ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

DESARROLLO DE UN PROTOTIPO DE PANEL DE CONTROL DE SERVICIO DE HOSTING JAVA PARA LA EMPRESA KAPYASOFT BASADO EN TEST-DRIVEN DEVELOPMENT

Trabajo de Titulación presentado para optar al grado académico de:

INGENIERO EN SISTEMAS INFORMÁTICOS

AUTOR: EDGAR FABIÁN CÓRDOVA JIMBO

TUTOR: ING. JORGE ARIEL MENÉNDEZ VERDECIA

Riobamba – Ecuador

2017

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

El Tribunal de Trabajo de Titulación certifica que: El trabajo de investigación: **DESARROLLO DE UN PROTOTIPO DE PANEL DE CONTROL DE SERVICIO DE HOSTING JAVA PARA LA EMPRESA KAPYASOFT BASADO EN TEST-DRIVEN DEVELOPMENT**, de responsabilidad del señor Edgar Córdova, ha sido minuciosamente revisado por los Miembros del Tribunal de Trabajo de Titulación, quedando autorizada su presentación.

	FIRMA	**FECHA**
Ing. Washington Luna **DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	_____	_____
Ing. Patricio Moreno **DIRECTOR DE ESCUELA DE INGENIERÍA EN SISTEMAS**	_____	_____
Ing. Jorge Menéndez **DIRECTOR DE TRABAJO DE TITULACIÓN**	_____	_____
Ing. Lorena Aguirre **MIEMBRO DEL TRIBUNAL**	_____	_____

Yo, Edgar Córdova soy responsable de las ideas, doctrinas y resultados expuestos en esta Tesis y el patrimonio intelectual de la Tesis de Grado pertenece a la Escuela Superior Politécnica de Chimborazo

EDGAR CÓRDOVA

DEDICATORIA

A mis Padres Mercy y Fausto, el origen de mi motivación, impulsores de cada idea y acto de bien. A mis hermanas Diana y Sandra, por su apoyo y aliento.

Con afecto

Edgar.

AGRADECIMIENTO

A la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO, por todo el saber adquirido, a mis profesores por sus lecciones aplicables en el campo profesional y a mi tutor, por su guía para desarrollar el presente Trabajo de Titulación.

Edgar.

TABLA DE CONTENIDOS

CAPÍTULO II

2. MARCO METODOLÓGICO

INDICE DE TABLAS

INDICE DE FIGURAS

INDICE DE GRAFICOS

INDICE DE ANEXOS

RESUMEN

El principal objetivo fue desarrollar un prototipo de panel de control de servicio de hospedaje web para aplicaciones desarrolladas en JAVA, para lo cual fue importante definir todos los procesos necesarios para realizarlo manualmente y poder automatizar dicho proceso. Se escogió una disciplina de programación poco usual llamada Test-Driven Development que se fundamenta en la escritura de pruebas de software al inicio del proyecto para construir la aplicación. Para lograr este propósito fue indispensable definir los principios básicos sobre pruebas, en especial pruebas de integración y pruebas unitarias y utilizar la herramienta de integración continua, JENKINS. Las pruebas son una etapa fundamental en el proceso de desarrollo de software y aunque la idea de realizar pruebas antes de tener el código fuente de un proceso, programa o aplicación pueda parecer ilógico, según el autor de la técnica Kent Beck, afirma que está más que comprobado que el desarrollo guiado por pruebas, garantiza principalmente la funcionalidad y calidad del proyecto. El prototipo desarrollado con el framework J.S.F., denominado PROTPANEL, entre sus principales funcionalidades, permite el registro de usuarios, la autenticación de los mismos, la gestión de aplicaciones y bases de datos, gestión de dominios, compilación del código fuente y despliegue de las aplicaciones web. La aplicación fue probada por el cliente del proyecto, propietario de la empresa KAPYASOFT y también en un experimento aplicado a dos grupos de estudiantes de la Escuela Superior Politécnica de Chimborazo siendo el prototipo, 3 veces más rápido a una guía de despliegue de aplicaciones web en la nube. En conclusión, el software creado mejora el tiempo de despliegue de aplicaciones web en JAVA y se recomienda el uso de herramientas informáticas para realizar los planes de pruebas, casos de pruebas, así como también para realizar la integración continua.

PALABRAS CLAVE: <TECNOLOGÍA Y CIENCIAS DE LA INGENIERÍA>, <SISTEMAS INFORMÁTICOS>, <HOSPEDAJE WEB>, <APLICACIÓN>, <JAVA>, <SERVIDOR DE APLICACIONES>, <DESARROLLO GUIADO POR PRUEBAS (TDD)>, <PRUEBAS>

SUMMARY

The principal objective was develop a Web Hosting service prototype control panel for JAVA developed applications, which it was important define all the necessary processes to make into manual and automatic. It choses an unusual discipline called TEST-DRIVEN DEVELOPMENT which is based on writing software test at the beginning of the project to build the application. To achieve this purpose it was indispensable to define the basics principles about the Tests, especially integration Tests and unit Tests and the use of a continuing integration tool called JENKINS. Testing it was fundamental stage in the software development software, although the idea of performing Tests before having the source code of process, program or application seems illogical, according the Kent Beck technique affirms that it is more than proven that guided development by test guarantees mainly the functionality project quality. The developed prototype with frameworks J.S.F. called PROTPANEL, among its main functionalities, permits user registration and authentication, management of applications and databases, Domain management, compilation of the source code and application deployment. The application was tested by the client of the project, owner of the KAPYASOFT enterprise and it was also applied as an experiment to two groups of ESPOCH students; the prototype being three times faster than a web application deployment guide on the cloud. In conclusion the created software improve the JAVA Web Application deployment time and it recommends the use of informatics tools for performing the Tests plans, test cases, as well as for performing the continuing integration.

Keyword: <WEB HOSTING>, <WEB APPLICATION>, <JAVA>, <POSTGRESQL>, <GLASSFISH>, <APPLICATION SERVER>, <TDD>, <DEVELOPMENT>, <TESTING>, <CONTINUING INTEGRATION>

INTRODUCCION

Antecedentes

KAPYASOFT es una empresa dedicada a proporcionar tecnologías innovadoras, desarrollo de aplicaciones informáticas, capacitaciones, consultoría y mantenimiento de los productos desarrollados, enfocándose las necesidades específicas de cada empresa o persona con el fin de contribuir a su desarrollo y competitividad y desea extender sus servicios a la venta de Cloud hosting JAVA.

La empresa carece de servidores físicos propios y aunque en el país sí existen empresas de Hosting JAVA, los costos son elevados, por lo que sub-contratar servicios de hospedaje web JAVA de terceros sería poco rentable.

Todos los procesos del levantamiento de los servidores hasta la publicación de sitios web se realizan por así decirlo de forma manual, dicho proceso incluye de manera general: instalación del Kit de Desarrollo JAVA, instalación del motor de bases de datos, configuración de las cuentas F.T.P., configuración de servicio de correo, instalación y configuración del servidor de aplicaciones, configuración de los directorios virtuales, subida y despliegue de las aplicaciones web y configuración de nombres de dominio (D.N.S.).

Del mismo modo una vez desplegada la aplicación es posible que se requieran hacer correcciones y con el fin de consumir el mínimo de recursos no se instala interfaces gráficas en servicios IaaS por lo que no es posible instalar entornos de desarrollo integrado (I.D.E.) y por ello es necesario construir de nuevo el archivo W.A.R. de la aplicación web, para poder utilizar los paneles de control que ya existen como los proporcionados por Glassfish y Tomcat, entre otros, por lo que el consumo por transferencia sube innecesariamente si hubiera una manera de corregir solo los archivos específicos a través de un editor web.

La mayoría de desarrolladores tardan mucho tiempo en la instalación y administración de servidores de aplicaciones bajo el Sistema Operativo Linux, lo cual no se encuentra automatizado (se hace a través de consolas Secure Shell (S.S.H.). con comandos de texto), lo que provoca un mayor tiempo de para desplegar, probar y publicar las aplicaciones desarrolladas.

Cuando se hace referencia a las aplicaciones web que se desplegarán se aclara que se tratan de sitios del tamaño de portales web y sus respectivos gestores de contenido y por lo menos una

base de datos con por lo menos 10 tablas o aplicaciones JAVA clasificadas como difíciles entre 10000 y 20000 líneas de código fuente.

El cliente supo manifestar que se requiere de mucho tiempo para realizar el proceso descrito anteriormente, que es repetitivo cada vez que se pone un proyecto en producción, por lo que se propone el presente trabajo de titulación "DESARROLLO DE UN PROTOTIPO DE PANEL DE CONTROL DE SERVICIO DE HOSTING JAVA PARA LA EMPRESA KAPYASOFT BASADO EN TEST-DRIVEN DEVELOPMENT".

Formulación del problema

¿Cómo afecta al tiempo que demora desplegar aplicaciones web creadas en JAVA el desarrollo de un prototipo de panel de control aplicado a la empresa KAPYASOFT utilizando Test-Driven Development?

Objetivos

Objetivos Generales

Desarrollar un prototipo de panel de control para la empresa KAPYASOFT utilizando Test-Driven Development para afectar el tiempo que demora desplegar una aplicación JAVA .

Objetivos Específicos

- Describir los procesos de despliegue de aplicaciones, el número de aplicaciones y medir el tiempo que se demora un desarrollador en desplegar una aplicación JAVA.
- Definir los fundamentos teóricos de Test-Driven Development en lo referente a pruebas unitarias que se aplicarán al desarrollo del prototipo de panel de control.
- Desarrollar un prototipo de panel de control para despliegue de aplicaciones JAVA usando Test-Driven Development.
- Comprobar si el tiempo de despliegue de aplicaciones JAVA es afectado con el desarrollo de un prototipo de panel de control.

Justificación

Justificación Teórica

Según la Comisión de Estudios de Auditoria, Consejo Profesional de Ciencias Económicas de la Capital Federal, Argentina, un panel o tablero de control constituye una herramienta básica para la gestión, ya que provee de la información necesaria para tomar un rápido conocimiento del estado de situación actual y la probable evolución de la empresa sujeta al análisis. De esta manera facilita la profundización del análisis en los casos que lo considere necesario.

La Real Academia de las Ciencias Físicas y Exactas define la automática como el conjunto de métodos y procedimientos para la substitución del operario en tareas físicas y mentales previamente programadas. De esta definición original se desprende la definición de la Automatización como la aplicación de la automática al control de procesos industriales. Aunque esta definición esté más orientada a la industria y fabricación de productos tangibles bien se puede aplicar a la generación de productos intangibles si se ve al desarrollo de software también como una industria.

Según el creador de las metodologías Test-Driven Development y XP, Kent Beck, vale la pena dedicar tiempo al estudio de la metodología T.D.D. y convertirla en la herramienta de diseño principal, ya que entre algunas de las razones destaca la calidad, la cual aumenta considerablemente, evitando código muerto, la detección de errores a tiempo y por la importancia que esta le da a las garantías de calidad.

Kent Beck también afirma que en el caso de equipos multi-disciplinarios mejora la integración y la comunicación, además que escribir el ejemplo o prueba antes que el código obliga al desarrollador a escribir el mínimo de funcionalidad necesaria, lo cual significa que el cliente tendrá lo que necesita y nada más.

Esta última característica podría ser apropiada para la elaboración de prototipos funcionales y listos para ser utilizados dentro de una empresa de software.

Justificación Aplicativa

El desarrollo de este prototipo de panel de control contribuiría a resolver la incógnita de que si automatizando todo el proceso de despliegue de aplicaciones web JAVA, ayudaría para reducir el tiempo de despliegue de dichas aplicaciones.

De todas las características y normas que debería seguir cualquier software que maneje datos de una cantidad masiva de usuarios, el prototipo propuesto se enfoca al proceso de despliegue de las aplicaciones mas no contempla situaciones de la post publicación del software como, la integridad de los datos, cyber ataques, validación de las políticas de cifrado, filtros de contenido inapropiado, indemnizaciones, gestión de la identidad, entre otras normas especificadas por el sitio web de la Gobernanza de Internet en Ecuador.

La práctica Test-Driven Development se aplicará al desarrollo del prototipo mas no tiene nada que ver con el proceso de desarrollo y despliegue de aplicaciones que se hospedarán. Del mismo modo solo se propone un prototipo puesto que un producto completo de panel de control requiere de más controles para satisfacer como las normas que se mencionan en el párrafo anterior y se explicarán mejor en el apartado 4.5 del marco teórico.

El prototipo debería permitir que otras aplicaciones creadas en un computador local sean desplegadas sin necesidad de configuraciones ni códigos adicionales, también permitiría enlazar los sitios desplegados con sus dominios DNS contratados a algún proveedor de nombres de dominio.

Otro aspecto importante de este prototipo es que reduciría trasferencia de archivos durante el despliegue y mantenimiento de las aplicaciones y visualizar los archivos de código fuente y los archivos ya desplegados de forma más organizada y jerárquica, todo en una misma interfaz y trabajando en el mismo servidor remoto lo que sería de mucha utilidad para realizar correcciones sin necesidad de subir al servidor todo el archivo JAVA reconstruido, lo que se traduciría en ahorro de costos por transferencia.

CAPITULO I

1. MARCO TEÓRICO REFERENCIAL

1.1. Definición de Test-Driven Development

Test-Driven Development (desarrollo dirigido por pruebas), o en modo abreviado, T.D.D., es un proceso de desarrollo de software que se basa en la idea de desarrollar primero pruebas y luego codificar y refactorizar el código construido. "Test-Driven Development es una práctica de programación que consiste en escribir primero las pruebas (generalmente unitarias), después escribir el código fuente que pase la prueba satisfactoriamente y por último, refactorizar el código escrito". (Herranz José, 2011, s.p.).

> Numerosos estudios se han llevado a cabo para comparar el desarrollo guiado por pruebas con técnicas de desarrollo más tradicionales como las inspecciones de código o pruebas después de que se ha escrito el código. Uno de estos estudios mostró que la prueba condujo a los desarrolladores a producir una mayor calidad del código, que aumentó un 18% más casos de prueba de caja negra funcional y el 80% de los desarrolladores que participaron en la encuesta afirmaron que Test-Driven Development fue más eficaz (George Boby., 2003, s.p.).

> Test-Driven Development funciona porque obliga al desarrollador a tener en cuenta el diseño a través de las pruebas (el elemento del sistema antes de la codificación) y a dar pequeños pasos al escribir el software. También destaca soluciones simples a problemas y no sólo ha demostrado que el desarrollo guiado por pruebas mejora la calidad del código, sino que las pruebas se pueden utilizar también como documentación del sistema. (Ambler Scott., 2006, s.p.).

Las pruebas son automatizadas, por ello es fácil volver a probar todo el agregando cada vez más funcionalidades. Pero hay que considerar que ejecutar continuamente pruebas aumentará el tiempo de desarrollo.

1.2. Objetivos de la técnica Test-Driven Development

- La implementación solo de las funciones exactas que el cliente necesita.
- La minimización de defectos en etapa de producción.
- La producción de software modular y reutilizable. (Beck Kent, 2003, s.p.)

Con frecuencia se malentiende la técnica Test-Driven Development, y se cree que se enfoca en hacer pruebas, pero esto no es del todo exacto ya que T.D.D. trata más de diseño y las pruebas automatizadas son un efecto colateral.

Tener un fallo en rojo utilizando la disciplina T.D.D., es normal, puesto que si existe un fallo, este se corrige entonces se construye el código y se consigue que la barra roja se convierta en una verde y esto es lo que busca la técnica.

1.3. El ciclo Test-Driven Development

La Figura No 1-1 muestra el ciclo de Test-Driven Development, con el código de colores de estados en los que puede caer una prueba.

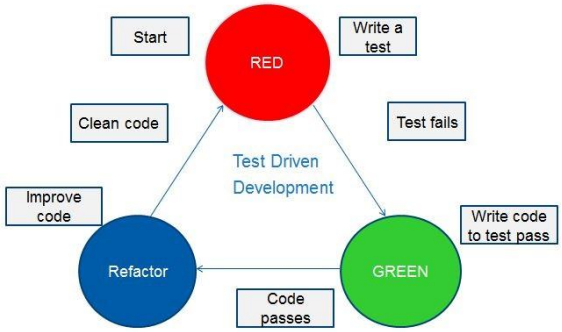


Figura No 1-1. Pasos del ciclo Test-Driven Development

Recuperado de: http://www.wideskills.com/junit/test-driven-development-junit

1.4. Ventajas de usar Test-Driven Development

Según el sitio PMOinformatica, las principales ventajas de utilizar el desarrollo guiado por pruebas, pueden ser:

- Mayor calidad,
- Diseño enfocado en necesidades y
- Mayor simplicidad en el diseño.

1.5. El algoritmo Test-Driven Development

Según el autor de esta técnica, los pasos del algoritmo para aplicar Test-Driven Development correctamente son:

1. Escribir la especificación del requisito (la prueba),
2. Ejecutar las pruebas especificadas,
3. Implementar el código o corregir errores y

4. Refactorizar para eliminar duplicidad y hacer mejoras en el código. (Beck Kent, 2002, s.p.)

Otra forma de enumerar las tres fases del ciclo es a través de un código de colores:

- Rojo para indicar fallo,
- Verde para indicar éxito o paso de prueba y
- Azul o Purpura para refactorización. (Beck Kent, 2002, s.p.).

Esta es una descripción metafórica del ciclo ya que las herramientas de pruebas suelen colorear en rojo aquellas especificaciones que no se cumplen y en verde las que tienen éxito, entonces cuando se escribe la prueba, el primer color es rojo porque todavía no existe código que implemente el requisito. Una vez implementado, este pasa a verde. Cuando se ha dado los tres pasos de la especificación que se ocupa, se toma la siguiente y se vuelve a repetir. (Beck Kent, 2003, s.p.)

No existe relación entre el tamaño y la aplicabilidad de T.D.D. El éxito está en saber separar y priorizar. Se puede usar la metodología Scrum para gestionar adecuadamente el *backlog* del producto o combinar XP y Scrum. La figura 2 muestra un diagrama de estados del funcionamiento de la técnica T.D.D. (Agiledata, s.f., s.p.)

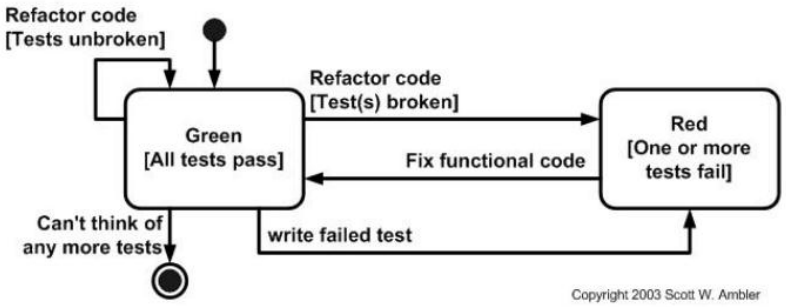


Figura No 2-1. Diagrama de estados de la técnica Test-Driven Development
Recuperado de: http://agiledata.org/essays/tdd.html

1.5.1. Escribir especificación de requisito

En esta parte se debe hacer dos preguntas clave:

¿Cómo se escribe una prueba para un código que todavía no existe? (Ble Carlos, s.f., s.p.)

La cual se responde con otra pregunta

¿Es posible escribir una especificación antes de implementarla? (Ble Carlos, s.f., s.p.)

Tomando como ejemplo, las especificaciones de requerimiento JAVA o J.S.R. (*JAVA Specification Request*), estas se escriben para que luego terceras partes las implementen. (Ble Carlos, s.f., s.p.)

La palabra especificación podría entenderse como algo que es inamovible, algo preestablecido y fijo, pero por el contrario una prueba se puede modificar. Para escribir una especificación, se debe pensar primero en cómo se quiere que sea la interfaz de programación de aplicaciones o Application Programming Interface (A.P.I.) del sujeto bajo prueba o *Subject Under Test* (S.U.T.) es decir, se realizan bosquejos antes. (Ble Carlos, s.f., s.p.)

Se debe intentar visualizar cómo sería el código del S.U.T. si ya estuviera implementado y cómo se comprueba, si efectivamente hace lo que le debe hacer. La principal diferencia con una J.S.R. es que no se diseñan todas las especificaciones antes, sino que se siguen los tres pasos del algoritmo T.D.D. (Ble Carlos, s.f., s.p.)

Cuando se piensa en que se debe probar una funcionalidad antes de haberla escrito, cambia considerablemente el código fuente resultante por lo que es importante tener especial cuidado de que el código sea cómodo, claro y cumpla con el objetivo. (Ble Carlos, s.f., s.p.)

1.6. Desarrollo dirigido por pruebas de aceptación (A.T.D.D.)

El sitio Agilealliance, muy reconocido en el mundo de las metodologías ágiles, sobre A.T.D.D., menciona:

> Análogamente al desarrollo impulsado por pruebas, el desarrollo impulsado por pruebas de aceptación (ATDD) implica a los miembros del equipo con diferentes perspectivas (cliente, desarrollo, pruebas), están colaborando para escribir pruebas de aceptación antes de implementar la funcionalidad correspondiente. Las discusiones colaborativas que se producen para generar la prueba de aceptación se refieren a 3 preguntas ¿qué problema estamos tratando de resolver?, ¿cómo podemos resolver este problema? Y qué pasa con las pruebas. (Agilealliance, s.f., s.p.).

1.7. Pruebas de software

> Las pruebas de software consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba, debidamente seleccionados de por lo general infinitas ejecuciones de dominio, contra la del comportamiento esperado. Son una serie de actividades que se realizan con el propósito de encontrar los posibles

fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo. (Ecured, s.f., s.p.)

1.7.1. Pruebas unitarias

Las pruebas unitarias comprueban el correcto funcionamiento de una unidad de software lo más pequeña posible, un experto en la materia dice sobre las pruebas unitarias:

> Verifican el comportamiento de una sola clase o método que es una consecuencia de una decisión de diseño. Este comportamiento es por lo general no relacionado directamente con los requerimientos excepto cuando un fragmento clave de lógica de negocio está encapsulado dentro de la clase o el método en cuestión. Estas pruebas son escritas por los desarrolladores para su propio uso; ayudan a los desarrolladores a describir lo que "parece" que resume el comportamiento de la unidad en forma de pruebas. (Meszaros Gerald, 2007, s.p.)

1.7.2. Principios de las pruebas unitarias

Las pruebas unitarias se diferencian de otros tipos de pruebas por tener ciertas características que se mencionan a continuación:

Atomicidad.- Las pruebas deben tener el mínimo código posible y probar un único método.
Independencia.- La prueba de un método o clase no debe afectar a las demás
Inocuidad.- La ejecución de una prueba unitaria debe ser inofensiva a los datos físicos del sistema.
Rapidez.- Las pruebas unitarias deben ejecutarse de forma rápida y automática. (Ble Carlos, s.f., s.p.)

1.7.3. Herramientas para realizar pruebas unitarias

1.7.3.1. dbUnit.

> dbUnit es una extensión de JUnit, utilizable con ANT, destinada a proyectos de base de datos que, entre otras cosas, pone la base de datos en un estado conocido entre los funcionamientos de prueba. Esta es una excelente manera de evitar la gran cantidad de problemas que pueden ocurrir cuando un caso de prueba corrompe la base de datos y hace controles posteriores o agravar el daño. (Dbunit, s.f., s.p.).

1.7.3.2. xUnit

XUnit es la propuesta de la empresa de software Microsoft, para realizar pruebas unitarias en sus aplicaciones basadas en .NET, según dos desarrolladores que utilizan este framework, afirman que: "xUnit.net es un framework de desarrollo de software, construido para soportar la técnica Test-Driven Development, con el objetivo de simplificar la construcción de pruebas." (Newkirk James, Wilson Brad, s.f., s.p.)

1.7.3.3. jUnit

JUnit es al igual que XUnit un framework, pero para las aplicaciones realizadas en JAVA, y de código abierto, que se puede descargar y agregarse como un empaquetado .jar al proyecto o instalarse como dependencia en aplicaciones Maven. La organización creadora dice sobre este framework: "JUnit es un marco simple para escribir pruebas repetibles. Es una instancia de la arquitectura xUnit para los marcos de pruebas unitarias." (Junit, 2002, s.p.)

1.7.3.4. Spock

Spock es un lenguaje de especificación de dominios (Domain Specific Language o abreviado D.S.L.) que utiliza jUnit para simplificar el realizar pruebas unitarias. Cierto autor dice sobre Spock: "El marco de pruebas Spock es un poderoso D.S.L., construido sobre Groovy que permite la escritura de pruebas fácilmente y extremadamente legible que se presta bien a los procesos de desarrollo de software como Test-Driven Development. y Behaviour-Driven Development. (Geraint Jones, 2002, s.p.)

1.7.3.5. Selenium

Selenium es un framework de pruebas que permite la escritura y ejecución automática de dichas pruebas pero para la interfaz de usuario o capa de presentación (en un modelo de n-capas). Según el sitio oficial Selenium "tiene el apoyo de algunos de los vendedores de navegadores más grandes y han tomado (o están tomando) los pasos para hacer de Selenium una parte nativa de su navegador. También es la tecnología principal en innumerables herramientas de automatización de navegadores, APIs y frameworks". (Seleniumhq, 2004, s.p.)

Proporciona dos entornos para ser utilizado, por medio de un complemento para el navegador o como un conjunto de librearías para el Entorno de Desarrollo Integrado, su funcionamiento se representa en la Figura No 3-1, en la página siguiente.

1.7.3.6. Mocha

Al igual que Selenium, Mocha es un framework de pruebas para la capa de presentación, Según su sitio web, dice: "Mocha es un framework de prueba de JavaScript para texto enriquecido en Node.js, en el navegador, haciendo pruebas asincrónicas, simples y divertidas. Con Mocha, las pruebas se ejecutan en serie, lo que permite informes flexibles y precisos, mientras que asigna excepciones a los casos de prueba." (Mochajs, s.f., s.p.)

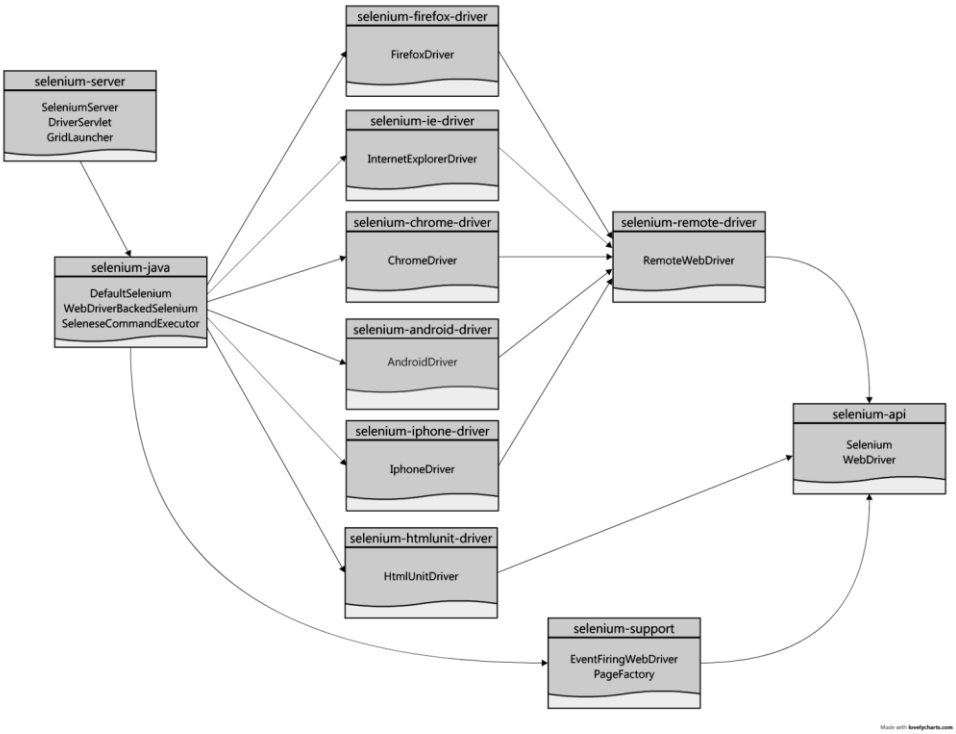


Figura No 3-1. Funcionamiento de Selenium

Recuperado de http://www.seleniumhq.org/images/maven.png

1.7.3.7. qUnit

"QUnit es un framework de pruebas potente y fácil de usar de Javascript. Utiliza jQuery, jQuery UI y jQuery para móviles y es capaz de probar cualquier código Javascript genérico, incluyéndose a sí mismo."(Qunit, s.f., s.p.)

1.7.3.8. JWebUnit

"JWebUnit es un framework de pruebas basado en JAVA para aplicaciones web. Envuelve los existentes frameworks de prueba como Selenium y HtmlUnit, con una interfaz unificada simple

pruebas para permitirle probar rápidamente la corrección de las aplicaciones web.” (Jwebunit, s.f., s.p.)

La Figura No 4-1 muestra la arquitectura del framework JWebUnit y su relación con Selenium y jUnit.

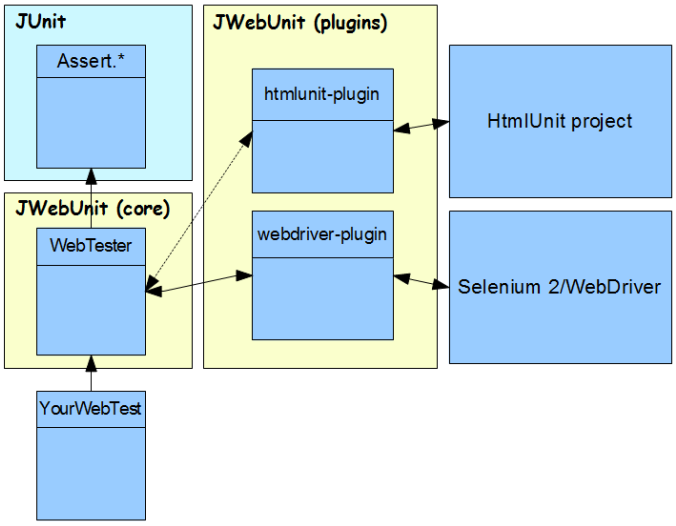


Figura No 4-1. Arquitectura *JWEBUnit*

Recuperado de https://jwebunit.github.io/jwebunit/

1.7.4. Características de las Pruebas unitarias

- Automatizable: Se deben poder ejecutar las pruebas de manera automática y lo más rápidamente posible.
- Completas: Deben comprobar la mayor cantidad de código posible.
- Repetibles: Se deben poder ejecutar en cualquier momento y tantas veces como sea necesario.
- Independientes: Cada una de las pruebas debe poder ejecutarse sin depender de las otras.
- Profesionales: Se deben tratar a las pruebas unitarias como si del mismo código de la aplicación se tratase, documentando y haciendo el código lo más eficiente posible. (Comas Sergi, 2016, s.f.)

1.7.5. Anatomía de una prueba unitaria

En teoría las pruebas unitarias están conformadas por tres partes principales conocidas comúnmente como: Organizar, Actuar y Afirmar o Arrange Act Assert (A.A.A.), a continuación, se explica cada una.

Organizar. - En esta sección se inicializan las variables que van a intervenir en la prueba incluyendo la instanciación de objetos con los atributos necesarios.
Actuar. - Ejecuta el código del método o clase a probar.
Afirmar- Consiste en una afirmación que permitirá saber si pasó o no la prueba comparando el resultado del método con el valor que debería devolver. (Ble Carlos, s.f., s.p.)

1.7.6. Implementación de una prueba unitaria

Para la implementación de pruebas unitarias se requiere primeramente crear una clase denominada test case o caso de prueba en la cual se prepararán sus variables se definirán las pruebas con instrucciones de inicialización (*set up*) y se compararan los resultados. (Ble Carlos, s.f., s.p.)

Anotaciones más usadas en JUnit:

@Before
@BeforeClass
@After
@AfterClass
@Ignore
@Test(timeout=500)
@Test(expected=IllegalArgumentException.class) (Ble Carlos, s.f., s.p.)

1.7.7. Plan de pruebas

Como se citó anteriormente, Test-Driven Development, trata más de diseño de pruebas que de la ejecución automática de pruebas. Así es que IBM en su sitio web de aprendizaje dice:

> El plan de pruebas describe el ámbito del esfuerzo de prueba general y proporciona un registro del proceso de planificación de prueba. El plan de pruebas puede configurarse para satisfacer las necesidades del equipo. Normalmente, un plan de pruebas identifica requisitos, riesgos, casos de prueba, los entornos de prueba a probar, objetivos empresariales y de calidad, planificaciones de prueba y otros elementos. Los planes de prueba relacionados pueden agruparse en una relación maestro-hijo anidada (IBM Knowledge Center, s.f., s.p.).

Para realizar el plan de pruebas existen algunas herramientas como por ejemplo Rational Team Concert 4.0.0 de I.B.M., un software de licencia comercial que entre muchas funciones

proporciona un gran repertorio de funciones para gestionar la calidad del software que se está desarrollando.

1.7.7.1. Casos de pruebas

"Un caso de prueba responde la pregunta ¿qué probaré?. Los casos de prueba se desarrollan para definir las cosas que es necesario validar a fin de asegurar que el sistema funciona correctamente y está construido con un alto nivel de calidad." (IBM Knowledge Center, s.f., s.p.)

Cuando se utiliza la técnica Test-Driven Development, es importante considerar el número total de Casos de prueba (T.C.), que se requieren para probar el software en construcción de manera eficiente y que abarque el mayor número de posibilidades para tener una densidad de errores mínima con el menor número de pruebas, esto es debido a que realizar pruebas también tiene un costo adicional en tiempo y procesamiento de datos.

"Para seleccionar casos de pruebas, las técnicas más comunes en procesos de prueba buscan seleccionar buenos casos de prueba, esto es, casos que tengan una alta probabilidad de encontrar un error" (Ramos I y Dolado J, 2010, p 46).

1.7.7.2. Conjunto de pruebas (Suit de pruebas)

Cuando se tienen que realizar varios tipos de pruebas sobre varias secciones de código de una aplicación, se utilizan conjuntos de pruebas o *suits*.

> Una suite de pruebas es una colección de casos de prueba que se agrupan para ejecutar pruebas. Las suites de pruebas también resultan útiles para realizar pruebas de: verificación de compilación, Pruebas aleatorias, Pruebas de integración, pruebas de verificación funcional, Pruebas de regresión (IBM Knowledge Center, s.f., s.p.)

