Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Desarrollo Orientado a Comportamiento.
Caso de Estudio:
Solicitudes de Jurado para Seminario y Trabajos Especiales de Grado

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela

Por los Bachilleres
Daniel Mariñan F.
Julio J. Montaño B.

para optar al título de
Licenciado en Computación

Tutora:
Profa. Jossie Zambrano

Caracas, Mayo de 2012

ACTA

Quienes suscriben, miembros del jurado designado por el Consejo de la Escuela de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela, para examinar el Trabajo Especial de Grado titulado **Desarrollo Orientado a Comportamiento. Caso de Estudio: Solicitudes de Jurado para Seminarios y Trabajos Especiales de Grado**, presentado por los bachilleres Daniel Mariñan F., C.I.: 18.707.369 y Julio J. Montaño B., C.I.: 18.617.291, a los fines de optar por el título de Licenciado en Computación, dejan constancia de lo siguiente:

Dicho trabajo, leído por cada uno de los miembros del jurado, se fijó el día Martes 29 de mayo de 2012, a las 2:00 PM, para que sus autores lo defendieran en forma pública en la Escuela de Computación, mediante una presentación oral de su contenido, luego de lo cual respondieron a las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo con la nota de _____ puntos.

En fe de lo cual se levanta la presente Acta, en Caracas a los veintinueve (29) días del mes de mayo del año dos mil doce (2012).

_____

Prof. Jossie Zambrano (Tutor)

_____ _____

Prof. Eugenio Scalise
(Jurado)

Prof. Sergio Rivas
(Jurado)

Agradecimientos

En primer lugar le agradecemos a Dios por protegernos en todo momento así como a nuestras familias y amigos. Gracias Dios por ayudarnos permanecer en el camino correcto y lograr los objetivos planteados.

A mis padres Luis y Mari, y hermanos Miguel y Diego, quienes sin pedírselo siempre están pendientes de mi y deseándome todo lo mejor. Gracias, los quiero mucho.

A mi Madre Esther por siempre creer en mí, a mi hermanita Chris por siempre apoyarme cuando lo necesitaba y cuando no, y a mi Papá Enrique por siempre estar allí para mí. Los quiero.

A Tibi, me has hecho una mejor persona desde que te conocí y cada vez que estoy contigo no paro de reír y de estar alegre. Gracias por todo tu apoyo incondicional, te adoro.

A Vanessa, por darme ese empujón que necesitaba para terminar esta tesis, por confiar en mí sin importar nada y por las infinitas palabras de apoyo, te amo mi bella novia.

Al Profesor Eugenio Scalise, gracias por la tolerancia y por aceptarnos bajo su tutela en nuestros momentos de presión. Gracias por ser un ejemplo para nosotros, sin Ud. no lo hubiésemos logrado.

A la Profesora Jossie por tener la paciencia más grande del mundo, aceptar nuestros errores buscando corregirlos y que no los volviéramos a cometer aun cuando tropezábamos una y otra vez.

A mi compañero de tesis Julio, gracias por todo, sabes que te quiero como a un hermano.

A Daniel por apoyarme cada vez que lo necesitaba y ser mi compañero en este trabajo de grado y en otros muchos proyectos durante la carrera.

A Las personas con las que compartimos probablemente más del 90% de la carrera y que todos los días hacían que nos dieran ganas de volver a la universidad y que nunca nos aburriéramos. Jesús, Julio, Pablo. Gracias por todo amigos.

A Lily por ayudarme en esos momentos de presión y por las infinitas correcciones y ayuda en nuestro TEG.

A Rosa, Loy, Keyla, Meche, Marco y a todos los demás con los que tuvimos la oportunidad de compartir en estos años. Muchas gracias muchachos, nunca los olvidaremos.

A todas las personas que hicieron esto posible, gracias.

Resumen

El presente Trabajo Especial de Grado (TEG) tiene como objetivo utilizar técnicas de desarrollo orientado a comportamiento (DOC), utilizando el enfoque “de afuera hacia adentro” (*Outside in* , donde primero se generan las pruebas y las interfaces, y después el código que vuelve funcional la aplicación); para la implementación de una aplicación web, tomando como caso de estudio la automatización del proceso de solicitud de jurados para Seminarios y TEG del Sistema de la División de Control de Estudios CONEST de la Facultad de Ciencias de la Universidad Central de Venezuela.

Este proceso es utilizado por las Escuelas que pertenecen a la Facultad de Ciencias (Biología, Computación, Física, Geoquímica, Matemáticas y Química) y está conformado por un grupo de tareas y actividades necesarias para la asignación de jurado para TEG y Seminarios, sin embargo, cada Escuela posee actividades, pasos y usuarios específicos para realizar este proceso y no poseen una comunicación efectiva con otras Escuelas. De este grupo de actividades se resaltan las siguientes: creación y envío de Solicitudes, Solicitud de Acta de notas realizadas por el Tutor Docente; verificación y modificación de solicitud de jurado realizadas por la Comisión de Trabajo Especial de Grado de la Escuela donde aplique; aprobación y modificación de solicitud de jurado, realizadas por el Consejo de Escuela; y finalmente la División de Control de Estudios, encargada de la comprobación y de la validación de los datos de estudiante, como por ejemplo, su expediente, notas y pensum. En este TEG se aplica un proceso estándar para las Escuelas cuyos principales actores son: Tutores, Comisión de TEG, Consejos de Escuela y la División de Control de Estudios, y se describen los procesos básicos del DOC (discusión, decisión, desarrollo y demostración).

Palabras Clave: Desarrollo Ágil, Desarrollo Web, Desarrollo Orientado a Comportamiento, *Cucumber*, CONEST.

Tabla de contenido

Tabla de figuras

Introducción

Dentro de los procesos más importante de las Escuelas de la Facultad de Ciencias se encuentra el referente a la solicitud de jurado para Seminario y Trabajo Especial de Grado (TEG), que se refiere a la asignación de jurado para la revisión de los TEG y Seminarios de estudiantes próximos a egresar de la institución. Para realizar este proceso deben intervenir diferentes entidades que se encargan de actividades específicas. El proceso es el siguiente: el Tutor debe realizar una solicitud de jurado frente a la Comisión de Grado o Consejo de Escuela, que se encarga de la validación y aprobación de la solicitud respectivamente. Una vez aprobada la solicitud, se envía a la División de Control de Estudios que se encarga de comprobar los requerimientos del estudiante antes de la presentación de su trabajo. Sin embargo las escuelas realizan este proceso de forma aislada, resultando en discrepancias en las actividades y documentos requeridos; pudiendo crear confusiones a los estudiantes y a la División de Control de estudios que se encarga de concentrar la información de todas las Escuelas de la Facultad.

En la actualidad este proceso es llevado manualmente, lo que genera una considerable dificultad al momento de obtener datos de los estudiantes asociados a la solicitud ya que la Facultad de Ciencias cuenta con una matrícula de más de 3.000 estudiantes de los cuales se debe llevar un registro de sus solicitudes y procesos frente a la Facultad, esto pudiera generar errores al momento de validar los requerimientos de los estudiantes y retrasa el proceso al tener que buscar de manera manual en sus expedientes. Además, las notificaciones del resultado de las solicitudes de jurado también son llevadas en papel, lo que dificulta la difusión a tutores y jurados externos a la Facultad. Tampoco se cuenta con un registro donde los tutores puedan obtener de manera sencilla información acerca de las solicitudes que están en proceso.

Es por eso que el Objetivo General de este Trabajo Especial de Grado consiste en desarrollar una aplicación Web que automatice los procesos relacionados a las solicitudes de jurado para Seminario y TEG que son similares para las todas las Escuelas de la Facultad de Ciencias utilizando desarrollo orientado a comportamiento que permite llevar un registro automatizado de las solicitudes de jurado de las distintas Escuelas, reducir los tiempo relacionados con el archivo y difusión de resultado de Solicitudes procesadas, y estandarizar el proceso de asignación de jurado atendiendo a las necesidades especificas y comunes de cada escuela, reduciendo los tiempos de procesamiento de las solicitudes de jurado para TEG y Seminario. El ciclo básico de desarrollo que se implementa y desarrolla en este TEG es

el siguiente: discusión, decisión, desarrollo y demostración. En este documento está estructurado de la siguiente manera:

Capítulo I: Se describe el marco conceptual donde se muestran los conocimientos básicos que fueron necesarios para la elaboración de este trabajo de investigación. Se explican las herramientas que se utilizan para elaborar la solución basada en procesos de desarrollo ágil.

Capítulo II: Comprende el marco aplicativo. Se demuestra cómo se utilizan los conocimientos y herramientas descritas en el marco teórico y tecnológico, para la realización de la solución propuesta. Se describe el problema que se solventa y el alcance que tendrá este TEG, describiendo el objetivo general y los objetivos específicos. Se menciona también la importancia de esta solución y las personas involucradas que se benefician.

Capítulo III: Conclusiones: Se mencionarán qué beneficios se lograron en el desarrollo de la solución, recomendaciones y limitaciones que lograron encontrarse.

Referencias bibliográficas.

Capítulo I: Marco Conceptual

En este capítulo se da a conocer los conceptos básicos y otros el lector debe comprender y que son la base para este Trabajo de investigación.

El Desarrollo Basado en Comportamiento (DOC) es un método que se ha sido perfeccionando a partir de lo que se conoce como Desarrollo Basado en Pruebas, ambos son métodos de desarrollo ágiles usados generalmente para proyectos en la web; es por esto que antes de abordarlos se ofrece un resumen sobre los procesos de desarrollo ágil, su origen, principios y características, e incluso algunos ejemplos donde se ve cómo aplicar esta filosofía al ambiente de desarrollo.

2.1 Desarrollo ágil

En febrero del 2001 un grupo de 17 de los principales expertos en procesos de desarrollo ágil (*The Agile Alliance*) se reunieron bajo la necesidad de crear documentación para un procesos de desarrollo que compitiera con otros ya establecidos, que se caracterizan principalmente por su rigidez y alta documentación en cada uno de sus procesos como RUP, orientada a documentos, para nombrar algunas. Entre ellos se encontraban expertos de procesos de desarrollo ágil como XP, SCRUM, programación pragmática, entre otros.

Este documento es lo que se conoce como el Manifiesto Ágil y pretende enunciar un conjunto de características y principios método lógicos que rigen los procesos basados en desarrollo ágil.

2.1.1 Principios del desarrollo ágil

El manifiesto busca resaltar los siguientes elementos del proceso de desarrollo (The Agile Alliance, 2001):

- *Individuos e interacciones* sobre procesos y herramientas: promover la comunicación del grupo de desarrollo y su conocimiento, y no las herramientas y tecnologías del entorno de trabajo.
- *Software funcionando* sobre documentación extensiva: se considera más útil módulos funcionando del sistema que documentación extensiva de dichos módulos.
- *Colaboración con el cliente* sobre negociación contractual: permitiendo una mayor participación del cliente en el desarrollo del software y no limitándose a la comunicación con el cliente por medio del contrato y reuniones periódicas.

- *Respuesta ante el cambio* sobre seguir un plan: haciendo más flexible el proceso de desarrollo, asumiendo cambios propios del proceso y nos asumiéndolos como una carga basada en el plan de acción.

2.1.2 Características del desarrollo ágil

A partir de sus principios se pueden enunciar 13 características generales que deben seguir los procesos de desarrollo ágil, estos son (The Agile Alliance, 2001):

1. *La mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor*.
2. *Se acepta que los requerimientos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente*.
3. *Se entrega frecuentemente software funcional, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.*
4. *Los responsables de negocio y los desarrolladores trabajan juntos de forma cotidiana durante todo el proyecto*.
5. *Los proyectos se desarrollan en torno a individuos motivados. Se les ofrece el entorno y el apoyo que necesitan, y se les confía la ejecución del trabajo.*
6. *El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.*
7. *El software funcionando es la medida principal de progreso.*
8. *Los procesos Ágiles promueven el desarrollo sostenible.*
9. *Los promotores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.*
10. *La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.*
11. *La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.*
12. *Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.*
13. *A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.*

2.1.3 Ventajas y desventajas del desarrollo ágil

"La principal ventaja de las metodologías ágiles es que proporcionan mejores resultados en los proyectos de alto riesgo" (Barbero, 2008), esto debido a los tiempos cortos de entrega entre cada interacción que permiten la adaptación y recuperación de un error cometido por el grupo de trabajo, "Gracias a la interactividad de las metodologías ágiles, estas están mucho más preparadas para asumir los cambios y los errores".

Otras de las ventajas del desarrollo ágil es que intenta mantener una mayor motivación del grupo de trabajo enfocándose menos en actividades de documentación y más en actividades de diseño e implementación de código, resultando en interacciones mejores y más rápidas. "Las metodologías ágiles se centran en las actividades más relevantes para la programación, que normalmente coinciden con las de mayor interés para los programadores."

(Colusso, Desarrollo ágil de software, 2009) Toma como punto importante que los equipos de programadores sean pequeños, preferiblemente en pares "revisando juntos el código y resolviendo problemas en lugar de tratar de cubrirlos, lo que repercute en un producto de mejor calidad, mejor documentado, y simple de mantener".

Sin embargo no todo es bueno un punto sensible para estos procesos de desarrollo como nos lo recuerda (Willian & Veronica, 2008) es la alta falta de documentación a un sistema desarrollado siguiendo procesos de desarrollo ágiles, "el código no puede tomarse como una documentación. En sistemas de tamaño grande se necesita leer los cientos o miles de páginas del listado de código fuente". A mayor documentación más entendible es el código y en proyectos grandes esto es un requerimiento para el código a implementar. Esto puede generar dependencia al grupo de trabajo ya que ellos son los únicos que lo entienden y resistencia al cambio debido al bajo nivel de entendimiento del código.

También es considerable tener en cuenta las "restricciones en cuanto a tamaño de los proyectos abordables". Los proyectos que son recomendables desarrollar con procesos de desarrollo ágiles son generalmente proyectos cortos con un nivel de simplicidad alto, grande proyectos que dependen de muchas personas no deberían ser desarrollados bajo procesos de desarrollo ágiles, sino bajo procesos de desarrollo más rígidos que se adapten al ambiente de desarrollo y al equipo de trabajo.

2.2 Desarrollo Orientado a Pruebas

Los procesos de desarrollo ágiles más conocidos, son Programación Extrema XP y SCRUM (José H. Canós, Métodologías Ágiles en el Desarrollo de Software, 2003), entre otros.

Antes de explicar el desarrollo orientado a comportamiento, primero se da a conocer a su predecesor: el desarrollo orientado a pruebas; del cual el DOC obtiene varios de sus principios esenciales.

El desarrollo orientado a pruebas (TDD, *test driven development*) es una forma de desarrollar basándose en que un algoritmo responda de una forma específica a unos datos específicos. Es decir, se diseñan primero las pruebas y luego se empieza a desarrollar, de forma que el código pueda superar estas pruebas (Ambler, 2010).

De forma resumida, el *TDD* o desarrollo orientado a pruebas se estructura en dos fases:

- Se diseña y escribe los casos de prueba y las pruebas unitarias en base a requerimientos del software y a la arquitectura del proyecto.
- Se codifica la aplicación de tal manera que el código implementado cumpla las pruebas diseñadas previamente.

El propósito del desarrollo guiado por pruebas es lograr un código que funcione y sea seguro ante los cambios. La clave es partir de los requerimientos. Estos deben ser traducidos a pruebas unitarias de tal forma que cuando todas las pruebas superen los requerimientos y todas sus derivadas, se podrá asegurar que la aplicación va a funcionar correctamente.

En la práctica, el diseño de pruebas unitarias es un proceso que va a convivir durante todo el ciclo de vida de un producto, desde su diseño inicial hasta la fase de explotación, con el mantenimiento de la misma.

Basándose en el principio YAGNI (por sus siglas en ingles, *You Ain't Gonna Need It? en español; No Lo Vas A Necesitar*), se pueden pedir unos casos de prueba que cubran la mayor parte de los casos posibles para la aplicación según los requisitos, de modo que se tengan los datos de entrada y los datos de salida de la aplicación. Es decir, no sólo saber qué es lo que se debe hacer, sino además tener un ejemplo de los datos que se deben poder conseguir a partir de cierta entrada para construir la aplicación con sólo y exclusivamente lo que nos pidan (Principio YAGNI/NLVAN).

Teniendo claro qué es lo que se debe hacer y además ilustrado con ejemplos palpables, pasar al cómo hacerlo es bastante simple puesto que lo que se tiene que hacer es realizar un programa que permita el paso de unos datos de una forma a otra.

Este enfoque TDD fusiona los conceptos de codificación y prueba, de modo que se hacen de forma conjunta al disponer datos para ello. Así mismo, incluye el diseño de las pruebas en el análisis y se usan como requisito para el diseño de la solución. Se ve claramente que las pruebas están incluidas en todas las etapas del desarrollo y no hay una etapa separada o específica para ellas.

2.2.1 Ventajas del Desarrollo Orientado a Pruebas

- Las pruebas son útiles porque ayudan a organizar mejor el código y a entender tanto los requisitos como las funcionalidades que debe implementar el software.
- Facilita la gestión de cambios, es decir, si hace falta cambiar algo en los requerimientos se puede hacer solamente con la funcionalidad como tal sin que afecte en mayor parte al proyecto.
- Automatiza las pruebas, con una evidente mejora en la productividad en el trabajo en equipo, lo que garantiza que los cambios introducidos son coherentes con los casos de prueba definidos.

