DESARROLLO DE UN SISTEMA DE INFORMACIÓN PARA LA COMERCIALIZACIÓN DE CUPONES DE ALTA SEGMENTACIÓN EN FACTURAS DE SERVICIOS PÚBLICOS[1]

Ana Milena Rojas Calero *

Institución Universitaria Antonio José Camacho, Facultad de Ingenierías. Colombia

amrojas@admon.uniajc.edu.co

Manuel Alejandro Pastrana Pardo

Institución Universitaria Antonio José Camacho, Facultad de Ingenierías. Colombia

mapastrana@admon.uniajc.edu.co

Michael Robert Rosero Salazar

Estudiante de Ingeniería de Sistemas, Institución Universitaria Antonio José Camacho, Facultad de Ingenierías. Colombia

Recibido/Received: 26/06/2018

Aceptado/Accepted: 31/08/2018

RESUMEN

El presente documento refleja el resultado del proyecto de grado "Desarrollo de un sistema de información para la comercialización de cupones de alta segmentación en facturas de servicios públicos" como respuesta a la necesidad de las empresas Carvajal S.A. y Gases de Occidente, en conjunto con Atinna S.A., que realiza estrategias publicitarias en ciudades como Bogotá y Cali destinadas a impulsar el crecimiento de las empresas mediante campañas de fidelización de clientes. En conjunto estas tres empresas idearon una nueva campaña basada en una aplicación web que revolucionará la forma de ver la publicidad a través de las estrategias que implementarán y garantizarán la suficiente interactividad, adaptabilidad y usabilidad para incrementar los niveles de satisfacción de sus usuarios al interactuar con el sistema, garantizando que así la publicidad sea mejor recibida y surta efecto. El papel de los autores fue garantizar, mediante un proceso formal de desarrollo de software que cumple con las etapas de análisis, diseño desarrollo, verificación y despliegue, que el aplicativo implementado cumpliera con los lineamientos requeridos para el éxito del proyecto.

PALABRAS CLAVES

Desarrollo web, arquitectura de software, web development, software architecture.

---

[1] Este artículo corresponde a un ejercicio investigativo realizado por el aspirante a Grado Michael Robert Rosero Salazar, bajo el acompañamiento de los profesores investigadores coautores de este trabajo.

* Autor para correspondencia/ Corresponding autor: Ana Milena Rojas Calero. Facultad de Ingenierías. Institución Universitaria Antonio José Camacho. Avenida 6N # 28N – 102. Cali – Colombia.

Sugerencia de cita/ Suggested citation: Rojas-Calero, A.M., Pastrana-Pardo, M.A. y Rosero-Salazar, M.R. (2017). Desarrollo de un sistema de información para la comercialización de cupones de alta segmentación en facturas de servicios públicos. *Revista ACTITUD, 15(1)*, 5-14.

ABSTRACT

This document shows the result of the system engineering degree project "development of an information system for the commercialization of highly segmented coupons in utility bills" in response to the needs of the companies Carvajal SA, Gases de Occidente and Atinna SA, who carries out advertising strategies in cities such as Bogotá and Cali that promote the growth of companies through customer loyalty campaigns. Together these three companies think up a new campaign based on a web application which will revolutionize the way of viewing advertising through the strategies to implement and guarantee enough interactivity, adaptability and usability to increase the levels of satisfaction of its users when interacting with the system, ensuring that advertising is better received and takes effect. The role of the authors was to guarantee through a formal process of software development that follows the stages of analysis, design, development, verification and deployment that the implemented application would comply with the guidelines required for the success of the project.

KEYWORDS

Web development, software architecture, web development, architecture software.

