UNIVERSIDAD TECNOLÓGICA ISRAEL

CARRERA DE SISTEMAS INFORMÁTICOS

TEMA:

"METODOLOGÍA PARA LA ADAPTACIÓN DE AGILE EN LA IMPLEMENTACIÓN DE UNA APLICACIÓN"

Trabajo de Graduación previo a la obtención del título de Ingeniero en Sistemas Informáticos

AUTOR:
Dennis Andrés Pazmiño Peña
TUTOR:
Ing. Juan Carlos Moreno Carrillo

Quito - Ecuador

2013

UNIVERSIDAD TECNOLÓGICA ISRAEL

APROBACIÓN DEL TUTOR

En mi calidad de Tutor del Trabajo de Graduación certifico:

Que el Trabajo de Graduación "METODOLOGÍA PARA LA ADAPTACIÓN DE AGILE EN LA IMPLEMENTACIÓN DE UNA APLICACIÓN", presentado por Dennis Andrés Pazmiño Peña, estudiante de la carrera de Sistemas Informáticos, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se designe, para su correspondiente estudio y calificación.

Quito, junio 2013

TUTOR

Ing. Juan Carlos Moreno Carrillo
C.C. 170639370-7

UNIVERSIDAD TECNOLÓGICA ISRAEL

AUTORÍA DE TESIS

El abajo firmante, en calidad de estudiante de la Carrera de Sistemas Informáticos, declaro que los contenidos de este Trabajo de Graduación, requisito previo a la obtención del Grado de Ingeniero en Sistemas Informáticos, son absolutamente originales, auténticos y de exclusiva responsabilidad legal y académica del autor.

Quito, junio del 2013

Dennis Andrés Pazmiño Peña
CC: 171502318-8

UNIVERSIDAD TECNOLÓGICA ISRAEL

APROBACIÓN DEL TRIBUNAL DE GRADO

Los miembros del Tribunal de Grado, aprueban la tesis de graduación de acuerdo con las disposiciones reglamentarias emitidas por la Universidad Tecnológica "ISRAEL" para títulos de pregrado.

Quito, junio del 2013

Para constancia firman:

TRIBUNAL DE GRADO

_____

PRESIDENTE

_____ _____

MIEMBRO 1 MIEMBRO 2

DEDICATORIA

A mi mamá, papá y mi hermano, quienes siempre esperaron este día, y a Diana quien siempre me dijo: Tu puedes.

Dennis

AGRADECIMIENTO

A la Universidad Israel por la oportunidad que me dio para enriquecer mis conocimientos, y a sus docentes por creer en mi.

Dennis

ÍNDICE GENERAL

CAPÍTULO II
MARCO TEÓRICO Y METODOLOGÍA

CAPÍTULO III
RESULTADOS

CAPÍTULO IV
CONCLUSIONES Y RECOMENDACIONES

C.- MATERIALES DE REFERENCIA

ÍNDICE DE CUADROS

ÍNDICE DE GRÁFICOS

UNIVERSIDAD TECNOLÓGICA ISRAEL
CARRERA SISTEMAS INFORMÁTICOS

TEMA:
"METODOLOGÍA PARA LA ADAPTACIÓN DE AGILE EN LA IMPLEMENTACIÓN DE UNA APLICACIÓN"

AUTOR
Dennis Andrés Pazmiño Peña
TUTOR
Ing. Juan Carlos Moreno

RESUMEN

Los proyectos de implementación han ido aumentando en los últimos tiempos, ya que suelen ser los primeros escalones para jóvenes ingenieros quienes empiezan a demostrar sus conocimientos, o muchas veces sus inicios en el mercado laboral. Sin embargo cuando no se cuenta con un equipo de trabajo numeroso, por falta de recursos económicos o porque los requerimientos de la implementación no lo necesita; se improvisa la ejecución del proyecto, dejando un cliente no satisfecho.

Esta investigación ha elaborado una metodología para adaptar el método Agile, en la implementación de aplicaciones, demostrando que por mas pequeño y simple que llegue a ser un proyecto, puede seguir una metodología y al final entregar un producto de calidad. Este trabajo ha sido evidenciado con la implementación de una aplicación para una pequeña empresa que tenía un requerimiento particular, esta implementación se logró siguiendo la metodología producto de esta investigación.

PALABRAS CLAVE: Implementación, metodología, calidad, producto, recursos, requerimientos, desarrollo.

UNIVERSIDAD TECNOLÓGICA ISRAEL
COMPUTER SYSTEMS CAREER

TOPIC:

"AGILE ADAPTATION METHODOLOGY FOR AN APPLICATION DEPLOYMENT"

AUTHOR

Dennis Andrés Pazmiño Peña

TUTOR

Ing. Juan Carlos Moreno

ABSTRACT

Implementation projects have been increasing in recent times, as they are often the first steps for young engineers who are beginning to show their knowledge, or many times its beginnings in the labor market. However when there is not a large team, for lack of funding or because the requirements of the implementation do not need it, you improvise the project, leaving a dissatisfied customer.

This research has developed a methodology to adapt the Agile method in application deployment, showing that by small and simple that it becomes a project, you can follow a methodology and ultimately deliver a quality product. This work has been demonstrated with the implementation of an application for a small business that had a particular requirement; this implementation was achieved following the methodology of this research product.

KEYWORDS: Implementation, methodology, quality, product, resources, requirements, development.

Introducción

El desarrollo o implementación de herramientas informáticas en el país, ha tenido mucho auge en los últimos tiempos, y este fenómeno se ha dado por la falta de control que las empresas o instituciones han estado experimentando con sus procesos.

Estos procesos luego de pasar por una etapa de regularización y estandarización, pasan a la mano de un arquitecto de software, quien sugiere el desarrollo o implementación de una solución, que permita mantener los procedimientos en correcta ejecución y control.

El principal problema de la empresa o institución hasta el momento parece estar resuelto, sin embargo esta nueva fase, en la que se sugiere el desarrollo o implementación de un herramienta informática, debe llevarse a cabo con una metodología eficiente, que permita al equipo de desarrollo o implementación organizar el proyecto, para así entregar un producto de calidad en un tiempo estimado, satisfaciendo los requerimientos del usuario final, que en este caso pasa a ser nuestro cliente.

Es ahí donde el siguiente problema aparece, y que se da en el lado técnico de la solución, ya que las empresas pequeñas que empiezan a emprender negocios informáticos o inclusive jóvenes emprendedores, carecen de todo un equipo necesario para la ejecución de una correcta metodología de desarrollo o implementación.

Las metodologías para desarrollo o implementación ágiles, han estado introduciéndose en el mercado de forma satisfactoria, ya que permite al equipo avanzar no solo de forma rápida si no, de forma ordenada y con la facilidad de realizar cambios sin impactar en el trabajo realizado.

CAPÍTULO I
EL PROBLEMA

Tema

Metodología para la adaptación de *Agile* en la implementación de una aplicación.

Línea de investigación con la que se relaciona

Implementación de aplicaciones informáticas basadas en metodologías estructuradas y heurísticas para su correcta planificación y ejecución dentro de un contexto individual.

- ¿Qué tipo de metodologías existentes ayudan a la implementación de proyectos de software para jóvenes emprendedores?
- ¿Las metodologías de implementación actuales, ayudan a guiar un proceso, para que sea planificado y ejecutado por una sola persona?
- ¿Qué elementos o documentos son necesarios para organizar un plan de trabajo para la implementación de una aplicación?
- ¿Cuál es el tiempo adecuado para la planificación, organización, ejecución y entrega de un proyecto de software?
- ¿Realmente es necesario el aplicar un criterio de planificación para realizar una implementación exitosa usando como recurso humano solo una persona?

Planteamiento del problema

¿Por qué no se puede seguir una metodología de implementación en proyectos pequeños?

El principal problema de los emprendimientos que parecen ser pequeños y de rápida implementación, es que no siempre se alinean a metodologías estructuradas que les permiten una mejor práctica y ejecución, dejando todo a la improvisación y a la costumbre, obteniendo como resultado, tiempos prolongados de implementación, requerimientos no soportados, errores en código y lógica, falta de documentación.

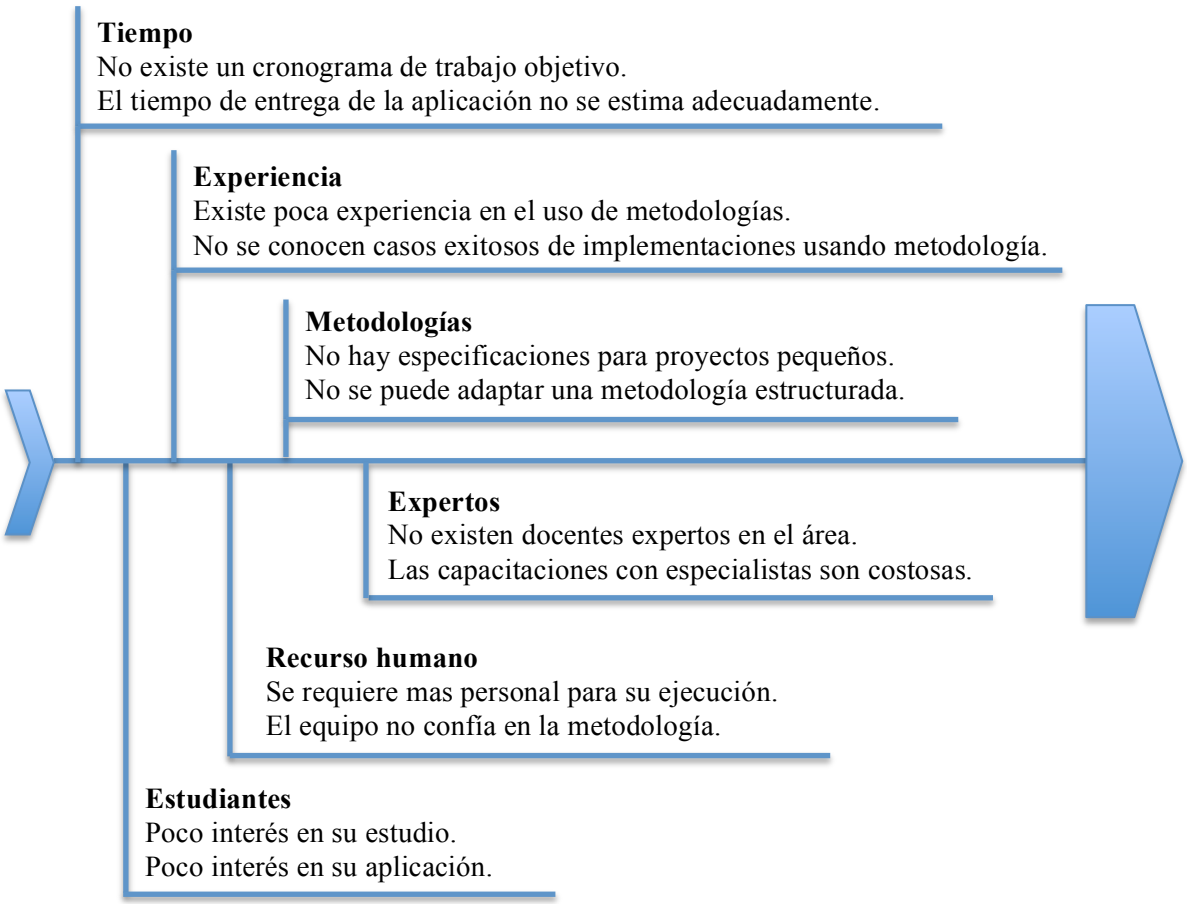


Gráfico Nº 1: Diagrama causa-efecto, primera pregunta.

Fuente: Dennis A. Pazmiño P.

¿Por qué un proyecto pequeño de implementación falla?

Los problemas que se dan con las implementaciones cortas, es que no suelen ser pequeñas como originalmente se las pensó, existe improvisación, falta de documentación y una mala ejecución de metodología que termina en una solución mal implementada, un cliente inconforme, y una esclavitud permanente con el proyecto, porque siempre se lo tiene que parchar, cubriendo requerimientos que nunca estuvieron claros o se mal interpretaros en el momento del desarrollo.

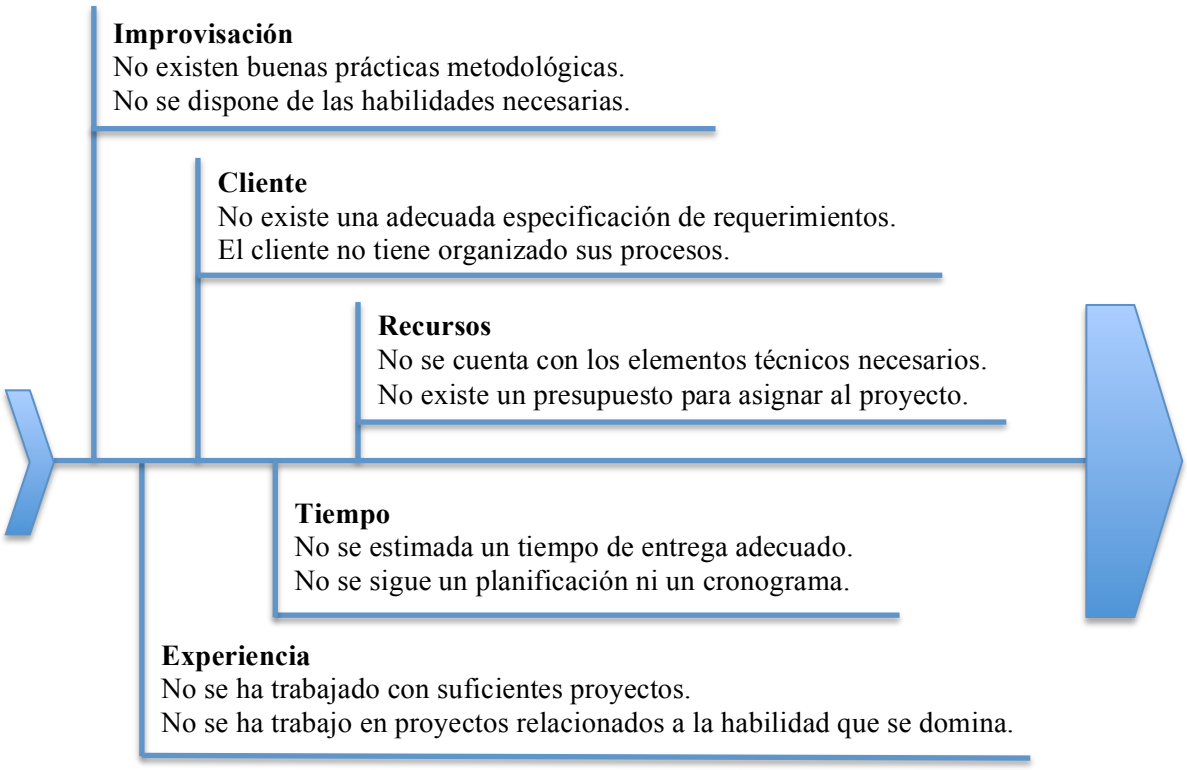


Gráfico Nº 2: Diagrama causa-efecto, segunda pregunta.

Fuente: Dennis A. Pazmiño P.

El desconocimiento de las metodologías que se ajusten a proyectos pequeños, crea una mala información respecto a ellos, ya que cuando se habla de metodología, siempre se piensa en un recurso humano numeroso, tiempo a invertir, proyectos con costos elevados, y todo eso como resultado; dinero a invertir.

Es importante tener claro lo que se quiere hacer, delimitar un alcance, y documentar todo en la medida que sea posible.

Contextualización

Las metodologías de desarrollo e implementación han existido desde los años 60 y estas han ido cambiando a medida que los requerimientos han ido variando, sin embargo, estos cambios no han sido tan sustanciales, ya que los nuevos emprendimientos se dan de forma individual.

El origen de estos procesos ordenados para guiar un proceso de desarrollo u implementación inicia en la década de los 60, donde grande empresas se inclinan por la automatización de sus procedimientos, basados en una estructura metodológica y su ciclo de vida.

La metodologías y modelos han sido creados para trabajar de forma independiente o de forma conjunta y los hay de toda índole, y cada una de ellas se las puede ubicar dentro de un contexto o proyecto, entre las más populares se encuentran las siguientes:

- **Modelo en Cascada**

  Es un modelo básico en el cual radica la finalización de una etapa determinada antes de iniciar una nueva. Este modelo contemple las siguientes etapas: Requisitos, diseño, implementación, verificación y mantenimiento.

- **Modelo Basado en Prototipos**

  Este modelo basa su comunicación con el cliente mediante prototipos que sirven para clarificar los requerimientos y funcionalidad.

- **Modelo Incremental o Evolutivo**

  La premisa de este modelo es mantener el proyecto, si ser retrasado por defectos o imprevistos, este avanza y en su marcha cambia y los corrige, minimizando su ciclo de vida.

- **Modelo Espiral**

  Este modelo establece sus etapas en una forma circular en la que se inicia desde el interior, pero este no sigue una prioridad, estas se eligen teniendo en cuenta el riesgo.

- **Modelo Orientado a Objetos**

  Esta técnica se caracteriza por la delegación de responsabilidades en base a los modelos de construcción de la aplicación.

- **Modelo Cascada con Sub-Proyectos**

  Este modelo tiene parte de la metodología en cascada, con el atenuante que en cada etapa se realizan proyectos definidos que al finalizar se evalúan.

- **Modelo Entrega por Etapas**

  En este modelo, el cliente evidencia el desarrollo y evolución de la aplicación con pequeñas entregas programadas, esto permite al desarrollador ir afinando detalles en cada entrega.

- **Modelos Ágiles**

  Los modelos ágiles basan el ciclo de vida del desarrollo del proyecto en pequeñas cascadas, que poco a poco se van elaborando y a la vez corrigiendo defectos. Estos modelos suelen tener parte de los anteriores modelos vistos.

Estos métodos se apoyan en herramientas adicionales como *UML*[1], que mediante diagramas de análisis y especificación, ayudan al equipo a entender lo que se quiere hacer, el tiempo estimado, los recursos necesarios y la forma en la que información y resultados deben moverse dentro del proceso.

Los jóvenes emprendedores no pueden seguir una metodología de implementación ya que los recursos con los que se cuentan en un inicio, no son los apropiados, sin embargo adaptar una de estas metodologías podría permitir al ejecutor entregar un producto de calidad y de forma profesional, sin retrasos o quejas por parte de su cliente.

Análisis Crítico

¿Por qué ninguna metodología de implementación existente contempla su ejecución con una sola persona en el equipo?

[1] *Unified Modeling Language*. Marzo 23 del 2013. http://www.uml.org

Si intentamos responder la anterior pregunta, tal vez se concluya fácilmente, que la ejecución de un plan de implementación obedece a un trabajo de equipo organizado, sin embargo, ahora se ven proyectos pequeños en los cuales solo una persona los lleva a cabo, no es por falta de recursos, si no porque no suelen ser proyectos grandes o importantes.

No porque una implementación sea pequeña, o la realice una sola persona, quiere decir que no tenga calidad o que sea un implementación sujeta a fallos y parches. Cuando un proyecto se ejecuta con un plan metódico y organizado, es posible minimizar los efectos negativos que puedan darse.

¿Por qué los pequeños emprendimientos no se guían por una metodología de implementación?

Las conocidas y exitosas metodologías que se usan en proyectos, no son adaptadas a iniciativas con recurso humano limitado, porque justamente su enfoque es distinto, pero es un tema discutible, ya que lo importante no es adaptar una metodología para el uso de una sola persona, si no, recabar información de todas las metodologías existentes y utilizar una sola de ellas para idear un plan objetivo para un proyecto.

Cuando la ejecución de una metodología se la realiza en equipo, la carga de trabajo se divide en función de las habilidades de los integrantes, al delegar el proyecto de cierta forma el líder del proyecto debe tener el máximo control de las actividades que se están llevando a cabo y como su equipo va avanzando. Esta modulación requiere estrategia, debido a que la responsabilidad que cada persona del equipo debe asumir, tanto en cumplimiento de tiempo de entrega, código limpio libre de errores, pruebas realizadas y corrección de defectos; tiene su tiempo y complejidad y ninguno de estos elementos debe fallar.

Prognosis

Las empresas jóvenes, luego de un tiempo, en el cual se ven maduras y con objetivos sostenibles, tienen a mejorar sus procedimientos, ya que esto no solo les da prestigio ante sus competidores si no también credibilidad frente a sus clientes.

Esta es una gran oportunidad para aquellas empresas de consultoría de software, desarrollo o implementación, de igual manera pequeñas, maduras pero con amplio crecimiento, para captar estos proyectos y llegar con propuestas frescas a clientes de bajo presupuesto.

Estas empresas informáticas no cuentan con todo un equipo para ejecutar una metodología de implementación, sea en cascada, *RUP*[2] o alguna otra de las conocidas. Las metodologías ágiles de desarrollo o implementación suelen ser la alternativa de primera elección para proyectos cortos, relativamente fáciles, de poco presupuesto y poco personal para su puesta en marcha.

Las metodologías no son aplicadas correctamente, se utiliza parte de ellas, parte de su enfoque o incluso parte de sus elementos para guiar un proceso, lo cual esta totalmente errado, ya que cada metodología tiene su principio y fin de ciclo y este se encuentra muy bien delimitado y definido. La aplicación de cualquier método tiene que ser de forma comprometida e integral.

Este conocimiento no necesariamente puede ser aplicado a pequeñas empresas que buscan entregar un producto de calidad, si no para cualquier joven ingeniero desee utilizar una metodología para entregar sus productos.

