



ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
Facultad de Ingeniería en Electricidad y Computación

INFORME DE PROYECTO DE GRADUACIÓN

**“INGENIERÍA INVERSA DE UN SOFTWARE PARA
MODELAR PROCESOS DE NEGOCIO BASADO EN
UNA NOTACION PROPIETARIA, UTILIZANDO
NOTACIÓN UML”**

Previo a la obtención del Título de:

**INGENIERO EN CIENCIAS COMPUTACIONALES
ORIENTACIÓN SISTEMAS DE INFORMACIÓN**

Presentado por:

HARRY ALBERTO CARPIO SALVATIERRA

GUAYAQUIL - ECUADOR

AÑO: 2015

AGRADECIMIENTO

A mis padres y hermanos por ser un apoyo constante e incondicional en todo momento.

Al PhD. Carlos Monsalve por la confianza transmitida para la realización de este proyecto.

Al Ing. Jorge Rodríguez por su predisposición, colaboración y guía.

Al MSc. Carlos Mera por su extraordinaria labor docente.

Al Ing. Francisco Ramírez por los conocimientos y consejos compartidos.

A quienes fueron mis profesores durante mi formación académica.

DEDICATORIA

A mis amados padres Luis y Sara, por ser la luz de mis días, por su amor, por su ejemplo de constancia, paciencia y esfuerzo.

A mis hermanos Yajaira, Ivonne y Byron.

A Liliana, por motivarme a dar siempre lo mejor, por la paciencia y confianza.

A todos quienes aportaron y me acompañaron de una u otra forma en mi formación de pregrado.

Lo logramos.

TRIBUNAL DE SUSTENTACIÓN


MSc. Sara Rios O.

SUBDECANA DE LA FIEC



Ing. Jorge Rodríguez E.

DIRECTOR DEL PROYECTO DE GRADUACIÓN



MSc. Carlos J. Mera G.

MIEMBRO DEL TRIBUNAL

DECLARACIÓN EXPRESA

“La responsabilidad del contenido de este Informe de Proyecto de Graduación, me corresponde exclusivamente; y el patrimonio intelectual del mismo a la Escuela Superior Politécnica del Litoral”

(Reglamento de exámenes y títulos profesionales de la ESPOL)



Harry Alberto Carpio Salvatierra

RESUMEN

En el presente proyecto se realiza un estudio e ingeniería inversa al BPMS PYX4, este sistema está desarrollado sobre el framework Ruby On Rails que está escrito en lenguaje Ruby, e implementa una notación de procesos de negocio propietaria llamada Qualigram que fue producto de un proyecto internacional de investigación como parte del “European Strategic Program for Research and Development in Information Technology”.

El objetivo de este trabajo es el de proveer la documentación suficiente para entender cómo el sistema PYX4 está implementado en relación a la característica funcional del diagramado de procesos, para hacerlo se usaran herramientas de análisis automático de ingeniería inversa especializadas en Ruby On Rails y se hará uso de análisis del código fuente.

El trabajo está dividido en revisión de literatura para comprender los conceptos necesarios para desarrollar el proyecto, análisis de requerimientos en donde se delimita las características específicas a ser analizadas, recuperación de ambiente en donde se despliega el sistema a partir de su código fuente, en esta etapa se incluye un instructivo para el despliegue en el Anexo A,

recuperación del diseño en donde se grafican los diagramas UML y finalmente en el análisis de la recuperación del diseño en donde se realiza una revisión.

Para finalizar se realiza una reflexión acerca de las implicaciones del proyecto así como las recomendaciones para trabajos futuros.

ÍNDICE GENERAL

AGRADECIMIENTO	II
DEDICATORIA	III
TRIBUNAL DE SUSTENTACIÓN	IV
DECLARACIÓN EXPRESA	V
RESUMEN	VI
ÍNDICE GENERAL.....	VIII
ABREVIATURAS Y SIMBOLOGÍA	XIII
ÍNDICE DE FIGURAS.....	XIV
ÍNDICE DE TABLAS	XVI
INTRODUCCIÓN.....	XVII
1. PROBLEMA DEL PROYECTO.....	1
1.1. Antecedentes.....	1
1.2. Descripción del problema	3
1.3. Justificación	4
1.4. Propuesta y alcance	4
1.5. Objetivos.....	5
1.5.1. General	5
1.5.2. Específicos.....	5
1.6. Metodología	6

2. REVISIÓN BIBLIOGRÁFICA	7
2.1. Ingeniería inversa	7
2.1.1. Dificultades.....	13
2.1.2. Diferentes enfoques	14
2.1.2.1. Ingeniería inversa de lógica o de proceso.....	15
2.1.2.2. Ingeniería inversa de datos.....	15
2.1.2.3. Ingeniería inversa de interfaces de usuario	16
2.1.3. Herramientas disponibles.....	17
2.1.3.1. Depuradores	17
2.1.3.2. Desensambladores	17
2.1.3.3. Descompiladores	18
2.1.3.4. Herramientas de inyección de fallos	18
2.1.3.5. Herramientas CASE.....	19
2.2. Lenguaje de modelado unificado (UML)	22
2.2.1. Paradigma orientado a objetos	22
2.2.1.1. Bases del modelo de OO	23
2.2.1.2. Características relacionadas al POO	25
2.2.2. Principios de modelado usando UML.....	25
2.2.3. Notación UML	27

2.2.3.1.	Elementos de UML	27
2.2.3.2.	Relaciones en UML.....	27
2.2.3.3.	Diagramas UML	28
2.3.	Ruby y Ruby on Rails	35
2.3.1.	Ruby.....	36
2.3.1.1.	Historia.....	36
2.3.1.2.	Enfoque.....	39
2.3.2.	Ruby on Rails	40
2.3.2.1.	Historia.....	41
2.3.2.2.	Enfoque.....	42
2.3.2.3.	Arquitectura.....	43
2.3.2.3.1.	La arquitectura MVC	43
2.3.2.3.2.	MVC en RoR.....	45
2.3.2.3.3.	Peticiones HTTP en RoR	47
3.	ANÁLISIS Y PREPARACIÓN DE AMBIENTE	49
3.1.	Análisis de ingeniería inversa	49
3.2.	Análisis de requerimientos.....	52
3.2.1.	Objetivo	52
3.2.2.	Alcance	52
3.2.3.	Descripción general.....	52

3.2.3.1.	Perspectiva del producto.....	52
3.2.3.1.1.	Interfaces del sistema.....	52
3.2.3.1.2.	Interfaces de usuario	53
3.2.3.1.3.	Interfaces de hardware	53
3.2.3.1.4.	Interfaces de comunicaciones	53
3.2.3.1.5.	Operaciones	53
3.2.3.2.	Funciones del producto.....	54
3.2.3.3.	Características de usuario	55
3.2.3.4.	Limitaciones	55
3.3.	Preparación de ambiente.....	55
3.4.	Consideraciones adicionales	56
4.	INGENIERÍA INVERSA Y RESULTADOS	57
4.1.	Arquitectura del sistema	57
4.1.1.	Componentes del sistema.....	58
4.1.2.	Arquitectura del despliegue.....	62
4.1.3.	Estructura de directorios	62
4.2.	Diagramas UML.....	66
4.2.1.	Diagrama de casos de uso.....	67
4.2.2.	Diagrama de clases – GraphController PYX4	68
4.2.3.	Diagrama de estados de gráfico.....	71

4.3. Diagrama de entidad relación	72
4.3.1. Relaciones de modelos en Rails	77
5. DISCUSIÓN DE RESULTADOS	80
5.1. Implicaciones	80
5.2. Limitaciones	81
CONCLUSIONES	82
RECOMENDACIONES	84
BIBLIOGRAFÍA	86

ABREVIATURAS Y SIMBOLOGÍA

BPM	Business Process Management
BPMN	Business Process Model Notation
BPMS	Business Process Management System
CASE	Computer-Aided Systems Engineering
ERB	Embedded Ruby
LOA	Levels of Abstraction
MVC	Model-View-Controller
OMG	Object Management Group
ORM	Object Relational Mapping
REST	Representational State Transfer
RoR	Ruby on Rails
SDLC	System Development Life Cycle
UML	Unified Modeling Language

ÍNDICE DE FIGURAS

Figura 2.1 Ingeniería directa, inversa y reingeniería	8
Figura 2.2 Relaciones entre abstracciones, artefactos de software, y dominios del mundo individual	12
Figura 2.3 Tipos de herramientas CASE según su etapa en el SDLC.....	20
Figura 2.4 Diagramas UML 2.0	29
Figura 2.5 Arquitectura MVC.....	44
Figura 2.6 Procesamiento de una petición HTTP desde la perspectiva MVC de RoR.....	47
Figura 3.1 Fases del proyecto.....	51
Figura 4.1 Diagrama de componentes RoR 3.2.12.....	61
Figura 4.2 Diagrama de despliegue de PYX4	62
Figura 4.3 Directorio de código fuente de PYX4 - RoR.....	63
Figura 4.4 Diagrama de casos de uso	67
Figura 4.5 Diagrama de clases GraphController.rb.....	70
Figura 4.6 Diagrama de estados de Gráfico en PYX4	71
Figura 4.7 Modelo ER generado por RE - Parte A.....	73
Figura 4.8 Modelo ER generado por RE - Parte B.....	74
Figura 4.9 Modelo ER generado por RE - Parte C.....	75
Figura 4.10 Contenido original de 20121002124403_create_graphs.rb	76

Figura 4.11 Contenido original de	
20121105134754_add_columns_to_graphs.rb.....	77
Figura 4.12 Diagrama de dependencia de modelos de PYX4	78

ÍNDICE DE TABLAS

Tabla 1.- Relaciones en UML	28
Tabla 2.- Diagramas UML usados con el "modelo 4+1" de vistas	34

INTRODUCCIÓN

La fase de mayor duración dentro del ciclo de vida del software es la de mantenimiento, por tanto, es la que suele demandar mayores recursos. La actividad de mantenimiento de sistemas debe estar ligada a un proceso estructurado y normado para controlar los cambios y nuevos requerimientos a implementar; siendo una de las mayores complicaciones el poder enfrentar la rotación del personal de desarrollo mientras se mantiene actualizado un repositorio de conocimiento del sistema, lo cual permitirá bajar los tiempos de respuesta y el número de desarrolladores necesarios que finalmente se traduce en una reducción de costos.

Los sistemas heredados por su naturaleza crítica están acompañados de altos costos de mantenimiento que pueden ser mermados si se dispone de la documentación suficiente para disminuir la curva de aprendizaje sobre la implementación del sistema, lo cual es necesario para introducir cambios en este.

El avance acelerado de las tecnologías de información, las mejoras en el hardware, el incremento en las velocidades de conexión a internet así como

la penetración de los sistemas en todos los niveles organizacionales humanos provoca un ambiente sumamente dinámico en donde innovar y mantenerse dentro de un proceso de mejora continua es vital para cualquier empresa que ofrece servicios tecnológicos. Enfocarse en resolver requerimientos y abstraer la rigurosidad técnica ha propiciado la implementación de frameworks de desarrollo, un conjunto de librerías y componentes enfocados al reuso de código, a niveles técnicos se puede hablar de la agilidad con que se mejoran estos sistemas gracias a la implementación de metodologías de desarrollo ágil como SCRUM por ejemplo, lo cual exige mantenerlos actualizados entre una versión y otra. Una de las desventajas de esto es que muchas veces los métodos y las funciones usadas en una versión son despreciados en otra, haciéndose necesario cambiar la implementación original, lo cual implica todo un proceso de reingeniería.

Además del software como tal, las metodologías también evolucionan creándose estándares para mejorar la interoperabilidad y la integración, lo cual es la motivación del presente trabajo. Aquí se tomará el sistema heredado PYX4 y se realizará el proceso de ingeniería inversa para elaborar su documentación, la cual estará disponible para que en trabajos futuros sirva como punto de partida para acoplar nuevos módulos a este.

CAPÍTULO 1

PROBLEMA DEL PROYECTO

1.1. Antecedentes

En la industria del software existe la necesidad constante de actualizar y dar mantenimiento a los sistemas, ya sea por la obsolescencia de componentes que han sido mejorados, detección de errores a ser corregidos o por la necesidad de atender nuevos requerimientos propios del ambiente dinámico de los negocios que soportan. Para llevar a cabo esta tarea, se requiere disponer de la documentación técnica adecuada como por ejemplo diagramas de clases, diagramas de secuencia u otros dependiendo de las necesidades a ser atendidas.

Durante el ciclo de vida del software existen factores determinantes, como por ejemplo los recursos disponibles para su implementación ya sean estos: tiempo, talento humano, infraestructura u otros. Estos recursos inciden en la calidad de los productos resultantes y debido a esto cuando se encuentran limitados, la mayoría de las veces los equipos de trabajo deben enfocarse en una tarea principal y/o en aquellas relacionadas a los requerimientos funcionales para cumplir con las fechas de entrega, lo que perjudica la mayoría de las veces en la tarea de documentación de la arquitectura del sistema.

En los entornos empresariales, BPM es una metodología ampliamente usada para soportar sus procesos, la cual se caracteriza por permitir modelarlos, organizarlos y documentarlos.

PYX4 es un Business Process Management System que implementa una notación propietaria llamada Qualigram que está orientada a la gestión del proceso [1], se encuentra implementado usando el framework Ruby On Rails escrito en lenguaje Ruby y cuenta con varios años de presencia en el mercado internacional.

1.2. Descripción del problema

El sistema PYX4 permite diseñar gráficamente los procesos de negocio por medio de una interfaz sencilla con el fin de que todos los miembros de la empresa puedan describir las actividades que desempeñan usando un lenguaje común, el lenguaje o notación es Qualigram que es el resultado de un proyecto internacional de investigación [2]. Sin embargo, Qualigram no es una notación estándar y en la actualidad PYX4 sólo tiene soporte para dicha notación, esto significa que el sistema no tiene la característica de permitir la importación de procesos modelados en BPMN, que es la notación estándar, y tampoco la exportación de los procesos modelados desde Qualigram hacia BPMN lo cual dificulta que el sistema pueda ser adoptado por organizaciones que ya tienen sus procesos definidos con BPMN.

Dado lo antes expuesto, se presenta la necesidad de agregar un módulo para el soporte de BPMN al sistema PYX4 por motivos de compatibilidad e integración. Sin embargo, los proveedores del sistema no disponen de la documentación técnica necesaria para realizar el proceso de ingeniería, por lo tanto la problemática que se atacará en este trabajo es la ingeniería inversa a partir del código fuente ya implementado para recuperar el diseño del sistema.

1.3. Justificación

En rasgos generales un proceso de ingeniería está compuesto de diferentes etapas, estas son levantamiento de requerimientos, análisis de requerimientos, diseño de solución e implementación y mantenimiento. Cuando el flujo de eventos sigue el orden descrito se trata de un proceso de ingeniería directa, si se requiere hacer algún cambio en una de las etapas, es decir, una reingeniería, será necesario regresar a la etapa anterior para hacer los cambios deseados, por ejemplo, si lo que se necesita es cambiar la implementación se debe de regresar al diseño o a etapas anteriores de ser requerido y trabajar en base a los nuevos requerimientos a atender.

En este proyecto se desea generar la documentación del diseño del software puesto que está no existe, por lo tanto es necesario realizar un proceso de ingeniería para que a futuro se pueda realizar cualquier tipo de reingeniería.

1.4. Propuesta y alcance

Cuando se trata de documentación y diseño de software se dispone del estándar UML. Este proyecto busca generar los diagramas UML

estructurales y de comportamientos del sistema PYX4. Además, se generará el modelo de datos usado por el sistema para llevar a cabo sus operaciones.

1.5. Objetivos

1.5.1. General

Elaborar la documentación técnica necesaria para su utilización en futuros proceso de reingeniería sobre PYX4.

1.5.2. Específicos

- Desplegar el sistema PYX4 proveyendo un instructivo para su realización.
- Realizar el proceso de ingeniería inversa a partir del código fuente del BPMS PYX4 documentando su arquitectura.
- Generar los diagramas de clases, componentes, despliegue y entidad-relación a partir de la ingeniería inversa para ser incorporados en la documentación del sistema.
- Validar que la documentación generada para PYX4 sea la idónea para futuros desarrollos o integración con nuevos componentes.

1.6. Metodología

La primera etapa de este trabajo consiste en la revisión de literatura relacionada a ingeniería inversa en ella se revisan las metodologías existentes así como las fortalezas y debilidades de cada una con el fin de determinar cuál o cuáles de ellas se adaptan a las necesidades del proyecto.

Una vez definida la metodología y requerimientos a atender, se procederá a realizar un estudio del framework Ruby on Rails sobre el cual se encuentra implementado el sistema.

Finalmente, se muestra el resultado de la ingeniería inversa y se generan los diagramas y documentación necesaria.

CAPÍTULO 2

REVISIÓN BIBLIOGRÁFICA

2.1. Ingeniería inversa

Para poder entender de qué se trata la ingeniería inversa se debe empezar clarificando el proceso de reingeniería, aquí se hará hincapié en la reingeniería de software.

La reingeniería es el proceso en el cual se realiza un examen, análisis y modificación de un sistema con el fin de reconstruirlo de una nueva forma dando como resultado una nueva implementación [3].

En la Figura 2.1 se ilustra el proceso de reingeniería de software que guarda concordancia con la definición del párrafo anterior. Aquí se muestra una fecha que va desde una implementación de código hacia otra, es decir que a partir de un sistema existente se obtiene una nueva implementación, el proceso consta de dos subprocesos en primer lugar se realiza un proceso de ingeniería inversa que tiene como objetivo lograr entender la manera en que funciona el software estudiado y en base a esto poder modificar su funcionalidad, rendimiento o implementación; en segundo lugar se tiene el proceso tradicional de ingeniería directa que comienza con los nuevos requerimientos desde los cuales se plantea un nuevo diseño y se termina con la nueva implementación.

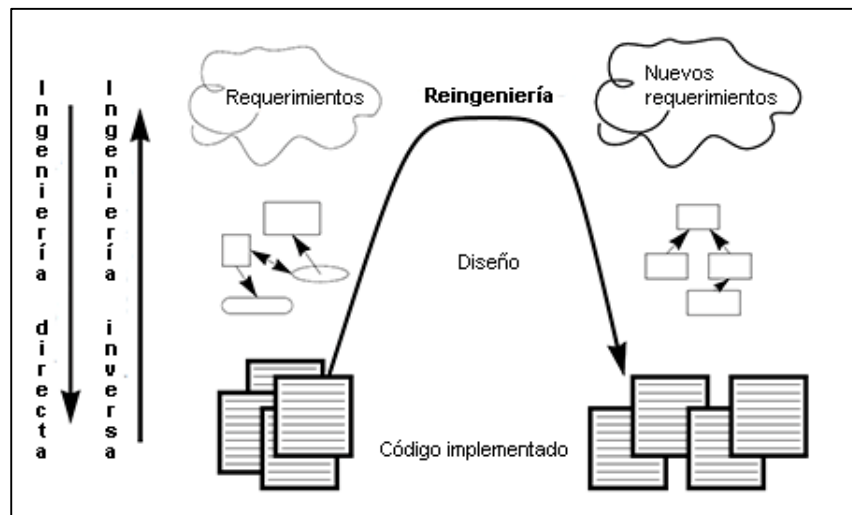


Figura 2.1 Ingeniería directa, inversa y reingeniería

Ahora bien, ¿por qué hacer reingeniería? En informática es recurrente el uso de sistemas legados, un legado es algo valioso que ha sido heredado, por tanto estos sistemas son aquellos que han sido heredados de una administración de tecnología a otra y que por su valor, que usualmente es elevado debido a que son fundamentales en los negocios que soportan, no pueden ser reemplazados fácilmente.

El surgimiento de metodologías de desarrollo ágil sumado a la rápida rotación de personal ha hecho que los sistemas de software puedan convertirse en legados más rápidamente de lo esperado y que por lo tanto, caigan en la obsolescencia, afortunadamente existen mecanismos para poder actualizarlos cuando dejan de cumplir con su propósito o necesitan cumplir con nuevos requerimientos. Sin embargo, cuando se trata de un sistema legado la tarea de reemplazarlo o actualizarlo está relacionada a costos muy elevados. El objetivo de la reingeniería es reducir la complejidad de un sistema legado lo suficiente como para poder seguir utilizándolo y adaptarlo a los nuevos requerimientos a un costo aceptable.

Las razones específicas de porqué se debería realizar reingeniería pueden ser tan variadas como las siguientes:

- Se desea desagregar un sistema monolítico debido a que sus partes desagregadas podrían ser más fácilmente comercializadas o combinadas de diferentes formas.
- Se desea aumentar el rendimiento del sistema, es decir, lograr que se ejecute con mayor velocidad o que consuma menos recursos.
- Se desea aumentar la portabilidad del sistema modificando su arquitectura y eliminando la dependencia entre el código implementado y la plataforma en que se lo despliega.
- Se desea obtener el diseño del sistema como una primera fase para lograr una nueva implementación.
- Se desea aprovechar una nueva tecnología, como por ejemplo nuevos estándares o librerías, con el fin de lograr bajar los costes de mantenimiento.
- Se desea reducir la dependencia humana para dar mantenimiento al sistema mediante la mejora de la documentación y hacerlo por consiguiente más fácil de mantener.

Como se ha mencionado, finalmente la reingeniería no se trata de una simple tarea de transformación de código fuente sino que va más allá, hasta la transformación del sistema en todos sus niveles. Es por esto que

cuando se requiere realizar reingeniería a un sistema del cual no se dispone la documentación adecuada surge la necesidad de llevar a cabo una ingeniería inversa como un requisito previo, puesto que no se puede transformar lo que no se entiende.

La ingeniería inversa tiene sus orígenes en el hardware; ocurrió cuando los fabricantes tecnológicos conseguían un producto de la competencia y lo desensamblaban para poder comprender su diseño y fabricación con el fin de duplicarlos o mejorar los propios. Por su parte, la ingeniería inversa de software usualmente se enfoca más que en un producto de la competencia, en un producto propio del que no existen especificaciones o estas están incompletas.

Según Chikofsky y Cross [5] la ingeniería inversa es el proceso de analizar un sistema con el propósito de identificar sus componentes y las interrelaciones entre los mismos y de crear representaciones del sistema en una nueva forma o en un nivel más alto de abstracción.

El nivel de abstracción en general hace referencia a la sofisticación de la información de diseño que es posible extraer del código fuente, y a medida que este nivel aumenta, aumenta la facilidad con que la

información proporcionada podrá ser comprendida por quien la recibe. Cabe destacar que dependiendo de la fase del desarrollo de software en que se encuentre se tendrán diferentes formas de abstracción como se muestra en la Figura 2.2, y para cada forma se tienen distintos niveles, aquí se hace referencia a los niveles de abstracción del diseño de la solución.

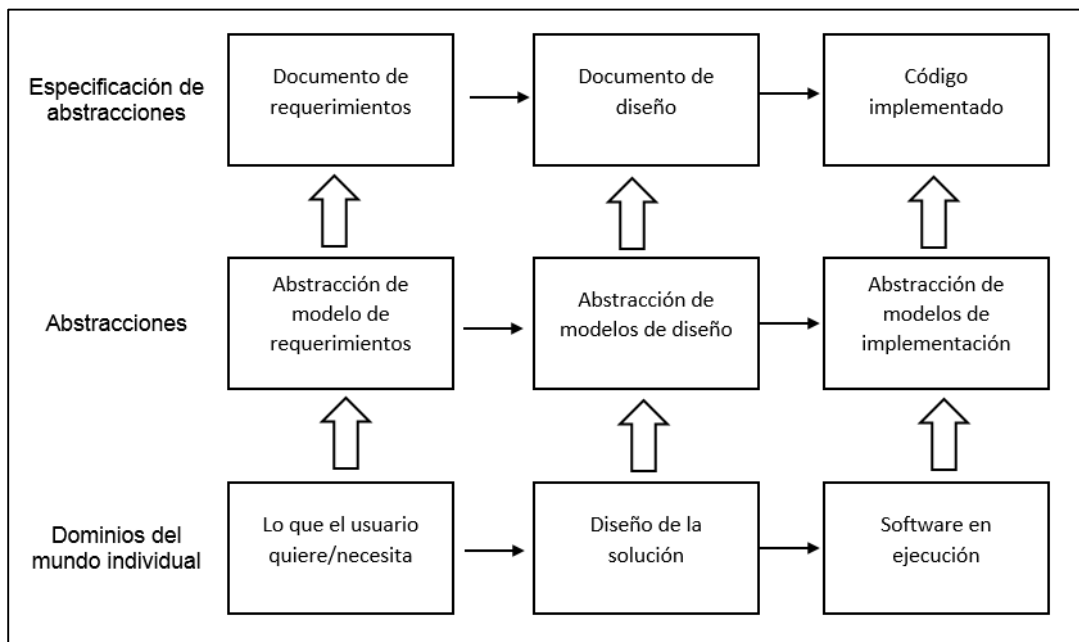


Figura 2.2 Relaciones entre abstracciones, artefactos de software, y dominios del mundo individual

Dependiendo del tipo de análisis que se realice, la ingeniería inversa puede ser clasificada como ingeniería inversa estática cuando se centra en describir la estructura del código fuente o como ingeniería inversa

dinámica cuando el análisis se lo realiza considerando el comportamiento en ejecución del software.

Considerando lo antes expuesto, se puede decir que la ingeniería inversa permite analizar y representar un software en una forma abstracta con el fin de facilitar su mantenimiento, reingeniería, reusabilidad y documentación.

2.1.1. Dificultades

Enfrentar un proyecto de ingeniería inversa conlleva las dificultades de cualquier proyecto de ingeniería de software además de las propias de esta metodología en particular. A continuación, se presentan las más relevantes.

Dificultades relacionadas a la dirección del proyecto:

- Mantener un propósito común en el equipo de ingeniería inversa.
- Mantener la visión arquitectónica durante el curso del proyecto.
- Mantener al equipo sincronizado.
- Elegir correctamente el problema en el cual enfocarse primero.
- Manejar los problemas reportados.

Dificultades relacionadas a las primeras impresiones:

- Tener una referencia de contexto histórico y político en el cual se desarrolló el sistema.
- Lograr una correcta impresión acerca de la calidad del código fuente.
- Identificar las partes de la documentación, si es que existe, que podrían ayudar a dar solución al problema a atacar.
- Lograr una correcta identificación de los escenarios típicos y de las principales características del sistema a través de la presentación de un demo por parte de los usuarios.
- Determinar la factibilidad de poder recompilar o ejecutar el código fuente.

2.1.2. Diferentes enfoques

Cuando se enfrenta un proceso de ingeniería inversa es posible centrarse en diferentes aspectos del software dependiendo de las necesidades que se requieran atender. El núcleo de este proceso es una actividad denominada extracción de abstracciones y puede estar enfocada en el procesamiento, los datos o las interfaces.

2.1.2.1. Ingeniería inversa de lógica o de proceso

El núcleo de todo software radica en las tareas que éste automatiza, las cuales son levantadas en una especificación de requerimientos y que al pasar por el proceso de ingeniería de software son implementadas y escritas en un lenguaje que pueda ser entendido por un equipo de cómputo; esto es lo que se conoce como código fuente del programa. El código fuente es la abstracción más detallada de un software legible para un humano y contiene todas las instrucciones necesarias para cumplir con los requerimientos.

La ingeniería inversa de lógica o de proceso es aplicada sobre el código fuente o en los documentos de diseño si es que existen. Para este proceso es posible encontrar diferentes herramientas que automatizan el análisis semántico del código, estas herramientas serán mencionadas en la sección 2.1.3 de este documento.

2.1.2.2. Ingeniería inversa de datos

En este enfoque lo que se busca es comprender cómo los datos son usados por el sistema en todos sus niveles. Los datos son

encapsulados en estructuras de datos y se pueden identificar dos tipos:

- Estructuras de datos internas

Corresponde a los datos definidos en el código fuente por medio de clases de objetos, de estas clases depende el funcionamiento lógico del software.

- Estructuras de bases de datos

Cuando se requiere analizar la información transaccional de un sistema que se encuentra almacenada en un esquema de bases de datos con el fin de poder modificarlo o usarlo como punto de partida para crear uno nuevo, es necesario comprender las relaciones entre las tablas o vistas ya existentes.

2.1.2.3. Ingeniería inversa de interfaces de usuario

En ciertas ocasiones se va a requerir realizar una reingeniería de la GUI con el fin de crear una nueva UI que sea más amigable para el usuario. Sin embargo, para poder llevar a cabo esta tarea es necesario entender cómo funciona la UI actual, cuáles fueron las

métricas que se consideraron y las que hay que mejorar, para hacerlo se requiere de la ingeniería inversa.

2.1.3. Herramientas disponibles

Dado que la ingeniería inversa enfocada a la lógica o al proceso consta de un análisis del código fuente y que este a su vez posee una estructura semántica conocida es posible automatizar su análisis para así poder obtener distintas abstracciones de diferentes niveles.

2.1.3.1. Depuradores

Los depuradores son programas muy usados en ambientes de desarrollo y su importancia radica en que tienen la capacidad de controlar otros programas permitiendo al desarrollador seguir la ejecución desde la perspectiva del código fuente línea a línea, permiten además insertar puntos de control y ver los valores que adquieren las variables en memoria en un determinado momento en la ejecución del programa. Si se suman todas estas características se tiene una herramienta efectiva para la detección fallos.

2.1.3.2. Desensambladores

Los lenguajes de alto nivel como los empleados en la programación de software al ser compilados o interpretados se convierten en un lenguaje que los equipos de cómputo son capaces de entender. El

nivel más bajo legible para los humanos es el lenguaje ensamblador cuyas instrucciones están fuertemente asociadas a la arquitectura del hardware sobre el cuál se ejecutan. Sin embargo, el lenguaje ensamblador para poder ser al fin ejecutado es convertido a lenguaje de máquina (binario). Se conoce como desensamblador a un programa que permite convertir lenguaje de máquina a lenguaje ensamblador.

2.1.3.3. Descompiladores

Así como se definieron los desensambladores, suponga que a partir del lenguaje de máquina o lenguaje ensamblador se requiera pasar a un lenguaje de alto nivel para facilitar la lectura humana del código, para realizar esta transformación se usan los descompiladores.

2.1.3.4. Herramientas de inyección de fallos

Las herramientas de inyección de fallos son aquellas que permiten insertar entradas malformadas a un sistema para provocar errores de los cuales al realizar un análisis es posible encontrar vulnerabilidades de seguridad o falta de validación en la entrada a los datos lo cual finalmente contribuye en la comprensión del software.

2.1.3.5. Herramientas CASE

Las herramientas CASE por sus siglas en inglés Computer Aided Systems Engineering, son sistemas que soportan o automatizan las actividades y metodologías propias de la ingeniería de software en cualquier etapa de su ciclo de vida y tienen como objetivo acelerar el desarrollo de un proyecto.

Según la etapa en las cuales son usadas, las herramientas CASE pueden dividirse como se muestra en la Figura 2.3. Aquellas usadas en las primeras etapas del SDLC como planeación, análisis y diseño son conocidas como herramientas CASE superiores, mientras que las que son usadas en la implementación, pruebas o mantenimiento son conocidas como herramientas CASE inferiores y por último las que pueden ser usadas en cualquier etapa son conocidas como herramientas CASE integradas.

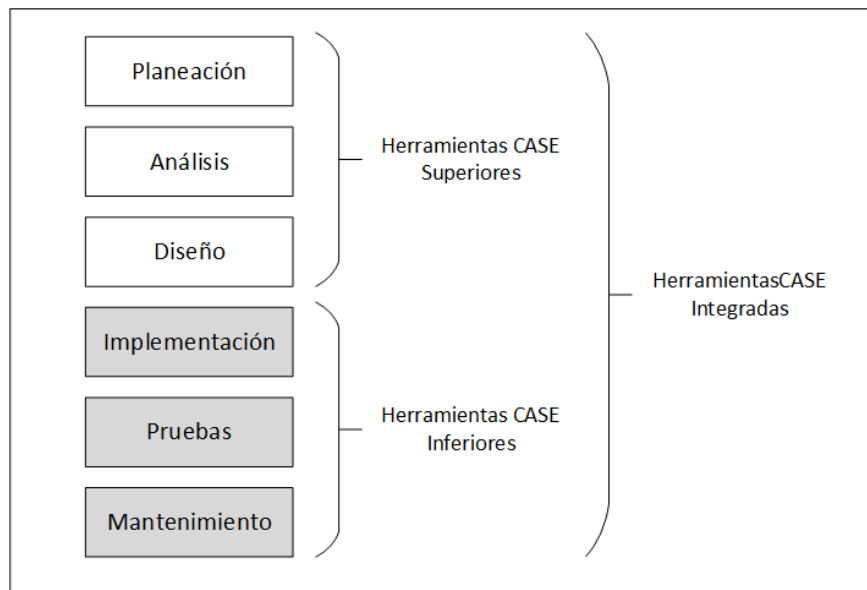


Figura 2.3 Tipos de herramientas CASE según su etapa en el SDLC

Existen varios tipos de estas herramientas, a continuación se listarán algunos:

- Herramientas de administración de proyectos.
- Herramientas de modelado de procesos.
- Herramientas de análisis.
- Herramientas de diagramación.
- Herramientas de documentación.
- Herramientas de diseño.
- Herramientas de prototipado.
- Herramientas de gestión de la configuración.

- Herramientas de control de cambios.
- Herramientas de programación.
- Herramientas de desarrollo web.
- Herramientas de aseguramiento de calidad.
- Herramientas de mantenimiento.

2.2. Lenguaje de modelado unificado (UML)

El lenguaje de modelado unificado o también conocido como notación UML, es un lenguaje gráfico que permite documentar, especificar y visualizar esquemas de sistemas de software desarrollados bajo el paradigma de la programación orientada a objetos. Actualmente es el estándar para la documentación y descripción gráfica del software y es regulado por el grupo de administración de objetos (OMG).

Cabe destacar que UML no es un método de desarrollo ni indica cómo realizar el diseño del software, sino que es una representación visual del diseño que se ha ideado para el software, es decir, UML es una representación en un nivel de abstracción alto de las características estructurales y comportamentales del sistema y lo hace por medio de diagramas que serán descritos en secciones posteriores de este documento.

2.2.1. Paradigma orientado a objetos

Un paradigma es un modelo o patrón estructurado que se sostiene en un contexto determinado, cuando se habla del paradigma orientado a objetos hace referencia a un paradigma de programación de sistemas

de software que se basa en la modularidad como fundamento para construir sistemas complejos. Se desarrolló debido a que sus predecesores, los paradigmas de desarrollo funcional y estructurado hacían difícil la tarea de construir sistemas que se adapten a lógicas de negocios muy complejas.

La modularidad de los POO hace referencia a la capacidad de dividir un problema en módulos o partes más pequeñas. Entre las ventajas de esta metodología se encuentra el manejo sencillo de código ya que permite que diferentes personas puedan trabajar simultáneamente en módulos distintos, la reutilización de módulos de terceros en proyectos diferentes y además las pruebas que se pueden realizar de cada uno de estos elementos separadamente.

2.2.1.1. Bases del modelo de OO

Para entender el modelo de OO se deben tener en cuenta los puntos expuestos a continuación.

- Clases: Una clase es una entidad genérica de un objeto del mundo real o una abstracción del mismo que posee propiedades (atributos) y comportamiento (métodos).

Considérese por ejemplo Perro como una clase que posee atributos como patas, cola y colmillos; y comportamientos tales como ladrar, aullar y comer.

- **Objetos:** Un objeto es una instancia específica de una clase; de esta forma, considerando el ejemplo anterior se puede decir que una instancia de la clase perro podría ser Snoopy, Scooby Doo o Lassie.
- **Herencia:** Como sucede en el mundo real, en la programación orientada a objetos es posible definir jerarquías de clases, es decir, que una clase puede tener una o más subclases que heredan todos sus atributos y métodos y que además puede tener los propios, de esta forma si seguimos con el ejemplo de la clase perro podríamos poner como subclases perro bombero o perro antinarcóticos puesto que estos poseen otros comportamientos (métodos) distintos a los de cualquier otro por el entrenamiento que reciben.
- **Envío de mensajes:** Un objeto como tal no puede estar aislado; si lo estuviera, su existencia no tendría sentido. La forma en que los objetos se comunican entre sí es por medio de mensajes. Los lenguajes de programación orientados a

objetos permiten enviar mensajes entre objetos de diferentes maneras siendo una de ellas la invocación de los métodos.