INTRODUCCIÓN

Los procesos de desarrollo de software se centran normalmente en el tema funcional, es decir, que el software haga lo que su especificación de requisitos funcionales expresa, pero para lograr esto sin sobrepasar el presupuesto de tiempo y dinero, en muchas ocasiones se ven abocados a sacrificar aspectos como: la usabilidad, la mantenibilidad, la modificabilidad que pueden resultar de vital importancia para quienes interactúan con el aplicativo, ya sea en su uso diario o en su mantenimiento a través del tiempo. Pero en realidad el desarrollo de toda aplicación que se realiza en las empresas, debe estar adecuado para garantizar la satisfacción sus usuarios mediante la capacidad de entender el sistema, usarlo de manera fácil e intuitiva y garantizar que su experiencia no resulte frustrante generando rechazo en su interacción. Como lo refleja el trabajo de (Hastie & Wojewoda, 2015).

De acuerdo a la información entregada por Bárbara & Frick, compañía que solicita el desarrollo, 'CUPONMIGO', que es como han denominado a la aplicación, espera interactuar en promedio con unos 6000 usuarios, en una concurrencia promedio de 300 a 3000 usuarios simultáneos, distribuidos en todos los departamentos de Colombia. Debido a este escenario altamente concurrente y transaccional es de gran importancia el diseño de la arquitectura de la solución que satisfaga los atributos de calidad mediante la implementación de patrones de diseño, como propone el trabajo de Sommerville (2011). Adicionalmente se hace necesario la correcta selección de tecnologías que serán usadas para la implementación.

La unión de la empresa Carvajal S.A., Gases de Occidente y Atinna S.A. genera la necesidad de crear un aplicativo por parte de Bárbara & Frick, centrando el proceso de desarrollo en la experiencia de usuario final. Este aplicativo registrará la información necesaria para la distribución de cupones promocionales en el Departamento del Valle del Cauca.

Por lo anterior, el objetivo general del proyecto es el construir una aplicación web para la empresa Bárbara & Frick, que permita poder distribuir pauta publicitaria a través de las facturas de servicios públicos que proporciona la empresa Gases de Occidente.

En este artículo se van a presentar los resultados del proyecto de grado necesarios para cumplir con los objetivos del proyecto de grado encaminados a la construcción del aplicativo que Bárbara & Frick ha solicitado.

MARCO TEÓRICO

Las etapas del proceso de desarrollo de software están consideradas por diversos autores de la siguiente manera:

- Análisis: etapa donde se recopila la información necesaria para funcionalmente determinar qué debe hacer el sistema, delimitar el alcance del mismo y delimitar las restricciones a las que se enfrenta el proyecto, como indican Pressman & Ph (n.d.).

- Diseño: etapa donde el equipo de trabajo determina cómo construir el sistema basado en los atributos de calidad del sistema, esto se hace de acuerdo a lo planteado por Pressman & Ph (n.d.). Aquí toma relevancia que cada atributo se correlaciona con un patrón de diseño, que no es más que una solución que ya existe, está probada y funciona idealmente para solucionar temas particulares, como muestra el trabajo de Larman (2004). En su mayoría estos patrones son provistos por frameworks de desarrollo.

- Desarrollo y calidad: dentro de los modelos tradicionales de desarrollo de software como Rational Unified Process (RUP) y Microsoft Solution Framework (MSF), las etapas de desarrollo de software y calidad o pruebas son secuenciales y hasta no terminar el desarrollo no se inician las pruebas. Pero bajo los modelos ágiles se plantea que estos dos pasos van unidos y se ejecutan al mismo tiempo, orientando un marco preventivo que potencie los resultados finales.

- Despliegue, etapa donde el equipo de trabajo pone en ambiente final, denominado también de producción, el aplicativo para ser utilizado por todos los usuarios que se ha estimado interactuarán con él, como indica Roger S. Pressman (2010).

Para efectos de este artículo los autores centran la atención en la etapa de diseño, donde se sustentan los aspectos guías de la arquitectura a través de los atributos de calidad y los patrones de diseño que soportan la solución. Para esto es importante referenciar el modelo de 4 vistas más 1 planteado por (Kruchten, 2006) que plantea una revisión de los componentes que integran la solución desde diversas perspectivas como lo muestra la Figura 1.