[2] *IBM Rational Unified Process (RUP)*. Marzo 25 del 2013, http://www.ibm.com/software/awdtools/rup

Delimitación del objeto de investigación

Esta investigación estará limitada a la entrega de un manual en el cual se detalle la forma de adaptar *Agile* a proyectos de implementación con una sola persona. Este manual explicará la forma en la que se debería establecer un cronograma objetivo de trabajo, los documentos necesarios, la forma en la que la comunicación con el cliente debe realizarse.

Este producto, no dará explicación de cómo ejecutar la metodología paso a paso, ya que ningún proyecto de implementación o cliente es igual a otro, la metodología deberá aplicarse al buen criterio de la persona que la ejecute basándose en la complejidad y tiempo estimado de complejidad de la aplicación.

Se realizará una investigación en la que se compare las metodologías mas usadas para la implementación de un software, y con esto identificar el porque *Agile* es la metodología que mejor podría adaptarse al trabajo de una sola persona.

Justificación

La metodología *Agile*, define principalmente cuatro roles, con los cuales un proyecto pueden empezar a ejecutarse. Estos roles son:

- *Product Owner*
- *Scrum Master*
- *Scrum Team*
- *Stakeholders*

Estos roles, se comunican en todo el proceso, diariamente y a toda hora, manteniendo una dinámica de colaboración, sin embargo, *Agile* mantiene un proceso estándar para la ejecución de la metodología que reza.

Este proceso y roles definidos, podrían ser modificado, en la medida en la que los requerimientos de un proyecto no sean grandes y el proceso de implementación sea relativamente sencillo, una o dos personas son quienes lo van a realizar, y por esta limitante, no puede ser aplicado *Agile* en su totalidad.

Sin embargo, los documentos que la metodología *Agile* utiliza para guiar el proceso, pueden ser utilizados y adaptados a un proceso de implementación que incluya a una sola persona involucrada en un proyecto, apoyándonos en un cronograma para su adecuada ejecución.

Objetivo General

Definir una metodología de adaptación individual de *Agile*, para la implementación de un sistema de recursos humanos, que servirá de modelo para el propósito.

Objetivos Específicos

- Analizar los elementos que conforman la metodología *Agile*.
- Determinar las ventajas del uso de la adaptación de la metodología *Agile* propuesta sobre la metodología RUP.
- Analizar la metodología ágil *Extreme Programming* como factor competidor frente a *Agile*.
- Aplicar la técnica empírica de las entrevistas a profesionales dedicados a la programación e implementación de proyectos, sobre el tema de metodologías.
- Verificar la consistencia y factibilidad de la propuesta con los especialistas de *Agile* de *IBM* Ecuador.

- Seleccionar un sistema de recursos humanos de mayor versatilidad y robustez para evidenciar la aplicación de la adaptación de la metodología.
- Determinar las características, requerimientos y procedimientos de puesta a punto del servidor de aplicaciones con el sistema operativo *CentOS*.

CAPÍTULO II
MARCO TEÓRICO Y METODOLOGÍA

Antecedentes investigativos / Estado del arte

Un estudio realizado en el 2011 por MSc(c). Javier Mogollón Afanador y MSc. Luis Alberto Esteban Villamizar de la Universidad de Pamplona en el Norte de Santander, en Colombia[3], afirma; luego de una encuesta realizada a un grupo de jóvenes desarrolladores sobre sus tendencias en cuanto a la elaboración de proyectos. Los resultados que arroga la encuesta son bastante interesantes y aportan en gran parte al objeto de investigación, ya que reflejan un nuevo aspecto a resaltar dentro de los proyectos de actualidad y es el desarrollo de aplicaciones utilizando una metodología híbrida, combinando la filosofía y documentos de otros modelos utilizados, para crear una "nueva" guía y así proveer un plan de trabajo.

En la Universidad Mayor de San Simón en Bolivia, el estudiante José Fernando Díaz Luizaga, ha elaborado como proyecto de grado un diseño de metodología para el trabajo de una sola persona combinando las metodologías *Extreme Programming* y *Scrum* brindando a los jóvenes desarrolladores una alternativa con la cual inclinarse en la elaboración de proyecto pequeños.

Las conclusiones de estos estudios son muy favorables, sin embargo se menciona que la combinación de las metodologías tienen validez, siempre y cuando estas obedezcan un orden, control y sobre todo, que se apliquen correctamente ya que

[3] MSc(c). Javier Mogollón Afanador, MSc. Luis Alberto Esteban Villamizar, El Desarrollo Individual De Proyectos De Software: Una Realidad Sin Método, ISSN: 1692-7257 - Volumen 1, 2011.

cuando estas no se alinean a la filosofía de trabajo original, los resultados esperados no son los mismos

Fundamentaciones

Es importante recalcar siempre que uno metodología siempre será una guía, una forma en la que un proyecto puede ser llevado con orden y control, pero sobre todo, entendiendo al cliente, llevando sus necesidades a un plano en el que ambas partes se sientan satisfechas y se logre el equilibro para poder trabajar con total seguridad.

Una implementación de software por mas pequeño que este sea, no puede pasar sin contemplarse requerimientos, ya que el usuario necesita algo, que no suele ser lo entregado. La comunicación cuando no existe una metodología se hecha a perder y es por eso que los programadores generalmente siguen trabajando luego de que la aplicación se encuentra en producción, ellos siguen elaborando parches que completan los requerimientos reales de la aplicación o en el peor de los casos, resuelven errores que nunca se vieron en el ambiente de desarrollo o, que el programador nunca contempló por desconocimientos del negocio.

El cliente no tiene los mecanismo para realmente expresar sus necesidades, ya que a veces no sabe de lo que existe en el mercado, sin embargo, cuando se discute con él sobre lo que se va a realizar, debido a su desconocimiento, el programador termina elaborando algo que el cliente no solo no entiende, si no que no era lo que necesitaba.

Marco conceptual

Para poner en marcha la adaptación de una metodología es necesario conocer lo que hacen las otras metodologías usadas con frecuencia, incluso verificar cual de

ellas es la ideal para adaptar a un proceso de implementación con una sola persona. Entre las mas conocidas se encuentra *RUP*, ahora reestructura luego de su adquisición por *IBM Corporation* en el 2003.

Metodología RUP

IBM define a *RUP* como "… un proceso de ingeniería de software. Proporciona un enfoque disciplinado para la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga las necesidades de sus usuarios finales, dentro de un horario predecible y presupuesto."[4]

Esta metodología mantiene seis principios, los cuales *IBM* define de forma muy clara y las destaca como prácticas que "… se centran en torno a un conjunto básico de principios que *IBM* ha encontrado caracterizar al software más exitoso del mundo y las organizaciones de sistemas:

- Adaptar el proceso.
- Equilibrar las prioridades de las partes interesadas.
- Colaborar entre los equipos.
- Demostrar valor iterativamente.
- Elevar el nivel de abstracción.
- Centrarse en la calidad."[5]

Este proceso se encuentra dividido en cuatro fases, al final de cada una se tendrá que exponer sus objetivos definidos y alcanzados para así dar por terminada la fase y continuar con la siguiente. Estas fases de definen de forma gráfica.

[4] *IBM Corporation, Rational Unified Process Best Practices for Software Development Teams, Rational Software White Paper TP026B, Rev 11/01*, 1998.
[5] *IBM Corporation, Rational Unified Process, RAD10971-USEN-00*, 2003.

Gráfico Nº 3: Fases de la metodología *RUP*.

Fuente: Philippe Kruchten, *A Rational Development Process*

El producto final podrá tener una siguiente versión o generación, si es que el desarrollo se detiene o se realizan cambios en los requerimientos, esta etapa se la conoce como Evolución. Cuando una nueva generación del producto es requerida, se inicia un nuevo ciclo, repitiendo las cuatro fases mencionadas anteriormente.

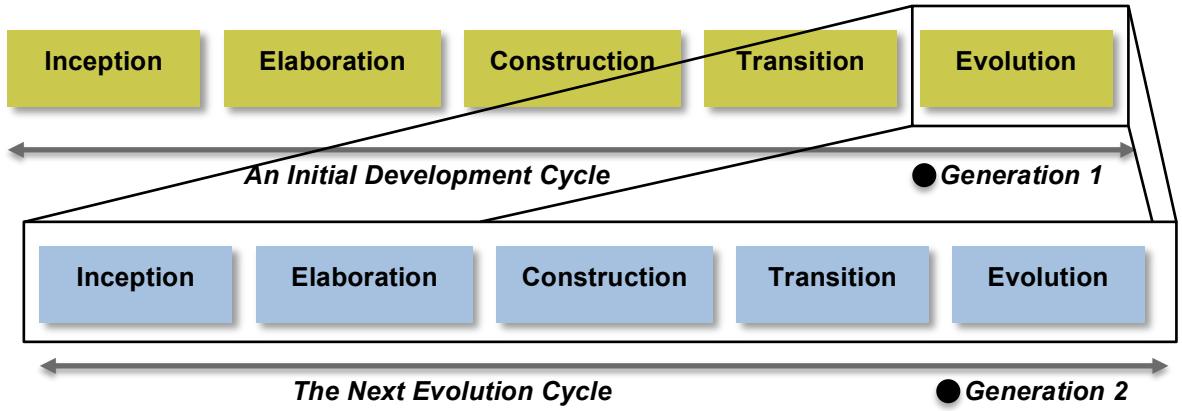


Gráfico Nº 4: Nueva generación en *RUP*.

Fuente: Philippe Kruchten, *A Rational Development Process*

Cada fase se encuentra muy bien definida, se sabe muy bien lo que se debe hacer, y se tiene claro el objetivo final de cada una de ellas.

Inception Phase / Fase inicial

En esta fase inicial es importante delimitar el alcance del proyecto, su modelo de negocio y los factores que de forma externa interactuarán con él, llamados actores.

También se debe identificar los casos de usos, sus especificaciones, y elaborar todo tipo de evaluación de riesgo, recursos tecnológico y humano; así como también un cronograma donde se detalle con fechas la finalización de cada uno de los hitos importantes que se hayan definido.

En el documento de *IBM* se definen los resultados que deberían obtenerse al finalizar esta fase:

- "Una visión del documento: una visión general de los requisitos del proyecto básico, características clave y restricciones principales.
- Un primer modelo de casos de uso (10% -20%) completo.
- Un glosario inicial del proyecto (opcionalmente puede estar parcialmente expresado como un modelo de dominio).
- Un modelo de negocio inicial, que incluye el contexto empresarial, criterios de éxito (ingresos proyección, mercado
- reconocimiento, y así sucesivamente), y las previsiones financieras.
- Una evaluación inicial de riesgos.
- "Un plan de proyecto, mostrando fases e iteraciones.
- Un modelo de negocio, si es necesario.
- Uno o varios prototipos."[6]

Toda fase debe tener una evaluación, es por eso que esta metodología de igual manera define criterios puntuales que deben ser revisados.

- "Interesados concurrencia en la definición del alcance y las estimaciones de costo / horario.
- Requisitos para comprender como se evidencia por la fidelidad de los casos de uso principales.
- La credibilidad de las estimaciones de costo / calendario, prioridades,

[6] *IBM Corporation, Rational Unified Process Best Practices for Software Development Teams, Rational Software White Paper TP026B, Rev 11/01*, 1998.

riesgos, y el proceso de desarrollo.

- La profundidad y amplitud de cualquier prototipo arquitectónico que se desarrolló.
- Los gastos reales frente a los gastos previstos.
- El proyecto puede ser cancelado o considerablemente repensarse si no logra pasar este hito."

***Elaboration Phase* / Fase de elaboración**

En esta fase se debe principalmente, analizar el dominio del problema, establecer una correcta arquitectura, desarrollar el plan elaborado en la fase anterior y eliminar los elementos que generen mayor riesgo al proyecto.

Los resultados de esta fase son los siguientes:

- "Un modelo de casos de uso (al menos el 80% completo) - todos los casos de uso y actores han sido identificados, y la mayoría de las descripciones de casos de uso se han desarrollado.
- Los requisitos suplementarios, capturar los requerimientos no funcionales y cualquier requisito que no están asociados con un caso de uso específico.
- La descripción de la arquitectura de software.
- Un prototipo de la arquitectura ejecutable.
- Una lista revisada de los riesgos y un modelo de negocio revisado.
- Un plan de desarrollo de todo el proyecto, incluyendo el plan de proyecto, mostrando las iteraciones y los criterios de evaluación para cada iteración.
- Un caso de desarrollo actualizado que especifica el proceso que se utilizará. Un manual de usuario preliminar (opcional)."[7]

Para la evaluación de esta fase, se definen varias preguntas que deberán ser respondidas.

[7] *IBM Corporation, Rational Unified Process Best Practices for Software Development Teams, Rational Software White Paper TP026B, Rev 11/01*, 1998.

- "¿Es la visión del producto estable?
- ¿Es la arquitectura estable?
- ¿El ejecutable de demostración muestra que los elementos de riesgo han sido abordados y se muestran con credibilidad?
- ¿Es el plan para la fase de construcción lo suficientemente detallado y precisa?
- ¿Es una copia de seguridad una base de estimaciones creíbles?
- ¿Todos los actores coinciden en que la visión actual se puede lograr si el plan actual se ejecuta para desarrollar el sistema completo, en el contexto de la arquitectura actual?
- ¿Es el gasto real de recursos frente a los gastos previstos aceptable?"

***Construction Phase* / Fase de construcción**

Los componentes restantes y demás características se encuentran ya desarrolladas e integradas en esta fase, también se realiza un control de los recursos utilizados y la forma para mejorar el rendimiento del equipo. En esta fase en se definen los resultados que se debería obtener:

- "El producto de software integrado en las plataformas adecuadas.
- Los manuales de usuario.
- Una descripción de la versión actual."[8]

Los criterios que se definen, deben responder a las siguientes preguntas:

- "¿Es esta versión del producto estable y maduro como para ser desplegado en la comunidad de usuarios?
- ¿Están todos los interesados listos para la transición a la comunidad de usuarios?

[8] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., *Agile software development methods Review and analysis, VTT Publications*. 2002.

- ¿Son los gastos reales de recursos frente a los gastos previstos todavía aceptable?"

***Transition Phase* / Fase de transición**

El propósito de esta fase es transferir el producto de software de la mano de los desarrolladores a la comunidad de usuarios, en esta fase de transición empiezan a verse los primero problemas relacionados con la funcionalidad de la aplicación. Es el momento en donde se empieza a desarrollar contenido adicional o no contemplado en la fase de elaboración.

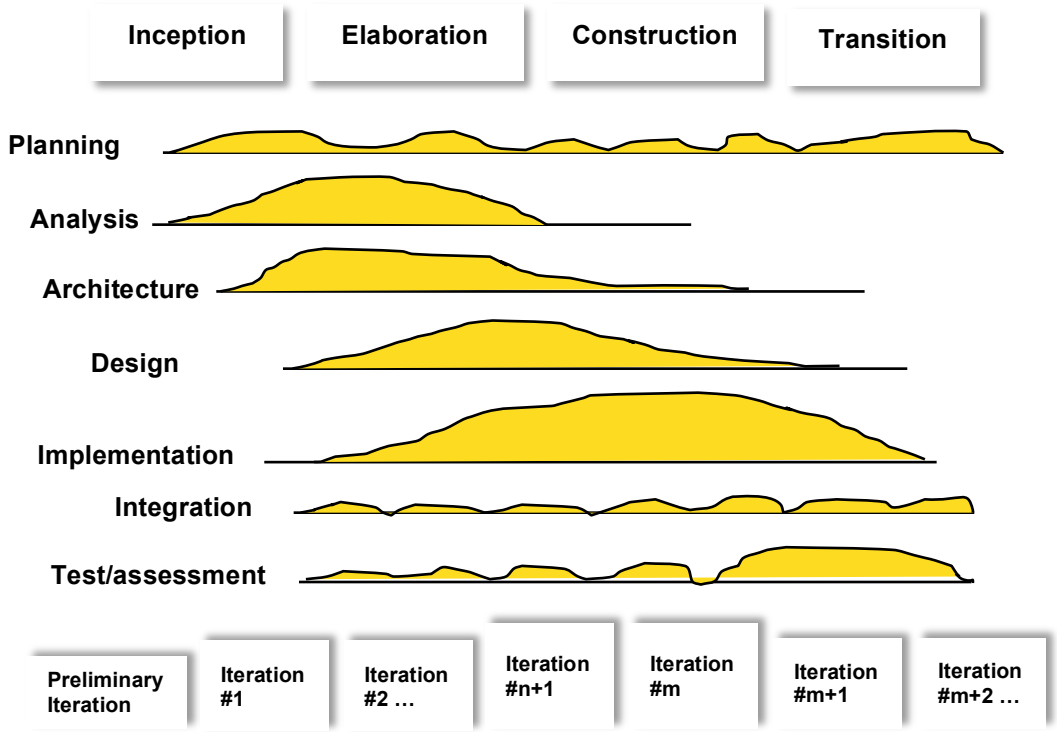


Gráfico Nº 5: Fases de procesos de *RUP*, según Philippe Kruchten.
Fuente: Philippe Kruchten, *A Rational Development Process*

La metodología *RUP* no solo tiene cuatro fases y los documentos que en cada fase se utiliza, si no que además, existen procesos de trabajo básicos que es como esta dividido el equipo de trabajo. Estos equipos de trabajo se dividen de acuerdo a sus

habilidades y se distribuye el trabajo para cada fase.

Mas tarde *IBM Corporation* reestructura los procesos planteados por Philippe Kruchten en su texto, los amplía agregando mas detalle y calidad al producto final; este cambio impacta de manera consistente en el proceso haciendo de él mas largo para el equipo de trabajo, hablando del tiempo que el proyecto tardaría en terminarse.

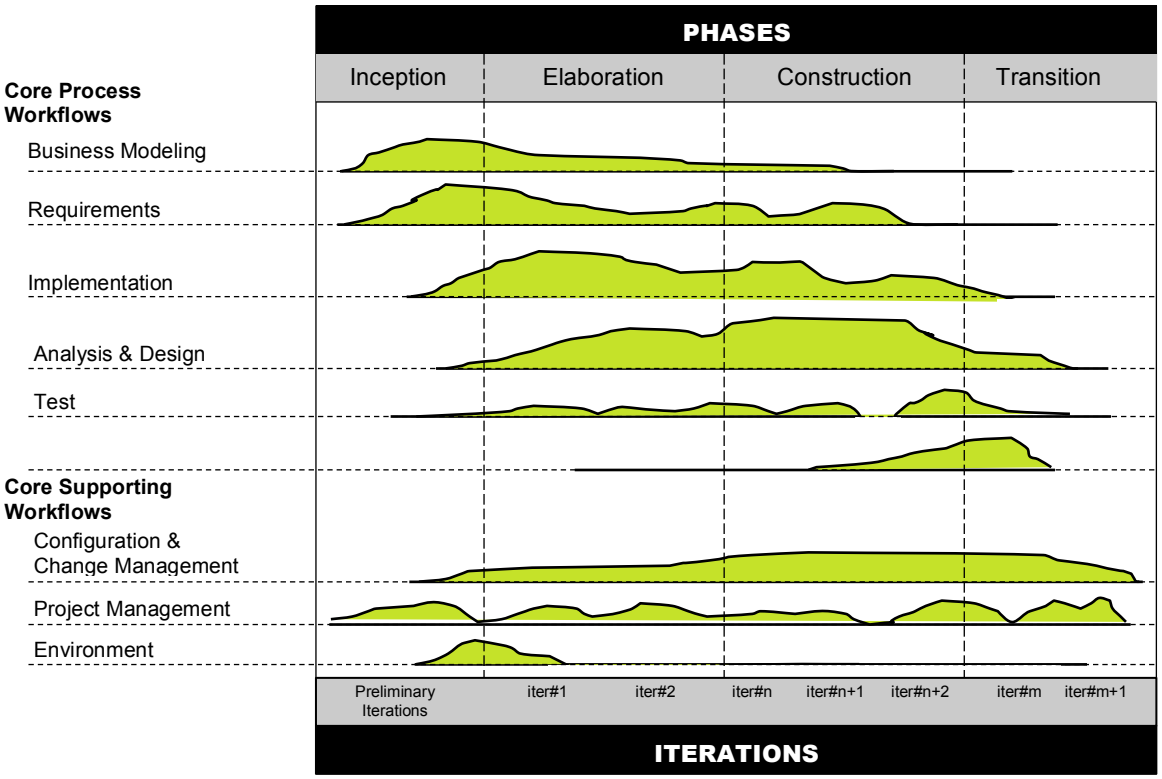


Gráfico Nº 6: Fases de procesos de *RUP*, según *IBM Corporation*.
Fuente: IBM, *Rational Unified Process Best Practices for Software Development Teams*

En esta fase se enfatiza que los resultados de esta fase están relacionados con:

- El logro de usuario auto-compatibilidad.
- Lograr la concurrencia interesada, que las líneas de base de despliegue son completas y coherentes con la evaluación de los criterios de la visión.

- El logro de línea de base como producto final rápido y rentable así como práctico.

Los documentos que usa RUP para dirigir su metodología varían según el criterio de quien lidere el equipo, ya que él puede considerar o no el uso de los que ciertos autores recomiendan. Entre los documentos que se consideran como generales en cada una de las etapas, se encuentran:

Inicio

- Documento Visión
- Diagramas de caso de uso
- Especificación de Requisitos
- Diagrama de Requisitos

Elaboración

- Documento Arquitectura que trabaja con las siguientes vistas:
    - Vista Lógica
        - Diagrama de clases
        - Modelo E-R (Si el sistema así lo requiere)
    - Vista de Implementación
        - Diagrama de Secuencia
        - Diagrama de estados
        - Diagrama de Colaboración
    - Vista Conceptual
        - Modelo de dominio
    - Vista física
        - Mapa de comportamiento a nivel de hardware.
- Diseño y desarrollo de casos de uso, o flujos de casos de uso arquitectónicos
- Pruebas de los casos de uso desarrollados, que demuestran que la arquitectura documentada responde adecuadamente a requerimientos funcionales y no funcionales.

