Xtreme Programming

Pushing the limits

Universidad de las Ciencias Informáticas

Facultad 8

Análisis, diseño e implementación de un

IDE Online

Trabajo de Diploma

Autores: José Ernesto Lara Rodríguez
Yurisbel Hernández Bernal

Tutor: Lic. Tomás Orlando Junco Vázquez

Co-tutores: Ing. Arniel Serrano Hernández
Ing. Enrique José Altuna Castillo
Ing. Humberto Rivero Guevara
Ing. Jorge Amado Soria Ramírez
Ing. Maikel Pereira Ojeda

Universidad de las Ciencias Informáticas
Curso 2009-2010
4ta Graduación

A nuestros amigos.

A nuestros padres.

¿Alguna vez te has preguntado hasta dónde llega la capacidad humana? ¿Alguna vez te has preguntado hasta dónde puedes empujar tus límites? ¿Alguna vez te has propuesto llegar… al extremo?

José Ernesto Lara Rodríguez, Iniciativa Xtreme, 2006

Agradecimientos

A todos mis compañeros en estos años de tristezas, alegrías y sacrificios, a los que una forma u otra han hecho posible este momento, a los que me han apoyado y los que pensaron que no llegaría hasta el final: para todos ellos es todo esto.

A Danay, Nani, Lizi y Cecilia el Dream Team de la 10 que estaban ahí para lo que hacía falta incluso sin que se lo pidiesen.

A la gente de CICPC por estar siempre juntos en tantas vacaciones. Pero en especial a Gley, a Oliv, a Chumaker, Cindy y Mavis por formar parte de ese piquete que con el que tantos momentos compartí.

Lara, Alejandro y Danelys por ser parte de mi escuela de la vida que me enseñaron a como ser mejor tanto personal como profesionalmente.

A Isa y a Denis por ser los amigos que me ayudaron en tantos momentos difíciles, por estar ahí siempre y saber que puedo contar con ellos para lo que sea.

A mis padres y mi hermano por ser la verdadera razón por la que he realizado todos mis estudios, por ser mi fuente de inspiración, pero especialmente a mi mamá por todo lo que me ha enseñado, por todo lo que ha luchado contra viento y marea por mí y por mi hermano para que podamos ser los hombres que hoy somos, por nunca cansarse y seguir luchando, pero sobre todo por ser el verdadero motivo por el que yo he llegado a ser quien soy.

Agradecimientos

A la UCi, por permitirme ser cosas que nunca imaginé que sería: estudiante de avanzada, profesor sin graduarme, orador, viajero internacional, y hasta líder. Por enseñarme muchas verdades de la vida. En general, por abrirme las puertas al mundo.

A mis amigos (mis tantos y tantos amigos), por compartir conmigo alegrías, trabajos, secretos y vida. Por convertirse en mi familia cuando me encontraba lejos de la mía.

A mis amigos de la FEU, compañeros de mil batallas.

A César Lage, René Lazo y Humberto Rivero, por ser las tres personas que más admiro en la UCi. Por enseñarme lo que es un líder, y haberme brindado el privilegio de trabajar a su lado.

Al Súper Team y mis demás amigos de CICPC (mi "familia grande"), por hacerme desear regresar a mi ambiente de trabajo cada día.

A Tomás Orlando Junco, por ser más que "El Boss" o "Mi Grande y Poderoso Tutor", por haber sido siempre un amigo.

A "Las Cabras", por admitirme con las puertas abiertas, incluso a pesar de ser un grupo elitista. Por no haber dudado ni un segundo cuando les pedí que fueran mis co-tutores.

A la "Iniciativa Xtreme", por haber acudido a mi llamado y haber comenzado a hacer cosas junto conmigo, y no haberse detenido jamás a pesar de las adversidades.

Al "Core" (Maikel Bradshaw, Gleisy Nordet, Cindy Santos, Olivia Almeida, Yoanna Himely), por haberse integrado al núcleo original y haber formado una amistad que no olvidaré jamás. Por hacerme maldecir el momento en que nos tengamos que separar.

Y al núcleo original: Alejandro Morales Torres ("Matojo"), Danelys Zamora Suri ("Nani, la que programa") y Yurisbel Hernández Bernal ("el Yuri-Sama"), por haberme acompañado durante los cinco años que duró esta aventura inigualable llamada Universidad. Por haber sido el núcleo que impulsó

A mi familia toda, por brindarme su amor sin condiciones y sin reparos.

Y en especial a mi madre, por ser mi ejemplo en la vida. Por indicarme el camino bueno, pero dejarme elegir por mí mismo. Por hacerme sentir orgulloso cada vez que hablo de ella.

Resumen

Este trabajo aborda la investigación y desarrollo de una aplicación web que consiste en una herramienta para el desarrollo de soluciones para problemas de programación: IDE Online. Incluye un estudio del estado del arte, las herramientas y frameworks utilizados. Las funciones implementadas fueron realizadas siguiendo las estrategias actuales de desarrollo de aplicaciones web, haciendo un uso extensivo de JavaScript y utilizando los frameworks Spring y GXT. Hace alusión a las estrategias de pruebas llevadas a cabo, y a las conclusiones y recomendaciones de interés.

Contenido

Introducción

Incontables son las ventajas que brindan las aplicaciones web. Para su ejecución sólo requieren de un navegador, lo cual hace que no sea necesario ningún sistema operativo específico, ni siquiera el uso de una PC. Tampoco necesitan grandes requisitos de hardware, ya que la lógica del negocio de la aplicación queda relegada al servidor, por lo que el cliente sólo necesita procesar el código asociado a la presentación de la misma. Además, con el auge de la Internet como red global, los datos y funcionalidades que ofrecen dichas aplicaciones pueden ser accedidos desde cualquier parte del mundo.

El desarrollo en este campo, unido a otros factores tecnológicos como son el incremento de las velocidades de conexión, la aparición de nuevos estándares como HTML 5 y CSS 3, la persistente reducción de los costos del hardware y la existencia de una plétora de dispositivos capaces de acceder a la web (reproductores MP3, teléfonos celulares, televisores, consolas de juego, etc.), han traído una tendencia cada vez más fuerte a migrar todos los servicios a la web. Términos como cloud computing y SaaS (Software as a Service) ganan fuerza en el mundo de la informática y se hacen cada vez más familiares entre adeptos y escépticos. La prestigiosa consultora Gartner identifica a la tecnología cloud computing como la primera de las tecnologías estratégicas de mayor importancia para las empresas en el 2010, con potencial para tener un impacto significativo en los próximos tres años. (1)

El liderazgo en este campo actualmente lo lleva Google, quien ofrece toda una suite ofimática desarrollada como aplicaciones web, llamada Google Docs. Principios en su diseño de sus aplicaciones, como son la simplicidad, el rendimiento, y la simulación de un entorno de escritorio en la web, han contribuido a su éxito. Otras grandes compañías comienzan a avanzar en la misma dirección: Microsoft anuncia el lanzamiento de Office 2010 Web Apps, que promete ser la migración a la web de la suite ofimática más exitosa del mundo, mientras que Apple da pasos cautelosos con el lanzamiento de iWork, MobileMe, entre otras.

Situación Problémica

Sin embargo, los entornos de desarrollo (IDEs, por sus siglas en inglés) tradicionales están desarrollados como aplicaciones de escritorio, trayendo consigo todas las desventajas típicas de este tipo de

aplicaciones: falta de portabilidad, dificultad o imposibilidad de cambiar de sistema operativo, e inconvenientes procesos de instalación y configuración.

Estos factores se acentúan en ambientes donde no se cuenta con una PC fija para desarrollar o no se controla la configuración de las PCs que se van a utilizar, como son los laboratorios docentes de las universidades, en los que se desarrollan actividades docentes y competitivas.

Problema a resolver

¿Cómo crear un IDE que no necesite de ningún sistema operativo específico ni instalación y configuración en la PC del usuario?

Objetivo General

Crear un IDE online, que elimine las limitaciones que se encuentran para desarrollar en cualquier momento y desde cualquier PC.

Objeto de Estudio

Desarrollo de entornos integrados de desarrollo (IDEs) y de aplicaciones web.

Objetivos específicos

- Analizar las tendencias actuales de soluciones similares.
- Diseñar e implementar un IDE online con las características descritas.
- Validar que el IDE implementado se adhiere a las funcionalidades descritas.

Campo de Acción

Desarrollo de IDEs online para docencia y competencias.

Resultados Esperados

Aplicación web consistente en un IDE online, y su aplicación en ambientes universitarios para docencia y competencias.

Métodos Científicos

Teóricos

- Analítico – Sintético: Es utilizado para analizar la situación y problemática existente y de ella sacar la conclusión de qué características debe tener la solución propuesta, así como para analizar la documentación y características de las herramientas y frameworks existentes y llegar a la conclusión de cuáles son los más adecuados para el desarrollo.
- Análisis Histórico – Lógico: Es utilizado para analizar la evolución histórica de las soluciones similares y de los IDEs clásicos, las tendencias más recientes de las aplicaciones web y basándose en esos datos complementar las características necesarias o deseables en la solución propuesta.

Empíricos

- Observación: Para realizar el estudio de las características, tendencias y expectaciones de los usuarios de las soluciones similares, es necesario observar dichas soluciones y las interacciones y opiniones existentes de los usuarios acerca de las mismas.
- Experimento: Para realizar el estudio de las herramientas y frameworks que se utilizarán, es necesario experimentar con ellos en ambientes controlados (aplicaciones de prueba), para descubrir sus características y su grado de adecuación a las necesidades del proyecto.

Estructura capitular

En el presente documento se aborda la investigación cuyo diseño metodológico se ha descrito. Su desarrollo comienza con un capítulo que describe la fundamentación teórica de dicha investigación, el cual incluye un estudio del estado del arte de los IDEs online en el mundo, y una descripción de las herramientas disponibles y los criterios que se tuvieron en cuenta para seleccionar las que se utilizaron en el desarrollo de la solución propuesta. A continuación se incluye un capítulo que describe el proceso de construcción de la solución, el cual comienza con la descripción, priorización y planificación de las historias de usuario que describen las funcionalidades a implementar, pasando luego a detallar la arquitectura de la solución y sus principales características, y concluye con un capítulo en el cual se describen los métodos utilizados para validar la solución desarrollada.

Fundamentación Teórica

Estado del arte

IDEs

Un IDE es una aplicación compuesta por un conjunto de herramientas útiles para un desarrollador. Puede ser exclusivo para un lenguaje de programación o bien, poder utilizarse para varios. Suele consistir de un editor de código (con facilidades como resaltado de sintaxis, completamiento de código y navegación entre clases), un compilador y herramientas de automatización de la compilación, un depurador y en algunos casos un constructor de interfaz gráfica. Dentro de los más populares a nivel mundial se encuentran el Eclipse y el Microsoft® Visual Studio®. (2)

Aplicaciones online

Las aplicaciones online son muy conocidas a nivel mundial debido al gran auge de Internet en los últimos años. Entre sus ventajas sobre las aplicaciones de escritorio se encuentran:

Portabilidad: Al presentar una interfaz construida a base de estándares como HTML, pueden ser ejecutadas desde cualquier plataforma, con el único requisito de contar con un navegador que soporte dichos estándares. Esto actualmente no es una limitante en ningún sistema operativo con interfaz gráfica.

Accesibilidad: Al estar publicadas en un servidor de Internet, en el cual guardan todos sus datos y configuración, generalmente pueden ser accedidas desde cualquier computadora con acceso a la red de redes. Así, un usuario puede consultar sus correos electrónicos, documentos y demás, desde cualquier parte del mundo.

Colaboración: Al centrar todas las conexiones de los usuarios en un mismo servidor central, se facilita la colaboración y comunicación entre los mismos, haciendo énfasis en el trabajo en equipo y la distribución de la información. (3)

Entre las aplicaciones online se destacan las desarrolladas por la empresa Google. Esta empresa ofrece toda una suite ofimática, compuesta por soluciones para edición de documentos, hojas de cálculo, presentaciones, correo electrónico, traducción y muchas más. Destacan por su uso extensivo de Java

Script en lugar de otras tecnologías como Adobe Flash, Java Applets, Microsoft Silverlight u otros, sin que ello les impida crear aplicaciones robustas e interfaces flexibles y amplias. El máximo exponente de estas características es su más reciente propuesta: Google Wave.

