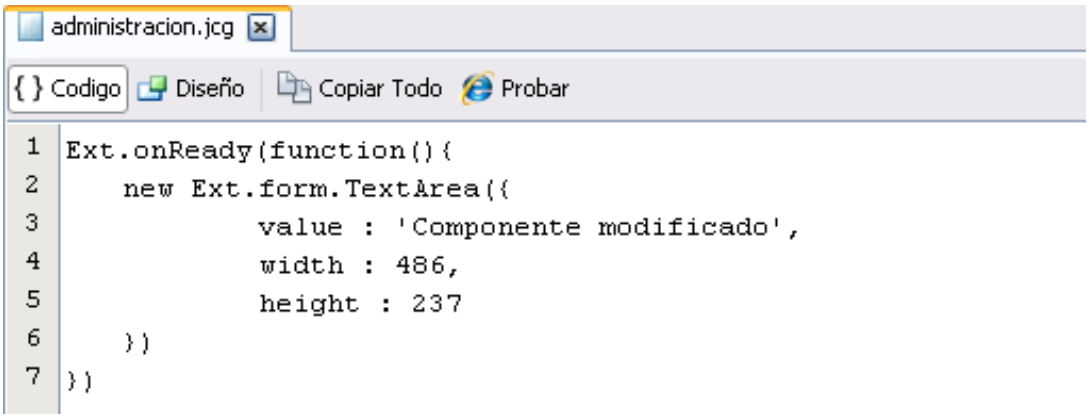
Flujo Normal de Eventos

Sección "Eliminar componente"

Acción del Actor	Respuesta del Sistema
1. El caso de uso inicia cuando el actor Desarrollador hace clic sobre un componente creado.	2. El sistema selecciona el componente.
3. El actor presiona la tecla suprimir.	4. El sistema elimina el componente. Terminando así el caso de uso.

Pos-condiciones:	• El sistema queda con un nuevo componente creado. • El sistema queda con un componente modificado. • El sistema queda sin un componente de los existentes.

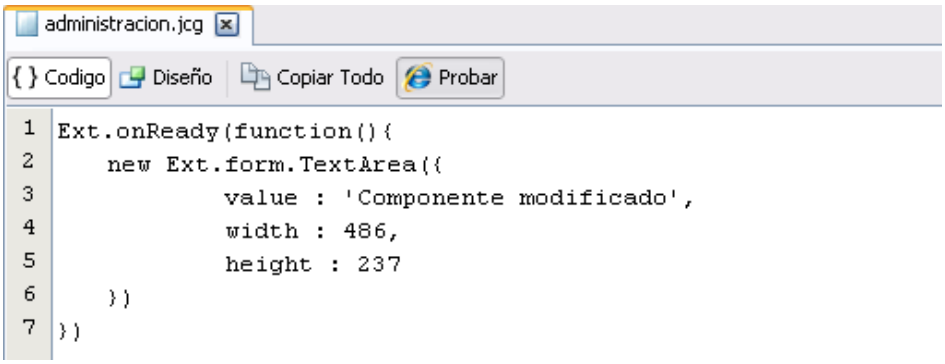
<table>
<tr><td>Caso de Uso:</td><td colspan="2">Generar Codigo</td></tr>
<tr><td>Actores:</td><td colspan="2">Desarrollador</td></tr>
<tr><td>Resumen:</td><td colspan="2">El caso de uso consiste en generar el código correspondiente al diseño del archivo seleccionado.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2">• El sistema debe estar instalado y ejecutándose correctamente.
• Debe estar seleccionado un espacio de trabajo.
• Debe estar creado un proyecto.
• Debe estar creado un archivo.</td></tr>
<tr><td>Prioridad:</td><td colspan="2">Crítico</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="3">Sección “Generar código”</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. El caso de uso inicia cuando el actor cambia a la vista de código.</td><td>2. El sistema genera el código correspondiente al diseño, lo muestra en el editor de código y guarda en el espacio de trabajo un fichero .js con el código generado. Terminando así el caso de uso.</td></tr>
<tr><td colspan="3">Prototipo de Interfaz</td></tr>
</table>



Pos-condiciones:	• El sistema queda con el código correspondiente al diseño generado y guardado.

Caso de Uso:	**Probar Codigo**
Actores:	Desarrollador
Resumen:	El caso de uso consiste en probar el código generado en el navegador seleccionado por defecto para abrir páginas web.
Precondiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • Debe estar seleccionado un espacio de trabajo. • Debe estar creado un proyecto. • Debe estar creado un archivo.
Prioridad:	Opcional

Flujo Normal de Eventos	
Sección "ProbarCodigo"	
Acción del Actor	**Respuesta del Sistema**

1. El caso de uso inicia cuando el actor cambia a la vista de código y selecciona la opción “Probar”.	2. El sistema guarda en el espacio de trabajo un archivo .html el cual incluye la librería Ext copiada inicialmente y el archivo .js generado, después de guardado es ejecutado. Terminando así el caso de uso.
Prototipo de Interfaz	
	
Pos-condiciones:	• El sistema queda con un nuevo archivo .html guardado.

Conclusiones del capítulo

En este capítulo se han abordado aspectos relacionados con la propuesta de implementar una herramienta que genere código JavaScript para la librería ExtJS con el objetivo de disminuir el tiempo de desarrollo de la interfaz de usuario del proyecto SINAPSIS u otras aplicaciones que utilicen la librería.

Se identificaron los requerimientos del sistema, obteniéndose así sus funcionalidades y un listado de requerimientos no funcionales a tener en cuenta para el desarrollo de la herramienta.

Se definieron los casos de uso del sistema, mostrándose el diagrama de casos de uso y la descripción de cada uno de ellos, lo que guiará el proceso de desarrollo a lo largo de los flujos de trabajo de diseño, implementación y prueba.

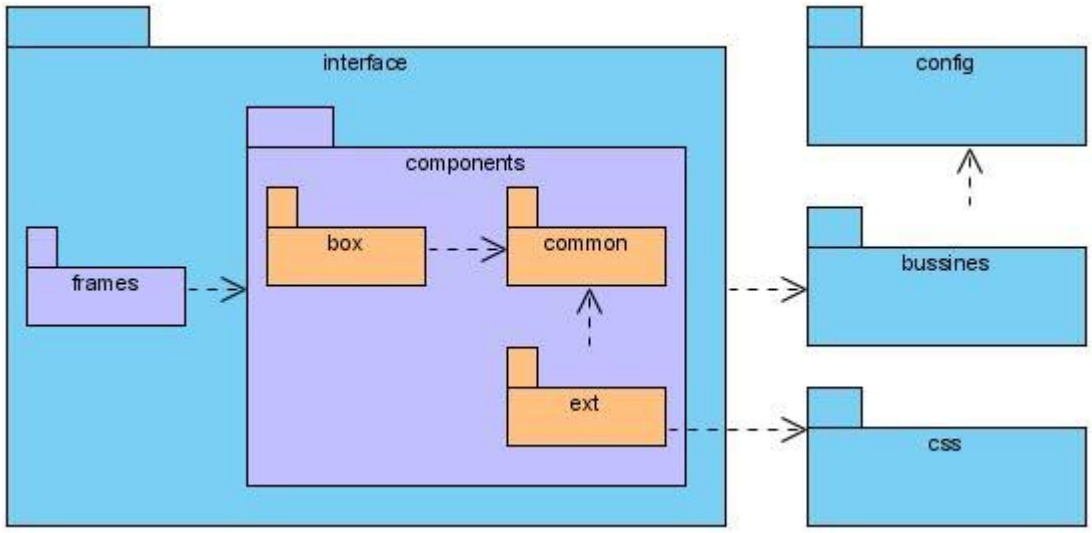
CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN

3.1 MODELO DE DISEÑO

En este modelo se crea un diseño que da respuesta a los requisitos del software teniendo en cuenta el leguaje de programación a utilizar, los componentes y las tecnologías empleadas. Con este modelo se tiene un punto de partida para comenzar la implementación de software.