2.2.2 Desventajas del Desarrollo Orientado a Pruebas

En su artículo (Dalke, 2009) menciona que “si se escriben las pruebas primero, también me preocuparé en que tengo que dar forma al código de las pruebas. Es decir que dado cualquier conjunto de datos, puedo encontrar un modelo que le corresponda. Ahora bien, ¿es el modelo válido y útil? la manera de verificar esto significa probar el modelo con datos que no han sido usados para crear el modelo”.

En principio, el modelado de código mediante las pruebas es una de las ventajas del TDD pero también podría ser una desventaja si se mira desde el punto de vista donde debido a las pruebas, se está haciendo un modelo que se ajuste a ellas pero no al comportamiento de los objetos. Es posible dependiendo del proyecto que se esté desarrollando y por consiguiente nació el desarrollo orientado a comportamiento

Esa es una de las grandes ventajas de TDD, el modelado del código mediante las pruebas, pero, ¿es posible que debido a las pruebas se esté haciendo un modelo que se ajuste a las pruebas pero no al comportamiento general? Es posible y realmente merece la pena mirar los modelos estadísticos para entenderlo un poco. El código, después de todo, debe ser sujeto a pruebas de sistema, de estrés, de rendimiento, de exploración que, por otro lado, están fuera del modelo bajo el cual se ha escrito el código.

2.2.3 Fases relevantes del TDD

Las principales fases del TDD se muestran en la figura 1.1 y se describen en las secciones a continuación.

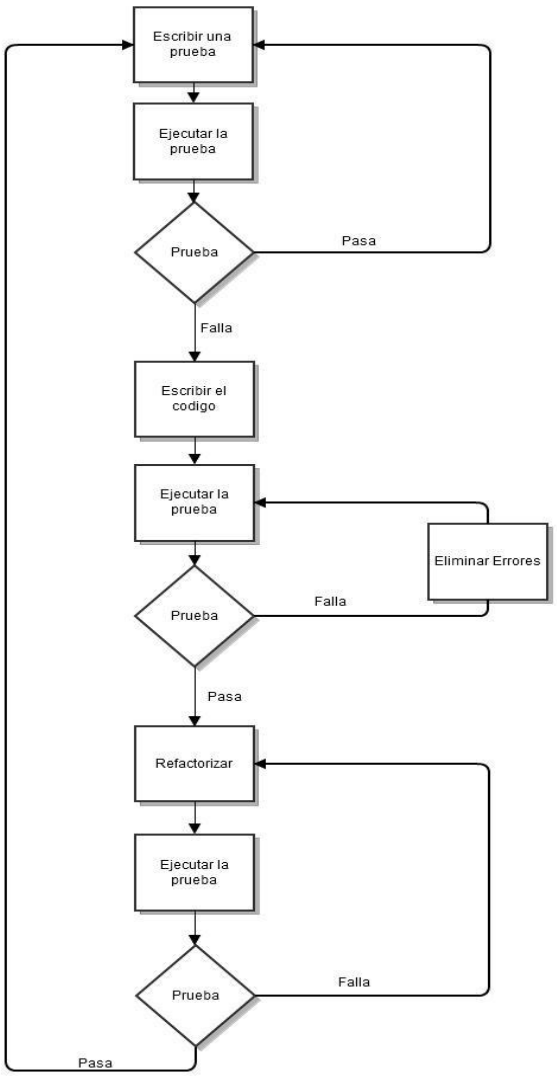

Figura 1.1 Fases relevantes del TDD

2.2.3.1 Escribir la Prueba (*Write Test*):

Las pruebas en TDD son llamadas pruebas de programador. Son como las pruebas unitarias pero con la diferencia de que están escritas sin que existan aún los métodos. (Astels, 2003).

Es importante que las pruebas sean escritas tal que las mismas sean independientes del orden, es decir, el resultados será el mismo independientemente de la secuencia en que se corran las pruebas. (Beck 2003).

Las pruebas se agrupan en conjuntos diferentes de acuerdo a lo que los comportamientos prueban. Esto permite probar una parte particular del sistema sin tener que ejecutar todas las pruebas. Un método de una prueba es la unidad más pequeña y debe probar un solo comportamiento, mientras que un caso de prueba reúne a las pruebas relacionadas.

Es una creencia común que un caso de prueba debe contener todas las pruebas de una clase específica. Este malentendido se ve reforzado por la mayoría de los *plugins* de Interfaces de Usuarios (IDE), ya que suelen generar un caso de prueba para cada clase conteniendo las pruebas para cada método.

Al escribir las pruebas se debe tener en cuenta que las pruebas deben concentrarse en probar múltiples métodos, es decir, si el sistema tiene que manejar múltiples entradas, entonces las pruebas deben reflejar las múltiples entradas. Es igualmente importante refactorizar las pruebas durante el ciclo de desarrollo, es decir, no debe haber una lista con 10 entradas si una lista de 3 entradas que lleva al mismo diseño y la misma implementación (Beck ,2003).

2.2.3.2 Correr la prueba (run test)

Las pruebas automáticas se deben ejecutar después de cada cambio del código de la aplicación con el fin de asegurar que los cambios no han introducido errores de la versión anterior del código (Astels 2003).

La ejecución de las pruebas se realiza generalmente con la ayuda de herramientas de pruebas unitarias, como JUnit (2005), que es un *framework* simple para escribir pruebas repetibles con Java.

En el ciclo TDD, la prueba se debe ejecutar después de escribir la prueba, después de escribir el código y también después de la refactorización.

2.2.3.3 Escribir código (*Write code*)

En TDD, la escritura de código es en realidad un proceso para hacer que la prueba funcione, es decir, escribir el código que pase la prueba.

Beck propone tres enfoques diferentes para hacer eso: la triangulación, "*fake it*", y la implementación obvia. Las dos primeras son las técnicas que son menos utilizadas en el trabajo de desarrollo en concreto. A pesar de ello, pueden ser útiles para la implementación de piezas más pequeñas en una solución más grande y más compleja, cuando el desarrollador no tiene una visión clara acerca de cómo poner en práctica o abstraer el sistema. A continuación se describen estos 3 enfoques:

1. "*fake it*" o fíngelo, puede emplearse por ejemplo, sustituyendo el valor devuelto de un método con una constante. Ofrece una forma rápida de hacer que la prueba pase. Tiene un efecto psicológico dándole confianza al programador para proceder con la refactorización y también se encarga del control de ámbito partiendo de un ejemplo concreto y luego generalizando desde allí. La implementación de "*fake it*" puede dar un nuevo impulso hacia la solución correcta, si el programador realmente no sabe por dónde empezar a escribir el código. (Beck, 2003)
2. La técnica de triangulación se puede utilizar para llegar a la abstracción correcta del comportamiento, es decir, la solución "*fake it*" no se puede utilizar más. Por ejemplo, hay por lo menos dos casos diferentes en la prueba del método que requieren diferentes valores de retorno y, obviamente, el retorno de la constante no satisface a ambos. Después alcanzar la implementación, la otra prueba pasa a ser redundante y debe ser eliminada. (Beck, 2003)
3. La implementación obvia se utiliza cuando el programador tiene confianza y sabe a ciencia cierta cómo implementar cierta operación. La práctica constante de la "implementación obvia" puede ser exhaustiva, ya que requiere la perfección. Cuando las pruebas comienzan a fallar de forma consecutiva, se recomienda utilizar la práctica del "*fake it*" o la "triangulación" hasta que vuelva la confianza. (Beck, 2003)

2.2.3.4 Refactorización (*Refactor*)

La refactorización es un proceso de mejora de la estructura interna mediante la modificación del código existente, sin cambiar su comportamiento externo. En TDD se utiliza para limpiar el código después de haber hecho lo más sencillo posible para hacer pasar la prueba. La Refactorización del código de la prueba es tan importante como refactorizar el código de la aplicación. (Astels 2003)

La idea de refactorización es llevar a cabo las modificaciones como una serie de pequeños pasos, sin introducir nuevos defectos en el sistema. Refactorización requiere un sólido conjunto de pruebas automatizadas, ya que generalmente se realiza manual, y el comportamiento externo del código puede cambiar sin querer en ese proceso, debido al factor humano. (Fowler 2002)

El TDD exige que los programadores no escriban código de funcionalidad si antes no han escrito pruebas unitarias que validen el código que se escribirá. La idea es bastante directa: De un requerimiento funcional se obtiene una responsabilidad atómica del actor (clasificador, clase, etc.) que se está modificando o creando, basados en esta responsabilidad se escribe el código que usaría esta funcionalidad, un código que además evalúa que la funcionalidad en cuestión haga lo que tiene que hacer. Este código es la prueba unitaria, que funciona como documentación y ejemplo de código. Con la prueba unitaria programada, se tiene que escribir el código para que la prueba tenga sentido semántico, o para que compile, en términos más directos. Cuando el código compila, se puede pasar a la parte en la que se programa lo necesario para pasar la prueba, y según el TDD, sólo se debe programar el código necesario para pasar la prueba. Si se ve que la prueba no es suficiente para garantizar la funcionalidad que se quiere, se hace otra prueba y se repite el proceso. De esta manera obtenemos un código probado que sólo hace lo que tiene que hacer.

2.3 Desarrollo Orientado a Comportamiento

El desarrollo orientado a comportamiento (DOC) es una técnica de desarrollo de software ágil que fomenta la colaboración entre los desarrolladores, control de calidad y participantes que no son técnicos ni de negocios en un proyecto de software (Viejo, 2010). Originalmente fue nombrado en 2003 por Dan North como respuesta al desarrollo orientado a pruebas (*Test Driven Development*, TDD), incluyendo las pruebas de aceptación (*Acceptance Test*) o las TDD orientadas al cliente, prácticas de desarrollo que se encuentran en *Extreme Programming* (XP).

DOC describe un ciclo de interacciones con resultados bien definidos, lo que resulta en la entrega del trabajo, el software probado que importa.

DOC se centra en la obtención de una comprensión clara del comportamiento de software deseado a través de discusiones con las partes interesadas. En cierto modo “hereda” del TDD escribiendo casos de pruebas en un lenguaje natural que incluso las personas sin experiencia en la programación podrían leer. Desarrolladores del DOC usan su lengua materna en combinación con el lenguaje omnipresente del diseño orientado a dominio para describir el propósito y beneficio de su código. Esto permite a los

desarrolladores centrarse en por qué el código debe ser creado, en vez de los detalles técnicos, y reduce al mínimo la traducción entre el lenguaje técnico en el que está escrito el código y el lenguaje de dominio hablado por el negocio, usuarios, grupos de interés, gestores de proyectos, etc.

Dan North creó el primer *framework* de DOC, *JBehave*, seguido por un *framework* para Ruby llamado *RBehave*, que se integró más tarde en el proyecto RSpec. Luego se creó Cucumber desarrollado por Aslak Hellesøy.

Cucumber es un Domain Specific Languaje (DSL, por sus siglas en inglés) que alienta el desarrollo orientado a comportamiento. Permite describir características o *features* antes de implementarlas.

En 2008, Chris Matts, que participó en las primeras discusiones en torno a DOC, se le ocurrió la idea de la inyección de característica (Feature Injection), lo que permite a DOC cubrir el espacio de análisis y proporcionar un tratamiento completo del ciclo de vida del software desde la visión a través de código y su entrega final.

2.3.1 Prácticas en DOC

Las prácticas en DOC ayudan a describir el comportamiento de un proceso en general. Primero viendo los actores que participan en el proceso; segundo, recorriendo las funcionalidades que estos realizan; tercero formulando las pruebas necesarias y por último escribiendo el código que superare estas pruebas que se desarrollaron. Las prácticas de DOC incluyen:

- Establecer los objetivos de los diferentes actores necesarios para la visión a ser implementada.
- Hacer surgir las características que permitan lograr esos objetivos mediante la *feature injection (*Inyección de características*).*
- La participación de los interesados en el proceso de implementación a través del desarrollo *outside-in (*desde afuera hacia adentro*).*
- Usar ejemplos para describir el comportamiento de la aplicación, o de trozos de código.
- Automatización de esos ejemplos para proporcionar información rápida y pruebas de regresión
- El uso del "*should*" (debe) al describir el comportamiento del software para ayudar a aclarar la responsabilidad y permitir a la funcionalidad del software ser cuestionada.
- El uso de "*ensure*" (asegura) a la hora de describir las responsabilidades del software para diferenciar los resultados en el ámbito del código en cuestión de los efectos secundarios de otros elementos de código.
- Uso de "*mocks*" (burlas) u obligando que la prueba pase así ésta no se haya escrito para poder desarrollar otros módulos deseados en ese momento.

2.3.2 Inyección de características (Feature Injection)

Una empresa puede tener varias visiones que dan valor a la empresa, por lo general ganando dinero, ahorrando dinero o protegiendo el dinero. Una vez que una visión es identificada por un grupo como la mejor visión para las condiciones, necesitarán ayuda adicional para que la visión sea un éxito.

Cada actor o *stakeholder* define los objetivos que necesitan para lograr el cumplimiento de la visión para tener éxito. Por ejemplo, un departamento legal puede pedir ciertos requisitos reglamentarios que deben cumplir. El director de marketing puede ser que desee que la comunidad que va a utilizar el software participe. Un experto en seguridad se asegura de que el software no será vulnerable a ataques de inyección SQL.

A partir de estos objetivos, se alcanza mediante temas generales o conjuntos de características que se definen. Por ejemplo, "permitir a los usuarios clasificar las contribuciones" o "auditar transacciones".

A partir de estos temas se pueden establecer las características (*user features)* y los primeros detalles de la interfaz de usuario.

2.3.3 Desde afuera hacia adentro (Outside-in)

DOC es conducido por el valor del negocio (wikimedia, 2010), es decir, el beneficio para la actividad que se acumula una vez que la aplicación está en producción. La única manera en que este beneficio se puede realizar es a través de la interfaz de usuario hacia la aplicación, por lo general (pero no siempre) una UI.

Cada elemento de código proporciona algún aspecto del comportamiento que, en colaboración con los otros elementos, proporciona el comportamiento de la aplicación.

La primera pieza de código de producción que los desarrolladores de DOC implementan es la interfaz de usuario. Los desarrolladores pueden beneficiarse de una rápida retroalimentación en cuanto si la interfaz de usuario se ve y se comporta adecuadamente. A través del código y utilizando principios de buen diseño y refactorización, los desarrolladores descubrieron que la interfaz de usuario colabora, y cada unidad de código a partir de entonces también. Esto les ayuda a cumplir con el principio de NLVAN, ya que cada pedazo de código de producción se requiere, ya sea por el negocio, o por otra pieza de código ya escrito.

2.3.4 La convención Given/when/Then

DOC es una variación de TDD. Mientras que en TDD se maneja el desarrollo indicando en primer lugar los requisitos como pruebas unitarias, en DOC se maneja en primer lugar el desarrollo indicando los requisitos (Barbero, 2008). La forma de los requisitos es sólida, lo que les permite ser interpretada por una herramienta que puede ejecutar de una manera que es similar a las pruebas unitarias.

Por Ejemplo,

GIVEN un empleado llamado Juan que gana 12 Bs la hora.

WHEN Juan trabaja 40 horas en una semana;

THEN A Juan se le pagara 480 bs en la noche del viernes.

La convención *Given/When/Then* es fundamental para la noción de DOC. Se conecta el concepto humano de causa y efecto, con el concepto de software de la entrada/proceso/salida, es decir, trata de hacer que la programación sea más natural y menos mecánica.

El beneficio argumentado es que el idioma de uso afecta a la manera de pensar, es decir, no es lo mismo que una persona piense en un idioma aprendido a que lo haga en su lenguaje materno. Para decirlo de otra manera, la convención *Given/When/Then* estimula mejor los procesos de pensamiento que la convención *AssertEquals(expected, actual).*

Cabe destacar que la convención *Given/When/Then*, puede ser escrita en múltiples idiomas haciendo una pequeña modificación en la configuración. Un ejemplo en español seria la convención Dado/Cuando/Entonces que significaría lo mismo que *Given/When/Then.*

En el ciclo básico de desarrollo para DOC se encuentran las siguientes tareas:

- Discusión: Desarrolladores, probadores, expertos de dominio y el dueño del producto se juntan y discuten la historia, gradualmente descomponiendo y destilando el comportamiento en un conjunto de especificaciones simples.
- Decisión: El dueño del producto decide cuando la especificación cumple suficientemente el comportamiento esperado y está de acuerdo con los casos de ejemplo y demostración de la historia y cierra el alcance de la misma.
- Desarrollo: Los probadores refinan los ejemplos de la especificación y los desarrolladores instrumentan las especificaciones creando primero pruebas que fallan y después implementando la funcionalidad.

- Demostración: Una vez que todas las pruebas pasan, la historia puede ser demostrada al dueño del producto.

2.3.5 Burlas (Mocks)

El Desarrollo Basado en Comportamiento, desafía a los desarrolladores a cuestionar cuáles son las responsabilidades que ellos asignan a sus clases, son apropiadas o pueden delegarse o moverse a otra clase. Cuestionarse de esta forma y utilizar Mocks para llenar los roles requeridos de colaboración de las clases, promueve el uso de interfaces basadas en roles, y esto también ayuda a mantener las clases pequeñas y desacopladas.

Los *Mocks* o burlas son una forma de hacer que el comportamiento sea válido sin ningún problema aun cuando todavía no se ha implementado, con el fin de desarrollar otros módulos primero que pueda necesitar que este comportamiento esté terminado.