Google Wave es la nueva plataforma de comunicación y colaboración propuesta por la empresa. Conjuga chat, correo, edición de documentos en tiempo real y en forma simultánea por distintos usuarios y permite además compartir imágenes con solamente arrastrarlas desde el escritorio.

Ilustración 1: Google Wave

Es una forma de "hacer la comunicación más fácil", y una de sus características principales es la simultaneidad. En un chat, mientras uno de los usuarios escribe, los demás pueden ver en tiempo real lo que está escribiendo. En cuanto a los documentos, una persona puede editar los textos de otra sin necesidad de autorización previa, y los cambios también pueden ser vistos en tiempo real. (4)

IDEs online

Luego de una breve panorámica de las aplicaciones online, resaltando sus ventajas sobre las aplicaciones de escritorio tradicionales, se analizarán soluciones existentes de este tipo y disponibles en Internet.

CodeRun

CodeRun es el IDE Online más completo de todos los analizados. Cuenta con un conjunto de características que lo acercan mucho a las funcionalidades más comunes brindadas por los IDEs de escritorio tradicionales. Entre ellas se encuentran: (5)

- Resaltado de sintaxis.
- Completamiento de código.
- Ejecución y depuración del lado del servidor.
- Posibilidad de crear y gestionar varios proyectos.
- Posibilidad de compartir un proyecto con otros usuarios.
- Repositorio SVN.

- Posibilidad de programar en varios lenguajes.
- Publicación y alojamiento de las soluciones desarrolladas.

TIDE

TIDE (Tiny IDE) es un IDE online para JavaScript. Su propósito es escribir, analizar o depurar pequeños programas escritos en JavaScript. Sin embargo, no ofrece gestión de proyectos ni persistencia de ningún tipo, solamente brinda un editor de código fuente y funcionalidades para probar y depurar dicho código en el navegador del cliente. Está desarrollado utilizando el framework Ext JS. (6)

Las características de su editor de sintaxis en tiempo real son:

- Resaltado de sintaxis.
- Depuración.

Metodologías de Desarrollo

Metodologías Robustas

Las metodologías tradicionales o robustas sirven para desarrollar soluciones empresariales muy complejas. Se basan en muchos años de experiencia en el uso de la tecnología orientada a objetos y en el desarrollo de software de misión crítica en una gran variedad de industrias. Guían a los equipos de proyecto en cómo administrar el desarrollo iterativo de un modo controlado mientras se balancean los requerimientos del negocio, el tiempo al mercado y los riesgos del proyecto. Las metodologías tradicionales buscan seguir una secuencia, en etapas validadas con tecnologías o manuales. Fueron pensadas para soportar la gran complejidad técnica de proyectos grandes, en los cuales se dificulta la comunicación y sincronización entre los miembros del equipo precisamente por el tamaño de éste. Aunque se ha dicho que también pueden ser utilizadas en proyectos pequeños, la gran cantidad de artefactos, roles y tareas que ellas definen generalmente no se adecuan a las características de dichos proyectos.

Metodologías Ágiles

Las metodologías ágiles cambian significativamente algunos de los énfasis de los métodos ingenieriles. La diferencia inmediata es que son menos orientados al documento, exigiendo una cantidad más pequeña de documentación para una tarea dada. De muchas maneras son más bien orientadas al

código, siguiendo un camino que dice que la parte más importante de la documentación es el código fuente. Estas metodologías ofrecen una buena solución para proyectos donde el entorno es inestable y los requisitos no se conocen con exactitud, ya que están pensadas para trabajar con incertidumbre. (7)

Estas características las hacen perfectas para el desarrollo que se propone, ya que se cuenta con un equipo pequeño (sólo dos personas), poco tiempo de desarrollo y alto riesgo. A continuación se analizan algunas de ellas:

Xtreme Programming (XP)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, ya que promueve el trabajo en equipo, se preocupa por el aprendizaje de los desarrolladores y propicia un buen clima de trabajo. Se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Esta metodología se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Para la especificación de las funcionalidades del sistema se utilizan tarjetas de papel (historias de usuarios) en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla a lo sumo en varias semanas.

SCRUM

Scrum, más que una metodología de desarrollo de software, es una forma de auto-gestión de los equipos de trabajo en forma general. Un grupo de integrantes del equipo de trabajo decide cómo hacer sus tareas y cuánto van a tardar en ello. Ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro.

Permite además seguir de forma clara el avance de las tareas a realizar, de forma que los jefes puedan ver día a día cómo progresa el trabajo. Sin embargo, no es una metodología de desarrollo, puesto que no indica qué se debe hacer para realizar el código. Debería, por tanto, complementarse con alguna otra metodología de desarrollo. Se lleva bien con las metodologías ágiles y en especial con XP.

En Scrum un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de dos semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite. (8) (9)

Conclusiones

Ambas metodologías se complementan bastante bien, ya que la primera está más enfocada al desarrollo en sí, mientras que la segunda hace mayor énfasis en la gestión del proyecto y el equipo. Por ello se decidió utilizar las dos, adecuándolas a las particularidades del ambiente en el que se encuentra el equipo de desarrollo.

Plataformas de Desarrollo

La selección de las herramientas de trabajo y lenguajes de programación depende de la plataforma de desarrollo así como del o los estilos arquitectónicos seleccionados. Las características de la solución propuesta dictan que la misma debe ser una aplicación web con una fuerte interacción con el sistema operativo y herramientas de depuración y ejecución. Es por esto que, de entre las grandes plataformas de desarrollo, se consideraron las plataformas J2EE y .NET.

J2EE

J2EE es la plataforma de desarrollo creada por Sun (recientemente adquirida por Oracle) para el desarrollo de todo tipo de aplicaciones para empresas y usuarios en general. Sun lo define como un estándar para el desarrollo de aplicaciones empresariales multicapa. A diferencia de la plataforma .NET, J2EE solamente soporta el lenguaje Java. Las aplicaciones Java están típicamente compiladas en un lenguaje intermedio llamado bytecode, que es normalmente interpretado o compilado a código nativo mediante la Máquina Virtual de Java (JVM). La JVM se sitúa en un nivel superior al hardware del sistema, y este actúa como un puente que entiende tanto el bytecode, como el sistema operativo sobre el que se pretende ejecutar. Las aplicaciones realizadas en J2EE se pueden dividir en dos, tres o más capas. En la primera capa es donde se encuentran las interfaces como páginas JSP, Servlet y Applet. En la segunda capa se encuentra los componentes Enterprise Java Beans (EJB), los servicios web y toda la lógica de negocio. La última capa es para acceder a bases de datos. Cada una de estas capas puede dividirse en subcapas.

Características de la Plataforma J2EE:

- **Portabilidad:** La plataforma J2EE cuenta con la máquina virtual de java para la mayoría de los sistemas operativos en cada una de sus versiones. Generalmente los proyectos de servicios web realizados en estos IDE generan una extensión (.WAR) que contiene toda la aplicación. Este archivo se puede transportar a diferentes sistemas operativos y es posible desplegarlo para ser accedido mediante un cliente utilizando algún servidor de aplicación que soporte las características con que se creó el servicio web.
- **Desempeño:** Moderado.
- **Interacción con el sistema operativo y otras aplicaciones:** La plataforma posee diversas funcionalidades que hacen posible la interacción con el sistema operativo, la gestión de procesos y la interacción con otras aplicaciones, sin dejar de brindar un nivel de abstracción suficiente como para no sacrificar portabilidad y solventar las diferencias entre los sistemas operativos. No obstante, si se decidiera sacrificar dicha portabilidad por algún motivo, las interfaces JNI proveen una vía de interactuar a bajo nivel con aplicaciones y bibliotecas de funciones (DLLs).
- **Servicios web:** Existen diversos frameworks que posibilitan el trabajo con servicios web de una manera sencilla, aunque en ocasiones su curva de aprendizaje puede ser algo elevada. (10)

Aunque existen diversos lenguajes de programación disponibles que se compilan a bytecodes de la JVM, el lenguaje por excelencia de la plataforma es Java.

Java es un lenguaje de programación muy extendido actualmente 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 y siempre ha estado enfocado a cubrir las necesidades tecnológicas más punteras. Desde su surgimiento se enfocó hacia el desarrollo empresarial de aplicaciones y las aplicaciones web.

Una de las principales características por las que el uso de Java se ha generalizado es que es un lenguaje independiente de la plataforma. Es una ventaja significativa para los desarrolladores de software, pues antes era necesario hacer un programa para cada sistema operativo. Esto lo consigue porque se ha creado una máquina virtual para cada sistema que hace de puente entre el sistema operativo y el programa de Java. La independencia de plataforma es una de las razones por las que Java

es interesante para Internet, ya que muchas personas deben tener acceso a las mismas aplicaciones con ordenadores distintos.

Actualmente Java se utiliza en un ámbito muy diverso. Casi cualquier cosa que se puede hacer en otro lenguaje se puede hacer también en Java, como programar páginas web dinámicas, acceder a bases de datos, utilizar XML y realizar cualquier tipo de conexión de red entre cualquier sistema.

Se debe tener en cuenta que los desarrolladores cuentan con una amplia experiencia con esta plataforma y lenguaje, así como con sus herramientas y frameworks más comunes. Además, las soluciones existentes dentro de la Iniciativa Xtreme (Jurado Online, CMS de competencias), están desarrolladas con esta tecnología, por lo tanto su uso facilitará en gran medida la integración con estos sistemas.

.NET

La plataforma .NET es un proyecto de Microsoft para el desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permite un rápido desarrollo de aplicaciones. Basado en esto, la empresa desarrolla una estrategia horizontal integrando todos sus productos, desde el sistema operativo hasta las herramientas de mercado.

El .NET Framework es un conjunto de servicios de programación diseñados para simplificar el desarrollo de aplicaciones en el entorno altamente distribuido de Internet. Incluye Common Language Runtime (CLR) y bibliotecas de clases. Cuando se compila cualquier código fuente soportado por .NET se compila a Microsoft Intermediate Language (MSIL). Para poder ejecutar MSIL se debe convertir mediante un compilador JIT o Jitter a código de máquina que se ejecuta en la plataforma del cliente.

Características de la Plataforma .NET

- **Portabilidad:** La portabilidad de .NET a través de archivos en formato PE (ejecutable portable), que contienen MSIL y los metadatos requeridos es mucho menor a la obtenida con J2EE, ya que no existen versiones del CLR para la mayoría de los sistemas operativos, solo para las versiones de Windows.
- **Desempeño:** Eficiente.
- **Interacción con el sistema operativo y aplicaciones:** La plataforma brinda excelentes funcionalidades para acceder a servicios del sistema operativo y gestionar procesos. Al ser

desarrollada casi exclusivamente pensando en Windows, explota al máximo las funciones del mismo.

- **Servicios web:** .Net Framework trae incluidas en su núcleo funcionalidades para publicar y consumir servicios web de una manera que se abstrae toda la complejidad del programador.

La plataforma .NET es la base de la nueva generación de software en la cual los servicios web son un medio que permitirá a distintas tecnologías inter-operar entre sí, así como conectar diversos sistemas operativos, dispositivos, información y usuarios dando a los desarrolladores las herramientas y tecnologías necesarias para desarrollar soluciones de negocios de manera rápida sin importar que involucren diversos medios y tecnologías. En la plataforma .NET el software es más concretamente de servicio y se enfoca a construir, instalar, consumir o integrar servicios que pueden ser accedidos mediante Internet. La idea central es que un usuario de Internet con un explorador pueda acceder a contenidos, no solo en forma de texto, imágenes o sonido, sino también pueda hacer uso de servicios web, los cuales se utilizan como bloques de construcción en este nuevo modelo de computación distribuida en internet. (10)

El lenguaje de programación insignia de la plataforma .NET es el C#.

