ADAPTACIÓN DEL MODELO DE NEGOCIO DE LA INDUSTRIA INMOBILIARIA ECUATORIANA AL SISTEMA LIBRE OPENERP PARA LA COMERCIALIZACIÓN DE BIENES INMUEBLES.

Israel Paredes Reyes[1], Katherine Robles Arévalo[2], Jenny Ruiz Robalino[3], Diego Marcillo[4]

1 Escuela Politécnica del Ejército, Ecuador, israelparedesreyes@gmail.com

2 Escuela Politécnica del Ejército, Ecuador, katherine.robles@live.com

3 Escuela Politécnica del Ejército, Ecuador, jaruiz@espe.edu.ec

4 Escuela Politécnica del Ejército, Ecuador, dmmarcillo@espe.edu.ec

RESUMEN

Las inmobiliarias en el Ecuador no cuentan con un sistema que integre la gestión de inventarios con la gestión de los clientes. El presente artículo describe el proceso de desarrollo del sistema denominado COMMIN. El objetivo es definir la adaptación del modelo de negocio de la industria inmobiliaria ecuatoriana al sistema libre OpenERP para la comercialización de bienes inmuebles. El propósito es optimizar el proceso que realizan las inmobiliarias del Ecuador ubicadas en la ciudad de Quito para poder comercializar bienes inmuebles. El sistema cuenta con dos módulos, el módulo de construcción y el módulo de comercialización. Para su análisis y desarrollo, primero se realizó el levantamiento de los requerimientos del software utilizando la Norma IEEE 830, luego se utilizó la metodología ágil XP (Programación Extrema) para tener una mejor forma de entendimiento de los requerimientos y a la vez retroalimentarse. Como resultado del presente proyecto se obtuvo un sistema adaptado a OpenERP que integra el departamento de construcciones con el departamento comercial, mejorando la comunicación interna dentro de la empresa, mejorando así los tiempos de sus procesos comerciales. Como conclusión, el presente proyecto es totalmente escalable y se puede integrar al resto de módulos del OpenERP permitiendo así la extensión a procesos más grandes como por ejemplo financiero o gestión de reclamos.

Palabras Clave: Inmobiliaria, OpenERP, módulo, metodología, XP

ABSTRACT

The estate in Ecuador does not have a system that integrates inventory management to customer management. This article describes the development process of the system called COMMIN. The aim is to define the business model adaptation estate industry Ecuadorian Free OpenERP system for marketing real estate. The purpose is optimizing the process of conducting Ecuador's real estate located in the city of Quito in order to market real estate. The system has two modules, the construction module y the marketing module. For analysis and development, we first conducted a survey of software requirements using IEEE Standard 830 and then used agile methodology XP (Extreme Programming) to have a better way of understanding of the requirements and also feedback. As a result of this project was obtained OpenERP system adapted to integrating the building department with the sales department, improving internal communication within the company, thus improving the time of their business processes. In conclusion, this project is fully scalable and can be integrated to other OpenERP modules allowing the extension to larger processes such as financial or claims management.

Keywords: Real Estate, OpenERP, module, methodology, XP

1. INTRODUCCIÓN

Actualmente la mayoría de empresas productoras, comerciantes y de servicio, utilizan aplicaciones informáticas para el control de sus procesos. Generalmente estos sistemas se especializan en una parte del negocio ya sea ventas, contabilidad, compras, entre otras; y pueden encontrarse implementados en ambientes cliente/servidor o directamente en la internet; sin embargo, este tipo de sistemas no aportan muchas ventajas en comparación a los sistemas netamente centralizados, ya que gran cantidad de los procesos se encuentran distribuidos y no se posee una conexión entre ellos. (Paredes Reyes & Robles Arévalo, 2013)

