Universidad de las Ciencias Informáticas

FACULTAD 6

Entorno Integrado de Desarrollo para Prácticas de Control Automático en un Sistema de Laboratorios Virtuales y a Distancia.

Trabajo de Diploma para optar por el título de
Ingeniero Informático.

Autores: Taimí Torres Cuello.

Lázaro Oropesa Mejías.

Tutores: MSc. Omar Mar Cornelio.

Ing. Yadian Fernández Pérez.

Co-tutor: Ing. Juan Carlos Fiorenzano.

La Habana, julio del 2015.

"Año 57de la Revolución"

"El futuro tiene muchos nombres. Para los débiles es lo inalcanzable, para los tenebrosos lo desconocido y para los valientes es la oportunidad."

Victor Hugo

Declaración de Autoría

Declaro ser el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2015.

_____
Taimí Torres Cuello
Autor

_____
Lázaro Oropesa Mejías
Autor

_____
MSc. Omar Mar Cornelio
Tutor

_____
Ing. Yadian Fernández Pérez
Tutor

_____
Ing. Juan Carlos Fiorenzano
Co-Tutor

Datos de Contacto

Autor: Taimí Torres Cuello

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: ttorres@estudiantes.uci.cu

Autor: Lázaro Oropesa Mejías.

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: loropesa@estudiantes.uci.cu

Tutor: MSc. Omar Mar Cornelio.

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: omarmar@uci.cu

Tutor: Ing. Yadian Fernández Pérez.

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: yfernandezp@uci.cu

Co-tutor: Ing. Juan Carlos Fiorenzano.

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: jcfiorenzano@uci.cu

Agradecimientos

- Agradecer antes a las razones de mí existir, a las responsables de que yo sea hoy, quien soy, a las que he tenido siempre su apoyo incondicional: a mi mamá Margarita y mi hermana Saday.
- Agradecer a mi tía Basilia y a su esposo Eladio por brindarme todo el apoyo que necesite todos estos años en la escuela saben que los quiero mucho.
- Agradecer a todo el resto de mi familia: mi prima Arletty que es como si fuera otra hermana, mis abuelos, mis tíos, Ariel y mi tía Tania.
- Agradecer a mi novio Alian por convertirse en alguien tan especial, gracias a su cariño, su comprensión, su amor.
- Agradecer a mi otra familia Gladis, Belkis, Niurkis, Evelin, Leidis gracias por hacerme parte de ella.
- Agradecer a los tutores por apoyarnos durante todo este periodo de tesis.
- Agradecer a Lázaro mi compañero por hacer que este sueño se hiciera realidad.
- Agradecer a todas aquellas amistades que hice en el transcurso de los años en la universidad: Nury, Yezenia, Yisell, Sara, Yulina, Delia que ya no está y a todos los chicos del aula fue un placer conocerlos a Migue, Lein gracias por hacerme reir tanto.
- A mi papá... que siempre estuvo ahí, que supo ser madre y padre.... a mi hermano por ser tanto hermano como amigo y por tener a esa cosa linda de mi sobrina que se llama Melisa. A mi compañera de tesis, que sin ella era por gusto. A mi abuelita Eneida, por empujarme cuando parecia que me echaba para atrás. A mi familia en general. A mis hermanos de la UCI... Leo, Leandro,Yoel, Cesar, Gianny... a los que no están presentes pero nunca se fueron... Miguel Varona, y mi hermanita menor Arianna López Alfonso. A todos los profes que me impartieron clases.

Dedicatoria

- A mi madre querida, por su amor y su sacrificio incondicional.
- A mi hermana por ser mi motor impulsor, por creer en mi cuando pensé que no podía, por ser mi guía ser como mi segunda mamá.
- A mi mamá y a mi abuelito que hoy no están pero los llevo en cada momento que vivo, a mi papá y a mi hermano una vez más porque ellos lo saben que somos los tres contra el mundo.

Resumen

El presente trabajo está enmarcado en la implementación de un Entorno Integrado de Desarrollo el cual va a estar integrado a un Sistema de Laboratorios Virtuales y a Distancia, con el objetivo de crear una interfaz web con conexión al software MATLAB, para que el usuario pueda elaborar prácticas de Control Automático. Para desarrollar el IDE fue elaborado el marco teórico, lo que aportó un gran conocimiento acerca de los Sistemas de Laboratorios Virtuales y a Distancia en el Control Automático y de los Entornos Integrales de Desarrollo. La metodología utilizada durante la investigación fue OpenUp y como lenguaje de programación se utilizó Java Web. Se diseñaron e implementaron las clases del Entorno Integrado de Desarrollo, aplicando las buenas prácticas del patrón arquitectónico Modelo Vista Controlador y de los patrones de diseño GRASP y GOF. Fueron realizadas varias pruebas, con el objetivo de comprobar el funcionamiento y la calidad la aplicación.

Palabras claves:

Sistemas de Laboratorios Virtuales y a Distancia, Control Automático, Entorno Integrado de Desarrollo, MATLAB.

Abstract

This work is framed in implementing an Integrated Development Environment which will be integrated into a system of virtual laboratories and distance, with the aim of creating a web interface connection to MATLAB software for the user to develop practices Automatic Control. To develop the IDE was developed the theoretical Framework, which brought a great knowledge about Virtual Systems Laboratories and Distance in Automatic Control and Integrated Development Environments. The methodology used during the research was OpenUP and Java programming language was used Web. They were designed and implemented classes Integral Development Environment, applying the best practices of Model View Controller architectural pattern and design patterns GOF and GRASP. Several tests were conducted, in order to check the operation and application quality.

Keywords:

Virtual Systems Laboratories and Distance, Automatic Control, Integrated Development Environment, MATLAB.

Índice

Índice

Índice de tablas

Índice de figuras

Introducción

La educación siempre ha sido una de las áreas fundamentales en el desarrollo de la humanidad, donde cada día es mayor el esfuerzo del hombre para mejorarla, con el objetivo de formar más y mejores profesionales en cada una de las esferas de la sociedad. En la actualidad el sistema educacional ha presentado disímiles dificultades, entre las cuales se encuentra la necesidad de dar respuesta a los diferentes cambios sociales, económicos y culturales, que surgen en la sociedad.

Con el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) se han presentado nuevas formas para perfeccionar y desarrollar la educación, como es el caso de la educación a distancia (ED)(Castellanos, Rodríguez y Hernández, 2005)**.** La ED se presenta como la solución idónea para un conjunto de colectivos que exigen el disponer de sistemas enseñanza más flexible, accesibles y adaptativos.

Dentro de todo el conjunto de tecnologías, se destacan dos herramientas que ofrecen un nuevo enfoque en el modelo educativo de la educación a distancia: los sistemas hipermedia como forma de estructurar la información y las redes de comunicación de área extendida como soporte de la información, es decir, la red Internet (Santana CHing, 2012).

Una de las modalidades más utilizadas de la ED son los Sistemas de Laboratorios Virtuales y a Distancia (SLVD) a través de Internet. Los SLVD son cada vez más populares dentro de las comunidades universitarias ya que permiten acceder a sistemas físicos costosos desde cualquier lugar. Esto permite al alumno desarrollar actividades sin necesidad de acudir al laboratorio donde se encuentra el sistema y en el horario que desee, contribuyendo a una mejor utilización de estos recursos.

En Cuba se hace uso también de los laboratorios virtuales. Por la importancia que tienen los mismos, la Universidad de las Ciencias Informáticas (UCI) es una de las universidades que lo utilizan para apoyar el Proceso de Enseñanza-Aprendizaje (PEA), en el trabajo docente. Algunos ejemplos de ellos son los utilizados en asignaturas como Matemática, Física y Sistema Operativo. Actualmente se desarrolla una plataforma a la que se van a integrar varias disciplinas, que va a facilitar la manipulación de modelos de simulación desarrollados en diferentes herramientas**.** En este tipo de SLVD se vuelve muy atractiva el área del Control Automático, la cual es una de las técnicas que más ha explotado las nuevas tecnologías para desarrollar herramientas que faciliten el aprendizaje a distancia (Rodríguez y Castellanos, 2004). Estas

deben permitir el control en tiempo real de sistemas físicos de forma remota a través de Internet, concediéndole al usuario la facultad de simular como ejecutar en tiempo real, un esquema de control sobre determinados sistemas físicos haciendo uso de herramientas como MATLAB/Simulink que permiten al usuario concentrarse en los aspectos de diseño e implementación de sistemas de control, en lugar de la costosa programación a bajo nivel(Dixon 2001)(Dixon, 2002).

Entre las soluciones encontradas en la bibliografía, el usuario codifica a través de algún editor ASCII (Notepad o Similar) para escribir un archivo de extensión *.m*, y luego subirlo al sitio web donde será compilado por MATLAB en el servidor.

Dentro de las dificultades que presenta la solución antes mencionada se encuentran: fragmentación del proceso de desarrollo, insuficiencia para la detección de errores y un proceso de codificación angustioso. Todo lo anteriormente incide de manera negativa en el tiempo de desarrollo. En (Uran, Hercog y Jezernik, 2010) y (Jiménez, 2005) logran integrar estas herramientas sobre una plataforma web eliminando estas deficiencias, pero presentan un carácter privativo. La actual plataforma del SLVD de la UCI no cuenta con un sistema para las prácticas de Control Automático que permita la integración eficiente de las herramientas que intervienen en ello.

Por todo lo antes expuesto se plantea como **problema de la investigación**: ¿Cómo integrar la elaboración de prácticas de Control Automático en un Sistema de Laboratorios Virtuales y a Distancia?

La investigación plantea como **objeto de estudio** los Sistemas de Laboratorios Virtuales y a Distancia en el Control Automático enmarcado en el **campo de acción** los Entornos Integrales de Desarrollo para la elaboración de prácticas de Control Automático en Sistemas de Laboratorios Virtuales y a Distancia.

Se define como **objetivo general** de la investigación: Desarrollar un Entorno Integrado de Desarrollo para la elaboración de prácticas de Control Automático integrado a un Sistema de Laboratorios Virtuales y a Distancia.

Para llevar a cabo el desarrollo del objetivo general se determinan las siguientes **preguntas científicas**:

- ¿Cuáles son los fundamentos teórico-metodológicos para realizar el Entorno Integrado de Desarrollo para elaborar las prácticas de Control Automático de un Sistema de Laboratorios Virtuales y a Distancia?

- ¿Qué diseño se puede aplicar para realizar el Entorno Integrado de Desarrollo para elaborar las prácticas de Control Automático de un Sistema de Laboratorios Virtuales y a Distancia?
- ¿Existe alguna solución en la actualidad que pudiera aportar a la resolución del problema?
- ¿Cómo implementar un módulo informático que permita elaborar las prácticas de Control Automático de un Sistema de Laboratorios Virtuales y a Distancia?
- ¿Cómo validar el módulo informático que permita elaborar las prácticas de Control Automático de un Sistema de Laboratorios Virtuales y a Distancia?

Para cumplir con el objetivo de las preguntas científicas se planifican las siguientes **tareas de investigación**:

Tareas de la investigación:

- Elaboración del marco teórico de los Sistemas de Laboratorios Virtuales y a Distancia en el Control Automático.
- Caracterización de las tecnologías y herramientas a utilizar en el proceso de desarrollo de un Entorno Integrado, en un Sistema de Laboratorios Virtuales y a Distancia en el Control Automático.
- Diseño de la solución propuesta de los Sistema de Laboratorios Virtuales y a Distancia en el Control Automático.
- Implementación de las funcionalidades de un Entorno Integrado de Desarrollo en un Sistema de Laboratorios Virtuales y a Distancia en el Control Automático.
- Validación de la solución mediante las pruebas al Sistema de Laboratorios Virtuales y a Distancia en el Control Automático.

Para la realización de las tareas fueron utilizados métodos de investigación teóricos y empíricos.

Métodos teóricos

Análisis-Síntesis: La investigación se encaminará a partir del análisis de los conceptos y la correspondiente selección y/o síntesis de las posibles herramientas a utilizar para el desarrollo del producto informático.

Histórico-Lógico: Se utilizó con el objetivo de estudiar los elementos presentes en otras soluciones que puedan ser de ayuda para el desarrollo de la investigación.

Métodos empíricos

Revisión de documentos: Permitirá la elaboración del marco teórico conceptual y el desarrollo del diseño de la propuesta de la interfaz web.

Estructura del documento

Este documento está compuesto por una introducción, tres capítulos, conclusiones, referencias bibliográficas, recomendaciones y anexos.

Capítulo 1: Fundamentos teóricos.

En este capítulo se plantean los fundamentos teóricos, en el cual se analizan definiciones importantes de los SLVD, las prácticas en la enseñanza del Control Automático y los IDEs, así como las metodologías y herramientas aplicables para resolver el problema de investigación.

Capítulo 2: Análisis y diseño.

En este capítulo se plantea el análisis y diseño de la solución. En él quedan definidas todas las funcionalidades y características que debe cumplir el Entorno Integrado de Desarrollo y además se realiza una descripción de su funcionamiento. Se identificaron los actores, así como los casos de usos del sistema, agrupando dentro de estos, los diferentes requisitos funcionales. Se realiza el diagrama de clases del diseño abordando acerca de los patrones de diseño y arquitectónicos.

Capítulo 3: Implementación y prueba.

En este capítulo se hace referencia a todos los elementos necesarios utilizados durante la implementación del IDE. Se realizaron los diagramas de componentes y de despliegue del sistema, mostrando la relación que existe entre los diferentes objetos que lo conforman. Fueron realizadas las pruebas al Entorno Integrado de Desarrollo para verificar el correcto funcionamiento del sistema.

Capítulo 1: Fundamentos teóricos

En este capítulo se realiza un estudio sobre los elementos que fundamentan la base teórica, durante el desarrollo de la investigación. Se hace un análisis de los principales conceptos y se incluye una descripción de los lenguajes, las herramientas, tecnologías y metodologías utilizadas para la confección del Entorno Integrado de Desarrollo.

1.1 Sistemas de Laboratorios Virtuales y a Distancia

La estructuración de la información mediante hipermedia, multimedia e Internet, es una herramienta valiosa y muy utilizada en la creación de sistemas de apoyo al aprendizaje y de experiencias educativas que no incluyen un componente importante en la práctica. De igual forma, los sistemas de enseñanza basada en Internet o *e-learning*, trasladan el entorno de enseñanza a espacios virtuales donde se puede enriquecer el proceso de autoaprendizaje. Pero para los casos donde es necesaria la realización de actividades prácticas en laboratorios convencionales (LC), las universidades se enfrentan a dificultades que incluyen la carencia de recursos en personas, espacios y problemas presupuestarios para la adquisición de equipo (Lorandi Medina et al, 2011).

Una solución a estos problemas se puede encontrar en la aplicación de los avances tecnológicos a la docencia e investigación, mediante la creación de laboratorios virtuales (LV), que gracias al desarrollo de las TIC se han convertido en un instrumento para la mejora del proceso de enseñanza-aprendizaje en los estudiantes.

Un LV es un espacio electrónico de trabajo concebido para la colaboración y la experimentación a distancia con objeto de investigar o realizar otras actividades creativas, elaborar y difundir resultados mediante tecnologías difundidas de información y comunicación (Vary, 1999).