3.1.1 Paquetes del Diseño

Paquete en que se agrupan las clases del diseño



3.1.2 Diagramas de Clases del diseño

Diagrama de clases del paquete: config

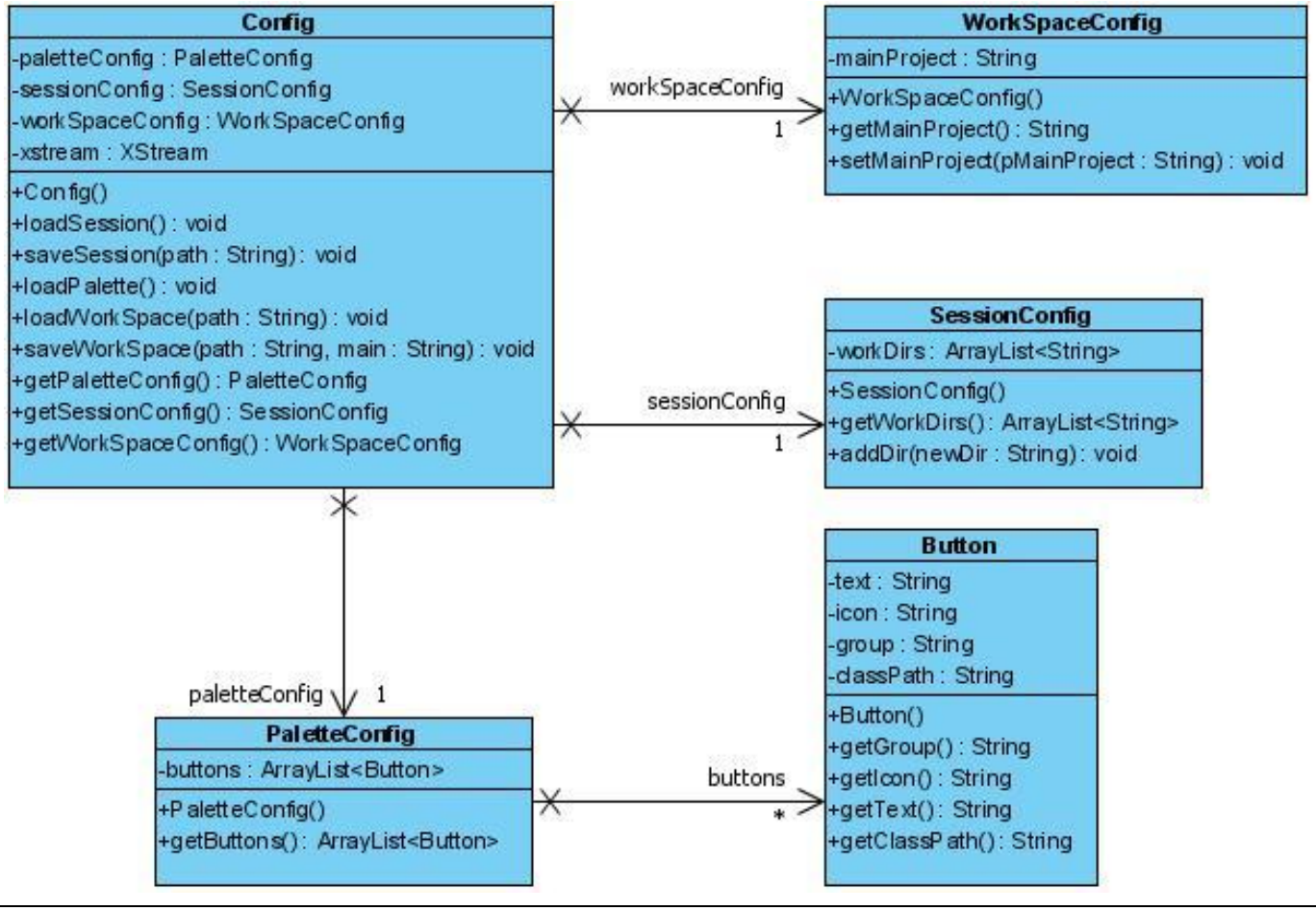


Diagrama de clases del paquete: bussines

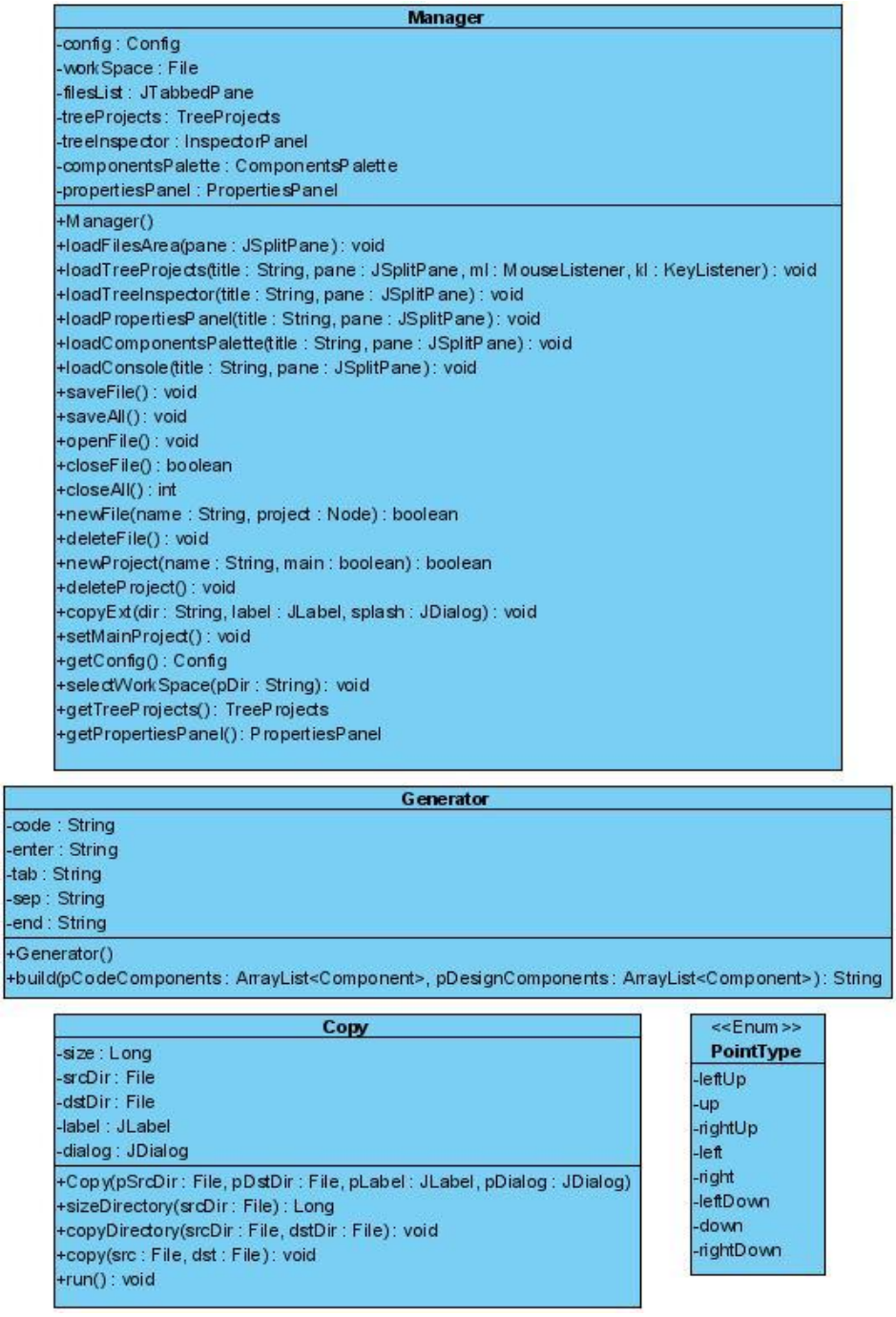


Diagrama de clases del paquete: css

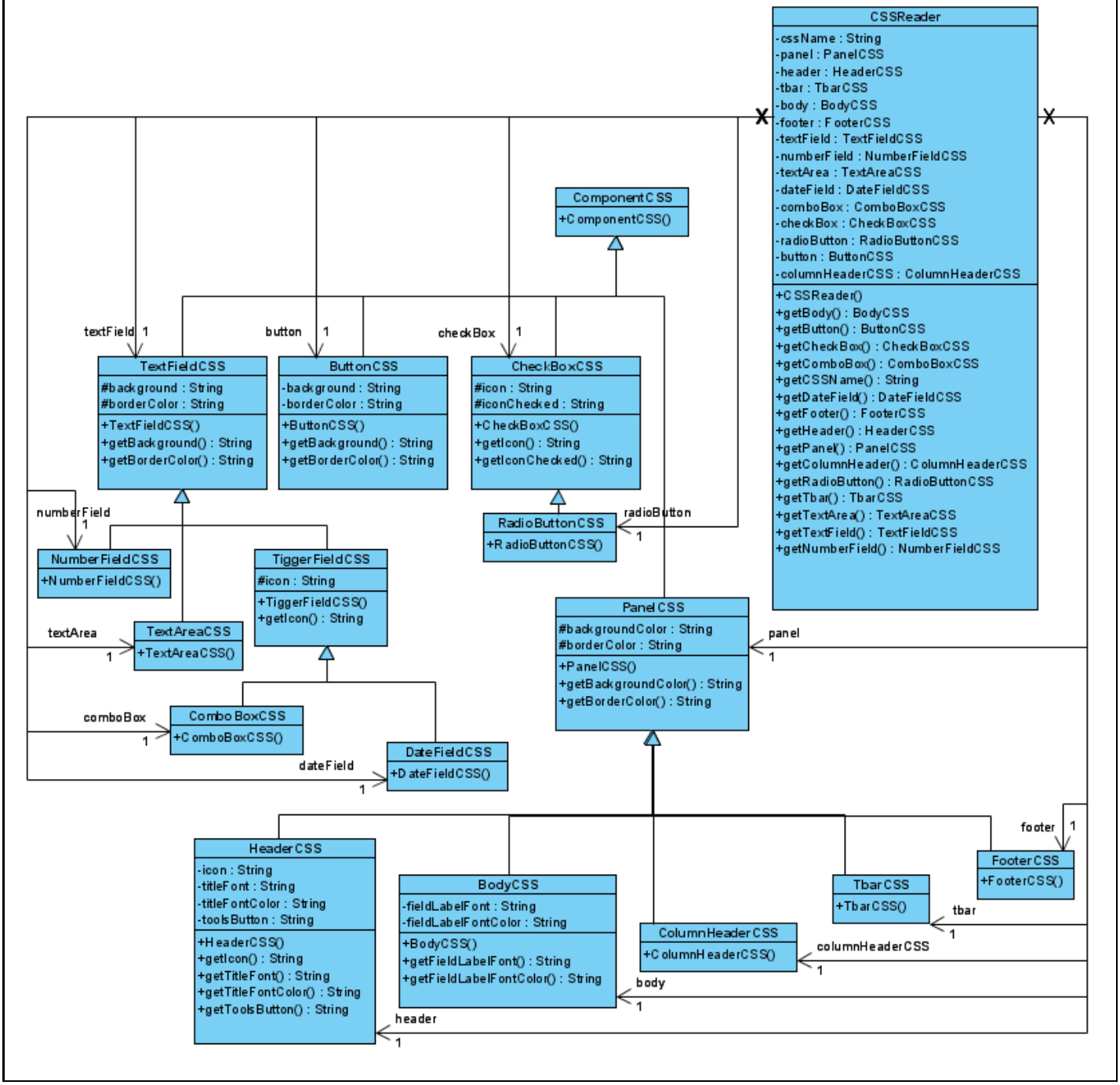


Diagrama de clases del paquete: frames

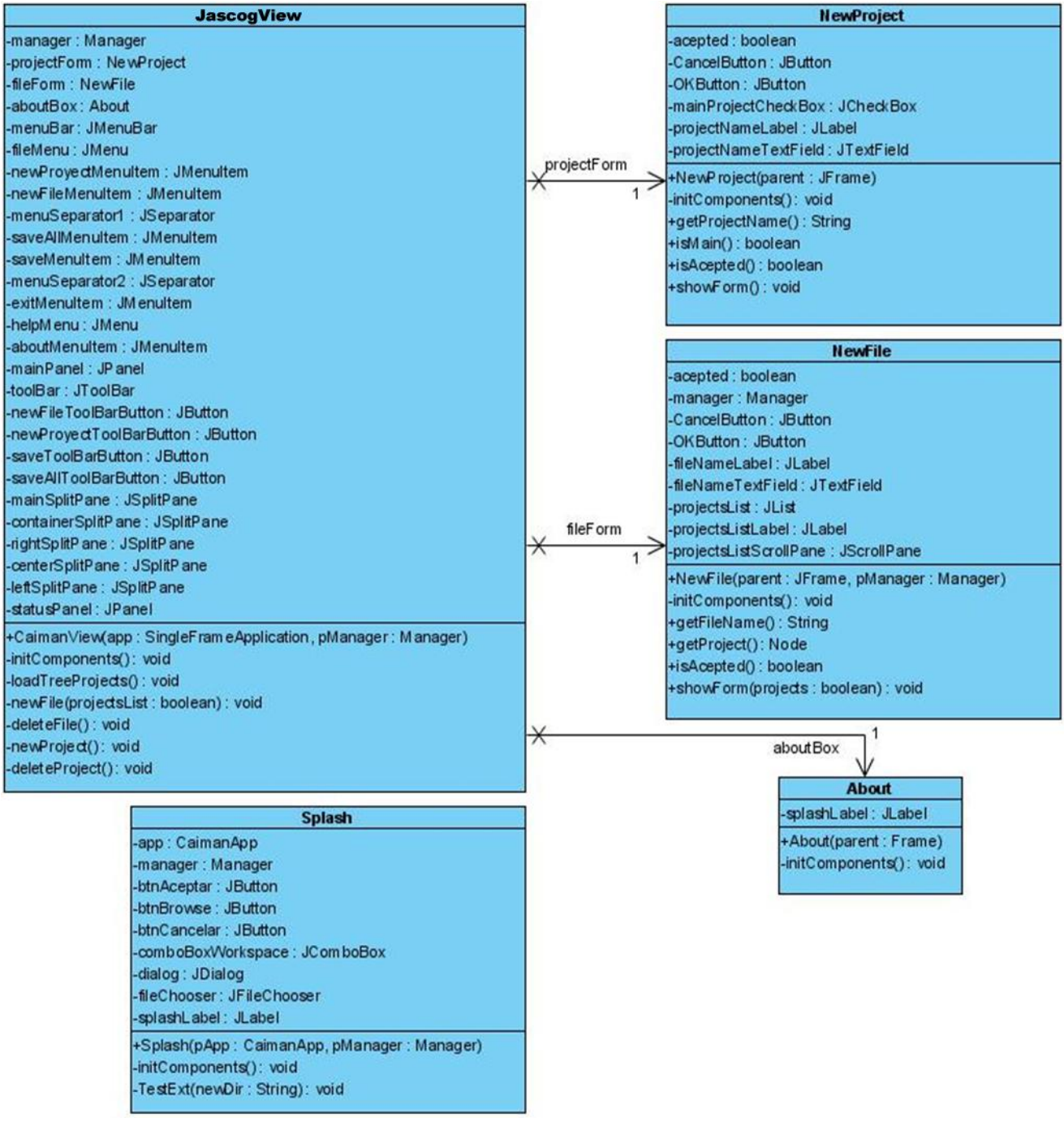


Diagrama de clases del paquete: common

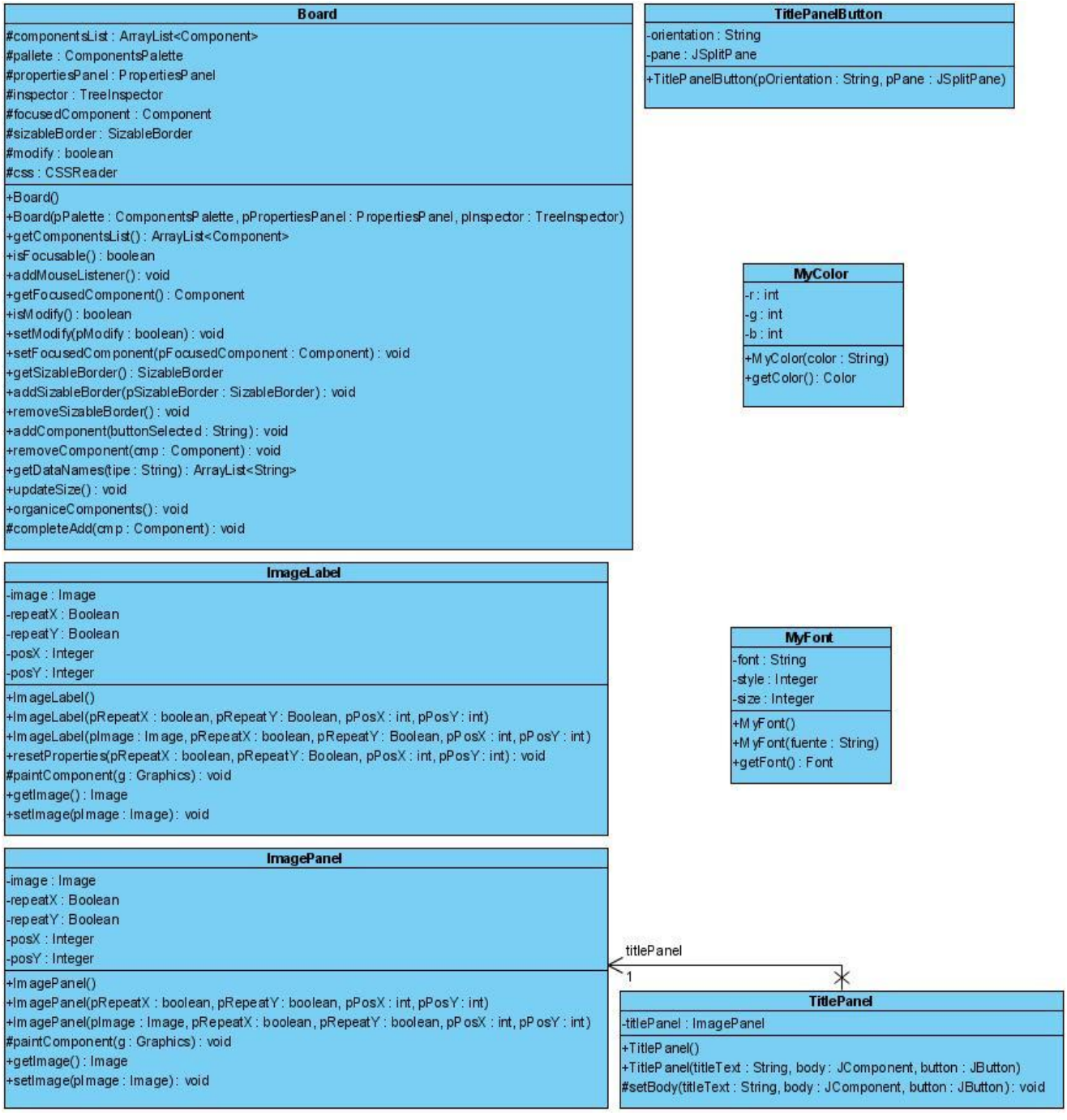


Diagrama de clases del paquete: box

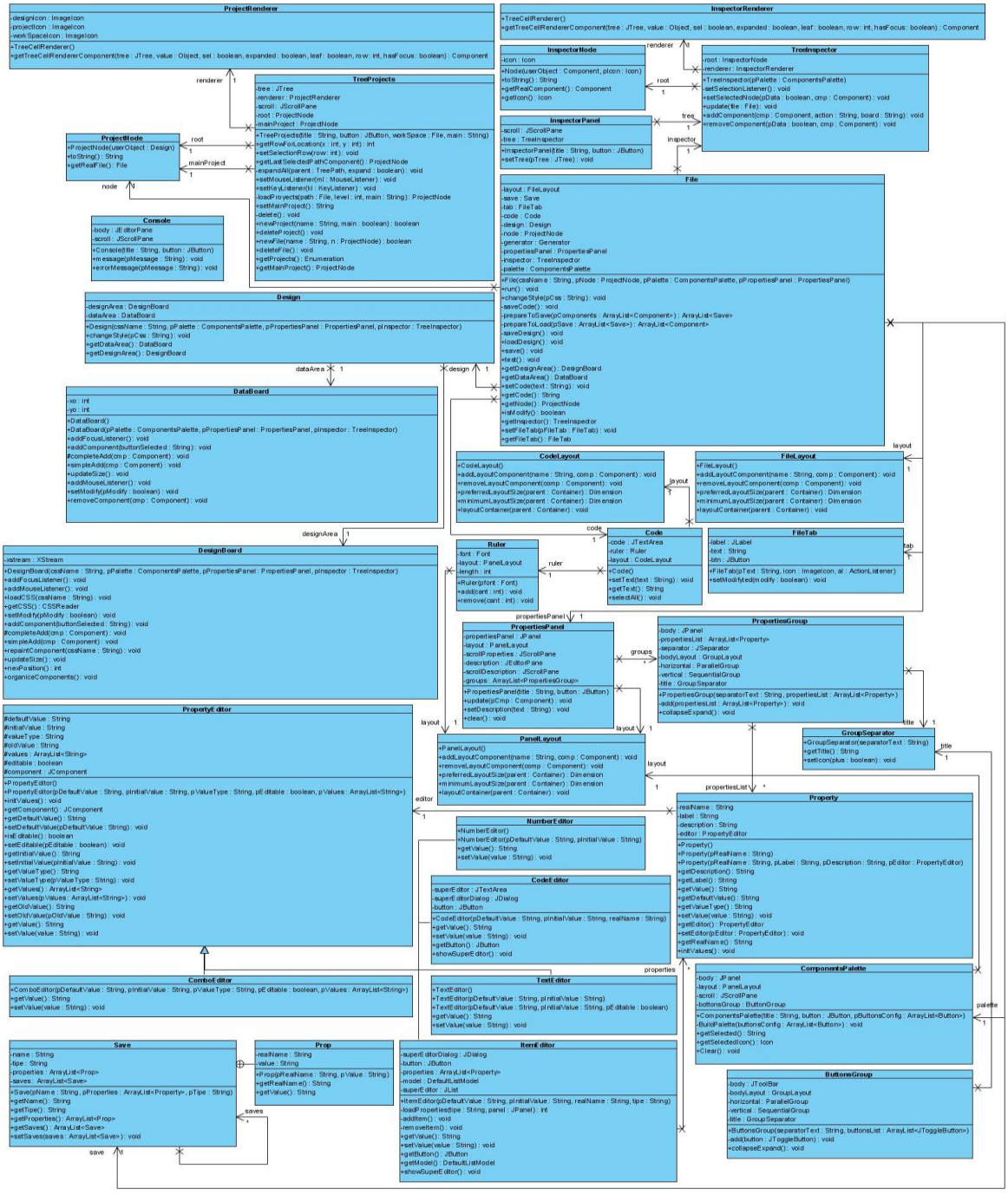
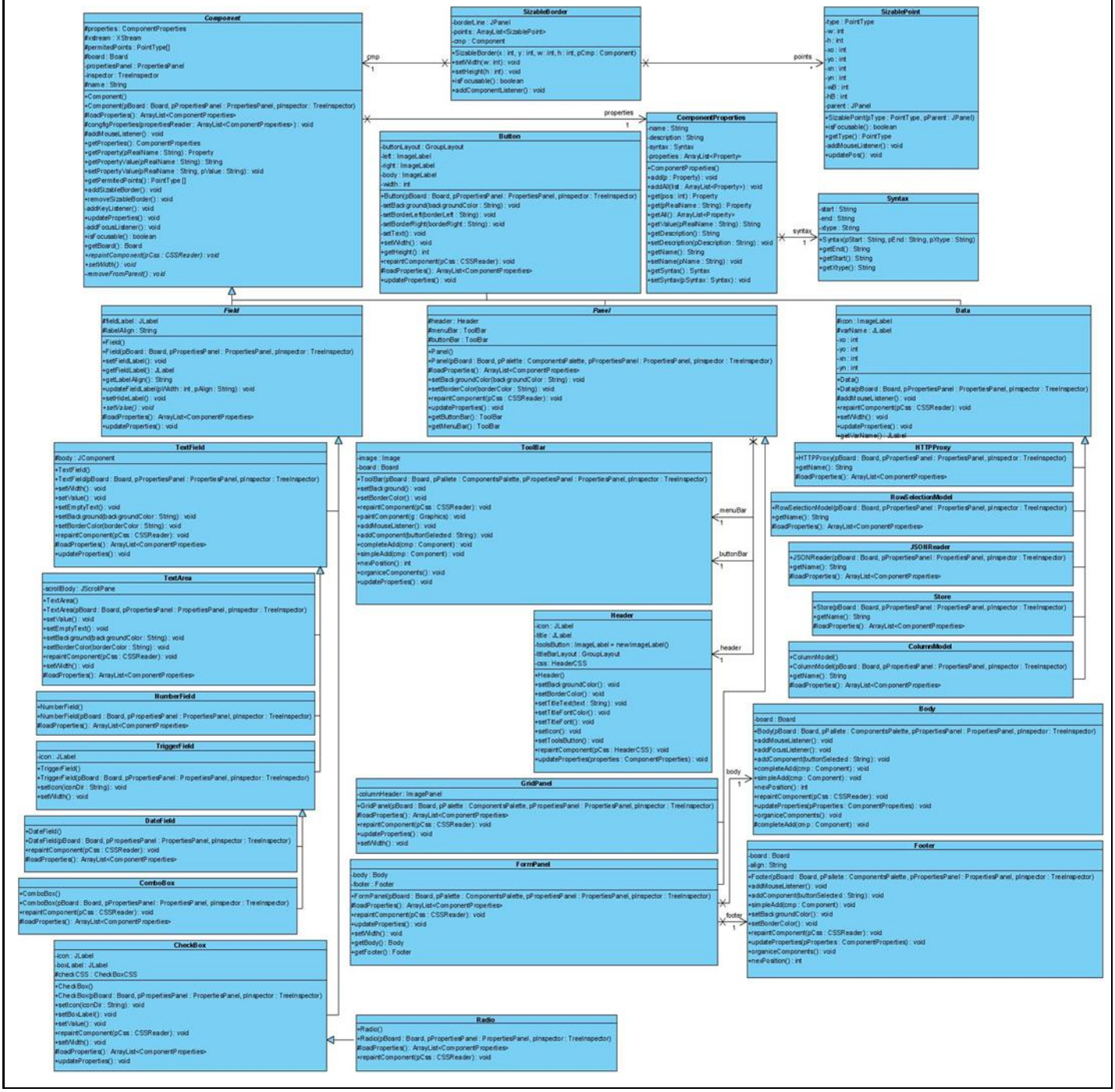


Diagrama de clases del paquete: ext



3.1.3 Validación por métricas orientadas a clases. Tamaño de clases

El tamaño de una clase puede determinarse empleando el número total de operaciones más el número de atributos. Una clase de tamaño grande tiene demasiadas responsabilidades, lo cual reduce la reutilización de la clase, al igual que se incrementa la implementación y comprobación de la clase.

