ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

"DESARROLLO DE PRUEBAS UNITARIAS AUTOMÁTICAS PARA EL SISTEMA WEB ESCOLÁSTICO DEL INSTITUTO TECNOLÓGICO SUPERIOR STANFORD"

Trabajo de titulación presentado para optar al grado académico de:

INGENIERO EN SISTEMAS INFORMÁTICOS

AUTOR: PAUCAR ATI EDUARDO FABRICIO

TUTOR: ING. RAÚL ROSERO MIRANDA

Riobamba-Ecuador

2017

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

El Tribunal del Trabajo de Titulación certifica que el: "DESARROLLO DE PRUEBAS UNITARIAS AUTOMÁTICAS PARA EL SISTEMA WEB ESCOLÁSTICO DEL INSTITUTO TECNOLÓGICO SUPERIOR STANFORD", de responsabilidad del señor Eduardo Fabricio Paucar Ati, ha sido minuciosamente revisado por los Miembros del Tribunal del Trabajo de Titulación, quedando autorizada su presentación.

NOMBRE	**FIRMA**	**FECHA**
Ing. Washington Luna Encalada **DECANO FACULTAD INFORMATICA Y ELECTRONICA**	_____	_____
Ing. Patricio Moreno **DIRECTOR ESCUELA INGENIERIA EN SISTEMAS**	_____	_____
Ing. Raúl Rosero Miranda **DIRECTOR DE TRABAJO DE TITULACIÓN**	_____	_____
Ing. Eduardo Villa **MIEMBRO DEL TRIBUNAL**	_____	_____

Yo, Eduardo Fabricio Paucar Ati soy responsable de las ideas, doctrinas y resultados expuestos en este Trabajo de Titulación y el patrimonio intelectual del mismo pertenece a la Escuela Superior Politécnica De Chimborazo.

_____

EDUARDO FABRICIO PAUCAR ATI

DEDICATORIA

Este trabajo se lo dedico a Dios por dame la fuerza y voluntad de seguir adelante día a día, A mis padres por ser ese pilar fundamental que siempre me apoyaron, con su voluntad de amor y cariño. A mi hijo Alan y esposa Wendy por ser las personas más maravillosas de puede existir, ya que con su apoyo moral y esas sonrisas de mi hijo me hicieron luchar para terminar este gran proceso. Sé que no he sido el mejor de los esposos y el padre ejemplar, pero le doy gracias a Dios por darme a una esposa maravillosa y aun hijo hermoso, los amo.

A mis dos hermanas Jesica y Betty, por estar ahí cuando más los necesito por su carisma, corazón, carácter y tenacidad para afrontar nuevos retos de la vida, ya que, con su apoyo, sus palabras alentadoras, hermano algún día terminaras de estudiar me hicieron luchar y no defraudarlas.

Fabricio

AGRADECIMIENTO

Al culminar el presente trabajo quiero agradecer a la Escuela Superior Politécnica de Chimborazo, Facultad de Informática y Electrónica, Escuela de Ingeniería en Sistemas por los conocimientos impartidos a lo largo de mi vida estudiantil y así poder formarme como un profesional.

Así mismo deseo agradecer al Ing. Raúl Rosero Miranda, por ser mi tutor de este trabajo, ya que con su amplio conocimiento, apoyo y guía se pudo culminar el presente trabajo de titulación.

Fabricio

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El objetivo del trabajo fue el desarrollo de pruebas unitarias automáticas para el sistema web escolástico del Instituto Tecnológico Superior Stanford (SACS), para la identificación de problemas se utilizaron técnicas como la entrevista y la observación, aplicadas al desarrollador del sistema, luego de lo cual se implementó la metodología de desarrollo ágil SCRUM para la creación del sistema de automatización de pruebas unitarias utilizando el framework de Junit, con el lenguaje de programación JAVA, con la tecnología SWING y AWT, además de las librerías de iText e IO, las cuales se comunican con cualquier aplicación de características similares al sistema SACS para implementar en estos automáticamente las pruebas unitarias de acceso a datos y permitir la obtención de información de las mismas, el desarrollo produjo el Sistema de Automatización de Pruebas Unitarias con un total de 986 líneas de código y un peso total de 65.42 Mb, el cual a través de las pruebas correspondientes a cada una de las 6 iteraciones reveló una mejora del 9% en el tiempo invertido en la resolución de conflictos sobre métodos de acceso a datos de un proyecto completo y una reducción promedio del 56% del tiempo invertido en la resolución de conflictos sobre métodos de acceso a datos de una iteración, volviendo así más eficiente el desarrollo de proyectos similares con respecto al tiempo, además al haber sido desarrollado bajo un ambiente de pruebas, obtenemos un código más fiable y limpio, siendo estos buenos indicadores para el desarrollo del proyecto; por los resultados favorables obtenidos se recomienda que el sistema de automatización de pruebas unitarias se convierta en un plugin para los Entornos de Desarrollo más populares como NetBeans y Eclipse.

PALABRAS CLAVE: <TECNOLOGÍA Y CIENCIAS DE LA INGENIERÍA>, <INGENIERÍA DE SOFTWARE>, <SWING>, <JAVA (LENGUAJE DE PROGRAMACIÓN)>, <CONJUNTO DE HERRAMIENTAS DE VENTANA ABSTRACTA (AWT)>, < SCRUM (METODOLOGÍA DE DESARROLLO ÁGIL)>

ABSTRACT

This work aimed to develop an automatic unit tests for the Scholastic Web System of the Stanford Higher Technological Institute (SACS). To identify the problems techniques such as interview and observation, applied to the system developer were used. After that, the SCRUM agile development methodology was implemented for the creation of the unit testing automation system using the Junit framework, with the JAVA programming language, SWING and AWT technology, as well as the iText and IO libraries. Communicating with any application with similar characteristics to the SACS system to automatically implement the unit tests of data access and allow the obtaining of information from them. The development produced the unit testing automation system with a total of 986 lines of code and a total weight of 65.42 Mb. Which, by means of the tests corresponding to each of the six iterations revealed an improvement of 82% in the time invested in the resolution of conflicts over data access methods of a complete project and an average reduction of 56% of the time invested in conflict resolution on data access methods of an iteration. This makes the development of similar projects more efficient with regard to time. Moreover, as it was developed under a test environment, a more reliable and clean code was obtained, being these good indicators for the development of the project; For the favorable results obtained it is recommended that the unit testing automation system becomes a plugin for the most popular Development Environments such as NetBeans and Eclipse.

KEYWORDS: <TECHNOLOGY AND ENGINEERING SCIENCES>. <SOFTWARE ENGINEERING>. <SWING> <JAVA (PROGRAMMING LANGUAGE)>. <SET OF ABSTRACT WINDOW TOOLS (AWT)>. < SCRUM (AGILE DEVELOPMENT METHODOLOGY)>.

INTRODUCCIÓN

Actualmente, en el Ecuador, así como en la mayoría de países en vías de desarrollo, las instituciones de educación superior se encuentran en constante evaluación por parte de entidades de gobierno, por ejemplo, el CEAACES y el SNIESE, además las tendencias a nivel mundial se enfocan a la automatización de procesos y la accesibilidad a la información.

En este entorno, se tiene al Instituto Tecnológico Superior Stanford, el cual se ve sujeto a todos los reglamentos y disposiciones de las entidades reguladoras correspondientes, por lo cual sus autoridades deciden automatizar sus procesos para mantener la información actualizada, así como disponible en cualquier ubicación geográfica a través de internet y sus mapas.

Al ser una institución educativa, el pilar fundamental sobre el que se edifica son los estudiantes y su relación enseñanza aprendizaje con los docentes, misma que genera los procesos escolásticos del negocio, por lo cual se toma este como el primer proceso a automatizar como institución.

Dando inicio así del desarrollo de Sistema Web Escolástico para el Instituto Tecnológico Superior Stanford, mismo que al estar basado en información sumamente sensible como son las calificaciones de los estudiantes, decide dar seguimiento a los métodos de acceso a datos para evitar incidencias en el manejo de los mismos.

En este entorno, se hace evidente que, al utilizar pruebas unitarias para los métodos relacionados al acceso a datos, se requieren demasiados recursos tanto económicos como humanos dado el volumen de métodos a analizar.

Así mismo ya analizados los métodos, los mismos generan otra gran cantidad de esfuerzo al ser comparados y estudiados entre sus diferentes versiones dada la naturaleza iterativa del desarrollo del sistema escolástico con la metodología SCRUM, siendo estos dos los procesos a automatizar para optimizar el tiempo invertido en la verificación de los métodos orientados al acceso a datos del Sistema Escolástico.

Tomando en cuenta lo antes mencionado, se plantea la relación del DESARROLLO DE PRUEBAS UNITARIAS AUTOMÁTICAS PARA EL SISTEMA WEB ESCOLÁSTICO DEL INSTITUTO TECNOLÓGICO SUPERIOR STANFORD, con la idea de convertirse en una herramienta para el Ahorro de recursos durante el desarrollo de sistemas similares orientado a la

verificación de los métodos relacionados al acceso a datos.

El presente trabajo se divide en 3 capítulos, los cuales se detallan a continuación:

Capítulo I: En este se describirá el problema enfatizándose en qué se va a hacer, porque se va a hacer, como se va a hacer y con qué herramientas, además se abarcarán las definiciones y conceptos sobre todas las herramientas, técnicas y tecnologías utilizadas durante el desarrollo del proyecto.

Capítulo II: En este se explicarán los procesos involucrados en la generación de las pruebas unitarias sobre los métodos con relación al acceso a datos y como automatizarlas, así como el funcionamiento de la metodología de desarrollo (SCRUM), mismos que se aplicaron durante el Capítulo III para obtener el producto final.

Capítulo III: En este se trata sobre la utilización de la metodología, herramientas, técnicas y tecnologías descritas en los capítulos anteriores enfatizando en cada uno de los productos obtenidos, y el uso de los mismos en el análisis e interpretación de resultados para comprobar el cumplimiento de los objetivos planteados para el proyecto.

Planteamiento del Problema

Antecedentes

Desde el punto de vista operativo, las calificaciones son el punto fundamental del negocio, dado que las mismas implican el avance del usuario (estudiante) dentro del negocio (Carrera Superior) y todas sus correspondientes motivos o consecuencias, motivo por el cual un mal manejo de los mismos por parte del sistema desarrollado o un error en el manejo y entrega de los mismos involucrarían problemas de índole legal al negocio, por lo tanto se decide asegurar el buen manejo de estos por parte del Sistema Escolástico utilizando como herramientas las pruebas unitarias durante el desarrollo del mismo.

En este contexto, este proceso ha demostrado ser bastante costoso a nivel económico y personal dado que por cada iteración del desarrollo existe una cantidad excesiva de métodos a analizar, luego de lo cual requiere un cruce de información con los resultados de las iteraciones anteriores, para posteriormente apenas iniciar con las correcciones en las incidencias encontradas, luego de lo cual vuelve a ser necesaria una nueva verificación contra los métodos recién modificados para

asegurar que la corrección haya eliminado la incidencia.

Como se puede evidenciar en el árbol de procesos anterior, este ha demostrado ser bastante costoso a nivel económico y personal dado que por cada iteración del desarrollo existe una cantidad excesiva de métodos a analizar, luego de lo cual requiere un cruce de información con los resultados de las iteraciones anteriores, para posteriormente apenas iniciar con las correcciones en las incidencias encontradas, luego de lo cual vuelve a ser necesaria una nueva verificación contra los métodos recién modificados para asegurar que la corrección haya eliminado la incidencia.

Cabe mencionar que además de los problemas técnicos encontrados, también se evidencian problemas humanos y personales que no se pueden solucionar a través de la elaboración de una herramienta software, por lo cual el presente desarrollo se orientará a los problemas técnicos, que por su naturaleza pueden ser automatizados, como son:

- El proceso es excesivamente repetitivo, es decir, la generación de las pruebas unitarias y la revisión de los resultados obtenidos.
- Una cantidad excesiva de métodos a analizar, tomando en cuenta que el proceso de análisis es similar en todos los métodos.
- Información poco confiable, dada las diferentes fuentes de información dado que no siempre la misma persona realizará el análisis ni generará de la misma manera la información por su condición humana.
- Información inoportuna, dado que el proceso de análisis es sumamente costoso en cuanto a esfuerzo.

En consecuencia, se detectan consecuencias a las que se llega por incurrir en los problemas anteriormente mencionados, las cuales incluyen, pero no se limitan a:

- Una mala gestión de recursos económicos dado que el personal de desarrollo incurre en un costo a causa de un proceso repetitivo automatizable.
- Incumplimiento en plazos establecidos dado que el análisis de incidencias muy frecuentemente conlleva a realizar el mismo más de una vez para comprobar los nuevos métodos.
- Mala gestión de Recursos Humanos dado que una persona podría invertir el tiempo en otra tarea mientras la información se obtiene de un proceso automatizado.

- Información inoportuna, dado que la versión a analizar ya es lanzada.

En consecuencia, para ayudar al proceso, se propone que el sistema a desarrollar, por su naturaleza homogenice las fuentes de información, además de homogenizará la misma para su fácil y oportuna interpretación dado que se encargará de manera automática de los procesos repetitivos.

Formulación del problema

¿Cómo optimizar la implementación de pruebas unitarias automáticas orientadas al acceso a datos el desarrollo de Sistema Web Escolástico del Instituto Superior Stanford, orientado a optimización de recursos humanos y económicos?

Sistematización del problema

¿Cómo ha reducido el tiempo requerido en el planteo de los casos de prueba la herramienta automatizada?

¿Involucra la implementación de la herramienta una homogenización de las fuentes de información?

¿Cómo ha optimizado el tiempo invertido en la detección de incidencias en cuanto al acceso a datos el proceso automatizado?

Justificación

En este apartado se explicarán los factores reglamentarios, así como los fundamentos conceptuales que apoyan al presente desarrollo y la orientación que tomará.

Justificación teórica

Basados en el reglamento de acreditación del CEAASES (R MODELO GENERAL PARA LA EVALUACIÓN DE CARRERAS CON FINES DE ACREDITACIÓN, 2011), se necesita dar

Seguimiento a la gran mayoría de procesos que se desarrollan tanto en el proceso educativo, son las cuales se desarrollan las avaluaciones de las universidades.

