Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

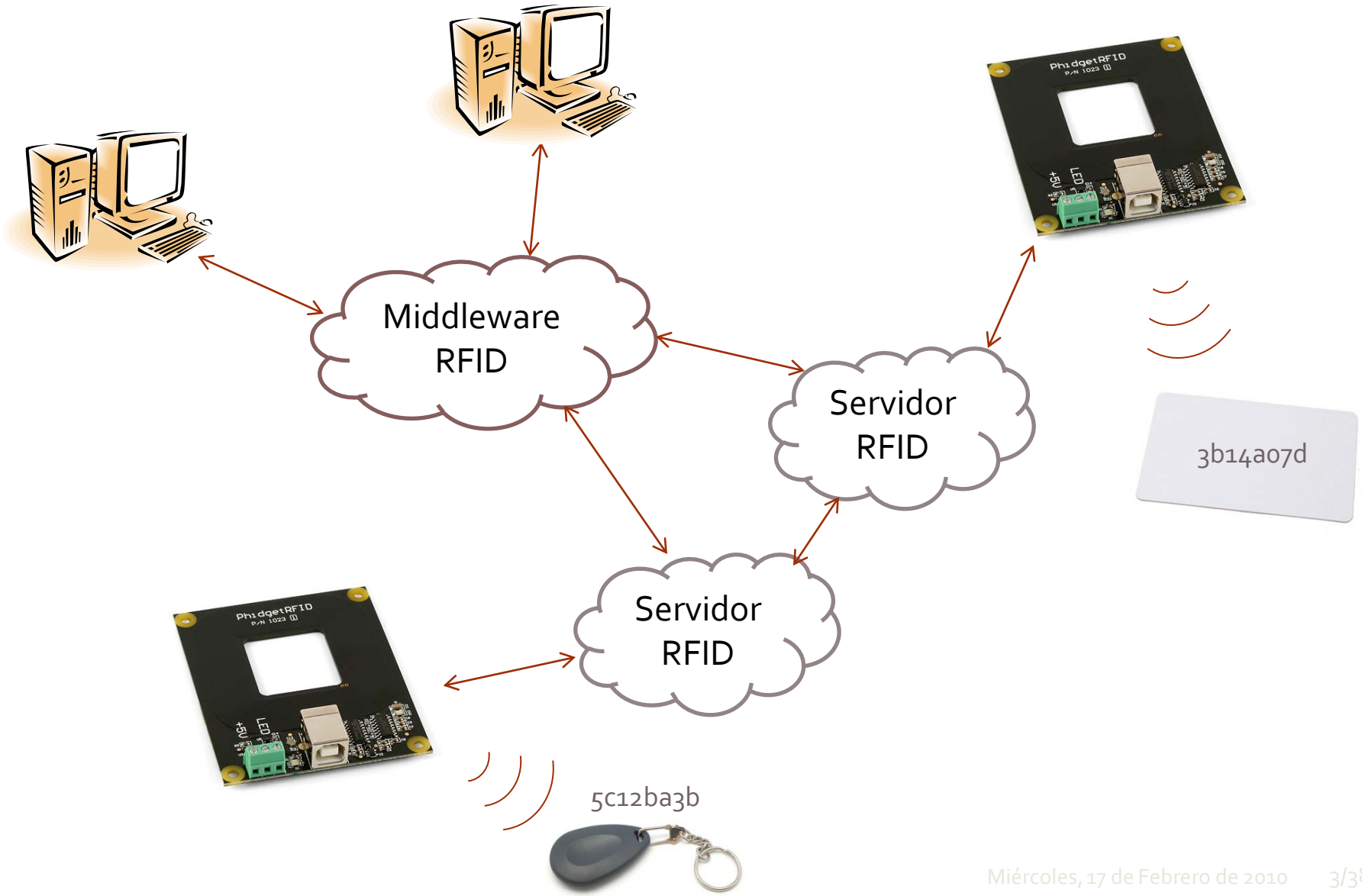
ELABORACIÓN DE UN GEM DE RUBY PARA CONSTRUIR UN MIDDLEWARE RFID BASADO EN LA ARQUITECTURA CLIENTE-SERVIDOR

Presentadores
Bachiller Daniel González Torres
Bachiller Joel Luis Ojeda Redondo
Tutor
Profesor Sergio Rivas

Introducción

- Promover el uso de la tecnología RFID como herramienta para la identificación de objetos
- La elaboración de un middleware RFID adaptable a cualquier tipo de aplicación
- Promover el desarrollo de aplicaciones con Ruby por sus características interesantes y convenientes para desarrollar software de manera ágil