Para medir el tamaño de las clases, se tienen en cuenta los aspectos siguientes:

- Total de operaciones, ya sean las de ella o las heredadas de las clases padres o interfaces que implementen.
- Cantidad de atributos, al igual que el anterior, tanto los de ella, como los de los padres.
- Promedio general de los dos anteriores para el sistema completo.

Para evaluar las métricas, son necesarios los valores de los umbrales, aspecto que es muy discutido por los especialistas en el tema, y dependen mucho del sistema que se esté diseñando. Las clases se clasifican en tres grupos, según su tamaño, los que se presentan en la siguiente tabla junto con los umbrales seleccionados para su clasificación.

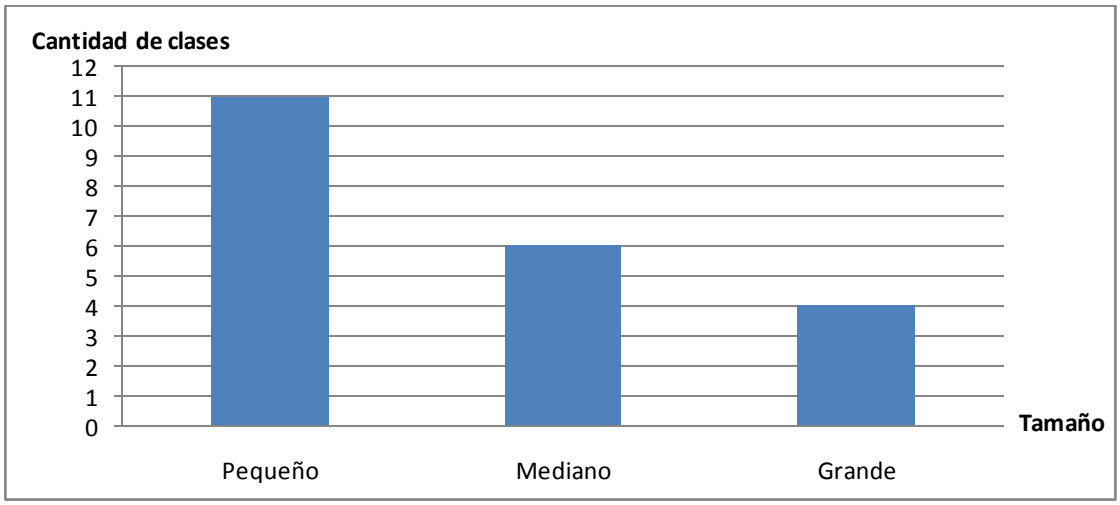
Clasificación	Valores de los Umbrales
Pequeño	<=20
Medio	>20 y<=30
Grande	>30

Esta métrica se aplico a las clases de mayor peso y prioridad obteniéndose los siguientes resultados:

Clases	No. Atributos	No. Operaciones
Manager	7	22
Generator	5	2
JascogView	28	7
NewProject	6	6
NewFile	9	6

Board	8	18
Component	7	20
Panel	10	29
Field	9	30
Data	13	27
ComponentProperties	4	13
SizableBorder	3	5
File	10	19
TreeProjects	5	16
TreeInspector	2	6
PropertiesPanel	6	4
Property	4	13
PropertyEditor	7	18
ComponentsPallete	4	5
Consle	2	3
Config	4	9

En las 21 clases presentadas se obtuvo un promedio de 7.28 atributos y 13.23 operaciones o métodos por clase. Atendiendo a los resultados arrojados por la métrica se tiene que:



3.1.4 Diagramas de interacción

Diagrama de secuencia del caso de uso: Gestionar Espacio Trabajo

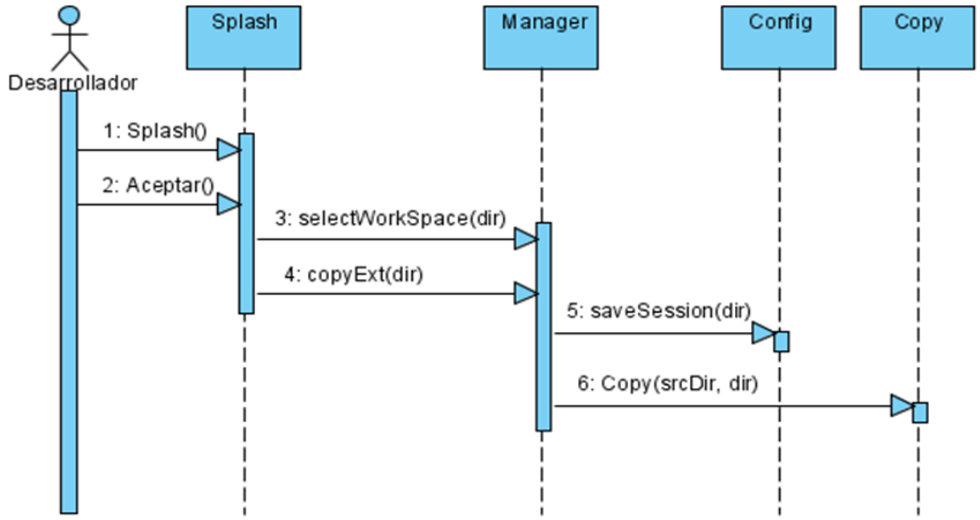


Diagrama de secuencia del escenario "Crear proyecto" del caso de uso: Gestionar Proyecto

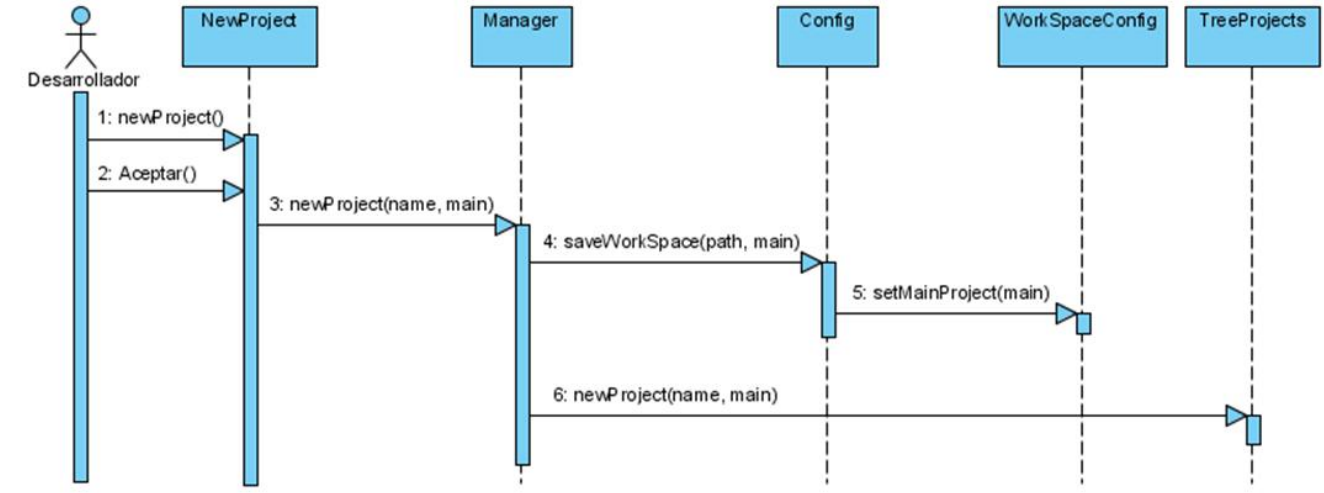


Diagrama de secuencia del escenario "Eliminar proyecto" del caso de uso: Gestionar Proyecto

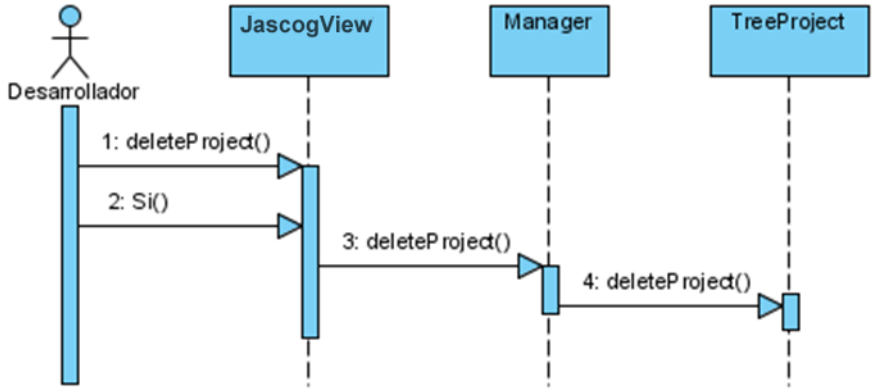


Diagrama de secuencia del escenario "Crear archivo" del caso de uso: Gestionar Archivo

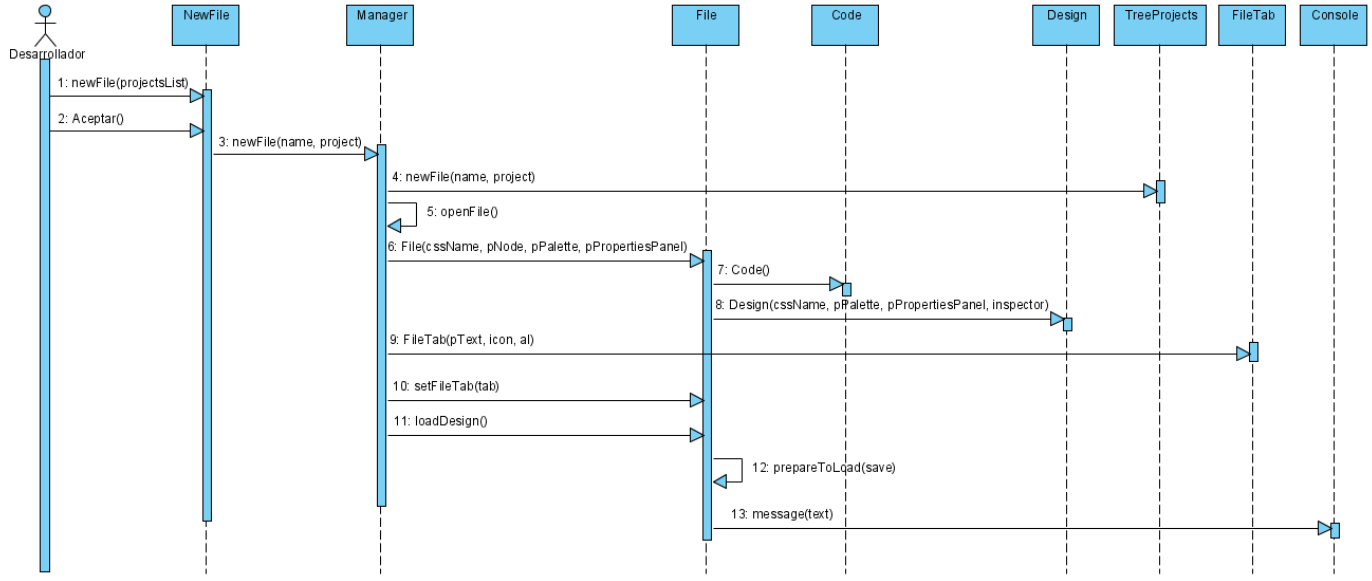


Diagrama de secuencia del escenario "Guardar archivo" del caso de uso: Gestionar Archivo

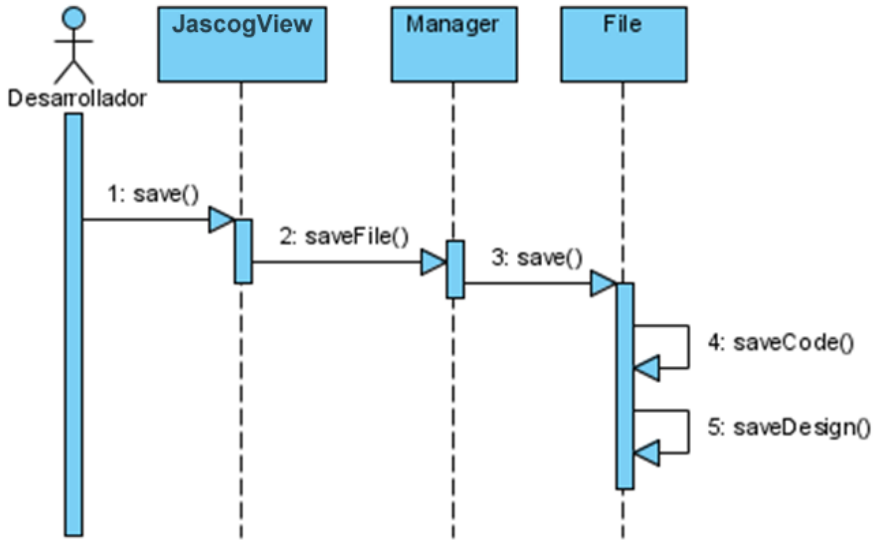


Diagrama de secuencia del escenario "Eliminar archivo" del caso de uso: Gestionar Archivo

