

Universidad de las Ciencias Informáticas

Facultad 3



Título: Configuración de un entorno de Integración Continua para un modelo de desarrollo de software libre para PHP.

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas.

Autores: Yosbel Peñate Barceló.
Neisy Gómez Rodríguez.

Tutor: Ing. Eidy Marien Carbó Peña.
Co-tutor: Ing. Damián Ilizastegui Arriba

Ciudad de La Habana, Junio de 2008

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Autora: Neisy Gómez Rodríguez

Autor: Yosbel Peñate Barceló

Tutor: Ing. Eidy Marien Carbó Peña

"La función de un buen software es hacer que lo complejo aparente ser simple"

Grady Booch.

Agradecimientos

- ❖ A Arno Schneider, gracias por ayudarnos...
- ❖ A Humberto Medina Fernández, la persona a la cual le agradezco inmensamente por esos momentos tan lindos que pasé junto a él, por quererme y apoyarme todo este tiempo, por ser mi compañero, por cuidarme y por estar conmigo en los momentos difíciles de mi vida, por ser incondicional, por haber sido mi hombro de apoyo, gracias por todo.
- ❖ A Ermita por ayudarme tanto y por preocuparse siempre por mi, muchas gracias no tengo como agradecerle.
- ❖ A todos mis primos en especial Kenia, Aloima, Yoanni, Yaima y Keini, por su preocupación y ayuda.
- ❖ A mis tíos Andrés, Rolando y Jesús por ayudarme y apoyarme en todo.
- ❖ A mis hermanas Lorenys y Loriet, por estar siempre pendiente de mi, y porque a pesar de los años que pasan sin vernos, nunca se olvidan de mi.
- ❖ A mis amigas del cuarto Lisandra, Anaelis, Guía y Tatiana, por ser mi familia todo este tiempo, por haber estado conmigo en los momentos difíciles de mi vida, por aconsejarme y por apoyarme en mis decisiones, gracias y así pasen mil años siempre estarán en mi corazón.
- ❖ A mi compañero de tesis Yosbel Peñate Barceló por haber sido un gran amigo, por su gran esfuerzo en este trabajo y por levantarme el ánimo en todo momento, por ayudarme y aconsejarme cuando lo necesitaba, gracias a ti, hoy soy lo que siempre desee.
- ❖ A mi tutora Eidy por la orientación en todos y cada uno de los capítulos, por el tiempo dedicado, porque sin su asesoramiento este trabajo no hubiera sido posible y a su novio Yasmani por el tiempo y esfuerzo que dedicó en ayudarnos.
- ❖ A Danny Bermúdez Rodríguez por escucharme y apoyarme en todo.
- ❖ A Damian, Yaisel por su ayuda incondicional.
- ❖ A Alexander Salgado por su gran ayuda y comprensión.
- ❖ A Yunior Rodríguez, gracias por tu ayuda, no tendré como pagar todo lo que hiciste por mí.
- ❖ A mi amigo Abdi Yasser Rodríguez, porque sin él hoy no hubiera estado en esta universidad.
- ❖ A mis amigas Masniubys, Yusnelkys, Mildrey, Margarita, Danay, Marianela, Yaritza y Ailín, por haberme ayudado y por preocuparse siempre por mí.
- ❖ A Alexander Medina y a Rubén Medina por apoyarme en todo y ayudarme tanto.
- ❖ A mis vecinos Idalma, Teresa y Xiomara por estar pendiente de mi.
- ❖ A todos mis profesores y a otros como Yoandris y Bolívar por ayudarnos.
- ❖ A mi comandante Fidel Castro por darme la oportunidad de graduarme.

Agradecimientos

- ❖ A Arno Schneider por su apoyo incondicional, gracias.
- ❖ A mis padres (mima y pipo) un agradecimiento especial por estar siempre presente, por ser mis guías en todo momento y aconsejarme cuando lo necesité, por enseñarme lo que es bueno y lo que es malo en la vida, en fin a ustedes les debo la vida y hoy les agradezco, porque gracias a ustedes he llegado hasta aquí, de corazón “gracias Mima y gracias Pipo”.
- ❖ A mi novia Dainerys Santos Hernández, por su amor y cariño, por estar siempre presente cuando la necesité, “Sin ti no lo hubiera logrado”, gracias.
- ❖ A mis hermanos y primos por su comprensión y apoyo.
- ❖ A todos los profesores que tuve desde que entré en la universidad, que de una forma u otra, me ayudaron.
- ❖ A mi compañera de tesis Neisy Gómez Rodríguez, la cual se ha convertido en una gran amiga, gracias por luchar junto a mí en todo momento, gracias por su entrega y delicadeza para lograr el fruto de este trabajo.
- ❖ A mi tutora Eidy Marien Carbó Peña por su paciencia y sacrificio, por estar presente siempre que la necesitamos y por haber sido nuestra guía en el desarrollo de este trabajo.
- ❖ Gracias a mis vecinos, que de una forma u otra me brindaron su apoyo.
- ❖ Gracias a Damian por su ayuda incondicional.
- ❖ Gracias a todo aquel que se preocupó por preguntarme cómo me iba en la tesis.
- ❖ Gracias a todos mis amigos y amigas, en especial Yadira, Yami, Dustel, Ana, Iliana, quienes me prestaron su PC siempre que la necesité.
- ❖ A Teresa, Noel, Santos, Jorge Luis, gracias a todos por haberme brindado su apoyo, por y por preocuparse siempre por mí.
- ❖ A mis amigos de la asociación de MAYA, los cuales siempre estuvieron al pendiente y apoyándome en el transcurso de mi trabajo.
- ❖ Y un agradecimiento especial a todas aquellas personas que de una forma u otra, estuvieron relacionados con nuestro trabajo y otros que nos ayudaron en el perfeccionamiento del mismo como Yoandris Espinosa y otros más.
- ❖ A Pascual Verdecia Vicet, mis más gratos agradecimientos por haberme ayudado y apoyado siempre y cuando lo necesité.
- ❖ Gracias a Fidel Castro Ruz y a la Revolución por darme la oportunidad de ser lo que siempre he soñado.

Dedicatoria

- ❖ A mi abuela Elisa Arteaga Bravo por ser la persona que más quiero en el mundo, la que me ha cuidado y apoyado durante todo este tiempo y la que hace que cada día tenga un motivo por el cual seguir adelante. A ti abuelita muchas gracias por lo que has hecho por mí, gracias por quererme, por preocuparte tanto, gracias por todo...
- ❖ A mi mamá Nery Rodríguez Arteaga, por ser la mejor madre del mundo, por quererme, por apoyarme, por preocuparse tanto, por ser mi guía en este largo camino, gracias mamita, te quiero muchísimo.
- ❖ A mi hermana Yoana Fernández Rodríguez por ayudarme, por quererme, por ser un tesoro preciado en mi vida.
- ❖ A mi papá Carlos Fernández Cuellar, por haberme querido como una hija todo este tiempo, por ayudarme y por quererme, por ser el mejor papá del mundo, no hubiera deseado tener otro que no fueses tu, gracias...
- ❖ A mi abuelo Inocente Rodríguez Arteaga por ser mi guía y mi ángel guardián, gracias abuelito por protegerme tanto, siempre estarás en mi corazón y en el de mis hijos.
- ❖ A mi gran amiga Dilibet Jáuriga Bombino, con la cual compartí momentos difíciles de mi vida, mi amiga del alma y mi hermana, desearía que estuvieras aquí conmigo, lo que hoy soy te lo debo también a ti, gracias amiga por tu amistad y por tus buenos consejos... No te olvidaré jamás...

Neisy

Esta tesis se la dedico a mi madre Aida Barceló y a mi padre Reinaldo Peñate por haber sido mis ejemplos, por ser las personas más inteligente y sabias que conozco. Son ustedes mi más preciado tesoro y mi único refugio después de la guerra. Nada material me ata a este mundo, lo único valioso para mí, tiene el brillo de sus vigilantes ojos, son ustedes mi definición de hogar y de muchas otras palabras que conozco. Si camino con confianza y sin miedos, como si no importaran consecuencias, es porque se que no estoy solo y que no importa cuan lejos me encuentre ni cuan difícil mi situación, yo estoy allí junto a ustedes.

Yosbel

Resumen

Actualmente en la Universidad de las Ciencias Informáticas existen una serie de proyectos que se ven afectados por las pérdidas de tiempo y de recursos, provocados por las extensas fases de integración del software. El propósito de la presente investigación consiste en implantar un entorno de Integración Continua (CI, del inglés Continuous Integrations), en el módulo Gestión de Cuadros y Personal de Apoyo (GCPA) del proyecto Fiscalía General de la República (FGR) como caso de estudio, con el objetivo de automatizar los procesos de compilación, construcción y pruebas de dicho módulo. Para ello es necesario la realización del estudio del Estado del Arte de la práctica de CI y de las herramientas que permiten crear un escenario de la misma, por último se selecciona un conjunto de herramientas de software libre que permiten disponer de este entorno en el proyecto FGR e implantarlo en el módulo seleccionado. Finalmente, se crea un paquete de instalación que facilita el despliegue de la Integración Continua.

PALABRAS CLAVES

Integración Continua.

Índice

Índice	1
Índice de Figuras	1
INTRODUCCIÓN	1
Fundamentación Teórica	5
1.1. Introducción	5
1.2. Introducción a la Integración Continua	5
1.3. Escenario de Integración Continua	8
1.4. Ventajas y desventajas de la Integración Continua	10
1.5. Reducir riesgos con la Integración Continua	12
1.6. Buenas prácticas de la Integración Continua	13
1.7. La Integración Continua y su relación con otras disciplinas del proceso de desarrollo de software	16
1.7.1. Control de versiones.....	16
1.7.2. Pruebas en el proceso de Integración Continua.....	20
1.7.3. Automatización y construcción.....	23
1.7.4. Mecanismo de retroalimentación.....	24
1.8. Servidores de integración	25
1.8.1. Cruise Control.....	26
1.8.2. Apache Continuum.....	26
1.8.3. Luntbuild.....	27
1.8.4. PhpUnderControl.....	27
1.8.5. Xinc.....	28
1.9. Conclusiones Parciales	29
Solución Propuesta	30
2.1. Introducción	30
2.2. Análisis crítico del entorno actual	30
2.3. Características del entorno propuesto	31
2.4. Selección de herramientas para un entorno de Integración Continua	33
2.5. Instalación y configuración del servidor Subversion	34
2.5.1. Instalación de Subversion y Apache.....	34
2.5.2. Seguridad.....	35
2.6. Instalación y configuración de las herramientas del servidor de Integración Continua	36
2.6.1. Configuración del Proxy.....	37
2.6.2. Instalación del cliente Subversion, Phing y PHPUnit.....	37
2.6.3. Instalación del Xinc.....	37
2.6.4. Configurar Interfaz Web.....	38

2.6.5. Seguridad.....	39
2.6.6. Configuración de Xinc.....	42
2.7. Caso de prueba.....	44
2.7.1. Pruebas Unitarias con PHPUnit.....	46
2.7.2. Compilación Automática con Phing.....	47
2.8. Adicionar proyecto a Xinc.....	49
2.9. Retroalimentación.	52
2.9.1. Interfaz Web.....	52
2.10. Conclusiones Parciales.	56
<i>Despliegue del entorno propuesto</i>	<i>57</i>
3.1. Introducción.....	57
3.2. Paquete de instalación Xinc+Xampp.....	57
3.3. Estructura de Xinc+Xampp.	59
3.3.1. Instalación	60
3.4. Aplicando Integración Continua al módulo GCPA de FGR.	61
3.4.1. Análisis del Buildfile.....	62
3.4.2. Archivo de configuración del Xinc.....	64
3.4.3. Resultados.....	65
3.5. Conclusiones parciales.....	67
<i>Conclusiones Generales</i>	<i>68</i>
<i>Recomendaciones</i>	<i>69</i>
<i>Bibliografía.....</i>	<i>70</i>
<i>Anexos.....</i>	<i>72</i>
<i>Glosario.....</i>	<i>76</i>

Índice de Figuras

Figura 1. 1 Proceso de desarrollo sin Integración Continua.	6
Figura 1. 2 Proceso de desarrollo con Integración Continua.	7
Figura 1. 3 Escenario de Integración Continua.	9
Figura 1. 4 Esquema de un sistema de control de versiones.	17
Figura 1. 5 Proceso de retroalimentación continua.	25
Figura 2. 1 Entorno actual de desarrollo de Fiscalía General de la República.....	30
Figura 2. 2 Entorno de Integración Continua propuesto para el proyecto F G R.	32
Figura 2. 3 Procedimiento de instalación de CI.	36
Figura 2. 4 Representación de una iteración.....	45
Figura 2. 5 Código de prueba unitaria.	46
Figura 2. 6 Archivo de configuración del Phing.	47
Figura 2. 7 Archivo de configuración de Xinc.	50
Figura 2. 8 Estado del build.	54
Figura 2. 9 Duración del Build en segundos.....	54
Figura 3. 1 Estado del Build.	65
Figura 3. 2 Duración del Build.	66
Figura 3. 3 Interfaz Web del Xinc.	66
Fig. 1 Interfaz Web del Xinc.	72
Fig. 2 Menú de proyecto	72
Fig. 3 Estadísticas del Xinc.	73
Fig. 4 Sumario 1 del Xinc.	73
Fig. 5 Sumario 2 del Xinc.	74
Fig. 6 Estructura de archivos de Xinc+Xampp.....	75

INTRODUCCIÓN

Entre los años 1965 y 1985 hubo una explosión en la demanda de sistemas computarizados a nivel mundial, la incapacidad de poder dar respuesta al mercado, provocó una verdadera “crisis de software” nombrada así por Edsger Dijkstra [1]. Esta crisis fue superada no tanto por la mejora en la gestión de proyectos sino más bien por los progresos en los procesos de diseños y metodologías de desarrollo de los mismos.

En los últimos 20 años las notaciones de modelado y posteriormente las herramientas han sido las "balas de plata" para el deseado éxito en el desarrollo de software[2]. Este proceso de desarrollo ha llevado asociado una tendencia hacia el control de procesos mediante la definición de actividades, artefactos y roles. Los esquemas tradicionales, han demostrado ser factibles para proyectos grandes, donde la atención a la planificación de los procesos es primordial. Pero ocurre que muchas veces este enfoque no resulta ser el más adecuado para muchos proyectos actuales, donde es necesario disminuir el tiempo de desarrollo sin afectar la calidad del mismo.

Aunque metodologías ingenieriles como CMMI (Modelo de Capacidad de Madurez Integrado) y RUP (Proceso Unificado De Desarrollo de Software) han marcado pautas en el modo de producir software, continúan existiendo problemas en este aspecto que hacen a los proyectos difíciles de predecir. Costosas fases previas de especificación de requisitos, análisis y diseño, la pérdida de flexibilidad ante los cambios y la excesiva documentación hacen el desarrollo de los proyectos más lentos.

Como respuesta a los problemas antes mencionados surgieron las metodologías ligeras o como comúnmente se les denominan, metodologías ágiles, con una forma más flexible de desarrollo. Estas introducen cambios significativamente importantes sobre los modelos existentes de la Ingeniería de Software, apoyándose en experiencias prácticas que han sido aplicadas con éxito.

Las metodologías ágiles han ido revolucionando la manera de producir software. Muchas empresas han encaminando su proceso bajo estas metodologías, debido a sus facilidades, ya que constituyen, en gran medida, la solución para muchos proyectos, aportando una elevada simplificación sin renunciar a las prácticas esenciales para asegurar la calidad del producto [3].

Actualmente existe una gran diversidad de metodologías ágiles como: Scrum, Crystal Clear, Microsoft Solution Framework Agile (MSF Ágil), Método de Desarrollo de Sistema Dinámico (DSDM), Programación Extrema (XP), esta última siendo una de las más populares. Una de las prácticas que propone XP es la Integración Continua, que permite comprobar de manera periódica que todos los cambios que realizan cada uno de los desarrolladores no producen problemas de integración con el código del resto del equipo. Aunque surgió con la metodología XP esta se ha extendido a otras como RUP. La Integración Continua no corrige errores, solo permite detectarlos en etapas tempranas del desarrollo, basado en el criterio de que cuanto más rápido se detecte un error, más fácil será corregirlo.

Cuba ha comenzado a introducirse en el mercado de la Industria de Software a nivel mundial, basándose en experiencias de varios países, obteniendo como beneficio la producción de sus propios productos de software. Actualmente se han adoptado políticas para el buen desarrollo de este sector, orientadas a lograr su introducción en este mercado.

La Industria Cubana del Software, en pleno desarrollo, acoge a la Universidad de las Ciencias Informáticas (UCI), jugando un importante papel, con tan solo 6 años de experiencias en este sector se han obtenidos logros capaces de plasmar los adelantos logrados de una manera rápida y comprometida y donde se llevan a cabo una serie de proyectos productivos utilizados como sustento en la informatización del país. La UCI desde sus inicios ha seguido las tendencias actuales de los problemas del software por lo que no se ve exenta de las dificultades que esta trae consigo.

Encuestas realizadas en varios proyectos productivos de la UCI demuestran que todos conocen el significado de las pruebas unitarias y la importancia que tienen estas en el proceso de desarrollo de un proyecto, sin embargo exponen que no las realizan por el motivo de que no se les exige como parte de su trabajo. Se escogió además para aplicar esta encuesta proyectos que tienen implementado un entorno de Integración Continua, Procyon y CICPS, detectándose igualmente la escasez de pruebas unitarias y poco conocimiento del funcionamiento del proyecto.

Es importante tener en cuenta que para obtener un buen entorno de Integración Continua en un proyecto productivo, cada uno de los miembros del equipo de desarrollo debe ser capaz de elaborar sus propias pruebas unitarias y conocer la importancia de los build automáticos.

La Facultad 3 actualmente cuenta con 8 proyectos productivos de gran envergadura, entre los cuales se encuentra el proyecto de Fiscalía General de la República (FGR) que se dedica a la automatización de la acción diaria de los fiscales. Este cuenta con nueve subsistemas entre los cuales se encuentra el módulo Gestión de Cuadros y Personal de Apoyo (GCPA) que actualmente está en desarrollo.

El proyecto FGR usualmente se ve afectado por una serie de problemas los cuales son:

- Existen problemas de visibilidad en el proyecto.
- Los desarrolladores realizan una serie de tareas repetitivas que pueden ser automatizadas.
- No adoptan prácticas como el desarrollo de pruebas unitarias.
- Detección de errores de integración de forma tardía.

Problema científico: ¿Cómo identificar y controlar los errores en las etapas tempranas del proceso de desarrollo de software para el módulo GCPA del proyecto Fiscalía General de la República?

Objetivo General:

Implantar un entorno de Integración Continua en el módulo GCPA del proyecto FGR como caso de estudio.

Objetivos Específicos:

- Realizar un estudio del Estado del Arte del proceso de Integración Continua.
- Realizar un estudio de las herramientas que permiten crear un escenario de Integración Continua.
- Seleccionar un conjunto de herramientas de software libre que permitan disponer de un entorno de Integración Continua en el proyecto Fiscalía General de la República e implantarlo en un caso de estudio.

En correspondencia con el objetivo y problema planteado se define como **objeto de estudio**, la práctica de integración dentro del proceso de desarrollo de software y siendo su **campo de acción**, la Integración Continua.

Hipótesis de Investigación:

Si se implanta un entorno de Integración Continua para el módulo GCPA del proyecto FGR se lograrán identificar y controlar los errores de integración en las etapas tempranas del proceso de desarrollo de software.

Los **métodos de investigación científicos** utilizados son:

Métodos Teóricos

- **Análisis-Síntesis:** Este método se utilizó con el objetivo de comprender a fondo el proceso de la Integración Continua, desglosándolo en los componentes que la conforman, como son: el control de versiones, el servidor de integración de build y los mecanismos de retroalimentación.
- **Histórico-Lógico:** Utilizado para el estudio consecuente del Estado del Arte siguiendo un curso lógico de los acontecimientos.

Métodos Empíricos

- **Observación:** Utilizado para ver como se realizan los procesos y poder describirlos para entender cómo funciona el proceso de Integración Continua.

Estructura de la Tesis

Este trabajo esta conformado por tres capítulos, en el primero se presenta el marco teórico y referencial de la investigación donde se realiza un análisis crítico y valorativo del estado del arte en el tema de la Integración Continua. En el Capítulo 2, se propone un paquete de herramientas, que sirve de guía al proceso de CI en el proyecto Fiscalía. En el tercer capítulo tiene se muestra la aplicación de dicho paquete en un caso de estudio, así como la descripción de los resultados obtenidos.

La biografía que se revisó para conformar la investigación en su mayoría fueron artículos y libros de actualidad publicados entre los años 2000 y 2007.