El DOC permite a los desarrolladores concentrarse en por qué el código debería ser creado, en vez de los detalles técnicos, y minimiza la traducción entre el lenguaje en el cual se escribe y el lenguaje hablado por los usuarios.

Capítulo II: Marco Aplicativo

En este capítulo se enuncian las bases del trabajo de investigación y se demuestra la aplicación de los conocimientos que se encuentran en la Capitulo I en una situación real.

Esta investigación consiste en aplicar desarrollo orientado a comportamiento para generar un módulo para el sistema CONEST (es un sistema de Gestión Académica y Administrativa, desarrollado en la Facultad de Ciencias). Este módulo se refiere a una actividad que no se realiza de forma automática y se busca un estándar donde todas las escuelas puedan utilizar una interfaz unificada para tratar de mejorar este proceso. El desarrollo orientado a comportamiento comprende 4 tareas básicas en su desarrollo: discusión, en la que se realizan reunión con los clientes para hacer el levantamiento de requerimientos; decisión, en la que se define el ciclo de desarrollo de la implementación; desarrollo, donde se codifica e implementa el código de la aplicación y finalmente demostración, donde se presenta el producto final del desarrollo al cliente a fin de validar el cumplimiento de los requerimientos identificados.

Este trabajo ayuda a la Facultad de Ciencias de la UCV a simplificar tanto el trabajo administrativo que se lleva a cabo, simplificar errores de datos que vienen dados por la redundancia de los mismos o por errores humanos, así como reducir el tiempo que tardaría realizar los procesos involucrados en la solicitud de jurado Seminario y TEG.

3.1 Planteamiento del Problema

El procesamiento de las solicitudes de jurado tanto para TEG como para Seminario, es un trabajo delicado que toma tiempo en revisiones y verificaciones de los expedientes de los estudiantes involucrados en la solicitud. Este trabajo se hace de forma manual y ocasiona que, en algunos casos, se retrasen los procesos a realizar, ya que la información se encuentra archivada en numerosos archivos de la División de Control de Estudios. Una vez procesada una solicitud, el resultado es difundido a través de actas físicas emitidas desde los consejos de escuela, lo que complica hacer llegar la información a tutores y jurados externos.

Tomando en cuenta aquellos procesos que son estrictamente necesarios para cada Escuela dentro de la Facultad de Ciencias, se busca automatizar el proceso de solicitud de jurado para TEG y seminario para aminorar tanto el tiempo de ejecución, como los recursos utilizados, facilitar la obtención de resultados frente a docentes, jurados y otros involucrados en una solicitud.

3.2 Objetivo general

- Desarrollar el módulo de solicitudes de jurado para Seminario y Trabajo Especial de Grado utilizando técnicas de desarrollo orientado a comportamiento.

3.3 Objetivos específicos

Para lograr el objetivo general se derivan los siguientes objetivos específicos:

- Describir desarrollo orientado a comportamiento como una metodología ágil.
- Levantar los requerimientos del módulo a desarrollar con los usuarios finales para identificar las distintas necesidades de cada tipo de usuario y de cada Escuela perteneciente a la Facultad de Ciencias.
- Discutir los procesos críticos y escenarios que deben ser desarrollados para procesar una solicitud de jurado para TEG y Seminario.
- Describir el ciclo de desarrollo, la clasificación de los escenarios a desarrollar en la aplicación.
- Desarrollar pruebas y código en lenguaje *Ruby*, utilizando el framework *Ruby On Rails* que esten apoyado por los escenarios descritos.
- Aplicar pruebas de aceptación del módulo desarrollado con los usuarios finales.

3.4 Justificación

A menudo los sistemas web son desarrollados siguiendo procesos de desarrollo ágiles, el problema aparece al momento de definir las pruebas, debido a la naturaleza cambiante del sistema y de la web, definirlas en una fase temprana del desarrollo resulta en muchos casos complicado.

Aplicar un módulo de pruebas a un sistema ya desarrollado, incrementado y modular resulta bastante complicado, una solución a este problema es aplicar desarrollo basado en comportamiento en el cual las pruebas son definidas desde el principio, sin embargo, supone regresar al proyecto a un estado de desarrollo inicial para poder ser adaptado a las pruebas de comportamiento.

El desarrollo orientado a comportamiento propone desarrollar las pruebas en una fase inicial de la implementación, teniendo claro cuáles son las funcionalidades que se desean lograr con la aplicación, y de esta manera evitar desarrollar módulos y funcionalidades de la aplicación que se escapan del alcance de la misma. La manera en que están descritas estas pruebas, sirven de documentación de la aplicación desarrollada, ya que están descritas en lenguaje natural señalando de manera puntual una funcionalidad y los pasos para cumplirla.

La automatización de un proceso sensible para las escuelas y donde intervienen tantos entes diferentes como lo el proceso de aprobación de solicitudes de jurado para TEG y Seminarios, es una

mejora en el sistema CONEST utilizado por la División de Control de Estudios, ya que además de incrementar las funcionalidades del sistema, provee de una herramienta mediante la cual se resuelve los problemas de seguimiento, control y difusión de los resultados de las aprobaciones de Solicitudes de jurados, así como reduce los recursos utilizados para dicho proceso, apoyándonos de la tecnología y dejando a un lado el proceso manual.

3.5 Estrategia general de solución

Para describir desarrollo orientado a comportamiento (DOC) como una metodología ágil, debido a que existe poca documentación formal al respecto, se busca información en medio no impresos, como el internet. Se toman ejemplos de experiencias de usuarios que ya han implementado esta tecnología logrando sus objetivos para que sirvan de guía.

Las reuniones con los diferentes tipos de usuarios involucrados de las distintas Escuelas son necesarias para unificar todos los requerimientos y realizar una interfaz que logre satisfacer las distintas necesidades individuales de la escuela, sin dejar de lado las tareas comunes realizadas por todas las escuelas.

Se escriben los escenarios de prueba bajo las gemas en *Ruby* '*Cucumber*' y '*Rspec*' usando la convención *Given/When/Then*, y a partir de ellas incrementar el código, generando vistas y sus controladores para que cumplir con las condiciones que le permitirán al sistema superar de forma exitosa las pruebas.

El módulo fue implementado a partir de una versión para desarrollo suministrada por el Prof. Sergio Rivas encargado del desarrollo del Sistema CONEST y desarrollada en la Escuela de Computación, la misma está implementada para la versión de *Ruby* 1.8.7, y *Ruby on Rails* 3.0.4; bajo ambiente Linux en su distribución Ubuntu versión 11.10.

Para el desarrollo web del sistema se utilizan diferentes herramientas, como por ejemplo *Haml* versión 3.1.3 para el desarrollo de las vistas de una manera más práctica, ordenada e intuitiva; *Prawn* 0.12.0 para la generación dinámica de PDF y *CoffeeScript* 2.2.0 para desarrollar código *Javascript* de una manera más sencilla.

Una vez terminada la aplicación, se presenta a los usuarios necesarios a fin de validar la funcionalidad y aceptabilidad del sistema.

A continuación se describe el desarrollo de la aplicación, así como algunos de sus artefactos y la descripción de sus resultados.

3.6 Desarrollo de la Aplicación

Antes de comenzar a explicar el desarrollo de la aplicación, se debe resaltar la existencia de un "kit de desarrollo" para módulos del Sistema CONEST que cuenta con el esqueleto básico de la aplicación, como las vistas, controladores y modelos básicos y comunes entre los módulos.

También fue suministrada la Base de Datos, la cual cuenta con más de 100 tablas, sin embargo sólo se un conjunto limitado de las mismas, éstas son las que tienen impacto sobre el módulo desarrollado y son demostradas en la figura 2.1.

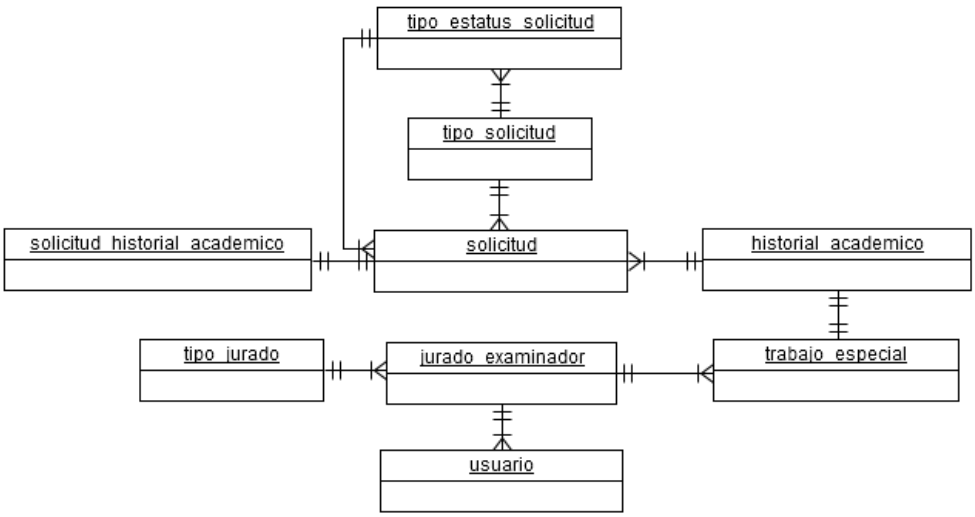

Figura 2.1 Diagrama de Tablas utilizadas en la Aplicación

La aplicación fue desarrollada utilizando la metodología de DOC, la cual se centra en la obtención de una comprensión clara del comportamiento de software deseado a través de discusiones con las partes interesadas tal como fue descrita en la sección 1.3 del Capítulo anterior, propone escribir casos de pruebas en un lenguaje natural que incluso las personas sin experiencia en la programación podrían leer.

Para la realización de la aplicación, se utiliza el enfoque "*Outside-In*" que sugiere la realización de las interfaces de la aplicación antes de los controladores. Esto para mejorar los tiempos de codificación ya que ayuda a identificar los segmentos de código necesarios para la aplicación.

La forma de escribir los requisitos viene atada a la convención *Given/When/Then* que es fundamental para la noción de DOC, la cual se conecta con el concepto humano de causa y efecto. Debido a que cuenta con la formalidad suficiente, se utiliza una herramienta llamada C*ucumber*, que ha sido escrita

para interpretar el requerimiento en lenguaje natural y luego manejar el sistema bajo prueba para asegurarse que el requerimiento trabaja según lo indicado.

Debido al amplio alcance de la aplicación se puede hacer una clasificación de estas tareas agrupándolas en niveles de detalle, de esta manera se obtiene el Enfoque Global y el Enfoque por Usuarios. Se comienza describiendo el Enfoque Global de la aplicación.

3.6.1 Enfoque Global

A continuación se comienza a describir la experiencia al realizar la aplicación, siguiendo las tareas básicas de discusión, desarrollo y demostración buscando describir de manera global la misma.

3.6.1.1 **Discusión:** se comienza por recopilar información necesaria sobre el proceso de solicitud de jurado para Tesis de Grado en cada una de las 5 escuelas que conforman la Facultad de Ciencias; esto se logra realizando reuniones con las partes pertinentes, la bitácora de estas reuniones se encuentra en los Anexos (Ver anexo 1). Una vez recopilada la información se pueden modelar los procesos críticos necesarios para el sistema, y los datos que el usuario debe ingresar para los mismos. El resultado de este levantamiento de información, está reflejado en los diagramas de actividades que se describen a continuación:

Como se describe en la Figura 2.2, el primer proceso define las actividades del tutor, éste envía solicitud de jurado al Consejo de Escuela para que éste la procese.

Figura 2.2 Entrega de Solicitud a Consejo de Escuela

El siguiente proceso, definido en la Figura 2.3 y descrito a continuación, muestra la comunicación y pasos a realizar entre el Tutor y la Comisión de Trabajo Especial de Grado (CTEG, en las Escuelas donde aplique):

- Enviar solicitud de jurado a la Comisión de TEG (CTEG): El Tutor envía la planilla de solicitud de jurado a la CTEG y espera resultado de aprobación por parte del Consejo de Escuela.
- Recibir solicitud de jurado: La CTEG recibe la planilla de Solicitud de jurado por parte del Tutor.
- Consultar Jurado: La CTEG estudia el jurado propuesto por el Tutor para garantizar que cada Jurado tiene la misma cantidad de trabajo entre los diferentes TEG de la Escuela.
- Asignar jurado tentativo: La CTEG puede o no modificar la solicitud inicial de jurado tomando en cuenta el estudio anterior.
- Enviar solicitud al Consejo de Escuela (CE): La planilla es enviada al CE para discusión.

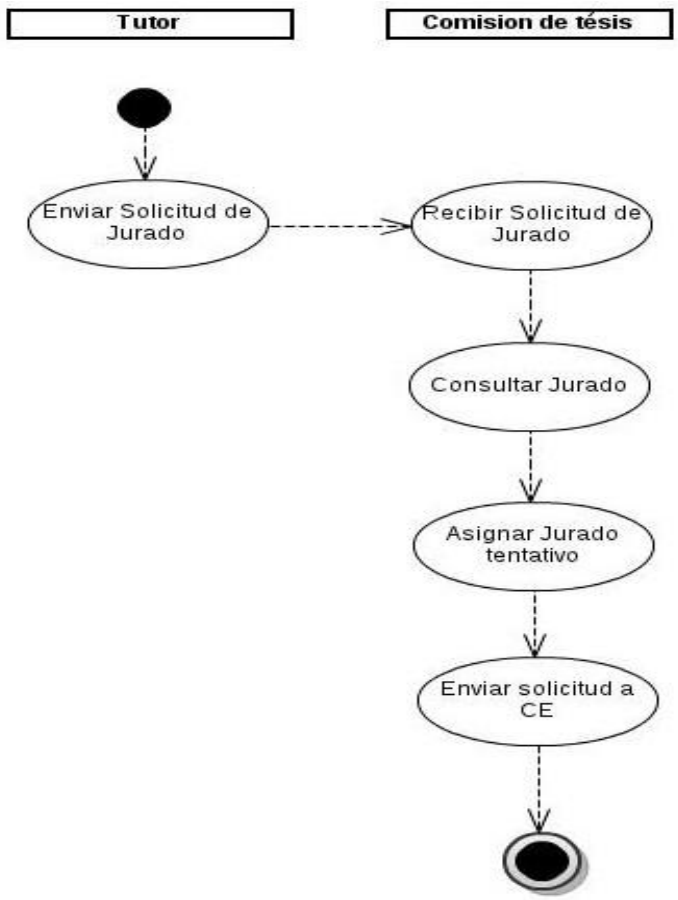

Figura 2.3 Sub-Proceso entre Tutores y Escuelas con Comisión de Grado

La Figura 2.4 nos muestra la interacción entre el Consejo de Escuela, la División de Control De Estudios (DCE), Tutores, jurados y estudiantes involucrados, sus tareas son las siguientes:

- Recibir Solicitud de jurado: El Consejo de Escuela recibe la planilla de solicitud de jurado
- Discutir punto de minuta en Consejo de Escuela de fecha DD/MM/AAAA: El Consejo de Escuela evalúa la solicitud y toma una decisión con respecto a su aprobación. Si la solicitud no es aprobada, se envía una notificación al Tutor indicando el motivo. Si la solicitud es aprobada, se asigna una fecha de presentación.
- Entregar documento físico a la División de Control de Estudios (DCE): Consejo de Escuela entrega el documento donde se encuentra el jurado aprobado y la fecha de discusión de la Solicitud.
- Recibir documento físico desde Consejo de Escuela: La DCE recibe el documento enviado por el Consejo de Escuela. La DCE también verifica los requisitos para la presentación de tesis del bachiller (revisa el Kardex, expediente y Pensum). Solamente si este cumple los requisitos, se continúa el proceso.
- Hacer efectiva la aprobación de jurado: La DCE debe ingresar al sistema los datos del jurado aprobado.
- Proceso en paralelo:
    - Enviar notificación a jurado/estudiante: La DCE envía una notificación al jurado aprobado y al Estudiante indicando la aprobación de la Solicitud de jurado para TEG.
    - Recibir notificación de DCE: El jurado aprobado y el estudiante reciben la notificación de la aprobación de la solicitud de jurado para TEG.
    - Enviar notificación a Tutor: La DCE envía una notificación al Tutor indicando la aprobación de su solicitud de jurado.
    - Recibir notificación de DCE: El Tutor recibe la notificación de la aprobación de la solicitud de jurado.
- Fin del procesamiento en paralelo
- Solicitar planilla de Acta Final de Notas: El Tutor solicita el Acta Final de Notas del estudiante a la DCE.
- Recibir solicitud de planilla de Acta Final de Notas: La DCE recibe la solicitud del acta final de notas del Estudiante.
- Imprimir Acta Final de Notas: La DCE imprime el Acta Final de Notas solicitada.

- Entregar Acta Final de Notas: La DCE entrega al Tutor el Acta Final de Notas solicitada.
- Recibir Acta Final de Notas: El Tutor recibe el Acta Final de Notas solicitada.
- Consignar Acta Final de Notas: El Tutor entrega a la DCE el Acta Final de Notas con la nota de presentación del jurado.