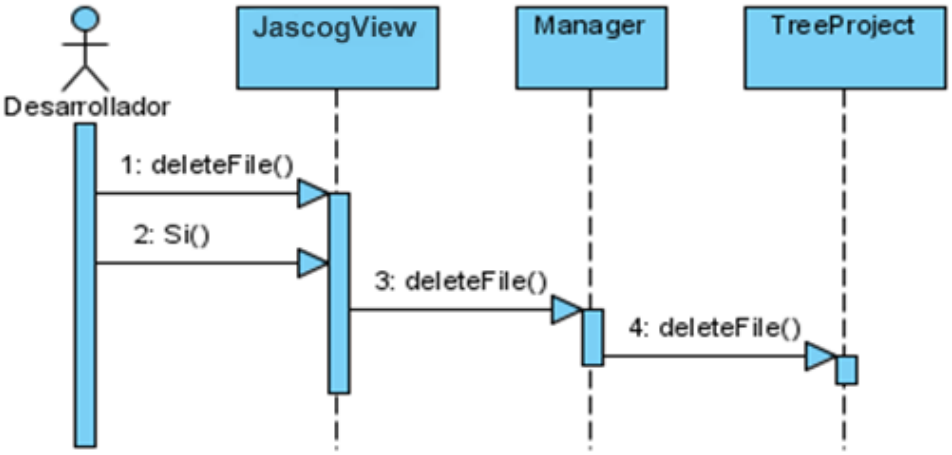


Diagrama de secuencia del caso de uso: Probar Codigo

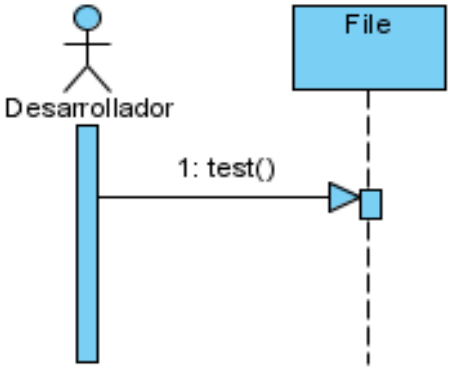


Diagrama de secuencia del caso de uso: Generar Codigo

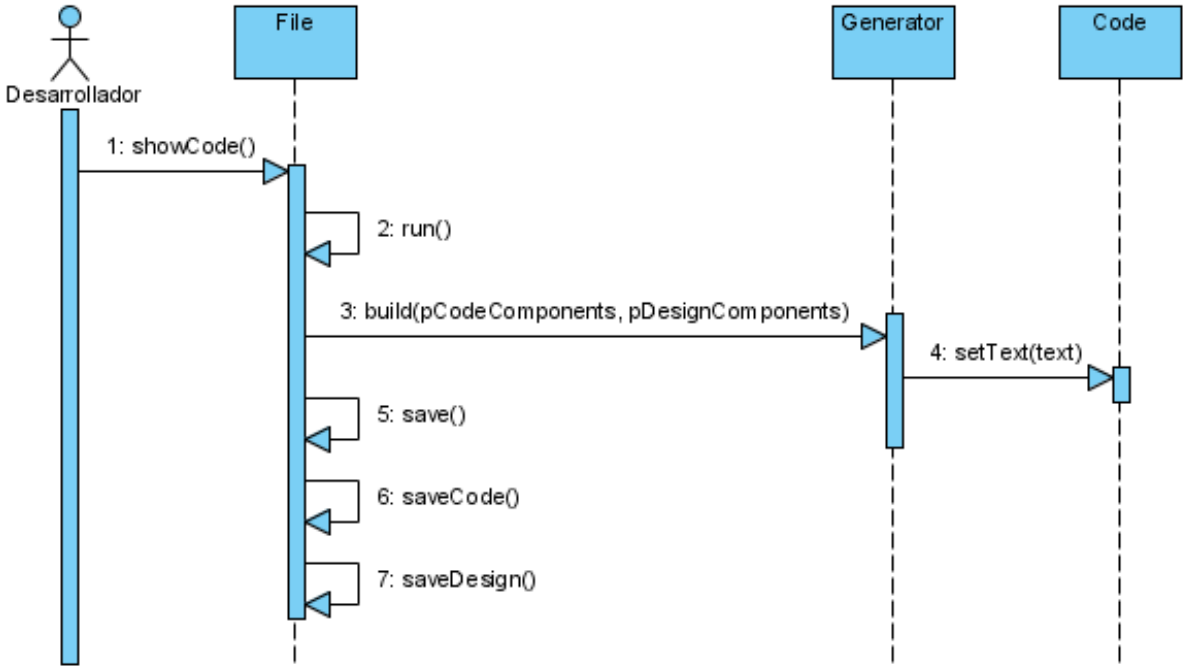


Diagrama de secuencia del escenario "Crear componente" del caso de uso: Gestionar Componente

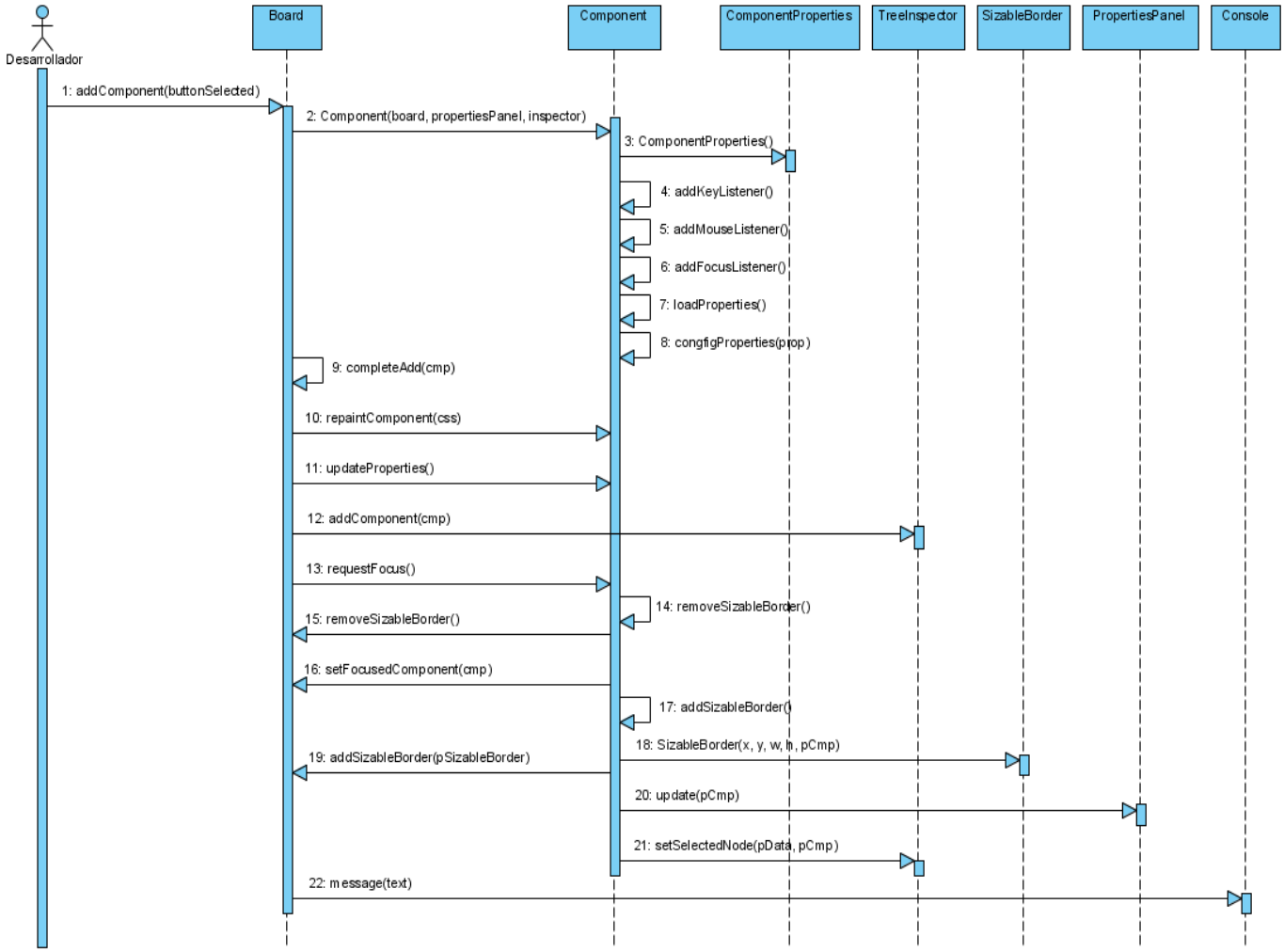


Diagrama de secuencia del escenario "Modificar componente" del caso de uso: Gestionar Componente

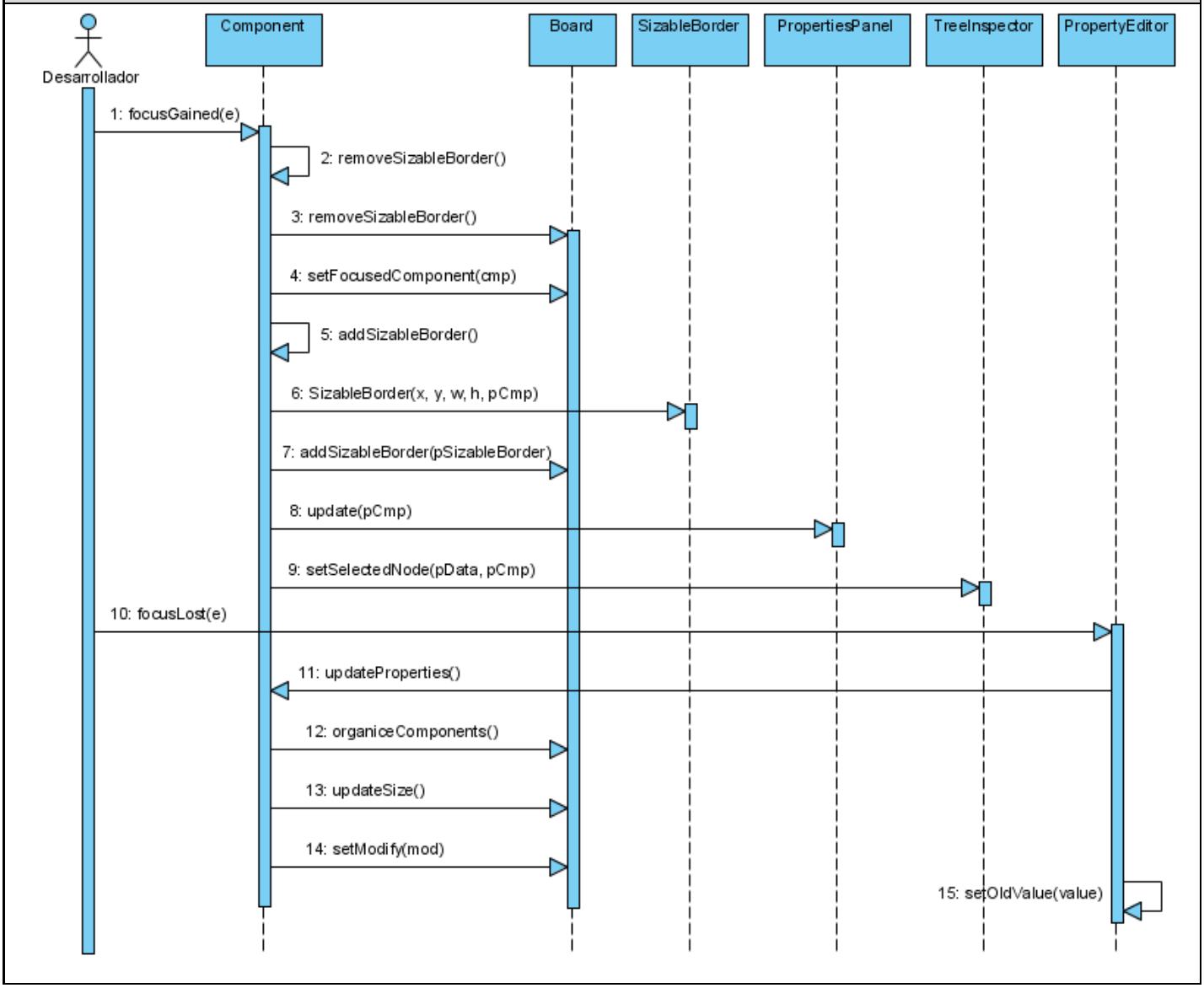
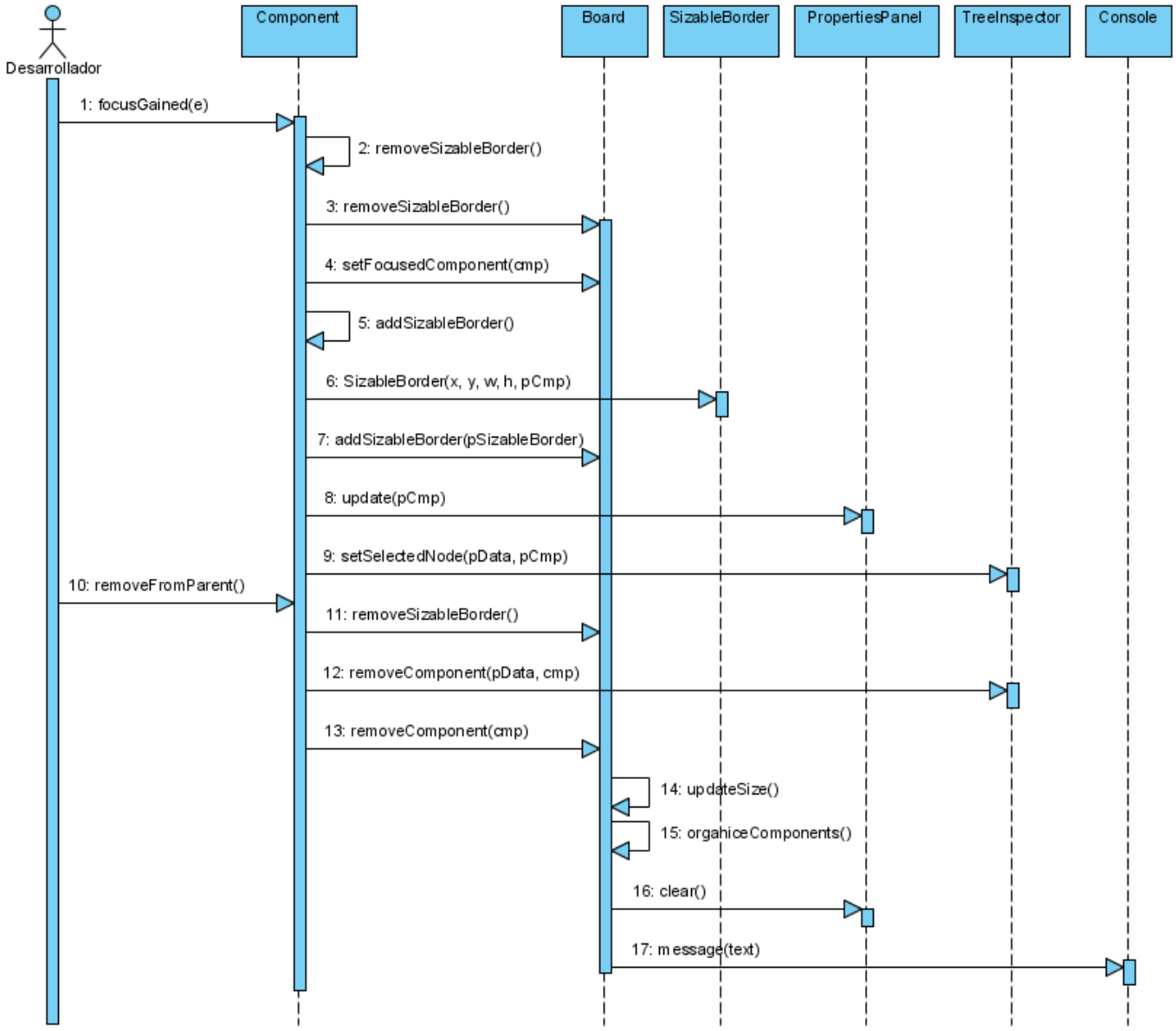
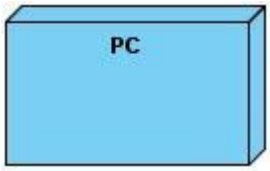


Diagrama de secuencia del escenario "Eliminar componente" del caso de uso: Gestionar Componente



3.1.5 Diagrama de despliegue

Diagrama de despliegue


3.1.6 Generación del código JavaScript

Cuando se genera el código JavaScript se crean tres archivos. El primero se llama “nombreArchivo.js” y contiene el código JavaScript generado, el segundo se llama “nombreArchivo.jcg” y contiene el diseño realizado, el tercero se llama “nombreArchivo.html” que contiene código html necesario para probar el código JavaScript generado.

Con la creación de este modelo de diseño se cuenta con la base necesaria para pasar al flujo de trabajo de implementación. Los diagramas y especificaciones de diseño propuestos constituyen una guía que puede ser fácilmente leída y comprendida. El uso de patrones de diseño ayudó a aplicar soluciones estandarizadas a problemas típicos del diseño aumentado así la calidad del diseño.