Las empresas se han visto en la necesidad de automatizar sus procesos por medio de aplicaciones informáticas. Existen muchas soluciones en el mercado, en su gran mayoría son privativas, cuyo soporte y mantenimiento le obliga al cliente a contratar nuevamente un proveedor específico convirtiéndose en una monopolización a costos demasiados elevados. (Paredes Reyes & Robles Arévalo, 2013)

Para el desarrollo del sistema de comercialización de bienes inmuebles, se realizó una investigación basada en las necesidades de una inmobiliaria en el Ecuador, específicamente en la ciudad de Quito para saber los problemas a los cuales se enfrentan. Para esto se programaron reuniones y levantamientos funcionales y se llegó a la conclusión de que las inmobiliarias presentan varios riesgos. Entre estos problemas está el de no poseer un módulo que les permita comercializar su producto final basado en la producción de los mismos, en dicho caso los inmuebles, porque en la actualidad toda la humanidad se encuentra en una era digital y ningún proceso debería ser manual, sino automatizado.

Para afrontar los problemas anteriormente mencionados se ha incorporado un sistema piloto para la construcción y comercialización de bienes inmuebles para las distintas inmobiliarias del Ecuador. Para el levantamiento de los requerimientos del software se utilizó el estándar IEEE 830 y la metodología ágil Programación Extrema (XP) para análisis y desarrollo.

2. METODOLOGÍA

Para el desarrollo del sistema de Adaptación del modelo de negocio de la industria inmobiliaria ecuatoriana al sistema libre OpenERP para la comercialización de bienes inmuebles, que se denominará COMMIN[1], se decidió utilizar una metodología ágil, la metodología es Programación Extrema (XP[2]), se escogió por las características (Beck, Extreme Programming Explained, 1999) que presenta:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.
- Programación en parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto para que el código sea revisado y discutido, mientras se escribe de esta manera se tiene más productividad.
- Frecuente integración del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve que todo el personal pueda corregir y extender cualquier parte del proyecto.
- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario.
- La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Cuanto más simple es el sistema, menos tendrá que comunicar sobre éste, lo que lleva a una comunicación más completa.

Se decidió utilizar una metodología ágil porque el proceso de implementación del sistema requería del menor tiempo posible y los requisitos de cada proceso no se encontraban totalmente definidos por parte de los usuarios lo cual implicaría cambios en un futuro. Adicionalmente el proyecto fue realizado por un equipo de trabajo y la manera de trabajar por historias de usuario facilitó el desarrollo colaborativo. Y gracias al pensamiento de Kent Beck[3] quien dijo: “Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar”. (Beck, Extreme Programming Explained, 1999), se optó por usar una metodología ágil.

Entre las diferentes metodologías ágiles existentes se escogió XP, porque “La Programación Extrema es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. Los Valores originales de la programación extrema son: simplicidad, comunicación, retroalimentación (feedback) y coraje. Un quinto valor, respeto, fue añadido en la segunda edición de Extreme Programming Explained.” (Beck, Extreme Programming Explained, 1999)

[1] COMMIN®: Denominación del presente proyecto para la construcción y comercialización de bienes inmuebles.
[2] XP: Extreme Programming
[3] Kent Beck: Creador de la metodología XP

El objetivo de la metodología XP es aumentar la productividad al desarrollar el software. Se basa en la adaptabilidad de cualquier cambio como medio para aumentar las posibilidades de éxito de un proyecto.