1.7.7.3. Script de pruebas

Los script de pruebas corresponden al código generado por el entorno de desarrollo integrado que se esté utilizando.

> Un *script de prueba* es un script manual o automático que contiene instrucciones para implementar un caso de prueba. Puede escribir scripts de prueba manuales para ser ejecutados por un probador humano o bien puede automatizar algunas o todas las

> instrucciones del script de prueba. Puede asociar scripts de prueba funcional, scripts de prueba de rendimiento y scripts de prueba de seguridad automatizados a un caso de prueba (IBM Knowledge Center, s.f., s.p.)

1.7.7.4. Ejecución de pruebas

Consiste en ejecutar el script de pruebas creado anteriormente, en la práctica se recomienda considerar todos los drivers requeridos para evitar confusiones entre errores en las pruebas por fallos en el software o por ejecución automática de las pruebas.

> La ejecución de pruebas es el proceso de ejecución de un caso de pruebas o combinación de varios casos de prueba en una suite de pruebas para luego ejecutar la suite de pruebas. Aquí se generan registros de ejecución de caso de prueba o registros de ejecución de suite de pruebas para definir los atributos de entorno para ejecutar la prueba y, a continuación, ejecutarlos (IBM Knowledge Center, s.f., s.p.)

1.7.8. Dobles de prueba y objetos MOCK

Los dobles de pruebas son objetos son objetos que se crean para suplantar a los originales al momento de la realización de la prueba (de allí el nombre doble de prueba). Un objeto MOCK por su parte es un tipo de doble de prueba que tiene alguna característica adicional como veremos citando a dos autores expertos en la materia:

> Los Mocks, son objetos pre-programados con expectativas que conforman la especificación de lo que se espera que reciban las llamadas", es decir, son objetos que se usan para probar que se realizan correctamente llamadas a otros métodos, por ejemplo, a una web API, por lo que se utilizan para verificar el comportamiento de los objetos. (Meszaros Gerard, 2006, s.p.)

> El término Objetos de Imitación (Mock Objects) se ha convertido en popular para describir objetos especiales que imitan a objetos reales para hacer pruebas. La mayoría de los entornos de programación ya tienen frameworks que facilitan la creación de objetos de imitación. Lo que a menudo es inadvertido sin embargo, es que los objetos de imitación son una forma especial de objeto para Pruebas, que proporciona un estilo diferente de escribir Pruebas. En este artículo explicaré cómo trabajan los objetos de imitación, cómo fomentan la escritura de Pruebas basados en la verificación del

> comportamiento y cómo la comunidad que hay alrededor de ellos los usa para desarrollar un estilo diferente de hacer pruebas. (Flower Martin, 2008, p. 1)

Martin Fowler también publicó un artículo: "Los mocks no son stubs", el mismo que señala que estos tipos pueden ser:

- **Dummy**: se pasa como argumento, pero nunca se usa realmente. Normalmente, los objetos dummy se usan sólo para rellenar listas de parámetros.
- **Fake**: tiene una implementación que realmente funciona, pero por lo general, toma algún atajo o cortocircuito que le hace inapropiado para producción (como una base de datos en memoria, por ejemplo).
- **Stub**: proporciona respuestas predefinidas a llamadas hechas durante los Pruebas, frecuentemente, sin responder en absoluto a cualquier otra cosa fuera de aquello para lo que ha sido programado. Los stubs pueden también grabar información sobre las llamadas; tal como una pasarela de email que recuerda cuántos mensajes envió.
- **Mock**: objeto pre-programado con expectativas que conforman la especificación de cómo se espera que se reciban las llamadas. Son más complejos que los stubs aunque sus diferencias son sutiles. Las veremos a continuación. (Flower Martin, 2008, s.p.)

1.7.9. Pruebas de caja blanca

"Las pruebas de caja blanca son una técnica de verificación que los ingenieros de software pueden utilizar para examinar si su código funciona como se espera." (Williams, 2006, p.60).

> Son pruebas estructurales. Conociendo el código y siguiendo su estructura lógica, se pueden diseñar pruebas destinadas a comprobar que el código hace correctamente lo que el diseño de bajo nivel indica y otras que demuestren que no se comporta adecuadamente ante determinadas situaciones. Ejemplos típicos de ello son las pruebas unitarias. (Luna J, 2009, p.45).

1.7.10. Pruebas de caja negra

"Uso de pruebas de caja negra son técnicas para examinar el diseño de alto nivel y la especificación de requisitos del cliente para planificar los casos de prueba para asegurar el código hace lo que pretende hacer." (Williams, 2006, p.35).

> Son pruebas funcionales. Se parte de los requisitos funcionales, para diseñar pruebas que se aplican sobre el sistema sin necesidad de conocer como está construido por

dentro (Caja negra). Las pruebas se aplican sobre el sistema empleando un determinado conjunto de datos de entrada y observando las salidas que se producen para determinar si la función se está desempeñando correctamente por el sistema bajo prueba. Ejemplos típicos de pruebas de caja negra son la comprobación de valores límite, pruebas de integridad de la base de datos, pruebas de situaciones de excepción, o pruebas de rendimiento del sistema. Presentan una limitación en cuanto a que es prácticamente imposible reproducir todo el espectro por la innumerable cantidad de combinaciones de entrada posibles, agravada por el desconocimiento de la lógica interna. (Luna, J., 2009, p.47)

1.7.11. Pruebas funcionales

Las pruebas funcionales se corresponden de alguna manera con los requerimientos funcionales, están en por encima de las pruebas unitarias, pero por debajo de las pruebas de aceptación. A continuación, una explicación más detallada.

Todas las pruebas son en realidad funcionales, puesto que todos ejercitan alguna función del S.U.T, aunque en el nivel más elemental sea un m método de una clase. No obstante, cuando se habla del aspecto funcional, se distingue entre prueba funcional y prueba no funcional. Una prueba funcional es un subconjunto de las pruebas de aceptación. Es decir, comprueban alguna funcionalidad con valor de negocio. Hasta ahora, todas las pruebas de aceptación que hemos visto son pruebas funcionales. Las pruebas de aceptación tienen un ámbito mayor porque hay requerimientos de negocio que hablan de tiempos de respuesta, capacidad de carga de la aplicación, etc.; cuestiones que van más allá de la funcionalidad. Una prueba funcional es una prueba de aceptación, pero, uno de aceptación, no tiene por qué ser funcional. (Ble Carlos, 2010, p.76)

1.7.12. Pruebas de aceptación

Las pruebas de aceptación se realizan ya con el cliente del producto, y están entre las pruebas finales del proyecto en marcha.

Análogo al desarrollo orientado a pruebas, esta práctica consiste en el uso de pruebas de aceptación automatizadas con la restricción adicional de que estas pruebas se escriben antes de implementar la funcionalidad correspondiente. En la situación ideal (sin

embargo, raramente alcanzada en la práctica), el propietario del producto, cliente o experto es capaz de especificar nuevas funcionalidades escribiendo nuevas pruebas de aceptación o casos de prueba, sin necesidad de consultar a los desarrolladores. (AgileAlliance, s.f., s.p.)

1.7.13. Pruebas de integración

Integran todos los tipos de pruebas y *suits* de pruebas en una sola ejecución, pruebas unitarias, pruebas de interfaces, funcionales, aceptación, etc. En este tipo de prueba se basa la entrega continua. El autor Carlos Ble, dice, sobre las pruebas de integración:

> Como su nombre indica, integración significa que ayuda a unir distintas partes del sistema. Una prueba de integración puede escribir y leer de base de datos para comprobar que, efectivamente, la lógica de negocio entiende datos reales. Es el complemento a los Pruebas unitarios, donde habíamos falseado el acceso a datos para limitarse a trabajar con la lógica de manera aislada. Una prueba de integración podría ser aquel que ejecuta la capa de negocio y después consulta la base de datos para afirmar que todo el proceso, desde negocio hacia abajo, fue bien. Son, por tanto, de granularidad más gruesa y más frágiles que las pruebas unitarias, con lo que el número de pruebas de integración tiende a ser menor que el número de pruebas unitarias. (Ble Carlos, 2010, s.p.)

> Una vez que se ha probado que dos módulos, objetos o capas se integran bien, no es necesario repetir la prueba para otra variante de la lógica de negocio; para eso habrá varias pruebas unitarias. Aunque los Pruebas de integración pueden saltarse las reglas, por motivos de productividad es conveniente que traten de ser inocuos y rápidos. Si tiene que acceder a base de datos, es conveniente que luego la deje como estaba. Por eso, algunos frameworks para Ruby y Python entre otros, tienen la capacidad de crear una base de datos temporal antes de ejecutar la batería de pruebas, que se destruye al terminar las pruebas. (Ble Carlos, 2010, s.p.)

> Concluimos el capítulo sin revisar otros tipos de Pruebas, porque este no es un libro sobre cómo hacer pruebas de software exclusivamente sino sobre cómo construir software basándolo en ejemplos que ilustran los requerimientos del negocio sin ambigüedad. Los Pruebas unitarios, de integración y de aceptación son los más importantes dentro del desarrollo dirigido por Pruebas. (Ble Carlos, 2010, s.p.)

1.7.14. Pruebas de sistema

> Es el mayor de las pruebas de integración, ya que integra varias partes del sistema. Se trata de una prueba que puede ir, incluso, de extremo a extremo de la aplicación o del sistema. Se habla de sistema porque es un término más general que aplicación, pero no se refiere a administración de sistemas. (Ble Carlos, 2010, s.p.)

1.7.15. Integración continua

El proceso o práctica de integración continua tiene como objetivo principal el de comprobar que las actualizaciones en el código fuente no generen retrasos y/o anomalías en el desarrollo de una aplicación.

"La integración continua fue utilizada por IBM para el desarrollo del OS/360 en los años 60." (Creative Commons, s.f., s.p.)

> La integración continua no es una herramienta sino una práctica derivada de la programación extrema o eXtreme Programming (XP). Los desarrolladores trabajando en una misma aplicación integran el programa en el que trabajan lo más a menudo posible. En cada integración se genera un proceso basado en una plataforma que verifica automáticamente el funcionamiento de la aplicación para detectar fallos cuanto antes. (Creative Commons, s.f., s.p.)

1.7.15.1. Hudson

"Hudson controla la ejecución de trabajos repetitivos, como la construcción de un proyecto de software" (Prakash Winston, 2014, s.p.). Hudson tiene dos tareas principales:

> 1. Hudson proporciona un sistema de integración continua fácil de usar, haciéndolo más fácil para los desarrolladores para integrar cambios en el proyecto y que facilita a los usuarios obtener una estructura fresca. La fabricación automatizada continua aumentando la productividad. (Prakash Winston, 2014, s.p.)

> 2. Monitores de ejecuciones de trabajos externos, mantiene esos productos y hace que sea fácil notar cuando algo está mal. (Prakash Winston, 2014, s.p.)

1.7.15.2. Jenkins

Jenkins es una herramienta web de integración continua, desarrollada a partir de Hudson (de allí la similitud en la interfaz), que realiza una ejecución automática de todas las pruebas de un proyecto (siempre que existan pruebas de integración definidas), en la Figura No 5-1, se observa el funcionamiento de Jenkins. El sitio web de Jenkins habla sobre su herramienta, señalando:

> Jenkins es un motor de automatización con todo un ecosistema de *plugins* de soporte para todas sus herramientas preferidas de tuberías, su objetivo es la integración continua, pruebas automáticas o entrega continua. Puede ser utilizado tanto en Windows como en Linux, para lo cual requiere del uso de repositorios dependiendo del tipo de aplicación que se implemente. (Jenkins, s.f., s.p.)

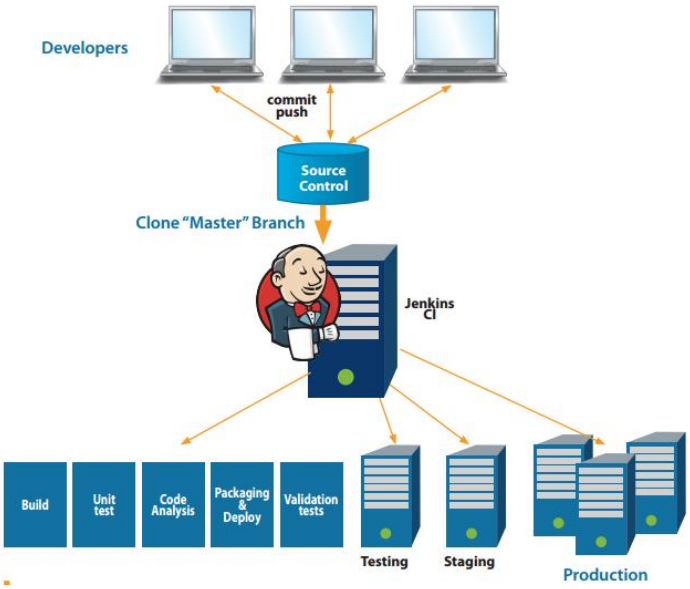


Figura No 5-1. Funcionamiento de Jenkins
Recuperado de: https://code2read.files.wordpress.com/2015/11/jenkins.png

JENKINS cuenta con una interfaz de usuario muy simple y fácil de usar, salvo por la configuración inicial que es más complicada. Sin embargo una vez configurado un proyecto permite integrar todas las pruebas realizadas de forma automática. La Figura No 6-1 muestra la interfaz inicial de JENKINS.

Figura No 6-1. Página inicial de Jenkins
Autor: Edgar Córdova

1.7.15.3. CruiseControl:

CruiseControl.NET es un servidor automatizado de integración continua, de código abierto, implementado con .NET Framework. El servidor automatiza el proceso de integración supervisando directamente el repositorio de control de origen del equipo. Cada vez que un desarrollador confirma un nuevo conjunto de modificaciones, el servidor iniciará automáticamente una compilación de integración para validar los cambios. Cuando se completa la compilación, el servidor notifica al desarrollador si los cambios que cometieron se integraron correctamente o no. (CruiseControl, s.f., s.p.)

1.7.16. Pruebas unitarias en Test-Driven Development

El uso de pruebas unitarias está muy extendido y existe una gran cantidad de información sobre cómo aplicarlos para algoritmos o pequeños programas que tienen datos de entrada y salida perfectamente claros y definidos, donde incluso a simple vista se sabe cuál será el resultado esperado dentro de estos programas están: algoritmos de búsqueda, ordenamiento, cálculo de áreas, series, etc. La Figura No 7-1, muestra la estructura del proceso de pruebas unitarias.

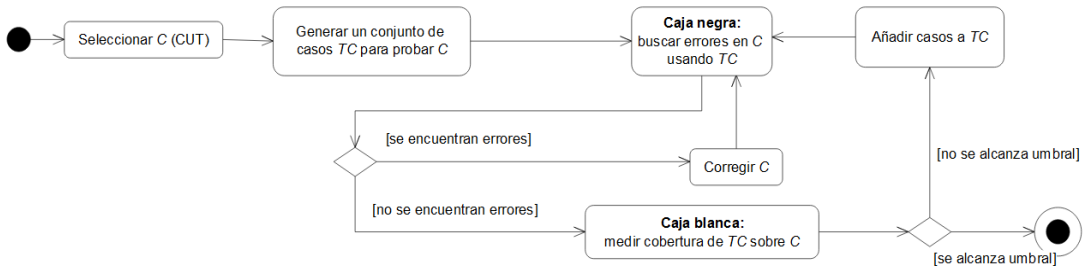


Figura No 7-1. Estructura del proceso de pruebas unitarias.
Autores: Macario Polo Usaola y Mario Piattini Velthuis.

Cuando se trata de proyectos medianos que incluyen una o más bases de datos resulta más complejo calcular el número de casos de prueba para pruebas unitarias sin olvidar que es posible

que se tenga más de una capa para un mismo modulo, sección o requerimiento, dependiendo de qué modelo o arquitectura se valla a utilizar. Resulta difícil hacer una prueba de caja negra para una base de datos o decidir qué nivel o capa de la aplicación se probará primero.

La situación mencionada en el párrafo anterior empeora al utilizar la técnica Test-Driven Development, donde las pruebas se realizan primero y no existe o no debería existir código alguno y en cada capa de la aplicación existen llamadas a métodos de otras clases, conexiones, bases de datos, controladores, etc., que requieren ser evaluados en fragmentos de código muy pequeños (una característica de las pruebas unitarias) y que pertenecen a un mismo requerimiento.

Si consideramos que el proceso de pruebas se puede representar mediante un gráfico en V, llamado Modelo V, donde se observa claramente que las pruebas unitarias están en lo más bajo de los niveles como muestra la Figura No 8-1.

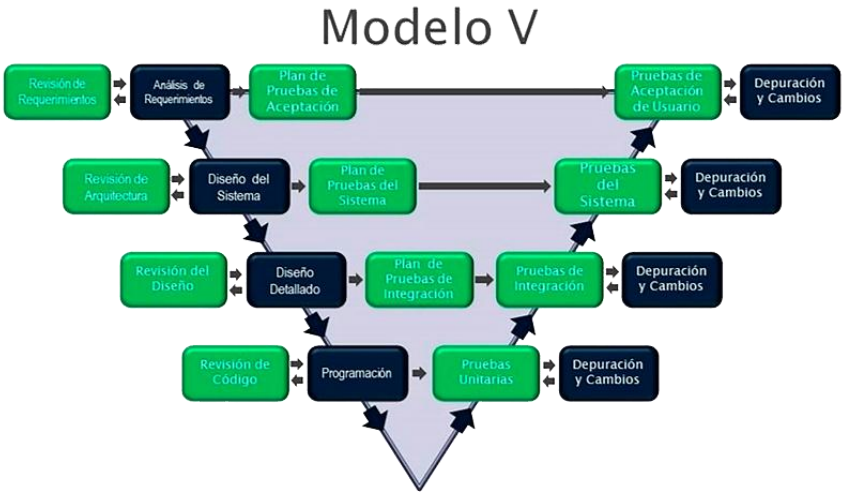


Figura No 8-1. ITSoluciones. Modelo V tradicional de pruebas de sofware.

Recuperado de:http://www.itesoluciones.com/wp-content/uploads/2015/11/pruebas-de-software.png

Desde luego la aplicación debe estar diseñada para empezar a construirla aplicando el desarrollo guiado por pruebas, es decir debe estar hasta la fase del diseño detallado, con el respectivo plan de pruebas que incluye además los casos de pruebas de cada requerimiento. Entonces si en el último nivel del modelo V, se encuentran las pruebas unitarias, como se trata de T.D.D., el bloque de programación iría después de las pruebas unitarias para cada requerimiento en forma de ciclo iterativo hasta que se complete el código como se muestra en la Figura No 9-1.

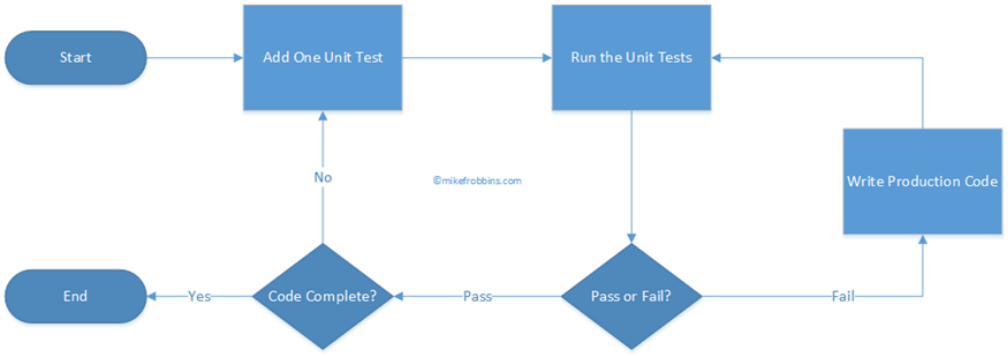


Figura No 9-1. Test Driven Development Workflow.

Recuperado de http://mikefrobbins.com/wp-content/uploads/2016/05/tdd-workflow875x400.png

1.7.16.1. Implementación de código

Se codifica lo mínimo necesario para que se cumpla la especificación y que la prueba pase. Típicamente, el mínimo código es el de menor número de caracteres porque mínimo quiere decir el que menos tiempo ha tomado escribirlo. No importa si el código tiene mala apariencia, posteriormente se arregla en el siguiente paso con las iteraciones. (Ble Carlos, 2010, s.p.)

Se puede tener algunas dudas sobre el comportamiento del S.U.T. ante distintas entradas, es decir, los distintos flujos condicionales que pueden entrar en juego; el resto de especificaciones de este bloque de funcionalidad. Se puede estar tentado a escribir el código que los gestiona sobre la marcha y, en ese momento, sólo la atención puede ayudar a contener el impulso y a anotar las preguntas que le han surgido al desarrollador en un lugar al margen para convertirlas en especificaciones que se retomarán después en iteraciones consecutivas. (Ble Carlos, 2010, s.p.)

1.7.16.2. Refactorización

Después del paso uno, escribir las pruebas y el paso dos, ejecutarlas, el paso tres implementar el código, la refactorización es cuarto paso en el algoritmo del desarrollo basado en pruebas.

Refactorizar no significa reescribir el código; reescribir es más general que refactorizar. Según Martín Fowler, refactorizar es *"modificar el diseño sin alterar su comportamiento"*. A ser posible, sin alterar su interfaz de programación de aplicación (A.P.I.) pública. (Ble Carlos, 2010, s.p.)

En este cuarto paso del algoritmo Test-Driven Development, rastrea el código (también el de la prueba) en busca de líneas duplicadas y se eliminan refactorizando. Además, se revisa que el código cumpla con ciertos principios de diseño y se refactoriza para que así sea. (Ble Carlos, 2010, s.p.)

Siempre que se llega al paso de refactorizar, y se elimina la duplicidad, se debe plantear si el método en cuestión y su clase cumplen el Principio de Única Responsabilidad o *Single Responsability Principle* (S.R.P.) y demás principios. (Ble Carlos, 2010, s.p.)

Dadas unas pre-condiciones, se aplican unos determinados cambios que mejoran el diseño del software mientras que su comportamiento sigue siendo el mismo. Mejora es una palabra ciertamente subjetiva, por lo que se emplea la métrica del código duplicado como parámetro de calidad. Si no existe código duplicado, entonces se ha conseguido uno de más calidad que el que presentaba duplicidad. (Ble Carlos, 2010, s.p.)

> Los IDE como Eclipse, Netbeans o VisualStudio, son capaces de llevar a cabo las refactorizaciones más comunes. Basta con señalar un bloque de código y elegir la refactorización Extraer-Método, Extraer-Clase, Pull-up, Pull-down o cualquiera de las muchas disponibles. El I.D.E. modifica el código por el desarrollador, asegurándose que no se cometen errores en la transición. (Ble Carlos, 2010, s.p.)
>
> La clave de una buena refactorización es hacerlo en pasos muy pequeños. Se hace un cambio, se ejecutan todas las pruebas y, si todo sigue funcionando, se hace otro pequeño cambio. Cuando se refactoriza, se piensa en global, se contempla la perspectiva general, pero se actúa en local. (Ble Carlos, 2010, s.p.)
>
> El verbo refactorizar no existe como tal en la Real Academia Española pero, tras discutirlo en la red, resulta la mejor traducción del término *refactoring*. La tarea de buscar y eliminar código duplicado después de haber completado los dos pasos anteriores, es la que más tiende a olvidarse. Es común entrar en la dinámica de escribir la prueba, luego el sujeto bajo pruebas(S.U.T.), y así sucesivamente olvidando la refactorización. Si de las tres etapas que tiene el algoritmo Test-Driven Development, se deja atrás una, lógicamente no está practicando T.D.D., sino otra cosa. (Ble Carlos, 2010, s.p.)

1.8. Normas para la protección de datos en la nube

En noviembre de 2013, el en ese entonces Secretario Nacional de la Administración Pública, Cristian Castillo, declaró que "la premisa detrás de la política pública [ecuatoriana] siempre ha sido garantizar la soberanía tecnológica" ("Una Minga por la Libertad Tecnológica", 2013), el reglamento emitido por la misma institución prohíbe a los servidores públicos el uso de nubes ubicadas fuera del territorio nacional. (Delgado Andrés, 2014, p.17)

Si bien es cierto que la seguridad informática es la principal preocupación, también debe considerarse la importancia de otras áreas como la gestión de la identidad, la gestión del control de acceso, la seguridad forense, la virtualización, la computación distribuida, entre otras. (Salazar, 2013, s.p.)

El 16 de marzo de 2010, se presentó ante la Asamblea Nacional el proyecto de "Ley de Protección a la Intimidad y a los Datos personales", el Legislativo resolvió su archivo por considerar que varias de las normas propuestas ya constan en la Constitución y en la legislación secundaria existente (Ley Orgánica de Garantías Jurisdiccionales y Control Constitucional, Ley de Transparencia y Acceso a la Información Pública), esto a pesar de que la propia constitución señala que debe existir una ley de protección a datos privados en registros públicos. ("Asamblea Nacional archiva el proyecto de Ley de Protección a la Intimidad y a los Datos Personales", 2010, s.p.)

Debido a la falta de una regulación específica para la Nube, los contratos o acuerdos de Nivel de Servicio representan el principal elemento de cumplimiento. Salazar (2013:108) propone una serie de principios para ser incluidos en un nuevo marco regulatorio específico para Ecuador, que abarque la protección de datos en la Nube:

Aprobación: Considerar si el tratamiento de datos necesita el consentimiento del titular, responsable de los datos, interesado de los datos, etc.

Delimitación de responsabilidades: Esclarecer cuáles son las responsabilidades específicas de las partes en un servicio de Cloud Computing.

Finalidad: Cual será la finalidad determinada para utilizar la plataforma de Cloud Computing, y si debe o no extenderse hacia otra finalidad.

Seguridad: Qué medidas técnicas y organizativas deben adoptarse para el tratamiento de datos en un entorno de Cloud. Si es posible establecerlo como un requisito para los proveedores de la Nube.

Transferencias Internacionales: Cómo debería realizarse el flujo transfronterizo de los datos personales y los niveles de protección que deben presentar los involucrados. Qué se reconoce por un adecuado nivel de protección.

Intervención de terceras partes que afecten el tratamiento de los datos: Qué requisitos deben cumplir las terceras partes cuando intervengan en un ambiente de Nube. Comunicación al cliente de quienes intervienen en el tratamiento de datos en la Nube.

Comunicación: Comunicación a los clientes acerca de todos los cambios y modificaciones importantes que se den en la plataforma de la Nube, que afecten el servicio, siendo claros y transparentes.

Indemnizaciones, garantías y sanciones: Cómo retribuyen los proveedores de servicios de la Nube al cliente en el caso de provocarle daños irreparables en el tratamiento de los datos.

Propiedad intelectual: Cómo se interpreta la propiedad intelectual de los datos colocados en una infraestructura de Nube. Qué sanciones se colocarían al mal uso o abuso de contenido con derechos de autor. (Salazar, 2013, s.p.)

1.9. Prototipos de software

1.9.1. Definición.

"Es un modelo del comportamiento del sistema que puede ser usado para entenderlo completamente o ciertos aspectos de él y así clarificar los requerimientos… Un prototipo es una representación de un sistema, aunque no es un sistema completo, posee las características del sistema final o parte de ellas" (Ingeniería en sistemas UNL, 2009, s.p.)

1.9.2. Características de los prototipos

- ✓ Funcionalidad limitada.
- ✓ Poca fiabilidad.
- ✓ Características de funcionalidad pobres.
- ✓ Alto grado de participación del usuario el cual evalúa los prototipos, propone mejoras y detalla requisitos.
- ✓ Alto grado de participación del analista de sistemas, ya que en muchos casos los usuarios no pueden indicar los requisitos sin tener experiencia con el sistema.
- ✓ El prototipo da mayor conocimiento al usuario y analistas ayudando a que el usuario aprenda a utilizar el sistema. (Ingeniería en sistemas UNL, 2009, s.p.) .

1.10. Paneles de control.

El Panel de Control en los servicios de Hosting se refiere a la interfaz que ofrece la compañía de Hosting para el mantenimiento y monitorización del sitio hospedado. Generalmente son en Sistemas Operativos GNU/Linux sin embargo también existen otras plataformas usadas para esto como Windows Server, BSD, etc. (Okhosting, s.f., s.p.)

Estas son algunas de las funciones que contienen la mayoría de los paneles:

- ✓ Estadísticas de visitas.
- ✓ Detalles sobre el ancho de banda usado.
- ✓ Manejo de archivos.
- ✓ Configuración de la cuenta de Email.
- ✓ Manejo de bases de datos
- ✓ Manejo de usuarios de FTP
- ✓ Acceso a logs del servidor.
- ✓ Manejo de sub-dominios. (Okhosting, s.f., s.p.)

CAPÍTULO II

2. MARCO METODOLOGICO

2.1. Tipo de investigación

Investigación experimental aplicada, puesto que se va a desarrollar una aplicación para una empresa y a la vez se pide comprobar su validez a través de métodos experimentales.

2.2. Métodos, técnicas e instrumentos

2.2.1. Metodología

La realización de este trabajo de titulación se ha realizado basándose en las fases de la metodología de Programación Extrema, por ser la más adecuada para aplicar el desarrollo guiado por pruebas según el propio autor de X.P. y T.D.D., mas, por cuestiones de escasez de personal la práctica de programación en parejas se pasa por alto. La técnica Test-Driven Development se aplica en un ciclo repetitivo hasta que la aplicación queda terminada, un diagrama aproximado de la metodología se observa en la Figura No 1-2.

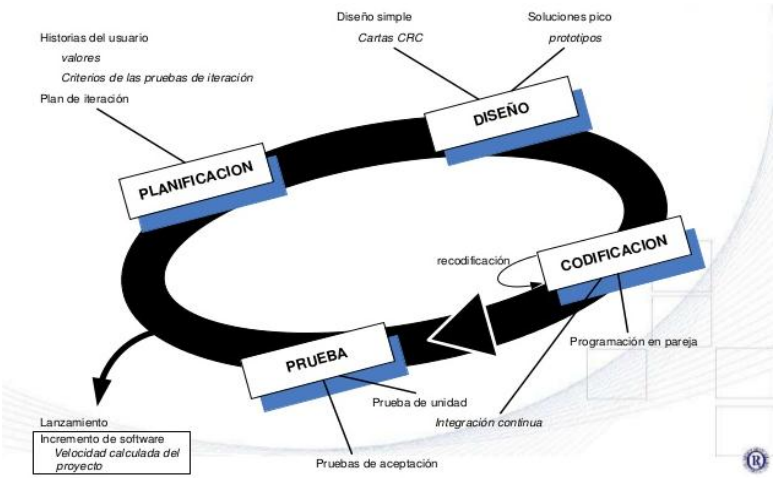


Figura No 1-2. Ciclo de proceso Extreme Programming
Recuperado de: https://www.slideshare.net/ELIUDLACSM/metodologia-xp-cortesserranoeliud

Para la realización del experimento se elabora una guía detallada de instalación de la colección de desarrollo JAVA o JAVA Development Kit (J.D.K.), instalación de PostgreSQL, Glassfish y finalmente el despliegue y ejecución de una pequeña aplicación web JAVA. Dicho manual se referencia en el Anexo D.

2.2.1.1. Primera fase: Planificación del proyecto

Comienza con el diseño de las historias de usuario, en varias reuniones se determinó los requisitos funcionales de la aplicación. Entonces se realizó el diseño U.M.L. Comenzando con los Diagramas de Casos de Uso, luego el diseño de la base de datos del cual se obtendrá el Diagrama de Clases y Objetos. En esta fase también se desarrolló los Diagramas de Secuencia y Colaboración, Diagrama de Componentes y de Despliegue.

2.2.1.2. Segunda fase: Diseño y mejoramiento de pruebas

Se configuró una herramienta de integración continua, luego se elaboró del Plan de Pruebas definiendo cada caso de pruebas para cada requerimiento. Del Plan de Pruebas se parte para escribir los Casos de Pruebas.

2.2.1.3. Tercera fase: Ejecución de pruebas

Comienza con la ejecución de las pruebas de la especificación inicial donde evidentemente todas fallarán. Sin embargo, es el paso inicial para empezar a escribir el código. De aquí en adelante se generará un bucle repetitivo entre la tercera fase, la cuarta y quinta fase, hasta que la aplicación esté terminada.

2.2.1.4. Cuarta fase: Codificación

Escribir el código luego de escribir la prueba, y asegurarse que la prueba o las pruebas concluyen satisfactoriamente. En esta fase también incluye la refactorización del código.

2.2.1.5. Quinta fase: Integración.

Se realiza con la herramienta de integración continua especificada en la segunda fase de la metodología, la herramienta llama a todas las pruebas de integración especificadas en el proyecto, es decir con el sufijo IT y a partir de aquí ejecutaron todas las demás, incluyendo las pruebas unitarias.

Test-Driven Development (T.D.D.), se aplica como técnica de desarrollo de software y sus que sus fundamentos y principios junto con los de pruebas unitarias e integración continua fueron estudiados detalladamente.

2.2.2. Técnicas

*2.2.2.1. Entrevista***:** Se realizó varias entrevistas con el representante legal de la empresa KAPYASOFT para determinar los requerimientos más importantes del prototipo de la aplicación.

2.2.2.2. Observación directa: Antes de realizar el prototipo se observará el lugar de trabajo de los desarrolladores de la empresa KAPYASOFT, el tiempo que se demoran en desplegar una aplicación desde cero en un mismo sistema operativo.

2.2.2.3. Consulta on-line: Se requiere recopilar información de servicios online similares locales y de otros países como Amazon, Nexus, Elastics y Jelastics.

2.2.2.4. Técnicas estadísticas. - Para eficiencia de la herramienta se utilizó la distribución t-student, una técnica descriptiva para análisis de medias y varianzas a través de la medición de tiempos de despliegue.

2.2.3. Instrumentos

Se utilizaron varias herramientas informáticas para el desarrollo de la aplicación, pruebas y comprobación de la eficiencia del prototipo creado, como muestra la **Tabla No 1-2**.