3.2 IMPLEMENTACIÓN

3.2.1 Diagramas de componentes

Diagrama de componentes del paquete: bussines

Diagrama de componentes del paquete: common

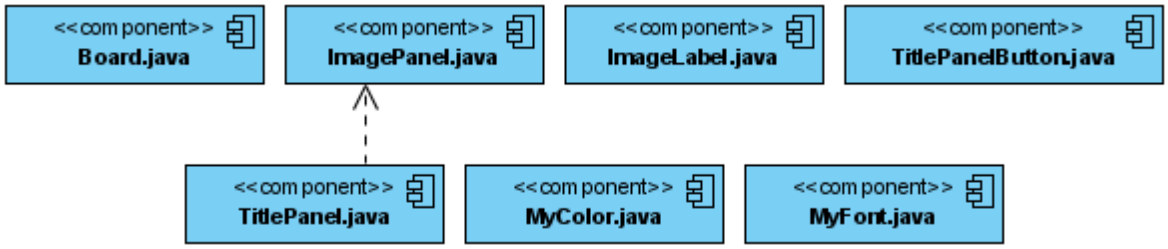


Diagrama de componentes del paquete: config

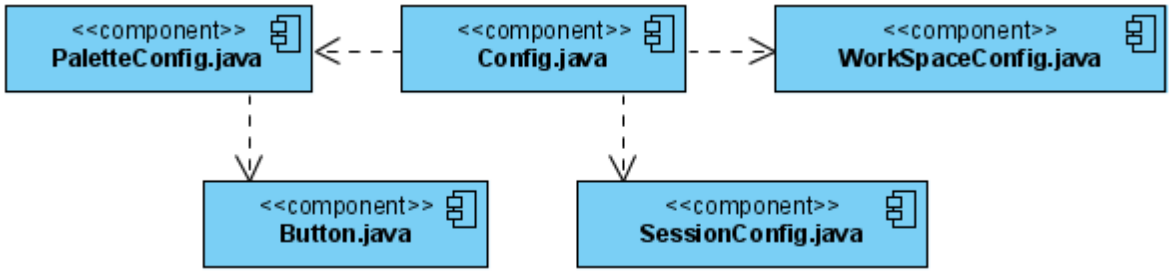


Diagrama de componentes del paquete: css

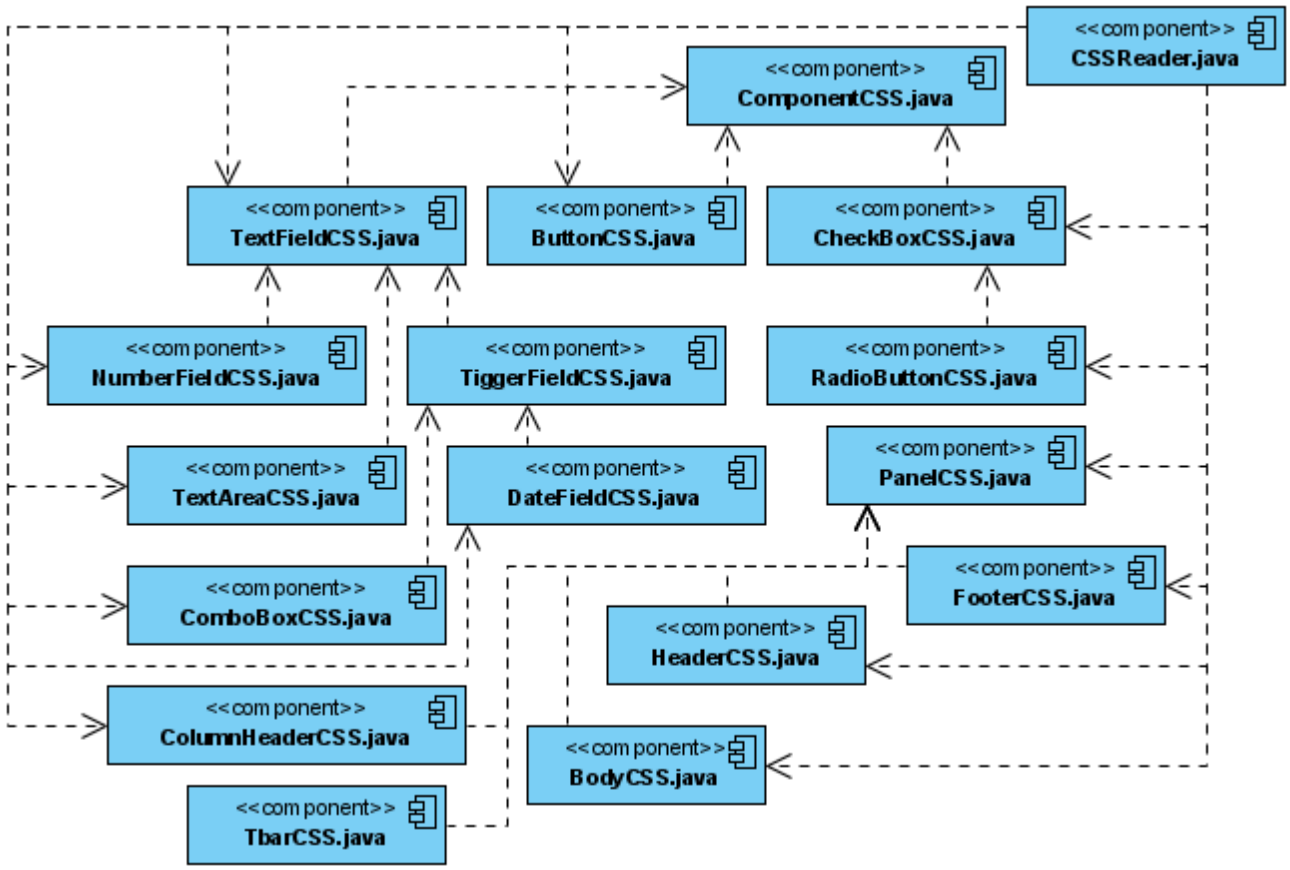


Diagrama de componentes del paquete: frames

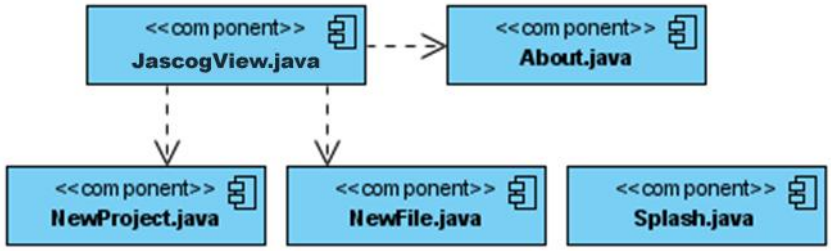


Diagrama de componentes del paquete: box

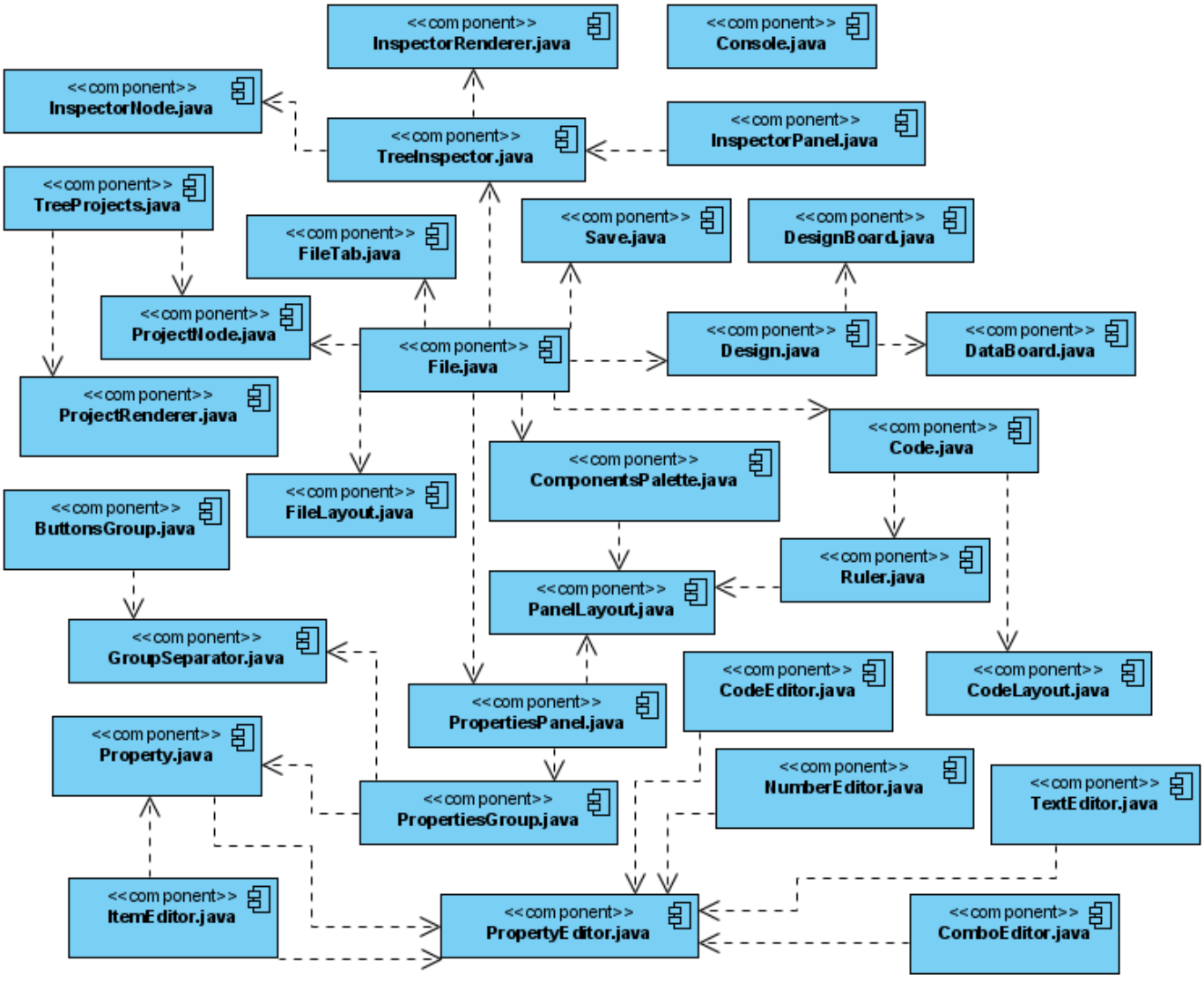


Diagrama de componentes del paquete: ext

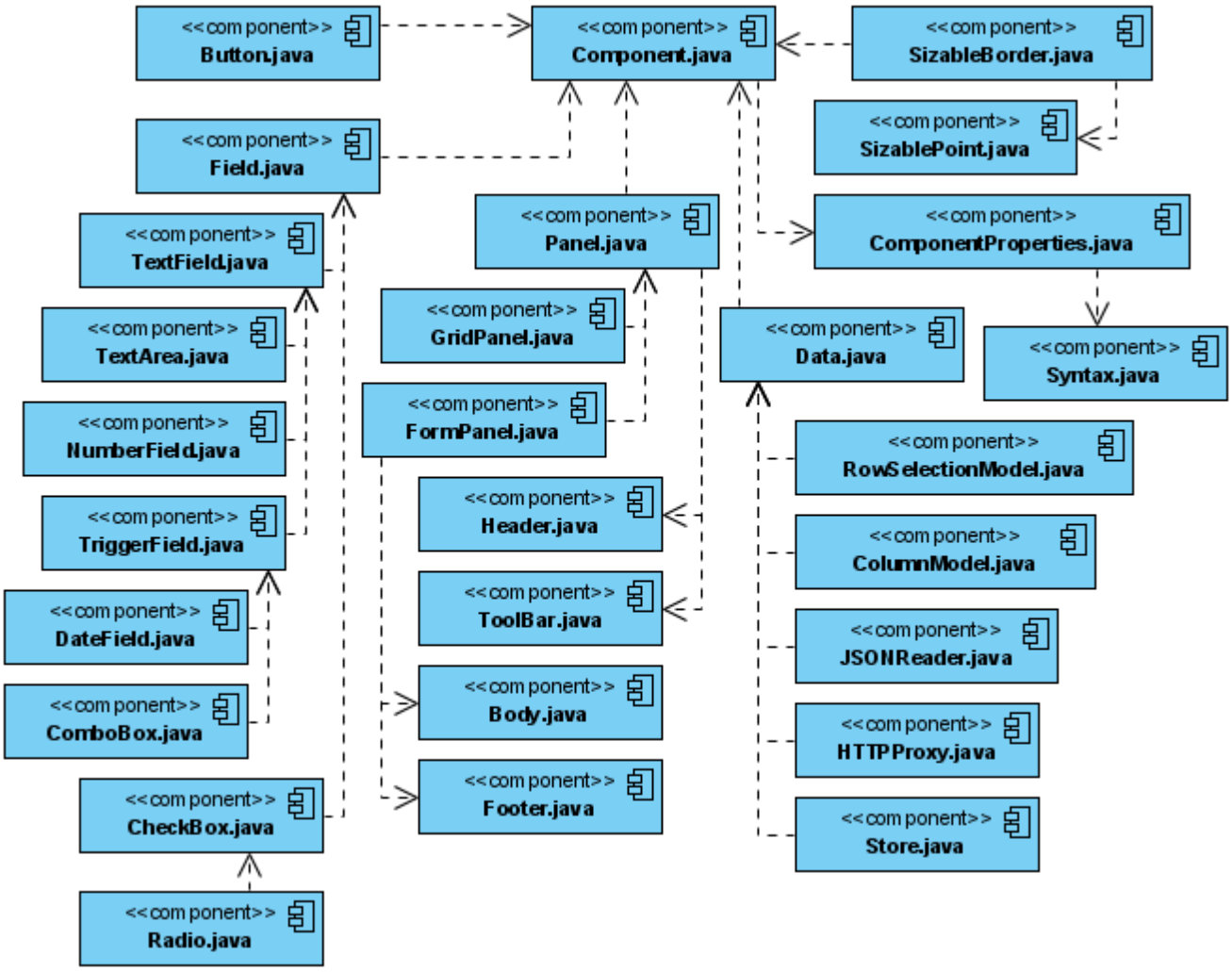
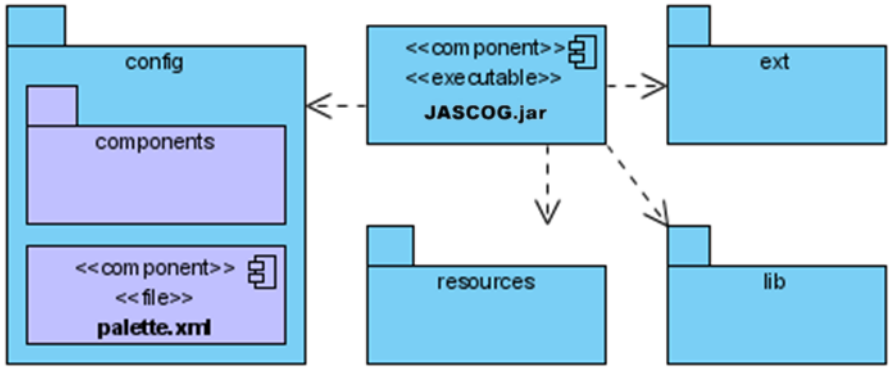


Diagrama de componentes de la aplicación



Conclusiones del capítulo

Este capítulo se centró principalmente en la realización del modelo de diseño del sistema, así como en la elaboración de los diagramas de clases y de interacción de los casos de uso detallados en el capítulo anterior.

CAPÍTULO 4: PRUEBA

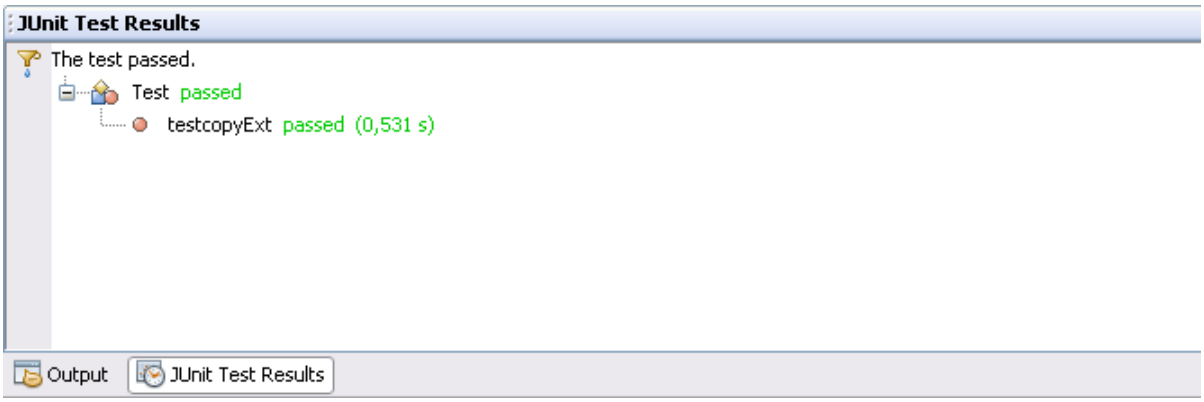
4.1 Pruebas de unidad

Cuando se implementan componentes de software, siempre resulta recomendable comprobar que el código que se ha escrito funcione correctamente. Para ello, se hace imperioso que se implementen pruebas que lo verifiquen; el modo que tienen muchos desarrolladores para respaldar la fiabilidad de los sistemas es la verificación empírica (no garantiza que no existan errores), siempre y cuando, se verifiquen detalladamente cada una de las partes que conforman al sistema, se desarrollen pruebas al código escrito, que aseveren la funcionalidad correcta de dicha codificación, y le permita al sistema corresponder con el funcionamiento esperado por los clientes.