Construcción

- Especificación de requisitos faltantes
- Diseño y desarrollo de casos de uso y/o flujos de acuerdo con la planeación iterativa
- Pruebas de los casos de uso desarrollados, y pruebas de regresión según sea el caso

Transición

- Pruebas finales de aceptación
- Puesta en producción
- Estabilización

Luego del estudio de la metodología *RUP*, basado en Kruchten junto con lo que plantea la metodología de *IBM*, se puede evidenciar que este proceso es perfecto para proyectos grandes de desarrollo para una nueva aplicación o para el reemplazo de una, donde se cuente con un equipo numeroso, se cuente con un presupuesto grande, se dispongan de varias habilidades en el equipo de trabajo y con un espacio para realizar todo el trabajo.

Esta metodología no es la ideal para una adaptación a una sola persona, y peor aún para un proyecto sencillo, ya que como se ha visto cuenta con varias fases y varios procesos a realizar, se requieren varias habilidades como componentes y en el caso de una sola persona no aplicaría.

Pasando a al campo de las metodologías ágiles, también se encuentra Extreme Programming, llamada también por las siglas *XP*, como metodología de primera elección por los estudiantes y jóvenes emprendedores quienes empiezan a desarrollar o implementar aplicaciones de forma comercial.

Extreme Programming / Programación Extrema

Esta metodología, mas que ser una estructura de procedimientos ordenados, es una disciplina que organiza a los integrantes del equipo y a sus habilidades, bajo un mismo sistema de colaboración rápido y sin mucho riesgo.

Este equipo de trabajo puede constituirse desde dos a diez integrantes, siempre y cuando su compromiso de trabajo en equipo sea lo que motive al equipo, este equipo siempre esta dispuesto a asumir retos sobre la marcha y aprender a fondo siempre que sea necesario. Sus ventajas frente a otras metodologías se mencionan en la siguiente lista:

- "Su respuesta temprana, concreta y continua de ciclos cortos.
- Su enfoque de planificación incremental, que rápidamente se origina en un plan general y que se espera que evolucione a través de la vida de la proyecto.
- Su capacidad de programar la aplicación de forma flexible, con la funcionalidad y respondiendo a las necesidades cambiantes del negocio.
- Su dependencia de las pruebas automatizadas escritas por los programadores y clientes para monitorizar el progreso del desarrollo, para permitir que el sistema evolucione, y para corregir defectos tempranamente.
- Su dependencia de la comunicación oral, las pruebas y el código fuente para comunicar la estructura del sistema y la intención.
- Su dependencia de un proceso de diseño evolutivo que dura como el sistema dura.
- Su dependencia de la estrecha colaboración de los programadores y sus habilidades ordinarias.
- Su dependencia de las prácticas que trabajan tanto en el corto plazo instintos de los programadores y los intereses a largo plazo del proyecto."[9]

[9] Kent Beck, *Extreme Programming Explained, Addison-Wesley Professional*, 1999.

Una de las ventajas que tiene *XP* frente a otras metodologías es que por su naturaleza ágil y des complicada, logra una retroalimentación constante y muy rápida dentro del mismo equipo y de forma externa con el cliente.[10]

XP para su desarrollo agrupa cuatro categorías, en las cuales se basa para su éxito:

- Retroalimentación a escala fina
- Proceso continuo en lugar de por lotes
- Entendimiento compartido
- Bienestar del programador

Retroalimentación a escala fina

En esta primera categoría se definen las entradas y los resultados esperados, se crea un documento o documento llamado *User Stories* / Historias de usuarios, donde se especifica lo que cada uno hace y su relación con las entradas y salidas de la aplicación. Esto sería una especia de Caso de Uso en la metodología *RUP*.

Es importante en esta categoría establecer la visita constante de un representante del cliente o el cliente mismo en el lugar donde se encuentran los desarrolladores para responder cualquier duda respecto a requerimientos o funcionalidad.

Finalmente se modulan los requerimientos de forma que el equipo en lo posible pueda trabajar en parejas y así se distribuya el trabajo de forma eficiente, basado en las habilidades de los programadores.

Proceso continuo en lugar de por lotes

Esta categoría contempla un punto muy importante en la metodología *XP*, y es la refactorización[11], esto permite a los programadores evaluar constantemente su

[10] Alistair Cockburn, *Agile Software Development, Addison-Wesley Professional*, 2000.
[11] Kent Beck, *Extreme Programming Explained, Addison-Wesley Professional*, 1999.

lógica y diseño, y al mismo tiempo recodificar lo necesario, esta característica esta sujeta a los siguientes criterios, propiamente recomendado por el autor.

"No es posible re factorizar el diseño del sistema todo el tiempo. Tomaría demasiado tiempo, sería muy difícil de controlar, y lo más probable es romper el sistema. a menos que:

- Usted está acostumbrado a la propiedad colectiva, por lo que no les importa hacer cambios donde sea necesario.
- Ha estándares de codificación, por lo que no tiene que volver a formatear antes de refactorización.
- Se programa en pares, por lo que son más propensos a tener el valor de hacer frente a una refactorización difícil, y es menos probable que se dañe algo.
- Tener un diseño simple, por lo que las refactorizaciones son más fáciles.
- Usted tiene las pruebas, por lo que sería menos probable que algo se dañe, sin conocerlo.
- Usted tiene la integración continua, por lo que si accidentalmente se daña algo en un distancia, o uno de sus conflictos refactorizaciones con el trabajo de otro, es posible saber en cuestión de horas.
- Está descansado, por lo que tiene más valor y son menos propensos a hacer errores.

Entonces, tal vez usted podría re factorizar cada vez que vea la oportunidad de hacer el sistema más sencillo, o reducir la duplicación, o comunicarse con mayor claridad."

La refactorización va de la mano con otro término muy presente en *XP*, el cual es *Pair Programming* / Programación en pares, el consiste en revisar, y evaluar código en parejas, permitiendo un nuevo punto de vista y rápidas mejoras en la funcionalidad.

Finalmente se realizan entregas pequeñas del producto y se lo ubica en un ambiente de prueba en tiempo real para el cliente, de esta forma el mismo tendrá la posibilidad de ir verificando la funcionalidad de sus requerimientos y dando una mejor retroalimentación al equipo de trabajo.

Entendimiento compartido

El código realizado es compartido con todo el equipo, todos son capaces de editarlo, evaluarlo y verificar su funcionalidad, así como cualquiera esta en la capacidad de levantar defectos para que sean corregidos de forma inmediata. La simplicidad de la codificación caracteriza a *XP*, ya se basa en entregar un producto funcional, acaparando todos los requerimientos contemplados por el cliente.

Bienestar del programador

Para *XP*, lo más importante es la integridad de las personas que ejecutan las actividades del proceso, es por eso que la metodología *XP* recomienda es no hacer horas prolongadas de programación, verificación o pruebas, ya que cuando se tiene a los programadores cansado, estos no son eficientes y no logran sus objetivos con claridad. Es importante que el tiempo efectivo de trabajo sea repartido en las parejas que antes se mencionaba.

El ciclo de vida de una aplicación en la metodología *Extreme Programming*, se divide en cinco fases[12]:

- *Exploration phase* / Fase de exploración
- *Planning phase* / Fase de planificación
- *Iterations to Release phase* / Fase de iteraciones
- *Productionizing phase* / Fase de producto

[12] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., *Agile software development methods Review and analysis, VTT Publications*. 2002.

- *Maintenance phase* / Fase de mantenimiento
- *Death phase* / Fase de muerte

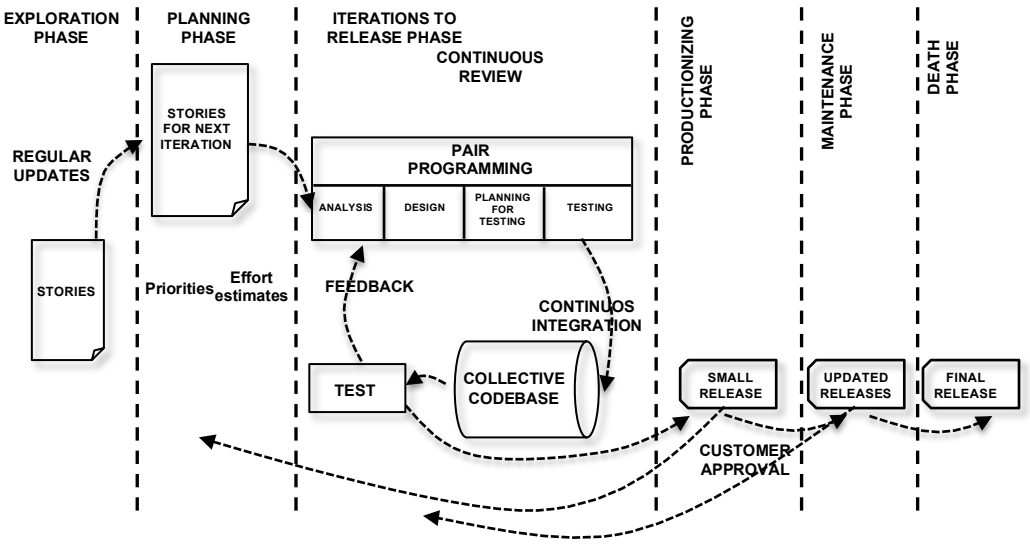


Gráfico Nº 7: Ciclo de vida del proceso *XP*

Fuente: Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., *Agile software development methods Review and analysis*

Exploration phase / Fase de exploración

En la fase de exploración, se definen los *User Stories* o *User Cards*, estas representarán un punto de verificación en la evaluación de funcionalidad de la aplicación, así como también los programadores se familiarizan con las aplicaciones a usar, la arquitectura a definida y demás factores que darán inicio al proyecto.

Planning phase / Fase de planificación

En la fase de planificación se organizan las primeras historias y se hace un consenso en las que las partes se ponen de acuerdo para el contenido de la primera versión del producto de software, esta fase no tarda mas de unos 2 días, ya que se

elaboran los cronogramas con tiempos estimados para la entrega del primer *release*.

***Iterations to Release phase* / Fase de iteraciones**

En esta fase se incluyen varias iteraciones antes de la primera versión de la aplicación, la primera iteración consiste en la elaboración de la arquitectura principal de la aplicación, las siguientes serán ejecutadas en conjunto el equipo de desarrollo con el cliente, basados en historias, para hacer un seguimiento de la funcionalidad. Al final de la última iteración la aplicación esta lista para salir a producción.

***Productionizing phase* / Fase de producto**

En esta fase el cliente revisa el producto antes de salir a la comunidad de usuarios y es en este momento cuando si este lo desea, podría agregar mas funcionalidad o detalles, que pueden ser agregados con un par de iteraciones adicionales. Se termina de elaborar la documentación para el usuario final y se termina de afinar los últimos detalles en cuanto al rendimiento de la arquitectura.

***Maintenance phase* / Fase de mantenimiento**

Una vez el producto se encuentra en producción, el equipo debe seguir elaborando iteraciones con el fin de ir detectando fallas, ya sea en rendimiento, o velocidad con carga y usuarios reales. Esta fase requiere incorporar al equipo nuevos miembros con habilidades mas especializadas.

***Death phase* / Fase de muerte**

En la fase de muerte, la aplicación tiene cubierto todas las historias de usuario, no existen nuevos requerimientos, la aplicación va estable en cuanto rendimiento con la carga real de información. El equipo debe realizar la documentación final,

respaldos necesarios y entrega de manuales de usuario e instalación al cliente y comunidad de usuarios.

Roles y responsabilidades

En esta metodología, definen de manera muy pragmática siete roles, cada uno de ellos con sus diferentes responsabilidades dentro del proyecto. En este caso uno de ellos podría manejar dos roles, sin embargo todo depende de la complejidad de los requerimientos, el tiempo estimado en cada iteración y las habilidades de los integrantes del equipo de desarrollo.

Lo roles definidos en *XP* según su autor son[13]:

***Programmer* / Programador**

Es quien elabora el código, de la forma mas limpia y clara posible para que el equipo en general pueda revisarlo y en algún momento modificarlo.

***Customer* / Cliente**

El cliente es quien elabora los *Test Script*, es quien proporciona los *User Stories* y los requerimientos de la aplicación, es también parte fundamental en la retroalimentación que el equipo debe mantener con el progreso del proyecto.

***Tester* / Ensayador**

Los ensayadores son quienes ayudan al cliente con la elaboración de los *Test Script*, ya que el cliente es quien entiende del negocio, pero no sabe como transmitir el concepto de los requerimientos a una forma en la que la comunidad

[13] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., *Agile software development methods Review and analysis, VTT Publications*. 2002.

de usuarios o las personas que realizarán las pruebas lo entiendan.

Tracker / Rastreador

El rastreador dentro de *XP*, es quien va haciendo un monitoreo de las actividades realizadas comparadas con la planificación del proyecto, esta persona retroalimenta al equipo y alerta sobre el consumo de tiempo y recursos.

Coach / Entrenador

Es la persona que esta a cargo de todo el proyecto, es quien conoce al equipo, sus habilidades, debilidades, la forma en la que trabajan, incluso tiene un amplio conocimiento del negocio para el cual se esta realizando el proyecto. Esta persona es la encargada de hacer reubicaciones de personal o es quien debido a su conocimiento del equipo, establece las parejas de trabajo y modula las actividades.

Consultant / Consultor

El consultor es un miembro externo del equipo, que apoya cuando este lo amerita, esta persona maneja un nivel especializado sobre una rama de estudio del proyecto. Esta persona no siempre se encuentra dentro del mismo ambiente, si no que colabora con visitas esporádicas o bajo demanda.

Manager / Gerente

El gerente o también llamado “Gran Jefe”, es quien toma las decisiones sobre el proyecto de forma administrativa, es quien generalmente identifica desde un punto de vista crítico los méritos y falencias del equipo de trabajo desde un óptica neutral.

***Scrum* en XP**

Una vez definidos las fases de *XP*, los roles y responsabilidades, lo siguiente por conocer es el *Scrum.* Este término que originalmente viene del mundo de los deportes, específicamente del *Rugby*, no es mas que una reunión corta entre los miembros del equipo, en la que se evalúa los avances del proyecto, esta reunión es planificada por el *Coach*, y es el quien delimita su frecuencia. Sin embargo dependiendo del proyecto, las reuniones de *Scrum* se pueden dar hasta de forma diaria.

Esta reunión, esta dividida en tres fases[14]:

- ***Pre-game phase* / Fase de pre juego**

  La fase de pre juego se encuentra sub dividida en dos fases, las cuales son Planeación y Arquitectura, en las que se habla de lo que su nombre dice respectivamente. Aquí se definen lo que a futuro convertirá al proyecto en sus cimientos para la posteridad, tanto en la parte de la construcción de la aplicación, como de la forma en la que los tiempos y recursos estarán divididos.

- ***Development phase* / Fase de desarrollo**

  La fase de desarrollo, es la mas larga de todo el proyecto, ya que es en la que el equipo de desarrollo mas tiempo dedicará, ya que en ella se trabajará exclusivamente en los temas que conciernen al producto para el cual se esta trabajando. En esta fase se realiza el *Sprint,* que consiste en la revisión y evaluación diaria del progreso del código y defectos.

[14] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., *Agile software development methods Review and analysis, VTT Publications*. 2002.

- ***Post-game phase* / Fase de post juego**

Para cuando inicia esta fase, el producto se encuentra ya en su etapa final, ya es una aplicación sólida, ya se han realizado y evaluado todo los *Test Scripts*, la documentación se encuentra realizada y entregada a las personas responsables.

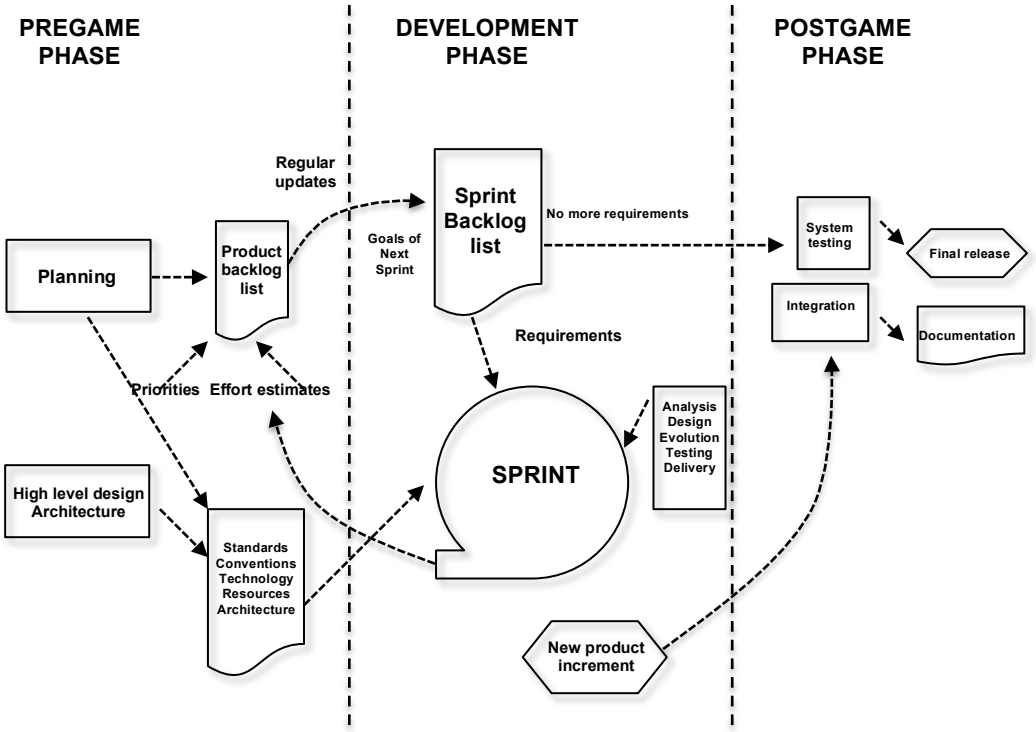


Gráfico Nº 8: Fases de *Scrum*

Fuente: Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., *Agile software development methods Review and analysis*

XP es una metodología muy interesante y dinámica, sin embargo, para in proyecto pequeño o para su adaptación a una sola persona, resulta complicado por la cantidad de roles que existen, la forma en la que se manejan las iteraciones y las fases que la conforman.

Cuadro Nº 1

Características de *Extreme Programming (XP)*

Extreme Programming (XP)	
Considerado uno de los principales factores de influencia de los métodos ágiles hoy en día, las unidades de programación extrema de negocios y de desarrollo de software para determinar objetivos comunes accesibles juntos. La lista de los principios que en la columna de la derecha se describen brinda un ambiente óptimo para lograr estos objetivos.	• Pequeñas iteraciones • Refactorización • Desarrollo basado en pruebas • Programación en parejas • Ritmo sostenible • Cuarenta horas a la semana • Equipo con propiedad de código • Los estándares de codificación • Diseño sencillo • Planificación de juego • La integración continua / construcción • El cliente en el sitio • Tarjetas de historia de usuarios • Prototipo de interfaces de usuario • Soporte de reuniones de seguimiento

Fuente: Kent Beck, *Extreme Programming Explained*

Comparando *RUP* con *XP* en cuanto a tiempo de implementación sobre el coste del cambio de metodología pueden verse mejor en los siguientes gráficos tomados de un *paper* de la USMP Universidad San Martin de Porres de Lima, Perú.[15]

[15] *Rational Unified Process* vs. *Extreme Programming (XP),* Miriam Milagros Diaz Flores, USMP Universidad San Martin de Porres, Lima, Perú.

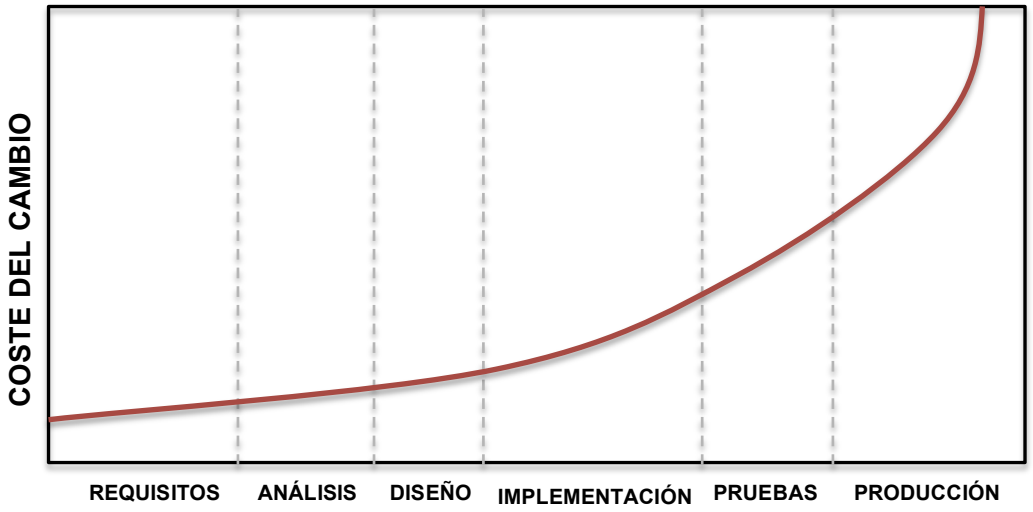


Gráfico Nº 9: Costo de cambio, *RUP*

Fuente: *Rational Unified Process* vs. *Extreme Programming (XP),* Miriam Milagros Diaz Flores, USMP Universidad San Martin de Porres

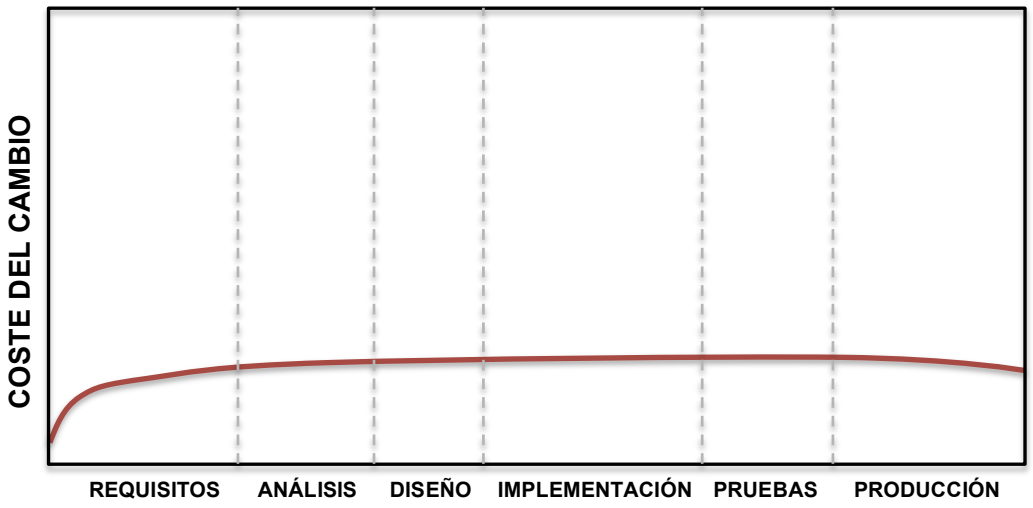


Gráfico Nº 10: Costo de cambio, *Extreme Programming (XP)*

Fuente: *Rational Unified Process* vs. *Extreme Programming (XP),* Miriam Milagros Diaz Flores, USMP Universidad San Martin de Porres

Agile

Agile es otra importante metodología de la cual últimamente se habla, por su versatilidad a la hora de planificar proyectos, ya que cuenta con una serie de elementos muy claros para guiar un proceso de implementación o desarrollo. Estos elementos que colaboran con una implementación de *Agile* pueden variar en función de los requerimientos tanto del programador, como de la persona que implementa, como del cliente.