Permite facilitar que se realicen prácticas o experiencias a un mayor número de estudiantes, aunque alumnos y laboratorios no coincidan en un espacio físico. Permite además simular muchos fenómenos físicos y modelar sistemas, conceptos abstractos, situaciones hipotéticas, controlando la escala de tiempo, la frecuencia, etc., ocultando si así se requiere el modelo matemático y mostrando solo el fenómeno simulado e inclusive de forma interactiva, llevando el laboratorio al hogar de estudiantes (Lorandi Medina, Alberto P et al, 2011).

1.1.1 Ventajas y desventajas de los Sistemas de Laboratorios Virtuales y Distancia

Ventajas de los Sistemas de Laboratorios Virtuales y Distancia: (Rosado y Herreros, 2009)

- Acerca y facilita a un mayor número de alumnos la realización de experiencias, aunque alumno y laboratorio no coincidan en el espacio. El estudiante accede a los equipos del laboratorio a través del navegador pudiendo experimentar sin riesgo alguno además, se flexibiliza el horario de prácticas y evita la saturación por el solapamiento con otras asignaturas.
- Los estudiantes aprenden mediante prueba y error, sin miedo a sufrir o provocar un accidente, sin avergonzarse de realizar varias veces la misma práctica, ya que pueden repetirlas sin límite; sin temor a dañar alguna herramienta o equipo. Pueden asistir al laboratorio cuando ellos quieran y elegir las áreas del laboratorio más significativas para realizar prácticas sobre su trabajo.
- En Internet encontramos multitud de simulaciones de procesos físicos (en forma de applets de Java y/o Flash). Con estos objetos dinámicos, el docente puede preparar actividades de aprendizaje que los alumnos han de ejecutar, contestando al mismo tiempo las cuestiones que se les plantean.

Desventajas de los Sistemas de Laboratorios Virtuales y Distancia: (Rosado y Herreros, 2009)

- Es importante que las actividades en el LV, vengan acompañadas de un guión que explique el concepto a estudiar, así como las ecuaciones del modelo utilizado. Es necesario que el estudiante realice una actividad ordenada y progresiva, conducente a alcanzar objetivos básicos concretos.
- El alumno no utiliza elementos reales en el LV, lo que provoca una pérdida parcial de la visión de la realidad. Además, no siempre se dispone de la simulación adecuada para el tema que el profesor desea trabajar. En Internet existe demasiada información, a veces inútil. Para que sea útil en el proceso de enseñanza-aprendizaje, se debe seleccionar los contenidos relevantes para los alumnos. Son pocas las experiencias realizadas con LV en los centros educativos, donde aún impera el uso de recursos tradicionales, tanto en la exposición de conocimientos en el aula como en el laboratorio.

1.1.2 Características de los Sistemas de Laboratorios Virtuales y a Distancia

Los SLVD presentan características comunes como son: (Rodríguez y Castellanos, 2004)

- **Disponibilidad:** Los sistemas de enseñanza basados en Web deben poder estar disponibles las 24 horas del día. Esto implica que el sistema debe de tener medidas de autoprotección para garantizar este aspecto. Todos los experimentos deben de ser equipados con dispositivos hardware y software que prevengan daños al equipo o al personal presente en el laboratorio.
- **Accesibilidad:** Debido a que el SLVD está montado sobre una plataforma Web, permite a los usuarios acceder al sistema desde cualquier parte del mundo. Para ello solo es necesaria una computadora con conexión a Internet y un navegador Web, como el Internet Explorer.
- **Facilidad de uso:** Para utilizar el SLVD los usuarios solo deben tener los conocimientos básicos de los sistemas de control, tales como el modelado de dispositivos físicos y el diseño de controladores. De esta forma el usuario se centra en aprender estos temas y evita todos los problemas asociados a la implementación y operación de los equipos usados en las prácticas.
- **Interfaz rápida y fácil:** La principal función de esta parte del sistema es conformar el pedido de las prácticas y mandarlo hacia el servidor Web. La interfaz de usuario del SLVD está basada en páginas HTML y ASP; esto permite que los usuarios puedan acceder al sistema de una forma rápida y sin necesidad de descargar o instalar ningún software adicional.
- **Administración de múltiples pedidos en forma paralela:** El SLVD es capaz de atender múltiples pedidos de prácticas de forma paralela, administrando centralizadamente, dispositivos similares que se encuentren geográficamente separados pero unidos por redes de área extensa. Esto permite una mayor disponibilidad de los equipos y un servicio más rápido y eficiente para los usuarios, lográndose con esto reducir los tiempos de espera para que un usuario realice una determinada práctica.
- **Desarrollo de controladores de forma remota utilizando Matlab y Simulink:** Posibilita a los usuarios diseñar sus propios controladores utilizando el ambiente Matlab-Simulink. Estos programas son un estándar en el área del control automático, por lo que los usuarios no necesitan perder tiempo aprendiendo nuevos lenguajes de programación para implementar un controlador nuevo, solo necesitan un conocimiento básico del ambiente de Matlab y Simulink.

Por lo visto anteriormente podemos afirmar que los LV son espacios de trabajo apoyados en las tecnologías y las comunicaciones a los cuales se pueden acceder o no mediante la web. Los LV son de suma importancia en la UCI dado el nuevo modelo de enseñanza-aprendizaje, donde se puede modelar procesos de diferentes asignaturas, lograr que el estudiante visualice y comprenda de una forma más fácil y didáctica lo que se les imparte en las clases. Con el uso de los LV se pretende

mejorar la calidad en el proceso de aprendizaje y aumentar de esta forma la preparación del estudiante.

1.2 Sistemas de Laboratorios Virtuales y a Distancia para la enseñanza de Control Automático

La enseñanza de la Ingeniería de Control requiere de un elemento que permita al estudiante poner en práctica todos los conocimientos que vaya adquiriendo a lo largo del estudio de la materia. Este papel en las enseñanzas tradicionales lo desempeña el laboratorio de prácticas, el cual, inexorablemente, requiere de la presencia física del estudiante para poder manipular los sistemas de control y las plantas existentes en un entorno controlado bajo la supervisión del profesor. Por consiguiente, trasladando este entorno práctico a la enseñanza a distancia, el elemento necesario para abordar la realización de prácticas sobre Sistemas de Control es la existencia de un laboratorio virtual y de tele presencia accesible a través de una red basada en protocolos TCP/IP que permita al alumno practicar de una forma lo más similar posible a como si estuviese en las dependencias del laboratorio, dándole la posibilidad de manejar las simulaciones o interactuar con las plantas reales (Sánchez, Dormido y Morilla, 2009).

Las aplicaciones del control automático en la actualidad son muy extensas, variadas e importantes. Quizás una de las más populares es la del control de robots manipuladores en la industria de manufactura. Desde la líneas de ensamblaje de automóviles hasta las celdas robotizadas de soldadura. Las razones principales para este éxito son la alta calidad del trabajo, el ahorro de tiempo y la reducción del costo de producción (Hernández, Silva Ortigoza y Carrillo Serrano, 2013).

En la actualidad son utilizados varios laboratorios virtuales que son utilizados en la enseñanza del Control Automático, algunos ejemplos de ellos son:

- **RECOLAB: Remote Control Laboratory** (Santana CHing, 2012)

Recolab es un laboratorio remoto del Departamento de Ingeniería de la Universidad Miguel Hernández en España. Este laboratorio permite al usuario aplicar una estructura de control a un servomotor, así también como a un cilindro deslizante. El sistema permite visualizar por medio de una cámara en línea la ejecución en tiempo real de las aplicaciones de control disponibles, además provee una gráfica y una tabla de resultados descargable. Recolab proporciona una arquitectura para la ejecución a distancia en tiempo real

de estructuras de control sobre sistemas físicos por medio de Matlab/Simulink. El sistema permite ejecutar a través de Internet aplicaciones de Simulink en tiempo real sobre sistemas físicos en el laboratorio.

- **ACT: Automatic Control Telelab** (Santana CHing, 2012)

Automatic Control Telelab es un laboratorio remoto para sistemas de Control Automático de la Universidad de Siena en Italia. Este laboratorio permite controlar servos de posición y de velocidad, así como un levitador magnético [1]y otros experimentos con fluidos. El ACT provee configuración en tiempo real, observación de los experimentos, así como la adquisición de datos de manera remota. Este laboratorio se basa en MATLAB/Simulink para el diseño de las estructuras de control y su interfaz no permite la manipulación de bloques gráficos, además para diseñar una nueva estructura de control el usuario primero debe crearla utilizando Simulink, para luego agregarla al sistema de interacción del laboratorio remoto.

- **TeleLab: Remote Automations Lab** (Santana CHing, 2012)

Telelab es una plataforma diseñada para proveer a los estudiantes acceso remoto a un controlador de lógica programable para realizar experimentos de automatización industrial. Los estudiantes pueden programar este dispositivo a través de Internet. Este controlador está físicamente conectado a un sistema real y cuenta con sensores y actuadores. Un sistema de adquisición de imagen permite a los estudiantes recibir audio y video en vivo con el fin de verificar el comportamiento del sistema según la aplicación programada.

1.3 Entorno de Integrado de Desarrollo

Un IDE es un conjunto de herramientas de programación, o sea, los componentes que lo conforman consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los

[1] es un método por el cual un objeto es mantenido a flote por acción únicamente de un campo magnético.

IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes (Mandonado, 2012).

Los IDEs ofrecen un marco de trabajo para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, el mismo puede funcionar como un sistema en tiempo de ejecución, donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto.

Ventajas de los IDEs.

- Agiliza el proceso de desarrollo para los usuarios.
- Permiten formatear el código.
- Permiten renombrar variables y funciones.
- Son capaces de detectar errores de forma rápida y sencilla.
- Facilitan el mantenimiento y lectura del código a través del resaltado de sintaxis.
- Permiten crear o cargar proyectos con facilidad.

Componentes que conforman los IDEs

- **Editor de código:** Es un procesador de textos orientado para escribir código fuente de aplicaciones en lenguajes de programación. Soportan varios lenguajes y son capaces de abrir varios archivos a la vez, resaltar su sintaxis y ofrecer ayudas contextuales a la hora de escribir o visualizar el código fuente de las aplicaciones.
- **Compilador:** Es un software que se encarga de traducir el programa hecho en lenguaje de programación, a un lenguaje de máquina que pueda ser comprendido por el equipo y pueda ser procesado o ejecutado por este. La importancia de los compiladores radica en que, sin estos programas no existiría ninguna aplicación informática, ya que son la base de la programación en cualquier plataforma.
- **Depurador:** Es una herramienta opcional para depurar o limpiar los errores de algún programa informático. Habitualmente, entre las opciones de compilación, deben añadirse instrucciones para generar información para el depurador.
- **Consola de salida:** En ella son mostrados todos los resultados y los diferentes mensajes de errores del programa.

1.3.1 Niveles que conforman el Entorno Integrado de Desarrollo en el Sistema de Laboratorio Virtual y a Distancia

El SLVD está compuesto por tres partes o niveles: la interfaz de usuario, el nivel de gestión de prácticas y el procesamiento de las prácticas (Santana CHing, 2012)

- **Interfaz de Usuario**

La interfaz de usuario es una de las partes de mayor importancia dentro de un SLVD ya que mediante ella el usuario interactúa con la aplicación. Debe ser flexible, amigable y permitir un fácil uso del sistema. Además debe ser implementada de forma tal que se pueda mantener y ampliar sin dificultad. La principal función de la interfaz de usuario consiste en mostrar las prácticas disponibles en el sistema, conformar el pedido de las prácticas con todos sus datos y enviarlo a la parte encargada de gestionarlos y, por último, presentar los resultados.

- **Gestión de las prácticas**

En la capa de gestión de prácticas el sistema se encarga de enviar los pedidos de prácticas realizados a través de la interfaz de usuario al nivel de procesamiento de prácticas, para luego devolver al usuario la respuesta de la ejecución, es decir, funciona como enlace entre la interfaz de usuario y la capa de procesamiento de prácticas.

- **Procesamiento de las prácticas mediante MATLAB/Simulink**

En la actualidad se utilizan variantes de procesamiento de prácticas como por ejemplo MATLAB. MATLAB/Simulink como herramienta de procesamiento de prácticas es muy potente debido a la gran variedad de paquetes de herramientas relacionados con el control que contiene, por lo que tiene un amplio uso en el área de control automático. Además, se desarrolla una gran cantidad de software para interactuar con MATLAB, entre ellos los necesarios para lograr la ejecución de las prácticas en tiempo real, independientemente de que MATLAB tenga su paquete de herramientas de tiempo real (*Toolbox Real-Time Workshop* y *Real- Time Windows Target*). Todo esto ha propiciado que el uso de MATLAB sea una de las variantes más utilizadas.

1.3.2 Soluciones existentes de Entornos Integrados de Desarrollo en Sistema de Laboratorios Virtuales y a Distancia

Durante la última década, se ha desarrollado variedades de laboratorios virtuales, enfocados en la educación a distancia, demostrando así un gran potencial para la formación de estudiantes en todas las esferas. Permiten por medio de simulaciones y experimentos en tiempo real, observar, supervisar e interactuar con una plataforma experimental, sin la necesidad de tener todo un equipamiento físico.

A continuación se muestran algunas soluciones existentes:

- **Diseño de un Entorno de Desarrollo Integrado para una Unidad Controladora de Procesos**

Es una iniciativa de parte de la Escuela de Ingeniería Electrónica, cuyo propósito es la creación de un laboratorio en línea enfocado a temas relacionados al área del Control Automático. Este laboratorio en línea, brindará a los estudiantes la posibilidad de realizar distintos experimentos de manera remota, ya sea, por medio de un sistema de red de área local (LAN) o Internet. El presente proyecto, trata del diseño y desarrollo de un nuevo sistema de interacción con el usuario denominado Entorno de Desarrollo Integrado para una Unidad Controladora de Procesos. El proceso de diseño consistió de cinco etapas: la creación de un lenguaje intermedio, el desarrollo de un traductor de código intermedio, el diseño de una interfaz gráfica, la creación de un traductor de lenguaje gráfico a código intermedio y finalmente la implementación de la herramienta de simulación (Caravaca Mora, 2010).

- **Herramienta en línea para la programación y depuración remota de funciones lógicas digitales**

Este trabajo se enfoca en el desarrollo de estos sistemas de laboratorio basados en web, orientado a fortalecer los procesos de enseñanza/aprendizaje en el área de sistemas digitales, mediante la programación remota de funciones lógicas; esto lo hace un recurso esencial en las didácticas orientadas a la experimentación en ingeniería. El usuario podría compilar, programar y depurar sus diseños en tiempo real y en forma remota a través de una aplicación web amigable, que permita la interacción y presentación de resultados; además, estarían disponibles en servidor los archivos correspondientes al planteamiento de las prácticas de laboratorio, que el docente con permisos de administrador ha subido previamente (Vera, Zúñiga y Bernal, 2013).

Estas herramientas al ser software propietario no permiten que puedan ser empleadas en otras soluciones.