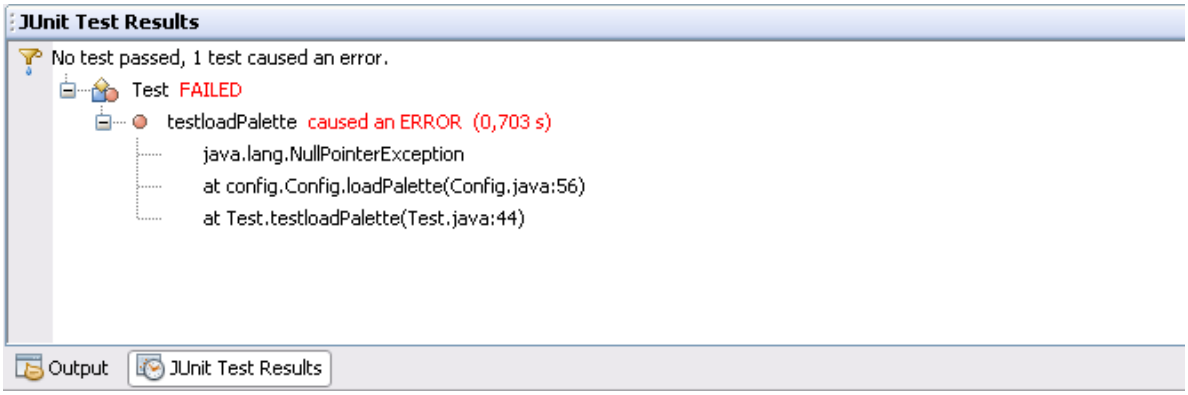
Una buena práctica resultaría llevar paralelamente el proceso de desarrollo y el proceso de evaluación o comprobación de los elementos de implementación que se van generando. Encontrar errores es una tarea desafiante para cualquier desarrollador, pero existen herramientas que proporcionan una manera sencilla, rápida y elegante para escribir pruebas y validarlas automáticamente. JUnit es precisamente una de estas herramientas, framework que se ha convertido en el estándar universal para realizar pruebas de unidad en Java.

JUnit tiene una manera muy peculiar para visualizar los reportes de las pruebas realizadas:

Prueba satisfactoria con JUnit



Prueba no satisfactoria con JUnit



4.1.1 Utilizacion de JUnit

La utilización del framework comienza con la elaboración de la clase donde se implementaran los casos de prueba la cual se muestra a continuación.

Se crea una clase heredando de TestCase, lo que garantiza que el framework compruebe nuestra clase.

```
public class Test extends TestCase {
```

Se declaran los atributos necesarios para realizar las pruebas.

```
private Manager manager;
private TreeProjects treeProjects;
private DesignBoard board;
private Generator generator;
.
.
.
```

Se sobrescribe el método setUp(), el cual se ejecuta antes de los casos de prueba y se usa para inicializar los objetos que se necesiten.

```
@Override
protected void setUp() throws Exception {
    super.setUp();
    manager = new Manager();
    board = new DesignBoard("", manager.getComponentsPalette(), manager.getP ...
    treeProjects = new TreeProjects("Proyectos", new TitlePanelButton("left" ...
    generator = new Generator();
}
```

Se crean los casos de prueba para lo cual es necesario crear los métodos iniciando su nombre con la palabra test seguido del nombre del caso de prueba.

```
public void testXXX() throws Exception {
    .
    .
    .
}
```

4.1.2 Métodos de JUnit

JUnit brinda una serie de métodos para realizar las pruebas.

Método	Descripción
assertTrue	Acepta el resultado si la condición es verdadera.
assertFalse	Acepta el resultado si la condición es falsa.
assertEquals	Acepta el resultado si los objetos son iguales.
asserNotNull	Acepta el resultado si el objeto no es null.
assertNull	Acepta el resultado si el objeto es null.
asserSame	Acepta el resultado si las dos referencias apuntan al mismo objeto.
assertNotSame	Acepta el resultado si las dos referencias no apuntan al mismo objeto.

4.1.3 Pruebas realizadas

La siguiente planilla recoge los principales pasos y aspectos que se tuvieron en cuenta a la hora de de realizar las pruebas sobre los distintos métodos.

Método	Objetivo	Resultados esperados	Resultados obtenidos
copyExt	Comprobar que se copia la librería ExtJS para el directorio seleccionado.	Se espera que la librería ExtJS este copiada en el directorio seleccionado.	La librería ExtJS fue copiada satisfactoriamente.
deleteDirectory	Comprobar que se elimina el directorio seleccionado.	Se espera que el directorio seleccionado sea eliminado.	El directorio seleccionado fue eliminado satisfactoriamente.
newProject	Comprobar que se cree un nuevo proyecto en el espacio de trabajo.	Se espera que se cree un nuevo proyecto en el espacio de trabajo.	El proyecto fue creado satisfactoriamente.
newFile	Comprobar que se cree un nuevo archivo en el proyecto seleccionado.	Se espera que se cree un nuevo archivo en el proyecto seleccionado.	El archivo fue creado satisfactoriamente.

loadProyects	Comprobar que se carguen los proyectos existentes en el espacio de trabajo.	Se espera que los proyect existentes en el espacio de trabajo sean cargados.	Los proyectos fueron cargados satisfactoriamente.
loadSession	Comprobar que la lista de espacios de trabajo seleccionados anteriormente, sea cargada.	Se espera que los espacios de trabajos seleccionados anteriormente sean cargados.	La lista de espacios de trabajo fue cargada satisfactoriamente.
loadPalette	Comprobar que se carga la paleta de componentes.	Se espera que la paleta de componentes se cargue.	La paleta de componentes fue cargada satisfactoriamente.
message	Comprobar que se adiciona un nuevo mensaje en la consola.	Se espera que se adicione un nuevo mensaje en la consola.	El mensaje fue adicionado satisfactoriamente.
update	Comprobar que se actualicen las propiedades del panel de propiedades.	Se espera que el panel de propiedades muestre la nueva lista de propiedades.	La lista de propiedades fue mostrada satisfactoriamente.
saveCode	Comprobar que el código generado sea guardado.	Se espera que el código generado sea guardado en un fichero .js.	El código fue guardado satisfactoriamente.
saveDesign	Comprobar que el diseño sea guardado.	Se espera que el diseño sea guardado en un fichero .jcg.	El diseño fue guardado satisfactoriamente.
loadDesign	Comprobar que el diseño sea cargado.	Se espera que el diseño guardado en el archivo .jcg sea cargado.	El diseño fue cargado satisfactoriamente.
test	Generar un fichero .html	Se espera que se genere	La página fue generada

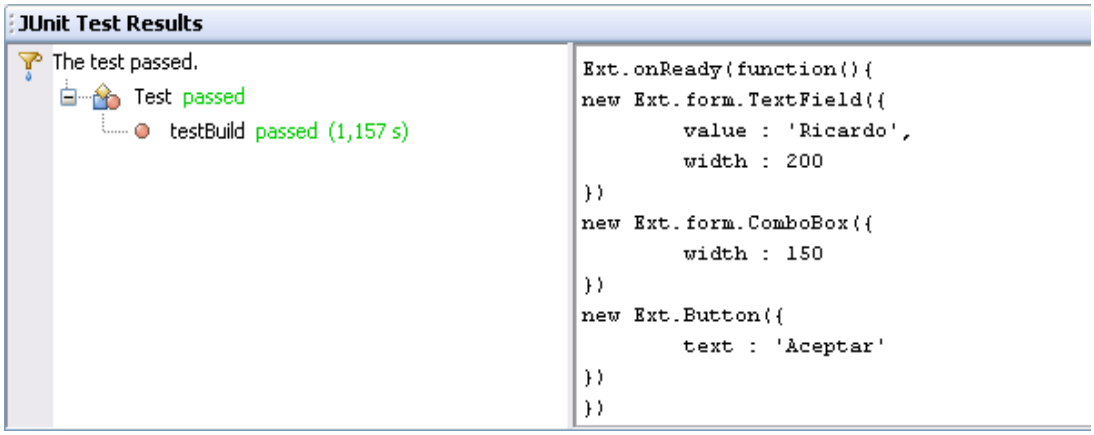
	para probar el código generado.	una página .html y sea ejecutada con un navegador web para probar el código generado.	y ejecutada satisfactoriamente.
addComponent	Adicionar en el área de diseño el componente seleccionado en la paleta de componentes.	Se espera que el componente seleccionado en la paleta de componentes, sea creado y adicionado en el área de diseño.	El componente fue creado y adicionado satisfactoriamente.
removeComponent	Eliminar el componente seleccionado.	Se espera que el componente seleccionado sea eliminado.	El componente fue eliminado satisfactoriamente.
loadProperties	Cargar las propiedades del componente creado.	Se espera que al crear un componente, este cargue sus propiedades.	El componente cargo sus propiedades satisfactoriamente.
configProperties	Crear el editor de cada propiedad.	Se espera que se cree el editor de cada propiedad cargada.	Los editores fueron creados satisfactoriamente.
addSizableBorder	Adicionar un borde al componente seleccionado.	Se espera que al seleccionar un componente aparezca un borde para poder resisarlo.	El borde fue adicionado satisfactoriamente.
removeSizableBorder	Eliminar el borde del componente seleccionado.	Se espera que el borde del componente seleccionado se eliminado.	El borde fue eliminado satisfactoriamente.

build	Convertir el diseño en código JavaScript.	Se espera que se genere el código JavaScript correspondiente al diseño.	El código fue generado satisfactoriamente.

4.1.4 Ejemplos de pruebas

Método	build

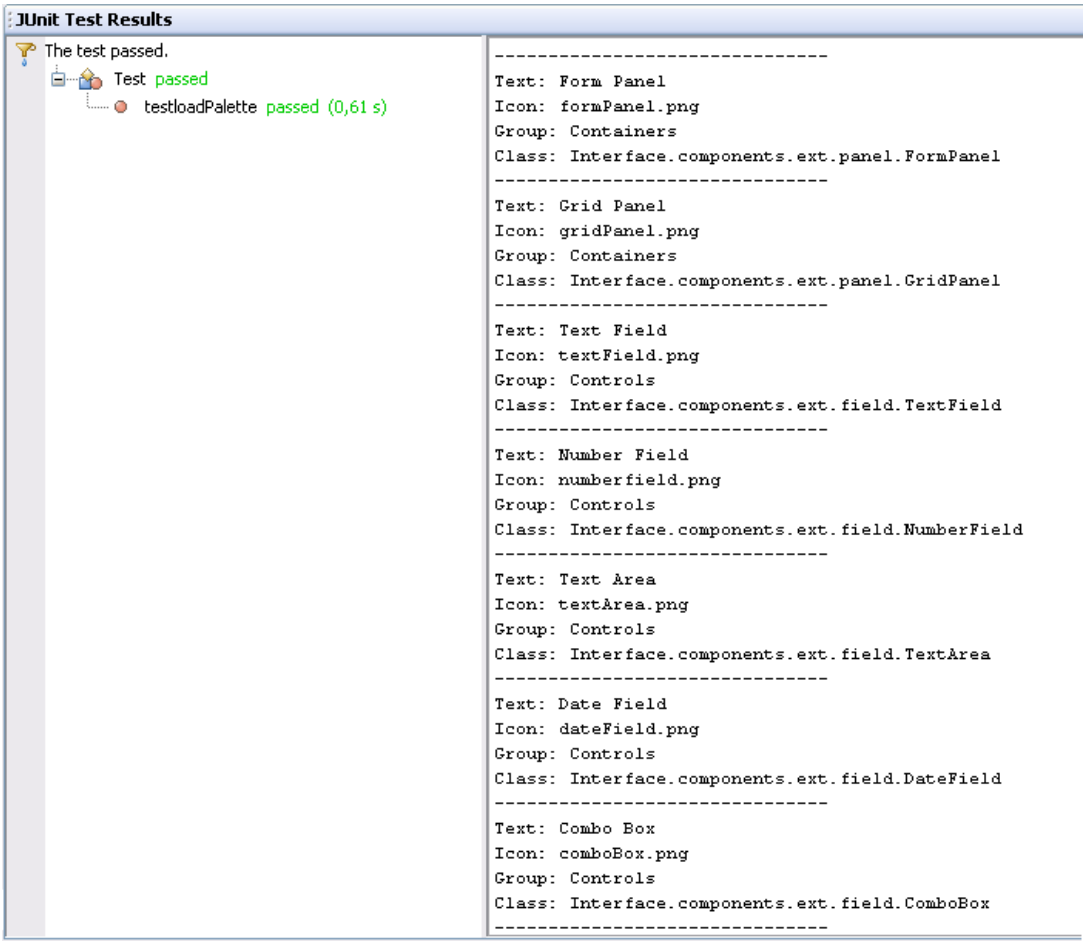
Implementación	`public void testBuild() throws Exception {`
	`board.addComponent("Interface.components.ext.field.TextField");`
	`board.getFocusedComponent().setPropertyValue("width", "200");`
	`board.getFocusedComponent().setPropertyValue("value", "Ricardo");`
	
	`board.addComponent("Interface.components.ext.field.ComboBox");`
	
	`board.addComponent("Interface.components.ext.button.Button");`
	`board.getFocusedComponent().setPropertyValue("text", "Aceptar");`
	
	`String code = generator.build(new ArrayList<Component>(), board.getComponentsList());`
	`System.out.println(code);`
	`assertNotNull(code);`
	`}`

Resultado	
Método	loadPalette

Implementación

```
public void testloadPalette() throws Exception {
    manager.getConfig().loadPalette();
    ArrayList<Button> list = manager.getConfig().getPaletteConfig().getButtons();
    for (int i = 0; i < list.size(); i++) {
        System.out.println("--------------------------------");
        System.out.println("Text: "+list.get(i).getText());
        System.out.println("Icon: "+list.get(i).getIcon());
        System.out.println("Group: "+list.get(i).getGroup());
        System.out.println("Class: "+list.get(i).getClassPath());
    }
    assertNotNull(list);
}
```

Resultado



4.2 Prueba Piloto

Las pruebas piloto son utilizadas para validar si una aplicación desarrollada cumple o no con su objetivo y con sus requisitos. Estas pruebas se realizan seleccionando una población real de los usuarios finales de la aplicación y se lleva a cabo un procedimiento en paralelo, lo cual significa que se realiza el proceso como se realizaba antes de la introducción del nuevo sistema y paralelo a esto se realizan las mismas actividades haciendo uso del la aplicación, al final se comparan los resultados obtenidos y se evalúa la aplicación.

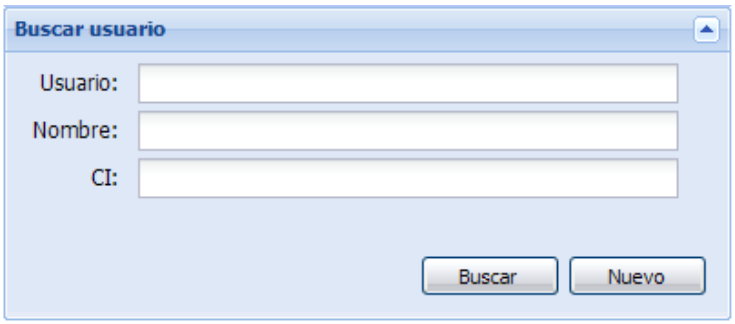
Para realizar la prueba piloto se seleccionaron siete especialistas, los cuales son los desarrolladores de la interfaz de usuario del proyecto SINAPSIS, estos especialistas cuentan con experiencia sobre el desarrollo con la librería ExtJS debido a que algunos formaron parte del equipo de desarrollo del proyecto CCV y otros han recibido la capacitación necesaria para asumir este rol dentro del proyecto.