“C Sharp” (C#) es el nuevo lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Sus principales creadores son Scott Wiltamuth y Anders Hejlsberg, este último también conocido por haber sido el diseñador del lenguaje Turbo Pascal y la herramienta RAD Delphi.

Aunque es posible escribir código para la plataforma .NET en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado en ella. Su sintaxis es muy similar a la C++, ya que la intención de Microsoft es facilitar la migración de códigos escritos en estos lenguajes y facilitar su aprendizaje a los desarrolladores habituados a ellos.

C# es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo. El hecho de ser relativamente reciente no implica que sea inmaduro, pues Microsoft ha escrito la mayor parte de la Biblioteca de Clase Base (BCL) usándolo, por lo que su compilador es el más depurado y optimizado de los incluidos en el .NET Framework SDK. (11)

Sin embargo, en el equipo de desarrollo no se cuenta con experiencia suficiente en esta plataforma y lenguaje, ya que aunque se ha utilizado en varios proyectos pequeños no se ha desarrollado en él ningún producto de gran envergadura.

Conclusiones

Los criterios fundamentales sobre los que se basó la selección de la plataforma de desarrollo fueron la experiencia de los desarrolladores con la misma y la alineación a las tecnologías utilizadas en las aplicaciones existentes en la infraestructura tecnológica de la Iniciativa Xtreme. Se selecciona entonces la plataforma J2EE para el desarrollo de la solución propuesta.

Frameworks soportados por Java

Framework es un concepto sumamente genérico, se refiere a ambiente de trabajo y ejecución, por ejemplo .NET es considerado un framework para desarrollar aplicaciones sobre Windows. En general los frameworks son soluciones completas que contemplan herramientas de apoyo a la construcción (ambiente de trabajo o desarrollo), motores de ejecución (ambiente de ejecución) o marcos en los cuales asentar la arquitectura de las aplicaciones.

Framework puede ser algo tan grande como .NET o Java, pero también el concepto se aplica a ámbitos más específicos. Por ejemplo, dentro de Java, en el ámbito específico de aplicaciones web, se tienen los frameworks: Struts y Java Server Faces, entre otros. Estos frameworks de Java en la práctica son conjuntos de librerías (APIs) para desarrollar aplicaciones web, más librerías para su ejecución (o motor), más un conjunto de herramientas para facilitar esta tarea.

Spring

Spring es un framework que se puede emplear en todo tipo de aplicaciones java, ya sean pequeñas aplicaciones web o voluminosos sistemas que distribuyen su carga entre varios servidores.

Entre otras cosas permite independizar la configuración de la aplicación del servidor en el que dicha aplicación se encuentre, evitando así tener que configurar recursos en cada uno de los servidores donde se despliegue o depender de descriptores específicos de determinados servidores comerciales.

Además, brinda importantes funcionalidades como son inyección de dependencias, programación orientada a aspectos, servicios web, etc.

Se puede configurar fácilmente para interactuar con Struts, JSF, Hibernate y muchos otros frameworks sirviendo como punto intermedio entre la capa de presentación y la de persistencia de una aplicación empresarial. (12)

Seam

Seam es un completo framework para la creación de aplicaciones Web 2.0 que unifica varias tecnologías como AJAX, Enterprise Java Beans (EJB3), Java Server Faces (JSF) entre otras. Seam introduce el concepto de contextos. Cada componente de Seam existe dentro de un contexto. El contexto conversacional por ejemplo captura todas las acciones del usuario hasta que éste sale del sistema o cierra el navegador, inclusive puede llevar un control de múltiples pestañas y mantiene un comportamiento consistente cuando se usa el botón de regresar del navegador. Seam puede ser integrado con las bibliotecas de componentes JSF JBoss RichFaces o con ICEsoft ICEFaces. Ambas bibliotecas poseen soporte para AJAX.

Conclusiones

Teniendo en cuenta las características expuestas, se selecciona Spring como framework para brindar servicios empresariales en la aplicación, ya que, aunque en funcionalidades está bastante parejo y compite con Seam, se utilizó una vez más la experiencia del equipo de desarrollo como factor de decisión.

Frameworks de JavaScript

Ext JS

Ext JS es una biblioteca JavaScript para construir aplicaciones de internet ricas en diseño y funcionalidades. El núcleo de Ext está lleno de características excepcionales orientado a un rápido desarrollo alentando el uso de código escalable y bien diseñado. Esta biblioteca proporciona abstracciones para manipulación del DOM, AJAX, eventos y eventos personalizados, animaciones, plantillas, mecanismos de programación orientada a Objetos y más. Cuenta con una extensa librería de componentes y soporte para los principales navegadores web. (13)

Google Web Toolkit (GWT)

GWT es un framework para construir aplicaciones web complejas. Su objetivo es permitir el desarrollo productivo de aplicaciones web de altas prestaciones, sin necesidad de que el desarrollador sea experto en las diferencias entre los navegadores, el uso del objeto XmlHttpRequest y JavaScript. GWT es utilizado en muchos productos de Google, incluyendo Google Wave y la nueva versión de AdWords. (14)

Este framework ofrece varias ideas revolucionarias, como son la posibilidad de programar todo el código del cliente en un lenguaje fuertemente tipado y robusto como Java, y luego traducirlo a JavaScript mediante un traductor especial. Brinda para ello una librería de emulación de las clases principales de la JDK, entre las que se encuentran todas las estructuras de datos con las que cuenta el lenguaje Java.

Permite el reciclado y la reutilización de objetos tales como widgets, scripts, etc. Brinda además funcionalidades para permitir el uso del botón “Atrás” del navegador dentro de una aplicación Ajax y una potente funcionalidad para la comunicación con el servidor llamada GWT-RPC, la cual permite invocar métodos definidos en el servidor e incluso intercambiar con ellos objetos de cualquier tipo. (15)

Ext GWT

Ext GWT brinda lo mejor de ambos mundos, ya que permite crear aplicaciones utilizando los controles y funcionalidades proporcionados por el framework Ext, pero generados mediante la filosofía del framework GWT, o sea, programados totalmente en Java y traducidos a JavaScript para su ejecución del lado del cliente. (16)

IDE

NetBeans

Ilustración 2: Logotipo de NetBeans

NetBeans IDE es una herramienta desarrollada por Sun Microsystems. Está escrito en Java, por lo que puede ser utilizado desde cualquier sistema operativo que tenga la máquina virtual de Java**.** Es un IDE multilenguaje completo y modular que tiene soporte para Java SE, Java EE, Java ME y con vinculación de gran cantidad de módulos (plugins). Cuenta con depurador, perfilador, herramientas para refactorizaciones y completamiento de código. (17)

Permite desarrollar aplicaciones:

- De escritorio
- Web
- Mobile
- Enterprise

Soporta varios lenguajes, entre ellos:

- Java
- C/C++
- Ruby on Rails
- PHP, Groovy, Python, JavaScript, entro otros

Características de NetBeans:

- Instalación y actualización más simple
- Diseñador visual de formularios para Swing GUI
- Profiling integrado, profiling “points”
- Características visuales para el desarrollo web
- Creador gráfico de juegos para celulares

Eclipse

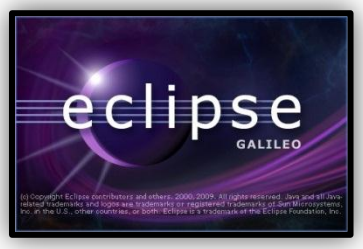

Ilustración 3: Logotipo de Eclipse

Eclipse SDK 3.5 (Galileo), es una plataforma para el desarrollo de software que incluye la plataforma Eclipse, entorno de desarrollo de plugins y Java Development Tools, además de la fuente y la documentación de usuario y programador. Es un software de desarrollo creado por IBM inicialmente para desarrollar sobre JAVA. Es un IDE multilenguaje que tiene vinculación con gran cantidad de plugins y cuenta con un completamiento de código muy superior al de NetBeans. (18)

El proyecto Eclipse se divide en tres sub-proyectos:

- El Core (núcleo): Incluye el Workspace, el Workbench, el subsistema de ayuda y la plataforma para trabajo colaborativo
- Java Development Toolkit (JDT), que brinda herramientas para desarrollar cualquier aplicación en java.
- Plugin Development Enviroment (PDE), brinda herramientas para el desarrollo de nuevos módulos.

Permite la realización de aplicaciones de todo tipo:

- Web
- Desktop
- Móviles
- Enterprise

Soporta (entre otros) los lenguajes:

- JAVA
- PHP
- Ruby (Ruby on Rails)
- C/C++
- Adobe Flexform

Características de Eclipse

- Estructura de plugins que hace sencillo añadir nuevas características y funcionalidades.
- Asistentes para la creación, exportación e importación de proyectos.
- Fácil vinculación con herramientas de control de versiones como Subversion, Git y Bazaar entre otros.

Conclusiones.

Teniendo en cuenta las características expuestas, se selecciona Eclipse SDK 3.5 (Galileo) IDE de desarrollo debido en gran medida a la flexibilidad de su editor de código, cuyas características lo hacen el editor de código más cómodo para los programadores entre los analizados. La existencia de plugins para

el trabajo con los frameworks seleccionados, las características de integración con servidores web que facilitan en gran medida el desarrollo de aplicaciones de este tipo, además de la amplia experiencia con el uso de este IDE con la que cuenta el equipo de desarrollo fueron otros factores que influyeron en la toma de esta decisión.

Herramientas de control de versiones

Para el desarrollo de un proyecto de software de esta envergadura se hace indispensable la utilización de una herramienta para el control de versiones debido a las necesidades de controlar los cambios realizados al código fuente.

Un sistema de control de versiones es un sistema de gestión de archivos y directorios, cuya principal característica es mantener el historial de cambios y modificaciones que se han realizado sobre dichos archivos a lo largo del tiempo. Es importante decir que estos sistemas no solo se limitan a gestionar archivos de texto sino que también gestionan documentos, imágenes y ficheros de todo tipo. Los sistemas de control de versiones utilizados a nivel internacional con gran aceptación y popularidad guardan toda la información en un repositorio central accesible a través de la red, permitiendo el trabajo colaborativo entre varios puestos de trabajo y a su vez proporcionando una mayor seguridad y disponibilidad de los datos.

De forma general estos sistemas brindan las siguientes características:

- Posibilidad de añadir, borrar, crear, modificar, mover, etc. cada uno de los elementos bajo control de versiones.
- **“check-out"**: Permite obtener una copia local de trabajo que puede ser examinada y/o modificada por el cliente.
- **“check-in”** (commit): Permite integrar en el repositorio los cambios realizados en la copia local.
- **“update”** (sync): Permite integrar en nuestra copia local los cambios realizados en el repositorio.
- Permite volver seguimiento a versiones anteriores de un archivo.

Subversion

Subversion es un sistema de control de versiones completamente equipado que fue originalmente diseñado para reemplazar a CVS. Desde entonces se ha expandido más allá de su objetivo original, pero

su modelo básico, el diseño y la interfaz fueron fuertemente influenciados por CVS por lo que debido a estas particularidades los usuarios de CVS se sienten muy cómodos al interactuar con Subversion. (19)

Ilustración 4: Logotipo de Subversion

Este sistema presenta varias características importantes que se plantean a continuación.

- Los directorios son versionados.
- Resolución de conflictos de forma interactiva.
- Gestiona de manera eficaz los archivos binarios.
- Opción de servidor independiente (svnserve).
- Bloqueo de archivos.
- Vinculaciones para lenguajes de programación.
- Vinculación de varios repositorios.
- Soporte para desarrolladores.
- Desarrollo Paralelo.

Dentro de las características planteadas anteriormente se profundiza en varias de ellas debido a la relevancia de las mismas.

Soporte para desarrolladores

Los desarrolladores desde sus entornos integrados de desarrollo (IDEs) como Eclipse y NetBeans entre otros tienen acceso a todas las funcionalidades que Subversion brinda mediante el uso de plugins.