2.3. Población y muestra

Para lograr el objetivo 4, se realizó un experimento tomando una muestra de 22 estudiantes de un curso de 46, Los estudiantes que se someten a este experimento se ha confirmado que tienen aprobados cursos de Linux, Aplicaciones web y en el manejo de servidores remotos en la Nube. Se divide a la muestra en dos grupos de 8 estudiantes donde al primer grupo se le proporciona una guía detallada para instalar una misma aplicación en un mismo servidor remoto con la instalación mínima del S.O. Centos 6.4, mientras que a la otra mitad se le proporciona una cuenta de acceso al panel de control PROTPANEL. El objetivo de cada estudiante es desplegar la aplicación y que la misma esté disponible en Internet. La hipótesis nula es que "NO existe una diferencia significativa entre los dos métodos de despliegue de aplicaciones web JAVA", por otro lado, la hipótesis alternativa dice que "SI existe una diferencia significativa utilizando la herramienta PROTPANEL y una guía detallada de despliegue de aplicaciones."

Tabla No 1-2. Instrumentos

No	Nombre	Versión	Propósito
1	JAVA, JDK	1.8	Lenguaje de programación
2	Netbeans	8.2	Entorno de desarrollo integrado
3	Jenkins	2.26	Integración continua
4	JUnit,	4.1	Pruebas unitarias y de integración
5	Selenium + Webdriver	2.0	Pruebas de interfaces
6	GIT	Sin versión	Repositorio de código fuente
7	TIMEPANTHER	Sin versión	Control de tiempo

Autor: Edgar Córdova

2.4. Ambientes de prueba

Debido a que la disciplina o técnica que se utilizó para la realización de la aplicación está basado en pruebas, desde el comienzo se tuvo que configurar o adecuar un ambiente de pruebas apropiado. Tal es el caso de la herramienta de entrega continua JENKINS, impulsada por la empresa ORACLE, que proporciona un ambiente de pruebas automático valiéndose para ello de un amplio conjunto de herramientas, *plugins*, paquetes y repositorios.

2.4.1. Configuración de JENKINS

Esta herramienta tiene dos modos de ser instalado en Linux: descargando el Archivo Web (WAR) o con el comando "yum install jenkins", en la terminal de Linux. Para el correcto proceso de integración y ejecución automática de pruebas, la herramienta JENKINS debe configurarse correctamente según el tipo de aplicación que se esté desarrollando. Comienza con la creación de una cuenta de administrador quien se hará cargo de la construcción y ejecución de la aplicación, como se aprecia en la Figura No 2-2.

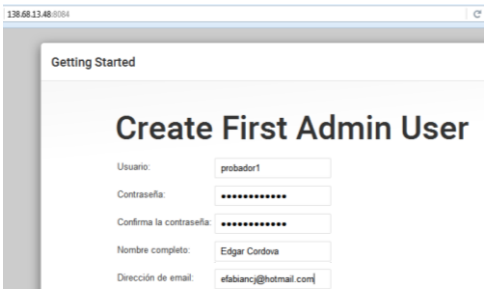


Figura No 2-2. Creación de la cuenta administrador de JENKINS
Autor: Edgar Córdova

A continuación, se le da un nombre al proyecto de integración y se escoge el tipo de proyecto como se indica en la Figura No 3-2, el cual dependiendo de la instalación previa de paquetes adicionales como ANT y MAVEN, entre los más importantes puede ser:

- ✓ Estilo Libre.- Cualquier tipo de aplicación.
- ✓ Proyecto MAVEN.- Proyecto MAVEN con dependencias.
- ✓ Pipeline.- Apto para construir gasoductos (antes flujos de trabajo) y agregar y coordinar otros proyectos ya existentes.
- ✓ Tarea Externa.- Para proyectos que ejecutan procesos en máquinas remotas.
- ✓ Para el proyecto PROTPANEL se seleccionó un proyecto estilo libre para poder manejar cualquier tipo de repositorio.

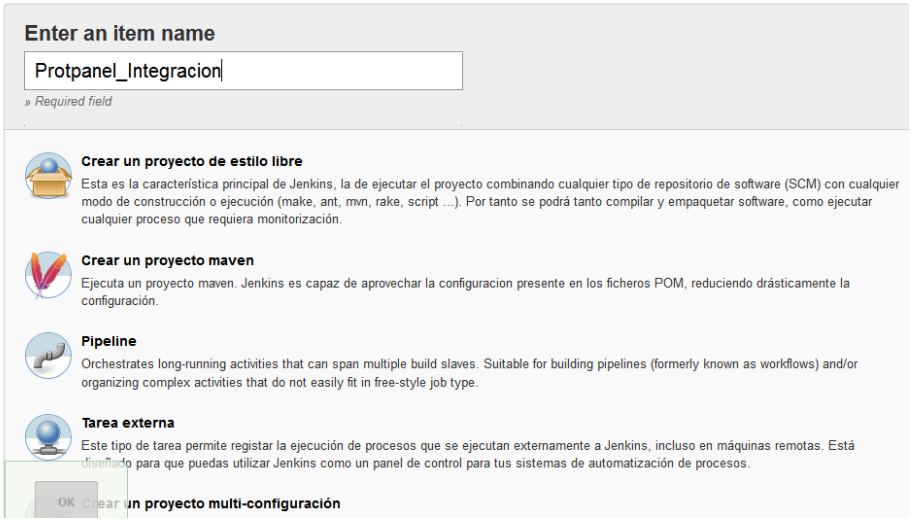


Figura No 3-2. Creación de un nuevo proyecto de integración.

Autor: Edgar Córdova

2.4.2. Instalando plugins necesarios

Para el correcto desempeño de JENKINS se requiere de un gran número de paquetes adicionales que se deben instalar luego de la instalación inicial. Dependiendo del tipo de aplicación que se

esté desarrollando, se deben instalar cuidadosamente. Sin embargo este paso puede llegar a ser un poco extenso y tedioso por lo que se resume los paquetes básicos:

Artifactory Plugin, Build timeout plugin, Credencials Binding Plugin, Deploy to container Plugin, Deployment notification, Email extension plugin, Extensive Choise Parameter plugin, Git Parameter plugin, GitHub Authentication plugin, GitHub Issues Plugin, Hudson Selenium plugin, GitHub Organization Folder Plugin, JavaTest report plugin, JUnit Attachments Plugin JUnit Plugin, JUnit Realtime Test Reporter Plugin, Project Health Report, Regression Report Plugin, Selenium builder plugin, Selenium HTML report, TestComplete xUnit Plugin, Test Selector, Workspaces cleanup plugin.

Si el proyecto que se va a integrar es con dependencias MAVEN entonces necesita estos paquetes:

- Maven Deployment Linker
- Maven to info Plugin
- Maven Integration Plugin
- Maven Invoker plugin
- Maven Metadata Plugin for Jenkins CI Server

Si desea lanzar la aplicación desde JENKINS es mejor desinstalar el paquete TimeStamper

2.4.3. Creación del repositorio GIT

Se creó un repositorio GIT para la aplicación PROTPANEL como indica la Figura No 4-2, necesaria para poder construir y analizar desde Jenkins, siempre trabaja con repositorios.

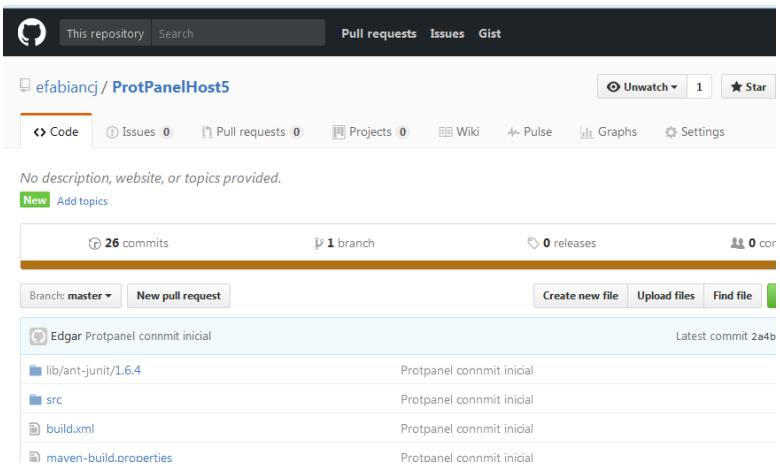


Figura No 4-2. Creación de repositorio GIT, para el código fuente
Autor: Edgar Córdova

2.4.4. Configuración del espacio de trabajo en JENKINS

Para poder compilar, construir y desplegar la aplicación desde JENKINS se requirió de la utilización de repositorios, Para la integración del proyecto PROTPANEL se escogió un repositorio GIT por adaptarse mucho mejor, ya que en Subversión se tuvieron inconvenientes con la estructura e directorios, entonces se creó un repositorio en el sitio Github.com y se pudo conectar sin inconvenientes, la configuración del repositorio se puede observar en la Figura No 5-2.

Una vez especificado el repositorio JENKINS importará el mismo a su propio repositorio interno y cada vez que se envié a construir el proyecto realizará esta importación siendo importante aclarar que si se desea un reporte de historial de pruebas es mejor no "limpiar el espacio e trabajo" ya que si se hace esto se comienza nuevamente con el ciclo de pruebas. La Figura No 6-2 muestra el espacio de trabajo de un proyecto JENKINS.

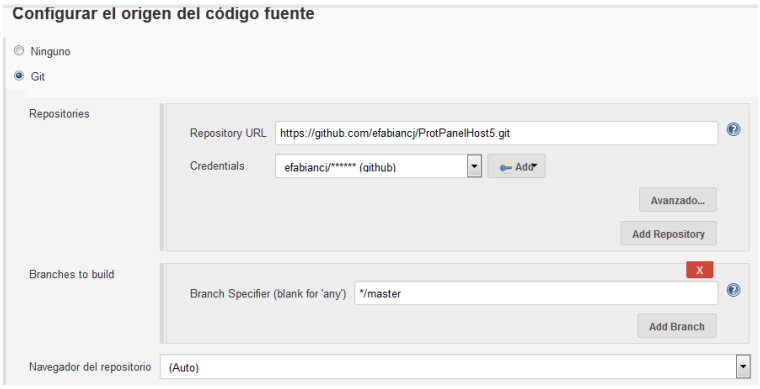


Figura No 5-2. Especificación del repositorio GIT para ejecutar pruebas del código fuente.
Autor: Edgar Córdova

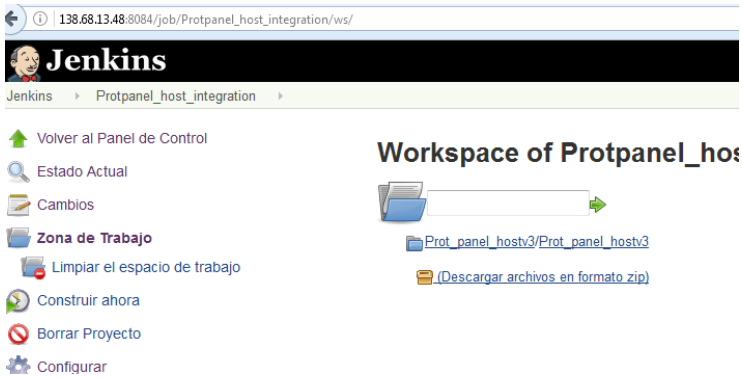


Figura No 6-2. Espacio de trabajo listo para construir y lanzar el proyecto
Autor: Edgar Córdova

En la configuración del proyecto de integración es vital especificar donde se encuentran las instalaciones de JDK, como indica la Figura No 7-2.

Figura No 7-2. Especificación de la instalación de JDK
Autor: Edgar Córdova

Se debe especificar los objetivos que pueden ser compilar, construir, desplegar. En el caso de proyectos MAVEN se requiere especificar el fichero POM raíz, como indica la Figura No 8-2.

Si se está utilizando ANT se debe especificar la versión de ANT y el destino en este caso es de prueba, como se indica en la Figura No 9-2.

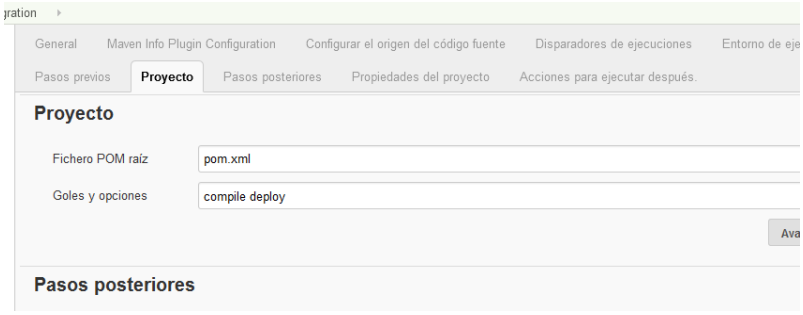


Figura No 8-2. Especificación de objetivos y archivo POM en caso de proyectos MAVEN
Autor: Edgar Córdova

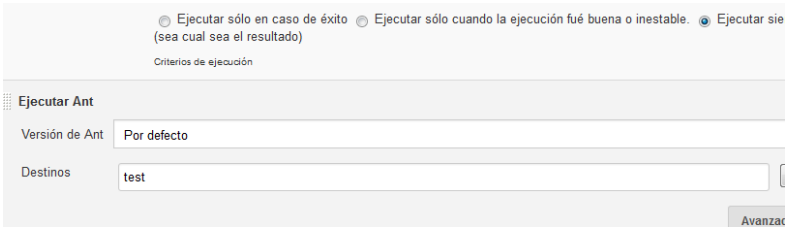


Figura No 9-2. Especificación de la versión de ANT y destino.
Autor: Edgar Córdova

Existe una sección en la configuración que indica que acciones se ejecutarán después la construcción el proyecto. Aquí se especifica el empaquetado que se desea obtener, el servidor de aplicaciones y la URL de lanzamiento como indica la Figura No 10-2.

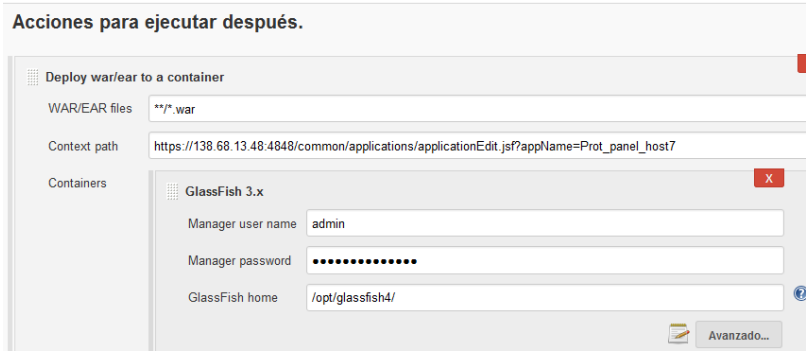


Figura No 10-2. Especificación de la versión del empaquetado, servidor de aplicaciones y U.R.L.
Autor: Edgar Córdova

2.4.5. Construcción del proyecto de integración continúa en JENKINS

La opción "Construir ahora", compiló el proyecto y como JENKINS encontró pruebas definidas en el código fuente, las ejecutó automáticamente como se aprecia en la Figura No 11-2 y generó una historia de tareas como se observa en la Figura No 12-2

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 9.116 s
[INFO] Finished at: 2017-02-03T03:10:34+00:00
[INFO] Final Memory: 24M/189M
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-deploy-plugin:2.7:deploy (default-deploy) on projec
Prot_panel_host7: Deployment failed: repository element was not specified in the POM inside distributionManagemen
element or in -DaltDeploymentRepository=id::layout::url parameter -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionException
[JENKINS] Archiving /var/lib/jenkins/jobs/Protpanel_Integration/workspace/repo_01/pom.xml to
com.kapyasoft/Prot_panel_host7/1.0-SNAPSHOT/Prot_panel_host7-1.0-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/jobs/Protpanel_Integration/workspace/repo_01/target/Prot_panel_host7-1.0-
SNAPSHOT.war to com.kapyasoft/Prot_panel_host7/1.0-SNAPSHOT/Prot_panel_host7-1.0-SNAPSHOT.war
channel stopped
ERROR: Unable to find build script at /var/lib/jenkins/jobs/Protpanel_Integration/workspace/repo_01/build.xml
Finished: FAILURE
```

Figura No 11-2. Mensaje de error por no especificar el repositorio MAVEN En el archivo POM del proyecto
Autor: Edgar Córdova

Figura No 12-2. Inicio de la construcción del proyecto
Autor: Edgar Córdova

Como se puede observar en la anterior figura también se tiene la opción de visualizar una consola con los pasos de construcción ejecutándose como indica la figura anterior. En la Figura No 13-2, se muestra un error por no encontrar el repositorio MAVEN, este error es muy común en proyectos MAVEN, la solución es modificar el archivo pom.xml del código fuente de la aplicación y especificar el repositorio.

```
<distributionManagement>
  <repository>
     <id>repo_server001</id>
     <name>repositorio_prot_panel_server001</name>
     <url>http://138.68.13.48:88/repositoriom3/</url>
  </repository>
</distributionManagement>

<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.10</version>
        <scope>test</scope>
    </dependency>
```

Figura No 13-2. Solución al error por no encontrar repositorio MAVEN.
Autor: Edgar Córdova

Ahora la construcción se ejecuta sin errores, pero, aunque se tenía creados las pruebas unitarias especificados, la herramienta JENKINS no las ejecutó, ya que JEKINS lo que hace son pruebas de integración.

Para ello MAVEN tiene previstos estas situaciones y tiene definido en su ciclo de vida las fases "pre-integration-test", "integration-test" y "post-integration-test", para ello se debe hacer uso del plugin de MAVEN Failsafe que con la configuración básica y cumpliendo con la convención de llamar a los Pruebas de integración con el sufijo "IT", Test, Testcase, etc. De esta manera se ejecutan todas las pruebas especificadas como indican la Figura No 14-2, la Figura No 15-2 y la Figura No 16-2.

Salida de consola Progreso:

```
Started by user Edgar Cordova
Building in workspace /var/lib/jenkins/jobs/Protpanel_Integration2/workspace
 > /usr/local/git/bin/git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > /usr/local/git/bin/git config remote.origin.url https://github.com/efabiancj/ProtPanelHost5.git # timeout=10
Fetching upstream changes from https://github.com/efabiancj/ProtPanelHost5.git
 > /usr/local/git/bin/git --version # timeout=10
using GIT_ASKPASS to set credentials github
 > /usr/local/git/bin/git fetch --tags --progress https://github.com/efabiancj/ProtPanelHost5.git +refs/heads/*:
/remotes/origin/*
 > /usr/local/git/bin/git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > /usr/local/git/bin/git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision fb8f60252f03ea1956f1ce49c7adc72272f45d7e (refs/remotes/origin/master)
 > /usr/local/git/bin/git config core.sparsecheckout # timeout=10
 > /usr/local/git/bin/git checkout -f fb8f60252f03ea1956f1ce49c7adc72272f45d7e
 > /usr/local/git/bin/git rev-list cbb1271993f6a1e64c1c96a31ef3c0c5f964e9c4 # timeout=10
Parsing POMs
```

Figura No 14-2. Ejecución del proceso de construcción del proyecto.
Autor: Edgar Córdova

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
parallel='none', perCoreThreadCount=true, threadCount=2, useUnlimitedThreads=false
Running com.kapyasoft.prot_panel_host5.especificaciones.Prot_panel_hostIT
contratar_plan
crear_nueva_aplicacion
validación_crear_nueva_aplicacion
crear_nueva_base_datos
listar_aplicaciones
listar_bases_datos
compilar_codigo_aplicacion
desplegar_desde_servidor_web
subir_archivos_aplicacion
modificar_aplicacion
eliminar_aplicacion
modificar_cliente
restaurar_base_datos
respaldar_base_datos
listar_dominios
visualizar_editar_codigo
reporte_consumo_recursos
Calcular_espacio_disco_utilizado
Calcular_memoria_por_aplicacion
Calcular_transferencia
crear_directorio_virtual_servidor_app
crear_script_directorios
crear_cuenta_ftp_o_repositorio
crear_script_cuentas_ftp_o_repositorios
ejecutar_script
listar_planes
autenticar_clientes
```

Figura No 15-2. Lista de pruebas generada durante la construcción.
Autor: Edgar Córdova

```
Results :

Failed tests:
  Prot_panel_hostIT.testContratar_plan:180 The test case is a prototype.
  Prot_panel_hostIT.testCrear_nueva_aplicacion:194 The test case is a prototype.
  Prot_panel_hostIT.testValidación_crear_nueva_aplicacion:208 The test case is a prototype.
  Prot_panel_hostIT.testCrear_nueva_base_datos:222 The test case is a prototype.
  Prot_panel_hostIT.testListar_aplicaciones:236 The test case is a prototype.
  Prot_panel_hostIT.testListar_bases_datos:250 The test case is a prototype.
  Prot_panel_hostIT.testCompilar_codigo_aplicacion:264 The test case is a prototype.
  Prot_panel_hostIT.testDesplegar_desde_servidor_web:278 The test case is a prototype.
  Prot_panel_hostIT.testSubir_archivos_aplicacion:292 The test case is a prototype.
  Prot_panel_hostIT.testModificar_aplicacion:306 The test case is a prototype.
  Prot_panel_hostIT.testEliminar_aplicacion:320 The test case is a prototype.
  Prot_panel_hostIT.testModificar_cliente:334 The test case is a prototype.
  Prot_panel_hostIT.testRestaurar_base_datos:348 The test case is a prototype.
  Prot_panel_hostIT.testRespaldar_base_datos:360 expected:<> but was:<null>
  Prot_panel_hostIT.testListar_dominios:374 expected:<> but was:<null>
  Prot_panel_hostIT.testVisualizar_editar_codigo:388 expected:<> but was:<null>
  Prot_panel_hostIT.testReporte_consumo_recursos:402 expected:<> but was:<null>
  Prot_panel_hostIT.testCalcular_espacio_disco_utilizado:418 The test case is a prototype.
  Prot_panel_hostIT.testCalcular_memoria_por_aplicacion:432 The test case is a prototype.
  Prot_panel_hostIT.testCalcular_transferencia:446 The test case is a prototype.
  Prot_panel_hostIT.testCrear_directorio_virtual_servidor_app:79 The test case is a prototype.
  Prot_panel_hostIT.testCrear_script_directorios:93 The test case is a prototype.
  Prot_panel_hostIT.testCrear_cuenta_ftp_o_repositorio:107 The test case is a prototype.
  Prot_panel_hostIT.testCrear_script_cuentas_ftp_o_repositorios:121 The test case is a prototype.
  Prot_panel_hostIT.testEjecutar_script:136 The test case is a prototype.
  Prot_panel_hostIT.testListar_planes:150 The test case is a prototype.
  Prot_panel_hostIT.testAutenticar_clientes:166 The test case is a prototype.
  Prot_panel_hostIT.testRegistro_cliente:51 The test case is a prototype.
  Prot_panel_hostIT.testValidacion_registro:65 The test case is a prototype.
  Prot_panel_hostIT.testReporte_pagos_mensuales:458 expected:<> but was:<null>
```

Figura No 16-2. Listado de pruebas fallidas durante la construcción.
Autor: Edgar Córdova

En la Figura No 17-2 y la Figura No 18-2, a continuación, se observa los resultados de la construcción del proyecto, aquí se importa primero el código fuente desde el repositorio GIT y se compila.

```
get-deps:
      [get] Getting: https://repo.maven.apache.org/maven2/junit/junit/4.10/junit-4.10.jar
      [get] To: /var/lib/jenkins/.m2/repository/junit/junit/4.10/junit-4.10.jar
      [get] Getting: https://repo.maven.apache.org/maven2/org/hamcrest/hamcrest-core/1.1/hamcrest-core-1.1.jar
      [get] To: /var/lib/jenkins/.m2/repository/org/hamcrest/hamcrest-core/1.1/hamcrest-core-1.1.jar
      [get] Getting: https://repo.maven.apache.org/maven2/ant/ant/1.5/ant-1.5.jar
      [get] To: /var/lib/jenkins/.m2/repository/ant/ant/1.5/ant-1.5.jar
      [get] Getting: https://repo.maven.apache.org/maven2/ant/ant-junit/1.6.5/ant-junit-1.6.5.jar
      [get] To: /var/lib/jenkins/.m2/repository/ant/ant-junit/1.6.5/ant-junit-1.6.5.jar
      [get] Getting: https://repo.maven.apache.org/maven2/javax/javaee-web-api/7.0/javaee-web-api-7.0.jar
      [get] To: /var/lib/jenkins/.m2/repository/javax/javaee-web-api/7.0/javaee-web-api-7.0.jar

compile:
    [javac] /var/lib/jenkins/jobs/Protpanel_Integration2/workspace/maven-build.xml:79: warning: 'includeantrunt
not set, defaulting to build.sysclasspath=last; set to false for repeatable builds

compile-tests:
    [javac] /var/lib/jenkins/jobs/Protpanel_Integration2/workspace/maven-build.xml:104: warning: 'includeantrun
was not set, defaulting to build.sysclasspath=last; set to false for repeatable builds
```

Figura No 17-2. Finalización de la construcción.
Autor: Edgar Córdova

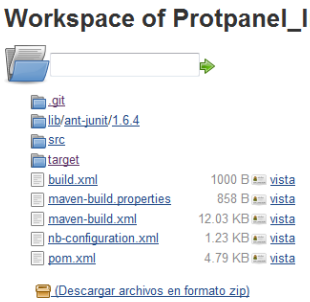


Figura No 18-2. Resultados obtenidos de la construcción.
Autor: Edgar Córdova

Dentro de la carpeta target, se descarga el empaquetado de archivo web (Web Archive o WAR) que luego servirá para desplegar la aplicación en el servidor de aplicaciones como se muestra en la figura la Figura No 19-2.

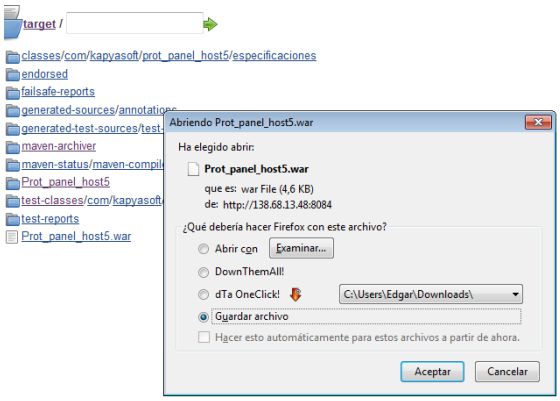


Figura No 19-2. Descarga el empaquetado WAR
Autor: Edgar Córdova

Se ha creado una aplicación solo con las especificaciones de las pruebas funcionales luego por la disciplina Test-Driven Development, se continua escribiendo más pruebas o especificaciones conforme se escribe el código para resolver y satisfacer las pruebas, también se generan automáticamente algunas clases ya que las especificaciones iniciales hacen referencia a clases que no existen y aunque la disciplina exige que no se cree más código que el necesario se asume que todas las clases resultado del diseño inicial UML y serán empleadas para construir la aplicación.

Un primer reporte de pruebas de JENKINS tendría la forma de la la Figura No 20-2.

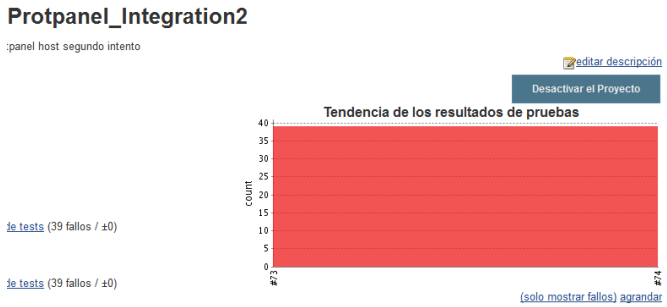


Figura No 20-2. Primer reporte de pruebas de JENKINS
Autor: Edgar Córdova

Se construye las clases con el mínimo código posible y de este modo tenemos más especificaciones que en un inicio todas fallarán porque no se ha escrito más que un código muy básico para que el proyecto se compile y sea construido por la herramienta de integración continua.

2.4.7. Preparación de pruebas unitarias en el proyecto

2.4.7.1. Instalación de librerías Selenium

Para poder utilizar las librerías de Selenium se invocó la dependencia desde MAVEN, sin embargo, también es posible utilizar Selenium en proyectos web JAVA normales sin dependencias. Para ello se debe ubicar en el directorio del repositorio .m2 y copiar los archivos con formato JAR, para posteriormente agregarlos al proyecto como librerías.

```
<dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-java</artifactId>
      <version>3.0.1</version>
      <scope>test</scope>
</dependency>
```

Para escribir las pruebas se hace uso de la herramienta *Firebug* que permite inspeccionar el código y saber cuál es el nombre del componente que se quiere probar en este caso se prueba el campo email del usuario que se registra, el componente se llama “formId:txtEmail”, como indica la Figura 21-2.

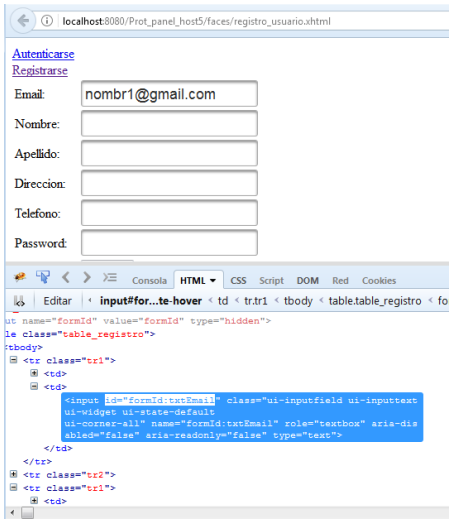


Figura No 21-2. Primera versión de la interface de registro de nuevo usuario
Autor: Edgar Córdova

Para que Selenium realice las pruebas automáticas de las interfaces, es necesario llamar a la clase WebDriver, con un amplio set de métodos para manejar las interfaces web, etiquetas HTML y URLs, como se muestra en la Figura No 22-2.

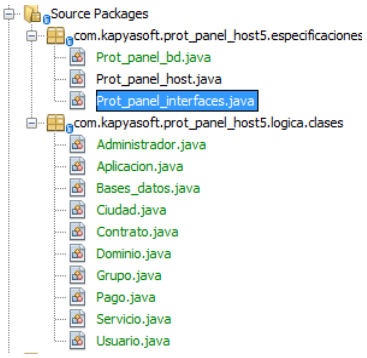


Figura No 22-2. Primera versión de especificaciones de pruebas.
Autor: Edgar Córdova

Con ayuda del entorno de desarrollo integrado (I.D.E.), NETBEANS, se generaron las pruebas para cada clase con especificaciones como indica la Figura No 23-2, a continuación:

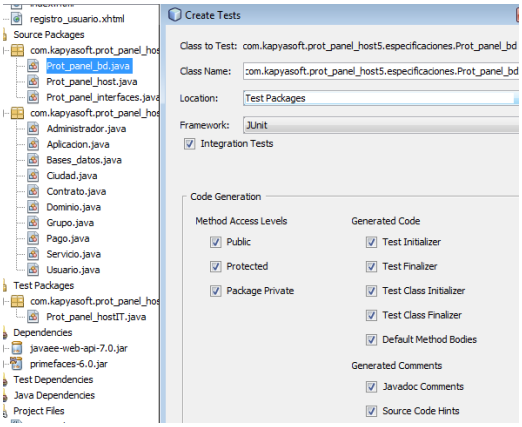


Figura No 23-2. Generación de pruebas desde clases de especificación de pruebas.
Autor: Edgar Córdova

Generando todas las pruebas y ejecutándolas la aplicación en un inicio tendrá la apariencia de la siguiente figura, como ya se ha especificado en el archivo pom.xml la dependencia al plugin de *failsafe* se espera que JENKINS las ejecute automáticamente como muestra la Figura No 24-2.

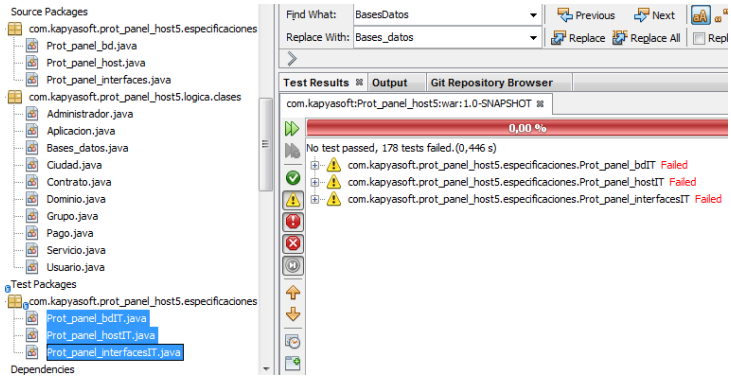


Figura No 24-2. Ejecución y fallo de todas las pruebas de la especificación
Autor: Edgar Córdova

Se envía las modificaciones al repositorio GIT, realizando una instrucción *commit* desde el entorno de desarrollo integrado (I.D.E.), como se indica en la Figura No 25-2.

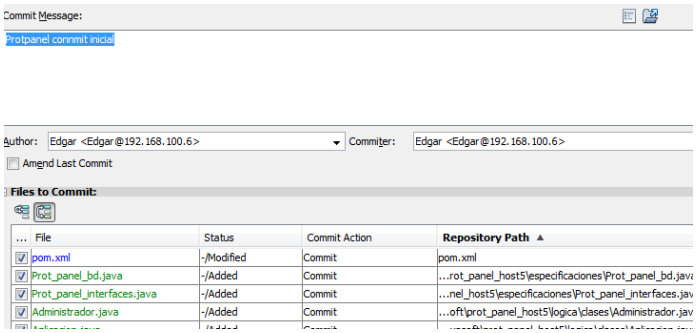


Figura No 25-2. Envío de las modificaciones al repositorio GIT
Autor: Edgar Córdova

Luego se debe hacer clic en enchufar (*push*), para conectar definitivamente todo el código a la raíz maestro de GIT como indica la Figura No 26-2.