2.2.1.2. Características relacionadas al POO

Además de las bases expuestas en la sección anterior existen otros conceptos relacionados a la programación orientada a objetos que sustentan la forma en que se definen los objetos.

- **Abstracción:** La abstracción es la capacidad de extraer las características y comportamientos irrelevantes de un objeto y mantener los relevantes.
- **Encapsulación:** Es la capacidad de ocultar las características y comportamientos, es decir, los atributos y métodos de una clase dentro de sí misma permitiendo que la clase sea vista como una caja negra reduciendo la complejidad, puesto que importa más el qué hace que no cómo lo hace.
- **Polimorfismo:** El polimorfismo es la capacidad que tienen las clases heredadas de que sus métodos tengan diferentes comportamientos al de la clase padre.

2.2.2. Principios de modelado usando UML

En 1998 Booch, Rumbaugh y Jacobson [15] publicaron principios que hasta la fecha no han sufrido mayores cambios. Estos principios reúnen

la experiencia de modelado en diferentes disciplinas de la ingeniería y son aplicables también al modelado usando UML.

- Principio I: La selección del modelo influye de manera profunda en cómo el problema es atacado y cómo la solución es formulada. Esto hace referencia a la necesidad de seleccionar el modelo indicado para cada caso.
- Principio II: Cada modelo puede ser representado en diferentes niveles de precisión. Con esto se establece que a pesar de que para determinado problema es posible realizar más de un modelo de la solución, cada uno puede tener un mayor detalle dependiendo de las necesidades del caso.
- Principio III: El mejor modelo está conectado con la realidad. En software, este enunciado hace referencia a la relación que debería haber entre el modelo del diseño y el código implementado. Si se considera el paradigma orientado a objetos y sus conceptos, es posible lograr la correspondencia entre estos dos aspectos.
- Principio IV: Ningún modelo por sí solo es suficiente. Los sistemas no triviales son mejor representados si se considera un grupo de modelos independientes en sí mismos pero relacionados entre sí. Por ejemplo, el mapa de un país es posible

representarlo con un modelo político donde se puedan ver sus provincias y ciudades, este sería un modelo por sí solo, pero el país se podría entender mejor si se considera además un mapa geográfico donde se ven sus montañas y valles.

2.2.3. Notación UML

Las especificaciones técnicas de la notación UML son extensas y complejas, se encuentran publicadas en el sitio web del OMG [18]. De manera general para realizar las representaciones en UML se deben considerar tres aspectos conocidos como bloques de construcción: los elementos, las relaciones y los diagramas.

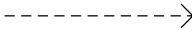

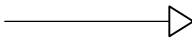
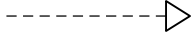
2.2.3.1. Elementos de UML

Son abstracciones en el modelo y son las unidades básicas para la construcción de los modelos. De acuerdo a su naturaleza, se clasifican en estructurales, comportamentales, de agrupación y de notación.

2.2.3.2. Relaciones en UML

Son entidades que relacionan a los elementos entre sí. Las relaciones pueden ser de cuatro tipos: de dependencia, de asociación, de generalización o de realización.

Tabla 1.- Relaciones en UML

Dependencia	Asociación y agregación
	
<p>Se establece entre dos elementos, uno dependiente y otro independiente y grafica que un cambio en el independiente puede afectar al dependiente.</p>	<p>La asociación representa una relación estructural entre dos clases. Existen dos tipos especiales de asociación, la agregación y la composición.</p>
Generalización	Realización
	
<p>Es una relación de especialización/generalización en cuanto los objetos de una clase especializada (hijo) pueden sustituir a los objetos de otra general (padre).</p>	<p>Es una relación entre clasificadores en donde se forma un contrato que consiste en que uno de ellos le garantiza al otro su cumplimiento.</p>

2.2.3.3. Diagramas UML

Son representaciones gráficas de un conjunto de elementos y/o relaciones que generalmente son visualizadas como un grafo conexo de nodos (elementos) y arcos (relaciones) que en conjunto permiten la visualización de una parte de un sistema. En UML 2.0 se definen trece tipos de diagramas diferentes [17], estos diagramas se encuentran agrupados en la Figura 2.4 de acuerdo a su naturaleza; en la parte izquierda se pueden apreciar los diagramas estructurales y en la derecha los de tipo comportamentales.

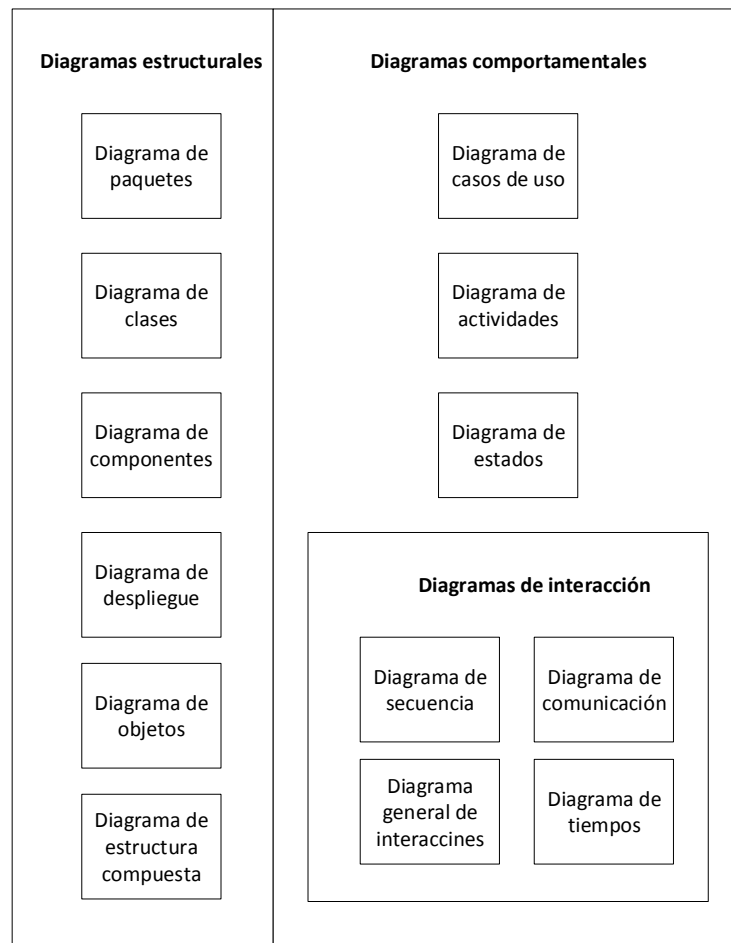


Figura 2.4 Diagramas UML 2.0

En la versión más reciente de UML, la versión 2.4, se ha agregado un diagrama adicional a los presentados en la Figura 2.4; el nuevo diagrama es conocido como diagrama de perfil. A continuación, se describirán brevemente los objetivos de cada uno de ellos.

- Diagrama de clases: es un diagrama que describe la estructura estática del sistema a nivel de las metaclasses. Una

metaclase es una abstracción que describe un conjunto de instancias que tienen características comunes [31].

- Diagrama de objetos: es una instancia del diagrama de clases, es decir, una captura de una clase en un momento específico que incluye los valores de las variables y su estado.
- Diagrama de paquetes: muestra los paquetes y las dependencias entre ellos. Un paquete es también conocido como namespace y agrupa elementos relacionados semánticamente con el fin de organizarlos de tal manera que facilite la lectura del modelo estructural del sistema.
- Diagrama de estructura compuesta: permite representar una estructura de elementos centrándose en los detalles y en sus relaciones internas.
- Diagrama de componentes: son representaciones de los componentes del sistema, es decir, de sus partes modulares, de las interfaces y puertos así como también de la relación entre ellos.
- Diagrama de despliegue: muestra la arquitectura del sistema en despliegue, la distribución de los artefactos del software respecto a los componentes físicos.

- Diagrama de perfil: es un diagrama auxiliar que permite definir estereotipos, etiquetas y restricciones personalizadas. Fue introducido al estándar con el fin de proveer un mecanismo ligero para la extensión de UML.
- Diagramas de casos de uso: describe las acciones que puede ejecutar el sistema al recibir un estímulo exterior (actor) para proveer un resultado observable.
- Diagrama de actividad: muestra la secuencia de eventos y las condiciones para que ellos ocurran entre los elementos del sistema.
- Diagrama de estados: también llamado diagrama de máquina de estados, es usado para expresar de manera discreta los estados de transición de los elementos del sistema.
- Diagrama de secuencia: se centra en expresar el estado de intercambio de mensajes entre los objetos.
- Diagrama de comunicación: provee características similares a la de los diagramas de secuencia pero se centra en la relación entre los objetos.

- Diagrama de interacción: es una variación del diagrama de actividad en el cual los nodos representan diagramas de interacción y se centra en el control de flujo.
- Diagrama de tiempos: muestra las interacciones basadas en el tiempo; como propósito principal mezclan los diagramas de actividad y secuencia.

El concepto de vista y diagramas UML

El concepto de vista hace referencia a mirar la arquitectura de un sistema desde una perspectiva en específico en la que se proyectan los aspectos más relevantes de la misma, por lo tanto, el concepto se puede entender como vista arquitectónica.

En 1995 Philippe Kruchten presentó el modelo “4+1” (que encaja en el estándar IEEE 1471-2000) de vistas para arquitectura de software [19] el cuál se basa en cuatro vistas bien diferenciadas y una vista adicional (“la vista +1”) que las relaciona. El modelo se utiliza para describir la arquitectura de un sistema de software intensivo. Las vistas que lo componen son: vista de diseño, vista de interacción, vista de implementación y vista de despliegue; en donde la vista que

las relaciona es conocida como vista de casos de uso. El modelo de Kruchten permite tener una visión global del sistema.

Es posible relacionar los diagramas UML con cada una de las vistas del modelo "4+1". En la Tabla 2, se detalla cada una de las vistas del modelo y se las relaciona con los respectivos diagramas UML.

Tabla 2.- Diagramas UML usados con el "modelo 4+1" de vistas

Vista	Descripción	Aspecto	Diagrama
Vista de casos de uso	Describe la funcionalidad del sistema tal cual es percibido por todos cada uno de los tipos de usuario (actores).	Estático	Casos de uso
		Dinámico	Interacción Estados Actividades
Vista de diseño	Describe las abstracciones propias de la programación orientada a objetos tales como clases, interfaces y colaboraciones cuya función es dar soporte para la implementación de los requerimientos funcionales.	Estático	Clases Objetos Estructura compuesta
		Dinámico	Interacción Estados Actividades
Vista de interacción	Se centra en flujo de control entre las partes del sistema y en los aspectos relacionados a los requerimientos no funcionales tales como rendimiento o capacidad de procesamiento. En su aspecto estático describe las clases activas y los mensajes entre ellas.	Estático	Clases Objetos
		Dinámicos	Interacción Estados Actividades
Vista de implementación	Describe los artefactos que se utilizan para ensamblar y poner en ejecución el sistema. Se centra en describir la arquitectura física del sistema.	Estáticos	Componentes (artefactos) Estructura compuesta
		Dinámicos	Interacción Estados Actividades
Vista de despliegue	Describe los elementos necesarios para la instalación y ejecución del sistema. Se centra en la distribución de las partes que conforman el sistema.	Estáticos	Despliegue
		Dinámicos	Interacción Estados Actividades

2.3. Ruby y Ruby on Rails

La comunicación entre el hombre y la máquina, en este caso entre el programador y la computadora es posible gracias a los lenguajes de programación. Estos lenguajes consisten en símbolos, caracteres y reglas bien estructuradas que han sido diseñadas usando abstracciones con el objetivo de que las personas puedan comprender fácilmente las instrucciones que las máquinas pueden ejecutar.

Los lenguajes de programación son representaciones con un alto nivel de abstracción respecto al lenguaje binario que las máquinas ejecutan, por lo tanto, necesitan de un proceso que lo traduzca hasta hacerlo ejecutable. Dependiendo de la naturaleza de este proceso son clasificados en lenguajes compilados o interpretados; los primeros son traducidos y “empaquetados” directamente en archivos en lenguaje de máquina antes de ser ejecutados, mientras que los segundos son traducidos cada vez que se van a ejecutar y son almacenados temporalmente en memoria.

La tarea de programar o desarrollar sistemas de software se ha ido diversificando y perfeccionando a lo largo de los años, han surgido diversos paradigmas que se han ajustado a los requerimientos y avances

tecnológicos y científicos, se han identificado patrones recurrentes al momento del desarrollo los cuales han dado paso a los patrones de diseño, también se han creado herramientas que agilitan su implementación como lo son las herramientas CASE y se han definido estructuras conceptuales y tecnológicas que facilitan y organizan el desarrollo, estas estructuras son conocidas como frameworks.

En esta sección se expondrá acerca de Ruby y Ruby on Rails, lenguaje de programación y framework respectivamente puesto que son con los que se ha implementado el sistema PYX4.

2.3.1. Ruby

2.3.1.1. Historia

Ruby fue concebido por el japonés Yukihiro Matsumoto en 1993 y su primera versión beta fue publicada en 1994 [21] como un proyecto de código abierto. El nombre de Ruby fue escogido para que haga alusión a Perl uno de los lenguajes en que se inspira.

Matsumoto emprendió la tarea de la creación de Ruby al enfrentarse con que en ese entonces no existía un lenguaje que satisfaga sus requerimientos a la perfección, estos requerimientos eran:

- Sintaxis simple.
- Orientación a objetos pura.
- Que disponga de iteradores y cierres.
- Manejo de excepciones.
- Recolector de basura.
- Portable.

Con estos requerimientos y tras la formación en 1996 de la comunidad de Ruby que comenzó a colaborar con su desarrollo, Ruby logró su primera versión estable (1.2) en Diciembre de 1998 que hasta ese momento sólo estaba disponible en Japón. Posteriormente Matsumoto y Keiju Ishitsuka escribieron el primer libro sobre Ruby titulado “The Object-oriented scripting Language Ruby” lo que propició que el lenguaje empiece a popularizarse en los países de habla inglesa.

Una de las versiones más significativas fue la 1.8 lanzada en el año 2003 que se mantuvo vigente por alrededor de 6 años, esta versión resultó incompatible con Ruby 1.9. La siguiente versión estable fue liberada en el 2011 y fue la 1.9.3 la cual integró cambios significativos que mejoraban el rendimiento y además permitía tareas de pruebas unitarias entre otros aspectos.

Uno de los hitos importantes fue la liberación de RubyGems en 2004 como sistema gestor de paquetes y librerías para Ruby lo cual aportó con la propagación del uso del lenguaje entre los desarrolladores.

La versión más reciente, Ruby 2.2, fue liberada en Diciembre del 2014 y agrega nueva funcionalidad y mejoras enfocadas a las necesidades modernas, además en Febrero del 2015 se anunció el fin del soporte para la popular versión 1.9.3 y además se ha sugerido que se actualicen las aplicaciones escritas en este lenguajes a la versión 2.0.0 o superior.

2.3.1.2. Enfoque

Ruby es un lenguaje basado en lenguajes como Perl, Smalltalk, Eiffel, Ada y Lisp e incorpora paradigmas como la programación funcional y la programación imperativa.

Matsumoto, su creador, se refiere a Ruby como un lenguaje enfocado en la simplicidad y productividad lo cual lo ha recalado al decir:

“Ruby es simple en apariencia, pero muy complejo por dentro, como el cuerpo humano.” [23]

Este enfoque hace que el lenguaje guarde más relación con cómo piensan los humanos a que cómo funcionan las máquinas, de aquí se desprende el principio básico de Ruby, el principio de la mínima sorpresa, a pesar de que esto puede sonar subjetivo puesto que dependiendo desde el lenguaje que domina cada desarrollador, los conceptos del uno y del otro pueden variar, por tanto el verdadero significado de este principio apunta a que la confusión de los usuarios experimentados del lenguaje se minimice.

Entre las principales características de Ruby se encuentran:

- Completamente orientado a objetos, es decir, todo es un objeto.
- Tipado dinámico, también conocido como duck typing.
- Todo se ejecuta imperativamente, para Ruby todo es una expresión.
- La gestión de paquetes está centralizada por medio de RubyGems.
- Manejo de excepciones.
- Recolección de basura.
- Posee una consola interactiva conocida como Interactive Ruby Shell.
- Es multiplataforma, se ejecuta sobre una máquina virtual.

2.3.2. Ruby on Rails

Ruby on Rails es un framework para desarrollo web de código abierto construido usando Ruby como lenguaje. En esta sección, se revisarán sus características.

2.3.2.1. Historia

La primera versión de RoR fue liberada en Julio del 2004 por David Heinemeier Hansson como un proyecto de código abierto pero no fue hasta Diciembre del 2005 que el framework alcanzo su primera versión estable (Rails 1.0).

El framework RoR fue un proyecto basado en el trabajo que desarrolló Heinemeir en Basecamp, una herramienta de gestión de proyectos.

Desde su origen se han ido integrando características como la integración de REST, routing para recursos, autenticación básica HTTP, almacenamiento de sesión en cookies, i18n para internacionalización, connection pools, JRuby que es un motor basado en la java virtual machine, incorporación de plantillas para las vistas, motor para consultas, controlador para email y se ha integrado el uso de otras librerías que soportan el desarrollo web como jQuery, SASS, CoffeeScript y Sprockets.

Actualmente, RoR es muy popular entre las personas que empiezan con el desarrollo web y es usado además como plataforma de entrada para cursos académicos. Rails ha sido implementado en grandes aplicaciones como Twitter en sus inicios, Soundcloud, Hulu o Github.

2.3.2.2. Enfoque

Ruby on Rails se basa en dos principios, DRY (Don't repeat yourself) o "no te repitas" y en la convención sobre la configuración, ambos orientados a que el desarrollo sea más rápido y requiera menos líneas de código.

- **No te repitas:** Se refiere a que las definiciones realizadas por el programador deberían hacerse una única vez dado que los componentes están integrados. RoR logra esto gracias a que implementa una metodología "full stack" o de pila completa lo cual hace posible que no haga falta establecer puentes entre los componentes pues ya se encuentran integrados. Cuando este principio es aplicado el resultado es que se tienen menos líneas de código en la aplicación lo cual quiere decir que su implementación fue más rápida y deja menos espacios a que

contenga errores, la gran ventaja de esto es que finalmente se tiene un código fácil de entender y mantener.

- **Convención sobre configuración:** RoR defiende hacer las cosas por convención y más no por configuración, para lograr esto posee una configuración predefinida basada en convenciones de desarrollo, este principio consiste en que el programador sólo necesita escribir la configuración no convencional, lo que ocurre en la minoría de las veces y por tanto, en la mayoría de los casos el tiempo destinado a configurar el framework es prácticamente nulo.

2.3.2.3. Arquitectura

Rails está basado en el patrón MVC que consiste en tres capas conocidas como modelo, vista y controlador.

2.3.2.3.1. La arquitectura MVC

En la Figura 2.5, se visualiza la arquitectura MVC y cómo las capas interactúan entre ellas. Las características de cada una de estas capas se presentan a continuación:

- **Modelo:** En esta capa es donde se trabaja con los datos, es decir, guarda relación con la base de datos de la aplicación.

Contiene los mecanismos para acceder a la información y poder agregar, eliminar o actualizar registros.

- Vistas: Es la capa en donde se visualiza la interfaz gráfica con que el usuario interactúa.
- Controlador: Esta capa funciona como puente entre la vista y el modelo, por tanto contiene el código necesario para responder a las acciones lanzadas desde la interfaz de usuario y también el código necesario para solicitar la información requerida al modelo.

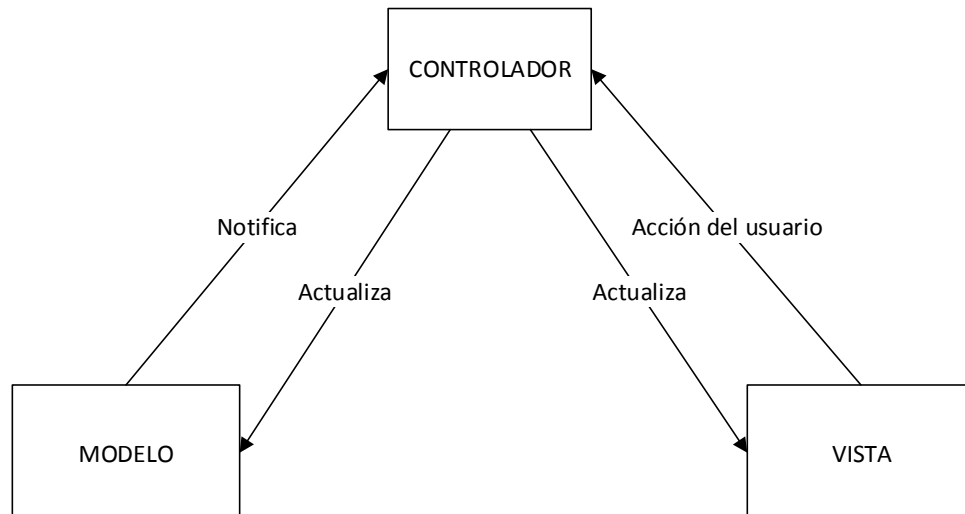


Figura 2.5 Arquitectura MVC

2.3.2.3.2. MVC en RoR

En una aplicación web desarrollada en Rails, el navegador envía una solicitud que es recibida por un servidor web y se transmite al controlador; el controlador interactúa con el modelo, un objeto en Ruby que representa un elemento de la página (vista) y se comunica con la base de datos. Cuando el modelo devuelve el resultado al controlador, este último genera una nueva vista y la devuelve al navegador. A continuación, se mencionan algunos aspectos de cada capa en de la arquitectura MVC en RoR.

- Modelo:
 - Una clase que representa una tabla.
 - Contiene los métodos para la manipulación de los datos.
 - Contiene las reglas de validación de los datos.
 - Contiene las declaraciones de las relaciones con otros modelos.
 - Las clases deben ser nombradas en singular y se guardan en archivos con extensión rb: documento.rb, usuario.rb, etc.
 - Los modelos son manejados por el ORM Active Record (modulo incluido en RoR).

- Vista:
 - Usa plantillas en formato Embedded Ruby por defecto.
 - Maneja HTML con Ruby embebido.
 - Una vista por cada acción de cada controlador.
 - El módulo para controlar las vistas es llamado Action View.

- Controlador:
 - Contiene la lógica de la aplicación y los cálculos.
 - Un controlador es una clase escrita en Ruby.
 - Cada método dentro del controlador corresponde a una acción.
 - El módulo de Rails para manejar los controladores es llamado Action Controller.

2.3.2.3.3. Peticiones HTTP en RoR

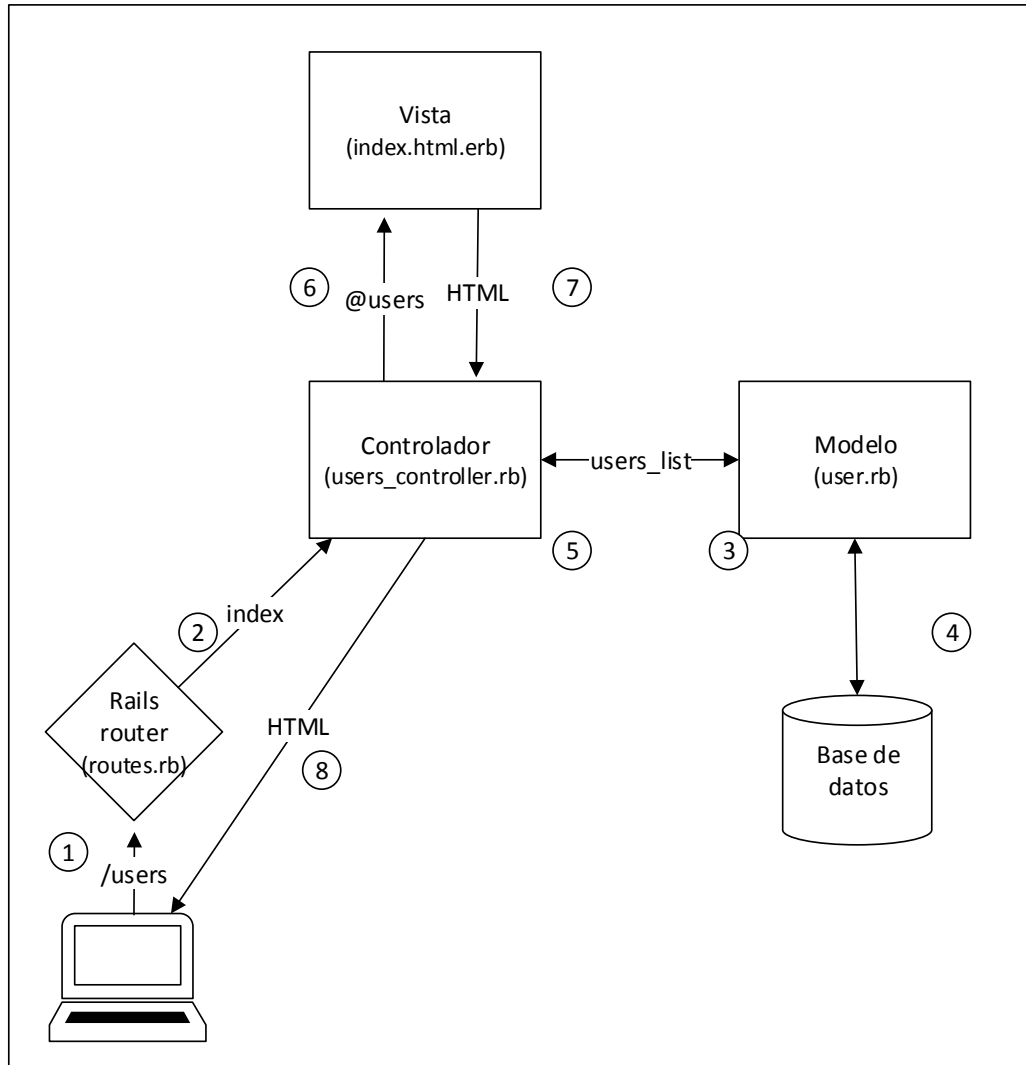


Figura 2.6 Procesamiento de una petición HTTP desde la perspectiva MVC de RoR

1. El navegador emite una solicitud a una URL, por ejemplo a `/users`.
2. Rails enruta `/users`, por medio del archivo `routes` a la acción `index` del controlador usuarios llamado `users_controller`.

3. La acción index pide al modelo de usuario "user.rb", recuperar los usuarios solicitados por el controlador.
4. El modelo de usuario obtiene los usuarios solicitados de la base de datos.
5. El modelo de usuario devuelve la lista de usuarios con el controlador.
6. El controlador captura los usuarios en la variable @users, que es pasada a la vista /users/index (o bien /users).
7. La vista utiliza código Ruby embebido para representar la página como HTML.
8. El controlador pasa el código HTML al navegador.

CAPÍTULO 3

ANÁLISIS Y PREPARACIÓN DE AMBIENTE

3.1. Análisis de ingeniería inversa

Como ya se ha comentado, en este trabajo se va realizar ingeniería inversa del sistema PYX4 enfocada en la lógica del sistema con el fin de documentar su arquitectura y facilitar su mantenimiento.

El proveedor del sistema ha concedido el acceso al código fuente del sistema, el cual se encuentra escrito en Ruby que es un lenguaje interpretado, gracias a esto no será necesario el uso de descompiladores para poder estudiar directamente el código.

La estrategia seleccionada para desarrollar este proceso de ingeniería inversa será la conocida como “outside-in”, la cual consiste en que a partir de estímulos externos al sistema, los cuales están relacionados a los requerimientos funcionales iniciales, se va a determinar los procesos lógicos que los manejan así como la integridad de los datos asociados. Como apoyo a esta estrategia se considerará el modelo “4+1” puesto que su punto de partida está directamente relacionado a los casos de usos los cuales corresponden al “+1” del modelo, esta última vista sirve como eje integrador a las demás vistas arquitectónicas del software, lo que finalmente facilitará la lectura de los diagramas generados.

En la Figura 3.1 se presenta el marco general con el cual se trabajará este proyecto, las fases a considerar son:

- Análisis de requerimientos: aquí se identifican los requerimientos del sistema del cual se hará la ingeniería inversa y que estén asociados a los nuevos. De este análisis se establecerán los escenarios y casos de uso que serán usados para el modelo “4+1”.
- Recuperación de la implementación: desde el código fuente suministrado procederemos a desplegar el sistema con el fin de poder realizar el estudio con el enfoque “outside-in” y considerando los casos de uso definidos anteriormente.

- Recuperación del diseño: una vez que el sistema se encuentra desplegado y los casos de uso definidos se procederá a realizar los diagramas UML. Su elaboración se hará desde un punto de vista estático y dinámico; para la parte dinámica se usarán herramientas de depuración y técnicas de inyección de código.
- Análisis de la recuperación: en esta parte se analizarán los diagramas y vistas arquitectónicas generadas con el fin de aportar descripciones que agreguen valor para quienes revisen la documentación.

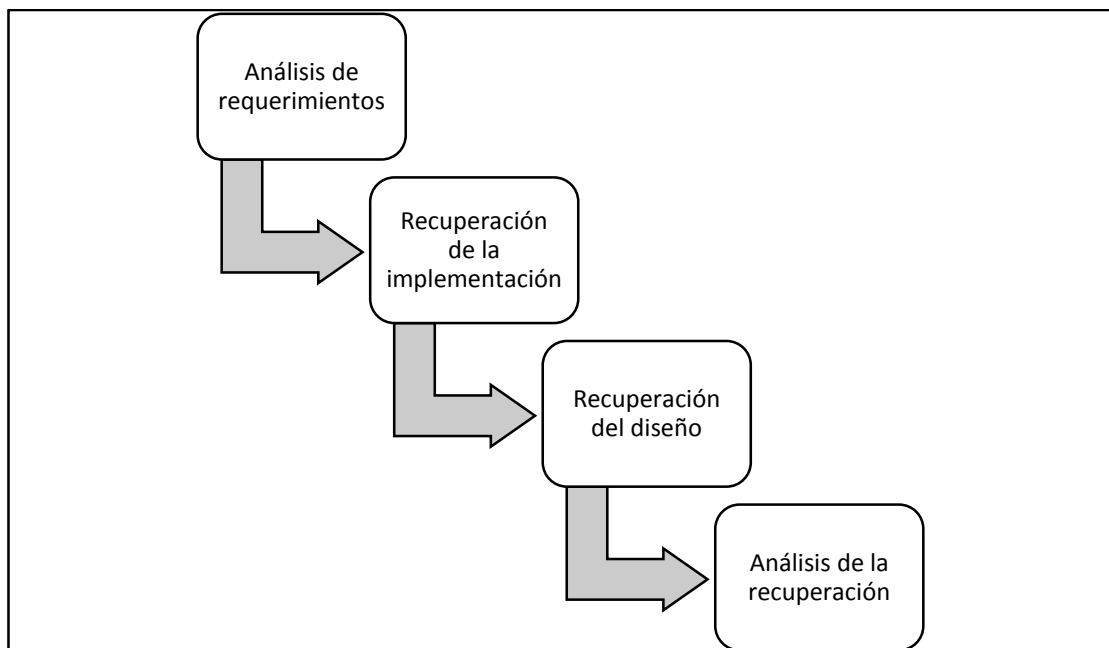


Figura 3.1 Fases del proyecto

3.2. Análisis de requerimientos

3.2.1. Objetivo

Definir los requerimientos a ser atendidos en el proceso de reingeniería del sistema con fin de direccionar hacia ellos el trabajo de ingeniería inversa.

3.2.2. Alcance

Estos requerimientos serán deducidos a partir de las interfaces de usuario del sistema PYX4 y tienen como fin sustentar los casos de uso desde los cuales se organizará la ingeniería inversa. Los requerimientos aquí presentados no corresponden a nuevos requerimientos.

3.2.3. Descripción general

3.2.3.1. Perspectiva del producto

3.2.3.1.1. Interfaces del sistema

- **Interfaz web:** Medio por el cual el usuario final, por medio de una aplicación web, interactúa con el sistema. En esta interfaz es posible acceder a todas las características que el sistema ofrece.

3.2.3.1.2. Interfaces de usuario

- **Interfaz de PXY4:** Esta interfaz maneja permisos para niveles de acceso, por tanto requiere que el usuario inicie sesión para decidir qué pantallas puede visualizar y a qué funcionalidades acceder.

3.2.3.1.3. Interfaces de hardware

El sistema requiere las siguientes interfaces de hardware:

- Tarjeta de red (como interfaz de comunicaciones).
- Monitor (como interfaz de salida).
- Teclado y mouse (como interfaces de entrada).

3.2.3.1.4. Interfaces de comunicaciones

Dada la naturaleza de una aplicación web, el sistema requiere ser ejecutado sobre un sistema operativo y hardware que soporte conexiones a redes de computadoras como una LAN o WAN con tecnología Ethernet.

3.2.3.1.5. Operaciones

En rasgos generales las operaciones de PYX4 son:

- a) Manejo de sesión de usuario.

- b) Dibujar procesos de negocio basados en la notación Qualigram.
- c) Manejar versiones de los procesos dibujados.
- d) Permitir que los diagramas pasen por un proceso de autorización en el cuál intervengan los actores relacionados.
- e) Cargar recursos que pueden estar relacionados a los diagramas de procesos.
- f) Cargar documentos que pueden estar relacionados a los diagramas de procesos.
- g) Administrar usuarios y grupos de usuarios, así como también sus permisos.
- h) Manejar notificaciones en la interfaz web del sistema acerca de eventos de los procesos en que el usuario que ha iniciado sesión esté involucrado.

3.2.3.2. Funciones del producto

A continuación, se nombran las funciones del producto:

- a) Documentar y gestionar procesos de negocio.
- b) Realizar autenticación de usuarios.
- c) Controlar las versiones de los procesos.

d) Cargar recursos relacionados a los procesos de negocio.

3.2.3.3. Características de usuario

El sistema está desarrollado para ser usado por cualquier miembro que sea parte de un proceso de negocio, por lo tanto serán personas experimentadas en el manejo de sistemas computacionales.

3.2.3.4. Limitaciones

Bajo la premisa de que los usuarios del sistema son parte de una organización, se desprende que se deben de manejar permisos para obedecer a la organización jerárquica y de especialización de funciones que puedan establecerse al interior de estas. La limitación evidente es que cada usuario puede sólo usar las funciones que le correspondan.

3.3. Preparación de ambiente

El instructivo detallado del proceso de despliegue del sistema se especifica en el ANEXO A de este documento. Como lineamientos generales se pueden considerar los siguientes puntos:

- Establecer el sistema operativo sobre el cual se desplegará el sistema. Ruby al ser un lenguaje portable permite elegir el sistema operativo, para fines de este estudio se realizará el despliegue de PYX4 sobre Windows 8.1.

- Instalar Ruby y Rails en las versiones especificadas.
- Instalar los sistemas requeridos por el software PYX4 para trabajar, estos son bases de datos y aplicaciones que contengan las librerías necesarias para que las gemas de Ruby se instalen.
- Migrar los modelos de datos desde Ruby hacia el motor de base de datos.
- Preparar los datos de configuración iniciales en la base de datos para poder tener acceso al sistema, esto es parámetros de inicialización, usuarios y permisos.
- Ejecutar el comando Rails para levantar el servidor y el sistema PYX4.

3.4. Consideraciones adicionales

Como generalidad se debe considerar que el rendimiento del despliegue realizado está relacionado a que la aplicación está corriendo sobre el servidor de aplicaciones WEBrick el cual está recomendado sólo para ambientes de desarrollo pues no ha sido diseñado para manejar un nivel alto de concurrencia.

