Ciencia Digital
www.cienciadigital.org

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

Control de calidad del software mediante pruebas automatizadas de integración y pruebas unitarias.

Quality Control of the software using automated testing of integration and unit tests.

Santiago Viteri Arias.[1], Tatiana Mayorga Soria. [2], Patricio Navas Moya. [3] & Patricio Molina Palma. [4]

Recibido: 13-03-2017 / Revisado: 10-05-2017 Aceptado: 11-06-2018/ Publicado: 01-07-2018

DOI: https://doi.org/10.33262/cienciadigital.v2i3.140

Abstract.

Currently testing to control the quality of software development is the most important process, if the goal is to automate institutional processes, so it requires more care and dedication in this phase. In the field of tests there are several manual ways to perform tests on a software product, although when running automated tests this process is done more efficiently and quickly. In the present investigation a comparative analysis of Automated Tests of Integration and Unit Tests will be carried out to investigate and analyze the best quality management, time and resources that are used at the time of testing a software product, to be able to apply the automated tests. has many tools, however one of the most used is the construction of test cases where we can define conditions on the different methods and thus assess whether they are correct or not, so that corrections can be established in time, for this we will have the Visual Studio tool In Mvc, TestAgent Management and a Management application in MVC to which the corresponding tests will be made

Keywords: Automated Tests, Unit Tests, Software, Quality.

Resumen.

En la actualidad las pruebas para controlar la calidad en el desarrollo de software es el proceso más importante, si el objetivo es automatizar procesos institucionales, por lo

[1] Universidad de las Fuerzas Armadas ESPE, Cotopaxi, Ecuador, csviteri1@espe.edu.ec
[2] Universidad de las Fuerzas Armadas ESPE, Cotopaxi, Ecuador, ptmayorga@espe.edu.ec
[3] Universidad de las Fuerzas Armadas ESPE, Cotopaxi , Ecuador, mpnavas@espe.edu.ec
[4] Universidad de las Fuerzas Armadas ESPE, Cotopaxi, Ecuador, pamolina6@espe.edu.ec

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

que requiere más cuidado y dedicación en esta fase. En el campo de pruebas existen varias formas manuales de realizar ensayos a un producto software, aunque al ejecutar pruebas automatizadas este proceso se realiza más eficiente y rápidamente. En la presente investigación se realizará un análisis comparativo de Pruebas Automatizadas de Integración y Pruebas Unitarias para investigar y analizar la mejor administración de calidad, tiempo y recursos que se utilizan al momento de realizar pruebas de un producto software, para poder aplicar las pruebas automatizadas se tiene muchas herramientas, sin embargo una de las más utilizadas es la construcción de casos de prueba donde podremos definir condiciones sobre los diferentes métodos y así evaluar si son correctos o no, de forma que se pueda establecer correcciones a tiempo, para esto se contará con la herramienta Visual Studio En Mvc, TestAgent Management y una aplicación de Gestión en MVC a la cual se le harán las pruebas correspondientes.

Palabras Claves: Pruebas Automatizadas, Pruebas Unitarias, Software, Calidad.

Introducción.

La intervención humana para realizar pruebas a los productos software, antes de entrar a producción, verifican si las interfaces son amigables, que tan intuitiva es la aplicación, que tan placentera es la experiencia de usuario. Sin embargo, en las pruebas manuales son más propensas a constantes errores, por ello se deben hacer más de una vez las pruebas, y, si es un sistema de varios módulos se consumen mucho tiempo y recursos. Las pruebas automatizadas es un elemento clave para garantizar la calidad, descartar inconsistencias y errores humanos de un producto software. “Son realizadas por una computadora” [1].

Las **pruebas unitarias** se ejecutan para comprobar el funcionamiento de los elementos de más bajo nivel de nuestro programa, puede ir desde probar el funcionamiento de una clase, hasta el funcionamiento de un método en particular[2].Las pruebas de integración son más complejas que las unitarias, estás están destinadas a comprobar el funcionamiento de dos o más componentes en el sistema y la forma en la cual funcionan en conjunto, de ahí su nombre, ya que si bien una clase puede funcionar adecuadamente por sí sola, luego al unirla a otra clase se pueden generar diversos problemas que debemos tener bajo control antes de pasar a producción. [3]

La calidad del *software* es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad. [4]

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

Pruebas Automatizadas.