En este entorno, tomaremos como ejemplo al Instituto Tecnológico Superior Stanford, el cual, estar sujeta al reglamento vigente de acreditación, se ve actualmente inmersa en diferentes problemas con respecto al seguimiento de los procesos necesarios.

Como pilar fundamental de los procesos educativos tenemos la correspondiente asignación y consulta de calificaciones de los estudiantes en sus diferentes periodos, mismo que será optimizado mediante la implementación del sistema web escolástico para la institución, proceso en el cual se ha detectado diversos problemas tratados en el punto antecedentes mismos que motivan el presente desarrollo de pruebas unitarias automáticas.

El proceso de aseguramiento de los datos almacenados en la base de datos para la correcta utilización de los mismos conlleva la comprobación de los métodos que acceden a los mismos para confirmar que ningún proceso realizado a futuro interfiera de manera inesperada en datos almacenados mediante métodos anteriores, dado que en caso de esta situación se podría perjudicar de alguna manera a los estudiantes dado que este proceso escolástico es el que maneja las calificaciones que son el pilar fundamental del proceso del negocio.

En relación a lo anterior mencionado, a nivel de país, cabe mencionar al artículo 81 de la Constitución Política de la República, que garantiza el derecho a acceder a las fuentes de información, como mecanismo para ejercer la participación democrática, también se debe mencionar el artículo 19 del Pacto Internacional de Derechos Civiles y Políticos así como el artículo 13 de la Convención Interamericana de Derechos Humanos, los cuales garantizan la calidad de información a la que se accede debe ser verídica y confiable, por lo cual es importante que los datos almacenados en la base de datos cumplan todas estas características.

Por los problemas mencionados, se propone automatizar el proceso de verificación de incidencias en el acceso a datos por parte de métodos orientados a evitar inconsistencias de datos y optimizar el tiempo de búsqueda, verificación y control de los mismos.

Justificación aplicativa

Para mejorar los procesos mencionados anteriormente, el sistema analizará el código fuente del proyecto en ese momento, la cual el código nos facilita en instituto Superior Stanford , con lo cual ubicará automáticamente los métodos que interactúen con la base de datos y por cada uno de estos creará el caso de prueba correspondiente para definir qué campos de la base de datos

fueron afectados por la ejecución de dicho método, además este "test", contendrá código para la retroalimentación y levantamiento de información misma que contendrá los campos afectados por la prueba así como el tiempo invertido en la misma.

Para lo antes mencionado el proyecto se dividirá en 3 módulos.

Constará con un módulo para el análisis del código y la detección de métodos que interactúan con la base de datos.

Para el proceso de generación automática de las pruebas unitarias de los métodos que interactúen con la base de datos se definirá el módulo de generación.

Para el levantamiento de información y datos relacionados a las incidencias se definirá el módulo de gestión.

Módulos Sistema

MÓDULO	DESCRIPCIÓN
Análisis	El sistema incluirá un módulo encargado de analizar el código existente del proyecto en busca de los métodos que interactúen con la base de datos para la posterior generación de las pruebas unitarias sobre estos.
Generación	Este módulo permitirá definir automáticamente los casos de prueba necesarios para cada uno de los métodos detectados por el módulo de análisis además de incluir el código requerido para obtener información
Gestión	Enfocado al proceso de levantamiento de información para el análisis de la misma, este módulo se encarga de medir, organizar y dar formato a la información obtenida de la ejecución de las pruebas unitarias.

Realizado por: Eduardo Paucar 2017

Dada la naturaleza del proyecto y su orientación a la fase de desarrollo, el único usuario que manejará el sistema en su momento será definido como desarrollador, dado que solo este podrá tener acceso al código fuente a analizar.

El presente proyecto dará como resultado la obtención de un sistema que permitirá optimizar los siguientes procesos:

- Detección de los métodos que interactúan con la base de datos.
- Generación automática de pruebas unitarias para los métodos detectados.
- Generación del informe de incidencias detectadas durante la ejecución de los casos de

Y las ventajas de su implementación serán:

- Integridad de datos.
- Menor consumo de tiempos en los Procesos de Generación de Pruebas Unitarias y Revisión de Incidencias.
- Menos consumo de tiempos y recursos en el Análisis de Incidencias.
- Accesibilidad a la Información.

OBJETIVOS

Objetivo general

Desarrollar pruebas unitarias automáticas para el sistema web escolástico del Instituto Tecnológico Superior Stanford para optimizar el proceso de aseguramiento de la integridad de datos en la base orientada a las aplicaciones java, con la reducción de tiempo y recursos económicos.

Objetivos específicos

- Definir los requerimientos de la aplicación “TEST GENERATOR”.
- Desarrollar un software que permita generar automáticamente pruebas unitarias de nuestro sistema web escolástico del Instituto Tecnológico Superior Stanford.
- Optimizar el tiempo invertido por el desarrollador en generar las pruebas unitarias para cada uno de los métodos que interactúan con la base de datos.
- Reducir el tiempo requerido para la detección de métodos que acceden a campos de la base de datos.

CAPITULO I

1. MARCO TEÓRICO REFERENCIAL

Como base cabe mencionar el énfasis mostrado actualmente por las empresas hacia la automatización de sus procesos para la optimización de sus recursos.
En este contexto personajes importantes al respecto del uso de tecnologías en los entornos de negocios mencionan ideas puntuales a tener en cuenta, por ejemplo (Iñiguez, 2011) para mejorar en la productividad y optimización de las empresas de deben optimizar nuevos resultados, idea que se puede aplicar también a los generadores de las herramientas para este fin, entre las cuales tenemos el Software.

Destacan, en este entorno como principales problemas a tener en cuenta, la heterogeneidad de la información, problema que el creador de la web contempló y se enfocó en solucionar, "The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect" (Tim Berners-Lee, 2016, párr. 1), idea que se puede aplicar también al micro entorno del desarrollo de software, dado que se menciona, la información debe ser universal y accesible para todo el mundo, en este caso particular para los desarrolladores, mismos que necesitan una información homogénea y comprensible.

Tomando en cuenta que el lenguaje de programación utilizado para el desarrollo del sistema escolástico a analizar es Java, se tiene este como opción, además las opiniones de Damián Pérez Valdés que escribe para MAESTROSDELWEB.com (Valdés, 2007), complementada con las opiniones de Manuel Pereira Gonzales (Gonzales, 2010), quienes mencionan lenguajes de programación basados en sus ventajas y dificultades para darnos un entorno de decisión, mismos que declaran como una herramienta acorde a nuestras necesidades al lenguaje JAVA, enfocándose en puntos como su condición multiplataforma y su comunidad Opensource, por lo cual se decide adoptar este mismo lenguaje para el presente desarrollo.

Otro punto referencial en los análisis antes mencionados, son los Frameworks o componentes para mejorar la productividad, por lo cual se consulta el blog de javahispano, mismo que tiene un análisis comparativo, en el cual menciona a awt, enfatizando.

- Componentes de disponibilidad.
- Facilidad de iniciar.
- Documentación sobre el tema. (javaHispano, 2012).

Para un mejor entendimiento del caso a estudiar se procederá a dar una breve introducción a los elementos mencionados como favoritos para el desarrollo de la solución software, así como de las bases teóricas en que se basará el proyecto y en que se basan los procesos a automatizar.

1.1 Herramientas empleadas para el desarrollo del software

1.1.1 Postgresql

Se hace necesario mencionar este elemento dado que el Sistema Web Escolástico a analizar lo tiene como motor de base de datos.

1.1.2 Definición

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. (rafaelma,2010, párr. 1)

Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarles a otras bases de datos comerciales. (rafaelma,2010, párr. 1)

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multi hilos para garantizar la estabilidad del sistema. Al momento de estar en un proceso un fallo en el sistema no afectará simbólicamente y seguirá en su funcionamiento normal. (rafaelma,2010,párr. 2)

1.1.3 Características

- Es una base de datos 100% ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) (“ACID”,2016,párr. 1)
- Integridad referencial.
- Replicación asincrónica/sincrónica / Streaming replication - Hot Standby
- Commit a 2 fases
- PITR – Recuperación a un punto del tiempo
- Copias de seguridad en caliente (Online/hot backups) (rafaelma,2010,pp.1)

- Unicode. (rafaelma,2010, pp.1)
- Juegos de caracteres internacionales. (rafaelma,2010, pp.1)
- Regionalización por columna. (rafaelma,2010, pp.1)
- Multi-Version Concurrency Control (MVCC). (rafaelma,2010, pp.1)
- Multiples métodos de autentificación. (rafaelma,2010, pp.1)
- Acceso encriptado vía SSL. (rafaelma,2010, pp.1)
- Actualización in-situ integrada (pg_upgrade). (rafaelma,2010, pp.1)
- SE-postgres. (rafaelma,2010, pp.1)
- Completa documentación. (rafaelma,2010, pp.1)
- Licencia BSD. (rafaelma,2010, pp.1)
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit. (Oracle, 2010)

1.1.4 Límites

- Máximo tamaño base de dato: Ilimitado (Depende de tu sistema de almacenamiento) (rafaelma, 2010)
- Máximo tamaño de tabla: 32 TB. (rafaelma, 2010)
- Máximo tamaño de fila: 1.6 TB. (rafaelma, 2010)
- Máximo tamaño de campo: 1 GB. (rafaelma, 2010)
- Máximo número de filas por tabla: Ilimitado. (rafaelma, 2010)
- Máximo número de columnas por tabla: 250 - 1600 (dependiendo del tipo.) (rafaelma, 2010)
- Máximo número de índices por tabla: Ilimitado. (rafaelma, 2010)

1.1.5 ¿Por qué Postgresql?

Como lo menciona Colcha Víctor en su estudio Análisis Comparativo entre los Motores de Base de Datos Postgresql y Firebird Aplicando al Gobierno Autónomo PostgreSQL obtiene un 77.6% de calificación haciéndolo así la mejor opción para una implementación en nuestro entorno. (Colcha Chimborazo, 2015).

1.1.6 Glassfish

1.1.6.1 ¿Qué es Glassfish?

Este término fue acuñado por sus creadores basados en un pez que es transparente es decir se puede ver sus huesos, esto fue tomado ya que querían que el proyecto fuera transparente además de que utiliza una licencia Open Source, específicamente la licencia Common Development and Distribution License (CDDL) v1.0 y la GNU Public License (GPL) v2. (Serra, 2010, p.96)

1.1.6.2 Para qué sirve Glassfish

Es un servidor de aplicaciones, además de ser una comunidad de consumidores que permite desplegar aplicaciones que han sido especificadas en la plataforma Java EE, fue creado por Su Microsystems. Cuenta con una versión comercial llamada Sun GlassFish Enterprise Server. Al ser una comunidad libre, sus interesados son capaces de incorporan nuevas características, además de realizar testeo para verificar las fallas y lograr una correcta funcionalidad. (Serra, 2010,96)

1.1.6.3 Cómo funciona un servidor de aplicaciones

Otorga funcionalidades built in, lo que evita escribir código fuente. Esto es gracias a que sus componentes se construyen dentro de un ambiente de ejecución virtual llamado dominio de ejecución.

Verifica si el desarrollador tiene los permisos necesarios para invocar a un determinado método dentro de los beans y sus implementaciones. (Serra, 2010, p.97)

1.1.7 Netbeans

Netbeans IDE es un entorno integrado de desarrollo modular con base estándar, desarrollado bajo plataforma Java, además el proyecto Netbeans consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general (framework) para compilar cualquier tipo de aplicación. (Oracle Corporation and/or its affiliates, 2017)

Permite el acoplamiento de los lenguajes de programación con las plataformas de los sistemas

operativos, lo que facilita el diseño y progreso de una aplicación ya sea móvil, web o de escritorio, brindando a los desarrolladores eficacia en cuanto a la creación, reajuste, compilación, refinación, prueba e implantación de las aplicaciones creadas. (Java, 2017)

1.2 Framework

1.2.1 Definición

En el documento "Análisis del rendimiento entre framework Java Server Face (JSF) y glassfish. Caso práctico en el sistema socio económico" el autor menciona:

Framework es un proceso en el cual se tiene como resultado una medida o cuantificación de la velocidad/resultado, en una computadora o sistema digital. Se debe tener muy claro que la palabra rendimiento no implica sólo del microprocesador o algún componente interno de transmisión de datos como equivocadamente se suele pensarse, sino más bien es la suma de todos sus componentes como la memoria, el bus de datos, los diversos dispositivos y su software (Villa Piray, 2014).

Dado que el framework a utilizar únicamente se utilizará en el entorno del desarrollo de la interfaz y que el plan a futuro es convertir el proyecto en un plugin, no se profundizará en este tema.

1.3 Pruebas Unitarias

1.3.1 Definición

Una prueba unitaria es una prueba que comprueba el funcionamiento de un proceso incluido en la aplicación, tanto en las condiciones favorables como en las no favorables (por ejemplo, entradas correctas e incorrectas del usuario para un determinado valor). La realización de pruebas unitarias permite mejorar el desarrollo de aplicaciones web, ya que se reducen los tiempos de depuración y corrección de incidencias (Luisataxi, 2013).

1.3.2 Características

Las pruebas unitarias se tienen que poder ejecutar sin necesidad de intervención manual. Esta característica posibilita que podamos automatizar su ejecución. -34- Las pruebas unitarias tienen que poder repetirse tantas veces como uno quiera. Por este motivo, la rapidez de las pruebas

tiene un factor clave. Si pasar las pruebas es un proceso lento no se pasarán de forma habitual, por lo que se perderán los beneficios que éstas nos ofrecen.

Las pruebas unitarias deben poder cubrir casi la totalidad del código de nuestra aplicación. Una prueba unitaria será tan buena como su cobertura de código. La cobertura de código marca la cantidad de código de la aplicación que está sometido a una prueba. Por tanto, si la cobertura es baja, significará que gran parte de nuestro código está sin probar. Las pruebas unitarias tienen que poder ejecutarse independientemente del estado del entorno. Las pruebas tienen que pasar en cualquier ordenador del equipo de desarrollo.