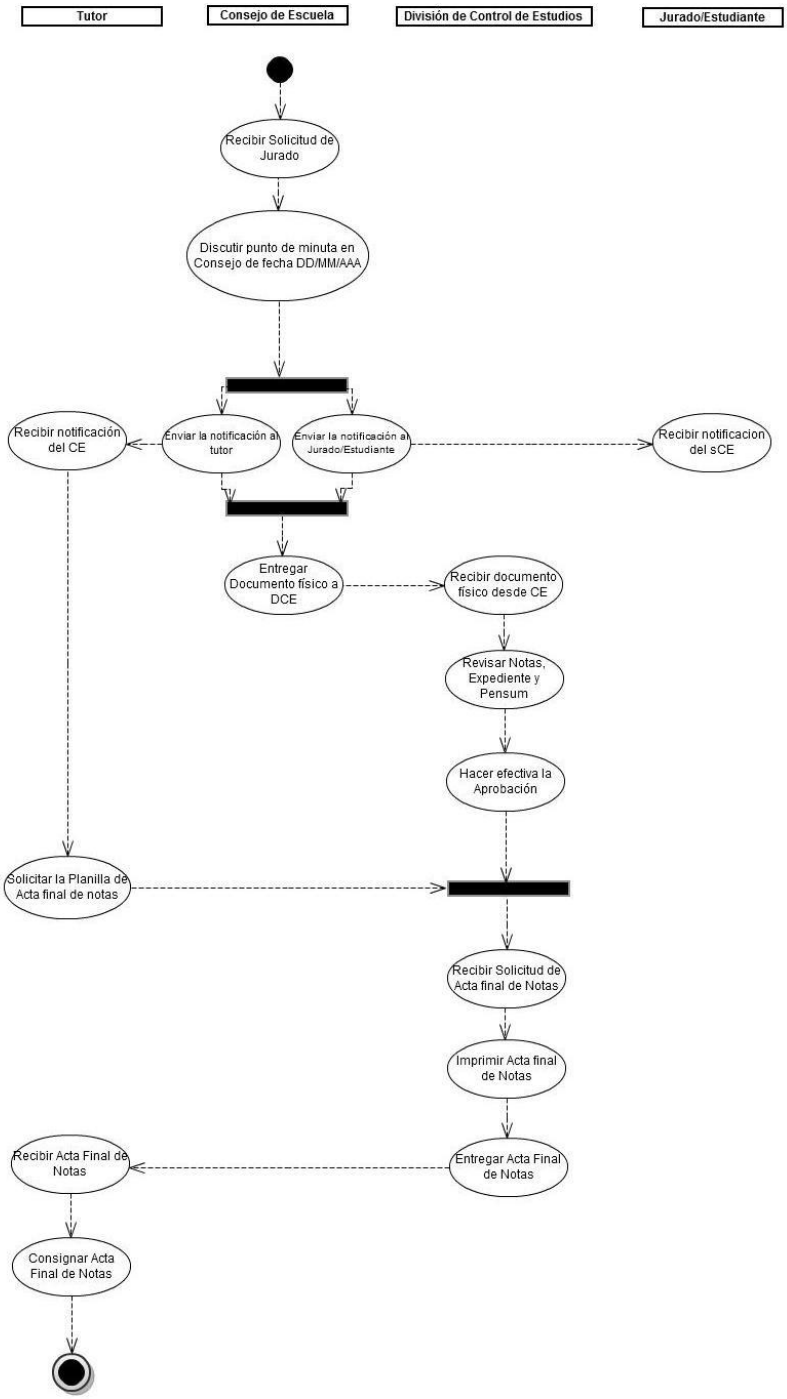

Figura 2.4 Proceso Interno entre Consejo de Escuela, Tutor, DCE y Estudiante

La Figura 2.5 muestra el proceso del procesamiento de las solicitudes de jurado para Seminario, sus tareas serán descritas a continuación:

- Solicitar la Planilla de Acta de Notas: una vez realizada la propuesta de jurado, el Tutor debe solicitar la impresión del Acta Final de Notas días antes de la presentación del Seminario.
- Recibir Solicitud de Acta Final de Notas: la DCE acusa de recibida la solicitud en línea de Impresión de Acta final de notas, y se envía una notificación al Tutor.
- Imprimir Acta de Notas: la DCE se realiza la impresión del Acta final de notas, de acuerdo con la Solicitud.
- Entregar Acta de Notas: la DCE una vez impreso el documento, se envía una notificación al Tutor para la entrega del documento físico.
- Recibir Acta de Notas: el Tutor acusa de recibido el documento físico y lo llena con los resultados de la presentación.
- Consignar Acta de Notas: el Tutor entrega el acta con la información de la presentación a la DCE.

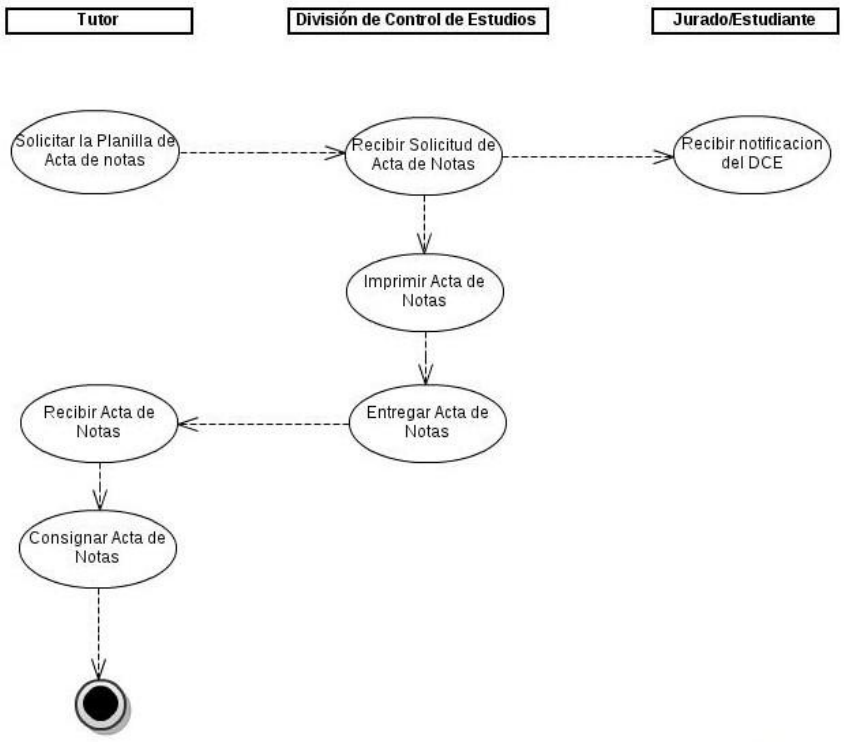

Figura 2.5 Solicitudes de jurado para Seminarios

Una vez definidos, los procesos fueron clasificados por usuarios para definirlos con mayor facilidad.

3.6.1.2 **Decisión:** Una vez definidos los procesos y sus actores, se pasa a definir las pruebas necesarias para cumplir los objetivos de cada proceso. Para esto se divide el proceso tomando en cuenta el actor principal involucrado en cada actividad. Las pruebas preliminares se mostraron a los tutores Eugenio Scalise , Jossie Zambrano y el profesor encargado del Sistema CONEST Sergio Rivas, quienes aprobaron las pruebas aunque observando que éstas pudieran evolucionar e incrementar a medida que se avanza en el desarrollo.

Ahora se enfoca en las pruebas definidas para los usuarios docentes.

3.6.2 Enfoque por usuario

3.6.2.1 Discusión: Una vez definidos los procesos en los cuales participaba cada tipo de usuario se procedió a refinarlos y clasificarlos de acuerdo a su funcionalidad dentro del sistema y su interacción con otros procesos y usuarios.

3.6.2.2 Decisión: Se identifican los procesos principales y necesarios para evitar describir procesos adicionales. Para cada proceso se escriben pruebas que busquen comprobar su funcionalidad y que serán implementadas en la siguiente tarea de desarrollo. Se describen las pruebas y se agrupan los pasos a fin de minimizar el código de pruebas sin comprometer el entendimiento de las mismas. Ya que se utiliza DOC las pruebas fueron definidas utilizando el DSL de Cucumber, el cual cuenta con las siguientes palabras clave**:**

- Escenario *(Feature)*: Describe la funcionalidad y todos los pasos que la misma tiene que seguir para poder completarse.
- Dado *(Given)*: Es siempre el contexto inicial donde se marca el comienzo de la funcionalidad. Describe un primer momento donde la sentencia debe ser verdadera para que el usuario pueda comenzar a realizar la funcionalidad descrita en el escenario.
- Y (*And*): Es la continuación de la sentencia que debe ser verdadera para que el escenario se realice correctamente.
- Cuando (*When*): Se refiere al evento que debe ocurrir cuando ya se está en el contexto inicial para que la funcionalidad pueda llevarse a cabo.

- Entonces (*Then*): Es el resultado que se debe asegurar si se dieron los pasos anteriores en el feature para saber que el comportamiento del software es correcto.

A continuación se describen las pruebas preliminares principales para cada usuario:

3.6.2.2.1 Docente

Es el usuario encargado de crear la solicitud frente al sistema y así comenzar una nueva instancia del proceso. Sirve también de intermediario con los estudiantes frente al sistema. Es el usuario al cual se responde por el procesamiento de la solicitud.

Para el usuario docente se identificaron 5 Escenarios, que son descritos a continuación:

Crear una nueva solicitud de jurado para Seminario

Para este Escenario se presenta la interfaz del formulario será presentada en la Figura 2.6, definida por los siguientes Pasos (*Steps*)

```
Escenario: Ver formulario de nueva Solicitud de Jurado
Dado que estoy en la pagina de inbox para Solicitudes Docentes
Cuando clickeo "Seminario"
Entonces debería estar en la pagina de formulario para nueva Solicitud de Jurado para
Seminario
Y debería ver "Solicitud de Jurado para Seminario"
Y debería ver "Título"
Y debería ver "Fecha aproximada de presentación"
Y debería ver "Datos de los estudiantes y tutores adicionales"
Y debería ver "Indique nombre, apellido o cédula del Estudiante"
Y debería ver "Indique nombre, apellido o cédula del Estudiante (En caso que sean más de
un estudiante)"
Y debería ver "Indique nombre, apellido o cédula del 2do tutor"
Y debería ver "¿Es Externo?"
Y debería ver "Seleccione su propuesta de Jurados"
Y debería ver "Nombre, apellido o cédula del 1er Jurado Principal"
Y debería ver "Nombre, apellido o cédula del 2do Jurado Principal"

Escenario: Crear nueva Solicitud de Jurado para Seminario con Tutor en el sistema y un
solo estudiante
Dado que estoy en la página de inbox para Solicitudes Docentes
Cuando voy a la página de formulario para nueva Solicitud de Jurado para Seminario
Y completo "tutor2" con "Nieves Adelis"
Y selecciono "Alvarez Adonahis" de "principal1"
Y selecciono "Bottini Adrian" de "principal2"
Y selecciono "De sousa Andreina" de "cedula_estudiante1"
Y completo "titulo_SEM" con "Titulo tentativo de SEM"
 TEG,..."
Y completo “fecha” con “01/01/2012”
Y apreto "Crear Solicitud"
Entonces debería ver "Solicitud y SEM Creados"
Y debería ver "Estudiantes:"
Y debería ver "De Sousa Andreina"
Y debería ver "Título:"
Y debería ver " Título tentativo de TEG"
Y debería ver "Fecha aproximada de presentación"
Y debería ver "01/01/2012"
Y debería ver "Principales:"
Y debería ver "Alvarez Adonahis"
Y debería ver "Bottini Adrian"
```

Crear una nueva solicitud de jurado para Trabajo Especial de Grado

Este Escenario fue presentado en el Diagrama de Actividades de la Figura 2.3 y la interfaz del formulario será presentada en La Figura 2.8, definida por los siguientes Pasos (*Steps*)

```
Escenario: Ver formulario de nueva Solicitud de Jurado
Dado que estoy en la página de inbox para Solicitudes Docentes
Cuando clickeo "Jurado para TEG"
```

```
Entonces debería estar en la página de formulario para nueva Solicitud de Jurado para TEG
Y debería ver "Solicitud de Jurado para Trabajo Especial de Grado"
Y debería ver "Título"
Y debería ver "Resumen"
Y debería ver "Palabras clave"
Y debería ver "Fecha aproximada de presentación"
Y debería ver "Datos de los estudiantes y tutores adicionales"
Y debería ver "Indique nombre, apellido o cédula del Estudiante"
Y debería ver "Indique nombre, apellido o cédula del Estudiante (En caso que sean más de
un estudiante)"
Y debería ver "Indique nombre, apellido o cédula del 2do tutor"
Y debería ver "¿Es Externo?"
Y debería ver "Seleccione su propuesta de Jurados"
Y debería ver "Nombre, apellido o cédula del 1er Jurado Principal"
Y debería ver "Nombre, apellido o cédula del 1er Jurado Suplente"
Y debería ver "Nombre, apellido o cédula del 2do Jurado Principal"
Y debería ver "Nombre, apellido o cédula del 2do Jurado Suplente"
```

```
Escenario: Crear nueva Solicitud de Jurado para TEG con Tutor en el sistema y un solo
Estudiante
Dado que estoy en la página de inbox para Solicitudes Docentes
Cuando voy a la página de formulario para nueva Solicitud de Jurado para TEG
Y completo "tutor2" con "Nieves Adelis"
Y selecciono "Alvarez Adonahis" de "principal1"
Y selecciono "Bottini Adrian" de "principal2"
Y selecciono "Figueredo Adriana" de "suplente1"
Y selecciono "Castillo Aiglee" de "suplente2"
Y selecciono "De sousa Andreina" de "cedula_estudiante1"
Y completo "titulo_TEG" con "Titulo tentativo de TEG"
Y completo "resumen_TEG" con "Resumen corto del TEG"
Y completo "palabras_TEG" con "Palabra clave para TEG, palabra clave TEG,..."
Y completo "fecha" con "01/01/2012"
Y apreto "Crear Solicitud"
Entonces debería ver "Solicitud y TEG Creados"
Y debería ver "Estudiantes:"
Y debería ver "De Sousa Andreina"
Y debería ver "Título:"
Y debería ver "Título tentativo de TEG"
Y debería ver "Resumen:"
Y debería ver "Resumen corto del TEG"
Y debería ver "Palabras claves:"
Y debería ver "Palabra clave para TEG, palabra clave TEG,..."
Y debería ver "Fecha aproximada de presentación"
Y debería ver "01/01/2012"
Y debería ver "Principales:"
Y debería ver "Alvarez Adonahis"
Y debería ver "Bottini Adrian"
Y debería ver "Suplentes:"
Y debería ver "Figueredo Adriana"
Y debería ver "Castillo Aiglee"
```

Ver solicitudes de jurado para Trabajo Especial de Grado y Seminario pendientes

Se presenta la interfaz de este escenario en la Figura 2.10, y se define de la siguiente manera:

```
Escenario: Ver estado de Solicitudes de Jurado para TEG
Dado que estoy en la página de inbox para solicitudes docentes
Entonces debería ver "Solicitudes de Jurado para TEG Pendientes"
Y debería ver "Solicitudes de Jurado para Seminario Pendientes"
```

Modificar solicitud de jurado para Trabajo Especial de Grado

Estos Escenarios pueden verse en las Figuras 2.13 y 2.14, definiéndose de la siguiente manera:

```
Escenario: Ver formulario de modificación de Solicitud de Jurado para TEG por tutor
Dado que he creado una nueva solicitud para TEG
Y que estoy en la página de inbox para Solicitudes Docentes
Cuando clickeo "Modificar Solicitud"
Entonces debería estar en la página de formulario para modificación de Solicitud de
Jurado para TEG por tutor
```

```
Escenario: Modificar Tutor en sistema por nuevo Tutor en el sistema para Solicitud de
Jurado para TEG por tutor
```

```
Dado que he creado una nueva Solicitud para TEG
Y que estoy en la página de inbox para Solicitudes Docentes
Cuando clickeo "Modificar Solicitud"
Y completo "tutor_2" con "Tutor Externo"
Y selecciono "Jurado 1" de "principal_1"
Y selecciono "Jurado 2" de "principal_2"
Y selecciono "Jurado 3" de "suplente_1"
Y selecciono "Jurado 4" de "suplente_2"
Y selecciono "Estudiante 1" de "cedula_estudiante_1"
Y selecciono "Estudiante 2" de "cedula_estudiante_2"
Y completo "titulo_TEG" con " Título tentativo de TEG"
Y completo "area_TEG" con "Área de Investigación de TEG"
Y completo "resumen_TEG" con "Resumen corto del TEG"
Y completo "palabras_clave_TEG" con "Palabra clave para TEG, palabra clave TEG,..."
Y completo “fecha” con “01/01/2012”
Y apreto "Modificar Solicitud"
Entonces debería ver "Solicitud y TEG Creados"
Y debería ver "Estudiantes:"
Y debería ver "Estudiante 1"
Y debería ver "Estudiante 2"
Y debería ver "Título:"
Y debería ver "Título tentativo de TEG"
Y debería ver "Área:"
Y debería ver "Área de Investigación de TEG"
Y debería ver "Resumen:"
Y debería ver "Resumen corto del TEG"
Y debería ver "Palabras claves:"
Y debería ver "Palabra clave para TEG, palabra clave TEG, ..."
Y debería ver "Fecha aproximada de presentación"
Y debería ver "01/01/2012"
Y debería ver "Principales:"
Y debería ver "Jurado 1"
Y debería ver "Jurado 2"
Y debería ver "Suplentes:"
Y debería ver "Jurado 3"
Y debería ver "Jurado 4"
```

3.6.2.2.2 Comisión de Trabajo Especial de Grado

Compuesta por miembros docentes de cada Escuela, sirve de filtro entre los documentos entregados por los tutores y el Consejo de Escuela, evitando errores relacionados con la falta de documentación y requerimientos estudiantiles para el procesamiento de las solicitudes.