Sin embargo existen definiciones adicionales que comparten la filosofía que mantiene *Agile*, una de las mas acertadas dice:

- "Individuos e interacciones sobre procesos y herramientas.
- Software funcionando sobre documentación extensiva.
- Colaboración con el cliente sobre negociación contractual.
- Respuesta ante el cambio sobre seguir un plan."[16]

Sus autores tienen unos principios con los cuales esta metodología funciona y que son una especie de lineamiento para quienes empiecen a aprender de ella.

"Principios del Manifiesto Ágil

Seguimos estos principios:

- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar

[16] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas, *Manifesto for Agile Software Development*, http://www.agilemanifesto.org/, 2001.

ventaja competitiva al cliente.

- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El software funcionando es la medida principal de progreso.
- Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia."

Una de las definiciones de *Agile* con la que mas estoy de acuerdo es con la que maneja la corporación *IBM*, en la que dice que utiliza el *feedback* continuo de las personas interesadas para ofrecer alta calidad en el código que se consume a través de las historias del usurario y una serie de iteraciones pequeñas[17]. Esta metodología como *IBM* la define, tiene cuatro componentes principales que la hacen robusta y confiable.

[17] *IBM Corporation, Agile concepts*, 2009.

- “Iteraciones estables.
- Comentarios y retroalimentación de los interesados.
- Equipos de trabajo auto dirigidos.
- Ritmo sostenible.”

Cuadro Nº 2

Características y beneficios de *Agile*

Características de *Agile*	Beneficios
Enfatiza el tamaño del código de trabajo como la principal medida de progreso.	• Se puede presentar al cliente un producto y obtener retroalimentación inmediata. • No gastar meses haciendo diseños de alto nivel en un ambiente cambiante. • Lograr mayor calidad y entregables.
El uso a corto plazo, en caja iteraciones. **Las características claves incluyen:** **• Código estable a finales de cortas iteraciones (2 a 4 semanas)** **• Significativa interacción y participación entre actores e interesados.** **• La confianza y la colaboración entre cliente y los equipos de desarrollo.**	• La rápida toma de decisiones, riesgos del proyecto reducidos (debido a los ciclos pequeños) • Expectativa de alta calidad de ingeniería ¡es decir, no hay tiempo para probar los defectos: un menor número de defectos puede ser inyectado en el código. • Más eficiente para que los desarrolladores centrarse en tareas más pequeñas incluidas en el cuadro de tiempo y establecer un rango para la entrega.
Usa un método en el cual los interesados se ven involucrados para una orientación efectiva.	• Representa necesidades de los interesados a través de casos de uso o medios similares (por ejemplo, las historias, usuarios o escenarios); busca la retroalimentación continua a través de visualizaciones de baja fidelidad, demos de código de iteraciones para dirigir las características del producto. • Reduce el riesgo de construir las cosas mal para el mercado, y aumenta la probabilidad de construir el producto adecuado para el mercado. • Rechaza explícitamente la noción de detallados requisitos del producto, documentos en el inicio del proyecto en favor de las ideas, como los temas de mercado y amplias definiciones de casos de uso. • Mayor consumismo porque, si las partes interesadas, socios no pueden hacer frente a la función en una iteración, la respuesta no será positiva.

Fuente: *Agile@IBM*, *IBM Corporation*

Scrum* en *Agile

En *Agile*, *Scrum* se define como las reuniones diarias que se planifican durante un mes, en el cual siempre se escuchará a las parejas desarrolladoras, si existen, y se irá monitoreando el estado de su trabajo, esto es importante ya que cuando en determinado momento no se avanza, lo que se debe hacer es revisar los estados a nivel general y verificar que grupo esta teniendo inconvenientes, si es posible, el *Scrum Master*[18] separará una pareja, en este caso la mas adelantada en sus labores, para ayudar a la que se encuentra atrás dentro del grupo.

Estos equipos trabajan de forma autónoma, toman decisiones y reaccionan frente a los problemas de manera eficiente, su colaboración con el grupo en general siempre es de forma positiva y manteniendo un clima laboral acogedor.

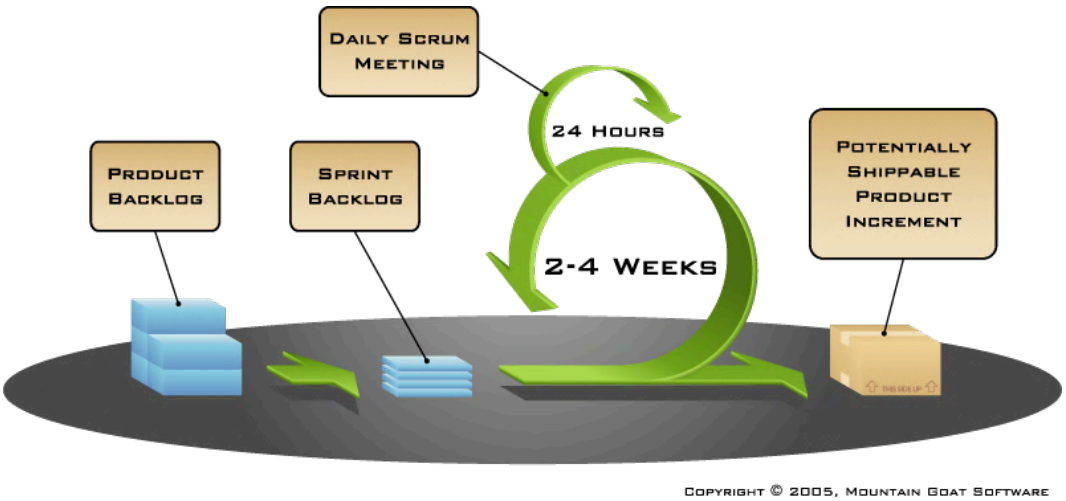


Gráfico Nº 11: Marco de *Scrum* en *Agile*

Fuente: *IBM Corporation*, *Agile* concepts

[18] Es la persona encargada de dirigir las reuniones diarias, sus principales funciones son: manejar el proyecto, resolver impedimentos, facilitar las labores, dirigir reuniones y autorizar los requerimientos.

Las partes de *Scrum* dentro de *Agile* son las siguientes:

Product Backlogs

Es el documento, con el cual se comunica la persona encargada del proyecto, con el cliente; este documento contiene los requerimientos y las especificaciones de la aplicación, los borradores de la funcionalidad e interfaces. Este documento no contiene un lenguaje técnico, si no que por el contrario es una forma en la cual el cliente expresa sus requerimientos.

En este documento, se especificará también la prioridad que tiene cada requerimiento, de esta forma se irá desarrollando los que mayor impacto conllevan para así dejan un tiempo prudente para los que pueden desarrollarse al final.

Sprint Backlogs

Es un documento que contiene en detalle una descripción de cómo el equipo va a realizar lo planificado, los requisitos necesarios, incluso la forma en la que el trabajo será repartido. Esta repartición debe ser estratégica, ya que por ejemplo si una tarea tarda alrededor de 16 horas, esta deberá dividirse en tareas mas pequeñas que se mas eficiente su trabajo. Esta división de actividades, no es repartida entre las personas miembros del equipo, si no que se las toma como una oportunidad en función de las habilidades del equipo.

Daily Scrum Meeting

Son las reuniones diarias, en las que se revisa el constante avance de las tareas, es una reunión en la que se busca conocer las necesidades del equipo y facilitarlas, proveer mecanismo para que las personas trabajen de mejor manera, incluso ayudar a quienes están teniendo problemas, sea por complejidad del requerimiento

o por falta de capacitación, desconocimiento de lo que debe realizar o por simple complejidad.

Potentially Shippable Product Increment

Es el primer producto resultante de la serie de mini iteraciones y cascadas que la metodología *Agile* contempla. Este primer producto debe pasar las pruebas de integración, y de usuario para ser implementado en producción.

Dentro de *Agile*, los roles que se manejan, son cuatros, estos en su momentos están en la capacidad de reemplazarse entre si, sin embargo, se encuentra muy bien definidos sus funciones:

***Product Owner* / Dueño del producto**

- "Responsable por el valor entregado por la versión liberada
- Desarrolla y mantiene la acumulación de productos
- Da prioridad a la pila de producto
- El poder de tomar decisiones para todos los clientes y usuarios
- Presenta y explica el *Product Backlog* al equipo
- Se comunica el progreso del proyecto a las partes interesadas
- Unidades de Experiencia de Usuario y Usuario Diseño
- Acepta y rechaza los resultados del trabajo"

Scrum Master* / Maestro de *Scrum

- "Responsable para maximizar la productividad de los equipos
- Configura y lleva a cabo reuniones
- Proteja el equipo de las influencias externas
- Elimina barreras"

Scrum Team* / Equipo de *Scrum

- "Responsable de la estimación y de comprometerse a trabajar
- Es auto organizado y transversal;
- Consta de siete más o menos dos ejecutantes
- Tiene plena autonomía y autoridad durante un *Sprint*[19]
- Colabora con el dueño del producto
- Demos a los propietario del productos"

***Stakeholders* / Personas interesadas**

- "Observa y asesora
- El material que posee es de suma importancia para el equipo de *Scrum*
- Proporciona información a través del *Product Owner*"

En esta metodología, la comunicación es importante, es por eso que se hace mucho énfasis en las reuniones diarias y/o programadas en el plan.

Uno de los casos de éxito con los cuales se evidencia la efectividad del uso de una metodología como *Agile,* dentro de implementaciones pequeñas es el caso de la aplicación *OATS* de *IBM*, la cual es desarrollada y puesta en producción en 58 países en los que *IBM* tiene presencia, desde Quito, Ecuador. Esta aplicación es de vital uso para los negocios con los cuales *IBM* trabaja y se organiza en equipos de trabajo tales como: desarrollo, despliegue y mantenimiento. Cada uno de estos grupos realiza sus tareas siguiendo la planificación que reza esta metodología. Es por eso que es viable adaptarla al trabajo de una sola persona, teniendo en cuenta todos los documentos que se usan como si fuese un trabajo de mas de una persona.

[19] Es un período de tiempo (1-4 semanas) en el que se produce el desarrollo de un conjunto de elementos de la totalidad de los requerimientos que el equipo se ha comprometido. También se conoce como un periodo de tiempo o de iteración.

La aplicación que se menciona en esta investigación funciona alrededor del mundo, y es usada por aproximadamente trece mil usuarios, el éxito de esta aplicación no se consolida con el hecho de solo haberla puesto en producción, si no por el hecho de atender actualizaciones y configuraciones mensuales, las cuales demuestran en cada ejecución la efectividad de la metodología utilizada.

Hipótesis de trabajo

Las metodologías de desarrollo o implementación de software que en la actualidad comúnmente se practican se pueden adaptar al trabajo de una sola persona.

Señalamiento de variables

- **Variable Independiente**
  Metodología de desarrollo e implementación

- **Variable Dependiente**
  Programador o persona que realice la implementación de una aplicación informática.

Enfoque de la modalidad

En esta investigación de procurará usar un enfoque cualitativo, ya que el formular preguntas dentro del marco de una entrevista, ayudará a resolver las inquietudes vertidas como hipótesis para este trabajo. No se usará un método numérico estadístico para la obtención de datos, si no que se enfocarán las preguntas en profesionales previamente seleccionados quienes emitirán un criterio mas especializado.

La idea es evidencias as cualidades que nos brinda cada metodología y así dirigir mejor una adaptación, teniendo en cuenta el criterio que cada profesional emita al respecto, no se requiere probar teorías ni demostrarlas, si no aplicarlas para el mejoramiento en la ejecución de un proyecto.

La observación ha sido un factor fundamental en el trabajo de investigación ya que se ha trabajado de forma intrínseca en el tema propuesto en esta tesis, así no solo la teoría de la investigación será el pilar mas importante, si no también la experiencia en el área de profundización.

Tipos de trabajo de investigación

Se realizarán entrevistas a profesionales del desarrollo e implementación de software, en las cuales se abordarán, preguntas clave, las cuales ayudarán a orientar y verificar la información estudiada en esta investigación.

Referencia estadística

Serán entrevistados contados seis profesionales los cuales han invertido mucho tiempo de sus actividades diarias en la planificación de proyectos utilizando metodologías de desarrollo e implementación, estos seis profesionales mantienen sus actividades económicas en diferentes ramas comerciales, así no será sesgado el análisis. Ramas como el desarrollo de software privado, proyectos en la industria farmacológica, aplicativos a nivel bancaría, proyectos de implementación a nivel social incluso profesionales *freelancer*.

Plan de recolección de la información

La entrevista a ejecutar será una entrevista de carácter periodístico, este instrumento de investigación es un método usado frecuentemente en un marco no formal, en el cual a manera de conversación se indaga sobre ciertos intereses y así buscar una opinión respecto a un tema en particular.

Planes de procesamiento y análisis de la información

Las respuestas de las preguntas realizadas en las entrevistas se tabularán en una hoja de cálculo, se realizará gráficos en los que se pueda evidenciar de mejor manera las respuestas obtenidas. La justificación de cada uno de los entrevistados a cada pregunta realizada servirá para análisis y juicio en la investigación.

CAPÍTULO III
RESULTADOS

Análisis

Esta investigación tiene un grupo objetivo específico, los jóvenes estudiantes quienes pudiesen desempeñar proyectos pequeños y de bajo presupuesto, y las empresas, quienes están ligadas directamente con clientes y proyectos de gran magnitud. Los estudiantes por la forma en la que podrían trabajar, pueden adoptar la propuesta que vierte esta investigación sin perder la calidad.

Se ha decidido realizar entrevistas a profesionales dedicados al desarrollo e implementación de proyectos, esta decisión se ha tomado teniendo en cuenta que las empresas y profesionales en general, tienen mas criterio para opinar respecto al tema, los estudiantes carecen del conocimiento específico de las metodologías para desarrollo de software. El espectro de clientes que estas empresas pueden alcanzar sería mayor, ya que podrían dedicarse a atender proyectos pequeños, utilizando menos recursos

Dentro del marco de la entrevista, se realizaron cinco preguntas, con las cuales se pretende obtener un criterio diferente y sobre todo especializado. Estos profesionales entrevistados tienen experiencia en diferentes metodologías y sin duda su aporte a esta investigación ha sido invaluable. Estas preguntas responden las inquietudes que se han experimentado a lo largo de la investigación y satisfacen el fin de la misma.

Se entrevistó a específicamente a seis profesionales en el tema del desarrollo d aplicaciones, estas fueron las preguntas:

¿Considera usted importante la utilización de una metodología para el desarrollo o implementación de un software?

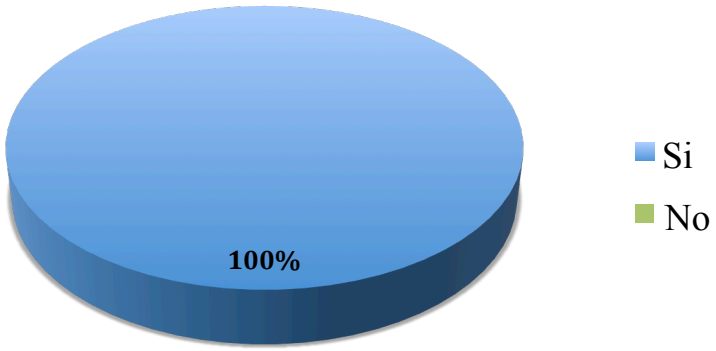


Gráfico Nº 12: Primera pregunta de la encuesta
Fuente: Dennis A. Pazmiño P.

El 100% de los profesionales entrevistados consideran importante la utilización de una metodología para el desarrollo o implementación de un software.

¿Utiliza usted o su equipo de trabajo alguna metodología conocida o propietaria para el desarrollo de sus proyectos?

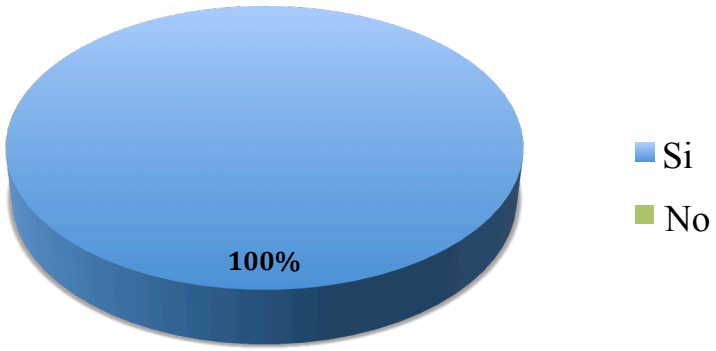


Gráfico Nº 13: Segunda pregunta de la encuesta
Fuente: Dennis A. Pazmiño P.

El 100% de los profesionales entrevistados utilizan con su equipo de trabajo una metodología para sus proyectos.

¿Cree usted viable la adaptación de una metodología de desarrollo para el trabajo de una sola persona?

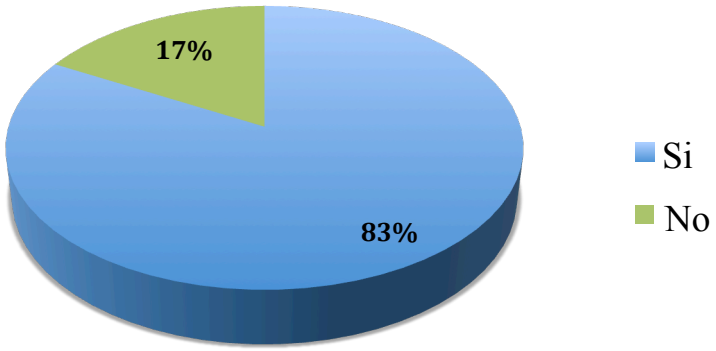


Gráfico Nº 14: Tercera pregunta de la encuesta
Fuente: Dennis A. Pazmiño P.

El 83% de los profesionales entrevistados creen viable la adaptación de una metodología de desarrollo para el trabajo de una sola persona, el 17% restante no lo está.

¿Atendería usted un proyecto pequeño y de bajo presupuesto, delegando a una sola persona la labor, siguiendo una adaptación metodológica que le permita realizar tal actividad sin problemas?

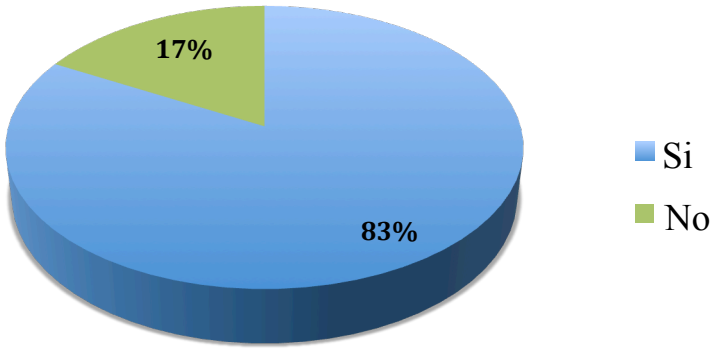


Gráfico Nº 15: Cuarta pregunta de la encuesta
Fuente: Dennis A. Pazmiño P.

El 83% de los profesionales entrevistados atendería un proyecto pequeño y de bajo presupuesto, dejando a una sola persona a cargo, el 17% restante no considera pertinente trabajar de esa forma.

¿Consideraría usted una metodología de desarrollo o implementación que permita el desarrollo de un proyecto con una sola persona, como una alternativa mas para atender nuevos proyectos?