1.4 Metodologías de desarrollo de software

La metodología de desarrollo de software es parte esencial para realizar cualquier proyecto o aplicación, es como una guía que define procesos, tareas y herramientas a utilizar para la realización del mismo. Todo desarrollo de software es riesgoso y bastante difícil de controlar, de ahí la importancia de contar con una metodología que indique paso a paso las actividades que deben desarrollarse, las personas involucradas en la realización de las mismas, así como el rol que desempeñan estas en aras de lograr un proceso de software exitoso. Las metodologías de desarrollo de software se pueden clasificar en dos grandes grupos: (Martinez Ajete y Francia Ofarril, 2014)

Metodologías Tradicionales

Dan nombre a las primeras metodologías que se utilizaron para desarrollar software y que actualmente aun gozan de gran popularidad. Fue a lo largo del siglo pasado la principal representante de este grupo de metodologías, actualmente se puede considerar el modelo de proceso unificado como su principal valedora. Se caracterizan por realizar un tratamiento predictivo de los proyectos y medir el progreso en términos de artefactos entregados, así como de la especificación de los requisitos, los documentos de diseño, planes de pruebas y revisiones de código (Trigas Gallego, 2009).

Metodologías Ágiles

Las metodologías ágiles surgen como una alternativa de las metodologías tradicionales. Se caracterizan principalmente por el uso de técnicas para agilizar el desarrollo del software, así como de una mayor flexibilidad para adaptarse a los cambios en los requisitos del proyecto (Carvajal Riola, 2008).

Para justificar y fundamentar que tipo de metodología seleccionar, se realizó una comparación entre las metodologías ágiles y las tradicionales, donde se muestra en la Tabla. 1: (Abad Londoño, 2014)

Tabla. 1: Comparación entre las metodologías ágiles y las metodologías tradicionales.

Metodologías ágiles	Metodologías tradicionales

Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo de desarrollo).	Impuestas internamente.
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (menos de 10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Muchos artefactos.
Pocos roles.	Muchos roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

A partir de lo analizado hasta el momento, entre los tipos de metodologías ágiles y tradicionales, se decidió utilizar una metodología ágil, debido a que sus características son bastantes similares a las del presente trabajo, por ejemplo: pocos integrantes en el equipo de desarrollo, pocos roles, la interacción entre el cliente y el equipo de desarrollo, entre otras características.

Dentro de las metodologías ágiles sobresalen SCRUM y XP (*Extreme Programming*).

SCRUM: desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Es una metodología de trabajo (no de análisis y diseño) usada para la planificación de los proyectos, basada en gestionar proyectos de software. Esta es especialmente indicada para proyectos con un rápido cambio de requisitos. Tiene como característica el desarrollo de software, el cual se realiza mediante iteraciones, denominadas "corridas" o *sprints*. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. Exige reuniones a lo largo del proyecto para coordinación e integración. SCRUM es un proceso en el que se aplican de manera regular un conjunto de mejores prácticas lo que permite obtener el mejor resultado posible de un proyecto. Permite realizar entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto (ORM Propel, 2011).

Programación Extrema o Extreme Programming (XP): es una metodología ágil para pequeños y medianos equipos, desarrollando software cuando los requerimientos son ambiguos o rápidamente cambiantes. En XP se realiza el software que el cliente solicita y necesita, en el momento que lo precisa, alentando a los programadores a responder a los requerimientos cambiantes que plantea el cliente en cualquier momento. Esto es posible porque está diseñado para adaptarse en forma inmediata a los cambios, con bajos costos asociados, en cualquier etapa del ciclo de vida. En pocas palabras, XP está expuesta al cambio (Calabrina y Píriz, 2003).

Otra metodología ágil que sobresale es la OpenUp. Es un proceso modelo, dirigido a la gestión y desarrollo de proyectos de software basados en desarrollo iterativo, ágil e incremental apropiado para proyectos pequeños y de bajos recursos; y es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo.

1.4.1 OpenUP

OpenUP mantiene las mismas características que RUP: contiene el desarrollo iterativo, casos de uso y escenarios de conducción de desarrollo, gestión de riesgos y el enfoque centrado en la arquitectura. Tiene los componentes básicos que pueden servir de modelo a procesos específicos. Además, la mayoría de los elementos están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances.

Es una metodología de desarrollo de software de código abierto diseñado para pequeños equipos organizados tomando una aproximación ágil del desarrollo, es iterativa, mínima porque solo incluye el

contenido del proceso fundamental, se caracteriza por ser un proceso completo ya que puede ser manifestado como proceso entero para construir un sistema y se comporta de forma extensible porque puede ser utilizado como base para agregar o para adaptar más procesos (Fernandes Agrela y Miranda Bousas, 2009).

El uso de esta metodología trae consigo beneficios como por ejemplo permite disminuir las probabilidades de fracaso en los proyectos pequeños e incrementar las probabilidades de éxito, además permite detectar errores tempranos a través de un ciclo iterativo. OpenUP se rige por varios principios entre los que se encuentran colaborar para sincronizar intereses y compartir conocimiento, que no es más que promover buenas prácticas para el trabajo en equipos.

Los roles propuestos para la solución son los roles de analista y desarrollador. El analista es el encargado de recopilar los requisitos propuestos por el cliente y de entender el problema a resolver. Mientras que el desarrollador es el encargado de desarrollar la aplicación, realiza el diseño teniendo en cuenta la arquitectura utilizada y las pruebas unitarias y de integración.

Por todo lo antes descrito se decide utilizar la metodología OpenUp, ya que esta es la que más se adapta a la presente investigación, es utilizada en proyectos sencillos, con bajo presupuesto y disminuye la probabilidad de fracaso de los mismos e incrementa las probabilidades de éxito. Permite detectar errores en fases tempranas a través de un ciclo iterativo. Además es un proceso de desarrollo de software mínimamente suficiente, que incluye solo el contenido fundamental, además no provee orientación sobre temas en los que el proyecto tiene que lidiar, como son: el tamaño del equipo, el cumplimiento, seguridad, orientación tecnológica, entre otras. Presenta un enfoque centrado al cliente y con iteraciones cortas.

1.5 Herramientas y técnicas utilizadas

En el proceso desarrollo de la aplicación es fundamental la correcta elección de las tecnologías y herramientas que mejor se adapten, teniéndose en cuenta las tendencias actuales y las novedades de cada una de estas. Para ello se tuvo en cuenta el uso de estándares y tecnologías que serán utilizadas en el IDE.

- **Lenguaje de programación**

Java 2 Enterprise Edition

Actualmente existen muchos lenguajes de programación para desarrollar aplicaciones web, dependiendo del lado en el que se ejecutan, se pueden clasificar en lenguajes del lado del cliente o lenguajes del lado del servidor, siendo estos últimos los más importantes en el desarrollo de aplicaciones. Entre los más utilizados se pueden citar PHP, Python, C# y Java (King, 2010).

La JSR-58 (*Java Specification Request*), más conocida como Plataforma Java 2 Enterprise Edition (J2EE), es la extensión de Java para el desarrollo web, representa la evolución de la plataforma de desarrollo del lado del servidor hacia una especificación madura y sofisticada (Allamaraju y Beust, 2011).

J2EE es considerada la mejor alternativa a la tecnología ASP.NET de Microsoft. Las aplicaciones desarrolladas bajo esta plataforma pueden ser desplegadas en cualquier sistema operativo. J2EE es una unidad de software funcional que está ensamblada dentro de las aplicaciones Java con sus clases relacionadas y archivos que lo comunican con otros componentes, dispone de varias herramientas de código abierto como IDEs, servidores, marcos de trabajo y APIs (por sus siglas en inglés, *Applications Programming Interface*). Posibilita una serie de librerías que facilitan el trabajo de los desarrolladores, tales como *Java Database Connectivity* (JDBC), *Enterprise Java Beans* (EJB), *Java Server Pages* (JSP), *Java Tag Library* (JSTL) y *Java Message Service* (JMS). Cuenta con varios marcos de trabajo que facilitan el desarrollo de las aplicaciones, tales como *Spring, Struts* y *Java Server Faces* (King, 2010).

JavaScript: es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Eguíluz Pérez, 2010).

Hoja de estilo en cascada (CSS, por sus siglas en inglés): CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con Lenguaje basado en marcas (HTML, por sus siglas en inglés). Es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. Se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página (Eguiluz, 2015).

Java Server Pages (JSP, por sus siglas en inglés): es una tecnología que permite incluir código Java en páginas web. El denominado contenedor JSP (que sería un componente del servidor web) es el encargado de tomar la página, sustituir el código Java que contiene por el resultado de su ejecución y enviarla al cliente. Así, se pueden diseñar fácilmente páginas con partes fijas y partes variables (Depto. CCIA, 2005).

- **Portlet**

Dentro de J2EE existen componentes que permiten combinar varias potencialidades de la plataforma, incluyendo el desarrollo de interfaces de usuario, ejemplo de ello lo constituyen los portlets.

Un Portlet, es un componente de interfaz de usuario reutilizable, que provee una vista de la información del sistema. Estos componentes producen fragmentos de lenguaje de marcado, que los dispone para ser agregados dentro de páginas en portales (Clay Richardson, 2004).

El Portlet tiene un ciclo de vida y según la especificación Java Portlet (JSR168) está basado en tres fases: (Moreño, 2011)

1. Inicio (*Init*): El usuario, al interactuar con el portal arranca el Portlet activando el servicio.
2. Gestión de peticiones (*Handle requests*): Procesa la petición mostrando diferentes informaciones y contenido según el tipo de petición. Los datos pueden redimir en sistemas diferentes. Dentro de esta fase se encuentra la Presentación, en la que el Portlet da salida a la información en código de hipertexto para su visualización en el navegador.
3. Destrucción (*Destroy*): Elimina el Portlet cuyo servicio deja de estar disponible.

- **Contenedores de Portlets en la plataforma J2EE**

En la plataforma J2EE existen varios contenedores de portlets que son capaces de manejar el ciclo de vida de un Portlet, entre los más utilizados están Apache Jetspeed, JBoss Portal y Liferay Portal. El primero es una plataforma de información basada en un portal, escrito completamente bajo código abierto, el segundo proporciona un entorno de código abierto, para alojar y servir aplicaciones en una interfaz web de portal, es además un CMS con múltiples funciones, mientras el último, goza de prestigio a nivel internacional (JBoss, 2007).

- **Liferay Portal**

Liferay Portal es un Sistema de Gestión de Contenidos basado en Java que permite la creación de portales Web de una manera sencilla y rápida. Cuenta con más de 60 portlets listos para su utilización. Liferay es una plataforma para el desarrollo, la integración y la colaboración. Ofrece todas las características necesarias para la creación de portales y aplicaciones web, todo ello desarrollado sobre una plataforma de código abierto completamente equipada (Scamercio, 2010).

Liferay Portal basa su contenido en portlets. Incluye muchas aplicaciones portlets tales como mensajería instantánea, foros y biblioteca de documentos. Soporta múltiples base de datos como: (Jonas, 2008)

PostgreSQL, MySQL, Oracle, SQL Server, Sybase e *InterBase*, se puede desplegar con muchos servidores como *Jetty, JBoss, Sun GlassFish, Oracle AS* y *Apache Tomcat*, es multiplataforma ya que puede ejecutarse en cualquier sistema operativo como Windows y Linux.

- **Marco de trabajo**

El desarrollo de aplicaciones promueve la reutilización de código, pues muchas de estas tienen funcionalidades iguales o muy parecidas a las de otra. El objetivo es aprovechar funciones generales o que ya están implementadas, a menudo estas funciones son agrupadas en librerías o marcos de trabajo.

Un marco de trabajo (*framework*, en inglés) es una pieza de software estructural. Se dice estructural porque la estructura es el objetivo de un framework que especifica todo requerimiento funcional. Un marco de trabajo trata de hacer generalizaciones acerca de las tareas comunes e intenta proveer una plataforma donde las aplicaciones pueden ser rápidamente construidas (Chad y Brown, 2007).

En la plataforma J2EE existen marcos de trabajos para diversos propósitos, dígase manipular bases de datos, desarrollar aplicaciones web, visualizar datos y gestionar peticiones AJAX (acrónimo del inglés *Asynchronous JavaScript And XML*).

- **Marcos de trabajos para aplicaciones web**

En la Tabla. 2 se hace una comparación entre estos marcos de trabajos para saber cuál es el más adecuado para desarrollar la aplicación (Yuan, 2012).

Tabla. 2: Comparación entre los marcos de trabajos Spring y Seam.

Desarrollo	Spring	Seam

Plugin IDE	SpringIDE	JBossTools
Documentación	Es muy completa y exhaustiva	Existe buena documentación
Pruebas	Spring tiene completamente integrado el testing. Al estar basado en Programación Orientada a Objetos (POO), se adapta perfectamente a JUnit. Para realizar las pruebas de integración, existen herramientas basadas en la capacidad de inyección de dependencias y de transaccionalidad.	Al ser POO todos los componentes, las pruebas unitarias resultan triviales. Con anotaciones, se logra recrear el entorno de forma sencilla. La herramienta SeamGen crea de forma automática los test unitarios y test TestNG permite gestionar simulaciones y respuestas JSF para cada acción mediante scripting.
Configuración	Ofrece la ventaja de "XML extensible". Spring permite que el resto de frameworks implicados se integren configurando un xml de configuración	Se realiza básicamente sobre anotaciones. Aun así, existe parte de la configuración que debe hacerse con XML (como la relativa a JSF). Esta pequeña parte es autogenerada por la herramienta SeamGen.
Navegación	Posee un módulo propio que controla la navegación denominado *Spring Web Flow*. Con él, se permite capturar los flujos de las páginas e integrarlo con otros frameworks (Struts, JSF, etc)	Permite elegir entre dos modos de navegación. El primero se basa en el estado de la aplicación usando el framework jPDL y el segundo basado en las reglas de navegación JSF Rules o Seam Rules.

Seguridad	Spring posee un framework de seguridad asociado, Spring security, que gestiona todo el mecanismo de login, autentificación y autorización	El Seam Security API es una parte de Seam que proporciona funcionalidades de autenticación y autorización basado en JAAS (*Java Authentication and Authorization Service*). Puede usarse el modo sencillo, por defecto, que se basa en roles o usar el framework JBoss Rules, que ofrece un sistema más poderoso basado en reglas. El sistema se encarga del manejo de errores de autorización o autenticación, permitiendo redirigir al usuario a una página determinada. Las restricciones pueden establecerse mediante el uso de anotaciones. Permite restringir tanto acciones como entidades o páginas o componentes de la página.
Servidor de aplicaciones	Si	Si

- **Spring Framework**

Uno de los marcos de trabajo más populares en la comunidad de Java es Spring Framework, este incluye varias librerías que ayudan a solucionar problemas comunes en el desarrollo de aplicaciones. Spring soporta la Programación Orientada a Aspecto e implementa el patrón Modelo-Vista-Controlador, separando la interfaz de usuario, los controladores y la capa de acceso a datos. Emplea el Objeto de Acceso a Datos (DAO, siglas en inglés de *Data Access Object*), que es un patrón de diseño para gestionar las bases de datos (Minter, 2008)

Spring es un marco de código abierto, creado por Rod Johnson y descrito en su libro Expert One-on- One: J2EE Design and Development. Fue creado para hacer frente a la complejidad del desarrollo de

aplicaciones empresariales. Sin embargo, la utilidad de Spring no se limita al desarrollo del lado del servidor. Cualquier aplicación Java puede beneficiarse de Spring en términos de simplicidad, capacidad de prueba y el acoplamiento débil (Craig y Breidebach, 2005)

Por lo antes descrito se escogió el *framework Spring* ya que permite producir código limpio, mejora la configuración de las aplicaciones, da soporte a la programación orientada a aspectos, tiene una escala de seguridad mediante la integración de *Spring Security.*

- **Servidores Web de Java**

Un servidor web no es más que un programa que se ejecuta de forma continua en un ordenador manteniéndose a la espera de peticiones por parte de un cliente.