Pruebas manuales son muy complejas y poco eficientes en el uso del tiempo y los recursos, desde hace un tiempo se maneja el esquema de pruebas automatizadas, esto se presenta de forma muy útil ya que en vez de pasar horas y horas pantalla tras pantalla, con solo correr un comando podremos obtener resultados de forma inmediata y sin errores.

Las pruebas desde sus inicios han requerido el factor humano, sin embargo, esto conlleva mucho tiempo y esfuerzo ya que debemos en primer lugar hace los diferentes casos que vamos a probar, establecer un listado de parámetros y mantenerlos por si ocurre un error poder detectarlo.

Pero no solo eso al momento de estar haciendo una *prueba de forma manual* podemos equivocarnos sin darnos cuenta, ya que como seres humanos estamos propensos a errores constantemente, aparte que si debemos probar un sistema de 20 o 30 módulos probablemente un día no sea suficiente.

Ahora aunque todo parezca desventajas, las pruebas manuales son muy buenas para detectar otro tipo de situaciones, por ejemplo podemos evaluar de forma personal si las interfaces son amigables, que tan intuitiva es la aplicación, que tan placentera es la experiencia de usuario, estos casos si son ideales para hacerlos de forma manual ya que es casi imposible que un programa de computadora nos pueda dar este tipo de información, sin embargo con algunas aplicaciones estadísticas se pueden obtener indicadores sobre el tema.

Se hará a un proyecto ya elaborado en el cual se va utilizar las pruebas automatizadas para investigar y optimizar las pruebas en un proyecto.

Para realizarlo, se utilizará visual studio y un proyecto en mvc realizado en el programa antes dicho.

Pruebas Humanas.

Las pruebas desde sus inicios han requerido el factor humano, esto conlleva mucho tiempo y esfuerzo ya que debemos en primer lugar hace los diferentes casos que vamos a probar, establecer un listado de parámetros y mantenerlos por si ocurre un error poder detectarlo [5]. Ahora aunque todo parezca desventajas, las pruebas manuales son muy buenas para detectar otro tipo de situaciones, por ejemplo podemos evaluar de forma personal si las interfaces son amigables, que tan intuitiva es la aplicación, que tan placentera es la experiencia de usuario, estos casos si son ideales para hacerlos de forma manual ya que es casi imposible que un programa de computadora nos pueda dar este tipo de información, sin embargo con algunas aplicaciones estadísticas se pueden obtener indicadores sobre el tema.

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

Pruebas Automatizadas.

Son las que se ejecutan sin la necesidad de la ayuda del hombre, cuando alguna prueba falla debe enviar una alarma para poder tomar las medidas necesarias, jamás se debe proseguir en el desarrollo si se tiene alguna alerta hasta la entrega o peor aún si entraría en producción dentro de estas podemos encontrar muchas pruebas pero principalmente se tiene las unitarias y las de integración. [6]

Pruebas Automatizadas de Integración.

Son las pruebas más inteligentes ya que ayudan a determinar si la integración de los datos es correcta y segura. Tener automatizadas estas pruebas son la forma más segura de de mejorar la calidad y sobre todo acelerar el desarrollo.

Desde hace un tiempo se maneja el esquema de pruebas automatizadas, esto se presenta de forma muy útil ya que en vez de pasar horas y horas pantalla tras pantalla, con solo correr un comando podremos obtener resultados de forma inmediata y sin errores. Son más complejas que las unitarias, estás están destinadas a comprobar el funcionamiento de dos o más componentes en el sistema y la forma en la cual funcionan en conjunto, de ahí su nombre, ya que, si bien una clase puede funcionar adecuadamente por sí sola, luego al unirla a otra clase se pueden generar diversos problemas que debemos tener bajo control antes de pasar a producción. [7]

Pruebas Automatizadas Unitarias.

Es como se comprueba el normal funcionamiento del código dentro de un programa, para la detección inmediata de defectos en métodos, mejor diseño detallado de la arquitectura. Código estructurado y menos complejo, documentación acorde a la realidad del aplicativo el tiempo invertido en el mantenimiento de la evolución. Dentro de este proceso el costo de las actividades influye en el resultado final ya que las pruebas unitarias sugieren que se debe probar luego de codificar mediante un debug del código. [2]

Pruebas automatizadas Unitarias vs Pruebas Automatizadas de Integración .

Pruebas manuales son muy complejas y poco eficientes en el uso del tiempo y los recursos, desde hace un tiempo se maneja el esquema de pruebas automatizadas, esto se presenta de forma muy útil ya que en vez de pasar horas y horas pantalla tras pantalla, con solo correr un comando podremos obtener resultados de forma inmediata y sin errores, En lo que difieren estas dos pruebas es en tiempo de realización.