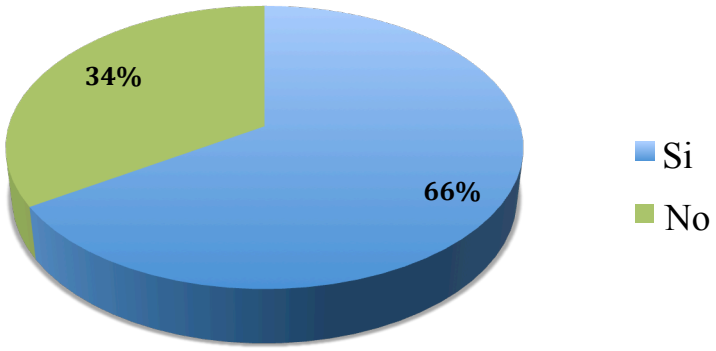


Gráfico Nº 16: Quinta pregunta de la encuesta
Fuente: Dennis A. Pazmiño P.

El 66% de los entrevistados esta de acuerdo en delegar a una sola persona un proyecto utilizando una metodología para trabajo personal, el 34% no lo considera adecuado.

Interpretación de datos

Según la información obtenida, de lo que se puede hablar con seguridad es que para los profesionales entrevistados, el utilizar una metodología en el desarrollo o implementación de un software es algo prioritario, es algo que no puede pasar por alto ya que según la justificación que cada uno dio a la primera pregunta, se puede evidenciar que todos prefieren el mantener el proyecto bajo un esquema organizacional y de control.

Cada uno de ellos ejerce su trabajo utilizando una metodología, lo cual verifica y asevera lo respondido en la primera pregunta. Las metodologías que estos profesionales utilizan para sus proyectos son varias de las que se ha analizado en

esta investigación, lo cual sigue siendo positivo, ya que con esta información se afirma que las metodologías estudiadas son las de primera elección por los programadores y líderes de proyecto.

En la tercera pregunta, se nota que el 17% de los profesionales no cree viable el hecho de que se pueda adaptar una metodología para el trabajo de una sola persona, sin embargo en sus justificaciones, mencionan que para una adecuada planificación es imperioso conformar un grupo de trabajo, lo cual en esta investigación se demuestra que no lo amerita siempre. Es posible trabajar con una sola persona, el punto importante es el ser conscientes que la adaptación propuesta aplica solo a proyectos pequeños.

El 83% de los profesionales entrevistados, confiarían un proyecto a una sola persona, entienden lo importante es saber delegar y mejor aún sabiendo que una metodología respaldará el trabajo de la persona a cargo del proyecto. No es dejar el proyecto a la deriva, ni tampoco delegar en todo el sentido de la palabra, si no empoderar al programador y hacer de el alguien productivo.

Los profesionales quienes administran sus propios negocios, o aquellos que destinan parte de su tiempo libre en proyectos *freelance* considerarían utilizar la metodología propuesta para ampliar el abanico de productos que podrían ofrecer, ya que con este concepto es posible atender de forma eficiente pequeños proyectos

Verificación de datos

Los datos y la información anteriormente indicada, tiene mucha concordancia con los hechos y anécdotas que se vivieron dentro de *IBM* Ecuador, trabajando en el área de *deployment*[20], en donde se utilizó la metodología Agile durante dos años,

[20] Etapa llamada también en español, etapa de despliegue, en la que se realizan configuraciones de la aplicación que no implican cambios drásticos de código.

para entregar software de calidad a clientes de diferentes partes del mundo, adicionalmente, se sabe que los proyectos en los que se aplica alguna metodología se puede evidenciar de mejor manera el trabajo de las personas que se encuentran inmersas, así como también la percepción de la calidad que se tiene al finalizar el plan.

Se ha cotejado la información obtenida, junto con conversaciones informales con colegas de trabajo e incluso de la universidad y se puede verificar que la forma idónea de trabajar en un grupo de desarrolladores es a través de una metodología, ya que no solo se promueve el trabajo en equipo y la eficiencia en la codificación, si no que también se fortifican los lazos entre las personas y se mejora el ambiente de trabajo adecuado. La adaptación de las metodologías para el trabajo de una sola persona, puede ser efectivo en la medida en la que la persona sepa como organizar de forma acertada el tiempo.

CAPÍTULO IV
CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Luego de analizar la metodología *Agile* en esta investigación, se puede determinar que los documentos en los que se basa su ejecución, son favorables para la adaptación al trabajo de un proyecto de forma individual, debido a su fácil interpretación de parte de los interesados. Esto se debe a la manera en la que permite exponer a las partes sus requerimientos sin necesidad de utilizar terminología técnica.

La metodología *RUP* es una excelente opción para el desarrollo e implementación de software, sin embargo para la planeación y ejecución de proyectos pequeños como se lo demuestra en esta investigación no es la adecuada, debido a la complejidad de sus documentos, el tiempo que tardan sus iteraciones y sobre todo la cantidad de roles que existen. Definitivamente no es la mejor elección para el tipo de proyectos que se menciona en esta investigación.

La programación en parejas que plantea la metodología *Extreme Programming* es una forma muy acertada de agilitar la codificación de una aplicación, no obstante, para el trabajo de forma individual no es el indicado. Otro punto negativo que tiene esta metodología para el fin de esta investigación, es la parte comunicacional con el cliente; ya que no se lo incluye durante el proceso.

IBM Ecuador, utiliza con frecuencia y como única alternativa metodológica para sus proyectos de desarrollo e implementación *Agile Plus*, que es una variante de la conocida metodología *Agile* pero con modificaciones propias que potencializan el trabajo en equipo y las reuniones de *Scrum*; los profesionales que trabajan con este método están de acuerdo en el uso de *Agile* en proyectos sin mayor envergadura, así como también en la adaptación al trabajo de forma individual siempre y cuando el uso de los documentos sea el correcto y se planifique bien su ejecución.

CentOS es el sistema operativo que los expertos recomiendan para el uso de servidores de aplicaciones en el mercado, ya que sus prestaciones y facilidad de instalación son un factor importante para decidir entre el resto de competidores. La experiencia de instalación fue muy favorable, no se presentaron problemas mayores y sobre todo, la comunidad detrás de CentOS esta siempre al día con las noticias y recomendaciones para su uso.

Recomendaciones

En la elaboración de un plan de trabajo, lo mas importante es tener claro los requerimientos que el cliente establece, y dentro de este marco, estimar de forma adecuada los tiempos de ejecución tanto para la fase de preparación como para la fase de pruebas. Para esta fase, el cliente debe considerar el personal y su disponibilidad de tiempo para colaborar y llevar a cabo esta fase sin ningún percance.

Es importante también concienciar al cliente que la elaboración del proyecto siguiendo la metodología planteada requiere en la fase de preparación, su colaboración para la construcción de los casos de prueba, estos documentos deberán pulirse a medida que se desarrollen ya que en la etapa final de las pruebas, el personal a probar la funcionalidad de la aplicación, puede o no estar familiarizado con el producto.

Las personas interesadas en la investigación del tema, deberán realizar un análisis de las metodologías que pueden usarse para adaptar a un proceso individual, ya que el material realizado puede estar sujeto a modificaciones y mejoras debido a la diversidad de proyectos que pueden darse.

La utilización de una de las tantas metodologías ágiles son favorables para la ejecución de un proyecto; el profesional interesado en la planificación de su trabajo debería utilizar con toda certeza la investigación propuesta para el mejoramiento del proyecto.

La documentación de todo lo realizado, ayudará al mejoramiento continuo del producto final, este material es fundamental inclusive en el caso de ejecutar una re ingeniería. La documentación que el programador entregará, debe permitir al cliente solucionar de forma sencilla problemas cotidianos.

El profesionalismo con que se trabaje a lo largo del proceso implica no solo la

entrega de un producto de calidad, si no también preocuparse por el desempeño de la aplicación a largo plazo, por tal motivo se recomienda realizar visitas de mantenimiento preventivo, o de recolección de nuevos requerimientos que puedan surgir con el uso del producto entregado, de esta forma no solo se otorgará un producto de calidad si no un servicio que asegure la estabilidad, escalabilidad y evolución del aplicativo.

Título de la propuesta de solución a ser implementada

Metodología para el trabajo individual de proyectos de desarrollo o implementación de software.

Datos informativos del beneficiario de la propuesta

Existen dos grandes grupos quienes son los beneficiarios directos de esta propuesta metodológica, estos grupos responden a las necesidades que el desarrollo e implementación de software implica.

- **Empresas desarrolladoras de software**

  Este grupo se caracteriza por utilizar equipos de trabajo números y bien organizados, este personal se divide en grupos y cada grupo atiende diferente proyectos de forma independiente, muchas veces el *Project Manager* se divide la carga de dos o mas proyectos y gestiona la responsabilidad de los programadores dependiendo la carga y el progreso del proyecto.

  Estos grupos de trabajo dividen sus actividades por módulos, y utilizan documentos muchas veces definidos por la misma empresa. Cada empresa

maneja una metodología y ambiente laborar favorable que a través del tiempo y por su experiencia ha dado resultados.

- **Comunidad desarrolladora**

  Quienes conforman este grupo son estudiantes universitarios dedicados a la programación o desarrolladores *freelance*[21], este grupo suele tener más dinámica que los trabajadores de las empresas privadas, ya que siempre se encuentran alineados con los temas recientes en la escena de la programación. Su experiencia se basa en los trabajos realizados.

  Este grupo también lo conforman aquellas personas que a pesar de contar con un trabajo de horario fijo, en sus tiempos libres atienden proyectos pequeños que no requieren de la totalidad de su tiempo.

Los programadores de las empresas privadas están sujetos a la metodología con que la empresa ha venido trabajando, lo cual es una forma muy organizada de desempeñar los roles asignados. Por otro lado, los programadores *freelance* utilizan sus propios métodos.

Justificación de la propuesta

Desde el punto de vista del cliente, quien en este caso requiere una solución de desarrollo o implementación de software, el hecho que su proyecto sea planificado o guiado por una metodología, sugiere profesionalismo y un enfoque diferente al acostumbrado a ver en proyectos donde la improvisación actúa como parámetro de primera elección.

Los proyectos llevados a cabo con una metodología satisfacen mejor las necesidades impuestas por el cliente, y facilitan la labor del programador o

[21] Trabajador autónomo o independiente

persona encargada de llevar a cabo la implementación en todo el proceso, ya que al involucrar en las pruebas a las personas por parte del cliente, quienes van a utilizar la solución, se obtiene una mejor retroalimentación con respecto a la puesta en marcha del proyecto.

Desde el punto de vista del desarrollador, el mostrar a un cliente un adecuado plan de trabajo que involucre a su personal, es una muestra de compromiso y profesionalismo ante el proyecto, este gesto no solo hace del programador un mejor elemento si no que el cliente tendrá mas confianza en la persona que esta desarrollando un aplicativo para el mejoramiento de algún proceso en su negocio.

Teniendo en cuenta el mismo razonamiento, desde el punto de vista de una compañía dedicada a la producción de software, el trabajar bajo una metodología, sin importar su filosofía es una manera eficiente de no solo controlar el proyecto si no el personal inmerso en el. La estimación de los costos debido a la cuantificación de horas-hombre se encuentra mucho mas clara.

Objetivos de la propuesta

La propuesta tiene definidos sus objetivos, y estos demostrarán su viabilidad en el momento de la planificación de un proyecto.

- Brindar herramientas al desarrollador para que se elabore un plan de trabajo para el desarrollo de su aplicación.
- Concienciar al desarrollador que la improvisación es un factor que no debería tomarse en cuenta para un proyecto sin importar tu tamaño.
- Demostrar que un proyecto pequeño y de bajo presupuesto puede ser entregado cumpliendo con los mandatos y filosofía de metodologías de desarrollo e implementación comprobadas a través del tiempo por grandes desarrolladores y oradores expertos en el tema.
- Involucrar al cliente y a un selecto grupo de su personal operativo para la

fase de pruebas de la aplicación, para así detectar los defectos que la aplicación pueda tener.

Análisis de factibilidad de implementación de la propuesta

Operativamente, esta propuesta es viable debido a que no se necesita una infraestructura para desarrollo, la forma de trabajo es de forma individual o en función del poco recurso humano existente, así como tampoco es necesario tener conocimiento previo de alguna metodología, ya que esta propuesta satisface los requerimientos de un programador con poca experiencia en planificación.

Esta propuesta tampoco requiere un ambiente de trabajo específico, al ser una labor individual, la persona que ejecute esta adaptación metodológica puede mantenerse en el lugar y bajo las condiciones a las que se encuentra acostumbrado. El horario dependerá de la disponibilidad del programador, sin embargo el seguir paso a paso la propuesta planteada requiere de disciplina y organización.

La aplicación de esta propuesta de forma efectiva permitirá al programador disponer de un cronograma de trabajo en el que pueda organizar mejor el tiempo, de esta forma se podrá disponer propiciamente del lapso en el que no se tenga carga con el proyecto y hacer mas eficiente la labor del programador. El cliente por otro lado tendrá el conocimiento de la forma en la que el proceso tendrá curso, proporcionando seguridad y compromiso.

Modelo operativo de ejecución de la propuesta

El siguiente modelo propuesto tiene como objeto detallar de forma práctica la forma de aplicar una metodología o la adaptación de una de ellas para el trabajo individual de un proyecto.

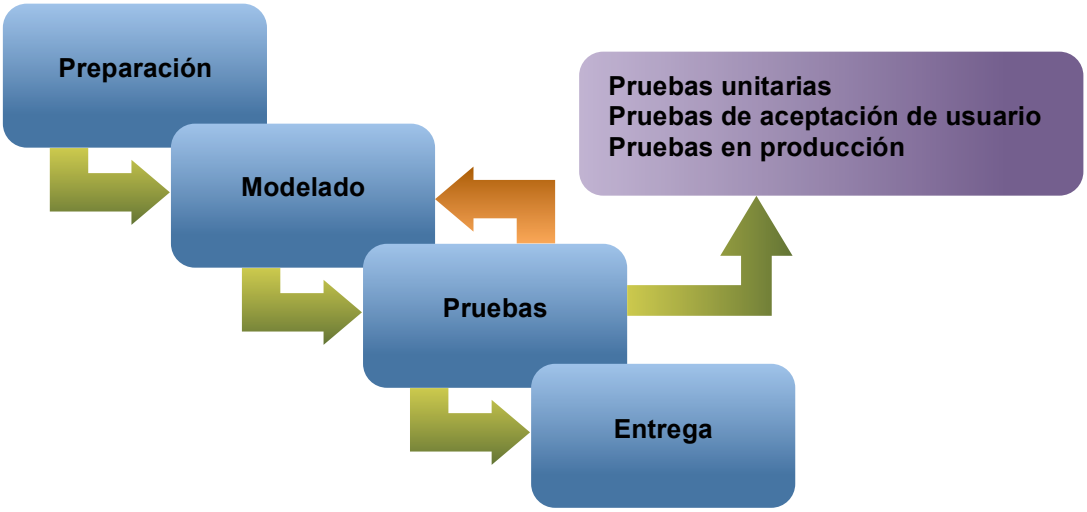


Gráfico Nº 17: Adaptación propuesta de *Agile*
Fuente: *Dennis Andrés Pazmiño Peña*

Esta adaptación de metodología esta compuesta de cuatro fases:

- **Preparación**

  En esta fase se debe realizar la recolección de los requerimientos, utilizando como documento principal, las **historias de usuario** (este documento es opcional y se encontrará a criterio del programador o realizador del proyecto), las cuales ayudarán a describir el proceso requerido, o el proceso manual a ser automatizado. En los anexos de esta investigación se encontrará un modelo a seguir para la recolección de información.

  Luego de tener claro lo que se debe realizar, se deberá establecer un tiempo estimado para su codificación en el caso de existir; este tiempo

deberá ser estimado y organizado en un Diagrama de Gantt[22], de esta forma se podrá visualizar todo el proceso e incluso ajustarlo a medida de su progreso, ya sea para extender el tiempo o para reducirlo.

En esta fase es importante también el comentar al cliente que para el éxito de la implementación del producto final, es clave que un grupo selecto de operativos por parte del cliente participen en la fase de pruebas, este personal debe ser quien en el futuro administre y maneje la aplicación, este grupo aportará con información que el gerente no conoce respecto a la parte operativa de su negocio.

El tiempo que esta fase debería durar no debe ser mayor a una semana, contando ocho horas por día, si el tiempo de las partes no es un inconveniente las horas por día podrían incrementarse.

Paralelamente a este proceso, el cliente deberá realizar los **casos de prueba**, que son documentos en los que se describe por medio de escenarios funcionales, la utilización de la aplicación y el requerimiento a ser probado. Este material deber ser proporcionado al programador al finalizar la fase de modelado, ya que este realizará las pruebas unitarias en su ambiente de trabajo.

- **Modelado**

Esta fase comprende el desarrollo como tal de la aplicación o la personalización de la aplicación en el caso de una implementación. El trabajo ya es netamente producto del programador, los requerimientos ya se encuentran claros y el cliente entiende el procedimiento con el cuál se trabajará. En esta fase el cliente siempre debe estar dispuesto a resolver las dudas que el programador tenga con respecto a la funcionalidad de la

[22] Gráfico en el cual se representa la disposición de las actividades a realizar en determinado proyecto, estimando su tiempo de realización y dependencia con las demás tareas.

aplicación o a la lógica del negocio.

El programador deberá organizar en esta fase los requerimientos de la aplicación, de acuerdo a su prioridad en cuanto a funcionalidad, ordenándolos para su desarrollo desde el de mayor complejidad hasta el mas sencillo de implementar. Esta persona debe establecer un horario en la mañana en la que se revise el estado de los requerimientos en el **documento de requerimientos** (este documento no tiene un formato pre establecido, ya que es el cliente quien proveerá al programador sus requerimientos, bien sea una presentación con diapositivas, un documento de texto, o inclusive una hoja de cálculo detallada), este estado será dado por un valor porcentual asignado por el programador al finalizar su jornada. El objetivo de este ejercicio será encontrar al otro día un documento en el que el programador puedo revisar como va su progreso.

La duración de esta fase depende de la complejidad de los requerimientos, no existe un manual para ello ya que cada proyecto tiene su nivel de complejidad, este tiempo será dispuesto por el desarrollador y aprobado por el cliente. La fecha de entrega del proyecto será definido por las partes teniendo en cuenta el desarrollo de todas las fases.

- **Pruebas**

Esta es una de las fases mas importantes del proceso ya que en esta se revisará la funcionalidad de la aplicación en conjunto con el personal seleccionado por parte del cliente. Esta fase de sub divide en tres etapas:

  - Pruebas unitarias
  - Pruebas de aceptación de usuario
  - Pruebas en producción

El inicio de esta etapa empieza con la ejecución de los **casos de prueba** de

la aplicación, esto lo realizará el programador en conjunto con el cliente o con el personal operativo. Cuando el programador haya finalizado la fase de modelado, empezará la etapa de pruebas unitarias, estas pruebas se realizarán en el ambiente de trabajo del programador y se deberán seguir todos los casos de prueba definidos. En este momento el programador podrá regresar a la fase de modelado para corregir los defectos encontrados y si es posible afinar los casos de prueba.

Una vez terminada la etapa de pruebas unitarias, el programador deberá proveer al grupo de operativos de parte del cliente un ambiente en el que ellos realicen los casos de prueba, este ambiente deberá tener las mismas características técnicas donde el sistema será implantado, de esta forma no solo se comprobará la funcionalidad del producto si no los imprevistos que puedan detectarse en el paso a producción.

El programador en esta etapa deberá proveer al grupo de personas que realiza los casos de prueba un archivo llamado **estado de defectos** en el que se detallen los problemas encontrados. Se deberá realizar una reunión diaria con este grupo de personas en el que se expondrán los inconvenientes encontrados, el programador paralelamente a la fase de pruebas, deberá regresar a la fase de modelado para corregir los problemas encontrados.

Cuando todos los defectos hayan sido corregidos, y los usuarios se encuentren de acuerdo con la funcionalidad de la aplicación, la siguiente etapa a realizar serán las pruebas en el ambiente de producción. El programador deberá realizar el paso del producción del aplicativo teniendo en cuenta las experiencias encontradas en la etapa anterior.

La etapa de pruebas de producción inicia cuando la aplicación se encuentra lista y puesta en marcha en el servidor o en el lugar de su ejecución, los casos de prueba deberán ser de nuevo ejecutados en lo

posible por un grupo diferente de personas, si no se cuenta con este personal, se puede realizar las pruebas con el anterior grupo de personas, incluyendo al programador. Para esta etapa el programador ya no debería estar resolviendo defectos de la aplicación, estos ya debieron ser resueltos en su totalidad en la anterior fase. No se deberá pasar a esta etapa si aún existen defectos por corregir.

En esta etapa de la fase de pruebas, la carga de trabajo del programador se reduce, y es momento para realizar la documentación de la aplicación, esta información será necesaria para finalizar la entrega del producto.

- **Entrega**

  Luego de finalizar satisfactoriamente la fase de pruebas, y ya el producto se encuentra implementado en los servidores de la empresa, se procede a entregar los manuales de usuario, instalación y demás documentación utilizada a lo largo del proceso. También se podrá realizar coordinar capacitación al personal adecuado siempre y cuando esto se haya acordado con el cliente.

Es parte fundamental del trabajo profesional del programador, coordinar con el cliente en un futuro una visita de mantenimiento o nuevos requerimientos que puedan emerger luego de la implementación, esto con el fin de dar a conocer al cliente el interés y profesionalismo con el cuál se realizó el acuerdo.

Documentos de la propuesta

- **Documento de requerimientos**

  Este documento proveerá al desarrollador el detalle de los requerimientos

que debe solventar con la aplicación. En la parte final de esta investigación, a manera de anexo se mostrará un formato en el cual el desarrollador o interesado en la propuesta podrá basarse.

- **Estado de defectos**

Este archivo será entregado a las personas encargadas de realizar las pruebas, cada persona tendrá que tener una copia para documentar los defectos encontrados, es importante que estos archivos sean consolidados por una persona designada y esta a su vez envíe al programador un solo archivo en el que contengan los defectos encontrados por el grupo.