La ejecución de una prueba no puede afectar la ejecución de otra. Después de la ejecución de una prueba el entorno debería quedar igual que estaba antes de realizar la prueba. Las diferentes relaciones que puedan existir entre módulos deben ser simuladas para evitar dependencias entre módulos. Es importante conocer claramente cuál es el objetivo del test.

Cualquier desarrollador debería poder conocer claramente cuál es el objetivo de la prueba y su funcionamiento. Esto sólo se consigue si se trata el código de pruebas como el código de la aplicación. Es importante tener en cuenta que, aunque estas son las características de una buena prueba, no siempre será posible ni necesario cumplir con todas estas reglas y será la experiencia la que nos guiará en la realización de las mismas (Luisataxi, 2013).

1.3.3 Beneficios

Con las pruebas unitarias todos ganan. La vida de desarrollador será mucho más fácil, ya que la calidad de su código mejorará, se reducirán los tiempos de depuración y la corrección de incidencias y por tanto el cliente estará mucho más contento porque la aplicación hace lo que él quiere que haga, por lo que ha pagado.

Las pruebas fomentan el cambio y la refactorización. Si consideremos que nuestro código es mejorable podemos cambiarlo sin ningún problema. Si el cambio no estuviera realizado correctamente las pruebas nos avisarán de ello.

Seguramente la frase “si funciona no lo toques” a más de uno les resultará familiar. Si hubiera pruebas unitarias, no sería necesario pronunciarla. Se reducen drásticamente los problemas y tiempos dedicados a la integración.

En las pruebas se simulan las dependencias lo que nos permite que podemos probar nuestro código sin disponer del resto de módulos. Por experiencia puede decir que los procesos de integración son más de una vez traumáticos, dejándolos habitualmente para el final del proyecto.

La frase "sólo queda integrar" haciendo referencia a que el proyecto está cerca de terminar suele ser engañosa, ya que el periodo de integración suele estar lleno de curvas. Las pruebas nos ayudan a entender mejor el código, ya que sirven de documentación. A través de las pruebas podemos comprender mejor qué hace un módulo y que se espera de él. Nos permite poder probar o depurar un módulo sin necesidad de disponer del sistema completo.

Aunque seamos los propietarios de toda la aplicación, en algunas situaciones montar un entorno para poder probar una incidencia es más costoso que corregir la incidencia propiamente dicha. Si partimos de la prueba unitaria podemos centrarnos en corregir el error de una forma más rápida y lógicamente, asegurándonos posteriormente que todo funciona según lo esperado (Universidad de Alicante, 2012).

Para el mundo Java hay unos estándares open source para el desarrollo de pruebas unitarias para los componentes desarrollados en lenguaje Java: JUnit. El framework JUnit se extiende a menudo con otra serie de frameworks que realizan pruebas unitarias más específicas, insertándose a menudo como plugins en los entornos de desarrollo (Universidad de Alicante, 2012).

1.4 Java

1.4.1 Definición

Es considerada como una tecnología enfocada para el desarrollo web, móvil, en la cual es una herramienta eficaz al momento de desarrollar una aplicación, en esta tecnología se puede crear páginas web, aplicaciones orientadas para la web que se ejecuta en un explorador ya sea Microsoft explore etc. (Java, 2016)

1.4.2 Características

La corporación Oracle describe a Java con las siguientes características:

Java ha sido probado, ajustado, ampliado y probado por toda una comunidad de desarrolladores, arquitectos de aplicaciones y entusiastas de Java. Java está diseñado para

permitir el desarrollo de aplicaciones portátiles de elevado rendimiento para el más amplio rango de plataformas informáticas posible. (Java, 2016)

Al poner a disposición de todo el mundo aplicaciones en entornos heterogéneos, las empresas pueden proporcionar más servicios y mejorar la productividad, las comunicaciones y colaboración del usuario final y reducir drásticamente el costo de propiedad tanto para aplicaciones de usuario como de empresa. Java se ha convertido en un valor impagable para los desarrolladores, ya que les permite: (Java, 2016)

- Escribir software en una plataforma y ejecutarla virtualmente en otra
- Crear programas que se puedan ejecutar en un explorador y acceder a servicios Web disponibles
- Desarrollar aplicaciones de servidor para foros en línea, almacenes, encuestas, procesamiento de formularios HTML y mucho más
- Combinar aplicaciones o servicios que utilizan el lenguaje Java para crear aplicaciones o servicios con un gran nivel de personalización.

Escribir aplicaciones potentes y eficaces para teléfonos móviles, procesadores remotos, micro controladores, módulos inalámbricos, sensores, gateways, productos de consumo y prácticamente cualquier otro dispositivo electrónico. (java, 2016)

1.5 Metodología SCRUM

1.5.1 ¿Qué es SCRUM?

Es un marco de trabajo para lograr que problemas complejos, así como adaptativos puedan ser abordados y así poder conseguir un producto de alto valor de manera productiva y creativa, utilizada desde principios de los 90.
Cabe mencionar que SCRUM no es un proceso o una técnica para construir productos, sino un marco de trabajo dentro del cual se pueden emplear varios procesos y técnicas. (Schwaber, y otros, 2016).

1.5.2 Teoría de SCRUM

Se basa en el control de procesos empírico, es decir que el conocimiento procede de la experiencia y fomenta el tomar decisiones de esta misma manera.
Soporte de 3 pilares fundamentales:

- Transparencia: Se define un estándar común

- Inspección: Se debe inspeccionar frecuentemente, (no tanto como para interferir en el trabajo) los artefactos y el progreso hacia un objetivo para detectar variaciones indeseadas, de preferencia lo deben hacer expertos.
- Adaptación: Si se detecta una variación considerable, de debe ajustar el producto o material, debe hacerse cuanto antes.

CAPITULO II

2 MARCO METODOLÓGICO

2.1 Tipo de investigación

Es una investigación de tipo aplicativa, ya que permiten aplicar metodologías de desarrollo de software, estándares, características en la cual nos enfocaremos al desarrollo de un sistema automático de pruebas unitarias para java, enfocando principalmente en los desarrolladores de aplicaciones softwares.

2.2 Métodos y técnicas

2.2.1 Metodología SCRUM

2.2.1.1 Definición

Se aprecia como una metodología ágil y flexible, ya que define un proceso de desarrollo iterativo e incremental que puede ser aplicado a cualquier producto software o la gestión de diferentes actividades complejas, proporcionando una sólida integración entre los equipos de desarrollo. Lo que permite enlazarlo con la participación activa de los clientes, aumentando que los requisitos y las peticiones del cliente se entiendan más rápidamente. (Bissi, 2007, pp.3-6)

2.2.1.2 Equipo SCRUM (Scrum Team)

Se definen como auto-organizados y multifuncionales y se diseñan para optimizar la flexibilidad, la creatividad y la productividad. (Schwaber, y otros, 2016)

Antes de iniciar con la descripción de los miembros del equipo de SCRUM vale la pena mencionar algunas definiciones utilizadas.

Gestión de la Lista del Producto (Product Backlog)

- ✓ Mencionar de manera óptima los requerimientos del Producto a realizar;

- ✓ Tener en claro cómo se encuentra las listas del producto a realizar para un óptimo desempeño de la misma.
- ✓ Trabajar de mejor manera en el equipo de trabajo, cliente-desarrollador.

- ✓ En la lista de requerimientos del producto sea transparente, entendible para el usuario y el desarrollador.
- ✓ El equipo de desarrollo entienda los requerimientos que cuentan dentro de la lista del producto.

2.2.1.3 Sprint Backlog

Es un rango o campo de tiempo que conlleva a desarrollarse un mes o menos en la cual se va desarrollando una parte del producto software la cual conlleva al término del sprint si la duración de desarrollo de cada Sprint consiste en el desarrollo y un incremento del producto para culminar en los tiempos establecidos. Al término del cada Sprint se comienza el desarrollo del Sprint siguiente para optimizar tiempos.

Los miembros del Equipo SCRUM son:

a) El Dueño del producto (Product Owner)

Debe maximizar el valor del producto y el trabajo del Equipo de Desarrollo, además de ser el único responsable de gestionar la Lista del Producto (Product Backlog).

b) Equipo de Desarrollo (Development Team)

Deben realizar el incremento de producto "Terminado" con el potencial de ser puesto en producción al final del Sprint.

c) Scrum Master: Es el encargado del seguimiento de la teoría por parte del equipo. (Schwaber, y otros, 2016)

La **Tabla 1-2** Contiene a las personas que conforman el equipo de desarrollo y correspondiente rol en el mismo.

Tabla 1-2: Equipo Scrum

Rol	**Persona**
El Dueño del producto (Product Owner)	Director de la Escuela de Ingeniería en Sistemas de la ESPOCH Decano de la facultad de informática y electrónica
Equipo de Desarrollo (Development	Eduardo Paucar
Scrum Master	Tutor del Trabajo de

Realizado por: Eduardo Paucar. 2017

2.2.1.4 Eventos SCRUM

- **El Sprint**

Los tiempos establecidos pueden ser de un tiempo de un mes o manos varían por la complejidad de cada sprint en la cual el avance de cada requerimiento se va incrementando al cabo de cada termino, es conveniente que los tiempos sea de un largo estimado ya que posee el mejor funcionamiento para dicho desarrollo y terminar en los tiempos establecidos al finalizar el sprint se da paso al siguiente:

- **Durante el Sprint:**

    - Objetivo del Sprint no realizan cambios significativos para el desarrollo (Sprint Goal).

    - No disminuyen los objetivos de alta calidad;

    - Al avance del producto se puede re planificar teniendo una reunión con el desarrollador y el cliente.

Cada Sprint puede considerarse un proyecto con una duración no mayor de un mes. Al igual que

los proyectos, los Sprints se usan para definirlo. Cada Sprint tendrá su propia definición de lo que se construirá, un diseño y un plan flexible. (Schwaber, y otros, 2016)

- **Planificación de Sprint**

  El trabajo a realizar durante el Sprint se planifica en la Planificación de Sprint. Este plan se crea mediante el trabajo colaborativo del Equipo Scrum completo. La Planificación de Sprint responde a las siguientes preguntas:

  - ¿Qué puede entregarse en el Incremento resultante del Sprint que comienza?
  - ¿Cómo se conseguirá hacer el trabajo necesario para entregar el Incremento? (Schwaber, y otros, 2016)

- **Objetivo del Sprint**

Los objetivos de cada sprint que se van a realizar, es un reto para el sprint. Hace referencia a la meta a lograr durante esta versión del producto, se lo crea durante el punto es ligeramente flexible siempre y cuando no comprometa el desarrollo del proyecto con respecto a la funcionalidad implementada en el Sprint. Los elementos de la Lista del Producto seleccionados ofrecen una función coherente que puede ser el objetivo del Sprint. El objetivo del Sprint puede representar otro nexo de unión que haga que el Equipo de Desarrollo trabaje en conjunto y no en iniciativas separadas. (Schwaber, y otros, 2016)

- **Scrum Diario**

Para el scrum diario se debe realizar reuniones con el cliente y el desarrollador para estar en continua comunicación y observar sus actividades, estas reuniones se realizaras con un tiempo de 30 minutos al día. Se realiza revisando conjuntamente con el cliente y el desarrollador el avance realizado al trabajo teniendo en cuenta desde el último Scrum Diario teniendo en cuenta las proyecciones y su fácil entendimiento y poder contemplarlo antes de pasar al siguiente paso. Las reuniones diarias se los realiza las dos partes implicadas, sin faltar uno al Scrum Diario teniendo en cuenta lugar y hora de los encuentros todos los días para reducir la complejidad. Durante la reunión, cada miembro del Equipo de Desarrollo.
Se basa en 3 líneas fundamentales orientados a lograr el Sprint, cada miembro del equipo define, que hizo, que va a hacer y que le impide seguir, orientándose así a una rendición de cuentas, una

planificación y una solución de conflictos, pero de una manera totalmente compresa y eficiente (Schwaber, y otros, 2016)

- **Revisión de Sprint**

Al final del Sprint se lleva a cabo una Revisión de Sprint para inspeccionar el Incremento y adaptar la Lista de Producto si fuese necesario. Durante la Revisión de Sprint, el Equipo Scrum y los interesados colaboran acerca de lo que se hizo durante el Sprint. Basándose en esto y en cualquier cambio a la Lista de Producto durante el Sprint, los asistentes colaboran para determinar las siguientes cosas que podrían hacerse para optimizar el valor. Se trata de una reunión informal, no una reunión de seguimiento, y la presentación del Incremento tiene como objetivo facilitar la retroalimentación de información y fomentar la colaboración. (Schwaber, y otros, 2)

- **Retrospectiva de Sprint**

El propósito de la Retrospectiva de Sprint es:

  - Inspeccionar cómo fue el último Sprint en cuanto a personas, relaciones, procesos y herramientas.
  - Identificar y ordenar los elementos más importantes que salieron bien y las posibles mejoras.
  - Crear un plan para implementar las mejoras a la forma en la que el Equipo Scrum desempeña su trabajo. (Schwaber, y otros, 2016)

2.2.1.5 Artefactos de SCRUM

- **Producto (Product Backlog)**

En Product Backlog se encuentra de forma ordenada en la cual es un requisito que contiene varios requisitos las cuales sirven para el desarrollo de un producto de software.
El cliente o dueño del producto (Product Owner), es dueño y responsable de lista de requerimientos que se van a desarrollar en el producto, la cual es el absoluto dueño de todo su contenido, disponibilidad, y ordenación.
Una Lista de Producto nunca está completa. El desarrollo más temprano de la misma solo refleja los requisitos conocidos y mejor entendidos al principio. La Lista de Producto evoluciona a medida que el producto y el entorno en el que se usará también lo hacen. La Lista de Producto

es dinámica; cambia constantemente para identificar lo que el producto necesita para ser adecuado, competitivo y útil.
Mientras el producto exista, su Lista de Producto también existe. La Lista de Producto, enumera todo lo que le falta al producto para entregas futuras. Los elementos de la Lista de Producto tienen como atributos la descripción, el orden, la estimación y el valor. (Schwaber, y otros, 2016)

- **Lista de Pendientes del Sprint (Sprint Backlog)**

La Lista de Pendientes del Sprint hace visible todo el trabajo que el Equipo de Desarrollo identifica como necesario para alcanzar el Objetivo del Sprint. La lista de requerimientos de los pendientes del Sprint es un plan con un nivel de detalle suficiente como para que los cambios en el progreso se puedan entender en el Scrum Diario. El Equipo de Desarrollo modifica la Lista de Pendientes del Sprint durante el Sprint y esta Lista de Pendientes del Sprint emerge a lo largo del Sprint.
Esto ocurre a medida que el Equipo de Desarrollo trabaja en lo planeado y aprende más acerca del trabajo necesario para conseguir el Objetivo del Sprint. A medida que el trabajo se ejecuta o se va completando, se actualiza la estimación de trabajo pendiente.
Se eliminan elementos considerados innecesarios. Solo el Equipo de Desarrollo puede cambiar su Lista de Pendientes del Sprint durante un Sprint. La Lista de Pendientes del Sprint es una imagen visible en tiempo real del trabajo que el Equipo de Desarrollo planea llevar a cabo durante el Sprint y pertenece únicamente al Equipo de Desarrollo. (Schwaber, y otros, 2016)

- **Incremento**

Es el conglomerado de todos los Sprints realizados hasta el momento, basados en que cada Sprint es un conjunto utilizable, este conglomerado debe ser un producto funcional hasta el punto en que este cumplido el Product Backlog, independientemente si se utilizará o no, este debe estar en la capacidad de funcionar en el ambiente de producción. (Schwaber, y otros, 2016)

2.3 Herramientas

2.3.1 Herramientas para el desarrollo del software

Netbeans 8.1: Desarrollo de
Generator testing Java: 1.8.0:
Lenguaje de Programación Windows

10: Versión del Sistema Operativo

Postgresql 9.5: Gestor de Base de

datos GlassFish 4.1.1: Servidor web

Junit 2.1: generador pruebas unitarias java.