CAPÍTULO I

Fundamentación Teórica

1.1. Introducción.

En este capítulo se describen de forma general los aspectos relacionados con el objeto de estudio y el campo de acción en el cual se trabaja. Esta sección constituye la base teórica para la comprensión del trabajo de diploma que se desarrolla y en el cual se abordan aspectos como: definición e introducción a la Integración Continua, se explica qué es un Controlador de Versiones (CV), cuál es su función e importancia dentro de un proyecto, cómo funcionan las pruebas automatizadas, en qué consisten los mecanismos de retroalimentación, etc. Por otra parte se analizan buenas prácticas en el uso de la Integración Continua así como, las ventajas y desventajas que ocasiona esta práctica dentro del proceso de desarrollo de software, riesgos que pueden ser eliminados con el uso de la misma, así como las herramientas utilizadas para su implantación.

1.2. Introducción a la Integración Continua.

El artículo “Continuous Integration” de Martin Fowler define CI como: “una práctica de desarrollo de software donde los miembros de un equipo integran su trabajo con frecuencia, cada persona integra por lo menos una vez al día conduciendo a múltiples integraciones diarias. Cada integración es verificada por una estructura automatizada (incluyendo pruebas) para detectar errores de integración lo más rápidamente posible. Muchos equipos encuentran que este procedimiento conduce a reducir perceptiblemente los problemas de integración y permite que un equipo desarrolle software cohesivo más rápidamente.” [4]

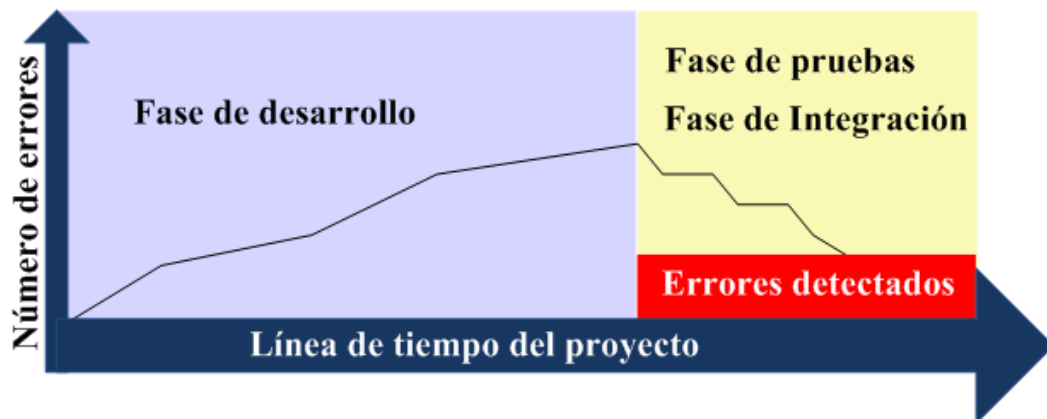


Figura 1. 1 Proceso de desarrollo sin Integración Continua.

A medida que aumenta el grado de independencia entre los desarrolladores, menos compatible se vuelven los diferentes módulos que ellos desarrollan y por tanto aumenta el número de problemas a la hora de integrarlos. Ejemplo de esto son los grandes proyectos de software libre (proyecto GNU) cuyos programadores están dispersos por todo el mundo y que por lo general su interacción solo se resume al correo electrónico, pero que colaboran entre sí para desarrollar proyectos de gran envergadura. En este tipo de proyectos sería engorroso dejar la integración para el final ya que el tiempo requerido para la búsqueda y erradicación de errores se haría extenso e imposible de predecir. En la figura 1.1 es posible apreciar como se comporta la cantidad de errores a lo largo del tiempo de desarrollo en los proyectos.

Para un buen desempeño de la Integración Continua, primeramente se necesita contar con un equipo disciplinado ya que en muchos proyectos no se trabaja en unión hasta que llega el momento de la integración final. La Integración Continua facilita que el desarrollo de los proyectos sea incremental, y brinda funcionalidades que permiten automatizar la ejecución de pruebas unitarias, esto permite que a medida que se va desarrollando el proyecto, se puedan añadir nuevas funcionalidades sin temor a que las demás dejen de funcionar.

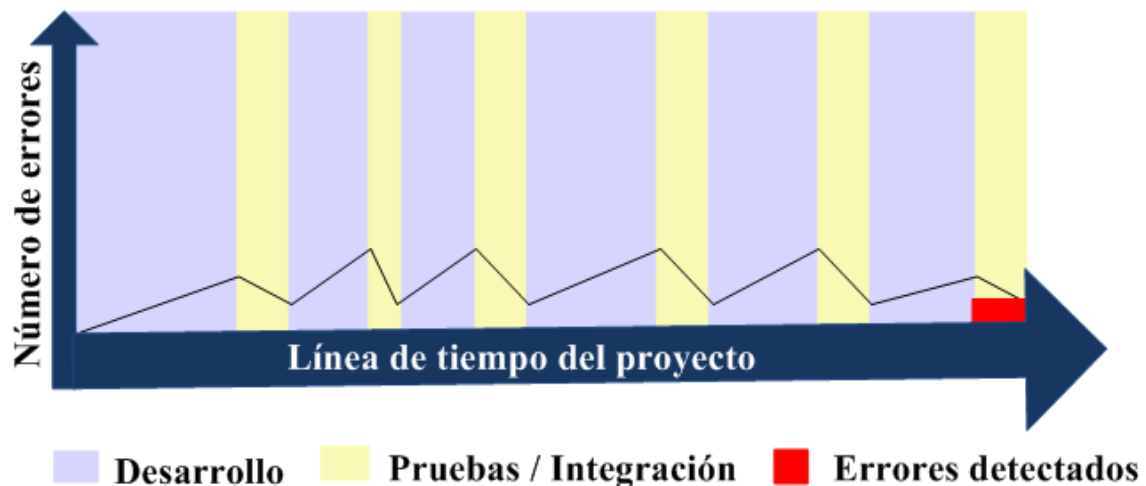


Figura 1. 2 Proceso de desarrollo con Integración Continua.

La Integración Continua es integrar un proyecto de una forma incremental y frecuente, permitiendo así encontrar problemas de integración en etapas tempranas y resolverlos rápidamente, reduciendo drásticamente las fases de integración y prueba (Figura 1.2). El proceso de integrar no es nuevo, constantemente los desarrolladores realizan build a lo largo del desarrollo del proyecto. Desde mucho antes del surgimiento del concepto de CI se consideraba como una buena práctica la realización de pruebas automáticas, aunque es válido especificar que no gozan de mucha aceptación entre los programadores ya que suelen ser consideradas como un trabajo innecesario. CI puede ser aplicado a cualquier proyecto, incluso en aquellos que son desarrollados por un único trabajador, pero su importancia se hace más perceptible a medida que aumenta la complejidad y el número de desarrolladores.[5]

La Integración Continua es una práctica que se aplica en todo tipo de entornos de desarrollo de software en la cual para automatizar el proceso de construcción, pruebas, despliegue y para que se realice diariamente es necesario [5]:

- Escoger un repositorio para almacenar las fuentes, donde se lleve el historial de todos los cambios del código y de donde sea posible obtener la última versión del proyecto.
- Automatizar el proceso de construcción de manera que a partir de las fuentes se pueda construir con un único comando todo el sistema.
- Realizar las pruebas de forma automática y de esta manera saber si todo está correcto o si existe algún problema.

¿Qué impide a los equipos de desarrollo el uso de Integración Continua?

Demasiados cambios: Algunas personas pueden sentir que son numerosos los procesos que necesitan cambios para poder introducir la Integración Continua al proyecto. Un aproximamiento incremental es más efectivo, por ejemplo: se puede comenzar con la construcción de la aplicación y pruebas con una frecuencia baja (ejemplo una vez al día), luego incrementar la frecuencia poco a poco mientras el equipo de desarrollo se sienta cómodo con los resultados.

Demasiados fallos en la construcción de la aplicación: Muchas personas creen que un build es solo compilar, pero no es así, ya que consiste entre otras cosas en la recopilación, las pruebas, inspección y despliegue. Build es el proceso que compila los códigos fuentes y verifica que el software funcione correctamente. Generalmente ocurren builds fallidos cuando los desarrolladores no hacen builds privados antes de subir el código al sistema de control de versiones, lo que provoca una rápida respuesta del sistema CI.

Incrementa los gastos al mantener un sistema de CI: Es una percepción errónea porque independientemente de que se tenga implementado un sistema CI o no, se tiene la necesidad dentro de un proyecto de integrar, realizar pruebas e inspecciones. Administrar un sistema CI es mejor que mantener un proceso manual.

Costo adicional en software y en hardware: Efectivamente CI requiere una máquina independiente ya que a menudo la integración es muy costosa, en lo que a tiempo de ejecución se refiere, pero resulta más costoso encontrar problemas en etapas tardías de un proyecto [5].

1.3. Escenario de Integración Continua.

Un escenario de Integración Continua comienza cuando el desarrollador realiza una serie de tareas que le son asignadas. Luego de finalizadas, este ejecuta un build privado, (que incluye pruebas unitarias que validan la calidad) y sube el código al repositorio de Control de Versiones. Hasta este punto el programador podría decir que su trabajo está completo, pero sucede que el servidor de integración estará constantemente chequeando la última versión del proyecto y realizando un conjunto de pruebas, (la frecuencia depende de como esté configurado el sistema de Integración Continua). En caso de fallos este enviará un correo electrónico con el reporte de errores a cada miembro del equipo, así resultará más fácil erradicar el problema.

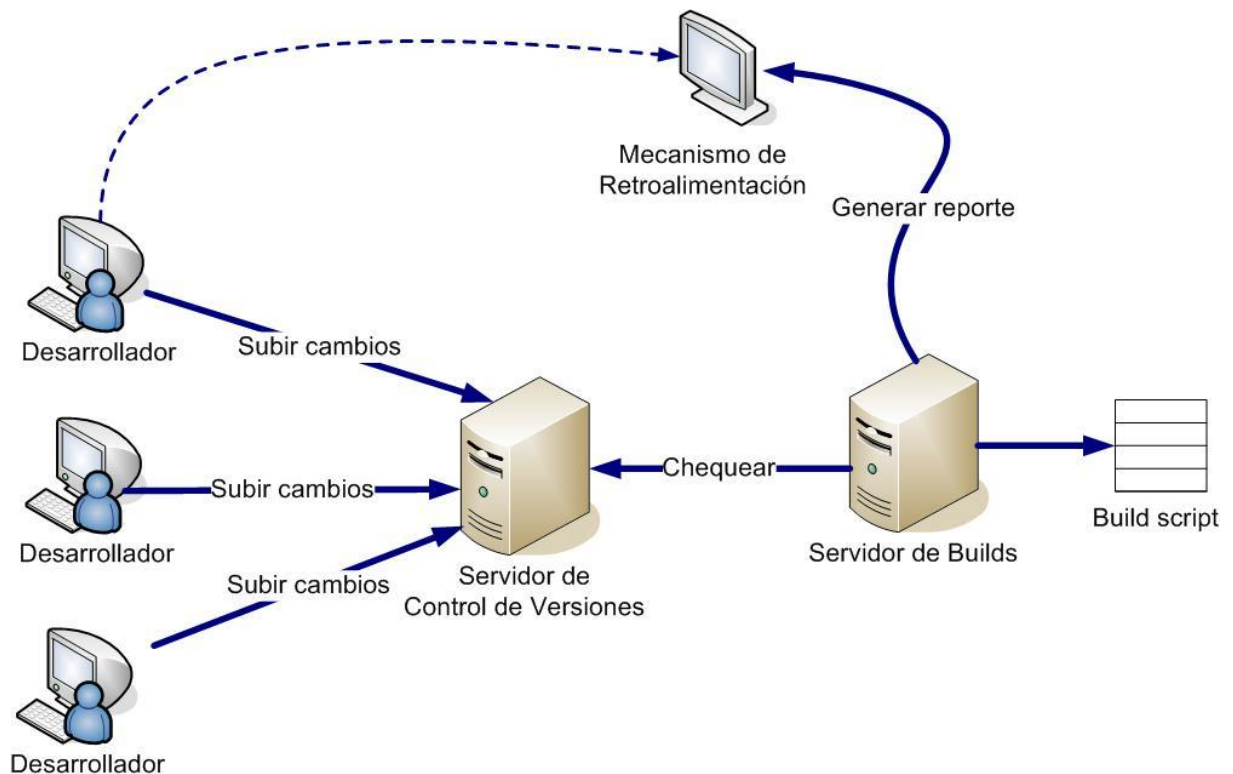


Figura 1. 3 Escenario de Integración Continua.

Como se puede observar en la Figura 1.3 para lograr un escenario de Integración Continua se cuenta con 4 componentes fundamentales [5]:

Un repositorio de CV: Para que la Integración Continua sea aplicada a la versión correcta del código, resulta necesario contar con un repositorio de control de versiones. Este permite obtener a los desarrolladores el historial de cambios realizados al proyecto en cualquier momento, es decir, es capaz de devolver cada versión del código hecha por el programador, permitiendo al equipo de desarrollo conservar, en caso de introducir errores, todo un registro con los cambios y de esta forma consultar versiones anteriores.

Un build script: No es más que un archivo de Lenguaje de Marcas Extensible (XML) con la configuración inicial del proceso que llevará a cabo la Integración Continua, el cual especifica datos como: la visibilidad del proyecto, cuando se aplicará CI, entre otros.

Un mecanismo de retroalimentación: Es necesario contar con un mecanismo de retroalimentación que informe inmediatamente de la ocurrencia de un problema. La retroalimentación no es más que la acción de informar acerca de la existencia de un error, no de solucionarlo como erróneamente se puede creer.

Para obtener los máximos beneficios es necesario definir quienes son los trabajadores del proyecto que recibirán las notificaciones y así establecer quien tomará las medidas necesarias para solucionarla. También es necesario determinar a qué persona se le enviará determinado reporte y que los miembros del proyecto establezcan qué hacer ante determinado evento. Por tanto con una pequeña planificación previa se puede elaborar un plan de contingencia para responder de forma rápida a las alertas y dar soluciones en un tiempo aceptable, si se tiene en cuenta que definir las responsabilidades correctamente es punto clave en la planificación de medidas rápidas y efectivas.

Un proceso para integrar los cambios del código fuente: Para integrar los cambios del código fuente se necesita de un proceso en memoria que cada cierto tiempo se active y realice la Integración Continua al código del proyecto. El servidor de integración, es el encargado de monitorear cada cierto tiempo al control de versiones en espera de actualizaciones en el proyecto. El responsable de la Integración Continua, configura el servidor de integración para que en intervalos definidos de tiempo descargue el proyecto, le realice las pruebas automatizadas, lo compile, y despliegue en un entorno semejante al final.

Los servidores de Integración Continua pueden ser configurados mediante ficheros que incrementan en gran medida sus ventajas en cuanto a la automatización se trata. Utiliza herramientas como NAnt o MSBuild para compilar el proyecto y mediante NUnit o alguna otra herramienta para correr las pruebas. Al terminar de ejecutar todas las herramientas, integra un reporte y lo publica en un sitio Web para que el equipo pueda ver el estado de desarrollo del proyecto.

La unión de todos estos componentes es lo que permite llevar a cabo un proceso de Integración Continua en la organización. Aunque estos no son los únicos elementos a tener en cuenta para una buena práctica dentro de un entorno de CI, son considerados los más básicos.

1.4. Ventajas y desventajas de la Integración Continua.

La Integración Continua constituye un cambio fundamental en las distintas organizaciones en cuanto a la forma de desarrollar software, ya que introduce nuevas reglas dentro del equipo de desarrollo, permitiendo un incremento de la calidad del mismo. Esta práctica no es muy fácil de aplicar pero su implementación trae consigo numerosas ventajas: [6]

- Disminuye la búsqueda de fallos a la hora de integrar el código, fallos que realmente son más difíciles de encontrar pues en muchas ocasiones suelen ser el resultado de un código desarrollado de manera independiente por diferentes desarrolladores.
- Permite identificar fallos en el entorno de producción en etapas tempranas, evitando largos períodos de integración finales en los mismos.
- Disminuye el tiempo de retroalimentación con el cliente al minimizar el paso de desarrollo a un entorno de integración.
- Aumenta la confianza de los desarrolladores al subir el código al control de versiones. El repositorio deja de ser un elemento estático dentro del proyecto, para convertirse en parte de un sistema dinámico. Esto permite al programador tener la seguridad de que el código que descarga del repositorio está correcto ya que ha pasado por una serie de pruebas de validación.
- Constante disponibilidad de los builds realizados, permitiendo el acceso a cada uno de estos así como sus correspondientes reportes que son almacenados en el servidor de integración. Este permite almacenar cada uno de los proyectos que fueron integrados correctamente y ponen a disposición de los desarrolladores, cada una de estas compilaciones enumeradas por fecha.

La implantación de la Integración Continua también trae consigo algunas desventajas que pueden relacionarse de la siguiente manera:

- **Sobrecarga por el mantenimiento del sistema:** Si el servidor de Integración Continua es detenido durante un tiempo considerable se acumula código en el control de versiones el cual podría tener errores que no serían detectados durante este período de tiempo. Además de que al comenzar nuevamente el servidor de integración podría sobrecargarse por existir un gran número de nuevos cambios,
- **Necesidad potencial de un servidor dedicado al build:** Existen varias razones por las cuales un servidor de integración necesita correr en un sistema dedicado, una es que las pruebas unitarias pueden llegar a ser muy costosas en tiempo de ejecución por lo tanto esto llevaría a conflicto con otros servicios que se ejecuten en el mismo, otra de las razones es que desde el mismo momento en que se aplica la Integración Continua, este se vuelve el proceso más importante dentro del ciclo de desarrollo ya que todos los actores dependen de su correcto funcionamiento.

- **El impacto inmediato al subir código erróneo provoca que los desarrolladores no suban su código frecuentemente como sería conveniente como copia de seguridad:** Un fallo en el servidor de integración provoca un paro en el desarrollo y un retraso en el proyecto.
- **Implica introducir una nueva filosofía de desarrollo:** Para aplicar Integración Continua dentro de un proyecto es necesaria la participación activa de cada uno de sus miembros, es decir primero se debe preparar a los desarrolladores y concientizarlos de la necesidad de llevar buenas prácticas de desarrollo lo cual a veces puede resultar complicado en grandes grupos de desarrolladores.

1.5. Reducir riesgos con la Integración Continua.

En todo proyecto existe una gran cantidad de riesgos que se deben manejar con cuidado y algunos de los cuales pueden ser reducidos mediante la Integración Continua, la cual ayuda a identificarlos y mitigarlos cuando ocurren. Algunos de estos riesgos son descritos a continuación [5]:

Carecer de software desplegable: En un panorama tradicional la integración y el despliegue de un proyecto se hace de manera manual y en etapas avanzadas del desarrollo. Existe una escasa confianza sobre si se podrá desplegar la última versión del software y extensas fases de integración antes de entregar internamente (equipo de prueba) o externamente (cliente). Sin embargo la Integración Continua propone obtener un software desplegable en cada integración exitosa del proyecto.

Descubrimientos tardíos de defectos: El riesgo anterior trae como consecuencia que muchos errores se mantengan latentes a lo largo del desarrollo y que las etapas de pruebas se hagan casi interminables. Es solo durante la integración y el despliegue que comienzan a descubrirse problemas que hasta entonces no se habían detectado. Errores que conducen a nuevos errores e incluso la dificultad de encontrar la raíz de estos, podrían paralizar un equipo durante días y retrasar la entrega de un proyecto. Es por esto que CI propone la búsqueda exhaustiva de fallos de integración en todo momento incluso desde etapas tempranas del desarrollo.

Carecer de visibilidad en el proyecto: En los proyectos tradicionales no existe un mecanismo automático que informe a sus miembros sobre el estado del mismo por lo que trae como consecuencia que el proyecto carezca de visibilidad. Aplicando Integración Continua es posible implementar un mecanismo de retroalimentación que informe a determinados miembros (o a todos) acerca del estado de la integración, estos mecanismos pueden ser el correo electrónico, Servicio de Mensajes Cortos (SMS), etc.

Baja calidad del software: Algunas personas creen que la baja calidad del software es únicamente diferido del costo del proyecto (después que se libera). La degeneración de código a causa de no seguir los estándares de diseño y codificación orientados por el arquitecto y la poca calidad del código conlleva a la aparición de errores potenciales graves. Esto se soluciona en gran medida mediante la Integración Continua, con la ejecución de las pruebas e inspecciones en cada cambio, de manera que se descubran posibles defectos que pueden ser introducidos en el código base.

1.6. Buenas prácticas de la Integración Continua.

Para un buen desarrollo de CI, primeramente se necesita contar con un equipo disciplinado, ya que en muchos proyectos no se percatan que se trabajan en equipo hasta el momento de la integración final. La Integración Continua, hace que el desarrollo del proyecto sea incremental, pero para que esto ocurra es muy importante poner en marcha algunas prácticas, aunque no es necesario que se realicen todas a la vez. Estas son [4]:

- **Mantener un único repositorio de control de versiones.**

Los proyectos de software incluyen archivos que necesitan ser organizados juntos, para construir un producto. Hacer seguimiento de todos estos requiere de un gran esfuerzo, por lo que no es de extrañar que a lo largo de los años los equipos de desarrollo de software hubiesen creado una herramienta para la gestión de los mismos. Estas herramientas llamadas, sistema de control de código fuente o de control de versiones, repositorios, entre otros nombres, son una parte fundamental de la mayoría de los proyectos de desarrollo, aunque no todos la usan. De aquí la importancia de que un proyecto cuente con un repositorio, para que no exista la duplicación de datos, ni confusiones. Además de que es imprescindible para realizar la Integración Continua que exista una única fuente de código.

