UNIVERSIDAD AUTÓNOMA DE ICA

FACULTAD DE INGENIERÍA, CIENCIAS Y ADMINISTRACIÓN

TESIS

"EFICACIA EN LA IMPLEMENTACIÓN DEL FRAMEWORK APLICADO A LOS PROCESO QA DENTRO DEL ÁMBITO DEL DISEÑO DE SOFTWARE EN UNA CONSULTORA"

PARA OPTAR EL TÍTULO DE:

INGENIERO DE SISTEMAS

PRESENTADO POR:

MARCO ANTONIO CHAVEZ ARRARTE

ICA – PERÚ

2019

DEDICATORIA

A mi familia, razón de mi existir.

RESUMEN

Actualmente muchas empresas siguen ciertas metodologías de desarrollo de software como el modelo en cascada, prototipado, incremental, scrum entre otros para así poder estructurar, planificar y controlar el proceso de desarrollo de software. La tendencia de la mayoría de estas empresas es utilizar metodologías ágiles, las cuales permiten una entrega de productos de software más rápido en comparación con las que siguen una metodología tradicional. Scrum es una metodología Ágil adaptable a los procesos de desarrollo de las empresas.

Uno de los principales procesos es el de QA, esta última busca garantizar que las especificaciones de los requerimientos mantengan concordancia antes, durante y después del desarrollo de software.

Belatrix es una consultora de software que brinda servicios de software a otras empresas, utiliza la metodología Scrum y siempre redefine sus procesos para dar un mejor servicio. Uno de los procesos que Belatrix ha mejorado es el proceso de QA.

Para el desarrollo óptimo de la ejecución de este proceso, se utilizará el framework de automatización para el proceso de QA, el cual busca apoyar al proceso en la reducción del tiempo de la actividad de la regresión, el cual consiste en la ejecución de todos los casos de pruebas.

INDICE

DEDICATORIA…….02

RESUMEN…….03

INDICE……...04

I. INTRODUCCIÓN……05
II. PROBLEMÁTICA DE LA INVESTIGACIÓN……...07
    a) DESCRIPCIÓN DE LA REALIDAD PROBLEMÁTICA……. 07
    b) FORMULACIÓN DEL PROBLEMA……..11
    c) JUSTIFICACIÓN DE LA INVESTIGACION……11
    d) HIPOTESIS……...12
    e) VARIABLES…… 12
III. OBJETIVOS DE LA INVESTIGACIÓN……....13
    a) OBJETIVO GENERAL……....13
    b) OBJETIVOS ESPECIFICOS……13
IV. MARCO TEORICO……14
V. METODOS O PROCEDIMIENTOS……..25
VI. RESULTADOS……26
VII. CONCLUSIONES Y RECOMENDACIONES……40
    a) CONCLUSIONES……40
    b) RECOMENDACIONES……..40
BIBLIOGRAFIA……..41

CAPÍTULO I: INTRODUCCIÓN

Para muchas organizaciones de desarrollo de software es de vital importancia reducir los costos de desarrollo mientras se mantiene alto la calidad del producto. Dado que las pruebas de QA comúnmente constituyen una parte significativa del tiempo de desarrollo, una forma de aumentar la eficiencia es encontrar más fallas tempranamente como se menciona en el artículo de Lars-Ola[1].

Belatrix es una empresa que se encarga de brindar servicios de software a diferentes clientes de Norteamérica y del mundo, por lo que constantemente sus clientes exigen un mejor servicios, por tal motivo Belatrix empezó con la reingeniería de los procesos de desarrollo de software. El proceso de QA pertenece a uno de los procesos de desarrollo de software y fue uno de los procesos que belatrix modificó añadiendo una actividad de Automatización con el que se podrá reducir el tiempo, siendo esto más atractivo para los clientes porque reducción de tiempo significa un costo más bajo y así continuar siendo atractivos para sus clientes a comparación de los clientes.

El presente Informe Profesional desarrolla la implementación del framework de automatización el cual representa la actividad de automatización del proceso de QA en la empresa Belatrix.

Debemos tener en cuenta que las actividades principales del proceso de QA que se realizan para todo desarrollo de software son:

1.Creación y casos de pruebas según los requerimientos.

2.Realizar el framework de automatización para automatizar las pruebas.

3.Validar las funcionalidades en base a los escenarios/casos de pruebas identificados.

4.Automatizar las funcionalidades de las pruebas de UI.

5.Reporte de la ejecución de los test cases automatizados.

6.Registro de las incidencias.

La actividad 1 consiste en identificar todas las casuísticas y flujos que se validará cuando el equipo de desarrollo despliegue la aplicación en el ambiente de pruebas.

La actividad 3 consiste en la ejecución de los casos de pruebas identificados en la actividad 1 y si se encuentra algún caso de prueba o escenario que no completa el flujo completo, entonces se ejecuta la actividad 7, el cual es identificar y crear una incidencia.

La actividad 2 consiste en diseñar una framework de automatización, en el que sobre este se ejecutar la actividad 4 y 5 las cuales consisten en que se pueda automatizar los casos de pruebas identificados en la actividad 1.

La actividad 6 consiste en obtener un reporte de ejecución de todos los casos de pruebas automatizados en la actividad 4 y 5.

Los sistemas de software pueden estar entre las construcciones más complejas en disciplinas de ingeniería y pueden abarcar años de desarrollo. La mayoría de los sistemas de software implementan, en parte, lo que ya se ha construido y tienden a seguir arquitecturas conocidas o casi conocidas. Aunque la mayoría de los sistemas de software no son del tamaño de Microsoft Windows 8, la complejidad del desarrollo de software puede aumentar rápidamente. Por lo tanto, entre estos métodos, lo más importante es el uso de patrones de arquitectura y diseño y marcos de software. Los patrones proporcionan soluciones conocidas para los problemas recurrentes que enfrentan los desarrolladores.

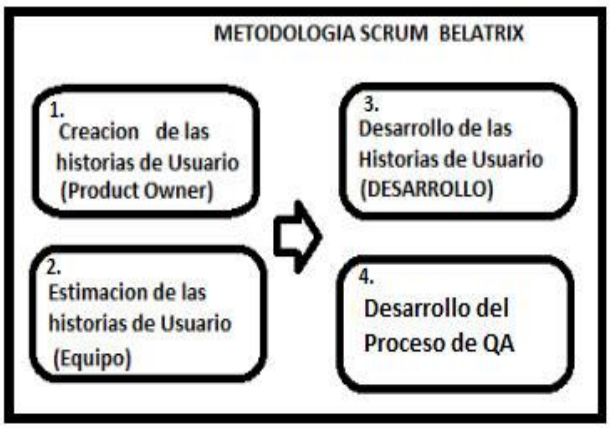
Mediante el uso de patrones conocidos, los componentes reutilizables se pueden construir en marcos. Los marcos de software proporcionan a los desarrolladores herramientas poderosas para desarrollar aplicaciones más flexibles y menos propensas a errores de una manera más efectiva.