2.4 Pruebas unitarias Automáticas

Mediante una entrevista no estructurada con el docente tutor y el Desarrollador de Sistema Escolástico en ese momento se define al proceso de análisis de información de pruebas unitarias sobre el acceso a datos como responsabilidad del desarrollador orientado a evitar la mala utilización de recursos tanto humanos como económicos. El **Figura 1-2** representa el diagrama del proceso efectuado para la generación del documento de análisis de incidencias en acceso a datos.

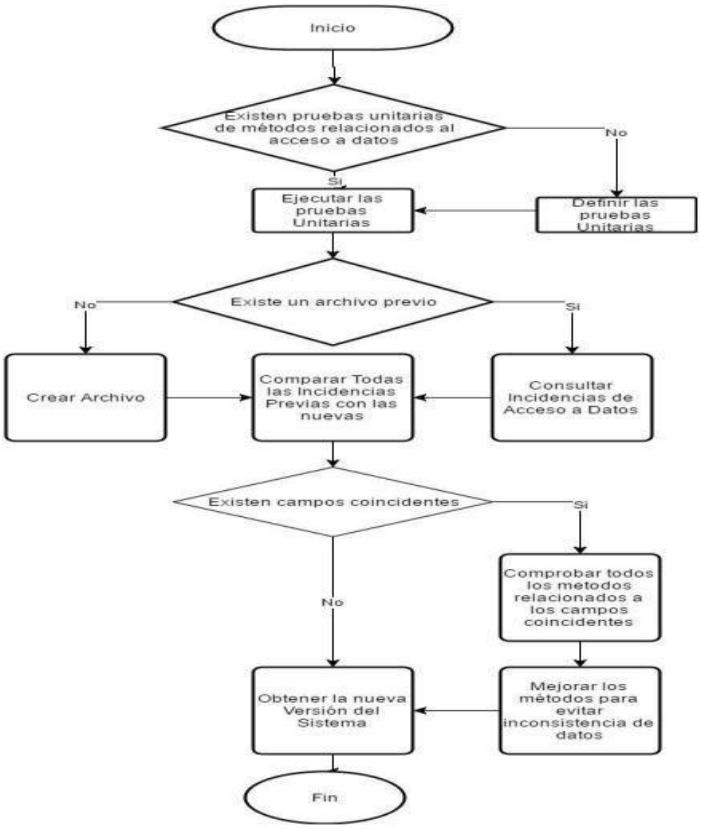

Figura 1-2. Diagrama de proceso de generación del documento de análisis de incidencias.

Realizado por: Eduardo Paucar. 2017

Como se puede apreciar en el gráfico, el proceso depende de que previamente se hayan generado las pruebas unitarias sobre el segmento del código a analizar, en el presente caso a cada una de las iteraciones, por lo cual y para asegurar la homogeneidad de los archivos a analizar, el presente

sistema se encargará de generar las pruebas unitarias de cada una de las clases del segmento del proyecto a analizar.

Las pruebas deben únicamente enfocarse a las clases que acceden a la información almacenada en la base de datos, por lo cual se recomienda el patrón de desarrollo MVC el cual separa en el modelo los métodos requeridos para el presente análisis.

Ya con las pruebas generadas, durante su ejecución se irán tomando las correspondientes medidas de incidencia, para generar el documento de información sobre incidencias según el formato adjunto en el **Anexo B**.

2.4.1 Pruebas de caja blanca

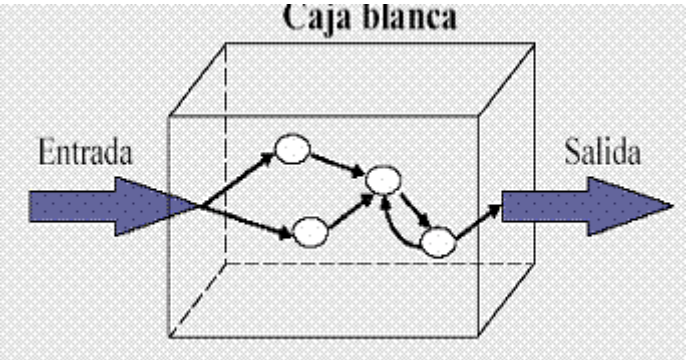

Figura 2-2. Diagrama de proceso pruebas unitarias de tipo caja blanca.

Realizado por: Eduardo Paucar. 2017

En la **figura 2-2** observamos el proceso de pruebas de caja blanca también conocidas como pruebas de caja de cristal o pruebas estructurales se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. El ingeniero de pruebas escoge distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y cerciorarse de que se devuelven los valores de salida adecuados.

Al estar basadas en una implementación concreta, si ésta se modifica, por regla general las pruebas también deberán rediseñarse.

2.5 Generación automática de casos de pruebas

Son especificaciones de secuencias de métodos, en las cuales estas clases pueden ser invocados por otras clases clientes, una secuencia de métodos de una clase pueden describir utilizando una expresión. El concepto fundamental en estas herramientas es el **caso de prueba** (*test case*), y la **suite** de prueba (*test suite*). Los casos de prueba son clases o módulos que disponen de métodos

para probar los métodos de una clase o módulo concreta/o. Así, para cada clase que quisiéramos probar definiríamos su correspondiente clase de caso de prueba. Mediante las suites podemos organizar los casos de prueba, de forma que cada suite agrupa los casos de prueba de módulos que están funcionalmente relacionados. (Casos de prueba., 2014: pp.1)

Las pruebas que se van construyendo se estructuran así en forma de árbol, de modo que las hojas son los casos de prueba, y podemos ejecutar cualquier subárbol (suite).

De esta forma, construimos programas que sirven para probar nuestros módulos, y que podremos ejecutar de forma automática. A medida que la aplicación vaya avanzando, se dispondrá de un conjunto importante de casos de prueba, que servirá para hacer pruebas de regresión. Eso es importante, puesto que cuando cambiamos un módulo que ya ha sido probado, el cambio puede haber afectado a otros módulos, y sería necesario volver a ejecutar las pruebas para verificar que todo sigue funcionando. (Casos de prueba., 2014: pp.2)

2.5.1 Medida de la calidad de los casos

Se generan de acuerdo al sistema a evaluar, realizando un estudio interno de los métodos existentes dentro de la aplicación, generando un número muy grande de casos de pruebas, su calidad y capacidad para detectar fallos que pueden existir en los métodos que accedan a la base de datos.

CAPITULO III

3 RESULTADOS Y DISCUSIÓN.

3.1. DESARROLLO DE PRUEBAS UNITARIAS AUTOMÁTICAS PARA SISTEMA WEB ESCOLÁSTICO DEL INSTITUTO TECNOLÓGICO SUPERIOR STANFORD

3.2 Introducción

Tomando como base todo lo planteado en el presente documento, se procede a la utilización del proceso SCRUM en el desarrollo, planteando así cada punto del presente capítulo como un informe de la experiencia obtenida durante el desarrollo del mismo, así como los problemas y soluciones que se presentaron en este proceso, de la misma manera que la adaptación de la metodología al caso real del presente desarrollo.

3.2.1 Estudio Preliminar

En el Instituto Tecnológico Superior Stanford acreditada como una de las instituciones de educación media, cuenta con varios sistemas web las cuales ayudan al desarrollo y fácil manejo de información de los estudiantes.

El instituto dentro de sus varias aplicaciones web no cuenta con una función principal dentro de su desarrollo que son las pruebas unitarias, las cuales son importantes al momento de acceder a la base de datos, para nuestro caso tomaremos el sistema web escolástico de dicha institución.

En la **figura 3-3** Podemos observar el sistema web escolástico

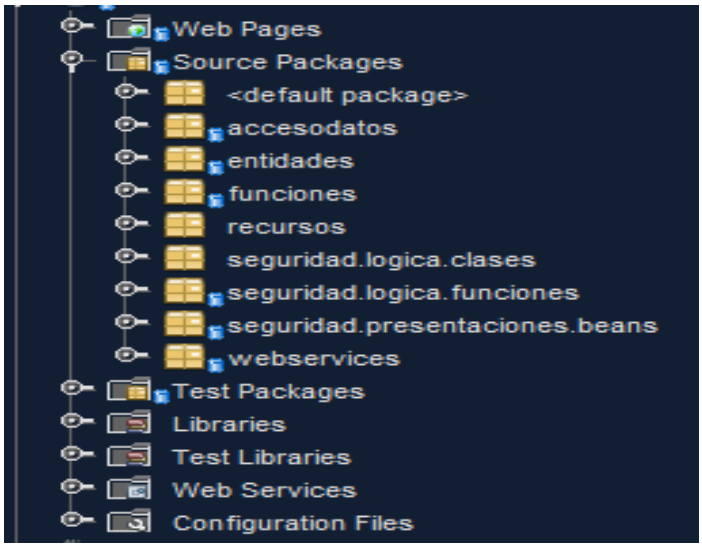

Figura 3-3. Paquetes del web escolástico

Realizado por: Eduardo Paucar. 2017

En la siguiente figura podemos observar que el sistema web escolástico del Instituto Tecnológico Superior Stanford no cuenta con un paquete de pruebas unitarias, dichas pruebas ayudaran al desarrollador a verificar en donde podrá tener problemas a futuro.

3.2.2 Desarrollo del proceso como funciona nuestro Generator testing Proceso

En la **figura 4-3** podemos observar como es el ciclo de vida de nuestra prueba unitaria, al momento de acceder a nuestro método, accederá a nuestra base datos posteriormente se generará automáticamente un método nuevo en nuestro caso una prueba unitaria.

Figura 4-3. Proceso caso de Prueba

Realizado por: Eduardo Paucar. 2017

3.2.3 Estudio de factibilidad

Con el fin de garantizar la inversión tanto de recursos humanos como económicos, se procede a realizar un estudio de la factibilidad y un análisis de riesgos para el desarrollo, con el fin de determinar los costos recursos en los diferentes puntos del desarrollo así como las predicciones de problemas que pueden presentarse durante cada una de las fases del ciclo de vida del proyecto, documentos en los que se concluye que la realización del proyecto es factible, dado que se cuenta o se puede acceder a los recursos necesarios para la correcta finalización del proyecto en los plazos estimados, además de catalogar a los riesgos como manejables y gestionables por las características de los mismos, convirtiendo así a este proyecto en viable, por lo cual se decide poner en marcha sudesarrollo.

El análisis estudio de factibilidad, así como el análisis de riesgos mencionado se encuentran en el manual técnico entregado a la institución adjunto al presente documento como lo evidencia el documento de entrega/recepción que se encuentra en el **Anexo A.**

3.2.4 Fase de planificación

Con el fin de lograr una calendarización y una óptima coordinación de las tareas planteadas, se realiza la planificación donde se elaboran los sprints de acuerdo a la prioridad que tenga cada requerimiento, las metáforas de la aplicación no son solicitadas por el usuario, pero son complemento importante en el desarrollo de la aplicación. La planificación se realizó en Project Profesional 2015. Y está representado mediante un Diagrama de Gantt **Anexo E.**

3.2.5 Requerimientos

Son necesidades del cliente, los cuales definen el funcionamiento del sistema que se pretende desarrollar. Estos requerimientos deben ser recopilados del cliente como una lista llamada Product Backlog establecida por orden de prioridad.

Después de obtener la lista de historias de usuario general del proyecto se realizó el proceso de estimación y priorización para cada una de las historias.
Describe la Pila del Producto, en la que:

- ID es el identificador para cada una de las tareas presentadas.
- Tareas realizadas son las tareas establecidas para cada módulo del sistema.
- Estimación está dada por la relación tiempo-hombre, el tiempo está dado por horas tomando en cuenta que un día laborable es de 8 horas y el trabajo es realizado por una sola persona.

La columna "Prioridad" de la Tabla 2-3 la determinamos según las necesidades de ejecución esta se describen en función de 3 parámetros Alta, Media y Baja, las cuales cuentan con un valor especificado.

Tabla 2-3 Prioridad de Ejecución

Prioridad	**Valor**
ALTA	**3**
MEDIA	**2**
BAJA	**1**

Realizado por: Eduardo Paucar. 2017

3.2.6 Tipos de Roles del Usuario

Dada la naturaleza del proyecto y su fin de proveer una herramienta de ayuda a los desarrolladores como usuarios finales, el presente sistema no presenta métodos de autenticación así que únicamente se definirá al Cliente como Usuario.