Pero no solo eso al momento de estar haciendo una prueba de forma manual podemos equivocarnos sin darnos cuenta, ya que como seres humanos estamos propensos a errores constantemente, aparte que si debemos probar un sistema de 20 o 30 módulos probablemente un día no sea suficiente.

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

Experimentos y resultados.

Se procede a detallar los resultados de las pruebas de rendimiento entre las Pruebas Automatizadas de integración, pruebas unitarias y humanas.

Para realizar estas pruebas se empleó la versión 14.0.25431.01 Update 3 del Visual Studio Enterprise, ASP.NET Framework versión 14.1.11107.0 para el modelo de desarrollo web MVC y una base de datos SQL en Microsoft SQL server 2012 instalados en un equipo con Windows 10 de 64 bits con 4 GB de memoria RAM.

Las pruebas se utilizaron el Marco de pruebas MSTest con sus respectivas clases y métodos.

Las pruebas se hicieron para las cuatro operaciones: inserción, actualización y borrado.

Para comparar la eficiencia de los diferentes tipos de pruebas automatizadas se empleó 4 modelos con sus respectivos controladores y vistas.

Para las operaciones CRUD se usaron los siguientes módulos:

- Cliente
- Editorial
- Compra
- Libro

En la primera prueba se insertó datos mediante el controlador de cada modelo, a una base de datos vacía y se tomó el tiempo que se demoró en realizar la operación.

Para la ejecución de las pruebas del CreateTest se utilizó el siguiente código.

```
public void CreateCliente()
    {
      CLIENTEController controller = new CLIENTEController();

      using (var ts = new TransactionScope())
      {
        Assert.IsNotNull(controller.Create(new Models.CLIENTE()
        {
          CLI_NOMBRE = "Santiago",
          CLI_TELEFONO = "0999999999",
          CLI_CORREO = "csviteri1@espe.edu.ec",
      }));
```

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

```
    }
}
```

Para crear las pruebas unitarias se usa el siguiente código

El tiempo resultante en segundos de las pruebas unitarias, incluyendo el promedio final de la operación de inserción mediante el método CreateTest()

Tabla 1. Resultados de la operación inserción mediante el método CreateTest

	Tiempo en Segundos		
Registros	**Prueba 1**	**Prueba 2**	**Promedio**
Cliente	0,082	0,040	0,061
Editorial	0,033	0,018	0,025
Compra	0,099	0,064	0,082
Libro	0,019	0,07	0,045

Elaborado por: Grupo de investigación.

Para realizar las pruebas de integración se realizó las operaciones CRUD.

En la primera prueba se insertó datos mediante interfaz gráfica UI Test, a una base de datos local vacía y se tomó el tiempo que se demoró en realizar la operación.

Para la ejecución de las pruebas del CreateTest se utilizó el siguiente código.

```
[TestMethod]
    [DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",
"|DataDirectory|\\Users.csv", "Users#csv", DataAccessMethod.Sequential),
DeploymentItem("Users.csv")]
    public void CodedUITestCreate()
    {
        this.UIMap.RecordedMethodCreateNew();
        this.UIMap.AssertMethodCreateViewOpened();
        var uiCreate =
this.UIMap.UIIndexCodedUItestautoWindow.UICreateCodedUItestautDocument;
        this.UIMap.RecordedMethodCreateParams.UIFirstNameEditText =
TestContext.DataRow[“Nombre"].ToString();
```

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

```
this.UIMap.RecordedMethodCreateParams.UILastNameEditText =
TestContext.DataRow["Apellido"].ToString();
this.UIMap.RecordedMethodCreateParams.UIEmailEditText =
TestContext.DataRow["Email"].ToString();
this.UIMap.RecordedMethodCreateParams.UIDateOfBirthEditText =
TestContext.DataRow["telefono"].ToString();
this.UIMap.RecordedMethodCreate();
var uiIndex =
this.UIMap.UIIndexCodedUItestautoWindow.UIIndexCodedUItestautoDocument;
```

