PREDESYS

Documentación del proyecto Predesys (versión 2)

Cubre las siguientes versiones de los componentes:
Predesys Server: 0.1.0
Predesys Service API: 0.1.0
Predesys Radar: 0.1.0

José Antonio Jiménez Carmona

5 de marzo de 2011

Índice general

Capítulo 1

Acerca del proyecto

1.1. Autoría

El autor de este proyecto es José Antonio Jiménez Carmona (el cual posee los derechos de autor), alumno de Ingeniería Técnica en Informática de Gestión en la Escuela Técnica Superior de Ingeniería Informática de la Universidad de Sevilla. Este proyecto conforma el proyecto Fin de Carrera del autor y tiene como tutor al profesor Pablo Neira Ayuso, del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla.

Puede contactar con el autor del proyecto por correo electrónico en la siguiente dirección:

`josantjim@gmail.com`

1.2. Licencia de este documento

Este documento se rige por la licencia *Reconocimiento-CompartirIgual 3.0* de Creative Commons. Esta licencia exige que en cualquier explotación de la obra hará falta reconocer la autoría y permite la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas. Puede consultar el contenido completo de la licencia en la siguiente dirección web:

`http://creativecommons.org/licenses/by-sa/3.0/es/legalcode.es`

1.3. Licencia del proyecto

El proyecto Predesys se rige por la licencia *GNU Lesser General Public License 3.0*. El contenido completo de la licencia (en inglés) está disponible en la siguiente dirección:

`http://www.gnu.org/licenses/lgpl-3.0-standalone.html`

Capítulo 2

Introducción

2.1. El porqué

La creación de este proyecto viene motivado por, además de la obligación de realizar un proyecto de final de carrera en mi titulación, mis inquietudes sobre algunas tecnologías de comunicación (como el bluetooth o la web), mis inquietudes sobre cómo integrar distintas plataformas informáticas (PCs, dispositivos móviles, microcontroladores, etc) y la conveniencia de tener un sistema de localización de personal en algunas organizaciones (como pueden ser las distintas asociaciones de estudiantes de la Escuela Técnica Superior de Ingeniería Informática de la Universidad de Sevilla o cualquier empresa).

2.2. De qué se trata

Predesys es un sistema software contenido en un servidor central y en un conjunto de servidores secundarios con el objetivo de detectar la presencia de dispositivos bluetooth (principalmente, teléfonos móviles) previamente registrados que se encuentren cerca físicamente de dichos servidores secundarios para ofrecer una serie de servicios de información a los usuarios de dichos dispositivos y a terceros. El sistema está pensado para implantarse en oficinas y otros tipos de instalaciones físicas. Está diseñado para los sistemas operativos Linux y es software libre (su licencia es LGPL 3.0).

2.3. Objetivos generales

Los objetivos principales del proyecto son:

1. Disponer de una o más máquinas capaces de detectar un dispositivo bluetooth concreto y realizar alguna tarea (servicio) asociada a dicha detección.

2. Disponer de un sistema de plugins para que terceros puedan programar tareas (servicios) adicionales para el sistema.

2.4. Trabajos relacionados

PowerWarning

PowerWarning es un detector de presencia de personas en una determinada sala. Consiste en utilizar un teléfono móvil que detecta cuándo unas puertas han sido abiertas y envía un SMS a otro teléfono para avisar a alguien de la intrusión.

Sitio web: `http://code.google.com/p/powerwarning`

BlueHoc

BlueHoc es un simulador de protolos de comunicación realizado por IBM. Permite que las aplicaciones software puedan enviar información a través de bluetooth creyendo que la están enviando por otro protocolo, como TCP/IP.

Sitio web: `http://bluehoc.sourceforge.net`

Capítulo 3

Planificación

3.1. Estimación

El proyecto consta de varios componentes bien diferenciados (ver sección *Arquitectura* para más información). La estimación del tiempo empleado necesario para realizar cada una de las partes del proyecto es la siguiente:

- **Diseñar la arquitectura del proyecto a un nivel esquemático:**
  Requiere definir qué componentes necesita el proyecto y cómo se comunicarán entre ellos.