En la columna Abierto / Cerrado del archivo propuesto, la persona encargada de las pruebas ubicará un nuevo defecto como **abierto**, de esta forma el programador entenderá que es un nuevo a resolver, en la columna Estado el programador pondrá a su criterio el estado de resolución de ese defecto, ya sea **bajo resolución**, cuando este requiere atención del programador y este ha empezado su resolución, **arreglado** cuando el defecto esta listo para se re probado por la persona que encontró el defecto y **desecho** cuando no es un error de la aplicación, si no un error en el caso de prueba o mala ejecución por parte de la persona que probó. En la columna Abierto / Cerrado la persona que esta comprobando la resolución del defecto deberá especificar como **cerrado**, así el grupo entenderá que ya no es problema a resolver y el desarrollador pasará al siguiente defecto.

En este archivo, también se encontrará una pestaña adicional llamada **evidencias,** en la cual la persona a realizar la comprobación podrá proporcionar mediante una captura de pantalla al desarrollador un mejor material donde pueda observar el error descrito. La existencia de esta evidencia podrá especificarse en la columna Evidencia del formato propuesto

- **Casos de prueba**

  Este documento contendrá las especificaciones de los escenarios que las personas encargadas de las pruebas deberán realizar y conforme a esa funcionalidad, evidenciar los defectos a revisar por parte del programador.

  En los anexos de esta investigación se encontrará un ejemplo de cómo elaborar este archivo conforme a los requerimientos a ser evaluados.

Perspectiva del impacto de la propuesta

Uno de los impactos que esta propuesta metodológica puede tener, es operativamente a nivel empresarial; las empresas dedicadas al desarrollo e implementación de solución de software, estarían en la capacidad de generar especial atención en proyectos pequeños y de bajo presupuesto, sin poner a disposición mas de una persona dedicar en su totalidad. Las compañías dedicadas a este tipo de emprendimientos ampliaría el catálogo de sus productos o incluso podría tener a nuevo personal trabajando de forma individual en los proyectos que esta proposición reza.

El impacto social que se evidenciaría, es la aceptación de los clientes a los desarrolladores que individualmente proponen una negociación ante un proyecto de bajo presupuesto, las expectativas de éxito por parte de los negociantes se vería respaldada y de esta forma aseguraría la calidad del producto final; este hecho a la postre traería nuevos clientes y podría así un desarrollador considerar aún mas formal su oficio.

REFERENCIAS

- Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., *Agile software development methods Review and analysis, VTT Publications*. 2002.

- *Agile@IBM, IBM Corporation, Learning the basics*, 2011

- Alistair Cockburn, *Agile Software Development, Addison-Wesley Professional*, 2000.

- *IBM Corporation, Agile concepts*, 2009

- *IBM Corporation, Rational Unified Process Best Practices for Software Development Teams, Rational Software White Paper TP026B, Rev 11/01*, 1998.

- *IBM Corporation, Rational Unified Process, RAD10971-USEN-00*, 2003.

- Jose Fernando Diaz Luizaga, Desarrollo De La Metodología Ágil Uni-Scrumx Para El Proceso De Desarrollo De Software De Proyectos Desarrollados Por Una Sola Persona, Universdad Mayor de San Simón, http://www.cs.umss.edu.bo/rep_tesis.jsp?codigo=1855&tipo_tes=1, 2010

- José H. Canós, Patricio Letelier y Ma. Carmen Penadés, Metodologías Ágiles en el Desarrollo de Software, DSIC -Universidad Politécnica de Valencia, 2003.

- Kent Beck, *Extreme Programming Explained, Addison-Wesley Professional*, 1999.

- Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew

Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas, *Manifesto for Agile Software Development*, http://www.agilemanifesto.org/, 2001

- MSc(c). Javier Mogollón Afanador, MSc. Luis Alberto Esteban Villamizar, El Desarrollo Individual De Proyectos De Software: Una Realidad Sin Método, ISSN: 1692-7257 - Volumen 1,
- http://www.unipamplona.edu.co/unipamplona/portalIG/home_40/recursos/03_v13_18/revista_17/03122011/07.pdf, 2011

- Philippe Kruchten, *A Rational Development Process, CrossTalk, 9 (7), STSC, Hill AFB, UT, pp.11-16*, Agosto 1995.

- *Rational Unified Process* vs. *Extreme Programming (XP),* Miriam Milagros Diaz Flores, USMP Universidad San Martin de Porres, Lima, Perú

ANEXOS

PREGUNTAS REALIZADAS EN LA ENTREVISTA

(Formato en blanco)

¿Considera usted importante la utilización de una metodología para el desarrollo o implementación de un software?

☐ Si Explique su respuesta anterior_____

☐ No _____

¿Utiliza usted o su equipo de trabajo alguna metodología conocida o propietaria para el desarrollo de sus proyectos?

☐ Si Explique su respuesta anterior_____

☐ No _____

¿Cree usted viable la adaptación de una metodología de desarrollo para el trabajo de una sola persona?

☐ Si Explique su respuesta anterior_____

☐ No _____

¿Atendería usted un proyecto pequeño y de bajo presupuesto, delegando a una sola persona la labor, siguiendo una adaptación metodológica que le permita realizar tal actividad sin problemas?

☐ Si Explique su respuesta anterior_____

☐ No _____

¿Consideraría usted una metodología de desarrollo o implementación que permita el desarrollo de un proyecto con una sola persona, como una alternativa mas para atender nuevos proyectos?

☐ Si Explique su respuesta anterior_____

☐ No _____

GLOSARIO

Término	Definición
Agile	Es una metodología que involucre a los interesados en el producto de forma continua para proporcionar un código de alta calidad, validado por los casos de prueba definidos por el cliente. Esta metodología es una sería de cascadas pequeñas en las que los participantes son activos en todo el proceso.
Artefacto	Son los documentos con los cuales la metodología se ayuda para mejorar la comunicación entre el cliente y el programador.
Test Cases / Casos de prueba	Son documentos que detallan a la persona encargada de realizar una prueba, el escenario correcto para seguir y evaluar.
Deployment / Despliegue	Es la etapa en la que se realizan configuraciones posteriores a la entrega del código núcleo de un aplicativo.
Documento de requerimientos	Es el formato propuesto para la recopilación de los requerimientos de parte del cliente, este archivo comprende las necesidades del cliente expuestas al programador.
Estado de Defectos	Es un documento en el cual el equipo encargado de realizar las pruebas documentará cada defecto encontrado para así enviar al programador a resolver.
Evidencia	Es el material con el cual el programador visualiza un defecto encontrado, esta evidencia puede ser tomada con una captura de pantalla, o un mensaje de error indicando el problema.
Herramienta	Para realizar una técnica, podemos apoyarnos en las herramientas software que automatizan su aplicación.
Iteration / Iteración	Es un periodo en el que el programador realizar una revisión sea de código o configuración, para satisfacer una de las etapas de una metodología. El tiempo de duración de este periodo

	esta dado en función a la metodología que se esta aplicando.
Metodología	Conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software.
Pair Programming / Programación en parejas	Es una característica de la metodología de desarrollo ágil *XP* en la que se utiliza una pareja de programadores para hacer mas eficiente la labor de codificación.
Procedimiento	Definición de la forma de ejecutar la tarea.
Producto	Resultado de cada etapa.
Re-factoring / Re-factorización	Es el proceso mediante el cual un proceso, ya revisado, pasa de nuevo a la etapa en la que originalmente terminó su etapa y regresa para realizarlo de nuevo con las mejoras necesarias.
Scrum	Es el proceso de la metodología *Agile* en el que se reúnen los desarrolladores y personal involucrado para compartir los avances, ayudar a los que tiene problemas e ir solventando dificultades en el desarrollo.
Self Directed teams / Equipos auto dirigidos	Es la facultad que tienen los equipos para organizar su trabajo de forma individual, sin la necesidad de un coordinador de actividades. Los grupos empoderados tienen la capacidad de resolver problemas por si solos.
Sprint	Dentro de la metodología *Agile*, es el nombre de las iteraciones, *Agile*, es una composición de varias *Sprint*.
Stakeholders / Interesados	Son las personas que encargan al grupo programador una tarea, en este caso podría considerarse a un interesado como un cliente.

Tarea	Actividades elementales en que se dividen los procesos.
Técnica	Herramienta utilizada para aplicar un procedimiento. Se pueden utilizar una o varias.
User Stories / Historias de usuario	Es un documento en el cual el cliente expresa lo que realiza o esta sujeto a una automatización, este detalle el cliente lo realiza con sus palabras, evitando así un lenguaje especializado como el *UML*.

APLICACIÓN

La aplicación utilizada para la demostración de la aplicación de la metodología propuesta se llama *OrangeHRM*.

OrangeHRM es una aplicación web especializada en gestionar los recursos humanos de una empresa y el ciclo de vida de los empleados. Permite acceso con rol de empleado o de administrador, con distintos módulos que permiten identificar a los empleados, escribir informes, analizar el rendimiento y horas de trabajo, administrar vacaciones y bajas, inclusive su perfil profesional.

Datos técnicos

Nombre: *OrangeHRM*
Versión: 2.6.0.2
Licencia: GPL v2
Plataforma: Servidor para Windows y Linux
Idioma: Multilenguaje
Web oficial: http://www.orangehrm.com/

Pre requisitos de instalación

Los pre requisitos mostrados a continuación son los recomendados por el fabricante de la aplicación, esta información se detalla en su sitio web.

http://www.orangehrm.com/installation-guide.shtml

- Apache HTTP Server 1.3 o posterior
- MySQL 5.0 o posterior
- PHP 5.2.4 o posterior

Análisis de la aplicación

Forma de distribución

La aplicación está disponible para descargar de forma gratuita, bajo la licencia antes comentada, y con el paquete en código fuente desde la sección de descargas.
Además la empresa desarrolladora ofrece un completo programa de soporte y servicios de pago que incluye formación, soporte técnico, *plugins* y personalización, entre otros. También existe una red de *partners* en la que las empresas proveedoras pueden formar parte [6] o bien las empresas que busquen servicios de *OrangeHRM* pueden buscar empresas especialistas.
También existe una versión de OrangeHRM en la nube accesible siempre desde cualquier sitio vía web, sin preocuparse de la instalación o seguridad de los datos. Este servicio lo ofrece la empresa desarrolladora, es de pago, pero cuenta con una demo de prueba.

Licencia de módulos/*extensions*

La aplicación dispone de *plugins*, pero al ser de pago, no se ha podido comprobar su licencia exacta.

Licencia

La licencia de OrangeHRM es GPL v2 (*GNU General Public License, version 2*) cuyos términos se pueden consultar.
Resumidamente define a la aplicación como software libre, con libertad de uso, modificación y distribución.

Funcionalidad

Completo ciclo de vida de los empleados

OrangeHRM incorpora las herramientas necesarias para gestionar la estructura de una pequeña o mediana empresa, teniendo registrados e identificados a todos los empleados, sus departamentos o áreas, horarios de trabajo, horas dedicadas, salarios, días de permiso o vacaciones. Además también permite hacer una buena gestión a la hora de hacer procesos de selección de trabajadores. El acceso a la aplicación se permite con rol de administrador de recursos humanos, o de empleado, pudiendo éste último pedir directamente permisos o escribir informes, entre otras pequeñas tareas a las que el administrador puede darle acceso.

Usabilidad

Diseño de la interfaz

La interfaz de *OrangeHRM* está dividida en módulos, agrupados bajo el menú de la aplicación en una barra superior. Estos módulos son desplegables que permiten con el ratón elegir la funcionalidad deseada.
La mayoría de páginas o vistas de esta aplicación web son formularios de datos o campos de búsqueda.

Módulos

- **Módulo de Administrador.** Este módulo solo es accesible con un usuario administrador de la aplicación. Permite definir la información general y estructura de la empresa, definir los puestos de trabajo de la compañía, las categorías salariales y demás datos de interés. Este módulo también es el

responsable de añadir todas las opciones necesarias en cada una de las categorías de los empleados, como lo son nacionalidad, idiomas, formación, aptitudes y habilidades, así como administrar los usuarios del sistema.

- **Módulo del Empleado.** Este módulo accesible por el trabajador le permite gestionar y editar su propia información, como datos de contacto, direcciones, personas de contacto alternativas, su propia foto, y otra información relativa a la formación académica o experiencia laboral. Esta información puede ser llenada por el mismo trabajador o por el administrador, teniendo en cuenta las hojas de vida del personal.

- **Módulo de Personal.** Este módulo permite al administrador gestionar a los empleados y consultar su información. Se pueden borrar o añadir nuevos registros, y así llenar todos sus datos relativos, entre una gran cantidad de opciones de carácter personal.

- **Módulo de Reportes.** Este módulo genera reportes totalmente configurables según las necesidades, para uno o un determinado número de empleados. Pueden definirse multitud de criterios a la hora de generar un informe, pudiendo guardar estos como plantillas para no repetir las mismas tareas. Estos reportes pueden ayudar al personal administrativo a tomar decisiones.

- **Módulo de Tiempo y Asistencia.** Este módulo para contabilizar horas o rentabilizar el tiempo, también llamado “time tracking”, permite registrar la cantidad de tiempo de un trabajador en una determinada tarea, administrando las propias y las de los empleados. También permite definir los días no laborables. El objetivo es optimizar los recursos reduciendo costes y errores.

- **Módulo de Beneficios.** Este módulo es una plataforma integrada para administrar tareas relacionadas con los beneficios. Permite al administrador definir coberturas médicas y definir el esquema de la nómina de la empresa. También permite asignar distintos tipos de beneficios ya sea de forma grupal o individual.

- **Módulo de Permisos.** Este módulo permite una completa gestión de los días de permiso de los trabajadores. Se pueden apreciar en una sola pantalla los días y horas de permiso de todos los trabajadores en un determinado año, definir los posibles tipos de permiso, establecer los días feriados, de vacaciones y los tipos de jornada laborables, dar permiso a un empleado y controlar así de forma correcta al personal.

- **Módulo de Reclutamiento.** Este módulo es una solución completa para el proceso de contratación, incluyendo las peticiones del personal, la aprobación de vacantes y la captura de la información de los candidatos. El módulo también permite a los profesionales de recursos humanos generar plantillas y documentación para mejorar el proceso de contratación.

Facilidad de uso

La aplicación es de fácil uso, debido en parte al diseño y la estructura de la aplicación. La organización en módulos independientes hace que para el usuario sea muy fácil el manejo de la misma y la identificación de las distintas opciones y funciones; todo ello sin que parezca que salimos de la aplicación y sin cambiar el aspecto radicalmente, siendo éste uniforme.
El uso de la herramienta es muy sencillo también debido a que las tareas se basan principalmente en editar datos o hacer clic con el ratón.

Accesibilidad

OrangeHRM no está dotado especialmente con funciones de fácil acceso para personas con problemas de accesibilidad de cualquier tipo. De todas formas la aplicación puede integrarse perfectamente con cualquier tecnología de asistencia del sistema operativo, y con cualquier opción relacionada con el navegador de internet.

Portabilidad y Adaptabilidad

OrangeHRM es una aplicación servidor disponible para *Windows* y *Linux*. Necesita para su instalación un servidor con los servicios *Apache*, *PHP* y *MySQL*, independientemente del sistema operativo en el que se decida instalar. Los requisitos e instrucciones de instalación pueden consultarse en la página del proyecto.

A nivel de cliente, *OrangeHRM* puede ser accedido desde cualquier plataforma o sistema operativo, tan solo hace falta conexión a la red apropiada (o en modo local) y un navegador de internet.

Instalación de XAMPP 1.8.1 en CentOS 6.4

XAMMP es una herramienta que incluye todos los pre requisitos necesarios para la instalación de OrangeHRM, su instalación e inicialización son simples comandos ejecutados en la terminal del equipo.

XAMPP es un servidor que trabaja de forma independiente y se encentra en el mercado de forma libre para su descarga, y utilización, su nombre se lo obtiene por las iniciales X (para cualquier sistema operativo) Apache, MySQL, PHP y Perl.

Para instalar e inicializar este servidor se deben ejecutar los siguientes comandos:

```
yum update

wget
http://www.apachefriends.org/download.php?xampp-
linux-1.x.x.tar.gz

tar xvfz xampp-linux-1.8.1.tar.gz -C /opt

yum -y install glibc* libstdc* gcc glibc.i686

/opt/lampp/lampp start
```

Instalación de OrangeHRM en CentOS 6.4

Descargar la versión 2.6 de *OrangeHRM* en la carpeta de descargas regulares, luego se deben seguir los siguientes comandos para su extracción en la carpeta pública del servidor XAMPP.

```
su -

cd /opt

tar -zxvf orangehrm-2.6.0.2.tar.gz

mv  orangehrm /var/www/html

cd /var/www/html

chown -R apache:apache orangehrm

chmod -R 755 /var/www/html/orangehrm/lib/confs/
```

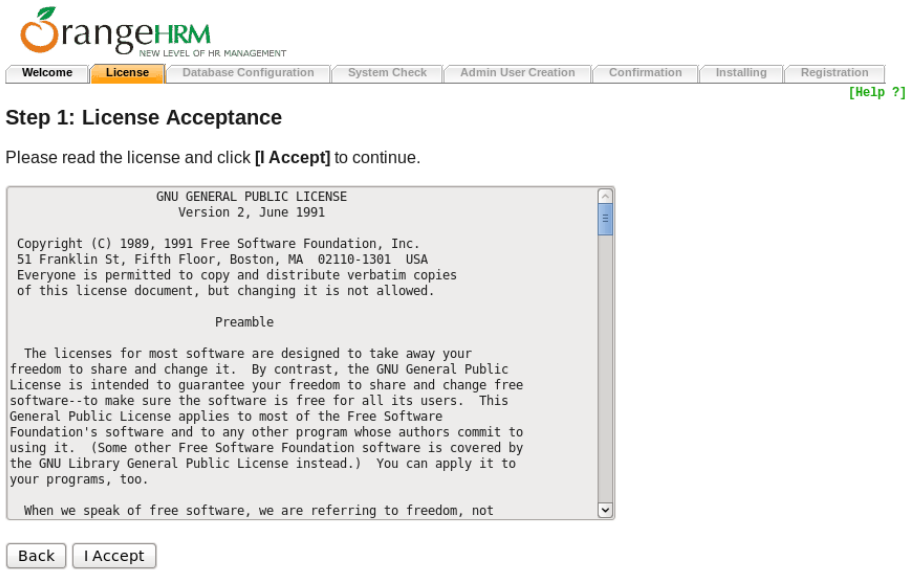
Instalación de OrangeHRM

La instalación de *OrangeHRM* se la realiza utilizando un navegador web. Una vez se ubique la carpeta descomprimida en la carpeta pública de XAMPP. Se debe escribir en la barra de direcciones del navegador:

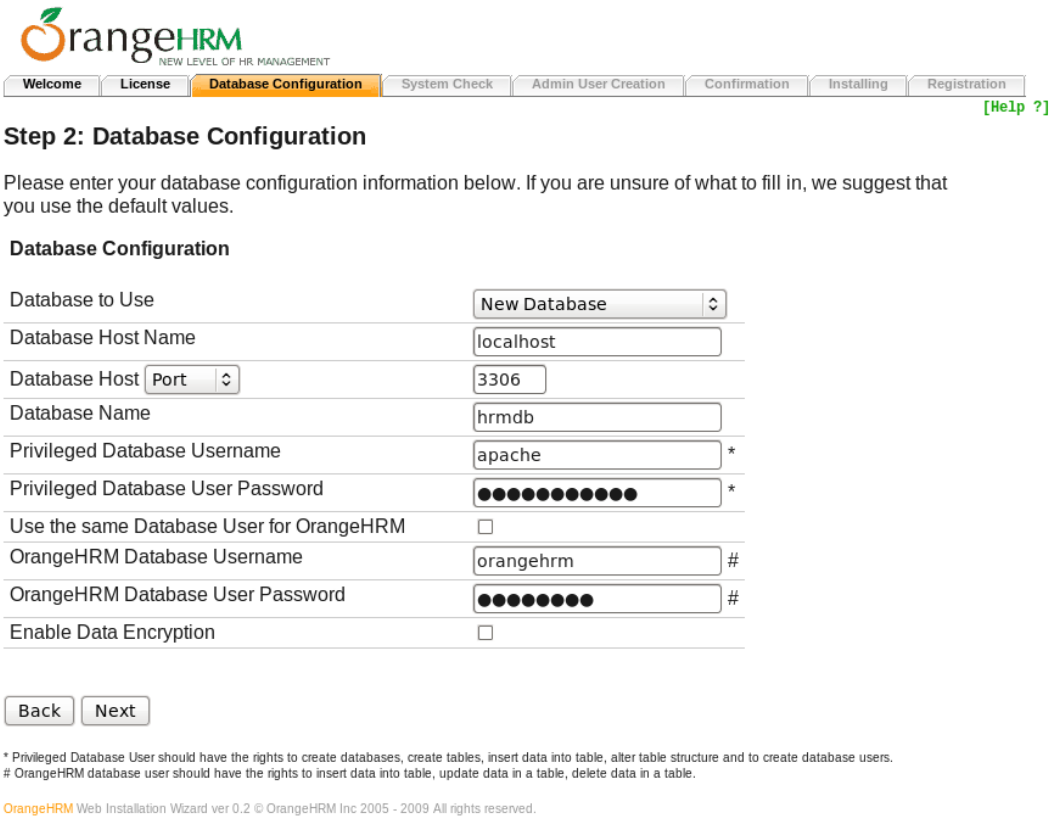
http://localhost/orangehrm

La primera pantalla dará la bienvenida a la instalación.