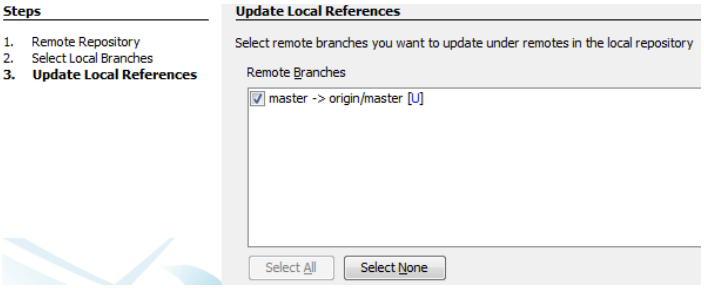


Figura No 26-2. Subir definitivamente todo el código a la raíz maestro de GIT.
Autor: Edgar Córdova

Se reinicia la aplicación y se vuelve a construir una vez más, el resultado fue el de abajo en la Figura No 27-2.

```
Prot_panel_hostIT.testCalcular_transferencia:446 The test case is a prototype.
Prot_panel_hostIT.testReporte_pagos_mensuales:458 expected:<> but was:<null>
Prot_panel_hostIT.testGestion_administrador:490 The test case is a prototype.
Prot_panel_hostIT.testCrear_nuevo_administrador:504 The test case is a prototype.
Prot_panel_hostIT.testValidacion_Crear_nuevo_administrador:518 The test case is a prototype.
Prot_panel_hostIT.testModificacion_administrador:532 The test case is a prototype.
Prot_panel_hostIT.testValidacion_Modificacion_administrador:546 The test case is a prototype
Prot_panel_hostIT.testListar_administradores:560 The test case is a prototype.
Prot_panel_hostIT.testReporte_usuarios:572 expected:<> but was:<null>
Prot_panel_hostIT.testReporte_ventas:586 expected:<> but was:<null>

Tests run: 178, Failures: 178, Errors: 0, Skipped: 0
```

Figura No 27-2. Aumento de código, pruebas y fallos
Autor: Edgar Córdova

2.4.8. Código de especificación de pruebas inicial

```
public class Prot_panel_host
{
  // ----------------------------------------------------------------------------
  //  Especificaciones para actor visitante
  //-----------------------------------------------------------------------------
    public int registro_cliente(Usuario usuario) throws Exception
    {
       //boolean res = FUsuarios.ingresar(usuario);
       return 0;
    }

    public int validacion_registro()
    {
       // int res = FUsuarios.repetidos();
        return 0;
    }

    public int crear_directorio_virtual_servidor_app(Usuario usuario)
    {
        crear_script_directorios(usuario);
        return 0;
    }


    public int crear_cuenta_ftp_o_repositorio(Usuario usuario)
    {
        crear_script_cuentas_ftp_o_repositorios(usuario);
        return 0;
    }

   //-----------------------------------------------------------------------------
   // ----------------------------------------------------------------------------
  //  Especificaciones para actor usuario - cliente
  //-----------------------------------------------------------------------------
    public boolean autenticar_clientes(String nombre, String clave)
    {
        return false;
```

```
}


public int contratar_plan()
{
    return 0;
}

public int crear_nueva_aplicacion(Aplicacion bd)
{
    return 0;

}
public int validación_crear_nueva_aplicacion()
{
    return 0;
}
public int crear_nueva_base_datos(Bases_datos bd)
{
    return 0;
}

public ArrayList<Aplicacion>  listar_aplicaciones(Usuario usuario)
{
    return null;
}

public ArrayList<Bases_datos> listar_bases_datos(Usuario usuario)
{
    return null;
}

public int compilar_codigo_aplicacion(Aplicacion aplicación,Usuario usuario)
{
    return 0;
}

public int desplegar_desde_servidor_web(Aplicacion aplicacion)
{
    return 0;
}

public int subir_archivos_aplicacion(Usuario usuario, String nombre_aplicacion)
{
   return 0;
}

public int modificar_aplicacion(Usuario usuario, String nombre_aplicacion)
{
   return 0;
}

public int eliminar_aplicacion(Aplicacion aplicación)
{
    return 0;
}

public int  modificar_cliente(Usuario usuario)
{
    return 0;
}

public int restaurar_base_datos(Bases_datos bd)
{
    return 0;
}

public String respaldar_base_datos(Bases_datos bd)
{
    return null;
}


public String  listar_dominios(Usuario usuario)
{
    return null;
}
```

```
    public String  visualizar_editar_codigo(Usuario usuario, Aplicacion aplicacion,
String archivo)
    {
        return null;
    }

    public String  reporte_consumo_recursos(Usuario usuario)
    {
        return null;
    }

    public int Calcular_espacio_disco_utilizado()
    {
        return 0;
    }

    public int Calcular_memoria_por_aplicacion()
    {
        return 0;
    }

    public int Calcular_transferencia()
    {
        return 0;
    }


    public String  reporte_pagos_mensuales(Usuario usuario)
    {
        return null;
    }
//-----------------------------------------------------------------------------------
// -----------------------------------------------------------------------------------
//  Especificaciones para actor administrador
//-----------------------------------------------------------------------------------

    public boolean autenticar_administradores(String nombre, String clave)
    {
        return false;
    }

    public int Gestion_administrador()
    {
        return 0;
    }

    public int Crear_nuevo_administrador()
    {
        return 0;
    }

    public int Validacion_Crear_nuevo_administrador()
    {
        return 0;
    }

    public int Modificacion_administrador()
    {
        return 0;
    }

    public int validacion_Modificacion_administrador()
    {
        return 0;
    }

    public ArrayList<Administrador> Listar_administradores()
    {
        return null;
    }

    public String  reporte_usuarios()
    {
        return null;
    }
```

```
    public String  reporte_ventas()
    {
        return null;
    }
}
```

2.4.9. Generación automática de pruebas

Las pruebas para la primera especificación, deben ser necesariamente pruebas de integración para que puedan ser reconocidas por la herramienta de integración continua JENKINS. Como resultado del proceso anterior se generan las pruebas para los métodos especificados en esa clase. En esta sección de código se cita solamente la prueba para "autenticar_clientes" para una mejor explicación.

```
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 *
 * @author Edgar
 */
public class Prot_panel_hostIT {

    public Prot_panel_hostIT() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }

/**
     * Test of autenticar_clientes method, of class Prot_panel_host.
     */
    @Test
    public void testAutenticar_clientes() {
        System.out.println("autenticar_clientes");
        String nombre = "";
        String clave = "";
        Prot_panel_host instance = new Prot_panel_host();
        boolean expResult = false;
    //    boolean result = instance.autenticar_clientes(nombre, clave);
        assertEquals(expResult, expResult);
        // TODO review the generated test code and remove the default call to fail.
       // fail("The test case is a prototype.");
    }
}
```

2.4.10. Escritura del código para pasar una prueba unitaria de la capa de Lógica de negocio

Debido a que el código es para cada prueba es extenso solo tomaremos un requerimiento como ejemplo, como indica fragmento de código a continuación.

```
@Test
    public void testObtener_Todos_Aplicacions() throws Exception {
        System.out.println("obtener_Todos_Aplicacions");
        ArrayList<Aplicacion> expResult = new ArrayList<Aplicacion>();
        Usuario usuario=new Usuario();
        usuario.setId(1);
        Aplicacion  a1 = new
Aplicacion(1,"aplic1",usuario,"E:\\Temp\\Proyectos\\Modulos_protpanel\\web\\usuarios_hos
t\\nmbre1a1\\aplic1\\","ninguno");
        Aplicacion  a2 = new
Aplicacion(2,"aplic2",usuario,"E:\\Temp\\Proyectos\\Modulos_protpanel\\web\\usuarios_hos
t\\nmbre1a1\\aplic2\\","ninguno");
        Aplicacion  a3 = new
Aplicacion(3,"aplic3",usuario,"E:\\Temp\\Proyectos\\Modulos_protpanel\\web\\usuarios_hos
t\\nmbre1a1\\aplic3\\","ninguno");
        Aplicacion  a4 = new
Aplicacion(4,"aplic4",usuario,"E:\\Temp\\Proyectos\\Modulos_protpanel\\web\\usuarios_hos
t\\nmbre1a1\\aplic4\\","ninguno");
        expResult.add(a1);
        expResult.add(a2);
        expResult.add(a3);
        expResult.add(a4);

        ArrayList<Aplicacion> result = FAplicacions.obtener_Todos_Aplicacions();
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
        //sucess("The test case is a prototype.");
    }
```

2.4.11. Escritura del código para pasar una prueba de integración en la capa de Presentación usando Selenium.

```
/**
 * Test of boton_logueo method, of class Prot_panel_interfaces.
 */

@Test
    public void boton_ingreso_nuevo_usuario_si_datos_correctos()
    {
        System.setProperty("webdriver.gecko.driver", Global.geckodriver);
        driver = new FirefoxDriver();
        String url = Global.url_app + "registro_usuario" + Global.ext;
        driver.get(url);

        //Now you can Initialize marionette driver to launch firefox
        DesiredCapabilities capabilities = DesiredCapabilities.firefox();
        capabilities.setCapability("marionette", true);

        WebElement email = driver.findElement(By.id("txtEmail"));
        WebElement nombre = driver.findElement(By.id("txtNombre"));
        WebElement apell = driver.findElement(By.id("txtApellido"));
        WebElement direcc = driver.findElement(By.id("txtDireccion"));
        WebElement telf = driver.findElement(By.id("txtTelefono"));
        WebElement clave = driver.findElement(By.id("txtClave"));

        WebElement submit_button = driver.findElement(By.id("btnRegistrar"));
        email.sendKeys("efabiancj@hotmail.com");
        nombre.sendKeys("Edgar");
        apell.sendKeys("Cordova");
        direcc.sendKeys("Sn Jse Tapi");
        telf.sendKeys("032602291");
        clave.sendKeys("123456");

        submit_button.click();
        WebElement elemento_mensaje = null;
        String mensaje = "";
        try
```

```
        {
           elemento_mensaje = driver.findElement(By.id("lblmensaje"));
           mensaje = elemento_mensaje.getText() ;
           System.out.println(mensaje);
          // Assert.assertTrue(mensaje.equalsIgnoreCase("Se ha ingresado un nuevo
usuario."));
           Assert.assertEquals(mensaje, "Se ha ingresado un nuevo usuario.");
        }
        catch(Exception ex)
        {
            fail("Error - No se ha ingresado el usuario."+ex.getMessage());
        }
        driver.close();
    }
```

2.5. Desarrollo de la aplicación

Como la técnica que se utiliza para desarrollar la aplicación es T.D.D., lo apropiado es utilizar una metodología ágil como X.P. Sin embargo hay que considerar que esta se debe adaptar o agregar la especificación y ejecución de pruebas al inicio de la planificación como exige T.D.D. Para ello es necesario especificar en un archivo .java las especificaciones de la aplicación cuyos métodos se van a desarrollar generando automáticamente a partir de estos métodos (vacíos), un código base o plantilla para ir ejecutando las pruebas y construyendo el código en base a los requerimientos solicitados.

2.5.1. Primera Fase: Planificación del proyecto.

2.5.1.1. Historias de usuario.

Las historias de usuario especifican los requisitos funcionales de la aplicación. Cada historia de usuario limita el alcance de cada requerimiento. Las tarjetas se adjuntan en el anexo A.

2.5.1.2. Diseño de Lenguaje de Modelado Unificado (U.M.L.)

2.5.1.2.1. Diagramas de casos de uso

Registro de usuario y listado de planes

Se ha identificado 3 actores un Visitante, quien podrá ver el listado de planes del servicio y decidir si desea registrase y contratar un plan. Un Usuario (cliente) quien se va a beneficiar del servicio que va a proporcionar la aplicación y que a su vez puede contratar más planes y un Administrador del sistema, quien gestionará a otros administradores y accederá a sus respectivos reportes. Los casos de uso o procesos con los que va a interactuar el Visitante se detallan en la Figura No 28-2.

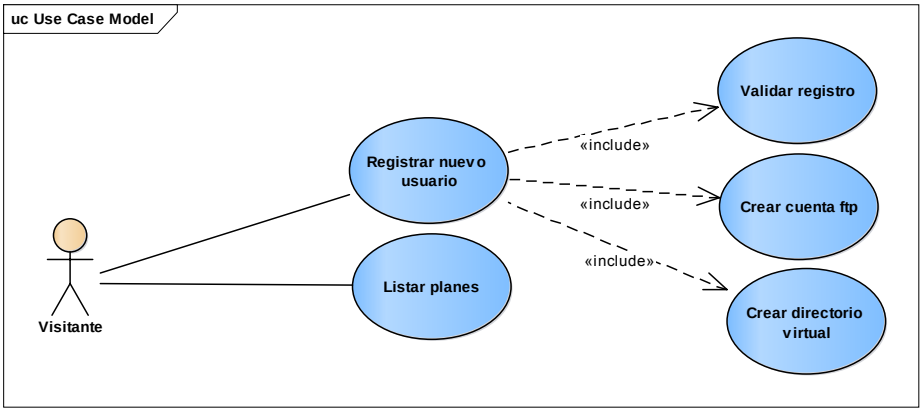


Figura No 28-2. Diagrama de Casos de uso No 1. Registro y listado de planes
Realizado por: Edgar Córdova, 2017

Reporte de Consumo y Pagos

El actor Usuario por su parte accede una vez autenticado al panel de control propiamente dicho en donde podrá acceder a la gestión de sus aplicaciones y a los reportes de consumo y pagos realizados o pendientes. El Usuario (cliente), es quien realiza la mayor cantidad de procesos, es para quien va diseñado el sistema. Los casos de uso o procesos con los que va a interactuar el Usuario en lo que se refiere a reportes se detallan en la Figura No 29-2.

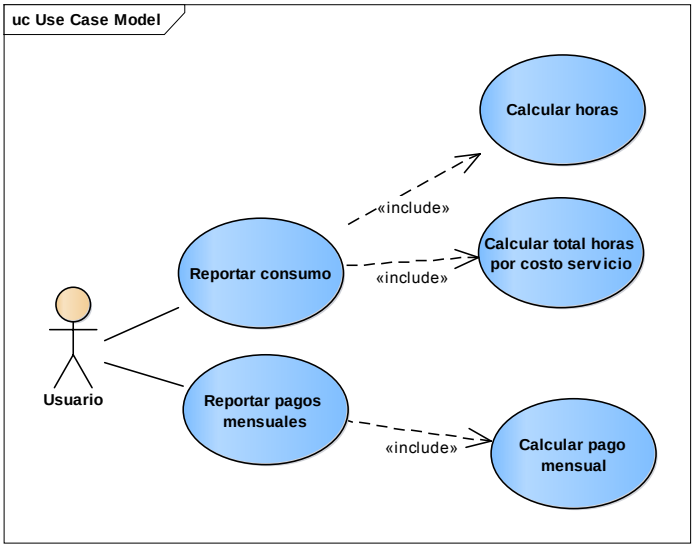


Figura No 29-2. Diagrama de casos de uso No 2
Realizado por: Edgar Córdova, 2017

Autenticación, Contrato, Crear aplicación Visualizar aplicaciones y bases de datos

El actor Usuario para poder acceder al panel debe estar autenticado después de como visitante se haya registrado, entonces accede a la página principal del panel, donde podrá contratar un nuevo plan, crear nuevas aplicaciones y subir sus respectivos archivos (ZIP o WAR), así como

ver las aplicaciones ya creadas y desplegadas. Estos casos de uso se visualizan en la Figura No 30-2.

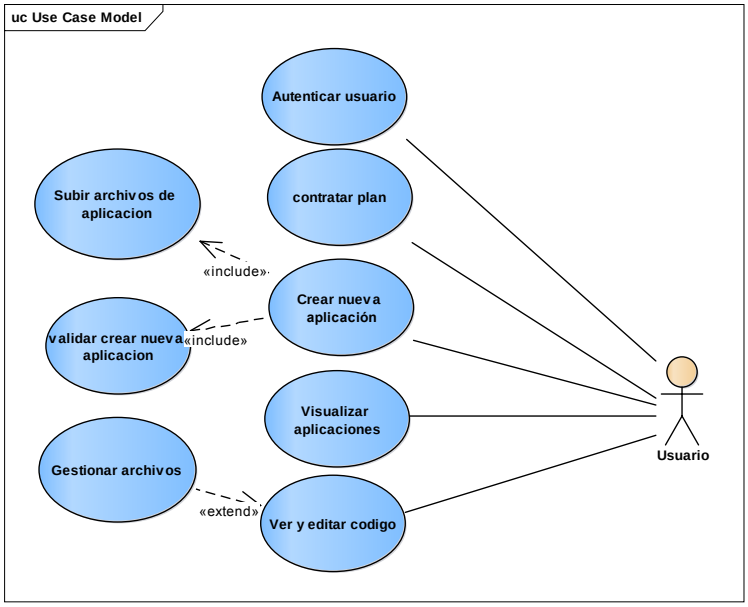


Figura No 30-2. Diagrama de casos de uso No 3
Realizado por: Edgar Córdova, 2017

Crear nueva Base de datos, Compilar código, ver y editar código

El Usuario también puede subir bases de datos y restaurarlas, aunque no podrá modificarlas mediante un gestor gráfico, pero si puede restaurarlas mediante archivos de respaldo que se pueden subir mediante un componente para subir archivos. En lo que se refiere a las aplicaciones si el código es subido mediante un archivo ZIP, este se guarda en un repositorio para luego compilarse por orden del Usuario y desplegarse en el servidor de aplicaciones Glassfish. La aplicación también debe permitir modificar archivos que contenga texto como TXT, HTML, XHTML, etc. Para lo cual es necesario algunas utilidades para el manejo de archivos, estos casos de uso de detallan en la Figura No 31-2

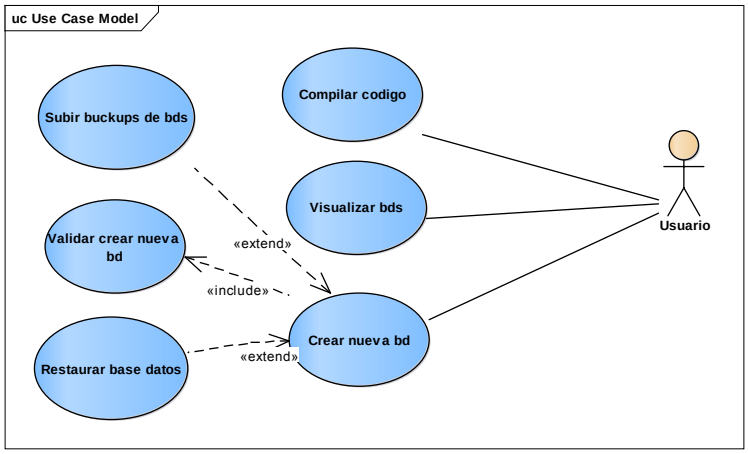


Figura No 31-2. Diagrama de Casos de uso No 4
Realizado por: Edgar Córdova, 2017

Gestión de administradores, Reporte de Ventas y Reporte de usuarios

Finalmente, el Administrador debe poder gestionar otros Administradores y listarlos. También acceder al reporte de planes contratados (ventas) y tener un listado de todos los usuarios en cierto intervalo de tiempo. Estos requerimientos se observan en la Figura No 32-2.

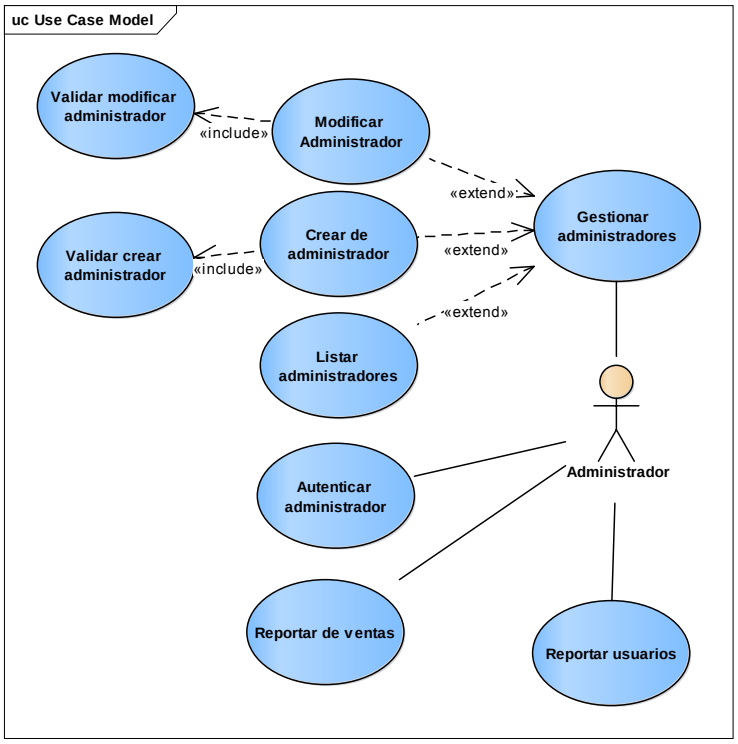


Figura No 32-2. Diagrama de Casos de uso No 5.
Realizado por: Edgar Córdova, 2017

2.5.1.2.2. Diagrama de clases

EL diagrama de clases consta de 10 clases principales que posteriormente se convertirán en el diagrama de bases de datos que servirán para re1solver los casos de uso. La clase usuario representa a los clientes que se registran para usar el panel de control, la clase Aplicación define a las aplicaciones web que el usuario desee crear y desplegar, esta clase pose atributos que guardan características importantes para una aplicación como su nombre y directorios de archivos de código fuente y empaquetados. La clase bases datos, es parecida a la clase aplicación menos el campo de directorio de despliegue de empaquetado WAR. La clase Administrador muy parecida a la clase Usuario con la variante del alias (nickname) y la ciudad. La clase dominios para relacionar las aplicaciones web con nombres de dominios. La clase ciudad para que el Usuario seleccione fácilmente sin necesidad de escribir. Una clase de Servicio para mostrar al usuario los servicios de hosting y precios. Una clase Pago para registrar todos los pagos realizados por un usuario. La clase Consumo para registrar el consumo del

servicio que realiza el usuario cada vez que crea una aplicación o base de datos. La representación gráfica de la base de datos se muestra en la Figura No 33-2.

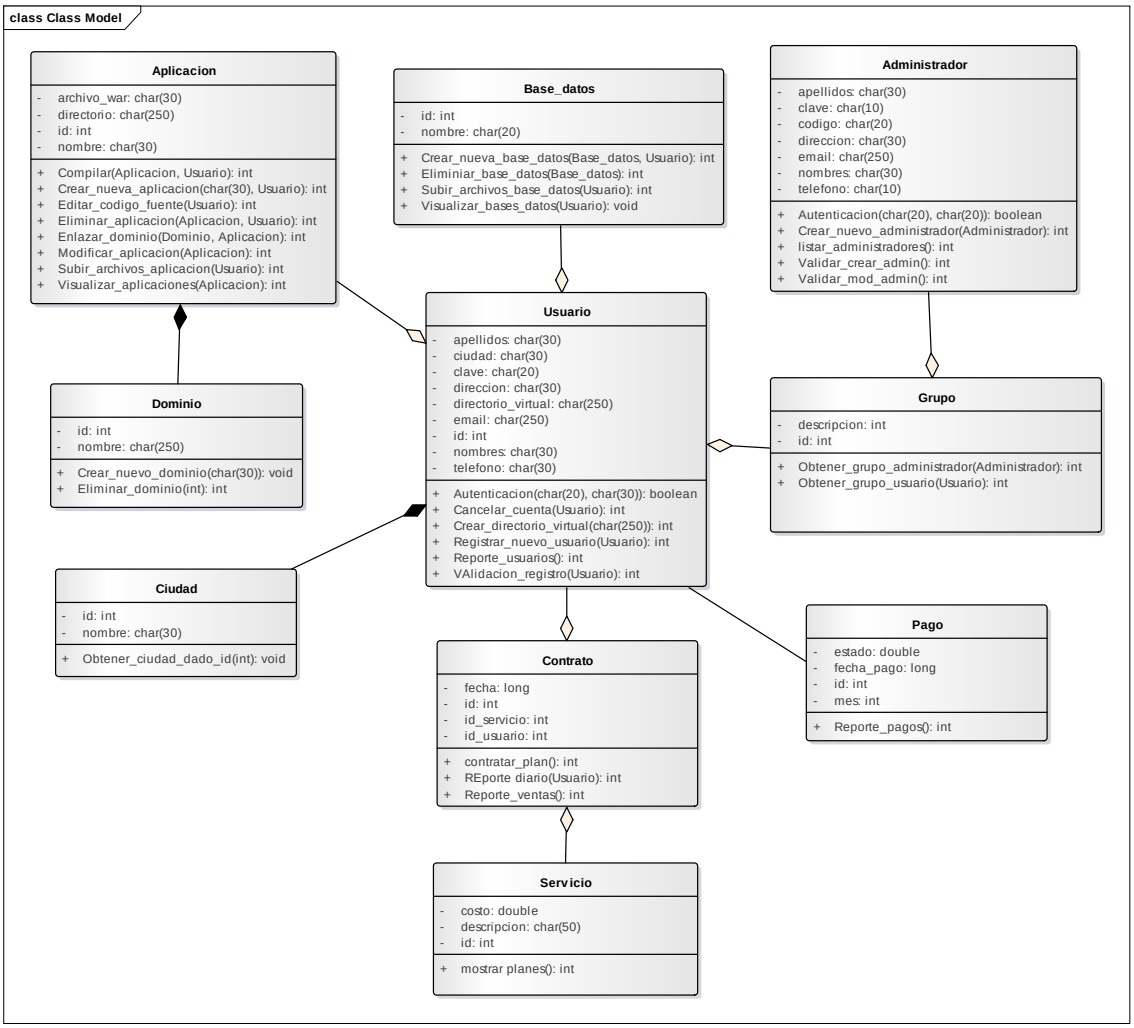


Figura No 33-2. Diagrama de clases
Realizado por: Edgar Córdova, 2017

2.5.1.2.3. Diagrama de objetos

Este diagrama muestra cómo interactúan las clases instanciadas y como fluyen los datos entre objetos. La mayoría de clases se relacionan con la clase Usuario. Primero un usuario se autentica y envía su información de inicio de sesión a la aplicación, el objeto aplicación si es instanciado requiere el número de usuario para realizar sus operaciones al igual que el objeto bases datos y pago. El objeto usuario también requiere de un grupo para estar bien instanciado al igual que un objeto ciudad, como muestra la Figura 34-2.

2.5.1.2.4. Diagrama de base de datos

Se tienen 10 tablas donde la tabla principal es usuario, quien es el que realizará la mayoría de tareas, las 3 tablas siguientes más importantes son aplicación, base datos y dominio y

posteriormente están administrador, ciudad, grupo, contrato, servicio y pagos, como se visualiza en la Figura No 35-2.

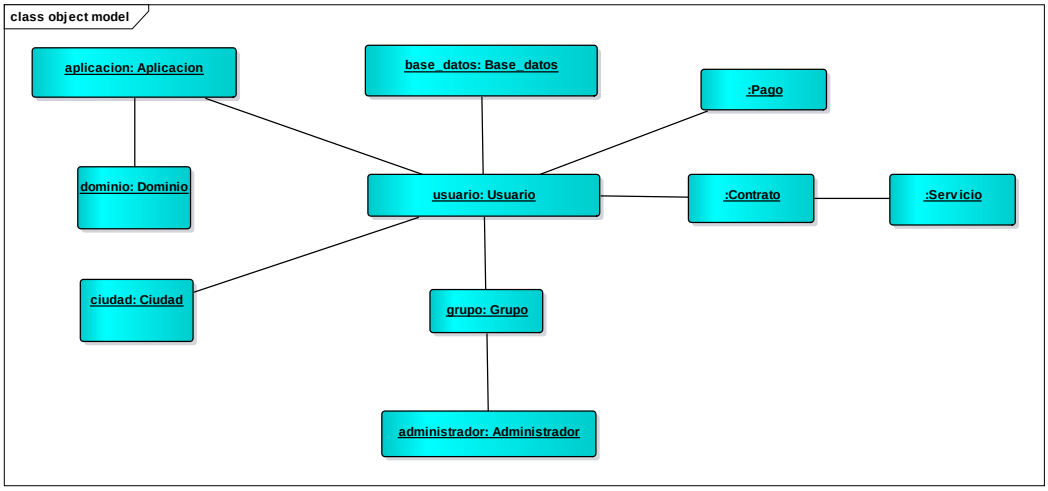


Figura No 34-2. Diagrama de objetos
Realizado por: Edgar Córdova, 2017

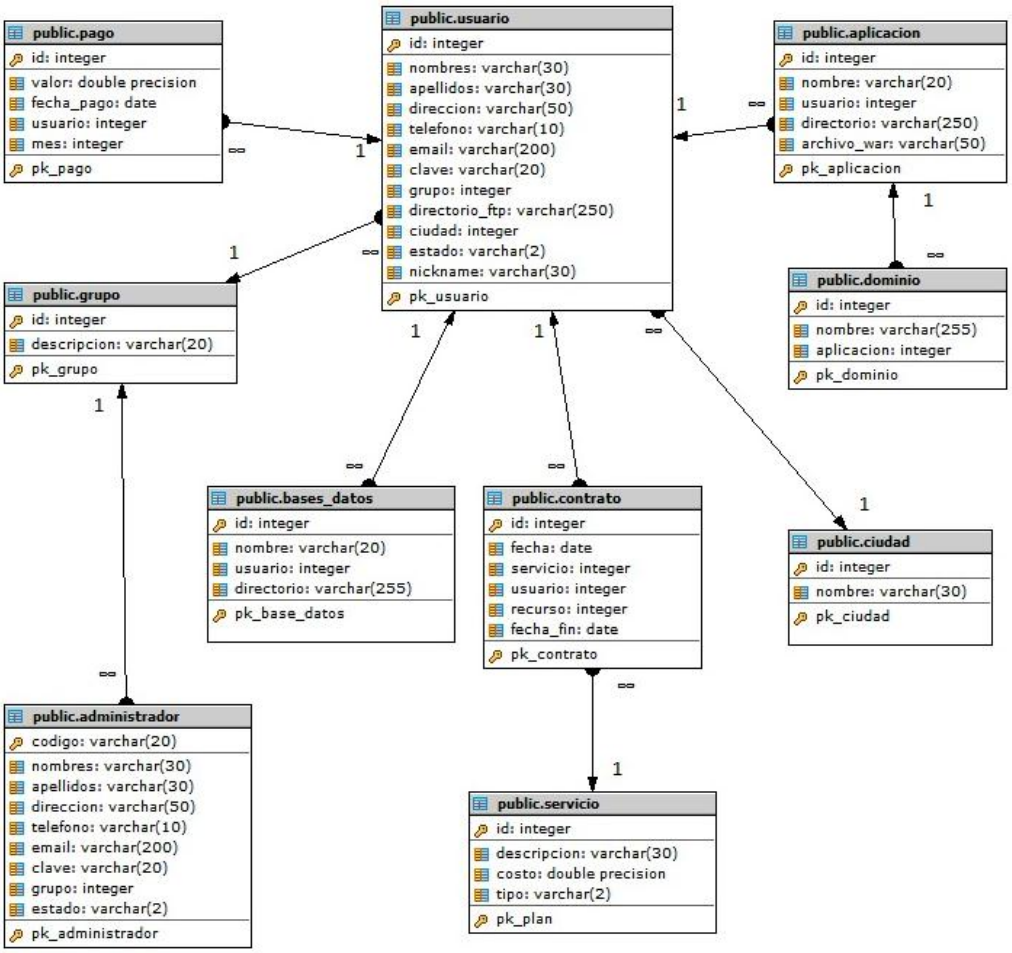


Figura No 35-2. Diagrama de bases de datos
Realizado por: Edgar Córdova, 2017

2.5.1.2.5. Diagramas de secuencia y colaboración

Crear nueva aplicación

El diagrama de secuencia del caso de uso crear nueva aplicación comienza con él la solicitud del usuario, entonces el programa despliega un formulario solicitando nombre de la aplicación (los demás atributos asignan internamente), luego el usuario ingresa el nombre y da clic en aceptar entonces creará una nueva aplicación. EL programa requiere que el usuario suba el archivo empaquetado con la extensión WAR o el código fuente en formato ZIP para lo cual la aplicación lanza un formulario para subir el mencionado archivo, esto de aprecia de forma gráfica en la Figura No 36-2.

El diagrama de colaboración de la Figura No 37-2, indica los mensajes que van y vienen entre objetos relacionados con la creación de una nueva aplicación. Al igual que el diagrama de secuencia comienza con la solicitud del usuario para crear una nueva aplicación luego el programa proporciona las interfaces necesarias para cumplir este objetivo. El formulario interactúa con el objeto aplicación para obtener los atributos y métodos necesarios para acceder a la base de datos e ingresar un nuevo registro.