- **Automatización del build.**

Con la automatización del build, a pesar de haber muchos programadores trabajando en distintos sectores y partes del código, todos van a compilar y generar el proyecto de la misma forma, para ello existe una serie de herramientas como Ant o Maven que son utilizadas con frecuencia en entornos de Integración Continua.

- **Integrar pruebas automatizadas dentro del proceso de compilación.**

Un programa se puede ejecutar sin ningún problema, pero esto no significa que haga lo que debe hacer. Una buena forma de capturar errores, con rapidez y eficacia, consiste en incluir pruebas automáticas en el proceso de construcción. Las pruebas necesitan poder ser lanzadas desde un único comando y chequeadas. El resultado de la ejecución de esta serie de pruebas debe indicar si alguna de ellas falla y por supuesto se debe tener en cuenta que en ellas, no se van a encontrar todos los errores.

- **Actualizar el repositorio de código frecuentemente.**

La integración permite comunicar a los desarrolladores sobre los cambios que han hecho y así conocerlos rápidamente. Es muy importante que cada desarrollador envíe su código al control de versiones diariamente, de esta forma es más útil para los desarrolladores pues mientras lo hagan con más frecuencia, habrá menos lugares en los cuales se tienen que buscar los errores de conflicto, y mayor será la rapidez para corregirlos. Teniendo desarrolladores subiendo el código cada pocas horas, se puede encontrar un error al poco tiempo de su aparición y si no ha habido demasiados cambios serán mas fácil de corregir los mismos pues los conflictos que se tardan varias semanas en detectar son más difíciles de corregir.

- **Con cada cambio de código, se debería lanzar el proceso de compilación en un entorno de integración.**

La Integración Continua implica que por cada cambio que realice el desarrollador en el control de versiones, se realicen las pruebas, se compile y se despliegue en un entorno semejante al final. De esta forma los errores serán detectados tan rápido como sea posible y ser una prioridad para cada desarrollador solucionarlos.

- **Mantener el proceso de compilación rápido.**

Un punto fundamental de la Integración Continua es realizar integraciones frecuentes y de esta forma mientras más se realicen mejor funcionará este proceso. Es por este motivo que las pruebas a realizar deben ejecutarse lo más rápido posible, tratando de reducir el tiempo que consumen estas y así el tiempo que consume el despliegue.

- **Realizar pruebas en un entorno igual al de producción.**

La idea de hacer el despliegue en un entorno semejante al de producción es buscar y erradicar errores bajo condiciones controladas. Es por esto que se debe mantener solo una máquina para hacer este tipo de pruebas ya que el cambio de entorno puede producir fallos en periodos avanzados del proyecto.

- **Fácil acceso a la última versión de los ejecutables.**

Es preciso que los desarrolladores tengan certeza de que el proyecto con el cual está trabajando es la última versión del mismo. Es inaceptable que un miembro itere sobre una versión mientras que el resto lo hace sobre otra ya que un error como este trae como consecuencia un largo período de corrección de errores, e incluso desechar módulos ya construidos. Por tanto la comunicación entre los miembros del equipo y la correcta implementación de un sistema de control de versiones se hace imprescindible dentro del grupo de desarrollo. Cada miembro debe conocer el lugar exacto donde se mantendrá la última versión del proyecto que no tiene errores.

- **Conocimiento por parte del equipo de desarrollo del estado actual del proyecto.**

Uno de los aspectos fundamentales de la Integración Continua es que cada miembro pueda ver el estado del proyecto en cada instante de tiempo, es por eso que un buen sistema de CI debe contar con una interfaz que permita monitorear cada una de las iteraciones que son realizadas sobre el código. Cruise Control por ejemplo provee de una interfaz Web donde se aprecian cada uno de los paquetes compilados a lo largo del período de desarrollo del proyecto así como gráficas y estadísticas, resultado de cada una de las compilaciones.

- **Automatizar el proceso de despliegue de las aplicaciones.**

La Integración Continua trata de realizar un conjunto de tareas varias veces al día, por tanto es conveniente automatizar cada una de estas. La aplicaciones deben ser desplegadas correctamente para que puedan estar al alcance de los miembros del equipo y además deberían ser desplegadas en un entorno semejante al de producción.

1.7. La Integración Continua y su relación con otras disciplinas del proceso de desarrollo de software.

La Integración Continua permite disminuir en gran medida el tiempo de desarrollo del proyecto, desde que comienza, hasta la finalización del mismo. Esto contribuye a trabajar con rapidez, mostrando al cliente como evoluciona el desarrollo en cada iteración y que con su ayuda se llegue a la solución que le sea más factible.

Con la implantación de CI también aparecen aparejados una serie de dificultades que vienen dadas a partir de otras prácticas en las cuales necesita apoyarse, por ejemplo [6]:

Controlador de Versiones (CV): En el proceso de Integración Continua es preciso tener un control de las diferentes versiones del código para de esta forma poder recuperar cualquier versión del proyecto, así como contar con un solo repositorio.

Pruebas Automáticas: Estas constituyen una parte importante del proceso de integración. Incluirlas a un proyecto y programar desarrollando pruebas es uno de los cambios de filosofía necesario para la implantación de CI.

Construcción y despliegue automatizados: Es necesario realizar las tareas de construcción y despliegue automatizados en un entorno semejante al final.

1.7.1. Control de versiones.

Un sistema de control de versiones es un sistema de gestión de archivos y directorios. Este mantiene el historial de los cambios y modificaciones que se han realizado sobre ellos a lo largo del tiempo. De esta forma, el sistema permite examinar el historial de cambios o recuperar versiones anteriores de un fichero, incluso si es eliminado. Los sistemas de control de versiones utilizan para su funcionamiento un repositorio central que es el lugar donde se almacenan los datos y la información.

Existen tres operaciones básicas en un control de versiones [7]:

- Bajar la versión completa del código.
- Actualizar cambios.
- Subir cambios.

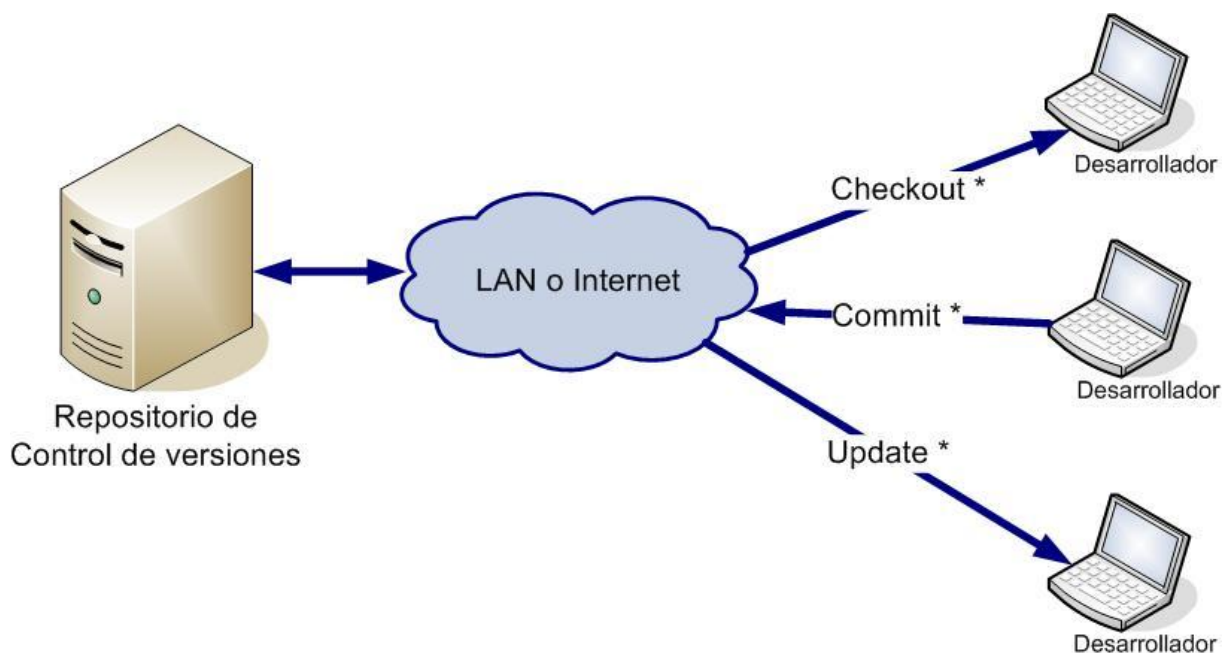


Figura 1. 4 Esquema de un sistema de control de versiones.

El control de versiones es una práctica importante para cualquier grupo de desarrollo de software. Existen varias herramientas para el control del código fuente pero sin importar el que se utilice, al menos deben cumplir con algunas de las siguientes características básicas [8]

- Proporcionar un lugar para almacenar el código fuente.
- Proveer un historial de lo que se ha hecho a lo largo del tiempo.
- Permitir el trabajo en paralelo de los desarrolladores, uniendo los esfuerzos más tarde.
- Proveer una manera de que los desarrolladores trabajen juntos sin interponerse en el camino de otro.

Con una herramienta sistema de control de versiones, trabajar en un equipo de trabajo es mucho más simple. Cada desarrollador tiene su propia carpeta de trabajo en su máquina, de esta forma los usuarios pueden realizar cambios en la misma sin tener que afectar al resto del equipo. El sistema de control de versiones ha sido durante mucho tiempo una herramienta muy importante para los programadores, los cuales normalmente perdían su tiempo realizando pequeños cambios al software y luego deshaciéndolos al día siguiente. Estos se clasifican en [9]:

Centralizados: En los sistemas centralizados el registro de cambios de un proyecto se almacena en un único servidor, que contiene toda la información del historial, de los cuales dependen distintos clientes para realizar operaciones como: modificar o consultar el historial del proyecto.

Distribuidos: Permiten que toda la información sobre el historial del proyecto no se mantenga en un único archivo central. En estos sistemas cada copia del proyecto, sostiene dicha información, permitiendo de esta forma que cada persona que trabaja con una copia disponga de las ventajas que brinda el control de versiones.

Existen diferentes herramientas para el control de versiones que pueden ser tanto privativas como de software libre ejemplo de ellos son: Sistema Concurrente de Versiones (CVS), Subversión, ClearCase, VisualSourceSafe, entre otros, que cuentan con distintas características que se necesitan estudiar para decidir cual de ellas utilizar en un proyecto. El actual código abierto de elección es el repositorio Subversion aunque vale destacar que CVS está todavía muy extendido y es uno de los más utilizados, pero Subversion es la opción moderna y contiene funcionalidades para el trabajo con ramas más potentes.

Un requisito fundamental para que sea posible la implantación de la Integración Continua es que todos los programadores trabajen simultáneamente sobre la misma base del código y esto solo es posible mediante el uso de herramientas de control de versiones, estas se analizan a continuación.

Sistema de Versiones Concurrentes (CVS)

Características:

- Utiliza una arquitectura cliente-servidor.
- Es un sistema de control de versiones bajo una licencia de tipo GNU/GPL, (Licencia Pública General de GNU)
- Permite a un grupo de desarrolladores trabajar y modificar concurrentemente ficheros organizados en proyectos, lo que significa que dos o más personas pueden transformar en un mismo fichero sin que se pierdan los trabajos de ninguna.
- Guarda las antiguas versiones de los ficheros, permitiendo de esta forma recuperar en cualquier momento versiones anteriores a la actual.
- Trabaja con código fuente de programas o con toda clase de documentos siempre que su formato sea completamente de texto, como pueden ser ficheros sgml/html/xml [10-12].

Subversion (SVN)

Subversión se creó con el objetivo de mejorar la funcionalidad de CVS, preservando su filosofía de desarrollo[13, 14].

Características:

- Mantiene versiones no sólo de archivos, sino también de directorios
- Es un sistema de control de versiones libre bajo una licencia de tipo Apache/BSD (Distribución de Software Berkeley) y se le conoce también como SVN por ser ese el nombre de la herramienta de línea de comandos.
- Es un sistema general que se puede utilizar para administrar cualquier conjunto de ficheros.
- Los archivos versionados no tienen cada uno un número de revisión independiente.
- Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos.
- Mayor eficiencia en la creación de ramas y etiquetas que en CVS.
- Realiza el seguimiento de las versiones de proyectos completos.
- Realiza el seguimiento del código modular (un archivo que se reutiliza, o se comparte, en varios proyectos).

Microsoft Visual SourceSafe (VSS)

Microsoft Visual SourceSafe conocida como VSS es una herramienta de control de versiones a nivel de archivos que forma parte de Microsoft Visual Studio[15].

Características:

- Permite que varias organizaciones trabajen en distintas versiones del proyecto al mismo tiempo y es muy importante en el proceso de desarrollo de software ya que se usa para mantener versiones de código paralelas.
- Ayuda al equipo a evitar la pérdida por accidente de archivos.
- Es un sistema de control de versiones bajo una licencia de tipo Propietaria.
- Permite realizar un seguimiento de las versiones anteriores de un archivo.
- Admite la bifurcación, el uso compartido, la combinación y la administración de versiones de archivos.

1.7.2. Pruebas en el proceso de Integración Continua.

Según un artículo de Paul Duvall [5], Integración Continua (CI), significa testeo continuo y se puede decir, que hay mucho de cierto en ello, porque básicamente cada esfuerzo de CI está encaminado a probar el software en cada momento de su ciclo de desarrollo llevándolo al límite, incluso cuando no se han desplegado pruebas automatizadas por parte del equipo de desarrollo (cosa que no es aconsejable). El servidor CI detecta errores en el proceso de compilación, empaquetado, despliegue, o problemas de compatibilidad, etc.

Una de las prácticas determinantes dentro de la Integración Continua es el desarrollo de las pruebas automatizadas, su implantación es uno de los retos más difíciles. Desarrollar un conjunto de pruebas completas del sistema, conlleva a cierta complejidad que depende del entorno de ejecución de la aplicación y de la interacción con los sistemas externos. Estas pruebas pueden ser divididas en diferentes conjuntos [6, 16] :

- Pruebas Unitarias.
- Pruebas de Integración.
- Pruebas de Rendimiento.
- Pruebas Funcionales.

En los modelos tradicionales el desarrollo de pruebas automatizadas se realiza una vez finalizado el proyecto, pero la mayoría de las veces suele suceder que este enfoque es poco efectivo. El desarrollo de pruebas automáticas brinda la facilidad de poder realizar cambios sin tener que volver a probar todo, pero dejarlas para el final priva de esta facilidad..

Existen múltiples criterios acerca de la realización de pruebas unitarias, algunos plantean que estas consumen un tiempo considerable durante su diseño e implementación, además de duplicar el trabajo del programador. Generalmente se asume que primero se debe programar y luego implementar las pruebas o al revés; nada más lejos de la realidad. Las pruebas deben elaborarse de forma paralela al desarrollo, de manera que son estas las que piden el código. Otro argumento es que su ejecución tarda, lo cual también es falso, las pruebas se ejecutan bastante rápido y si alguna se demora demasiado puede ser dividida en segmentos más pequeños.

Aunque muchos de los argumentos citados anteriormente podrían parecer válidos en determinado momento, lo cierto es que la diferencia entre un buen programador y uno malo está dada en que el buen programador utiliza pruebas para detectar sus errores lo antes posible. Mientras más rápido los errores salgan a la luz mucho más sencillo será encontrar su origen y erradicarlos.

De hecho se puede decir que el verdadero poder de la Integración Continua está en la realización de pruebas automatizadas, su diseño, elaboración y despliegue es una tarea de gran importancia. Solo sería necesario citar que el tiempo invertido por un programador detectando y corrigiendo errores supera al que utiliza en cualquier otra actividad. Algunos tipos de pruebas, prácticas, y herramientas que los desarrolladores pueden utilizar son [5]:

Automatización de pruebas unitarias: Las pruebas unitarias son una forma de probar el correcto funcionamiento de un módulo de código, mediante el uso de framework como NUnit, JUnit o PHPUnit.

Automatización de pruebas de componentes: Automatizar las pruebas de componente es mucho más complicado que las pruebas unitarias, generalmente estas envuelven más objetos y toman más tiempo ejecutarlas. Es posible usar herramientas como JUnit, NUnit, DbUnit, y NDbUnit en caso de usar bases de datos.

Automatización de pruebas de sistema: Las pruebas de sistema toman mucho más tiempo que las anteriores ya que envuelven varios componentes. Estas tienen como objetivo verificar que el comportamiento externo del software, satisface los requisitos establecidos por los clientes y futuros usuarios del mismo.

Automatización de pruebas de funcionalidad: Se realizan desde una perspectiva del usuario. Pueden usarse herramientas como Selenium (para Web) y Abbot (para GUI).

Crear categorías de pruebas: Para facilitar las pruebas es posible agruparlas por categorías de acuerdo a complejidad o tiempo de ejecución.

Ejecutar las pruebas más rápidas primero: Ejecutar las pruebas que requieren menos tiempo de ejecución.

Sin las pruebas automatizadas es muy difícil para los diseñadores u otros equipos de trabajo tener confianza en los cambios que se realizan en el software. La mayoría de los diseñadores que utilizan un sistema de CI utilizan herramientas para realizar pruebas unitarias como JUnit, NUnit, u otros xUnit frameworks. También se puede ejecutar categorías diferentes de pruebas de un proceso de CI para acelerar sus builds. A continuación se describen dos herramientas para la realización de las pruebas automatizadas, estas son:

JUnit

- Framework de pruebas unitarias creado por Erich Gamma y Kent Beck.
- Herramienta de código abierto que se ha convertido en el estándar para las pruebas unitarias en Java y que es soportado por la mayoría de los Entorno de Desarrollo Integrado (IDEs), como Eclipse o Net-Beans.
- Tiene licencia Licencia Pública Común (CPL), que permite usarlo de forma gratuita para cualquier aplicación, sea ésta comercial o de código abierto.
- Consiste en un conjunto de clases, fácil de usar y aprender que les permite a los programadores escribir sus pruebas unitarias en el lenguaje Java.
- Posee una comunidad mucho mayor que el resto de los frameworks de pruebas en Java [17, 18].

PHPUnit

- PHPUnit nace del lado de conceptos de metodologías ágiles, programación extrema, etc; sin embargo, esto no significa que deba utilizarse con metodologías ágiles, entre sus características se encuentra:
- Framework para las pruebas unitarias en específico a PHP.
- Se encuentra bajo licencia BSD.
- Forma parte del grupo de frameworks de xUnit
- Almacena los resultados en una base de datos de pruebas.
- Se integra con varias aplicaciones de test.
- Facilita la creación de pequeños scripts que ayudan a testear las aplicaciones y analizar los resultados [19, 20].

1.7.3. Automatización y construcción.

La Integración Continua consiste en llevar un proceso de construcción automático diario. Para esto es necesario automatizar todas las tareas que se realizan, para conseguir construir un sistema a partir del código que se ha desarrollado y lograr que sea desplegado y esté disponible para poder ser utilizado.

Automatizar un proceso puede ser una tarea de mucho esfuerzo, se deben automatizar todas las tareas que se realizan y de esta forma conseguir construir el sistema a partir del código que se ha desarrollado para lograr que sea desplegado y esté disponible para poder ser utilizado. Para automatizar todos estos procesos existen algunas herramientas las cuales permiten llevar a cabo la mayoría de las tareas que se ejecutan de forma manual, estas son [6]:

Ant

Ant [21, 22] es una herramienta gratuita de construcción de dominio público utilizada en la compilación y creación de programas Java.

- Se basa en la ejecución de tareas que se modelan en un fichero el XML.
- Es una de las herramientas J2EE de construcción más usada.
- Es multiplataforma, ya que está escrito en Java.
- Licencia Apache 2.0.
- Utiliza ficheros de construcción escritos en XML.
- Puede extenderse fácilmente creando nuevas tareas.

Maven

- Maven es una herramienta para la gestión y comprensión de proyectos Java
- Es capaz de encargarse de la gestión de dependencias, construcción de los artefactos, generación de la documentación, etc.
- Su configuración es más simple, reutilizable y consistente que la de Ant.
- Una vez instalado y configurado, no es necesario mantenerlo.
- Es rápido el acceso a los paquetes ya descargados.
- Licencia Apache 2.0.
- Independencia de la conexión a internet, excepto cuando se necesita algún paquete nuevo [23].

Phing

- Phing es un sistema de compilación automático basado en Apache Ant.
- Puede ser configurado para que realice tareas de compilación sin la intervención por parte del usuario.
- Permite realizar tareas de administración de proyectos, como: copiar, pegar archivos, ejecución de SQL, soporte CVS, Repositorio de Extensiones y Aplicaciones para PHP, Pear, etc.
- Permite la posibilidad de poder usar una serie de tareas ya predefinidas o por el contrario programar nuevas tareas adicionándolas a la librería de tareas disponible lo cual proporciona gran escalabilidad.
- Es extensible y utiliza archivos XML de configuración.
- Uso sencillo.
- Licencia LGPL (Licencia Pública General Limitada).
- Ofrece tareas de ayuda en las pruebas de código, además de documentación.