CAPITULO II: PROBLEMÁTICA DE LA INVESTIGACIÓN

2.1 DESCRIPCIÓN DE LA REALIDAD PROBLEMÁTICA

La metodología Scrum en Belatrix se consta de varias actividades. Como se puede observar, en la figura 2encontramos estas actividades, las cuales se encuentran enumerados en orden de ejecución: primero se crean nuevas historias de usuario (nuevos requerimientos), se realiza la estimación, luego se inicia el desarrollo (programación en código) y se finaliza con la ejecución del proceso de QA, el cual busca garantizar la calidad del desarrollo.

Se debe tener en cuenta que la actividad 1 y 2 se ejecutan en paralelo asimismo como las actividades 3 y 4.



A continuación se detalla en qué consiste cada actividad de la metodología scrum:

1. Creación de las historias de Usuario (Product Owner)

Su objetivo es adecuar el modelo a emplear para la solución del Requerimiento, expresado en términos de objetos y relaciones entre ellos.

El desarrollo de las historias de Usuario es una abstracción resumida y precisa de lo que debe de hacer el Requerimiento deseado y no de la forma en que se

hará. Los elementos del modelo deben ser conceptos del dominio de aplicación y no conceptos informáticos tales como estructuras de datos.

Los pasos a seguir son:

De acuerdo a la prioridad de la tarea o Proyecto, el Jefe de Proyecto asigna al Product Owner y coordinan el tiempo para el análisis de los requerimientos y presentación de las historias de usuario (documento solución al Requerimiento).

Para una Historia de Usuario: los pasos a seguir del Product Owner son:

-Realiza el análisis según la solicitud de servicio, plantea la solución y completa los documentos: Historia de Usuario.

-Presenta solución al Usuario y la reformula hasta obtener su aprobación.

-Presenta solución al A l Equipo Scrum.

2. Estimación los Historias de usuario (Equipo)

Su objetivo es identificar cuánto esfuerzo tomará desarrollar y completar la historia de Usuario por el equipo se tiene que ealua Defiitio of Doe

Para ello se tomará los números de Fibonacci para puntuar la historia

Los pasos a seguir son:

Todo el equipo indica cuánto tomará completar la historia de Usuario desde su perspectiva de su Rol (Desarrollador, QA) en el que se justificara la puntuación Dada; los números estén dentro del rango: 1, 2, 3, 5, 8, 13, 21.

3. Desarrollo de las historias de Usuario (DESARROLLO)

Los desarrolladores empiezan con la creación de código para los servicios y la parte de la interfaz de usuario así como la creación de las pruebas unitarias.

4. Pruebas de las de las historias de Usuario (QA)

El equipo de QA se encarga de la creación de los casos de pruebas, ejecución de las pruebas exploratorias, ejecución de los casos de pruebas, registro de

incidencias. Según la figura 3, el procedimiento consiste primero en Realizar la creación de los casos de pruebas, mientras el equipo de desarrollo está realizando la programación, cuando el equipo desarrollo termina sus actividades, el equipo de QA empieza con la ejecución de las pruebas exploratorias y la ejecución de los casos de pruebas.

a. Creación de los casos de pruebas consiste en analizar las historias de usuarios y crear nuevos casos de pruebas en Test Link los cuales serán ejecutados cuando el desarrollo termine. Se toma en cuenta como caso de prueba a un conjunto de pasos redactados en un documento con una finalidad de validación, en este caso será el test link el repositorio de todos los casos de pruebas creados.

Pruebas de Regresion Sprint 1 # de Personas	900 casos de pruebas
Tester 1(160 casos)	20 horas
Tester 2(170 casos)	20 horas
Tester 3(170 casos)	20 horas
Tester 4(200 casos)	24 horas
Tester 5(200 casos)	24 horas

Pruebas de Regresion Sprint 3 # de Personas	962 casos de pruebas
Tester 1(192 casos)	24 horas
Tester 2(200 casos)	24 horas
Tester 3(170 casos)	20 horas
Tester 4(200 casos)	24 horas
Tester 5(200 casos)	24 horas

Pruebas de Regresion Sprint 4 # de Personas	1000 casos de pruebas
Tester 1(200 casos)	24 horas
Tester 2(200 casos)	24 horas
Tester 3(200 casos)	24 horas
Tester 4(200 casos)	24 horas
Tester 5(200 casos)	24 horas

Pruebas de Regresion Sprint 2 # de Personas	932 casos de pruebas
Tester 1(192 casos)	24 horas
Tester 2(170 casos)	20 horas
Tester 3(170 casos)	20 horas
Tester 4(200 casos)	24 horas
Tester 5(200 casos)	24 horas

b. Ejecución de las pruebas exploratorias consiste en validar la aplicación de manera ligera y superficial, sin ser meticulosos en las validaciones para identificar rápidamente incidencias, generalmente esta validación demora poco tiempo en comparación de la ejecución de los casos de pruebas, sin embargo la idea de las pruebas exploratorias es detectar rápidamente incidencias.

c. Ejecución de los casos de pruebas consiste en seguir los pasos que indican los casos en pruebas en la aplicación desarrollada, se realiza cuando el desarrollo de las historias de usuario ya está terminado.

d .Registro de la incidencia consiste en que si cuando se están ejecutando los casos de pruebas en la aplicación, si se encuentra alguna incidencia, se registrara en Jira, para que el equipo de desarrollo lo resuelva.

e. Pruebas de Regresión (QA) Las pruebas de regresión es crucial para la fase de desarrollo, eliminando los errores en producción y el desarrollo del producto, como se indica en la sección de background de AZAD [3].

La regresión de casos de pruebas se realiza después de que todos los casos de pruebas del sprint se hayan ejecutado satisfactoriamente, asimismo las pruebas exploratorias y que todas las incidencias se hayan cerrado, para así dar paso a la regresión el cual consiste en ejecutar todos los casos de pruebas creados anteriormente para garantizar que las funcionalidades anteriores sigan funcionando como se espera.

En las figuras se observa la ejecución de 900 casos de pruebas distribuidas en 5 testers (analistas de QA), también se puede visualizar la cantidad de casos de pruebas asignados.

Las pruebas de regresión tomaban aproximadamente 3 días(24 horas), ejecutadas por 5 personas.

El cuello de botella que encontramos es la cantidad de días y personas que nos toma la regresión después de finalizar cada sprint, como se puede observar

también es que mientras más sprints se desarrollan la cantidad de casos de pruebas aumenta y esto requerirá mayor cantidad de días y más personas.

El principal problema es la cantidad de tiempo que demora actualmente la regresión de los casos de pruebas manuales después de cada desarrollo.

2.2 FORMULACIÓN DEL PROBLEMA

2.2.1 PROBLEMA PRINCIPAL

¿Cuáles son los framework de automatización para el proceso de QA en un proyecto de desarrollo?

2.2.2 PROBLEMA SECUNDARIOS

- ¿Cuál es el tiempo que toma la regresión de los casos de pruebas?
- ¿Cuál es la intervención Humana en la ejecución de los casos de pruebas.?
- ¿Cuál es la actividad nueva denominada Automatización de casos de pruebas dentro del proceso de QA?

2.3 JUSTIFICACIÓN DE LA INVESTIGACIÓN

El presente documento brinda información la configuración del framework de automatización, Casos de pruebas Automatizados y un Manual de Usuario de cómo crear un test Automatizado Los casos de pruebas previamente deben estar definidos.