  Tiempo estimado: 2 semanas

- **Completar las implementaciones de las primeras versiones del Servidor y de la API de Servicios:**
  Esto implica el diseño de cómo debe accederse a los servicios que ofrezca el Servidor, el diseño del sistema de plugins para que terceros puedan programar servicios adicionales, y el sistema de autentificación para determinados servicios que requieran la identificación del usuario.

  Tiempo estimado: 2 meses

- **Completar la implementación de la primera versión del Radar:**
  Esta parte del proyecto sólo requiere los conocimientos de las partes anteriores, por lo que el tiempo para realizarla no debería ser mucho.

  Tiempo estimado: 1 semana

3.2. Tiempo empleado

- **Diseñar la arquitectura del proyecto a un nivel esquemático:**
  Tiempo empleado: 2 semanas

  El tiempo empleado ha sido el previsto.

- **Completar las implementaciones de las primeras versiones del Servidor y de la API de Servicios:**
  Tiempo empleado: 3 meses

  Se ha superado el tiempo estimado inicialmente en un 50 %, debido a problemas surgidos en la investigación sobre la distribución de aplicaciones para Linux. Estos problemas han

sido mi desconocimiento casi total de cómo debe organizarse el código fuente del proyecto y cómo debe distribuirse el código fuente a desarrolladores y el programa final a los usuarios. Para la distribución del programa a los usuarios, ha sido necesario investigar acerca de cómo se crean los *paquetes Debian* (archivos que contienen un programa listo para instalar), para lo cual, la documentación existente en Internet es bastante confusa.

- **Completar la implementación de la primera versión del Radar:**
  Tiempo empleado: 1 semana

  El tiempo empleado ha sido el previsto.

Capítulo 4

Elicitación de requisitos

4.1. Objetivos específicos

4.2. Casos de uso

Capítulo 5

Arquitectura

5.1. Sistemas distribuidos

Según Wikipedia (http://es.wikipedia.org):

Un sistema distribuido se define como una colección de computadoras separadas físicamente y conectadas entre sí por una red de comunicaciones distribuida; cada máquina posee sus componentes de hardware y software que el usuario percibe como un solo sistema (no necesita saber qué cosas están en qué máquinas). El usuario accede a los recursos remotos (RPC) de la misma manera en que accede a recursos locales, o un grupo de computadores que usan un software para conseguir un objetivo en común.

Los sistemas distribuidos deben ser muy confiables, ya que si un componente del sistema se descompone otro componente debe de ser capaz de reemplazarlo, esto se denomina Tolerancia a Fallos.

El tamaño de un sistema distribuido puede ser muy variado, ya sean decenas de hosts (red de área local), centenas de hosts (red de área metropolitana), y miles o millones de hosts (Internet); esto se denomina escalabilidad.

La computación distribuida ha sido diseñada para resolver problemas demasiado grandes para cualquier supercomputadora y mainframe, mientras se mantiene la flexibilidad de trabajar en múltiples problemas más pequeños. Por lo tanto, la computación en grid es naturalmente un entorno multiusuario; por ello, las técnicas de autorización segura son esenciales antes de permitir que los recursos informáticos sean controlados por usuarios remotos.

5.2. Arquitectura Cliente-Servidor

Según Wikipedia (http://es.wikipedia.org):