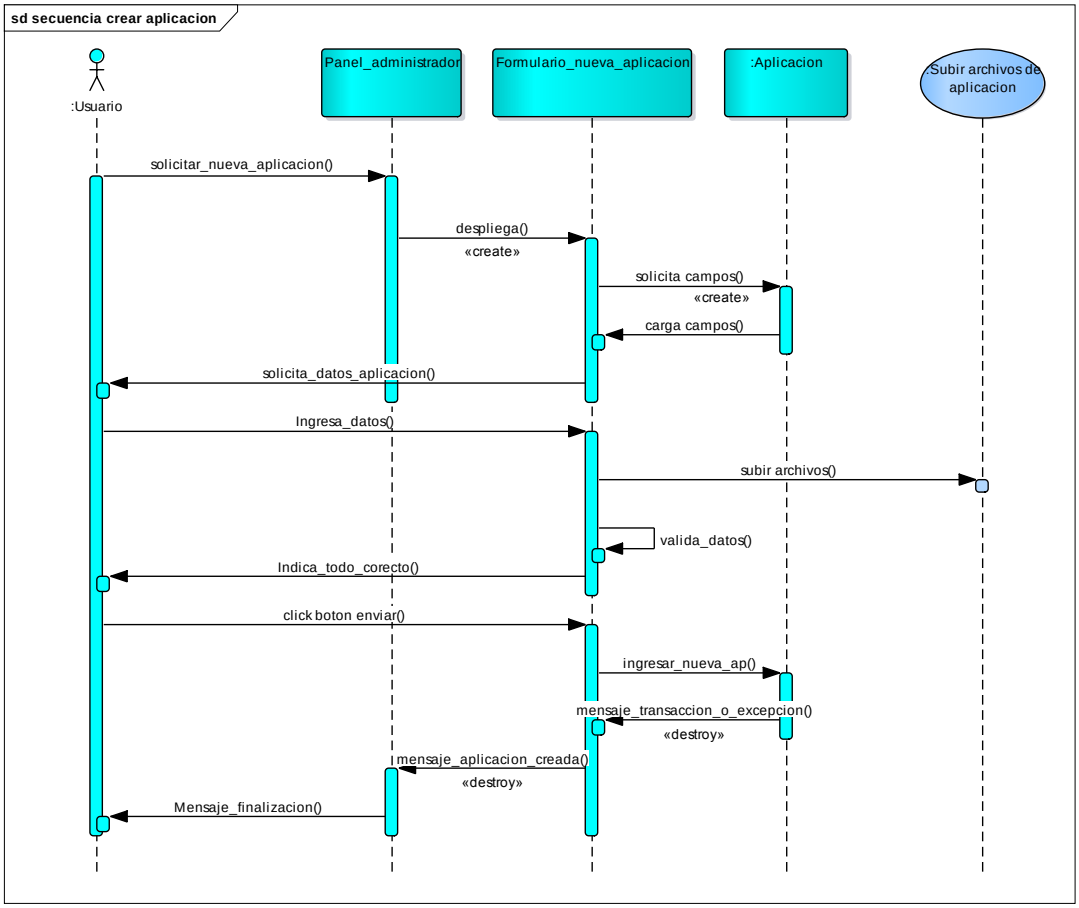


Figura No 36-2. Diagrama de secuencia "Crear nueva aplicación"
Realizado por: Edgar Córdova, 2017

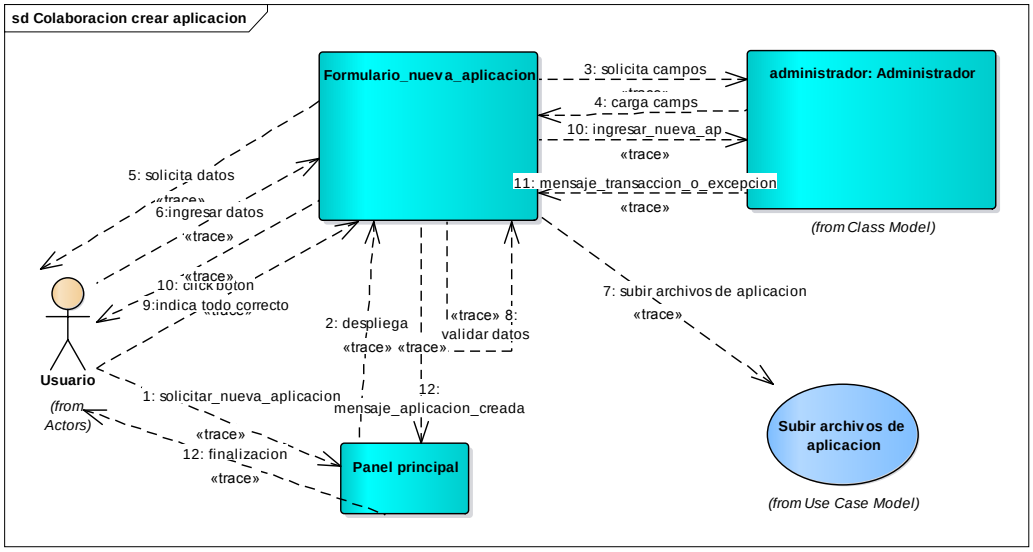


Figura No 37-2. Diagrama de colaboración para crear una nueva aplicación
Realizado por: Edgar Córdova, 2017

Visualizar aplicaciones

El diagrama de secuencia de este requerimiento es sencillo simplemente requiere una solicitud del usuario y el sistema visualizará una interfaz con todas las aplicaciones del ese Usuario. Para ello accede a la clase Aplicación que tiene el método para obtener aplicaciones por usuario. La Figura 38-2, muestra esta secuencia.

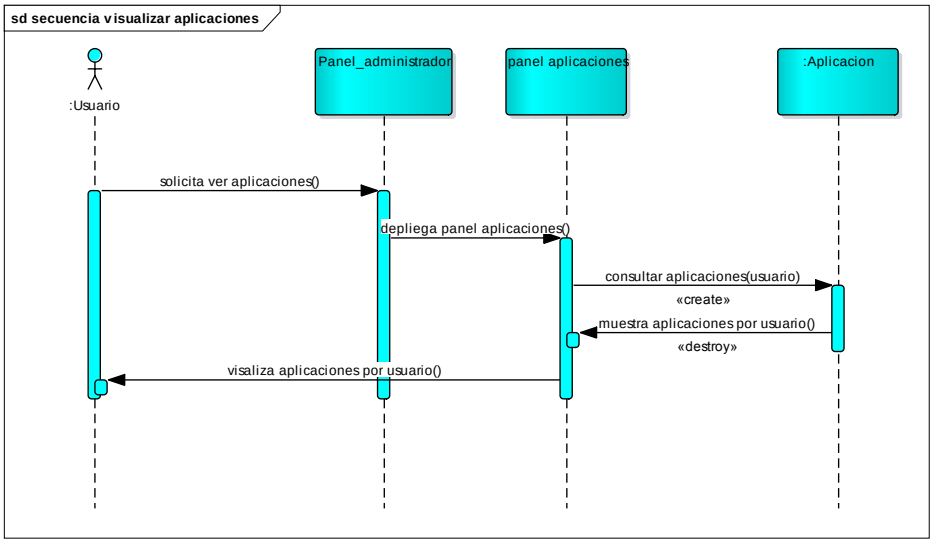


Figura No 38-2. Diagrama de secuencia para visualizar aplicaciones
Realizado por: Edgar Córdova, 2017

El diagrama de colaboración de este requerimiento modela la interacción entre el panel principal el cual también interactúa con la interfaz de aplicaciones y para que todo esto sea posible los datos se obtienen de una lista de objetos de tipo aplicación. La Figura No 39-2 única esta interacción.

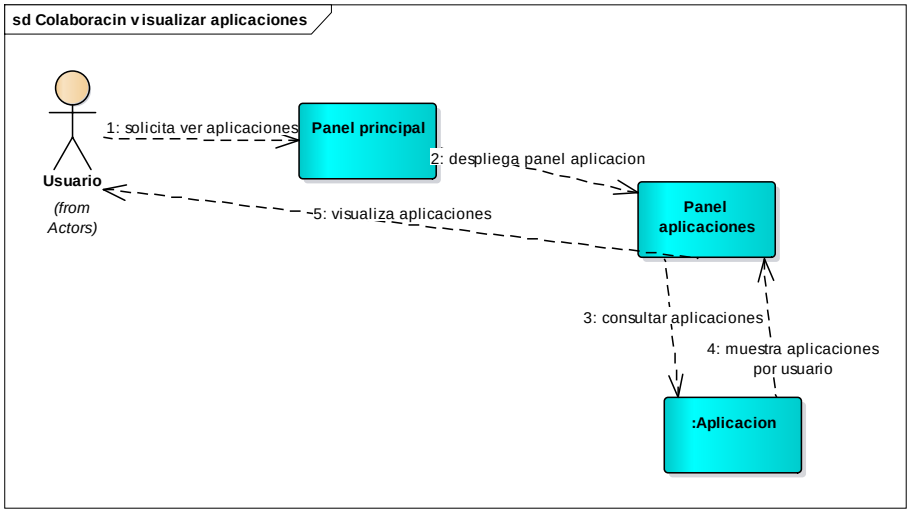


Figura No 39-2. Diagrama de colaboración para visualizar aplicaciones
Realizado por: Edgar Córdova, 2017

Abrir y editar código

El Diagrama de secuencia para abrir y editar código empieza con el usuario solicitando editar código fuente de la aplicación, luego la interfaz de aplicaciones despliega el formulario de edición de texto al usuario quien realiza los cambios al archivo a través de un cuadro de texto sencillo. El diagrama de la Figura No 40-2 muestra la secuencia de edición de un archivo de texto.

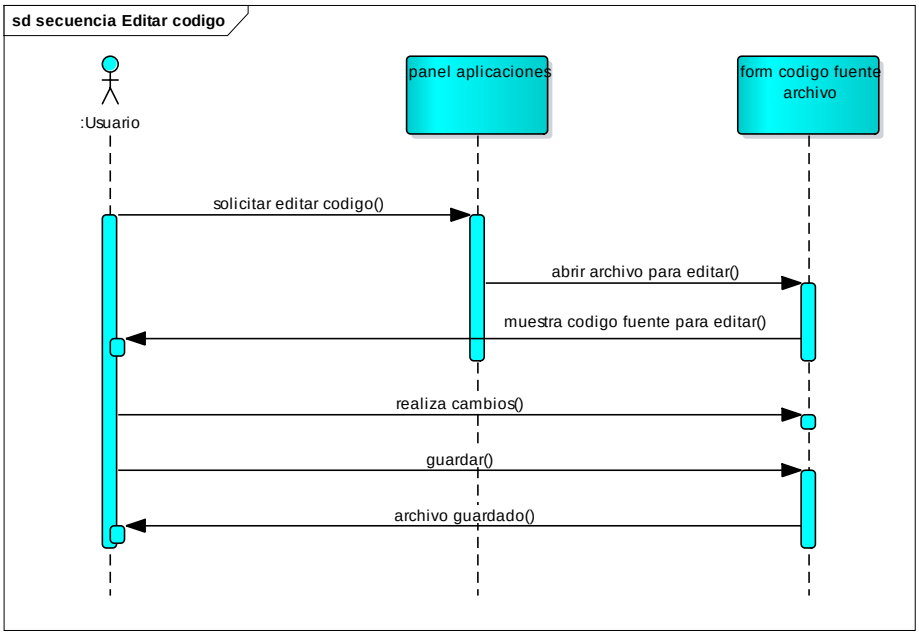


Figura No 40-2. Diagrama de secuencia para editar código fuente
Realizado por: Edgar Córdova, 2017

El diagrama de colaboración de la Figura No 41-2, indica el flujo de mensajes entre el usuario y editor de código fuente de las aplicaciones que se hospedan en el panel de control.

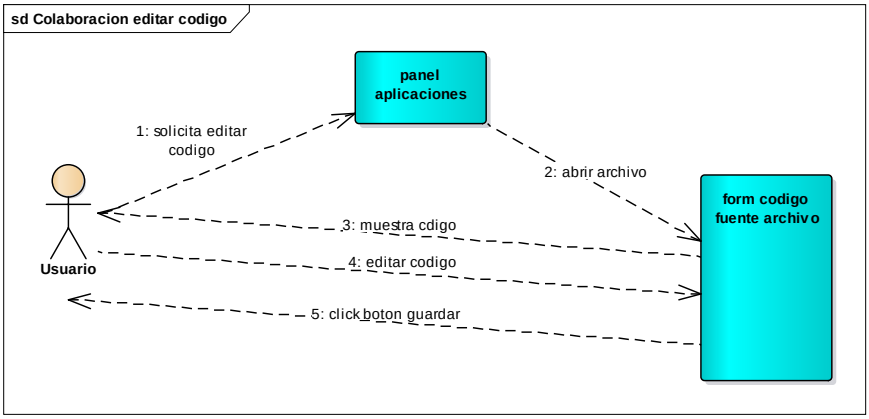


Figura No 41-2. Diagrama de colaboración para editar código fuente
Realizado por: Edgar Córdova, 2017

Autenticación

El proceso de autenticación inicia con el usuario solicitando bloguearse, el panel de administrador despliega un formulario de autenticación para ingresar su correo y su clave. Una vez el usuario ingresa su correo y clave, se envía el formulario a un controlador de acceso donde se busca el usuario, se obtiene sus datos se averigua a que grupo pertenece, entonces se acepta o se rechaza la petición de autenticación como indica la Figura No 42-2, a continuación.

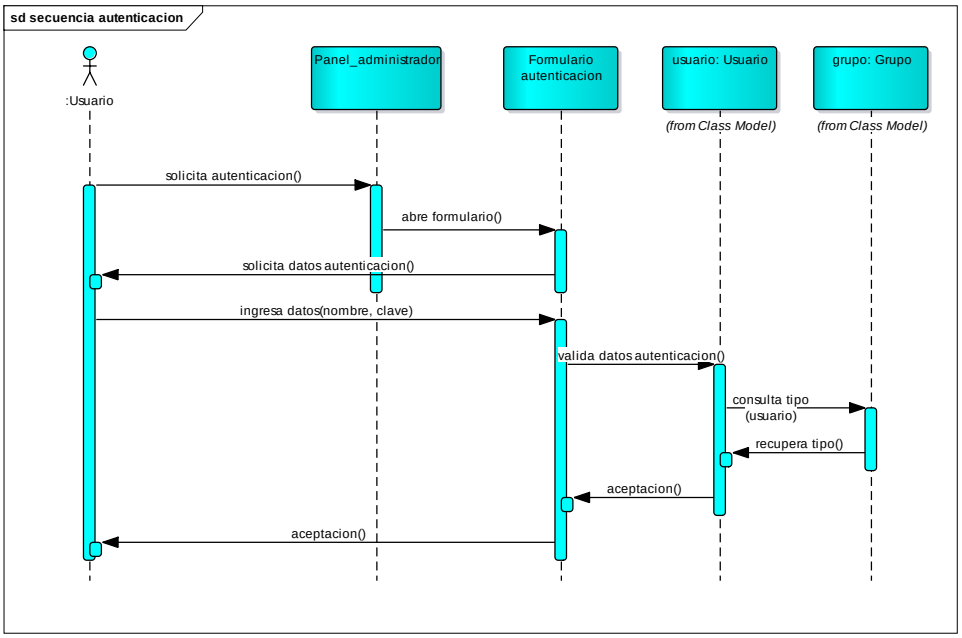


Figura No 42-2. Diagrama de secuencia para autenticación
Realizado por: Edgar Córdova, 2017

Los objetos como usuario, paneles, formularios y controladores de acceso colaboran entre sí a través de mensajes. El actor usuario, interactúa en mayor manera con interfaces y paneles, siendo el resto de interacciones, internas, es decir, ciertos componentes con otros. Esto se indica a través de la figura 43-2.

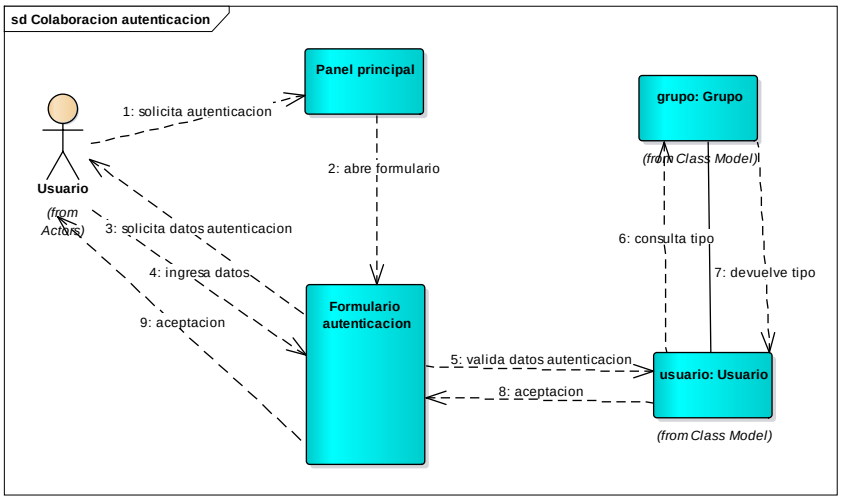


Figura No 43-2. Diagrama de colaboración para autenticación
Realizado por: Edgar Córdova, 2017

Compilar y construir

En la Figura No 44-2, la secuencia de esta funcionalidad comienza con el usuario solicitando la compilación y empaquetado de su código fuente previamente enviado en formato ZIP. Una vez el código descomprimido puede ser visualizado y compilado a través de un botón. Aquí se genera el empaquetado WAR que será desplegado posteriormente.

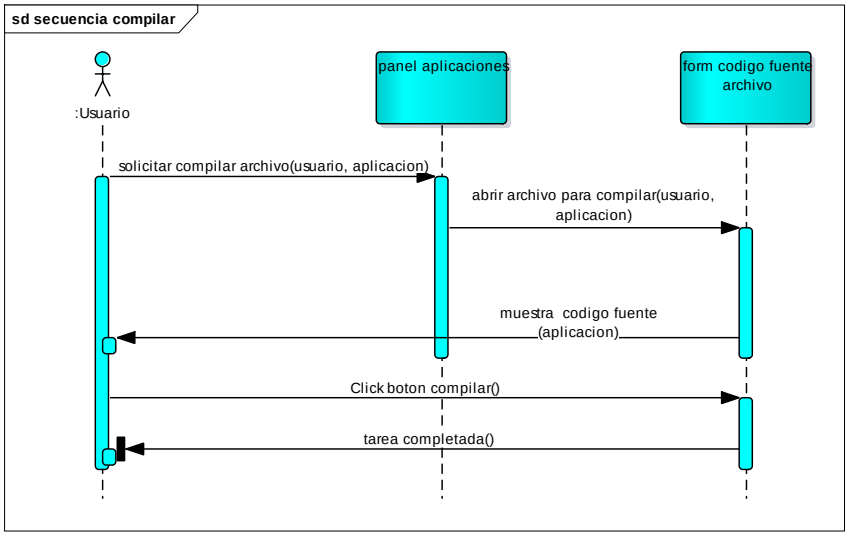


Figura No 44-2. Diagrama de secuencia para compilar y construir desde el código fuente.
Autor: Edgar Córdova

El usuario envía un mensaje solicitando compilar, el objeto panel despliega un formulario con los archivos del código fuente. Este formulario colabora con el actor Usuario proporcionando el botón de compilar, mientras que el usuario da clic en el botón para comenzar el proceso de compilado. Una vez finalizado se enviará un mensaje de tarea completada al usuario. La Figura No 45-2 muestra este diagrama.

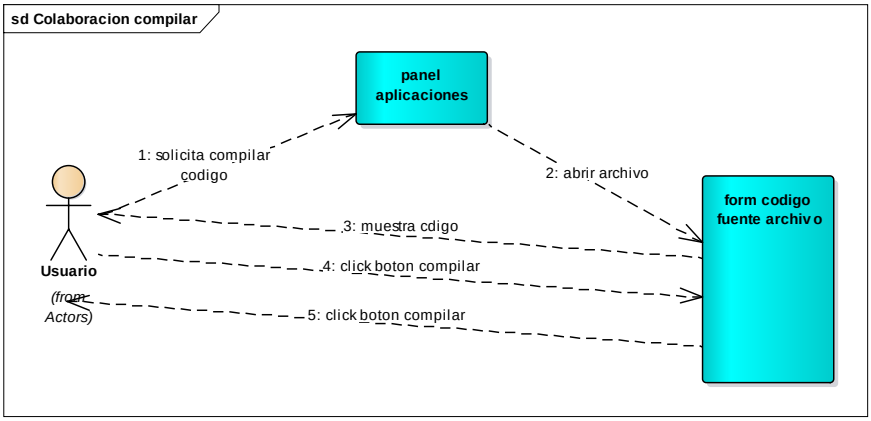


Figura No 45-2. Diagrama de colaboración para compilar y construir desde el código fuente
Realizado por: Edgar Córdova, 2017

Crear nueva base datos

La creación de una nueva base de datos tiene una secuencia muy similar a crear una nueva aplicación con la variante del despliegue del archivo WAR, que en este caso es la restauración de la base de datos en el motor de base de datos, como muestra la Figura No 46-2 a continuación.

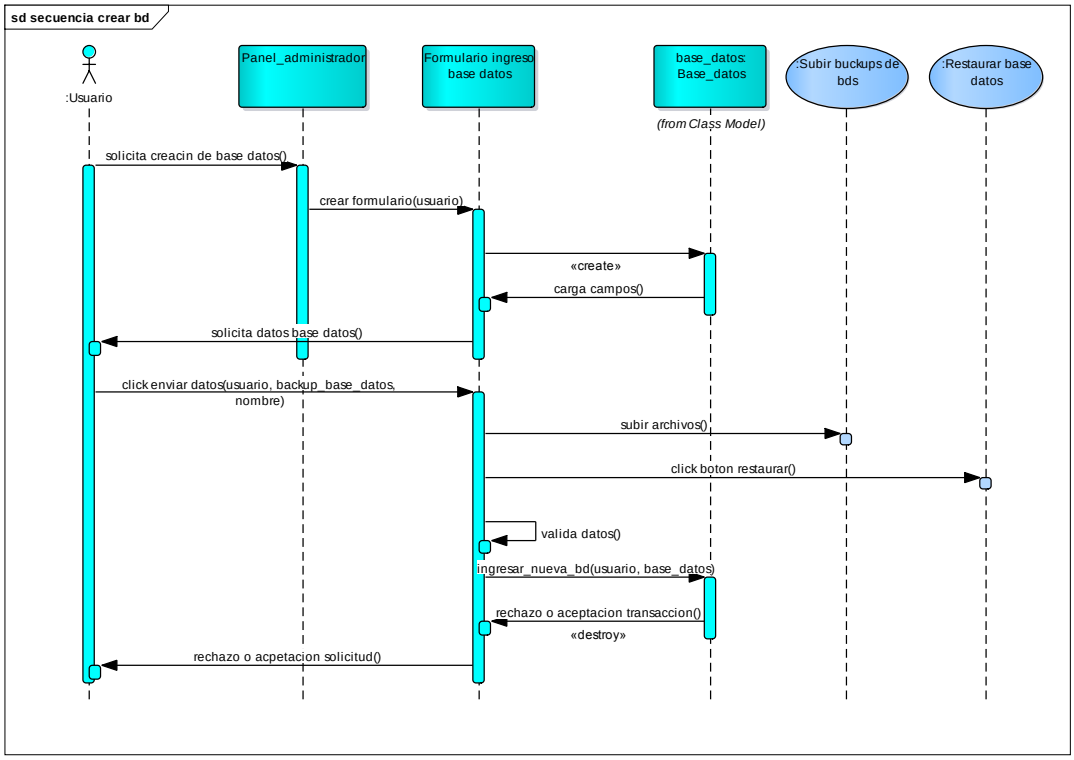


Figura No 46-2. Diagrama de secuencia para crear y restaurar una nueva base de datos
Realizado por: Edgar Córdova, 2017

El Diagrama de Colaboración para crear una nueva base de datos refleja la compartición de mensajes entre el actor Usuario, el panel principal, formulario de ingreso de base de datos, los procesos de subir archivos y restaurar la base de datos y el objeto base de datos con los métodos para ingresar una nueva base de datos, la Figura No 47-2 muestra el diagrama.

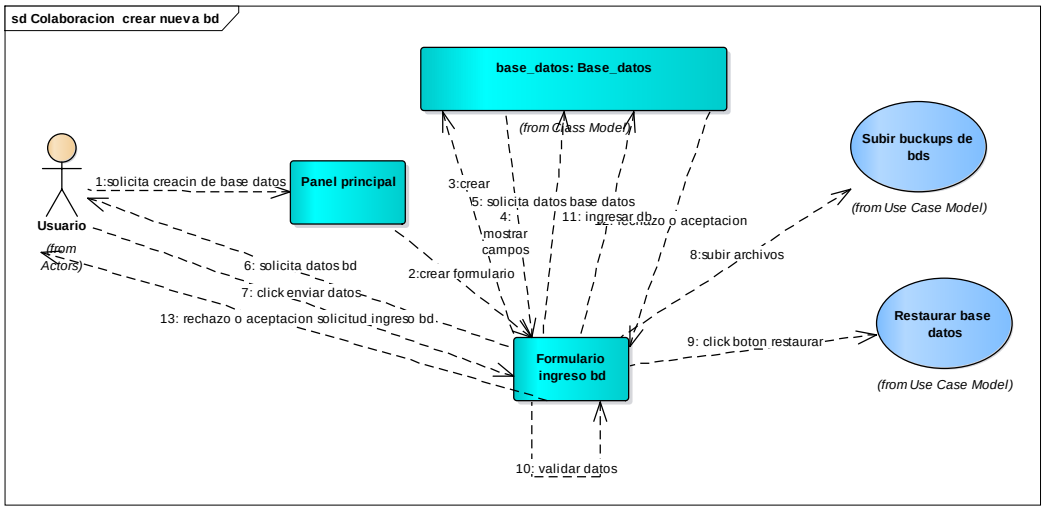


Figura No 47-2. Diagrama de colaboración para crear y restaurar una nueva base de datos
Realizado por: Edgar Córdova, 2017

Visualizar las bases de datos

Para visualizar las bases de datos el usuario solicita al panel de cliente, ver las bases de datos creadas por el usuario. Entonces se despliega una interfaz con todas las bases de datos del usuario en una tabla luego de consultar previamente al objeto base datos que contiene el método para listar las bases de datos, como indica la Figura No 48-2.

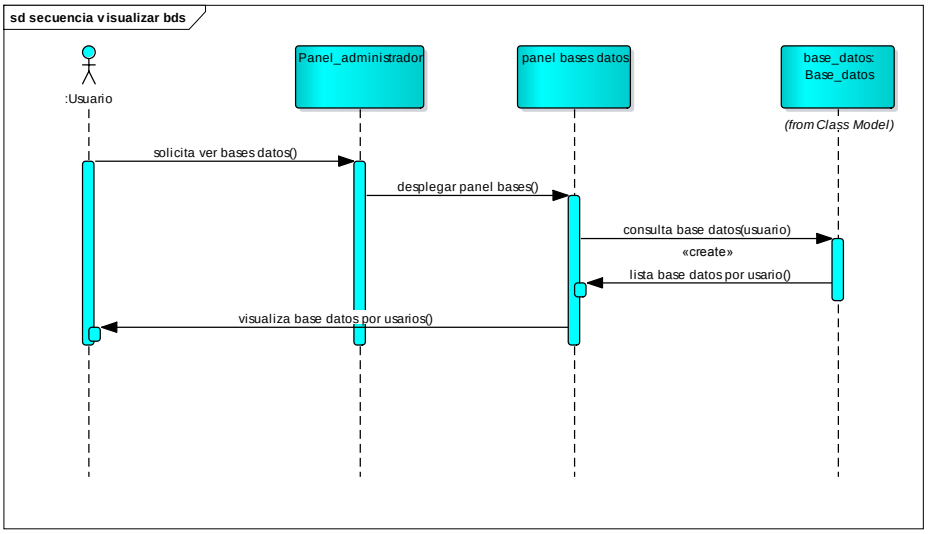


Figura No 48-2. Diagrama de secuencia para visualizar las bases de datos
Realizado por: Edgar Córdova, 2017

El diagrama de colaboración se muestra en la Figura No 49-2, donde se aprecia la manera en que el usuario inicia el proceso enviando un mensaje solicitando listar las bases de datos, el mensaje es atendido por el panel de cliente, luego por un panel de base de datos y finalmente por el objeto base de datos que realiza la consulta y devuelve un listado que será visualizado finalmente por el usuario, como muestra la Figura No 50-2.

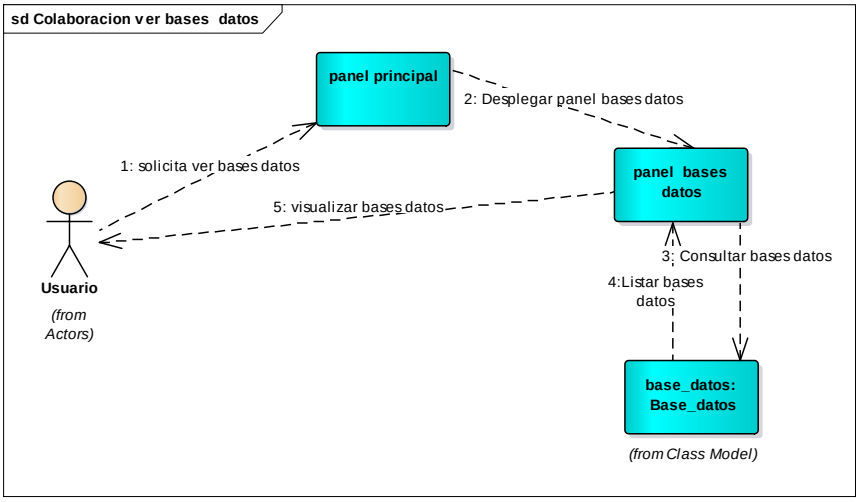


Figura No 49-2. Diagrama de colaboración para visualizar las bases de datos
Realizado por: Edgar Córdova, 2017

Gestionar usuarios

La gestión de usuarios es realizada por un administrador con estado Activo, inicia solicitando la visualización de todos los usuarios registrados, visualizando principalmente su estado el cual se puede actualizar de Activo a Inactivo y Baja, dependiendo de la situación, como indica la Figura No 50-2.

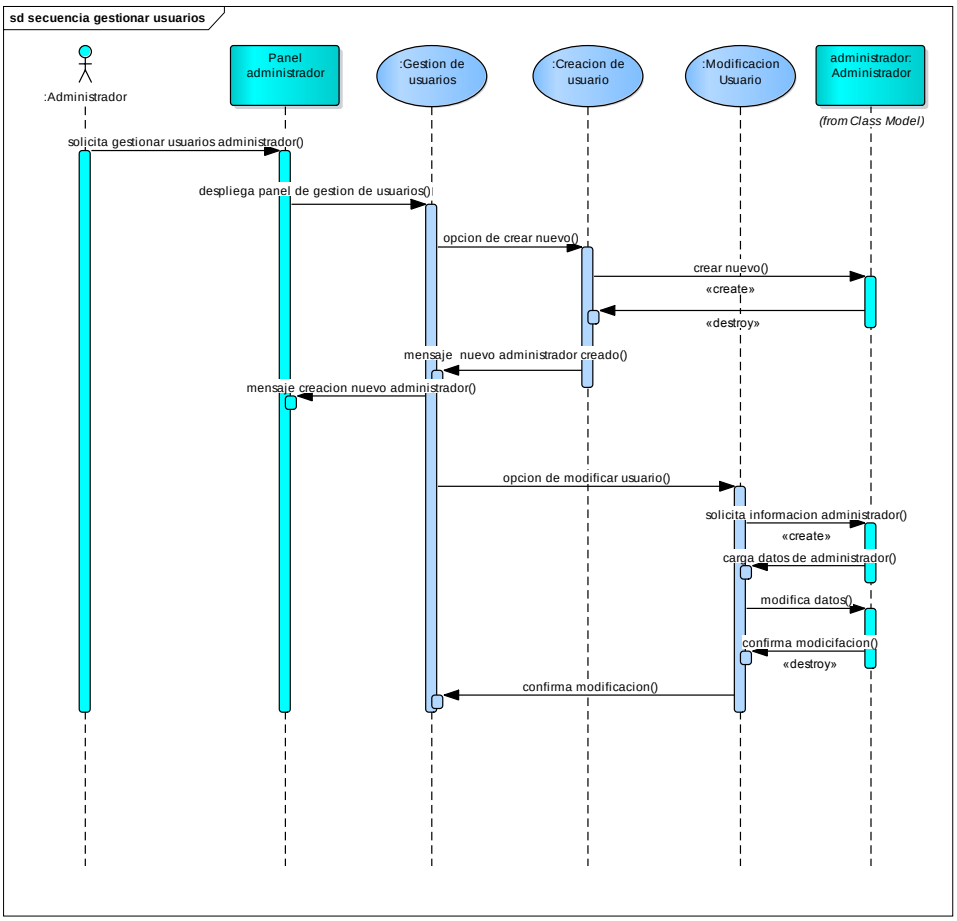


Figura No 50-2. Diagrama de secuencia para gestionar usuarios
Realizado por: Edgar Córdova, 2017

El diagrama de colaboración de la Figura No 51-2, muestra los mensajes enviados entre los objetos que participan en el proceso de gestión de usuarios.

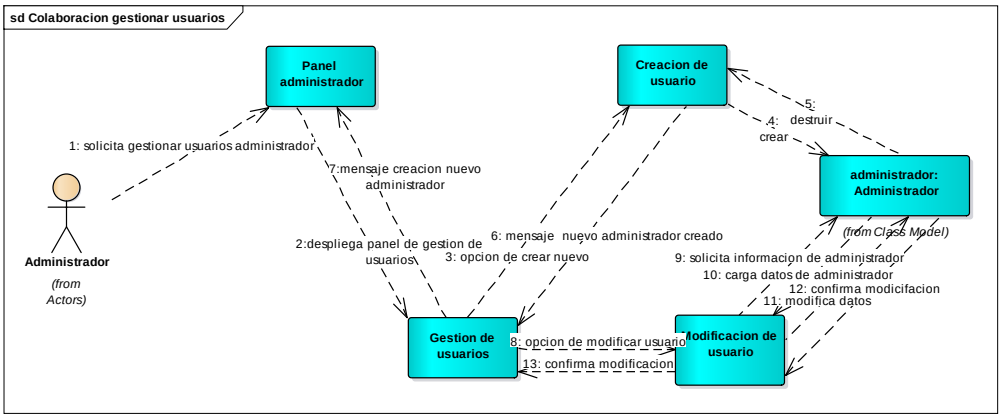


Figura No 51-2. Diagrama de colaboración para gestionar usuarios
Realizado por: Edgar Córdova, 2017

Registro nuevo usuario

El Registro de nuevos usuarios es un proceso que lo realiza el visitante s así lo desea, inicia con la solicitud del actor Visitante a la página de inicio, este desplegará un formulario de registro solicitando los datos personales del nuevo usuario. El usuario ingresa sus datos personales, cada campo es validado por el formulario, luego se envía el formulario el mismo que llamará al método ingresar usuario del objeto usuario. Posteriormente crea los directorios necesarios para almacenar los archivos de la aplicación y bases de datos. Esto se observa de mejor manera en la Figura No 52-2.

El diagrama de colaboración de la Figura No 53-2, muestra el envío de mensajes entre los diferentes objetos que intervienen en el proceso de registro de nuevos usuarios.