2.4 HIPÓTESIS Y VARIABLES DE LA INVESTIGACIÓN

2.4.1 HIPÓTESIS GENERAL

Existe una alta eficacia en la implementación del framework aplicado a los proceso dentro del ámbito del diseño de software en una consultora

2.5 VARIABLES

VARIABLE 1:

FRAMEWORK

DEFINICIÓN CONCEPTUAL DE LA VARIABLE 1

Un marco de software es una plataforma concreta o conceptual donde el código común con funcionalidad genérica puede ser selectivamente especializado o anulado por desarrolladores o usuarios. Los marcos toman la forma de bibliotecas, donde una interfaz de programa de aplicación (API) bien definida es reutilizable en cualquier lugar dentro del software en desarrollo.

CAPÍTULO III: OBJETIVOS DE LA INVESTIGACIÓN

3.1 OBJETIVO GENERAL

Determinar los framework de automatización para el proceso de QA en un proyecto de desarrollo.

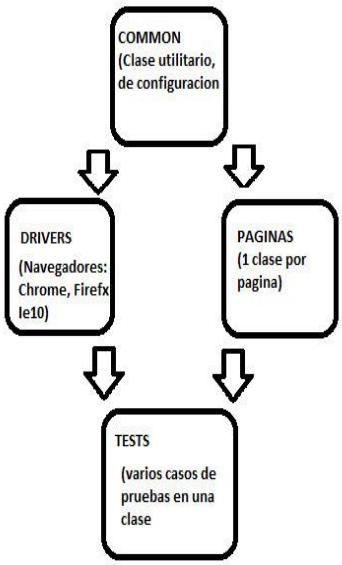
3.2 OBJETIVOS ESPECÍFICOS

1. Determinar es el tiempo que toma la regresión de los casos de pruebas.
2. Determinar la intervención Humana en la ejecución de los casos de pruebas.
3. Establecer la actividad nueva denominada Automatización de casos de pruebas dentro del proceso de QA
4. Crear el framework de automatización.

CAPÍTULO IV: MARCO TEÓRICO

DISEÑO DE LA ARQUITECTURA

La arquitectura se puede visualizar la figura siguiente, básicamente contiene folders donde estarán contenidos las clases organizados de una manera óptima, la carpeta COMMON contiene la clase Utilitario el cual en su mayoría posee métodos estáticos los cuales serán consumidos en todo el proyecto y clases de configuración, la carpeta DRIVERS, contiene las clases que configuran ciertos navegadores que se aperturan cuando el test se ejecuta, cada clase en la carpeta driver puede ser de tipo Chrome, ie, firefox, safari, opera, etc. la carpeta PÁGINAS contiene las clases que están relacionada a cada página de la aplicación, es decir la pagina Login posee su clase correspondiente, la página de bienvenida similarmente y es ahí donde se aloja los elementos Web. La Carpeta TESTS contiene la clase de los casos de pruebas divididos por funcionalidades, es decir todos los casos de pruebas de login irán en la clase Login Test.

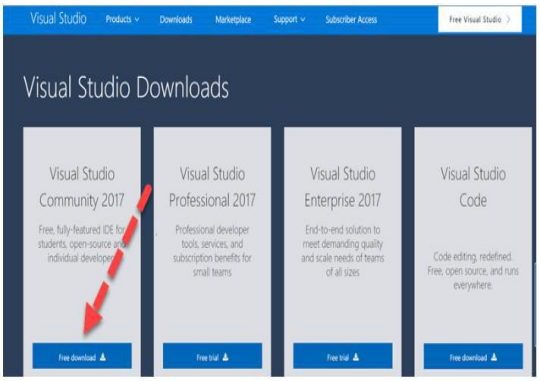


ELECCIÓN DE LA TECNOLOGÍA Y DE LAS LIBRERÍAS.

Para la poder construir el framework de automatización desde cero debemos identificar las herramientas tecnologías, teniendo en cuenta que actualmente muchas de ellas tienen mayor soporte técnico de otras, para esto se identificó las buenas prácticas que usan muchas compañías en el ámbito de la automatización de pruebas. Para esto se escogió la librería Selenium debido a que se integra fácilmente con TestNG, el cual me permite generar reportes y porque posee paralelismo en comparación a otras librerías. Selenium permite la conexión con los diferente browsers como Chrome, Firefox, Internet Explorer entre otros.

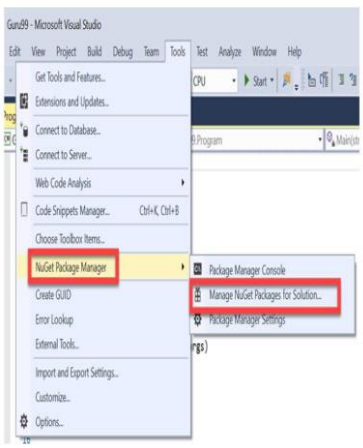
Se escogió el lenguaje C# el cual requiere que se instale el visual Studio Comunity porque está orientado a POO ,porque es Open Source y porque posee librerías como Linq, Bogus y fluent Assertion los cuales

son fácilmente añadidos al framework de automatización para realizar validaciones, crear data y realizar conexión a base de datos, como se puede visualizar en las figuras siguietes, se tiene una libre descarga para el uso personal sin pago alguno.

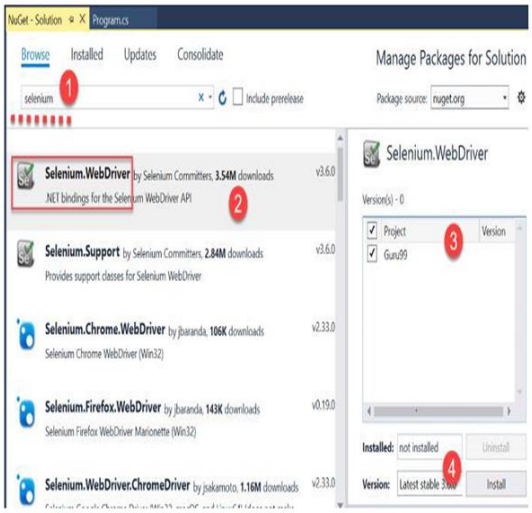


Además, se muestra el visual studio listo para añadir los nugets(las librerias que se utilizaran en la automatizacion) . Se crea un proyecto nuevo y se instala la herramienta Selenium añadiendo un Nuget .

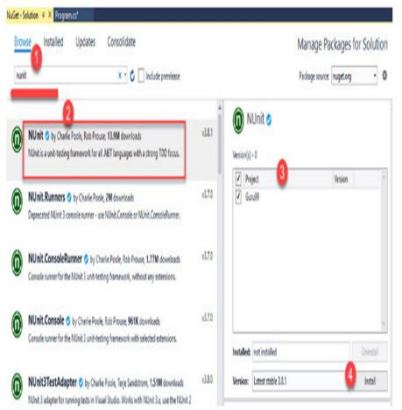
Se realiza una búsqueda de la librería Selenium Web Driver, el cual nos permite interactuar con Los diferentes browsers (Chrome, Safari, Internet Explorer, Firefox). en la *figura siguiente* se visualiza la busqueda realizada



con exito y se encontro la libreria Selenium.