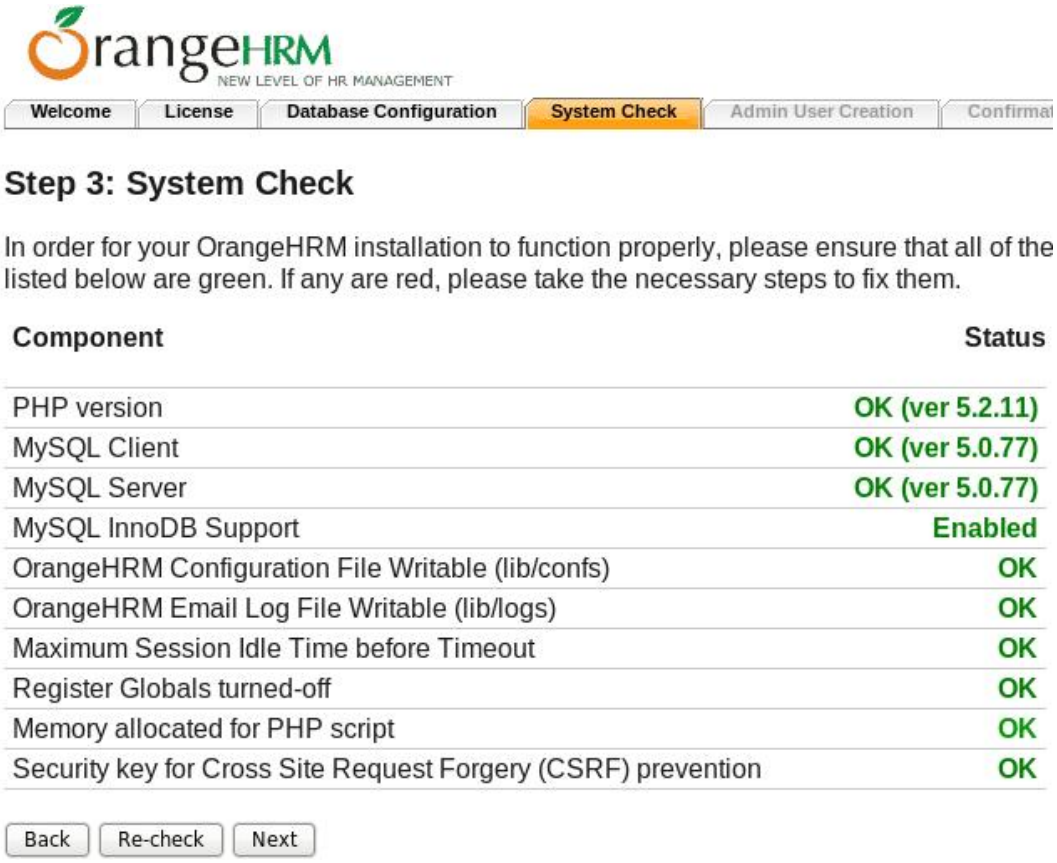
El paso 1 mostrará la licencia de la aplicación y se pasará al siguiente paso luego de presionar el botón **I Accept.**



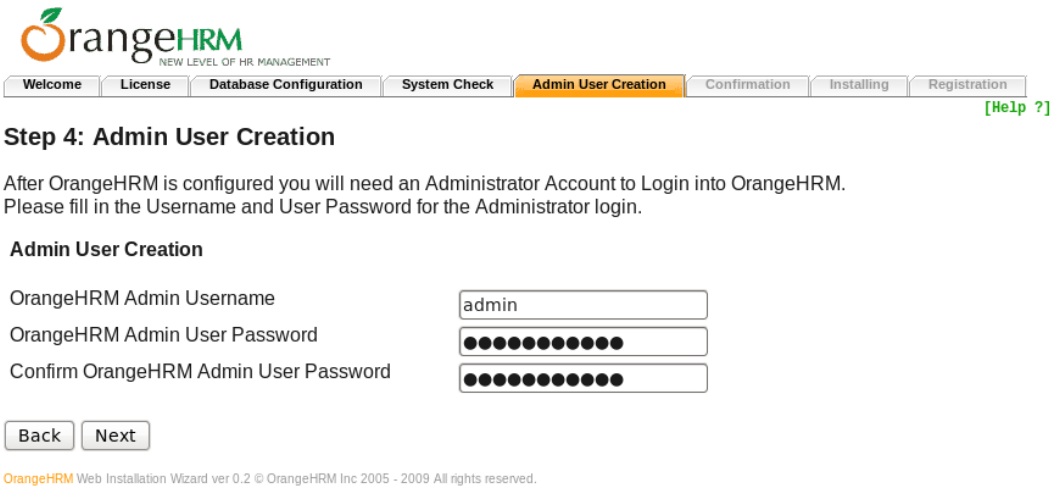
En el paso 2 de la instalación se deberá especificar si ya existe una base de datos o si se desea crear una nueva, se deberá especificar el nombre de ella, el puerto y el nombre de los usuarios con privilegios para la creación.



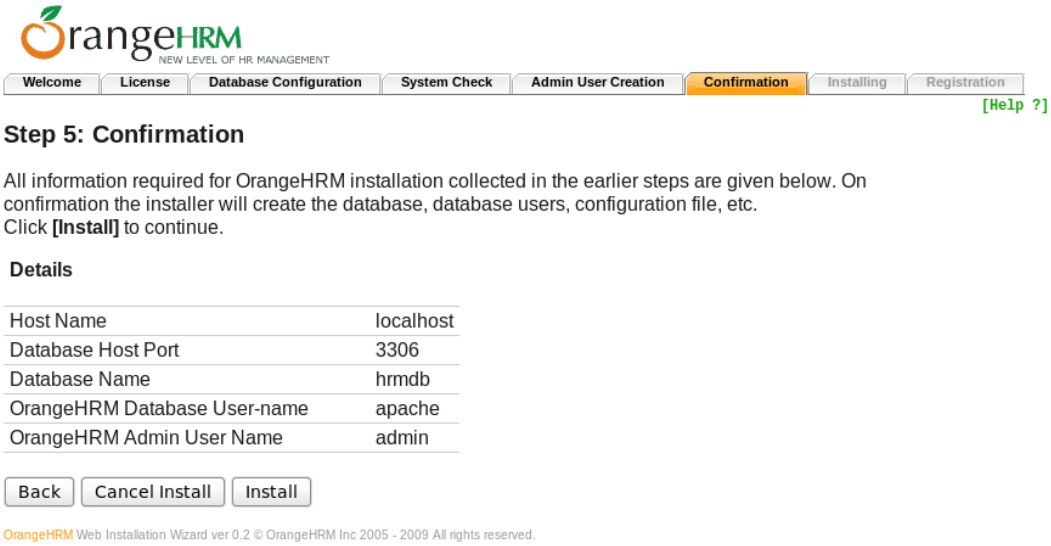
El paso 3 es una verificación que realiza el asistente de la instalación y puede ser guía para la verificación de los componentes requeridos para la instalación.



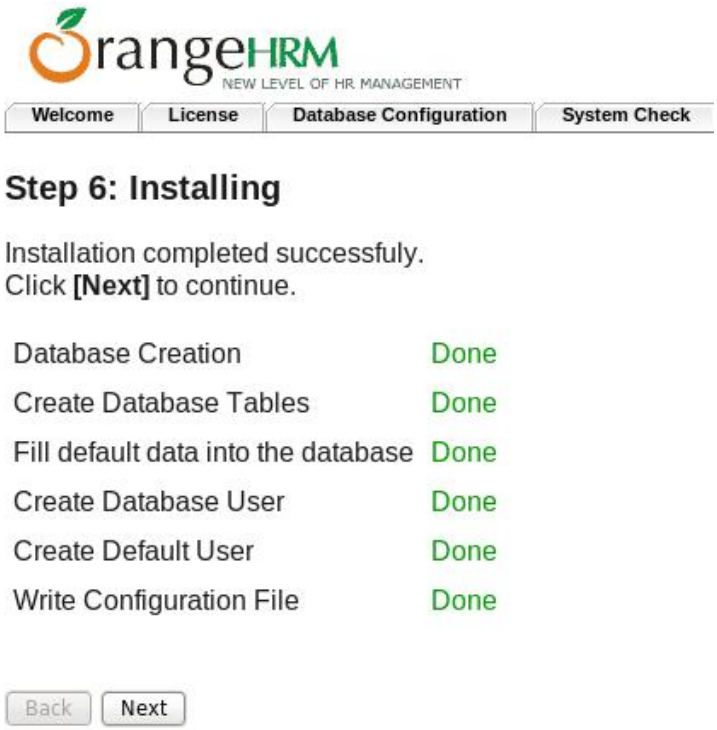
En el paso 4 se procederá a la creación del usuario de administración de *OrangeHRM*, para continuar con la instalación se debe presionar el botón **Next**.



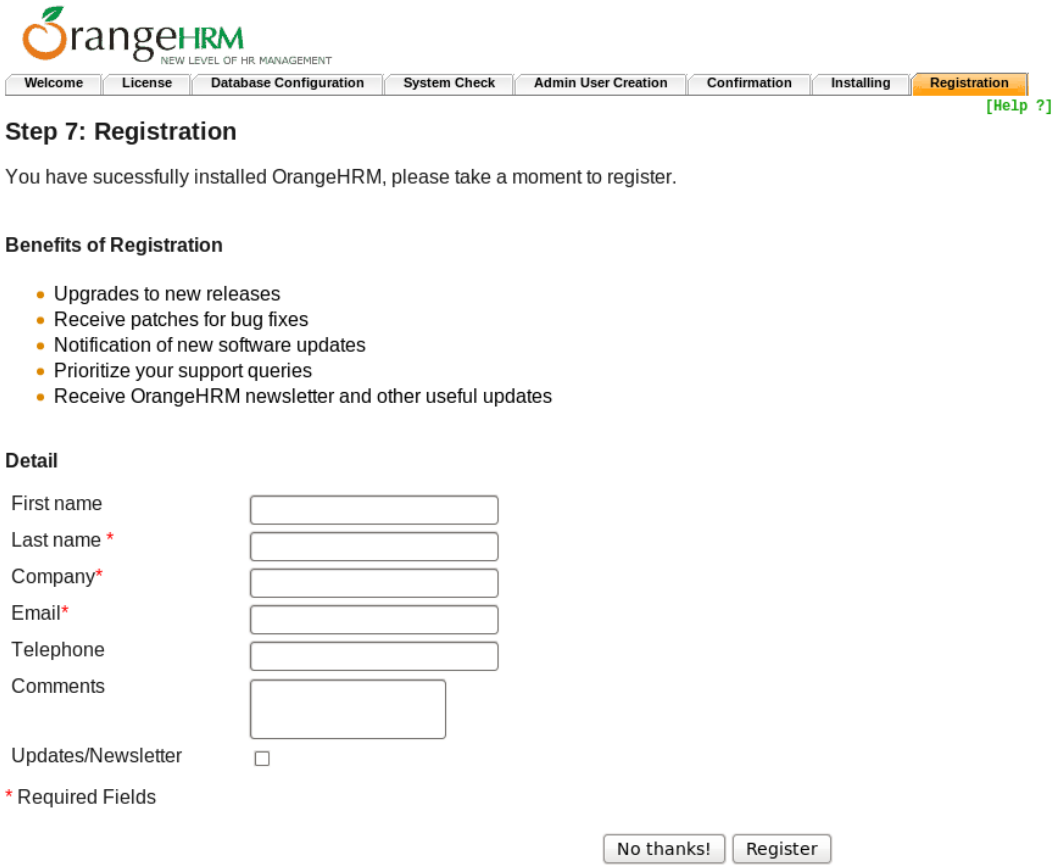
El paso 5 realizará una confirmación respecto al nombre de la base de datos, el usuario de la base de datos, el usuario administrador y el puerto en el cual correrá la aplicación.



EL paso 6 realizará la instalación verificando cada punto especificado.



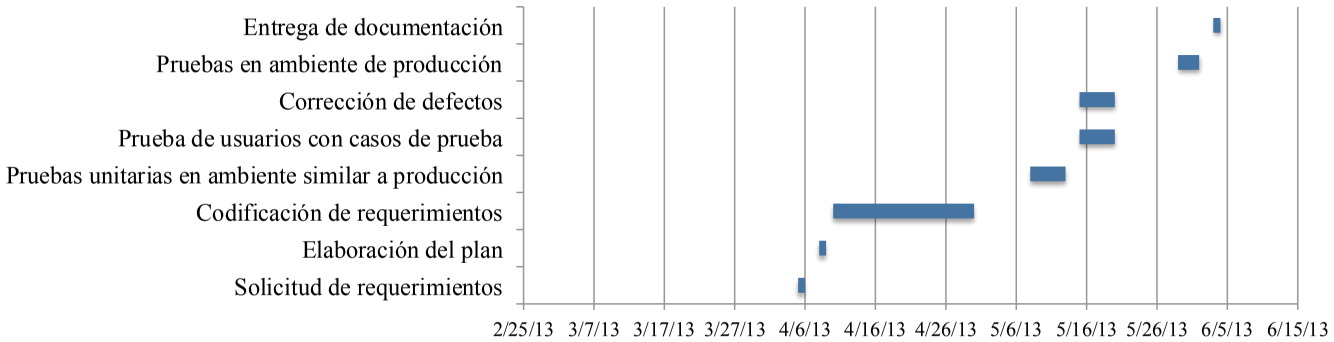
Al final de la instalación y de forma opcional, el asistente para la instalación pedirá un registro de usuario para enviar noticias, información sobre paquetes adicionales e información interesante sobre el producto.



El asistente terminará la instalación mostrando al usuario la pantalla principal de inicio de la aplicación.

CRONOGRAMA DE ACTVIDADES
(Formato usado en la investigación)

Cod	Actividad	Inicio	Duración (días)	Fin
A	Solicitud de requerimientos	4/5/13	1	4/6/13
B	Elaboración del plan	4/8/13	1	4/9/13
C	Codificación de requerimientos	4/10/13	20	5/7/13
D	Pruebas unitarias en ambiente similar a producción	5/8/13	5	5/14/13
E	Prueba de usuarios con casos de prueba	5/15/13	5	5/21/13
F	Corrección de defectos	5/15/13	5	5/21/13
G	Pruebas en ambiente de producción	5/29/13	3	5/31/13
H	Entrega de documentación	6/3/13	1	6/3/13



* Las actividades en el Diagrama de Gantt se leen de abajo hacia arriba.

ESTADO DE DEFECTOS

(Formato en blanco)

Cod	Nombre	Fecha	Descripción	Evidencia	Severidad	Abierto / Cerrado	Estado	Comentarios	Fecha Resolución	Responsable
				Si	Media	Abierto	Nuevo			
				No	Baja	Cerrado	Arreglado			

ESTADO DE DEFECTOS

(Formato usado en la investigación)

Cod	Nombre	Fecha	Descripción	Evidencia	Severidad	Abierto / Cerrado	Estado	Comentarios	Fecha Resolución	Responsable
OR01	Probador A	6/17/2013	Existe un error al ingresar a la aplicación	No	**Alta**	**Cerrado**	Arreglado	No existían permisos para escritura en la base de datos	6/18/2013	Dennis P.
OR02	Probador B	6/17/2013	Error al ingresar al módulo de Skills	No	Baja	**Cerrado**	Desecho	No era error, hubo mala ejecución del Caso de Prueba	6/17/2013	Dennis P.

CASOS DE PRUEBA

(Formato en blanco)

Proyecto:			
Caso de prueba:			
Responsable:			
Paso	**Descripción**	**Estado** (SI / NO)	**Comentarios / Evidencia**

CASOS DE PRUEBA

(Formato usado en la investigación)

Proyecto: Implementación de **Orange HRM**			
Caso de prueba: Acceso al sitio			
Responsable: Probador A			
Paso	**Descripción**	**Estado** (SI / NO)	**Comentarios / Evidencia**
1	Ingresar el sitio web de la aplicación. http://localhost/orangehrm	Si	
2	Verificar que **NO** aparezcan errores al ingresar a la pantalla principal de la aplicación.	Si	
3	Ingresar al sistema usando el ***user***: Admin y el ***password***: dpazmino	Si	
4	Verificar que **NO** se presente errores al pasar la pantalla de acceso.	Si	
5	Verificar que se visualicen de forma correcta los menús de la parte superior de la pantalla.	Si	
6	Salir de la aplicación utilizando correctamente la opción de salida.	Si	

CASOS DE PRUEBA

(Formato usado en la investigación)

Proyecto: Implementación de **Orange HRM**			
Caso de prueba: Agregar usuarios administrador			
Responsable: Probador B			
Paso	**Descripción**	**Estado** (SI / NO)	**Comentarios / Evidencia**
1	Ingresar el sitio web de la aplicación. http://localhost/orangehrm	Si	
2	Verificar que **NO** aparezcan errores al ingresar a la pantalla principal de la aplicación.	Si	
3	Ingresar al sistema usando el ***user***: Admin y el ***password***: dpazmino	Si	
4	Verificar que **NO** se presente errores al pasar la pantalla de acceso.	Si	
5	Verificar que se visualicen de forma correcta los menús de la parte superior de la pantalla.	Si	
6	Ir al menú ***Admin*** > ***Users*** > ***HR Admin Users***	Si	
7	Presione el botón ***Add***	Si	
8	Ingrese los datos mandatorios en el formulario que se muestra, al finalizar presione el botón ***Save***	Si	
9	Regresará a la pantalla anterior visualizando el usuario que se acaba de agregar.	Si	
10	Repita desde el paso 7 al 9 agregando tres usuarios más	Si	

CASOS DE PRUEBA

(Formato usado en la investigación)

Proyecto: Implementación de **Orange HRM**			
Caso de prueba: Agregar un *Skill* al sistema			
Responsable: Probador C			
Paso	**Descripción**	**Estado** (SI / NO)	**Comentarios / Evidencia**
1	Ingresar el sitio web de la aplicación. http://localhost/orangehrm	Si	
2	Verificar que **NO** aparezcan errores al ingresar a la pantalla principal de la aplicación.	Si	
3	Ingresar al sistema usando el ***user***: Admin y el ***password***: dpazmino	Si	
4	Verificar que **NO** se presente errores al pasar la pantalla de acceso.	Si	
5	Ir al menú ***Admin*** > ***Skills*** > ***Skills***	Si	
6	Ingrese los datos mandatorios en el formulario que se muestra, al finalizar presione el botón ***Save***	Si	
7	Regresará a la pantalla anterior visualizando el usuario que se acaba de agregar.	Si	

CASOS DE PRUEBA

(Formato usado en la investigación)

Proyecto: Implementación de **Orange HRM**			
Caso de prueba: Agregar un *Job* al sistema			
Responsable: Probador C			
Paso	**Descripción**	**Estado** (SI / NO)	**Comentarios / Evidencia**
1	Ingresar el sitio web de la aplicación. http://localhost/orangehrm	Si	
2	Verificar que **NO** aparezcan errores al ingresar a la pantalla principal de la aplicación.	Si	
3	Ingresar al sistema usando el ***user***: Admin y el ***password***: dpazmino	Si	
4	Verificar que **NO** se presente errores al pasar la pantalla de acceso.	Si	
5	Ir al menú ***Admin*** > ***Job*** > ***Job Titles***	Si	
6	Presione el botón ***Add***	Si	
7	Ingrese la información obligatoria y al finalizar presione el botón ***Save***.	Si	
8	Presione el botón ***Edit*** en el nuevo formulario que aparece en pantalla.	Si	
9	Ingrese la información de ***Employment Status***	Si	
10	Regresará a la pantalla principal de los ***Jobs*** visualizando el registro que se acaba de agregar.	Si	

CASOS DE PRUEBA

(Formato usado en la investigación)

Proyecto: Implementación de **Orange HRM**			
Caso de prueba: Agregar la información general de la compañía			
Responsable: Probador D			
Paso	**Descripción**	**Estado** (SI / NO)	**Comentarios / Evidencia**
1	Ingresar el sitio web de la aplicación. http://localhost/orangehrm	Si	
2	Verificar que **NO** aparezcan errores al ingresar a la pantalla principal de la aplicación.	Si	
3	Ingresar al sistema usando el ***user***: Admin y el ***password***: dpazmino	Si	
4	Verificar que **NO** se presente errores al pasar la pantalla de acceso.	Si	
5	Ir al menú ***Admin*** > ***Company Info*** > ***General***	Si	
6	Presione el botón ***Edit*** al final del formulario, e ingrese los datos mandatorios, al finalizar, presione el botón ***Save***	Si	

CASOS DE PRUEBA

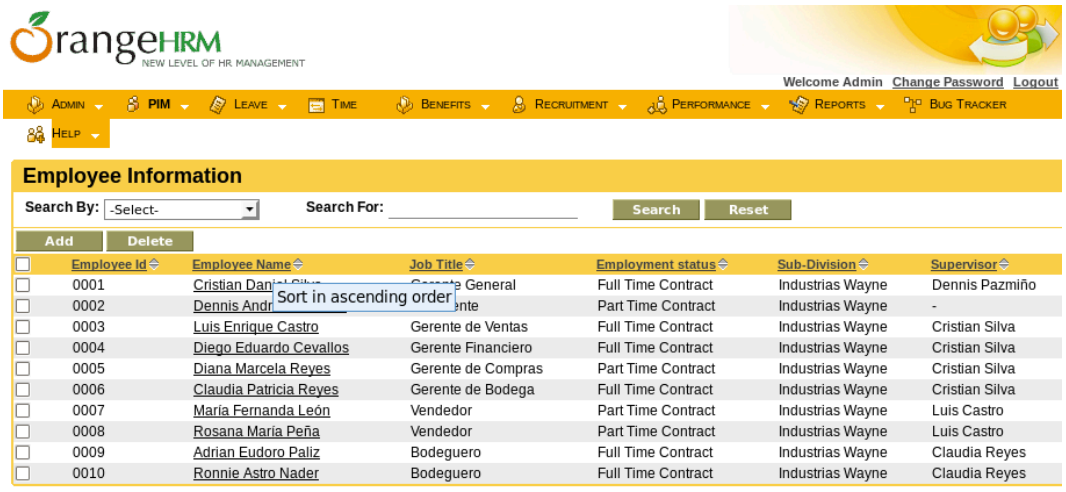
(Formato usado en la investigación)