Jetty

Jetty es un servidor web escrito totalmente en Java que incluye, un contenedor de Servlets[2]. Tiene un tamaño reducido y un rendimiento alto, lo que lo ha convertido en uno de los preferidos para desarrollar productos embebidos que requieran un servidor HTTP. No suelen encontrarse funcionando por si mismos sino integrados con servidores de aplicaciones como JBoss y Jonas. También ejecutándose en múltiples sistemas embebidos y ordenadores de mano (Mateu, 2010).

JBoss

Es un servidor de aplicaciones J2EE de código abierto implementado en Java puro. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo que lo soporte. Los principales desarrolladores trabajan para una empresa de servicios, JBoss Inc, adquirida por Red Hat en Abril del 2006, fundada por Marc Fleury, el creador de la primera versión de JBoss. El proyecto está apoyado por una red mundial de colaboradores. Los ingresos de la empresa están basados en un modelo de negocio de servicios. JBoss implementa todo el paquete de servicios de J2EE (Alferez Sanchez, 2010)

Las características destacadas de JBoss incluyen:

[2] es una clase del lenguaje de programación Java que es usada para extender la habilidad de los servidores de albergar aplicaciones que son accesibles bajo el modelo de programación petición-respuesta.

1. Producto de licencia de código abierto sin coste adicional.
2. Flexibilidad consistente.
3. Confiable a nivel de empresa.
4. Incrustable, orientado a arquitectura de servicios.

- **Apache Tomcat**

Apache Software Foundation es la organización que más prestigio tiene debido a que sus proyectos contienen gran estabilidad y confiabilidad. Dentro de los servidores web que tiene esta organización se encuentra el Apache Tomcat.

Apache Tomcat es un servidor web, de código abierto que une la tecnología *Java Servlet* y *Java Server Pages.* Es mantenido y desarrollado por miembros de la *Apache Software Foundation* y voluntarios independientes. Fue escrito en Java, por lo que funciona en cualquier sistema operativo que disponga de la máquina virtual. Es un producto muy robusto, altamente eficiente y uno de los más potentes contenedores de Servlets existentes. Su único punto débil reside en la complejidad de su configuración, dado el gran número de opciones existentes. Es desarrollado y publicado bajo la licencia Apache 2.0. Permite montar servicios web mediante Axis2 (Durán y Medel, 2012)

Se escogió *Apache Tomcat* como servidor web ya que al ser un servidor HTTP posee menos memoria, mientras que los servidores Java EE pesan cientos de megas. Tomcat es muy popular para aplicaciones web o aplicaciones que utilizan como *frameworks Spring* que no requieren de un completo servidor de aplicaciones Java EE.

- **Entorno Integrado de Desarrollo. Eclipse-Luna**

Para el lenguaje Java son empleados en la actualidad diferentes entornos de desarrollos, cada uno con sus particularidades y características. Algunos de estos entornos de desarrollo son Java Studio**,** Java Studio, Eclipse, entre otros. A continuación se describe este último entorno de programación.

Es un poderoso Entorno de Desarrollo, donde inicialmente fue creado por *International Business Machines Corp* (IBM) para el desarrollo de aplicaciones utilizando Java, actualmente su desarrollo es llevado a cabo por la Fundación Eclipse, una organización independiente y sin ánimos de lucro que fomenta una comunidad de código abierto, así como una serie de servicios. Eclipse se encuentra basado en componentes para brindar todas sus funcionalidades, a diferencia de otros entornos monolíticos que

incluyen todas las funcionalidades, las necesite el usuario o no y permite extenderse a otros lenguajes como C++ y PHP (Jiménez Milia y Ferrer Obregón, 2008).

- **CodeMirror**

CodeMirror es un editor de código fuente que funciona en el navegador, ya que está implementado en JavaScript. Trae soporte para numerosos lenguajes de programación y *addons*[3] para añadirle funcionalidades de edición más avanzadas. Tiene disponible una API de programación y temas CSS para personalizar la aplicación y extenderla para añadirle una nueva funcionalidad. Está soportado por todos los navegadores actuales que soporten HTML5: Firefox desde la versión 3, Chrome en cualquiera de sus versiones, Safari desde la versión 5.2, Internet Explorer desde la versión 8, Opera desde la versión 9. Dentro de las principales características se pueden destacar: (CodeMirror, 2015)

1. Soporta hasta 60 lenguajes de programación por defecto.
2. Autocompletado (XML).
3. Asociaciones de teclas configurables.
4. Interfaz de buscar y reemplazar.
5. Emparejamiento de llaves y etiquetas.
6. Posibilidad de usar distintos estilos de letra y tamaños en el mismo editor.
7. Varios temas.
8. Posibilidad de redimensión para ajustarse al contenido.
9. Interfaz
10. Automatizar actividades manuales y mejora la visión general de la ingeniería ayudando a garantizar completamente programable.
11. Posibilidad de crear zonas de texto de solo lectura.

- **Herramientas CASE. Visual Paradigm**

Las herramientas de Ingeniería del Software Asistida por Computadora (por sus siglas en inglés, CASE) proporcionan la posibilidad de la calidad del producto antes de llegar a construirlo. Tienen como objetivo

[3] son programas que sólo funcionan anexados a otro y que sirven para incrementar o complementar sus funcionalidades

incrementar la productividad y la calidad del ciclo de vida de un proyecto, mejorar la planificación del mismo, así como reducir el tiempo y coste de su desarrollo. Existen numerosas herramientas CASE, algunas de ellas son: *Umbrello, Rational Rose* y *Visual Paradigm* (Pressman, 2001).

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite representación de todo tipo de diagramas, generando código a partir de ellos.

Su objetivo fundamental es reducir el tiempo de desarrollo de analistas, arquitectos, diseñadores y desarrolladores. Es una herramienta multiplataforma que proporciona soporte a varios lenguajes en generación de código e ingeniería inversa a través de plataformas Java (Sierra, 2013).

- **MATLAB**

MATLAB (*Matrix Laboratory*): es un programa interactivo de uso general. Es un instrumento computacional simple, versátil y de gran poder para aplicaciones numéricas, gráficas y contiene gran cantidad de funciones predefinidas para aplicaciones en ciencia e ingeniería. La interacción se realiza mediante instrucciones (denominados comandos) y también mediante funciones y programas en un lenguaje estructurado. Los objetivos básicos con los cuales opera MATLAB son matrices. La asignación de memoria a cada variable la realiza de forma dinámica y eficiente, por las que no son necesarias las declaraciones de variables antes de su uso (Rodríguez Ojeda, 2007).

Características de MATLAB:

1. Cálculo numérico rápido y con alta precisión.
2. Capacidad para manejo matemático simbólico.
3. Programación mediante un lenguaje de alto nivel.
4. Facilidades básicas para diseño de una interfaz gráfica.
5. Extensa biblioteca de funciones.
6. Paquetes especializados para algunas ramas de ciencia e ingeniería.
7. Soporte para programación estructurada y orientada a objeto.

Operación:

1. Simple y eficiente.
2. Interactivo y programable.
3. Sistema de ayuda en línea.

4. Interacción con otros entornos.

Uso interactivo de MATLAB:

El entorno de MATLAB está organizado mediante ventanas. Las principales son:

Command Window: es la ventana de comandos para interactuar con MATLAB.

Command History: contiene el registro de comandos que han sido ingresados.

Wordspace: contiene la descripción de las variables usadas en cada sección.

Como parte de la solución se realizó un **servicio web.**

Un servicio web se define como un mecanismo de comunicación distribuida que permiten que las aplicaciones compartan información y que además invoquen funciones de otras aplicaciones independientemente de cómo se hayan creado las aplicaciones, cuál sea el sistema operativo o la plataforma en que se ejecutan y cuáles los dispositivos utilizados para obtener acceso a ellas (Arias y Fernández, 2010).

Utilidades de los servicios web (Arias y Fernández, 2010)**.**

Permiten que varias aplicaciones compartan información e invoquen funciones de otras aplicaciones independientemente de cómo hayan sido creadas (lenguaje de programación), cómo se ejecutan (sistema operativo y plataforma), los dispositivos utilizados para acceder a ellas.

Crean una especie de red informática mundial (WWW, por sus siglas en inglés) paralela para máquinas:

- WWW humana (personas accediendo a páginas Web)
- WWW cibernética (aplicaciones accediendo a servicios Web)

Para la realización del servicio web fue utilizado GlassFish, el mismo es un servidor de aplicaciones desarrollado por Sun Microsystems que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación. Soporta las últimas versiones de tecnologías Servlets, Java API para Servicios Web (JAX-WS), Arquitectura Java para Enlaces XML (JAXB), Metadatos de Servicios Web para la Plataforma Java y muchas otras tecnologías (Manchado Serra y Franco Puntes, 2010).

Conclusiones parciales

- En el presente capítulo se estudiaron los Sistemas de Laboratorios Virtuales existentes identificándose que no implementan Entornos Integrales para el desarrollo de las prácticas.
- Una vez estudiada las tecnologías, herramientas y metodologías para el Proceso de desarrollo de software se identificó el Eclipse para el desarrollo de la aplicación como entorno de desarrollo.
- Se fundamentó la selección de Liferay Portal como portal de aplicaciones y contenedor de Portlet, Apache Tomcat para montar el servidor web y como marco de trabajo se optó por Spring.
- Se analizaron las principales metodologías de desarrollo de software que existen en la actualidad y se determinó utilizar como guía para la investigación, la metodología ágil OpenUP teniendo en cuenta las características del sistema, lo que permitió llevar a cabo un correcto proceso de desarrollo de software.
- Se definió como herramienta CASE Visual Paradigm 8.0, para permitir la realización con facilidad de los procesos de modelado e implementación del IDE.

Capítulo 2: Análisis y Diseño del Entorno de Desarrollo

En este capítulo se realiza el análisis y diseño de la solución. En él quedan definidas todas las funcionalidades y características que debe cumplir el IDE a través de los Requisitos Funcionales y Requisitos no Funcionales. Se identifican los casos de usos del sistema, realizando la descripción de cada uno de ellos, agrupando dentro de estos, los diferentes requisitos funcionales. Se modela el diagrama de clases del diseño, abordando acerca de los patrones utilizados, patrones de diseño y patrones arquitectónicos.

2.1 Propuesta de la solución

En este trabajo se desea desarrollar un Entorno Integrado de Desarrollo que permita la elaboración de prácticas de Control Automático en un Sistema de Laboratorios Virtuales y a Distancia. El IDE estará basado en portales y permitirá establecer una conexión a MATLAB mediante JMATLAB[4], donde el mismo debe permitir al usuario realizar prácticas de Control Automático.

2.2 Descripción del modelo conceptual

Un modelo conceptual es una representación de conceptos en un dominio del problema, permite descomponer el problema en unidades comprensibles como conceptos, permite esclarecer un conjunto de términos del dominio. En este modelo se pueden ver cuáles son los términos importantes y cómo se comunican entre ellos (Larman, 1999).

En la presente investigación se realiza el modelo conceptual ver Fig. 1, el cual se descompuso el sistema en varios conceptos fundamentales, lo que permite lograr una mayor comprensión del problema a resolver. A continuación se expone el diagrama de la investigación:

[4] Librería que permite conectarse a MATLAB, fue creada por el profesor Edel Moreno Lemus del Departamento de Desarrollo de Componentes.

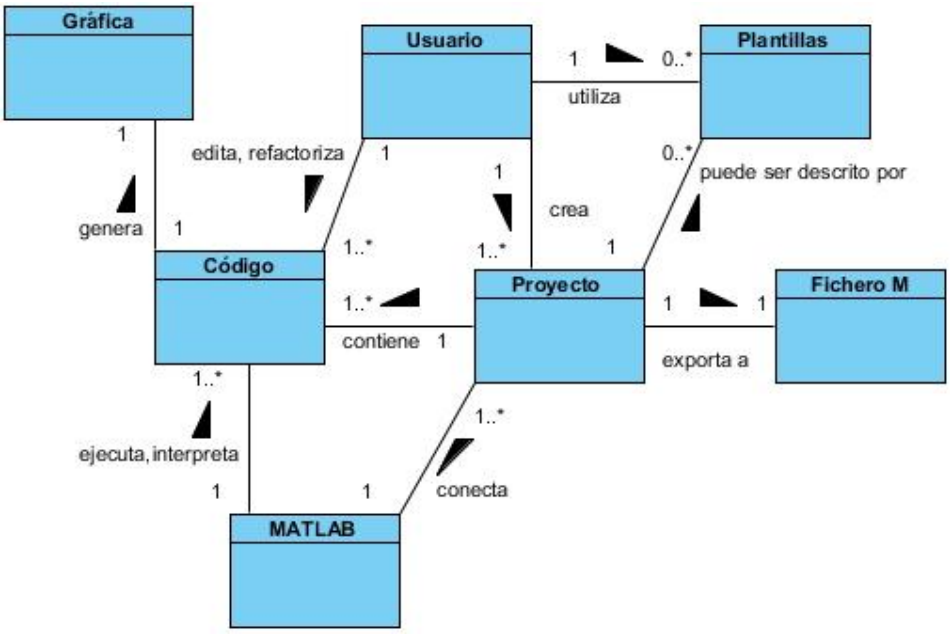

Fig. 1: Modelo conceptual.

A continuación en la Tabla. 3 se muestra las descripciones de cada uno de los conceptos asociados al modelo.

Tabla. 3: Descripción de los elementos del modelo de conceptual.

Usuario	Persona que interactúe con el software.
Plantillas	Redacción de plantillas de código, que al crearse un nuevo programa este inicializa con el código, de forma tal que se agiliza el proceso de desarrollo.
Código	Código introducido por el usuario al crear un proyecto.
Gráfica	Gráfica que se genera del código.
Fichero m	Fichero de extensión m que se genera cuando se exporta el proyecto.
MATLAB	Programa encargado de interpretar y ejecutar el código.
Proyecto	Nuevo proyecto creado por el usuario el cual contiene plantillas.

2.3 Especificación de requisitos

La especificación de requisitos es un proceso fundamental para el desarrollo de cualquier software. Este tiene como principales objetivos permitir a los clientes describir con claridad lo que desea obtener, una vez terminado el software, por lo que el cliente debe tener activa participación durante este proceso. Por otra parte, permite a los integrantes del equipo de desarrollo construir un software que satisfaga las necesidades del cliente.