1.7.4. Mecanismo de retroalimentación.

Una de las pautas esenciales que propone la Integración continua es la implementación de un mecanismo de retroalimentación robusto el cual permita la solución de errores de una manera rápida como un punto crítico en el desarrollo del software. La pregunta que debe hacerse todo programador es si es aconsejable seguir programando para saber si se tiene o no un error. En la mayoría de los casos es aconsejable resolver el problema antes de continuar, porque puede darse el caso de tener que reprogramar todo de nuevo hasta ese punto y a veces es inevitable que un error conlleve a otro o que simplemente sea cambiado algo que no tenía que cambiarse. Es por esta razón que la Integración Continua aconseja el uso de medios que alerten lo más pronto posible de la existencia de fallos.

Muchos servidores de CI proveen facilidades para la retroalimentación haciendo uso del correo electrónico, para enviar mensajes acerca del estado del proyecto, incluso mensajes SMS, a teléfonos celulares donde el responsable es informado en cualquier momento que se está dando un problema y así poder responder con medidas de contingencia previamente establecidas por el equipo de desarrollo [5].

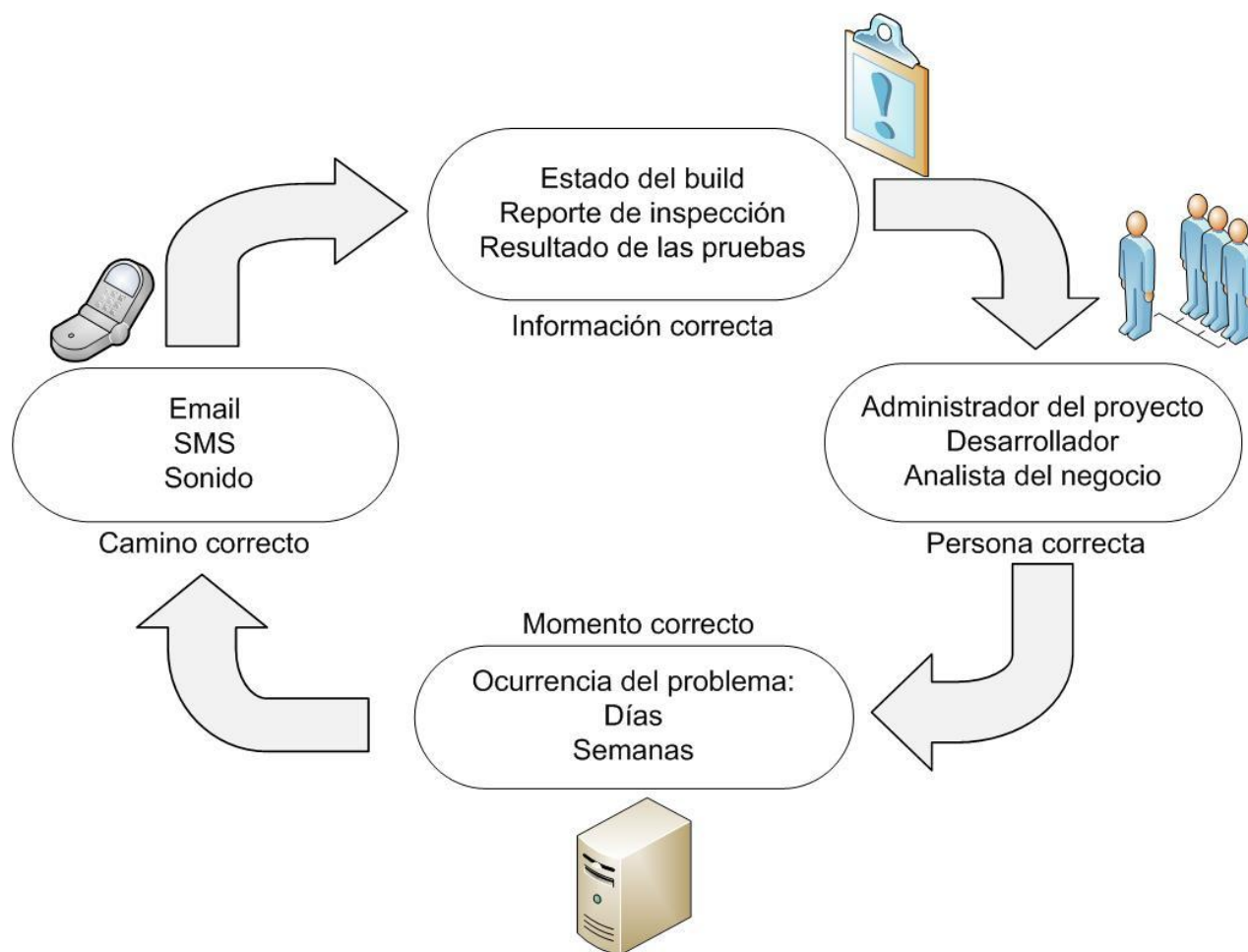


Figura 1. 5 Proceso de retroalimentación continua.

1.8. Servidores de integración.

La implementación de un entorno de Integración Continua no es una tarea fácil ya que requiere de una gran complejidad técnica. Automatizar el proceso de construcción, pruebas y despliegue sin apoyarse en una herramienta resulta bastante costoso. Existen diferentes herramientas tanto libres como propietarias que permiten la automatización de estas tareas repetitivas, la selección de una de ellas debe basarse según las particularidades de cada equipo de desarrollo. A continuación se muestran un conjunto de servidores de integración y se mencionan algunas de sus características principales.

1.8.1. Cruise Control.

- Cruise Control es una herramienta de código abierto distribuido bajo licencia BSD.
- Es una herramienta que surge en un principio para realizar la Integración Continua a proyectos en Java.
- Basa su funcionamiento en plugins, que se le añaden a un núcleo básico, permitiendo así que los usuarios avanzados puedan extender su funcionalidad.
- Es compatible con algunas herramientas como son Apache, Ant, Maven y NAnt.
- Ofrece soporte para una gran variedad de sistemas de control de versiones incluyendo CVS, SVN y Visual Source Safe (VSS).
- Soporta medios de almacenamiento como por ejemplo servidores Protocolo de Transferencia de Archivos (FTP) o Web.
- Presenta un variado número de plugins que le permiten interactuar con una amplia variedad de mecanismos de retroalimentación, entre los cuales se encuentran el correo electrónico y algunos protocolos de mensajería instantánea como el Jabber.
- Fácil proceso de instalación que no requiere un alto nivel de conocimientos por parte del usuario.
- Configuración a partir de Lenguaje de Marcas Extensibles (XML), en el que se definen, entre otras cosas, las tareas a realizar en el ciclo de construcción para cada uno de los proyectos registrados. [25, 26]

1.8.2. Apache Continuum.

- Continuum es un servidor de Integración Continua para la construcción de proyectos en Java.
- Producto de código abierto y distribuido bajo licencia Apache 2.
- Para la construcción de sus proyectos brinda soporte para el uso de Ant, Mavent 1 y 2 y Shell Scripts como herramientas de automatización.
- Brinda soporte para los sistemas de control de versiones: CVS, Subversion, Clearcase, Perforce,
- En su versión 1.0.3 brinda soporte para el envío de correos electrónicos así como a cuatro mecanismos de notificación, tres de ellos son protocolos de mensajería. [26, 27]

1.8.3. Luntbuild.

- Distribuido bajo licencia Apache 2.0, que permite definir y planificar las distintas tareas a realizar de integración de los proyectos de software; existe también una versión comercial de este proyecto llamada QuickBuild.
- Se integra con diversas utilidades de construcción de software, Ant, Maven1 y Maven2.
- Brinda soporte a los sistemas de control de versiones: CVS, Subversion, ClearCase, y Perforce. Tiene varios métodos para notificar los resultados de la construcción de los builds, entre los que se destacan: el envío de correos electrónicos y el uso del protocolo de mensajería de Jabber, etc.
- Toda la configuración, manejo y administración del sistema se hace mediante una interfaz Web, que es ejecutada por el servidor HTTP que viene embebido en el sistema, aunque brinda la posibilidad de utilizar algún otro servidor [26, 28].

1.8.4. PhpUnderControl.

- Es distribuido bajo las normas de la licencia BSD lo cual supone una desventaja ya que una herramienta que se distribuye bajo la misma permite que la compañía o entidad tenga la libertad de cambiar la licencia a otra más restrictiva cuando lo estime conveniente. O sea no existe seguridad de que el software sea siempre de código abierto sino que puede la licencia ser cambiada a software privativo. Otras características significativas de PhpUnderControl son:
- Facilidad de uso e instalación por parte del usuario.
- Es configurado a través de archivos de configuración de XML.
- Para el uso de las pruebas unitarias es compatible con PHPUnit.
- Genera automáticamente la documentación a partir de código PHP haciendo uso de PhpDocumentor.
- Realiza las tareas de compilación de forma automática haciendo uso de Ant que está desarrollado en java y además es extensible a través de este lenguaje.
- Posibilita a través de la herramienta PHP_CodeSniffer implementar reglas de codificación, las cuales tienen gran importancia a la hora de establecer estándares de programación entre los miembros de un equipo de desarrollo.

- Utiliza el correo electrónico y el formato de datos RSS como mecanismo de retroalimentación además usa X10, Jabber entre otros.
- Es compatible con Windows, Unix y cualquier plataforma que soporte el marco de Java.
- PhpUndercontrol es un plugins de CruiseControl por tanto necesita de un entorno Java y una instalación funcionando de CruiseControl.
- Para su mantenimiento es necesario poseer conocimientos de Java.

1.8.5. Xinc.

- Este servidor es distribuido bajo licencia LGPL que diferencia de la licencia utilizada por PhpUnderControl y Cruise Control garantiza que un software nunca pueda ser privatizado. Además permite al software privado utilizar librerías y componentes libres, siempre y cuando se especifique qué componentes son utilizados. Otras características significativas de Xinc son:
- Realiza las tareas de compilación de forma automática usando Phing, la cual es una herramienta totalmente implementada en PHP.
- Genera automáticamente la documentación a partir de código PHP haciendo uso de PhpDocumentor.
- Utiliza el correo electrónico y el formato de datos RSS como mecanismo de retroalimentación.
- Facilidad de uso e instalación por parte del usuario.
- Es configurado a través de archivos de configuración de XML.
- Para el uso de las pruebas unitarias es compatible con PHPUnit.
- Utiliza Xdebug como herramienta para la depuración de código.
- Es compatible con los sistemas operativos Ubuntu, Debian y Fedora.
- Visualización de gráficas con estadísticas sobre la compilación.
- Muestra resúmenes y detalles de las pruebas unitarias.
- Utiliza comandos Xinc-setting y Xinc-build que permiten cambiar la configuración.
- Cronometra el tiempo.

1.9. Conclusiones Parciales.

En este capítulo se realizó el estudio del Estado del Arte de la Integración Continua, cumpliendo parcialmente algunos objetivos específicos planteados.

- La Integración Continua es una práctica muy importante para proyectos grandes y rápidos, aunque exige una gran disciplina a los desarrolladores.
- CI permite detectar en etapas tempranas del proyecto posibles problemas de integración que pueden tener soluciones muy complejas posteriormente. Al ser validado continuamente, con una serie de pruebas, en un entorno similar al final, se alcanza en el mismo, un alto grado de confianza.
- La Integración Continua minimiza el tiempo de puesta en producción del proyecto, acortando el ciclo desde que el cliente pide un proyecto nuevo o un cambio hasta que puede tenerlo en producción.
- Crea una motivación en el cliente, mostrándole a cada momento cómo evoluciona el desarrollo del proyecto y de esta forma pudiendo llegar a la mejor solución.
- Para la implantación de esta práctica el movimiento de Software libre proporciona herramientas muy potentes (CruiseControl, Ant, CVS, JUnit, etc) que facilitarán este camino.

CAPÍTULO II

Solución Propuesta

2.1. Introducción.

En el presente capítulo se realiza una propuesta de cómo aplicar un entorno de Integración Continua en el proyecto de Fiscalía General de la República de la Facultad 3, partiendo del análisis crítico de cómo se lleva a cabo su proceso de desarrollo de software al cual se le inserta la práctica de CI. Además se presenta la selección de un conjunto de herramientas que soportarán esta práctica en dicho proyecto y se mostrará cómo instalar y configurar cada una de ellas para obtener los máximos beneficios.

2.2. Análisis crítico del entorno actual.

En la actualidad el proceso de desarrollo del proyecto Fiscalía General de la República incluye una serie de flujos y acciones que rigen el comportamiento del mismo y posibilitan llevar a cabo el desarrollo del proyecto. En la siguiente figura se representa este entorno y una breve descripción de como fluye el proceso de desarrollo actualmente en el mismo.

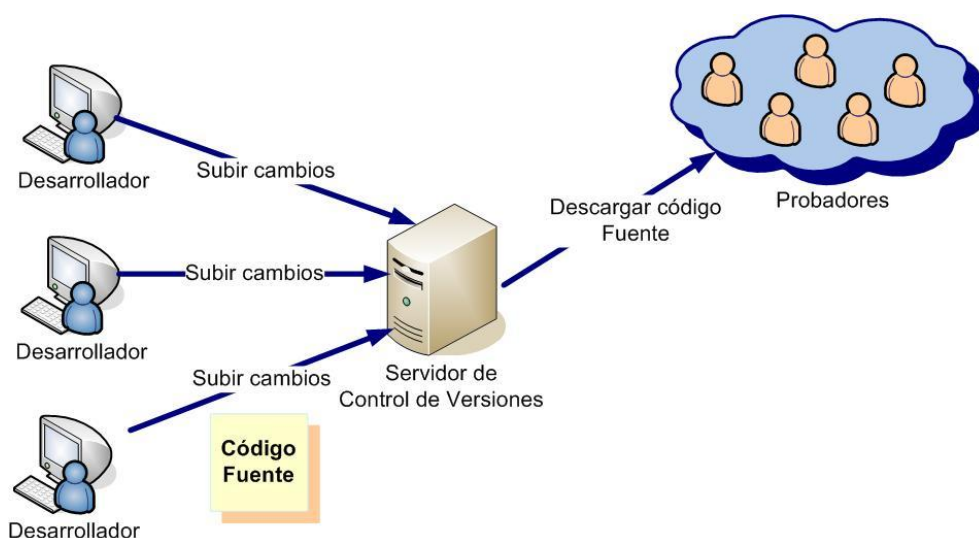


Figura 2. 1 Entorno actual de desarrollo de Fiscalía General de la República.

Durante el desarrollo del proyecto los programadores trabajan en los diferentes módulos asignados. Luego de terminar cada tarea, estos suben su código al servidor de control de versiones que se encuentra en un ordenador dentro del proyecto. Un pequeño grupo de calidad es el encargado de realizar las pruebas, descargando para esto el código del control de versiones. Es preciso resaltar que este trabajo se realiza de forma manual, es decir, no se utilizan herramientas que automaticen la realización de las pruebas, o al menos no se les exigen a los probadores usarlas.

El proyecto de FGR no aplica la Integración Continua como práctica para mejorar el proceso de desarrollo de software detectándose los siguientes problemas:

- Las pruebas se realizan de una forma primitiva, los probadores en muchos casos prueban el código pasando los datos a mano y analizando los resultados. Esta forma de probar métodos y funcionalidades trae como consecuencia que no todo el código sea probado como exigen las normas internacionales de calidad y generalmente los usuarios finales son víctimas de un gran número de problemas con la consecuente pérdida de prestigio y confianza de los clientes.
- Referente a la manera en que el código es probado, el problema radica en el hecho de que a veces el intervalo de tiempo que transcurre entre la programación de una funcionalidad y el momento en que es probada por el equipo de calidad interna puede ser de semanas e incluso meses. Como consecuencia de esto, la búsqueda de errores puede volverse todo un reto, incluso para los programadores que implementaron los códigos.

2.3. Características del entorno propuesto.

En el presente trabajo los autores proponen un modelo de un entorno de Integración Continua para ser instaurado en el proyecto Fiscalía General de la República el cual utiliza plataforma libre y se desarrolla en el lenguaje PHP. Este modelo se basa en el análisis del entorno propuesto por Martin Fowler que se muestra en el capítulo 1, Figura 1.3. Existen algunas diferencias entre el modelo teórico de Integración Continua y el propuesto por los autores, debido fundamentalmente a que las políticas de seguridad de la universidad no permiten implementar un sistema de retroalimentación automático (como debe ser) basado en servicios de mensajería electrónica.

En la siguiente figura se muestra el esquema del entorno de Integración Continua propuesto para el proyecto Fiscalía General de la República.

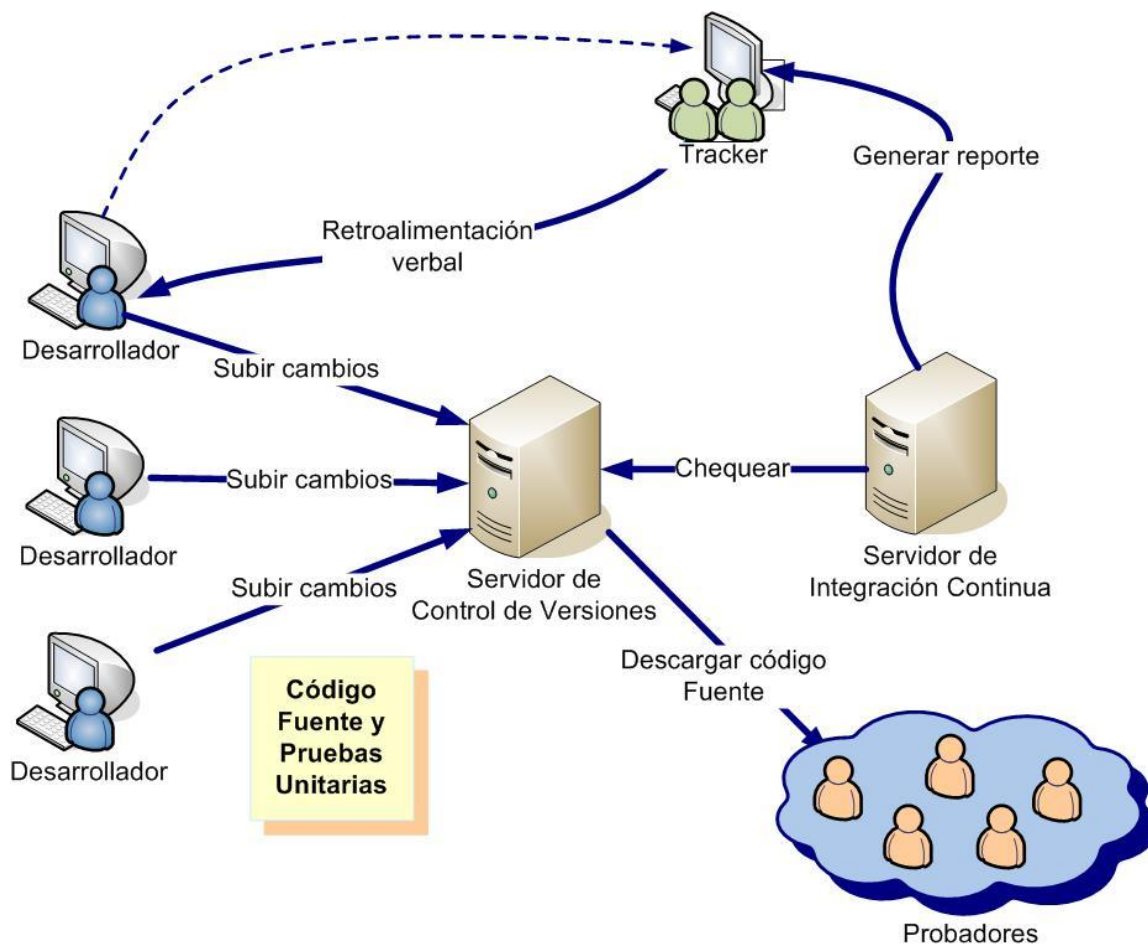


Figura 2. 2 Entorno de Integración Continua propuesto para el proyecto F G R.

Descripción del flujo de trabajo en el entorno propuesto:

Primeramente los desarrolladores realizan su trabajo y elaboran las pruebas unitarias que lo validan, luego realizan un build privado para asegurarse de que no existen errores. Posteriormente estos suben los cambios al control de versiones (código y nuevas pruebas unitarias), que son almacenados en el repositorio. El servidor de Integración Continua, en intervalos de 20 minutos (es configurable) chequea si existe una nueva versión del proyecto y si han ocurrido cambios desde la última consulta descarga el código, realiza un build al proyecto, ejecuta las pruebas unitarias y publica los resultados en la Web. Si el build falla, entonces existe un trabajador llamado Tracker o Rastreador que es el encargado de monitorear el estado de la integración, buscar el origen del error e informarle de esto al programador que introdujo el problema y exigirle que lo solucione inmediatamente.