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño

del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

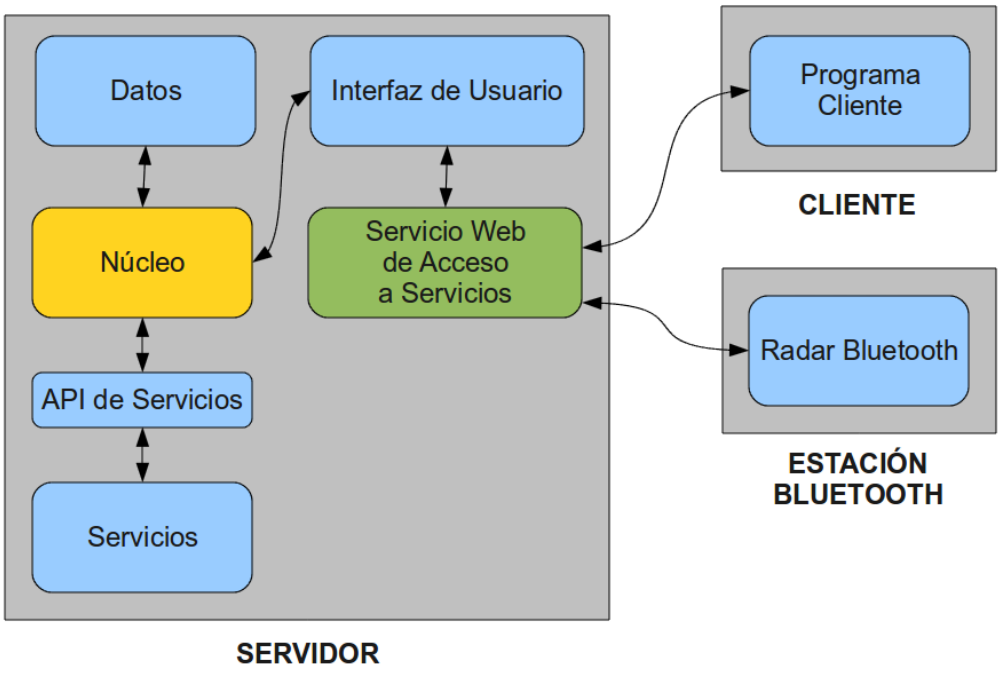
Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

La red cliente-servidor es aquella red de comunicaciones en la que todos los clientes están conectados a un servidor, en el que se centralizan los diversos recursos y aplicaciones con que se cuenta y que los pone a disposición de los clientes cada vez que estos son solicitados. Esto significa que todas las gestiones que se realizan se concentran en el servidor, de manera que en él se disponen los requerimientos provenientes de los clientes que tienen prioridad, los archivos que son de uso público y los que son de uso restringido, los archivos que son de sólo lectura y los que, por el contrario, pueden ser modificados, etc. Este tipo de red puede utilizarse conjuntamente en caso de que se este utilizando en una red mixta.

5.3. Arquitectura de Predesys

A continuación, se expone una introducción a la arquitectura del proyecto:



Como puede verse en la figura, el soporte físico para Predesys consta de un Servidor (un PC), 1 ó más Estaciones Bluetooth (PCs) y 1 ó más Clientes (PCs o dispositivos móviles).

Predesys es un sistema que detecta dispositivos bluetooth por el Radar y ejecuta tareas automáticas cuando se detectan dichos dispositivos y otras tareas que son ejecutables por los usuarios mediante el Servicio Web de Acceso a Servicios. Estas tareas se definen por medio de "servicios", que son plugins que tiene instalados el Servidor.

Globalmente, hay 2 tipos de servicios, los servicios llamables y los automáticos. Los servicios automáticos sólo los ejecuta el Radar, que lo hace cada cierto tiempo y los llamables los ejecuta el usuario (el programa Cliente) cuando quiera. Todos los servicios se ejecutan por medio del Servicio Web de Acceso a Servicios y pueden requerir autentificación por medio de un nombre de usuario y una contraseña (en el caso de los servicios automáticos se requiere autentificación ya que sólo una máquina Estación debería poder ejecutarlos).

Los servicios automáticos pueden, además de realizar alguna tarea, al finalizar ésta, enviar una orden al Radar para que envíe algún mensaje con cualquier información por bluetooth a los dispositivos bluetooth detectados.