Una buena especificación de requisitos de software ofrece una serie de ventajas entre las que destacan el contrato entre cliente y desarrolladores, reduce el esfuerzo en el desarrollo, brinda una base para la estimación de costes y planificación, además de ser un punto de referencia para procesos de verificación y validación, y sirve como base para la identificación de posibles mejoras en los procesos analizados (Monferrer Agut, 2001).

2.3.1 Requisitos funcionales

Los requisitos funcionales son los encargados de describir las determinadas funciones o condiciones que debe brindar el sistema. Describen además las posibles entradas y salidas, así como la respuesta que tiene el sistema ante algunas situaciones que se presenten en su interacción con el cliente.

Para la realización de la investigación se identificaron los siguientes Requisitos Funcionales (RF):

- **RF1:** Completar código en el editor de texto.
- **RF2:** Refactorizar código fuente en el editor de texto.
- **RF3:** Ejecutar código fuente.
- **RF4:** Crear nuevo proyecto mediante una plantilla.
- **RF5:** Exportar proyecto a la extensión .m de MATLAB.
- **RF6:** Generar código fuente de una función seleccionada.
- **RF7:** Mostrar gráfica de coordenadas de una función.

2.3.2 Requisitos no Funcionales

Los requisitos no funcionales son las propiedades o cualidades que el producto debe presentar. Estos requisitos definirán las características del producto final y muchas veces están implicados en el éxito final que se desea alcanzar.

Para la realización de la investigación se identificaron los siguientes Requisitos no Funcionales (RnF):

- **Requisitos de Software**

Cliente

Debe tener una computadora con Sistema Operativo (Ubuntu LTS 14.04 o Windows7 o superior) que cuente con navegador web, donde se recomienda que sea Mozilla Firefox 32.0 o superior.

Servidor

Debe tener un servidor con Sistema Operativo Ubuntu LTS 14.04 o Windows7 o superior y se requiere que cuente con Liferay Portal 6.2 con Servidor Web apache-tomcat-7 o superior, debe tener instalado MATLAB en su versión 7.6.0.324 (R2008a).

- **Requisitos de Hardware**

Cliente

Debe poseer una tarjeta de red a 100 Megabytes (Mb) o superior, donde se requiera que sea un Procesador Intel Pentium IV o superior, de 1Gb de RAM o superior.

Servidor

Debe poseer un Procesador Corei3 o superior que cuente con una memoria RAM 2.0 Gigabytes (Gb) o superior y que tenga de capacidad de disco duro 80 Gb libres o superior. Requiere una tarjeta de red a 100 Megabytes (Mb) o superior.

2.4 Modelo de casos de uso del sistema

El modelo de casos de uso del sistema representa las relaciones existentes entre actores y casos de uso. Los actores son terceros fuera del sistema que interactúan con él, puede ser cualquier persona, individuo, grupo, entidad, organización, máquina o sistema de información externo; con los que el sistema interactúa y los casos de uso son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de

valor para sus actores (López Duque y Oceguera Ravelo, 2010). Durante la investigación fue identificado un único actor del sistema, que se describe a continuación:

Tabla. 4: Descripción de los actores del sistema.

Actor	Descripción
Usuario	Persona que puede ejecutar, completar y refactorizar el código, exporta el proyecto a la extensión m, crea proyecto mediante plantillas y grafica una función.

A partir de los siete RF antes descritos, se pudieron confeccionar los siguientes Casos de Usos (CU), los cuales se mencionan a continuación:

- **CU1:** Completar código en el editor de texto.
- **CU2:** Refactorizar código fuente en el editor de texto.
- **CU3**: Ejecutar código fuente.
- **CU4:** Crear nuevo proyecto mediante una plantilla.
- **CU5:** Exportar proyecto a la extensión .m de MATLAB.
- **CU6:** Generar código de una función seleccionada.
- **CU7**: Mostrar gráfica de coordenadas de una función.

Diagrama de caso de uso del sistema

En la Fig. 2 se muestra el diagrama de casos de uso del sistema que describe las acciones que el usuario realiza mediante la utilización del IDE, así como las funciones que este brinda.

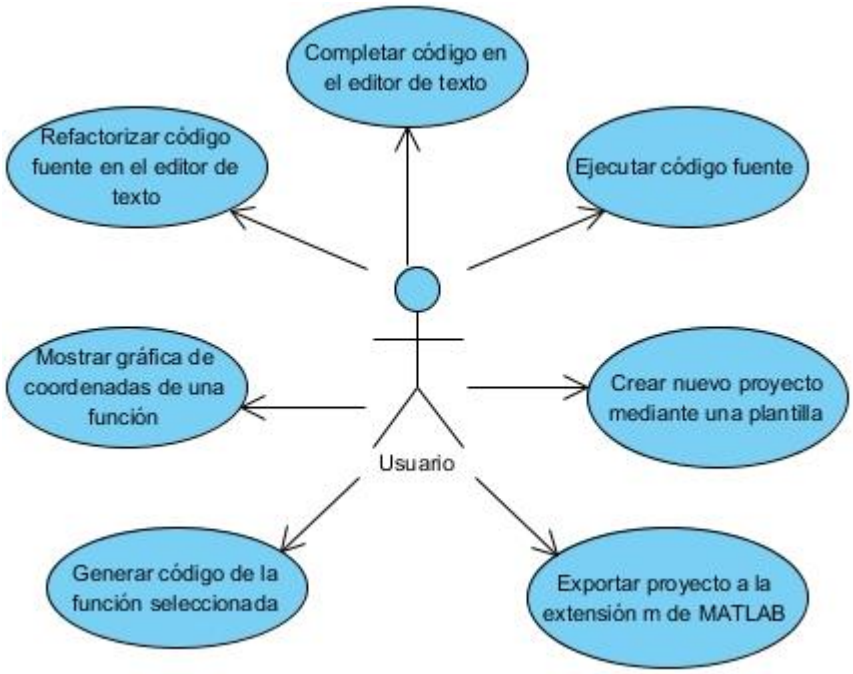

Fig. 2: Diagrama de caso de uso del sistema.

2.4.1 Descripción textual de los casos de uso del sistema

Cada CU es descrito por a partir de su descripción textual. A continuación se presenta en la Tabla. 5 un ejemplo textual del CU ¨Ejecutar código fuente¨.

Tabla. 5: Descripción textual del caso de uso " Ejecutar código fuente".

Caso de Uso	*Ejecutar código fuente*
Actores	*Usuario*
Resumen	*El caso de uso inicia cuando el usuario ejecuta el código y el sistema debe mostrar el resultado.*
Complejidad	*Alta*
Prioridad	*Crítico*
Precondiciones	
Postcondiciones	
Referencias	*RF3*
Flujo de eventos	
Flujo básico: *Ejecutar código fuente*	

	Actor	**Sistema**

1.	*Escribe el código y presiona el botón ejecutar.*	
2.		*Compila el código y verifica que el código entrado esté correcto.*
3.		*El sistema muestra que el código ha sido ejecutado satisfactoriamente mostrando en un mensaje el resultado.*
Flujos alternos: *Códigos incorrectos*		
	Actor	**Sistema**
2a		*Muestra un mensaje de error y especifíca que tipo de error ocurrió.*

Prototipo de Interfaz

Relaciones	**CU Incluidos**	*No procede*
	CU Extendidos	*No procede*

Requisitos no funcionales	

2.5 Arquitectura de Software

La arquitectura de software es importante porque afecta el desempeño y la potencia, así como la capacidad de distribución y mantenimiento de un sistema. Los componentes individuales implementan los requerimientos funcionales del sistema. Los requerimientos no funcionales dependen de la arquitectura del sistema, es decir, la forma en que dichos componentes se organizan y se comunican. En muchos sistemas, los requerimientos no funcionales están también influidos por componentes individuales, pero no hay duda de que la arquitectura del sistema es la influencia dominante (Sommerville, 2011).

2.6 Patrón arquitectónico

Un patrón arquitectónico es de vital importancia ya que la estructura de un sistema influye directamente sobre la capacidad que presenta un sistema para satisfacer los atributos de calidad. Algunos de estos ejemplos de estos atributos de calidad son el tiempo de respuesta del sistema a las peticiones que se le hacen, la usabilidad, que tiene que ver con qué tan sencillo les resulta a los usuarios realizar determinadas operaciones con el sistema, como determinar qué tan sencillo es realizar cambios al mismo (Cervantes, 2010)

2.6.1 Patrón arquitectónico Modelo Vista Controlador (MVC)

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes diferentes: el modelo, la vista y el controlador. El patrón MVC se ve frecuentemente en aplicaciones web (Trott y Shalloway, 2000).

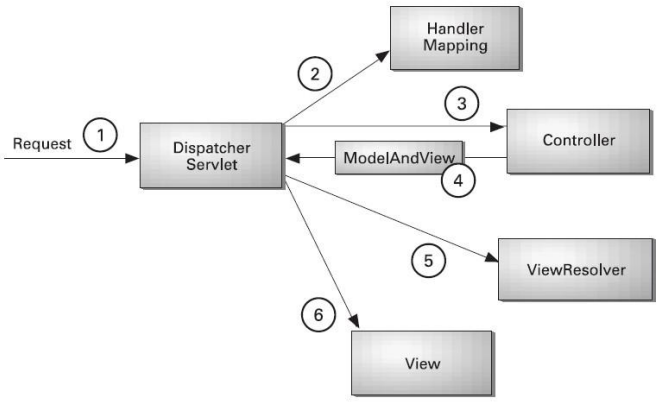

Fig. 3: Patrón Arquitectónico Modelo-Vista-Controlador.

Modelo: Maneja los datos del sistema y las operaciones asociadas a esos datos.

Vista: Define y gestiona cómo se presentan los datos al usuario.

Controlador: Dirige la interacción del usuario.

Ventajas del patrón MVC (Sommerville, 2011)

- Permite que los datos cambien de manera independiente de su representación y viceversa.
- Soporta en diferentes formas la presentación de los mismos datos, y los cambios en una representación se muestran en todos ellos.
- Clara separación entre interfaz, lógica de negocio y de presentación.
- Sencillez para crear distintas representaciones de los mismos datos.
- Reutilización de los componentes.
- Simplicidad en el mantenimiento de los sistemas.
- Facilidad para desarrollar prototipos rápidos.
- Los desarrollos suelen ser más escalables.

Desventajas del patrón MVC (Sommerville, 2011)

- Puede implicar código adicional y complejidad de código cuando el modelo de datos y las interacciones son simples.
- La curva de aprendizaje para los nuevos desarrolladores se estima mayor que la de modelos más simples.
- La distribución de componentes obliga a crear y mantener un mayor número de ficheros.

El patrón MVC es muy utilizado hoy en día, es un diseño que ofrece consistencia y baja complejidad en el desarrollo de software. En esencia el concepto, es tomar una capa intermedia para comunicar los datos o la lógica del negocio con la vista, entonces si se quiere modificar la vista no se tiene que modificar la lógica de negocio y viceversa.

Presentación de la Vista Lógica del sistema

Para comprender la estructura y la organización del diseño del sistema, se utiliza una Vista lógica de la arquitectura donde ilustra las realizaciones de subsistemas, paquetes y clases que abarcan el comportamiento significativo arquitectónicamente. En la Fig. 4 se muestra la Vista lógica del sistema.

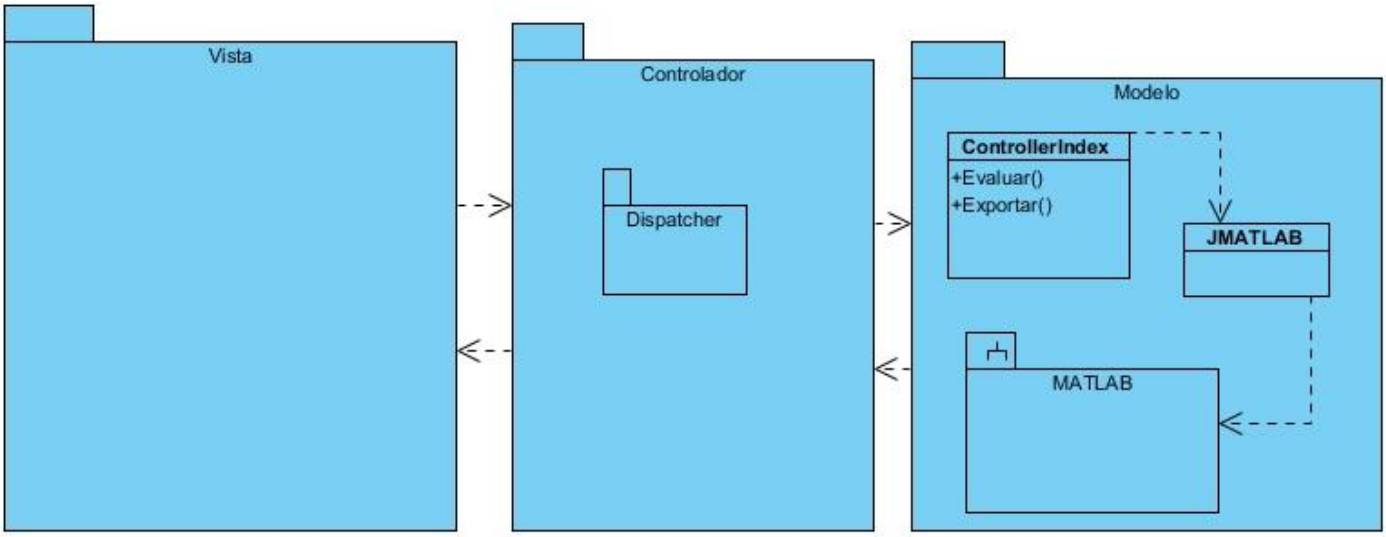

Fig. 4: Vista Lógica del Sistema.

A continuación se realiza la descripción de cada uno de los paquetes por la cual va a estar conformada la vista:

Vista: En este paquete se encuentran las clases que implementa la vista del sistema. Utilizando las Páginas Servidoras, Páginas Clientes y Formularios correspondientes para cada acción.

Controlador: En este paquete se encuentra la clase Dispatcher, es la encargada de controlar el flujo central de eventos provenientes de la capa Vista hacia la capa del Modelo.

Modelo: Es el paquete donde se encuentra la clase Index.java que es la lógica del sistema, la clase JMATLAB.java que es la encargada de permitir la comunicación entre las aplicaciones para compartir

información, el subsistema MATLAB es de la que se van a extraer los datos para calcular y graficar las funciones.

2.7 Modelo del diseño del sistema

El modelo de diseño detalla los modelos de análisis, tomando en cuenta todas las implicaciones y restricciones técnicas. El propósito es especificar una solución que trabaje y pueda ser fácilmente convertida en código fuente y construir una arquitectura simple y extensible. Las clases definidas en el análisis se detallan y se añaden nuevas clases para manejar áreas técnicas como bases de datos, interfaces de usuario, dispositivos, entre otros.