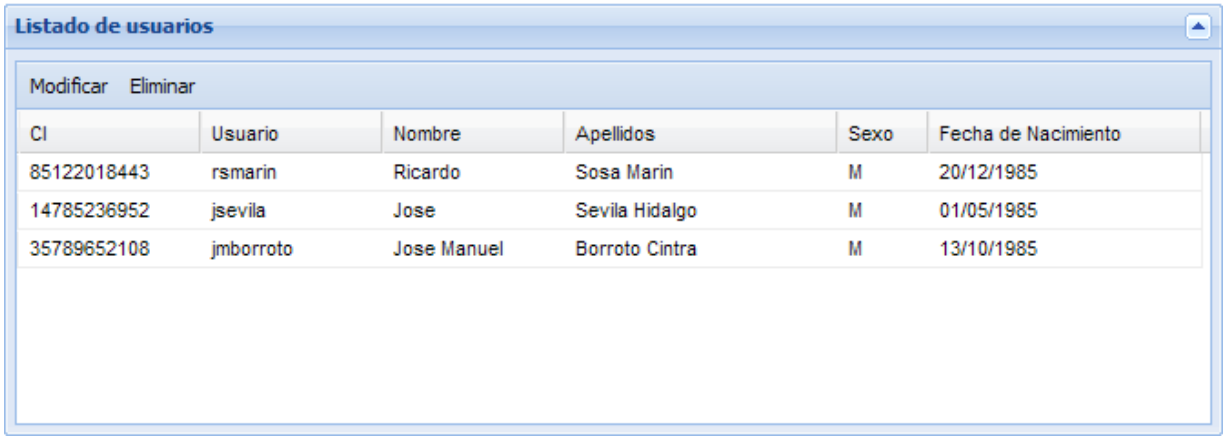
Especialista	Tiempo de experiencia
Especialista #1	3 años
Especialista #2	2 años
Especialista #3	2 años
Especialista #4	1 año
Especialista #5	1 año
Especialista #6	5 meses
Especialista #7	4 meses

Para realizar la prueba piloto se decide desarrollar las siguientes interfaces de usuario las cuales están categorizadas con una complejidad media:

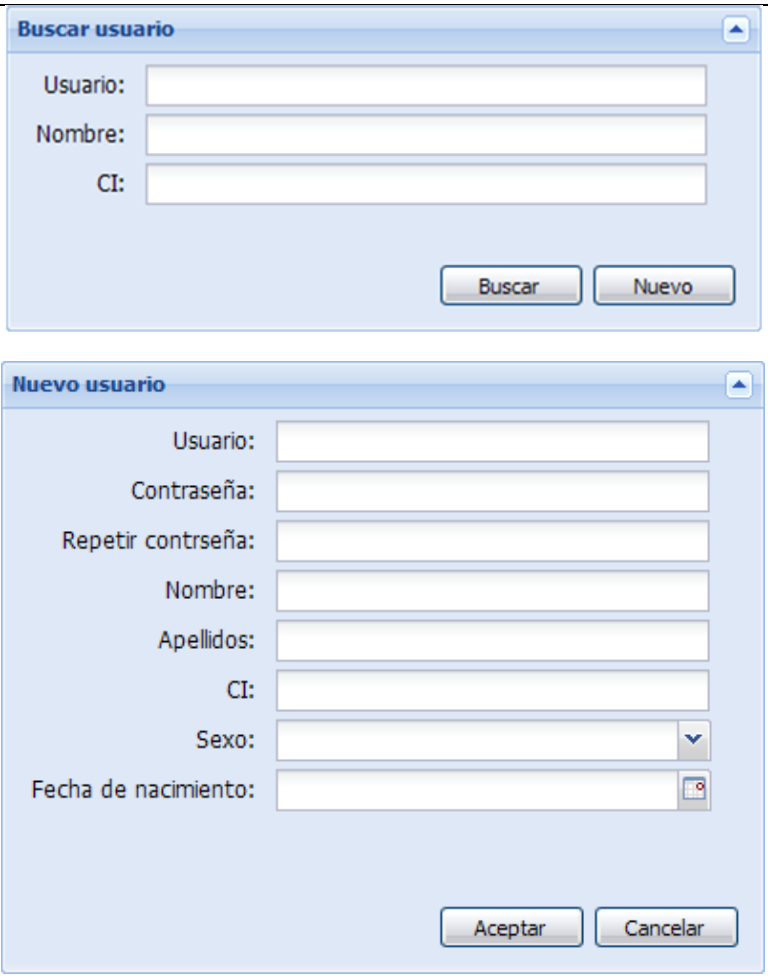
Caso de Uso:	**Gestionar Usuario**
Resumen:	El caso de uso consiste en buscar, crear o eliminar un usuario.
Flujo Normal de Eventos	

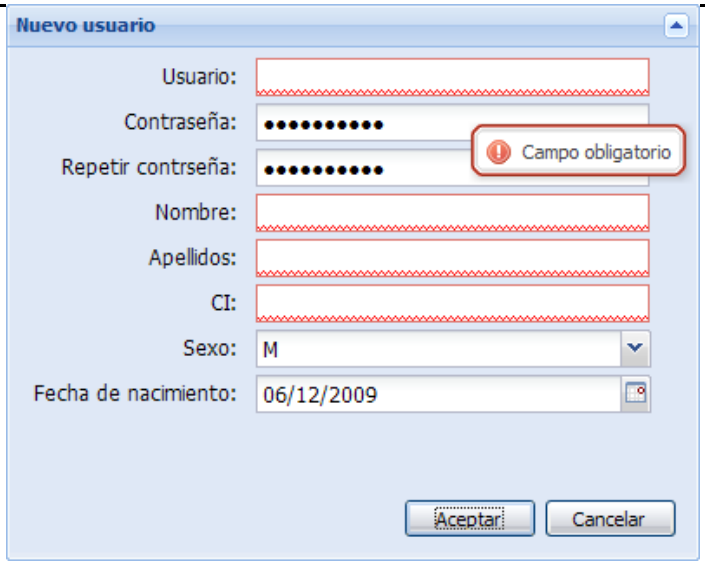
<table>
<tr><th colspan="2">Sección “Buscar usuario”</th></tr>
<tr><th>Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td></td><td>1. El sistema muestra una interfaz para buscar usuarios a partir de los siguientes datos:
➢ Usuario
➢ Nombre
➢ CI</td></tr>
<tr><td>2. El actor introduce los datos necesarios y selecciona la opción “Buscar”.</td><td>3. El sistema muestra un listado de usuarios que cumplen con el criterio de búsqueda. Los datos a mostrar son:
➢ CI
➢ Usuario
➢ Nombre
➢ Apellidos
➢ Sexo
➢ Fecha de Nacimiento
4. Terminado así el caso de uso</td></tr>
<tr><td colspan="2">Prototipo de Interfaz</td></tr>
</table>

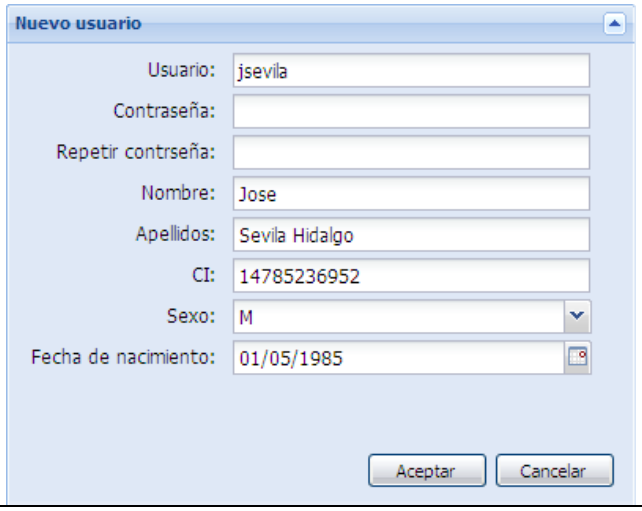


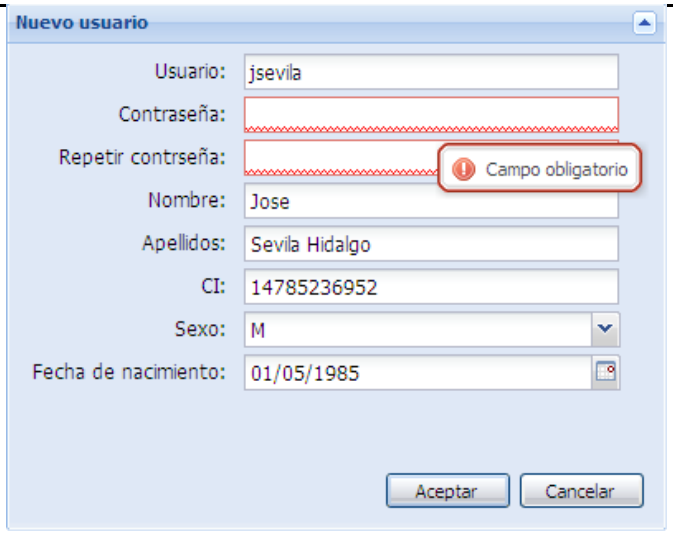


Flujo Normal de Eventos	
Sección "Crear usuario"	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción "Nuevo".	2. El sistema muestra una interfaz para crear nuevos usuarios a partir de los siguientes datos: ➢ Usuario ➢ Contraseña ➢ Repetir contraseña ➢ Nombre ➢ Apellidos

	➢ CI ➢ Sexo ➢ Fecha de Nacimiento
3. El actor introduce los datos necesarios y selecciona la opción “Aceptar”.	4. El sistema verifica que los campos obligatorios no estén vacios, crea el nuevo usuario y regresa al formulario para buscar usuario. Terminando así el caso de uso.
Prototipo de Interfaz	
	

Flujos alternos	
Flujo alterno al paso 4 "Campos vacios"	
	5. a El sistema muestra un mensaje de error. Terminando así el caso de uso.
Prototipo de Interfaz	
	
Flujo Normal de Eventos	
Sección "Modificar usuario"	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona el usuario que desea modificar y selecciona la opción "Modificar".	2. El sistema muestra una interfaz para modificar usuarios a partir de los siguientes datos: ➢ Usuario ➢ Contraseña ➢ Repetir contraseña

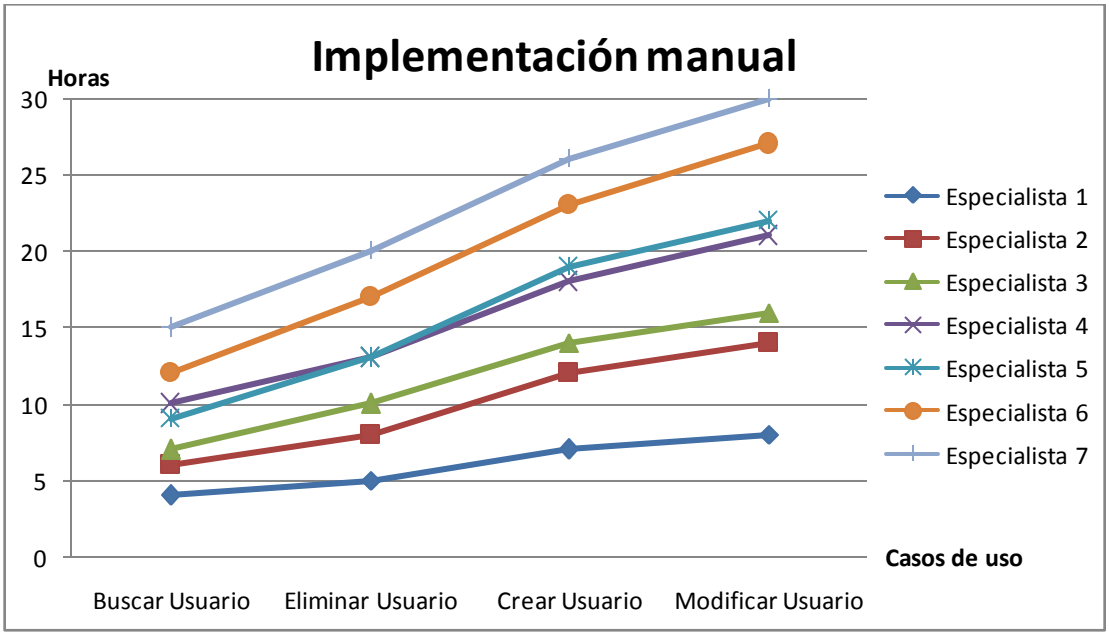
	➢ Nombre ➢ Apellidos ➢ CI ➢ Sexo ➢ Fecha de Nacimiento
3. El actor introduce los datos necesarios y selecciona la opción “Aceptar”.	4. El sistema verifica que los campos obligatorios no estén vacios, modifica el usuario y regresa al formulario para buscar usuario. Terminando así el caso de uso.
Prototipo de Interfaz	
 	

Flujos alternos	
Flujo alterno al paso 4 "Campos vacios"	
	4. a El sistema muestra un mensaje de error. Terminando así el caso de uso.
Prototipo de Interfaz	
	
Flujo Normal de Eventos	
Sección "Eliminar usuario"	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona el usuario que desea eliminar y selecciona la opción "Eliminar".	2. El sistema muestra una interfaz para preguntar si realmente desea eliminar el usuario.
3. El actor selecciona la opción "Si".	4. El sistema elimina el usuario seleccionado.

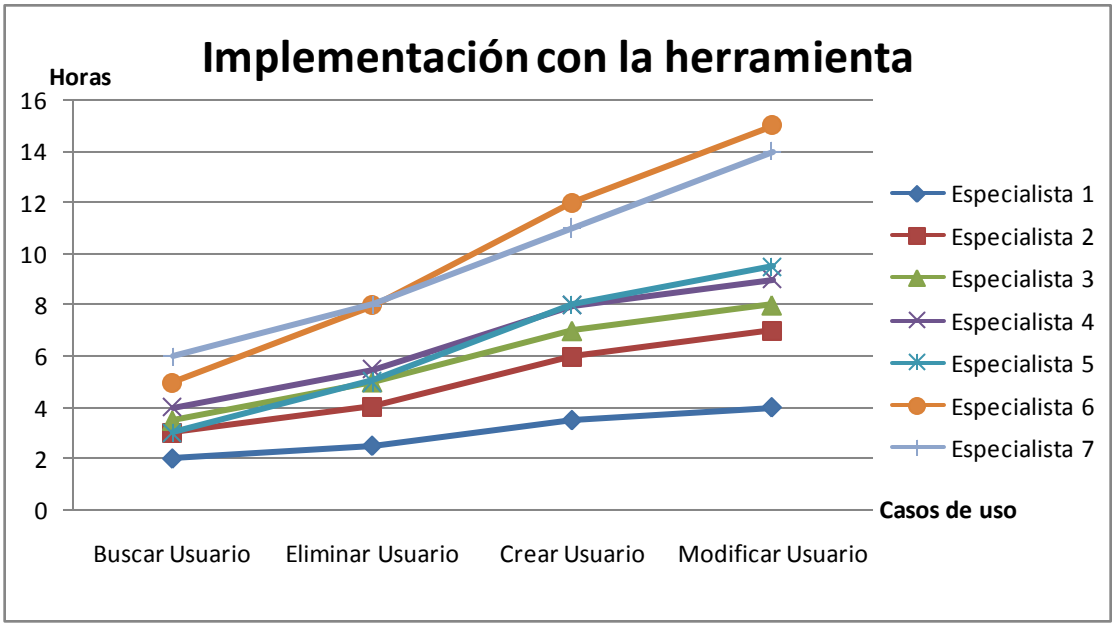
Prototipo de Interfaz


4.2.1 Resultados obtenidos

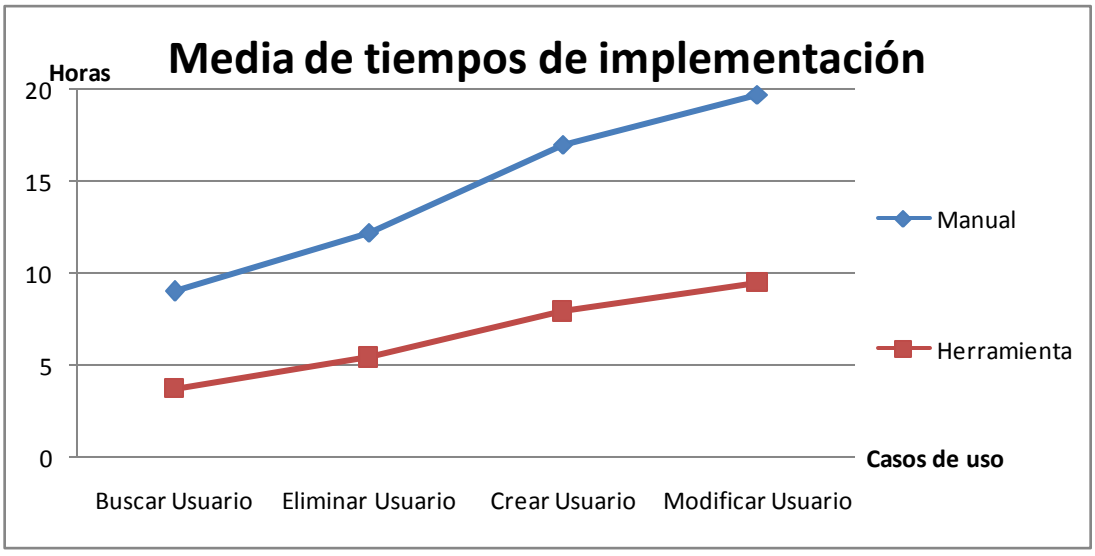
Después de implementar las interfaces de usuario de forma manual se obtuvo la siguiente relación "Horas - Caso de uso".



