

**UNIVERSIDAD CENTRAL DEL ECUADOR**

**FACULTAD DE INGENIERÍA CIENCIAS FÍSICAS Y MATEMÁTICA**

**CARRERA DE INGENIERÍA INFORMÁTICA**

**TÍTULO DEL TRABAJO**

PROYECTO INVESTIGATIVO DE INTEGRACION CONTINUA EN JAVA CON METODOLOGIAS DE DESARROLLO TDD.

**BYRON CABEZAS**

**JAIME ALCOCER**

Quito, 1 de Junio del 2021

**ACEPTACIÓN DEL TUTOR**

Por la presente dejo constancia que he leído el plan del trabajo de titulación, modalidad………….………..…………………….…… , cuyo título es PROYECTO INVESTIGATIVO DE INTEGRACION CONTINUA EN JAVA CON METODOLOGIAS DE DESARROLLO TDD.

resentado por………………………………….…………………………………………………………………………….

y en tal virtud acepto asesorar al estudiante en calidad de Tutor, durante el desarrollo de mismo hasta la elaboración del informe final, presentación y evaluación.

Dado en la ciudad de Quito, a los 1 dias del mes de junio

De 2021

NOMBRES Y APELLIDOS

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

FIRMA

cc…………………………………………………..

**PLAN DEL TRABAJO DE TITULACIÓN**

**CONTENIDO**

pág

|  |  |
| --- | --- |
| 1. Datos generales…………………………………………………………………..….. |  |
| 1. Análisis del problema…………………………………………………………….. |  |
| 2.1 Antecedentes…………………………………………………………………… |  |
| 2,2, Descripción del problema………………………………………………… |  |
| 1. Investigación bibliográfica……………………………………………………… |  |
| 3.1 Estado actual del conocimiento.………………………………………… |  |
| 3.2 Bibliografía comentada…………………………………………………….. |  |
| 1. Hipótesis/Justificación……………………………………………………………. |  |
| 1. Objetivos………………………………………………………………………………… |  |
| 5.1. Objetivo general……………………………………………………………….. |  |
| * 1. Objetivos específicos………………………………………………………….  1. Alcance…………………………………………………………………………………… 2. Limitaciones…………………………………………………………………………… |  |
| 1. Metodología experimental….…………………………………………………… |  |
| 1. Cronograma de actividades…………………………………………………….. |  |
| 1. Contenido de la memoria………………………………………………………… |  |
| 1. Presupuesto……………………………………………………………………………. |  |
| 1. Anexos……………………………………………………………………………………. |  |

**PLAN DEL TRABAJO DE TITULACIÓN**

1. **DATOS GENERALES**

|  |
| --- |
| **TÍTULO DEL TRABAJO:**  PROYECTO INVESTIGATIVO DE INTEGRACION CONTINUA EN JAVA CON METODOLOGIAS DE DESARROLLO TDD. |
| **MODALIDAD DEL TRABAJO DE TITULACIÓN:**   1. **PROYECTOS DE INVESTIGACIÓN** 2. **PROYECTO INTEGRADOR** |
| **AUTOR (ES):**  **Byron Cabezas**  **Jaime Alcocer** |
| **LÍNEA DE INVESTIGACIÓN (SOLO PARA LA MODALIDAD 1 ):** |
| **POSIBLE TUTOR:** |
| **FECHA DE PRESENTACIÓN:** |
| **LUGAR DE EJECUCIÓN:** |

1. **ANÁLISIS DEL PROBLEMA**

**2.1 ANTECEDENTES**

**En 1996, CI fue adoptada por los creadores de la Metodología de Programación Extrema, Kent Beck y Ron Jeffries . La integración continua se ha convertido en uno de los doce principios clave de la programación actual.**

**Actualmente la integración continua es practicada por organizaciones de una amplia gama de industrias. En 2018, un gran proveedor de la nube realizó una encuesta entre especialistas en TI de empresas de los sectores de servicios, educación y finanzas. De los seis mil encuestados, el 58% dijo que usa herramientas y principios de CI en su trabajo.**

**Entre las diferentes metodologías de desarrollo se encuentran los métodos agiles, y dentro de esta nos centramos en la metodología de desarrollo guiado por pruebas (TDD) donde la mayor parte del esfuerzo se concentra en realizar pruebas unitarias, esta metodología tiene un simple planteamiento donde se conciben las pruebas unitarias, luego escribir el código y seguir con una refactorización cumpliendo su objetivo de mejorar la producción e código.**