Sistema RFID



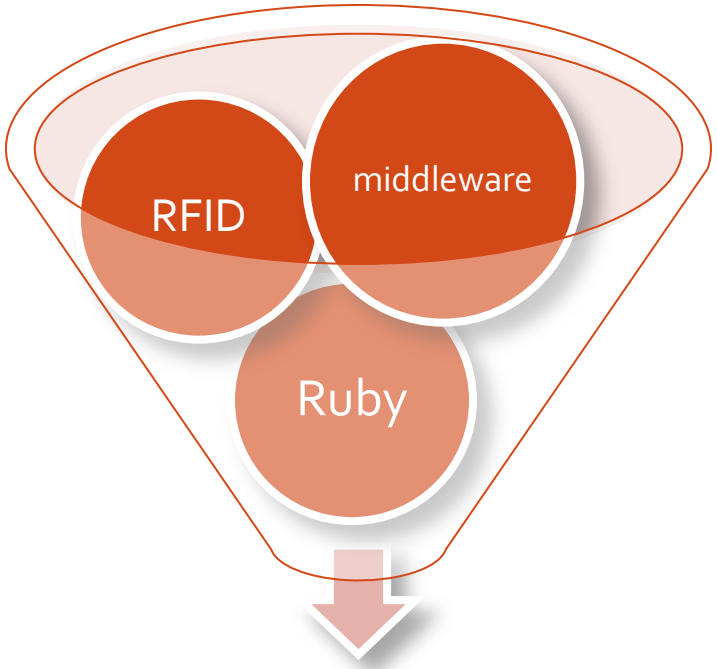
PROPUESTA

Contexto

- Tecnologías de identificación y movilidad
- RFID posee cualidades superiores frente a otras tecnologías como códigos de barras o bandas magnéticas
- Completos middleware RFID de gestión, seguimiento, ubicación, rastreo, localización, lectura, escritura, cálculos estadísticos, identificación, y demás funcionalidades

Contexto

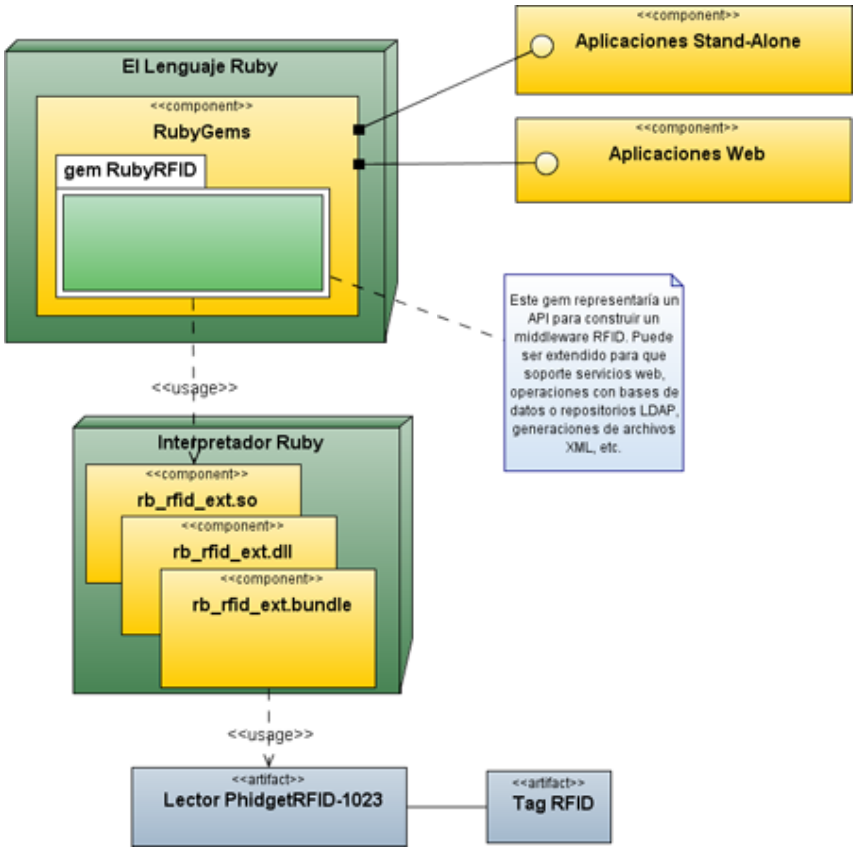
- Ruby es un lenguaje de programación orientado a objetos
- Las aplicaciones o librerías de código (gems) pueden ser distribuidas a través del sistema RubyGems
- Estos gems conforman paquetes de código que pueden ser instalados a través del sistema RubyGems, resolviendo automáticamente cualquier dependencia
- Técnicas, herramientas y código para integración de otros lenguajes como Java, C o C++ con Ruby