CAPÍTULO 4

INGENIERÍA INVERSA Y RESULTADOS

4.1. Arquitectura del sistema

En esta sección se presentará la arquitectura de la implementación del sistema que corresponde a la misma implementada por Ruby on Rails; de este último se detallarán sus componentes o módulos y se los representará por medio de un diagrama de componentes de UML. Se mostrará también un modelo del despliegue del sistema representado en un diagrama de despliegue y finalmente se revisará la estructura de archivos del framework, esto es, la descripción de cómo se distribuyen sus directorios y los archivos dentro de ellos.

4.1.1. Componentes del sistema

Los componentes en Rails se expresan como módulos, un módulo en Ruby es una colección de “objetos” agrupados bajo un nombre único. Estos objetos pueden ser clases, métodos, constantes u otros módulos. Los módulos no son instanciables y cumplen con dos funciones principales: proveer un mecanismo de encapsulación de objetos y permitir añadir funcionalidad a una clase por medio de los conceptos de interface o herencia.

Rails hace uso de esta característica de Ruby para definir los módulos que hacen parte su núcleo y definen su arquitectura lógica. A continuación, se presentarán los módulos existentes en la versión 3.2.12 de RoR:

- a) Action Pack: provee las características necesarias para soportar las capas de vista y controlador del modelo MVC, para hacerlo cuenta con tres submodulos.
 - i. Action View: su función es administrar el renderizado de plantillas para generar las vistas HTML que se muestran al usuario cuando hace una petición. Este componente tiene soporte para tres tipos de plantillas: rhtml para la generación de vistas HTML por medio de ERB que es una

plantilla HTML con código Ruby embebido, rjs para crear código javascript dinámico y rxml para construir archivos XML usando Ruby.

- ii. Action Controller: procesa las peticiones de la aplicación, extrae sus parámetros y los envía a la acción correspondiente. Además, controla la generación de las vistas y redirección, la gestión de las sesiones de usuarios, así como el almacenamiento en caché, entre otros.
 - iii. Action Dispatch: este componente es el encargado de enrutar y despachar las peticiones web y las envía hacia donde se ha configurado. Maneja además el procesamiento HTTP avanzado como son las cookies, sesiones, entre otros.
- b) Action Mailer: provee los métodos y funciones usadas para que la aplicación pueda enviar y recibir correos electrónicos.
- c) Active Record: este componente permite el mapeo objeto-relacional para clases, es decir, permite conectar las tablas de la base de datos con su representación en clases de Ruby. Active record como tal es un patrón arquitectónico usado para

administrar la data de una base de datos relacional como si fueran objetos.

El módulo es la base para la capa de Modelo del patrón MVC y proporciona independencia entre la base de datos, funcionalidad CRUD con cero configuración, entre otros.

- d) Active Model: provee una interface entre los módulos Action Pack y Active Record. Por ser una interface permite que Rails pueda trabajar con un ORM diferente a Active Record si la aplicación lo requiere.
- e) Active Resource: administra la conexión entre servicios web RESTful y objetos del negocio. Sigue el mismo principio que el módulo de Active Record con la diferencia que en vez de mapear registros desde tablas a objetos, lo hace desde un recurso REST remoto.
- f) Active Support: este módulo está diseñado como una colección de clases utilitarias que facilitan el desarrollo en RoR e incluye soporte, por ejemplo, para zonas horarias, internacionalización y pruebas.
- g) Railties: es el núcleo del código de Rails y el encargado de “engranar” todos los módulos descritos anteriormente. Además provee el motor de generación de código Rails, el cual puede

generar escenarios de pruebas unitarias y funcionales para las vistas y controladores; también se pueden agregar datos de prueba usando YAML.

La forma en que estos componentes interactúan es presentada en el diagrama de componentes de la Figura 4.1.

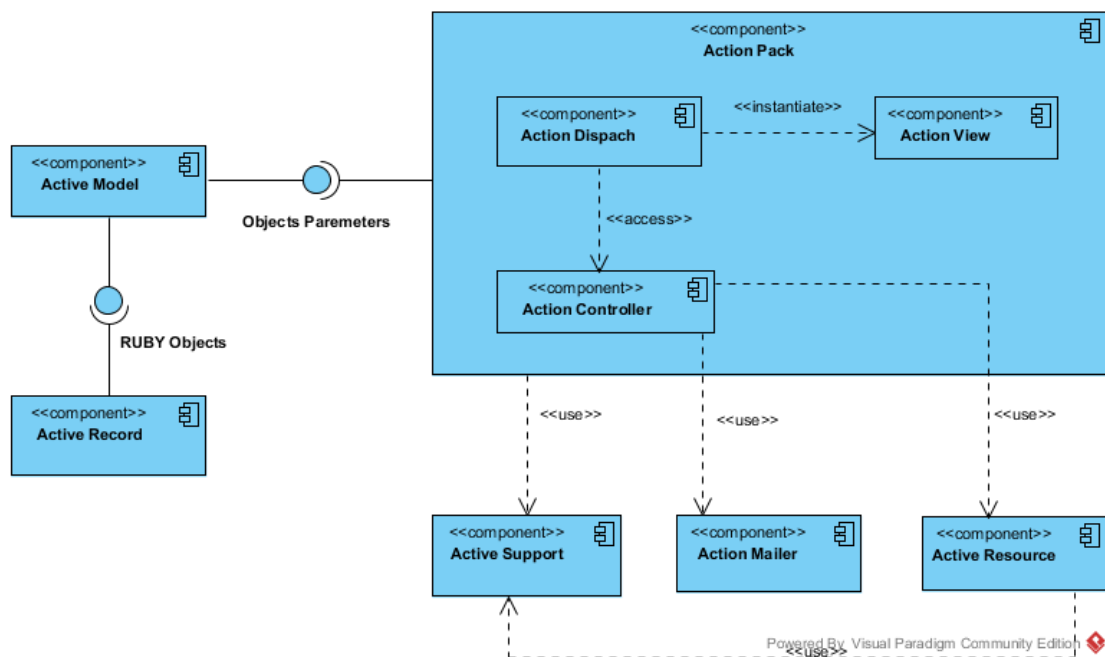


Figura 4.1 Diagrama de componentes RoR 3.2.12

4.1.2. Arquitectura del despliegue

PYX4 es un sistema web pasado en la arquitectura cliente-servidor y posee tres capas o niveles: el cliente, servidor de aplicaciones y servidor de bases de datos. En la figura 4.2, se muestra en el diagrama de despliegue de PYX4.

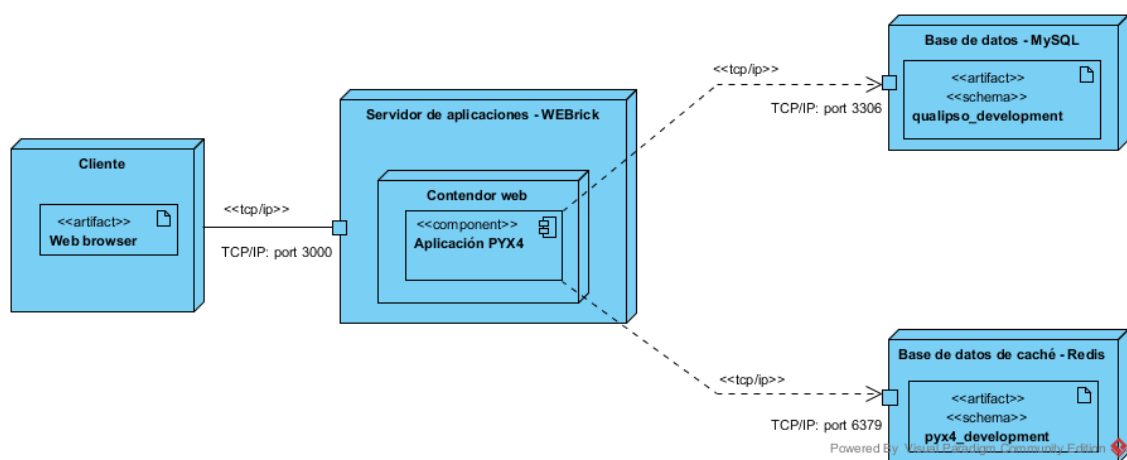


Figura 4.2 Diagrama de despliegue de PYX4

4.1.3. Estructura de directorios

A nivel de código, el sistema está estructurado según la organización de directorios propuesta por RoR, esto facilita la navegación entre los archivos del proyecto puesto que delimita las posibilidades de búsqueda de acuerdo a la funcionalidad que cumple el archivo que se desea encontrar. El código fuente del proyecto se encuentra dentro de una carpeta llamada “qualipso” y dentro de ella el resto de directorios del framework.

En la Figura 4.3, se muestra una captura de la vista del directorio raíz del código fuente del sistema con la carpeta “app” extendida en donde se ha resaltado las carpetas que contiene los archivos correspondientes del patrón MVC, la especificación del propósito de cada una de estas carpetas y archivos están disponibles en la web de RoR [29] y se detallan a continuación.

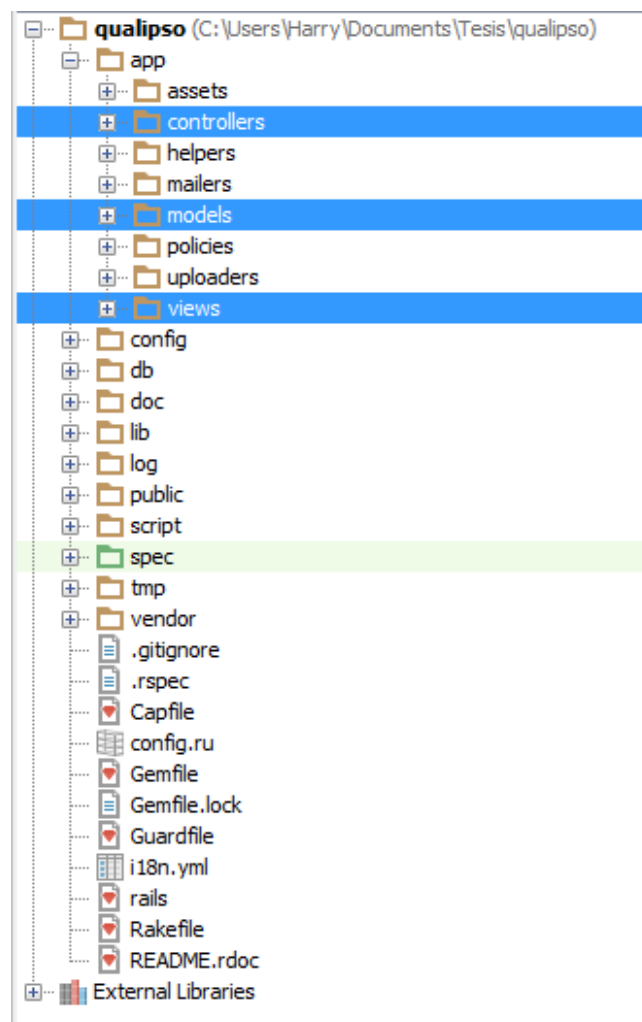


Figura 4.3 Directorio de código fuente de PYX4 - RoR

- Directorios:

- app/

- Contiene todos los recursos web de la aplicación y aquellos involucrados con el patrón MVC.

- config/

- Contiene los archivos usados para configurar características de la aplicación tales como reglas de ejecución, enrutamiento, base de datos y otros.

- db/

- Contiene el schema de base de datos y sus migraciones.

- doc/

- Contiene la documentación de la aplicación.

- lib/

- Contiene los módulos extendidos para la aplicación.

- log/

- Contiene los archivos log de la aplicación.

- `public/`

Contiene archivos estáticos en su formato original y recursos compilados.

- `script/`

Contiene los scripts de Rails usados para iniciar la aplicación, además puede contener otros scripts que el desarrollador requiera para desplegar o ejecutar la aplicación.

- `test/`

Contiene los archivos de pruebas unitarias y otros recursos relacionados.

- `tmp/`

Contiene archivos temporales.

- `vendor/`

Es usado para alojar el código de terceros, como gemas, plugins u otros.

- Archivos:

- config.ru

- Configuración Rack para servidores de este tipo, es usada para iniciar la aplicación.

- Gemfile y Gemfile.lock

- Especifican las gemas de las que depende la aplicación para ejecutarse.

- Rakefile

- En este archivo se localizan y cargan las tareas que pueden ser ejecutadas desde la línea de comandos

- README.doc

- En formato de texto plano es escribe un breve manual de instrucciones de la aplicación.

4.2. Diagramas UML

En esta sección se presentarán los diagramas UML relacionados a la funcionalidad de crear los diagramas de procesos en PYX4.

4.2.1. Diagrama de casos de uso

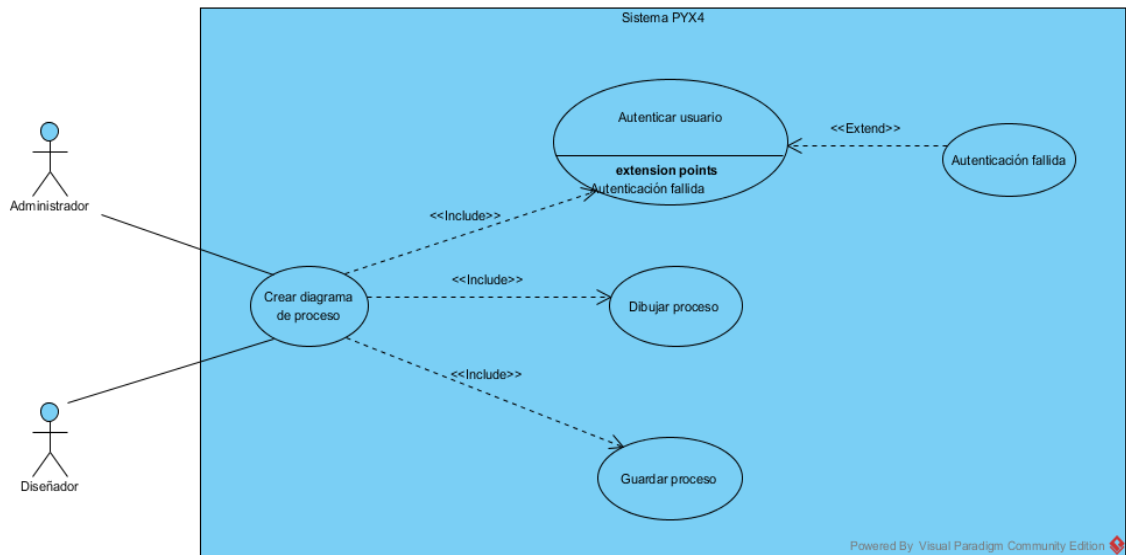


Figura 4.4 Diagrama de casos de uso

En la Figura 4.4, se muestra el diagrama de casos de uso que incluye los casos relacionados al modelado del proceso, donde podemos ver que los actores que pueden ejecutar las funciones de modelado son el administrador y el diseñador, además se visualiza que la tarea de crear el diagrama del proceso consta de otras subtareas como por ejemplo iniciar sesión, dibujar el proceso y guardarlo; siendo estos últimos los casos que se procederán a analizar.

4.2.2. Diagrama de clases – GraphController PYX4

En el sistema interactúan múltiples clases dedicadas a diferentes propósitos. En esta sección se presenta el diagrama correspondiente a las clases dedicadas a controlar el proceso de dibujar el gráfico del modelo de negocio, las cuales están centradas en la clase escrita en el archivo */app/controllers/graph_controller.rb*.

En RoR podemos distinguir dos tipos de clases, por una parte las clases de los modelos ubicadas en la carpeta */app/models* y por otra las clases de los controladores ubicadas en la carpeta */app/controllers/*. Por definición del modelo MVC, las clases de controladores invocan a las clases de los modelos, por lo tanto estas poseen como atributos variables de los tipos de datos de los modelos y además definen métodos para responder a las peticiones del usuario lanzadas desde las vistas. Por su parte las clases de modelos contienen definiciones de atributos propios del modelo de datos y además los métodos para validarlos y filtrarlos, así como también para insertar, leer, actualizar y eliminar sus registros; se debe recordar que por cada modelo se define una tabla en la base de datos y que este modelo puede comunicarse

con la base de datos gracias a que hereda de la clase *ActiveRecord::Base* la cual es parte central de RoR.

En la Figura 4.5, se observa el diagrama de clases para GraphController en donde a las clases de los modelos sólo se ha incluido sus atributos puesto que si bien es cierto también tienen métodos, estos no cobran mayor relevancia dentro del proceso de graficado.

4.2.3. Diagrama de estados de gráfico

En el sistema, un gráfico al igual que un documento pasa por diferentes estados antes de ser finalmente publicado. En la Figura 4.6, se muestra el diagrama de estados de un diseño de procesos en donde se ve que el estado inicial es el identificado como “In progress” luego de ser creado, y continúa hasta estar en estado “Applicable”; en este último estado no es posible retroceder a estados previos.

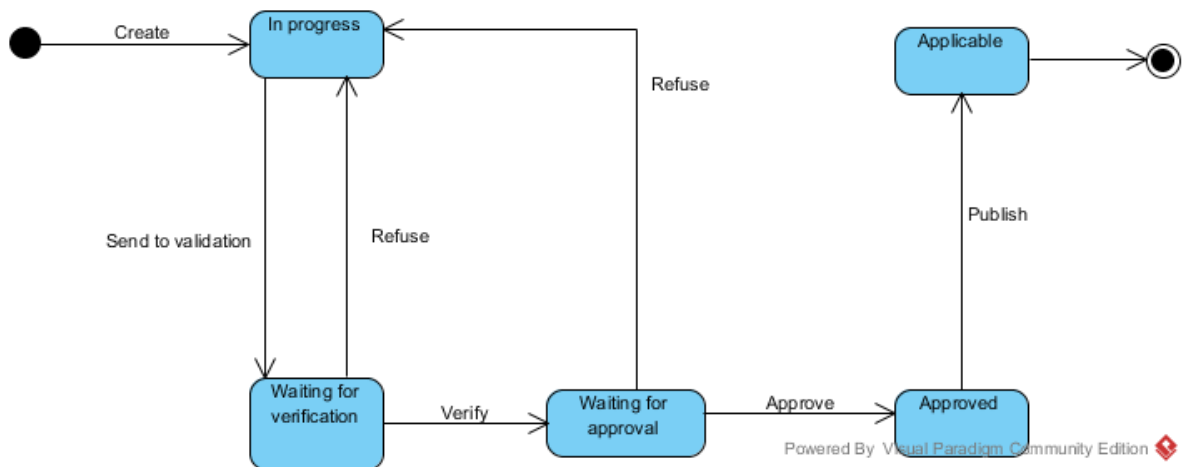


Figura 4.6 Diagrama de estados de Gráfico en PYX4

4.3. Diagrama de entidad relación

Un diagrama ER es un gráfico usado para describir conceptos tales como entidad, atributo y relaciones que en conjunto tienen como fin el modelado de datos y está estrechamente relacionado a la base de datos. Cuando se trabaja en un proyecto de ingeniería directa, se empieza definiendo este diagrama y posteriormente se lo implementa dentro del motor de base de datos.

En este trabajo se ha realizado el proceso de ingeniería inversa desde las tablas migradas del código fuente Rails del sistema PYX4 con el fin de obtener una vista del modelo de datos. Sin embargo, el resultado fueron entidades no relacionadas como se ve en las figuras 4.7, 4.8 y 4.9.



Figura 4.7 Modelo ER generado por RE - Parte A



Figura 4.8 Modelo ER generado por RE - Parte B

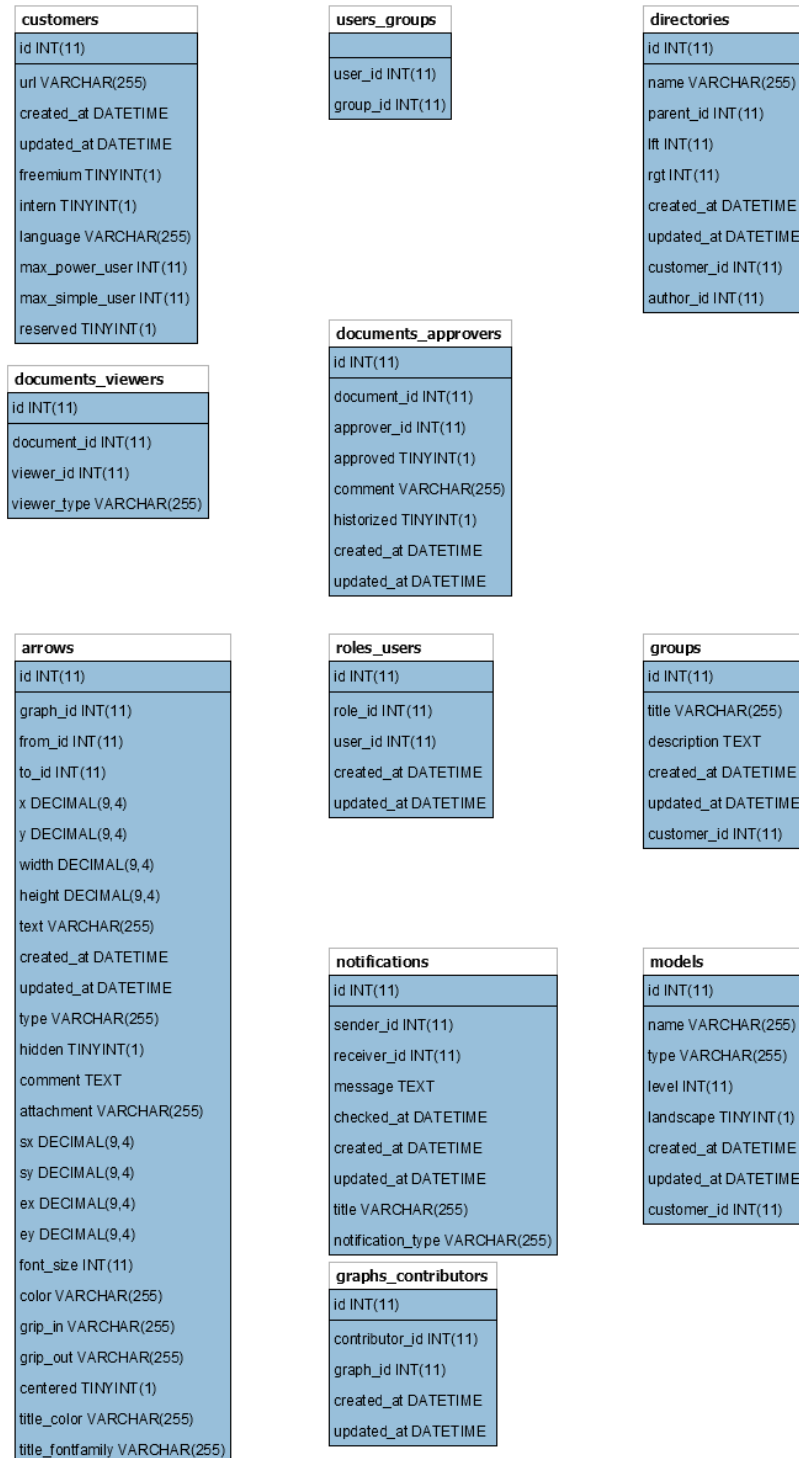
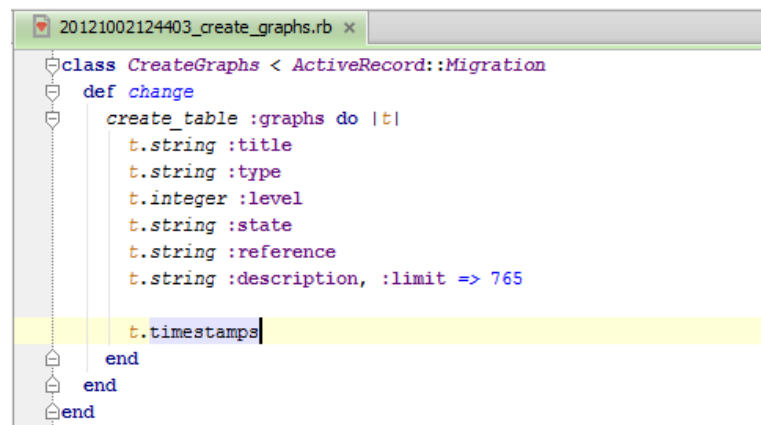


Figura 4.9 Modelo ER generado por RE - Parte C

La causa de tener entidades aisladas se debe a la forma en que Rails realiza la migración de las tablas y sus índices por medio de ActiveRecord, este proceso involucra un archivo en que se definen las características de las tablas a migrar, para explicarlo es posible fijarse en archivo `20121002124403_create_graphs.rb` ubicado en la carpeta `/db/migrate` en el cual se define el código capturado en la Figura 4.8 en donde vemos las columnas declaradas para la tabla “graph”.

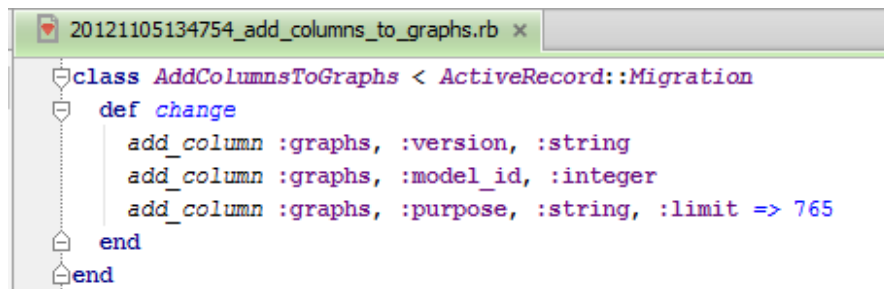


```
class CreateGraphs < ActiveRecord::Migration
  def change
    create_table :graphs do |t|
      t.string :title
      t.string :type
      t.integer :level
      t.string :state
      t.string :reference
      t.string :description, :limit => 765
      t.timestamps
    end
  end
end
```

Figura 4.10 Contenido original de `20121002124403_create_graphs.rb`

En la Figura 4.9 se ve cómo se agrega la columna para guardar el valor del identificador de una de las tablas con que se relaciona, “model”. Sin embargo, no se declara de forma explícita la relación entre ambas lo cual ocasiona que el motor de la migración no cree las claves foráneas que emplea el motor de ingeniería inversa de MySQL para dibujar las relaciones en el diagrama ER. Según se expresa en la documentación

oficial de RoR 3.2.12 [34], en la versión actual para declarar las relaciones en las migraciones se dispone del método “add_reference”, el cual no está disponible para la versión en que se ha implementado PYX4.



```
20121105134754_add_columns_to_graphs.rb x
class AddColumnsToGraphs < ActiveRecord::Migration
  def change
    add_column :graphs, :version, :string
    add_column :graphs, :model_id, :integer
    add_column :graphs, :purpose, :string, :limit => 765
  end
end
```

Figura 4.11 Contenido original de 20121105134754_add_columns_to_graphs.rb

4.3.1. Relaciones de modelos en Rails

La integridad de los datos en RoR está definida directamente en los modelos de objetos que son migrados a la base de datos, por tanto, las relaciones entre las entidades que se buscaba obtener al hacer ingeniería inversa al esquema de datos es posible observarlas en un diagrama de dependencia de modelos en donde se grafican los modelos, las relaciones entre ellos y la cardinalidad de estas.

En el diagrama postrado en la Figura 4.10 se observan los conceptos equivalentes al modelo de entidad relación lo cual permite entender cómo se relacionan los modelos de datos del sistema PYX4.

CAPÍTULO 5

DISCUSIÓN DE RESULTADOS

En este capítulo se realizará el análisis de los resultados obtenidos y la comprensión lograda acerca del sistema referenciando a las implicaciones y las limitaciones.

5.1. Implicaciones

PYX4 fue desarrollado para desempeñarse en un ambiente web. Para los fines de este trabajo se desplegó todos sus componentes en un solo equipo físico, lo cual en ambiente de producción o desarrollo no necesariamente debe ser así, es decir que se pueden separar cada uno de los módulos en servidores diferentes dependiendo de los recursos de hardware y de red disponibles.

La información más relevante acerca de los gráficos de modelo de negocio que se dibujan en el sistema es la almacenada en la base de datos pues es la que se puede interpretar para definir las relaciones entre los elementos propios de la notación implementada.

5.2. Limitaciones

Los diagramas aquí presentados están enfocados a satisfacer información específica y no corresponden a todos los diagramas posibles del sistema. Las herramientas de análisis automatizado disponible están orientadas a análisis estructurales de los modelos de datos en RoR, lo cual requiere que el análisis dinámico se realice manualmente. Como referencia en el Anexo B se ha agregado un diagrama en donde se grafican los todos controladores del sistema.

La funcionalidad del modelador de procesos en PYX4 se encuentra implementada usando el lenguaje Coffeescript que es compilado a Javascript y por lo tanto se ejecuta del lado del navegador y en el servidor no ocurre nada hasta que el diagrama es guardado. Debido a esto el código escrito en Ruby en el que se centró este trabajo, sólo interviene a la hora de escribir los registros correspondientes en la base de datos.

CONCLUSIONES

1. Todo proceso de ingeniería inversa debe de estar correctamente delimitado con el objetivo de orientar el análisis en la dirección necesaria.
2. EL proceso de ingeniería inversa disminuye la complejidad del sistema analizado puesto que la documentación resultante permite comprenderlo.
3. Gracias al análisis realizado es posible detectar qué elementos del sistema pueden ser reusados y cuáles requieren mantenimiento más urgente.

4. El desarrollo de RoR y de sus librerías está fuertemente ligado a su comunidad y puede presentar variaciones importantes entre una versión y otra.
5. Las herramientas existentes para el análisis automático de la arquitectura lógica de un sistema escrito en RoR poseen limitaciones a la hora de analizar la relación entre los Controladores y Modelos.
6. En RoR un Modelo corresponde a una tabla de la base de datos.
7. El mejor diagrama para entender la manera en que los datos se relacionan en RoR es el diagrama de modelos.
8. Para obtener la información almacenada en la base de datos relacionada a la representación de un gráfico de proceso diseñado en PYX4 se debe iniciar el análisis desde la tabla Graph.

RECOMENDACIONES

1. El sistema se encuentra escrito usando versiones legadas de Ruby y de Ruby on Rails por lo cual, si existe la necesidad de agregar nuevas características, se debería empezar procurando actualizarlo a las versiones más recientes lo que bien puede desencadenar una serie de errores de compilación y ejecución; sin embargo, es un ejercicio necesario para procurar disminuir la dificultad de mantenimientos futuros. Adicionalmente, la comunidad de Ruby recomienda migrar las aplicaciones escritas en versiones anteriores a las 2.0 hacia versiones posteriores puesto que el soporte en estas se ha discontinuado y da lugar a tener vulnerabilidades de seguridad no deseadas, así mismo las versiones más recientes de RoR soportan características acordes a las tendencias actuales del desarrollo web.

2. Para mantener la información accesible para todo el equipo de software se recomienda levantar un servidor con un sistema dedicado a este fin, como por ejemplo un wiki. Es importante mencionar también que Ruby tiene una herramienta dedicada a generar la documentación de los sistemas desarrollados en este lenguaje llamada RDoc, enfocada a los atributos y métodos de sus clases. Un trabajo a futuro estaría dedicado a subir a un servidor el instructivo de despliegue y la información técnica de la implementación.

BIBLIOGRAFÍA

- [1] Globalliance France, The PYX4 Solution, <http://www.pyx4.co.uk/our-software-solution>, consultado en Marzo del 2015.
- [2] Monsalve C., Abran A., April A., Measuring Software Functional Size from Business Process Models, International Journal of Software Engineering and Knowledge Engineering Vol. 21, 2011.
- [3] Demeyer S., Ducasse S., Nierstrasz O, Object-Oriented Reengineering Patterns, Square Bracket Associates, 2009.
- [4] Abbas A., Jeberson W., Klinsega V., The Need of Re-engineering in Software Engineering, International Journal of Engineering and Technology Volume 2 No. 2, 2012.
- [5] Chikofsky E., Cross J., Reverse Engineering and Design Recovery: A Taxonomy, IEEE Software, 1990.
- [6] Dongre P., Upadhyay A., Tapsavi N., Reverse Engineering Approach to Instatement of Design Artifacts, International Journal of Scientific Engineering and Technology, Volumen No. 1, 2012.
- [7] Tsui F., Gharaat A., Duggins S., Jung E., Measuring Levels of Abstraction in Software Development, International Conference on Software Engineering and Knowledge Engineering (SEKE), 2011.

[8] Tutorials Point (I) Pvt. Ltd., Software Case Tools Overview, http://www.tutorialspoint.com/software_engineering/case_tools_overview.htm, consultado en Marzo del 2015.

[9] Dennett D. (Center for Cognitive Studies), Cognitive Science as Reverse Engineering: Several Meanings of “Top-Down” and “Bottom-Up”, <http://users.ecs.soton.ac.uk/harnad/Papers/Py104/dennett.eng.html>, consultado en Marzo del 2015.

[10] Riva C., Selonen P., Systä T., Xu J., UML-Based Reverse Engineering and Model Analysis Approaches for Software Architecture Maintenance, IEEE International Conference on Software Maintenance (ICSM'04), 2004.

[11] Budd T., An Introduction to Object-Oriented Programming, Addison Wesley Longman 3rd Ed, 2002.

[12] Akin J., Object Oriented Program Concepts, <https://www.clear.rice.edu/mech517/Books/oop3.pdf>, consultado en Marzo del 2015.

[13] Martinez A., Introduction to Object-oriented Programming (OOP), <https://weblogs.java.net/blog/potty/archive/2014/01/20/introduction-object-oriented-programming-oop-part-i> , consultado en Marzo del 2015.

[14] Eckel B., Thinking in Java, Prentice Hall 4th Ed., 2006.

- [15] Booch G., Rumbaugh J., Jacobson I., The Unified Modeling Language User Guide, Addison Wesley 1st Ed., 1998.
- [16] Booch G., Maksimchuk R., Engle M., Young B., Conallen J., Houston K., Object-Oriented Analysis And Design With Applications, Addison Wesley 3rd Ed., 2007.
- [17] Rumbaugh J., Jacobson I., Booch G., The Unified Modeling Language Reference Manual., Addison Wesley 2nd Ed., 2004.
- [18] Object Management Group, Unified Modeling Language Resource Page, <http://www.uml.org>, consultado en Marzo del 2015.
- [19] Kruchten P., Planos Arquitectónicos: El Modelo de “4+1” Vistas de la Arquitectura del Software., http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:modelo4_1.pdf, consultado en Marzo del 2015.
- [20] PuntoAbierto, Diferencias entre Frameworks y Lenguajes, <http://puntoabierto.net/blog/diferencias-entre-frameworks-y-lenguajes/> , consultado en Marzo del 2015.
- [21] Ruby, Acerca de Ruby, <https://www.ruby-lang.org/es/about/> , consultado en Marzo del 2015.

[22] Stewart B., An Interview with the Creator of Ruby, <http://www.linuxdevcenter.com/pub/a/linux/2001/11/29/ruby.html> , consultado en Marzo del 2015.

[23] Matsumoto Y., Matsumoto hablando en la lista de correo Ruby-Talk, <http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/2773> , consultado en Marzo del 2015.

[24] Basu S., Ruby on Rails Study Guide: The History of Rails, <http://code.tutsplus.com/articles/ruby-on-rails-study-guide-the-history-of-rails--net-29439> , consultado en Marzo del 2015.

[25] Chrome, MVC Architecture, https://developer.chrome.com/apps/app_frameworks, consultado en Marzo del 2015.

[26] Corbridge R., Introducing Ruby ON Rails Part One, <http://www.softwaredeveloper.com/features/intro-to-ruby-on-rails-042507/> , consultado en Abril del 2015.

[27] IEEE, IEEE Recommended Practice for Software Requieriments Specifications, IEEE Std 830-1998, 1998.

[28] Software Engineering Institute, How to Document the Architecture of Your Application Using UML and More, <http://www.scribd.com/doc/179906596/How-to-Document-the-Architecture-of-Your-Application-Using-UML-and-More> , consultado en Abril del 2015.

[29] Rubyonrails.org, Getting Started with Rails, http://guides.rubyonrails.org/v3.2.12/getting_started.html , consultado en Abril del 2015.