**2.2 PLANTEAMIENTO DEL PROBLEMA**

**Actualmente los softwares juegan un papel de vital importancia en las empresas, estos proyectos de desarrollo de software han presentado en los últimos años una serie de deficiencias que no permiten lograr su objetivo y que el desarrollo de software se realice en el menor tiempo posible y con alta calidad. Por lo tanto, se requiere de un adecuado desarrollo de software que permita responder al dinamismo del negocio y de la organización. Una de las técnicas del desarrollo de software que ha aparecido hace pocos años es la integración continua, que detallaremos en este documento.**

**La investigación a desarrollarse busca integrar la metodología TDD que básicamente implementa pruebas antes de desarrollo de código de esta manera se garantiza un código funcional y sobretodo independiente, además de que cada cambio que se realice no modifique otros módulos del desarrollo, adicional a esto se involucra la integración continua en el desarrollo de software de una organización, aplicando estas metodologías el área de TI se garantiza como muy robusto y factible para desarrollarse a lo largo de los años con nuevas tecnologías.**

Como se menciona más adelante en la bibliografía encontrada está demostrado que esta metodología está diseñada para aumentar la productividad de los programadores y en consecuencia reducir el número de bugs calendario y costos, sin embargo no se menciona el problema de la integración continua y como es la interacción entre diferentes participantes que problemas han acarreado , pues bien el desarrollo eficiente es importante pero el enfoque individual que se le da a esta metodología hace que la integración sea un pilar fundamental a analizar.

1. **INVESTIGACIÓN BIBLIOGRÁFICA**

**3.1 ESTADO ACTUAL DEL CONOCIMIENTO**

**Actualmente en los equipos de desarrollo se instala los conocidos hooks con comprobaciones básicas durante los mensajes de commit e incluso antes de hacer push a una rama o a máster; en general, sin embargo, se usa hooks en los servidores que ejecutarán los pulls sobre el mismo.**

**A diferencia de los hooks locales, que se ejecutan en el mismo entorno de desarrollo que el que está usando quien lo desarrolle, para los remotos habrá que crear una configuración para que ejecute esos tests y nos diga cuáles han pasado o cuales no. Estas máquinas más adelante se combinan con las de despliegue continuo, no permitiendo el mismo si algún test no ha pasado.**

**En general, la integración continua se hace en la nube, por el simple hecho de que los equipos de desarrollo están distribuidos y también los repositorios suelen ser servicios externos a los ordenadores de la empresa; lo que no quiere decir que se haga siempre en un servicio cloud contratado, sino porque se suele hacer en máquinas dedicadas específicamente para ello; es más general, sin embargo que una máquina virtual se descarga los ficheros, ejecuta los tests y crea un informe, enviando también un correo al autor indicándole el resultado**

**Un sistema bastante popular de integración continua es Jenkins. Jenkins se lo puede usar en la nube o instalarla en un ordenador propio. Sin embargo, está enfocado sobre todo a Java (y no hay un servicio gratuito que se pueda ejecutar).**

**Actualmente para trabajar con un sistema de CI hay que hacerlo en varios pasos**

**Primeramente, se deber conectar con GitHub por lo que se puede usar directamente el usuario propio. A través de un proceso de autorización, acceder al contenido e incluso informar del resultado de los tests.**

**Activar el repositorio en el que se vaya a aplicar la integración continua. Travis permite hacerlo directamente desde tu configuración; en otros se dan de alta desde la web de GitHub; también en algunos casos todos los repositorios estarán autorizados con sólo autorizar el usuario. Por supuesto, en el caso de GitHub Actions y GitLab pipelines no hace falta llevar a cabo este paso.**

**Crear un fichero de configuración con la configuración necesaria para ejecutar estos tests y añadirlo al repositorio. En sistemas de CI integrados en un repositorio, como GitHub actions, este paso es el único necesario.**

International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 18 (2017) pp. 7705-7716 © Research India Publications. http://www.ripublication.com

[**http://www.ripublication.com/ijaer17/ijaerv12n18\_81.pdf**](http://www.ripublication.com/ijaer17/ijaerv12n18_81.pdf)

Comparing Test-Driven Development and Pair Programming to Improve the Learning of Programming Languages

**https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1042&context=jmwais**

**3.2 REVISIÓN BIBLIOGRÁFICA COMENTADA**

La integración continua es de mucha importancia en el proceso de desarrollo de software. Automatización de las tareas integradas es de alta prioridad ya que estas tareas se realizan para ejecutarse periódicamente.