2.7.1 Diagrama de clases del diseño del sistema

Los diagramas de clases del diseño describen gráficamente las especificaciones de las clases del software y de las interfaces en una aplicación (Larman, 1999). Son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

A continuación en la Fig. 5 se realiza el diagrama de clases del diseño para el CU “Ejecutar código fuente”.

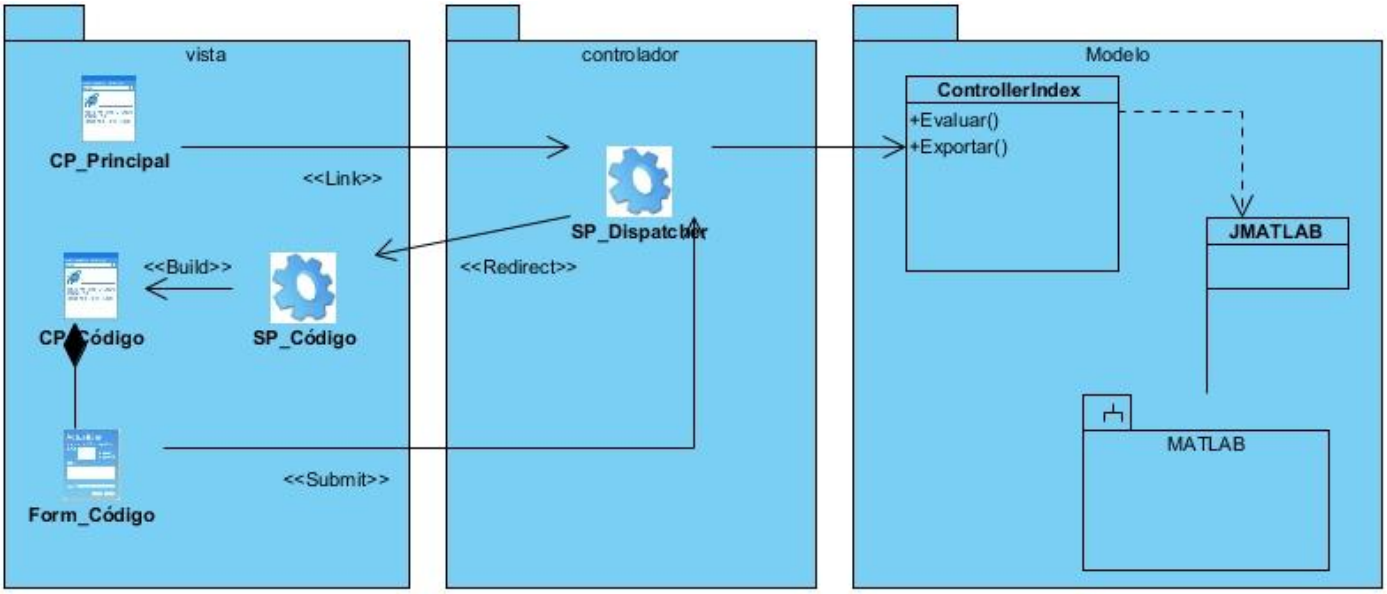

Fig. 5: Diagrama de clases CU "Ejecutar código fuente”.

2.7.2 Patrones de diseño

Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos (Mendoza, 2013). Un patrón de diseño es una solución repetible a un problema recurrente en el diseño de software. Esta solución no es un diseño terminado que puede traducirse directamente a código, sino más bien una descripción sobre cómo resolver el problema (Visconti y Astudillo, 2011).

En resumen los patrones de diseños se pueden definir de la siguiente manera:

1. Describen un problema recurrente y una solución.
2. Cada patrón nombra, explica, evalúa un diseño recurrente en sistemas orientados a objetos.

En la construcción del diseño de la aplicación informática a implementar, se utilizaron los patrones GRASP (acrónimo del inglés *General Responsibility Assignment Software Patterns*). Se considera que más que patrones son una serie de "Buenas Prácticas" de aplicación recomendable en el diseño de software.

Para la realización fueron necesarios utilizar varios de estos patrones de diseño, a continuación se mencionan los que fueron utilizados:

Patrones GRASP

- **Creador**

El patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases, guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos, el propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento (Veloso Hernández, 2003).

Aplicación: En la clase ControllerIndex.java se evidencia este patrón ya que es la clase encargada de crear instancias de la ConectarProxy.java

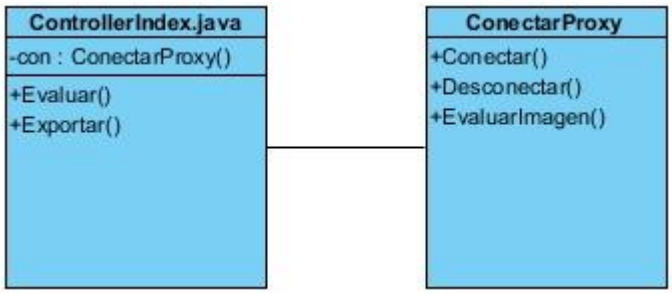

Fig. 6: Patrón creador.

- **Experto**

El GRASP de experto en información es el principio básico de asignación de responsabilidades en diseño orientado a objetos. Indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo. El cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos (Veloso Hernández, 2003)

Aplicación: Este patrón se puede evidenciar la clase del sistema en la Fig. 7, la clase ControllerIndex.java tiene toda la información necesaria para evaluar el código y exportar el proyecto a la extensión .m de MATLAB.

Fig. 7: Patrón experto.

- **Controlador**

El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado, sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control y es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación (Veloso Hernández, 2003).

Aplicación: Este patrón se ve evidenciado en la Fig. 5 en la clase SP_Dispatcher ya que esta es la clase que tiene la responsabilidad de manejar el flujo de eventos del sistema.

Patrones GOF

- **Fachada**

Ayuda a crear o provee de una interfaz unificada para manejar un conjunto de objetos en un subsistema. De esta forma, se estará definiendo una interfaz de alto nivel, que hará los subsistemas más fáciles de manejar.

En la clase SP_Código se ve evidenciado dicho patrón ya que es la encargada de crear la interfaz con la que va a interactuar el usuario.

Conclusiones parciales

- Una vez concluido el proceso de análisis y diseño se identificaron, siete Requisitos Funcionales y tres Requisitos no Funcionales permitiendo conocer las funcionalidades y características que debe brindar la aplicación.
- Se identificaron los casos de usos del sistema, realizando la descripción de cada uno de ellos y se estableció qué requisitos funcionales se cubren en cada caso de uso, permitiendo solucionar problemas existentes.
- Se realizó el diagrama de clases del diseño, en el cual se identificaron los patrones de diseño y arquitectónico, donde el uso de estos patrones posibilitó una correcta asignación de responsabilidades a cada una de las clases.

Capítulo 3: Implementación y prueba del Entorno de Desarrollo

En este capítulo se hace referencia a todos los elementos necesarios utilizados durante la implementación del IDE. Se realizaron el diagrama de componentes del sistema y el diagrama de despliegue, mostrando la relación que existe entre los diferentes objetos que componen los diagramas. Fueron realizadas las pruebas al sistema para comprobar su funcionamiento.

3.1 Modelo de despliegue

El modelo de despliegue se crea durante las últimas actividades del flujo de trabajo del diseño y suministra la descripción de la distribución física del sistema. Se utiliza como entrada fundamental en las actividades de diseño e implementación. Además son los complementos de los diagramas de componentes que unidos, proveen la vista de implementación del sistema. Describen la topología del sistema, la estructura de los elementos de *software* y *hardware* que ejecuta cada uno de ellos.

En la fig. 8 se muestra una representación del modelo de despliegue del sistema.

Fig. 8: Diagrama de despliegue.

A continuación se muestra la descripción de los componentes del modelo.

- **PC_Cliente**: representa una computadora (PC) que mediante un navegador web, realiza las peticiones al servidor de aplicaciones a través del protocolo HTTPS por el puerto 443 al servidor de aplicaciones.
- **Servidor de aplicaciones**: representa un servidor destinado a responder las peticiones hechas por la PC_Cliente**.** Cuenta con un servidor Apache Tomcat 7.0.42, el cual cuenta con Liferay 6.2 para desplegar el Portlet. Cuenta con un servicio web como mecanismo de comunicación distribuida para que las aplicaciones compartan información. Debe estar instalado el MATLAB en su versión 7.6.0.324 (R2008a) donde actúa como un subsistema para realizar las prácticas.

3.2 Diagrama de componentes

Un diagrama de componentes se representa como un grafo de componentes de software unido por medio de relaciones de dependencia. Entre los componentes se encuentran las clases, interfaces, librerías y componentes ejecutables, muestra un conjunto de ficheros relacionados entre sí para lograr una completa funcionalidad del sistema. En la Fig. 9 se hace una representación del modelo de componentes del CU ¨Ejecutar código fuente¨.

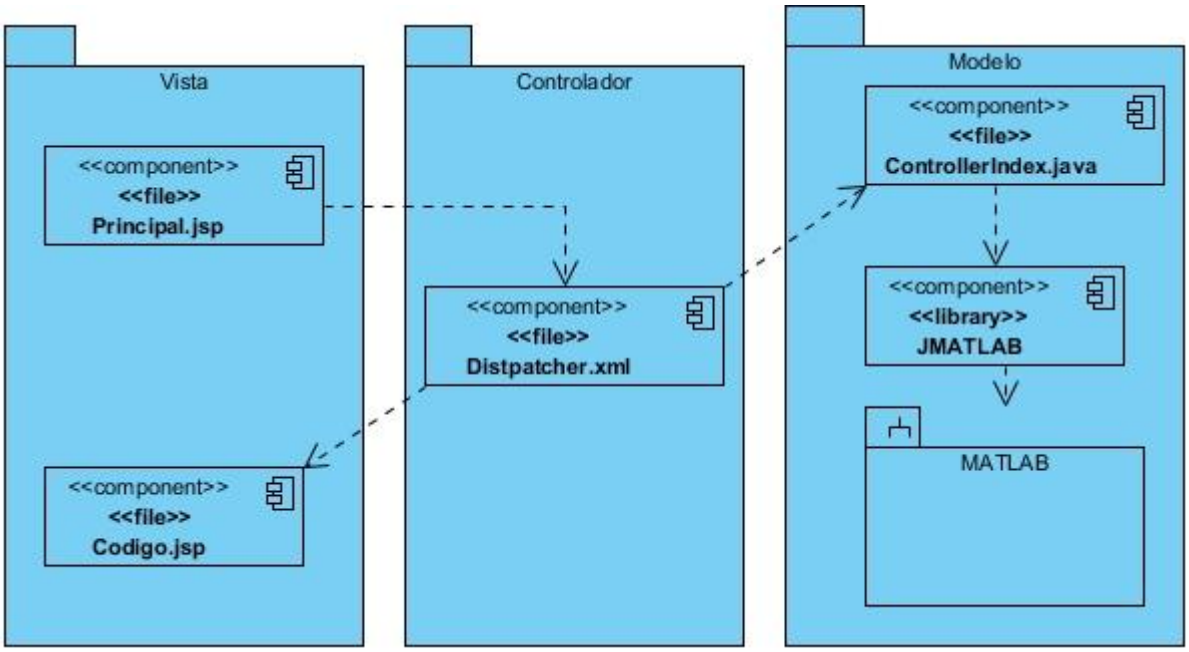

Fig. 9: Diagrama de componentes del CU ¨Ejecutar código fuente¨.

A continuación se hace una descripción a partir de la Tabla. 6 de los componentes del diagrama antes mencionado.

Tabla. 6: Descripción de los componentes del diagrama del CU "Ejecutar código fuente"´.

Principal.JSP	Página JSP que permite al usuario ingresar el código de programación para su posterior ejecución.
Dispatcher.XML	Archivo XML encargado de atender todas las peticiones del usuario y delegar estas al controlador indicado.

Código.JSP	Página JSP encargada de visualizar el resultado de la ejecución del código entrado por el usuario.
ControllerIndex.JAVA	Clase JAVA que se encarga de gestionar peticiones y respuestas de la vista.
JMATLAB.JAVA	Clase JAVA que permite la comunicación entre las aplicaciones.
MATLAB	Componente que contiene todos los datos que son solicitados por el usuario.

3.3 Pruebas de software

Las pruebas intentan demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo. Al probar el software, se ejecuta un programa con datos artificiales. Hay que verificar los resultados de la prueba que se opera para buscar errores, anomalías o información de atributos no funcionales del programa (Sommerville, 2011).

Ventajas en la inspección del software sobre las pruebas: (Sommerville, 2011)

- Durante las pruebas, los errores pueden ocultar otras fallas. Cuando un error lleva a salidas inesperadas, nunca se podrá asegurar si las anomalías de salida posteriores se deben a un nuevo error o son efectos colaterales del error original. Puesto que la inspección es un proceso estático, no hay que preocuparse por las interacciones entre errores. En consecuencia, una sola sesión de inspección descubriría muchos errores en un sistema.
- Las versiones incompletas de un sistema se pueden inspeccionar sin costos adicionales. Si un programa está incompleto, entonces es necesario desarrollar equipos de prueba especializados para poner a prueba las partes disponibles. Evidentemente, esto genera costos para el desarrollo del sistema.
- Además de buscar defectos de programa, una inspección puede considerar también atributos más amplios de calidad de un programa, como el cumplimiento con estándares. Pueden buscarse ineficiencias, algoritmos inadecuados y estilos de programación imitados que hagan al sistema difícil de mantener y actualizar.

3.3.1 Niveles de pruebas utilizados

Pruebas de Unidad

Las pruebas de unidad es la primera fase de las pruebas dinámicas y se realizan sobre cada módulo del software de manera independiente. El objetivo es comprobar que el módulo, entendido como una unidad funcional de un programa independiente, está correctamente codificado. En estas pruebas cada módulo será probado por separado y lo hará, generalmente, la persona que lo creó.

Se prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada hacia y desde la unidad de programa que está siendo probada. Se examinan las estructuras de datos locales para asegurar que los datos que se mantienen temporalmente, conservan su integridad durante todos los pasos de ejecución del algoritmo. Se prueban las condiciones límites para asegurar que el módulo funciona correctamente en los límites establecidos como restricciones de procesamiento. Se ejercitan todos los caminos independientes (caminos básicos) de la estructura de control con el fin de asegurar que todas las sentencias del módulo se ejecutan por lo menos una vez, finalmente se prueban todos los caminos de manejo de errores (Pressman, 2001).

Tanto las pruebas de caja blanca como de caja negra han de aplicar para probar de la manera más completa posible un módulo. Las pruebas de caja negra (los casos de prueba) se pueden especificar antes de que módulo sea programado, no así las pruebas de caja blanca (Juristo, Moreno y Vegas, 2005)

Pruebas del Sistema

La prueba del sistema está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas (Pressman, 2001).

Pruebas de integración

Para que los módulos de un programa funcionen bien por separado es necesario probarlos conjuntamente: un módulo puede tener un efecto adverso o inadvertido sobre otro módulo; las subfunciones, cuando se combinan, pueden no producir la función principal deseada; la imprecisión aceptada individualmente puede crecer hasta niveles inaceptables al combinar los módulos; los datos pueden perderse o malinterpretarse entre interfaces. El objetivo de las pruebas de integración es que se

necesita probar el software ensamblando todos los módulos probados previamente (Juristo, Moreno y Vegas, 2005).