El Servidor contiene toda la lógica de los servicios y los datos de los mismos. Todo el control del Servidor lo ejerce el Núcleo, que es el componente principal, y es el que ejecuta los servicios (que son scripts que realizan una determinada tarea). Los Servicios pueden acceder a los Datos pero no pueden modificarlos (si requirieran guardar datos, deberían gestionar su propia base de datos). El Núcleo es el intermediario entre los Servicios y los Datos, ya que debe controlar y asegurar el correcto funcionamiento del sistema y limitar los datos a los que pueda acceder cada parte del sistema. La forma en la que los Servicios acceden a los Datos y a cualquier funcionalidad del sistema es a través de la API de Servicios, que provee varias funciones limitadas que acceden al Núcleo.

Las estaciones bluetooth pueden ser 1 ó más, todas con conectividad por TCP/IP con el servidor. Tienen tan sólo 1 función muy básica aunque muy importante. Esta función es detectar periódicamente la presencia de dispositivos bluetooth que estén físicamente cerca de la Estación Bluetooth y enviar sus direcciones MAC al Servidor, a través de uno de los Servicios (en este caso, un Servicio especial de acceso restringido a las Estaciones Bluetooth) para ejecutar a su vez todos los servicios automáticos que estén instalados en el Servidor y, en el caso de que alguno lo especifique, enviar mensajes por bluetooth a los dispositivos móviles.

Capítulo 6

Implementación

Capítulo 7

Distribución

Los tres componentes de Predesys (Servidor, API de Servicios y Radar) se distribuyen en 2 archivos de formatos distintos por cada uno de ellos. Los formatos son el formato *tarball* (con extensión *.tar.gz*) y formato de *paquete Debian* (con extensión *.deb*).

El *tarball* es un archivo comprimido que contiene el programa junto a un script de instalación. Tiene la ventaja de que puede instalarse en cualquier sistema Linux pero tiene la desventaja de no instalar las dependencias que no estén instaladas en el sistema, es decir, el software necesario para hacer funcionar al programa.

El *paquete Debian* es un archivo que contiene el programa listo para instalar en cualquier sistema operativo Linux tipo Debian (por ejemplo, el propio Debian o Ubuntu). Tiene la ventaja de que es muy fácil de instalar y comprueba todas las dependencias (descargándolas e instalándolas automáticamente si el sistema no las tuviera instaladas) pero tiene la desventaja de que, en principio, sólo puede instalarse en los sistemas operativos Linux de tipo Debian.

7.1. Servidor

Dependencias

El Servidor requiere tener los siguientes paquetes de software instalados en el sistema (los nombres son los establecidos en los sistemas Linux tipo Debian, en otros sistemas pueden ser distintos) para poder ejecutarse:

- python (versión igual o superior a la 2.6.6)
- python-libxml2 (versión igual o superior a la 2.7.7)
- python-sqlalchemy (versión igual o superior a la 0.6.3)
- python-mysqldb (versión igual o superior a la 1.2.2)
- sysv-rc (versión igual o superior a la 2.87)

En el caso de que quiera instalar el Servidor desde el archivo *tarball*, necesitará, para poder instalarlo, el siguiente paquete:

- make (versión igual o superior a la 3.81)

Generación del archivo *tarball*

Partiendo del directorio raíz del código fuente del repositorio del proyecto, navegue hasta el directorio *predesys-server/scripts* y ejecute el siguiente script:

```
generate-tarball.sh
```

Y, en el directorio generado *output*, apareceré el archivo *tar.gz* resultante, el cual estará listo para distribuirse e instalarse.

Generación del archivo *paquete de Debian*

Partiendo del directorio raíz del código fuente del repositorio del proyecto, navegue hasta el directorio *predesys-server/scripts* y ejecute el siguiente script:

```
generate-deb.sh
```

Y, en el directorio generado *output*, apareceré el archivo *.deb* resultante, el cual estará listo para distribuirse e instalarse.