Para este usuario se identificaron 2 escenarios generales:

- Ver solicitudes de jurado para Trabajo Especial de Grado pendientes por procesar.
- Modificar solicitud de jurado para Trabajo Especial de Grado.

Debido a su alto parecido y similitud a las interfaces desarrolladas para el Consejo de Escuela, estas serán omitidas.

3.6.2.2.3 Consejo de Escuela

Está, al igual que la Comisión de Trabajo Especial de Grado, conformado por miembros docentes de cada Escuela respectiva de la Facultad. Son los encargados de emitir decisiones definitivas frente a una solicitud de jurado y hacerlas llegar a la División de Control de Estudio y a quien pueda interesar.

Para este usuario se identificaron 2 escenarios, que son descritos a continuación:

Ver Solicitudes de jurado para Trabajo Especial de Grado pendiente

Este Escenario será presentado en la Figura 2.18, y se define a continuación:

```
Escenario: Ver estado de Solicitudes de Jurado para TEG
```

```
Dado que estoy en la página de inbox para Solicitudes Docentes
Entonces debería ver "Solicitudes de Jurado para TEG Pendientes"
```

Modificar solicitud de jurado para Trabajo Especial de Grado

Estos Escenarios se presentarán en a las Figuras 2.18 y 2.19, definiéndose de la siguiente manera:

```
Escenario: Ver formulario de modificación de Solicitud de Jurado para TEG por consejo de
escuela
Dado que he creado una nueva solicitud para TEG
Y que estoy en la página de inbox para Solicitudes Docentes
Cuando clickeo "Modificar Solicitud"
Entonces debería estar en la página de formulario para modificación de Solicitud de
Jurado para TEG por consejo de escuela
```

```
Escenario: Modificar tutor en sistema por nuevo tutor en el sistema para Solicitud de
Jurado para TEG por consejo de escuela
Dado que he creado una nueva solicitud para TEG
Y que estoy en la página de inbox para Solicitudes Docentes
Cuando clickeo "Modificar Solicitud"
Y selecciono "Jurado 1" de "principal_1"
Y selecciono "Jurado 2" de "principal_2"
Y selecciono "Jurado 3" de "suplente_1"
Y selecciono "Jurado 4" de "suplente_2"
Y completo “fecha” con “01/01/2012”
Y aprieto "Modificar Solicitud"
Entonces debería ver "Solicitud de Jurados Modificada"
Y debería ver "Estudiantes:"
Y debería ver "Estudiante 1"
Y debería ver "Estudiante 2"
Y debería ver "Titulo:"
Y debería ver "Titulo tentativo de TEG"
Y debería ver "Area:"
Y debería ver "Area de Investigacion de TEG"
Y debería ver "Resumen:"
Y debería ver "Resumen corto del TEG"
Y debería ver "Palabras claves:"
Y debería ver "Palabra clave para TEG, palabra clave TEG, ..."
Y debería ver "Fecha aproximada de presentacion"
Y debería ver "01/01/2012"
Y debería ver "Principales:"
Y debería ver "Jurado 1"
Y debería ver "Jurado 2"
Y debería ver "Suplentes:"
Y debería ver "Jurado 3"
Y debería ver "Jurado 4"
```

3.6.2.2.4 División de control de Estudios

Son los encargados de hacer cumplir la decisión del Consejo de Escuela frente a una solicitud de jurado, así como de validar y actualizar el expediente de los estudiantes involucrados. También emiten el Acta Final de Notas.

Para este usuario se identificó 1 escenario, que es descrito a continuación:

Ver solicitudes de jurado para Trabajo Especial de Grado y Seminario pendientes

Este Escenario será presentado con la Figura 2.20 y se describe de la siguiente manera:

```
Escenario: Ver estado de Solicitudes de Jurado para TEG
Dado que estoy en la página de inbox para Solicitudes Docentes
Entonces debería ver "Solicitudes de Jurado para TEG Pendientes"
Y debería ver “Solicitudes de Jurado Para Seminario Pendientes”
```

3.6.2.3 Desarrollo: A partir de estas pruebas se comenzó a definir las interfaces necesarias. Una vez definido el comportamiento, se procede a la implementación funcional de los controladores; para esto fue necesario un periodo de adaptación y entendimiento de las tecnologías, como lenguaje *HAML*

(*HTML Abstraction Markup Language*), *CoffeScript*; y ambiente de base de datos utilizados en el motor funcional del sistema**.**

A continuación se muestra una breve descripción de cada tecnología:

3.6.2.3.1 Prawn

Prawn es un esfuerzo conjunto entre el equipo principal (Gregory Brown, James Healy, Brad Ediger, Daniel Nelson, Jonathan Greenberg) y decenas de colaboradores de la comunidad *Ruby*. La meta de *Prawn* es llevar la modularidad y la reutilización de código del desarrollo web para la generacion de PDF.

Es una biblioteca para la generación de PDF en *Ruby* que proporciona una gran cantidad de funcionalidades tratando de mantenerse simple y al mejor rendimiento razonable. Estas son unas de las características más importantes de *Prawn*:

- Soporte de dibujo de vectores, incluyendo las líneas, polígonos, curvas, elipses, etc.
- Amplio soporte para renderizar texto.
- Soporte para tipos de fuentes , tanto las creadas dentro como los tipos de fuente 'TrueType'
- Una variedad de herramientas de bajo nivel para las necesidades básicas de diseño, incluyendo un sistema para incrustación de imágenes con opciones flexibles de escala.
- Las características de seguridad incluyen el cifrado y protección mediante contraseña.
- Herramientas para representar el contenido repetible (es decir, encabezados, pie de página y números de página).
- Características integrales de internacionalización, incluyendo soporte completo de UTF-8 basado en fuentes y renderización de texto de izquierda a derecha.
- Entre otras características.

A continuación código *Prawn* que generaría el encabezado de las actas que genera el sistema:

```
  image "#{Prawn::BASEDIR}/images/logo_ciencias.jpg",:at=> [485,725] ,
:width=>50,    :valign=>"top"
   image "#{Prawn::BASEDIR}/images/logo_ucv.jpg", :at=> [10,725] , :width=>60
   text "UNIVERSIDAD CENTRAL DE VENEZUELA", :align=>:center,:size=>18
   text "FACULTAD DE CIENCIAS", :align=>:center,:size=>18
   text "ESCUELA DE #{@carrera[:nombre_corto]}", :align=>:center,:size=>18
```

3.6.2.3.2 Haml

Haml (*Abstracción HTML Markup Language*) es un lenguaje de marcado que se utiliza para describir el *XHTML* de cualquier documento web sin el uso de la codificación en líneas tradicionales. Está diseñado para hacer frente a muchos de los defectos tradicionales de los motores de plantillas, así como hacer el código de descripción de páginas lo más elegante posible.

El objetivo de *Haml* es el incremento de la productividad que se lograría en 3 factores:

- Simplificación de código: *Haml* simplemente requiere escribir menos código.
- Representación visual de la jerarquía del documento: Esto permite que el código se vea más claro, así como eliminar la redundancia de cerrar todas las etiquetas. A cambio se tiene la desventaja de que *Haml* es muy estricto con la identación.
- Compatibilidad: *Haml* puede coexistir con las vistas *rhtml* e incluso mezclar código ERB y *Haml*.

Haml usa la identación del código no como la buena práctica que suele ser, sino como una obligación que tiene significado propio logrando así una manera más condensada y gráfica de crear plantillas *HTML*.

Usando *Haml* se logran plantillas con código *Ruby* embebido, concisas y legibles, al tiempo que mantiene la misma flexibilidad que las tradicionales plantillas ERB, y se escribe más rápido.

A continuación un ejemplo de código *Haml*

```
 =form_tag ({:action => "actualizar"}, {:id => "formulario"}) do
 %fieldset
   %legend
     Seleccione su propuesta de Jurados
   %table
     %tr
       %td
         %p
           =label :primer_principal,:primer_principal, "Nombre, apellido o cédula del 1er
Jurado Principal"
     %tr
       %td
         %p
           =
select_tag('principal1',options_from_collection_for_select(@usuariosDocentes, "cedula" ,
"nombre_corto" ))

   %p
     %input{:name => "nombre_seccion", :value=>@nombre_seccion,:type=>"hidden"}
     =submit_tag "Actualizar"
     %span{:id => "cargando" }
       =image_tag "cargando.gif", :width => 14
```

Se puede ver que la identación juega un papel muy importante en esta herramienta y gracias a ella definir qué código está dentro de cierta etiqueta.

3.6.2.3.3 CoffeeScript

CoffeeScript es un lenguaje para ser compilado en *JavaScript*, es decir, los archivos .coffee (archivos *CoffeeScript*) no son interpretados directamente por el navegador, sino que serán compilados en un archivo .js que será el utilizado posteriormente por éste. *CoffeeScript* puede funcionar para cualquier "tipo" de *JavaScript*: navegador, node.js, etc.

CoffeeScript es un lenguaje de programación que se traduce a código *JavaScript*. El lenguaje añade cierta sintaxis parecida a *Ruby*, *Python* y *Haskell*, para mejorar y clarificar la brevedad de las expresiones de *JavaScript*, así como para añadir características más sofisticadas como comprensión de arreglos y reconocimiento de patrones.

CoffeeScript traduce a *JavaScript* y los programas requieren entonces de menos líneas, algo así como un tercio del código original en *JavaScript*, sin que haya efectos sustanciales de desempeño al ejecutarlos. También se pueden utilizar todas las librerías *JavaScript* existentes.

CoffeeScript soluciona ese problema (la difícil lectura del *JavaScript*) con simplificaciones sintácticas que ayudan a escribir menos código de forma más rápida, fácil de leer y, de forma indirecta, un mejor rendimiento en el código. En el momento de traducir, realiza ciertas optimizaciones que mejoran el código frente a uno escrito de forma manual. Por lo tanto, se puede aprender mucho del código traducido intentando entender porqué *CoffeeScript* hace las cosas de una manera u otra.

La sintaxis dulce es un término que se utiliza en la ciencia de la computación para referirse a los lenguajes de programación que son diseñados para hacer los códigos más fáciles de expresar y de leer.
Se coloca un código de ejemplo en *CoffeScript* que nos permitiría utilizar las funciones "*tablesorter*" y "*datepicker*" en la solicitud de jurado:

```
$ ->
        $( ".tablesorter" ).tablesorter();

$ ->
       $( ".datepicker" ).datepicker();
```

Esta sería la salida en lenguaje JavaScript:

```
$(function() {
  return $(". tablesorter "). tablesorter ();
});

$(function() {
  return $(".datepicker").datepicker();
});
```

A continuación se muestran las vistas generadas para satisfacer cada funcionalidad.

3.6.2.3.4 Docente

Definidas las pruebas en la sección 2.6.2.2.1 se muestran a continuación el conjunto de interfaces que satisfacen las pruebas para los usuarios Docente.

Crear una nueva solicitud de jurado

Este formulario permite al docente tutor la interfaz para introducir una nueva solicitud de jurado para Seminario y TEG así como las vistas de confirmación de creación.

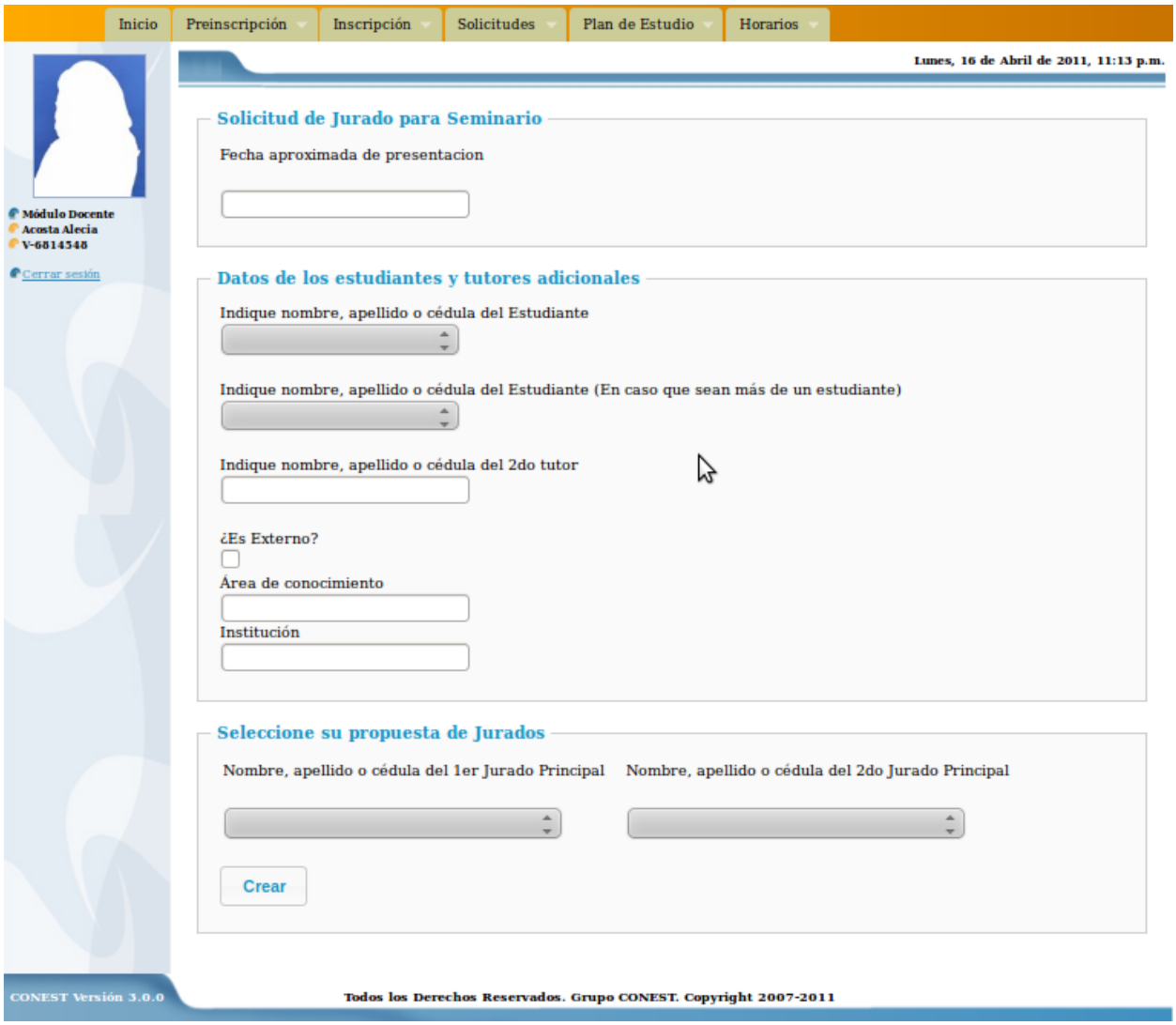

Figura 2.6 Formulario de creación de solicitud de jurado para Seminarios

Para ejemplificar el uso de *Haml*, se presenta una sección del código que genera esta vista:

```
=link_to 'Solicitud de Jurado', url_for(:action => 'index_docente', :controller =>
'solicitud_jurado')
> Nueva Solicitud de Jurado para Seminario
=form_tag ({:action => "crearSeminario"}, {:id => "formulario"}) do
  %fieldset
    %legend Solicitud de Jurado para Seminario
    %p
      Fecha aproximada de presentación
    %p
      %input.datepicker{:name => "fecha", :type => "text"}
  %fieldset
    %legend Datos de los estudiantes y tutores adicionales
    %p
```

```
       =label :nombre, :nombre, "Indique nombre, apellido o cédula del Estudiante"      =
select_tag('cedula_estudiante1',options_from_collection_for_select(@usuarios, "cedula" ,
"nombre_corto"))
    %p
      =label :TUTOR_externo,:TUTOR_externo, "¿Es Externo?"
      %input{:name => "TUTOR_externo", :type=>"checkbox", :id=>"x"}
      %div{:id=>"y"}
        =label :externo_nombre_label, "Nombre"
        %input{:name => "externo_nombre", :type=>"text"}
        %select{:id=>"externo_nacionalidad", :name =>"externo_nacionalidad"}
          %option{:value=>"V"}
            Venezolana
```

Primero se crea el *form_tag* que lleva los campos del formulario al método designado en el Controlador, y luego se pasa a escribir mediante los *label* y los *input* los distintos campos requeridos en el formulario. La Sintaxis para estos casos es muy similar a la utilizada en HTML.

Ahora se muestra parte del código Ruby del controlador invocado por esta vista cuando se llama a la funcionalidad de crear la solicitud de jurado para Seminario.

```
       def crearSeminario
         bitacora("creada solicitud de jurado para seminario")
         begin
           #Se obtiene el listado de materias desccritas como 'Seminarios'
           @estudiante_1=Usuario.order("primer_nombre").where(:cedula=>
params[:cedula_estudiante1]).first

          historial_academico_estudiante=HistorialAcademico.where(:materia_codigo=>
materiasTEG,:estudiante_cedula=>params[:cedula_estudiante1],
:periodo_academico_id=>periodo_actual).first

           bitacora(@string,params[:cedula_estudiante1])

              if(params[:cedula_estudiante2]!=null)

                 historial_academico_estudiante2=HistorialAcademico.where(
:materia_codigo=>materiasTEG, :estudiante_cedula=>params[:cedula_estudiante2],
:periodo_academico_id=> periodo_actual).first
                @estudiante_2=Usuario.order("primer_nombre").where(:cedula=>
params[:cedula_estudiante2]).first
                  if historial_academico_estudiante2[:carrera_id] ==
historial_academico_estudiante[:carrera_id]
                      #Se crea una nueva solicitud
                      #Se crea una nueva solicitud_historial_academico, asociada a la solicutd
anterior
                      #Se crean los jurados, principales y suplentes y se asocia el tutor a la
materia y sección en cuestión
              .
              .
              .
```

Primero se crea una entrada en la bitácora que se encarga de llevar un control de las actividades de los usuarios frente al sistema, esta información es necesaria y sensible ya que a partir de ella se presta servicio.