Proyecto: Implementación de **Orange HRM**			
Caso de prueba: Agregar empleados al sistema			
Responsable: Probador D			
Paso	**Descripción**	**Estado** (SI / NO)	**Comentarios / Evidencia**
1	Ingresar el sitio web de la aplicación. http://localhost/orangehrm	Si	
2	Verificar que **NO** aparezcan errores al ingresar a la pantalla principal de la aplicación.	Si	
3	Ingresar al sistema usando el ***user***: Admin y el ***password***: dpazmino	Si	
4	Verificar que **NO** se presente errores al pasar la pantalla de acceso.	Si	
5	En la pantalla principal del sistema, (*Employee Information*) presione el botón ***Add*** y agregue 10 usuarios.	Si	
6	Complete la información de cada usuario.	Si	

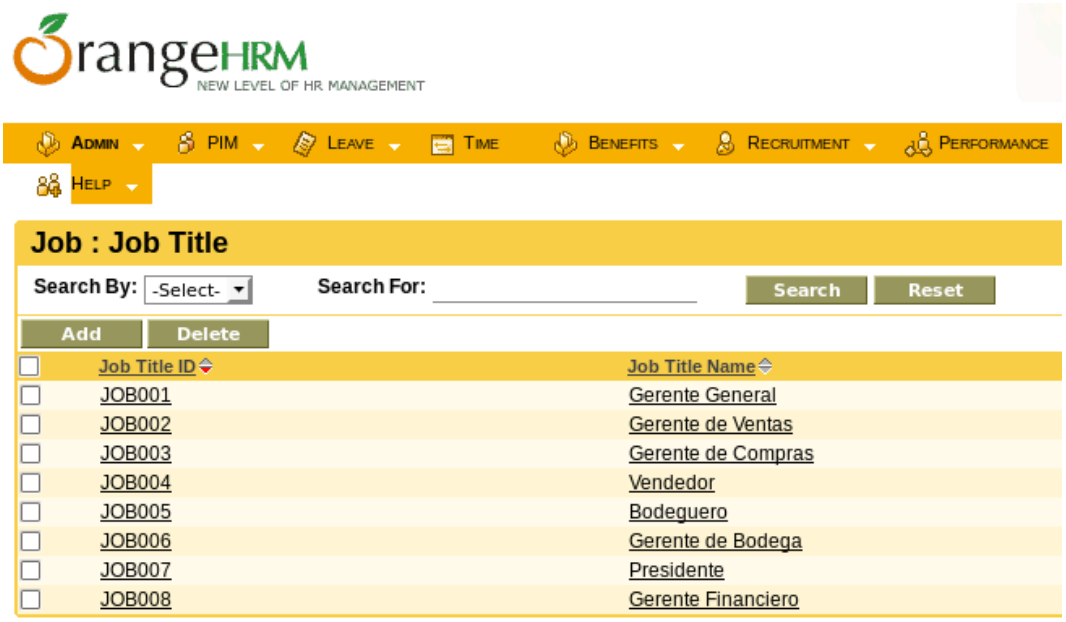
CAPTURAS DE PANTALLA

Las siguientes capturas de pantalla han sido tomadas de la aplicación, evidenciando la realización de los casos de prueba propuestos.

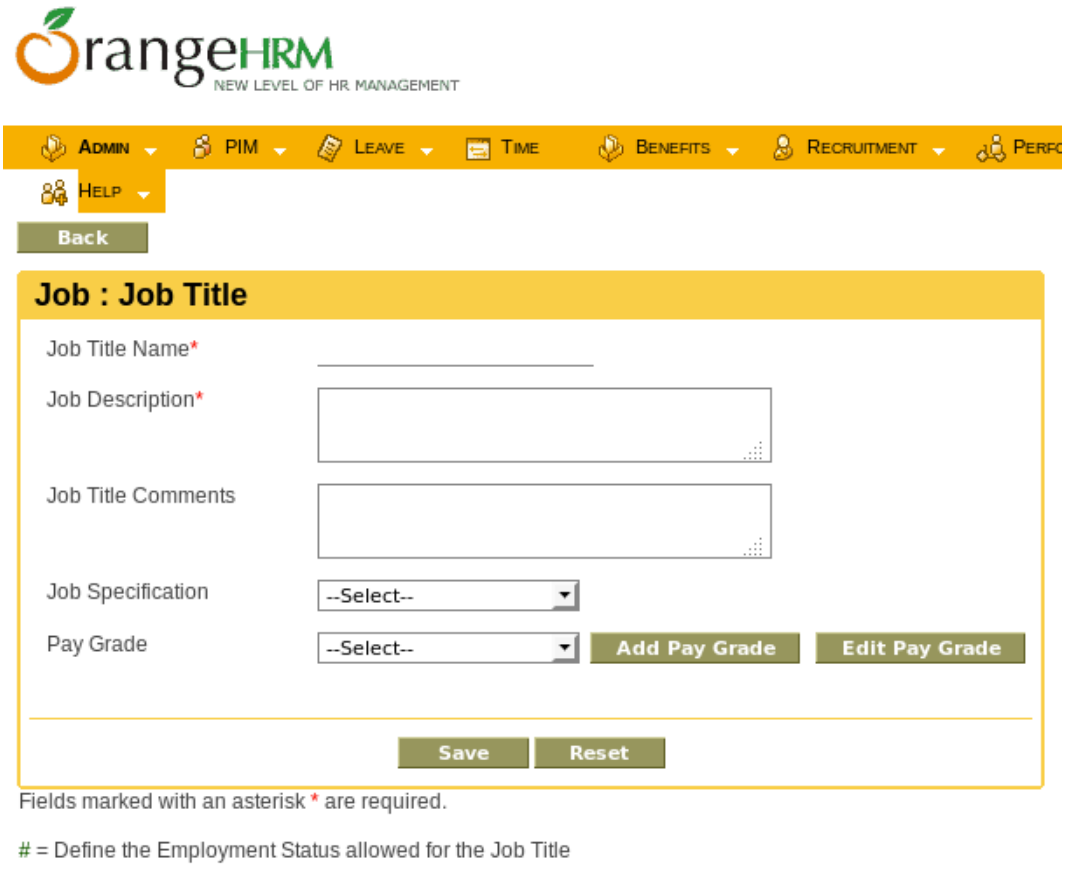
Pantalla de inicio de la aplicación



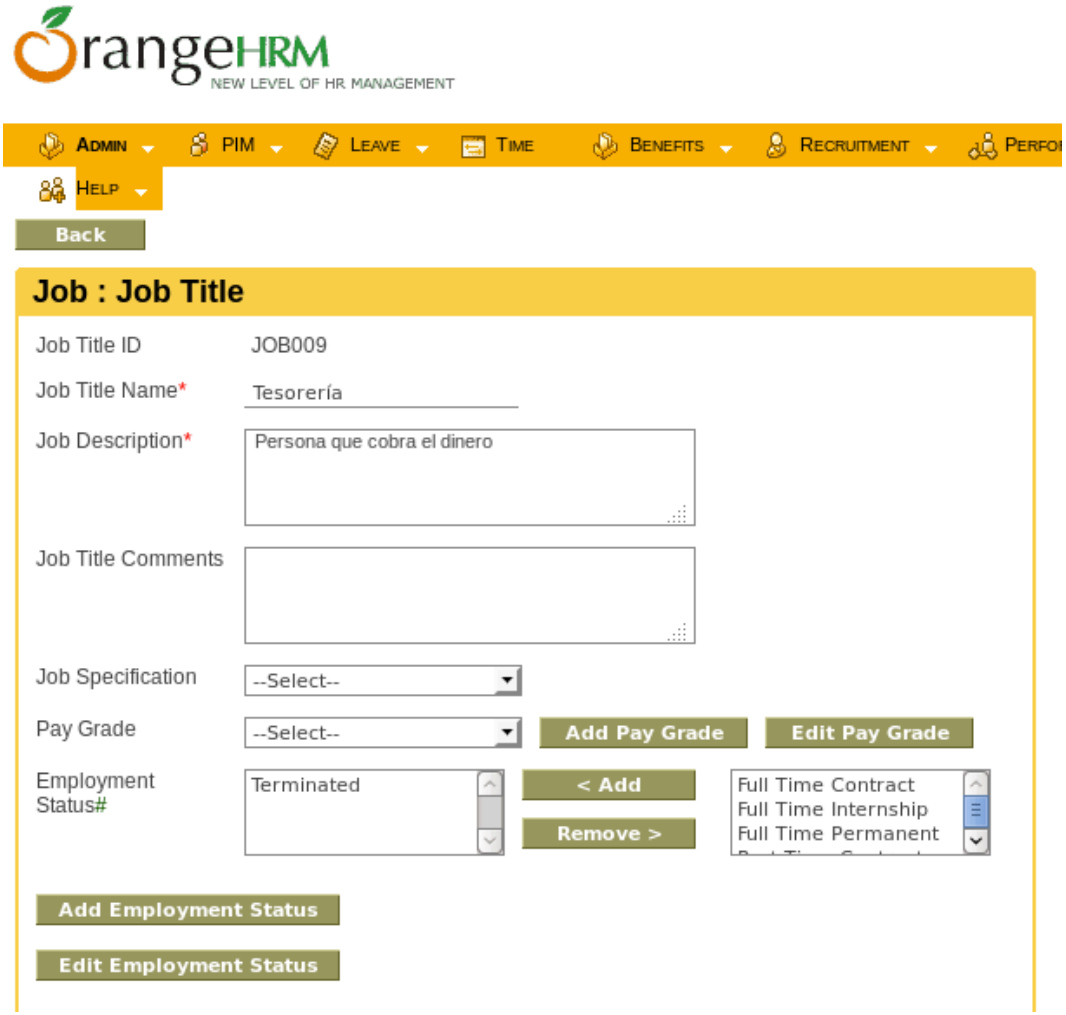
Pantalla principal luego del acceso



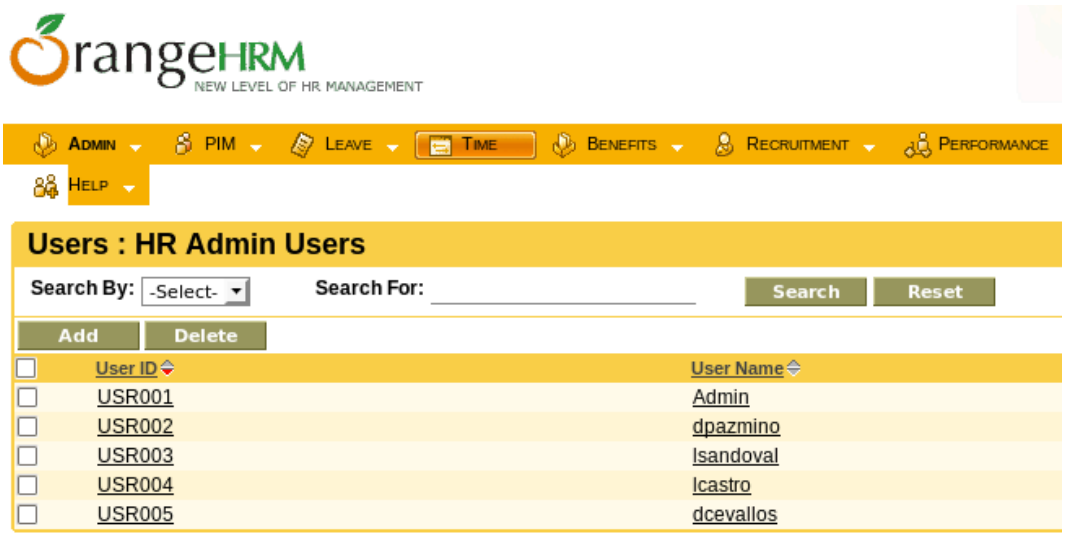
Pantalla donde se muestran los *Jobs* creados



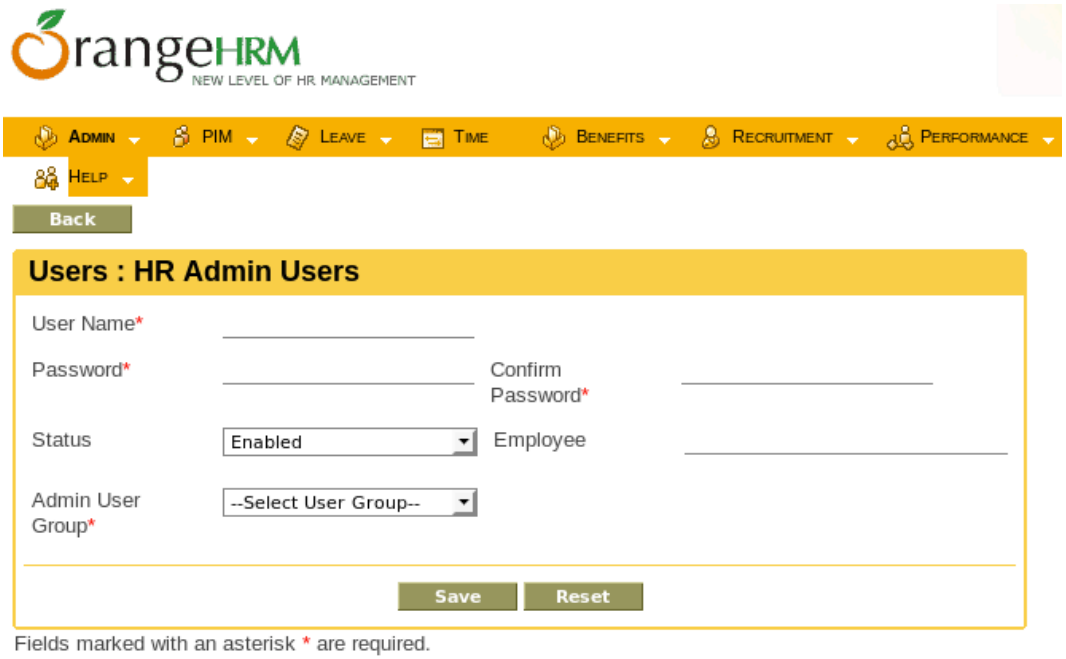
Pantalla donde se ingresa la información de un nuevo *Jobs*



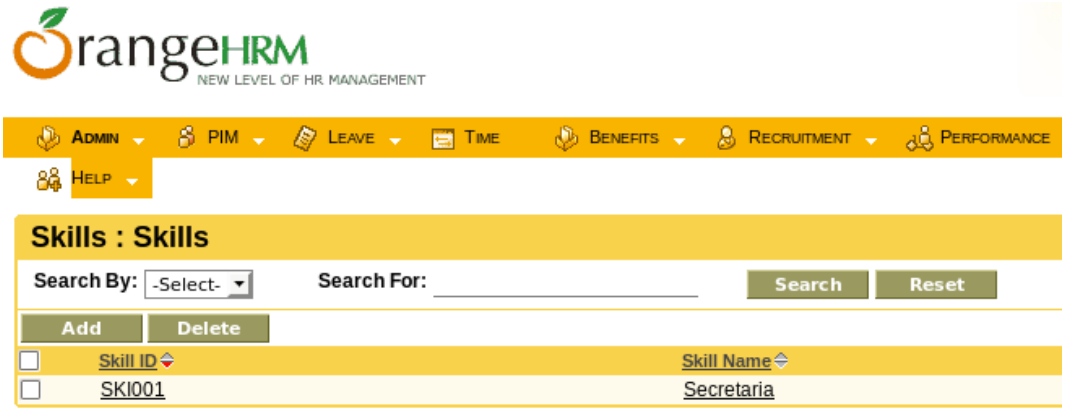
Pantalla donde se complementa la información cuando se ingresa un nuevo *Job*



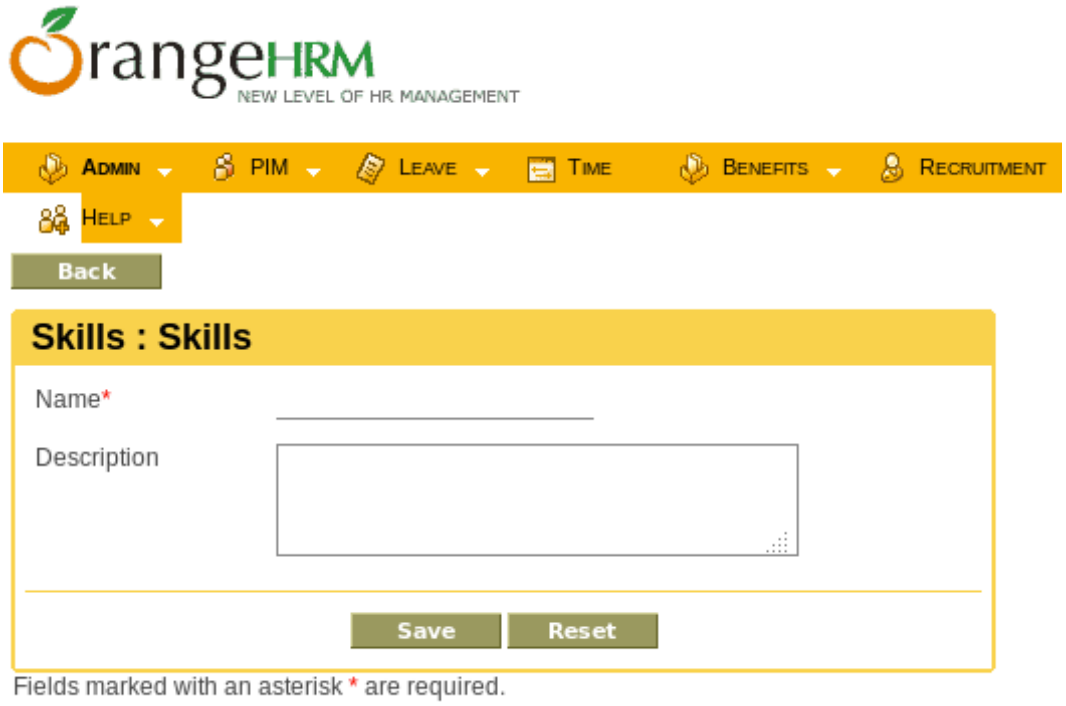
Pantalla donde se muestran los usuarios administradores creados



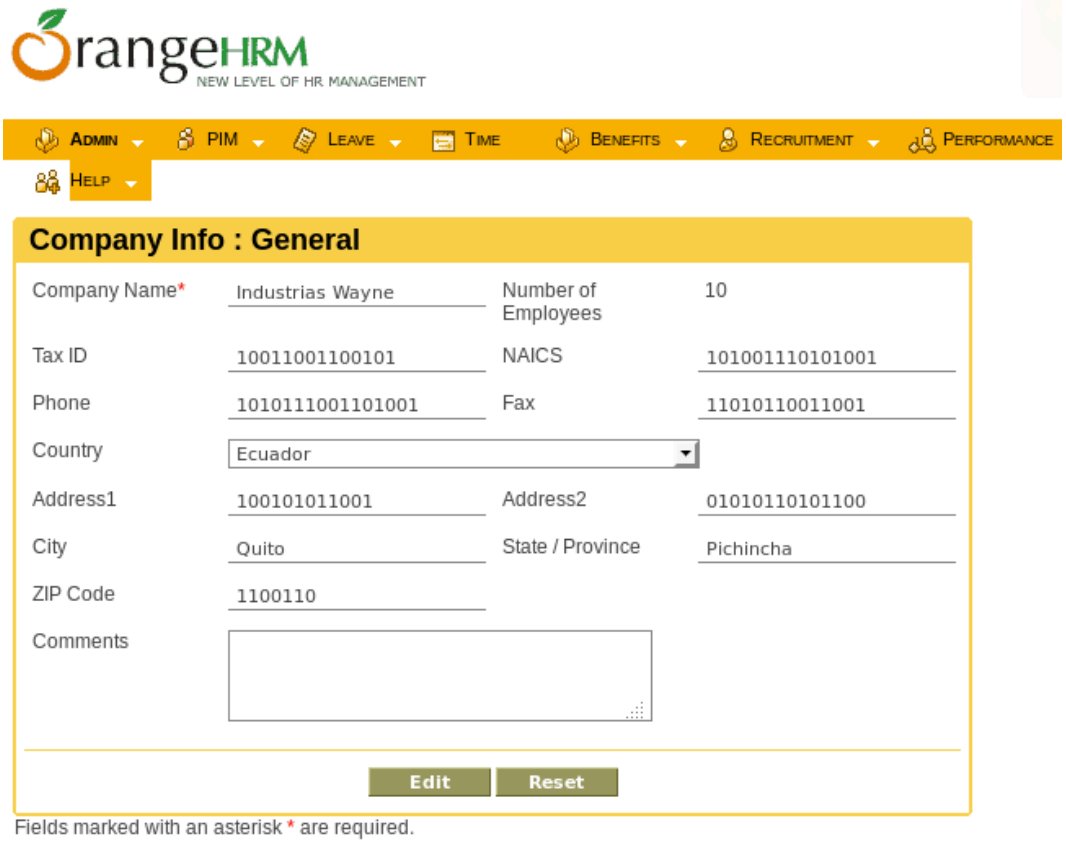
Pantalla donde se ingresan los datos para la creación de un nuevo usuario administrador



Pantalla donde se muestran los *Skills* creados



Pantalla donde se ingresa un nuevo *Skill*



Pantalla donde se ingresa los datos generales de la empresa para la cual es implementada la solución.