Los tipos y roles del usuario se evidencian en la Tabla 3-3.

Tabla 3-3: Tipos y Roles del Usuario

Dada la naturaleza del proyecto y su fin de proveer una herramienta de ayuda a los desarrolladores como usuarios finales, el presente sistema no presenta métodos de autenticación así que únicamente se definirá al Cliente como Usuario.

La **Tabla 3-3** Especificaron de las funciones del Usuario determinado

Rol	Funciones
Usuario	• Establecimiento de parámetros de uso. • Generación de Pruebas unitarias. • Generación de información sobre el acceso a datos.

Realizado por: Eduardo Paucar. 2017

3.2.7 Arquitectura del sistema

Se decide por el uso de la arquitectura a N Capas complementada por la arquitectura cliente servidor, dado que la interacción entre cada una de las partes de MVC se comporta de esta manera tomando en cuenta el Hardware a disponibilidad del desarrollador.

En dependencia del funcionamiento de la aplicación a analizar, las capas estarán distribuidas en un mismo o varios servidores.

En la Capa de Negocio se desplegará la aplicación dividida lógicamente en:

- Vistas. - Formularios de JFrame que permiten la interacción del usuario con la aplicación.
- Controladores. - Clases en Java que contienen la lógica del negocio.
- Funciones. - Clases encargadas de la creación y organización de las sentencias SQL.
- Acceso a Datos. - Clases encargadas de realizar la conexión a la base de datos y las correspondientes consultas, además retornan los conjuntos de resultados.

3.2.8 Artefactos

3.2.8.1 Pila de Producto (Product Backlog)

Por lo mencionado en las tablas anteriores y la naturaleza del negocio se definen 3 Product Owner, encargados la definición de las Historias de Usuario (Requerimientos del Sistema), pero la requerimentación se realizó por parte del Ing. Raúl Rosero en su representación y dado que es un experto en el área a tratar.

Durante la implementación de la técnica de entrevista, para mantener los requerimientos claros y breves, se orientó a no profundizar en descripciones ni procesos para poder cubrirlos correctamente durante el proceso de construcción de la solución.

Definida con los requerimientos del cliente y del representante del Product Owner, la pila del producto consta de 13 funcionalidades específicas a implementar para construir la visión del producto que detallan, se la presenta de manera priorizada, la **Tabla 4-3** muestra la lista derequisitos planteados en el orden que se definen apoyado por el desarrollador en cuanto a la importancia de cada uno.

Tabla 4-3: Product Backlog

Historia	Nombre de la Historia
1	Diseño de interfaces
2	Definición de la arquitectura del sistema.
4	Selección del proyecto a analizar.
5	Detección de los métodos que accedan a la base de datos.
6	Generación automática de pruebas unitarias para los métodos
7	Detección de campos accedidos por un método.
8	Medición del tiempo requerido por una prueba.
9	Medición del tiempo requerido por un conjunto de pruebas.
10	Generación de un informe sobre los resultados obtenidos durante la
11	Detección del estado actual de la base de datos antes de la prueba del método.
12	Detección del estado de la base de datos posterior a la ejecución del método.
13	Identificación de los campos modificados por la ejecución de la prueba en la base de datos.

Realizado por: Eduardo Paucar. 2017

3.2.9 Sprint Backlog

Para determinar el esfuerzo que se necesita en el desarrollo de cada Historia Técnica o de Usuario se empleó el método T-Shirt, para ello se definió las tallas S, M, L y XL.

- ✓ Talla S 2.5 puntos
- ✓ Talla M 5 puntos
- ✓ Talla L 10 puntos
- ✓ Talla XL 15 puntos

Sprint 1 talla L o 2 tallas M o 4 tallas S.

Se define como 1 punto de función a 3 horas de trabajo realizadas en el mismo día, además se establecen los lunes, martes, miércoles, jueves y viernes como días laborables, un Sprint tendrá 3 semanas de duración, por lo que cada Sprint tendrá 15 puntos de función que corresponden a 45 horas de trabajo, teniendo en cuenta que se trabaja 5 días a la semana se tiene un total de 15 días, por lo cual se debe trabajar un total de 3 horas diarias, mismas que se realizarán de 15:00 a 18:00.

Con las historias de usuario como base, se generan 8 pilas del Sprint (Sprint Backlog) considerando agrupar las funcionalidades relacionadas en un mismo Sprint y teniendo en cuenta que la duración de cada Sprint debe ser la misma.

En el Sprint 1 han sido tratadas las historias técnicas mientras que en los 7 Sprint restantes se desarrollaron las funcionalidades requeridas por el usuario obteniendo así un total de 120 puntos de función. En la **Tabla 5-3**, se detallan los Sprint Backlog del proyecto.

Tabla 5-3: Sprint Backlog

Sprint	Historia	Nombre de la Historia	Puntos
1	**MT_01**	Conexión a Base de Datos	2.5
	MT_02	Diseño de interfaces	2.5
	MT_03	Definición de la arquitectura del sistema.	2.5
	HU_01	Selección del proyecto a analizar.	2.5
	HU_02	Detección de los métodos que accedan a la base	5
2	**HU_03**	Generación automática de pruebas unitarias para los métodos de ingreso de datos.	10
	HU_04	Detección de campos accedidos por un método.	5

3	**HU_05**	Medición del tiempo requerido por una prueba.	5
	HU_06	Generación automática de pruebas unitarias para los métodos de consulta de datos.	10
4	**HU_07**	Generación automática de pruebas unitarias para los métodos de consulta de datos.	10
	HU_08	Medición del tiempo requerido por un conjunto	5
5	**HU_09**	Generación automática de pruebas unitarias para los métodos de consulta de datos.	15
6	**HU_10**	Detección del estado actual de la base de datos antes de la prueba del método.	15
7	**HU_11**	Detección del estado de la base de datos posterior a la ejecución del método.	15
	HU_12	Identificación de los campos modificados por la ejecución de la prueba en la base de datos.	10
8	**HU_13**	Generación de un informe sobre los resultados obtenidos durante la ejecución de las pruebas.	5

Realizado Por: Eduardo Paucar. 2017

Para mantener la homogeneidad se planifico en puntos de función y cabe mencionar que se obtuvo una buena primera estimación lo que conllevó a no tener retrasos inesperados durante el desarrollo.

3.2.9.1 *Historias de Usuario*

Se utilizan para definir los requisitos y mantenerlos organizados, se caracterizan por ser escritas por el usuario con un bajo nivel de detalle y en su propia terminología.

Como complemento a los requisitos del usuario, se complementan las Historias de Usuario con las Historias Técnicas, las cuales sirven tanto para el desarrollo del sistema como para su posterior uso.

Se toma la misma definición que utiliza el desarrollador del sistema en otro de sus productos desarrollado con la misma metodología, para cada las historias de usuario el prefijo HU_ Historias Técnicas el prefijo MT_, mismo que se complementará con el número secuencial correspondiente y un identificador (nombre) que exprese de la manera más clara posible el requerimiento.

Esta información se complementará con información sobre la prioridad que tiene este requerimiento y a quien beneficiará, información determinada por el cliente, así como campos técnicos como las observaciones, mismos que son definidos por el desarrollador según su necesidad.

Inicialmente se han identificado 13 historias de usuario además de 3 historias técnicas, estudiando las mismas, en la **Tabla 5-3** se han establecido y organizado.
Como se mencionó anteriormente, cada Historia tanto de usuario como técnica esta descrita por 7 campos como lo sugiere el desarrollador:
ID: Es el campo compuesto por el identificados, sigue lo antes establecido en cuanto a prefijos.
Nombre: Conjuntamente definido con el cliente, especifica de manera puntual lo que se pretende resolver.
Usuario: Para quien es útil y para que lo necesita.
Sprint Asignado: Número del Sprint.
Descripción: en este campo de describiremos paso a paso que se va realizando en nuestro requerimiento.
Puntos estimados: para llenar este campo se realizará el esfuerzo estimado que se va realizando en cada requerimiento.
Puntos reales: se calculan los puntos reales que se van realizando dentro de nuestra historia de usuario.
En la **Tabla 6-3** se presenta una historia de usuario como ejemplo de lo mencionado.

Tabla 6.3: Formato Historia de Usuario

Historia de Usuario	
Número: MT_01	Conexión a Base de Datos
Modificación de historia de usuario: Desarrollador	
Usuario: Desarrollador	**Sprint Asignada:** 1
Prioridad en el Negocio:	**Puntos Estimados:** 2.5
Riesgo en el Desarrollo:	**Puntos Reales:** 2.5
Descripción: Como principal acceso de desarrollador deseo ingresar a la base de datos	
Observaciones: • Se solicita que utilice la tecnología JDBC dado que el sistema escolástico así lo maneja. • Se requiere el diagrama entidad relación de la base terminada del sistema web. • Se requiere el diccionario de datos correspondiente.	

Realizado Por: Eduardo Paucar. 2017

Para utilizar las historias de usuario correspondientes a cada Sprint en el desarrollo del mismo se toman en cuenta 2 consideraciones.

- Las Historias se convierten en tareas al comienzo del Sprint.
- La realización de las tareas de ingeniería de estiman en días ideales

Tareas de ingeniería. - Se las puede entender como las partes necesarias para construir una Historia de Usuario o técnica, llevarán las iniciales TI_ acompañadas con el número secuencial correspondiente, cabe mencionar que se van creando a medida que van apareciendo y sin guardar relación con la fase del proyecto actual sino con la funcionalidad a la que corresponden.
En la **Tabla 7-3** se muestra como ejemplo la tarea de ingeniería TI_05 misma que es parte de la Historia Técnica 1 (MS_01), cada una de las tareas de ingeniería adjuntas en el **Manual Técnico** siguen este modelo que se tomó del desarrollador (Mejia Broncano, 2016) para mantener un formato estándar.

Tabla 7-3: Formato Tarea de Ingeniería

TAREA DE INGENIERÍA	
Historia de Usuario: MS_01 Conexión a Base de Datos	
Número de Tarea: TI_5	**Nombre de Tarea:** Diccionario de datos
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Programador Responsable: Eduardo Paucar	
Descripción: La creación del diccionario de datos con sus respectivos tipos de datos y sus restricciones	
Pruebas de Aceptación; Que el diccionario de datos incluya todas las tablas de la base de datos. Que los tipos de datos especificados en los campos de la base, estén dentro de los soportados por postgresql.	

Realizado Por: Eduardo Paucar. 2017

Pruebas de Aceptación. - Son las validaciones que debe pasar una tarea de ingeniería para poder ser tomada como correcta.
De la misma manera la **Tabla 8-3**, representa un ejemplo de una tarea de ingeniería con el modelo tomado del desarrollador (Mejia Broncano, 2016), en la cual se pueden visualizar los campos que contiene y la función que desempeña.

Tabla 8-3: Formato Prueba de Aceptación

Prueba de Aceptación	
Código: PA_05	**Historia de Usuario:** Conexión a Base de Datos
Nombre: Dentro de diccionarios de datos contemplara, todas las tablas de nuestro proyecto	
Responsable: Eduardo Paucar	
Descripción: el en diccionario de datos debe estar todos los datos que contienen nuestras	
Condiciones de Ejecución: El responsable revisara visualmente que cada una de las tablas estén documentadas en el diccionario de datos.	
Pasos de ejecución: las tablas den estar ordenadas de forma alfabética, de la base de datos que se está desarrollando. Deben estar contempladas todas las tablas en orden.	
Resultado esperado: en el diccionario de datos deben estar todos los documentos o registros	
Evaluación de la prueba: Exitosa.	

Realizado Por: Eduardo Paucar. 2017

Todos los documentos representados en la **Tabla 7-3** y **Tabla 8-3,** tareas de ingeniería y pruebas de aceptación respectivamente, obedecen al mismo diseño y se las puede encontrar detalladas en el **Manual Técnico** del presente desarrollo.

Cabe recalcar que todos los instrumentos generados para el presente desarrollo tienen un programador como responsable, en este caso Eduardo Paucar.

Al inicio del Sprint y siguiendo lo que recomienda la metodología para relacionar los actores con las funciones que desempeña se deberían definir diagramas para apoyo, en nuestro caso al no tener más que 1 actor este proceso se omite**.**

De la misma manera se manejan las tarjetas CRC, mismas que tienen todas como responsable al único desarrollador Eduardo Paucar, por lo que al Inicio del Sprint como lo marca la metodología se plasma para la construcción de cada una de las clases requeridas y la orientación hacia las funciones específicas de las mismas, se utilizan como herramienta los Diagramas de Caso de Uso, dado que estos nos permiten relacionar los actores con cada una de las labores que realizarán durante el funcionamiento del proceso a automatizar, a continuación, se detalla el caso de uso general de las funciones de la herramienta software.

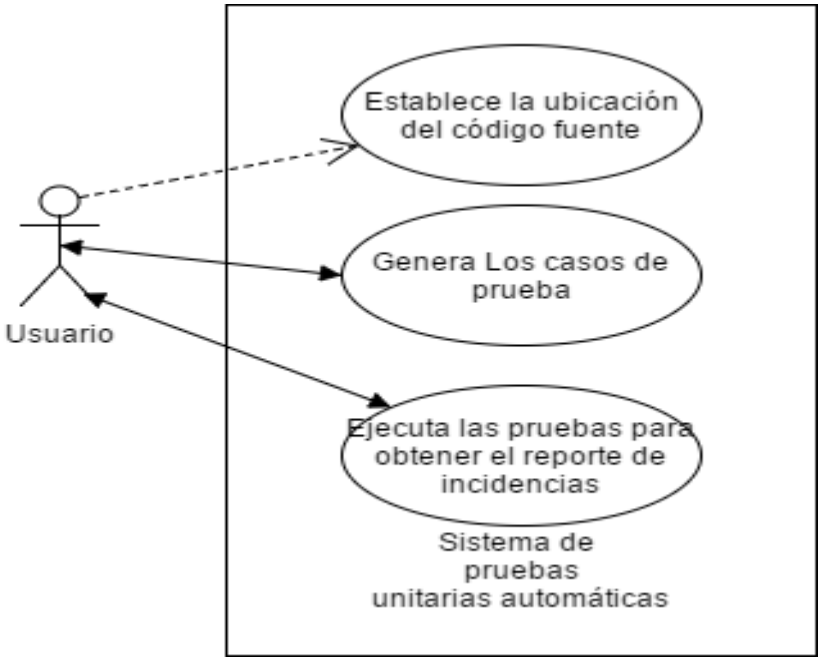

Figura 5-3. Diagrama de caso de uso proceso de detección de métodos

Realizado Por: Eduardo Paucar. 2017

Como se puede apreciar en el **Figura 5-3,** el usuario definido en el apartado correspondiente, utiliza el Sistema de pruebas unitarias automáticas como herramienta para optimización en los 3 procesos definidos en los objetivos, por cada Historia de Usuario se definió un caso de uso correspondiente a esa función utilizando la nomenclatura estándar de los mismos, los que puedes ser consultados en el **Manual Técnico** del producto.