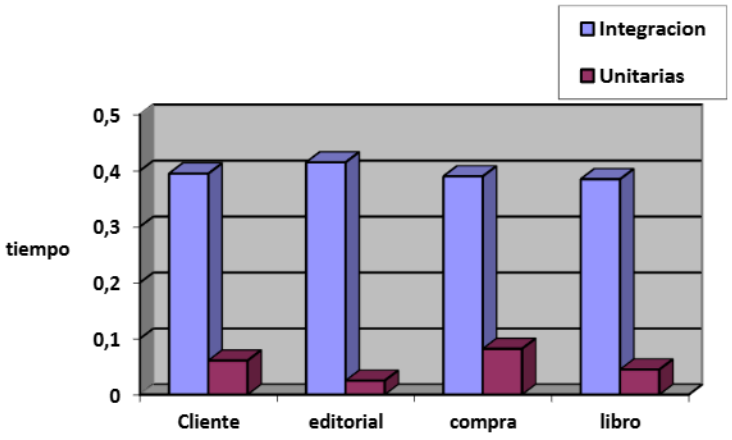
Tabla 2. Resultados de la operación inserción en los Test de Integración.

	Tiempo en Segundos		
Registros	**Prueba 1**	**Prueba 2**	**Promedio**
Cliente	0,41	0,38	0,395
Editorial	0,46	0,37	0,415
Compra	0,42	0.36	0.39
Libro	0,38	0.39	0,385

Elaborado por: Grupo de investigación.

Para las operaciones de inserción los resultados se muestran en la Figura 1.

Figura 1. *Resultados de las pruebas de inserción*



Elaborado por: *Grupo de investigación.*

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

En la segunda prueba se actualizo datos mediante el controlador de cada modelo, a una base de datos y se tomó el tiempo que se demoró en realizar la operación.

Para la ejecución de las pruebas del EditTest se utilizó el siguiente código.

```
[TestMethod()]
    public void EditTestCliente()
    {
        CLIENTEController controller = new
CLIENTEController();
        using (var ts = new TransactionScope())
        {
            Assert.IsNotNull(controller.Edit(new
Models.CLIENTE()
            {
                CLI_ID=4,
                CLI_NOMBRE = "Santiago",
                CLI_TELEFONO = "0999999999",
                CLI_CORREO =
"csviteri1@espe.edu.ec",
            }));
        }
    }
```

El tiempo resultante en segundos de las pruebas unitarias, incluyendo el promedio final de la operación de actualizar datos mediante el método EditTest()

Tabla 3, Resultados de la operación de actualización de datos mediante el método EditTest

	Tiempo en Segundos		
Registros	**Prueba 1**	**Prueba 2**	**Promedio**
Cliente	0,08	0,19	0,135
Editorial	0,03	0,55	0,29
Compra	0,041	0,1	0,07
Libro	0,012	0,07	0,04

Elaborado por: Grupo de investigación.

Para las pruebas de integración del editar se utilizó el siguiente código.

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

```
var uiIndex =
this.UIMap.UIIndexCodedUItestautoWindow.UIIndexCodedUItestautoDocument;

uiIndex.UIItemTable.UIBobCell.SearchProperties[HtmlHyperlink.PropertyNames.Inner
Text] = TestContext.DataRow["FirstName"].ToString();

uiIndex.UIItemTable.UIBobCell.SearchProperties[HtmlHyperlink.PropertyNames.Inner
Text] = TestContext.DataRow["LastName"].ToString();

uiIndex.UIItemTable.UIBobCell.SearchProperties[HtmlHyperlink.PropertyNames.Inner
Text] = TestContext.DataRow["Email"].ToString();

uiIndex.UIBobsmithsomedomaincoHyperlink.SearchProperties[HtmlHyperlink.Property
Names.InnerText] = TestContext.DataRow["DateOfBirth"].ToString();
        this.UIMap.AssertMethodUserModificar
```

Tabla 4. Resultados de la operación de actualización de datos en los Test de Integración

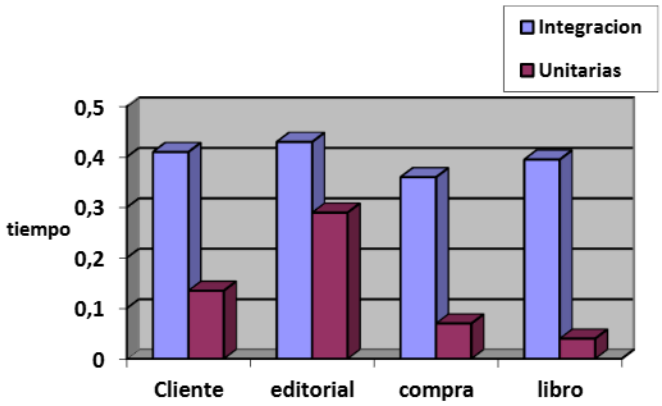
	Tiempo en Segundos		
Registros	**Prueba 1**	**Prueba 2**	**Promedio**
Cliente	0,43	0,39	0,41
Editorial	0,43	0.36	0.395
Compra	0,45	0.40	0.425
Libro	0,40	0.37	0,385

Elaborado por: Grupo de investigación.