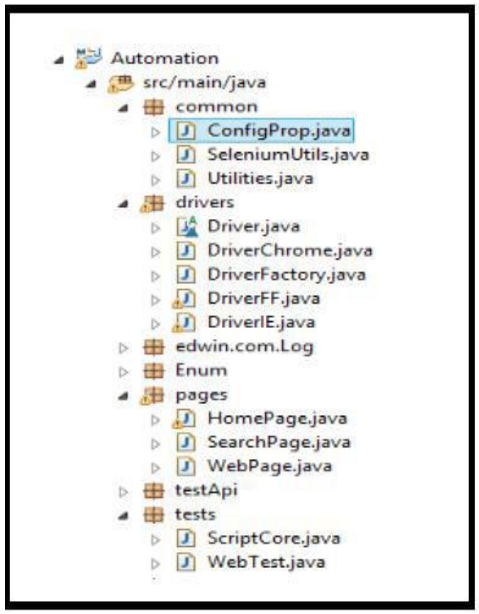


Se instala otra librería llamada Nunit el cual nos brindara ciertos métodos para validar los resultados esperados de nuestro test, como se puede visualizar en la figura siguiente obtenemos la búsqueda del Nunit, una libreria util para validar nuestros escenarios.

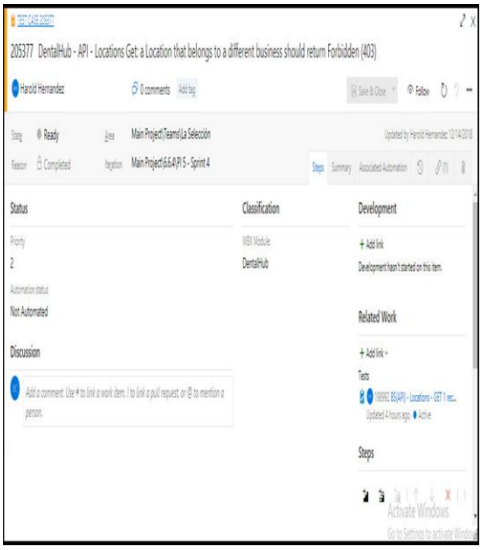


Además se puede apreciar las clases divididas dentro de las carpetas anteriormente señaladas. en la carpeta drivers se puede visualizar Drive

Chrome, Driver FF, Driver IE las cuales indican que solo se esta ejecutando las pruebas automatizadas para los navegadores, Chrome, Internet explorer y firefox. La idea es separar la arquitectura para tener más código mantenible, si en algún momento queremos añadir otro navegador, solo sería incluir una clase más y no modificar las anteriores clases.



Podemos visualizar un caso de prueba específica los pasos del caso de prueba, en el que se especifica el flujo completo de una validación. Se divide en pasos a seguir y un resultado esperado. Al seguir estos pasos en la aplicación se debería comportar como tal, sino es así entonces se crea una incidencia.



Se especifica los pasos del caso automatizado, y la navegación en la aplicación de acuerdo siempre a los pasos de casos de pruebas.

```
package tests;
import org.testng.annotations.Test;

public class ScriptCore extends WebTest{
    public ScriptCore(){
        super();
        // TODO Auto-generated constructor stub
    }

    @Test
    public void ValidateFirstResultsInHomePage(){
        home.goHomePage().getHeader().changePrincipalLanguage();
        home.getHeader().insertSearchText(SearchConstants.SHOES.toString()).clickButtonSearch();
        search.selectShoesBrand(BrandConstants.PUMA.toString()).selectShoesSize(SizeConstants.TEN.toString()).validateResultIs
        search.changePriceDescendant().displayFirstResult().SortFirstSevenResultNames();
        search.changePriceDescendant().displayFirstResult();
```

De la figura anterior se tiene que en cada sprint el equipo de QA formado por 5 personas toma 120 horas la ejecución de 932 casos de pruebas y la figura muestra un reporte donde la ejecución toma 109 minutos ejecutar los mismos tests, pero ya automatizados por lo que se puede decir que después de automatizar 932 test se pudo evaluar el tiempo de reducción que se ganó con la automatización fue el siguiente:

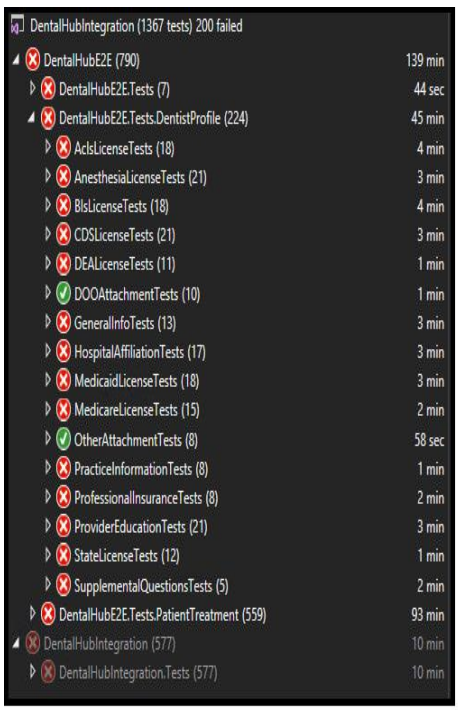
1 mes = 2 sprints	Duración en Horas/Hombres Sin la automatización	Duración en Horas/Hombres Con la automatización
3 meses de trabajo = 6 sprints	6 sprint x3 días x 5 personas x 8 Horas = 720 horas	6 Sprint x 2 horas= 12 horas
6 meses de trabajo = 12 Sprint	12 sprint x3 días x 5 personas x 8 Horas = 1440 horas	12 sprint x 2 horas= 24 horas

Se puede visualizar en el cuadro anterior que sin la automatización en 3 meses de trabajo se requirió 720 horas para ejecutar 932 test manualmente, esto debido a que en 3 meses de trabajo se tienen aproximadamente 6 sprints, y en cada término de 1 sprint se tiene que ejecutar la regresión que toma 3 días utilizando 5 recursos de QA los cuales laboran 8 horas diarias resultan 720 horas.

Y utilizando la automatización esto toma solamente 2 horas ejecutar 932 test por lo que 3 meses de trabajo sólo genera el uso de 12 horas. La reducción del tiempo es aproximadamente del 98% del tiempo total.

Regresion	# Casos de pruebas	Tiempo	Costo
Personas	930		
QA 1	185	3 dias (60 casos de pruebas ejecutados por dia)	24 horas-Hombre
QA 2	185	3 dias (60 casos de pruebas ejecutados por dia)	24 horas-Hombre
QA 3	185	3 dias (60 casos de pruebas ejecutados por dia)	24 horas-Hombre
QA 4	185	3 dias (60 casos de pruebas ejecutados por dia)	24 horas-Hombre
QA 5	185	3 dias (60 casos de pruebas ejecutados por dia)	24 horas-Hombre
			120 horas

Se muestra un reporte de la ejecución de los test automatizados, en donde se visualiza la ejecución de los casos de pruebas y el tiempo de ejecución es de 139 minutos en para los tests E2E y de 10 minutos para los de tipo Integration.