[30] uml-diagrams.org, Diagrams overview, <http://www.uml-diagrams.org/uml-24-diagrams.html> , consultado en Abril del 2015.

[31] uml-diagrams.org, UML Classifier, <http://www.uml-diagrams.org/classifier.html>, consultado en Abril del 2015.

[32] Heroku dev center, Ruby Default Web Server, <https://devcenter.heroku.com/articles/ruby-default-web-server>, consultado en Abril del 2015.

[33] B. Knapp, The Ruby Web Benchmark Report, <http://www.madebymarket.com/blog/dev/ruby-web-benchmark-report.html>, consultado en Abril del 2015.

[34] rubyonrails.org, A Guide to Active Record Associations, http://guides.rubyonrails.org/v3.2.12/association_basics.html#belongs_to-association-reference, consultado en Abril del 2015.

ANEXO A

INSTRUCTIVO DE DESPLIEGUE EN WINDOWS

Versión	Fecha de modificación	Descripción de cambios
1.0	13/04/2015	Se agrega instrucción para instalación de Redis.
1.0.1	14/04/2015	Se profundiza en la solución para problema de instalación de gema Rmagick.
1.0.2	15/04/2015	- Se agrega una sección con la instalación de ImageMagick y sus respectivas capturas - Se indica cómo levantar el servidor Redis.
1.0.3	16/04/2015	Se agrega en la sección de problemas frecuentes las instrucciones de cómo proceder frente a un error con la gema Rake.
1.0.4	16/04/2015	Se agregan las instrucciones para hacer el cambio de contraseña.

Acerca del despliegue del software

El software PYX4 es una aplicación web desarrollada con el framework RoR, por lo cual puede ser desplegado en Windows, MacOS o Linux puesto que Ruby, el lenguaje en que está escrito, es portable.

Antes de empezar

Este manual se basará en el proceso a realizar sobre un equipo con las siguientes características:

- Procesador : Intel Core i7
- RAM: 4 GB
- S.O: Windows 8.1 x64

PYX4 está desarrollado considerando las siguientes versiones:

- Ruby 1.9.3
- Rails 3.2.12

Instalar Ruby y Rails

En Windows se dispone de paquetes “.exe” que permiten la instalación de Ruby y RoR por medio de un asistente gracias a iniciativas como “rubyinstaller.org” y “railsinstaller.org “, los programas que se necesitan para ejecutar RoR son:

- Ruby
- Rails
- Devkit

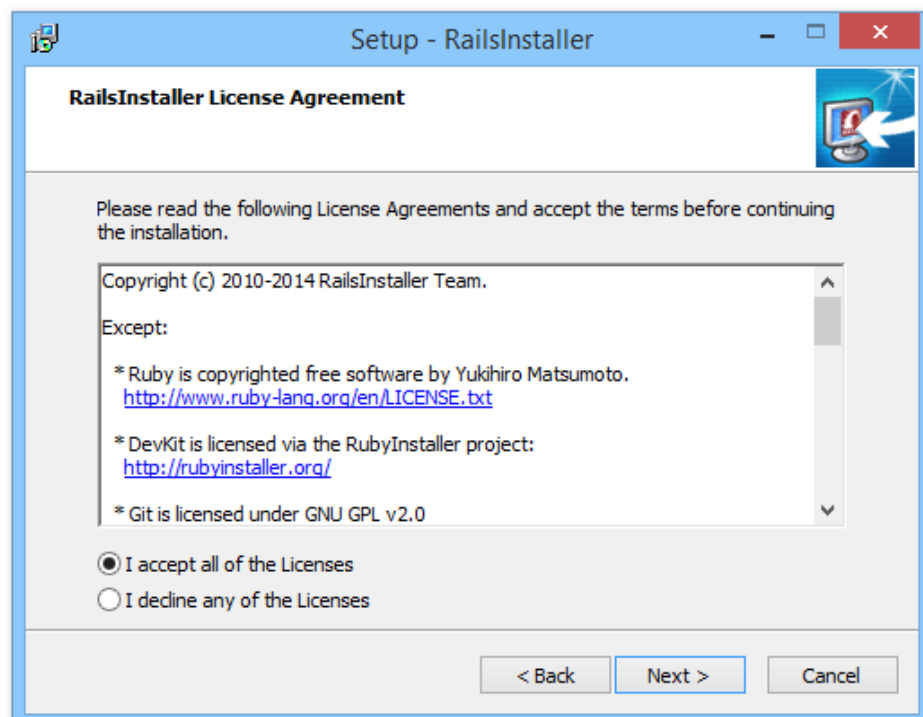
El proyecto “railsinstaller.org” es patrocinado por la compañía Engine Yard Inc. Desde su sitio web (<http://railsinstaller.org/>) se puede encontrar el instalador railsinstaller-2.2.5.exe en el siguiente link:

<https://s3.amazonaws.com/railsinstaller/Windows/railsinstaller-2.2.5.exe>

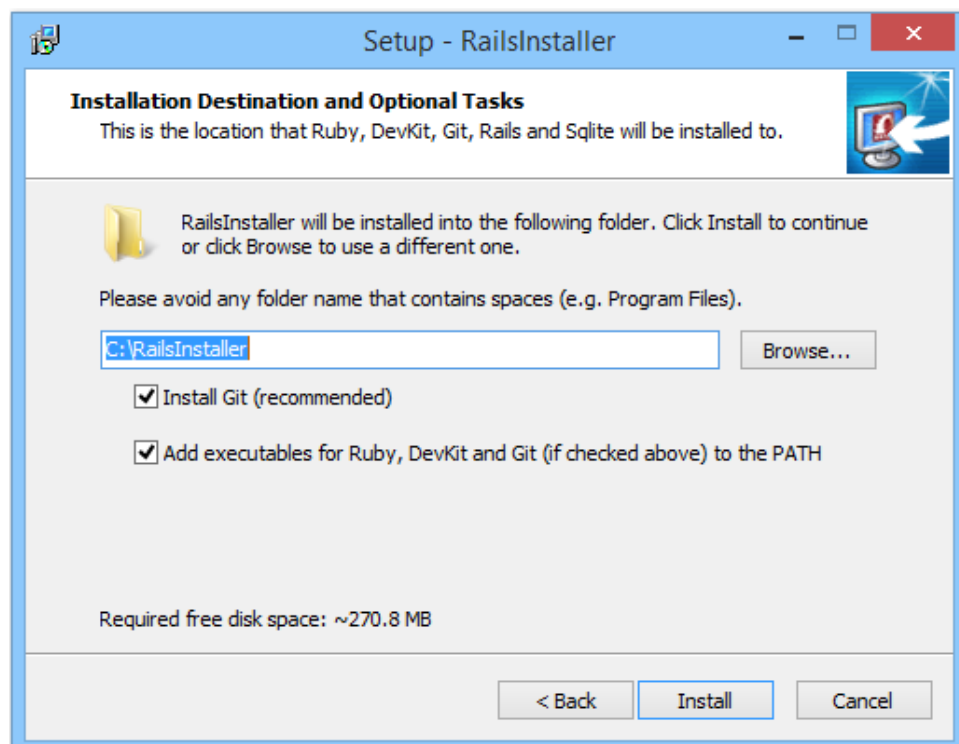
Cuando la descarga se complete, se debe ejecutar el archivo railsinstaller-2.2.5.exe. A continuación, en la ventana que se abre se da clic en “Next >”.



Seguido de esto, se deben aceptar los términos de la licencia para luego dar clic en "Next".



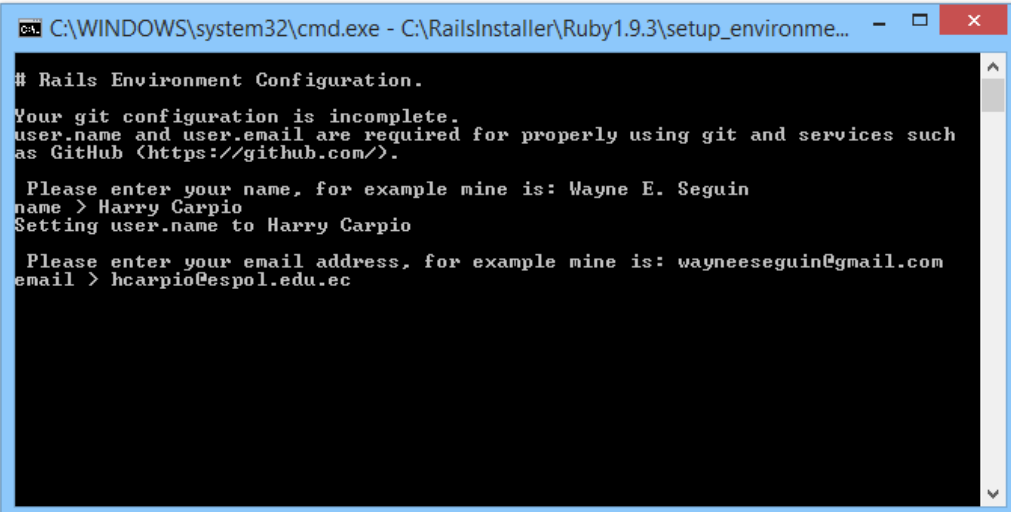
En la siguiente ventana, se debe elegir el directorio de instalación, para lo cual en este caso se dejará el que parece de defecto. Adicionalmente, se debe asegurarse que estén habilitadas ambas casillas de verificación para luego dar comienzo a la instalación dando clic en el botón “Install”.



Asegurarse que la casilla de verificación en el cuadro para configurar git y ssh esté activada para luego dar clic en “Finish”.

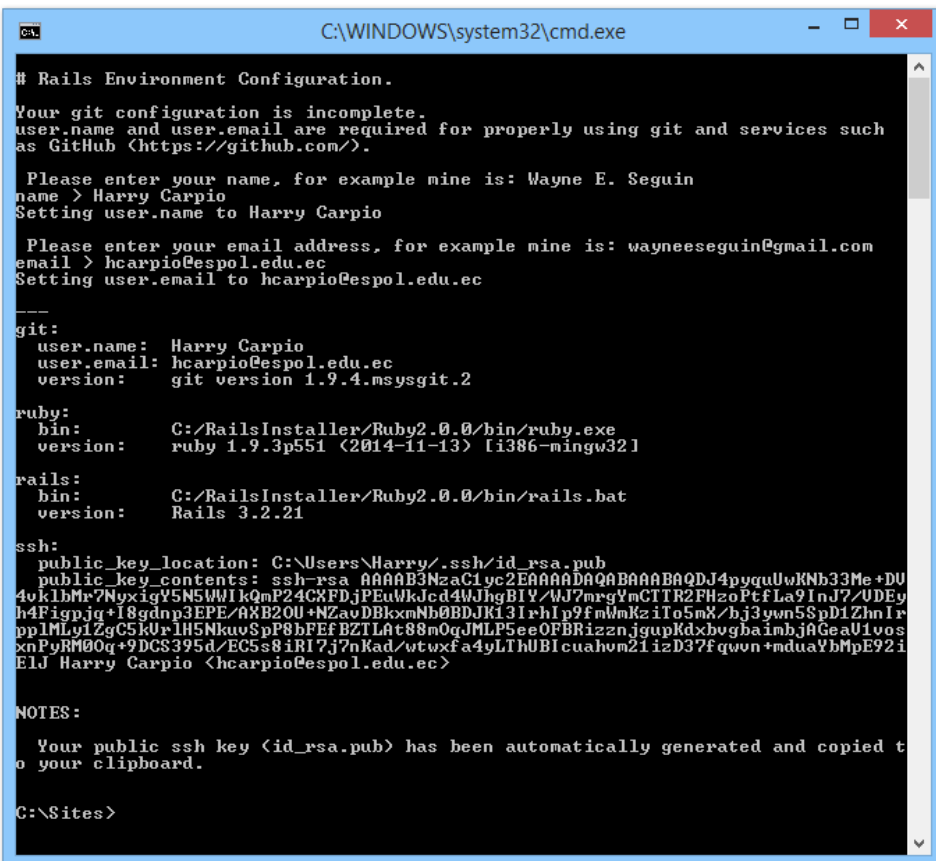


Lo siguiente es ingresar la información de nombre y correo electrónico solicitados.



```
C:\WINDOWS\system32\cmd.exe - C:\RailsInstaller\Ruby1.9.3\setup_environe...  
# Rails Environment Configuration.  
Your git configuration is incomplete.  
user.name and user.email are required for properly using git and services such  
as GitHub (https://github.com/).  
  
Please enter your name, for example mine is: Wayne E. Seguin  
name > Harry Carpio  
Setting user.name to Harry Carpio  
  
Please enter your email address, for example mine is: wayneeseguin@gmail.com  
email > hcarpio@espol.edu.ec
```

En este ejemplo se han ingresado los datos y finalmente aparecerá la pantalla de confirmación, con lo cual se ha configurado con éxito los parámetros de git y ssh.



```

C:\WINDOWS\system32\cmd.exe

# Rails Environment Configuration.

Your git configuration is incomplete.
user.name and user.email are required for properly using git and services such
as GitHub (https://github.com/).

Please enter your name, for example mine is: Wayne E. Seguin
name > Harry Carpio
Setting user.name to Harry Carpio

Please enter your email address, for example mine is: wayneeseguin@gmail.com
email > hcarpio@espol.edu.ec
Setting user.email to hcarpio@espol.edu.ec

---
git:
  user.name: Harry Carpio
  user.email: hcarpio@espol.edu.ec
  version: git version 1.9.4.msysgit.2

ruby:
  bin: C:/RailsInstaller/Ruby2.0.0/bin/ruby.exe
  version: ruby 1.9.3p551 (2014-11-13) [i386-mingw32]

rails:
  bin: C:/RailsInstaller/Ruby2.0.0/bin/rails.bat
  version: Rails 3.2.21

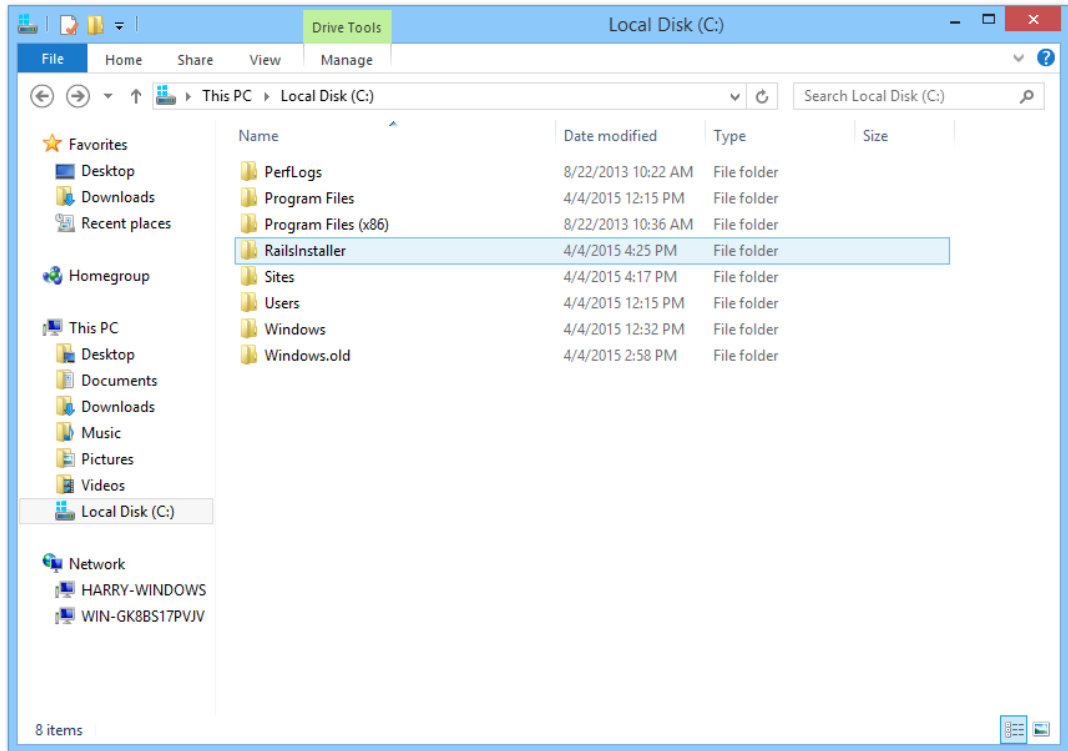
ssh:
  public_key_location: C:\Users\Harry\.ssh\id_rsa.pub
  public_key_contents: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDJ4pyquUwKNb33Me+DU
4uklbMr7NyxigY5N5WJlkQmP24CXFDjPEuWkJcd4WJhgBIY/WJ7mrgVmCTTR2FHzoPtfLa9InJ7UDEy
h4Figpjjg+18gdnp3EPE/AXB2OU+NZa0DBkxmNb0BDJK13lrlhp9fmlmkziTo5mX/hj3ywn5SpD1ZhnIr
pp1MLy1ZgC5kUy1H5Nkuo$pp8bFEfBZTLAt88mOqJMLP5eeOPBRizznjgupKdxbvqbaimbjAGeaU1vos
xnPyRM00q+9DCS395d/EC5s8iRI7j7nkad/wtuxfa4yLthUBIcuahvm21izD37fqwn+mduaYbMpE92i
ElJ Harry Carpio <hcarpio@espol.edu.ec>

NOTES:
  Your public ssh key (id_rsa.pub) has been automatically generated and copied t
o your clipboard.

C:\Sites>

```

Una vez finalizada la instalación, se puede ver que dentro del directorio raíz C:/ se han creado dos carpetas: RailsInstaller y Sites las cuales serán utilizadas más adelante.



Desplegar aplicación Rails

Antes del despliegue

Antes de poder desplegar la aplicación se necesita tener acceso a una base de datos MYSQL, en este instructivo se revisará cómo instalarla en el mismo equipo donde desplegaremos el sistema PYX4.

Instalar MySQL

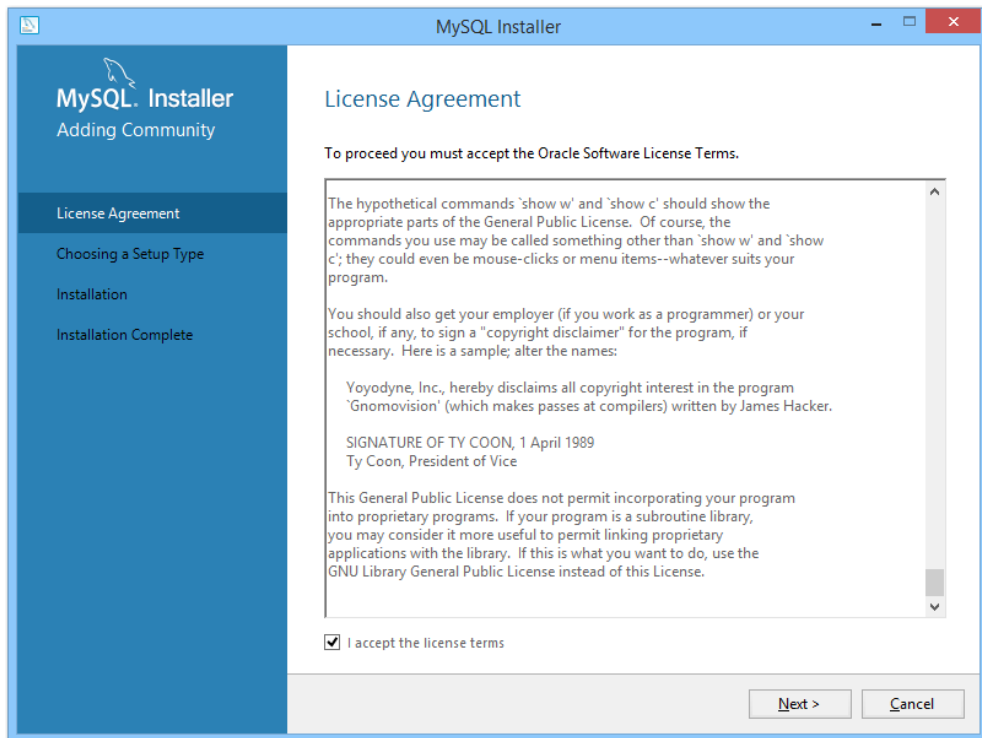
Mysql es una base de datos de código abierto y está disponible en versión empresarial y community; esta última es posible utilizarla gratuitamente. Se procede a instalar la versión más reciente descargando el instalador desde:

<http://dev.mysql.com/downloads/windows/installer/5.6.html>

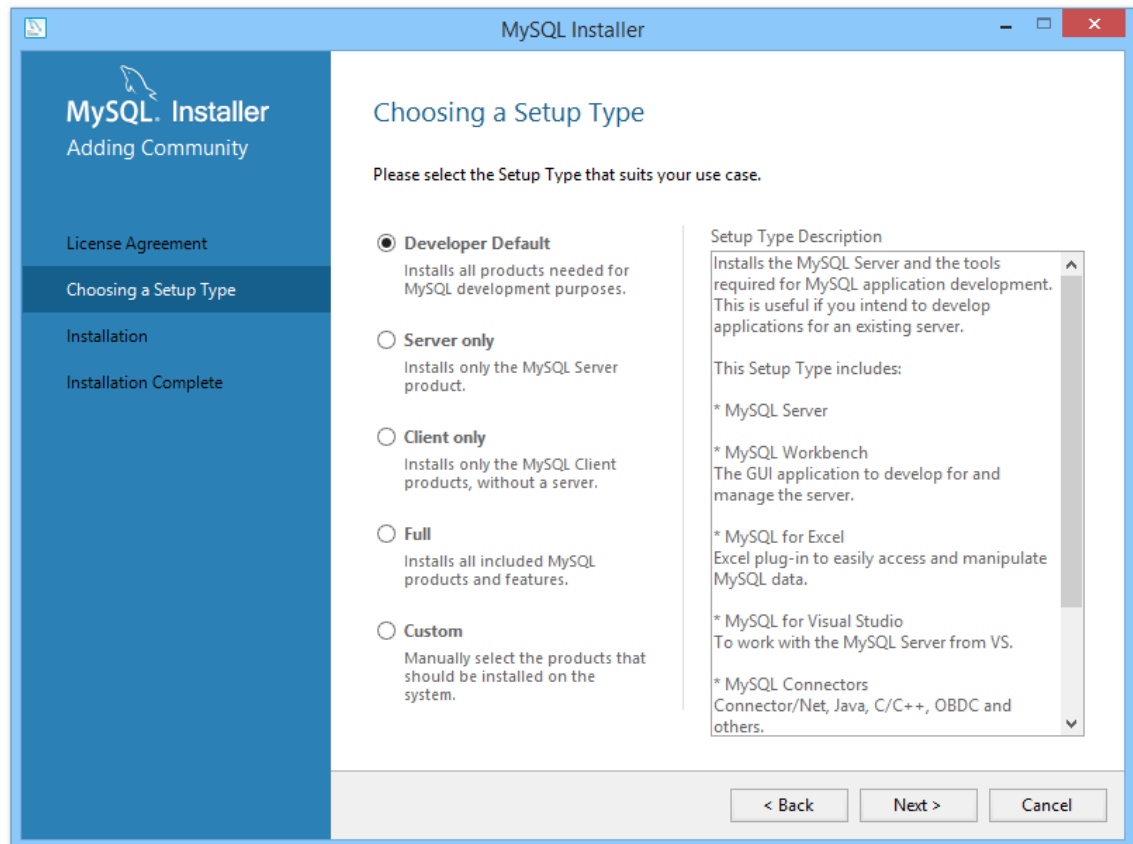
Aquí se puede ver que sólo está disponible el MSI para 32 bits pero puede ser instalado en sistemas tanto de 32 como de 64 bits.

Al abrir el ejecutable se solicitará permiso para ejecutar como administrador el cual debe ser concedido.

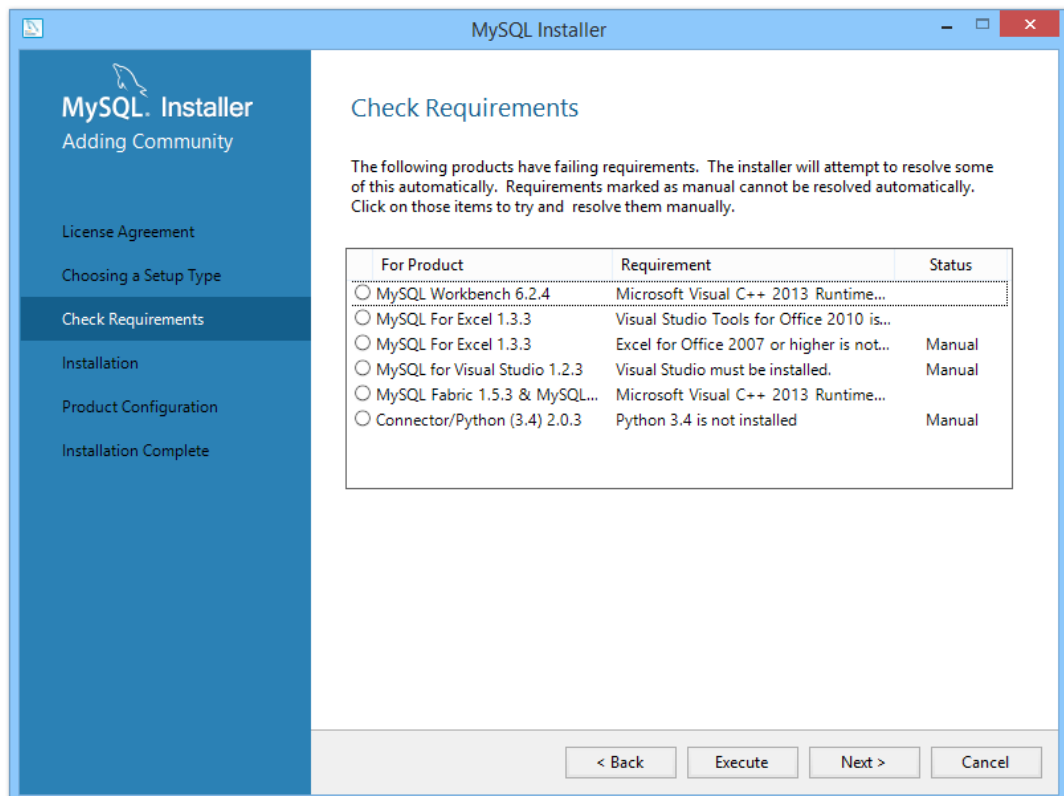
Luego de aceptar los términos de la licencia, se da clic en “Next >”



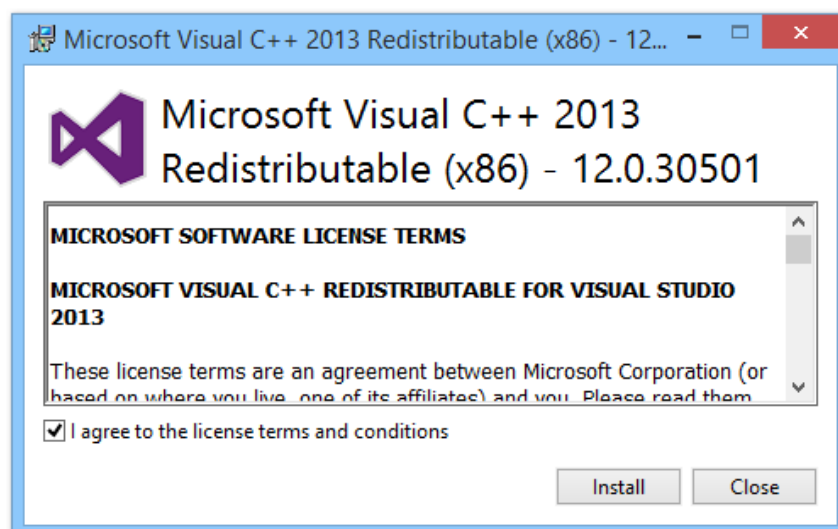
Seleccionar como tipo de instalación "Developer Default" con la cual se instalarán las herramientas y librerías usadas comúnmente por los desarrolladores.



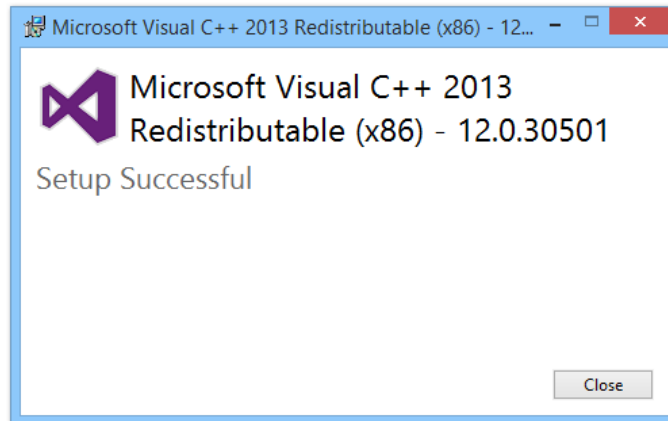
A continuación, se muestra un cuadro donde el instalador chequeará que el sistema tenga instaladas las librerías necesarias para ciertos componentes. Las que nos interesan para nuestro propósito son las relacionadas a MySQL Workbench, un entorno gráfico que nos permitirá interactuar con nuestra base de datos, para instalarlo se debe presionar el botón “Execute”.



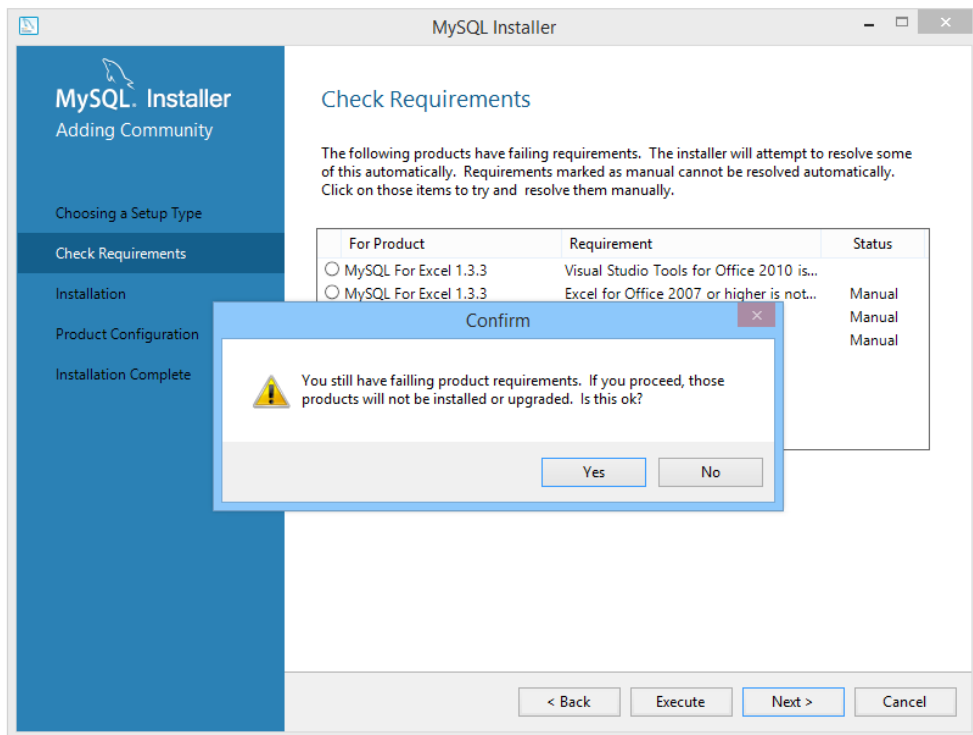
El asistente de instalación buscará en internet el instalador y lo ejecutará. Luego de aceptar los términos de la licencia, se da clic en “Install”.



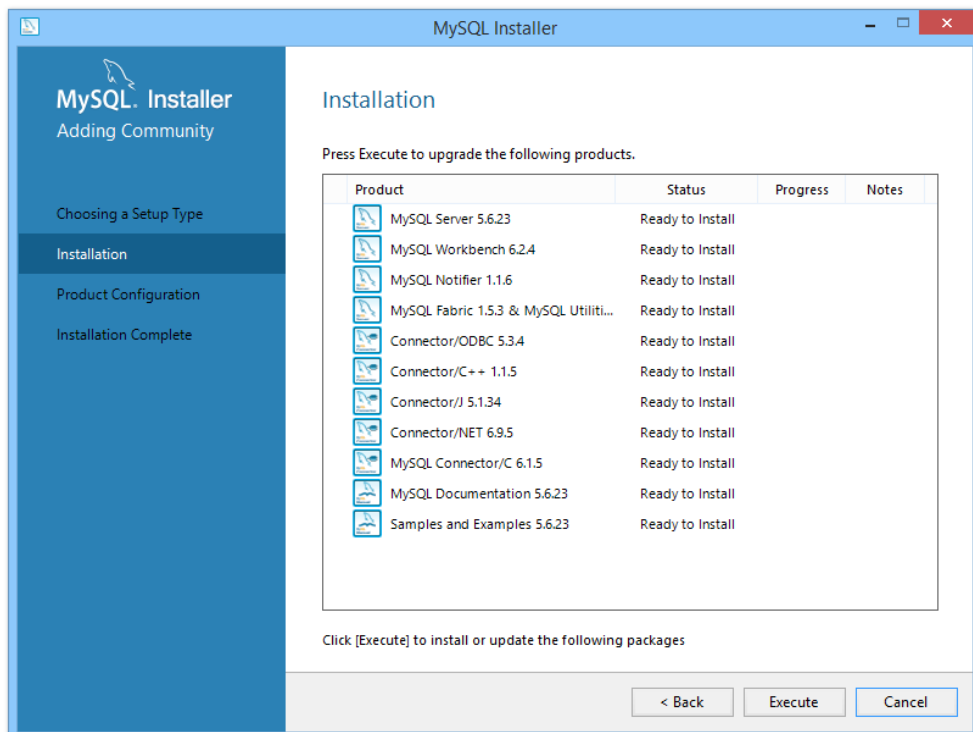
Cuando la instalación se complete aparecerá una ventana como la siguiente y luego de dar clic en “Close” esta se cerrará.



Si al regresar a la ventana de instalación de MySQL no se encuentra habilitado el botón “Next”, se debe cerrar la ventana y volver a repetir los pasos anteriores. Se podrá apreciar que MySQL Workbench ya no está entre la lista de componentes sin requerimientos no encontrados. Se debe dar clic en “Next >” y aparecerá un anuncio como el mostrado en la siguiente figura, y se procede a confirmar que no hay ningún inconveniente.



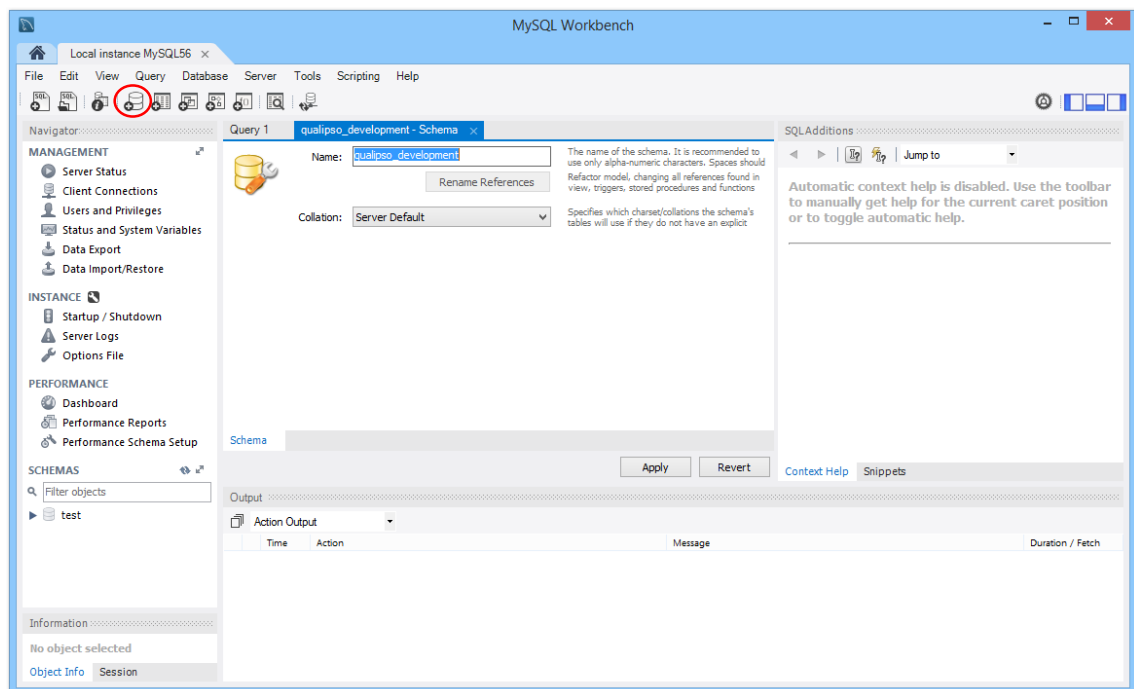
Finalmente iniciamos la instalación dando click el botón “Execute”.