En la propuesta se propone introducir el rol de Tracker dentro del proyecto, el cual va a ser el responsable de la Integración Continua y cuyas responsabilidades dentro del proyecto serán las siguientes [30]:

- **Controlar la marcha de las pruebas unitarias:** Es responsable de monitorear el estado de las pruebas unitarias y errores que ocurren en las integraciones.
- **Monitorear el estado del Build:** Chequea cada cierto tiempo el estado del build usando la interfaz Web del servidor de Integración Continua.
- **Informar a los programadores de los resultados del Build:** Responsable de informar a los programadores si ha sucedido un fallo en el build.
- **Dar mantenimiento al servidor de Integración Continua:** Configura tanto el servidor de Integración Continua y otros servicios tales como Apache, MySQL y PHP.
- **Desplegar políticas de contingencia:** Responsable de llevar a cabo todas aquellas acciones que son planificadas por el equipo de desarrollo para casos en que los builds sean fallidos.

2.4. Selección de herramientas para un entorno de Integración Continua.

En el análisis de la Figura 2.2, fue presentado cómo la Integración Continua consta de 5 elementos fundamentales: repositorio de control de versiones, servidor de integración, compilación automática, pruebas unitarias y mecanismo de retroalimentación.

En el capítulo anterior se mostró cómo cada uno de estos elementos estaba sustentado por un grupo de herramientas que automatizaban completamente el proceso de Integración Continua, llevando a cabo así un conjunto de tareas que los desarrolladores realizan a diario durante el desarrollo de todo producto de software.

Tomando como premisa el estudio realizado los autores proponen que para la implementación de un entorno de Integración Continua basado en tecnología libre y específicamente para utilizar como lenguaje de programación el PHP, se utilicen el conjunto de herramientas que se listan a continuación:

- Subversion → Repositorio de control de versiones.
- Xinc → Servidor de Integración.
- Phing → Compilación automática.
- PHPUnit → Pruebas Unitarias.
- Apache → Mecanismo para la retroalimentación.

Después de seleccionadas las herramientas necesarias para crear el entorno de Integración Continua propuesto se mostrará el proceso de instalación y configuración de cada una de ellas.

2.5. Instalación y configuración del servidor Subversion.

Uno de los aspectos más importantes para la implantación de la Integración Continua es tener habilitado un repositorio de control de versiones. Estos repositorios deben cumplir como premisas:

- Todo el proyecto debe estar bajo el mismo repositorio.
- El repositorio debe estar en un servidor dedicado.
- Cada miembro del equipo de trabajo posee su clave particular de acceso al servidor.

El cumplimiento del primer punto asegura que la Integración Continua se esté aplicando a la única versión del proyecto. El segundo permite evitar conflicto con otros servicios y el tercero no solo evita problemas de seguridad, sino que permite identificar unívocamente a los usuarios que provoquen los conflictos.

2.5.1. Instalación de Subversion y Apache.

Los autores proponen instalar Subversion y Apache, esto no solo proporciona la facilidad de poder acceder al contenido del repositorio a través del protocolo HTTP, sino que brinda la posibilidad de aprovechar las políticas de seguridad de Apache.

Para la instalación de Subversión y Apache son necesarios los siguientes paquetes:

- subversion: Servidor subversión.
- apache2: Servidor Apache 2.
- apache2-utils: Utilidades para Apache 2.
- libapache2-svn: Paquete que permite integrar Apache a subversión.

Después de instalar los paquetes necesarios se crea la carpeta raíz del servidor de control de versiones. Se debe activar el módulo del Apache DAV_SVN el cual permite a Apache utilizar las librerías del Subversion, es posible hacer esto mediante el comando.

```
# a2enmod dav_svn
```

El siguiente paso a seguir es dar permisos al servidor Apache para que pueda leer y escribir en el repositorio creado, esto se hace ejecutando los siguientes comandos:

```
# chown -R root.www-data /var/svn  
# chmod -R g+rw /var/svn
```

2.5.2. Seguridad.

Con Apache es posible elaborar una política de seguridad que puede ser de dos tipos:

- **Repositorio privado:** Solo un conjunto de usuarios bien definidos tienen acceso al repositorio.
- **Acceso restringido:** Cualquiera puede acceder al repositorio, pero solo un grupo de usuarios tiene permiso para modificarlos.

Por las características que posee el proyecto Fiscalía General de la República, los autores proponen implementar la política de seguridad del tipo: Acceso restringido. La forma de implementar esta política se describe a continuación.

```
<Location /rn>  
  DAV svn  
  #SVNPath /var/svn/docRN  
  SVNParentPath /var/svn  
  SVNListParentPath on  
  AuthType Basic  
  AuthName "Repositorio Doc RN"  
  AuthUserFile /etc/apache2/pws.passwd  
  AuthzSVNAccessFile  
  /etc/apache2/auth.passwd  
  Require valid-user  
</Location>
```

La directiva *AuthUserFile* permite especificar un fichero de claves, el mismo tendrá almacenado los usuarios que podrán acceder al directorio especificado. Más adelante en el subepígrafe 2.6.5 se explicará con más detalle cómo gestionar ficheros de claves en Apache, usando el comando *htpasswd*.

2.6. Instalación y configuración de las herramientas del servidor de Integración Continua. Continua.

Para facilitar la implantación de la Integración Continua y lograr un correcto funcionamiento de sus herramientas se necesita contar con el siguiente procedimiento de instalación propuesto por los autores. Algunas de las herramientas necesarias se encuentran disponibles desde los repositorios habilitados por la universidad, como es el caso del cliente Subversion, el resto se propone instalarlas como se muestra a continuación.

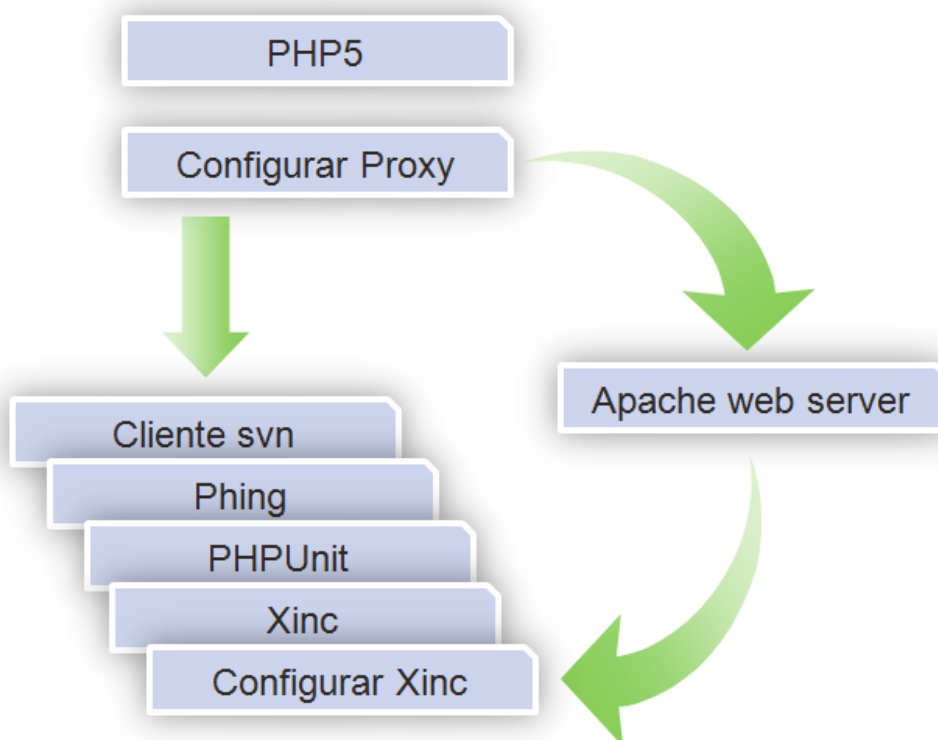


Figura 2. 3 Procedimiento de instalación de CI.

2.6.1. Configuración del Proxy.

Por las particularidades de la UCI y antes de usar Pear para descargar contenido de internet es necesario configurar el proxy del ordenador, esto se realiza mediante el comando *export* y de la siguiente manera:

```
export http_Proxy=http://<usuario_uci>:<pass_uci>@10.0.0.1:8080
```

Nota: El *<usuario_uci>* es el usuario de dominio UCI y *<pass_uci>*, la contraseña.

2.6.2. Instalación del cliente Subversion, Phing y PHPUnit.

Para la instalación del cliente Subversion, Phing y PHPUnit, se necesita contar con dos pasos fundamentales:

1. Conectarse a los repositorios en donde se encuentran disponibles las herramientas mediante el comando *pear channel-discover*. Estos repositorios son los siguientes:

- **Phing:** *pear.phing.info*
- **PHPUnit:** *pear.phpunit.de*

2. Instalar desde el repositorio las diferentes herramientas de la siguiente manera:

- **Phing:** *pear install phing/phing*
- **PHPUnit:** *pear install phpunit/phpunit*

2.6.3. Instalación del Xinc.

Para la instalación del Xinc se cuenta con los mismos pasos anteriores mediante los comandos:

```
pear channel-discover pear.xinc.eu  
pear install xinc/Xinc
```

Luego para poder visualizar reportes ofrecidos por esta herramienta es necesario disponer de un servidor Web, en su caso, Apache el cual se instala desde los repositorios de la UCI.

Para la inicialización del servidor Xinc, se ejecuta el comando el cual debe ser llamado con permisos de súper usuario mediante el comando *sudo* de Linux.

```
/etc/init.d/xinc start
```

El comando *xinc* permite controlar el estado del servidor Xinc, con el cual es posible detenerlo, reiniciarlo, así como conocer su estado (corriendo o detenido).

Nota: Aunque es posible instalar Xinc de manera manual esto requiere de ciertos conocimientos sobre el lenguaje bash por lo que los autores no lo aconsejan.

2.6.4. Configurar Interfaz Web.

Al instalar el servidor Apache es necesario configurarlo para poder visualizar los reportes del Xinc. Para esto se crea un enlace. Esto se hace creando un enlace al archivo *www.conf* en el directorio */etc/apache2/sites-enabled/* mediante el siguiente comando:

```
$ sudo ln -s /etc/xinc/www.conf /etc/apache2/sites-enabled/
```

Posteriormente se debe reiniciar el servidor Apache para que este cargue los cambios, mediante el comando *apache2ctl* y con permisos de súper usuario:

```
$ sudo apache2ctl restart
```

Además es necesario habilitar el módulo de Apache denominado *rewrite* el cual permite gestionar direcciones URL y el archivo del Apache *httpd.conf* adicionando la siguiente línea:

```
LoadModule rewrite_module /usr/lib/apache2/modules/mod_rewrite.so
```

Finalmente, se reinicia el servidor Apache y el servidor Xinc. La interfaz Web del Xinc estará disponible en la dirección *http://127.0.0.1:8080*.

2.6.5. Seguridad.

La seguridad en una entidad es un tema primordial ya que en algunas ocasiones es necesario restringir el acceso a los reportes del servidor de Integración Continua. Xinc no cuenta con herramientas que garanticen que personas ajenas al proyecto no puedan acceder a través de la red a datos confidenciales del mismo, pero es posible utilizar el servidor Apache para manejar las solicitudes y gestionar niveles de seguridad.

Es posible gestionar el acceso al servidor Web de Apache mediante reglas. Cuando Apache recibe una petición de una página Web y antes de devolver el resultado, lleva a cabo varias acciones para verificar que la petición está autorizada. Las distintas acciones que lleva a cabo para confirmar la validez de la aplicación, se pueden agrupar en tres tipos [31]:

- **Autenticación:** La autenticación es el proceso por el cual se verifica la identidad de una persona. De una forma simple, este proceso se puede llevar a cabo mediante un nombre de usuario y una contraseña, pero se pueden llegar a utilizar otros métodos para validar la identidad de una persona, como mediante el uso de certificados, tarjetas etc.
- **Autorización:** La autorización es el proceso por el cual se verifica que un usuario con una identidad conocida, tiene acceso al recurso solicitado. Para llevar a cabo esta acción, se suelen utilizar listas de permisos en las cuales se enumeran cada una de las acciones que puede realizar un usuario, o las que no puede hacer. Normalmente, para simplificar la gestión de estos ficheros, los usuarios se suelen unir en grupos proporcionando los permisos al grupo.
- **Control de Acceso:** El control de acceso es el proceso por el cual se verifica que la máquina desde la que se ha hecho la petición, tiene acceso al recurso. Los controles de acceso se utilizan para limitar y controlar las máquinas que tienen acceso a un recurso independientemente del usuario que accede, ya que estos controles se llevan a cabo antes de que se realice el proceso de autenticación.

La autenticación en Apache puede estar gestionada por distintos módulos, dependiendo de la forma de implementación. Si decide llevarla a cabo gestionando ficheros con listas de usuarios y contraseñas, se deberá utilizar el módulo *mod_auth*.

La autorización a recursos es gestionada o bien mediante la directiva `<directory>` en el fichero principal de configuración, o bien mediante la configuración de la carpeta a través de ficheros `.htaccess`. En todo caso y para poder llevar a cabo la configuración de las tres características aquí enumeradas, autenticación, autorización y control de acceso, es necesario tener la directiva `AllowOverride` con el valor `AuthConfig`, para así permitir el uso de las distintas directivas de autenticación.

- Autorización y autenticación de usuarios.

Para configurar el servidor Apache y que este sea capaz de autenticar a los usuarios y verificar la autorización del mismo al recurso solicitado, es necesario realizar las siguientes acciones.

- Crear un fichero que almacenará los usuarios.
- Crear un fichero con grupos de usuarios (si es necesario).
- Definir las directivas en el fichero de configuración o mediante un fichero `.htaccess`.

En los ficheros de usuarios de Apache, en cada línea se especifica un usuario, escribiendo el nombre de este separado de dos puntos, seguido de la contraseña encriptada con el Algoritmo de Resumen de Mensaje 5 (MD5).

En los ficheros de grupos de Apache, en cada línea se especifica un grupo escribiendo el nombre del grupo seguido de dos puntos, y a continuación separado por espacios, los nombres de los usuarios. Es recomendable que tanto los ficheros de usuarios como los de grupos, se encuentren almacenados fuera de los directorios publicados, para que de esta forma nadie pueda descargarlos. De esta forma, sólo el súper usuario debería estar autorizado a escribir en él, mientras que solo el usuario que ejecuta el servicio Web, debería estar autorizado para leerlo.

El fichero de grupos se puede crear manualmente bajo el directorio `/opt/lampp/etc/grupos`, pero el fichero con los usuarios es recomendable crearlo mediante la utilidad `htpasswd`, que se encuentra en la carpeta con los binarios de Apache.

Para crear un fichero de usuario se utilizará la siguiente sintaxis:

```
htpasswd -c etc/apache2/usuarios usuario 1
```

La especificación `-c` se utiliza para crear un fichero nuevo, por lo que sólo se deberá poner la primera vez que se crea el fichero, sino lo borrará. Luego de ejecutar este comando será preciso introducir la contraseña dos veces.

El término `<usuario>` representa el usuario y `<contraseña_enscriptada>` es la contraseña. Luego para agrupar los usuarios en diferentes grupos, se usa un archivo que se ha creado bajo el mismo directorio que el de los usuarios y que tiene el siguiente formato:

```
usuariosAutenticados:usuario1
usuariosAutenticados:usuario2
root:administrador
```

Por último, es necesario especificar el directorio que se le aplicarán las directivas creadas, esto se hace creando un archivo llamado `.htaccess` con el contenido mostrado a continuación, y colocándose en el directorio que se quiere proteger.

```
AuthType Basic
AuthName "ServidorDeIntegracion"
AuthUserFile /etc/apache2/usuarios
AuthGroupFile /opt/apache2/grupos
```

A continuación se muestran las especificaciones de cada una de las directivas que aparecen en el archivo de la figura anterior.

- Con `"AuthType Basic"` se estará protegiendo la carpeta con autenticación básica, es decir que la clave que el usuario introduzca se transmitirá sin cifrar por la Web.
- Con `"AuthName "ServidorDeIntegracion"` se asociará esta carpeta con el dominio `"ServidorDeIntegracion"`, nombre con el que lo identificará el cliente.
- Con `"AuthUserFile /etc/apache2/usuarios"` y `"AuthGroupFile /opt/apache2/grupos"` se definirá la ubicación tanto de los ficheros de usuarios como los ficheros de grupos y se almacenarán en la carpeta `/etc`, con nombres usuarios y grupos respectivamente.

- Con "*Require group usuariosAutenticados*" se autorizará el acceso al contenido de esta carpeta a todos los usuarios que forman parte del grupo de "*usuariosAutenticados*", por lo que en la práctica se autorizará el acceso al contenido a los usuarios *usuario1* y *usuario2*.

Los módulos que intervienen en la autenticación y autorización son: *core* y *mod_auth* las directivas que se requieren de este último son las siguientes:

- ***AuthUserFile***: Se utiliza para especificar la ruta donde se almacenará el fichero de usuarios.
- ***AuthGroupFile***: Se utiliza para especificar la ruta donde se almacenará el fichero de grupos.

Las directivas de *core* necesarias para complementar la configuración del módulo son:

- ***AuthType***: Selecciona el tipo de autenticación de usuarios que se utilizará. Puede variar en dependencia del tipo de permisos que requiera un directorio. Los valores posibles son *Basic* y *Digest*. Con *Basic*, la transferencia de las claves se hará sin cifrar, y con *Digest* se harán cifradas.
- ***AuthName***: Especifica un nombre del dominio para el cual se solicita el acceso, este nombre figurará en la pantalla donde se pide la clave, y a su vez servirá para que el cliente identifique la contraseña que debe utilizar al enviar una petición a un área determinada.
- ***Require***: Selecciona los usuarios que pueden acceder a un área determinada, los usuarios se pueden determinar a través de nombres o grupos.

2.6.6. Configuración de Xinc.

Para un correcto funcionamiento Xinc permite modificar algunas opciones. En ocasiones es necesario especificarle al servidor donde debe buscar las herramientas de las cuales depende para su funcionamiento (Phing, SVN) esto puede realizarse a través del comando *xinc-settings*, el cual permite modificar las siguientes opciones:

Datos específicos sobre la configuración del servidor Xinc:

- ***version***: Versión del Xinc que se está utilizando.
- ***etc***: Directorio raíz del Xinc.
- ***etc_conf_d***: Directorio de configuración del servidor.
- ***status_dir***: Directorio donde se puede encontrar los datos que se muestran en la interfaz Web.

- **project_dir:** Directorio donde se almacenan los proyectos.
- **www_dir:** Lugar donde es almacenada la interfaz Web para ser mostrada en el servidor Web Apache.
- **www_port:** Puerto en el cual se encuentra disponible la interfaz Web en el ordenador.
- **www_ip:** Dirección de internet en la que está disponible el sitio.
- **log_dir:** Directorio en el cual se almacenan los log de estado del servidor de Integración Continua.

Xinc permite modificar algunas características de la interfaz Web, para lograr ajustar su apariencia a las empresas que lo utilizan:

- **title:** Cuando se modifica este dato es posible cambiar el texto de presentación de Xinc.
- **logo:** Esta opción permite personalizar el logo de la interfaz Web del Xinc, siempre y cuando el usuario lo desee.

Como se planteó anteriormente puede suceder el caso en que algunas de las dependencias de Xinc se instalen en directorios definidos por el usuario; en este caso es necesario definir en el Xinc donde se puede encontrar. Las dependencias que Xinc utiliza y que pueden ser configuradas son:

Directorio donde se instala el comando *phing*:

```
[phing]  
path = /<dirección alternativa>/phing
```

Directorio donde se instala el comando *svn*:

```
[svn]  
path = /<dirección alternativa>/svn
```

Es posible modificar el archivo donde es almacenada esta información sin usar el comando específico para esto. En el directorio */lib/php/data/xinc* se encuentra un archivo de texto llamado *xinc*, en el que se almacena la configuración del servidor de Integración Continua.

Principales directorio de Xinc.

Algunos de los directorios más importantes de Xinc son los siguientes:

1. ***/etc/xinc***: Directorio que almacena los archivos de configuración de Xinc.
2. ***/var/xinc***: Directorio donde están los proyectos y la información de estado.
3. ***/var/log***: Donde se archivan los log del servidor Xinc.
4. ***/etc/init.d***: En este directorio esta el archivo ejecutable que permite a Xinc correr como proceso en el ordenador.
5. ***/var/www/***: Xinc: Directorio donde es instalada la aplicación Web de Xinc.

Estos son básicamente los lugares donde se pueden encontrar los principales archivos que permiten a Xinc ejecutarse y ser configurado.

2.7. Caso de prueba.

Luego de instalar satisfactoriamente el servidor Xinc, el próximo paso a seguir será adicionar nuevos proyectos. Básicamente un proyecto en PHP desde el punto de vista de la Integración Continua está conformado por tres partes fundamentales, la primera es el código y conjunto de utilidades propias del lenguaje, la segunda son las pruebas unitarias, y la tercera los archivos de configuración. En este último pueden ser identificados dos tipos de archivos fundamentales, de configuración del Xinc y los del Phing o comúnmente llamados Buildfiles. Un proceso Integración Continua se ejecuta cada cierto tiempo (iteración), realizando un conjunto de tareas sobre el proyecto, estas son básicamente las siguientes:

- Actualización del proyecto.
- Realización del Build.
- Ejecución de pruebas unitarias.
- Despliegue.
- Retroalimentación.