FASES DE LA METODOLOGÍA XP

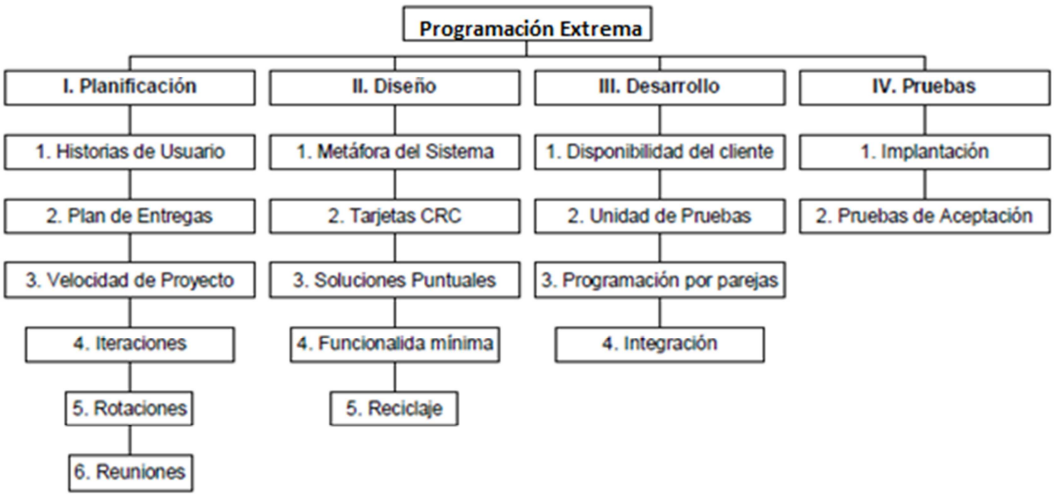

Figura 1: Fases de la metodología XP (Beck, Extreme Programming Explained, 1999)

El cliente recibe todos los entregables de esta metodología por parte de los desarrolladores del sistema.

1ª Fase: Planificación del proyecto (Paredes Reyes & Robles Arévalo, 2013)**.**

1. **Historias de usuario**, se realizó un formato creado por los desarrolladores del sistema para especificar las necesidades escritas por los usuarios.
2. **Planes de entregas**, se realizó una estimación de tiempo para la realización de cada historia de usuario creado. Se usó un formato realizado por los desarrolladores del presente proyecto.
3. **Velocidad del Proyecto**, se realizó una estimación del tiempo para la realización del proyecto. En esta sección se usó un formato realizado por los desarrolladores del presente proyecto.
4. **Iteraciones**, se identificó las distintas historias de usuario que se van a desarrollar en una iteración específica. En esta sección se identificó 3 iteraciones por cada historia de usuario.
5. **Rotaciones**, es la forma como va variando el proyecto. En esta sección se encontraron algunas variaciones, porque los usuarios finales pedían varios cambios al momento de la implementación del sistema, porque existió ambigüedad por parte del cliente al momento de levantar los requerimientos.
6. **Reuniones diarias**, con el fin de facilitar la comunicación entre el grupo de trabajo y el cliente se realizaron reuniones diarias con el Jefe de Sistemas de la empresa, también se hicieron reuniones con los usuarios finales para el levantamiento de los requerimientos para tener un mejor enfoque del proceso que ellos siguen para la construcción y comercialización de bienes inmuebles.

2ª Fase: Diseño (Paredes Reyes & Robles Arévalo, 2013)**.**

1. **Metáfora del sistema**, se hizo uso de los estándares de programación para facilitar el manejo consistente de los nombres de las clases y los métodos.
2. **Tarjetas C.R.C.**[4], no se usó. Porque en esta metodología no se está obligado a cumplir con todas las fases, y con las historias de usuario fue suficiente.
3. **Soluciones puntuales**, se propusieron soluciones puntuales a los distintos problemas técnicos o de diseño que se presentaron en su momento. En esta sección se realizaron varias reuniones para dar solución a los diferentes problemas presentados ante los requerimientos tanto en análisis como en diseño.
4. **Funcionalidad mínima**, en esta sección se usó la funcionalidad requerida tanto por parte de los usuarios como de los desarrolladores, porque al tener una funcionalidad extra sería un desperdicio de tiempo y recursos.
5. **Refactorizar**, se revisó las veces necesarias el código, para procurar optimizar su funcionamiento, sin alterar su funcionalidad. En esta sección siempre se realizaron reuniones entre los desarrolladores para optimizar recursos en cuanto al código fuente.