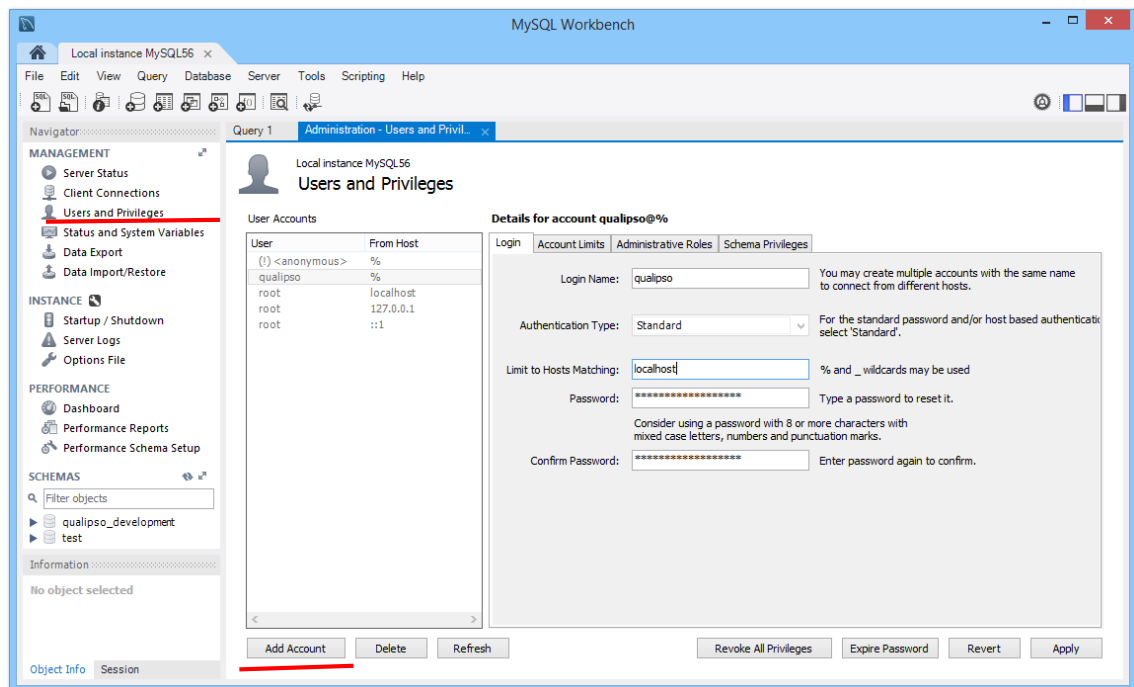
Crear Schema y Usuario

Para poder desplegar exitosamente el sistema es necesario tener conexión a una base de datos MySQL que tenga el schema y usuario declarados en el archivo ubicado en “C:\Sites\qualipso\config\database.yml”, esto se lo hace abriendo la herramienta MySQL Workbench 6.2 CE.

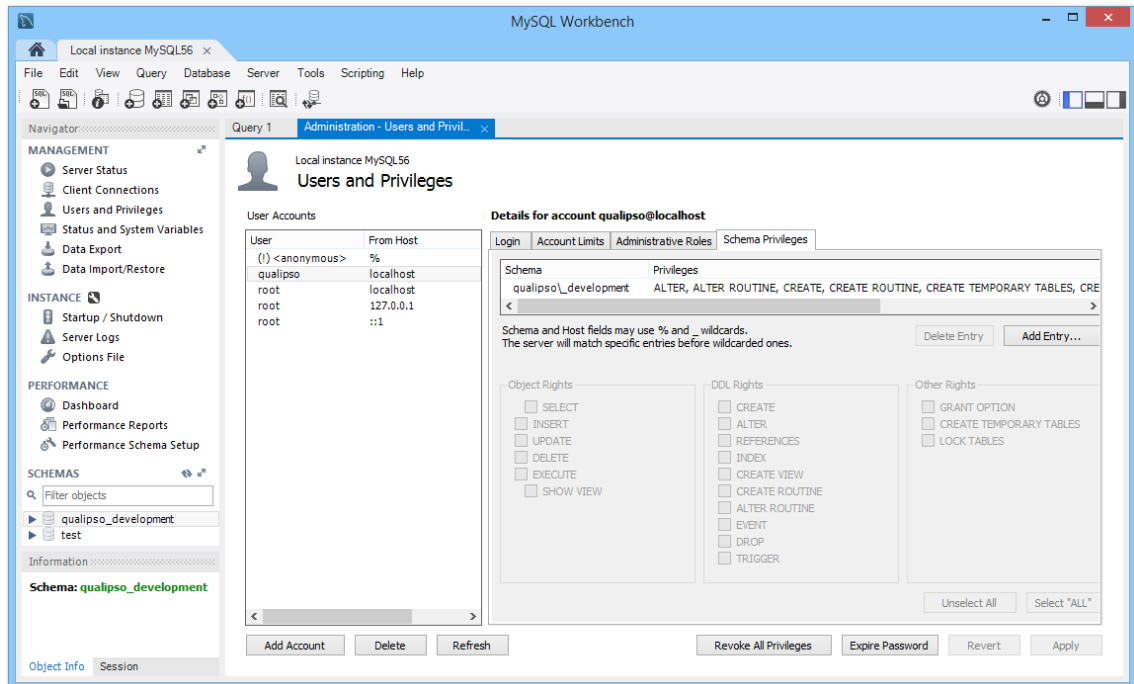
A continuación, se debe crear un schema llamado qualipso_development, luego de lo que cual se debe confirmar dando clic en “Apply”.



A continuación, se debe crear el usuario con nombre “qualipso” y contraseña “qualipso” en dos pasos: dentro de la pestaña Login y luego en la pestaña Schema Privileges.



En esta parte se debe dar clic en el botón “Add Entry...”. Se deberá seleccionar el schema “qualipso_development” para luego dar clic en el botón “Select All” y otorgarle al usuario todos los permisos sobre el schema. Con esto se termina de configurar la base de datos.



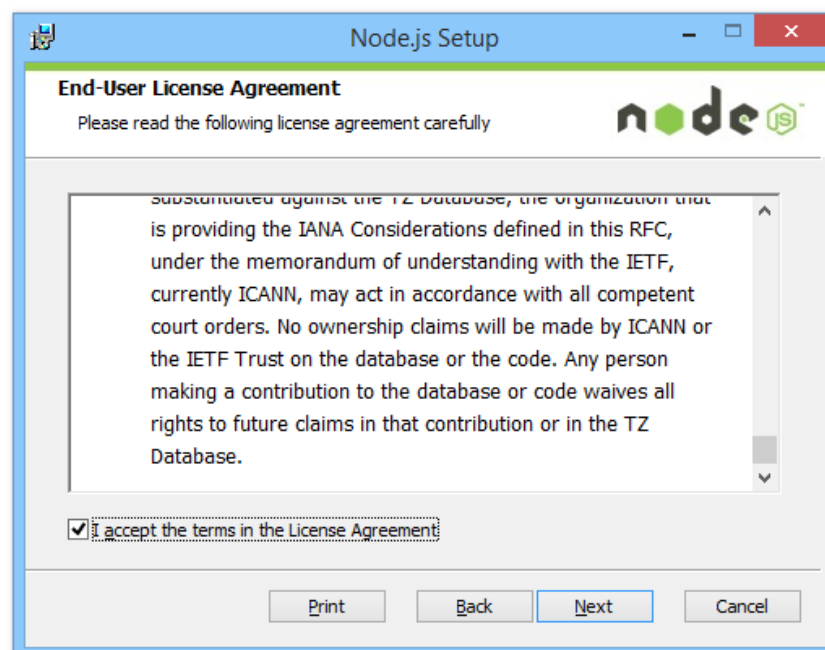
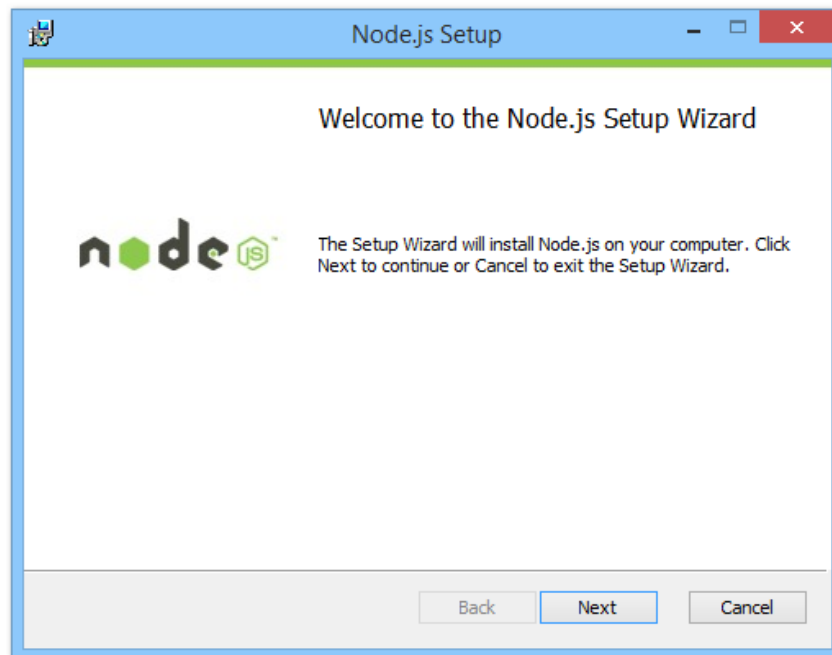
Instalar NodeJS

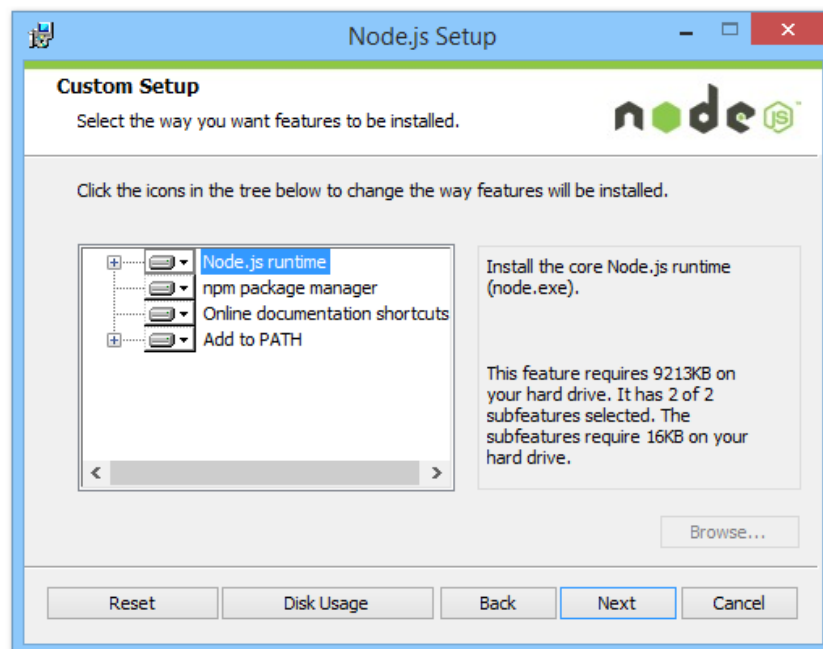
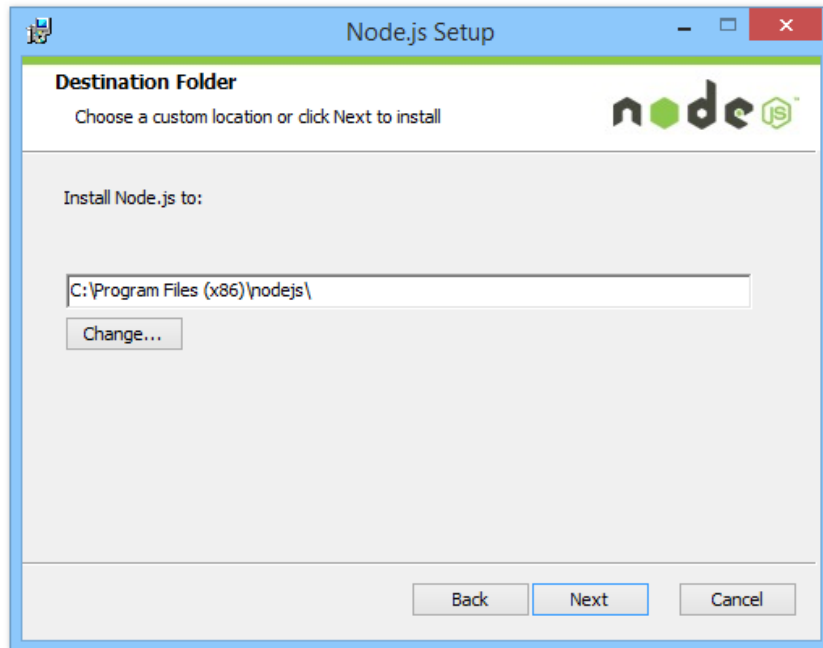
RoR trabaja con el motor Javascript V8 para realizar el procesamiento de scripts JS. Para lograr que este motor trabaje sin conflictos y dadas las pruebas realizadas se recomienda que se verifique que no se tenga la gema libv8 instalada e instalar NodeJS. Para verificarlo se puede ejecutar el comando “gem list” y revisar que no se encuentre en el listado; si está presente se la debe desinstalar usando el comando “gem uninstall libv8”.

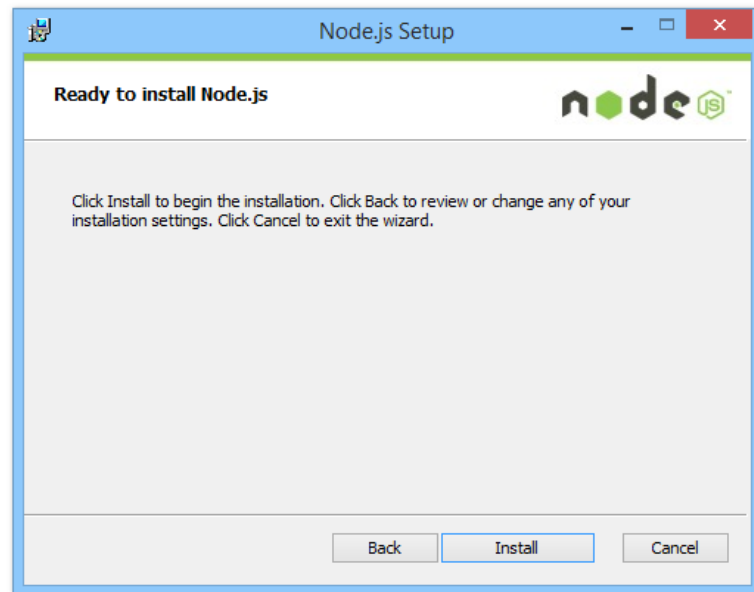
Dado que se ha eliminado la gema, se debe proveerle al sistema los binarios necesarios, para esto se debe instalar NodeJS, se sugiere la versión node-v0.12.2-x86.msi disponible en el siguiente link:

<https://nodejs.org/download/>

Una vez que se ha descargado, se procede a instalarlo manteniendo las opciones de instalación por defecto.







Cuando la instalación se complete, se debe reiniciar el equipo.

Instalar Redis

Redis es una base de datos de código abierto que se ejecuta en memoria dinámica y está basada en una estructura de tablas hash, es decir, en elementos del tipo clave / valor.

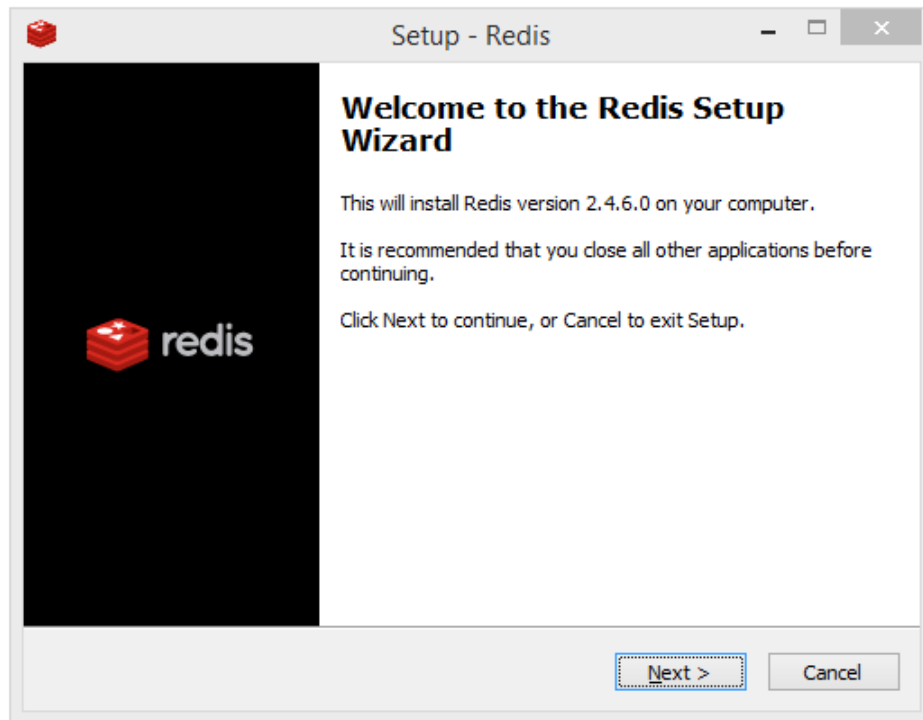
El hecho de que la base esté cargada en memoria la hace muy veloz en comparación a las bases de datos relacionales como Mysql o Postgresql. Esta característica hace que sea usada comúnmente como servidor de datos de caché. A pesar de que, como se ha dicho, la base está cargada en memoria, ofrece un mecanismo para persistir los datos en memoria de almacenamiento.

Redis está escrita en ANSI C, por lo tanto se ejecuta en sistemas basados en POSIX como Linux, OS X o BSD. Sin embargo, ha sido desarrollada y probada en sobre Linux y OS X y es recomendado desplegarlo sobre Linux. Si se lo desea usar sobre Windows hay que compilar el código fuente para obtener el ejecutable. Una alternativa menos complicada es utilizar recursos de terceros que faciliten su instalación puesto que no existe un soporte oficial para Windows.

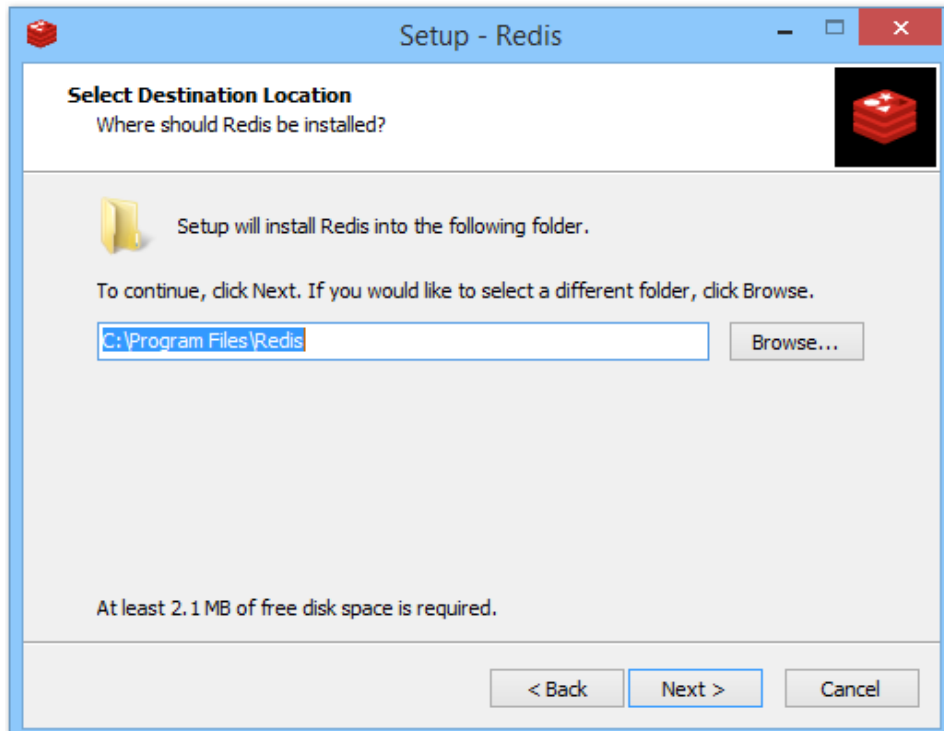
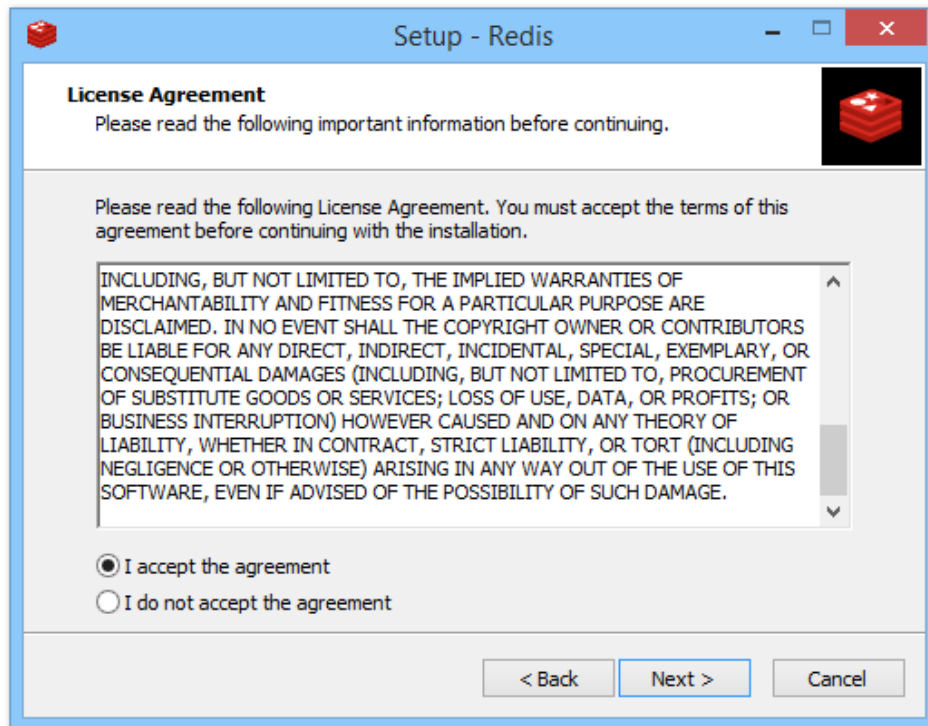
Sólo para fines de desarrollo es posible ejecutar este servidor sobre Windows. Para hacerlo se recomienda instalar Redis desde un “.exe” desarrollado como un proyecto abierto alojado en Github, descargando la versión más reciente para la arquitectura del sistema host. El enlace en donde es posible encontrar este ejecutable es el siguiente:

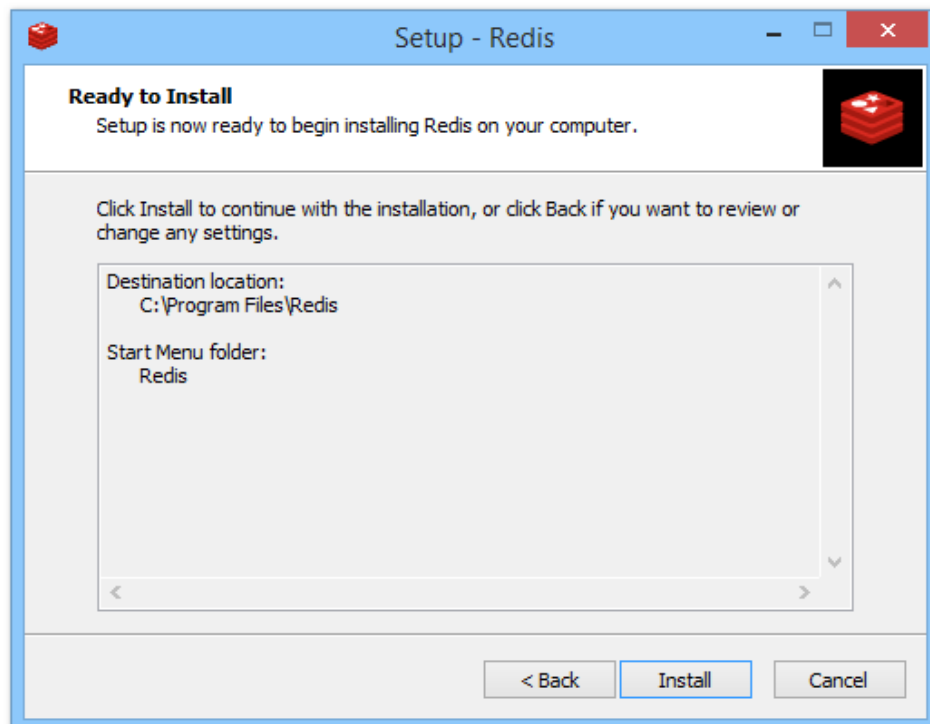
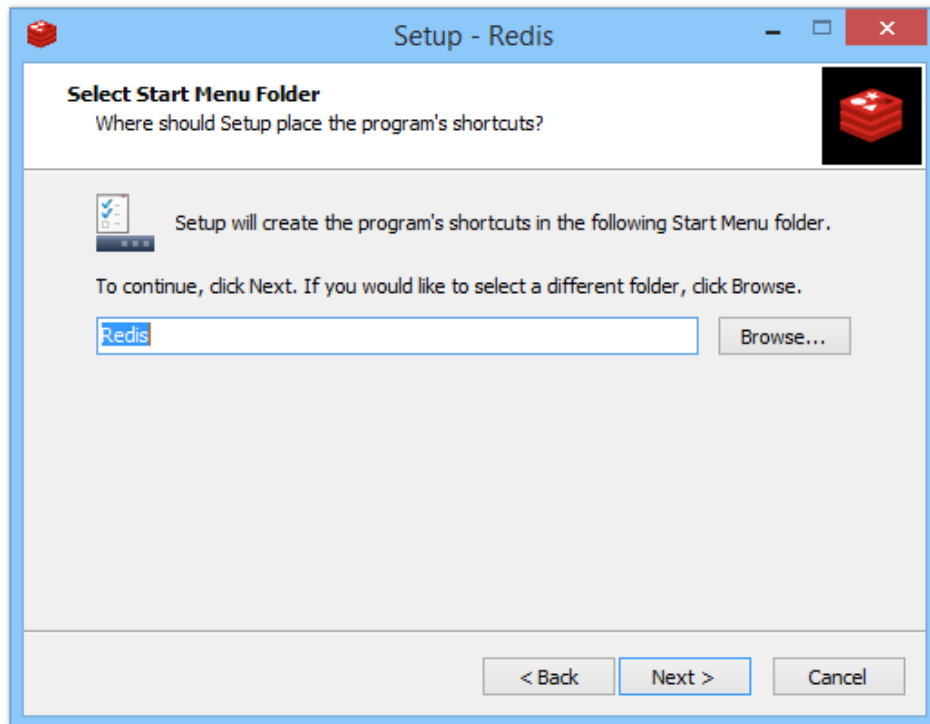
<https://github.com/rgl/redis/downloads>

Cuando se complete la descarga se debe ejecutar el archivo, en donde se abrirá una ventana como la mostrada.



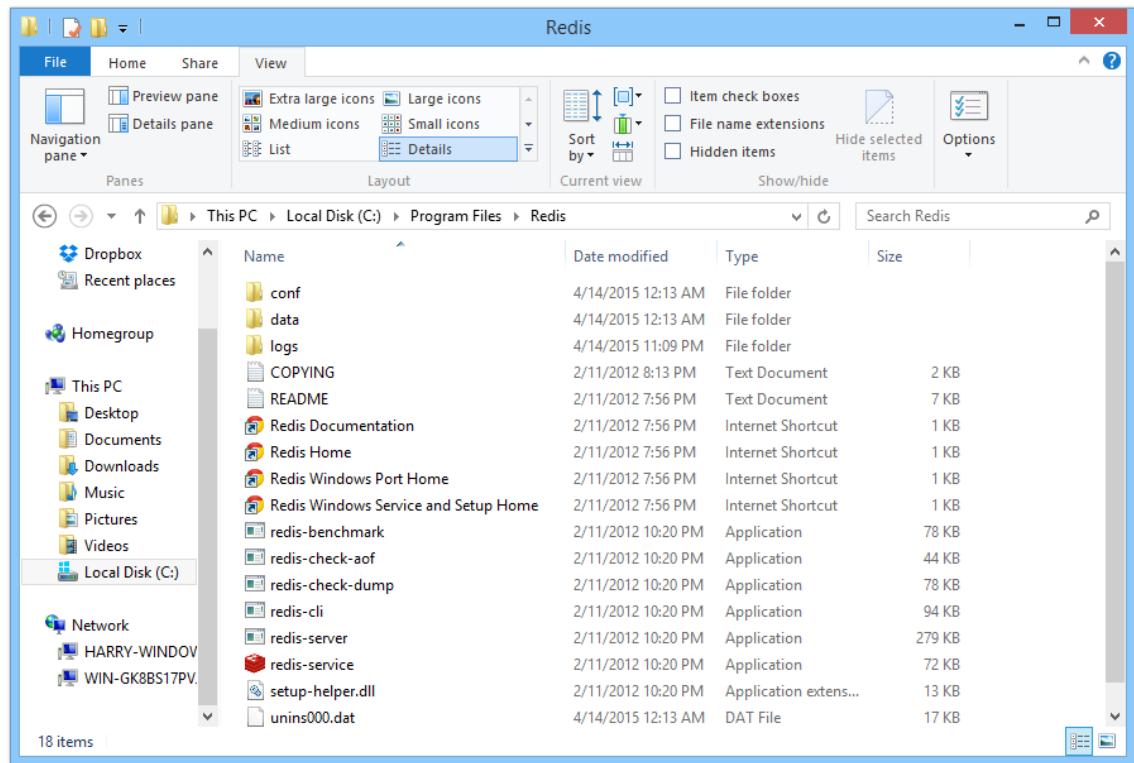
Este paquete facilita la instalación gracias a que permite dejar los parámetros de instalación por default y continuar presionando el botón "Next >" de cada ventana hasta llegar por último al botón "Install" que inicia la instalación. A continuación, se mostrará la secuencia de ventanas de este asistente:





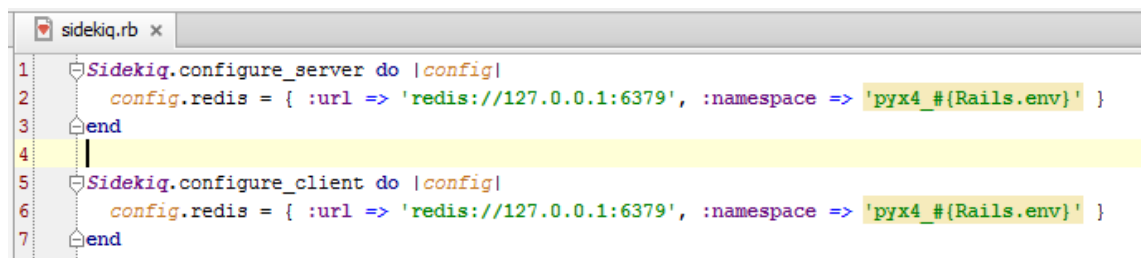


Al completar el proceso se tendrá instalado Redis el cual debería estar instalado como Servicio de Windows. Debe estar ejecutandose cuando se requiera utilizar el sistema PYX4. Si el servicio Windows de Redis tiene problemas para ejecutarse se recomienda iniciarlo como aplicación desde el archivo "redis-server.exe" ubicado en "C:\Program Files\Redis", el directorio deber verse como el mostrado en la siguiente imagen:



Cambiar los parámetros de conexión a Redis

PYX4 usa Redis para manejar datos en caché, para lograrlo Rails utiliza dos gemas llamadas redis y redis-namespace; estas gemas permiten armar una interfaz entre un servidor Redis y la aplicación RoR. Sin embargo, dado el soporte para IPV6 que implementan estas gemas es necesario asegurar que los parámetros de conexión ubicados en el archivo que se encuentra en el path “\qualipso\config\initializers\sidekiq.rb” se vean como en la siguiente imagen:



```
sidekiq.rb x
1 Sidekiq.configure_server do |config|
2   config.redis = { :url => 'redis://127.0.0.1:6379', :namespace => 'pyx4_#{Rails.env}' }
3 end
4
5 Sidekiq.configure_client do |config|
6   config.redis = { :url => 'redis://127.0.0.1:6379', :namespace => 'pyx4_#{Rails.env}' }
7 end
```

Lo relevante aquí es que en la url no se especifique la cadena “localhost” sino que en su lugar debe de escribirse la dirección IP loop-back, es decir, “127.0.0.1”.

Instalar ImageMagick

ImageMagick es un paquete de utilidades para edición, procesamiento y visualización de imágenes desde línea de comandos que soporta múltiples formatos. La herramienta provee un API que ha facilitado que pueda ser usada desde diferentes lenguajes por medio de una interface. El proyecto cuenta con una licencia Apache y es de código abierto.

El sistema PYX4 implementa ImageMagick y utiliza como interface una gema llamada “rmagick”. Para que esta gema pueda ser compilada e instalada se requieren que las librerías de ImageMagick estén disponibles en el sistema host, es posible descargarlas, compilarlas y configurar las variables de entorno correspondientes para que posteriormente la instalación de la gema se complete exitosamente.