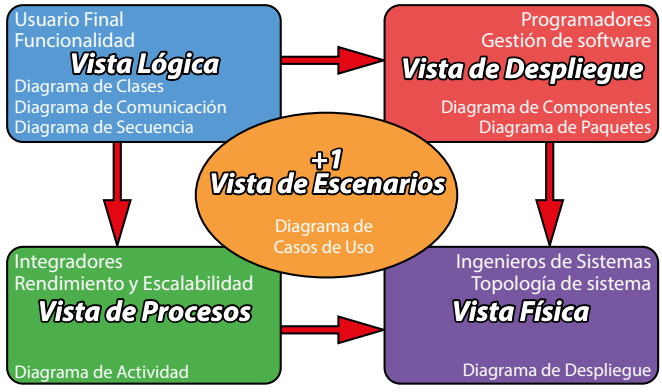

Fig. 1. Modelo 4 vistas + 1 Modelo (Moya, 2012)

Cada vista de las anteriormente referenciadas en la Figura 1, presenta las decisiones de diseño reflejadas a través de diagramas UML específicos que denotan la implementación de patrones de diseño que satisfacen los atributos de calidad propuestos. A continuación, se detalla un poco más cada una:

- **La vista lógica:** con base en la definición propuesta en el trabajo de Kruchten (2006) y reforzada en el trabajo de Roger S. Pressman (2010), la vista lógica comprende las abstracciones fundamentales del sistema a partir del dominio de problemas. En esta vista se representa la funcionalidad que el sistema proporcionará a los usuarios finales. Es decir, se ha de representar lo que el sistema debe hacer, y las funciones y servicios que ofrece. Para completar la documentación de esta vista se pueden incluir los diagramas de clases, de comunicación o de secuencia de UML.

- **La vista de proceso:** siguiendo bajo la definición del trabajo de Kruchten (2006) esta vista comprende el conjunto de procesos de

ejecución independiente a partir de las abstracciones anteriores y su comunicación entre sí. Roger S. Pressman (2010) señala que esta vista se representa desde la perspectiva de un integrador de sistemas, y de manera específica describe el flujo de trabajo paso a paso, los componentes negocio y operacionales que conforman el sistema. Para completar la documentación de esta vista se puede incluir el diagrama de actividad de UML.

- **La vista física:** según Kruchten (2006), esta vista muestra desde la perspectiva de un ingeniero de sistemas todos los componentes físicos del sistema, así como las conexiones físicas entre esos componentes que conforman la solución (incluyendo los servicios).

- Para completar la documentación de esta vista se puede incluir el diagrama de distribución de UML.

- **La vista de desarrollo:** Para definir esta vista, Kruchten (2006) invita a ver el sistema desde la perspectiva de un programador el sistema a partir de los componentes que lo integran, reflejando dentro de los diagramas UML los componentes necesarios que permiten la composición de capas de software y dan paso al flujo de mensajes modelando comportamientos predeterminados que garanticen óptimos resultados operativos. Para completar la documentación de esta vista, Roger S. Pressman (2010) sugiere que se pueden incluir diagramas de componentes y/o de paquetes.

- **Vista de Escenarios (+1):** Kruchten (2006) indica que esta vista es representada por casos de uso o por historias de usuario y tiene la función de unir y relacionar las otras 4 vistas. En complemento de lo anterior, Roger S. Pressman (2010) sugiere que en esta vista se pueden incluir el diagrama de casos de uso de UML, diagramas de escenario y diagramas de estado.

- En el caso del framework ágil SCRUM, esta vista no se usa debido a que es representada de una manera más flexible y precisa, con el uso de las historias de usuario.