Selenium: Selenium se compone de múltiples herramientas de automatización de software, como Selenium IDE, Selenium RC (selenium 1.0), y Selenium webdriver (selenium 2.0). Selenium IDE es un entorno de desarrollo integrado para construir los guiones de prueba. Es un complemento de Firefox que le permite grabar, editar y depurar los casos de prueba de selenio. Graba todo las acciones realizadas por el usuario final y generar los scripts de prueba. El control remoto de selenio (RC) fue selenio principal proyecto desde hace mucho tiempo. Selenium RC es más lento que el webdriver de selenium porque usa el programa de script java llamado núcleo de selenio. Selenium RC requiere iniciar el servidor antes de ejecutar los scripts de prueba, como se menciona en el artículo SATISH [2].

TestNG: TestNG se compone de múltiples herramientas de automatización de

software, como TestNG Selenium IDE, TestNG Selenium RC (selenium 1.0), y Selenium webdriver (selenium 2.0). Selenium IDE es un entorno de desarrollo integrado para construir los guiones de prueba. Es un complemento de Firefox que le permite grabar, editar y depurar los casos de prueba de selenio. Graba todo las acciones realizadas por el usuario final y generar los scripts de prueba. El control remoto de selenio (RC) fue selenio principal proyecto desde hace mucho tiempo.

Selenium RC es más lento que el webdriver de selenium porque usa el programa de script java llamado núcleo de selenio. Selenium RC requiere iniciar el servidor antes de ejecutar los scripts de prueba, como se menciona en el artículo SATISH [2].

SeleniumIDE: Selenium IDE se compone de múltiples herramientas de automatización de software, como Selenium IDE, Selenium RC (selenium 1.0),

y Selenium webdriver (selenium 2.0). Selenium IDE es un entorno de desarrollo integrado para construir los guiones de prueba. Es un complemento de Firefox que le permite grabar, editar y depurar los casos de prueba de selenio. Graba todo las acciones realizadas por el usuario final y generar los scripts de prueba. El control remoto de selenio (RC) fue selenio principal proyecto desde hace mucho tiempo.

Selenium RC es más lento que el webdriver de selenium porque usa el programa de script java llamado núcleo de selenio. Selenium RC requiere iniciar el servidor antes de ejecutar los scripts de prueba, como se menciona en el artículo SATISH [2].

IP flotante: En términos generales, una IP flotante es una dirección IP pública enrutable que no se asigna a ninguna instancia de manera automática.

En lugar de eso, el propietario de un proyecto la pone a disposición de una o varias instancias de manera temporal. La instancia correspondiente dispone así tanto de una IP estática que le ha sido concedida automáticamente para la comunicación en un ámbito de red privado no enrutable, como también de una IP flotante asignada manualmente. Esto hace accesibles los servicios de una entidad a usuarios fuera de una nube.

En aquellas situaciones configuradas para que tenga lugar la outaió po eo,

diha dieió IP flota (inglés: floating = flotar) o se desliza con dinamismo hacia otra unidad activa en la red con el objetivo de que dicha unidad se haga cargo sin demoras de, por ejemplo, la función de una instancia que ya no está activa y de que, en su lugar, responda a las peticiones entrantes.

CAPÍTULO V: MÉTODOS O PROCEDIMIENTOS

5.1 TIPO Y NIVEL DE INVESTIGACIÓN

TIPO DE INVESTIGACIÓN

La presente investigación es de carácter analítico.

NIVEL DE INVESTIGACIÓN

La presente investigación es de nivel experimental-

5.2 MÉTODO Y DISEÑO DE LA INVESTIGACIÓN

MÉTODO DE LA INVESTIGACIÓN

Cuantitativo: porque utiliza el cuestionario donde es valorado con numeraciones para establecer resultados.

CAPÍTULO VI: RESULTADOS

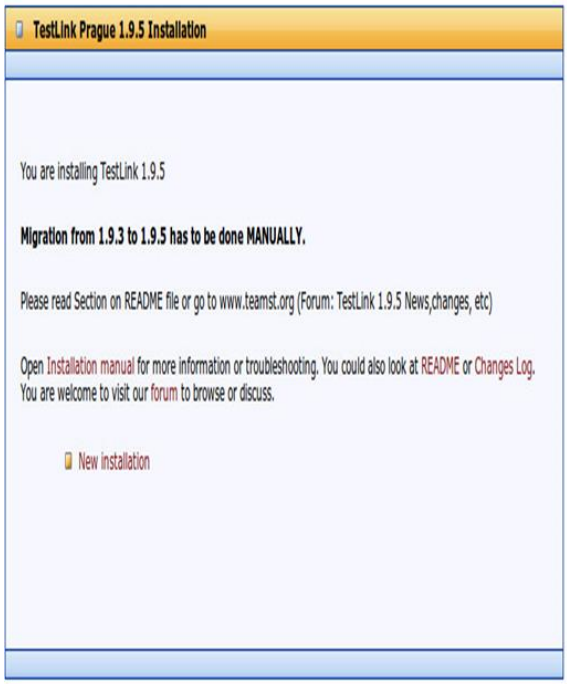
INSTALAR TESTLINK

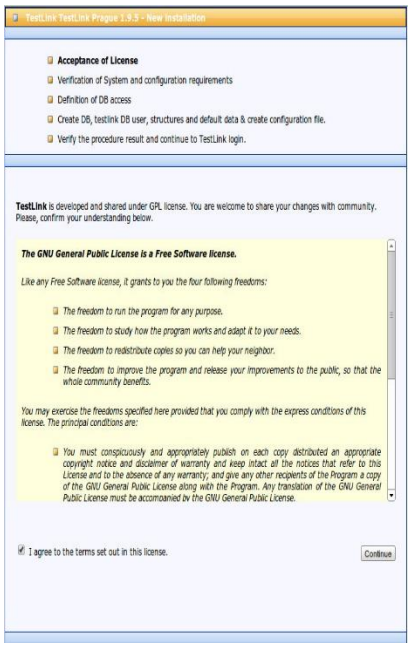
El desarrollo de software es importante en la industria de fabricación y procesos. Hoy en día el software y las computadoras son a menudo una parte integrada de los procesos industriales. Sin embargo, todavía hay espacio para mejorar en diferentes partes del proceso de producción. Una ilustración de esto es la iniciativa del gobierno alemán "Industria 4.0" [4].

El objetivo de la iniciativa, que se lanzó por primera vez en 2011, es informatizar la industria manufacturera de manera similar a como la idea detrás de "internet de las cosas" [5] conecta elementos cotidianos a internet. Conectar partes de un proceso de fabricación a Internet o una intranet facilita el seguimiento del uso de los recursos y el seguimiento de los procesos logísticos. Otra ventaja de la computarización adicional de los procesos de fabricación es que puede contribuir a un entorno de trabajo mejor y más eficiente. Los controles de diferentes máquinas pueden, por ejemplo, integrarse y operarse desde una sala de control central.