Integración de la Tecnología RFID con Ruby

Problemática

- Se presenta el interés de explotar y aprovechar al máximo las ventajas que ofrece la tecnología RFID
- Ruby que por su facilidad de aprendizaje y desarrollo ágil, ofrece el soporte práctico para la implementación de un sistema RFID
- Aprovechar las conveniencias del sistema RubyGems para hacer más práctico el uso y distribución
- Es posible que otros desarrolladores puedan participar en mejoras a futuro



Objetivos

Ojetivo General

Desarrollo de un *gem* de Ruby que permita implementar un middleware RFID

De esta manera, se desarrollará un producto que podrá ser usado y extendido por cualquier desarrollador de la comunidad de Ruby, ya que estará disponible en los repositorios de RubyForge

- Utilizar la propuesta del proceso de desarrollo XP la metodología de diseño AM
- Patentar el desarrollo del gem bajo una licencia pública (código abierto)
- Publicar el gem en los repositorios de RubyForge
- Proponer la integración del API para Java o C/C++ del dispositivo PhidgetRFID-1023 con módulos hechos en Ruby

Tecnologías, Plataformas, Procesos y Metodologías

Herramientas de software libre

- El lenguaje de programación Ruby y el sistema de gestión de paquetes RubyGems
- El intérprete de Ruby 1.8.6
- El sistema de control de versiones Subversion 1.4.5
- El API para C/C++ del dispositivo PhidgetRFID-1023
- Los dispositivos lectores y tags de la familia PhidgetRFID

Plataformas

- Ubuntu Linux 32-bits

Procesos y Metodologías de Desarrollo

- Proceso de desarrollo XP, complementado con las prácticas de diseño y modelación de la metodología AM

LA TECNOLOGÍA RFID

Etiquetas (tags) y Lectores RFID

Etiquetas

- Presentación
- Uniones
- Capacidad de Almacenamiento
- Energía
- Estándares

Lectores

- API del Lector
- Subsistema de Antenas
- Comunicaciones
- Manejador de Eventos

Middleware RFID

•¿Por qué es necesario un *middleware* RFID?

- Considerar un escenario simple de una cadena de tiendas de artículos electrónicos

Adaptador de Lectura

- Encapsula las interfaces propietarias de cada lector

Manejador de Eventos

- Filtra, consolida y transforma observaciones e interrogaciones
- Provee mecanismos de:
  - observación en conjunto
  - por tiempo
  - a través de múltiples lectores

Interfaz de Nivel de Aplicación

- Estandariza el registro y la recepción de eventos
- Provee un API estándar para configurar, monitorear y administrar el middleware RFID y los lectores que controla

EL LENGUAJE RUBY

Características de Ruby

- Lenguaje de programación orientado a objetos
- Viendo todo como un objeto
- La flexibilidad de Ruby
- Los Bloques, una funcionalidad realmente expresiva
- Ruby y el Mixin
- Más allá de lo básico

  - YARV
  - JRuby

Integración con C

- Las extensiones en C o C++ para Ruby no son tan difíciles de crear
- Lo más difícil es encontrar las descripciones detalladas para las funciones del API C de Ruby

```
require `mkmf'
create_makefile(`myextension')
```

```
#include "ruby.h"
void Init_myextension() {
      /* codigo de la extension */
}
```

```
gcc -fPIC -I/usr/local/lib/ruby/1.6/i686-
linux -g -O2 o main.o
gcc -shared -o myextension.so main.o -lc
```

RubyGems

- ¿Qué es un Gem?
- ¿Qué es RubyGems?
- ¿Cómo Instalar RubyGems?
- ¿Cómo Utilizar RubyGems?
- Ventajas

Provee un mecanismo estándar para describir requerimientos y software

Provee un repositorio central de software

Permite la distribución de *gems* usando un servidor

Maneja las dependencias de software automáticamente

Maneja diferentes versiones de software inteligentemente

Puede ser usado transparentemente en lugar de librerías regulares de Ruby

Permite usar la misma tecnología en diferentes sistemas operativos

APLICACIÓN

Adaptación de XP y AM

El desarrollo se basó en iteraciones de una semana de duración.

En la actividad de planificación se definen las actividades y responsabilidades.

En la actividad de diseño se elabora una solución a la problemática planteada en la iteración actual.

En la actividad de codificación se implementa la solución o una aproximación de la misma.

En la actividad de pruebas se verifica el correcto funcionamiento de lo implementado.