La construcción estable se canaliza hacia la regresión, esto es para asegurar que los cambios recientes no afecten a las características existentes. Jenkins es una mejor solución para la integración continua en software desarrollo.

Automation in Testing with Jenkins for Software Development Shridhar Prabhu1 , Manoj Naik2 , Firdosh A D3 , Sohan S A4 , Neeta B Malvi5, Journal of University of Shanghai for Science and Technology ISSN: 1007-6735

<https://jusst.org/wp-content/uploads/2021/06/Automation-in-Testing-with-Jenkins-for-Software-Development.pdf>

Jenkins es una plataforma de automatización de código abierto ampliamente

utilizado para construir e implementar software. Es altamente extensible

a través de complementos, proporciona una GUI conveniente para crear y editar

procesos de compilación, y actualmente se utiliza en varios sitios de HPC como un

herramienta de automatización de uso general

<https://doi.org/10.1145/3219104.3219147>

<https://dl.acm.org/doi/abs/10.1145/3219104.3219147>

1. **HIPÓTESIS Y/O JUSTIFICACIÓN**

|  |
| --- |
| Hipótesis |
| De qué manera influye la incorporación de la integración continua en el desarrollo de software. |
| Las pruebas constantes de la integración continua influyen en el desarrollo de software. |
| La retroalimentación constante de la integración continua influye en el desarrollo de software |

**La integración continua es una práctica de desarrollo de software donde los miembros de un equipo integran su trabajo con frecuencia, por lo general cada persona integra por lo menos una vez al día – que conduce a múltiples integraciones por día. Cada integración es verificada por una construcción automatizada (incluidas pruebas) para detectar errores de integración lo más rápido posible.**

**4.1 HIPÓTESIS Y JUSTIFICACIÓN (MODALIDAD 1)**

**Justificación:**  
En el desarrollo de software, es importante no solo producir la mayor cantidad de código funcional (metodología TDD) sino también integrarlo, pues es aquí donde se invierte la mayor parte del tiempo del desarrollo de un proyecto, por lo cual es fundamental contar con una herramienta que nos permita la integración continua, donde intervienen otros procesos como automatización de ciertos procesos, desarrollo de un software con mayor calidad, reducción de errores, aumentar la productividad de los desarrolladores, traduciéndose en disminución de tiempo y por lo tanto costos de producción.

**Hipótesis:**

1. **OBJETIVOS**

**5.1 OBJETIVO GENERAL**

Integrar las metodologías de desarrollo ágil tales como la integración continua y desarrollo guiado por pruebas (TDD), para mejorar la producción de software funcional y su posterior integración en conjunto.

**5.2 OBJETIVOS ESPECÍFICOS**

**-**definir métricas que nos permitan identificar el avance en cada cierto punto de la codificación e integración las cuales serán índices de calidad del código.

- realizar monitorización continua mediante las métricas que previamente serán definidas

- automatizar las pruebas, asegurando que para los determinados casos pueda realmente recoger un error

- implementar plugins en el software Jenkins que faciliten la integración de las metodologías así como el versionamiento

- analizar de la arquitectura que se implementara

1. **ALCANCE**

El presente proyecto pretende integrar metodologías de desarrollo agiles como la integración continua y la TDD, para que se aumente la productividad de los desarrolladores, código de mayor calidad, reducción de calendario y costos todos estos factores nos aseguraran que nuestro software que se desarrolle tiene calidad, esto con el apoyo de métricas definidas por nuestro equipo noes permitirán medir dicha calidad.

La herramienta denominada Jenkins nos permitirá la integración continua validar y versionar nuestro código para posteriormente se pueda refactorizar el código, validarlo y finalmente pasarlo a despliegue

1. **LIMITACIONES**

* El problema a analizar se centrará en el desarrollo de una aplicación, mas no en su documentación intensiva.
* El proyecto será ejecutado en un ambiente controlado donde nosotros especificaremos las métricas y herramientas a utilizar
* Experiencia limitada nos hace ver una implementación e interacción entre participantes de diferentes capacidades y experiencia
* Las pruebas manuales serán limitadas, pues nos concentraremos en hacer las pruebas automaticamente

1. **METODOLOGÍA EXPERIMENTAL DEL TRABAJO DE TITULACIÓN**

La integración continua (CI) es la práctica de automatizar la integración de los cambios de código de varios contribuidores en un único proyecto de software. Es una de las principales prácticas recomendadas de DevOps, que permite a los desarrolladores fusionar con frecuencia los cambios de código en un repositorio central donde luego se ejecutan las compilaciones y pruebas. Las herramientas automatizadas sirven para verificar que el nuevo código es correcto antes de la integración.