Desarrollo Paralelo

Subversion permite el desarrollo paralelo, para que miembros individuales del equipo de trabajo puedan completar las diferentes partes y versiones de un proyecto al mismo tiempo, sin tener la necesidad de esperar por que otro compañero termine las tareas que está realizando.

Git

Git es un sistema de control de versiones, diseñado para el trabajo con proyectos de cualquier tamaño con gran rapidez y eficiencia. Tiene una forma diferente y revolucionaria en su sistema de guardado

haciendo que cada copia de trabajo sea un repositorio en sí mismo y contenga todo el historial de modificaciones. Esta es una característica que lo hace muy eficiente porque se puede disponer de toda la información necesaria para trabajar sin conexión y sincronizar los cambios una vez restablecida la conexión. (20)

Ilustración 5: Logotipo de Git

Entre sus características se encuentran:

Eficiencia

Git es un sistema de control de versiones que fue desarrollado para ser eficiente y como resultado se obtiene un sistema altamente eficiente y veloz en el que duplicar un repositorio lleva escasos segundos.

Desarrollo no lineal

Múltiples ramas paralelas que se dividen y se unen en distintos puntos es una de las características que brinda este sistema que evita la separación del código en ramas de trabajo que pueden o no incorporarse al código base.

Limpieza del espacio de trabajo

Git a diferencia de los otros sistemas de control de versiones únicamente crea un fichero oculto en la raíz de la copia de trabajo en la que se almacena toda su información sobre versiones.

Otros aspectos

Desaparición del número de versión del repositorio, aunque dispone de algunos sistemas para emularlo como un “descriptor” de versión.

Git, a diferencia de otros sistemas de control de versiones, almacena el fichero completo en lugar de diferencias respecto al anterior.

Autentificación criptográfica del repositorio

Conclusiones

Teniendo en cuenta las características expuestas, se selecciona Subversion v1.5.5 como herramienta de control de versiones para ser utilizado en el desarrollo de la aplicación, ya que en funcionalidades está bastante parejo y compite con Git, pero sin embargo se utilizó una vez más la experiencia del equipo de desarrollo y su fácil vinculación con las herramientas de trabajo para la toma de decisión de la herramienta de control de versiones a utilizar en la fase de desarrollo de la aplicación.

Servidores de Aplicación

Un servidor de aplicaciones es un software que al igual que un servidor web brinda la posibilidad de publicar aplicaciones a través de la red y utilizando el protocolo http. Los servidores de aplicación se diferencian de los servidores web por su frecuente integración con bases de datos y por el uso extensivo del contenido dinámico, además proporciona servicios de 'middleware', trabajando como un intermediario para la seguridad y el mantenimiento, además de proveer acceso a los datos.

JBoss

JBoss es uno de los servidores de aplicaciones más utilizados a nivel internacional que implementa la plataforma de Java EE 5, es un servidor implementado completamente en Java por lo que puede ser utilizado desde cualquier sistema operativo que cuente con la máquina virtual de java.

Ilustración 6: Logotipo de JBoss

Características de JBoss:

- Todas las funciones están basadas en estándares.
- Soporta versión 3.0 de la especificación de Servlet.
- Incrustable, orientado a arquitectura de servicios.
- Flexibilidad consistente.
- Servicios del middleware para cualquier objeto de Java. (21)

Jetty

Jetty es un servidor de aplicaciones pequeño y flexible, características que son muy importantes a tener en cuenta, ya que la memoria es un recurso valioso para cualquier aplicación. Es utilizado en una amplia variedad de proyectos y productos como teléfonos móviles, herramientas como el IDE de Eclipse y en frameworks como GWT.

Ilustración 7: Logotipo de Jetty

El servidor de aplicaciones Jetty en su versión más reciente 7.0 presenta una serie de características que se relacionan a continuación:

- Soporte para la versión 2.5 de la especificación de los Servlets y JSP 2.1.
- Soporta algunas características de la versión 3.0 de la especificación de los Servlets, como los fragmentos.
- Todas las funciones están basadas en estándares
- Flexible y extensible
- Ocupa poco espacio
- Integrable
- Servidor HTTP Asincrónico (22)

Tomcat

Apache Tomcat funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de Java Server Pages (JSP) de Sun Microsystems. Debido que fue escrito en Java, funciona en cualquier sistema operativo que cuente con la máquina virtual Java.


Ilustración 8: Logotipo de Apache Tomcat

Características de Apache Tomcat.

- Soporte para la versión 2.5 de la especificación de los Servlets y JSP 2.1.
- Todas las funciones están basadas en estándares
- Servidor HTTP Asincrónico (23)

GlassFish

GlassFish es un servidor de aplicaciones que implementa la plataforma JavaEE5, soporta las últimas versiones de tecnologías como: JSP, JSF, Servlets, EJBs, Java API para servicios web (JAX-WS), Arquitectura Java para Enlaces XML (JAXB) y muchas otras tecnologías (24)

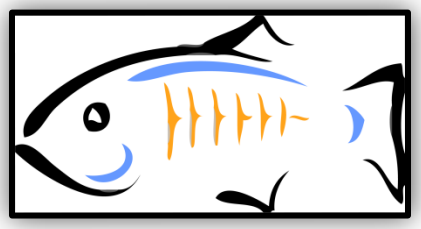

Ilustración 9: Logotipo de GlassFish

Conclusiones

Teniendo en cuenta las características expuestas, se selecciona Apache Tomcat como servidor de aplicaciones. Debido a que todos los servidores analizados están muy parejos en lo referente a sus funcionalidades, se utilizó la experiencia del equipo de desarrollo como factor fundamental en la toma de esta decisión.

Desarrollo de la solución

Exploración

En esta fase, el cliente plantea las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo y se prueba la tecnología. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

Las historias de usuarios son la técnica utilizada para especificar los requisitos del software. Son pequeñas tarjetas en las que el cliente describe brevemente los requisitos que el sistema debe poseer, sean funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que un programador (o par de programadores) pueda implementarla. Son tareas que responden a objetivos específicos que un programador debe realizar para darles respuesta.

Una vez terminada una tarea se realizan un conjunto de pruebas unitarias para comprobar el funcionamiento de los componentes implementados, también se realizan pruebas de integración para probar la interacción de estos nuevos componentes con el resto del sistema. (25)

Las historias de usuarios que son definidas en la fase de exploración inicial del proyecto tributan al primer sprint, estas historias de usuario sirven para conformar la arquitectura y para entregar una primera porción de valor del software al cliente.

Las historias de usuario fueron estimadas solo tomando en cuenta su relevancia en el negocio y la dificultad de su implementación, debido a que estos criterios son suficientes para determinar su importancia.

Plantilla de Historias de Usuarios.

La metodología seleccionada no propone ningún formato específico para las historias de usuario. Plantea que pueden ser tan simples como una tarjeta de cartón o un post-it, con una descripción de la funcionalidad deseada, y deja los detalles a consideración del equipo de desarrollo.

Teniendo en cuenta los datos necesarios para la planificación y estimación de las historias de usuario y las plantillas utilizadas en desarrollos anteriores, se propone entonces la utilización de la siguiente plantilla:

Historia de Usuario	
Número: *Número de la historia de usuario*	**Nombre**: *Nombre de la historia de usuario*
Usuario: *Usuario del sistema que utiliza o realiza la acción*	**Estimación:** *Tiempo estimado para realizar la actividad*
Prioridad: *Cuán importante es para el cliente o el equipo de desarrollo*	**Riesgo**: *Nivel de dificultad para el desarrollador*
Sprint: *Sprint al que corresponde*	
Descripción: *Breve reseña de la historia de usuario, describiendo las acciones del usuario.*	
Prueba Funcional:	
Notas: *Observaciones de interés*	

Tabla 1 - Plantilla de historias de usuario

Cada historia de usuario fue asignada a un sprint, así como una estimación como se verá con mayor detalle en la próxima sección. En total se especificaron 18 historias de usuario.

Historia de Usuario	
Número: *1*	**Nombre**: *Registrar usuario*
Usuario: *Todos*	**Estimación:** *3 días*
Prioridad*: Alta*	**Riesgo**: *Medio*
Sprint: *1er sprint*	
Descripción: *El usuario introduce sus datos en el sistema y este verifica los mismos, de estar correctos crea el nuevo usuario y automáticamente el usuario es autentificado en el sistema.*	
Prueba Funcional: *Se permite al usuario autentificarse en el sistema con los datos introducidos.*	
Notas:	

Tabla 2: Historia de usuario Registrar usuario

Historia de Usuario

Número: *2*	**Nombre**: *Autentificar usuario*
Usuario: *Todos*	**Estimación:** *3 días*
Prioridad: *Alta*	**Riesgo**: *Alto*
Sprint: *1er sprint*	
Descripción: *El usuario introduce su ID de usuario y su contraseña y el sistema verifica los datos contra el archivo de usuarios, en caso de no encontrar coincidencias brinda la opción de registrarse.*	
Prueba Funcional: *Se muestra el ambiente de trabajo del usuario autentificado.*	
Notas:	

Tabla 3: Historia de usuario Autentificar usuario

Historia de Usuario	
Número: *3*	**Nombre**: *Mostrar ambiente de trabajo del usuario autentificado*
Usuario: *Todos*	**Estimación:** *4 días*
Prioridad: *Alta*	**Riesgo**: *Alto*
Sprint: *1er sprint*	
Descripción: *Una vez autentificado el usuario, se debe mostrar su ambiente de trabajo.*	
Prueba Funcional: *Se muestran los elementos definidos.*	
Notas: *El ambiente de trabajo debe incluir un selector de proyectos, un árbol donde se muestren los contenidos del proyecto seleccionado, un editor (inicialmente en blanco) y herramientas para la edición y gestión de los proyectos y sus elementos.*	

Tabla 4: Historia de usuario Mostrar ambiente de trabajo del usuario autentificado

Historia de Usuario	
Número: *4*	**Nombre**: *Crear proyecto*
Usuario: *Autentificados*	**Estimación:** *2 días*
Prioridad: *Alta*	**Riesgo**: *Alto*
Sprint: *1er sprint*	
Descripción: *El usuario especifica el nombre proyecto y el lenguaje en el que desea programar.*	
Prueba Funcional: *El proyecto aparece en el ambiente de trabajo del usuario autentificado.*	
Notas:	

Tabla 5: Historia de usuario Crear proyecto

Historia de Usuario	
Número: *5*	**Nombre**: *Eliminar proyecto*
Usuario: *Autentificados*	**Estimación:** *½ día*
Prioridad: *Media*	**Riesgo**: *Bajo*
Sprint: *1er sprint*	
Descripción: *El usuario especifica el proyecto que desea eliminar.*	
Prueba Funcional: *El proyecto deja de aparecer en el ambiente de trabajo del usuario autentificado.*	
Notas: *Se debe pedir una confirmación al usuario, para reducir el caso de eliminaciones accidentales de los proyectos.*	

Tabla 6: Historia de usuario Eliminar proyecto

Historia de Usuario	
Número: *6*	**Nombre**: *Cambiar nombre a un proyecto*
Usuario: *Autentificados*	**Estimación:** *½ día*
Prioridad: *Alta*	**Riesgo**: *Medio*
Sprint: *2do sprint*	
Descripción: *El usuario especifica el nuevo nombre del proyecto.*	
Prueba Funcional: *El proyecto aparece con el nombre nuevo en el ambiente de trabajo del usuario autentificado.*	
Notas: *Se deben hacer las refactorizaciones pertinentes al proyecto.*	

Tabla 7: Historia de usuario Cambiar nombre a un proyecto

Historia de Usuario	
Número: *7*	**Nombre**: *Crear archivo dentro del proyecto*
Usuario: *Autentificados*	**Estimación:**
Prioridad: *Alta*	**Riesgo**: *Alto*
Sprint: *1er sprint*	
Descripción: *El usuario especifica el nombre, tipo de archivo, la plantilla que desea utilizar y la ubicación del mismo dentro del proyecto.*	
Prueba Funcional: *El archivo aparece en el ambiente de trabajo del usuario autentificado, dentro de la*	

estructura del proyecto especificado.
Notas: *Se deben hacer las refactorizaciones pertinentes al proyecto.*