En Windows, si no se desea descargar y compilar el código fuente, se cuentan con ejecutables “.exe” que facilitan el proceso de instalación, el cual consiste de manera general en dos pasos:

1. Instalar ImageMagick-6.7.7-9-Q16-windows-static.exe
2. Instalar ImageMagick-6.7.7-9-Q16-windows-dll.exe

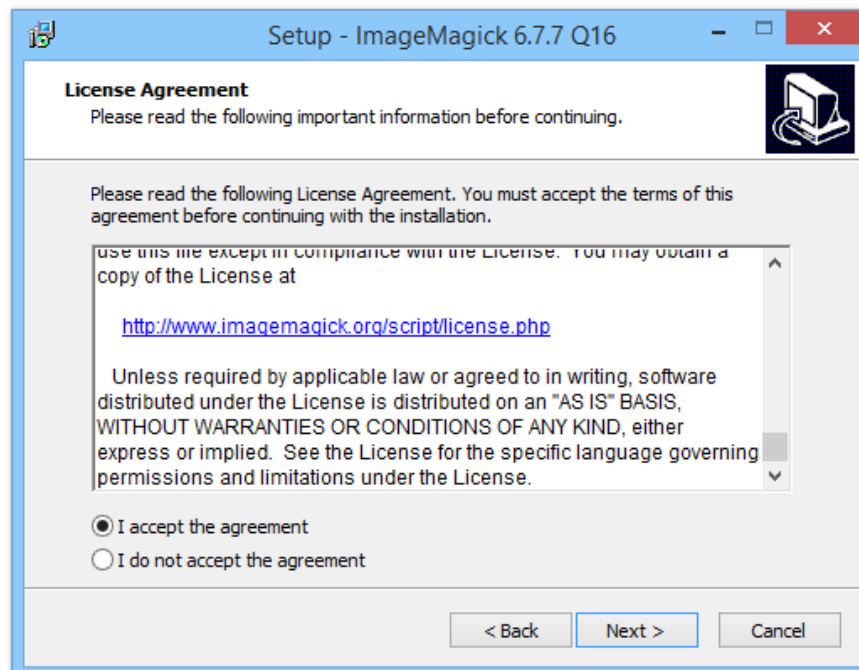
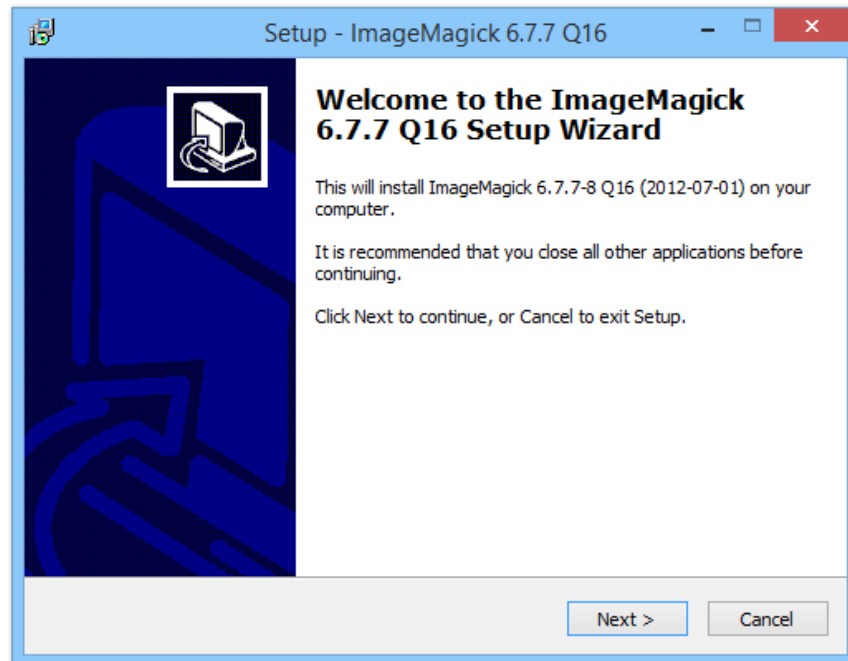
Se recomienda el uso de la versión 6.7.7-9-Q16 de ImageMagick, los archivos binarios se encuentran disponibles en el siguiente enlace:

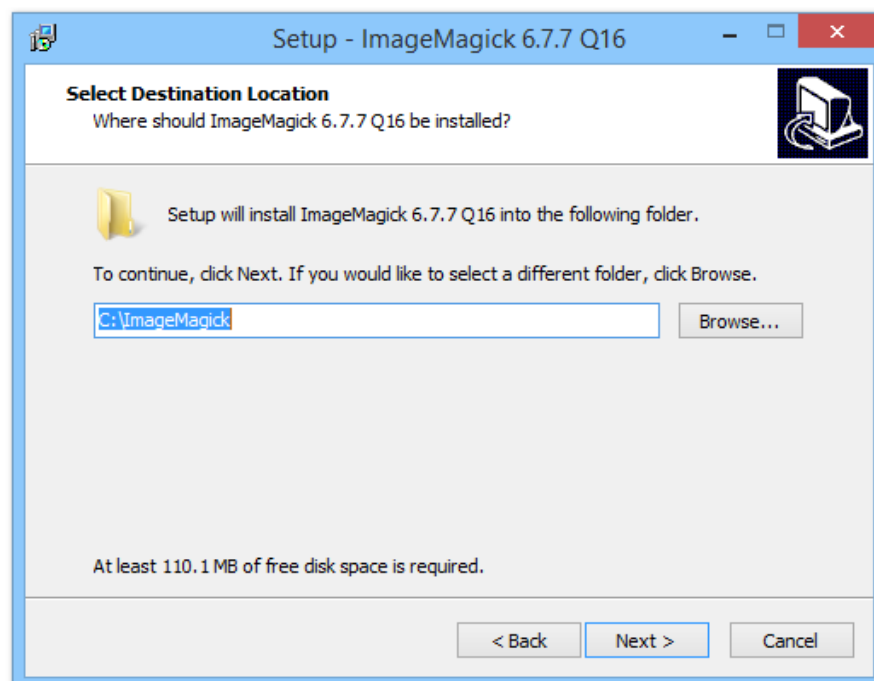
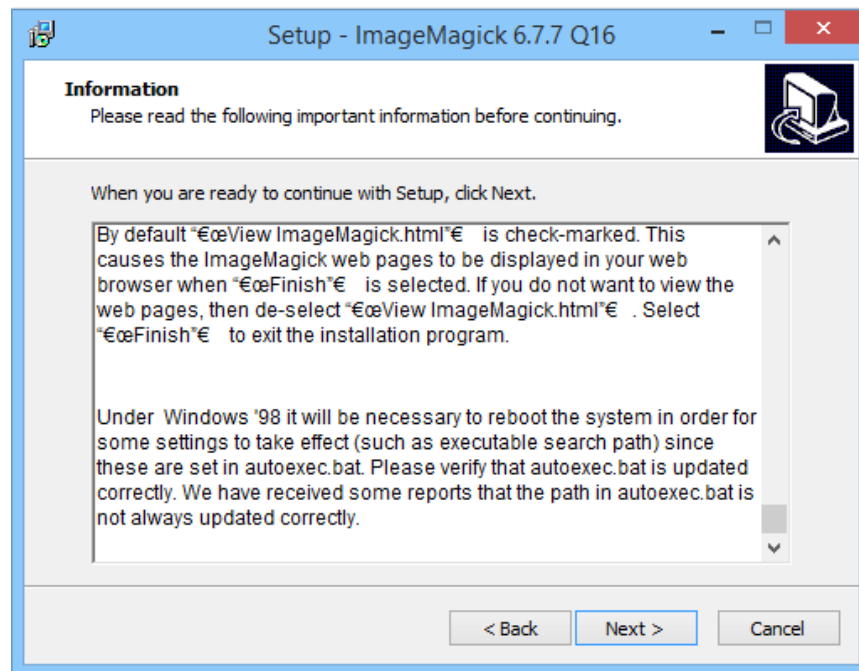
<http://ftp.sunet.se/pub/multimedia/graphics/ImageMagick/binaries/>

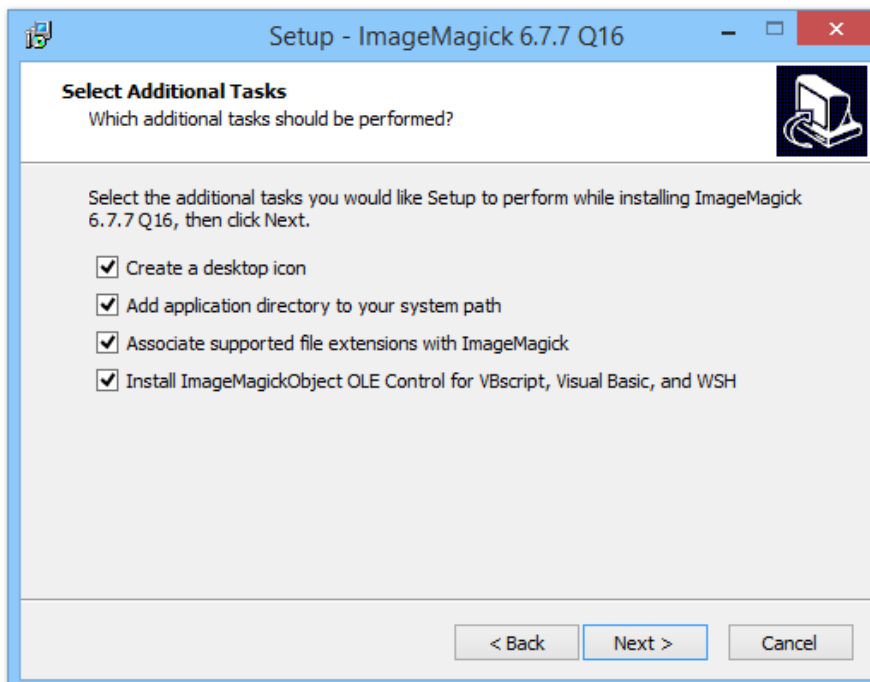
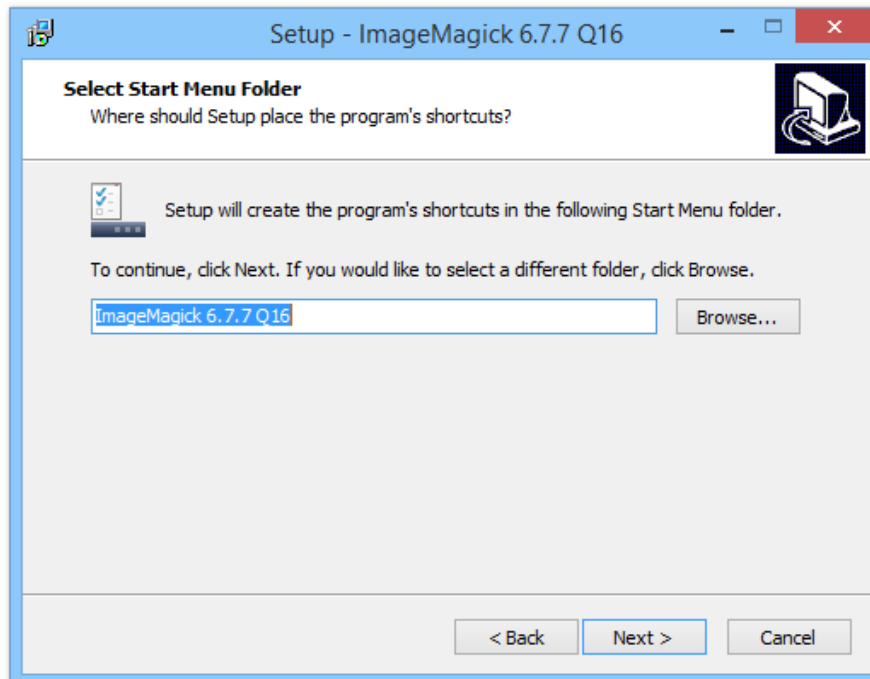
Instalación static

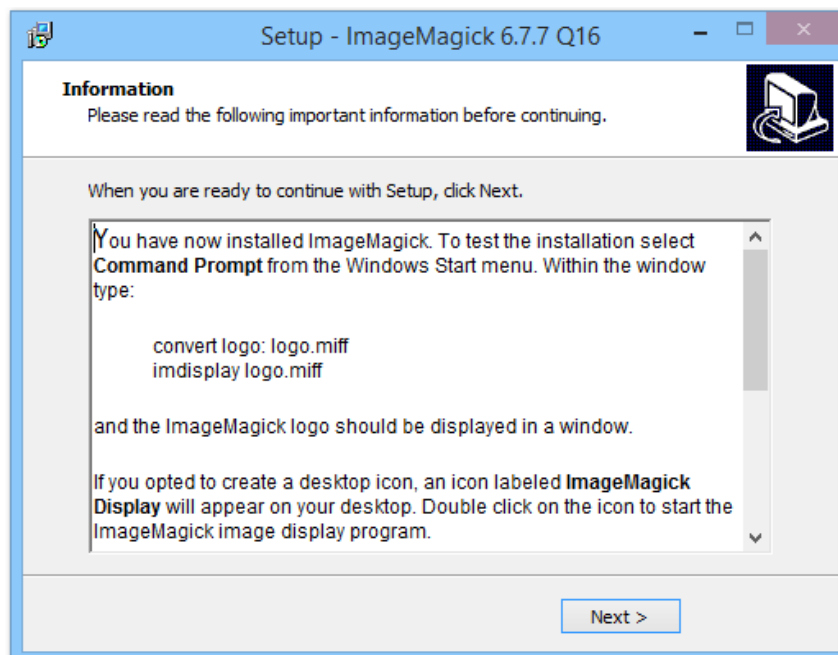
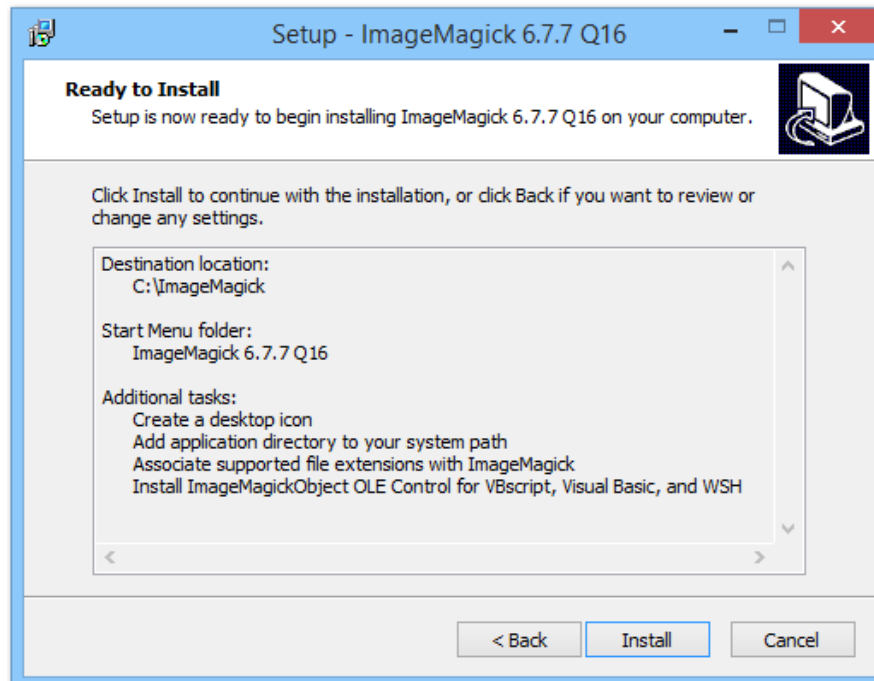
Al ejecutar el archivo ImageMagick-6.7.7-9-Q16-windows-static.exe aparecerá una serie de ventanas a las cuales en su mayoría se debe mantener su configuración por defecto y continuar con el asistente de instalación; es importante en que la ventana con los cuadros de verificación se seleccionen las mismas opciones que aquí se muestran.

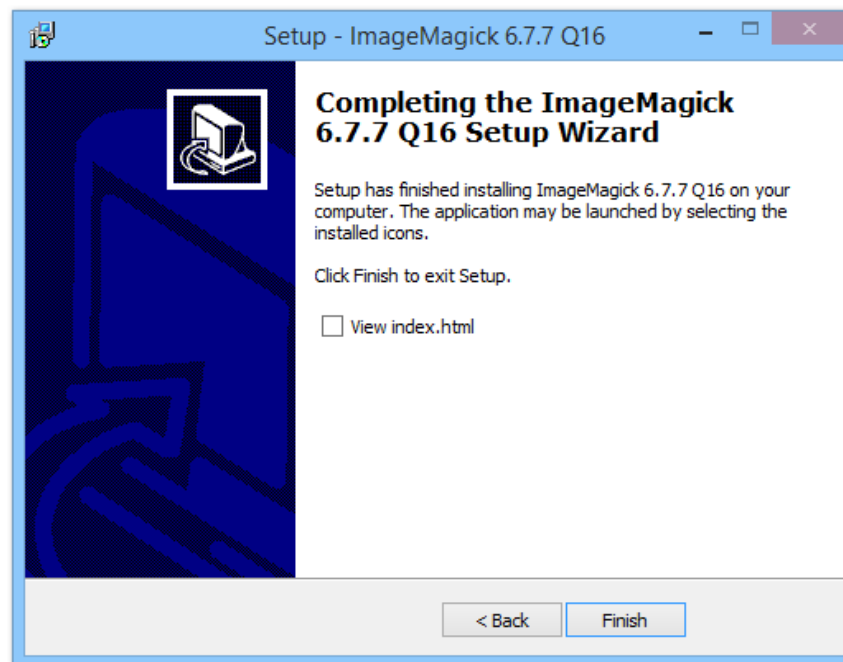
A continuación, se presentará la secuencia de ventanas del asistente de instalación antes de comenzar con el proceso:











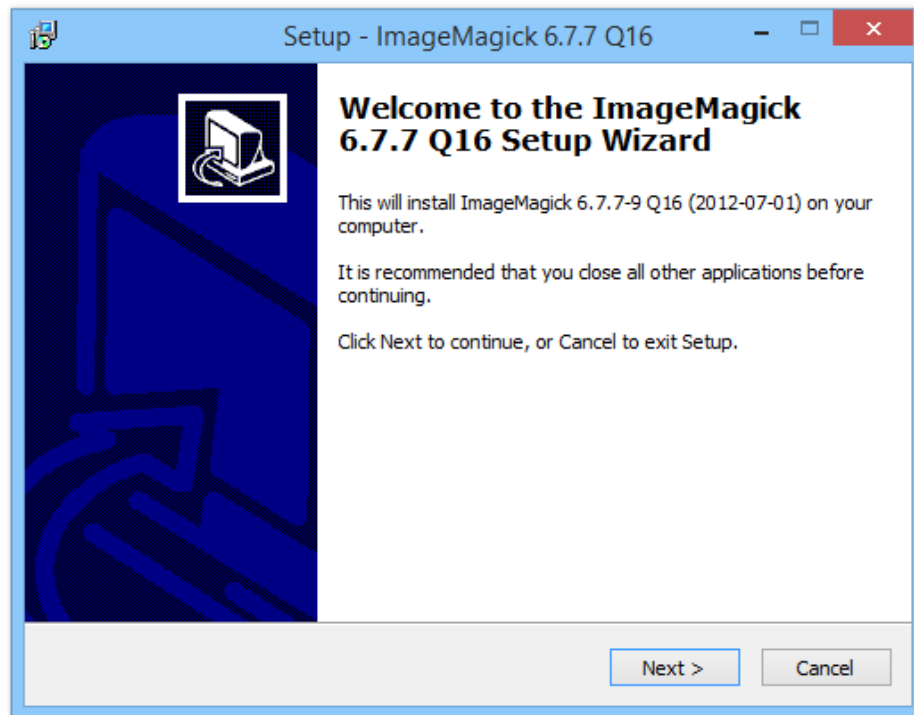
Al completarse la instalación se habrán configurado las variables de entorno e instalado los componentes para la ejecución del software ImageMagick.

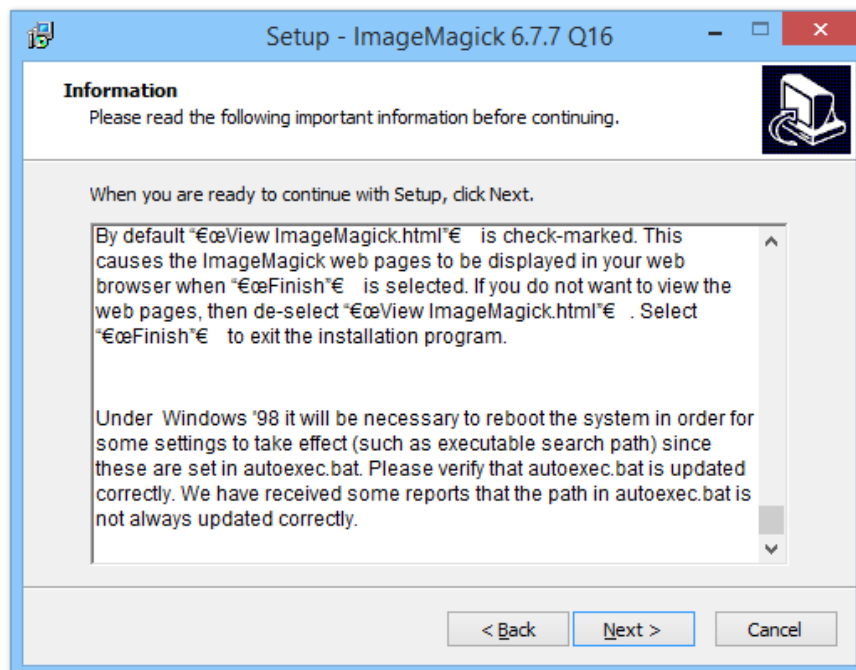
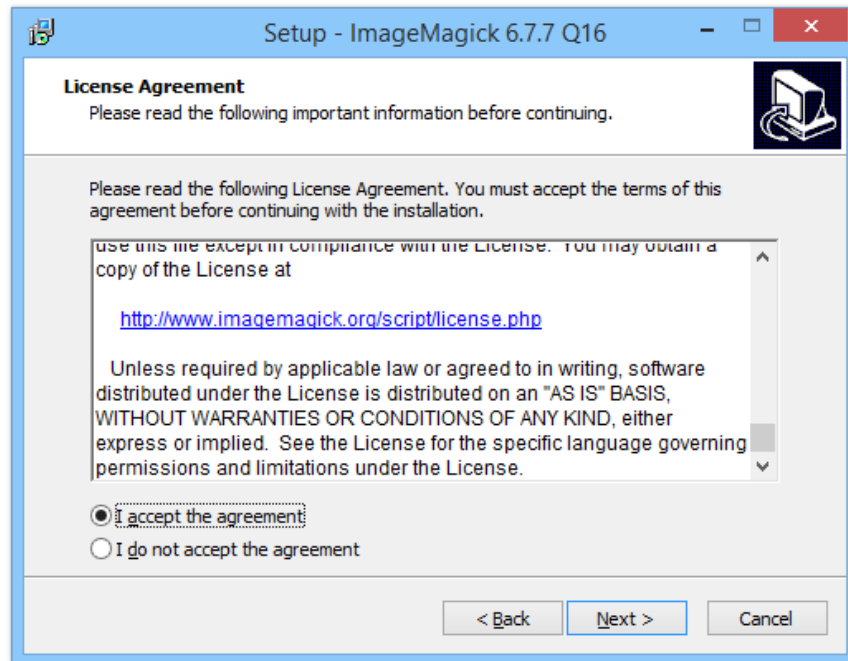
Instalación de dll

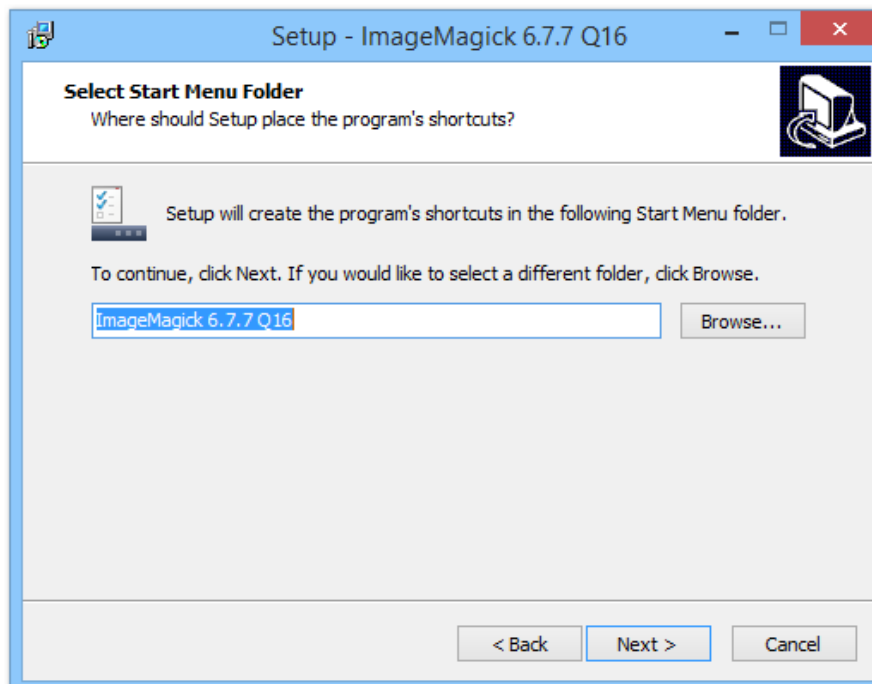
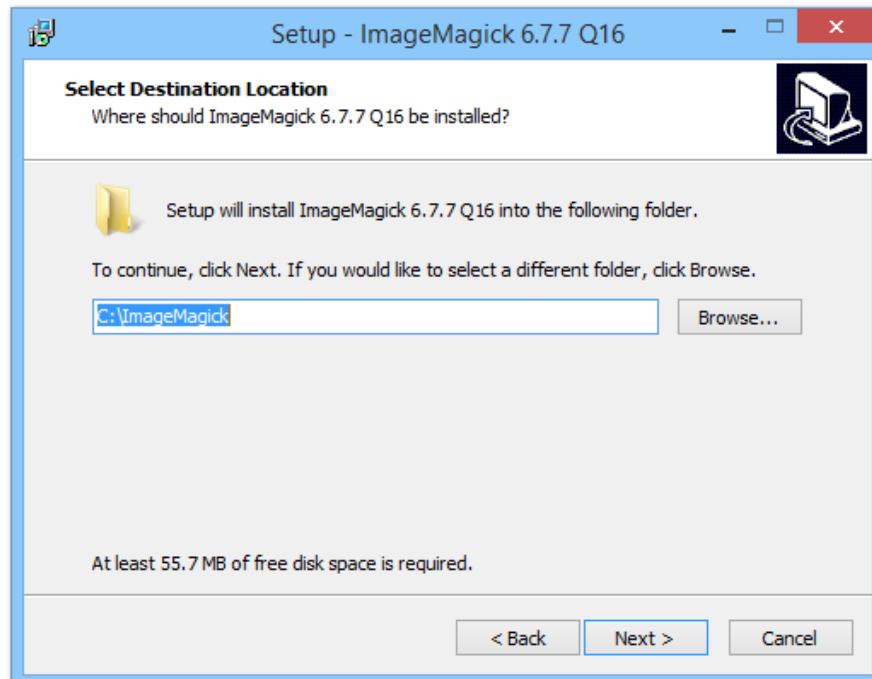
El siguiente paso es proporcionarle al sistema las librerías necesarias para compilar ImageMagick. Esto es necesario porque posteriormente la gema las requerirá para instalarse.

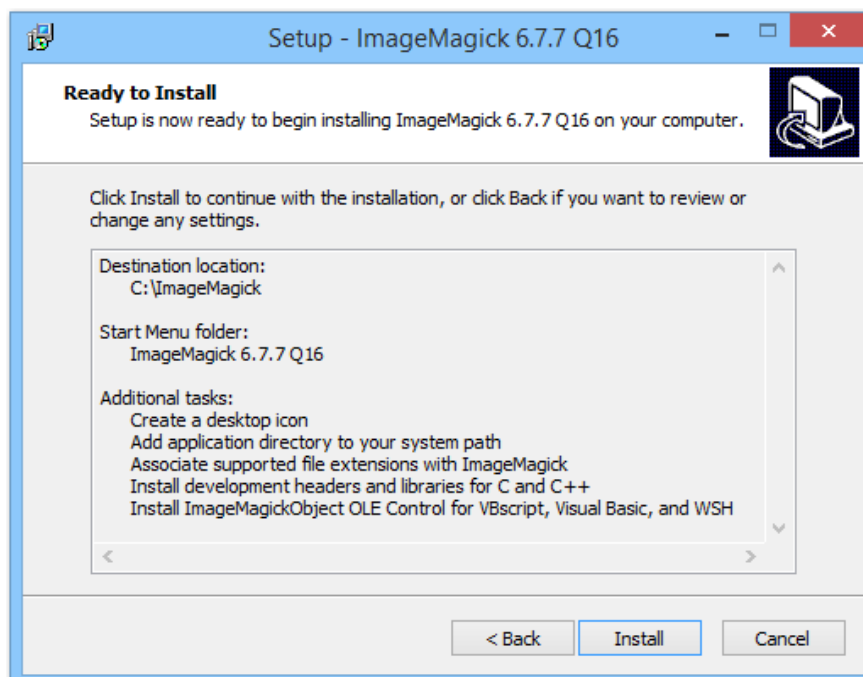
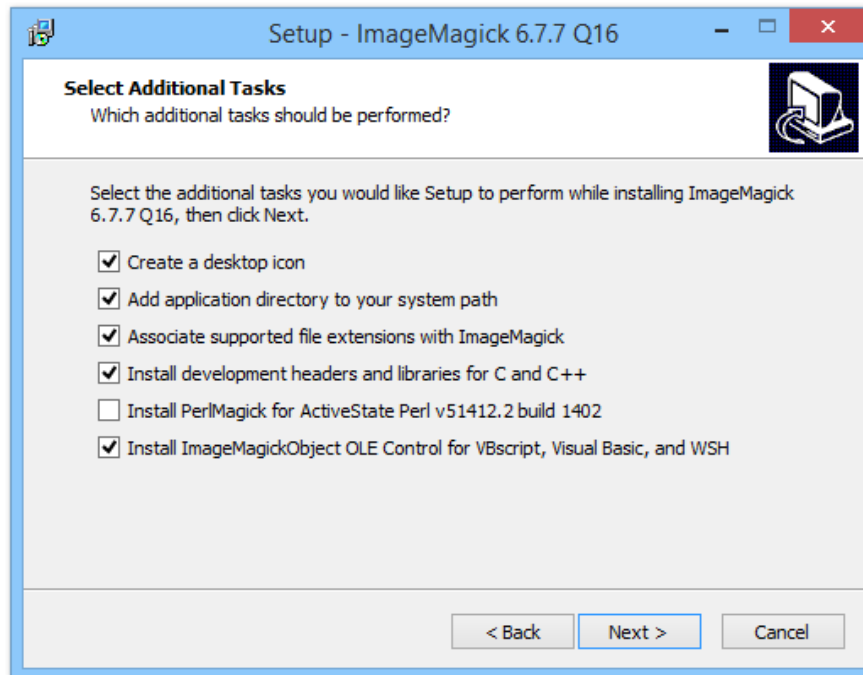
Las ventanas del asistente de instalación son muy similares a las del paso anterior. Aquí nuevamente habrá que poner mayor atención al momento de seleccionar las opciones de la ventana con los cuadros de verificación. El archivo a ejecutar es "ImageMagick-6.7.7-9-Q16-windows-dll.exe".

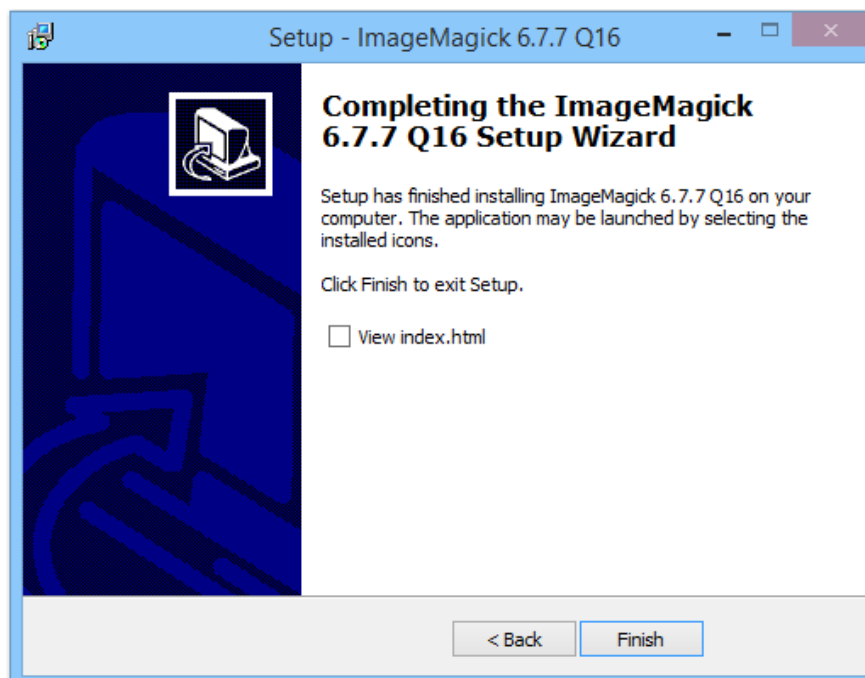
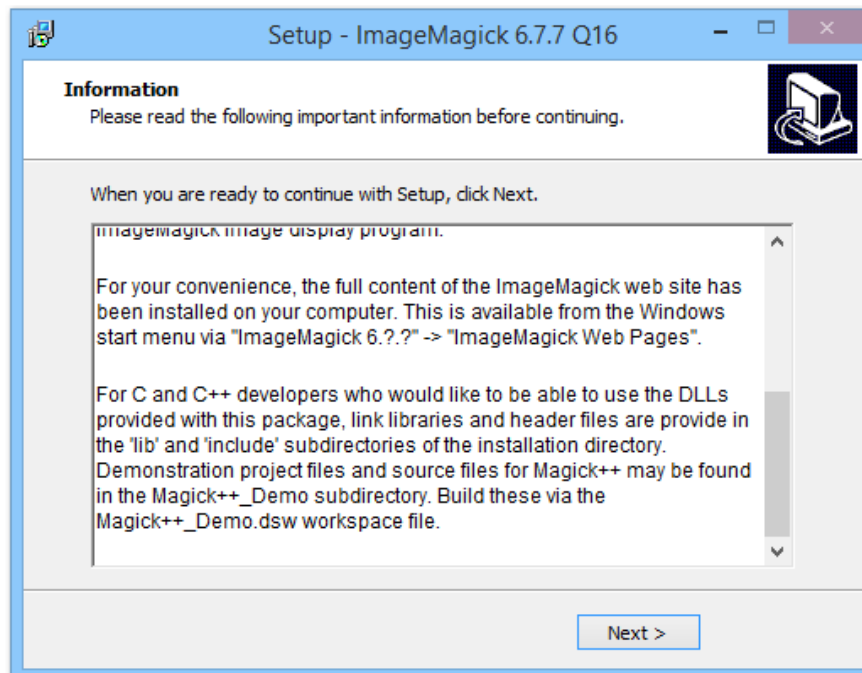
A continuación, se presentarán las capturas de la secuencia de ventanas que aparecen durante la instalación, poniendo atención a que las opciones de cada una sean las mismas y continuar con el proceso.