Estas pruebas fueron realizadas al IDE implementado, para verificar el correcto funcionamiento de integración con la plataforma.

3.3.2 Tipos de pruebas

Pruebas de funcionalidad: son aquellas pruebas que tienen por objetivo probar que el sistema cumpla cumplan con las funciones específicas para los cuales ha sido creado.

3.3.3 Métodos utilizados para realizar las pruebas

Método de caja negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software o sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de caja blanca. La prueba de caja negra intentan encontrar errores de las siguientes categorías: funciones incorrectas o ausentes (Pressman, 2001)

1. Errores de interfaz.
2. Errores en estructuras de datos o en accesos a bases de datos externas.
3. Errores de rendimiento.
4. Errores de inicialización y de terminación.

Técnica partición de equivalencia (Pressman, 2001)

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase que, de otro modo, requieran la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Una clase de equivalencia representa un conjunto de estados validos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

Método de caja blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba.

Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que: (Pressman, 2001)

1. Garantice que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
2. Se ejercite todas las decisiones lógicas en sus vertientes verdaderas y falsas.
3. Se ejecute todos los bucles en sus límites y con sus límites operacionales.
4. Se ejercite las estructuras internas de datos para asegurar su validez.

El camino básico es una técnica de la prueba de caja blanca. Permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa (Pressman, 2001).

Antes de considerar el método del camino básico se debe introducir una sencilla notación para la representación del flujo de control, denominada grafo de flujo. El uso de un grafo de flujo, se considera como la representación del diseño procedimental, en ella se usa un diagrama de flujo para representar la estructura de control del programa (Pressman, 2001).

En el diagrama de flujo cada círculo, denominado nodo del grafo de flujo, representa una o más sentencias procedimentales. Un solo nodo puede corresponder a una secuencia de cuadros de proceso y a un rombo de decisión. Las flechas del grafo de flujo, denominadas aristas o enlaces, representan flujo de control y son análogas a las flechas del diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental (Pressman, 2001).

Con el cálculo de la complejidad ciclomática se sabe que cantidad de caminos se deben de buscar, está basada en la teoría de grafos.

En la Fig. 10 se realizó la prueba de caja blanca desarrollada al método “saveData”, donde el mismo permite establecer o no la conexión con MATLAB y se obtuvieron los siguientes resultados:

```
public void saveData(ResourceRequest resourceRequest,ResourceResponse resourceResponse,Model model)
        Gson gson= new Gson();
        codigo cod = gson.fromJson(resourceRequest.getParameter("codigo"), codigo.class);
        System.out.println(cod);                                                          1

        ConectarProxy con= new ConectarProxy();

        if(con.conectados()==true){                                                       2

                logger.info("estamos conectados");
                                                                                          3
        }
        else{

                logger.info("no se conecta");                                             4

        }

                Gson gson= new Gson();
                codigo cod = gson.fromJson(resourceRequest.getParameter("codigo"), codigo.class);

                resourceResponse.getWriter().write("true");

               System.out.println(e.getMessage());                                        5
               resourceResponse.getWriter().write(gson.toJson("false"));
        }
```

Fig. 10: Código del método de caja blanca.

A partir del código se obtuvo el siguiente grafo:

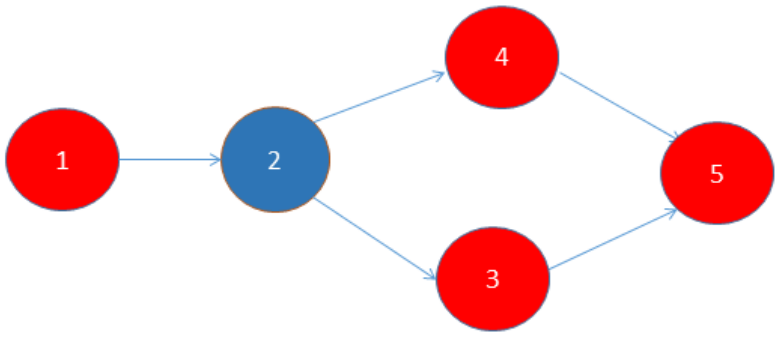

Fig. 11: Grafo de la prueba de caja blanca.

Se calculó la complejidad ciclomática del grafo para saber la cantidad de caminos independientes resultante donde se obtuvieron los siguientes resultados:

1. V(G)=2
2. V(G)=5-5+2=2

3. V(G)=1+1=2

Caminos independientes

Camino 1: 1,2,3,5

Camino 2: 1,2,4,5

Casos de pruebas de los caminos básicos

Camino 1: valor (conectado)=true

Resultados esperados: se conecta

Camino 2: valor (conectado)=false

Resultados esperados: no se conecta

3.4 Casos de pruebas

Los casos de prueba se realizan para validar el correcto funcionamiento de la aplicación y hacer la correcta liberación del producto. Durante esta fase de las pruebas de software, se comprueba la implementación de cada una de las funcionalidades mediante el uso de casos de pruebas, utilizando la técnica seleccionada con anterioridad.

A continuación se realiza el caso de prueba de caja negra para el CU “Ejecutar código fuente”, el cual consiste en comprobar cómo se comporta el sistema a medida que se van cambiando los datos utilizados.

En el caso de la prueba “Ejecutar código fuente” se utiliza como variable editor de código donde a continuación se muestra la descripción en la Tabla. 7.

Tabla. 7: Descripción de la variable del CU "Ejecutar código fuente".

No.	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Editor de código	campo de texto	No	Es un procesador de textos orientado para escribir código fuente de aplicaciones en lenguajes de programación. Soportan varios lenguajes y son capaces de resaltar su

				sintaxis.

En la siguiente Tabla. 8 se definen V que representa un valor válido, NV para datos no válidos e I que representa un valor irrelevante.

Una vez realizada la descripción de las variables se procede a la creación de la matriz de datos donde se prueba la validez de los datos introducidos en el sistema para el caso de prueba “Ejecutar código fuente”.

Tabla. 8: Matriz de datos.

Escenario	Descripción	Variable	Respuesta del sistema	Flujo Central
		1		
EC 1.1 Ejecutar introduciendo el código correctamente.	Se presiona el botón Enter ejecutando la función o la declaración de variable sin errores léxicos.	V x=30 sen(x) V roots([1 10 100])	El sistema muestra el resultado de la operación evaluada.	1- El usuario introduce el Código. 2- Presiona la tecla Enter. 3- El editor de código muestra el resultado y posiciona el cursor en la siguiente línea.
EC1.2 Ejecutar introduciendo código incorrecto.	Se presiona el botón enter ejecutando la función o la declaración de variable con errores de cualquier tipo.	NV sen(asd)	El sistema muestra un error de variable indefinida.	1- El usuario introduce el Código. 2- Presiona la tecla Enter. 3- El editor de código muestra el error y posiciona el cursor en la siguiente línea.

1.3 Ejecutar cuando no haya código en el editor de texto.	Se deja el editor vacío y el sistema no ejecuta.	I	El sistema mantiene deshabilitado el botón Ejecutar, impidiendo que se pueda ejecutar sin haber introducido el código.	1- El usuario no introduce el Código. 2- Presiona la tecla Enter. 3- El editor de código no muestra el resultado.

3.5 Resultado de las pruebas

Durante la aplicación del método de caja negra al sistema fueron detectadas un total de quince no conformidades, las mismas fueron corregidas en varias iteraciones del método. A continuación se ejemplifican cinco de las no conformidades.

Tabla. 9: Tipos de no conformidades.

Números	Caso de Prueba	Tipo de error	Descripción
1	Ejecutar código fuente	Error de interfaz	El botón ejecutar no se deshabilita cuando el editor de texto no tiene ningún código introducido.
2	Crear nuevo proyecto mediante plantillas	Funcional	Al cargar la plantilla no mostraba el código de la misma.
3	Exportar proyecto a la extensión .m de MATLAB	Funcional	Cuando se exportaba el proyecto lo hacía con el código comentariado.
4	Mostrar gráfica	Error técnico	Cuando se graficaba la función no coincidían los valores de la gráfica con los de la propia función

	de coordenadas de una función.	Ortografía	Cuando se mostraba la gráfica la palabra ¨gráfica¨ aparecía sin tilde

Como parte de la ejecución de las pruebas de caja negra se realizaron tres iteraciones de pruebas que se representan en la Tabla. 10. En la primera iteración se encontraron quince no conformidades. Una vez corregidas, se procedió a realizar una segunda iteración, en la que se identificaron cinco no conformidades nuevas y finalmente se realizó una última iteración en la que no se encontraron deficiencias, razón por la que se definió no realizar más iteraciones.

Tabla. 10: Resumen de las no conformidades encontradas.

Casos de Pruebas	Iteración 1	Iteración 2	Iteración 3
Completar código en el editor de texto.	2	1	-
Refactorizar código fuente en el editor de texto.	3	1	-
Ejecutar código fuente.	2	0	-
Crear nuevo proyecto mediante plantillas.	1	1	-
Exportar proyecto a la extensión .m de MATLAB.	3	0	-
Mostrar gráfica de coordenadas de una función.	4	1	-

Generar código fuente de una función seleccionada.	2	1	-

En la siguiente gráfica se puede apreciar los resultados obtenidos a partir de la tabla anterior.

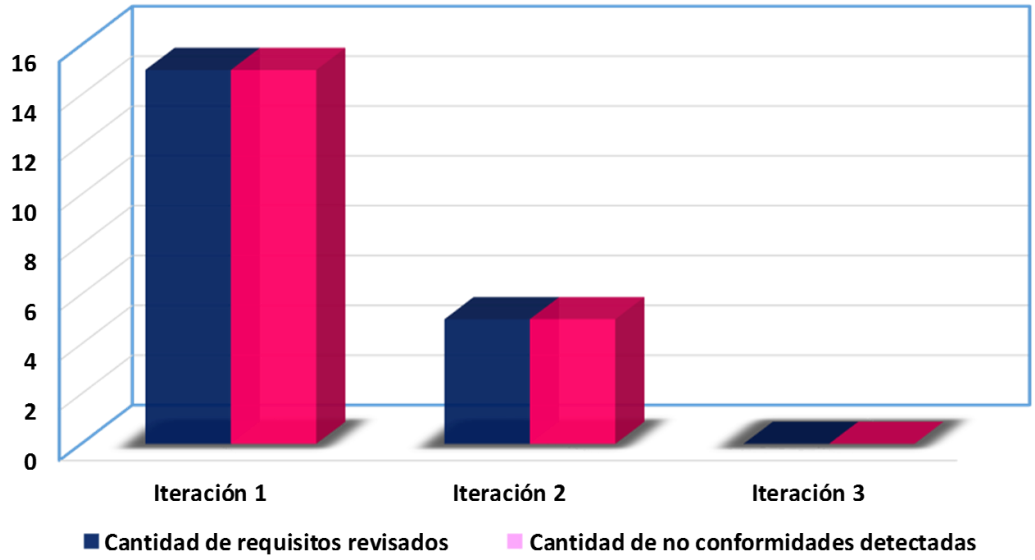

Fig. 12: Resultado de las pruebas realizadas.

Conclusiones parciales

- En el presente capítulo se realizó el diagrama de despliegue donde se representa la distribución física del sistema, lo que permitió conocer la estructura de los elementos de *software* y *hardware* que ejecuta cada uno de ellos.
- En el diagrama de componentes se representan los diferentes elementos que participan en el sistema, para lograr comprender la estructura y como se organizan los componentes que intervienen en la implementación.
- Una vez terminada la implementación del IDE fueron realizadas pruebas de integración, de sistema y unidad permitiendo corregir las no conformidades y así lograr un buen funcionamiento de la aplicación.

Conclusiones generales

Una vez concluida la investigación se arribaron a varias conclusiones, las mismas son descritas a continuación:

- Con la elaboración del marco teórico de la investigación se logra profundizar y conocer acerca de los Sistemas de Laboratorios Virtuales y a Distancia en el Control Automático así como identificar la necesidad de crear una herramienta de implementación.
- Con el análisis y diseño del Entorno Integrado de Desarrollo, se logra una mayor comprensión de la solución, definiendo los requisitos, además de generar los artefactos que propone la metodología OpenUp, lo que contribuyó en gran medida a realizar el proceso de implementación.
- Con la implementación del Entorno Integrado de Desarrollo se logra crear una interfaz para la web la cual accede al software matemático MATLAB, permitiendo que los usuarios puedan realizar las prácticas.
- Con la aplicación de las diferentes pruebas al Entorno Integrado de Desarrollo, se logran corregir todas las deficiencias y no conformidades que fueron detectadas durante el proceso de prueba, permitiendo entregar un software con calidad logrando satisfacer las necesidades del cliente.

Recomendaciones

- Se recomienda que el Caso de Uso ¨ Refactorizar código fuente en el editor de texto ¨ aparte de cumplir con la funcionalidad de buscar y remplazar código pueda organizar el código mediante teclas predefinidas.
- Se recomienda que la aplicación pueda contener la opción de hacer y deshacer, como por ejemplo la opción de ctrol + z.

Referencias bibliográficas

ABAD LONDOÑO, J.H. 2014. Lecciones Aprendidas en Desarrollo de Software: Tabla comparativa entre Metodologías Tradicionales y Ágiles. [en línea]. [Consulta: 19 junio 2015]. Disponible en: http://www.lecciones-aprendidas.info/2014/07/tabla-comparativa-entre-metodologias.html.

ALFEREZ SANCHEZ, J. 2010. Instalación,Configuración y Administración del Servidor de Aplicaciones JBoss. . S.l.: s.n.,

ALLAMARAJU, S. y BEUST, C., 2011. *Programación Java Server con J2EE*. 2011. S.l.: s.n.

ARIAS, J. y FERNÁNDEZ, N., 2010. *Servicios Web*. 2010. S.l.: s.n.

CALABRINA, L. y PÍRIZ, P. 2003. *Metodología XP*. Universidad ORT. Uruguay: Cátedra de Ingeniería de Software.

CARAVACA MORA, O.M., 2010. *Diseño de un Entorno de Desarrollo Integrado para una Unidad Controladora de Procesos*. 2010. S.l.: s.n.

CARVAJAL RIOLA, J.C., 2008. *Metodoloías ágiles. Herramientas y modelo de desarrollo para aplicaciones Java EE como metodología empresarial. Tesis Final de Máster*. 2008. S.l.: s.n.

CASTELLANOS, S., RODRÍGUEZ, A. y HERNÁNDEZ, L., 2005. *Laboratorio a distancia para la prueba y evaluación de controladores a través de Internet.* 2005. S.l.: s.n. Universidad Central «Marta Abreu» de las Villas

CERVANTES, Humberto, 2010. *Arquitectura de Software*. 2010. S.l.: s.n.

CHAD, M.D. y BROWN, D. 2007. Página de inicio de Mozilla Firefox. [en línea]. [Consulta: 17 junio 2015]. Disponible en: about:home.

CLAY RICHARDSON, W. 2004. *Professional Portal Development with Open Source Tools*. Indianapolis, Indiana: s.n. ISBN 459.