3ª Fase: Codificación (Paredes Reyes & Robles Arévalo, 2013)**.**

1. **Disponibilidad del cliente**, el cliente tuvo una disponibilidad baja, porque por lo general siempre se encontraban en reuniones internas. Hubo una disponibilidad de una reunión por semana.
2. **Unidad de pruebas**, se utilizaron el Plan de Pruebas[5] y Casos de Prueba[6] antes de empezar a codificar, lo cual hizo más sencillas y efectivas las pruebas.
3. **Programación por parejas**, el desarrollo del sistema se realizó en equipo de trabajo de dos personas, y luego se llevó a cabo una integración paralela. Cabe recalcar que con dicha integración no se garantiza la consistencia y la calidad, es recomendable hacer pruebas exhaustivas.
4. **Integración**, se dejó la optimización para el final. Una vez que el código requerido estuviese completo se procedió a la integración de las partes.

4ª Fase: Pruebas (Paredes Reyes & Robles Arévalo, 2013)**.**

1. **Implantación**, al momento de implementar el sistema, se realizaron plan de pruebas, casos de prueba, una vez realizadas dichas pruebas se colocaron los resultados en el documento de reporte de errores[7].
2. **Test de aceptación**, se crearon pruebas de aceptación a partir de las historias de usuario. Estos test son creados y usados por los clientes para comprobar que las distintas historias de usuario cumplen su cometido. El cliente uso sus propios formatos para validar al sistema.

3. HERRAMIENTAS

[4] CRC: Class, Responsabilities and Colaboration

[5] Plan de Pruebas: Es el documento en el cual se establece un estándar de pruebas que se va aplicar al presente proyecto, con el fin de validar el cumplimiento de los requerimientos planteados en función de la IEEE 830.

[6] Casos de Prueba: Es el documento en el cual consta la forma como se llevaran a cabo las distintas pruebas por cada requerimiento planteado. En esta sección se hizo uso de un formato propio para los casos de prueba.

[7] Reporte de errores: Es un documento en el cual se detallan los diferentes errores que se presentaron al momento de realizar las pruebas, y si se corrigieron o no.(Referencia documento tal)

Las herramientas y tecnologías de software libre que se utilizaron para el desarrollo de esta propuesta, acordes a las necesidades el proyecto y al framework OpenObject[8], son las que se describen a continuación.

3.1 Lenguaje de programación: Python

Al utilizar OpenERP se requiere tener conocimientos en el lenguaje de programación Python ya que es la base de OpenObject, sin embargo en el proceso de implementación se encontraron grandes ventajas del lenguaje frente a otros, la principal es la facilidad de escribir código como si de pseudo código se tratase.

Python[9] es un lenguaje de programación de alto nivel cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y conteo de referencias para la administración de memoria, es fuertemente tipado y multiplataforma. (González Duque) **,** (Van Rossum, 2005)

3.2 Base de datos: PostgreSQL (Robles Prado, 2002)

PostgreSQL[10] es el motor de base de datos que utiliza OpenObject, éste incluye la capa de persistencia que maneja la base de datos, todos los objetos dentro de OpenObject heredan de la clase **ORM[11]** que posee métodos para realizar CRUD a las tablas de Postgres por lo que el desarrollador no requiere construir esta conexión, es decir no es necesario hacer un pool de conexiones hacia la base de datos, porque ya viene dada por OpenObject.

PostgreSQL es un sistema de base de datos relacional libre (open source) que utiliza OpenObject para representar el modelo relacional de los sistemas administrativos. PostgreSQL, tiene más de 15 años de activo desarrollo y arquitectura probada, por lo que se considera un sistema seguro y confiable para la integridad de los datos. Funciona en todos los sistemas operativos importantes, incluyendo Linux[12], UNIX[13] y Windows[14].