Después de cada iteración el servidor de Integración Continua genera un conjunto de artefactos que son usados para documentar, retroalimentar, y determinar el grado de avance del proyecto. La siguiente figura representa una iteración, los recursos que necesita y los artefactos que son generados.

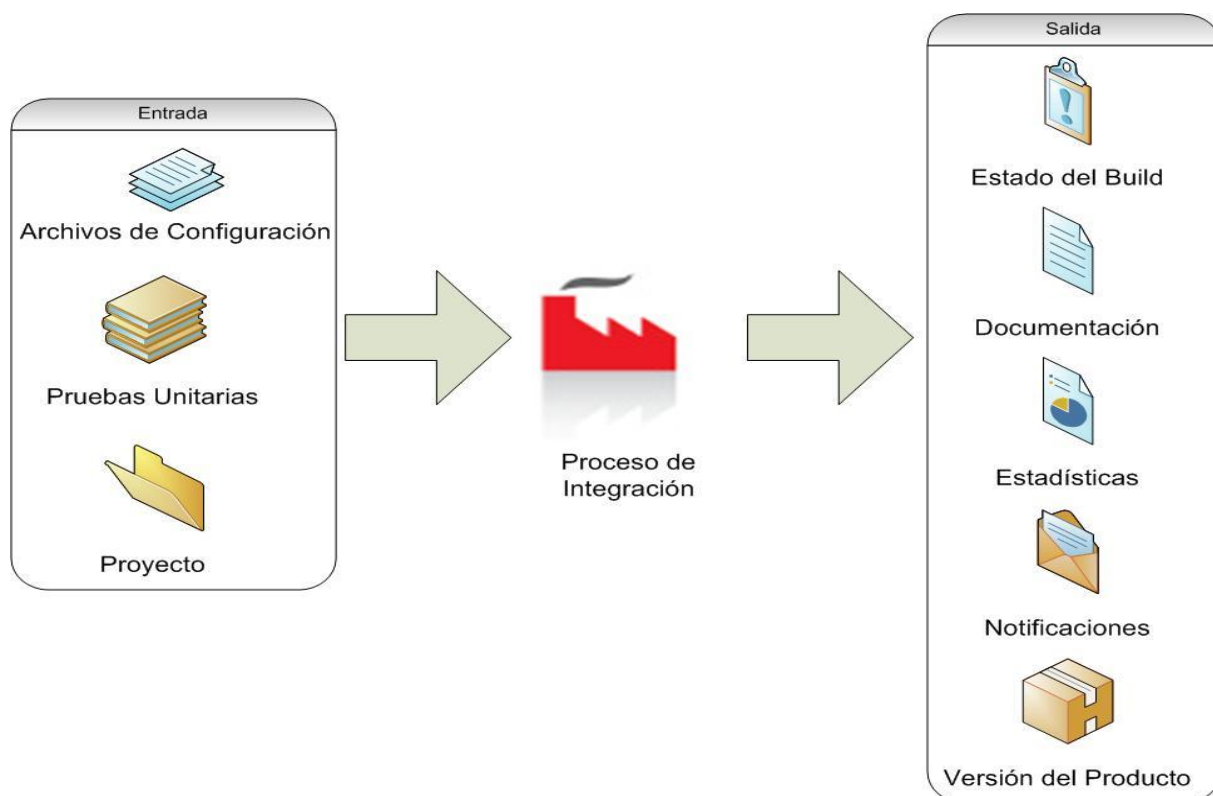


Figura 2. 4 Representación de una iteración.

Para un mejor entendimiento de cómo desplegar Integración Continua se hará uso de un proyecto escrito en PHP que consta de los siguientes artefactos básicos:

El proyecto está compuesto por tres archivos escritos en PHP:

- Page.php: es una clase que posee un método llamado *getOutput()* que devuelve una cadena.
- Index.php: se vale de la clase Page.php para mostrar lo devuelto por el método *getOutput()*.
- PageTest.php: es una prueba unitaria que valida al método *getOutput()* de la clase Page.

Todo el proyecto se encuentra bajo el directorio *Project*, esto es lo básico que un proyecto debe tener para implementar Integración Continua. A continuación se explicará con más detalles cada uno de los pasos a seguir para integrar este proyecto.

2.7.1. Pruebas Unitarias con PHPUnit.

En este punto se dará una pequeña explicación acerca de las pruebas unitarias en PHPUnit basándonos en el caso de prueba. Los pasos básicos para escribirlas para PHPUnit son los siguientes:

- Las pruebas de una clase *Class* van dentro de una clase *ClassTest*.
- *ClassTest* hereda (la mayor parte del tiempo) de *PHPUnit_Framework_TestCase*.
- Las pruebas son métodos públicos que no esperan parámetros y son nombrados *test**.
- Dentro de los métodos de pruebas, la afirmación de métodos tales como *assertEquals()* se utilizan para afirmar que el valor real coincide con un valor esperado.

A continuación se expone la prueba unitaria realizada sobre el caso de prueba que dará una idea general de cómo utilizar PHPUnit para el desarrollo de pruebas unitarias.

```
1  <?php
2  require_once 'PHPUnit/Framework/TestCase.php';
3  require_once 'Page.php';
4
5  class PageTest extends PHPUnit_Framework_TestCase
6  {
7      public function testGetOutput ()
8      {
9          $page = new Page ();
10         $this->assertNotEquals(0, strlen($page->getOutput ()));
11     }
12 }
13 ?>
```

Figura 2. 5 Código de prueba unitaria.

En toda prueba unitaria escrita para PHPUnit es necesario incluir la librería *TestCase.php* y la clase para la que se está diseñando así como cualquier otra que se necesite. Además toda prueba que usa el framework PHPUnit es una clase que extiende de la clase *PHPUnit_Framework_TestCase* para de esta forma poder utilizar métodos de validación proporcionados por el marco. En este caso la prueba *PageTest* posee un método llamado *testGetOutput()* que valida la salida del método *getOutput()* de la clase *Page*. La validación consiste en “asegurar” que el método *getOutput()* nunca devuelva una cadena de longitud cero a través de la función *assertNotEquals()* proporcionada por el framework PHPUnit.

2.7.2. Compilación Automática con Phing.

En ocasiones es posible identificar una serie de acciones que son realizadas continuamente y que son posibles de automatizar, lo cual puede ahorrar un tiempo preciado. Un ejemplo de tarea repetitiva dentro de un proyecto productivo puede ser el despliegue de aplicaciones en entornos de producción. Estas pueden conllevar a múltiples acciones que con determinada regularidad deben ser repetidas una y otra vez, algunas de estas tareas pueden ser:

- El despliegue de aplicaciones conlleva a múltiples configuraciones de archivos.
- Creación de directorios.
- Copia de archivos en directorios específicos del sistema.
- Creación de empaquetados complejos.

Como se hacía mención en el primer capítulo, Phing es un sistema de compilación automática para PHP basado en Apache Ant. La diferencia básica entre un compilador tradicional como el de PHP y uno automático como Phing, es el hecho de que este último resuelve varios problemas como el de configuración (Buildfile), ya que estos se escriben en XML el cual es un lenguaje descriptivo ampliamente extendido.

Para compilar un proyecto usando Phing es necesario especificar qué pasos seguirá mediante un archivo de configuración (Buildfile), en el cual se definirán las tareas que se realizarán durante la integración. En la siguiente figura se muestra el Buildfile del caso de prueba seleccionado.

```
1  <?xml version="1.0"?>
2  <project name="Simple Project Build File" basedir="/var/projects/project" default="build">
3    <target name="build" depends="test">
4      <move file="index.php" tofile="/opt/lampp/htdocs/project/index.php" overwrite="true"/>
5      <move file="Page.php" tofile="/opt/lampp/htdocs/project/Page.php" overwrite="true"/>
6    </target>
7    <target name="test">
8      <phpunit haltonfailure="true" printsummary="true">
9        <batchtest>
10         <fileset dir=".">
11           <include name="*Test.php"/>
12         </fileset>
13       </batchtest>
14     </phpunit>
15   </target>
16 </project>
```

Figura 2. 6 Archivo de configuración del Phing.

Dentro de un mismo Buildfile se puede tener solo un elemento Project (`<project></project>`). Como se puede apreciar dentro del elemento Project existe un conjunto de elementos Target (`<target>...</target>`) los cuales identifican cada uno de los objetivos o grupos de acciones que se realiza sobre el proyecto para cumplir determinada misión.

Un aspecto importante que se debe tener en cuenta es que puede ser que alguno de los objetivos que se quiere realizar dependa de otro, como por ejemplo: primero se tiene un objetivo que es copiar un archivo a un directorio determinado y luego otro que es crear un empaquetado del mismo tomándolo del directorio en donde fue copiado. Estas dependencias deben ser especificadas en el atributo de *Target* llamado *depends*, por tanto, no importa el orden en que los objetivos sean especificados siempre y cuando se definan correctamente sus dependencias, un ejemplo sería:

```
<target name="TaskD" depends="TaskA, TaskB, TaskC" />
```

Esto significa que *TaskD* depende de los objetivos *TaskC*, *TaskB* y *TaskA* por tanto *TaskD* no se realiza hasta que no se ejecuten sus dependencias.

Dentro de cada elemento *Target* se definen elementos denominados *Task* (tareas), estos elementos a diferencia de los objetivos poseen diferentes nombres ya que en realidad son fragmentos de código escritos en PHP que ejecutan tareas concretas. Estos códigos ejecutan determinada acción al ser invocados por Phing y cuentan con un formato específico que es:

```
<nombre atributo="valor1" atributo1= "valor 2" ... />
```

Como se puede observar estos códigos se definen con un nombre y cualquier cantidad de atributos. El número de tareas definidas difiere de una versión a otra del Phing. También es bueno recalcar que existen otros tipos de tareas que soportan tipos de datos más complejos.

A grandes rasgos lo que hace el Buildfile del caso de prueba es, primero ejecuta la prueba unitaria *PageTest* y luego despliega el proyecto. Como se puede observar Phing es capaz de realizar tanto la compilación como el despliegue con apenas una llamada a una aplicación además de otras tareas que permiten crear empaquetados de código, escribir archivos de texto, crear directorios, etc.

2.8. Adicionar proyecto a Xinc.

Para verificar que Xinc está corriendo en el sistema luego de ser instalado, se ejecuta el siguiente comando:

```
/etc/xinc/init.d/xinc status
```

Es bueno recordar que para que un proceso de Integración Continua brinde todos sus frutos, estos deben contar con pruebas unitarias que chequeen la validez de sus métodos. Para configurar un nuevo proyecto en Xinc, es necesario crear un archivo de configuración bajo el directorio `/etc/xinc/conf.d` cuyo nombre debe tener el siguiente formato: `<nombre_nuevo_proyecto>.xml`. Los proyectos deben ser colocados bajo el directorio `/var/xinc/projects/`.

El primer paso para aplicar Integración Continua al caso de prueba descrito en el punto anterior y el cual a partir de ahora será llamado *project*, es colocar el proyecto bajo un directorio predeterminado para esto, que en el caso de Xinc es `/var/xinc/projects/`.

Se crea una nueva carpeta para el proyecto, luego se coloca el contenido del proyecto bajo este directorio. El siguiente paso a seguir es configurar Xinc para que reconozca el nuevo proyecto y le realice las tareas correspondientes, para esto se crea un archivo de configuración de nombre `XincBuildFile.xml` bajo en el directorio `/etc/xinc/conf.d` con el siguiente contenido:

```

1  <?xml version="1.0"?>
2  <xinc>
3  <project name="CasopPrueba">
4    <schedule interval="120"/>
5    <configuration>
6      <setting name="loglevel" value="1"/>
7      <setting name="timezone" value="US/Eastern"/>
8    </configuration>
9    <property name="dir" value="/opt/lampp/var/xinc/projects/project"/>
10   <modificationset>
11     <buildalways/>
12   </modificationset>
13   <builders>
14     <phingbuilder buildfile="${dir}/build.xml" target="build" />
15   </builders>
16   <publishers>
17     <artifactspublisher file="${dir}/report.txt"/>
18     <onfailure>
19       <email to="ypenate@estudiantes.uci.cu"
20             subject="${project.name} build ${build.number} failed"
21             message="The build failed."/>
22     </onfailure>
23   </publishers>
24 </project>
25 </xinc>

```

Figura 2. 7 Archivo de configuración de Xinc.

Nota: Es necesario poseer conocimientos básicos sobre el lenguaje de marcado XML para poder entender la configuración anterior.

Antes de continuar con una descripción del archivo de configuración, es bueno citar algunas características esenciales que hacen válido un archivo de configuración del Xinc y es que primeramente se debe especificar la versión del lenguaje XML que se está usando ya que es posible escribir estos archivos tanto con la versión 1.0 como con la 2.0 indistintamente. Un archivo de configuración está definido solo por un elemento `<xinc>...</xinc>` y por uno o varios elementos `<project>...</project>` esto sugiere que dentro de un solo archivo de configuración es posible configurar de uno a más proyectos.

A continuación se dará una pequeña descripción de alguna de las sentencias que se exponen en el archivo de configuración anterior:

Línea 4. La etiqueta `<Schedule interval="120"/>` define cada cuántos segundos se realizará el proceso de Integración Continua.

Líneas 5 a 8. Las etiquetas `<configuration>... </configuration>` definen características como que uso de horario se usara en el proyecto entre otros.

Línea 9. La etiqueta `< property />` se utiliza para definir variables que almacenen determinados datos, esto resulta útil para hacer más entendible lo que se escribe, en este caso se crea una variable *dir* que almacena el directorio donde se encuentra el proyecto.

Líneas 10 a 12. Las etiquetas `<modificationset>... </modificationset>` son para definir cuando se realizaran las integraciones. `<buildalways/>` le dice a Xinc que siempre integre. Es probable que se quiera integrar solo cuando se realicen cambios sobre el proyecto para esto se usan otras ordenes además que hay que tener un controlador de versiones.

Líneas 14. Se ejecuta el build sobre el proyecto, usando Phing, con la propiedad *buildfile* se le dice a Phing donde esta el archivo de configuración que usara para el build y *target* que tarea ejecutara primero de las definidas dentro del Buildfile.

Líneas 16 a 23. Dentro de las etiquetas `<publishers>... </publishers>` se despliegan tareas para desplegar la retroalimentación.

Línea 17. La etiqueta `<artifactspublisher.../>` permite publicar archivos en la Web del Xinc en este caso es un archivo de texto llamado *report.txt*.

Línea 18 a 22. `<onfailure>... </onfailure/>` con estas etiquetas es posible definir acciones para el caso específico de que el build falle, que en este caso es el envío de un correo electrónico, es bueno aclarar que la funcionalidad `< email />` no funciona en la UCI por las características de los servidores de correos de la misma.

Línea 19: Una de las tareas más importante dentro de la Integración Continua es el correo electrónico, el cual facilita implementar un sólido sistema de retroalimentación que permite alertar inmediatamente a los usuarios del proyecto cuando ha fallado el build y de esta manera tomar determinadas medidas de contención (define que deben hacer los desarrolladores cuando falla el build). Lamentablemente los servicios de correo electrónicos de la Universidad no permiten implementar este mecanismo de retroalimentación.

2.9. Retroalimentación.

La retroalimentación es uno de los procesos más importantes dentro de la Integración Continua, esta permite resolver rápidamente los problemas que aparecen durante la integración del proyecto. El proceso CI propone que esta sea llevada a cabo de manera automática mediante correo electrónico, mensajería instantánea u otra.


En estos momentos Xinc tiene soporte para realizar la retroalimentación mediante el correo electrónico sobre el protocolo SMTP. Las políticas de seguridad de la Universidad no permiten implementar la retroalimentación a través de este medio.


2.9.1. Interfaz Web.


En el (Anexo 1) se puede apreciar la interfaz Web del Xinc, la misma está constituida por 3 zonas fundamentales las cuales se han enmarcado por líneas gruesas y enumeradas, estas zonas son:

- **Zona 1:** Acceso a proyectos. En esta zona se muestra un árbol de directorios que representa cada uno de los proyectos a los cuales se les está aplicando el proceso de Integración Continua, en este caso son tres proyectos. El primero denominado “SimpleProject2” y el segundo llamado “SimpleProject”, son proyectos que vienen por defecto en las versiones de Xinc a modo de ejemplos, el tercero “Caso_de_Prueba” es el caso de prueba donde se está trabajando.
- **Zona 2:** Vistas: Esta zona está diseñada para mostrar a los usuarios diferentes datos acerca de la Integración Continua, como son el estado del build, las gráficas de las estadísticas, artefactos generados durante el proceso de integración que los administradores del servidor Xinc deseen publicar. En el Anexo 1, se muestra el “Dashboard”, en este puede apreciarse el estado de cada uno de los proyectos, la versión del último build exitoso, la fecha en que se realizó la última compilación, así como enlaces que permiten acceder a información más detallada sobre el proyecto.
- **Zona 3:** Pestañas de acceso. En la zona superior derecha podemos encontrar las pestañas que dan acceso a cada una de las páginas que son abiertas, las cuales brinda gran facilidad a la hora de gestionar la interfaz.

El estado actual de la compilación se representa mediante tres tipos de iconos. Estos pueden ser vistos en el “Dashboard” a la derecha de cada proyecto e indican cuando la compilación está fallando, ha sido satisfactoria o fue interrumpida.

 **Build fallido:** Significa que existen problemas en el Build. Este icono es sinónimo de alerta ya que es producto de un problema durante la compilación o por la ejecución fallida de las pruebas unitarias. Los responsables de monitorear la Integración Continua deben tomar medidas inmediatamente que aparezca esta notificación. Es bueno decir que la misma desencadena el mecanismo de retroalimentación del servidor de CI.

 **Build Satisfactorio:** Esta notificación es opuesta al “Build fallido”. Significa que los Build se están sucediendo satisfactoriamente. Aunque esto no significa que no puedan ocurrir fallos en cualquier momento ya que esta se actualiza en cada nuevo Build.

 **Build detenido:** Esta notificación se origina cuando por algún motivo Xinc no está realizando los build sobre determinado proyecto. Estos problemas pueden darse por diferentes motivos, una mala configuración de los archivos de configuración del Xinc, puede ser una de ellas.

En la figura (véase Anexo 2) se puede apreciar el “Acceso a proyectos” en el cual se encuentra a modo de árbol de directorios, diferentes proyectos que están en el servidor. Estos son los mismos que se pueden apreciar en el “Dashboard” y si se despliegan aparece un enlace a las estadísticas.

Una de las herramientas más interesantes de Xinc y que dan una idea de cómo ha sido la trayectoria del proyecto a lo largo del desarrollo, son las representaciones gráficas de las estadísticas. Estas dan una visión del proyecto a lo largo del proceso de desarrollo desde sus inicios, permitiendo crear estudios sobre productividad, crear estrategias de desarrollo y predecir comportamientos futuros. Las estadísticas pueden ser accedidas de diferentes formas, una a través del “Acceso a proyectos”, la otra a través del “Dashboard”, donde cada proyecto cuenta con un enlace a sus gráficas correspondientes (véase Anexo 3).

En la página de gráficas estadísticas se pueden encontrar tres tipos diferentes de gráficos. El llamado Estado del Build (Build Status) es una gráfica de pastel que representa el porcentaje de build que han fallado (color azul), contra el porcentaje de build satisfactorios (color verde). Aunque este tipo de gráfico es muy representativo, solo brinda una vista global del lo que ha ocurrido a lo largo del desarrollo, no proporciona información sobre en qué momento específico han ocurrido los fallos, ni cuáles han sido las fechas en las que se detectaron más problemas.

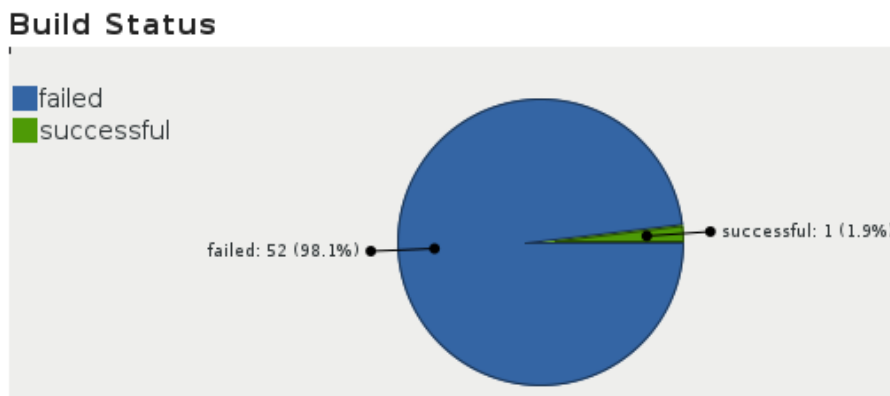


Figura 2. 8 Estado del build.