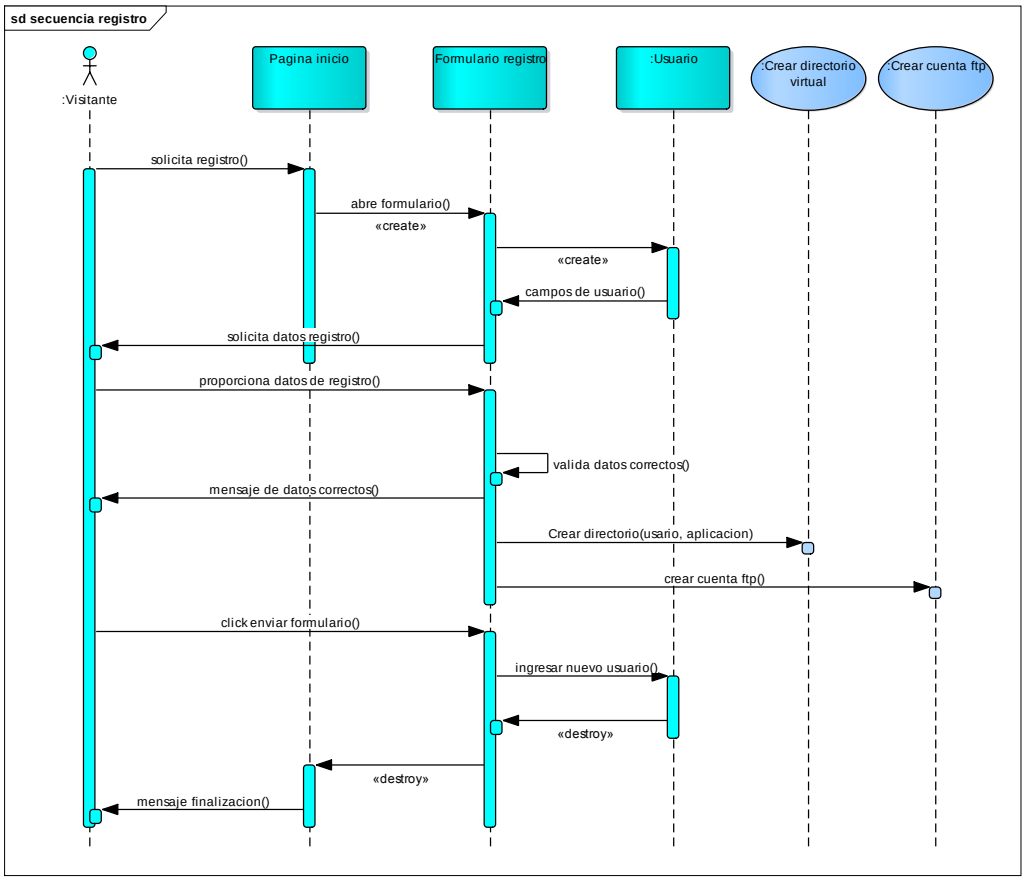


Figura No 52-2. Diagrama de secuencia para registrar nuevos usuarios
Realizado por: Edgar Córdova, 2017

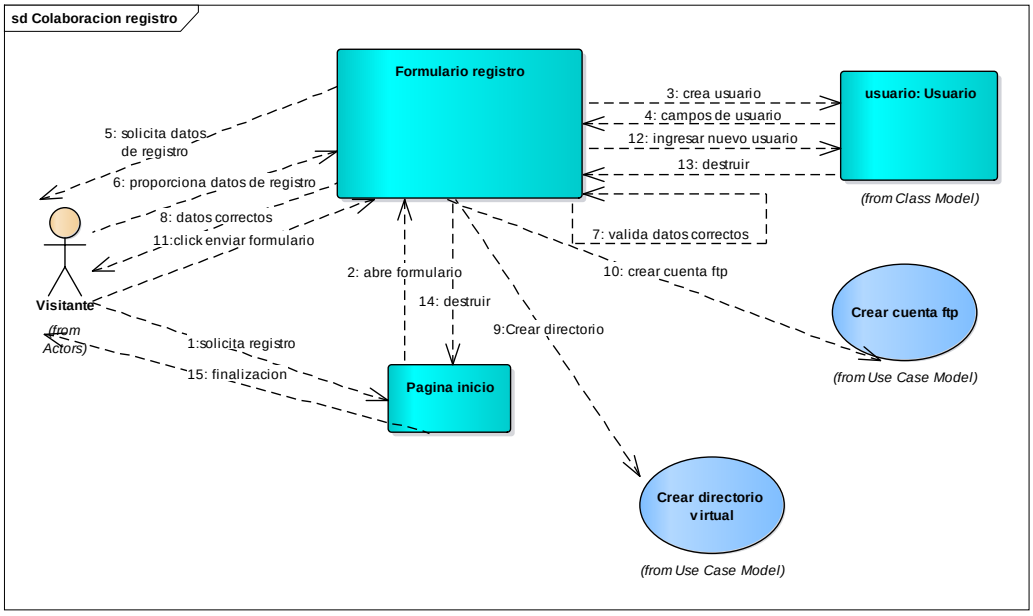


Figura No 53-2. Diagrama de colaboración para registrar nuevos usuarios
Realizado por: Edgar Córdova, 2017

2.5.1.2.6. Diagrama de componentes

La aplicación tiene una arquitectura n capas siguiendo el Modelo Vista Controlador (M.V.C.). En la capa de presentación se utiliza páginas web dinámicas XHTML que a su vez dependerán de otros componentes como JQUERY, Hojas de estilo y la librería PRIMEFACES. Aquí se incluye otro componente que es el informe de pruebas con la librería Selenium. El componente controladores dependerá del paquete Lógica del negocio y esta a su vez de la librería Postgresql. Durante el desarrollo la aplicación definirá varios tipos de pruebas como: pruebas unitarias, de interfaces y de integración, dependiendo las mismas de las librerías jUnit y Selenium. El gráfico del diagrama se muestra en la página 81, en la Figura No 54-2.

2.5.1.2.7. Diagrama de despliegue

El diagrama de muestra la interacción de la aplicación con el medio ambiente, en este caso interactúa primero con los componentes del servidor de aplicaciones Glassfish, también con el componente J.D.K. de JAVA y con el componente de base de datos de PostgreSQL. Todo esto se desenvuelve en un entorno virtualizado dentro de una nube en un servicio IasS. A la aplicación se van a conectar vía Internet uno o más navegadores web. El despliegue de la aplicación se indica en la Figura No 55-2, en la página siguiente

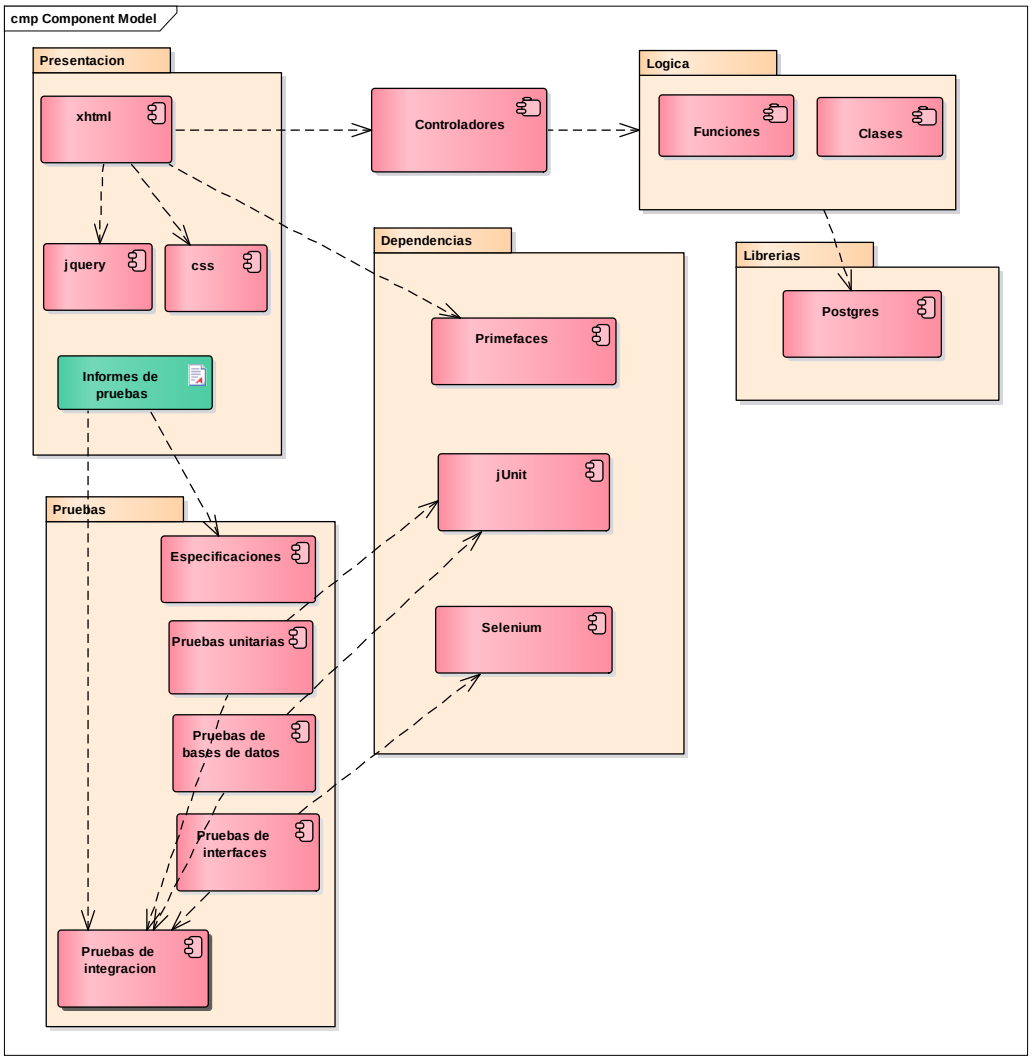


Figura No 54-2. Diagrama de componentes de la aplicación
Realizado por: Edgar Córdova, 2017

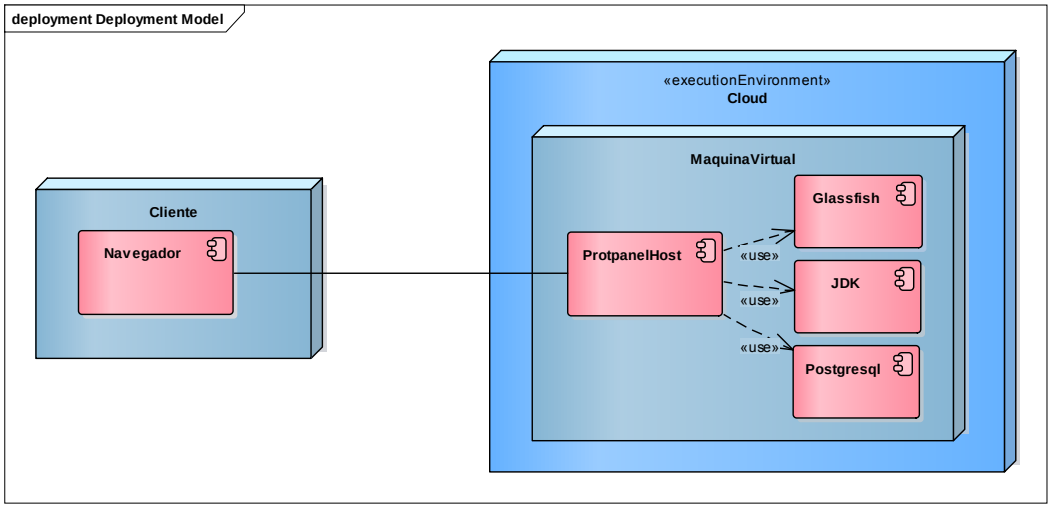


Figura No 55-2. Diagrama de despliegue de la aplicación
Realizado por: Edgar Córdova, 2017

2.5.2. Segunda Fase: Diseño y mejoramiento de pruebas.

2.5.2.1 Diseño de Pruebas- Plan de Pruebas

En la sección de Ambientes de prueba ya se realizó todo el proceso de configuración de la herramienta de integración. El paso siguiente fue realizar el respectivo plan de pruebas como indica la Tabla No 2-2, todos estos formatos fueron obtenidos de la web de conocimiento de IBM, donde de varios formatos se seleccionó la plantilla de Plantilla de plan de prueba de software por ser la que más se ajustó a las características de la aplicación a construir y la técnica a utilizar.

Tabla No 2-2. Plan de pruebas PROTPANEL

Sección del plan de prueba	Plantilla de plan de prueba de software
Resumen	Plan de pruebas para el trabajo de titulación: Prototipo de panel de control para servicio de hosting para la empresa Kapyasoft. Los principales objetivos son: Calidad, seguridad, eficiencia del software a construir. El alcance es realizar las pruebas unitarias, pruebas de integración e Integración continua de los requerimientos del 1 al 14 de lasa historias de usuario definidas. En las tablas del 4 al 18.
Ámbito de plan de prueba	Prototipo de panel de control, hospedaje de aplicaciones web JAVA, seguridad, integridad, privacidad.
Objetivos empresariales	Promocionar el servicio de hospedaje de aplicaciones JAVA por parte de la empresa Kapyasoft
Objetivos de prueba	- Contar con un prototipo funcional de calidad. - Registros de pruebas correctamente documentadas.
Requisitos	- Implementar casos de prueba. - Implementar código - Refactorizar
Enlaces de colección de requisitos	- Req1. Ingresar nuevo usuario - Req 2. Crear directorios virtuales de usuario - Req 3. Crear una cuenta ftp usuario - Req 4. Crear nueva aplicación - Req 5. Subir archivos de aplicación - Req 6. Compilar código fuente de aplicación - Req 7. Autenticación de usuarios clientes - Req 8. Compilar código fuente de aplicación - Req 9. Editar_código_fuente_aplicación - Req 10. Restaurar base de datos - Req 11. Respaldar base de datos
Análisis de riesgo	Riesgo 1. Las pruebas no cubran todas la posibilidades.
Planificaciones de prueba	1. Plan de pruebas (E.A.) 2. Casos de pruebas (E.A.) 3. Script de pruebas (*JUnit*) 4. Ejecución de pruebas *(Junit*) 4.1. Interfaz (Selenium) 4.2. Controladores (Junit) 4.3. Logica(Junit) 4.4. Base de datos(jUnit) 5. Registro de pruebas(Jenkings)
Valoración de prueba	Número de pruebas de pruebas realizadas Número de iteraciones hasta lograr una aceptable funcionalidad.
Entornos de prueba	Servicio de cloud IasS, con
Detalles de entorno de prueba de software	prcesador de 2hz, 4 gb RAM, 2 TB de disco duro, tasa de transferencia de 10GBits

Identificación de prueba	- Pruebas unitarias - Pruebas de integración - integración continua
Equipo de prueba	- Edgar Córdova
Objetivos de calidad	Integridad Seguridad Eficiencia Eficacia
Suites de pruebas	Pruebas interfaces Pruebas de BD Pruebas de lógica
Casos de prueba	CP_01_01. Ingresar_usuario_ingreso_correcto CP_01_02. Ingresar_usuario_datos_vacios_o_null CP_01_03. Ingresar_usuario_dato_entero_repetido CP_02_01. Crear directorios virtuales de usuario_ingreso_correcto CP_02_02. Crear directorios virtuales de usuario_datos_incorrectos CP_03_01. Crear nueva aplicación_ingreso_correcto CP_03_02. Crear una nueva aplicación _datos_incorrectos CP_04_01. Subir archivos de aplicación_ingreso_correcto CP_04_02. Subir archivos de aplicación_ingreso_incorrecto CP_04_03. Subir archivos de aplicación_campos_null CP_05_01. Compilar codigo fuente de aplicación_ingreso_correcto CP_06_01. Autenticación de usuarios clientes_ingreso_correcto CP_06_02. Autenticación de usuarios clientes_nombre_usuario_no_existe CP_06_03. Autenticación de usuarios clientes_clave_no_existe CP_06_04. Autenticación de usuarios clientes_clave_datos_null CP_07_01. Editar_codigo_fuente_aplicacion_ingreso_correcto CP_06_01. Restaurar base de datos_ingreso_correcto CP_06_02. Restaurar base de datos_nombre_archivo_vacio CP_06_03. Restaurar base de datos_nombre_archivo_null CP_07_01. Respaldar base de datos_ingreso_correcto CP_07_02. Respaldar base de datos_nombre_archivo_vacio
Recursos	Conexión a internet Servicio Cloud IasS
Adjuntos	Anexo A

Autor: Edgar Córdova

Los casos de prueba para cada requerimiento se explican en el Anexo B.

2.5.3. Tercera Fase: Ejecución de pruebas

2.5.3.1. Primera entrega - Especificación de pruebas.

Para resolver la primera especificación de pruebas, que se generaron como pruebas de integración, se tuvieron que especificar nuevos métodos que se contemplan en el diseño U.M.L. pero que están dentro de otros requerimientos, sin embargo, estos a su vez necesitan de otros métodos que no se contemplaron a simple vista en el diseño por esta razón es que el número de pruebas se dispara durante su codificación y refactorización, como se aprecia en el Gráfico No 1-2.

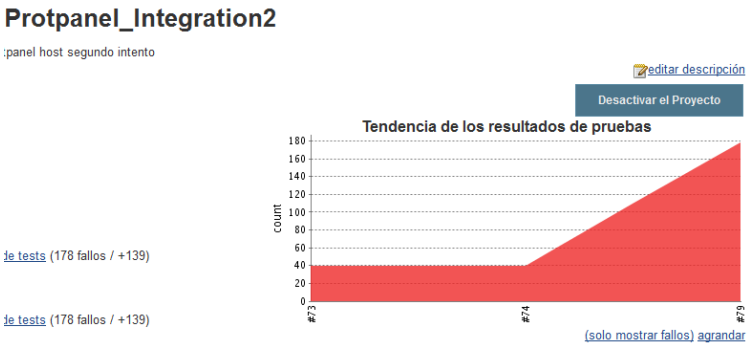


Gráfico No 1-2. Aumento del número de pruebas desde la especificación inicial
Realizado por: Edgar Córdova, 2017

2.5.3.2. Segunda entrega - Incremento de requisitos y pruebas

En una segunda tanda de pruebas se incrementa más aun el número de pruebas por las razones antes explicadas, pues a pesar que se desea realizar una entrega de un primer módulo este se encuentra ligado a controladores, catálogos, clases, por ello las pruebas continuaron aumentando y fallando, como se observa en el Gráfico No 2-2.

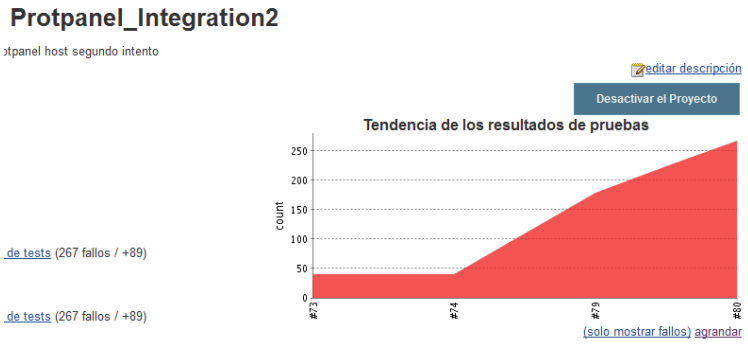


Gráfico No 2-2. Aumento del número de pruebas en la segunda entrega
Realizado por: Edgar Córdova, 2017

2.5.3.3. Tercera entrega - Registro y autenticación

Recién en una tercera ronda de pruebas se aprecian resultados entregables, Estos corresponden al requerimiento de registro y autenticación, como indica el Gráfico No 3-2.

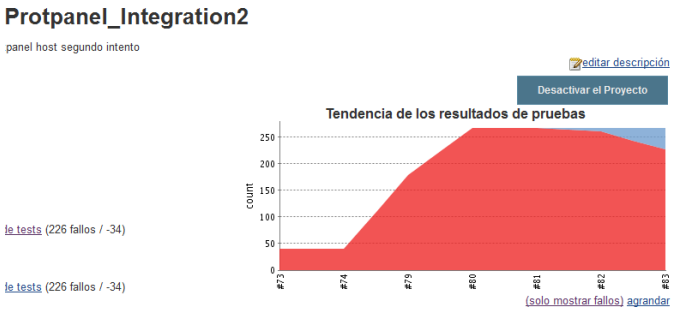


Gráfico No 3-2. Paso de varias pruebas en la tercera entrega
Realizado por: Edgar Córdova, 2017

2.5.3.4. Cuarta entrega de pruebas - Incremento por implementación de otras clases adicionales.- En una cuarta tanda de pruebas los métodos aumentan intentando resolver los requerimientos funcionales iniciales se hace uso de otras clases que no se contemplaban en el diseño, como manejo de archivos, cadenas de caracteres, comandos de la consola de Linux, etc., como se aprecia en el Gráfico No 4-2.

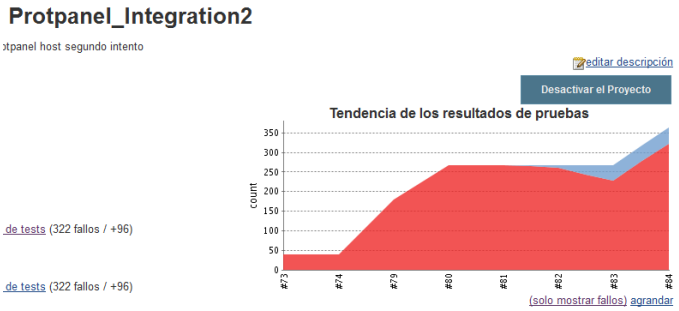


Gráfico No 4-2. Aumento del número de pruebas en la cuarta entrega
Realizado por: Edgar Córdova, 2017

2.5.3.5. Quinta entrega - crear aplicaciones, crear bases de datos, despliegue y restauración.- En esta nueva entrega apreciamos la programación de los utilitarios para archivos, cadenas de caracteres, comandos, etc. El número de pruebas se mantiene constante y se han resuelto los métodos correctamente, como muestra el Gráfico No 5-2.

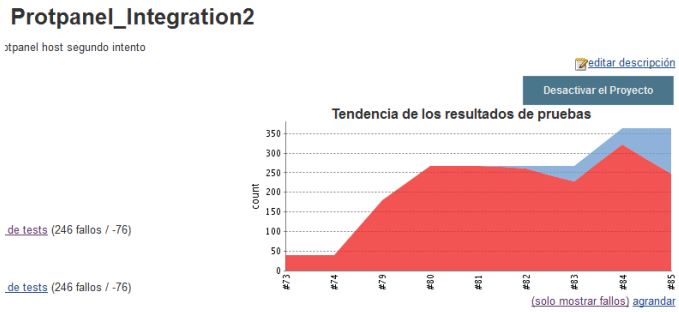


Gráfico No 5-2. Paso de varias pruebas en la quinta entrega
Realizado por: Edgar Córdova, 2017

2.5.3.6. Sexta entrega -Ver aplicaciones, bases de datos, editar código fuente, compilar

En una sexta entrega de la aplicación observamos un poco más del 50% de las especificaciones, ya resueltas, hasta aquí se tiene las principales funcionalidades de la aplicación como la creación de la aplicación, subida de archivos y el despliegue del empaquetado WAR en el servidor de aplicaciones, como se aprecia en el Gráfico No 6-2.

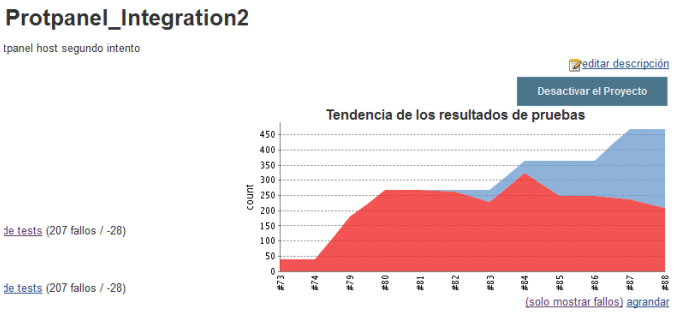


Gráfico No 6-2. Pruebas satisfactorias en más de un 50% en la sexta entrega
Realizado por: Edgar Córdova, 2017

2.5.4. Cuarta fase: Implementación

2.5.4.1. Autenticación de usuarios

El formulario de autenticación solicitará el correo electrónico del usuario registrado y su clave de acceso para ingresar al panel principal, como indica la Figura No 56-2.

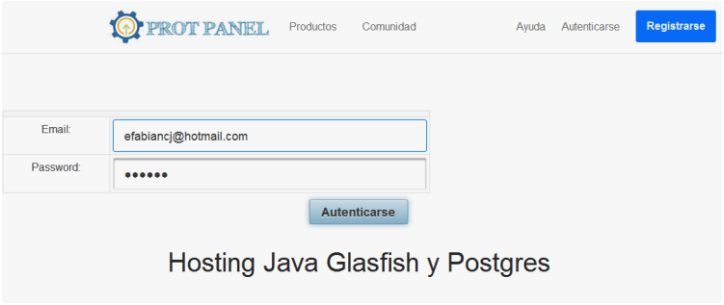


Figura No 56-2. Formulario de Autenticación de usuarios
Realizado por: Edgar Córdova, 2017

2.5.4.2. Panel principal del cliente

El panel principal del cliente proporciona 3 funcionalidades sobre las aplicaciones, bases de datos y dominios y otras 2 sobre reportes de consumo y ventas, como muestra la Figura No 57-2.

Figura No 57-2. Panel principal del cliente
Realizado por: Edgar Córdova, 2017

2.5.4.3. Panel de Visualización de aplicaciones

Esta interfaz muestra todas las aplicaciones creadas por usuario, su código, nombre de la aplicación, directorio donde ha sido creado y el archivo WEB desplegado, como se aprecia en la Figura No 58-2.

Figura No 58-2. Panel de visualización de aplicaciones
Realizado por: Edgar Córdova, 2017

2.5.4.4. Creación de una nueva aplicación

EL formulario de nueva aplicación pide seleccionar el tipo de servicio de aplicaciones que desea, luego un nombre para la aplicación, el usuario será el que haya iniciado sesión, y el botón nueva aplicación creará la aplicación, la misma que no puede ser igual a otra creada por el mismo o por otro usuario. Al instante de crear la nueva aplicación se creará también un nuevo contrato con la fecha actual de ese momento y la fecha de finalización en blanco, como indica la Figura No 59-2.

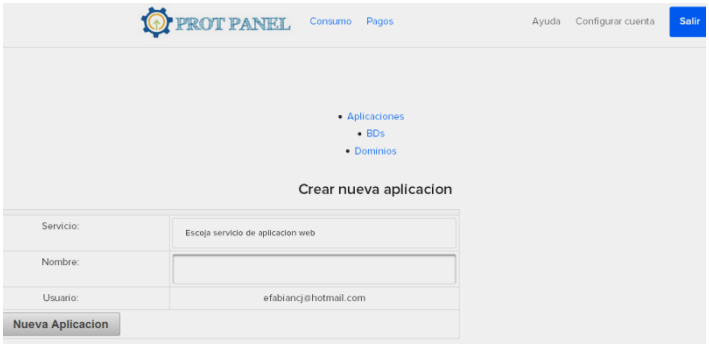


Figura No 59-2. Formulario para crear una nueva aplicación
Realizado por: Edgar Córdova, 2017

2.5.4.5. Ver detalles de aplicación

La Figura No 60-2, muestra el formulario de detalles de aplicación muestra la información de una aplicación de forma más detallada con el nombre de la aplicación, el usuario, directorio,

archivo WEB subido, aplicación desplegada, enlace de lanzamiento de aplicación y con más opciones como subir archivos de aplicación, desplegar el archivo subido, anular el despliegue y eliminar la aplicación creada. Si se han subido archivos con formato ZIP (código fuente), es posible también ver los archivos.

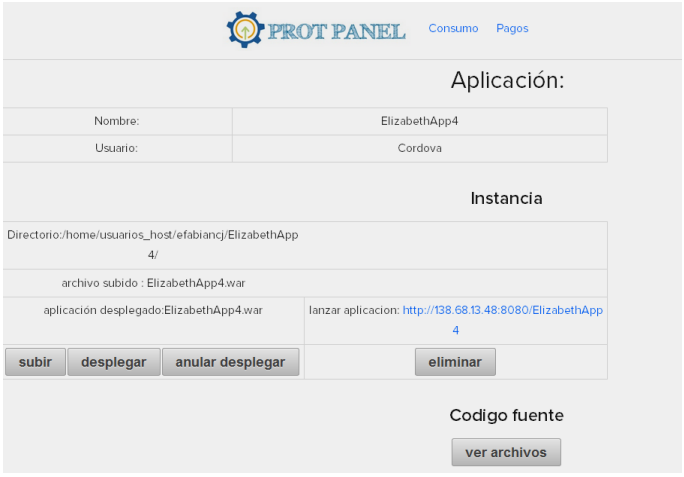


Figura No 60-2. Formulario para visualizar detalles de aplicación
Autor: Edgar Córdova

2.5.4.6. Subir archivos

Este formulario permite subir archivos hacia el servidor los mismos que pueden ser archivos con la extensión .war, .ear, .jar y .zip. Si el archivo enviado al servidor mediante este formulario, es de formato ZIP, luego de guardarlo en el directorio temporal de la aplicación, lo descomprime y luego lo elimina, quedando los archivos de código fuente listos para ser modificados, como indica la Figura No 61-2.

Figura No 61-2. Formulario para subir archivos al servidor
Realizado por: Edgar Córdova, 2017

2.5.4.7. Ver archivos

A través de esta funcionalidad es posible ver todos los archivos subidos por el usuario, en forma de tabla. En cada fila se muestra un elemento que puede ser archivo o directorio. Si es directorio al seleccionar el elemento y dar clic en "VER", se mostrará el contenido del directorio, mientras que si es archivo se visualizará su contenido, en el área de texto debajo, aquí se lo podrá

modificar y guardar. También aquí están los botones de “construir” y “desplegar”, desde el código fuente, esta opción eliminará cualquier archivo WEB, subido o creado anteriormente, como muestra la Figura No 62-2.

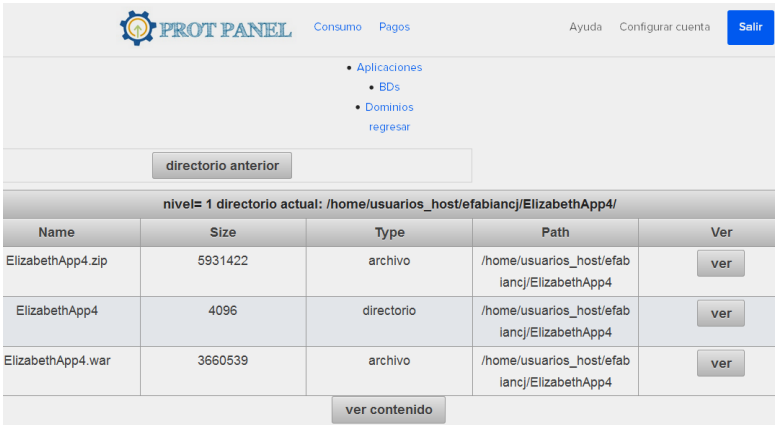


Figura No 62-2. Panel de visualización de archivos subidos por el usuario
Realizado por: Edgar Córdova, 2017

2.5.4.8. Visualizar bases de datos

Esta interfaz muestra todas las bases de datos creadas por el usuario, identificador, nombre y directorio donde se encuentra guardado. Desde aquí se puede seleccionar un elemento de la tabla y dando clic en el botón “VER”, se puede ir formulario de detalles de la base de datos seleccionada, como se aprecia en la Figura No 63-2.

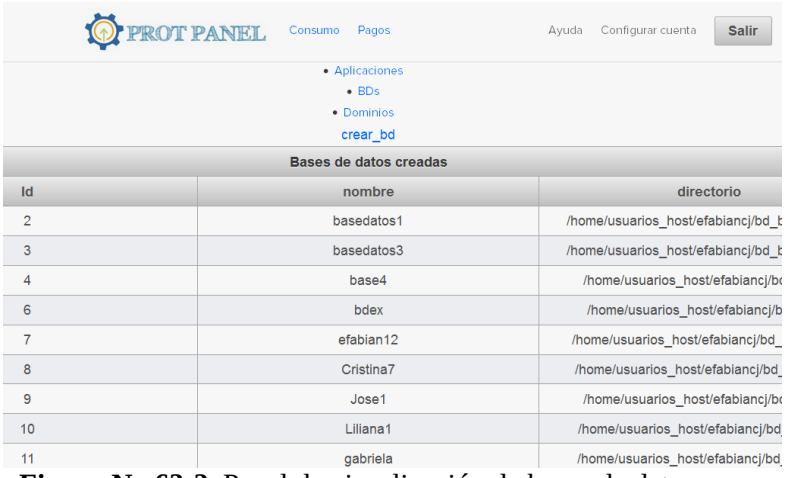


Figura No 63-2. Panel de visualización de bases de datos
Realizado por: Edgar Córdova, 2017

2.5.4.9. Ver detalles de bases de datos

El formulario de detalles de bases de datos, muestra la información de una base de datos de forma más detallada como el nombre de la base de datos, nombre del usuario, archivos enviados y con más opciones como subir archivos de respaldo de base de datos, restaurar la base de datos. En esta sección es también posible eliminar la base de datos, como indica la Figura No 64-2.

Figura No 64-2. Formulario para ver detalles de la base de datos
Realizado por: Edgar Córdova, 2017

2.5.4.10. Crear nueva base de datos

EL formulario pide ingresar un nombre para la nueva base de datos, el campo usuario es el mismo nombre con el que se inició sesión. El botón nueva base de datos, ingresará un nuevo registro en la base de datos de la aplicación y también creará una base de datos real en el motor de base de datos en este caso Postgresql, y también un nuevo contrato de servicio, esto se puede observar en la Figura No 65-2.

Figura No 65-2. Formulario para crear una nueva base de datos
Autor: Edgar Córdova

2.5.4.11. Panel de nombres de dominio

EL formulario pide ingresar un nombre para el nuevo dominio y seleccionar una aplicación ya que todo dominio que se crea debe pertenecer a una sola aplicación. Dando clic en el botón "nuevo dominio" se creará un nuevo dominio, para la aplicación. Cabe indicar que un dominio podrá tener solamente una aplicación, pero una aplicación podrá tener varios nombres de dominio, como indica la Figura No 66-2.

Figura No 66-2. Panel de visualización de nombres de dominio
Autor: Edgar Córdova

2.5.4.12. Reporte de consumo

En esta sección es posible visualizar todos los servicios contratados por el cliente, el costo total en función del tiempo de consumo. Se debe seleccionar la opción de consumo, luego ingresar la fecha desde cuando se desea el reporte y finalmente clic en "mostrar". Se visualizará, el id de contrato, el tipo de servicio, la fecha de inicio del contrato, la fecha de finalización del contrato, costo por hora del servicio y el costo total, como indica la Figura No 67-2.