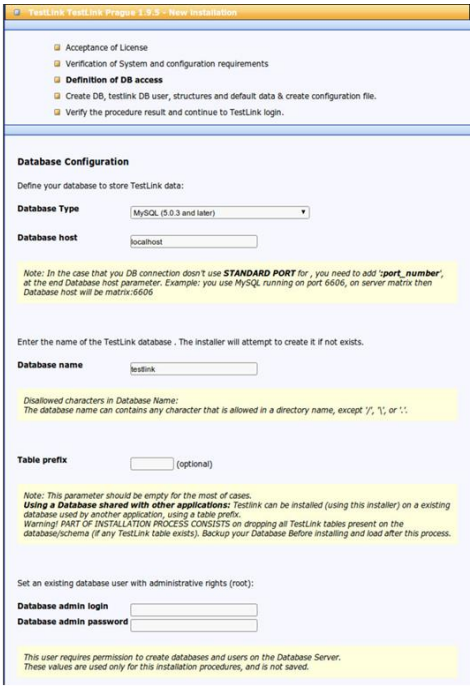
Otra parte de un proceso industrial que puede mejorarse mediante la informatización y la automatización es la etapa de prueba. La mayoría de los productos creados en la industria de fabricación y procesos generalmente se prueban. La prueba puede realizarse, por ejemplo, midiendo la resistencia a la tracción de un material, el color de una tela o el tinte de un parabrisas para un automóvil. El propósito de la prueba es verificar que el producto cumpla con sus especificaciones (por ejemplo, un pedazo de cartón debe poder soportar una cierta cantidad de fuerza antes de que se rasgue). A menudo, cuando se prueba un producto o material, ya sea de madera contrachapada o tela o cartón, se utiliza una variedad de equipos para probar diferentes

características. Un desafío es conectar los diferentes instrumentos de prueba a la red de la planta de producción.

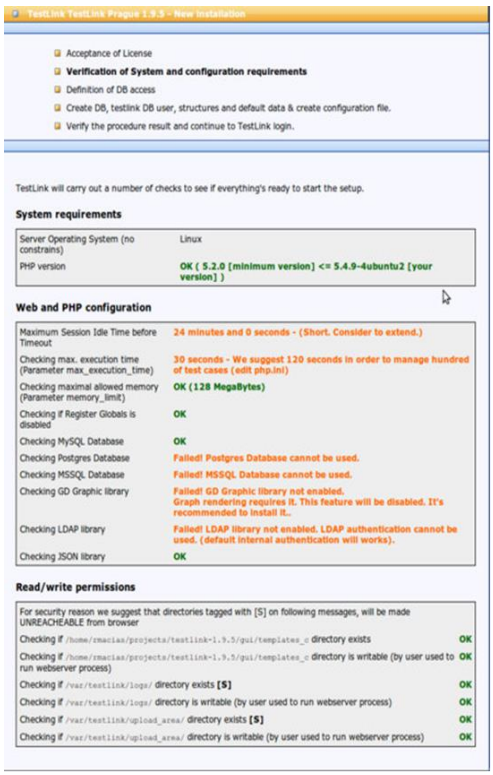




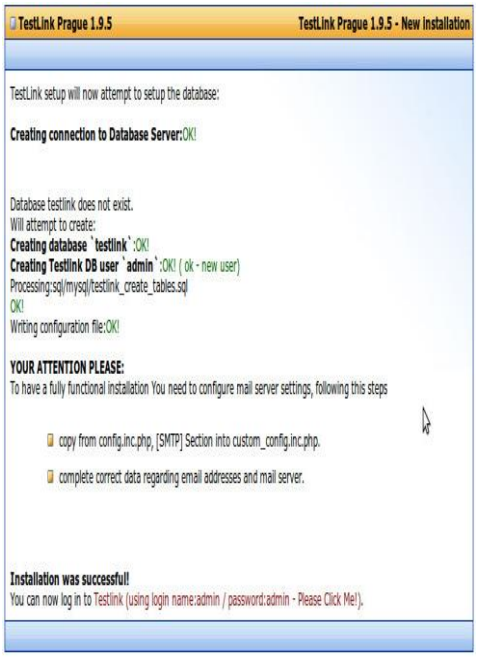
Se debe ingresar los datos por complete.



Además, se debe asegurar los requisitos.



Después de esto la instalación estara complete.



Luego se deberá configurar el software

oconfig.inc.php: contiene configuración principal.

oconfig_db.inc.php: contiene parametros de configuración de acceso a la bd.

ocustom_config.inc.php: sirve para modificar los valores por defecto de los parametros de config.inc.php, esto nos facilita la

modificacion

Test Project y minimo dos usuarios:

u usuaio “eio Teste e agado de ellea esos Test Case aíos o el escenario de prueba. (Steps)

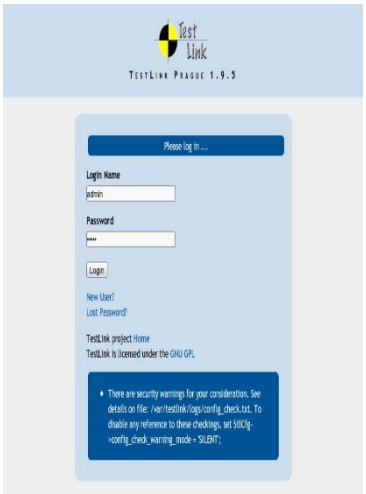
Una vez creados estos Steps, podemos linkear los Test Cases a un Test Plan y a un Build creados anteriormente.

Podemos crear unas palabras clave “Keyword” para tener un filtro de test.

Una vez creado todo esto probaremos los test, y reflejaremos los resultados en esta aplicación.

7. Uso de la aplicación

Nos conectaremos con el usuario admin que hemos creado:

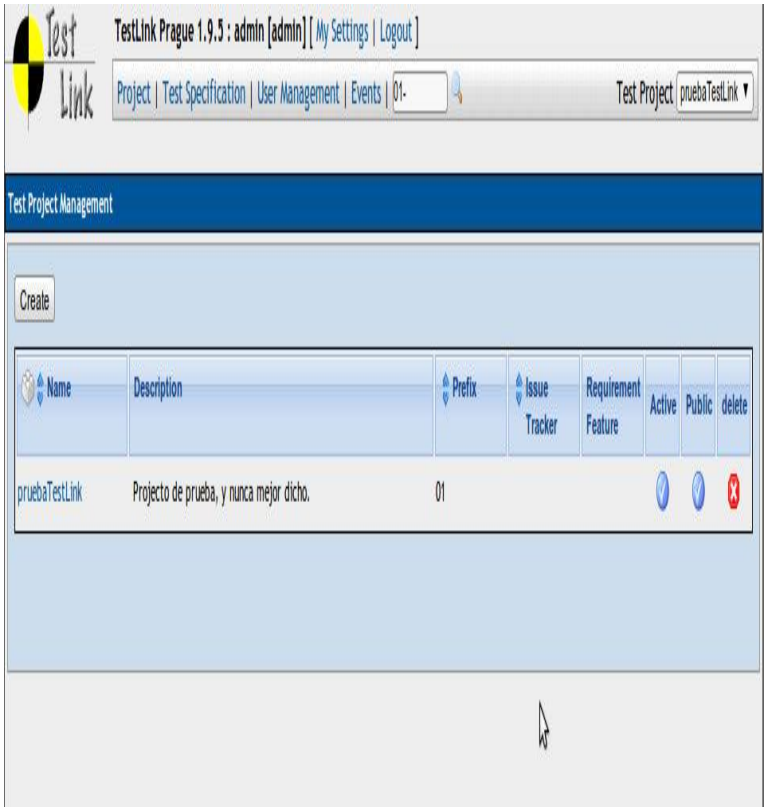


– Test project :

Vamos a crear un projecto para test (necesitamos derechos de administrador).