7.2. API de Servicios

Dependencias

La API de Servicios requiere tener los siguientes paquetes de software instalados en el sistema (los nombres son los establecidos en los sistemas Linux tipo Debian, en otros sistemas pueden ser distintos) para poder ejecutarse:

- predesys-server (versión igual o superior a la 0.1.0) - Se trata del Servidor de Predesys
- python (versión igual o superior a la 2.6.6)
- python-setuptools (versión igual o superior a la 0.6.14)

En el caso de que quiera instalar la API de Servicios desde el archivo *tarball*, necesitará, para poder instalarlo, el siguiente paquete:

- make (versión igual o superior a la 3.81)

Generación del archivo *tarball*

Partiendo del directorio raíz del código fuente del repositorio del proyecto, navegue hasta el directorio *predesys-service-api/scripts* y ejecute el siguiente script:

```
generate-tarball.sh
```

Y, en el directorio generado *output*, apareceré el archivo *tar.gz* resultante, el cual estará listo para distribuirse e instalarse.

Generación del archivo *paquete de Debian*

Partiendo del directorio raíz del código fuente del repositorio del proyecto, navegue hasta el directorio *predesys-service-api/scripts* y ejecute el siguiente script:

```
generate-deb.sh
```

Y, en el directorio generado *output*, apareceré el archivo *.deb* resultante, el cual estará listo para distribuirse e instalarse.

7.3. Radar

Dependencias

El Radar requiere tener los siguientes paquetes de software instalados en el sistema (los nombres son los establecidos en los sistemas Linux tipo Debian, en otros sistemas pueden ser distintos) para poder ejecutarse:

- python (versión igual o superior a la 2.6.6)
- python-libxml2 (versión igual o superior a la 2.7.7)
- python-bluez (versión igual o superior a la 0.18)
- python-lightblue (versión igual o superior a la 0.3.2)
- sysv-rc (versión igual o superior a la 2.87)

En el caso de que quiera instalar el Radar desde el archivo *tarball*, necesitará, para poder instalarlo, el siguiente paquete:

- make (versión igual o superior a la 3.81)

Generación del archivo *tarball*

Partiendo del directorio raíz del código fuente del repositorio del proyecto, navegue hasta el directorio *predesys-radar/scripts* y ejecute el siguiente script:

```
generate-tarball.sh
```

Y, en el directorio generado *output*, apareceré el archivo *tar.gz* resultante, el cual estará listo para distribuirse e instalarse.

Generación del archivo *paquete de Debian*

Partiendo del directorio raíz del código fuente del repositorio del proyecto, navegue hasta el directorio *predesys-radar/scripts* y ejecute el siguiente script:

```
generate-deb.sh
```

Y, en el directorio generado *output*, apareceré el archivo *.deb* resultante, el cual estará listo para distribuirse e instalarse.

Capítulo 8

Instalación

Los 3 componentes de Predesys (Servidor, API de Servicios y Radar) se pueden instalar cada uno bien mediante el archivo *tarball* o bien mediante el archivo *paquete de Debian*. El Servidor y la API de Servicios deben ser instalados en la misma máquina; el Radar puede instalarse en la misma máquina o en otra distinta (requiere tener un dispositivo bluetooth).

Después de instalar el Servidor, es necesario configurarlo mediante su interfaz de usuario.

8.1. Servidor

Instalación mediante el archivo de distribución *tarball*

Descomprima el archivo *.tar.gz* en cualquier directorio y desde el directorio extraído del archivo comprimido (que contiene el programa), ejecute como administrador la siguiente orden:

```
make install
```

El Servidor quedará instalado. Su Servicio Web de Acceso a Servicios quedará instalado como un servicio del sistema (que se arranca en cada inicio del sistema) y estará ejecutándose.

Si quisiera desinstalarlo, debería ejecutar como administrador la siguiente orden también desde el directorio extraído del archivo comprimido:

```
make uninstall
```

Para parar, iniciar de nuevo o reiniciar el servicio web, ejecute como administrador la siguiente orden (da igual el directorio donde se encuentre):

```
service predesys-server-webservice [opción]
```

Donde `[opción]` debe ser `stop`, `start` o `restart`, respectivamente.

Una vez instalado el Servidor, éste debe configurarse para establecer los parámetros de la base de datos a utilizar (ver capítulo *Uso y Configuración*). Después de hacer esto, el Servidor estará listo.