Iteraciones

- Planificación
- Diseño
- Codificación
- Pruebas

Subversion

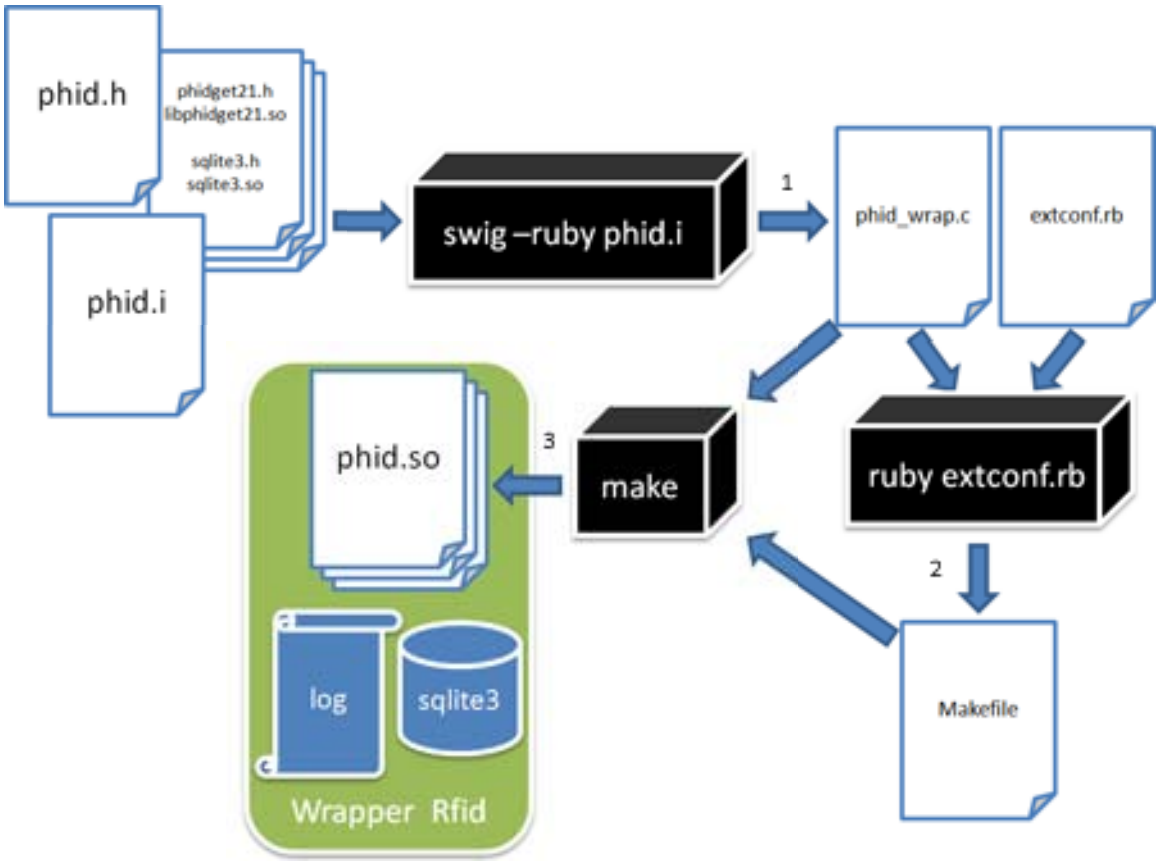
- assembla.com

Desarrollo

- **Iteración 0:** se desarrolló la extensión (wrapper) de Ruby para administrar el dispositivo PhidgetRFID-1023
- **Iteración 1:** se agregaron nuevas funcionalidades al wrapper y se creó la clase de Ruby que utiliza la extensión (desarrollo del adaptador de lectura)
- **Iteración 2 (continuación de it. 1):** se agregó una implementación de un log con una base de datos *Sqlite* y con un archivo de texto plano
- **Iteración 3 (continuación de it. 1):** se desarrolló la estructura definitiva del wrapper para establecer el estándar a seguir
- **Iteración 4:** se desarrolló un servidor sencillo basado en comandos (desarrollo del manejador de eventos)
- **Iteración 5:** se desarrolló un cliente para establecer comunicación con otros servidores (desarrollo de la interfaz de nivel de aplicación)
- **Iteración 6:** se empaquetó la aplicación en un gem de Ruby y se importó a su repositorio de RubyForge
- **Iteración 7:** se hicieron mejoras, correcciones y otras modificaciones del gem y la extensión de Ruby

Iteración 0

El proceso de desarrollo de la extensión de Ruby para utilizar el dispositivo PhidgetRFID a través de su API del lenguaje C, se simplifica utilizando la herramienta SWIG.