Como se mencionó anteriormente, el eje central para el uso de estas vistas son los atributos de calidad y su correlación con los patrones de diseño. La siguiente tabla menciona los principales:

Atributo de Calidad	Descripción	Tácticas / Patrón de Arquitectura
Confidencialidad	Es la ausencia de acceso no autorizado a la información (Barbacci & Kazman, 1997)	**Patrón rol base access:** es un enfoque de restricción del acceso al sistema a los usuarios autorizados mediante roles.
		Patrón autenticador: mediante este patrón con un grado de complejidad garantiza que no cualquier personal no autorizado pueda ingresar a nuestra aplicación.
		Patrón trusted subsystem: mediante esta técnica se prevé protección contra el acceso a datos e información no autorizado, ya sea accidental o deliberadamente.
Portabilidad	Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos.(Barbacci & Kazman, 1997)	**Patrón multi-Layers:** Consiste en estructurar aplicaciones que pueden ser descompuestas en grupos de subáreas, las cuales se clasifican de acuerdo a un nivel particular de abstracción.

Usabilidad	Se refiere a la facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto. (Barbacci & Kazman, 1997)	**Patrón aprendibilidad:** Mediante este patrón, se prevé que el usuario entienda el mecanismo del sistema y el objetivo de que cumpla con todos los pasos del mismo.
		Patrón eficiencia: Con este patrón se garantiza que las respuestas del controlador se muestren de manera rápida y efectiva. Mediante uso de llamadas AJAX.
		Patrón protección contra errores de usuario: Mediante este patrón, se evita que el usuario cometa errores.
		Patrón estética de la interfaz de usuario: Mediante este patrón se trata que la interfaz agrada y satisfaga la interacción con el usuario.
Mantenibilidad	Es la capacidad de someter a un sistema a reparaciones y evolución (Barbacci et al., 1995).	**Patrón Model-ViewControler:** Divide una aplicación interactiva en tres componentes. El modelo (model) contiene la información central y los datos. Las vistas (view) despliegan información al usuario. Los controladores (controllers) capturan la entrada del usuario. Las vistas y los controladores constituyen la interfaz del usuario.
Modificabilidad	Capacidad de modificar el sistema de manera rápida y a bajo costo (Folmer, Gurp, & Bosch, 2005)	

METODOLOGÍA

El proceso de desarrollo de software usado para la elaboración del producto se centra en el marco de trabajo ágil SCRUM y en las etapas tradicionales descritas a continuación en detalle:

ANÁLISIS

En esta fase se elicitan los requisitos o necesidades funcionales y las restricciones del sistema a través del product backlog, que contiene la recopilación de la información mediante historias de usuarios, detallando en estas, los criterios de aceptación para cumplir con la funcionalidad del sistema

Debido a que las historias de usuario, no poseen un estándar dentro del mundo ágil, a diferencia de la norma IEEE 830 en su última revisión y que aplica para los casos de uso utilizados en metodologías como RUP, deben ser elaboradas con un modelo usado y reconocido en la industria. Por lo anterior, este proyecto se basa en la estructura propuesta por Cohn (2004) para su construcción: Yo como [usuario] requiero [una funcionalidad] para [un motivo], bajo los criterios de aceptación [definidos].

Una vez obtenida la lista de funcionalidades a través de historias de usuario, se realizan una serie de validaciones con el objetivo de constatar la conformidad por parte del cliente con lo recopilado en el artefacto mencionado, y aclarar dudas respecto a funcionalidades o requerimientos antes de comenzar el diseño y la construcción. Lo anterior no implica que esta fase termine aquí, dado que en cada planning meeting nuevamente se revisa el detalle actual de las historias de usuario y estas son susceptibles al refinamiento constante, debido a estar escritas en lenguaje natural y no en un lenguaje de única interpretación como Business Proces Models, como sugiere Pastrana, Ordóñez, Ordonez, Thom, & Merchan (2018).

PLANEACIÓN

Determina el alcance total del esfuerzo requerido para elaborar el proyecto. Según PMI (2004), el orden de avance de la planificación, inicia con el registro de las partes interesadas y el acta de constitución de proyecto; con estos resultados se procede a:

- Elaborar el plan de gestión de los interesados, el cual se va a representar en una matriz de los interesados.
- Elaborar la matriz de la comunicación de los interesados y el equipo.
- Elaborar el presupuesto.
- Elaborar el plan de gestión de riesgos, el cual se representa en la matriz de riesgos.