Para las operaciones de actualización de datos, los resultados se muestran en la Figura 2.

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

Figura 2. Resultados de las pruebas de inserción.



Elaborado por: Grupo de investigación.

En la tercera prueba se eliminó datos mediante el controlador de cada modelo, a una base de datos y se tomó el tiempo que se demoró en realizar las operaciones respectivas.

Para la ejecución de las pruebas del DeleteTest se utilizó el siguiente código.

```
[TestMethod()]
    public void DeleteTest()
    {
        bibliotecaEntities db = new bibliotecaEntities();

        CLIENTE c = new CLIENTE();
        try
        {
            CLIENTEController controller = new
CLIENTEController();
            ActionResult result = controller.Delete(1) as
ActionResult;
            db.Entry(c).State = EntityState.Deleted;
            db.SaveChanges();
            Assert.IsNotNull(result);
        }
        catch (Exception e)
        {
        }
    }
```

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

El tiempo resultante en segundos de las pruebas unitarias, incluyendo el promedio final de la operación de eliminar datos mediante el método DeleteTest()

Tabla 5. Resultados de la operación de la eliminación de datos mediante el método DeleteTest()

	Tiempo en Segundos		
Registros	**Prueba 1**	**Prueba 2**	**Promedio**
Cliente	0,010	0,014	0,012
Editorial	0,062	0,021	0,042
Compra	0,021	0,015	0,018
Libro	0,044	0,061	0,053

Elaborado por: Grupo de investigación.

Resultados de las pruebas de integración.

Tabla 6. Resultados de la operación de eliminación de datos en los Test de Integración.

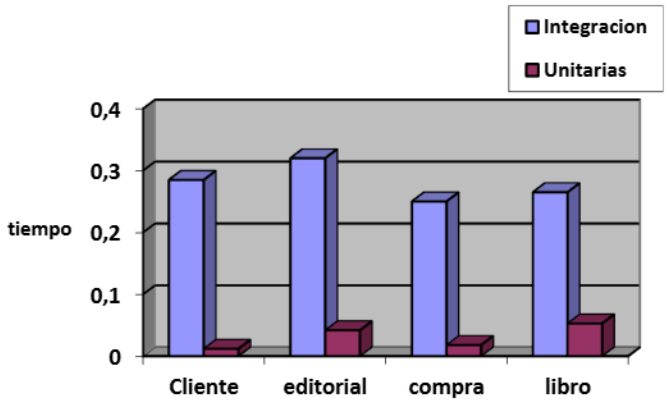
	Tiempo en Segundos		
Registros	**Prueba 1**	**Prueba 2**	**Promedio**
Cliente	0,29	0,28	0,285
Editorial	0,31	0,33	0,32
Compra	0,26	0,24	0,25
Libro	0,28	0,25	0,265

Elaborado por: Grupo de investigación.

Para las operaciones de eliminación de datos, los resultados se muestran en la Figura 3.

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

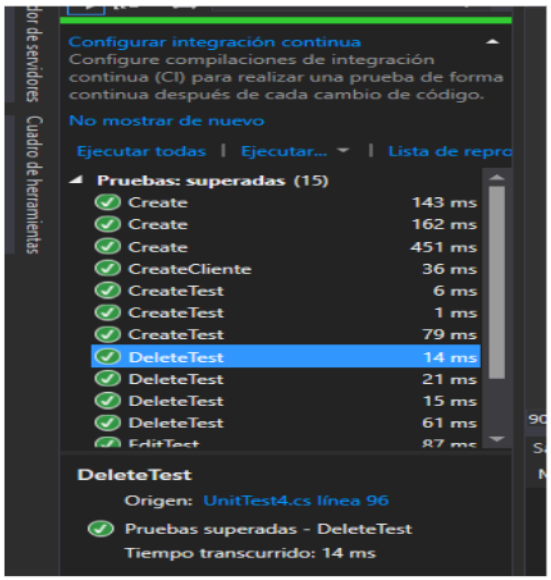
Figura 3. Resultados de las pruebas de eliminación.