La gráfica “Duración del Build en segundos” (Build duration in seconds), representa cómo evoluciona el tiempo de duración de los builds. Esto proporciona información valiosa sobre la complejidad de las pruebas y del proyecto en general, la cual es útil para los desarrolladores ya que a través de esos datos se puede observar en qué momento han ocurrido las iteraciones de la Integración Continua. Con estos datos es posible también definir cuando se deben tomar medidas para reducir el costo de las pruebas unitarias, agrupándolas.

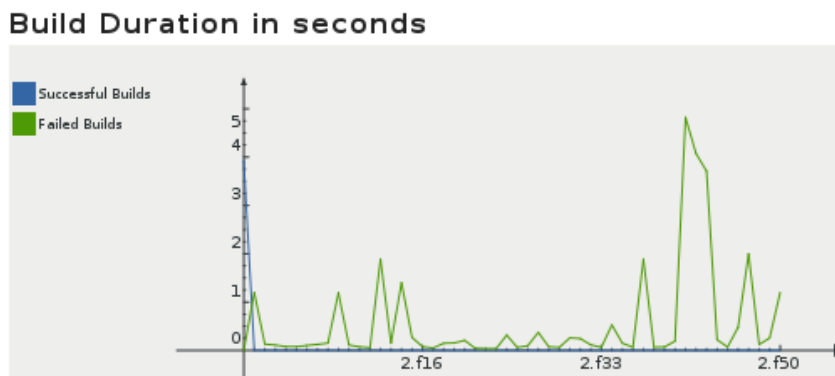


Figura 2. 9 Duración del Build en segundos.

Los usuarios de Xinc cuentan también con la posibilidad de acceder a información más detallada sobre cada uno de los Build de un proyecto, así como la posibilidad de obtener cada versión que se generó como producto de cada integración, las cuales pueden ser publicadas si así se especifica, en el archivo de configuración del Xinc. Es posible acceder a este sumario (Summary) a través del “Dashboard”, haciendo clic sobre el enlace que se encuentra en el nombre del proyecto que se desea explorar (véase Anexo 4).

En esta misma figura también se muestra la apariencia del sumario de uno de los proyectos. En este, se puede ver a la derecha, un historial de builds realizados al proyecto, bajo el nombre "All Builds". Cada versión del build consta de un icono de estado, la versión y la fecha en que se ejecutó el mismo. Si se da clic sobre algunos de estos builds, se mostrará información sobre esa compilación, en la lista que lleva por nombre "Log Message".

La lista "Log Message" muestra información sobre cada uno de los pasos que se realizaron en cada integración. En ella puede verse el orden de cada tarea que fue ejecutada, así como sus mensajes de estado correspondientes (si falló o no). A través de esta lista se pueden encontrar problemas de configuración, así como detectar cuáles pruebas unitarias y tareas han fallado y la fecha en que ocurrió.

En la figura (ver Anexo 5) se puede observar el resto de la información disponible en el sumario de cada build, en esta parte es posible identificar tres grupos de datos diferentes:

- **Modification Summary:** Muestra información sobre datos obtenidos del repositorio control de versiones como por ejemplo: cuantos archivos habían sido modificados en el momento en que se realizó el build, si fue adicionado alguno a los ya existentes, etc.
- **Change Log:** Indica datos sobre los usuarios que realizaron los cambios en el código del repositorio. En caso de que el build fallase, es posible determinar quien es el responsable mediante esta información.
- **Artifacts:** Publicación de artefactos generados por la compilación, documentación ágil, paquete con la versión del código que fue integrado.

Como es posible observar la interfaz Web de Xinc posee una gran organización que no da al traste con su flexibilidad para publicar contenido e informar acerca de los datos de las integraciones. Es posible mediante esta analizar cómo interaccionan los desarrolladores con el proyecto que se encuentra en el repositorio así como determinar los cambios que afectan el buen funcionamiento del mismo.

2.10. Conclusiones Parciales.

En este capítulo se realizó el estudio de las herramientas necesarias para crear un escenario de Integración Continua en el proyecto FGR, donde se llega a las siguientes conclusiones:

- Con el entorno propuesto se obtiene una mayor visibilidad del proyecto.
- Las herramientas seleccionadas garantizaron la introducción de esta nueva práctica.
- La instalación y configuración de estas herramientas no es un proceso difícil pero se debe seguir las secuencias de pasos propuestos en el capítulo.

De esto se pudo discernir que el servidor Xinc junto con PHPUnit y Phing es una alternativa viable para implantar la Integración Continua, no sólo en este proyecto sino en todos aquellos que tengan características semejantes.

Despliegue del entorno propuesto

3.1. Introducción.

En este capítulo se explican los detalles de cómo se aplica el proceso de Integración Continua al módulo GCPA del proyecto Fiscalía General de la República. Se analiza cómo se configuran las diferentes herramientas, utilizando un paquete de instalación alternativo propuesto por los autores. Igualmente se ven detalles sobre los diferentes procesos que se llevan a cabo durante un ciclo de integración del módulo GCPA. Además es posible analizar el funcionamiento de la generación automática de artefactos que sirven a los desarrolladores para tener una mejor visión del proyecto.

3.2. Paquete de instalación Xinc+Xampp.

A raíz del estudio del servidor de Integración Continua Xinc y de acuerdo a las características y necesidades del proyecto FGR de la Universidad de las Ciencias Informáticas (UCI), surge la idea de mejorar de algún modo la manera de aplicar este servidor; no solo a este proyecto, sino a todo aquel que tenga características semejantes. Algunos de los problemas fundamentales que fueron detectados a la hora de aplicar el servidor Xinc fueron:

- Xinc depende de una serie de herramientas las cuales se descargan e instalan por separado.
- Ausencia de un repositorio Pear en la Universidad.
- Limitada capacidad de descarga de contenido de internet del proyecto Fiscalía.

El servidor Xinc es un proyecto realizado en PHP y para su ejecución depende de una serie de herramientas las cuales se descargan e instalan por separado, estas son descritas a continuación:

- Phing, cliente subversion y el compilador PHP5.
- Requiere del PHPUnit para agregar la funcionalidad de pruebas unitarias.
- Posee una aplicación Web para visualizar reportes que necesita un servidor, en su caso particular el Apache Web Server.

Al igual que Xinc las herramientas Phing y PHPUnit están disponible en internet para ser instaladas a través de los repositorios Pear (esto no quiere decir que sea la única forma de instalarlos), el resto de las herramientas son de uso común por parte de los programadores de la Universidad, por lo que pueden ser descargadas e instaladas mediante los repositorios que están habilitados.

Ausencia de un repositorio Pear en la Universidad.

Aunque Pear se ha convertido en un estándar para muchos desarrolladores de soluciones informáticas basadas en PHP, en lo que a distribución se trata, la UCI no cuenta con un repositorio de este tipo del cual los proyectos productivos puedan servirse. Los principales problemas con los repositorios Pear es que consumen las cuentas de los usuarios que lo usan para descargar e instalar paquetes desde internet, además de depender de conexiones externas que pueden fallar en determinado momento.

Limitada capacidad de descarga de contenido de internet del proyecto Fiscalía.

Como se ha mencionado en puntos anteriores, para la instalación de Xinc y sus dependencias es aconsejable instalarlos desde los repositorios Pear de sus respectivos proyectos, esto implica utilizar una conexión a Internet y el proyecto FGR no cuenta con una cuota de descarga para esto, por tanto queda a disposición de sus miembros aportar sus cuentas de Internet, lo cual no es una opción fiable.

A partir de los problemas detectados, una posible solución puede ser crear un paquete que tenga contenidas todas las herramientas necesarias y que cumpla con las siguientes características:

- **Fácil de instalar:** Reducir al máximo el número de pasos a seguir para su instalación.
- **Portable:** Posible transportación de un lugar a otro, todo en un mismo paquete.
- **Pre configurado:** Configuración básica para un correcto funcionamiento.
- **Código abierto y protegido con copyleft:** Bajo licencia GPL o alguna de sus variantes.
- **Multiplataforma:** Que permita su ejecución en diferentes sistemas operativos ó al menos que se aplique en un principio sobre la distribución Debian.

En respuesta a todo lo planteado anteriormente se decide escoger el paquete de instalación Xampp, el cual agrupa algunas de las herramientas más utilizadas por los desarrolladores de PHP y que cumple además con los parámetros establecidos anteriormente. Xampp es un paquete de instalación que permite desplegar un entorno de desarrollo en PHP rápidamente y que cuenta con herramientas como: Apache, MySQL, PHP, entre otras. Basándose en esto los autores adicionaron las herramientas Xinc, Phing, PHPUnit a las ya existentes dentro del paquete facilitando el proceso de instalación y configuración del servidor de Integración Continua.

3.3. Estructura de Xinc+Xampp.

Xinc+Xampp está compuesto por una serie de directorios que asemejan en gran medida la estructura de archivos de un sistema Unix. Este es en un principio un conjunto de herramientas agrupadas dentro de un mismo paquete. La estructura de su sistema de archivo facilita su despliegue en múltiples plataformas sin que de una a otra existan divergencias notables (ver Anexo 6, donde es posible analizar su árbol de ficheros).

- La instalación se hace bajo del directorio */opt/* en la carpeta */lampp*, este es el directorio raíz de Xinc+Xampp.
- En el directorio */opt/lampp/htdocs/* se almacenan los sitios Web que se publican incluyendo la interfaz del Xinc.
- Bajo */opt/lampp/etc/* se encuentran los archivos de configuración de las diferentes herramientas que proporciona Xinc+Xampp como el archivo de configuración del Apache, *httpd.conf* y el del PHP llamado *php.ini*.
- En */opt/lampp/bin/* residen todos los archivos ejecutables y binarios.
- Los archivos de configuración de Xinc+Xampp se hallan bajo el directorio */opt/lampp/etc/xinc/conf.d*. Los proyectos se colocan en */opt/lampp/var/xinc/projects*.
- Las librerías de las diferentes herramientas se encuentran bajo */opt/lampp/lib/php* incluyendo las del Phing y las de PHPUnit.

3.3.1. Instalación

Para la instalación de Xinc+Xampp es preciso utilizar los siguientes comandos con permisos de súper usuario:

1. Descomprimir Xinc-Xampp en /opt:

```
$ tar xvfz Xinc-Xampp-1.tar.gz -C /opt
```

2. Darle permiso de ejecución al script de PHPUnit mediante el comando:

```
$ /opt/lampp/bin/chmod +x phpunit
```

3. Iniciar el servidor Apache y otros servicios:

```
$ /opt/lampp/lampp start
```

4. Iniciar el servidor Xinc:

```
$ /opt/lampp/lampp --xinc start
```

Estos son los comandos necesarios para instalar y iniciar el servidor Xinc y el servicio Web. Los autores adicionan nuevas funcionalidades como por ejemplo: para verificar que el demonio (daemon) de Xinc está corriendo, ejecutar el siguiente comando:

```
$ /opt/lampp/lampp --xinc status
```

Otra funcionalidad que es adicionada por los autores, es la posibilidad de que Xinc+Xampp se inicie al cargarse el sistema operativo, esto resulta conveniente ya que la PC servidor puede ser reiniciada y aun así no es necesario volver a ejecutar los comando de los pasos 4 y 5. Para configurar esta opción, se debe ejecutar el siguiente comando (esta funcionalidad ha sido probada en la distribución de Linux Ubuntu y Debian):

```
$ /opt/lampp/lampp makesresident
```

Para deshabilitar la opción anterior:

```
$ /opt/lampp/lampp unmakeresident
```

Estas son algunos de los comandos más importantes a la hora de instalar el servidor de Integración Continua usando el paquete de instalación Xinc+Xampp.

3.4. Aplicando Integración Continua al módulo GCPA de FGR.

Basándose en la solución propuesta en el capítulo 2, a continuación se exponen los pasos a seguir para realizar el despliegue del entorno de Integración Continua en el proyecto FGR.

1. Se instala y configura el Servidor de Integración Continua Xinc.
2. Se configura un nuevo proyecto.
3. Se crea un archivo de configuración del Phing (Buildfile) para el proceso de compilación automática, que define las tareas a realizar en cada integración.
4. Se adiciona un nuevo proyecto al servidor de Integración Continua y se definen que artefactos pueden ser generados.

Luego de instalar satisfactoriamente las herramientas que lleva a cabo el proceso de Integración Continua, se hace necesaria la creación de una nueva carpeta, para almacenar el código que se quiere integrar bajo el directorio `/opt/lampp/var/xinc/projects` y que se actualiza desde el repositorio. Este proceso se puede realizar de dos formas:

Utilizando la línea de comandos: El proceso de configuración de un directorio para que se actualice el proyecto desde el repositorio controlador de versiones es el siguiente.

- Desde la consola ir al directorio de proyectos del Xinc+Xampp.
- Ejecutar el siguiente comando:

```
$ svn co http://<servidor_svn>/<directorio_raiz> project
```

- Con el comando anterior se crea un nuevo directorio llamado *project*, para actualizar el contenido del mismo desde el control de versiones se utiliza el siguiente comando:

```
$ svn update
```

Utilizando el rapidSubversion: Para adicionar un nuevo proyecto con una herramienta visual como esta, se deben seguir los siguientes pasos:

1. Se debe instalar el rapidSubversion desde los repositorios de la universidad.
2. Luego se debe crear una nueva carpeta bajo el directorio *opt/lampp/var/xinc/projects* y nombrarla FGR.
3. Finalmente, mediante la interfaz del rapidSubversion, se asocia el directorio creado con la dirección del repositorio donde se encuentra el código del proyecto (este paso descarga el código en el directorio creado).

3.4.1. Análisis del Buildfile.

Para cada iteración del proceso de Integración Continua son definidos una serie de objetivos y dentro de estos un conjunto de tareas, para el caso del módulo GCPA del proyecto Fiscalía General de la República se definieron las siguientes tareas:

1. **Obtener la última versión del proyecto desde el repositorio.**
2. **Liberar el proyecto:** Dejar el proyecto listo para su despliegue.
3. **Directorio de pruebas unitarias:** Define en qué directorio los desarrolladores deben almacenar las pruebas unitarias que realizan.
4. **Crear un paquete:** Se crea un paquete que contiene la última versión del proyecto.
5. **Crear nuevamente un proyecto Symfony:** Se realiza lo opuesto al paso 1.


```

1 <?xml version="1.0"?>
2 <project name="Proyecto_Fiscalia" basedir="/opt/lampp/var/xinc/projects/FGR"
  default="build">
3   <target name="build" depends="Pruebas_Unitarias">
4     <tar destfile="fgr.tar.gz" basedir="." />
5     <exec command="symfony unfreeze" dir="/opt/lampp/var/xinc/projects/FGR" />
6   </target>
7   <target name="Pruebas_Unitarias" depends="Freeze">
8     <phpunit haltonfailure="true" printsummary="true">
9       <batchtest>
10        <fileset dir="./test/unit">
11 <include name="*Test.php"/>
12        </fileset>
13      </batchtest>
14    </phpunit>
15  </target>
16    <target name="Freeze" depends="Update" >
17    <exec command="symfony freeze" dir="/opt/lampp/var/xinc/projects/FGR" />
18  </target>
19  <target name="Update">
20    <exec command="svn update" dir="/opt/lampp/var/xinc/projects/FGR" />
21  </target>
22 </project>

```

La primera tarea definida es necesario realizarla, ya que el proyecto FGR utiliza el marco de trabajo Symfony, el cual brinda a los desarrolladores facilidades a la hora de programar, por tanto antes de ejecutar las pruebas unitarias y de crear un paquete listo para el despliegue, es necesario liberar el proyecto mediante el comando *symfony freeze*, este comando es el encargado de adicionar al proyecto todas las librerías que se utilizan en el mismo, con el objetivo de poder desplegar el proyecto en cualquier sistema sin necesidad de tener instalado el marco de trabajo Symfony. La cuarta tarea tiene como objetivo convertir nuevamente el código en un proyecto Symfony a través de comando *symfony unfreeze*.

Algunas de las tareas mostradas en la figura anterior, se explicaron en el Capítulo 2. A continuación se expone la explicación de algunas de las tareas que no han sido explicadas aún:

Línea 4 - El objetivo de esta tarea es empaquetar los archivos que se le especifican, que en este caso particular es el código del proyecto.

Línea 17 - La tarea *exec* permite al Phing ejecutar comandos de la terminal. Esto proporciona a Phing una gran ventaja ya que permite extender sus funcionalidades.

Línea 20 - Utilizando la tarea *exec* se actualiza el directorio donde está el proyecto usando el comando *svn update*.

3.4.2. Archivo de configuración del Xinc.

A continuación se muestra la sección del archivo de configuración del Xinc que permite aplicarle al módulo de GCPA del proyecto FGR el proceso de Integración Continua. Algunas de las expresiones que se pueden analizar en este archivo de construcción, ya fueron analizadas en el capítulo anterior, por tanto a grandes rasgos es posible determinar que es lo que hace:

1. Chequea el proyecto cada 20 minutos.
2. Define algunos parámetros de configuración.
3. Crea una variable *dir* y en ella almacena la dirección donde se encuentra el proyecto.
4. Compara la copia local del proyecto con la última versión del mismo y en caso de existir cambios realiza la integración, de lo contrario no realiza ninguna tarea.
5. Ejecuta el Phing utilizando el archivo de configuración definido.
6. Publica en la Web los artefactos definidos.

```
<?xml version="1.0"?>
<xinc>
.
1. <project name="ModuloFiscaliaGCPA">
2. <schedule interval="1200"/>
3. <configuration>
4. <setting name="loglevel" value="1"/>
5. <setting name="timezone" value="US/Eastern"/>
6. </configuration>
7. <property name="dir"
   value="/opt/lampp/var/xinc/projects/FGR"/>
8. <modificationset>
9. <svn directory="${dir}" update="true"/>
10. </modificationset>
11. <builders>
12. <phingbuilder buildfile="${dir}/build.xml" target="build" />
13. </builders>
14. <publishers>
15. <artifactspublisher file="${dir}/symfony"/>
16. <artifactspublisher file="${dir}/fgr.tar.gz"/>
17. </publishers>
18. </project>
</xinc>
```

Una de las tareas más significativas que aparecen en el archivo de configuración y que no fue tratada en el capítulo anterior, es *svn* en la **línea 9**. Esta permite definir que solo se realiza la integración si se comprueba que existe algún cambio en el código del repositorio (si los desarrolladores han realizado nuevos cambios) de lo contrario no sucede nada.

También es posible ver en la **línea 16** la publicación del artefacto *fgr.tar.gz* que no es más que el paquete generado por el Buildfile del Phing descrito anteriormente y que es colocado en la Web.

3.4.3. Resultados

Luego de adicionar el proyecto al servidor de Integración Continua Xinc, se probó el entorno propuesto introduciendo fallos intencionales para analizar su comportamiento ante los cambios y errores durante los primeros días de la compilación, obteniéndose los siguientes resultados de sus estadísticas:

Build Status

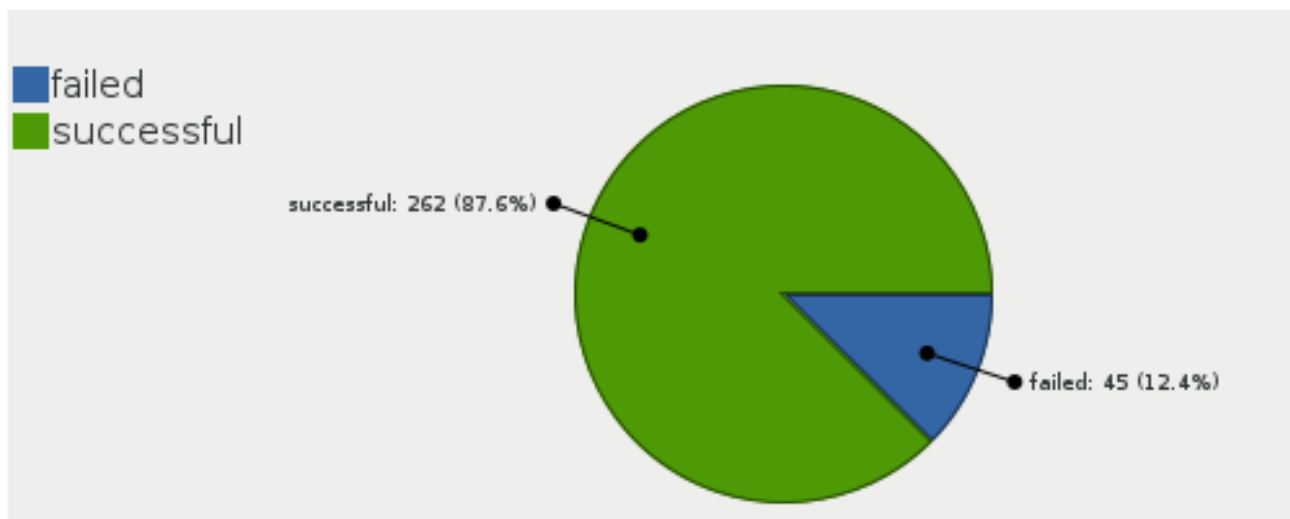


Figura 3. 1 Estado del Build.