Durante la planeación SCRUM juega un papel muy importante, previo al inicio del proyecto, el product backlog, que es estimado mediante la técnica SCRUM POKER y priorizado por medio de MoSCoW para una mejor comprensión del trabajo total a realizar.

DISEÑO

Tomando como referencia los requisitos del cliente, se procede a modelar el software bajo el estándar UML dentro de un modelo 4+1 de Kruchten (2006).

Así mismo, con la herramienta mockflow se construyeron los prototipos de pantalla, asimismo, el diseño gráfico utilizado para dar una sensación visual agradable al usuario fue realizado con Ilustrator.

Es importante resaltar que la vista física, representa el uso de modelo en la nube tipo IaaS (Infraestructura como Servicio). Los servicios de Amazon Web Services permiten utilizar contenedores de sistemas operativos y de base de datos que simplifican el modelo de mantenimiento y administración de los mismos, por lo que se adopta el uso de un contenedor EC2 con el sistema operativo de Linux.

CONSTRUCCIÓN

Basado en la etapa de análisis y la solución obtenida en la etapa de diseño, se procede a la codificación del proyecto, donde resalta el estilo arquitectural Modelo Vista Controlador. A su vez debe tener en cuenta las pruebas funcionales, y los estándares de codificación para garantizar el buen funcionamiento de la aplicación y el cumplimiento de los objetivos del proyecto.

Las herramientas a utilizar para esta etapa son el editor Brackets. Desde la capa de lógica y acceso a datos, el lenguaje PHP en su versión 7.0 y para la capa de presentación se utiliza el framework para aplicaciones de internet enriquecida (Rich Internet Applications o RIA) Bootstrap, la librería JQuery. Por último, como servidor de aplicaciones se utiliza APACHE 2.4.

PRUEBAS E INTEGRACIÓN

Las pruebas de software son la investigación empírica y técnica realizada para facilitar a los interesados información sobre la calidad del producto (Kaner, 2006).

Para realizar esto, se hace la implementación de pruebas automatizadas a partir del uso de PHPUNIT y la herramienta de integración continua Jenkins, estas pruebas se centran en comprobar el funcionamiento del código y que este cumpla con todos los requisitos solicitados de manera preventiva y antes de las pruebas con el usuario final. Además de validar el código, también se verifican distintas variables, como la revisión de los parámetros en los métodos, el tipo de dato que se devuelve, el estado final de los métodos, entre otros.

Una de las ventajas de las pruebas unitarias es que fomentan al cambio, y permiten la reestructuración del código implementado (refactorización), puesto que permiten hacer pruebas sobre los cambios y verificar que no se han introducido errores (regresión).

DESPLIEGUE

El despliegue se realizó considerando una concurrencia aproximada de 3000 usuarios simultáneos, en todo el país. Para esto se realizaron constantemente pruebas no funcionales de carga, estrés y volumen que permitieran prever posibles fallos antes de su puesta en marcha definitiva y garantizar así su correcta operación en el ambiente de producción.

RESULTADOS

Para la etapa inicial del proyecto se definió un documento de product backlog aprobado por todos los interesados que dio paso a la especificación funcional del sistema. Durante la etapa de diseño posterior a la selección de atributos de calidad y patrones, se elaboró el documento de arquitectura de software en el que destacan a nivel de UML los resultados de la toma de decisiones. A continuación, se expone la vista de despliegue que sirve de guía principal para el equipo a la hora de interpretar el modelo a seguir en la construcción del proyecto.