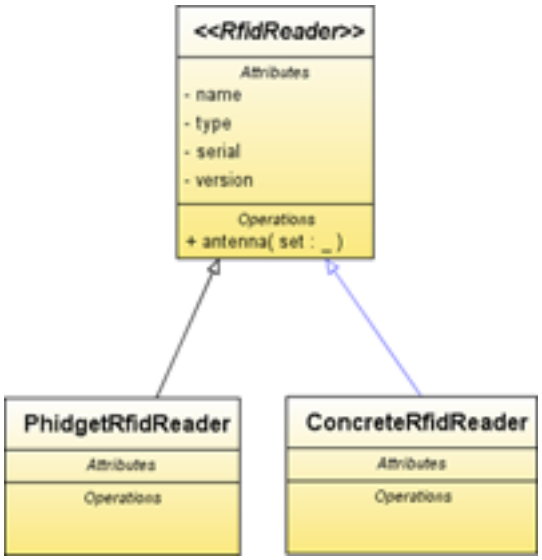


Desarrollo

- **Iteración 0:** se desarrolló la extensión (wrapper) de Ruby para administrar el dispositivo PhidgetRFID-1023
- **Iteración 1:** se agregaron nuevas funcionalidades al wrapper y se creó la clase de Ruby que utiliza la extensión (desarrollo del adaptador de lectura)
- **Iteración 2 (continuación de it. 1):** se agregó una implementación de un log con una base de datos *Sqlite* y con un archivo de texto plano
- **Iteración 3 (continuación de it. 1):** se desarrolló la estructura definitiva del wrapper para establecer el estándar a seguir
- **Iteración 4:** se desarrolló un servidor sencillo basado en comandos (desarrollo del manejador de eventos)
- **Iteración 5:** se desarrolló un cliente para establecer comunicación con otros servidores (desarrollo de la interfaz de nivel de aplicación)
- **Iteración 6:** se empaquetó la aplicación en un gem de Ruby y se importó a su repositorio de RubyForge
- **Iteración 7:** se hicieron mejoras, correcciones y otras modificaciones del gem y la extensión de Ruby

Iteraciones 1, 2 y 3

- En estas iteraciones se definieron las funcionalidades determinantes del wrapper
- Se delimitaron las funcionalidades más importantes del componente de *interfaz de nivel de aplicación*



Utilización del patrón *Template Method*

Estructura de la tabla *log*

Desarrollo

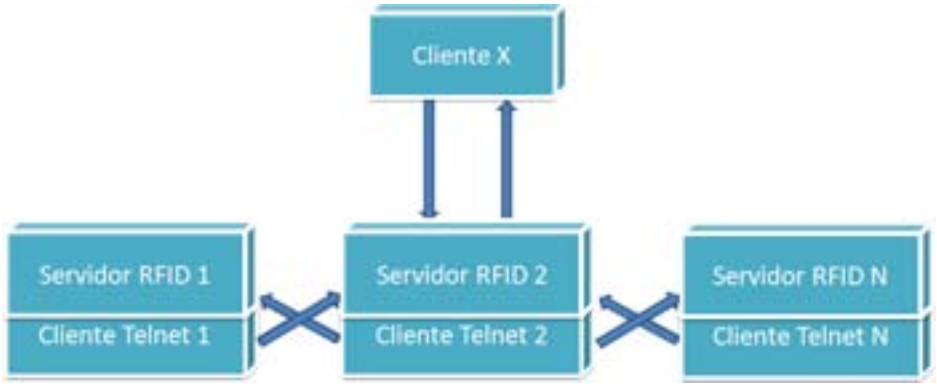
- **Iteración 0:** se desarrolló la extensión (wrapper) de Ruby para administrar el dispositivo PhidgetRFID-1023
- **Iteración 1:** se agregaron nuevas funcionalidades al wrapper y se creó la clase de Ruby que utiliza la extensión (desarrollo del adaptador de lectura)
- **Iteración 2 (continuación de it. 1):** se agregó una implementación de un log con una base de datos *Sqlite* y con un archivo de texto plano
- **Iteración 3 (continuación de it. 1):** se desarrolló la estructura definitiva del wrapper para establecer el estándar a seguir
- **Iteración 4:** se desarrolló un servidor sencillo basado en comandos (desarrollo del manejador de eventos)
- **Iteración 5:** se desarrolló un cliente para establecer comunicación con otros servidores (desarrollo de la interfaz de nivel de aplicación)
- **Iteración 6:** se empaquetó la aplicación en un gem de Ruby y se importó a su repositorio de RubyForge
- **Iteración 7:** se hicieron mejoras, correcciones y otras modificaciones del gem y la extensión de Ruby

Esquema de interconexión entre servidores

Cada servidor es a su vez un cliente de otros servidores.

Un servidor puede resolver una consulta o un comando de manera local y/o retransmitirlo a otro servidor.

Un cliente (aplicación externa) puede conectarse a un servidor y debe poder administrar el sistema completo a través de ese único servidor.