Durante un período de 2 meses fue probado el funcionamiento del servidor de Integración Continua, el cual chequea el control de versiones en busca de cambios cada 20 minutos. Como resultado de esto se obtuvo de 307 builds para un 100%, el 87,6% satisfactorio y el 12,6% fallidos. A continuación se puede apreciar la gráfica de la evolución del tiempo de duración de los builds.

Build Duration in seconds

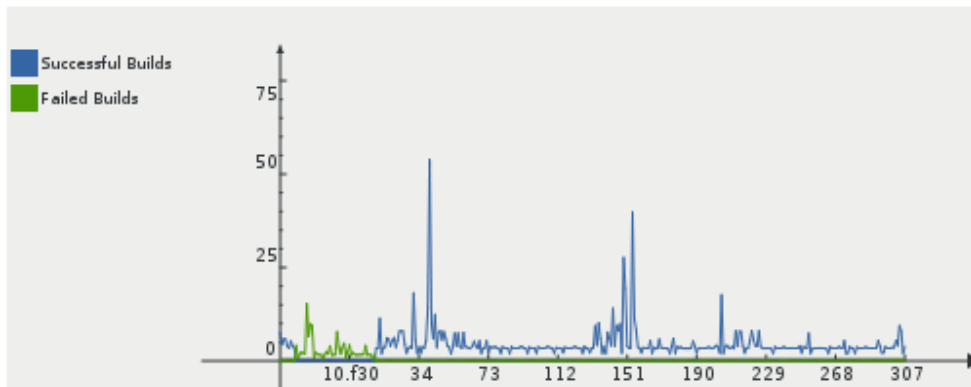


Figura 3. 2 Duración del Build.

Finalmente se puede observar cómo queda configurada la interfaz Web del Xinc. También se pueden observar datos como la hora y la fecha en que se realiza la primera integración.

The screenshot shows the Xinc web interface for 'Proyecto Fiscalía General de la Republica powered by Xinc'. The left sidebar contains a menu with 'Dashboard', 'Projects', 'CasoPrueba', and 'ModuloFiscaliaGCPA'. The main content area is titled 'Dashboard' and features a 'Projects' table. The table lists two projects: 'CasoPrueba' and 'ModuloFiscaliaGCPA'. Each project entry includes a date and time, a build ID, and a status icon. The 'CasoPrueba' project has a status icon of a green flag, indicating a successful build. The 'ModuloFiscaliaGCPA' project also has a status icon of a green flag, indicating a successful build. Below each project name, there are links for 'Statistics' and 'Graphs', with a 'NEW' badge next to the 'Statistics' link. An 'Icon Legend' on the right side of the dashboard defines the status icons: a green flag for 'Build passed', a red flag with a lightning bolt for 'Build failed', and a red flag with a lightning bolt and a stop sign for 'Build stopped'.

Project Name	Date and Time	Build ID	Status
CasoPrueba	2008-05-31 21:52:00-US/Eastern	BUILD.62	Build passed
ModuloFiscaliaGCPA	2008-04-31 21:50:54-US/Eastern	BUILD.1	Build passed

Figura 3. 3 Interfaz Web del Xinc.

3.5. Conclusiones parciales.

Después de realizar el análisis de este capítulo se obtuvieron los siguientes resultados:

- Se introdujo un paquete de herramientas que facilitó el despliegue de un entorno de Integración Continua basado en Xinc, demostrándose la alta configurabilidad del mismo.
- Se instaló y configuró un servidor de Integración Continua usando el paquete de instalación propuesto por los autores facilitando la instalación del servidor de Integración Continua.
- Aplicar el proceso de Integración Continua al módulo GCPA del proyecto Fiscalía General de la República permitió la automatización del proceso de construcción y pruebas, la introducción del uso de las pruebas unitarias y el conocimiento del estado de visibilidad del proyecto.
- El paquete Xinc+Xampp constituye la unión del procedimiento de instalación descrito en el capítulo 2 al paquete ya existente “Xampp”, garantizando portabilidad, flexibilidad, y fácil instalación.

De esta forma se dio cumplimiento al último objetivo específico planteado, terminando exitosamente el trabajo investigativo realizado.

Conclusiones Generales

A partir de los problemas fundamentales que posee el proyecto de Fiscalía General de la República de la Facultad 3 y con el fin de dar cumplimiento al objetivo general del presente trabajo se definieron los objetivos específicos de la investigación, con los que se arribó a las siguientes conclusiones:

- Se llevó a cabo el estudio del Estado del Arte de la Integración Continua, lo que permitió tener una mejor visión de los grandes beneficios que puede traer su aplicación para los proyectos productivos.
- Se realizó un análisis de algunas herramientas que permiten crear escenarios de Integración Continua para diferentes lenguajes, proporcionando datos valiosos para aquellos proyectos que deseen implantarlo.
- Se seleccionó un conjunto de herramientas de software libre que permitieron implantar un entorno de Integración Continua en el módulo GCPA del proyecto Fiscalía General de la República en el lenguaje de programación PHP.
- Se creó el paquete Xinc+Xampp basado en el entorno propuesto facilitando la instalación y configuración del servidor de Integración Continua.

Recomendaciones

- Continuar profundizando el tema de Integración Continua y aplicarlo en el proyecto de Fiscalía General de la República.
- Estudiar cómo implementar un sistema de retroalimentación en la UCI por correo electrónico o por Servicio de Mensajería Instantánea (Jabber).
- Extender las herramientas propuestas para adaptarlas a las necesidades de la Universidad.
- Continuar la investigación sobre herramientas de Integración Continua para otras plataformas.
- Desarrollar o extender una herramienta de CI para lograr independencia tecnológica.

Bibliografía

- [1]. Ahuanar, E., *Introducción a la Ingeniería de Software* 2007.
- [2]. *Metodologías Ágiles en el Desarrollo de Software* 2003.
- [3]. Patricio Orlando Letelier Torres, M.C.P., *Métodologías ágiles para el desarrollo de software: [eXtreme Programming (XP)*. 2006.
- [4]. Fowler, M., *Continuous Integration*. 2006.
- [5]. Duvall, P.M., *Continuous Integrations* ed. T. Addison-Wesley. 2007.
- [6]. Sánchez, J.P., *Integración Continua utilizando herramientas Open Source*. 2004.
- [7]. Meli, M.A., *Técnicas y Herramientas de Desarrollo Ágil para Microsoft.net framework*. 2005.
- [8]. Victoria Caballero, M.N., *Qué es un Sistema de Control de versiones?* 2008.
- [9]. Leandro Lucarella, A.B. *Introducción a los sistemas de control de versiones*. 2006.
<http://www.lug.fi.uba.ar/documentos/scms/>.
- [10]. Moreno, J.M.N.y.B.M., *Control de versiones*. 2004.
- [11]. L., F.M.C., *Uso practico de CVS para control de versiones*. 2003.
- [12]. Paz, J.D., *Desarrollo en comunidad con eXtreme Programming*. 2007.
- [13]. Ben Collins-Sussman, B.W.F., C. Michael Pilato, *Control de versiones con Subversion*. 2004.
- [14]. Ramírez, A., *Subversion*. 2004.
- [15]. *Visual SourceSafe* 2005. 2007 [cited; Available from: [http://msdn.microsoft.com/es-es/library/ms181038\(vs.80\).aspx](http://msdn.microsoft.com/es-es/library/ms181038(vs.80).aspx)].
- [16]. Hernán, S.M., *Diseño de una Metodología Ágil de Desarrollo de Software*. 2004.
- [17]. Berzal, F., *Pruebas de unidad con JUnit*. 2007.
- [18]. Juan Manuel Doderó, C.F.a.L., *JUnit: Framework de casos de prueba*. 2002.
- [19]. Bergmann., S. *PHPUnit* 2002. <http://www.phpunit.de/>.
- [20]. Bergmann, S., *PHPUnit Pocket Guide*. 2008.
- [21]. Lozano, M.A. *Automatización con Ant*. 2004. <http://www.jtech.ua.es/tutoriales/apuntes/sesion-ant-apuntes.htm>.
- [22]. *The apache Ant project*. 2007 [cited; Available from: <http://ant.apache.org/>].
- [23]. *Apache Maven Project*. 2007 [cited; Available from: <http://maven.apache.org/>].
- [24]. Andreas Aderhold, A.B., Manuel Holtgrewe, Hans Lellelid, *Phing Guide* 2004.
- [25]. *CruiseControl*. 2007 [cited; Available from: <http://cruisecontrol.sourceforge.net/index.html>].

- [26]. Duvall, P. Automation for the people: Choosing a Continuous Integration server. 2005 [cited; Available from: <http://www-128.ibm.com/developerworks/java/library/j-ap09056/index.html>.
- [27]. Apache Maven Continuum. 2007. <http://continuum.apache.org/>.
- [28]. Luntbuild - automate and manage your builds. 2006.
- [29]. Pichler, M. phpUnderControl - Continuous Integration for PHP. 2008. <http://www.phpundercontrol.org/about.html>.
- [30] Letelier, Patricio. Metodologías Agiles y XP. 2007. www.dsic.upv.es/asignaturas/facultad/lsi/doc/MetodologiasAgilesyExtremeProgramming.ppt.
- [31] Authentication, Authorization, and Access Control. 2008. <http://httpd.apache.org/docs/1.3/howto/auth.html>

Anexo 1: Interfaz Web del Xinc.

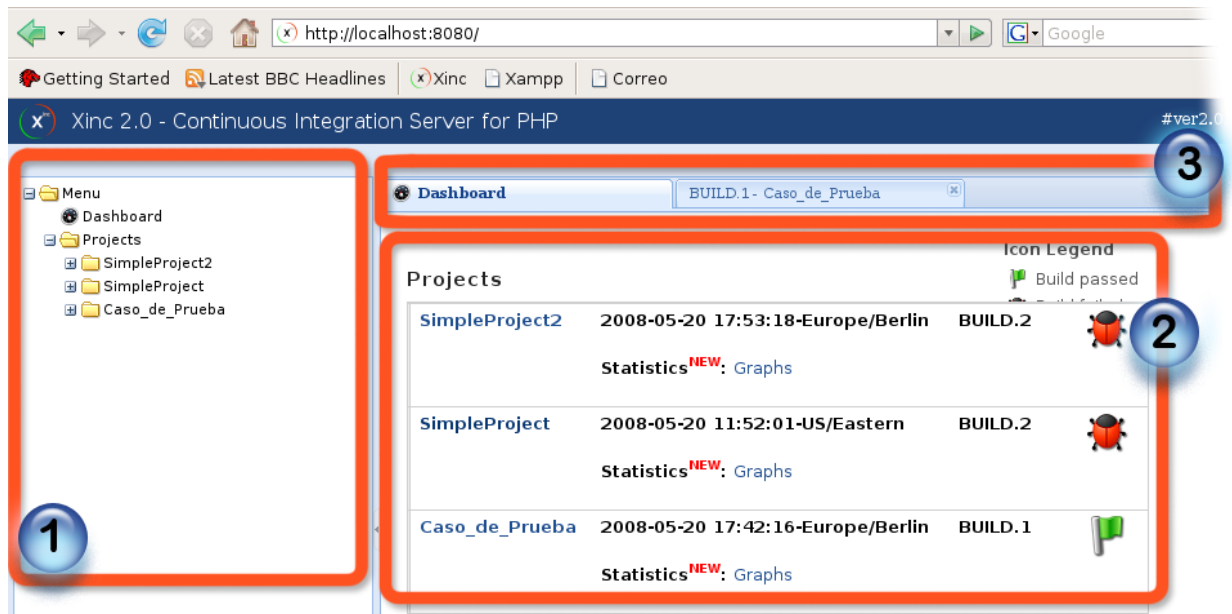


Fig. 1 Interfaz Web del Xinc.

Anexo 2: Menú de proyecto.

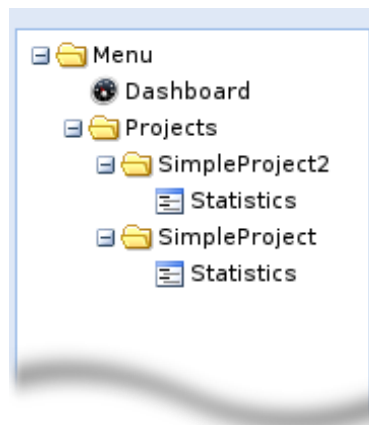


Fig. 2 Menú de proyecto

Anexo 3: Estadísticas del Xinc.

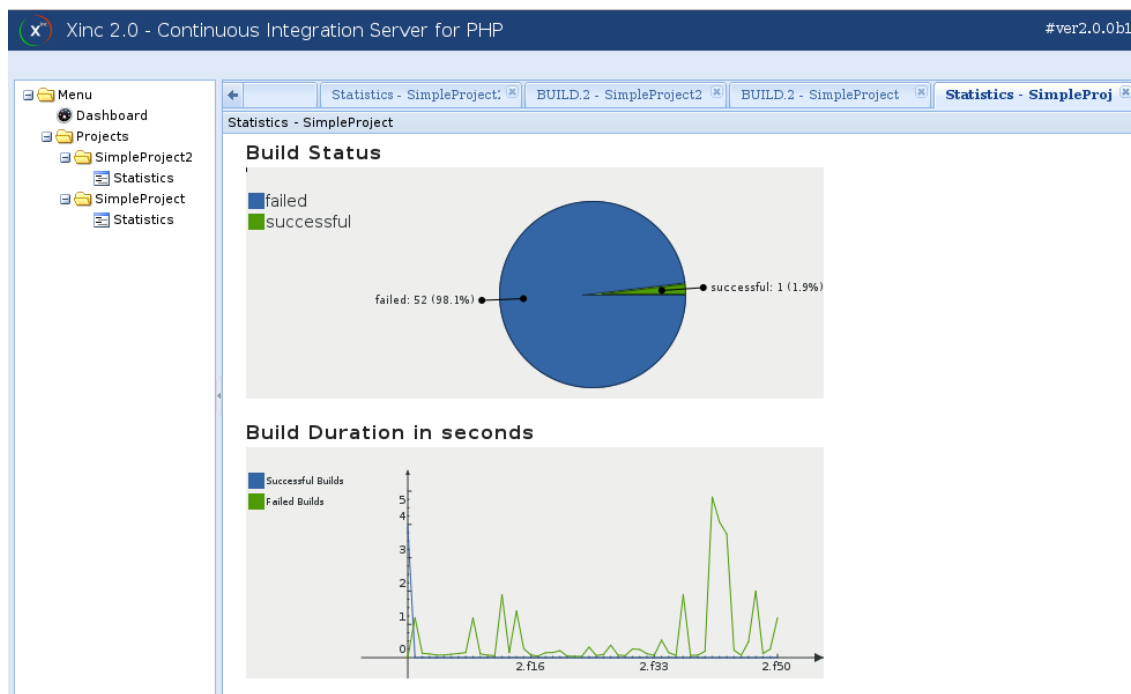


Fig. 3 Estadísticas del Xinc.

Anexo 4: Sumario 1 del Xinc.

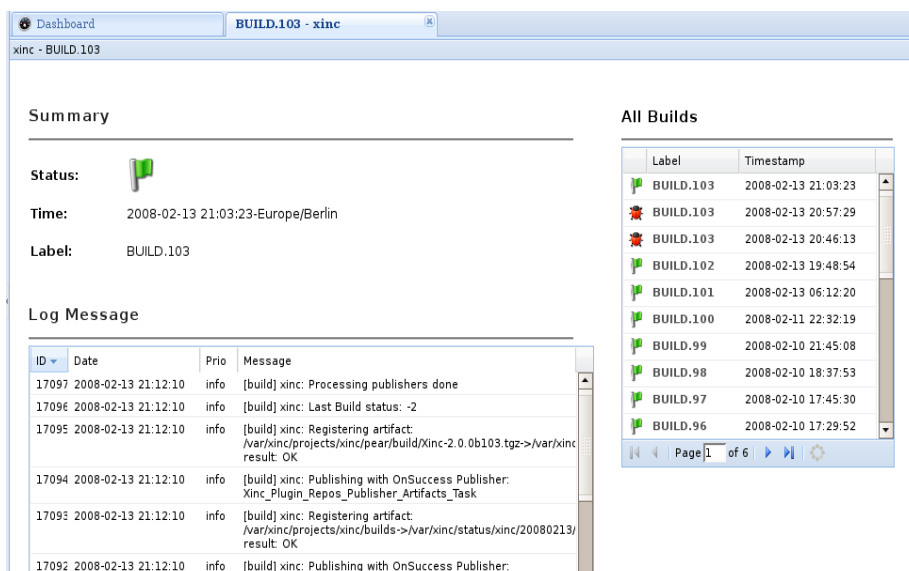


Fig. 4 Sumario 1 del Xinc.

Anexo 5: Sumario 2 del Xinc.

Modification Summary

Previous Revision: 291
Revision: 292
Files Modified: 1
Files Added: 0
Files Deleted: 0
Files Merged: 0
Files in conflict: 0

Change Log

Revision: 292	Author: arnoschn
----------------------	-------------------------

4 Issue 118: Changing request path retrieval to use REQUEST_URI if REDIRECT_URL is not set

Artifacts

- Artifacts
 - doc
 - report
 - builds
 - Xinc-2.0.0b103.tgz

Fig. 5 Sumario 2 del Xinc.

Anexo 6: Estructura de archivos de Xinc+Xampp.

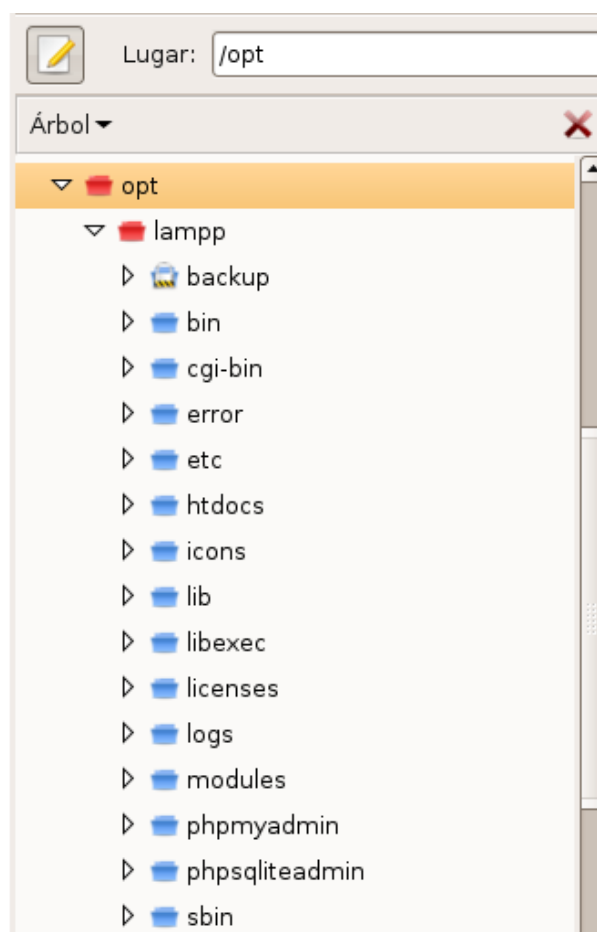


Fig. 6 Estructura de archivos de Xinc+Xampp.

Glosario

Bash: Archivo de procesamiento por lotes.

BSD: Distribución de Software Berkeley del inglés “Berkeley Software Distribution”.

Build: Resultado del proceso de compilación y empaquetado de un producto de software.

Commit: Es cuando subes los cambios del trabajo al repositorio de control de versiones.

Copyleft: Copia permitida, con la condición de que el trabajo derivado se mantenga con el mismo régimen de derechos de autor que el original.

CPL: Licencia Pública Común del inglés, “Common Public License”.

Exchange: Servidor de correo electrónico de Microsoft.

FGR: Fiscalía General de la República.

Framework: Estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

GNU/GPL: Licencia Pública General de GNU, del inglés GNU General Public License.

GNU: Es un acrónimo recursivo que significa GNU No es Unix.

GUI: Interfaz Gráfica de Usuario del inglés “Graphical User Interface”.

HTTP: Protocolo de Transferencia de Hipertexto, del inglés “HyperText Transfer Protocol”.

J2EE: Del inglés “Java 2 Enterprise Edition”. Es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java.

LGPL: Licencia Pública General Limitada, del inglés “Less General Public Licence”.

PEAR: Del inglés PHP Extension and Application Repository es un repositorio de extensiones (clases, librerías) y aplicaciones para PHP.

Plugin: Aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica.

RSS: Formato de datos utilizado para redifundir contenidos a suscriptores de un sitio Web.

Script: Guión o conjunto de instrucciones.

Shell: Interfaz de línea de comando.

SMTP: Protocolo de Transferencia Simple de Correo, del inglés “Simple Mail Transfer Protocol”.

Tracker: Trabajador encargado de monitorear el proceso de Integración Continua dentro de un proyecto de software.

X10: Protocolo de comunicaciones para el control remoto de dispositivos eléctricos.

XML: Lenguaje de Marcas Extensibles del inglés “Extensible Markup Language”.