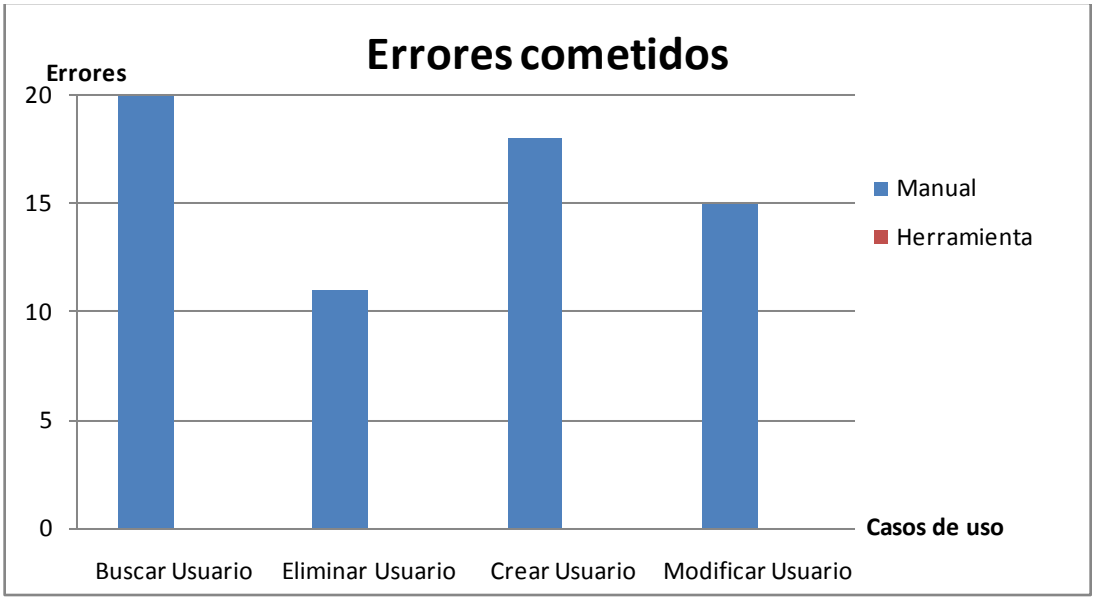
Después de implementar las interfaces de usuario haciendo uso de la herramienta generadora de código se obtuvo la siguiente relación “Horas - Caso de uso”.



Con el objetivo de comparar los resultados obtenidos en la implementación manual y en la implementación haciendo uso de la herramienta, se decide calcular una media de los tiempos empleados en implementar cada caso de uso, estos tiempos se muestran en la siguiente relación “Horas - Caso de uso”.



Después de implementar las interfaces de usuario de forma manual y haciendo uso de la herramienta se obtuvo la siguiente relación “Errores - Caso de uso”.



Conclusiones del capítulo

Una vez terminadas las pruebas de unidad haciendo uso del framework JUnit, se puede llegar a la conclusión de que el código implementado corresponde con el funcionamiento esperado y funciona correctamente.

Después de analizar los resultados obtenidos en la prueba piloto se puede ver que la herramienta generadora de código JavaScript cumple con su objetivo, observándose una disminución del 50% para el tiempo y 100% para los errores.

Conclusiones

Con esta investigación, guiada por el proceso de desarrollo RUP, se realizó un generador pasivo de código JavaScript para la librería AJAX ExtJS, que contribuirá a disminuir el tiempo de desarrollo de la interfaz de usuario del proyecto SINAPSIS u otras aplicaciones Web que utilice la librería ExtJS.

La presente investigación cumplió con los objetivos propuestos obteniéndose los siguientes resultados:

- Se realizó un levantamiento de los requisitos que debe cumplir una herramienta de generaron de código.
- Se diseñó una herramienta de generación de código que cumple con los requisitos propuestos.
- Se implementó una herramienta de generación de código que cumple con las funcionalidades propuestas.
- La herramienta es útil y fácil de usar, lo cual ha propiciado el uso de la misma en proyectos productivos.
- Se cuenta con la documentación del proyecto que permite que futuros desarrolladores continúen mejorando el generador.

El trabajo realizado constituye un punto de partida más sólido para aquellos que deseen desarrollar una herramienta de este tipo en la Universidad de las Ciencias Informáticas, la cual por sus características es un lugar propicio para el diseño y construcción de herramientas de desarrollo de software.

Recomendaciones

Para que la herramienta generadora de código JavaScript brinde mayores facilidades de uso se recomienda:

- Elaborar de una herramienta que facilite la creación y edición de los XML que representan a los componentes Ext.
- Implementar más componentes Ext para el generador.
- Implementar nuevas funcionalidades para dar más comodidades al usuario.
- Desarrollar la ayuda de la herramienta.
- Impartir cursos de capacitación a los desarrolladores para que introduzcan más eficazmente esta herramienta en el desarrollo de sus proyectos.

GLOSARIO

Base de datos:
Conjunto de información relacionada que se encuentra agrupada ó estructurada.

Código:
Cifras, clave. En informática se utiliza para referirse a un conjunto de instrucciones en un lenguaje de programación.

Compilador:
Programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar.

Clase:
Declaración o abstracción de un objeto cuando se programa usando el paradigma de orientación a objetos.

DOM:
Acrónimo de “Document Object Model”, es la estructura de objetos que genera el navegador cuando se carga un documento, estos se pueden alterar mediante Javascript para cambiar dinámicamente los contenidos y aspecto de la página.

Framework:
Estructura predefinida para la creación de aplicaciones. Puede estar formado por un conjunto de librerías y clases o por una arquitectura que facilita el desarrollo de software.

Hipertexto:
Nombre que recibe el texto que en la pantalla de una computadora te conduce a otro texto relacionado. La forma más habitual de hipertexto en documentos es la de hipervínculos o referencias cruzadas automáticas que van a otros documentos.

Hipervínculo:

Un hipervínculo es una conexión de una página a otro destino, El destino es con frecuencia otra página Web, pero también puede ser una imagen, una dirección de correo electrónico, un archivo o un programa.

IDE:

Ambiente integrado de desarrollo (Integrated development environment). Es un conjunto de software que permite el desarrollo de aplicaciones.

JSON:

Acrónimo de “JavaScript Object Notation", es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

Modelo:

Representación abstracta de la realidad. Diagramas que representan la estructura de un sistema dado.

Multiplataforma:

Es un término usado para referirse a los programas, sistemas operativos o lenguajes de programación que puedan funcionar en diversas plataformas como Windows o Linux.

NeXT:

Fue una compañía informática, conocida entre el público por sus avanzados ordenadores y en el mundo de la programación por sus plataformas de desarrollo (orientadas a objetos). NeXT se fusionó con Apple Inc. el 20 de diciembre de 1996, en una compra de aproximadamente $375 millones, junto con 1,5 millones de acciones de Apple, y actualmente no negocia como entidad separada. El software desarrollado por Next es la base para el sistema operativo Mac OS X.

Repositorio:

Sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos.

Servidor Web:

Es un programa que corre sobre el servidor que escucha las peticiones HTTP que le llegan y las satisface. Dependiendo del tipo de la petición, el servidor Web buscará una página Web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición.

Software:

Término genérico que designa al conjunto de programas que posibilitan realizar una tarea específica en un ordenador.

UNIX:

Sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas Mcllroy.

XMLHttpRequest:

Es una interfaz empleada para realizar peticiones HTTP y HTTPS a servidores WEB. Para los datos transferidos se usa cualquier codificación basada en texto, incluyendo: texto plano, XML, JSON, HTML y codificaciones particulares específicas. La interfaz se presenta como una clase de la que una aplicación cliente puede generar tantas instancias como necesite para manejar el diálogo con el servidor. El uso más popular de esta interfaz es proporcionar contenido dinámico y actualizaciones asíncronas en páginas WEB mediante tecnologías construidas sobre ella como por ejemplo AJAX

XSLT:

Es un estándar de la organización World Wide Web Consortium que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML.

BIBLIOGRAFÍA

1. **Cañedo Andalia, Rubén.** *Aproximaciones para una historia de Internet.* Ciudad de La Habana : Red Telemática de Salud en Cuba (Infomed), 2004.

2. **Lasala, Pilar y Salazar, José Luis.** *Introducción Histórica a Internet y la World Wide Web.* s.l. : http://metodosestadisticos.unizar.es/asignaturas/10234/lasala/tema1.pdf.

3. Tiranga. *Historia de los navegadores web.* [En línea] 8 de Septiembre de 2008. [Citado el: 22 de Marzo de 2009.] http://www.taringa.net/posts/info/1525077/Historia-de-los-navegadores-web.html.

4. **Van Der Henst, Christian.** maestros del web. *Los diferentes lenguajes de programación para la web.* [En línea] 2 de Noviembre de 2007. [Citado el: 20 de Marzo de 2009.] http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/.

5. **Eguíluz Pérez, Javier.** *Introducción a CSS.* 2008.

6. **Eguíluz Pérez, Javier .** *Introducción a AJAX.* 2008.

7. RibosoMatic. *Listado de librerías, frameworks y herramientas para AJAX, DHTML y JavaScript.* [En línea] 17 de Noviembre de 2007. [Citado el: 30 de Marzo de 2009.] http://www.ribosomatic.com/articulos/top-librerias-ajax-dhtml-y-javascript/.

8. **Herrington, Jack.** *Code Generation In Action.* s.l. : Manning, 2006.

9. **Molina Moreno, Pedro Juan.** *Especificación de interfaz de usuario: De los requisitos a la generación automática de código.* s.l. : Universidad de Valencia, 2003.

10. **Alvarez, Miguel Angel.** desarrolloweb.com. *Dreamweaver.* [En línea] 2001. [Citado el: 23 de Marzo de 2009.] http://www.desarrolloweb.com/articulos/332.php.

11. **Web Dev Team.** Quanta Plus. [En línea] 2005. [Citado el: 22 de Marzo de 2009.] http://quanta.kdewebdev.org/.

12. **YesSoftware.** YesSoftware. *CodeCharge Studio Feature List.* [En línea] 2009. [Citado el: 21 de Marzo de 2009.] http://www.yessoftware.com/products/features.php?product_id=1.

13. **Alachisoft.** Alachisoft. *Rapidly Generate .NET Applications with O/R Mapping.* [En línea] 2008. [Citado el: 21 de Marzo de 2009.] http://www.alachisoft.com/tdev/index.html.

14. Sparx Systems. *Enterprise Architect.* [En línea] SOLUS S.A., 2000. [Citado el: 20 de Mayo de 2009.] http://www.sparxsystems.com.ar/products/ea/index.html.

15. Embarcadero. *Embarcadero ER/Studio.* [En línea] Embarcadero Technologies, 2009. [Citado el: 25 de Marzo de 2009.] http://www.embarcadero.com/products/er_studio/index.php.

16. **Gamma, Erich, y otros.** *Design Patterns: Elements of reusable object-oriented software.* s.l. : Addison-Wesley, 1995.

17. **Larman, Craig.** *UML Y PATRONES.* s.l. : PEARSON, 2003.

18. **Abrahamsson, Pekka, y otros.** Agile Software Development Methods. Review and Analysis. [En línea] 2002. http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf. ISBN 951-38-6010-8.

19. **Molpeceres, Alberto.** Procesos de desarrollo: RUP, XP y FDD. [En línea] 15 de Diciembre de 2002. http://www.javahispano.org/contenidos.downloadatt.action?id=71.

20. **Palacio, Juan.** Gestión y Modelos para la Eficiencia en Empresas de Desarrollo de Software. [En línea] http://www.baquia.com/marketing/Gestion_y_modelos_eficiencia_software.pdf.

21. **Aguilar, Catherine.** Aplicación de Conceptos de Gestión de Proyectos y Gestión de Riesgos en el Desarrollo de Productos Nuevos en el Campo de la Tecnología de Información. Universidad de Puerto Rico : s.n., Diciembre 2005.

22. **Jeffries, Ron, Anderson, Ann y Hendrickson, Chet.** *Extreme Programming Installed.* s.l. : Addison Wesley, Octubre 2000. ISBN 0-201-70842-6.

23. **Marches, Michele, y otros.** *Extreme Programming Perspectives.* s.l. : Addison Wesley, Agosto 2002. ISBN 0-201-77005-9.

24. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : PEARSON EDUCACIÓN, 1999. 84-7829-036-2.

25. **Kroll, Per y Kruchten, Philippe.** *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP.* s.l. : Addison Wesley, Abril 2003. ISBN 0-321-16609-4.

26. Kioskea.net. *Lenguajes de programación.* [En línea] [Citado el: 28 de Mayo de 2009.] http://es.kioskea.net/contents/langages/langages.php3.

27. **Alvarez, Miguel Angel .** desarrolloweb.com. *Descripción y características de este potente y moderno lenguaje de programación.* [En línea] 18 de Julio de 2001. [Citado el: 22 de Marzo de 2009.] http://www.desarrolloweb.com/articulos/497.php.

28. **Blanco González, Victorino.** desarrolloweb.com. *Conocemos Java, con un poco de historia previa y la visión general de las características actuales de la tecnología.* [En línea] 20 de Octubre de 2004. [Citado el: 23 de Marzo de 2009.] http://www.desarrolloweb.com/articulos/1670.php.

29. Java. *Conozca más sobre la tecnología Java.* [En línea] Sun Microsystems. [Citado el: 27 de Marzo de 2009.] http://www.java.com/es/about/.

30. **González Seco, José Antonio.** *El lenguaje de programación C#.* 2001.

31. **Ceballos Sierra, Javier.** *El Lenguaje De Programación C#.* s.l. : Ra-ma, 2001.

32. **Cortijo Bon, Francisco.** Desarrollo Profesional de Aplicaciones. *Introducción al lenguaje de programación C#.* [En línea] [Citado el: 25 de Mayo de 2009.] http://decsai.ugr.es/~cb/CSharp/languaje/intro.xml.

33. **Sun Microsystem .** NetBeans. *Welcome to the NetBeans Community.* [En línea] [Citado el: 19 de Marzo de 2009.] http://www.netbeans.org/about/index.html.

34. **Sun Microsystems.** NetBeans. *NetBeans IDE 6.5 Features.* [En línea] [Citado el: 19 de Marzo de 2009.] http://www.netbeans.org/features/index.html.

35. **The Eclipse Foundation.** Eclipsepedia. *The Official Eclipse FAQs.* [En línea] 2009. http://wiki.eclipse.org/The_Official_Eclipse_FAQs.

36. **Kendall, Kenneth y Kendall, Julie.** *Análisis y Diseño de Sistemas.* s.l. : Prentice-Hall. 978-970-26-0577-5.

37. **IBM España S.A.** IBM. [En línea] [Citado el: 23 de Marzo de 2009.] http://www-142.ibm.com/software/dre/ecatalog/detail.wss?locale=es_ES&synkey=M221280M46834Z27.

38. **Company Headquarters.** Visual Paradigm. *10 Reasons to Choose Visual Paradigm.* [En línea] 2006. [Citado el: 21 de Marzo de 2009.] http://www.visual-paradigm.com/aboutus/10reasons.jsp.

39. **Lee, Justing.** TheServerSide. *Test Framework Comparasion.* [En línea] Julio de 2005. [Citado el: 15 de Mayo de 2009.] http://www.theserverside.com/tt/articles/content/TestFrameworkComparison/article.html.

40. **W3C Communications Team.** World Wide WEB Consortium. *XML in 10 points.* [En línea] 2 de Junio de 2003. [Citado el: 26 de Marzo de 2009.] http://www.w3.org/XML/1999/XML-in-10-points.html.

41. XSTREAM. [En línea] 2008, 6 de Diciembre. [Citado el: 30 de Marzo de 2009.] http://xstream.codehaus.org/.