Ya con todas estas herramientas generadas se procede con el desarrollo de las tareas asignadas para cada sprint teniendo en cuenta todo lo analizado en la documentación generada.

3.2.9.2 Casos de Prueba

Dentro de los casos de pruebas se ejecutarán con un modelo de casos de pruebas en la cual describiremos paso a paso como se van desarrollando.

Caso de prueba: Nombre significativo del caso de prueba que permita identificar el propósito de la prueba.

Identificador caso de prueba: Identificador único del caso de prueba.

Se recomienda que inicie la nomenclatura del nombre: NombreCasoDePrueba. Donde CP corresponde a las siglas de casos de prueba, el NombeCasoDePrueba corresponde al nombre significativo asignado en el campo.

Función probar: Definir el modulo, servicio o función que probara con el caso de prueba.

Objetivo: Describir que funcionalidad que será probada con el caso de prueba.

Descripción: Describir y explicar el propósito el caso de prueba.

Criterios de éxito: Definir los criterios de aceptación, que permiten determinar que el caso de prueba ejecuta

Criterios de falla: Definir los criterios que permiten determinar que el caso de prueba ejecutado

Precondiciones: Describir las condiciones y el estado en las que se debe encontrar el sistema para la ejecución del caso de prueba, en caso de ser necesario incluir los casos de pruebas que se deben ejecutar.

Perfil del usuario: Perfil del usuario en el sistema con el que se ejecutara la prueba.

Necesidades para el caso de prueba: Definir las necesidades para la ejecución de los casos de pruebas, como por ejemplo los datos de pruebas, las condiciones adicionales a tener en cuenta, configuración de la prueba.

Autor: Nombre de la persona que diseña el caso de prueba

Fecha de creación: Fecha en la que se diseña el caso de

prueba. **Flujo del caso de prueba**

- ✓ **No paso:** Orden en el que se ejecuta el paso
- ✓ **Usuario del sistema:** Acción del usuario en el sistema, definir las entradas requeridas en el paso y que realiza el usuario durante el paso, en caso que presente entradas, describir que hace el usuario con las entradas.

Post condiciones: Describir el estado del sistema luego de la ejecución de caso de prueba.

Código (CP): Añadiremos el código que se probó y observaremos el resultado.

Tabla 9-3: Caso de prueba (CP-insert)

Caso de prueba	
Identificador caso de prueba	CP001
Función probar	Control de notas. fadicionalesestuddiantes.
Objetivo	Verificar que los métodos que se encuentra en nuestra función sean correctamente funcional.
Descripción	Realizar un test de la función FAdicionalesEstudiantes. TestObtenerdadocodigo.
Criterios de éxito	En el siguiente caso de prueba (CP) se realizó exitosamente.

<table>
<tr><td>Criterios de falla</td><td colspan="3">Al momento del enviar un error, se muestra ya sea por el cambio de estado de true a false.

Por el tipo de dato int o un tipo de dato serial. Búsqueda dado un</td></tr>
<tr><td>Precondiciones</td><td colspan="3">Las precondiciones de para la ejecución de nuestro test se lo realiza previamente ejecutado nuestro realizado test.</td></tr>
<tr><td>Perfil del usuario</td><td colspan="3">Administrador.</td></tr>
<tr><td>Necesidades para el caso de prueba</td><td colspan="3">Debemos tener en cuenta que datos se los está enviando, para la ejecutar nuestro prueba.</td></tr>
<tr><td>Autor</td><td colspan="3">Fabricio Paucar</td></tr>
<tr><td>Fecha de creación</td><td colspan="3">18-06-2017</td></tr>
<tr><td rowspan="3">Flujo del caso de prueba</td><td>No paso</td><td>Usuario del sistema</td><td>Sistema</td></tr>
<tr><td>1</td><td>Para ejecutar se resuelve nuestro test (CP) simplemente ejecutamos nuestra función</td><td>Envía respuesta de</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>Post condiciones</td><td colspan="3">Si algún método tiene un error el sistema enviar un tests passed fallida.</td></tr>
<tr><td>Código (CP)</td><td colspan="3">@Test
public void testobtenerdadocodigo () throws Exception {
System.out.println("obtenerdadocodigo");
String metodo = "obtenerdadocodigo"; String clase = "FAdicionalesEstudiante"; Date vinicio = new Date();
int param1 = 3;
Boolean expResult = true; AdicionalesEstudiante result = null;
try {
result = FAdicionalesEstudiante.obtenerdadocodigo(param1);
} catch (Exception e) { result = null;
}
Boolean res = false; if (result != null) {
res = true;
} else {
res = false;
}
assertEquals(expResult, res); Date vfinal = new Date();
long duracion = Util.DiferenciaFechas(vinicio, vfinal);
Util.exportar(clase, metodo, duracion);
}</td></tr>
</table>

Realizado por: Eduardo Paucar. 2017

En la tabla siguiente vamos a demostrar, cual es el proceso de cómo se ejecuta nuestro caso de uso, para lo cual realizaremos a nuestra función determinada, en la cual un caso de uso lo realizaremos al finsertest en la cual esto nos retornara un tipo de dato entero o un tipo de dato serial, los demás casos de pruebas se encuentran en el **Anexo C**.

Tabla 10-3: Caso de prueba (CP-Delete)

Caso de prueba			
Identificador caso de prueba	CP008		
Función probar	Control de notas. FEstudioTe		
Objetivo	Verificar que los métodos que se encuentra en nuestra función sean correctamente funcional.		
Descripción	Realizar un test de la función FEstudioTest. testEliminar()		
Criterios de éxito	Al momento de ejecutar nuestra función y el caso de uso update nos devolverá un tipo de dato entero que representa su código.		
Criterios de falla	Por el tipo de datos que se envía String o int. Búsqueda dado un código.		
Precondiciones	Las precondiciones de para la ejecución de nuestro test se lo realiza previamente ejecutado nuestro realizado test.		
Perfil del usuario	Administrador.		
Necesidades para el caso de prueba	Debemos tener en cuenta que datos se los está enviando, para la ejecutar nuestro prueba.		
Autor	Fabricio Paucar		
Fecha de creación	18-06-2017		
Flujo del caso de prueba	No paso	Usuario del sistema	Sistema
	1	Para ejecutar se resuelve nuestro test (CP) simplemente ejecutamos nuestra función FAdicionalesEstudiantes.java	Envía respuesta de fallida o correcta

Post condiciones	Si algún método tiene un error el sistema enviar un tests passed fallida.

Código (CP)	@T
	est
	public void testEliminar() throws Exception {
	System.out.println("Eliminar");
	String metodo = "Eliminar"; String clase = "FEstudioSE"; Date vinicio = new Date();
	EstudioSE param1 = new EstudioSE(); Boolean expResult = true;
	boolean result = false; try {
	result = FEstudioSE.Eliminar(param1);
	} catch (Exception e) { result = false;
	}
	Boolean res = false; if (result != false) {
	res = true;
	} else {
	res = false;
	}
	assertEquals(expResult, res); Date vfinal = new Date();
	long duracion = Util.DiferenciaFechas(vinicio, vfinal);
	Util.exportar(clase, metodo, duracion);
	}

Realizado por: Eduardo Paucar 2017

En la tabla siguiente vamos a demostrar, cual es el proceso de cómo se ejecuta nuestro caso de uso, para lo cual realizaremos a nuestra función determinada, en la cual un caso de uso lo realizaremos al delete de nuestro método en la cual esto nos retornara un tipo de dato entero que representara como nuestro código, los demás casos de pruebas se encuentran en el **Anexo C**.

3.2.9.3 Diseño de interfaz test generator

Definir iremos un estándar sumamente amigable con el usuario, que permitan facilitar su fácil manejo establecido diferentes colores, formas etc.

Figura. Algo podemos observar una interfaz sencilla diseñada en un jframe la cual se cargará nuestro proyecto desarrollado exclusivamente en java.

Botón seleccionar: Es en encargado de seleccionar el proyecto a analizar y seleccionar.

Botón Generar: Es el en cargado de generar automáticamente nuestras pruebas unitarias, una vez seleccionado nuestro proyecto.

En la **Figura 6-3** podemos observar cómo está diseñado nuestra interfaz de nuestra aplicación de pruebas unitarias.

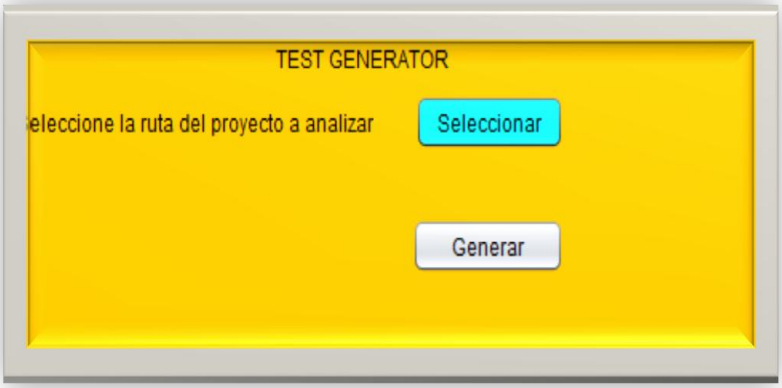

Figura 6-3. Pantalla

Realizado por: Eduardo Paucar. 2017

En la **Figura 7-3** Con el botón seleccionar escogemos el proyecto a ejecutarse para la realización de las pruebas unitarias automáticas.

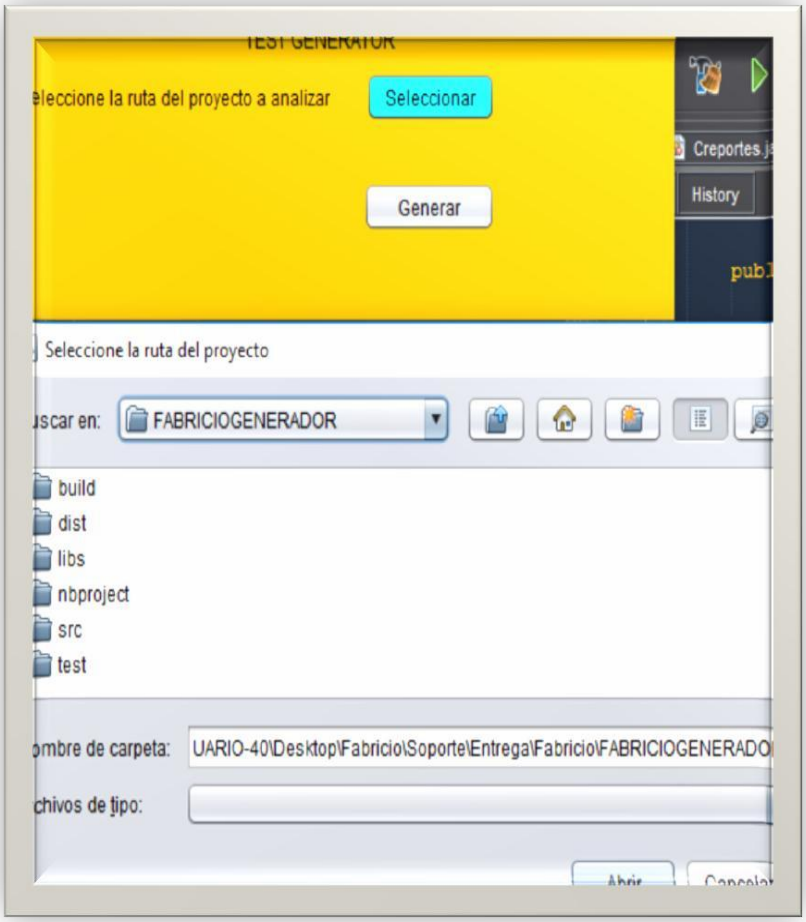

Figura 7-3. Pantalla Cargar Proyecto

Realizado Por: Eduardo Paucar. 2017

Cada especificación fue determinada por el desarrollador para que sea una interfaz sencilla y amigable ya que todo el proceso de generación se realizara, internamente automatizando pruebas unitarias.

Para la implementación de este requerimiento no funcional se ha realizado su respectiva historia técnica, tareas de ingeniería y pruebas de aceptación las cuales están desarrolladas en el **Anexo B.**

3.2.10 Codificación

Con el objetivo de programar el sistema "TEST GENERATOR", en el lenguaje de programación específico, se desarrolló en orden de prioridad las historias de usuario, que inician en el Sprint 4 hasta el Sprint 8.

Cada historia de usuario desarrollada desempeña un requisito funcionalidad específico del sistema, estos plasman en teoría la codificación de los requerimientos establecidos por el usuario. Se de elaboran casos de pruebas para cada método, para el desarrollo del sistema se ha utilizado el IDE Netbeans 8.1, JDK 1.8 bajo el lenguaje de programación java, framework junit y el servidor GlassFish 4.1.1.

En el **Anexo B** se presentan las Historias de Usuario de la aplicación "TEST GENERATOR" cada una con sus respectivas tareas de Ingeniería y estas a su vez cuentan con sus pruebas de aceptación respectivas.

Una vez realizado la implementación del sistema se realizó un conteo de las clases, métodos y las líneas de código del que está constituido el sistema.