Tabla 8: Historia de usuario Crear archivo dentro del proyecto

Historia de Usuario	
Número: *8*	**Nombre**: *Eliminar archivo dentro del proyecto*
Usuario: *Autentificados*	**Estimación:** *½ día*
Prioridad: *Alta*	**Riesgo**: *Bajo*
Sprint: *1er sprint*	
Descripción: *El usuario especifica el archivo que desea eliminar.*	
Prueba Funcional: *El archivo deja de aparecer en la estructura del proyecto al que pertenece.*	
Notas:	

Tabla 9: Historia de usuario Eliminar archivo dentro del proyecto

Historia de Usuario	
Número: *9*	**Nombre**: *Cambiar nombre de archivo*
Usuario: *Autentificados*	**Estimación:** *½ día*
Prioridad: *Alta*	**Riesgo**: *Bajo*
Sprint: *2do sprint*	
Descripción: *El usuario especifica el nuevo nombre del archivo.*	
Prueba Funcional: *El archivo aparece con el nuevo nombre en la estructura del proyecto especificado.*	
Notas:	

Tabla 10: Historia de usuario Cambiar nombre de archivo

Historia de Usuario	
Número: *10*	**Nombre**: *Abrir archivo*
Usuario: *Autentificados*	**Estimación:** *½ día*
Prioridad: *Alta*	**Riesgo**: *Medio*
Sprint: *1er sprint*	
Descripción: *El usuario especifica el archivo que desea abrir.*	
Prueba Funcional: *El contenido del archivo se muestra en el editor.*	

Notas:

Tabla 11: Historia de usuario Abrir archivo

Historia de Usuario	
Número: *11*	**Nombre**: *Compilar proyecto*
Usuario: *Autentificados*	**Estimación:** *3 días*
Prioridad: *Alta*	**Riesgo**: *Alto*
Sprint: *1er sprint*	
Descripción: *El usuario decide compilar el proyecto y el sistema por su parte muestra las salidas del compilador mostrando los errores o informando al usuario que el código está correcto.*	
Prueba Funcional: *El ejecutable aparece dentro de la estructura del proyecto, o aparecen en una ventana los mensajes generados por el compilador.*	
Notas:	

Tabla 12: Historia de usuario Compilar proyecto

Historia de Usuario	
Número: *12*	**Nombre**: *Ejecución remota del proyecto*
Usuario: *Autentificados*	**Estimación:** *2 días*
Prioridad: *Alta*	**Riesgo**: *Alto*
Sprint: *2do sprint*	
Descripción: *El usuario selecciona ejecución remota del proyecto y el sistema ejecuta el mismo.*	
Prueba Funcional: *Aparecen en una ventana las salidas generadas por el ejecutable.*	
Notas:	

Tabla 13: Historia de usuario Ejecución remota del proyecto

Historia de Usuario	
Número: *13*	**Nombre**: *Descargar ejecutable*
Usuario: *Autentificados*	**Estimación:** *½ día*
Prioridad: *Alta*	**Riesgo**: *Bajo*
Sprint: *2do sprint*	
Descripción: *El usuario especifica el proyecto específico y la dirección de salida el archivo.*	

Prueba Funcional: *El navegador brinda el ejecutable para su descarga.*
Notas:

Tabla 14: Historia de usuario descargar ejecutable

Historia de Usuario	
Número: *14*	**Nombre**: *Descargar proyecto completo*
Usuario: *Autentificados*	**Estimación:** *1 día*
Prioridad: *Alta*	**Riesgo**: *Medio*
Sprint: *2do sprint*	
Descripción: *El usuario especifica el proyecto que desea descargar así como la dirección de salida.*	
Prueba Funcional: *El navegador brinda un archivo compactado para su descarga.*	
Notas:	

Tabla 15: Historia de usuario Descargar proyecto completo

Historia de Usuario	
Número: *15*	**Nombre**: *Descargar archivos individuales del proyecto*
Usuario: *Autentificados*	**Estimación:** *½ día*
Prioridad: *Alta*	**Riesgo**: *Bajo*
Sprint: *2do sprint*	
Descripción: *El usuario especifica el archivo que desea descargar así como la dirección de salida.*	
Prueba Funcional: *El navegador brinda el archivo especificado para su descarga.*	
Notas:	

Tabla 16: Historia de usuario Descargar archivos individuales del proyecto

Historia de Usuario	
Número: *16*	**Nombre**: *Importar proyecto*
Usuario: *Autentificados*	**Estimación:** *1 día*
Prioridad: *Alta*	**Riesgo**: *Bajo*
Sprint: *2do sprint*	
Descripción: *El usuario especifica el proyecto que desea importar.*	
Prueba Funcional: *El proyecto aparece dentro del ambiente de trabajo del usuario.*	

Notas:

Tabla 17: Historia de usuario Importar proyecto

Historia de Usuario	
Número: *17*	**Nombre**: *Importar archivo del proyecto*
Usuario: *Autentificados*	**Estimación:** *½ día*
Prioridad: *Alta*	**Riesgo**: *Bajo*
Sprint: *2do sprint*	
Descripción: *El usuario especifica el archivo que desea importar así como la ubicación dentro del proyecto.*	
Prueba Funcional: *El archivo aparece dentro de la estructura del proyecto especificado.*	
Notas:	

Tabla 18: Historia de usuario Importar archivo del proyecto

Historia de Usuario	
Número: *18*	**Nombre**: *Guardar archivo*
Usuario: *Autentificados*	**Estimación**: *½ día*
Prioridad: *Alta*	**Riesgo**: *Bajo*
Sprint: *1er sprint*	
Descripción: *El usuario selecciona guardar archivo y el sistema guarda el archivo que actualmente está siendo modificado.*	
Prueba Funcional: *El contenido actualizado del archivo puede ser consultado al abrirlo nuevamente.*	
Notas:	

Tabla 19: Historia de usuario Guardar archivo

Estimación y Planificación

En el primer sprint se crea un sistema con la arquitectura del sistema completo, esto se debe a que son seleccionadas para el desarrollo las historias que harán cumplir la estructura del sistema completo. El cliente es quien decide cuales de las historias de usuarios serán seleccionadas para realizar en cada sprint. Al final del último sprint, el sistema está listo para su despliegue en producción.

Debido a la flexibilidad de XP, este no define métricas para la estimación de historias de usuarios, entonces el equipo de desarrollo y el cliente tienen la libertad de decidir si desean o no utilizar métricas de estimación. En este caso se decidió utilizar el popular método *Planning Poker*, analizando a la hora de realizar la estimación solo la complejidad de realización de la tarea y su importancia en el negocio.

Planning Poker es un sencillo método de estimación, en el que se presentan los requisitos a estimar uno por uno haciendo una descripción de los mismos, estos se discuten haciendo énfasis en los detalles más relevantes o que no hayan quedado claros. Una vez terminado el período de discusión cada una de las personas implicadas en el proceso elije una carta (numeradas generalmente con números de la serie Fibonacci) que representa su estimación del trabajo que lleva realizar la tarea analizada.

El proceso de estimación de realiza de forma personal y privado hasta el momento que todos los involucrados tengan una estimación, una vez que todos tengan una propuesta las estimaciones personales se hacen públicas para evitar que la estimación de una persona influya en la de otra. Una vez mostradas todas las estimaciones son analizadas y si la diferencia de días es muy grande el proceso es realizado nuevamente debido a que en la mayoría de las ocasiones esto se debe a que la información que manejan los participantes no es exacta o totalmente comprensible por lo que una segunda fase ayuda a eliminar las dudas o interrogantes y de esta forma lograr una estimación confiable.

Ésta es una forma de analizar los requisitos de forma ágil, y producir estimaciones acerca del tiempo de desarrollo que sean respetadas por todos los implicados en el desarrollo, porque todos participaron en su confección.

Planning Poker plantea que las tareas deben ser lo suficientemente sencillas como para ser realizadas en un tiempo menor que el de un sprint. Si esto no sucede las tareas deben ser divididas en sub-tareas y ser analizadas por separado. También plantea que una tarea no debe de ser realizada en un tiempo menor que la mitad de un día ideal (día de trabajo al 100%, sin ningún tipo de interrupción): si esto sucede las tareas deben agruparse y ser analizadas en conjunto. (26)

Aún con la debida planificación los riesgos son altos debido al corto tiempo, la complejidad de las historias de usuarios y el escaso conocimiento de los frameworks de trabajo. Debido a estas circunstancias, se tomaron medidas que ayudan al desarrollo de la solución de forma segura y eficiente. Estas medidas son:

Mantener control de versiones del desarrollo: Para poder regresar a una versión anterior y funcional si ocurriese algún problema.

Uso frecuente de la aplicación: Para comprobar el funcionamiento de los componentes y evitar que algunos dejen de funcionar debido a la incorporación de las nuevas funcionalidades de la aplicación o a la constante refactorización del código.

Planificación de historias de usuario del 1er sprint

Para el 1er sprint se seleccionaron las historias de usuario que ayudarán a moldear la arquitectura del sistema, así como las funcionalidades más básicas del mismo. Aunque este sprint tiene una menor cantidad de historias de usuario, en su planificación se tuvo en cuenta que se está asentando la arquitectura y definiendo la manera de utilizar los frameworks y demás herramientas, por lo que el desarrollo se ralentiza. Es por este motivo que historias del tipo “Autentificar usuario” fueron planeadas para 3 días, aun siendo una funcionalidad sencilla de implementar.

- Autentificar usuario
- Registrar usuario
- Crear proyecto
- Crear archivo dentro del proyecto
- Compilar proyecto
- Abrir archivo
- Guardar archivo
- Mostrar ambiente de trabajo del usuario autentificado
- Eliminar proyecto
- Eliminar archivo dentro del proyecto

Planificación de historias de usuario del 2do sprint

- Cambiar nombre proyecto
- Cambiar nombre archivo
- Ejecución remota del proyecto
- Descargar archivos individuales del proyecto

- Descargar proyecto completo
- Descargar ejecutable
- Importar proyecto
- Importar archivo del proyecto

Al final del 2do sprint se hará una nueva especificación de historias de usuario, para incluir funcionalidades más complejas como el resaltado de sintaxis, completamiento de código y la depuración remota del proyecto, las cuales se realizarán en un tercer sprint.

Descripción de la arquitectura

La IEEE presenta la siguiente definición de la arquitectura de software:

La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. (27)

La arquitectura es probablemente el elemento más importante durante la construcción del software, y de ella depende en gran medida el éxito del desarrollo. Involucra decisiones de alto nivel que comienzan incluso antes de la selección de un lenguaje o plataforma de desarrollo.

Principales decisiones arquitectónicas

Basándose en los requisitos especificados en las historias de usuario, se tomaron las siguientes decisiones arquitectónicas de alto nivel:

- Se debe reforzar la separación entre el cliente y el servidor
- Se deben guardar los archivos de código fuente como archivos físicos dentro del disco duro del servidor.
- No se utilizará sistema gestor de bases de datos (SGBD).
- La arquitectura de la aplicación debe ser modular.

Las otras decisiones arquitectónicas como la selección de la plataforma de desarrollo, lenguaje, frameworks, etc. no se discuten aquí porque fueron abordadas suficientemente en el capítulo anterior. A continuación se abordan estas decisiones en detalle.