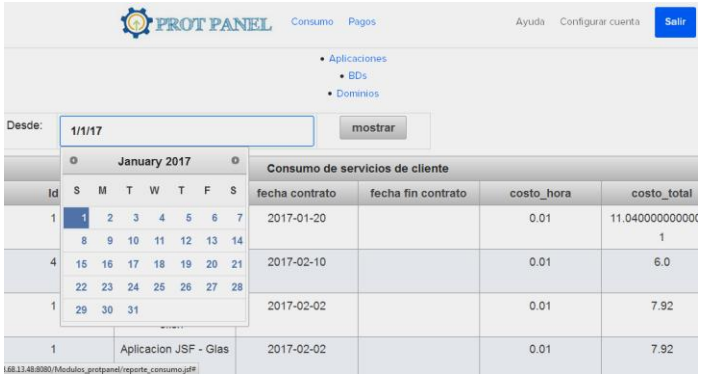


Figura No 67-2. Reporte de consumo de servicios
Autor: Edgar Córdova

2.5.4.13. Reporte de pagos

En esta sección es posible visualizar todos los pagos realizados por el usuario por el cliente, por mes y la fecha en la que se realizó el pago. Se ingresa la fecha desde cuando se desea el reporte y luego en el botón "mostrar", entonces se visualizará, el número del pago, el mes al que corresponde el pago, el valor pagado y la fecha del pago, como se puede apreciar en la Figura No 68-2.

Figura No 68-2. Reporte de pago de servicios
Autor: Edgar Córdova

2.5.5. Quinta fase: Integración

En esta fase se aplica todo lo anteriormente visto con la herramienta de integración continua JENKINS, construcción de proyecto, ejecución automática de pruebas de integración, etc. En esta fase es posible tener un prototipo totalmente funcional listo para ser desplegado en el servidor de aplicaciones.

CAPÍTULO III

ANALISIS DE RESULTADOS

3.1. Resultados de las pruebas realizadas a la aplicación PROTPANEL

Como se aprecia en el Gráfico No 1-3, en total se generaron 420 pruebas automáticamente partiendo de las especificaciones iniciales, clases y catálogos de funciones, así como también de los controladores para las páginas J.S.F., entre pruebas de integración y pruebas unitarias, y 36 pruebas se crearon manualmente posterior a la generación automática según las necesidades que iban apareciendo durante el desarrollo (resultó que aunque se invirtió bastante tiempo diseñando las pruebas, aparecían nuevos métodos que era necesario implementar para que otros funcionen).

De las 456 pruebas generadas se pasaron 265, el resto de pruebas, como se generaron automáticamente correspondían a métodos que no se llegaron a utilizar, por lo que no fue necesario implementarlos y llegaron a existir o especificarse porque a partir del diseño UML de la aplicación se generaron clases y estas clases, con ayuda de un generador de código muy básico, generaron catálogos necesarios para construir la aplicación.

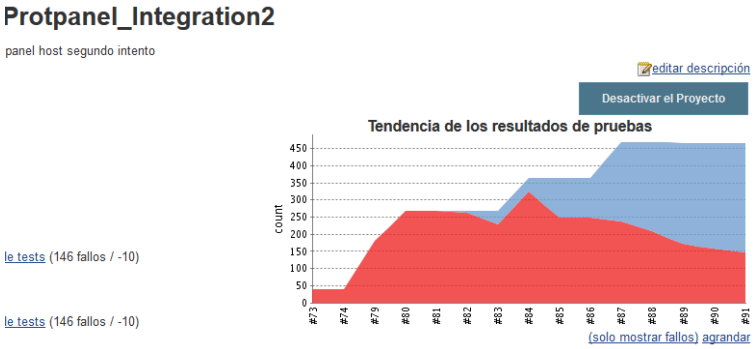


Gráfico No 1-3. Pruebas satisfactorias en más de un 74% en la séptima entrega
Realizado por: Edgar Córdova, 2017

Sin embargo al finalizar la refactorización del código, y eliminar código innecesario, se logró pasar todas las pruebas con lo cual se obtuvieron colateralmente otra tabla de valores con los tiempos de ejecución de las pruebas que se muestran a continuación:

3.2. Resultados del experimento aplicado a los estudiantes.

Del experimento piloto se aprendió que el manual o guía del primer tratamiento necesitaba algunas correcciones y aclaraciones adicionales para que los individuos puedan seguir las

instrucciones sin inconvenientes ni demoras innecesarias. Los datos obtenidos de este primer intento de experimento se observan en la Tabla No 1-3.

Tabla No 1-3. Datos Experimento piloto para comprobar la guía de despliegue (tratamiento) del experimento

No	Tiempo(min) Utilizando guía
1	55,06
2	39,27
3	46,32
4	51,52
5	52,48
6	48,35
7	58,08
8	64,06

Autor: Edgar Córdova

Una vez depurado el material, se realizó el experimento a un grupo de estudiantes obteniendo los siguientes resultados, utilizando una herramienta en línea llamada "TIMEPANTHER" para registrar el tiempo de cada unidad experimental en cada grupo, como indican la Figura No 1-3 y la Figura No 2-3.

En la Tabla No 2-3 y Tabla No 3-3 se han transformado los valores obtenidos de minutos a decimales para realizar los respectivos cálculos.

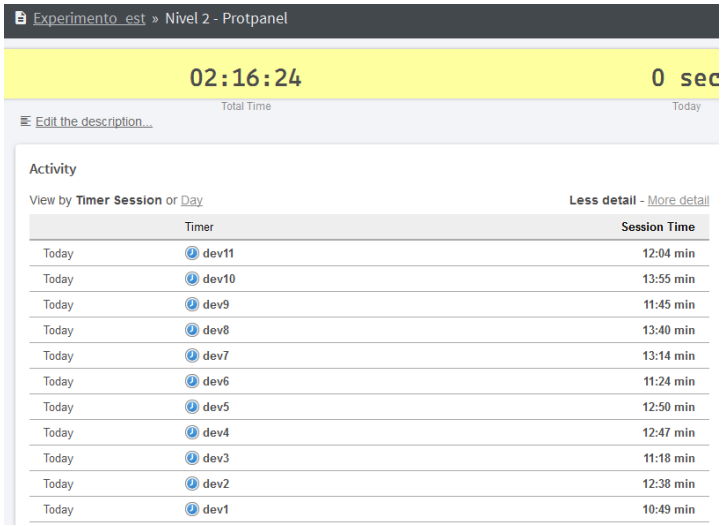


Figura No 1-3. Tiempo hecho por grupo de estudiantes utilizando PROTPANEL

Autor: Edgar Córdova

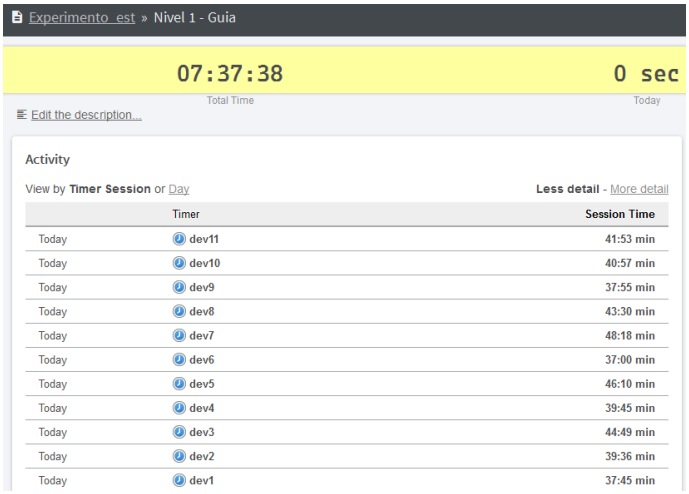


Figura No 2-3. Tiempo registrado por los estudiantes utilizando la guía de despliegue.
Autor: Edgar Córdova

Tabla No 2-3. Tiempos de ambos tratamientos transformados en minutos en forma decimal

Tratamiento 1	Tratamiento 2
37,75	10,81
39,60	39,63
44,82	11,30
39,75	12,78
46,16	12,83
37,00	11,40
48,30	13,23
43,50	13,66
37,91	11,75
40,95	13,91
41,88	12,06

Autor: Edgar Córdova

El Gráfico No 2-3 muestra la comparación de tiempos de ambos tratamientos aplicados sobre ambos grupos de estudiantes. Muestra una considerable mejora en el tiempo de despliegue de aplicaciones.

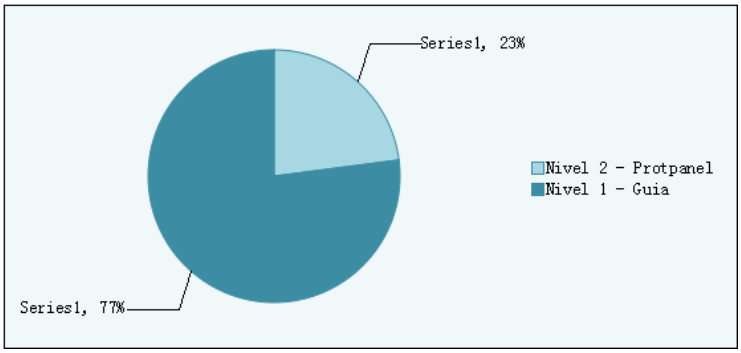


Gráfico No 2-3. Comparación de tiempo de ambos tratamientos.
Autor: Edgar Córdova

Esta diferencia significativa también se puede comprobar por medios estadísticos. Para ello se realizará un análisis de las varianzas utilizando la distribución Ji-Cuadrada.

H_o = NO existe una diferencia significativa entre los dos métodos de despliegue de aplicaciones web JAVA.

H_1 = SI existe una diferencia significativa utilizando la herramienta PROTPANEL y una guía detallada de despliegue de aplicaciones.

Nivel de significancia = α = 0.05
Muestra = n = 6
Grados libertad = r = n - 1=5

$$\begin{cases} H_o: & \mu_1 = \mu_2 \\ H_1: & \mu_1 \neq \mu_2 \end{cases}$$

Calculamos la varianza con la formula

$$S^2 = \frac{\sum_{i=1}^{n}(y_1 - \bar{y})^2}{n-1}$$

Tabla No 3-3. Cálculo de sumas y diferencias de cuadrados para determinar la varianza

N°	GUÍA	PROTPANEL	DIFERENCIA Y1	DIFERENCIA Y2	CUAD DIF Y1	CUAD DIF Y2
1	37,75	10,81	-3,85	-4,04	14,84	16,33
2	39,60	39,63	-2,00	24,78	4,01	614,00
3	44,82	11,30	3,22	-3,55	10,36	12,61
4	39,75	12,78	-1,85	-2,07	3,43	4,29
5	46,16	12,83	4,56	-2,02	20,78	4,08
6	37,00	11,40	-4,60	-3,45	21,18	11,91
7	48,30	13,23	6,70	-1,62	44,87	2,63
8	43,50	13,66	1,90	-1,19	3,60	1,42
9	37,91	11,75	-3,69	-3,10	13,63	9,62
10	40,95	13,91	-0,65	-0,94	0,42	0,89
11	41,88	12,06	0,28	-2,79	0,08	7,79
Suma	**457,62**	**163,36**				
Medias(u)	**41,60**	**14,85**				
Sumas cuad					**74,58**	**663,22**
VARIANZA					**14,92**	**132,64**

Autor: Edgar Córdova

$$S_p^2 = \frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2} = \frac{(10)(14.92)+(10)(132.64)}{11+11-2} = \frac{(149.2)+(1326.4)}{20} = \frac{(1475)}{20} = 73.78$$

La prueba t de dos muestras

$$t_0 = \frac{\bar{y}_1 - \bar{y}_2}{S_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}} = \frac{41.60-14.85}{8.59\sqrt{\frac{1}{11}+\frac{1}{11}}} = \frac{26.75}{3.66} = 7.31$$

Donde y1 y y2 son las medias muestrales, n1 y n2 son los tamaños de las muestras, S_p^2 es una estimación de la varianza $\sigma_1^2 = \sigma_2^2 = \sigma^2$

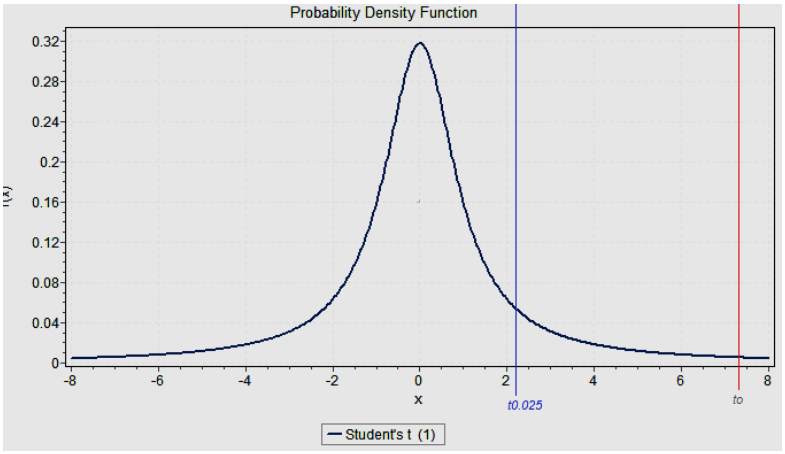


Gráfico No 3-3. Campana de Gauss de las observaciones realizadas.

Autor: Edgar Córdova

Puesto que $t_0 = 7.31 > t_{0.025} = 2.015$, se rechaza H_o y se acepta H_1

3.3. Tiempo requerido para desarrollar el prototipo PROTPANEL utilizando Test-Driven Development

El gráfico 4-3 muestra una comparación de tiempos entre la codificación y el tiempo de diseño de la aplicación. Se aprecia que diseñar el plan de pruebas y los casos de pruebas para las especificaciones iniciales para aplicar T.D.D., se gastó un 16.5% de la duración del proyecto.

Time Breakdown

Project / Timer	Total Time
Implementacion	**251:25:35**
Especificacion pruebas	08:13:44
Configuracion de ambiente de pruebas	27:56:12
Escritura de pruebas	63:08:02
Codificacion	108:34:03
refactorizacion	43:33:34
Diseño	**49:11:43**
Casos de pruebas	28:45:47
Plan de pruebas	20:25:56
uml	0 sec

Gráfico No 4-3. Comparación de tiempo de Codificación y el tiempo de diseño utilizando T.D.D.

Autor: Edgar Córdova

Sin embargo la comparación de tiempos dentro de la implementación la codificación tiene un

42.73%, seguido de la escritura de pruebas con 24.85%, más de la mitad de la codificación, la refactorización del código también tuvo una duración significativa en T.D.D. con un 17.14% al igual que la configuración del ambiente de pruebas indispensable para la integración continua, siendo la parte más pequeña la escritura de las especificaciones iniciales que no deben confundirse con el diseño del plan de pruebas.

Time Breakdown

Timer	Total Time
Especificacion pruebas	08:13:44
Configuracion de ambiente de pruebas	30:35:57
Escritura de pruebas	63:08:02
Codificacion	108:34:03
refactorizacion	43:33:34

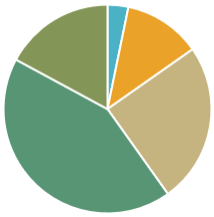


Gráfico No 5-3. Comparación de tiempo de Codificación, escritura de pruebas y refactorización de diseño utilizando T.D.D.
Autor: Edgar Córdova

3.3. Trabajos futuros

Las pruebas de software son una parte esencial del desarrollo de software independiente de la metodología que se emplee. Sin embargo, escribir las pruebas toma un tiempo considerable, por ello se puede desarrollar una aplicación que ayude a generar pruebas automáticas para probar interfaces.

EL proceso de configuración del ambiente de pruebas de integración también tiene su grado de dificultad, y toma cierto tiempo adaptarlo a la aplicación en desarrollo por ello, se puede extender la funcionalidad del prototipo a integrar las aplicaciones con un servidor JENKINS listo para funcionar con todos los plugins necesarios ya instalados.

CONCLUSIONES

Los principios de Test-Driven Development, establecen reglas estrictas de escritura de código, que no permiten realizar la codificación a menos que se escriban las pruebas primero, esto asegura que el cliente final obtenga exactamente lo que solicita y nada más, sin embargo en proyectos medianos o con modelos por capas, se puede generar código automáticamente para intentar satisfacer las pruebas, lo que no se debe olvidar o dejar pasar por ninguna razón es la refactorización, ya que si no se hace esto último no se está aplicando T.D.D.

Escribir las pruebas primero, puede retrasar la implementación de la aplicación debido al tiempo que demora el diseño de planes de prueba y casos de prueba (un 16.5% del tiempo del total del proyecto y 42.73% del tiempo en la codificación). Sin embargo conforme avanza el proyecto, por la práctica que adquiere el desarrollador para elaborar las pruebas, se compensa este tiempo en el producto final que garantiza la funcionalidad de la aplicación y disminuye considerablemente los errores, con el código fuente óptimo para para que funcione la aplicación.

Utilizando el prototipo PROTPANEL, se mejoró significativamente el tiempo de despliegue de una aplicación web JAVA, siendo un 54% más rápido en comparación de una guía detallada para instalación de JAVA, Postgresql y Glassfish.

El número de aplicaciones de la empresa es de una media de 2 aplicaciones web JAVA por mes, teniendo retrasos al momento de probar y poner en producción las aplicaciones, debiendo instalar para ello el conjunto de desarrollo JAVA (J.D.K.), los motores de bases de datos PostgreSQL o MySql y los servidores de aplicaciones Glassfish o Tomcat.

Aunque el resultado de los experimentos mostraron que para el despliegue manual demoró una media de 45.6 minutos se debe considerar que no se ha tomado en cuenta la seguridad del servidor, acceso mediante clientes ftp, repositorios para versionado de las aplicaciones, réplicas de bases de datos, entre otras características que debería cumplir un servidor web profesional y de las que le hacen falta al prototipo.

El prototipo desarrollado con la técnica Test-Driven Development y denominado PROTPANEL, no presenta errores relacionados con las pruebas implementadas.

RECOMENDACIONES

Probar utilizar alguna herramienta de integración continua como JENKINGS para llevar un registro de la ejecución de las pruebas e igual para el diseño del plan de pruebas se puede usar TESTLINK pero en versiones superiores a 2.9 ya que las anteriores tienen algunos errores.

Al tratarse de archivos y al no existir ninguna librería para el I.D.E. que ayude a mapear el sistema de directorios, es recomendable hacer una copia del árbol de directorios en un solo directorio el cual hará de directorio raíz temporal y se borrará una vez terminada la prueba.

Al igual que para los archivos, antes de ejecutar las pruebas sobre bases de datos, es necesario realizar respaldos de la base de datos antes de cada prueba, por si no se dispone de medios para deshacer todos los cambios hechos a la base, se otro modo en el caso de inserciones de nuevos registros, siempre se tendrían que hacer cambios en los datos de entrada de las pruebas, lo que tomaría demasiado tiempo en Test-Driven Development y también al realizar integración continua.

Buscar una herramienta que ayude a generar casos de prueba que es una parte muy importante en T.D.D. y que toma un tiempo considerable.

Liberar la memoria luego de ejecutar las pruebas puesto que algunos controladores como Geckodriver para pruebas con *Selenium*, no se cierran solos y se pueden tener instancias del mismo programa tantas veces se ejecuten las pruebas.

Si se va a utilizar el software de JENKINS para hacer integración continua y en el mismo servidor físico, se debe parametrizar aplicación en cuanto al uso de directorios ya que JENKINS tiene su propia jerarquía de directorios y tiende cambiar las ubicaciones de otros programas como MAVEN, Subversion, etc., afectando el funcionamiento de la aplicación sobre la cual se aplican las pruebas y si se trabaja con directorios y archivos físicos puede provocar que las pruebas siempre fallen.

Tener un servidor para pruebas y otro para la aplicación en producción, debido a que ejecutar las pruebas consume muchos recursos de hardware (procesador y memoria), esto puede ralentizar otras aplicaciones que se estén ejecutando en la misma maquina (aunque sea esta virtual), o que la herramienta informática que ejecuta las pruebas simplemente se salte las pruebas devolviendo un error .

GLOSARIO

ANT	Herramienta desarrollada en JAVA para proyectos JAVA, para realizar tareas repetitivas basado en archivos de configuración X.M.L.
CENTOS	Distribución del Sistema Operativo Linux, muy utilizada para servidores web.
CLOUD	Nube en español, es un término informático para referirse a la gran cantidad de servidores, que prestan una gran variedad de servicios en Internet, entre estos el de correo, datos, IasS, PasS, BasS, etc.
FAILSAFE	*Plugin* diseñado para ejecutar pruebas de integración, para controlar que si existe algún fallo lo haga de manera segura.
FIREBUG	Plugin para el navegador Firefox, para inspeccionar condigo de páginas web.
FRAMEWORK	Es un esquema o estructura que sirve para organizar un proyecto de software.
GIT	Repositorio de código fuente para versionado de aplicaciones en internet.
GLASSFISH	Servidor de aplicaciones web de uso gratuito para aplicaciones JAVA.
GECKODRIVER	Driver utilizado por el método WebDriver de Selenium para llamar a las funciones del navegador, muy utilizado para realizar pruebas automáticas de software sobre interfaces web.
IASS	Infraestructura como Servicio, es una máquina virtual con un sistema operativo básico instalado.
IBM	Máquinas de Negocios Internacionales, es una empresa fabricante computadores de mucho prestigio a nivel mundial, una de las primeras, que empezó a vender ordenadores. En la actualidad vende super ordenadores y servidores y también ofrece servicios de Cloud.
IT	Sub-fijo al final del nombre de una clase JAVA para indicar que se trata de una prueba de integración.
JAVA	Lenguaje de programación desarrollado por Sun Microsystems, y actualmente impulsado por la empresa Oracle. Tiene su propia Interfaz de programación de Aplicaciones por lo que puede ser independiente del Sistema Operativo y se basa en un componente llamado J.R.E. (JAVA Runtime Environment), que contiene la Máquina Virtual de JAVA, el núcleo de su plataforma.

JENKINS	Herramienta de integración continua utilizada para ejecutar todas las pruebas definidas en la aplicación de manera automática y llevar un registro de las pruebas.
LINUX	Sistema Operativo basado creado por Linus Torvalds, bajo licencia GNU (acrónimo recursivo GNU No es Unix), ideal para el levantamiento de servidores web en internet por ocupar poco espacio, ser más estable.
MAVEN	Es una herramienta de código abierto para la gestión de proyectos y librerías creadas por terceros. Permite ubicar librerías en repositorios centrales (en Internet) y locales (en la misma maquina).
NETBEANS	Entorno de desarrollo integrado (I.D.E.) de uso gratuito para aplicaciones JAVA, PHP, C, entre otros.
NICKNAME	Apodo o alias, para identificarse en algún sistema de gestión de manera única.
POSTGRESQL	Motor de base de datos de uso gratuito desarrollado en JAVA.
POM	Archivo de configuración de proyectos MAVEN donde se pueden agregar dependencias, propiedades, repositorios, etc. Funciona con Lenguaje de Etiquetado eXtensible o más conocido por sus siglas en ingles X.M.L.
PRIMEFACES	Librería de código abierto para aplicaciones web en JAVA especialmente para el Framework J.S.F (JAVA Server Faces)
PROTPANEL	Prototipo de software de panel de control para servicio de hosting JAVA
SVN	Herramienta para control de versiones con una estructura más simple que GIT.
TESTCASE	Sub-fijo agregado al final del nombre de una clase JAVA, para indicar que se trata de un caso de pruebas.
TEST	Subfijo al final del nombre de una clase JAVA indicando que se trata de una prueba unitaria.
TIMEPANTHER	Herramienta online para controlar el tiempo de diferentes tareas simultáneamente.
ZIP	Formato de compresión de archivos sin pérdida de datos.

BIBLIOGRAFIA

1. **AMBLER**, Scott. *Introduction to Test Driven Development (TDD).*2002-2013. [Fecha de consulta: 18-12-2016], disponible en http://agiledata.org/essays/tdd.html

2. **BLÉ,** Carlos**.** *Diseño Ágil con TDD. ¿Qué es el desarrollo dirigido por Pruebas?* Los Angeles.: Safecreative. 2010. 308p.

3. **BLÉ,** Carlos. *Desarrollo Dirigido por Pruebas de Aceptación (ATDD).* 2013. [Fecha de consulta: 12-08-2016], disponible en: http://librosweb.es/libro/tdd/capitulo_3.html

4. **BLÉ,** Carlos. *¿Qué es el Desarrollo Dirigido por Pruebas? (TDD).* 2013 [Fecha de consulta: 12-08-2016], disponible en https://librosweb.es/libro/tdd/capitulo_2.html

5. **DELGADO,** Andrés. *Gobernanza de Internet en Ecuador: Infraestructura y acceso Encuentro Nacional de Gobernanza de Internet en Ecuador*, 2014, [Fecha de consulta: 06-11-2016], disponible en: http://delgado.ec/research/es/Gobernanza_Internet_Ecuador_2014.pdf

6. **GONZALES,** Liliana. *Método para generar casos de prueba funcional en el desarrollo de software*.(2009) [Fecha de consulta: 03-011-2016], disponible en: *https://dialnet.unirioja.es/descarga/articulo/4845741.pdf*

7. **HAMMELL,** Thomas, **GOLD,** Russell y **SNYDER,** Tom. *Getting started with Test-Driven Development*. 6 de diciembre 2004 [Fecha de consulta: 18-08-2016], disponible en: http://www.javaworld.com/article/2073090/testing-debugging/getting-started-with-test-driven-development.html

8. **HERRANZ,** Roldán. *TDD como metodología de diseño de software*. 17 de enero del 2011 [Fecha de consulta: 12-08-2016], disponible en: https://www.paradigmadigital.com/dev/tdd-como-metodologia-de-diseno-de-software/

9. **INGENIERIA DE SISTEMAS UNL**. *Prototipos Informaticos*. (2009*).* [Fecha de consulta: 22-12-2016], disponible en: https://sistemas2009unl.wordpress.com/prototipos-informaticos/

9. HAUBERT, Eliezer. *Test Driven Development: Pruebas de Software a priori.* (2009). [Fecha de consulta: 12-11-2016], disponible en http://neuronadigital.blogspot.com/2009/11/test-driven-development-pruebas-de.html

10. **LORDUDUM**. *Introducción a TDD (Test Driven Development)*. 2011 [Fecha de consulta: 12-08-2016], disponible en http://lordudun.es/2011/04/introduccion-a-tdd-test-driven-development/

11. MESZAROS, Gerard. *xUnit Test Patterns: Refactoring Test Code.* México.: Pearson Education, 2007. 944p. ISBN 13:987-0-13-149505-0

12. TUYA, Javier, **RAMOS,** Isabel y **DOLADO,** José Javier. *Técnicas cuantitativas para la gestión en la ingeniería del software*. La Coruña-España.: Netbiblo. 2007. 373p.

13. RUST, Alan, **BISHOP,** Brian y **MCDAID,** Kevin. Institute of Technology. *Investigating the Potential of Test-Driven Development for Spreadsheet Engineering*. [Fecha de consulta: 12-08-2016], disponible en: *https://www.dkit.ie/system/files/RustBishopMcDaidEusprig2006%5B1%5D.pdf*

14. **WIDESKILLS**. *Test Driven Development - JUnit*. (s.f). [Fecha de consulta: 20-08-2016], disponible en: http://www.wideskills.com/junit/test-driven-development-junit

15. WILLIAMS, Laurie, **MAXIMILIEN,** Michael, **VOUK** Mladen., *Test-Driven Development as a Defect-Reduction Practice*. State University, Department of Computer Science. North Carolina. [Fecha de consulta: 03-011-2016], disponible en: https://pdfs.semanticscholar.org/641e/066bd9f5568f38b34a558d0ab2ec737e5bf1.pdf

ANEXOS

Anexo A

Historias de Usuario

HISTORIA DE USUARIO	
Número: 1	**Nombre:** Registro nuevo usuario
Usuario: Visitante	
Número de modificación: 0	**Número de iteración:1**
Prioridad: Alta	**Puntos estimados:**
Riesgo en desarrollo:	
Descripción: El visitante al sitio podrá crear una cuenta personal, proporcionando sus datos personales.	
Observaciones: Por medio de un formulario se le pide al usuario ingrese un nombres, apellidos, teléfono, correo electrónico, ciudad, dirección y una clave o contraseña.	

HISTORIA DE USUARIO	
Número: 2	**Nombre:** Crear directorios virtuales de usuario
Usuario: Cliente	
Número de modificación:0	**Número de iteración:**1
Prioridad: Alta	Puntos estimados
Riesgo en desarrollo:	
Descripción: Al usuario recién registrado el sistema le genera un directorio único para almacenar sus archivos	
Observaciones: Los directorios que se generan son un directorio con su nombre de usuario, otro directorio para los archivos .war y .zip, otro directorio para las bases de datos.	

HISTORIA DE USUARIO	
Número: 3	**Nombre:** Creación de una cuenta ftp
Usuario: Cliente	
Número de modificación:0	**Número de iteración:** 1
Prioridad: Alta	Puntos estimados
Riesgo en desarrollo:	
Descripción: El sistema creará una cuenta ftp relacionada a los directorios recién creados para el usuario	
Observaciones: Se creará una cuenta ftp a cada usuario nuevo que se haya registrado.	

HISTORIA DE USUARIO	
Número: 4	**Nombre:** Crear nueva aplicación
Usuario: Cliente	
Número de modificación:0	**Número de iteración:1**
Prioridad: Media	Puntos estimados
Riesgo en desarrollo:	
Descripción: El usuario puede crear una nueva aplicación ingresando el nombre y subiendo los archivos necesarios	
Observaciones: En un cuadro de texto se ingresa el nombre de la aplicación con la que se mostrará en el sistema, también solicitará el archivo codificado .war o el codigo fuente comprimido .zip	

HISTORIA DE USUARIO	
Número: 5	**Nombre:** Subir archivo de aplicación
Usuario: Cliente	
Número de modificación:0	**Número de iteración:1**
Prioridad: Media	Puntos estimados
Riesgo en desarrollo:	
Descripción: El cliente podrá subir archivos de la aplicación en formato .war o .zip	
Observaciones: Si el archivo enviado es .war se pasa directamente al directorio para ser desplegado. Si es .zip primero se descomprime, se compilará y finalmente se desplegará.	

HISTORIA DE USUARIO	
Número:6	**Nombre:** Compilar y desplegar código fuente
Usuario: Cliente	
Número de modificación:0	**Número de iteración: 1**
Prioridad: Alta	Puntos estimados
Riesgo en desarrollo:	
Descripción: El usuario podrá compilar su código fuente	
Observaciones: Si se a enviado un archivo .zip una vez descomprimido se ejecuta el comando javac para compilar el código fuente y generar el archivo .war.	

HISTORIA DE USUARIO	
Número:7	**Nombre:** Autenticación de usuarios clientes
Usuario: Cliente	
Número de modificación:0	**Número de iteración:**
Prioridad: Alta	Puntos estimados
Riesgo en desarrollo:	
Descripción: El cliente podrá autenticarse con su nombre de usuario y contraseña.	
Observaciones: En el cuadro de texto para el nombre de usuario también se acepta, el correo electrónico único proporcionado por el cliente	

HISTORIA DE USUARIO	
Número:8	**Nombre:** Crear una nueva base de datos
Usuario: cliente	
Número de modificación:0	**Número de iteración:**
Prioridad: alta	Puntos estimados:
Riesgo en desarrollo:	
Descripción: El cliente podrá crear una nueva base de datos.	
Observaciones: EN un cuadro de texto se ingresa el nombre y en otro formulario se sube el archivo plano SQL que se restaurará inmediatamente después de subir el archivo.	

HISTORIA DE USUARIO	
Número:9	**Nombre:** Restaurar base de datos
Usuario: cliente	
Número de modificación:0	**Número de iteración:**
Prioridad: alta	Puntos estimados:
Riesgo en desarrollo:	
Descripción: El cliente podrá respaldar una base de datos seleccionada	
Observaciones: AL visualizar las bases de datos podrá seleccionar cualquier base y escoger si desea sacar un respaldo de la base.	

HISTORIA DE USUARIO	
Número:10	**Nombre:** Respaldar base de datos
Usuario: cliente	
Número de modificación:0	**Número de iteración:**
Prioridad: alta	Puntos estimados:
Riesgo en desarrollo:	
Descripción: El cliente podrá respaldar una base de datos seleccionada	
Observaciones: AL visualizar las bases de datos podrá seleccionar cualquier base y escoger si desea sacar un respaldo de la base.	