3.2.11 Grafica de Producto (Burn Up)

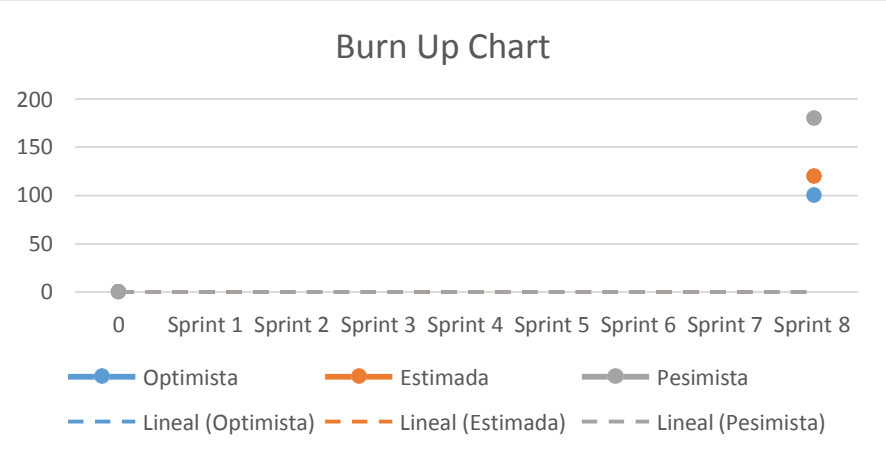

Figura 8-3. Diagrama de Caso de Uso de la Gestión de Feriados

Realizado por: Eduardo Paucar. 2017

Según lo expresado se estiman 120 puntos de función, los cuales se distribuyen en 8 Sprint, el desarrollo se puede ver comprometido tal como lo expresa el estudio de factibilidad y el análisis de riesgos por dificultades en el uso de la tecnología y lo proceso del complejo que conlleva trabajar directamente sobre el código en desarrollo, lo cual nos lleva a una estimación pesimista de 180 puntos de función según la gestión de riesgos realizada.
Por otro lado, al tener el desarrollador experiencia en el lenguaje de programación, así como en el motor de base de datos y tener la apertura del desarrollador del Sistema Escolástico y documentación de proyectos similares por el mismo, se estima de manera optimista 100 puntos de función.

3.2.12 Manual de usuario

Este manual permitirá a cada uno de los usuarios que manejarán el sistema tener una guía clara de cómo deberá usar el mismo, así como sus funcionalidades y características.

Se ha realizado un documento en Microsoft Word 2013 que consta de 10 hojas de contenido, que detalla los pasos con su respectiva imagen de cada una de las opciones de la aplicación, para que el desarrollarlo java utilice de manera más fácil de tal forma que no exista inconvenientes en su uso. El manual de usuario se encuentra adjunto al presente trabajo de titulación.

3.2.13 Determinación de la ejecución de TEST GENERATOR

3.2.13.1 Creación de Paquete funciones automáticamente

Funciones

En la **Figura 9-3** el paquete funciones se generará automáticamente nuestras nuevas clases .java las cuales cuentan con los casos de pruebas que se acceden a la base de datos.

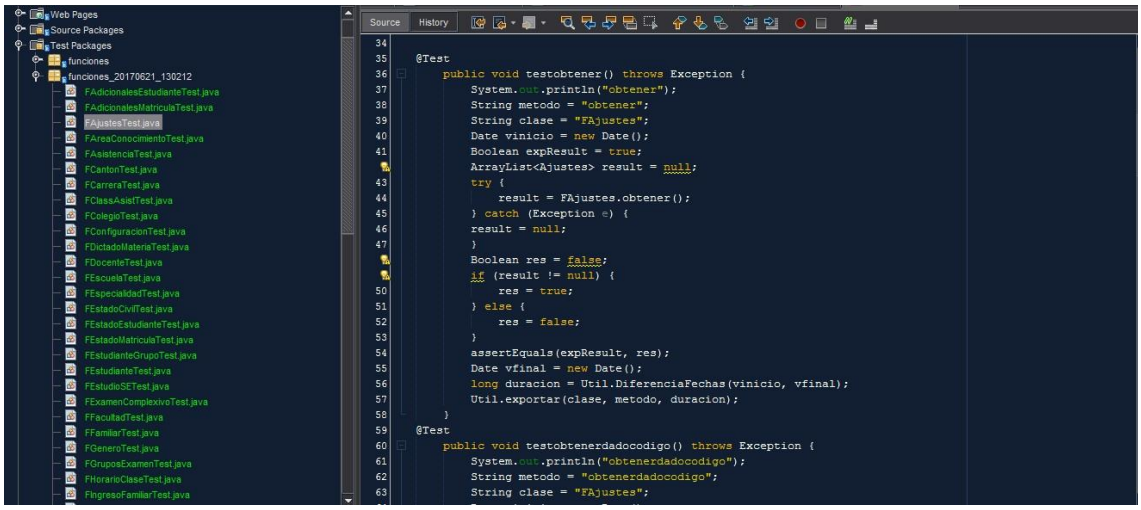

Figura 9-3. Paquete funciones

Realizado por: Eduardo Paucar. 2017

3.2.13.2 Ejecución de una clase caso de prueba.

En la siguiente **Figura 10-3** una vez q tenemos nuestras clases creadas con nuestro código automático internamente, procedemos a evaluar ejecutando, file test, la cual pasar a evaluar cada método que contiene.

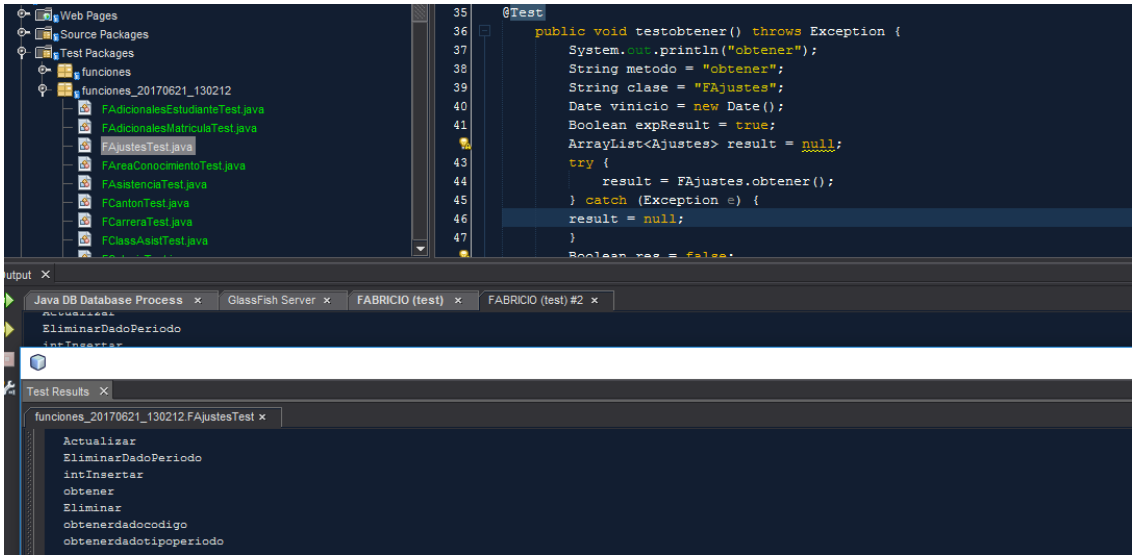

Figura 10-3. Ejecución Test

Realizado por: Eduardo Paucar. 2017

3.2.13.3 Reporte TEST GENERATOR

En la figura nos indica un reporte en la cual contendrá una matriz mostrándonos los métodos que fueron evaluados, dentro de esas clases y a su vez que contendría código de test de cada método accedido.

- **Método:** Son los métodos que fueron ejecutados de nuestro test.
- **Tiempo:** se tomará el tiempo en milisegundos.
- **Campos:** son los campos que se encuentran en nuestra base de datos y las cuales fueron accedidas.

Si en nuestra matriz observamos que tenemos un número 1 es significa que dentro de ese campo fue accedido por nuestro método.

Si en nuestra matriz observamos que tenemos un numero 0 es significa que dentro de ese campo no pertenece o no fue accedido ya que pertenece a otro método.

En la **Figura 11-3** observamos cómo está establecido nuestra matriz generado por nuestra aplicación.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	C
1	metodo	tiempo	colegio.codi	colegio.nom	configuracio	configuracio	contrato_do	contrato_do	contrato_do	contrato_do	contrato_do	contrato_do	coorequisito	coorequisito	cooreq
2	FFacultad obtenerdadocodigo	297	0	0	0	0	0	0	0	0	0	0	0	0	
3	FColegio obtenerdadocodigoligero	217	1	1	0	0	0	0	0	0	0	0	0	0	
4	FColegio obtener	93	1	1	0	0	0	0	0	0	0	0	0	0	
5	FColegio obtenerdadocodigo	22	1	1	0	0	0	0	0	0	0	0	0	0	
6	FColegio obtenerligero	52	1	1	0	0	0	0	0	0	0	0	0	0	

Figura 11-3. Matriz Test

Realizado por: Eduardo Paucar. 2017

3.3 Sprint

3.3.1 Sprint 1

El Sprint 1 estuvo orientado a las metáforas del sistema, por lo cual se obtuvieron productos intangibles pero apoyados en documentos de ayuda como el diccionario de datos de la base del Sistema Web Escolástico, misma que se evidencia en el **Manual Técnico** correspondiente y que nos será de gran utilidad para entender el funcionamiento del mismo.

3.3.2 Sprint 2-8

Los Sprint ubicados entre 2 y el 8 estuvieron enfocados a la codificación del proyecto, cada uno de estos generó un proyecto utilizable con funcionalidades reducidas e independiente, mismo

que en sí mismo ya puede ser implementado para la toma de métricas, pero orientándonos en que el objetivo es optimizar el recurso, se realizará este proceso cuando el proyecto terminado ahorre la mayor cantidad de esfuerzo posible.

El desarrollo, mismo que se encuentra explicado en el manual técnico entregado a los responsables de los proyectos de la facultad según lo evidencia el **Anexo A,** produjo el DESARROLLO DE PRUEBAS UNITARIAS AUTOMÁTICAS PARA EL SISTEMA WEB ESCOLÁSTICO DEL INSTITUTO TECNOLÓGICO SUPERIOR STANFORD obteniendo el total de 986 líneas de código y un peso de la aplicación alrededor de un total de 65.42 Mb.

El uso de esta versión del sistema será el que se utilice para las mediciones correspondientes de tiempos para el cumplimiento de los objetivos.

3.3.3 Reunión y Cierre de Sprint

Cada una de las entregas se realizó en las fechas estimadas, dado que se planificaron en horario de consulta del tutor del proyecto.

3.4 Análisis de Resultados

Como se menciona en anteriores, se requiere que la información entregada al desarrollador sea homogénea para facilitar su manejo y comprensión, por lo cual, al tener el presente sistema un formato automático para la generación del informe de incidencias, además de la generación automática de las pruebas unitarias correspondientes a los métodos relacionados con el acceso a datos, las mismas se verán homogéneas dado que existe solo una fuente de obtención de información, por cuanto queda demostrado que el Sistema ayuda a la homogenización.

3.4.1 Mejora de Procesos

Para la demostración de los objetivos se procederá a la comparación de los correspondientes procesos realizados de la manera tradicional y utilizando la herramienta desarrollada, para así poder definir en caso de existir, el valor que represente la mejora en tiempos y recursos obtenida.

Con el fin de comprobar la existencia de algún tipo de mejora en relación a los recursos requeridos por el proceso de establecer:

- **Objeto de Experimentación:** Sistema de generación de pruebas unitarias automáticas para sistema web escolástico del Instituto Tecnológico Superior Stanford.
- **Sujetos de Experimentación:** Sistema Web Escolástico del Instituto Tecnológico

Superior Stanford

- **Población:** 6 Iteraciones

3.4.2 Métricas

La decisión de medición tomada presenta las siguientes ventajas:

- Dada la similar experiencia en lenguajes de programación, economiza recursos en cuanto a capacitación previa a un sujeto de pruebas para el levantamiento de información.
- El nivel de conocimientos de uso del sistema es standard así que las medidas obtenidas serán más reales.
- Teniendo en cuenta que el desarrollador ya había realizado el análisis de los métodos relacionados al acceso a datos, la realización del mismo proceso por segunda vez nos hubiese entregado información irreal sobre el tiempo requerido, problema que al afrontarlo con nuestra decisión de medición se evita dado que el encargado de probarlo estará en la misma condición para los 2 procedimientos.

Aunque se presentan las siguientes desventajas

- No se puede utilizar técnicas como la encuesta, dado que solo se presenta 1 usuario
- El usuario a tomar las medidas no es imparcial y objetivo al entregar información

3.4.2.1 Muestra

Para una población de 6 iteraciones a analizar, procedemos a realizar el un cálculo del tamaño de la muestra que la consideramos para este caso.
Con una significancia (α) de 0.05 (5%) que corresponde a un nivel de confianza (1- α) del 95% (0.95) se tienen los siguientes datos:

- e=5%=0,05
- Z=1,96
- N=163
- p= 0,5
- q=0,5

al momento de obtener los datos requeridos damos paso a aplicar en la formula correspondiente:

$$x = \frac{1.96^2 * 6 * 0.5 * 0.5}{0.1 * (6 - 1) + 1.96^2 * 0.5 * 0.5}$$

x = 5.7624/1.4604 x=3.945

Valor que se aproxima a 4 iteraciones utilizando la función techo dado que una iteración es una unidad indivisible convirtiéndola en un tipo de dato entero.

3.4.2.2 Medida

La información obtenida representa el tiempo invertido en cada uno de los procesos involucrados en la automatización, en la **Tabla 11-3** se presenta un resumen organizado de las 4 iteraciones analizadas y los 3 procesos correspondientes a cada uno.