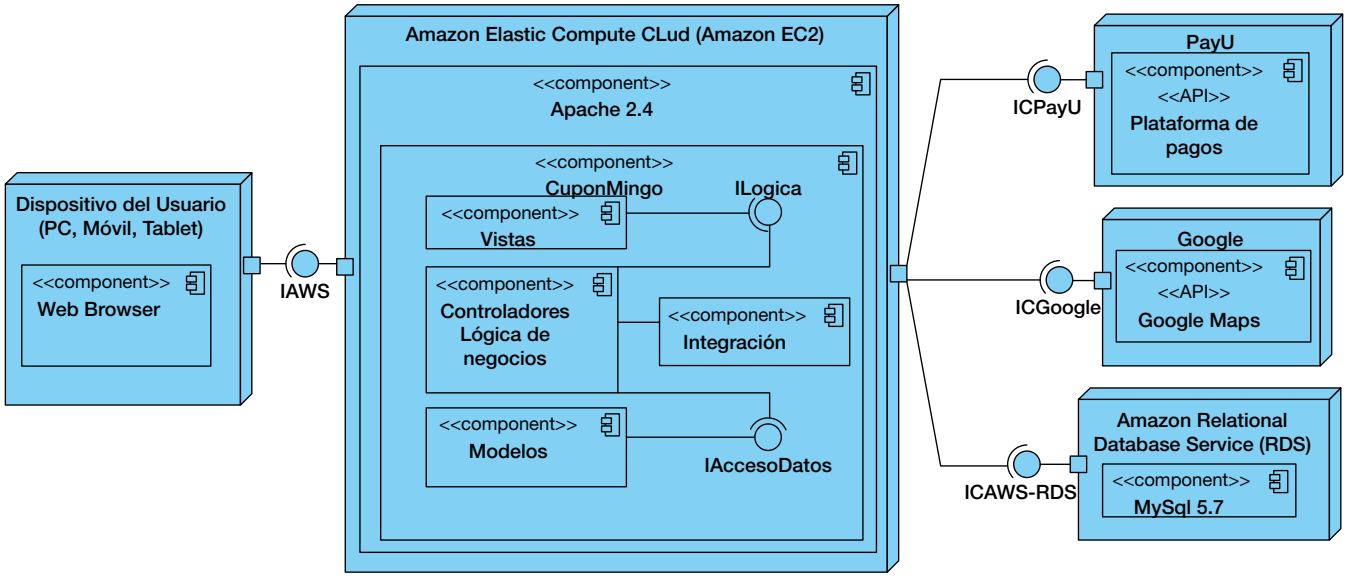

Fig. 2. Vista de despliegue. Realizado por los autores

En la figura 2 se aprecia que el estilo arquitectural principal está guiado por el Modelo Vista Controlador, este modelo genera una interacción del usuario final con la vista que, mediante peticiones HTTP/POST, se comunica con los controladores pertenecientes a una capa de lógica de negocios completamente separada, tanto de la capa anterior denominada presentación y de la siguiente capa de acceso a datos, que se encarga de la manipulación de los modelos y las peticiones a la base de datos.

Lo anterior lleva al equipo de trabajo a seleccionar las siguientes tecnologías para la elaboración del proyecto:

TECNOLOGÍAS	
BD	Mysql 5.7
FRAMEWORK TEST	PhpUnit
MAQUETACIÓN	Html5- css3
LENGUAJE FRONT-END	Javascript (jquery)
LENGUAJE BACK-END	Php 7.0
FRAMEWORK FRONT-END	Boostrap 3.7
SERVIDOR DE APLICACIONES	Apache 2.4
SERVIDOR WEB	Amazon Elastic Compute Cloud (Amazon EC2)
SEVIDOR DE BASES DE DATOS	Amazon Relational Database Service (RDS)
SISTEMA DE GESTIÓN DE BD	Workbench 6.3
INTEGRACIÓN CONTINUA	Jenkins 2.46
SERVIDOR DE REPOSITORIOS	Bitbucket
CLIENTE GIT	SourceTree 2.4.7
EDITOR DE CÓDIGO	Brackets 1.10

Durante el proceso de calidad, las pruebas unitarias realizadas mediante el framework PhpUnit juegan un papel crucial en el resultado final del producto de software. Adicionalmente, se hace uso de la herramienta de integración continua Jenkins y del versionador de código Bitbucket. Gracias a este ecosistema de calidad preventiva, a raíz de cada actualización del código fuente en el repositorio todo el proceso se inicia y se mide constantemente el avance o retroceso de los resultados, permitiendo la mejora continua como una constante durante todo el proceso. En la Figura 3 se refleja el comportamiento de estas herramientas:

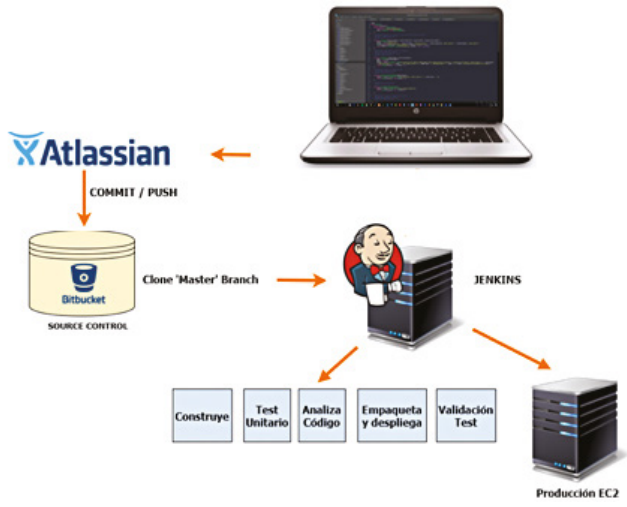

Fig. 3. Integración y Pruebas Unitarias. Realizado por los autores

A continuación, se incluye un ejemplo práctico sobre la clase OperacionesDB.php para mayor comprensión del lector sobre las pruebas unitarias:

- Test sobre un método 'create', que se encarga de insertar información en la base de datos.

```
public function testCreate(){
  $info = array("nombre_departamento"=>"NOVACÁ","total_personas"=>2423319);
  $this->assertNotNull($this->InstanceOperacionesDb->create('sp_loc_departamento',$info));
  $this->assertGreaterThan(0, $this->InstanceOperacionesDb->create('sp_loc_departamento',$info));
}
```

Fig. 4. Prueba Unitaria Método Aplicación realizado por los autores.

La figura 4 analiza el método 'create', en donde se realizan las respectivas aserciones. Adicionalmente, se valida que el registro no retorne ningún valor nulo, y también se valida que el resultado del registro debe retornar un valor mayor a 0.

Visto desde Jenkins la Figura 5 muestra el resultado de la ejecución de la prueba unitaria.

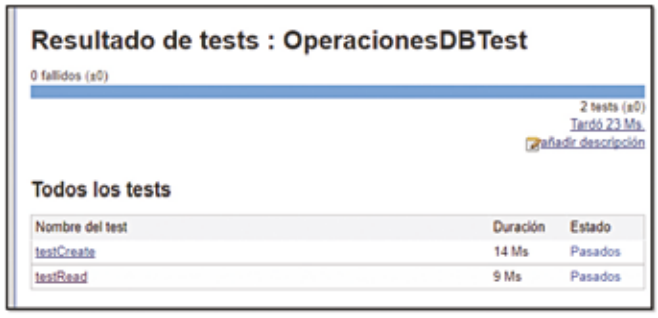

Fig. 5. Resultado de la Prueba unitaria en Jenkins. Realizado por el autor

Adicionalmente, por parte de BitBucket, también se realiza una revisión de la calidad de los resultados a través del avance del proyecto. En la figura 6 se aprecia el comportamiento de las pruebas unitarias del proyecto indicando la cantidad de errores en color rojo y las pruebas exitosas en azul.