HISTORIA DE USUARIO	
Número:11	**Nombre:** Visualizar consumo de recursos
Usuario: cliente	
Número de modificación:0	**Número de iteración:**
Prioridad: alta	Puntos estimados:
Riesgo en desarrollo:	
Descripción: El cliente podrá visualizar el consumo de recursos de memoria, disco y trasferencia de datos.	
Observaciones:	

ANEXO B

Casos de prueba

Una vez detallados los requerimientos de la aplicación y el plan de pruebas se especifican los casos de pruebas para cada especificación:

Caso de prueba 1.1. Ingresar_usuario_ingreso_correcto

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req. 1. Crear nuevo usuario. Si todos los datos son ingresados correctamente.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 1. Ingresar nuevo usuario
Riesgo	- Dependencias con otros Requerimientos.
Condición previa	- Recibe un objeto usuario con todos sus atributos. `New Usuario(1,"jcampos","Juan","Campos","Riobamba","123456","Av. Canonigo Ramos 41-25", "jcampos@gmail.com","0951252362","/home/jcampos1")`
Condición posterior	- Mensaje: `"nuevo usuario creado correctamente, generado directorio virtual y cuenta ftp".`
Resultados esperados	- Ingreso de un nuevo registro usuario en la base de datos
Scripts de prueba	Generado aut - Junit
Registros de ejecución de caso de prueba	Generado aut - Jenkins
Adjuntos	

Caso de prueba 1.2. Ingresar_usuario_datos_vacios_o_null

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req. 1. Crear nuevo usuario. Uno o más campos son null
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 1. Ingresar nuevo usuario
Riesgo	- Dependencias con otros Requerimientos.
Condición previa	- Recibe un objeto usuario con todos sus atributos. `New Usuario()`
Condición posterior	- Mensaje: `"A ocurrido un error al ingresar nuevo usuario, uno o mas capos son null".`
Resultados esperados	- No se realiza ningún cambio en la base de datos
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución	Generado aut - Jenkins

de caso de prueba	Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 1.3. Ingresar_usuario_dato_entero_repetido

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req. 1. Crear nuevo usuario. Un campo de tipo entero ingresado repetido
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 1. Ingresar nuevo usuario
Riesgo	- Dependencias con otros Requerimientos.
Condición previa	- Recibe un objeto usuario con todos sus atributos. `New Usuario(1,”Luis”,”Avalos”,”Riobamba”,”654321”,”Av. Daniel Leon Borja 11-32”,”lavalos@gmail.com”,”0961245812”,”/home/lavalos1”)`
Condición posterior	- Mensaje: “`A ocurrido un error al ingresar nuevo usuario, el id esta repetido ”.`
Resultados esperados	- No se realiza ningun cambio en la base de datos
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 2.1. Crear directorios virtuales de usuario_ingreso_correcto

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-2. Crear directorios virtuales de usuario. Todos los datos ingresados correctamente.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 2. Crear directorios virtuales de usuario
Riesgo	- Dependencias con otros Requerimientos.
Condición previa	- Nuevo usuario ingresado en la base de datos - Recibe un objeto usuario con todos sus atributos. - Campo directorio correcto `New Usuario(1,“Juan”,”Campos”,”Riobamba”,”123456”,”Av. Canonigo Ramos 41-25” , ”jcampos@gmail.com”,”0951252362”,”/home/jcampos1”)`
Condición posterior	- Mensaje: “`nuevo directorio virtual juanc1 creado correctamente”.`
Resultados esperados	- Crea un nuevo archivo con el nombre del usuario concatenado con el la primera letra del apellido, concatenado con número de id

Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 2.2. Crear directorios virtuales de usuario_datos_incorrectos

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-2. Crear directorios virtuales de usuario. Nombre de usuario_incorrecto
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 2. Crear directorio virtual de usuario
Riesgo	- Dependencias con otros Requerimientos.
Condición previa	- Nuevo usuario ingresado en la base de datos - Recibe un objeto usuario con todos sus atributos. - Campo directorio correcto `New Usuario(1,"Juan@","Campos","Riobamba","123456","Av. Canonigo Ramos 41-25" , "jcampos@gmail.com","0951252362","/home/jcampos1")`
Condición posterior	- Mensaje: "`nombre de usuario incorrecto`".
Resultados esperados	- Ningun archivo creado
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 2.3. Crear directorios virtuales de usuario_datos_incorrectos

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-2. Crear directorios virtuales de usuario. Campos de usuario null.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 2. Crear directorio virtual de usuario
Riesgo	- Dependencias con otros Requerimientos.
Condición previa	- Nuevo usuario ingresado en la base de datos - Recibe un objeto usuario con todos sus atributos. - Campo directorio correcto `New Usuario()`

Condición posterior	- Mensaje: "uno o mas campos son null".
Resultados esperados	- Ningun archivo creado
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 3.1. Crear nueva aplicación_ingreso_correcto

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-4. Crear nueva aplicación. Todos los datos ingresados correctamente.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 4. Crear nueva aplicación
Riesgo	- Dependencias con otros Requerimientos.
Condición previa	- Usuario autenticado - Recibe un objeto `Aplicación` con todos sus atributos. - nombre de aplicación `New Aplication(1,"aplicacion1.war","aplicacion1","/home/jcampos1")`
Condición posterior	- Mensaje: "nueva aplicación creado correctamente".
Resultados esperados	- Crea un nuevo registro en aplicación en la base de datos
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 3.2. Crear una nueva aplicación _datos_incorrectos

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-4. Crear una nueva aplicación. Nombre de aplicación
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 4. Crear nueva aplicación
Riesgo	- Dependencias con otros Requerimientos.
Condición previa	- Usuario autenticado

	- Recibe un objeto `Aplicación` con todos sus atributos. - nombre de aplicacion `New Aplication(1,"aplicacion1.war","1#aplicacion","/home/jcampos 1")`
Condición posterior	- Mensaje: `"nombre de aplicacion incorrecto".`
Resultados esperados	- Ninguna aplicacion creada
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 3.3. Crear una aplicacion _datos_incorrectos

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-4. Crear una nueva aplicación. Campos de usuario null.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 3. Crearde una cuenta ftp
Riesgo	- Dependencias con otros Requerimientos.
Condición previa	- Nuevo usuario ingresado en la base de datos - Recibe un objeto `Aplicación` con todos sus atributos. - Campo directorio correcto `New Aplicación()`
Condición posterior	- Mensaje: `"uno o mas campos de aplicación son null".`
Resultados esperados	- Ninguna aplicación creada
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 4.1. Subir archivos de aplicación_ingreso_correcto

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-5. Subir archivos de aplicación. Todos los datos ingresados correctamente.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 5. Subir archivo de aplicación
Riesgo	- Dependencias con otros Requerimientos.

Condición previa	- Usuario autenticado - Archivos de aplicacion (war, zip). - Nombre de aplicación `nombre archivo = "aplicacion1.zip"`
Condición posterior	- Mensaje: "`archivo de aplicación subido correctamente`".
Resultados esperados	- Guarda un nuevo archivo en el directorio virtual del usuario
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 4.2. Subir archivos de aplicación_nombre_incorrecto

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-5. Subir archivos de aplicación. Nombre de archivo increcto.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 5. Subir archivo de aplicación
Riesgo	- Dependencias con otros Requerimientos.
Condición previa	- Usuario autenticado - Archivos de aplicacion (war, zip). - Nombre de aplicación `nombre archivo= 11221#aplicacion.zip`
Condición posterior	- Mensaje: "`nombre de archivo de aplicación incorrecto`".
Resultados esperados	- Ningun archivo subido
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 4.3. Subir archivos de aplicación_campos_null

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-5. Subir archivos de aplicación. Campos null.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 5. Subir archivo de aplicación
Riesgo	- Dependencias con otros Requerimientos.

Condición previa	- Usuario autenticado - Archivos de aplicacion (war, zip). - Nombre de aplicación `nombre_archivo = null`
Condición posterior	- Mensaje: "`nombre de archivo de aplicación incorrecto`".
Resultados esperados	- Ningun archivo subido
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 5.1. Compilar codigo fuente de aplicación_ingreso_correcto

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-6. Compilar codigo fuente de aplicación. Todos los datos ingresados correctamente.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 6. Compilar codigo fuente de aplicación
Riesgo	- Dependencias con otros Requerimientos.
Condición previa	- Usuario autenticado - Archivos de aplicacion descomprimidos. - Nombre de aplicación `nombre archivo = "aplicacion1.zip"`
Condición posterior	- Mensaje: "`compilacion satisfactoria`".
Resultados esperados	- Compila codigo fuente de aplicacin y genera nuevo .war en directorio virtual de usuario.
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 7.1. Autenticación de usuarios clientes_ingreso_correcto

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-7. Autenticación de usuarios clientes. Todos los datos ingresados correctamente.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 7. Autenticación de usuarios clientes

Riesgo	- Dependencias con otros Requerimientos.

Condición previa	- Nombre de usuario
	- Clave de usuario
	
	`nombre_usuario = ”jcampos”`
	`clave_usuario = “123456”`

Condición posterior	- Mensaje: “`autenticación satisfactoria`”.
Resultados esperados	- Redireccionamiento automatico a index_usuario.xhtml.
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 7.2. Autenticación de usuarios clientes_nombre_usuario_no_existe

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-7. Autenticación de usuarios clientes. Nombre de usuario no_existe.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 7. Autenticación de usuarios clientes
Riesgo	- Dependencias con otros Requerimientos.

Condición previa	- Nombre de usuario
	- Clave de usuario
	
	`nombre_usuario = ”jcampos123123”`
	`clave usuario = “123456”`

Condición posterior	- Mensaje: “`autenticación fallida, nombre no existe`”.
Resultados esperados	- Redireccionamiento al inicio.xhtml.
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 7.3. Autenticación de usuarios clientes_clave_no_existe

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-7. Autenticación de usuarios clientes. clave no_existe.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 7. Autenticación de usuarios clientes

Riesgo	- Dependencias con otros Requerimientos.

Condición previa	- Nombre de usuario
	- Clave de usuario
	
	`nombre_usuario = ”jcampos”`
	`clave_usuario = “111222”`

Condición posterior	- Mensaje: “`autenticación fallida,` clave `no existe”.`
Resultados esperados	- Redireccionamiento al inicio.xhtml.
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 7.4. Autenticación de usuarios clientes_clave_datos_null

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-7. Autenticación de usuarios clientes. Datos_null.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 7. Autenticación de usuarios clientes
Riesgo	- Dependencias con otros Requerimientos.

Condición previa	- Nombre de usuario
	- Clave de usuario
	
	`nombre usuario = null`
	`clave_usuario = null`

Condición posterior	- Mensaje: “`autenticación fallida,` datos null”.
Resultados esperados	- Redireccionamiento al inicio.xhtml.
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 8.1. Compilar codigo fuente de aplicación_ingreso_correcto

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-6. Compilar codigo fuente de aplicación. Todos los datos ingresados correctamente.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 6. Compilar codigo fuente de aplicación
Riesgo	- Dependencias con otros Requerimientos.

Condición previa	- Usuario autenticado - Archivos de aplicacion descomprimidos. - Nombre de aplicación `nombre archivo = "aplicacion1.zip"`
Condición posterior	- Mensaje: "`compilacion satisfactoria`".
Resultados esperados	- Compila codigo fuente de aplicacin y genera nuevo .war en directorio virtual de usuario.
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 9.1. Editar_codigo_fuente_aplicacion_ingreso_correcto

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-9. Editar_codigo_fuente_aplicacion. Todos los datos ingresados correctamente.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 9. Editar_codigo_fuente_aplicacion
Riesgo	- Dependencias con otros Requerimientos.

Condición previa	- Nombre de usuario
	- Nombre aplicacion
	- Nombre archivo
	
	`nombre_usuario = "jcampos"`
	`Nombre_aplicacion = "aplicacion1"`
	`Nombre archivo = "aplicacion1"`

Condición posterior	- Mensaje: "`archivo modificado`".
Resultados esperados	- Abre archivo .java, xhtml, jsf, css, etc. - Escribe en archivo .java, xhtml, jsf, css, etc. - Guardar archivo.java, xhtml, jsf, css, etc.
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 10.1. Restaurar base de datos_ingreso_correcto

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-10. Restaurar base de datos. Todos los datos ingresados correctamente.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 10. Restaurar base de datos
Riesgo	- Dependencias con otros Requerimientos.

Condición previa	- Nombre de usuario
	- Nombre archivo backup
	
	`nombre_usuario = ”jcampos”`
	`Nombre_archivo = “bd1.backup”`

Condición posterior	- Mensaje: `“Restauración satisfactoria”.`
Resultados esperados	- Restaura bd - Redireccionamiento automatico a visualizar_bds.xhtml.
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 10.2. Restaurar base de datos_nombre_archivo_vacio

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-10. Restaurar base de datos. nombre_archivo no_existe.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 10. Restaurar base de datos
Riesgo	- Dependencias con otros Requerimientos.

Condición previa	- Nombre de usuario
	- Nombre archivo backup
	
	`nombre usuario = ”jcampos”`
	`Nombre archivo = “”`

Condición posterior	- Mensaje: `“restauracion fallida, nombre achivo vacio”.`
Resultados esperados	- No Restaura bd - No redirecciona a ningun lugar
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 10.3. Restaurar base de datos_nombre_archivo_null

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-10. estaurar base de datos nombre_archivo no_existe.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 10. Restaurar base de datos
Riesgo	- Dependencias con otros Requerimientos.

Condición previa	- Nombre de usuario
	- Nombre archivo backup
	
	`nombre_usuario = ”jcampos”`
	`Nombre_archivo = null`

Condición posterior	- Mensaje: “`restauracion fallida, nombre achivo vacio`”.
Resultados esperados	- No Restaura bd - No redirecciona a ningun lugar
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 11.1. Respaldar base de datos_ingreso_correcto

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-11. Respaldar base de datos. Todos los datos ingresados correctamente.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 11. Respaldar base de datos
Riesgo	- Dependencias con otros Requerimientos.

Condición previa	- Nombre de usuario
	- Nombre bd
	
	`nombre_usuario = ”jcampos”`
	`Nombre_archivo = “bd1”`

Condición posterior	- Mensaje: “Respaldo `satisfactorio`”.
Resultados esperados	- Respalda bd - Redireccionamiento automatico a guardar_archivo.xhtml.
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

Caso de prueba 11.2. Restaurar base de datos_nombre_archivo_vacio

Sección de un caso de prueba	Plantilla de caso de prueba clásica
Resumen	Caso de prueba para el req-11. Respaldar base de datos. Nombre de usuario no_existe.
Requisitos	- Implementar caso de prueba. - Implementar código - Refactorizar
Enlaces de requisitos	Req. 11. Respaldar base de datos
Riesgo	- Dependencias con otros Requerimientos.

Condición previa	- Nombre de usuario
	- Nombre bd
	
	`nombre_usuario = ”jcampos”`
	`Nombre_archivo = “bd1”`

Condición posterior	- Mensaje: “`respaldo fallido, nombre achivo vacio”.`
Resultados esperados	- No Respalda bd - No redirecciona a ningun lugar
Scripts de prueba	Generado aut - Junit Link- anexo21.
Registros de ejecución de caso de prueba	Generado aut - Jenkins Link- anexo22.
Adjuntos	Ninguno

ANEXO C

Fotografías del Experimento

Foto No 1. Grupo de estudiantes utilizando la herramienta Protpanel

Foto No 2. Accediendo a la nube Digitalocean para crear serividores a cada estudiante

Foto No 2. Accediendo a la nube Digitalocean para crear serividores a cada estudiante

ANEXO D

Guía de despliegue de una aplicación web JAVA en Glassfish

Instrumento:	Guía de despliegue de aplicaciones web java, postgresql, Glassfish de Experimento
Unidades experimentales:	Estudiantes de Ingeniería en sistemas
Investigador:	Edgar Córdova
Objetivo:	Desplegar una aplicación web en un servidor IasS remoto

Instrucciones para el estudiante:

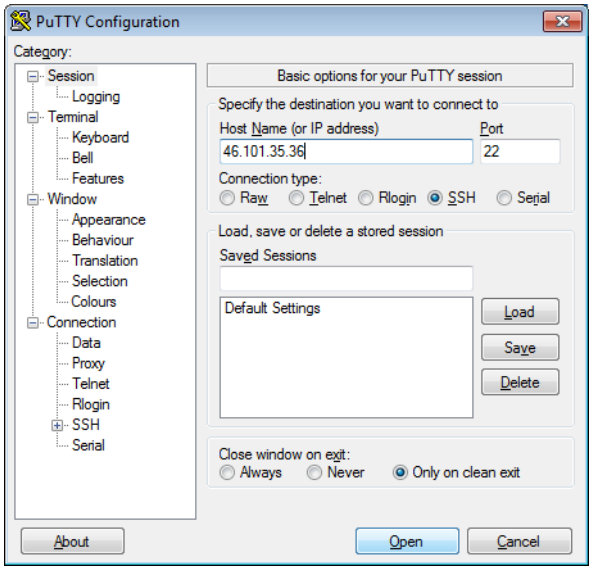
- Si debe realizar otras actividades enviar mensaje de "detener" para pausa el tiempo
- Copiar y pegar en su herramienta SSH(putty), el texto resaltado en amarillo, el resaltado en verde se elimina

Descargar el archivo .zip del siguiente enlace:
https://www.dropbox.com/s/wq0o01h4ph9vnv0/experimento.zip?dl=0

1. Descomprimir y entrar en la carpeta
2. Si no tiene instalado, instalar pgadmin 3
3. **Enviar mensaje de iniciando**
4. Ejecutar Putty.- (no requiere instalación)
5. Ingresar IP asignada (104.236.100.229) sus datos de acceso son:

```
Droplet Name: UExperimental01
IP Address:   138.197.80.126
Username:     root
Password:     expert2017
posgres:      expert2017
glassfish:    fish2017
```

En todos los casos reemplzar la IP de ejemplo: 46.101.35.36 por `104.236.100.229`



1. Instalación de JDK

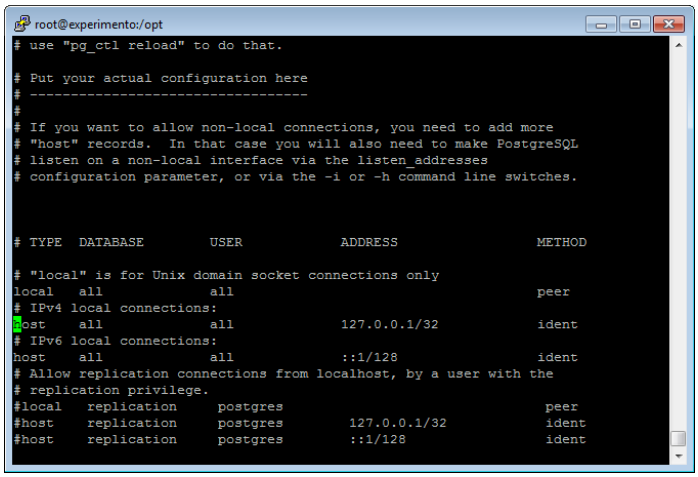
```
[root@UExperimental01 /]#cd /opt

[root@UExperimental01 /]#wget --no-cookies --no-check-certificate --header "Cookie:
gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-cookie"
"http://download.oracle.com/otn-pub/java/jdk/8u60-b27/jdk-8u60-linux-x64.rpm"

[root@UExperimental01 /]#sudo yum localinstall jdk-8u60-linux-x64.rpm
[root@UExperimental01 /]#rm jdk-8u60-linux-x64.rpm
[root@UExperimental01 /]#java -version

[root@UExperimental01 /]#export JAVA_HOME=/usr/java/jdk1.8.0_60/jre
[root@UExperimental01 /]#sudo sh -c "echo export JAVA_HOME=/usr/java/jdk1.8.0_60/jre >>
/etc/environment"
```

2 Instalación de Postgresql 9.4

```
[root@UExperimental01 /]#rpm -Uvh http://yum.postgresql.org/9.4/redhat/rhel-6-
x86_64/pgdg-redhat94-9.4-2.noarch.rpm

[root@UExperimental01 /]#yum update
[root@UExperimental01 /]#yum install postgresql94-server postgresql94-contrib
[root@UExperimental01 /]#service postgresql-9.4 initdb
[root@UExperimental01 /]#service postgresql-9.4 start
[root@UExperimental01 /]#chkconfig postgresql-9.4 on
[root@UExperimental01 /]#su - postgres

-bash-4.1$psql
postgres=# \password postgres
Enter new password:expert2017
Enter it again:expert2017

postgres=# CREATE EXTENSION adminpack;CREATE EXTENSION
postgres=#\q

$ createuser desarrollador
$ createdb pgbasedev

$ psql

psql (9.4.0)
Type "help" for help.

postgres=# alter user desarrollador with encrypted password 'expert2017';
ALTER ROLE
```

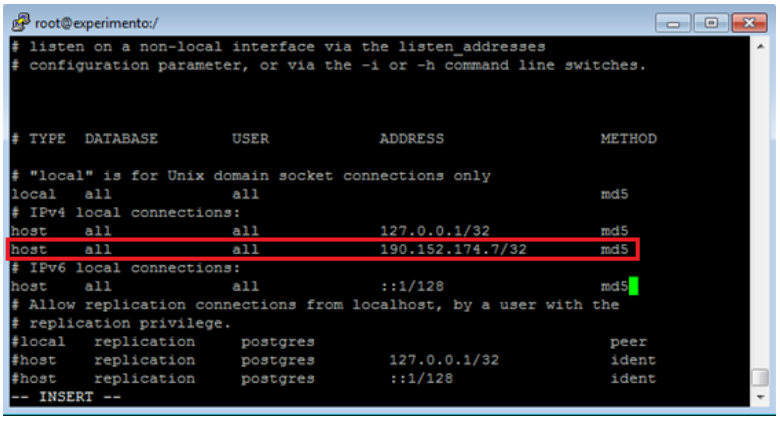
```
postgres=# grant all privileges on database pgbasedev to desarrollador;
GRANT
postgres=#\q
postgres=#exit

[root@UExperimental01 /]#vi /var/lib/pgsql/9.4/data/pg_hba.conf
```

Debemos proporcionar permisos de acceso remoto en el archivo pg_hba.conf, debemos ubicarnos al final de este archivo.



En el navegador accedemos al sitio www.cualesmiip.com para saber cuál es la IP real en Internet, para tener acceso remoto al servidor.

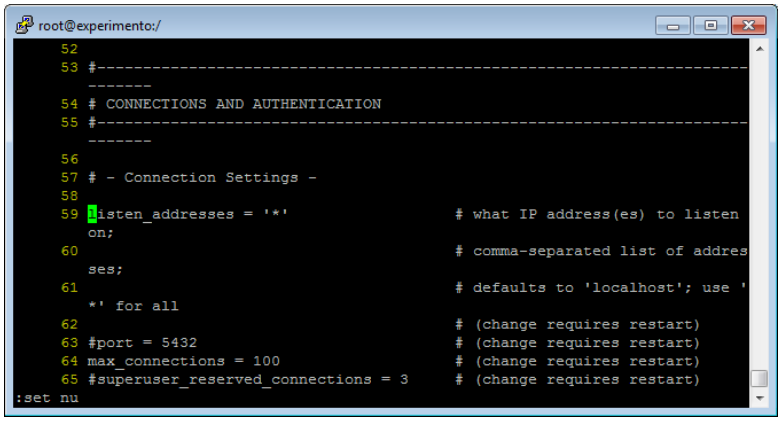
Colocamos la IP obtenida creando una nueva línea host como se indica en la figura debajo. La máscara sigue siendo la misma (/32).



Guardar y salir con la tecla ESC y luego: wq
Luego abrimos otro archivo:

```
[root@UExperimental01 /]#vi /var/lib/pgsql/9.4/data/postgresql.conf
```

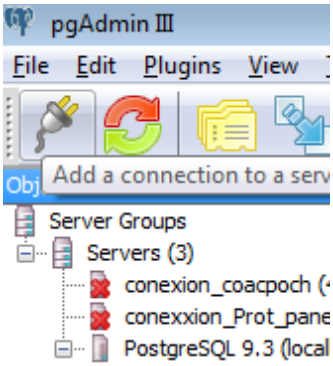
(Para activar números de linea damos ESC, luego: set nu)
Cambiamos la linea #listen_addresses borrando el numeral #
También activamos la linea 63 #port = 5432 borrando #



Guardar y salir con la tecla ESC y luego :wq

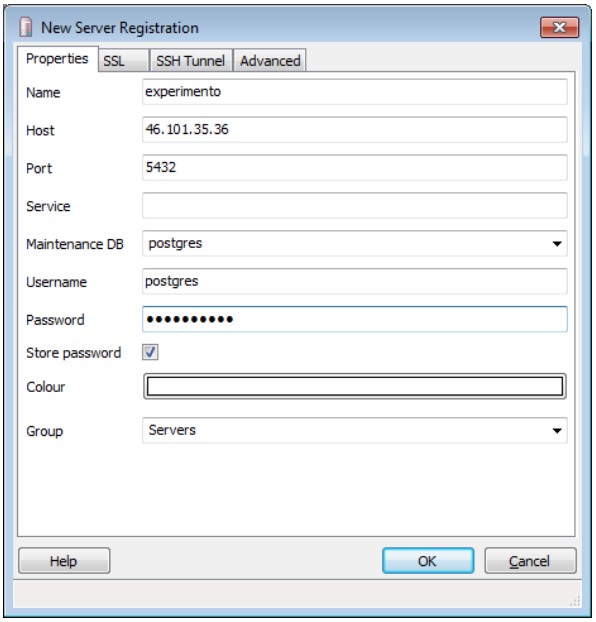
```
[root@UExperimental01 /]#setsebool -P httpd can network_connect_db 1
[root@UExperimental01 /]#service postgresql-9.4 restart
```

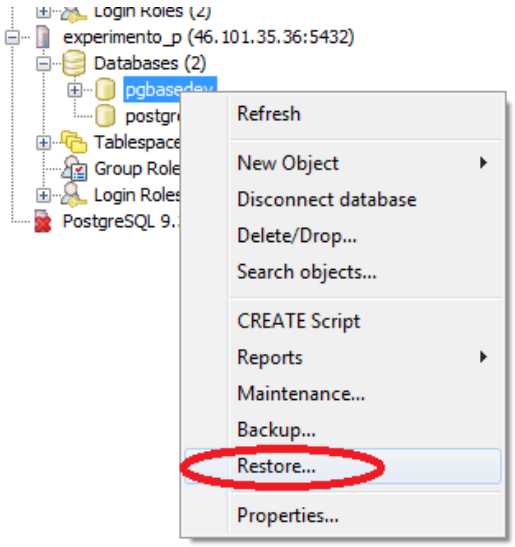
Abrir pgadmin
Crear nueva conexion

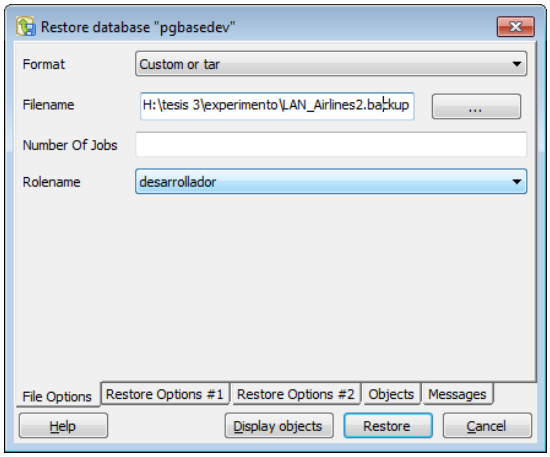


La clave es expert2017

El host es: 104.236.100.229





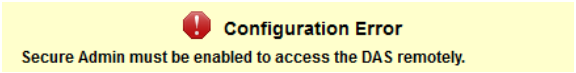


Instalar glassfish

```
[root@UExperimental01 /]#wget http://download.java.net/glassfish/4.1/release/glassfish-4.1.zip
[root@UExperimental01 /]#yum install unzip
```

```
[root@UExperimental01 /]#unzip glassfish-4.1.zip

[root@UExperimental01 /]# sudo useradd --system glassfish -d /opt/glassfish4/
[root@UExperimental01 /]# sudo chown -R glassfish glassfish4

[root@UExperimental01 /]#sudo chmod -R +x glassfish4/bin
[root@UExperimental01 /]#sudo chmod -R +x glassfish4/glassfish/bin
[root@UExperimental01 /]#cd glassfish4
[root@UExperimental01 /]#sudo -u glassfish bin/asadmin start-domain domain1
Waiting for domain1 to start ...........
Successfully started the domain : domain1
domain  Location: /opt/glassfish4/glassfish/domains/domain1
Log File: /opt/glassfish4/glassfish/domains/domain1/logs/server.log
Admin Port: 4848
Command start-domain executed successfully.
[root@UExperimental01 /]#
[root@UExperimental01 /]# sudo -u glassfish bin/asadmin  change-admin-password
Enter admin user name [default: admin]>admin
Enter the admin password>
Enter the new admin password>fish2017
Enter the new admin password again>fish2017
Command change-admin-password executed successfully.
[root@UExperimental01 /]#
```

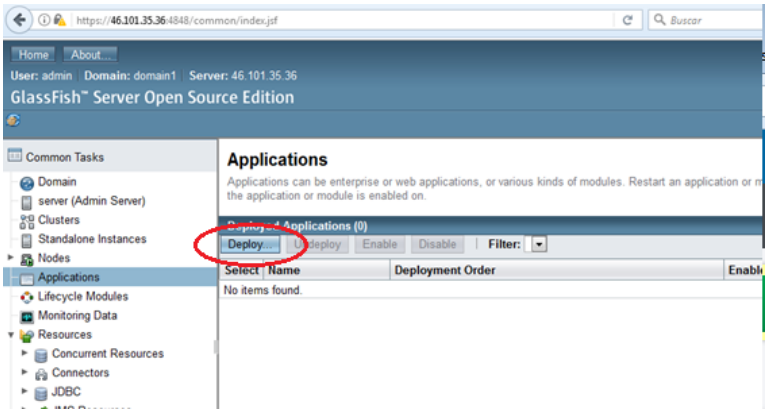
Escribimos la url del servidor en el navegador http://104.236.100.229:4848



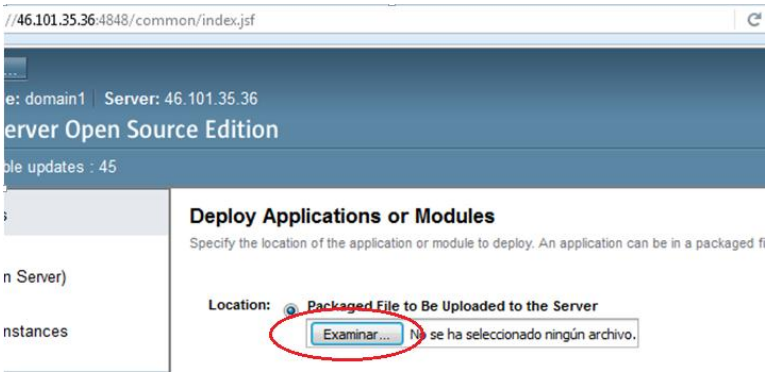

```
[root@UExperimental01 glassfish4]# sudo -u glassfish bin/asadmin enable-secure-admin
Enter admin user name>admin
Enter admin password for user "admin">fish2017
You must restart all running servers for the change in secure admin to take effect.
Command enable-secure-admin executed successfully.

[root@UExperimental01 /]#cd ..
[root@UExperimental01 /]#chmod 777 -R glassfish4
[root@UExperimental01 /]#reboot

[root@UExperimental01 /]#cd /opt
[root@UExperimental01 /]#cd glassfish4
[root@UExperimental01 /]#sudo -u glassfish bin/asadmin start-domain domain1
```

Escribimos nuevamente la url del servidor en el navegador http://104.236.100.229:4848
User name:admin
Pasword:fish2017

DAmos clic en Deploy

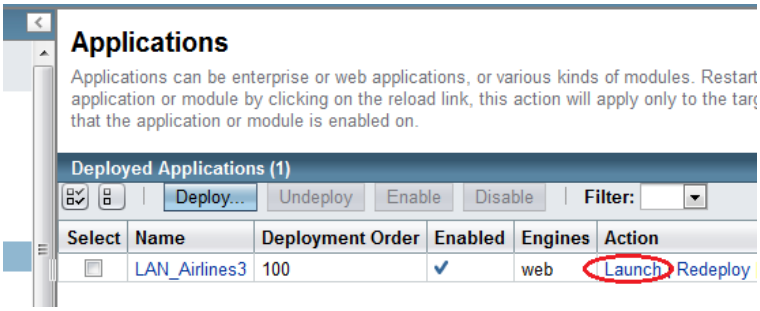


Ahora buscamos el archivo con extensión .war en el archivo LANAirline.zip dentro de la carpeta "disp.", de la aplicación que queremos desplegar

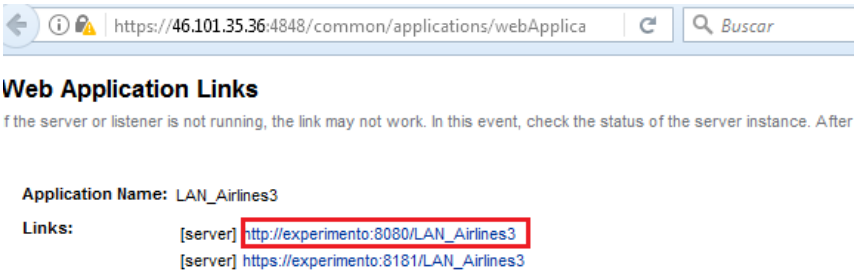


Una vez localizada damos clic en la parte superior derecha en OK

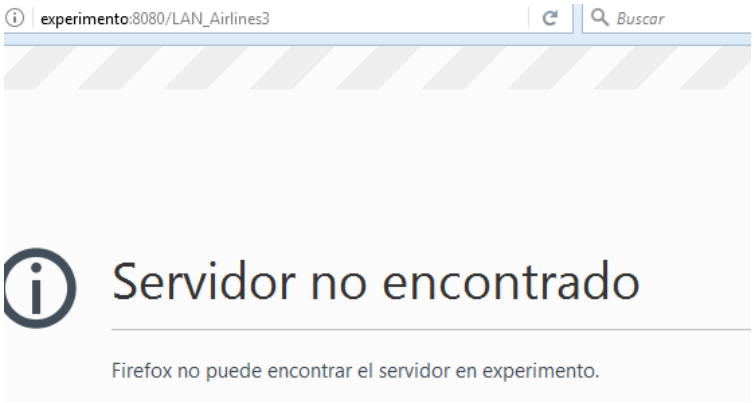
La aplicación ya está desplegada, para comprobar damos clic en lanzar(launch)



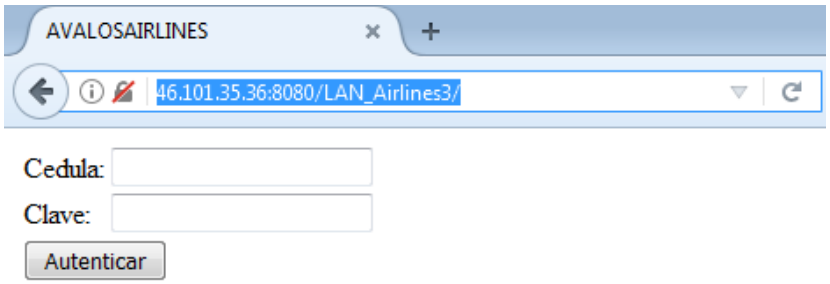
Clic en el primer link



Como no tenemos configurado ningún nombre de dominio la URL no encuentra el servidor, pero sirve para reemplazar la palabra experimento en la barra de direcciones por la IP del servidor remoto sobre el cual estamos trabajando 104.236.100.229



Una vez reemplazado por la IP 104.236.100.229, accederemos a la aplicación desplegada



Para finalizar enviar mensaje con la URL de la aplicación desplegada funcionando.