CODEMIRROR, Comunity 2015. CodeMirror. [en línea]. [Consulta: 17 junio 2015]. Disponible en: http://codemirror.net/.

CRAIG, W. y BREIDEBACH, R. 2005. Spring in action. . Packt Publishing. S.l.: s.n., ISBN 1-932394-35-4.

DEPTO. CCIA, 2005. *JSP Básico*. 2005. S.l.: s.n.

DIXON, W.. 2001. *Towards the standardizatoin of a matlab-based control systems laboratory experience for undergraduate students.* Proceedings of the American Control Conference. S.l.: s.n.

DIXON, W.. 2002. *A matlab-based control system laboratory experience for undegraduate student Toward standardization and shared resources.* IEEE Transactions on Edution. S.l.: s.n. ISBN 0018-9359.

DURÁN, A. y MEDEL, R. 2012. Introducción a Apache Tomcat 5.5. [en línea]. [Consulta: 17 junio 2015]. Disponible en: http://www.lsi.us.es/docencia/get.php?id=1923.

EGUILUZ, J. 2015. *Introduccion a HTML*. S.l.: s.n.

EGUÍLUZ PÉREZ, J. 2010. *Introducción a Java Script*. S.l.: s.n.

FERNANDES AGRELA, A.K. y MIRANDA BOUSAS, C.M., 2009. *Tesis Doctoral*. 2009. S.l.: s.n.

HERNÁNDEZ, G., SILVA ORTIGOZA, R. y CARRILLO SERRANO, R.V. 2013. *Control Automático. Teoría de diseño, construcción de prototipo, modelo, identificación y pruebas experimentales.* México: s.n.

JBOSS 2007. Plataforma de aplicaciones JBoss enterprise para portales. [en línea]. [Consulta: 17 junio 2015]. Disponible en: www.redhat.es/jboss.

JIMÉNEZ, L.M. 2005. *Recolab: Laboratorio Remoto de control utilizando Matlab/Simulink.* Dpto.Ingenieria de Sistemas Industriales, Universidad Miguel Hernández,Elche(Alicante),Spain: s.n. ISBN 1697-7912.

JIMÉNEZ MILIA, J.P. y FERRER OBREGÓN, R. 2008. Análisis de un IDE para múltiples plataformas con tecnologías y herramientas libres para desarrollar software educativo en formato multimedia. Subsistema de gestión de código. . La Habana: s.n.,

JONAS, J. 2008. *Liferay Portal Enterprise Intranets*. Birmingham,. Birmingham, E.E. U.U: s.n. ISBN 978-1-84719-272-1.

JURISTO, N., MORENO, A.M. y VEGAS, S., 2005. *Técnicas de evaluación de software*. 2005. S.l.: s.n.

KING, R. 2010. The Top 10 Programming Languages. [en línea]. [Consulta: 17 junio 2015]. Disponible en: http://spectrum.ieee.org/at-work/tech-careers/the-top-10.

LARMAN, C. 1999. *UML y Patrones. Introducción al análisis y diseño.* México: s.n. ISBN 970-17-0261-1.

LÓPEZ DUQUE, M. y OCEGUERA RAVELO, R., 2010. *Herramienta en Matlab para la obtención de datos y ficheros electroencefalograma del proyecto Mapeo Cerebral Humano Cubano*. 2010. S.l.: s.n.

LORANDI MEDINA, A.P., GARCÍA REYNOSO, A.C., HERMIDA SABA, G., LADRÓN DE GUEVARA DURÁN, E. y HERNÁNDEZ SILVA, J., 2011. *Canihuá: Un portal de Laboratorios Virtuales*. 2011. S.l.: s.n.

LORANDI MEDINA, A.P., SABA HERMIDA, G., HERNÁNDEZ SILVA, J. y LADRÓN DE GUEVARA DURÁN, E., 2011. *Los Laboratorios Virtuales y Laboratorios Remotos en la Enseñanza de la Ingeniería*. 2011. S.l.: Revista internacional de educación en ingeniería.

MANCHADO SERRA, D. y FRANCO PUNTES, D., 2010. *Estudio del servidor de aplicaciones Glassfish y de las aplicaciones J2EE.* 2010. S.l.: Escola d'Enginyeria.

MANDONADO, D. 2012. El codigo k. [en línea]. [Consulta: 17 junio 2015]. Disponible en: http://www.elcodigok.com.ar/2007/09/que-son-los-ide-de-programacin.html.

MARTINEZ AJETE, M. y FRANCIA OFARRIL, D., 2014. *Desarrollo del Portal Web de la Organización Nacional de Bufetes Colectivos.* 2014. S.l.: s.n.

MATEU, C. 2010. *Desarrollo de Aplicaciones Web*. Barcelona: s.n. ISBN 84-9788-118-4.

MENDOZA, J. 2013. Diseño del sistema de tarjeta de crédito con UML. [en línea]. [Consulta: 17 junio 2015]. Disponible en: http://sisbib.unmsm.edu.pe/bibvirtualdata/Tesis/Basic/mendoza_nj/Cap5.pdf.

MONFERRER AGUT, R. 2001. Especificación de Requisitos Software según el estándar de IEEE 830. . Universitat Jaume I: s.n.,

MOREÑO, J.J. 2011. Curso Java y Tecnologías Java EE. [en línea]. [Consulta: 17 junio 2015]. Disponible en: Curso Java y Tecnologías Java EE.

ORM PROPEL 2011. ORM Propel. [en línea]. [Consulta: 17 junio 2015]. Disponible en: http://www.ultimocombate.com/python-vs-php_id512.html.

PRESSMAN, R.S. 2001. *Ingeniería de software. Un enfoque práctico*. S.l.: McGraw-Hill Companies. ISBN 84-481-3214-9.

RODRÍGUEZ, A. y CASTELLANOS, S., 2004. *Remote laboratory system(RLC):Learning laboratory in the automatic control at distance.* 2004. S.l.: s.n. Porto Alegre

RODRÍGUEZ OJEDA, L., 2007. *MATLAB conceptos básicos y programación*. 2007. S.l.: s.n.

ROSADO, L. y HERREROS, J.. 2009. Nuevas aportaciones didácticas de los laboratorios virtuales y remotos en la enseñanza de la Física. . Madrid : Recent Research Developments in Learning Technologies: s.n.,

SÁNCHEZ, J., DORMIDO, S. y MORILLA, F., 2009. *Laboratorios virtuales y remotos para la práctica a distancia de la Automática*. 2009. S.l.: s.n.

SANTANA CHING, I., 2012. *Herramienta para la docencia en la automática orientada hacia la metodología de la ECTS.Tesis Doctoral*. 2012. S.l.: s.n.

SCAMERCIO, F. 2010. Introducion a Liferay Portal. [en línea]. [Consulta: 17 junio 2015]. Disponible en: http://francescoscamarcio.com/2010/12/10/introducion-a-liferay-portal.

SIERRA, D. 2013. Visual Paradigm For Uml. [en línea]. [Consulta: 17 junio 2015]. Disponible en: http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

SOMMERVILLE, I. 2011. *Ingeniería de Software*. Novena. S.l.: s.n. ISBN 978-607-32-0603-7.

TRIGAS GALLEGO, M., 2009. *Gestión de proyectos informáticos*. 2009. S.l.: s.n.

TROTT, J. y SHALLOWAY, A., 2000. *Design Patterns Explained. MODELER.* 2000. S.l.: s.n.

URAN, S., HERCOG, D. y JEZERNIK, K. 2010. *Web-based MATLAB and Controller design learning.* S.l.: University of Maribor, Maribor, Slovenia.

VARY, J.P. 1999. Informe de la reunión de expertos. . S.l.: s.n.,

VELOSO HERNÁNDEZ, P., 2003. *Uso de patrones de arquitectura*. 2003. S.l.: s.n.

VERA, A., ZÚÑIGA, N. y BERNAL, Á. 2013. *Herramienta en línea para la programación y depuración .* Herramienta en línea para la programación y depuración . Grupo GADyM,: s.n.

VISCONTI, M. y ASTUDILLO, H. 2011. Fundamentos de Ingeniería de Software. [en línea]. [Consulta: 17 junio 2015]. Disponible en: http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

YUAN, M. 2012. Seam frameword.Esperience the evolution of Java EE. . Prentice-Hell. S.l.: s.n.,

Anexos

Fig. 13: CU "Completar código en el editor de texto."

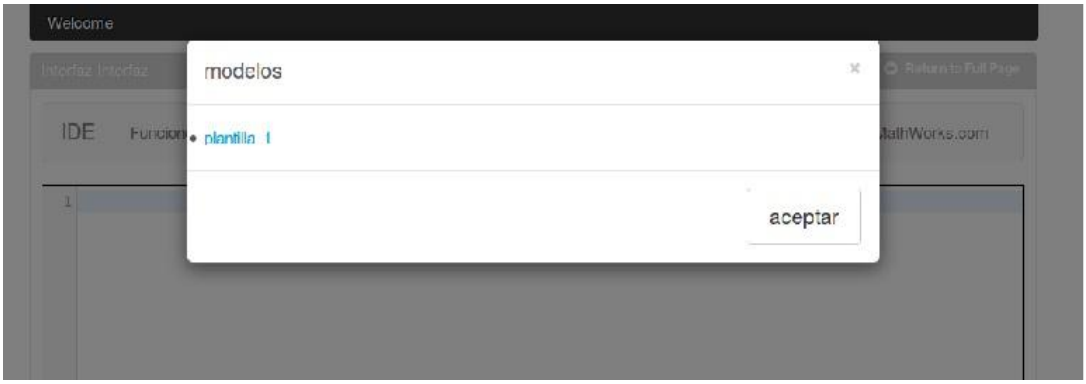

Fig. 14: CU "Crear nuevo proyecto mediante una plantillas".

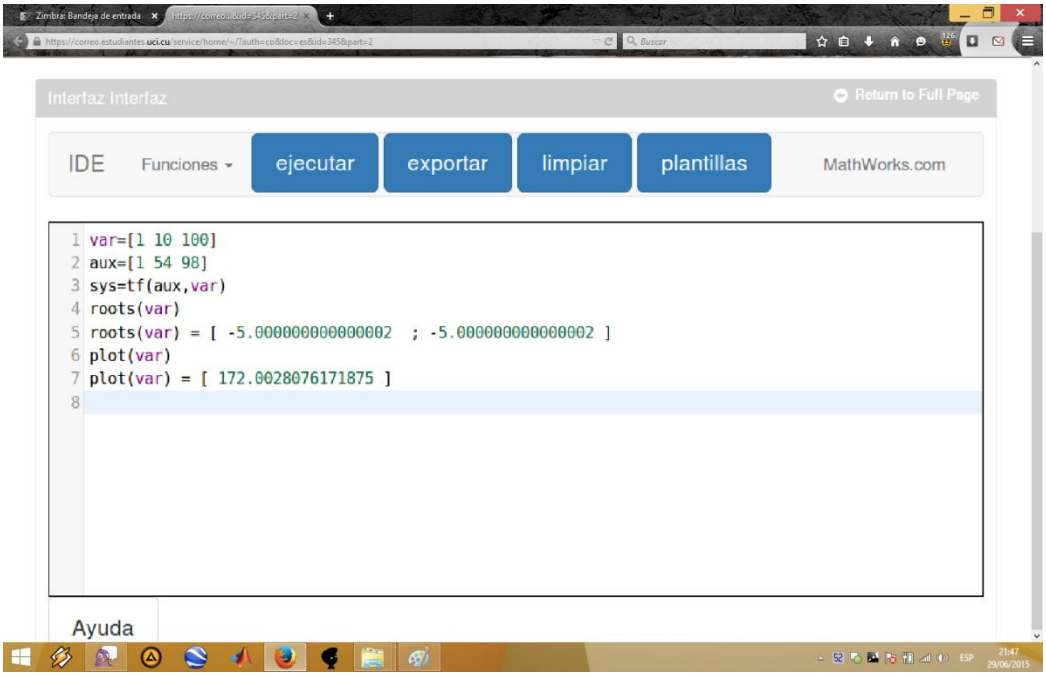

Fig. 15: Ejemplo de prácticas de Control Automático.

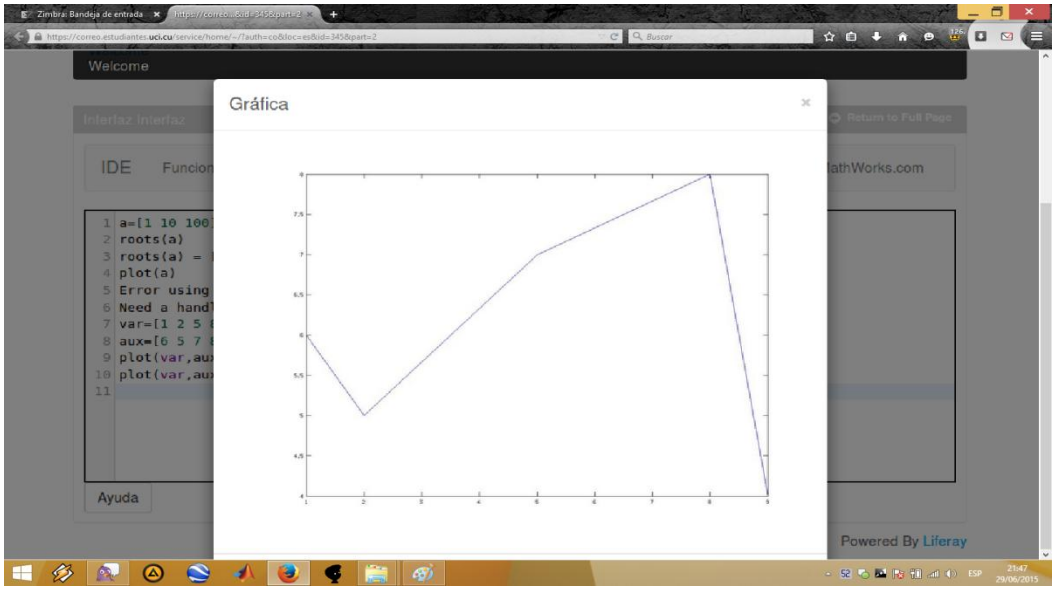

Fig. 16: CU "Mostrar gráfica de una función seleccionada".

Fig. 17: CU "Ejecutar código fuente"

Tabla. 11: Descripción del CU ¨Completar código en el editor de texto¨.

Caso de Uso	*Completar código en el editor de texto.*	
Actores	*Usuario*	
Resumen	*El caso de uso inicia cuando el usuario presiona las teclas ctrol+ espacio y el sistema muestra una lista de palabras reservadas del lenguaje.*	
Complejidad	*media*	
Prioridad		
Precondiciones		
Postcondiciones		
Referencias	*RF1*	
Flujo de eventos		
Flujo básico: *Completar código en el editor de texto*		
	Actor	**Sistema**
1.	*Presiona ctrol+ espacio.*	

2.		Muestra *una lista de palabras reservadas del lenguaje según la necesidad del usuario.*

Prototipo de Interfaz

Relaciones	**CU Incluidos**	*No procede*
	CU Extendidos	*No procede*
Requisitos no funcionales		

1