Además, el código fuente de PostgreSQL está disponible bajo la licencia BSD, esta licencia proporciona la libertad de usar, modificar y distribuir PostgreSQL. Todas las modificaciones, mejoras, o cambios que se realicen quedan a decisión del usuario de publicar o no hacerlo.

3.3 Bitbucket: Repositorio de versiones

Se usó Bitbucket entre los diferentes repositorios disponibles porque es un repositorio en la nube que permite el trabajo colaborativo de código de hasta 5 personas en sus cuentas gratuitas,

[8] OpenObject: Framework de OpenErp para el desarrollo de sistemas.
[9] Python: Lenguaje de Programación interpretado.
[10] PostgreSQL: Base de Datos relacional
[11] ORM: Mapeo Objeto-Relacional (Object-Relational mapping), técnica de programación para convertir datos entre un lenguaje de programación y una base de datos usando un motor de persistencia.
[12] Linux: Núcleo del sistema operativo libre basado en Unix.
[13] UNIX: Sistema operativo libre
[14] Windows: Corresponde al nombre de una familia de sistemas operativos privados comercializados por la empresa Microsoft.

además permite definir si un repositorio es privado o compartido; al ser un proyecto de tesis cuyo trabajo fue realizado por 2 personas, bitbucket era la solución perfecta.

Bitbucket es un servicio de alojamiento basado en web, para los proyectos que utilizan el sistema de control de revisiones Mercurial y Git. El servicio está escrito en Python.

3.4 Mercurial: Control de versiones

Se utilizó Mercurial porque el repositorio Bitbucket trabaja con este control de versiones y además esta implementado haciendo uso del lenguaje de programación Python, para el control de versiones por parte de los desarrolladores del sistema. Por medio de este sistema de control de versiones multiplataforma, se tiene una información detallada de los cambios que se suben al repositorio. En el caso de presentarse errores, se puede evaluar y de ser el caso retornar a versiones anteriores de código.

Mercurial fue escrito originalmente para funcionar sobre Linux. Ha sido adaptado para Windows, Mac OS X y la mayoría de otros sistemas tipo Unix. Mercurial es, sobre todo, un programa para la línea de comandos. El creador y desarrollador principal de Mercurial es Matt Mackall[15]. El código fuente se encuentra disponible bajo los términos de la licencia GNU GPL[16] versión 2, lo que clasifica a Mercurial como software libre. (Tiny, 2009)

Para el acceso a repositorios mediante red, Mercurial usa un protocolo eficiente, basado en HTTP[17], que persigue reducir el tamaño de los datos a transferir, así como la proliferación de peticiones y conexiones nuevas. Mercurial puede funcionar también sobre ssh, siendo el protocolo muy similar al basado en HTTP.

4. ANÁLISIS, DISEÑO E IMPLEMENTACIÓN

Para el análisis, diseño e implementación del sistema, se siguió un orden específico. Para el análisis, primero se realizó el levantamiento de los requisitos utilizando el estándar IEEE 830, luego se realizó las historias de usuario por cada requerimiento planteado. Para el Diseño, primero se realizó el modelado de los casos de uso con su respectiva especificación, luego se realizó el modelado del sistema (modelo conceptual, lógico y físico de datos). Una vez implementado el sistema, se realizaron el plan de pruebas, los casos de pruebas, el reporte de errores, para validar el cumplimento de los requerimientos planteados por parte del cliente.

4.1 Arquitectura MVC de openObject (Pinckaers, Gardiner, & Van Vossel, 2011)

OpenERP utiliza el patrón de arquitectura de software Modelo Vista Controlador como se observa en la Figura 2.