Reforzar la separación entre el cliente y el servidor

Para que la solución sea verdaderamente utilizable y cumpla su propósito de poder reemplazar en cierta medida a los IDEs tradicionales, debe tener un rendimiento similar al de éstos. Para lograrlo, se decidió diseñar la aplicación viéndola en todo momento como dos aplicaciones separadas: la que correrá en el servidor y suministrará los datos, y la que correrá dentro del navegador cliente. Para reforzar este paradigma, se decidió dotar a la aplicación cliente un alto nivel de autonomía: una aplicación que cuente de su propio motor de comunicaciones, que genere completamente y actualice su presentación, que intercambie con el servidor sólo los datos imprescindibles para su funcionamiento. El servidor en ningún momento debe generar código HTML del cliente, solamente los datos que éste le solicite en formato XML, sin información de presentación. Con esto se logra reducir el tamaño de los datos que viajan por la red, aumentando así el rendimiento y la velocidad de respuesta de la aplicación cliente. Operaciones como el resaltado de sintaxis y el completamiento de código deben ser ejecutadas completamente en el cliente, sin que medien peticiones de datos al servidor.

La desventaja de esta decisión es que al inicio de la aplicación es necesario descargar gran cantidad de datos hacia el cliente, incluyendo código JavaScript (el cual incluye el gestor de comunicaciones, la lógica necesaria para generar la interfaz, y los elementos para dar soporte a los distintos lenguajes de programación: analizadores léxicos, sintácticos y semánticos) y los datos necesarios para su funcionamiento, como son las bases de datos de los elementos semánticos que serán utilizados en el completamiento de código.

No obstante, se decidió que es mejor esperar una sola vez al inicio que esperar muchas veces durante el uso de la aplicación.

Guardar elementos de los proyectos como archivos físicos

Debido a la necesidad de pasarle los archivos de código fuente y las opciones de los proyectos a los compiladores, éstos deben existir como archivos físicos en el disco duro del servidor, por lo menos en el momento de la compilación. Por esto se decidió mantenerlos siempre en el disco duro y no guardarlos en una base de datos. Si se hiciera esto último habría una gran pérdida de rendimiento durante compilación, ya que sería necesario extraer todos los archivos de la base de datos, almacenarlos temporalmente y luego de compilarlos, eliminarlos.

No utilizar SGBD

Los archivos de código fuente y las informaciones de los proyectos constituyen el grueso de los datos que maneja la aplicación, y como ya se explicó, deben ir almacenados como archivos físicos en el disco duro. Fuera de éstos, sólo se maneja la información de los usuarios a los que les es permitido acceder al sistema. Se consideró entonces que utilizar un SGBD para manejar tan pocos datos añadiría una capa de complejidad extra tanto al desarrollo como al despliegue de la aplicación, la cual no se compensa con las ganancias que reportaría.

La arquitectura de la aplicación debe ser modular

Los diseños modulares están presentes en todas las ramas de la industria. Un sistema modular brinda grandes ventajas, entre ellas la reutilización de componentes, y una gran capacidad de adaptación y escalabilidad mediante el reemplazo de componentes que comparten una interfaz común.

La solución propuesta debe soportar el desarrollo en múltiples lenguajes. Un diseño modular facilitaría en gran medida la adición de nuevos lenguajes soportados, así como nuevas funcionalidades en forma de plugin.

Estructura de la aplicación

Teniendo en cuenta las decisiones arquitectónicas expresadas, se decidió organizar la aplicación de la siguiente manera:

Un paquete de primer nivel donde residan las funcionalidades centrales del IDE y las que den soporte a la arquitectura modular, llamado “core”.

Dentro de un paquete “modules”, debe ir un paquete para cada módulo que se desarrolle (actualmente sólo existe el módulo para C++, llamado “cpp”).

Cada uno de estos paquetes, tanto el core como los módulos, deben dividirse en tres paquetes, para dar cumplimiento a la arquitectura que propone el framework GWT: “client”, “server” y “common”.

En el paquete “client” se ubicarán las clases que serán traducidas a JavaScript y serán utilizadas solamente en el cliente. Este paquete se desarrolla como aplicación autónoma, que intercambia datos con el servidor utilizando la tecnología GWT-RPC.

En el paquete “common” se ubican las clases que serán traducidas a JavaScript pero también serán utilizadas en el servidor. Típicamente en esta clasificación caen las clases del modelo de datos que se intercambian con el cliente y las interfaces necesarias para la tecnología GWT-RPC.

En el paquete “server” se ubican las clases que sólo serán utilizadas en el servidor: la aplicación servidor en sí. También se ubican las clases del modelo de datos que no es necesario que viajen al cliente, o que no es posible por motivos de seguridad, como la información de los usuarios.

Vale destacar que el framework sólo propone la creación de los paquetes “client” y “server”: se decidió incluir el paquete “common” por motivos de organización y posibles optimizaciones al código compilado.

Con esto se logra una separación estructural entre la aplicación cliente y la aplicación servidor. Se imponen además las siguientes restricciones:

- Las clases en el paquete “client” no pueden interactuar con las clases en el paquete “server”, y viceversa.
- Las clases en los paquetes “client” y “server” pueden interactuar libremente con las clases en el paquete “common”.
- En los paquetes “client” o “common” no puede ir ninguna clase que contenga información sensible o protegida, como por ejemplo las clases relativas al sistema de seguridad.

Estructura de la aplicación servidor

Para definir la organización estructural de la parte servidor de cada módulo, se utilizó el estilo arquitectónico de n capas. Este estilo plantea que las clases pertenecientes a cada capa de la arquitectura sólo se deben comunicar con las clases en la misma capa o la capa inmediatamente inferior. De esta manera, cada capa encapsula las funcionalidades de las capas inferiores y los cambios realizados en una capa sólo implican cambios como máximo en la capa inmediatamente superior.
Se definieron tres capas, como ya es común en las arquitecturas para aplicaciones web. Dichas capas se detallan a continuación:

- **Capa de acceso a datos:** Las clases de esta capa se encargan de la persistencia y carga de las entidades de la aplicación. Son las encargadas de interactuar con el sistema de archivos y crear o leer los archivos XML que contienen los datos de las entidades. Fueron diseñadas según el patrón

DAO y como convención se decide nombrarlas tomando el nombre de la entidad con la que se relacionan, agregándole la terminación “DAO”.

- **Capa de lógica del negocio:** En esta capa se agrupan las clases encargadas de administrar la lógica del negocio: gestionar las entidades, invocar a las clases correspondientes de la capa de acceso a datos y brindar a la capa de presentación todos los servicios necesarios. Como convención de nombres, se decide utilizar el nombre de la entidad a la que se asocian y la terminación “Manager”.
- **Capa de comunicación con el cliente:** Como la mayoría de la lógica de la presentación se ejecuta en el cliente, esta capa consta sólo de clases sencillas consistentes en servicios de la tecnología GWT-RPC cuya única función es invocar a las capas inferiores y comunicar los resultados al cliente. Como convención, se decide que su nombre tenga la terminación “Service”.

Para lograr una separación estructural entre las capas descritas, se decide agrupar las clases dentro de los distintos módulos y submódulos en paquetes de la siguiente manera:

- **Paquete “dao”:** Contiene las clases pertenecientes a la capa de acceso a datos.
- **Paquete “exceptions”:** Contiene las clases que representan excepciones generadas por el módulo o submódulo.
- **Paquete “managers”:** Contiene las clases pertenecientes a la capa de lógica del negocio.
- **Paquete “model”:** Contiene las distintas entidades que conforman el modelo de datos.
- **Paquete “services”:** Contiene las clases servicios pertenecientes a la capa de comunicación con el cliente.

En caso de ser necesario crear alguna clase que no pertenezca a alguna de estas clasificaciones se crean paquetes adicionales en los módulos (por ejemplo: “servlet” o “util”).

Estructura de la aplicación cliente

La aplicación cliente consta casi exclusivamente de clases relacionadas con la lógica de presentación. Se definen para ella dos capas:

- La capa de presentación, propiamente dicha.
- Una capa encargada de la gestión de las entidades y su sincronización con el servidor.

Para la confección de la capa de presentación fueron definidos tres tipos de componentes fundamentales, atendiendo a las características comunes entre los IDEs de escritorio más populares:

- **Pantalla:** Una pantalla define una vista de la aplicación. Contiene a todos los demás componentes y define la interacción del usuario en un momento dado. Eclipse llama las pantallas de trabajo "perspectivas". Visual Studio no les da nombre, pero se aprecia el cambio en el entorno de trabajo cada vez que cambia la actividad que se está realizando. Las pantallas de trabajo constan generalmente de la zona del editor, en la zona superior cintas de opciones y en la zona inferior una barra de estado. La principal diferencia entre ellas radica en el conjunto de paneles que se muestran.
- **Panel:** Un panel brinda algunas herramientas o informaciones relevantes para la tarea que está siendo desempeñada por el usuario. Se encuentran rodeando al editor, ya sea en su parte izquierda, derecha, superior o inferior. Eclipse llama a los paneles "vistas" y Visual Studio les llama "ventanas". Ejemplos de paneles son el explorador de proyectos y la consola.

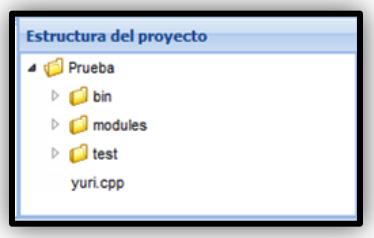

Ilustración 10: Panel "Estructura del proyecto"

- **Diálogo:** Un diálogo brinda opciones que no son utilizadas frecuentemente o solicita los datos necesarios para completar una tarea accedida desde la cinta de opciones o algún panel.

Se define entonces la estructura de la parte cliente de los módulos de la siguiente manera:

- **Paquete "manager":** Contiene las clases pertenecientes a la capa de gestión y sincronización de las entidades.
- **Paquete "ui":** Contiene las clases encargadas de la presentación. Según el tipo de componente se dividen en los siguientes paquetes:
  - **dialogs**
  - **panels**
  - **screens**

Validación de la solución

Para que la solución se considere lista para ser desplegada debe pasar previamente por una rigurosa etapa de pruebas, con el fin de analizar si cumple con las funcionalidades requeridas. A continuación se presentan las estrategias utilizadas para validar la solución y una descripción de las mencionadas pruebas.

Pruebas

En la metodología XP las pruebas son un elemento importante a lo largo de todo el desarrollo del software, estimulando a los desarrolladores a probar constantemente, permitiendo aumentar la calidad de los sistemas reduciendo el número de errores no detectados y a su vez disminuyendo el tiempo transcurrido entre la aparición de un error y su corrección. XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, pruebas de integración para verificar el sistema una vez añadida una nueva funcionalidad y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida por el cliente.

XP es una metodología de desarrollo que hace énfasis tanto en las pruebas unitarias como en las de integración y aceptación. Un programador que se esté desarrollando su aplicación bajo las pautas de esta metodología debe siempre probar su código de forma unitaria y una vez probado el mismo debe integrarlo a la aplicación y hacer las pruebas de integración correspondientes.

Pruebas de caja blanca

Las pruebas de caja blanca son realizadas mediante un seguimiento del código fuente mientras se ejecutan los casos de prueba, analizando detenidamente cada bloque de instrucciones verificando que no existan errores.

Para la ejecución de las pruebas de caja blanca es necesario tener acceso al código fuente del programa para poder analizar y comprobar su lógica interna. Al sistema desarrollado se le aplicaron las pruebas unitarias y de integración como XP propone.

Pruebas unitarias

Las pruebas unitarias son un tipo de pruebas de caja blanca que prueba a cada clase en aislamiento como elemento unitario. El objetivo fundamental de estas pruebas es asegurar el correcto funcionamiento de las interfaces y el flujo de datos entre componentes.

JUnit

JUnit es un framework de pruebas creado por Kent Beck y Erich Gamma en el que se define un conjunto amplio de clases que automatizan la ejecución de pruebas unitarias para software orientado a objetos, en particular de programas Java. Permite la ejecución de clases Java de manera controlada para poder evaluar el funcionamiento de cada uno de los métodos de la clase y analizar si esta se comporta de forma correcta.