Elaborado por: Grupo de investigación.

Para la realización de pruebas unitarias se recurrió a la utilización del Unitest que mide las consultas y como fueron desarrolladas de acuerdo a los niveles de calidad de la información

Figura 4. Pruebas Unitarias con Unitest.



Elaborado por: Grupo de investigación.

En la figura anterior se puede ver como se realizan las consultas y el tiempo que debió transcurrir para crear, almacenar, actualizar y eliminar la respectiva información.

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

Algunos Errores Comunes .

La secuencia de acciones desde el punto de vista de código, el **RecordedMethodIndex** abre la aplicación de la URL indicada, e inmediatamente después, el **AssertMethodIndexPageLoaded** comprueba que existe el control "Crear nuevo". Esta comprobación puede suceder demasiado pronto si la página web no se cargue por completo todavía, para solucionar este error y que la aplicación espere a que cargue la página se pondrá el siguiente código en el UIMap Test.

```
públicos vacíos AssertMethodIndexPageLoaded ()
{
uICreateNewHyperlink.WaitForControlReady ();
   Assert.AreEqual (...);
}
```

Conclusiones.

- En el ciclo de desarrollo software en la etapa de pruebas, aplicar un método eficiente para detectar errores y resolverlos a tiempo antes de unir los diferentes módulos desarrollados del proyecto, permite un ahorro del tiempo y evita tener errores a futuro, en sí todas las pruebas automatizadas tienen confiabilidad para realizarlas, pero todas tienen sus ventajas y desventajas como se ve en este artículo.
- Los diferentes tipos de pruebas empleados en el presente trabajo fueron los que más se utilizaban en un ambiente de desarrollo software y por ende fueron parte de este estudio de eficiencia y fiabilidad, las pruebas utilizadas fueron Unitarias y Pruebas de Integración.
- Se destacó las pruebas Untarías en tiempos de ejecución y evaluación de las operaciones de inserción, actualización y eliminación de datos.
- Para la detección de errores se usa el método Assert que devuelve un valor verdadero o falso para cada prueba realizada a las operaciones antes dichas de los datos y muestra si la prueba se realizó con éxito o no. El mejor en responder a este método fueron las pruebas Unitarias que permitieron detectar errores de forma más eficiente y con un mayor grado de exactitud, es decir, ahorra recursos, costos, evita redundancia de software y tiempo a comparación con las pruebas de integración con UI ya que el hecho de utilizar una interfaz gráfica hace que la ejecución de las pruebas se demoren más tiempo, estas pruebas tienen la misma fiabilidad de calidad que las unitarias pero ocupan más tiempo y recursos como se ven en los gráficos estadísticos (Figura 1, Figura 2, Figura 3).

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

Referencias bibliográficas.

Dustin, E., Rashka, J., & Paul, J. (1999). *Automated software testing: introduction, management, and performance*. Addison-Wesley Professional.

Esmite, I., Farías, M., Farías, N., & Pérez, B. (2007). Automatización y gestión de las pruebas funcionales usando herramientas open source. In *XIII Congreso Argentino de Ciencias de la Computación*.

Aristegui, J. L. (2010). Los casos de prueba en la prueba del software. *Lámpsakos*, (3), 27-34.

Graham, D., & Fewster, M. (2012). *Experiences of test automation: case studies of software test automation*. Addison-Wesley Professional.

Mosley, D. J., & Posey, B. A. (2002). *Just enough software test automation*. Prentice Hall Professional.

Dustin, E., Rashka, J., & Paul, J. (1999). *Automated software testing: introduction, management, and performance*. Addison-Wesley Professional.

Anon, (2017). [online] Available at: Article title: Pruebas automatizadas con ASP.NET MVC - Solvetic - Página 2 Website title: Solvetic.com URL: https://www.solvetic.com/tutoriales/article/1318-pruebas-automatizadas-con-aspnet-mvc/?_st=1 [Accessed 24 Feb. 2017].

ISSN: 2602-8085
Vol. 2, N°3, p. 101-115, Julio - Septiembre, 2018

Para citar el artículo indexado.

Viteri S., Mayorga T., Navas P. & Molina P. . (2018). control de calidad del software mediante pruebas automatizadas de integración y pruebas unitarias. *Revista electrónica Ciencia Digital* **2**(3), 101-115. Recuperado desde: http://cienciadigital.org/revistacienciadigital2/index.php/CienciaDigital/article/view/140/125