Tabla 11-3: Resumen de mediciones

	Clases	**Métodos**	**Métodos Relacionados**	**Tiempo en detección (s)**	**Tiempo en generación (min)**
Iteración 3	8	46	40	1408	208:16
Iteración 4	6	40	36	1034	152:21
Iteración 5	5	26	25	876	61:35
Iteración 6	12	72	64	3011	315:42

Realizado por: Eduardo Paucar. 2017

De la tabla anterior se omiten los datos de las iteraciones 1 y 2 del proyecto, dado que el cálculo de la muestra realizado en el punto 3.4.2.1 nos establece que únicamente se necesitan analizar 4 iteraciones, por lo cual se escogen las que más volumen de incidencias pueden presentar dado que al ser las iteraciones finales estas ya tienen un registro previo con quien compararlas, lo que no sucede con la iteración uno por ejemplo que al ser la primera no presentará incidencias, además la iteración uno contiene gran cantidad de catálogos en su desarrollo, tablas que son de escaso acceso durante el funcionamiento del negocio.

También se omite el proceso de búsqueda y resolución de incidencias dado que este proceso será analizado posteriormente para la comprobación de otra mejora, el presente análisis se enfocará

en la reducción en tiempo de búsqueda de métodos y la creación de pruebas unitarias de los mismos.

Como punto de comparación, se tiene la **Tabla 12-3,** la cual contiene los valores producidos por el aplicativo, se lo probó en un entorno de uso regular del equipo, siendo únicamente el aplicativo el programa en primer plano cargado en el sistema después de un reinicio, los valores son referenciales al equipo mencionado en el hardware existente para el desarrollo y estos pueden variar sustancialmente en dependencia de los recursos disponibles en el equipo de pruebas en el momento del levantamiento de información y las características del equipo.

Tabla 12-3: Resumen de mediciones con el Sistema

	Clases	**Métodos**	**Métodos Relacionados**	**Tiempo en detección (s)**	**Tiempo en generación (s)**
Iteración 3	8	46	40	3.004	3.12
Iteración 4	6	40	36	4.206	3.16
Iteración 5	5	26	25	3.514	3.31
Iteración 6	12	72	64	4.136	4.61

Realizado por: Eduardo Paucar. 2017

Como se puede evidenciar en la Tabla 11-3, se hizo necesario el cambio de las unidades en la columna 6, dado que medirlo en minutos era demasiado extenso, mientras que en la Tabla 12-3, medirlo en segundos era demasiado pequeño mientras que medirlo en horas era demasiado extenso, dándonos una prevista inicial de una mejora en el proceso, misma que será comprobada de una manera estadística.

También cabe mencionar que el proceso que más homogeneidad presenta es el proceso de generación, dado que es directamente dependiente a la cantidad de métodos relacionados, mientras que en el tiempo de detección de métodos relacionados se tiene valores fluctuantes, se asume debido a la naturaleza de lectura y clasificación del proceso y a la cantidad de falsos positivos que puedan presentarse.

Ambos procesos detectaron la misma cantidad de métodos interactúan con la base de datos.

El total de clases del proyecto a analizar son 278 en 52 clases, de las cuales fueron analizados en las 4 iteraciones 32 clases con un total de 184.

Antes de proceder a declarar la diferencia entre estos 2 procesos, se debe calcular si en realidad existeuna diferencia significativa entre estos 2 grupos de datos, por lo cual se establece la prueba de Wilcoxon para pares de datos relacionados como proceso estadístico para la demostración, el

cual de concluir que existe diferencia significativa nos permitirá calcular el valor de la misma, mientras que si este proceso estadístico resulta falso significará que la diferencia entre ambos procesos es despreciable, por lo cual se procede a la aplicación de la prueba de Wilcoxon. Para la implementación de la prueba, se define como:

H0 = los datos entre los procesos no presentan diferencia significativa (hipótesis nula)

H1 = los datos entre los procesos si presentan diferencia significativa (hipótesis del investigador) La **Tabla 13-3,** muestra de forma organizada los valores requeridos para el análisis de Wilcoxon.

Tabla 13-3: Análisis Búsqueda de Métodos

P. Manual	**P. Automatizado**	**Signo 1 +**	**Diferencia absoluta**	**Rango**	**Rango con signo**
		-1 = -			
1408	3.00	1	1404.99	4	4
1034	4.20	1	1029.79	5	5
876	3.51	1	872.486	4	4
3011	4.13	1	3006.86	7.5	7

Realizado Por: Eduardo Paucar. 2017

Con lo mencionado en la tabla anterior se procede al uso de la herramienta Wilcoxon para la medición con lo que se obtiene como resultado

Figura 12-3. Detalle de resultados
Realizado por: Eduardo Paucar. 2016

Como lo menciona la herramienta, al tener menos de 20 pares, la evaluación sigue la formula Z-valué, con el nivel de significancia plantado (0.05), por lo cual tenemos.

-2.873 < 0.05;

Valor que al ser menor comprueba la hipótesis del investigador descartando la hipótesis nula, razón por la que se declara que existe diferencia significativa entre los 2 rangos de valores, misma que es de un 99.77% en el promedio de la muestra.

El mismo proceso se aplica a la diferencia en tiempo de generación de pruebas unitarias, mismo que refleja un resultado del 99.87% de mejora en el tiempo de generación de las pruebas unitarias para los métodos encontrados, el resumen de mediciones se expresa en la **Tabla 14-3.**

Tabla 14-3: Análisis Generación de Pruebas Unitarias

P. Manual (min)	**P. Automatizado (seg)**	**Signo 1 + -1 = -**	**Diferencia absoluta (seg)**	**Rango**	**Rango con signo**
208:16	3.12	1	12486.48	8.5	8.5
152:21	3.165	1	9128.435	7.1	7.1
61:35	3.315	1	3677.685	6.8	6.8
315:42	4.615	1	18920.585	8.5	8.5

Realizado por: Eduardo Paucar. 2017

Con lo cual se evidencia que existe una reducción considerable entre los tiempos invertidos en la realización de estos procesos, comprobando así la mejora de eficiencia planteada en los objetivos, al proveer información homogenizada, esta facilita el análisis por parte del desarrollador, lo cual mejora también el proceso de búsqueda y resolución de incidencias.

CONCLUSIONES

- Al analizar el proceso de control de notas del sistema web escolástico del Instituto Tecnológico superior Stanford, se establecieron 13 requerimientos funcionales, los cuales se desarrollaron como historias de usuario y 3 requerimientos no funcionales los cuales contribuyen a mejorar la calidad del producto “TEST GENERATOR”.

- Mediante la utilización de la aplicación “TEST GENERATOR” se pudo mejorar los procesos de desarrollo de pruebas unitarias, que se elaboraban manualmente por el desarrollador java, ya que gracias a esta aplicación se automatiza su desarrollo reduciendo el tiempo de los procesos.

- Mediante el uso de la aplicación desarrollada enfocado a los desarrolladores java, conlleva una reducción mayor al 90% del tiempo en promedio de elaboración de pruebas unitarias y detección de métodos que acceden a la base de datos.

- Se concluye que la aplicación desarrollada reduce los tiempos detección de los métodos que se relaciona con la base de datos y elaboración automáticamente de pruebas unitarias.

RECOMENDACIONES

- Se recomienda la implementación de un módulo de detección de conexiones en el presente sistema para poder convertirlo en multipropósito, dado que actualmente está personalizado para proyectos de similares características al aquí analizado.

- En el desarrollo de pruebas unitarias automáticas se debe contemplar dentro de la planificación de la elaboración de un proyecto de desarrollo ya que esto conlleva a garantizar que el desarrollo de la aplicación sea un software de calidad.

- Para futuras aplicaciones que se realizan hoy en día se debe implementa como factor importante el desarrollo de pruebas unitarias ya que conlleva un rol importante dentro del desarrollo, la cual ayudara a detentar los errores que existan dentro la aplicación conllevando a una refactorización del código.

- Las pruebas deben usarse en todo el ciclo de desarrollo de un proyecto, sobre todo debe formar parte del día a día del programador, formulando filosofías como la TDD (Test Drive n Development) donde el método es probar antes de implementar.

GLOSARIO

MVC

Siglas del patrón de diseño Modelo-Vista-Controlador, que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

PUNTO DE FUNCIÓN

Es un método utilizado en ingeniería del software para medir el tamaño del software. Fue definida por Allan Albrecht, de IBM, en 1979 y pretende medir la funcionalidad entregada al usuario independientemente de la tecnología utilizada para la construcción y explotación del software, y también ser útil en cualquiera de las fases de vida del software, desde el diseño inicial hasta la implantación y mantenimiento.

UML

Lenguaje Unificado de Modelado (Unified Modeling Language). Es un lenguaje de modelado de sistemas o de algún software en específico. El lenguaje gráfico, estandarizado por el OMG (Object Management Group), de mucha ayuda en las tareas de visualizar, especificar, construir y documentar un sistema o algún otro proyecto de desarrollo de software.

BIBLIOGRAFÍA

1. **AGUIRRE BUENAÑO, Tania Paola, & MONCAYO ALVAREZ, Andrea Isabel**. *Análisis de frameworks MVC de Java para el desarrollo de aplicaciones web empresariales. Caso práctico: Sistema de Bienestar Politécnico* (tesis). (Ingeniería). Escuela Superior Politécnica de Chimborazo, Facultad de Informática y Electrónica, Escuela de Ingeniería en Sistemas. Riobamba-Ecuador. 2003. pp. 55-56

2. **BEMONTE, O** *Introducción al lenguaje de programación Java. Una guía básica* [enlínea]. [Consulta: 01 mayo 2017]. Disponible http://www3.uji.es/~belfern/pdidoc/IX26/Documentos/introJava.pdf.

3. **BERNERS-LEE, T**. *World Wide Web Consortium Launches International Program Office for Web Accessibility Initiative* [en línea]. Washington, DC:, 22 octubre 1997.[Consulta: 19 mayo 2017]. Disponible en: https://www.w3.org/Press/IPO-announce.

4. **CASOS DE PRUEBAS*:junit*** [en línea]. Universidad de Alicante, pp.1-2. [Consulta 04 de 01 de 2017]. Disponible en http://www.jtech.ua.es/j2ee/publico/lja-2012-13/sesion04-apuntes.html.

5. **EXTREMEPROGRAMING**. *extremeprograming.org*. [En línea]. 2013 [Consulta: 01 mayo 2017]. Disponible enhttp://www.extremeprogramming.org.

6. **RAFAEL, mauricio**. *Postgres.org*. [En línea]. 2010 [Consulta: 01 mayo 2017]. Disponible en http://www.postgres.org.

7. **GONZALES, M**. *Ciencia, tecnología y administración pública*. [en línea].Colombia:2011. [Consulta: 12agosto2016].Disponible en http://manuelpereiragonzalez.blogspot.com/2010/12/diez-motivos-para-programar-en-java.html.

8. **GRUPO DE INGENIERÍA DE SOFTWARE.** Introducción a las Aplicaciones Web [en línea].2014.[Consulta: 15 enero 2017]. Disponible en: http://www.lsi.us.es/docencia/get.php?id=854.

9. **IÑIGUEZ, S.** *Que_es_la_automatizacion_de_procesos. [*En línea]. 2011. [Consulta: 11 abril 2017]. Disponible en https://es.over-blog.com/Que_es_la_automatizacion_de_procesos-1228321767-art127041.html.

10. **JAVA.** conozca mas sobre la tecnología de java [en línea]. Estados Unidos: Oracle, 204. [Consulta: 01 mayo 2017]. Disponible https://www.java.com/es/about/

11. **JAVAHISPANO.** *javaHispano.org*. [En línea]. 2017. [Consulta: 01 marzo 2017]. Disponible en http://www.javahispano.org.

12. **KUMAR, Manoj**. *Multi-Objective Optimization of Software Test Case Using Evolutionary and Soft Computing Techniques* (tesis). (Doctor of Philosophy). School of Mathematics and Computer Applications, Thapar University. Punjab-India. 2013. pp. 66-71

13. **LUISATAXI, José Rodolfo**. *"COMPARACIÓN DE HERRAMIENTAS BENCHMARKING PARA PRUEBAS UNITARIAS WEB DE APLICACIONES JSF. CASO PRÁCTICO: SISTEMA DE GESTIÓN DE VIÁTICOS DE LA ESPOCH"* (tesis). (Ingeniería). Escuela Superior Politécnica de Chimborazo, Facultad de Informática y Electrónica, Escuela de Ingeniería en Sistemas. Riobamba-Ecuador. 2013.

14. **MEJÍA BRONCANO, Miguel Alexander**. *"SISTEMA DE SEGUIMIENTO A PLANIFICACIÓN DE PEA PARA LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA DE LA ESPOCH*" (tesis). (Ingeniería). Escuela Superior Politécnica de Chimborazo, Facultad de Informática y Electrónica, Escuela de Ingeniería en Sistemas. Riobamba-Ecuador. 2016.

15. **MANCHADO, D. Serra; PUNTES, Daniel Franco**. *Estudio del servidor de aplicaciones Glassfish y de las aplicaciones J2EE* [en línea]. 2010, pp.96-97. [Consulta 06 de 04 de 2017]. Disponible en https://core.ac.uk/download/pdf/13325903.pdf.

16. **SÁNCHEZ, J.** ¿Qué es un „framework"? [en línea]. 2016. [Consulta: 9 abril 2017]. Disponible en: http://jordisan.net/blog/2006/que-es-un-framework/.

17. **SCHWABER, Ken, & SUTHERLAND, Jeff**. La Guía de Scrum [en línea].julio 2016. [Consulta: 19 agosto 2016]. Disponible en: http://www.scrumguides.org/docs/scrumguide/v2016/2016- Scrum-GuideSpanish.pdf#zoom=100.

18. **VALDÉS, D.** Maestros del Web. [En línea]. 2009 [Consulta: 9 abril 2017]. Disponible en: http://www.maestrosdelweb.com/los-diferentes-lenguajes-de-programacion-para-la-web/.

19. **WILLIAMS. N.** Java for Web Applications [en línea]. Indianápolis – Estados Unidos: Wrox, [Consulta: 01 mayo 2017]. Disponible http://site.ebrary.com/lib/espoch/detail.action?docID=10842292&p00=java+8.

ANEXOS

Anexo A

Certificado entrega de código por la institución.

Anexo B

- Manual técnico
- Historias de usuario
- Pruebas de aceptación

Anexo C

Casos de Pruebas

Anexo D

Manual de usuario

Anexo E

Diagrama de Gantt