Se nos presenta la siguiente guia:

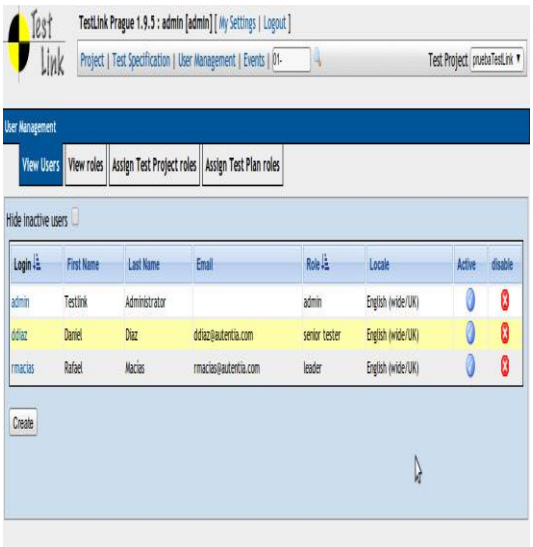
Lo rellenamos y al crearlo llegamos a una tabla con nuestros projectos



Despues crearemos dos usuarios:

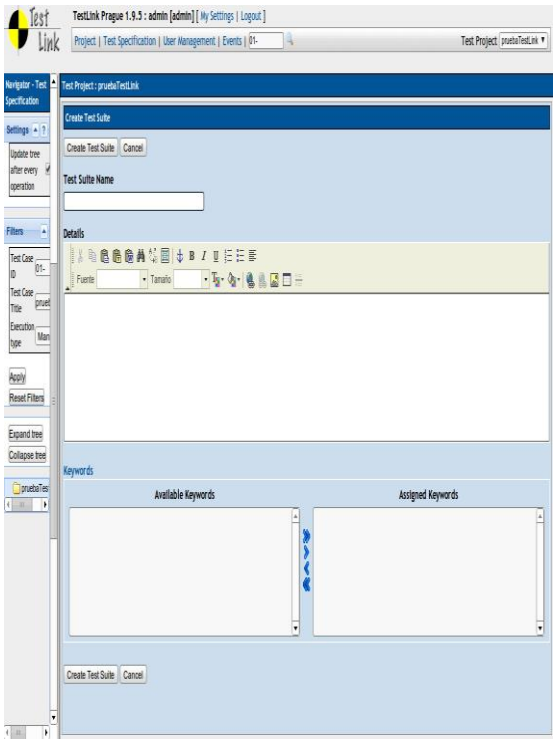
●Leader user

●Senior Tester

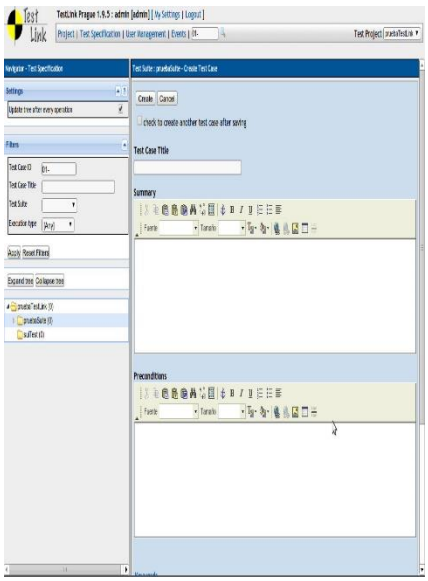


El leader del test project declara los requisitos del Software y con estos crea unos casos de test que incluirá en unos suites.

En esta aplicación crearemos primero un Suite de pruebas:



Y a continuación de este estableceremos los casos de test.



En sumary indicaremos la especificacion que queremos de nuestro software, por ejemplo:

–“Posibilidad de logear en la aplicacion”

En preconditions especificaremos las precondiciones para esa summary:

–“Tener iniciada la aplicación”

Otro ejemplo más claro seria:

–Summary: “Quiero que se cargue un menú de inicio”

–Preconditions: “El usuario se haya conectado”

De momento dejamos los steps sin definir.

Ahora crearemos un Test Plan, al que ligaremos todos los test cases.

En el panel de “Test Plan” hacemos click en build y nos dice que tenemos que crear uno, sera nuestro control de versión de cada test.

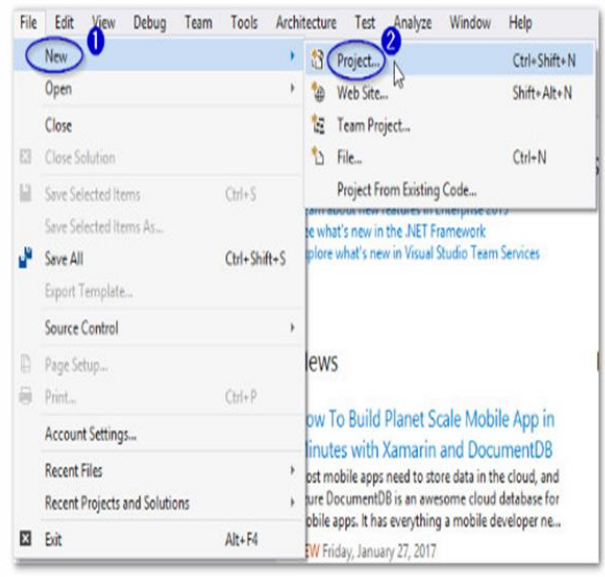
Es decir para el test “Probar un caso de test” creamos una build “Probar un caso de test

INSTALAR SELENIUM CON C#

Para mayor detalle puede encontrar el tuorial completo en la siguiente pagina web: http://learn-automation.com/install-selenium-webdriver-with-c/

Crear un Nuevo proyecto en Visual Studio:

Paso 1) En el File Menu, Click New > Project



Step 2) En la siguiente pagina ,

1.Seleccionar the option 'Visual C#'

2.Click on Console App (.Net Framework)