Luego, se obtiene el listado de materias descritas como 'Seminarios', que estén disponibles, para luego obtener el listado de estudiantes que ha inscrito alguna de estas materias; esta será la lista de estudiantes disponible en el formulario de creación.

Si un estudiante ya tiene una Solicitud creada en el sistema, se muestra un mensaje de error ya que un estudiante no puede tener más de una solicitud de jurado en procesamiento.

Luego de las verificaciones necesarias, se crean las solicitudes y se asocia a los expedientes de los estudiantes correspondientes utilizando la tabla solcitud_historial_academico.

Figura 2.7 Pantalla de confirmación de creación de solicitud para Seminario (Docente)

Figura 2.8 Pantalla de confirmación de creación de solicitud para TEG.

Figura 2.9 Formulario de creación de Solicitud para TEG (Docente)

Ver Solicitudes de Jurado para Trabajo Especial de Grado pendiente

Es la vista principal donde un docente puede observar el estado de las solicitudes creadas.

Figura 2.10 Listado de solicitudes pendiente

Para esta funcionalidad, se muestra la vista implementada por el siguiente código Haml:

```
=@notice
%h1
  Solicitudes de Jurado para Trabajo Especial de Grado
  %div
    %table.tablesorter
      %thead
        %tr
          %th
            Estudiantes
          %th
            Título del Trabajo Especial de Grado
          %th
            Estado
          %th
            Acciones
        %tbody
          -For solicitud in @solicitudes_historial_academicoTEG
            %tr
              %td
                -For solicitud_historial in
@historiales_academicos=SolicitudHistorialAcademico.where(:nombre_seccion=>solicitud["nombre_secc
ion"])
                %p
                    =Usuario.where(:cedula=>solicitud_historial["estudiante_cedula"]).first.
nombre_corto
              %td
                  =TrabajoEspecial.where(:carrera_id=>session[:docente].organizacion_id,:
periodo_academico_id=>solicitud.periodo_academico_id,:nombre_seccion=> solicitud.nombre_seccion,
:materia_codigo=>solicitud.materia_codigo).first.titulo
              %td
                  =TipoStatusSolicitud.where(:id=>Solicitud.where(:id=>solicitud.
solicitud_id).first.tipo_status_solicitud_id).first.descripcion
```

Se listan todas las solicitudes asociadas al usuario (docente) que las creo, mostrando su estado actual y una lista de opciones que se pueden realizar con la solicitud (dependiendo de su estatus). Se puede ver los estudiantes involucrados así como el título de la Solicitud, dependiendo si es TEG o Seminario.

La identación juega un papel fundamental en la programación en Haml ya que al haber un error en ella, ocurren errores en la interpretación del código. Por cada solicitud que exista en la base de datos asociada a ese docente, se imprimirá en la vista la misma.

Como corresponde, el código Ruby que implementa la funcionalidad es el siguiente:

```
  def index_docente
  usuario = Usuario.verificar_login(user,clave)
  session[:usuario]=usuario
  session[:tipo_modulo] = TipoModulo.find TipoModulo::DOCENTE
  session[:docente] = usuario.docente
  bitacora("visitando el listado de solicitudes de jurado pendiente")
     @notice=params["notice"]
     @jurados=JuradoExaminador.where(:periodo_academico_id=>"01-
2010",:tipo_jurado_id=>['TU'],:usuario_cedula=>session[:usuario]["cedula"]).group("nombre_seccion
")

    @solicitudesTEG=Array.new(0,String)
    @solicitudesSEM=Array.new(0,String)

    @jurados.each do |jurado|

        @solicitud_historial_academico=SolicitudHistorialAcademico.where
(:periodo_academico_id=>"period_actual",:nombre_seccion=>jurado["nombre_seccion"],:materia_codigo
=>jurado["materia_codigo"])

       @solicitud_historial_academico.each do |solicitud_historial|
        @solicitud=Solicitud.where(:id=>solicitud_historial["solicitud_id"]).first
      @tipoSolicitud=TipoSolicitud.where(:id=>@solicitud["tipo_solicitud_id"]).first
      if (@tipoSolicitud["id"] == 'JURTEG')
       @solicitudesTEG << solicitud_historial["solicitud_id"]
      else
       if (@tipoSolicitud["id"] == 'JURSEM')
         @solicitudesSEM << solicitud_historial["solicitud_id"]
      end
     end
    end
  end

 @solicitudes_historial_academicoTEG=SolicitudHistorialAcademico.order
("nombre_seccion").where(:solicitud_id=>@solicitudesTEG).all

 @solicitudes_historial_academicoSEM=SolicitudHistorialAcademico.order
("nombre_seccion").where(:solicitud_id=>@solicitudesSEM).all
  End
```

Para el usuario docente que se encuentre en sesión, primero se crea una entrada en la bitácora con la actividad que se está realizando (en este caso, “visitando el listado de solicitudes de jurado pendiente”)

Luego, se buscan todas las solicitudes que existan donde el docente es jurado principal y luego se agrupan por nombre de sección (para que dos estudiantes que estén haciendo juntos el TEG y el Seminario salgan juntos) y ordenan los modelos que se pasan a la vista.

Modificar solicitud de jurado para Seminario

Permite la modificación de los datos de una solicitud por parte del tutor, siempre y cuando este en procesamiento.

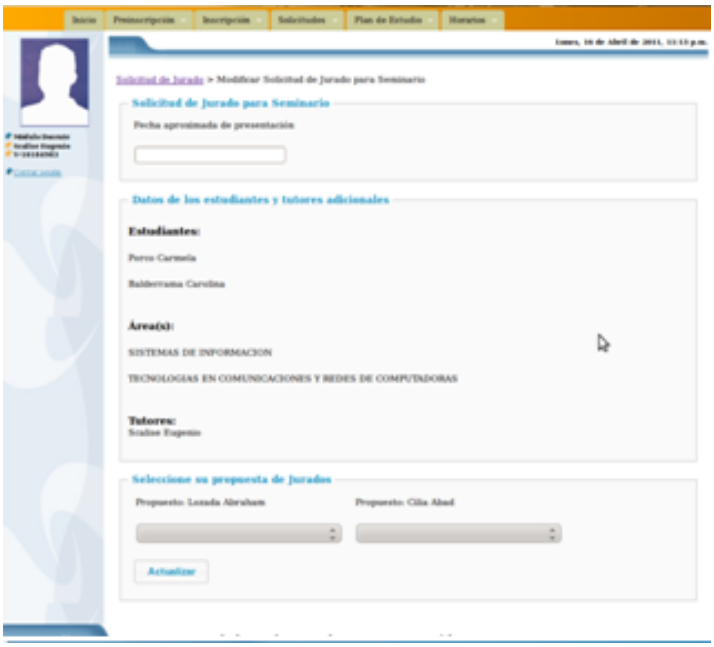

Figura 2.11 Formulario de modificación de Solicitud de Jurado para Seminario

Figura 2.12 Pantalla de confirmación de actualización de Solicitud para Seminario

Modificar solicitud de jurado para Trabajo Especial de Grado

Provee un método para la modificación de los datos de una solicitud por parte del docente encargado, solo si esta no ha sido enviada a procesamiento.

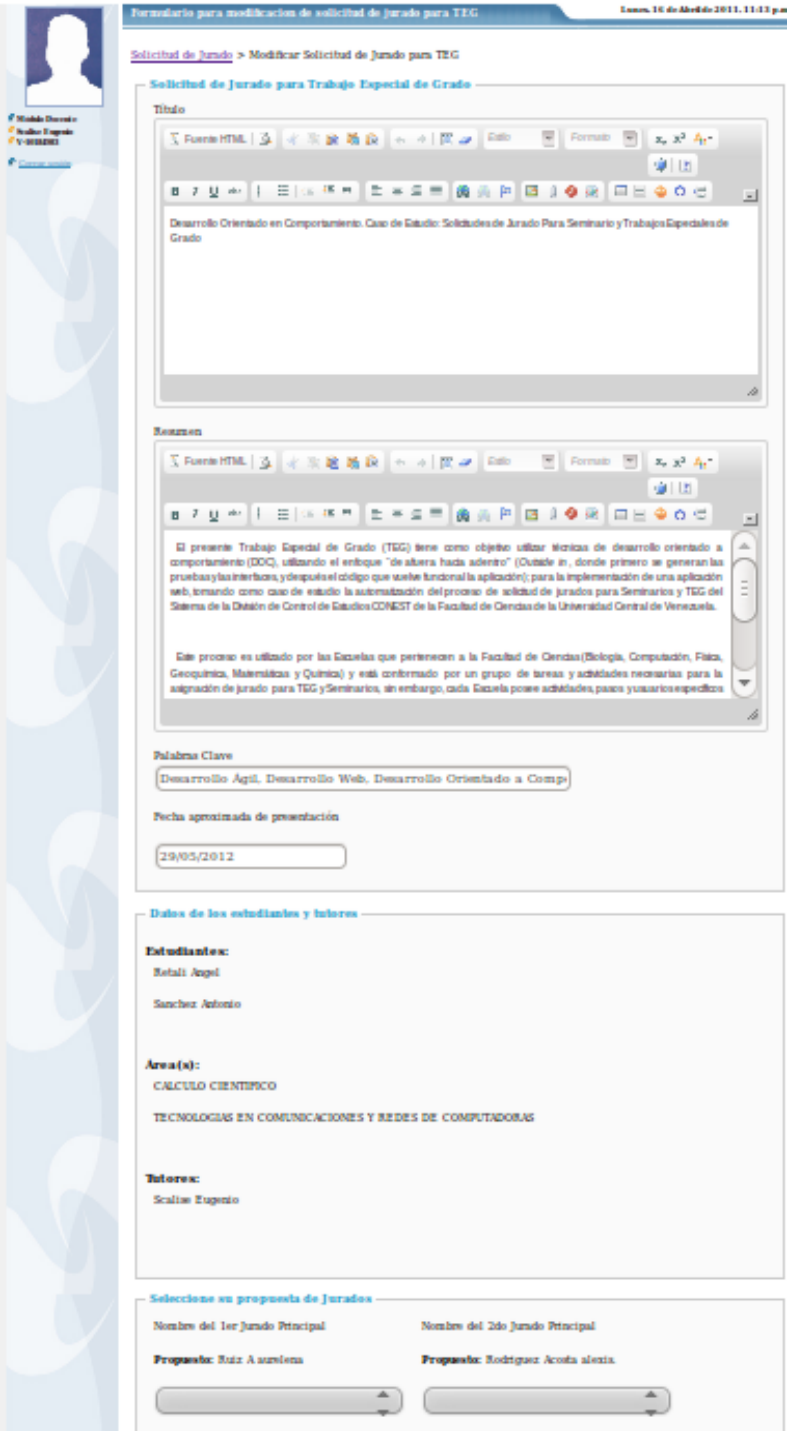

Figura 2.13 Formulario de modificación de solicitud de jurado para TEG. (Docente)

Figura 2.14 Pantalla de confirmación de actualización de Solicitud para TEG.

3.6.2.3.5 Consejo de Escuela

Conformada por docente de la Escuela, es la encargada de la aprobación o rechazo de las solicitudes estudiantiles.

Ver solicitudes de jurado para Trabajo Especial de Grado pendiente

En esta vista se pueden observar todas las solicitudes de jurado creadas en una escuela, y su estado actual.

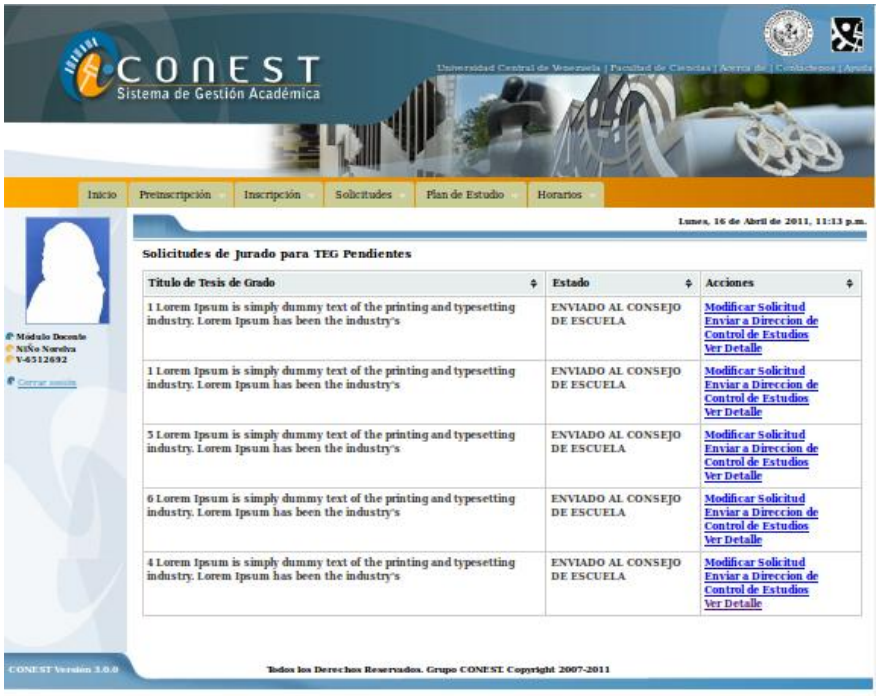

Figura 2.15 Lista de solicitudes de jurados pendientes

Modificar solicitud de jurado para Trabajo Especial de Grado

Permite al consejo de Escuela modificar los datos que considere necesarios en una solicitud de jurado.

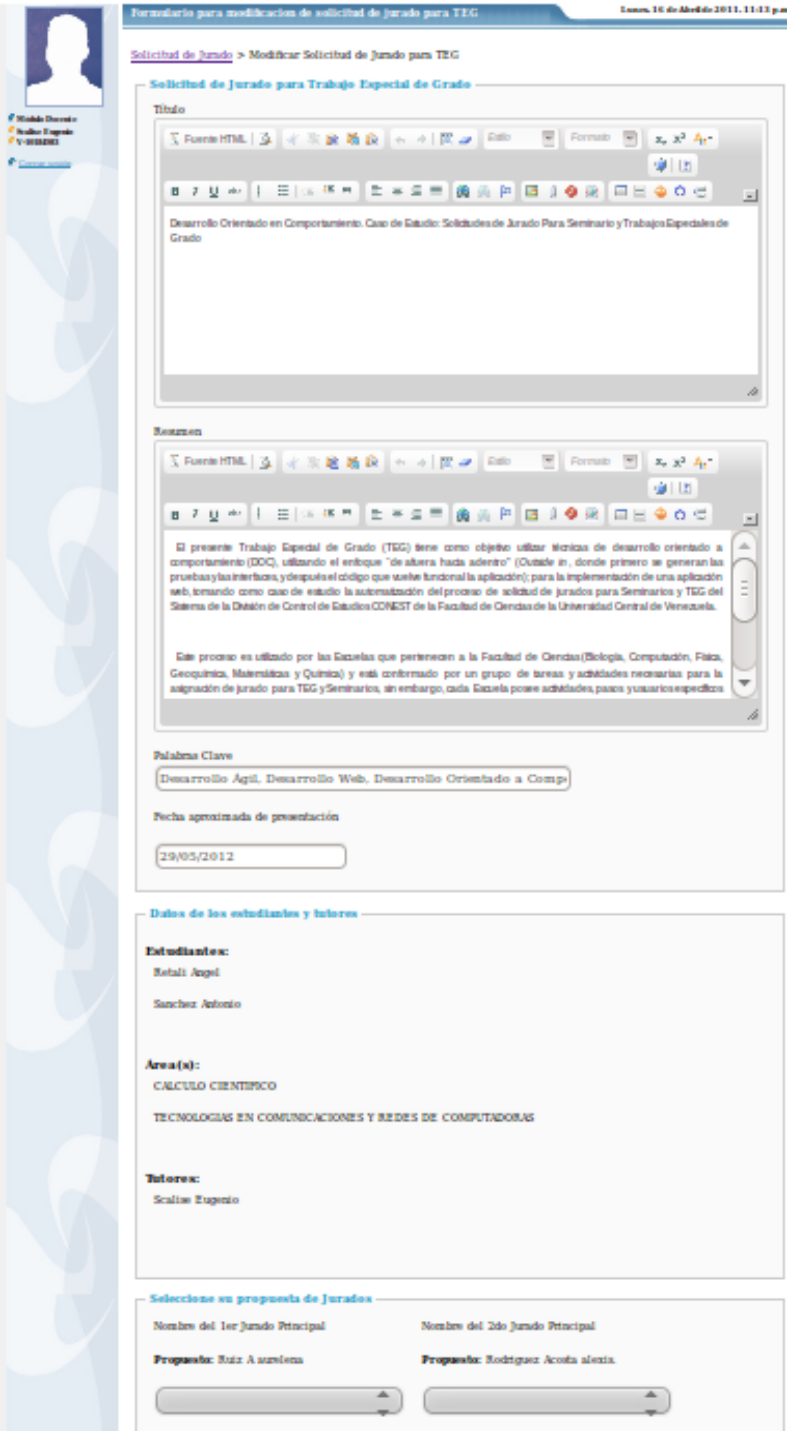

Figura 2.16 Formulario de modificación de solicitud de jurado para TEG. (Consejo de escuela)

Figura 2.17 Pantalla de confirmación de actualización de solicitud para TEG. (Consejo de escuela)

3.6.2.3.6 División de control de Estudios

Es el ente encargado de la recibir y archivar el resultado de la aprobación de las Solicitudes Estudiantiles por parte de los Consejos de Escuelas.