JUnit es también un medio de controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

En la actualidad las herramientas de desarrollo más populares como NetBeans y Eclipse cuentan con plugins que generan de forma automática las plantillas necesarias para la creación de las pruebas de una clase Java, facilitando al programador enfocarse en la prueba y el resultado esperado. JUnit proporciona una forma sencilla, rápida y elegante de escribir pruebas y validarlos de forma automática, y hace énfasis en el desarrollo orientado a pruebas (TDD por sus siglas en inglés) como práctica recomendada de la metodología XP. (28)

Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra ejecutadas por el cliente o el equipo de desarrollo, con el objetivo de comprobar que la solución implementada cumple con las funcionalidades descritas y descartar los posibles errores.

Debido a la importancia de estas pruebas se plantean una serie de casos de prueba para la validar el funcionamiento de la aplicación, los cuales servirán como un apoyo a los usuarios que vayan a ejecutar las pruebas, en caso de que los mismos no estén familiarizados con el uso del software. Para especificar

dichos casos de prueba se propone utilizar la siguiente planilla, ya que la metodología de desarrollo no define ninguna:

Caso de prueba de aceptación	
Número de historia de usuario: *Numero de historia de usuario correspondiente al caso de prueba*	**Código**: *Código que identifica el caso de prueba*
Condiciones de ejecución: *Condiciones necesarias para ejecutar la prueba*	
Entrada / Pasos de ejecución : *Valores de entrada*	
Resultados de la prueba: *Salida de la ejecución*	

Tabla 20: Plantilla de casos de prueba

Para validar la solución implementada se definen los siguientes casos de prueba de aceptación:

Caso de prueba de aceptación	
Historia de usuario: *2*	**Código**: *HU2-CP1*
Condiciones de ejecución: *El usuario debe estar registrado en la aplicación.*	
Entrada / Pasos de ejecución: *El usuario introduce sus datos (usuario y contraseña) y oprime el botón "Aceptar". El sistema luego de enviados los datos por parte del usuario verifica los mismos y si coinciden muestra la pantalla de bienvenida, en caso de no coincidir se le informa al usuario que los datos no coinciden.*	
Resultados de la prueba: *Si los datos introducidos por el usuario son correctos entonces el sistema debe autentificar al usuario y mostrar la pantalla de bienvenida.*	

Tabla 21: Caso de prueba Autentificar usuario

Caso de prueba de aceptación	
Historia de usuario: *1*	**Código**: *HU1-CP2*
Condiciones de ejecución: *El usuario debe de haber seleccionado la opción Registrar usuario.*	

Entrada / Pasos de ejecución: *El usuario introduce identificador de usuario, la contraseña y una rectificación de esta última, el sistema debe verificar que no exista un usuario con el mismo identificador en caso de no existir crea el nuevo usuario con la contraseña escrita anteriormente y le muestra al usuario la pantalla de inicio de sesión. En caso de que el identificador de usuario exista el sistema le informará al usuario que ese identificador ya está en el sistema.*
Resultados de la prueba: *Si los datos introducidos por el usuario son correctos el sistema debe verificar los mismos y si no existe ningún usuario con ese identificador el usuario es creado y está listo para iniciar sesión.*

Tabla 22: Caso de prueba Registrar usuario

Caso de prueba de aceptación	
Historia de usuario: *3*	**Código**: *HU3-CP3*
Condiciones de ejecución: *El usuario debe haber iniciado sesión.*	
Entrada / Pasos de ejecución: *Una vez seleccionada la opción Nuevo proyecto el sistema brinda la posibilidad de escoger el tipo de proyecto que desea crear, una vez seleccionado el tipo de proyecto el usuario debe de introducir el nombre del mismo y presionar el botón "Crear". El sistema verifica que el usuario no tenga ningún proyecto con ese nombre, en caso de que exista alguno, el sistema informa al usuario que existe un proyecto con el mismo nombre. Si no existe un proyecto con ese nombre entonces el sistema procede a crear el proyecto y el usuario es enviado a la pantalla de trabajo con el nuevo proyecto abierto.*	
Resultados de la prueba: *Una vez creado el proyecto el usuario sea enviado a la pantalla de trabajo con el proyecto abierto. Si ocurren errores en la creación del proyecto debe mostrarse un mensaje al usuario.*	

Tabla 23: Caso de prueba Crear nuevo proyecto

Caso de prueba de aceptación	
Historia de usuario: *4*	**Código**: *HU4-CP4*
Condiciones de ejecución: *El usuario debe haber iniciado sesión, tener un proyecto válido activo y un archivo abierto.*	

Entrada / Pasos de ejecución: *El usuario selecciona la opción de guardar archivo y el sistema toma el texto del archivo activo y lo guarda en el servidor, en caso de existir algún problema el usuario es informado de lo ocurrido.*
Resultados de la prueba: *Si el usuario selecciona la opción Guardar archivo, el sistema debe guardar los datos del archivo seleccionado y mostrar un mensaje de error si ocurriese alguno.*

Tabla 24: Caso de prueba Guardar archivo

Caso de prueba de aceptación	
Historia de usuario: *5*	**Nombre**: *HU5-CP5*
Condiciones de ejecución: *El usuario debe estar debe haber iniciado sesión y tener activo un proyecto valido*	
Entrada: *El usuario selecciona la opción de compilar proyecto y el sistema compila el proyecto activo mostrando los resultados de la compilación, en caso de ser fallida la compilación por errores de código el sistema debe mostrar un mensaje de error informando que la compilación ha encontrado errores y a su vez informando al usuario que los detalles de estos están en la parte inferior de la pantalla de trabajo. En caso de no existir errores en la compilación el sistema le informa al usuario que la compilación ha sido exitosa.*	
Resultados de la prueba: *El proyecto es compilado mostrando (en caso de existir) los errores en la parte inferior de la pantalla de trabajo o informando que la compilación ha sido exitosa, además en la parte derecha de la pantalla de trabajo es agregado el archivo .exe que corresponde con el ejecutable del proyecto en cuestión.*	

Tabla 25: Caso de prueba Compilar proyecto

Caso de prueba de aceptación	
Historia de usuario: *6*	**Nombre**: *HU6-CP6*
Condiciones de ejecución: *El usuario debe haber iniciado sesión.*	

Entrada / Pasos de ejecución: *El usuario selecciona la opción de abrir un proyecto y es mostrada una ventana en la que el usuario selecciona el proyecto que desea abrir. El sistema abre el proyecto y muestra la pantalla de trabajo, mostrando en el panel derecho (estructura del proyecto) los archivos que corresponden con el proyecto abierto. En caso de ocurrir algún error el usuario es informado sobre ello con un mensaje.*
Resultados de la prueba: *Una vez abierto el proyecto se muestra la pantalla de trabajo y en la parte derecha de esta se muestran todos los archivos que componen al proyecto abierto recientemente, en caso de ocurrir algún error el usuario es informado.*

Tabla 26: Caso de prueba Abrir proyecto

Caso de prueba de aceptación	
Historia de usuario: *7*	**Código**: *HU7-CP7*
Condiciones de Ejecución: *El usuario debe haber iniciado sesión, y tener activo un proyecto válido.*	
Entrada / Pasos de ejecución: *El usuario en el panel de estructura del proyecto selecciona un archivo y al dar clic derecho sale un menú de opciones y selecciona la opción de eliminar. El sistema elimina el archivo seleccionado por el usuario y actualiza el panel de estructura del proyecto. En caso de ocurrir algún error se le informa al usuario los detalles del error.*	
Resultados de la prueba: *El archivo es eliminado correctamente del proyecto y es actualizado el panel de estructura del proyecto y si ocurre algún error se muestra un mensaje con los detalles del error.*	

Tabla 27: Caso de prueba Eliminar archivo

Caso de prueba de aceptación	
Historia de usuario: *8*	**Código**: *HU8-CP8*
Condiciones de ejecución: *El usuario debe haber iniciado sesión, tener activo un proyecto válido y debe haber compilado este proyecto con antelación.*	

Entrada / Pasos de ejecución: *El usuario selecciona la opción de descargar ejecutable por el menú de acciones básico de la pantalla de trabajo o con clic derecho sobre el archivo .exe del panel de estructura del proyecto. El sistema debe ofrecer el archivo ejecutable del proyecto en cuestión para su descarga. Si ocurre algún error el sistema muestra un mensaje.*
Resultados de la prueba: *Si los pasos realizados por el usuario son realizados correctamente el sistema descargará un archivo .zip que contiene el ejecutable del proyecto activo.*

Tabla 28: Caso de prueba Descargar ejecutable

Caso de prueba de aceptación	
Historia de usuario: *9*	**Código**: *HU9-CP9*
Condiciones de Ejecución: *El usuario debe haber iniciado sesión y tener activo un proyecto válido.*	
Entrada / Pasos de ejecución: *El usuario selecciona la opción de descargar proyecto en el menú de acciones básicas de la pantalla de trabajo. El sistema debe crear un archivo compactado que contenga los archivos del proyecto y ofrecerlo para su descarga. En caso de ocurrir algún error, se muestra un mensaje.*	
Resultados de la prueba: *Si los pasos son realizados correctamente el sistema descarga un archivo .zip con toda la información del proyecto activo.*	

Tabla 29: Caso de prueba Descargar proyecto

Caso de prueba de aceptación	
Historia de Usuario: *10*	**Código**: *HU10-CP10*
Condiciones de Ejecución: *El usuario debe haber iniciado sesión y tener activo un proyecto válido.*	
Entrada / Pasos de ejecución: *El usuario selecciona la opción de descargar archivo en el menú de acciones básicas de la pantalla de trabajo y el sistema ofrecerá el archivo activo en la pantalla de trabajo para su descarga. El usuario también puede acceder a la funcionalidad mediante el panel de estructura del proyecto a la derecha de la pantalla de trabajo, dando clic derecho en el archivo deseado y seleccionando la opción "Descargar". En caso de ocurrir algún error se mostrará un mensaje.*	
Resultados de la prueba: *Si los pasos son realizados correctamente el usuario habrá descargado el archivo deseado para su PC, si ocurre algún error entonces es informado de los detalles del mismo.*	

Tabla 30: Caso de prueba Descargar archivo

Caso de prueba de aceptación

Historia de Usuario: *11*	**Código**: *HU11-CP11*
Condiciones de ejecución: *El usuario debe haber iniciado sesión.*	
Entrada / Pasos de ejecución: *El usuario selecciona la opción de importar proyecto y el sistema mostrará una ventana para que el usuario seleccione en su PC el proyecto que desea importar al servidor. Si el usuario selecciona un archivo que no sea de extensión .zip, el sistema mostrará un mensaje de error informándole al usuario que sólo puede importar archivos de extensión .zip. Si el usuario seleccionó correctamente el archivo, el sistema subirá al servidor el proyecto y lo abrirá de forma automática para que el usuario comience a trabajar sobre él.*	
Resultados de la prueba: *Si los pasos realizados por el usuario fueron realizados correctamente, el sistema importará el proyecto seleccionado, lo abrirá de forma automática y mostrará al usuario la pantalla de trabajo con el proyecto anteriormente importado en estado activo.*	

Tabla 31: Caso de prueba Importar proyecto

Caso de prueba de aceptación	
Historia de Usuario: *12*	**Código**: *HU12-CP12*
Condiciones de ejecución: *El usuario debe haber iniciado sesión y tener activo un proyecto válido.*	
Entrada / Pasos de ejecución: *El usuario selecciona la opción de importar un archivo al proyecto mediante el panel de estructura del proyecto localizado en la parte derecha de la pantalla de trabajo haciendo clic derecho en una de las carpetas con las que cuenta dicho proyecto o en la raíz del mismo. El sistema mostrará una ventana para que el usuario seleccione en su PC el archivo que desea importar, una vez que el usuario seleccione el archivo el sistema lo subirá al servidor y lo guardará en la carpeta seleccionada por el usuario. Después de importado el archivo, el sistema actualizará el panel de estructura del proyecto mostrando el archivo importado. En caso de que ocurra algún error se debe mostrar un mensaje.*	
Resultados de la prueba: *Si los pasos realizados por el usuario fueron realizados correctamente, el sistema importará el archivo seleccionado y actualizará el panel de estructura del proyecto situado en la parte derecha de la pantalla de trabajo.*	