Desarrollo

- **Iteración 0:** se desarrolló la extensión (wrapper) de Ruby para administrar el dispositivo PhidgetRFID-1023
- **Iteración 1:** se agregaron nuevas funcionalidades al wrapper y se creó la clase de Ruby que utiliza la extensión (desarrollo del adaptador de lectura)
- **Iteración 2 (continuación de it. 1):** se agregó una implementación de un log con una base de datos *Sqlite* y con un archivo de texto plano
- **Iteración 3 (continuación de it. 1):** se desarrolló la estructura definitiva del wrapper para establecer el estándar a seguir
- **Iteración 4:** se desarrolló un servidor sencillo basado en comandos (desarrollo del manejador de eventos)
- **Iteración 5:** se desarrolló un cliente para establecer comunicación con otros servidores (desarrollo de la interfaz de nivel de aplicación)
- **Iteración 6:** se empaquetó la aplicación en un gem de Ruby y se importó a su repositorio de RubyForge
- **Iteración 7:** se hicieron mejoras, correcciones y otras modificaciones del gem y la extensión de Ruby

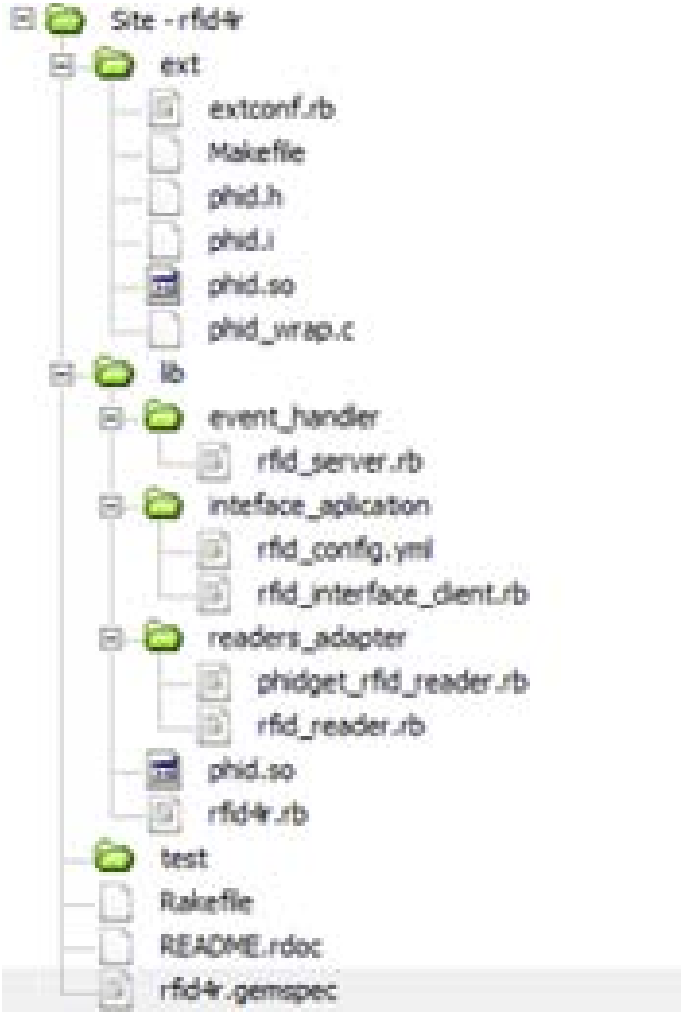
Árbol de archivos del gem

Las convenciones de RubyGems exigen que todo gem esté estructurado con subcarpetas específicas.