Ver solicitudes de jurado para Trabajo Especial de Grado pendiente

La DCE cuenta con la siguiente interfaz de control para las solicitudes que les han sido enviadas desde los distintos Consejos de Escuela de la Facultad de Ciencias.

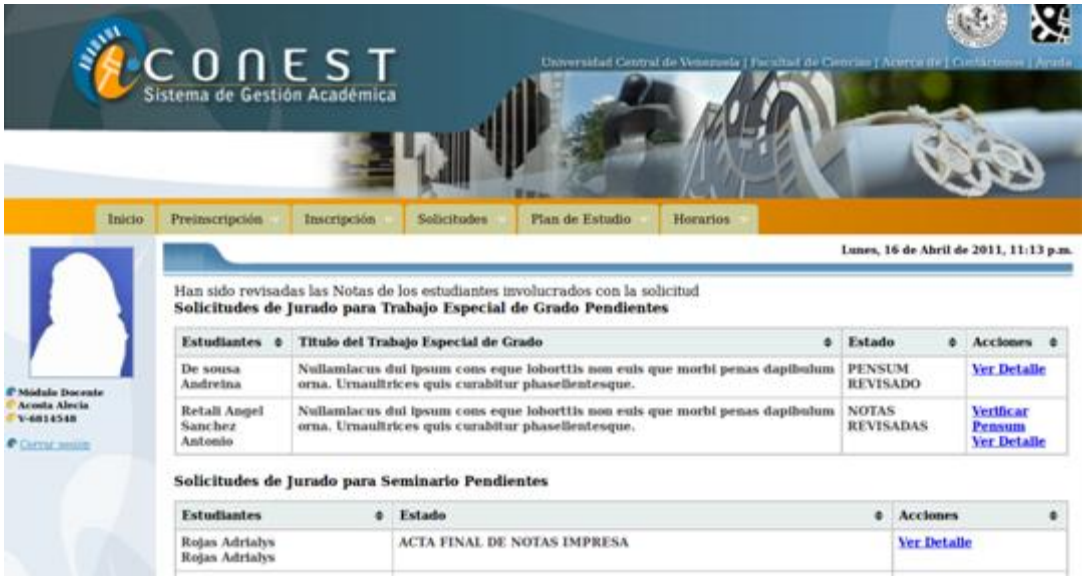

Figura 2.18 Lista de solicitudes de jurados pendientes (DCE).

3.6.2.4 **Demostración:** Una vez terminada la implementación de la aplicación se aplicó una encuesta a diferentes tipos de usuarios de las diferentes Escuelas de la Facultad de Ciencias, con el fin de validar la aplicación frente a los usuarios finales que usarán el sistema. Cabe destacar que debido a la complicada agenda que presenta el personal de la Escuela de Biología no fue posible recolectar información de dicha escuela.

A continuación se presentan los resultados de la encuesta, divida por preguntas, el modelo de la encuesta será colocada en los anexos (ver anexo 2).

3.6.2.4.1 Muestra de manera rápida y sencilla la información necesaria para realizar las tareas relacionadas a las solicitudes de jurado para TEG y seminario.

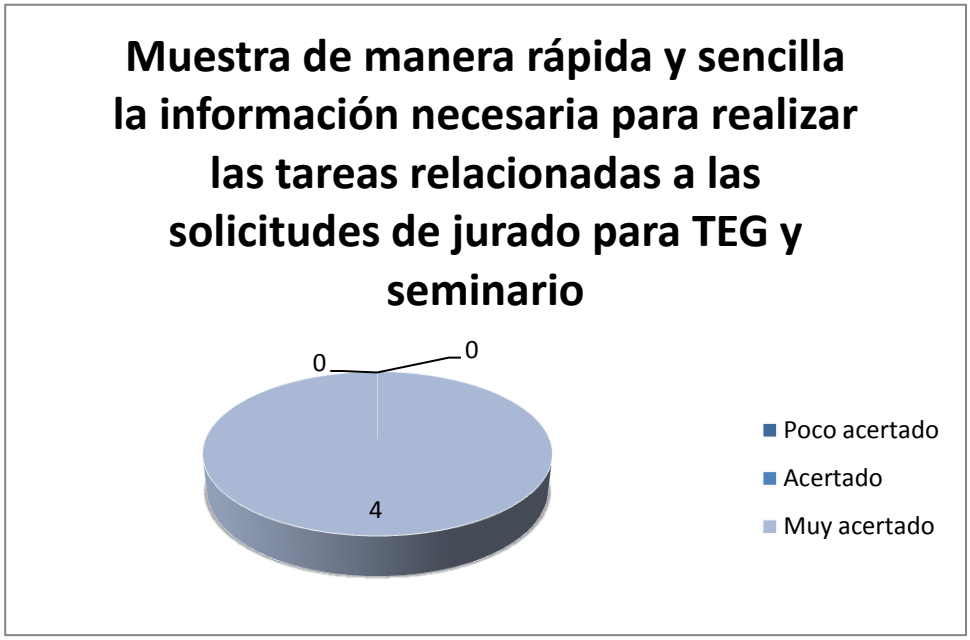

Figura 2.19 Gráfico Demostración Pregunta 1.

La Figura 2.19 muestra que los usuarios encuentran la aplicación fácil de usar, lo que sugiere una rápida y sencilla transición entre el nuevo modulo automatizado y la manera en que se trabaja actualmente en las escuelas.

3.6.2.4.2 Minimiza las posibilidades de cometer errores y provee *feedback* en caso de cometerlo.

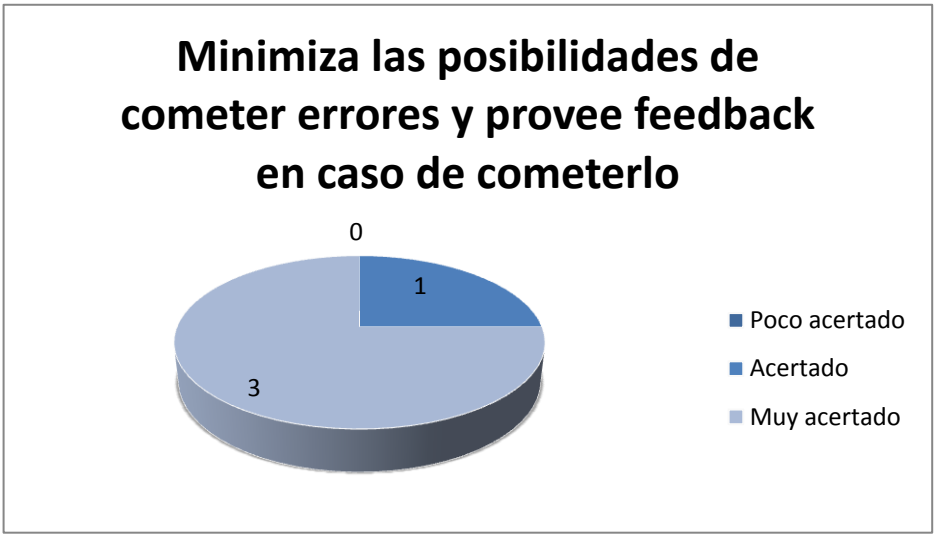

Figura 2.20 Gráfico Demostración Pregunta 2

En este punto se busca validar que el nuevo módulo presenta una ventaja considerable tomando en cuenta el error humano presente en la manera en que trabajan las Escuelas actualmente. Los resultados se muestran en la Figura 2.20.

3.6.2.4.3 Provee una correcta automatización de los procesos actuales llevados a cabo para las tareas relacionadas a las solicitudes de jurado para TEG y Seminario.

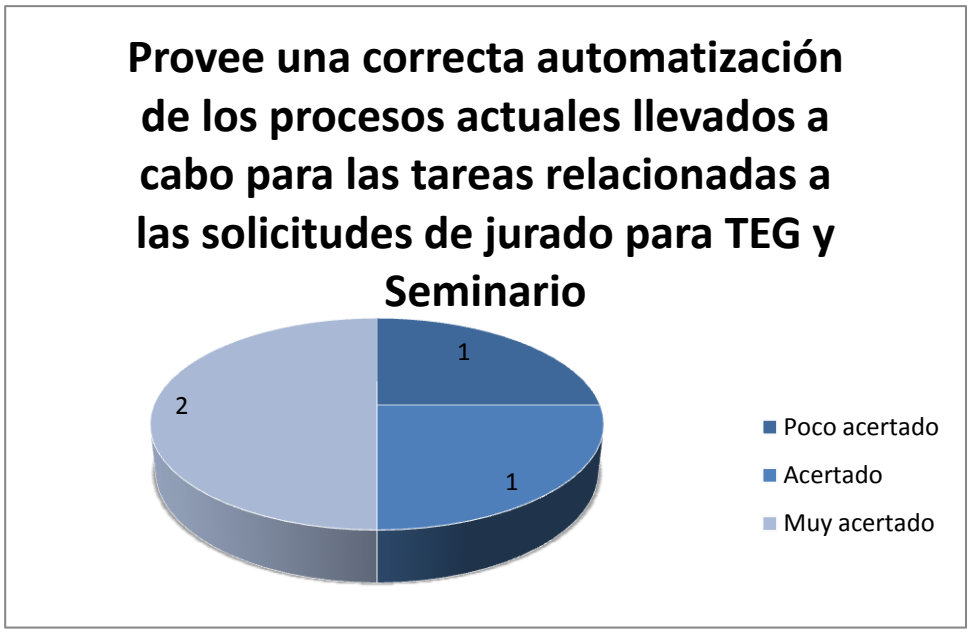

Figura 2.21 Gráfico Demostración Pregunta 2

Se muestra en la Figura 2.21 que mediante las encuestas realizadas a miembros de la Facultad, el sistema desarrollado para la solicitud de jurado para T.E.G y Seminarios cubre las necesidades necesarias para ser usado en una nueva versión del sistema CONEST.

3.6.2.4.4 La información, diseño, apariencia y organización de los elementos mostrada es consistente y posee un estándar en la aplicación.

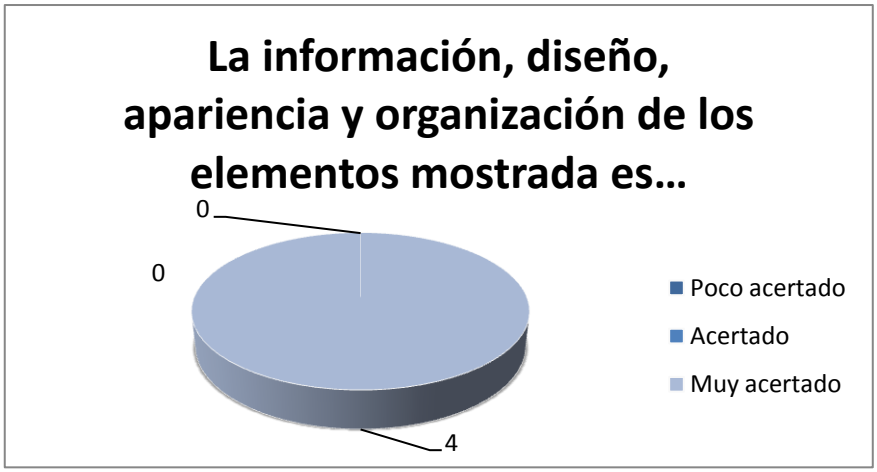

Figura 2.22 Gráfico Demostración Pregunta 4

Con este punto se recolecta información sobre que tan estandariza y útil es la información mostrada por el módulo de la aplicación. En la Figura 2.22 muestra que la mayoría de los usuarios piensan que el modulo cumple con esta cualidad.

Los formularios presentados recopilan la información necesaria para cumplir con las tareas y son fáciles de entender.

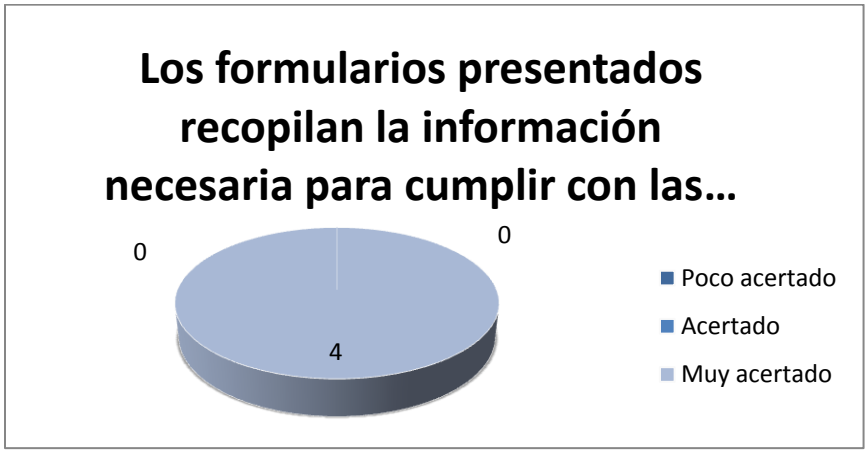

Figura 2.23 Gráfico Demostración Pregunta 5.

Los usuarios encuestados consideran que este módulo cumple con esta cualidad, como se muestra en la Figura 2.23.

Aunque la cantidad de personas encuestadas no es suficiente para obtener un resultado concluyente, podríamos describir una tendencia. En cuanto a la información que despliega la aplicación, el diseño que mantiene, y el minimizar las posibilidades de cometer errores entregando *feedback* en caso de ser necesario, las personas entrevistadas piensan en general que se logra satisfactoriamente. Sin embargo, en cuanto a la pregunta sobre si posee una correcta automatización de los procesos actuales que podemos ver en la Figura 2.21, no podemos obtener un resultado concluyente aunque la tendencia es positiva.

3.6.3 Despliegue de la aplicación

La aplicación fue instalada en el servidor `Yagrumo` de la escuela de computación, en ambiente Linux, utilizando lenguaje *Ruby* en su versión 1.8.7, con el *framework Ruby On Rails* versión 3.1.0. La base de datos selecciona fue *Mysql*, motor de base de datos bajo el cual se encuentra la estructura de tablas de base de datos proporcionada antes de iniciar a desarrollar la aplicación.

Los archivos de la aplicación y el *backup* de la base de datos fueron movidos al servidor mediante comandos *SSH* (*Secure SHell*) a través de la consola del servidor, donde también fue necesaria la instalación del *framework* y de gemas como `*Nokogiri*`, `*Prawn*` y otras que conforman el ambiente de ejecución de la aplicación.

La aplicación se ejecuta bajo el servidor *WEBrick* embebido en *Ruby* en el puerto 3000 del servidor.

Capítulo III: Conclusiones, recomendaciones y limitaciones

El objetivo de este TEG consiste en el desarrollo de un módulo de CONEST que automatice los procesos relacionados a las solicitudes de jurado para Seminario y TEG de la Facultad de Ciencias de la Universidad Central de Venezuela utilizando las técnicas de DOC. Se logra a través de la obtención de un producto que cumple con todos los requerimientos planteados por los objetivos generales y específicos.

Aún cuando el proceso de levantamiento de información se dificulta desde un comienzo, se logra captar todos los requerimientos. Los procesos y actividades tomados en cuenta en este TEG involucran a un gran número de personas en cada una de Escuelas de la Facultad y contactarlos a todos es una tarea ardua, sin embargo, algunas de ellas facilitan el proceso, puesto que brindan la información necesaria para realizar el análisis de requerimientos y tienen la disponibilidad para formar parte del proceso de desarrollo.

La discusión de los procesos críticos y escenarios necesarios para procesar las solicitudes de Jurado para TEG y Seminario, se realiza sin ningún problema. Sin embargo, al momento de llegar al desarrollo de las pruebas se observa que estas pueden incrementar, es decir, generándose nuevas pruebas a medida que se desarrolla el sistema.

La aplicación de las pruebas de aceptación del modulo también se desarrolla de forma exitosa, teniendo en cuenta que muchos usuarios facilitan esta tarea gracias a su disponibilidad.

De las distintas tecnologías utilizadas como Haml, Prawn, CoffeeScript , se logra concluir que son muy útiles al momento de saber utilizarlas aunque la curva de aprendizaje de algunas es un poco empinada. Por lo tanto, toma un tiempo en lograr entender cómo funcionan y cómo se utilizan (por ej. CoffeeScript) aunque luego facilitan bastante el trabajo.

El desarrollo orientado a comportamiento permite el desarrollo en conjunto de la documentación de pruebas de la aplicación y el código que logra el correcto funcionamiento de la aplicación. Al leer las pruebas se observa que se describen las funcionalidades del sistema en un lenguaje entendible y por lo tanto, cualquier persona puede comprender el fin de cada funcionalidad.