[15] MattMackall: Creador y desarrollador principal de Mercurial.
[16] GPL: General Public License
[17] HTTP: Hypertext Transfer Protocol

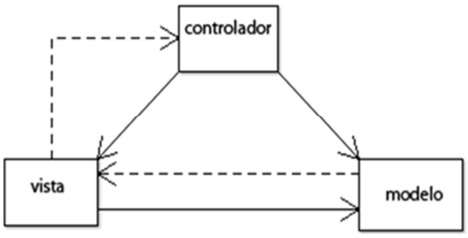

Figura 2: Modelo Vista Controlador (Pinckaers, Gardiner, & Van Vossel, 2011)

Modelo Vista Controlador (MVC), es un patrón de arquitectura de software que separa en tres componentes distintos:

- **Modelo:** Los datos de la aplicación
- **Vista:** La interfaz de usuario
- **Controlador:** La lógica de control

El patrón MVC se ve frecuentemente en aplicaciones web donde:

- La vista es la página HTML[18] y el código que provee de datos dinámicos a la página.
- El modelo es el Sistema de Gestión de Base de Datos y la lógica de negocio.
- El controlador es el responsable de recibir los eventos de entrada desde la vista, consultar datos del modelo, realizar los cálculos necesarios y solicitar nuevas vistas.

4.2 Modelo Vista Controlador en OpenERP (Pinckaers, Gardiner, & Van Vossel, 2011)

En OpenERP, se puede aplicar este modelo-vista-controlador de la siguiente manera:

- **Modelo (model):** Los objetos de OpenERP con sus columnas que normalmente se guardan en las tablas de PostgresSQL con sus campos. Permite la creación/actualización automática de las tablas y acceder a las tablas sin usar SQL.
- **Vista (view):** Listas, formularios, calendarios, gráficos, definidas en archivos XML. En estos archivos también se definen menús, acciones, informes, asistentes, etc.
- **Controlador (controller):** Métodos Python definidos dentro de los objetos de OpenERP que proporcionan la lógica: Validación de datos, cálculos, etc.

4.3 Arquitectura técnica OpenERP (Cliente – Servidor) (Pinckaers, Gardiner, & Van Vossel, 2011)**,** (Tiny, 2009)

OpenERP posee una arquitectura multiusuario de tres capas. La capa de negocio se define como un núcleo permitiendo así adicionar múltiples módulos, dando una funcionalidad diferente a OpenERP como se observa en la Figura 3.

[18] HTML: Hypertext Markup Language

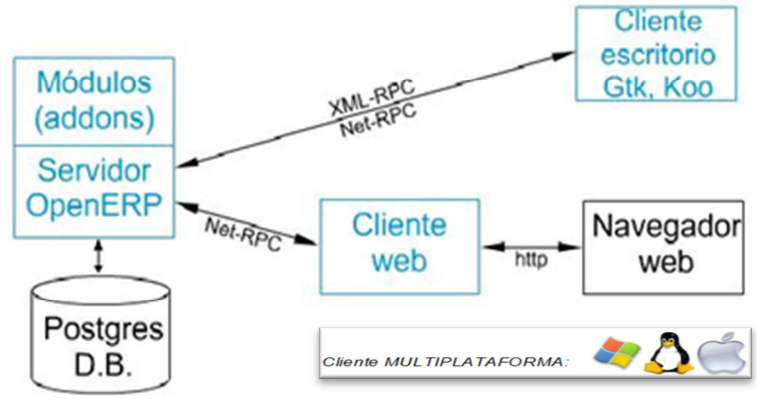

Figura 3: Arquitectura técnica OpenERP (Tiny, 2009)

El lenguaje para escribir los módulos es Python. La funcionalidad de los módulos es expuesta utilizando el protocolo XML-RPC y NET-RPC (configurable en el servidor). Los módulos hacen uso de la herencia de las clases ORM de OpenERP para persistir los datos en Postgres. Los módulos pueden insertar datos en su instalación por medio de archivos XML, CPV y YML.