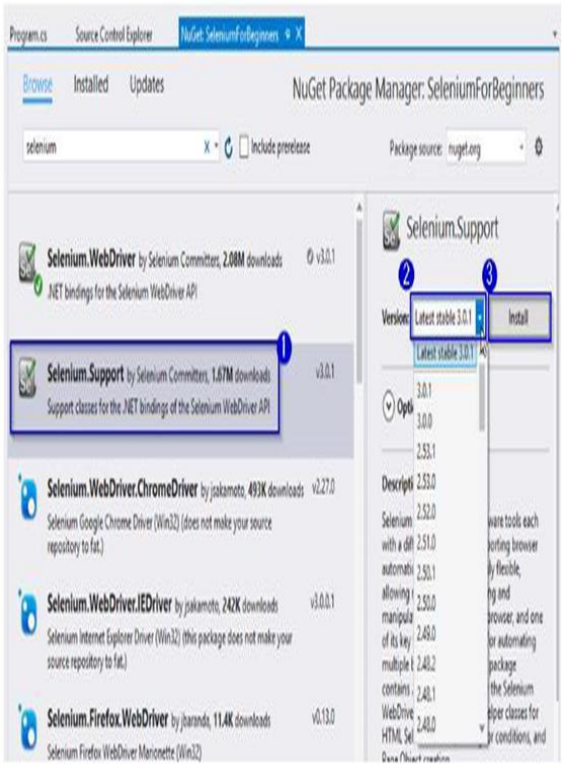
3.Ingresar nombre como "Guru99"

4.Click OK

Step 3) se muestra creado correctamente

Paso 4) seleccionar estas opciones Navigate to Tools -> NuGet Package Manager ->

Manage NuGet Packages for Solution



Las pruebas de software han demostrado su valor para el desarrollo de software cada vez más en el último década. Con el reconocimiento de los beneficios de las pruebas de software, varias pruebas de software Las herramientas de gestión (TMT) han surgido en el mercado. Aunque existen diferentes enfoques, no hay un método para una evaluación sistemática de TMT.

Este es un problema porque, según nuestro conocimiento, evaluar TMT es más bien una tarea subjetiva, Depende en gran medida de las opiniones de los evaluadores en lugar de basarse en el enfoque objetivo. El mismo problema se aplica cuando se solicita a los gerentes de pruebas que evalúen si sus utilizado

TMT cumple con las expectativas de la empresa. Para comprender la importancia y la necesidad de la evaluación de TMT, realizamos una Estudio de literatura sobre procesos de prueba de software y estudios de mercado de TMT existentes. Entonces mapeamos juntos las actividades de prueba identificadas y artefactos de prueba. Los resultados nos ayudan a formular y diseñar un cuestionario en línea y realizar una encuesta TMT dentro de las empresas de TI de Estonia. Sobre la base de los resultados de la encuesta, se crea un marco para evaluar el software TMT.

Un tal El marco podría potencialmente ayudar a las empresas a medir la idoneidad de TMT para las empresas. Objetivos y disminuir la subjetividad de la evaluación TMT. El marco también proporciona prueba y los gerentes de proyecto entienden si sus TMT actuales cumplen con los requisitos de la compañía. esperanzas de heredar. Validamos el marco con un estudio de caso realizado entre Calidad. Especialistas en aseguramiento para recopilar información sobre la usabilidad del marco.

Las posibilidades para futuros trabajos basados en esta tesis son numerosas. El marco se puede hacer. en una aplicación para facilidad de uso y distribución más amplia. Expandiendo la investigación a otros Los países europeos son otra opción viable. También ampliando los requisitos de TMT basados en Se pueden tener en cuenta las nuevas tendencias en las pruebas. En conclusión, creemos esta tesis.

Durante la evolución de la ingeniería de software ha habido muchas definiciones de lo que la prueba es. A menudo se percibe como una "bala mágica" (Myrvold, 2011) que resolverá el problema de encontrar errores

después de que el producto ha sido entregado al cliente. Sin embargo, las pruebas no pueden establecer que un producto funciona correctamente en todas las condiciones, pero solo puede establecer que no funciona correctamente en condiciones específicas (Kaner et al, 1999). El objetivo de las pruebas es ejecutar un sistema intensivo de software o partes del sistema de forma controlada. ambiente y bajo circunstancias controladas para detectar desviaciones de la especificación y para comprobar si el sistema cumple los criterios de aceptación definidos (Pohl, 2010).

Por esta definición, aborda solo la ejecución de pruebas y no le preocupan otros Ciclo de vida del software. Otra definición es la siguiente: las pruebas de software son un elemento crítico de garantía de calidad del software (SQA) que tiene como objetivo determinar la calidad del sistema y su Modelos relacionados (Keyes, 2003). Aquí la calidad del sistema puede significar diferentes cosas para diferentes partes interesadas. Por ejemplo, para el ingeniero de software, la calidad representa el sistema correspondencia a los requisitos; mientras que para el usuario final también significa la facilidad de uso del sistema. Por lo tanto, las pruebas de software deben cubrir las expectativas internas y externas o en En otras palabras, es parte de la garantía de calidad del software. SQA es un enfoque formal para el software desarrollo, pruebas de regresión automatizadas, gestión de la configuración, control de versiones, creación de perfiles y liberar el control con el objetivo de cero defectos (Britannica, 2003).

En las pruebas de software, la terminología puede variar según la certificación (es decir, internacional Junta de Calificación de Pruebas de Software (ISTQB), Instituto de Aseguramiento de la Calidad (QAI), Instituto Internacional para

Pruebas de Software (IIST)) utilizado en la organización. Tesis actual se refiere a la terminología utilizada en la certificación ISTQB cuando corresponda.

El ciclo de vida de las pruebas de software es parte del Ciclo de vida del desarrollo de software (SDLC). Eso define un conjunto de etapas que describen qué actividades de prueba realizar y cuándo realizarlas. Estas etapas son planificación, análisis, diseño, construcción, pruebas, pruebas finales y Implementación, post implementación (Keyes, 2003). Las pruebas se pueden dividir en Pruebas funcionales y no funcionales. Al igual que otros procesos, SQA puede basarse en diferentes modelos de prueba de software que se describen.

CAPÍTULO VII: CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

El framework de automatización fue adecuadamente implementado, y sirvió para apoyar al área de QA en la mejora del tiempo de ejecución de los casos de pruebas a partir de inicios del 2018.

•Se redujo el tiempo de la ejecución de los casos de pruebas, de 3 días a 2 horas

•Se evitó la Intervención humana en la ejecución

•Se añadió una actividad al proceso de QA.

•Se instaló el ambiente de automatización al equipo del área de QA.

RECOMENDACIONES

Replicar el modelo de implementación de framework de automatización para otros proyectos donde aún trabajan con ejecución manual y que como trabajos futuros, se puede identificar como implementar la ejecución de los test automatizados contra servidores en la Nube usando Browserstack.

BIBLIOGRAFIA

[1] LARS-OLA D. , LARS L. ; Results from introducing component-level test automation and Test-Driven Development, Ronneby, Sweden

[2] SATISH G. , RAHUL J. ; Analysis and Design of Selenium WebDriver Automation Testing Framework, Pune , India

[3] AZAD M. , SIEVERS J. ; n-Tiered Test Automation Architecture for Agile Software, California , USA

[4] Rouse, M. (2005) Framework. http://whatis.techtarget.com http://whatis.techtarget.com/definition/framework.

[5] Gamma, H.J. (1994) Design Patterns: Elements of Reusable Object Oriented Software. Addison-Wesley, Sydney, Zurich, Urbana, Hawthorne.

[6] Patterns, D. (2014) Design Patterns. http://www.oodesign.com/

[7] Wikimedia (2013) Introduction to Software Engineering.

[8] Sanna, E.A. (2013) 15 Top Factors that Impact Application Performance. http://www.apmdigest.com