Un sistema de control de versiones del código fuente es el punto clave del proceso de CI. El sistema de control de versiones también se complementa con otras comprobaciones como las pruebas automatizadas de calidad del código, las herramientas de revisión de estilo de sintaxis y mucho más.

La integración continua se utiliza generalmente junto con un flujo de trabajo de desarrollo de software de metodología ágil. Una organización compilará una lista de tareas que constituyan una hoja de ruta de productos. A continuación, estas tareas se distribuyen entre los miembros del equipo de ingeniería de software para la entrega. Al utilizar la CI, estas tareas de desarrollo de software pueden desarrollarse independientemente y en paralelo entre los desarrolladores asignados. Una vez que una de estas tareas esté completa, un desarrollador introducirá el nuevo trabajo en el sistema de CI para integrarlo con el resto del proyecto.

A continuación, un ejemplo practico detallado de cómo debería estructurarse una correcta integración continua de código.

1. **CRONOGRAMA DE ACTIVIDADES DEL TRABAJO DE TITULACIÓN**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TIEMPO (meses) (semanas)**  **ACTIVIDAD** | **1** | | | | **2** | | | | **3** | | | | **4** | | | | **5** | | | |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
| Se Incluyen las actividades del proyecto para el cumplimiento de los objetivos |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**AÑO 201…..**

1. **CONTENIDO DE LA MEMORIA DEL TRABAJO DE TITULACIÓN**

**10.1 MODALIDAD PROYECTOS DE INVESTIGACIÓN**

|  |
| --- |
| **El trabajo de titulación comprenderá las siguientes partes:**   1. **PRELIMINARES:**    1. **Pasta o cubierta**    2. **Guardas (páginas en blanco)**    3. **Portada**    4. **Derechos de Autor**    5. **Página de Aprobación del Tutor**    6. **Página de Aprobación del Tribunal**    7. **Página de dedicatoria (opcional)**    8. **Página de agradecimientos (opcional)**    9. **Contenido**    10. **Listas especiales (tablas, figuras, anexos, etc.)**    11. **Glosario (opcional)**    12. **Resumen y palabras claves (castellano e inglés)** 2. **CUERPO DEL DOCUMENTO:**    1. **Introducción**    2. **Teoría ( máximo 10 páginas)**    3. **Metodología experimental**    4. **Cálculos y resultados**    5. **Discusión de resultados**    6. **Conclusiones**    7. **Recomendaciones**    8. **Citas bibliográficas** 3. **ELEMENTOS COMPLEMENTARIOS:**    1. **Bibliografía**    2. **Anexos** |

**10.2 MODALIDAD PROYECTOS INTEGRADORES**

|  |
| --- |
| **El trabajo de titulación comprenderá las siguientes partes:**   1. **PRELIMINARES:**    1. **Pasta o cubierta**    2. **Guardas (páginas en blanco)**    3. **Portada**    4. **Derechos de Autor**    5. **Página de Aprobación del Tutor**    6. **Página de Aprobación del Tribunal**    7. **Página de dedicatoria (opcional)**    8. **Página de agradecimientos (opcional)**    9. **Contenido**    10. **Listas especiales (tablas, figuras, anexos, etc.)**    11. **Glosario (opcional)**    12. **Resumen y palabras claves (castellano e inglés)** 2. **CUERPO DEL DOCUMENTO:**    1. **Introducción**    2. **Teoría ( máximo 10 páginas)**    3. **Estudio de Mercado,**    4. **Tamaño y Localización**    5. **Estudio Técnico (Ingeniería del proyecto industrial)**    6. **Estudio Económico Financiero (Costo y rentabilidad del proyecto)**    7. **Discusión**    8. **Conclusiones**    9. **Recomendaciones**    10. **Citas bibliográficas** 3. **ELEMENTOS COMPLEMENTARIOS:**    1. **Bibliografía**    2. **Anexos** |

1. **PRESUPUESTO DEL PROYECTO**

|  |
| --- |
|  |