Instalación mediante el archivo de distribución *paquete Debian*

Desde el directorio donde esté el archivo *.deb*, ejecute como administrador la siguiente orden:

```
dpkg -i [NombreArchivo]
```

Donde `[NombreArchivo]` es el nombre del archivo *.deb*. El Servidor quedará instalado. Su Servicio Web de Acceso a Servicios quedará instalado como un servicio del sistema (que se arranca en cada inicio del sistema) y estará ejecutándose.

Si quisiera desinstalarlo, debería ejecutar como administrador la siguiente orden (da igual el directorio donde se encuentre):

```
apt-get purge predesys-server
```

Para parar, iniciar de nuevo o reiniciar el servicio web, ejecute como administrador la siguiente orden (da igual el directorio donde se encuentre):

```
service predesys-server-webservice [opción]
```

Donde `[opción]` debe ser `stop`, `start` o `restart`, respectivamente.

Una vez instalado el Servidor, éste debe configurarse para establecer los parámetros de la base de datos a utilizar (ver capítulo *Uso y Configuración*). Después de hacer esto, el Servidor estará listo.

8.2. API de Servicios

Instalación mediante el archivo de distribución *tarball*

Descomprima el archivo *.tar.gz* en cualquier directorio y desde el directorio extraído del archivo comprimido (que contiene el programa), ejecute como administrador la siguiente orden:

```
make install
```

La API de Servicios quedará instalada. El módulo Python instalado se llama *predesys*.

Si quisiera desinstalarlo, debería ejecutar como administrador la siguiente orden también desde el directorio extraído del archivo comprimido:

```
make uninstall
```

Instalación mediante el archivo de distribución *paquete Debian*

Desde el directorio donde esté el archivo *.deb*, ejecute como administrador la siguiente orden:

```
dpkg -i [NombreArchivo]
```

Donde `[NombreArchivo]` es el nombre del archivo *.deb*. La API de Servicios quedará instalada. El módulo Python instalado se llama *predesys*.

Si quisiera desinstalarlo, debería ejecutar como administrador la siguiente orden (da igual el directorio donde se encuentre):

```
apt-get purge predesys-server
```

8.3. Radar

Instalación mediante el archivo de distribución *tarball*

Descomprima el archivo *.tar.gz* en cualquier directorio y desde el directorio extraído del archivo comprimido (que contiene el programa), ejecute como administrador la siguiente orden:

```
make install
```

El Radar quedará instalado. Este componente quedará instalado como un servicio del sistema (que se arranca en cada inicio del sistema) y estará ejecutándose.

Si quisiera desinstalarlo, debería ejecutar como administrador la siguiente orden también desde el directorio extraído del archivo comprimido:

```
make uninstall
```

Para parar, iniciar de nuevo o reiniciar el servicio, ejecute como administrador la siguiente orden (da igual el directorio donde se encuentre):

```
service predesys-radar [opción]
```

Donde `[opción]` debe ser `stop`, `start` o `restart`, respectivamente.

Instalación mediante el archivo de distribución *paquete Debian*

Desde el directorio donde esté el archivo *.deb*, ejecute como administrador la siguiente orden:

```
dpkg -i [NombreArchivo]
```

Donde `[NombreArchivo]` es el nombre del archivo *.deb*. El Radar quedará instalado. Este componente quedará instalado como un servicio del sistema (que se arranca en cada inicio del sistema) y estará ejecutándose.

Si quisiera desinstalarlo, debería ejecutar como administrador la siguiente orden (da igual el directorio donde se encuentre):

```
apt-get purge predesys-radar
```

Para parar, iniciar de nuevo o reiniciar el servicio, ejecute como administrador la siguiente orden (da igual el directorio donde se encuentre):

```
service predesys-radar [opción]
```

Donde `[opción]` debe ser `stop`, `start` o `restart`, respectivamente.

Capítulo 9

Uso y configuración

Capítulo 10

Desarrollo de servicios