Como proceso ágil, el DOC ofrece un desarrollo lento al principio del proyecto debido a que se tiene que lograr un completo entendimiento de las funcionalidades y requerimientos del sistema pero posee beneficios como poder retroceder en el desarrollo para cambiar las funcionalidades, también en caso de

que una persona se va a integrar al grupo de trabajo, se le hace mucho más fácil la etapa de aprendizaje del proyecto como tal, además y se tiene un producto probado y con estas pruebas escritas.

En un futuro, se pueden implementar mejoras que ayuden a favorecer al módulo desarrollado. Algunas de nuestras recomendaciones son: implementar una funcionalidad para que se pueda cancelar una Solicitud que ya fue creada por un Tutor, que el tutor tenga posibilidad de ver todas las solicitudes que ha creado en vez de solo poder ver las del período académico actual, que el estudiante tenga la posibilidad de ver en qué estado se encuentra la solicitud, entre otras.

La Facultad de Ciencias cuenta con una matrícula de más de 3000 alumnos y teniendo en cuenta que en las distintas escuelas la cantidad de profesores que son miembros de las comisiones de tesis, mesas de tesis, etc. son pocos, esto podría causar un efecto parecido al embudo de botella donde la cantidad de solicitudes sean mayores a la cantidad de solicitudes procesadas en un tiempo limitado.

En el desarrollo del sistema se encontraron herramientas nuevas ya implementadas para el desarrollo base de la aplicación de CONEST, como por ejemplo el estándar para la vista *Haml*, el *framework* de *Javascript, Coffee-Script* y la generación de documentos PDF con la gema *Prawn*; se necesitó de un tiempo de adaptación y aprendizaje de estas tecnologías, una de las principales limitaciones fue la escasa documentación, refiriéndonos principalmente a *Haml* y *Coffee-Script*.

Referencias

Alexandrou, M. (2011). *Crystal Methods Methodology*. Retrieved 2010 йил Diciembre from http://www.mariosalexandrou.com/methodologies/crystal-methods.asp

Ambler, S. W. (2010). *Introduction to Test Driven Design (TDD).* Retrieved 2010 йил noviembre from http://www.agiledata.org/essays/tdd.html

Barbero, M. J. (2008 йил 17-4). *Cuaderno de Notas de Miguel Jaque Barbero.* Retrieved 20 йил 2011-12 from http://migueljaque.com/index.php/metodologias/25-introducci%C3%B3n/57-ventajasmetodologiasagiles

Colusso, R. (2009 йил 17-Octubre). *Desarrollo ágil de software*. Retrieved 2010 йил Noviembre from http://knol.google.com/k/desarrollo-%C3%A1gil-de-software#

Colusso, R. (2009). *Desarrollo ágil de software.* Retrieved 2011 йил 24-2 from http://knol.google.com/k/desarrollo-%C3%A1gil-de-software#

Dalke, A. (2009 йил 29-Diciembre). *Problems with TDD.* Retrieved 2010 йил 10-Noviembre from http://www.dalkescientific.com/writings/diary/archive/2009/12/29/problems_with_tdd.html

Daycrom Software Factory. (n.d.). *Daycrom.* Retrieved 2010 йил 27-12 from http://www.daycrom.com/metodologias-agiles

Díaz, J. R. (n.d.). *Desarrollo Software Agil*. Retrieved 2010 йил Noviembre from http://www.slideshare.net/jrramon/desarrollo-agil

Factory, L. S. (n.d.). *Desarrollo Orientado por Pruebas.* Retrieved 2010 йил noviembre from http://www.logiciel.es/tdd

Gonzales, E. (2010 йил 15-Noviembre). *desarrollo web agil.* Retrieved 2010 йил Noviembre from https://docs.google.com/leaf?id=0B0nKdTGFYvevYzlmMzViNTctNTZhOC00OGFiLThhZWUtZmQ0OGEyYmZmODQ1&sort=name&layout=list&pid=0B0nKdTGFYvevOTc1YzBlOTgtNGRlMS00NGRlLWFmMjgtOTQ0YmNhOGEzNDQ4&cindex=1

José H. Canós, P. L. *Métodologías Ágiles en el Desarrollo de Software.* Universidad Politécnica de Valencia.

José H. Canós, P. L. (2003). *Métodologías Ágiles en el Desarrollo de Software.* Universidad Politécnica de Valencia.

Manchón, E. (2006 йил 26-Octubre). *Desarrollo ágil.* Retrieved Octubre йил 2010 from http://www.desarrolloweb.com/articulos/desarrollo-agil.html

Mayer, T. (2009 йил 20-Septiembre). *Agile Anarchy*. Retrieved 2010 йил Diciembre from http://agileanarchy.wordpress.com/2009/09/20/simple-scrum/

Metodologías de Desarrollo Ágil. (2007 йил 25-Agosto). Retrieved 2010 йил Noviembre from http://www.caudalweb.com/blog/metodologias-de-desarrollo-agil/

ProyectosAgiles.org. (n.d.). Retrieved 2010 йил Diciembre from http://www.proyectosagiles.org/

Schwaber, K., & Beedle, M. (2002). *Agile Software Development with Scrum (Series in Agile Software Development).*

ScrumAlliance. (n.d.). Retrieved 2010 йил Diciembre from http://www.scrumalliance.org/articles

The Agile Alliance. (2001). *Manifesto for Agile Software Development*. Retrieved 2010 йил Octubre from http://agilemanifesto.org/

Viejo, B. (2010). *Desarrollo Orientado a Pruebas (TDD).* Retrieved 2010 йил noviembre from http://bosqueviejo.net/2009/01/08/desarrollo-orientado-a-pruebas-tdd/

Wigahluk en julio 3, 2. (2007, julio 3). *La desorientación con las pruebas….* Retrieved noviembre 2010, from http://wigahluk.wordpress.com/2007/07/03/la-desorientacion-con-las-pruebas/

Wikimedia. (n.d.). *Desarrollo ágil de software*. Retrieved Octubre 2010, from http://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software

Willian, s., & Veronica, a. (2008, 9 22). *Métodos Ágiles.* Retrieved 2 24, 2011, from http://metodosagiles.blogspot.com/

ANEXO 1: Levantamiento de información sobre los distintos procesos que se llevan a cabo en las Escuelas para solicitudes de jurado para TEG.

Escuela de Computación

Prof. Eugenio Scalise y Profa. Jossie Zambrano, profesores de la Escuela de Computación.

Lugar: Escuela de Computación

Fecha: 14 de Junio de 2011

Resumen de la reunión:

Se nos informó del proceso que se lleva a cabo en el sistema CONEST para la selección de jurados, el proceso empieza al momento en que el tutor solicita el jurado y se puede resumir en los siguientes pasos:

1. El tutor envía la planilla de Solicitud de jurado a la Comisión de Tesis y espera resultado de aprobación por parte del Consejo de Escuela. Consejo de Escuela aprueba y asigna la fecha de presentación de la tesis.
2. La Comisión de Tesis recibe la planilla de Solicitud de jurado. Se envía un *feedback* con la aprobación definitiva al Estudiante y al Tutor.
3. La Comisión consulta al jurado propuesto por el Tutor para garantizar una taza justa de trabajo a cada jurado entre las diferentes Tesis de la Escuela.
4. La Comisión de Tesis puede o no modificar la Solicitud inicial de jurado tomando en cuenta la consulta anterior.
5. La planilla es enviada al Consejo de Escuela para su discusión.
6. El Consejo de Escuela recibe la planilla de Solicitud de jurado.

7. El Consejo de Escuela evalúa la Solicitud y toma una decisión con respecto a su aprobación. Si la Solicitud no es aprobada, se envía una notificación al Tutor indicando el motivo. Si la Solicitud es aprobada, se asigna una fecha de presentación.

8. Consejo de Escuela asigna fecha de presentación para la Tesis.

9. Consejo de Escuela entrega el documento donde se encuentra el jurado aprobado y la fecha de presentación de la Tesis.

10. La DCE recibe el documento enviado por el Consejo de Escuela donde se encuentra la información del jurado aprobado y la fecha de presentación de la tesis. La DCE también verifica los requisitos para la presentación de tesis del bachiller. Solo si este cumple los requisitos, se continúa el proceso.

11. La DCE debe ingresar al sistema los datos del jurado aprobado y fecha de presentación para la Tesis inscrita.

12. Proceso en paralelo:

    a. Enviar notificación a jurado/estudiante: La DCE envía una notificación al jurado aprobado y al Estudiante indicando la aprobación y fecha de presentación de la Tesis.

    b. Recibir notificación de DCE: El jurado aprobado y el estudiante reciben la notificación de la aprobación y fecha de presentación de la Tesis.

    c. Enviar notificación a Tutor: La DCE envía una notificación al Tutor indicando la aprobación y fecha de presentación de la Tesis.

    d. Recibir notificación de DCE: El Tutor recibe la notificación de la aprobación y fecha de presentación de la Tesis.

13. Fin del proceso en paralelo.

14. El Tutor solicita el acta final de notas del Estudiante a la DCE.

15. La DCE recibe la Solicitud del Acta Final de Notas del Estudiante.

16. La DCE imprime el Acta Final de Notas solicitada.

17. La DCE entrega al Tutor el Acta Final de Notas solicitada.

18. El Tutor recibe el Acta Final de Notas solicitada.

19. El Tutor entrega a la DCE el Acta Final de Notas con la nota de presentación del jurado.

Adicionalmente se nos dieron unos requerimientos a tomar en cuenta a la hora de desarrollar:

1. Al acceder como Control de Estudios al sistema, poder tener una vista donde se vea quiénes tienen jurado y quiénes son.

2. Respetar los estándares en *Ruby*, utilizar *CSS3*, *MySQL*, *JQuery*.

Escuela de Química

Prof. José Martínez, integrante de la Grado de la Escuela de Química

Lugar: Edif. De Química, Piso 2, Centro de Equilibrio en Solución

Fecha: 8 de Julio de 2011

Resumen de la reunión:

Se nos informó del proceso que se lleva a cabo en la Escuela de Química para la selección de jurados, el proceso es empieza al momento en que el alumno inscribe seminario, y se puede resumir en los siguientes pasos:

1. El alumno inscribe la materia Seminario, habiendo consultado con su tutor tema, título de la tesis, jurado tentativo, plan de trabajo.
2. El alumno redacta el manuscrito del Seminario de Investigación.
3. El tutor llena la Planilla de Inscripción de Seminario y Tesis de Grado con los datos consultados por su tutorado y envía la planilla de aceptación del TEG a la Comisión de tesis de química que cumple entre otras, las funciones de comisión de Grado discutidas en escuelas anteriores.
4. La Comisión revisa la planilla, evaluando el título, tutor, jurado tentativo de la presentación y créditos del estudiante y la envía a Consejo de Escuela.
5. El Consejo de Escuela de Química revisa la planilla y el manuscrito, evaluando el título, tutor, jurado tentativo de la presentación y créditos del estudiante.
6. Consejo de Escuela evalúa la Planilla y la envía a Control de Estudios, indicando su decisión.
7. Control de Estudios recibe la planilla, donde se indica si la inscripción del estudiante es válida o no y notifica al estudiante.
8. El estudiante debe buscar en la oficina del Departamento de Química, las actas de aprobación del Proyecto y asignación de jurados; cinco (5) actas en original, una para cada miembro del jurado (principales y suplentes), y una que deben firmar todos los miembros jurados, esta última debe ser entregada en la oficina del Departamento de Química.
9. El estudiante contacta al jurado asignado para establecer fecha de presentación y su tutor debe solicitar a Control de Estudios un Acta de Notas donde se coloca la nota y debe ser enviada a control de estudios, culminando el proceso.

Escuela de Biología

Prof. Palmira Guevara, integrante de la Comisión de Asuntos Académicos, Departamento de Biología Celular.

Lugar: Laboratorios Docentes de Biología/Aula 2

Fecha: 20 de Julio de 2011

Resumen de la reunión:

Se nos informó del proceso que se lleva a cabo en la Escuela de Biología para la selección de jurados, el proceso es empieza al momento en que el alumno inscribe seminario, y se puede resumir en los siguientes pasos:

1. El alumno inscribe la materia Seminario, habiendo consultado con su tutor tema, título de la tesis, jurado tentativo, plan de trabajo.
2. El tutor llena la Planilla de Inscripción de Seminario y Tesis de Grado con los datos consultados por su tutorado y envía la planilla a la Comisión de Asuntos Académicos de Biología Celular que cumple entre otras, las funciones de Comisión de Grado discutidas en escuelas anteriores.
3. La Comisión de Asuntos Académicos de Biología Celular revisa la planilla, evaluando el título, tutor, jurado tentativo de la presentación y créditos del estudiante y la envía a consejo de escuela.
4. Consejo de Escuela evalúa la Planilla y la envía a Control de Estudios, indicando su decisión.
5. Control de estudios recibe la planilla, donde se indica si la inscripción del estudiante es válida o no y notifica al estudiante.
6. El estudiante contacta al jurado asignado para establecer fecha de presentación y su tutor debe solicitar a Control de Estudios un Acta de Notas donde se coloca la nota y debe ser enviada a control de estudios, culminando el proceso.

Escuela de Física

Prof. Ernesto Fuenmayor, Director de la Escuela de Física.

Lugar: Escuela de Física

Fecha: 25 de Julio de 2011

Resumen de la reunión:

Se nos informó del proceso que se lleva a cabo en la Escuela de Física para la selección de jurados, el proceso empieza al momento en que el alumno inscribe seminario, y se puede resumir en los siguientes pasos:

1. El alumno inscribe la materia Seminario, habiendo consultado con su tutor tema, título de la tesis, jurado tentativo, plan de trabajo.
2. El tutor llena la Planilla de Inscripción de Seminario y Tesis de Grado con los datos consultados por su tutorado y envía la planilla al Consejo de Escuela.
3. El Consejo de Escuela recibe la planilla, a través de la Comisión de Grado de la Escuela que realiza las validaciones de créditos del estudiante para presentar seminario y tesis, y puede o no modificar la selección de jurados en la planilla y el plan de trabajo, que pudiera afectar la fecha tentativa de presentación.
4. Consejo de Escuela evalúa la Planilla y la envía a Control de Estudios, indicando su decisión.
5. Control de Estudios recibe la planilla, donde se indica si la inscripción del estudiante es válida o no y notifica al estudiante.
6. El estudiante contacta al jurado asignado para establecer fecha de presentación y su tutor debe solicitar a Control de Estudios un Acta de Notas donde se coloca la nota y debe ser enviada a Control de Estudios.
7. El estudiante presenta Seminario y si es aprobado, el tutor envía la Planilla de Inscripción de Seminario y Tesis de Grado llenada al inicio de semestre a la comisión de Grado de la escuela que puede evaluar el cumplimiento de plan de trabajo, y modificarlo, acción que en algunos casos afecta también la fecha tentativa de presentación; y/o reasignar jurado a la presentación de la tesis. Generalmente el jurado presente para la presentación de seminario, repite para la presentación de la tesis.
8. Consejo de escuela evalúa la Planilla y la envía a Control de Estudios, indicando su decisión.
9. Control de estudios recibe la planilla, donde se indica si la inscripción del estudiante es válida o no y notifica al estudiante.
10. El estudiante contacta al jurado asignado para establecer fecha de presentación y su tutor debe solicitar a Control de Estudios un Acta Final de Notas donde se coloca la nota, y se envía a control de estudios, culminando el proceso.

Escuela de Matemática.

Prof. Francisco Tovar, Director de la Escuela de Matemática.

Lugar: Escuela de Matemática, Laboratorios de Computación Grafica.

Fecha: 26 de septiembre de 2011

Resumen de la reunión:

Se nos informó del proceso que se lleva a cabo en la Escuela de Matemática para la selección de jurados, el proceso empieza al momento en que el alumno inscribe el TEG, y se puede resumir en los siguientes pasos:

1. El alumno inscribe la materia TEG, habiendo consultado con su tutor tema, título de la tesis, jurado tentativo, plan de trabajo.
2. El tutor llena la Planilla de Inscripción del Proyecto de TEG con los datos consultados por su tutorado y envía la planilla al consejo de escuela.
3. Consejo de Escuela evalúa la Planilla y la envía a Control de Estudios, indicando su decisión.
4. Control de Estudios recibe la planilla, donde se indica si la inscripción del estudiante es válida o no y notifica al estudiante.
5. El estudiante contacta al jurado asignado para establecer fecha de presentación y su tutor debe solicitar a Control de Estudios un Acta de Notas donde se coloca la nota y debe ser enviada a Control de Estudios, culminando el proceso.

ANEXO 2: Modelo de Encuesta aplicada a usuarios finales del sistema CONEST (Módulo de solicitudes de jurado para Seminario y TEG)

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Encuesta

Trabajo Especial de Grado: Desarrollo Orientado a Comportamiento. Caso de Estudio: Solicitudes de Jurado para Seminarios y Trabajos Especial de Grado.

Siendo 1 - Poco acertado 2- Acertado 3 - Muy acertado, indique si el sistema cumple con las siguientes afirmaciones:

Afirmaciones	1	2	3	Observaciones
Muestra de manera rápida y sencilla la información necesaria para realizar las tareas relacionadas a las solicitudes de jurado para TEG y seminario				
Minimiza las posibilidades de cometer errores y provee *feedback* en caso de cometerlo				
Provee una correcta automatización de los procesos actuales llevados a cabo para las tareas relacionadas a las solicitudes de jurado para TEG y Seminario				
La información, diseño, apariencia y organización de los elementos mostrada es consistente y posee un estándar en la aplicación				
Los formularios presentados recopilan la información necesaria para cumplir con las tareas y son fáciles de entender				

Recomendaciones generales