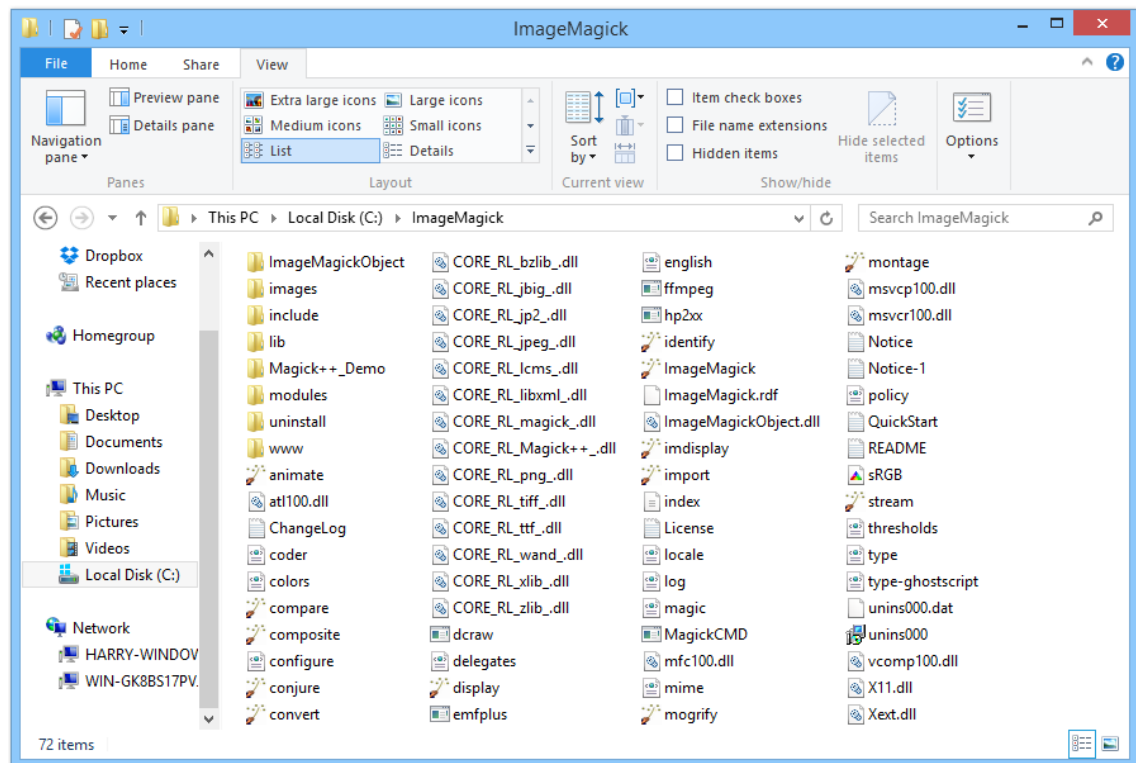






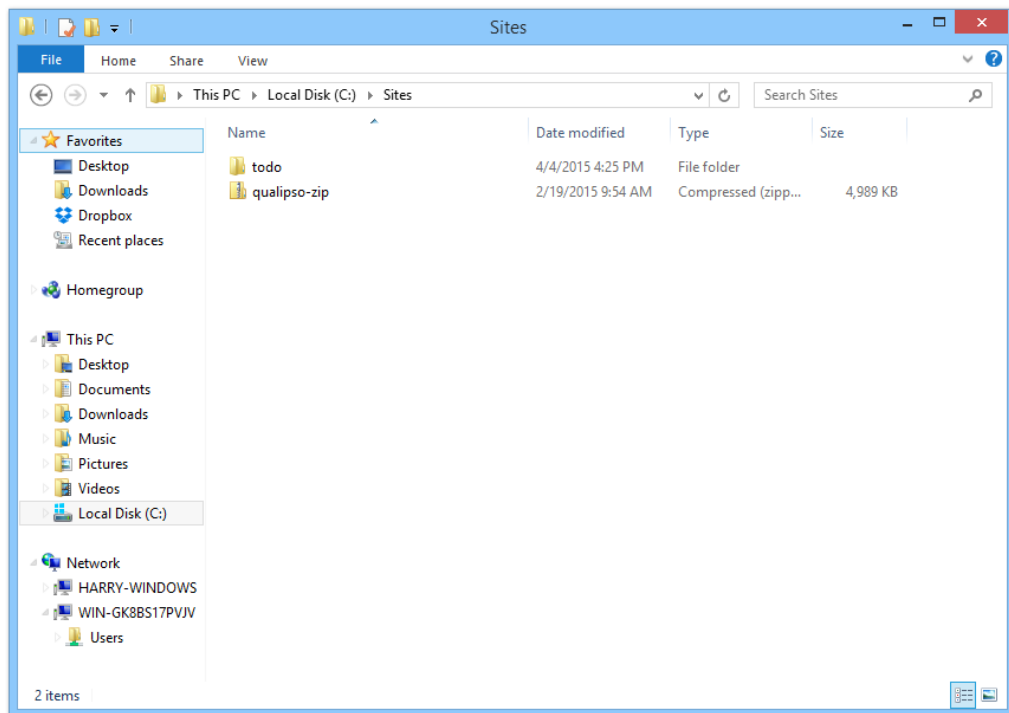


Finalmente, se deberá tener un directorio como el mostrado en la siguiente imagen:

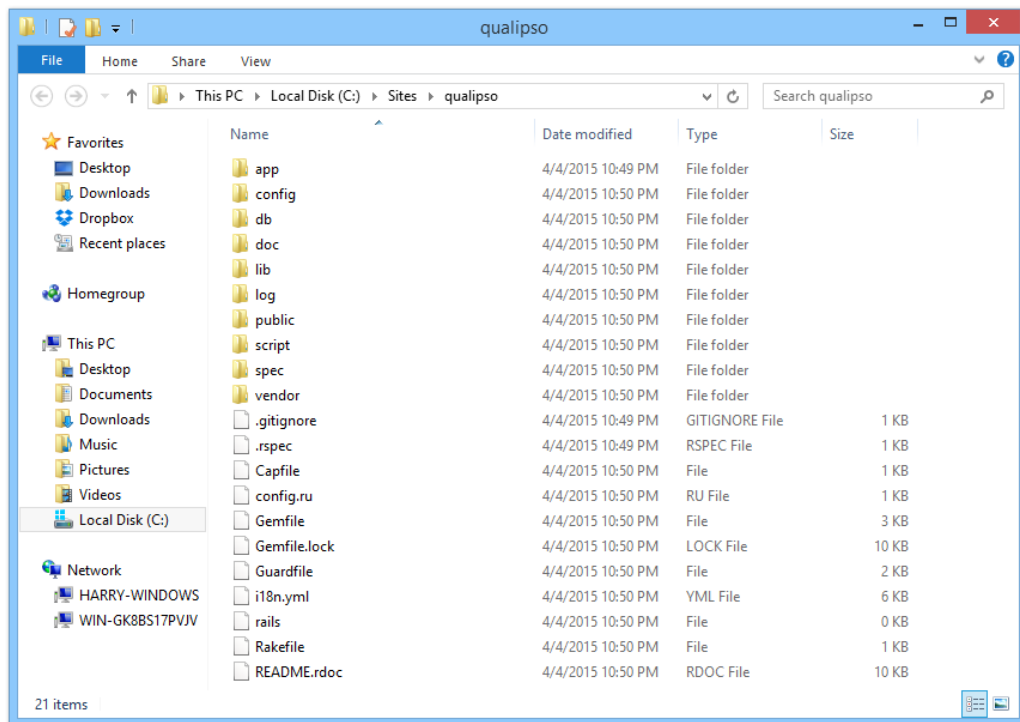


Despliegue de PYX4

Ahora se usará la carpeta Sites que se mencionó antes, colocando el archivo zip “qualipso-zip” dentro de Sites



Aquí dentro se lo debe descomprimir de tal forma que quede el directorio “C:\Sites\qualipso”.



Luego se debe abrir el archivo Gemfile con un editor de texto, se recomienda notepad++ o similar donde se puedan ver resaltadas las palabras reservadas del lenguaje. En el archivo se procede a comentar la línea 28, los comentarios en Ruby se hacen con un “#” al iniciar la línea a comentar. Luego se deben guardar los cambios.


```

1 source 'https://rubygems.org'
2
3 gem 'rails', '3.2.12'
4
5 # Bundle edge Rails instead:
6 # gem 'rails', :git => 'git://github.com/rails/rails.git'
7
8 gem 'mysql2'
9
10
11 # Gems used only for assets and not required
12 # in production environments by default.
13 group :assets do
14   gem 'sass-rails', '~> 3.2.3'
15   gem 'coffee-rails', '~> 3.2.1'
16   gem "spine-rails"
17   gem 'eco'
18   #gem 'stylus'
19   # See https://github.com/sstephenson/execjs#readme for more supported runtimes
20   # gem 'therubyracer', :platforms => :ruby
21
22   gem 'uglifier', '>= 1.0.3'
23 end
24
25 gem 'jquery-rails'
26 gem 'jquery-ui-rails'
27 gem 'jquery-scrollto-rails'
28 #gem "therubyracer", ">= 0.8.2"
29
30 # To use ActiveModel has_secure_password
31 # gem 'bcrypt-ruby', '~> 3.0.0'

```

Luego se debe ejecutar el archivo que se encuentra en “C:\RailsInstaller\DevKit\devkitbars.bat”. La ejecución se la realiza dando clic sobre dicho archivo. Se abrirá brevemente una consola y se cerrará; este proceso configura variables de ejecución de DevKit.

A continuación se debe ejecutar el archivo ubicado en “C:\RailsInstaller\DevKit\msys.bat”; se abrirá una consola de comandos. En esta consola se debe escribir:

```
cd "C:\Sites\qualipso"
```

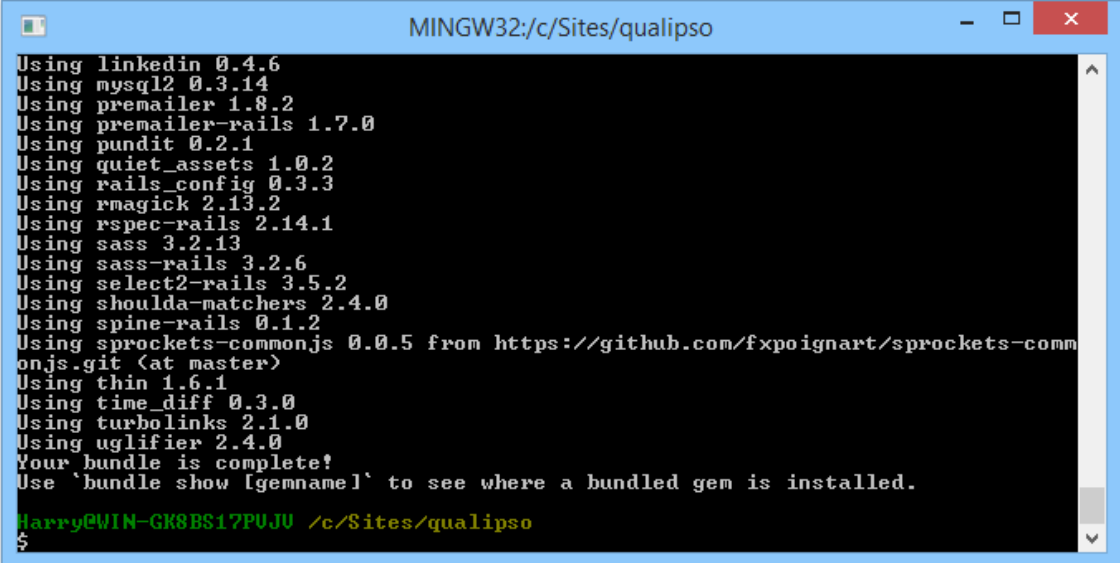
Esto ubicará la consola en la carpeta del sistema PYX4. Luego de debe ejecutar el comando “bundle install” para instalar las gemas y sus dependencias declaradas en el archivo Gemfile.



```
MINGW32:/c/Sites/qualipso
Harry@WIN-GK8BS17PUJU ~
$ cd "C:\Sites\qualipso"
Harry@WIN-GK8BS17PUJU /c/Sites/qualipso
$ bundle install
```

Al ejecutar el comando anterior se podrían presentarse una serie de problemas debido a que se está desplegando un software con versiones de librerías legadas. En la sección de Problemas Frecuentes se han resuelto algunos de estos.

Si la ejecución del comando “bundle install” fue exitosa, se tendrá una ventana como la siguiente:



```
MINGW32:/c/Sites/qualipso
Using linkedin 0.4.6
Using mysql2 0.3.14
Using premailer 1.8.2
Using premailer-rails 1.7.0
Using pundit 0.2.1
Using quiet_assets 1.0.2
Using rails_config 0.3.3
Using rmagick 2.13.2
Using rspec-rails 2.14.1
Using sass 3.2.13
Using sass-rails 3.2.6
Using select2-rails 3.5.2
Using shoulda-matchers 2.4.0
Using spine-rails 0.1.2
Using sprockets-commonjs 0.0.5 from https://github.com/fxpoignart/sprockets-commonjs.git (at master)
Using thin 1.6.1
Using time_diff 0.3.0
Using turbolinks 2.1.0
Using uglifier 2.4.0
Your bundle is complete!
Use 'bundle show [gemname]' to see where a bundled gem is installed.
Harry@WIN-GK8BS17PUJU /c/Sites/qualipso
$
```

Migrar modelos a schema

Rails permite migrar desde modelos escritos en Ruby hacia tablas de un schema. Aquí es importante que exista el schema y el usuario declarados en “database.yml”, creados anteriormente. Para realizar la migración, se debe ejecutar el comando “rake db:migrate”.

```

MINGW32:/c/Sites/qualipso
Harry@WIN-GK8BS17PUJU /c/Sites/qualipso
$ rake db:migrate

```

Crear registros iniciales

Para crear los registros que necesita PYX4 para poder ejecutarse por primera vez, se debe de ejecutar el siguiente script:

```

BEGIN;
INSERT INTO `customers` VALUES ('1', '127.0.0.1', '2015-02-27 22:12:01', '2015-02-27 22:12:01', '1', '0', 'en', null, null, '0');
INSERT INTO `users` VALUES ('1', 'developer@espol.edu.ec', 'Harry', 'Carpio', '2015-02-28 12:10:06', '2015-04-05 22:01:39', '$2a$10$W0uTILTtX4CDWitQ1GrRXe5SbMnooySvHuz87PkSQkFnsAhz2q1ya', null, null, 'PUZ69yVWfgcTaRJ2NP4u', null, null, '8', '2015-04-05 22:01:39', '2015-03-19 00:50:02', '127.0.0.1', '127.0.0.1', '1', null, null, null, '0', null, null, null, null, null, null, null, 'man', null, null, null, null, '0', 'admin', 'en', null);
INSERT INTO `contributions` VALUES ('1', 'Hola, prueba comment.', '1', 'Graph', '1', '2015-02-28 22:30:25', '2015-02-28 22:30:25');
INSERT INTO `arrows` VALUES ('1', '1', '1', '3', '394.1667', '118.9984', '100.0000', '25.0000', 'Basket1', '2015-02-28 19:49:16', '2015-02-28 20:12:33', 'straight', '0', null, 'none', null, null, null, '14', '#ffffff', 'top', 'bottom', '1', '#000000', 'Arial'), ('2', '1', '3', '2', '660.8333', '280.5012', '100.0000', '25.0000', 'Basket2', '2015-02-28 19:49:16', '2015-02-28 20:12:33', 'straight', '0', null, 'none', null, null, null, '14', '#ffffff', 'top', 'bottom', '1', '#000000', 'Arial');
INSERT INTO `directories` VALUES ('1', 'dir_test', null, '1', '4', '2015-02-28 13:51:35', '2015-02-28 13:51:35', '1', '1'), ('3', 'prueba', '1', '2', '3', '2015-04-13 20:47:50', '2015-04-13 20:47:50', '1', '1');
INSERT INTO `documents_publishers` VALUES ('1', '1', '1', null, null, '2015-02-28 23:12:15', '2015-02-28 23:12:15'), ('2', '2', '1', null, null, '2015-03-01 00:14:15', '2015-03-01 00:14:15');
INSERT INTO `documents` VALUES ('1', 'DocumentHarryTest', '', 'FirstTests', '1.0', 'docx', '2015-02-28 23:04:37', '2015-02-28 23:13:48', '1', '1', 'Plantilla.docx', '1', null, 'approved', null, '0', null, '1', null), ('2', 'SecondTest', '', 'Segunda prueba', '1.0', 'docx', '2015-03-01 00:00:52', '2015-03-01 00:15:22', '1', '1', 'documento.docx', '1', null, 'approved', null, '0', null, '2', null);
INSERT INTO `documents_approvers` VALUES ('1', '1', '1', '1', 'Aprobado', '0', '2015-02-28 23:12:14', '2015-02-28 23:13:48'), ('2', '2', '1', '1', 'se envia a aprobar el documento de prueba 2', '0', '2015-03-01 00:14:14', '2015-03-01 00:15:22');
INSERT INTO `documents_logs` VALUES ('1', '1', 'created', null, '1', '2015-02-28 23:04:37', '2015-02-28 23:04:37'), ('2', '1', 'wf_started', null, '1', '2015-02-28 23:12:21', '2015-02-28 23:12:21'), ('3', '1', 'verified_by', 'Ok, aqui verificando', '1', '2015-02-28 23:12:47', '2015-02-28 23:12:47'), ('4', '1', 'verified', null, null, '2015-02-28 23:12:47', '2015-02-28 23:12:47'), ('5', '1', 'approved_by', 'Aprobado', '1', '2015-02-28 23:13:48', '2015-02-28 23:13:48'), ('6', '1', 'approved', null, null, '2015-02-28

```

```

23:13:48', '2015-02-28 23:13:48'), ('7', '2', 'created', null, '1', '2015-03-01 00:00:52', '2015-03-01 00:00:52'), ('8', '2',
'wf_started', null, '1', '2015-03-01 00:14:19', '2015-03-01 00:14:19'), ('9', '2', 'verified_by', 'verificando segundo documento
de prueba', '1', '2015-03-01 00:14:49', '2015-03-01 00:14:49'), ('10', '2', 'verified', null, null, '2015-03-01 00:14:49', '2015-03-
01 00:14:49'), ('11', '2', 'approved_by', 'se envia a aprobar el documento de prueba 2', '1', '2015-03-01 00:15:22', '2015-03-
01 00:15:22'), ('12', '2', 'approved', null, null, '2015-03-01 00:15:22', '2015-03-01 00:15:22');
INSERT INTO `documents_verifiers` VALUES ('1', '1', '1', '1', 'Ok, aqui verificando', '0', '2015-02-28 23:12:12', '2015-02-28
23:12:47'), ('2', '2', '1', '1', 'verificando segundo documento de prueba', '0', '2015-03-01 00:14:13', '2015-03-01 00:14:49');
INSERT INTO `documents_viewers` VALUES ('2', '1', '1', 'User'), ('3', '2', '1', 'User');
INSERT INTO `elements` VALUES ('1', '1', 'graphstart', null, '142.5000', '120.0000', '70.0000', '18.0000', 'Start', '2015-02-28
19:49:15', '2015-02-28 19:49:15', 'graphstart', null, null, null, null, '#ffffff', null, '100', 'middle', '0', '0', '0', null, null,
'Arial', null), ('2', '1', 'graphend', null, '675.8333', '428.0000', '70.0000', '18.0000', 'End', '2015-02-28 19:49:16', '2015-02-28
19:49:16', 'graphend', null, null, null, null, '#ffffff', null, '101', 'middle', '0', '0', '0', null, null, 'Arial', null), ('3', '1',
'instruction', null, '613.3333', '120.0000', '195.0000', '43.0000', 'Actividad 1', '2015-02-28 19:49:16', '2015-02-28 20:12:32',
'instruction', null, null, null, null, '#ffffff', 'none', '102', 'middle', '0', '0', '0', null, null, 'Arial', null);
INSERT INTO `flags` VALUES ('1', '1', '1', '2015-02-28 14:10:16', '2015-02-28 14:10:21', '1');
INSERT INTO `graph_publishers` VALUES ('1', '1', '1', null, null, '2015-02-28 20:03:21', '2015-02-28 20:03:21');
INSERT INTO `graphs` VALUES ('1', 'Test', 'process', '2', 'new', 'First graph creation test', null, '2015-02-28 19:40:39', '2015-
02-28 19:40:39', '0', '1', '1', '1', '1', null, null, '1', '0'), ('2', 'graph prueba', 'process', '2', 'new', 'referencia', null,
'2015-03-07 05:46:10', '2015-03-07 05:46:10', '1', '0', '1', null, '1', '1', '1', '1', null, null, '2', '0');
INSERT INTO `graphs_approvers` VALUES ('2', '1', '1', '0', '2015-02-28 20:03:29', '2015-02-28 20:03:29', null, '0');
INSERT INTO `graphs_logs` VALUES ('1', '1', 'created', null, '1', '2015-02-28 19:40:39', '2015-02-28 19:40:39'), ('2', '2',
'created', null, '1', '2015-03-07 05:46:10', '2015-03-07 05:46:10');
INSERT INTO `graphs_verifiers` VALUES ('1', '1', '1', '0', '2015-02-28 20:03:27', '2015-02-28 20:03:27', null, '0');
INSERT INTO `graphs_viewers` VALUES ('1', '1', 'User', '1');
INSERT INTO `groupdocuments` VALUES ('1', '1', '2015-02-28 23:04:37', '2015-02-28 23:04:37'), ('2', '1', '2015-03-01
00:00:52', '2015-03-01 00:00:52');
INSERT INTO `groupgraphs` VALUES ('1', '1', 'process', '2', '2015-02-28 19:40:39', '2015-02-28 19:40:39', '0'), ('2', '1',
'process', '2', '2015-03-07 05:46:10', '2015-03-07 05:46:10', '0');
INSERT INTO `models` VALUES ('1', null, 'process', '2', '0', '2015-02-28 14:38:28', '2015-02-28 14:38:29', '1');
INSERT INTO `notifications` VALUES ('1', '1', '1', 'A verification request has been sent for the document', '2015-03-01
01:13:13', '2015-02-28 23:12:21', '2015-03-01 01:13:13', 'Verification request.', 'information'), ('2', '1', '1', 'A verification
request has been sent for this document <a
href=\http://localhost/documents/1/show_properties\>DocumentHarryTest</a>.', '2015-03-01 01:13:13', '2015-02-28
23:12:22', '2015-03-01 01:13:13', 'Verification request.', 'action'), ('3', '1', '1', 'A approval request has been sent for the
document <a href=\http://localhost/documents/1/show_properties\>DocumentHarryTest</a>.', '2015-03-01 01:13:13',
'2015-02-28 23:12:47', '2015-03-01 01:13:13', 'Approval request', 'information'), ('4', '1', '1', 'A approval request has been
sent for the document <a href=\http://localhost/documents/1/show_properties\>DocumentHarryTest</a>.', '2015-03-01
01:13:13', '2015-02-28 23:12:48', '2015-03-01 01:13:13', 'Approval request', 'action'), ('5', '1', '1', 'The document <a
href=\http://localhost/documents/1/show_properties\>DocumentHarryTest</a> has been approved. You can publish it.',
'2015-03-01 01:13:13', '2015-02-28 23:13:49', '2015-03-01 01:13:13', 'document approved.', 'action'), ('6', '1', '1', 'The
document <a href=\http://localhost/documents/1/show_properties\>DocumentHarryTest</a> has been approved.',
'2015-03-01 01:13:13', '2015-02-28 23:13:49', '2015-03-01 01:13:13', 'document approved.', 'information'), ('7', '1', '1', 'A
verification request has been sent for the document', '2015-03-01 01:13:13', '2015-03-01 00:14:19', '2015-03-01 01:13:13',
'Verification request.', 'information'), ('8', '1', '1', 'A verification request has been sent for this document <a
href=\http://localhost/documents/2/show_properties\>SecondTest</a>.', '2015-03-01 01:13:13', '2015-03-01 00:14:20',
'2015-03-01 01:13:13', 'Verification request.', 'action'), ('9', '1', '1', 'A approval request has been sent for the document <a
href=\http://localhost/documents/2/show_properties\>SecondTest</a>.', '2015-03-01 01:13:13', '2015-03-01 00:14:49',
'2015-03-01 01:13:13', 'Approval request', 'information'), ('10', '1', '1', 'A approval request has been sent for the document
<a href=\http://localhost/documents/2/show_properties\>SecondTest</a>.', '2015-03-01 01:13:13', '2015-03-01
00:14:49', '2015-03-01 01:13:13', 'Approval request', 'action'), ('11', '1', '1', 'The document <a
href=\http://localhost/documents/2/show_properties\>SecondTest</a> has been approved. You can publish it.', '2015-
03-01 01:13:13', '2015-03-01 00:15:22', '2015-03-01 01:13:13', 'document approved.', 'action'), ('12', '1', '1', 'The document
<a href=\http://localhost/documents/2/show_properties\>SecondTest</a> has been approved.', '2015-03-01 01:13:13',
'2015-03-01 00:15:22', '2015-03-01 01:13:13', 'document approved.', 'information');
INSERT INTO `resources` VALUES ('1', 'Recurso de prueba', 'http://ddf912383141a8d7bbe4-
e053e711fc85de3290f121ef0fe3a1f.r87.cf1.rackcdn.com/Ruby_on_Rails_logo.jpg', '1', '2015-03-01 01:48:03', '2015-03-01
01:50:04', '1', 'Imagen', 'Prueba de recurso con imagen.');
```

```

INSERT INTO `roles` VALUES ('1', 'Rol de prueba', 'extern', '2015-03-01 01:55:01', '2015-03-01 01:57:26', 'Mision de prueba',
'Actividades para el rol de prueba', '1', null, '1', 'Proposito del rol de prueba');
INSERT INTO `schema_migrations` VALUES ('20121002114906'), ('20121002122958'), ('20121002124403'),
('20121002133148'), ('20121003072522'), ('20121003072820'), ('20121003151152'), ('20121004080517'),
('20121004112232'), ('20121022103558'), ('20121105134754'), ('20121105151326'), ('20121109060744'),

```

```

('20121109112505'), ('20121116080814'), ('20121116121726'), ('20121129102556'), ('20121213134053'),
('20121214082142'), ('20130107114910'), ('20130108120452'), ('20130109071106'), ('20130121143822'),
('20130124143701'), ('20130131102056'), ('20130131112452'), ('20130214074807'), ('20130308130443'),
('20130320083344'), ('20130322103106'), ('20130326134708'), ('20130402114658'), ('20130405114303'),
('20130517112015'), ('20130527073943'), ('20130527131903'), ('20130528074008'), ('20130528101838'),
('20130528114330'), ('20130531080133'), ('20130604114807'), ('20130605123037'), ('20130610075735'),
('20130610131852'), ('20130724064249'), ('20130725120850'), ('20130809122654'), ('20130809151755'),
('20130910065932'), ('20130917073707'), ('20130918124056'), ('20130918124323'), ('20131008120816'),
('20131021102130'), ('20131022090336'), ('20131029070033'), ('20131029133619'), ('20131030104852'),
('20131030114931'), ('20131112071033'), ('20131112071107'), ('20131118045616'), ('20131119070536'),
('20131119134633'), ('20131119135246'), ('20131126112625'), ('20131212075746'), ('20131212080418'),
('20140106130129'), ('20140107134143'), ('20140108104700'), ('20140120070800'), ('20140121113042'),
('20140128124508'), ('20140130062645'), ('20140130104731'), ('20140203105640'), ('20140206104854'),
('20140210064308'), ('20140210110435'), ('20140211062618'), ('20140211075749'), ('20140212071958'),
('20140212072859'), ('20140217064356'), ('20140217064628'), ('20140217064657'), ('20140218054628'),
('20140220102218'), ('20140304151521'), ('20140304151603'), ('20140304152201'), ('20140304152415'),
('20140306050314'), ('20140306061643'), ('20140307111607'), ('20140310070541'), ('20140310071430'),
('20140310071546'), ('20140310071712');
INSERT INTO `schema_migrations` VALUES ('20140310105905'), ('20140310112450'), ('20140310112528'),
('20140310112557'), ('20140310112658'), ('20140313113945'), ('20140314073516'), ('20140314111942'),
('20140318061531'), ('20140318073523'), ('20140318111720'), ('20140321111114'), ('20140321111545'),
('20140322120053'), ('20140322121415'), ('20140322121637'), ('20140322122311'), ('20140322140830'),
('20140324100010'), ('20140409065718'), ('20140409122057'), ('20140409122317'), ('20140422063320'),
('20140422123749'), ('20140425040957'), ('20140502132147'), ('20140502133659'), ('20140506101432'),
('20140519104002'), ('20140521061743'), ('20140603074354'), ('20140702084546'), ('20140703110205'),
('20140703113023'), ('20140703113905'), ('20140703125005'), ('20140710111231'), ('20140711061304'),
('20140711140007'), ('20140716055017'), ('20140716072508'), ('20140718074501'), ('20140725112721'),
('20140729133540'), ('20140730111423'), ('20140731115001'), ('20140801111532'), ('20140808133423'),
('20140812055807'), ('20140825092513');
COMMIT;

ALTER TABLE `arrows` AUTO_INCREMENT=3;
ALTER TABLE `contributions` AUTO_INCREMENT=2;
ALTER TABLE `customers` AUTO_INCREMENT=2;
ALTER TABLE `directories` AUTO_INCREMENT=4;
ALTER TABLE `document_publishers` AUTO_INCREMENT=3;
ALTER TABLE `documents` AUTO_INCREMENT=3;
ALTER TABLE `documents_approvers` AUTO_INCREMENT=3;
ALTER TABLE `documents_logs` AUTO_INCREMENT=13;
ALTER TABLE `documents_verifiers` AUTO_INCREMENT=3;
ALTER TABLE `documents_viewers` AUTO_INCREMENT=4;
ALTER TABLE `elements` AUTO_INCREMENT=4;
ALTER TABLE `favorites` AUTO_INCREMENT=1;
ALTER TABLE `flags` AUTO_INCREMENT=2;
ALTER TABLE `graph_publishers` AUTO_INCREMENT=2;
ALTER TABLE `graphs` AUTO_INCREMENT=3;
ALTER TABLE `graphs_approvers` AUTO_INCREMENT=3;
ALTER TABLE `graphs_contributors` AUTO_INCREMENT=1;
ALTER TABLE `graphs_logs` AUTO_INCREMENT=3;
ALTER TABLE `graphs_roles` AUTO_INCREMENT=1;
ALTER TABLE `graphs_verifiers` AUTO_INCREMENT=2;
ALTER TABLE `graphs_viewers` AUTO_INCREMENT=2;
ALTER TABLE `groupdocuments` AUTO_INCREMENT=3;
ALTER TABLE `groupgraphs` AUTO_INCREMENT=3;
ALTER TABLE `groups` AUTO_INCREMENT=1;
ALTER TABLE `lanes` AUTO_INCREMENT=1;
ALTER TABLE `models` AUTO_INCREMENT=2;
ALTER TABLE `notifications` AUTO_INCREMENT=13;
ALTER TABLE `pastilles` AUTO_INCREMENT=1;
ALTER TABLE `recordings` AUTO_INCREMENT=1;
ALTER TABLE `resources` AUTO_INCREMENT=2;
ALTER TABLE `roles` AUTO_INCREMENT=2;

```

```

ALTER TABLE `roles_users` AUTO_INCREMENT=1;
ALTER TABLE `super_admins` AUTO_INCREMENT=1;
ALTER TABLE `taggings` AUTO_INCREMENT=1;
ALTER TABLE `tags` AUTO_INCREMENT=1;
ALTER TABLE `users` AUTO_INCREMENT=2;

```

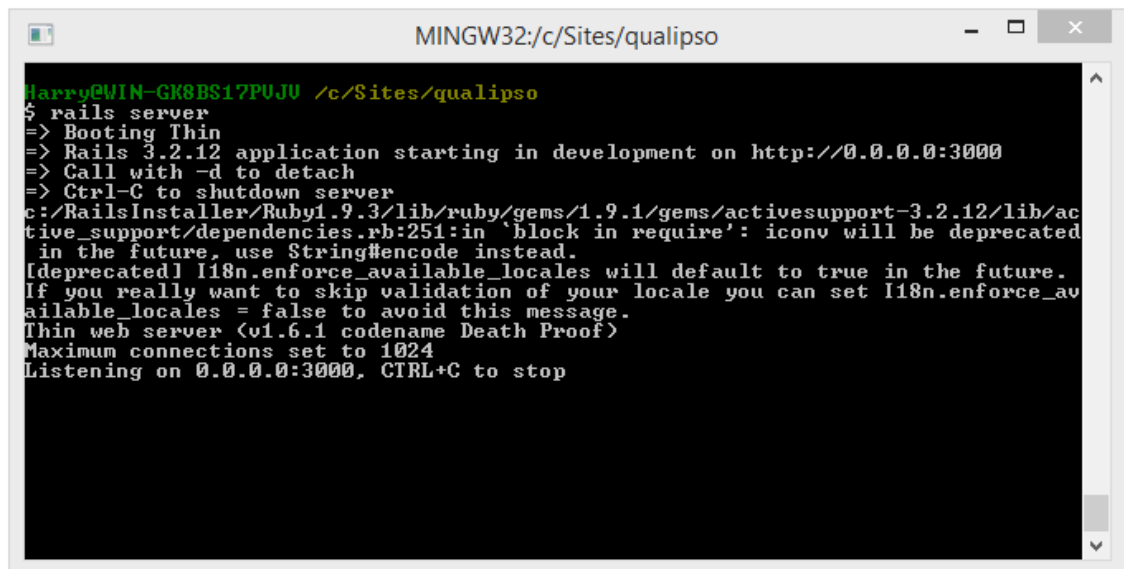
El script puede ser ejecutado como consulta dentro de MySQL Workbench.

The screenshot shows the MySQL Workbench interface. The main window displays a SQL query with 18 lines of code, including a BEGIN statement, several INSERT INTO statements for various tables (customers, users, contributions, arrows, directories, document_publishers, documents, documents_approvers, documents_logs, documents_verifiers, documents_viewers, elements, flags, graph_publishers, graphs, graphs_approvers), and a final INSERT INTO statement for graphs_logs. The Output window at the bottom shows the execution results for the ALTER TABLE statements, indicating that 0 rows were affected for each, with no duplicates or warnings.