Tabla 32: Caso de prueba Importar archivo

Caso de prueba de aceptación	
Historia de Usuario: *13*	**Código**: *HU13-CP13*

Condiciones de ejecución: *El usuario debe haber iniciado sesión y tener activo un proyecto válido.*
Entrada / Pasos de ejecución: *El usuario accede a la funcionalidad haciendo clic derecho en el panel de estructura del proyecto y seleccionando la opción "Cambiar nombre". El sistema mostrará una ventana en la que el usuario introducirá los datos del nombre que desea poner al archivo y seleccionará la opción aceptar. Si los datos son introducidos de forma correcta, el sistema intentará cambiar el nombre del archivo y actualizará el panel de estructura del proyecto. Si ocurre algún error, se le informará al usuario mediante un mensaje.*
Resultados de la prueba: *Si los valores fueron introducidos de forma correcta, el sistema cambiará el nombre del archivo seleccionado y actualizará el estado del proyecto.*

Tabla 33: Caso de prueba Cambiar nombre de archivo

Caso de prueba de aceptación	
Historia de Usuario: *14*	**Código**: *HU14-CP14*
Condiciones de ejecución: *El usuario debe haber iniciado sesión y tener activo un proyecto válido.*	
Entrada / Pasos de ejecución: *El usuario selecciona la opción de cambiar nombre del proyecto. El sistema mostrará una ventana en la que el usuario introducirá los datos del nombre que desea poner al proyecto. Si los datos son introducidos de forma correcta, el sistema intentará cambiar el nombre del archivo y actualizará el panel de estructura del proyecto. Si ocurre algún error, se le informará al usuario mediante un mensaje.*	
Resultados de la prueba: *Si los valores fueron introducidos de forma correcta el sistema cambiará el nombre del proyecto activo y actualizará el estado del mismo.*	

Tabla 34: Caso de prueba Cambiar nombre proyecto

Conclusiones

Como resultado de este trabajo se realizó un estudio de las soluciones similares existentes en la red, así como de los antecedentes a esta aplicación: las aplicaciones web y los IDEs tradicionales. Se realizó además un estudio de las herramientas disponibles para el desarrollo, seleccionando las más adecuadas según diversos criterios.

Se obtuvo un IDE implementado como una aplicación web. El mismo constituye una aproximación a los conceptos de cloud computing y SaaS, ayuda a limar las limitantes de los IDEs tradicionales y brinda toda la potencialidad y las ventajas inherentes a las aplicaciones web a los usuarios institucionales de los IDE tradicionales.

Constituye también un apoyo significativo a las actividades docentes y competitivas que se desarrollan en las universidades y otros entornos, brindando una alternativa a los IDEs tradicionales en caso de no encontrarse éstos instalados en las PCs utilizadas para estas actividades, o de fallar los mismos por virus u otros factores.

Por último, la solución propuesta fue validada utilizando métodos de pruebas de caja blanca y de caja negra, y se comprobó que la misma responde a los objetivos planteados.

Recomendaciones

Como primera recomendación se sugiere continuar con el desarrollo de la aplicación, y a corto plazo agregarle funcionalidades avanzadas con las que cuentan los IDEs tradicionales, como son el resaltado de sintaxis, el completamiento de código, ayudas para las refactorizaciones, navegación del código y otras que ayuden a que se convierta en un verdadero reemplazo para los IDEs de escritorio en ambientes docentes o competitivos. A mediano y largo plazo, se recomienda añadirle funcionalidades para trabajo en equipo (compartición de proyectos, edición colaborativa de archivos, etc.), editores visuales, capacidades para el trabajo con aplicaciones web y otras funcionalidades de corte productivo, dándole así alcance para la producción empresarial.

Se recomienda comenzar la explotación del IDE dentro de la UCi, tal que pueda ser una opción a utilizar durante eventos docentes o competitivos cuando no se cuente con entornos de desarrollo más apropiados.

Se recomienda extender el soporte del IDE para todos los lenguajes utilizados en la docencia (Java y C#) y posteriormente otros lenguajes que se seleccionen según su frecuencia de uso.

Se recomienda además lograr la distribución del IDE en un paquete de instalación suficientemente sencillo como para poder ser distribuido a otras redes o al menos a otros usuarios para su uso local.

Referencias bibliográficas

1. **Gartner, Inc.** Gartner Identifies the Top 10 Strategic Technologies for 2010. *Gartner Technology Business Research Insight.* [En línea] Gartnet, Inc., 20 de Octubre de 2009. [Citado el: 21 de febrero de 2010.] http://www.gartner.com/it/page.jsp?id=1210613.

2. **ALEGSA.** ALEGSA.com.ar. *Definición de IDE.* [En línea] [Citado el: 2 de febrero de 2010.] Disponible en: http://www.alegsa.com.ar/Dic/ide.php.

3. **LOGin Desarrollos.** LOGin. *Características y ventajas de las aplicaciones online.* [En línea] LOGin Desarrollos. [Citado el: 12 de febrero de 2010.] Disponible en: http://www.logindesarrollos.com/es/Servicios/Desarrollo-de-aplicaciones-online.

4. **Freddy Vega, John.** Google Wave todo lo que tienes que saber. *CristalLAB.* [En línea] [Citado el: 9 de febrero de 2010.] Disponible en: http://www.cristalab.com/blog/google-wave-todo-lo-que-tienes-que-saber-c73716l/.

5. **Aldaco, Alberto.** Coderun: IDE para programadores. *Portafolio en mano.* [En línea] Miguel Lorenzo Romero, 28 de Junio de 2009. [Citado el: 10 de febrero de 2010.] Disponible en: http://www.portafolioblog.com/2009/06/coderun-ide-para-programadores-gratuito/.

6. **Schuringa, Jon.** TIDE. *TIDE.* [En línea] [Citado el: 12 de febrero de 2010.] Disponible en: http://tide4javascript.com/.

7. **Canós, José, Letelier, Patricio y Penadé, Carmen María.** Todo Ágil. *WillyDev.* [En línea] 2005. [Citado el: 8 de febrero de 2010.] Disponible en: http://www.willydev.net/descargas/prev/TodoAgil.pdf.

8. **Kniberg, Henrik.** *Scrum and Xp From The Trenches.* s.l. : Enterprise Software Development Series.

9. **Abellán, Javier.** SCRUM. *Chuidiang.com.* [En línea] [Citado el: 16 de febrero de 2010.] Disponible en: http://www.chuidiang.com/ood/metodologia/scrum.php.

10. **Santos Jaimes, MSc. Luz Marina y Omar Por, Ing. Jorge.** PLATAFORMAS J2EE Y .NET EN EL DESARROLLO DE SERVICIOS WEB. *Universidad de Pamplona.* [En línea] 2009. [Citado el: 18 de

febrero de 2010.] Disponible en: http://www.unipamplona.edu.co/unipamplona/hermesoft/portalIG/home_18/recursos/01_general/documentos/articulos/rcta/17032009/rcta_v1n13_17_plataformas_j2ee.pdf.

11. **González Seco, José Antonio.** Qué es C#. *Desarrollo Web tu mejor ayuda para aprender a hacer webs.* [En línea] 11 de Octubre de 2001. [Citado el: 9 de febrero de 2010.] Disponible en: http://www.desarrolloweb.com/articulos/561.php.

12. **Walls, Craig y Breidenbach, Ryan.** *Spring in Action, Second Edition.* s.l. : Manning Publications Co., 2008. 1-933988-13-4.

13. **Ext, LLC.** Ext JS: Cross-Browser Rich Internet Application Framework. *Ext JS.* [En línea] 2006. [Citado el: 13 de febrero de 2010.] Disponible en: http://www.extjs.com/products/extjs/.

14. **Google, Inc.** Google Web Toolkit Overview. *Google Web Toolkit.* [En línea] 16 de Mayo de 2006. [Citado el: 12 de febrero de 2010.] Disponible en: http://code.google.com/intl/en-US/webtoolkit/overview.html.

15. **Geary, David y Gordon, Rob.** *Google™ Web Toolkit Solutions.* s.l. : Pearson Education, Inc, 2008. 0-13-234481-5.

16. **Ext, LLC.** Ext GWT: Rich Internet Application Framework for GWT. *Ext JS.* [En línea] 2006. [Citado el: 13 de febrero de 2010.] Disponible en: http://www.extjs.com/products/gxt/.

17. **Oracle Corporation.** NetBeans. *NetBeans IDE 6.7 Release Information.* [En línea] Oracle Corporation, 2010. [Citado el: 12 de mayo de 2010.] Disponible en: http://netbeans.org/community/releases/67/.

18. **Eclipse, Inc.** Eclipse Documentation - Current Release (Eclipse Galileo). *Help Eclipse SDK.* [En línea] [Citado el: 02 de mayo de 2010.] Disponible en:http://help.eclipse.org/galileo/index.jsp.

19. **The Apache Software Foundation.** Subversion. *Apache Subversion.* [En línea] [Citado el: 05 de mayo de 2010.] Disponible en:http://subversion.apache.org/.

20. **Chacon, Scott.** Git. *Git - Fast Version Control System.* [En línea] [Citado el: 05 de mayo de 2010.] Disponible en: http://git-scm.com/.

21. **Red Hat, Inc.** JBoss AS - JBoss Community. *JBoss Application Server.* [En línea] JBoss Enterprise. [Citado el: 05 de mayo de 2010.] Disponible en: http://www.jboss.org/jbossas.

22. **Codehaus Foundation.** Jetty - Jetty WebServer. *Jetty.* [En línea] Mort Bay. [Citado el: 05 de mayo de 2010.] Disponible en: http://jetty.codehaus.org/jetty/.

23. **The Apache Software Foundation.** Apache Tomcat. *Apache Tomcat.* [En línea] The Apache Software Foundation, 17 de 2 de 2010. [Citado el: 04 de mayo de 2010.] Disponible en: http://tomcat.apache.org.

24. **Oracle Corporation.** Oracle Sun Developer Network (SDN). *GlassFish Community.* [En línea] Oracle Corporation. [Citado el: 05 de mayo de 2010.] Disponible en: https://glassfish.dev.java.net/.

25. **Villafuerte, Victor.** Extreme Programming Explained. *Ciclo de vida.* [En línea] 2009. [Citado el: 20 de marzo de 2010.] http://extremeprogramming.host56.com/ARTICULO5.php.

26. **Cohn, Mike.** *Agile Estimating and Planning.* s.l. : Prentice Hall, 2005. 978-0131479418.

27. **Reynoso, Billy.** *Architect Academy Webcast #1: Seminario de Arquitectura de Software.* [Webcast] Buenos Aires : Microsoft Corp., Microsoft Corp., 2005.

28. **Molpeceres, Alberto.** Java Hispano. *Introducción a JUnit.* [En línea] Asociación javaHispano (SUN Microsystem), 1 de Septiembre de 2001. [Citado el: 25 de abril de 2010.] http://www.javahispano.org/contenidos/es/introduccrion_a_junit/.

Glosario de términos

Cloud Computing: Se refiere a la computación basada en Internet, donde los recursos compartidos, la información y el software radican en un servidor y se proveen a las computadoras u otros dispositivos a demanda.

Pruebas de caja negra: Son una estrategia de pruebas que se enfocan directamente en el exterior del módulo, sin detenerse en analizar el código fuente. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y la integridad de la información externa se mantiene. Se centran en los requisitos funcionales del software.

Pruebas de caja blanca: Son una estrategia de pruebas que se enfoca en comprobar los caminos lógicos del software. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado. Requieren de conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas de diseño o el código.

SaaS: (Software as a Service) Es una forma de distribuir aplicaciones a través de internet en la que los usuarios no pagan licencias para instalar las aplicaciones en sus computadoras. Se basa en que los datos y programas se almacenan en un ambiente seguro centralizado, que es de fácil acceso y sencilla administración. Cada usuario en la red tiene su propio perfil, accesible desde un directorio común, sin estar atado a una computadora especifica. Los usuarios almacenan sus datos en un repositorio central y no en máquinas locales.