Los archivos fuente deben estar en una carpeta con el nombre *lib*

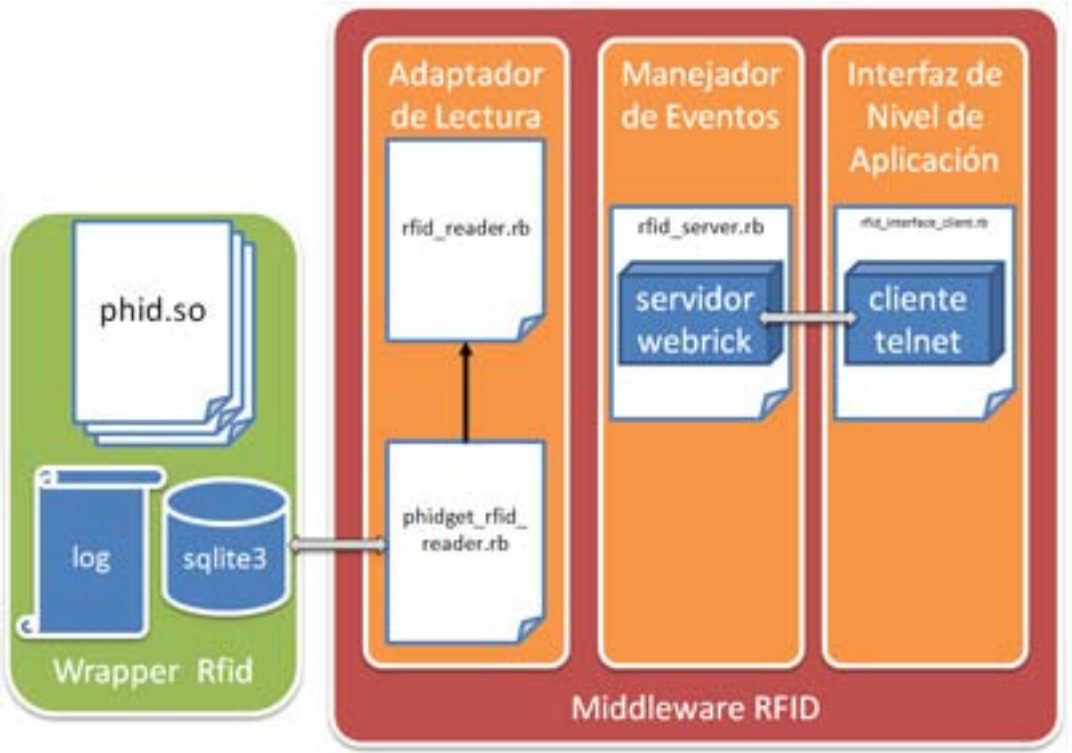
Si el gem contiene extensiones, deben estar en una carpeta *ext*

Otras consideraciones contemplan la adecuada documentación, manuales o instructivos de uso, etc.



Estructura del gem

Esta estructura muestra de manera sencilla en qué componente del middleware se pueden ubicar los módulos y clases de Ruby que tiene el gem.