5. RESULTADOS

Se obtuvieron los siguientes resultados:

- Se logró implementar el sistema de comercialización de inmuebles en un 100% a una empresa cuyo negocio principal es la industria inmobiliaria.
- Se logró obtener un sistema totalmente escalable, cuyo núcleo se basa en la producción y comercialización de bienes inmuebles.
- Satisfacción de los usuarios, debido a la interfaz intuitiva y el fácil manejo de la plataforma, porque cuenta con diferentes menús para todo el proceso de construcción y comercialización.
- Los tiempos de respuesta son favorables, oscila entre los 3-5 segundos por petición. Con el sistema que utilizaban antes en la empresa inmobiliaria, el tiempo de respuesta era desesperante, porque por petición se demoraba 1-2 minutos para una simple consulta, como por ejemplo revisar cuantos inmuebles están creados, esto causaba malestar por parte de los usuarios finales.
- Reducción en tiempos de desarrollo de software en un 75%, gracias al uso de la plataforma OpenObject.

6. TRABAJOS RELACIONADOS

"Implantación de una herramienta ERP software libre y desarrollo del anexo transaccional para la Empresa de Distribución de Leche Andina para Imbabura". Trabajo realizado por Angela Natalia Rojas Tobar para la universidad Técnica del Norte donde se adapta un flujo de negocio a 3 sistemas ERP libres, uno de ellos OpenERP. El proyecto fue realizado en la ciudad de Ibarra en Junio del 2011.

"Estudio e implementación del sistema openERP en la Empresa de Economía Solidadaria Pakariñan Turismo comunitario". Trabajo realizado por Jorge Luis Bravo Campoverde, Edwin

Alejandro Tenesaca Gómez, en donde se adaptan algunos módulos y se desarrollan nuevos. El trabajo fue realizado para la universidad de Cuenta.

Respecto a la implementación de un sistema a la industria inmobiliaria ecuatoriana no se han encontrado trabajos relacionados.

7. CONCLUSIONES Y TRABAJO FUTURO

Al utilizar la plataforma OpenObject para el desarrollo del sistema, se logró que el manejo de la base de datos sea transparente para el desarrollador, gracias a las funciones ORM definidas por la plataforma. OpenObject al ser una plataforma de desarrollo, utiliza sus propias definiciones para mostrar al usuario final cada campo dependiendo de su tipo, si se necesita cambiar la forma de visualizar la misma se debe modificar el núcleo del cliente Web. El uso del lenguaje de programación Python facilitó el desarrollo del sistema, porque es un lenguaje fácil de entender al momento de programar, a diferencia de otros lenguajes las sentencias no terminan con un punto y coma porque cuenta con indexado obligatorio; tampoco necesita compilación para el control de errores, porque estos se realizan en tiempo real.

El trabajo futuro consiste en integrar este proyecto llamado COMMIN a otros módulos como son el módulo de inventarios para saber con cuántos inmuebles cuenta actualmente la empresa, y con el módulo de contabilidad para tener todo el manejo de la parte contable en función de la venta de dichos inmuebles.

8. REFERENCIAS BIBLIOGRÁFICAS

Beck, K. (199). *Extreme Programming Explained.* Embrace Change.

Gili Busquet, R. (2010, 11 20). Análisis del mercado inmobiliaria urbano residencial. *El Comercio*, p. 2.

González Duque, R. (n.d.). *Python para todos.*

Paredes Reyes, I., & Robles Arévalo, K. (2013, 08 16). Adaptación del modelo de negocio de la industria ecuatoriana inmobiliaria al sistema libre OpenERP para la comercialización de bienes inmuebles. Quito, Pichincha, Ecuador: ESPE.

Pinckaers, F., Gardiner, G., & Van Vossel, E. (2011). *Open ERP Version 6: a modern approach to integrated business management.*

Robles Prado, T. (2002). *Introducción a Python y PostgreSQL.*

Tiny, S. (2009). *Open Object Developer Book.*

Van Rossum, G. (2005). *Guía de Aprendizaje de Python. Python Software Foundation.*