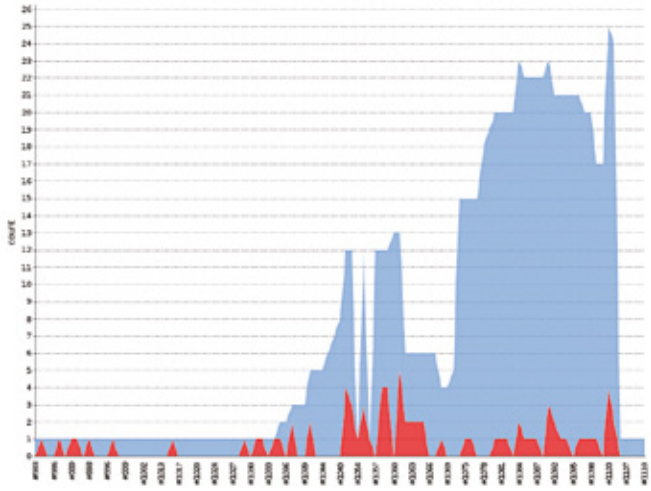

Figura 6 Gráfico de total de Pruebas Unitarias. Realizado por los autores.

Finalmente, el proyecto fue elaborado y entregado a satisfacción de todos los interesados y actualmente está puesto en producción.

CONCLUSIONES

El trabajo de grado permitió realizar la aplicación de todos los conocimientos adquiridos durante la formación profesional en la Institución Universitaria Antonio José Camacho en un ambiente real que reta estos conocimientos y donde el resultado final es un producto de alta calidad que cumple con lo requerido por todos los involucrados en el proceso.

Adicionalmente, el realizar el desarrollo de software dentro de un marco ágil como SCRUM permite que el proceso sea muy fluido, retroalimentando constantemente al equipo sobre cómo lograr el estado ideal a partir de la mejora continua y las buenas prácticas de calidad incluidas dentro del proyecto.

También se evidencia que el uso de patrones de diseño de software facilita significativamente la selección de tecnologías y el uso de frameworks para agilizar el proceso de desarrollo de software.

REFERENCIAS BIBLIOGRÁFICAS

Barbacci, M. R., & Kazman, R. (1997). Software architecture evaluation panel. In *Proceedings Twenty-First Annual International Computer Software and Applications Conference (COMPSAC'97)* (pp. 160–161). Retrieved from http://doi.org/10.1109/CMPSAC.1997.624780

Cohn, M. (2004). *User Stories Applied: For Agile Software Development (Addison Wesley Signature Series). Writing* (Vol. 1). http://doi.org/10.1017/CBO9781107415324.004

Folmer, E., Gurp, J. Van, & Bosch, J. (2005). Software *Architecture* Analysis of Usability. Architecture, 38–58. Retrieved from http://www.springerlink.com/index/87jl9wkck2gwyyd4.pdf

Hastie, S., & Wojewoda, S. (2015). Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch. Retrieved from http://www.infoq.com/articles/standish-chaos-2015

Kaner, C. (2006). Exploratory Testing. *Quality Assurance International,* (c), 1–47. Retrieved from http://www.kaner.com/pdfs/ETatQAI.pdf

Kruchten, P. (2006). Planos Arquitectónicos: El Modelo de " 4 + 1 " Vistas La Arquitectura del Software. *IEEE Software,* 12(6), 1–16.

Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Analysis*. Retrieved from http://doi.org/10.1016/j.nec.2006.05.008

Pastrana, M., Ordóñez, H., Ordonez, A., Thom, L. H., & Merchan, L. (2018). Optimization of the Inception Deck Technique for Eliciting Requirements in SCRUM Through Business Process Models. *Business Process Management Workshops,* 4928(January), 649–655. Retrieved from http://doi.org/10.1007/978-3-319-74030-0_52

PMI. (2004). *PMI Fundamentos para la dirección de proyectos (Guía del PMBOK). Fundamentos*. Retrieved from http://doi.org/10.15611/ie.2014.1.14

Pressman, R.S. (2010). *Ingenería de software - Un enfoque práctico. Danielr.Obolog.Es.* http://doi.org/http://zeus.inf.ucv.cl/~bcrawford/Modelado%20UML/Ingenieria%20del%20Software%207ma.%20Ed.%20-%20Ian%20Sommerville.pdf

Sommerville, I. (2011). *Ingeniería del software. Software engineering.* http://doi.org/10.1111/j.1365-2362.2005.01463.x