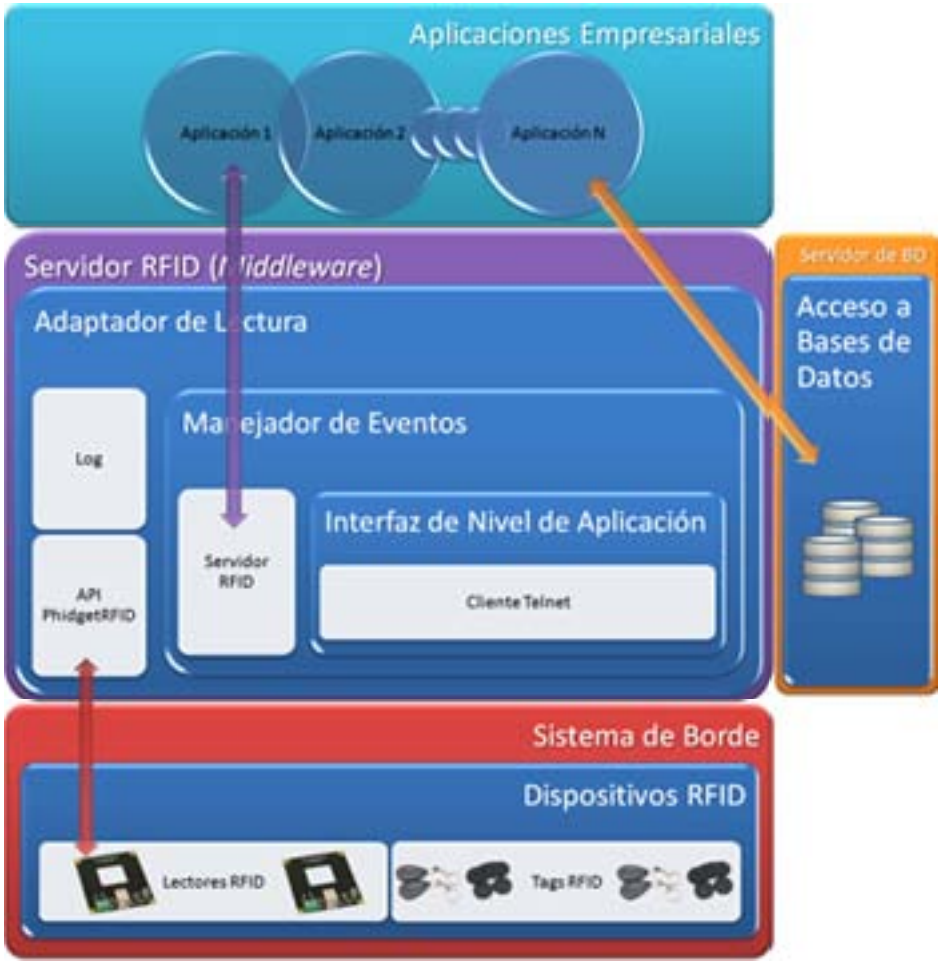
Desarrollo

- **Iteración 0:** se desarrolló la extensión (wrapper) de Ruby para administrar el dispositivo PhidgetRFID-1023
- **Iteración 1:** se agregaron nuevas funcionalidades al wrapper y se creó la clase de Ruby que utiliza la extensión (desarrollo del adaptador de lectura)
- **Iteración 2 (continuación de it. 1):** se agregó una implementación de un log con una base de datos *Sqlite* y con un archivo de texto plano
- **Iteración 3 (continuación de it. 1):** se desarrolló la estructura definitiva del wrapper para establecer el estándar a seguir
- **Iteración 4:** se desarrolló un servidor sencillo basado en comandos (desarrollo del manejador de eventos)
- **Iteración 5:** se desarrolló un cliente para establecer comunicación con otros servidores (desarrollo de la interfaz de nivel de aplicación)
- **Iteración 6:** se empaquetó la aplicación en un gem de Ruby y se importó a su repositorio de RubyForge
- **Iteración 7:** se hicieron mejoras, correcciones y otras modificaciones del gem y la extensión de Ruby

Arquitectura del gem

Partiendo de la configuración del middleware RFID se pueden identificar aisladamente los componentes del sistema RFID completo.

Dependiendo de las posibles peticiones que se puedan presentar por parte de las aplicaciones, el servidor RFID debe estar en la capacidad de comunicarse con otros servidores RFID.



CONCLUSIONES

Conclusiones

- El resultado fue un producto de software llamado *rfid4r*, destinado a proveer un middleware para sistemas RFID
- Se desarrolló utilizando el lenguaje Ruby y el lenguaje C para la elaboración de una extensión de Ruby para el manejo del dispositivo PhidgetRFID-1023
- El producto fue empaquetado en la forma de un *gem* de Ruby
- La implementación se hizo utilizando el proceso de desarrollo XP y la metodología AM
- El gem está implementado utilizando la propuesta de arquitectura cliente-servidor

Conclusiones

- El servidor define un protocolo basado en comandos simples y las respuestas son enviadas en formato de texto plano
- Las pruebas se hicieron de manera convencional (manualmente)
- El desarrollo estuvo enfocado en la plataforma Linux
- El proceso de adaptación para otras plataformas es muy similar
- El gem está publicado en el repositorio de RubyForge (http://rfid4r.rubyforge.org/)
- Los objetivos planteados para el desarrollo de este trabajo fueron alcanzados

Mejoras a Futuro

- Implementar el gem contemplando el soporte para las plataformas Windows y Mac
- Agregar soporte para más dispositivos lectores-escritores, etiquetas activas y futuras generaciones de dispositivos y etiquetas
- Agregar soporte para servicios web
- Implementar módulos usando técnicas avanzadas de programación como Comet
- Mejores interfaces de comunicación implementando nuevos protocolos, comandos y agregando políticas y niveles de seguridad

Recomendaciones

- Revisar la documentación *ri* y *rdoc* para entender el funcionamiento del gem
- Editar el archivo *rfid_config.yml* para configurar tantos servidores como sean necesarios
- Es necesario instanciar un servidor RFID por cada lector que se quiera utilizar
- Mantener al día, desarrollar pruebas constantemente, actualizar y mejorar el gem desde su repositorio http://rfid4r.rubyforge.org

Recomendaciones (aplicaciones externas)

- Realizar operaciones con bases de datos o repositorios LDAP para relacionar o asociar datos e información
- Generación de archivos XML y reportes en PDF
- Integración con APIs de GoogleMaps para implementar sistemas de posicionamiento en tiempo real
- Generación de gráficas y estadísticas

Aportes de la Investigación

- La implementación y adaptación de metodologías y procesos de desarrollo como XP y AM para la implementación de un producto de software
- El desarrollo de una extensión de Ruby utilizando SWIG y el módulo *mkmf*
- El desarrollo de un programa Cliente-Servidor sencillo en Ruby
- El desarrollo de un middleware RFID para administrar dispositivos y hacer consultas
- El desarrollo de un gem de Ruby

Referencias

- Beck, K. (1999). Extreme Programming Explained. Addison-Wesley Professional
- agilemodeling.com (2008). Agile modeling (am) home page effective practices for modeling and documentation. http://www.agilemodeling.com
- Glover, B. and Bhatt, H. (2006). RFID Essentials. O'Reilly Media
- Ilyas, S. A. M. (2008). RFID Handbook. CRC Press - Taylor & Francis Group
- Thomas, D., Fowler, C., and Hunt, A. (2005). Programming Ruby: The Pragmatic Programmers Guide. The Pragmatic Bookshelf
- Berube, D. (2007). Practical Ruby Gems. Apress

PREGUNTAS

FIN DE LA PRESENTACIÓN