Time	Action	Message	Duration / Fetch
61 17:41:04	ALTER TABLE `roles` AUTO_INCREMENT=2	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.000 sec
62 17:41:04	ALTER TABLE `roles_users` AUTO_INCREMENT=1	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.000 sec
63 17:41:04	ALTER TABLE `super_admins` AUTO_INCREMENT=1	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.000 sec
64 17:41:04	ALTER TABLE `taggings` AUTO_INCREMENT=1	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.047 sec
65 17:41:04	ALTER TABLE `tags` AUTO_INCREMENT=1	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
66 17:41:04	ALTER TABLE `users` AUTO_INCREMENT=2	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.000 sec

Levantar servidor

Ruby posee un pequeño web server llamado WEBrick el cual permite desplegar las aplicaciones Rails para fines de desarrollo. Para iniciarlo basta con ejecutar el comando “rails server” tras lo cual tendremos una salida como la siguiente:

A screenshot of a terminal window titled "MINGW32:/c/Sites/qualipso". The terminal shows the execution of the "rails server" command. The output includes: "Booting Thin", "Rails 3.2.12 application starting in development on http://0.0.0.0:3000", "Call with -d to detach", "Ctrl-C to shutdown server", and "Listening on 0.0.0.0:3000, CTRL+C to stop". There are also several deprecation warnings from the ActiveSupport gem.

```
Harry@WIN-GK8BS17P0JU /c/Sites/qualipso
$ rails server
=> Booting Thin
=> Rails 3.2.12 application starting in development on http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
c:/RailsInstaller/Ruby1.9.3/lib/ruby/gems/1.9.1/gems/activesupport-3.2.12/lib/active_support/dependencies.rb:251:in `block in require': iconv will be deprecated in the future, use String#encode instead.
[deprecated] I18n.enforce_available_locales will default to true in the future. If you really want to skip validation of your locale you can set I18n.enforce_available_locales = false to avoid this message.
Thin web server (v1.6.1 codename Death Proof)
Maximum connections set to 1024
Listening on 0.0.0.0:3000, CTRL+C to stop
```

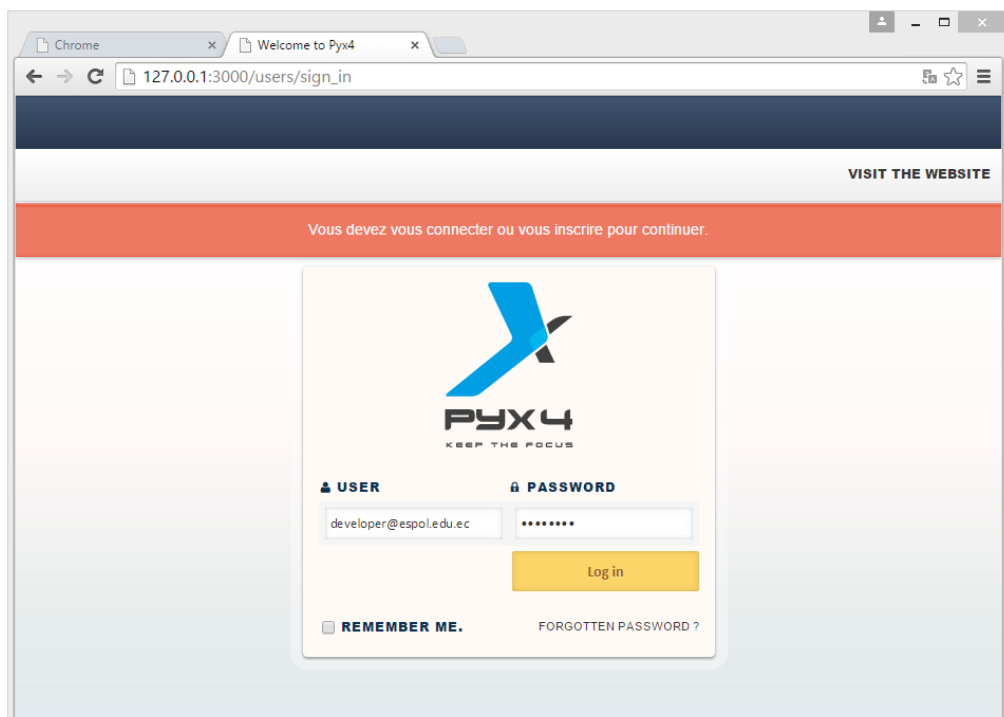

Ingresar a aplicación Web

Iniciar sesión

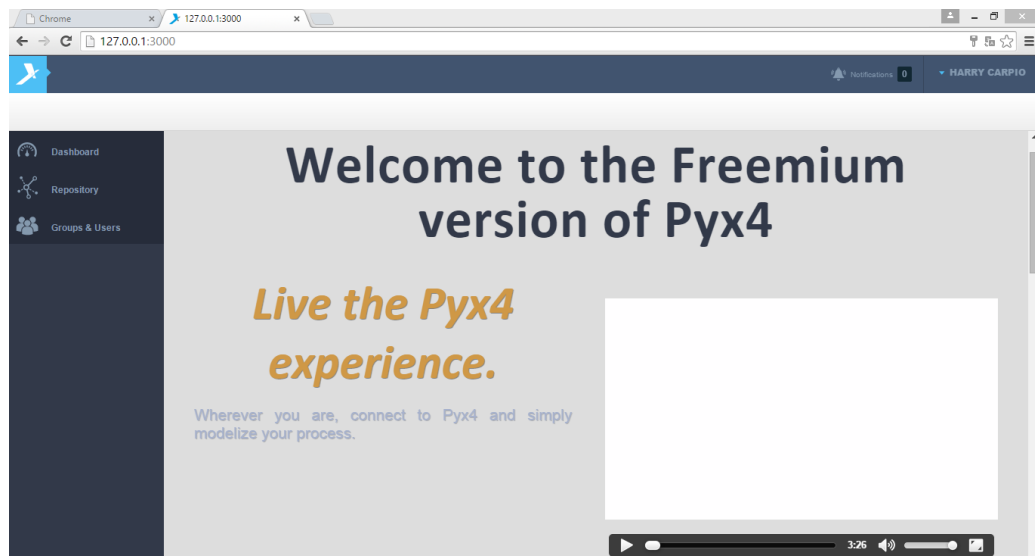
Una vez que se ha ejecutado el servidor web, se debe abrir un navegador y en la barra de direcciones ingresar: <http://127.0.0.1:3000>. Luego se usarán las siguientes credenciales:

User: developer@espol.edu.ec

Password: password



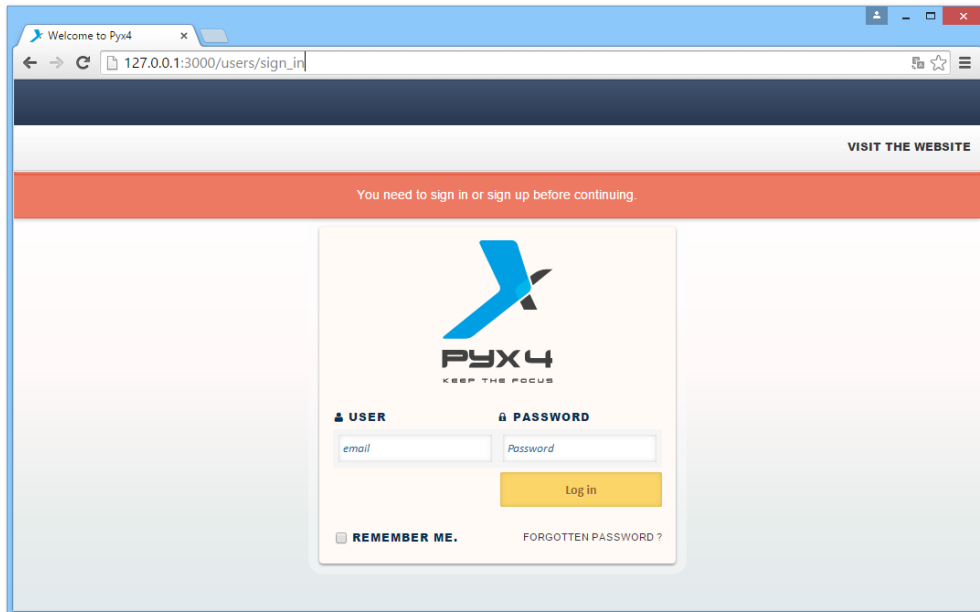
Luego de este paso se tendrá acceso a la aplicación web PYX4 y se podrán explorar sus características.



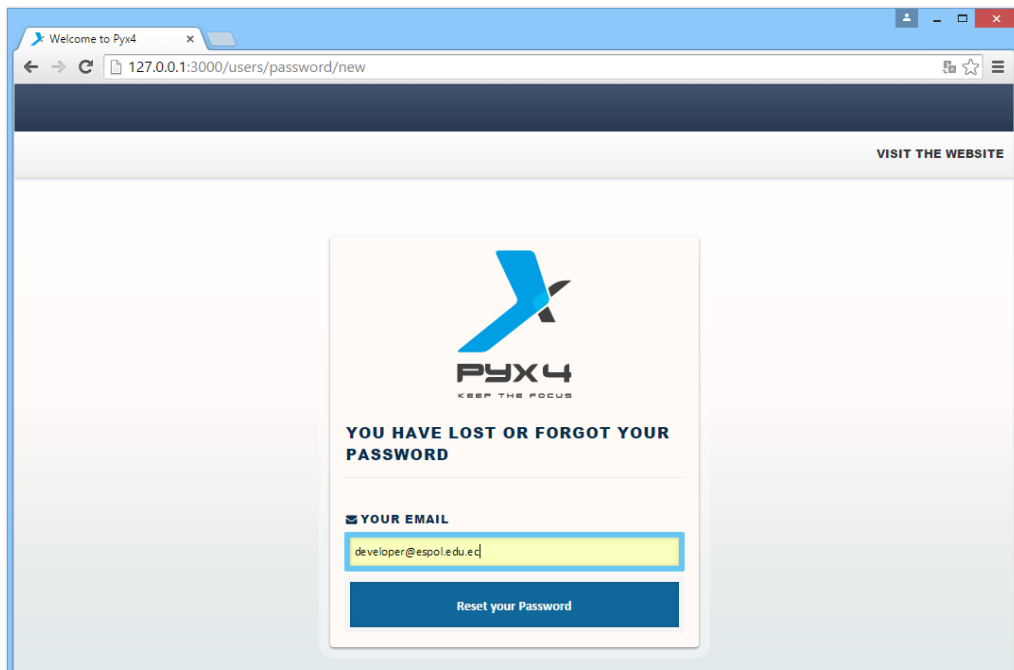
Cambiar contraseña (Opcional)

Si el inicio de sesión falla es posible cambiar la contraseña por defecto. Para hacerlo sin dificultades se requiere que el sistema esté configurado para que abra los archivos con extensión “html” con un navegador, esto es importante a la hora de generar la vista que redirecciona hacia la vista de cambio de contraseña.

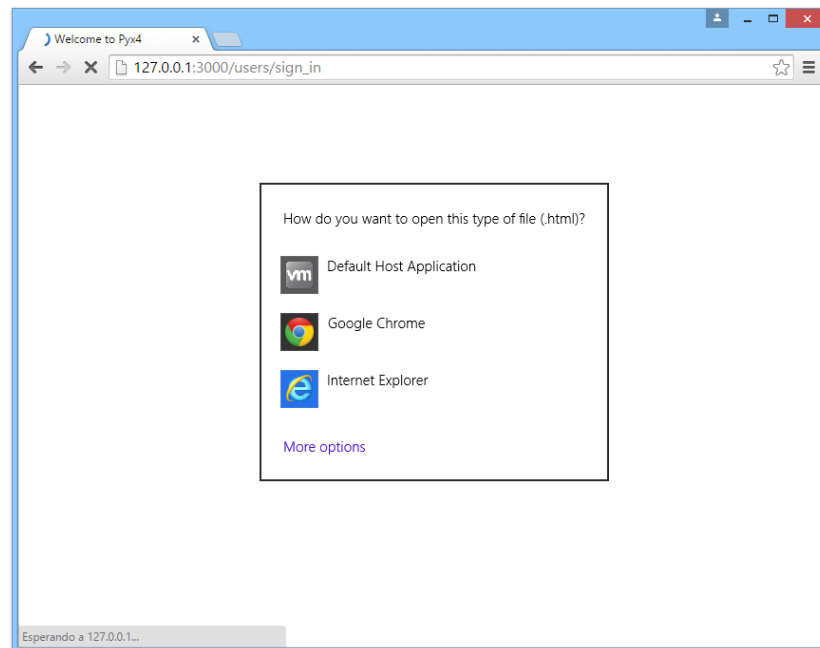
Se comienza desde la página de inicio de sesión “Login” en donde se selecciona el enlace “FORGOTTEN PASSWORD?”



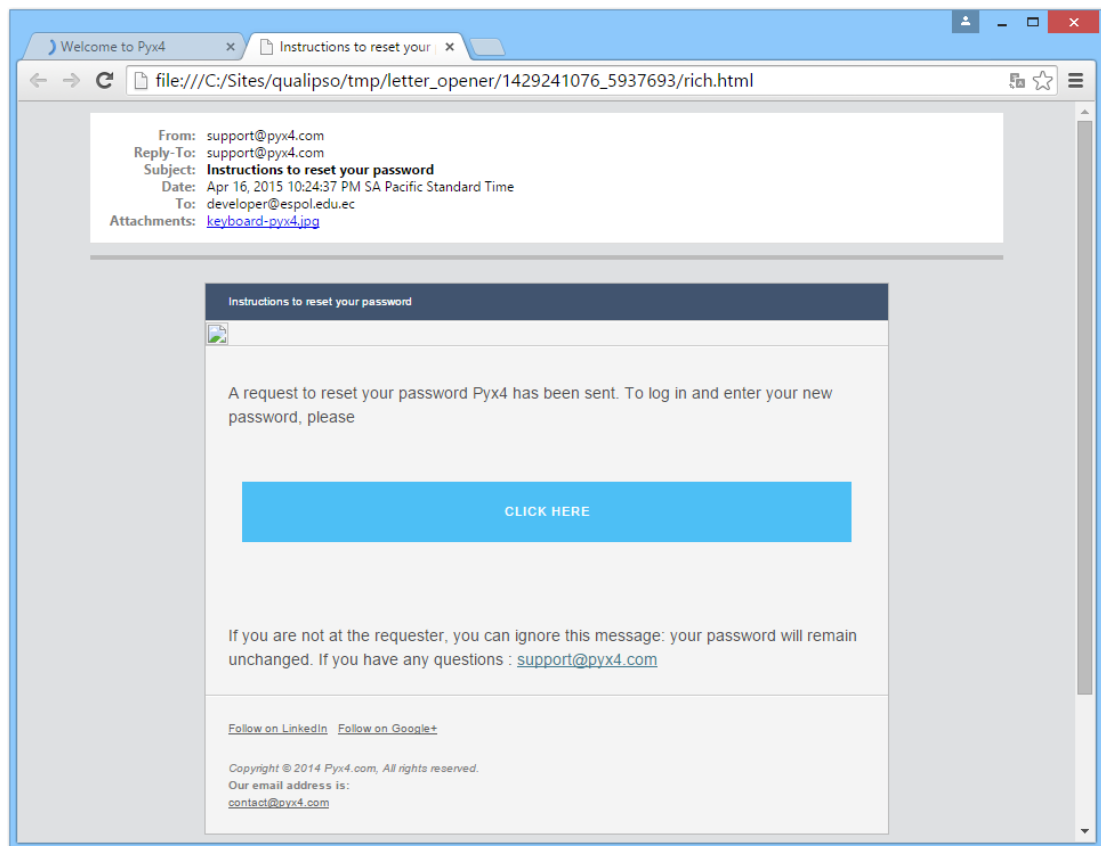
Al seleccionarlo aparecerá una página para ingresar el email del usuario al cual se le desea cambiar la contraseña, en este caso será developer@espol.edu.ec



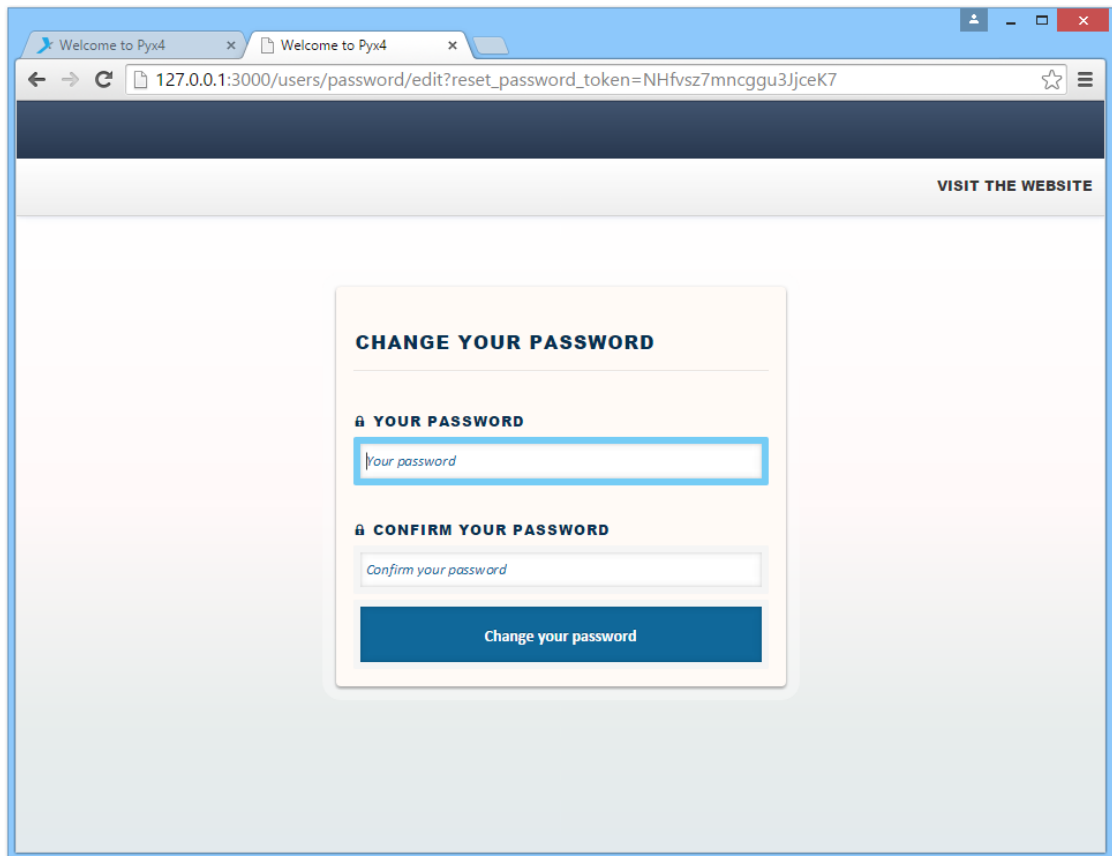
Si no se tiene seleccionada ninguna aplicación por defecto en Windows para abrir los archivos “html”, el sistema operativo preguntará con qué programa se desea abrir la página generada. Se debe seleccionar un explorador web.



A continuación aparecerá la página que debería haber sido enviada al correo del usuario. Sin embargo, como se está usando un correo de prueba, deberá visualizarla de esta manera.



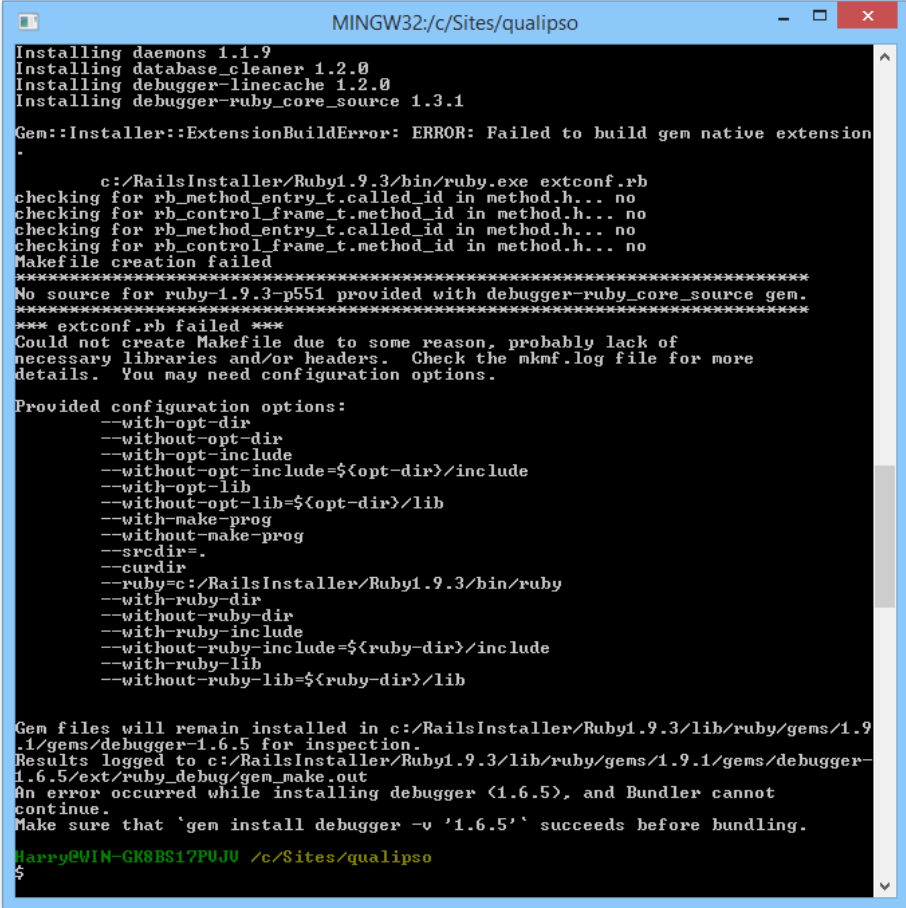
Dando clic en el botón indicado, se continúa en la página para hacer el cambio de contraseña en donde se la debe ingresar dos veces, la segunda como confirmación. Tras esto el cambio estará completo y la sesión será iniciada.

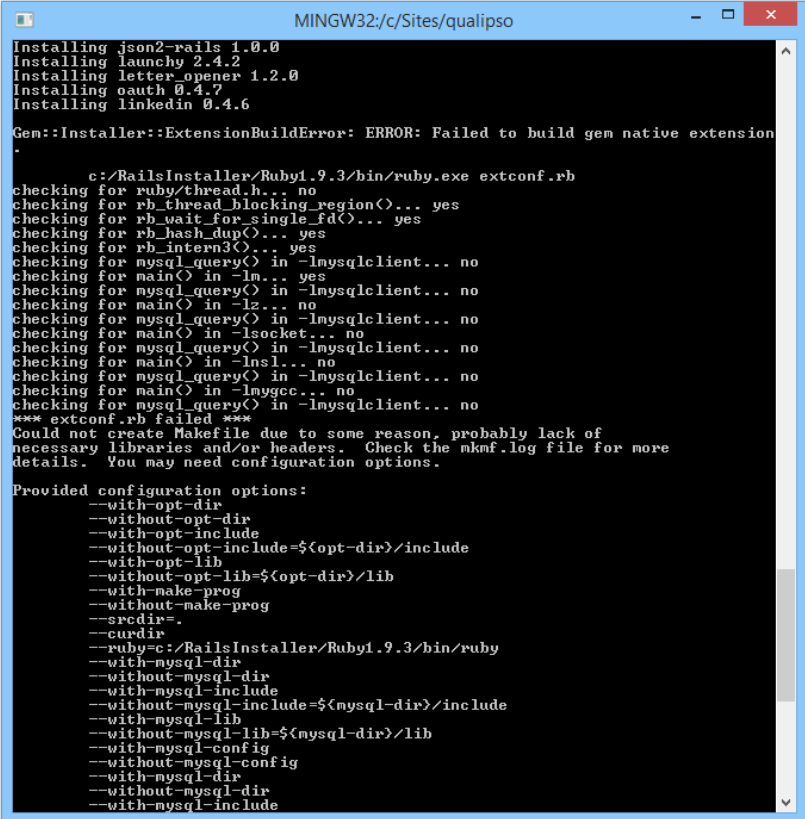


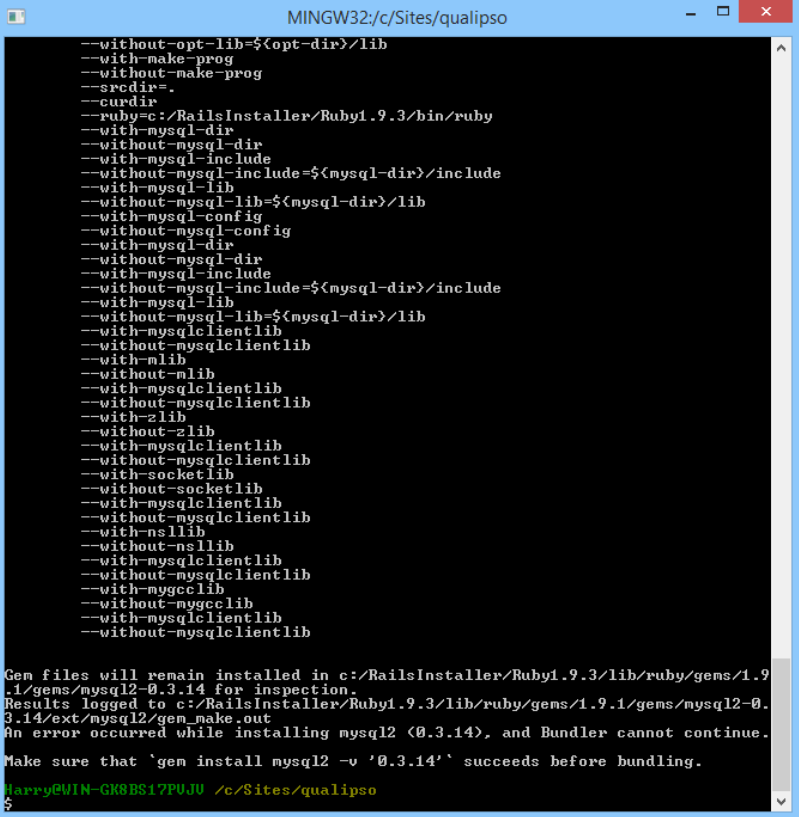
The image shows a web browser window with two tabs titled "Welcome to Pyx4". The address bar displays the URL: `127.0.0.1:3000/users/password/edit?reset_password_token=NHfvsz7mncgggu3JjceK7`. The page content includes a dark blue header with the text "VISIT THE WEBSITE" on the right. The main content area features a light-colored box with the following elements:

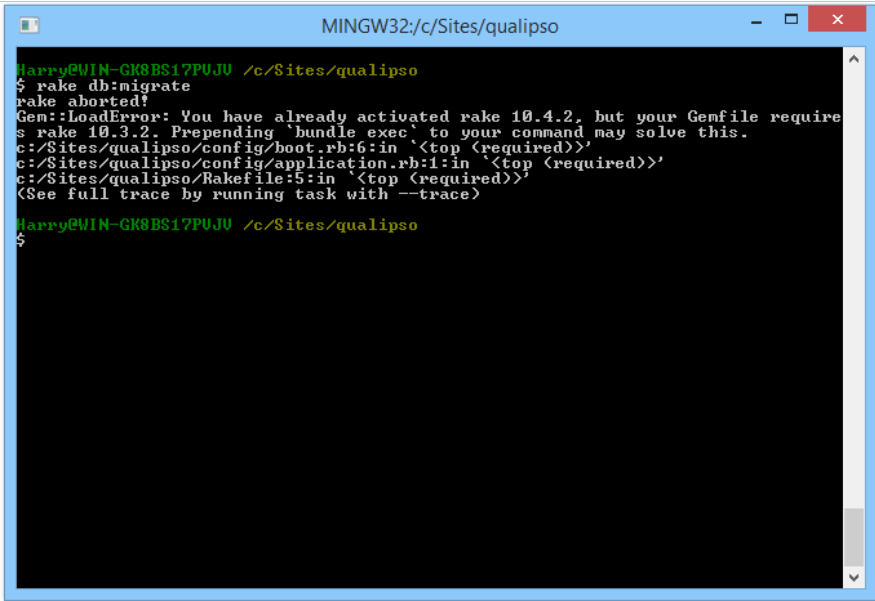
- CHANGE YOUR PASSWORD**: A section header.
- YOUR PASSWORD**: A label above a text input field containing the placeholder text "Your password".
- CONFIRM YOUR PASSWORD**: A label above a text input field containing the placeholder text "Confirm your password".
- Change your password**: A dark blue button with white text.

Problemas frecuentes con gemas

Gema	Debugger
Versión	1.6.5
Descripción	Es una implementación rápida del estándar Ruby debugger. Provee el manejo de punto de interrupción y consolidaciones de marcos de pila entre otras características.
Error	 <pre> MINGW32:c:/Sites/qualipso Installing daemons 1.1.9 Installing database_cleaner 1.2.0 Installing debugger-linecache 1.2.0 Installing debugger-ruby_core_source 1.3.1 Gem::Installer::ExtensionBuildError: ERROR: Failed to build gem native extension . c:/RailsInstaller/Ruby1.9.3/bin/ruby.exe extconf.rb checking for rb_method_entry_t.called_id in method.h... no checking for rb_control_frame_t.method_id in method.h... no checking for rb_method_entry_t.called_id in method.h... no checking for rb_control_frame_t.method_id in method.h... no Makefile creation failed ***** No source for ruby-1.9.3-p551 provided with debugger-ruby_core_source gem. ***** *** extconf.rb failed *** Could not create Makefile due to some reason, probably lack of necessary libraries and/or headers. Check the mkmf.log file for more details. You may need configuration options. Provided configuration options: --with-opt-dir --without-opt-dir --with-opt-include --without-opt-include=\${opt-dir}/include --with-opt-lib --without-opt-lib=\${opt-dir}/lib --with-make-prog --without-make-prog --srcdir=. --curdir --ruby=c:/RailsInstaller/Ruby1.9.3/bin/ruby --with-ruby-dir --without-ruby-dir --with-ruby-include --without-ruby-include=\${ruby-dir}/include --with-ruby-lib --without-ruby-lib=\${ruby-dir}/lib Gem files will remain installed in c:/RailsInstaller/Ruby1.9.3/lib/ruby/gems/1.9 .1/gems/debugger-1.6.5 for inspection. Results logged to c:/RailsInstaller/Ruby1.9.3/lib/ruby/gems/1.9.1/gems/debugger- 1.6.5/ext/ruby_debug/gem_make.out An error occurred while installing debugger (1.6.5), and Bundler cannot continue. Make sure that `gem install debugger -v '1.6.5` succeeds before bundling. Harry@WIN-GK8BS17PUJV /c:/Sites/qualipso \$ </pre>
Descripción solución	Ejecutar comando: gem update debugger
Exit code	Successfully installed debugger-ruby_core_source-1.3.8 Gems updated: debugger-ruby_core_source Installing ri documentation for debugger-ruby_core_source-1.3.8... Installing RDoc documentation for debugger-ruby_core_source-1.3.8...

Gema	MySQL2
Versión	0.3.14
Descripción	Una simple y rápida librería MySQL para Ruby, vinculada a librería libmysql.
Error	 <pre> MINGW32/c:/Sites/qualipso Installing json2-rails 1.0.0 Installing launchy 2.4.2 Installing letter_opener 1.2.0 Installing oauth 0.4.7 Installing linkedin 0.4.6 Gem::Installer::ExtensionBuildError: ERROR: Failed to build gem native extension . c:/RailsInstaller/Ruby1.9.3/bin/ruby.exe extconf.rb checking for ruby/thread.h... no checking for rb_thread_blocking_region()... yes checking for rb_wait_for_single_fd()... yes checking for rb_hash_dup()... yes checking for rb_intern3()... yes checking for mysql_query() in -lmysqlclient... no checking for main() in -lm... yes checking for mysql_query() in -lmysqlclient... no checking for main() in -lz... no checking for mysql_query() in -lmysqlclient... no checking for main() in -lsocket... no checking for mysql_query() in -lmysqlclient... no checking for main() in -lnsl... no checking for mysql_query() in -lmysqlclient... no checking for main() in -lmygcc... no checking for mysql_query() in -lmysqlclient... no *** extconf.rb failed *** Could not create Makefile due to some reason, probably lack of necessary libraries and/or headers. Check the nkmf.log file for more details. You may need configuration options. Provided configuration options: --with-opt-dir --without-opt-dir --with-opt-include --without-opt-include=\${opt-dir}/include --with-opt-lib --without-opt-lib=\${opt-dir}/lib --with-make-prog --without-make-prog --srcdir=. --curdir --ruby=c:/RailsInstaller/Ruby1.9.3/bin/ruby --with-mysql-dir --without-mysql-dir --with-mysql-include --without-mysql-include=\${mysql-dir}/include --with-mysql-lib --without-mysql-lib=\${mysql-dir}/lib --with-mysql-config --without-mysql-config --with-mysql-dir --without-mysql-dir --with-mysql-include </pre>

	 <pre> --without-opt-lib=\${opt-dir}/lib --with-make-prog --without-make-prog --srcdir=. --curdir --ruby=c:/RailsInstaller/Ruby1.9.3/bin/ruby --with-mysql-dir --without-mysql-dir --with-mysql-include --without-mysql-include=\${mysql-dir}/include --with-mysql-lib --without-mysql-lib=\${mysql-dir}/lib --with-mysql-config --without-mysql-config --with-mysql-dir --without-mysql-dir --with-mysql-include --without-mysql-include=\${mysql-dir}/include --with-mysql-lib --without-mysql-lib=\${mysql-dir}/lib --with-mysqlclientlib --without-mysqlclientlib --with-mlib --without-mlib --with-mysqlclientlib --without-mysqlclientlib --with-zlib --without-zlib --with-mysqlclientlib --without-mysqlclientlib --with-socketlib --without-socketlib --with-mysqlclientlib --without-mysqlclientlib --with-nsllib --without-nsllib --with-mysqlclientlib --without-mysqlclientlib --with-mysqclib --without-mysqclib --with-mysqlclientlib --without-mysqlclientlib </pre> <p> Gem files will remain installed in c:/RailsInstaller/Ruby1.9.3/lib/ruby/gems/1.9.1/gems/mysql2-0.3.14 for inspection. Results logged to c:/RailsInstaller/Ruby1.9.3/lib/ruby/gems/1.9.1/gems/mysql2-0.3.14/ext/mysql2/gem_make.out An error occurred while installing mysql2 (0.3.14), and Bundler cannot continue. Make sure that `gem install mysql2 -v '0.3.14` succeeds before bundling. </p> <pre> Harry@WIN-GK8BS17PUJU /c/Sites/qualipso \$ </pre>
Descripción solución	<p>Paso 1.- Copiar los archivos libmysql.dll y libmysql.lib que se encuentran dentro del directorio “C:\Program Files (x86)\MySQL\MySQL Connector.C 6.1\lib” en de la carpeta “C:\RailsInstaller\Ruby1.9.3”.</p> <p>Paso 2.- Copiar los archivos libmysql.dll y libmysql.lib que se encuentran dentro del directorio “C:\Program Files (x86)\MySQL\MySQL Connector.C 6.1\lib” en de la carpeta “C:\RailsInstaller\Ruby1.9.3\bin”.</p> <p>Paso 3.- Ejecutar el comando: gem install mysql2 -v 0.3.14 -- '--with-mysql-dir="C:\Program Files (x86)\MySQL\MySQL Connector.C 6.1"'</p>
Exit code	<p>Successfully installed mysql2-0.3.14 1 gem installed Installing ri documentation for mysql2-0.3.14... Installing RDoc documentation for mysql2-0.3.14...</p>

Gema	Rake
Versión	10.3.2
Descripción	<p>Rake es una gema que implementa un comportamiento similar al que posee el comando Make en sistemas UNIX, es una herramienta para automatizar la construcción de binarios y para administrar tareas.</p> <p>Durante la instalación de Ruby o RoR puede instalarse la versión más reciente de esta gema, lo cual puede causar debido a cambios que pueden afectar al sistema PYX4, por tanto se requiere dejar instalada la versión correspondiente.</p>
Error	 <pre> MINGW32:/c/Sites/qualipso Harry@WIN-GK8BS17PUJU /c/Sites/qualipso \$ rake db:migrate rake aborted! Gem::LoadError: You have already activated rake 10.4.2, but your Gemfile requires rake 10.3.2. Prepending 'bundle exec' to your command may solve this. c:/Sites/qualipso/config/boot.rb:6:in `<top (required)>' c:/Sites/qualipso/config/application.rb:1:in `<top (required)>' c:/Sites/qualipso/Rakefile:5:in `<top (required)>' (See full trace by running task with --trace) Harry@WIN-GK8BS17PUJU /c/Sites/qualipso \$ </pre>
Descripción solución	<p>Desinstalar todas las versiones superiores a la 10.3.2 de la gema rake, para hacerlo ejecutar el comando:</p> <pre>gem uninstall rake</pre> <p>Aparecerá una lista de las versiones instaladas, para desinstalarlas seleccionar su número, de manera similar a como se muestra en la captura:</p>

	 <pre>MINGW32:/c/Sites/qualipso Harry@WIN-GR8BS17PUJU /c/Sites/qualipso \$ gem uninstall rake Select gem to uninstall: 1. rake-0.9.2.2 2. rake-10.3.2 3. rake-10.4.2 4. All versions > 3</pre>
Exit code	Successfully uninstalled rake-{version}

Gema	Rmagick
Versión	2.13.2
Descripción	Provee una interface entre Ruby e ImageMagick.
Error	 <pre> MINGW32:/c/Sites/qualipso Installing pundit 0.2.1 Installing quiet_assets 1.0.2 Installing rails_config 0.3.3 Gem::Installer::ExtensionBuildError: ERROR: Failed to build gem native extension . c:/RailsInstaller/Ruby1.9.3/bin/ruby.exe extconf.rb checking for Ruby version >= 1.8.5... yes Invalid drive specification. Unable to get ImageMagick version *** extconf.rb failed *** Could not create Makefile due to some reason, probably lack of necessary libraries and/or headers. Check the mkmf.log file for more details. You may need configuration options. Provided configuration options: --with-opt-dir --without-opt-dir --with-opt-include --without-opt-include=\${opt-dir}/include --with-opt-lib --without-opt-lib=\${opt-dir}/lib --with-make-prog --without-make-prog --sreaddir= --curdir= --ruby=c:/RailsInstaller/Ruby1.9.3/bin/ruby Gem files will remain installed in c:/RailsInstaller/Ruby1.9.3/lib/ruby/gems/1.9 .1/gems/rmagick-2.13.2 for inspection. Results logged to c:/RailsInstaller/Ruby1.9.3/lib/ruby/gems/1.9.1/gems/rmagick-2 .13.2/ext/RMagick/gem_make.out An error occurred while installing rmagick (2.13.2), and Bundler cannot continue. Make sure that `gem install rmagick -v '2.13.2'` succeeds before bundling. Harry@WIN-GK8BS17PUJU /c/Sites/qualipso \$ </pre>
Descripción solución	Ejecutar comando: <code>gem install rmagick -v '2.13.2' -- --platform=ruby '--with-opt-include=C:\ImageMagick\include --with-opt-lib=C:\ImageMagick'</code>
Exit code	Successfully installed rmagick-2.13.2 1 gem installed Installing ri documentation for rmagick-2.13.2... Installing RDoc documentation for rmagick-2.13.2...

ANEXO B

CONTROLADORES RAILS DE PYX4

