Sistemas Informáticos
Curso 2006/07

Localización de dispositivos móviles en interiores usando redes Wireless

Adolfo González Blázquez
Pablo Pedro Mulas Gómez
Rafael Rivera Retamar

Dirigido por:
Prof. Manuel Ortega Ortíz de Apodaca
Dpto. Ingeniería del Software e Inteligencia Artificial

Facultad de Informática
Universidad Complutense de Madrid

ÍNDICE DE CONTENIDO

1. Palabras clave....3

2. Resumen del proyecto....4
2.1. Resumen....4
2.2. Abstract....4

3. Introducción....5

4. Java en PDAs y dispositivos móviles....7
4.1. Java en Microsoft Windows XP....7
4.2. Java en Microsoft Windows Mobile / Pocket PC....7
4.3. Java en sistemas GNU/Linux....8
4.4. Java en GNU/Linux para dispositivos móviles como PDAs....9

5. Herramientas de configuración de tarjetas wireless....10
5.1. Herramientas wireless en sistemas GNU/Linux....10
5.2. Herramientas wireless en sistemas Microsoft Windows....12

6. Sistemas Operativos en PDAs....14
6.1. Windows Mobile en una PDA....14
6.2. GNU/Linux en una PDA....15

7. Modelos de localización....17
7.1. Localización usando RTT mediante llamadas Ping....17
7.1.1. Proceso de localización....18
7.1.2. Resultados de la investigación en este modelo....20
7.2. Localización usando fuerzas de señal (Fingerprinting)....21
7.2.1. Proceso de localización....21
7.2.2. Resultados de la investigación en este modelo....24
7.3. Localización con el tiempo de respuesta desde la capa de enlace de datos 802.11....26

8. Algoritmos para la localización....27
8.1. Algoritmo de k-ésimos vecinos más cercanos (k-closest neighbors)....27
8.2. Algoritmo de filtrado por caminos....29

9. Aplicaciones prácticas de la localización....30

10. Software implementado....31
10.1. Indoor Location – Aplicación de fingerprinting y geolocalización....31
10.2. Escáner de red para Windows en modo consola – WifiScanner.exe....35
10.3. Editor de mapas - Map_Editor....37
10.3.1. Implementación de Map_Editor....37
10.3.2. Posibles mejoras en Map_Editor....41

11. Resultados del proyecto....43

Apéndice 1. Instalar GNU/Linux en una PDA....46
Apéndice 2. Diagramas UML de la aplicación Indoor Location....52
Apéndice 3. Código fuente de la aplicación Indoor Location....60
Glosario....162
Referencias....165
Agradecimientos....168

AUTORIZACIÓN

Por la presente, los autores de este proyecto autorizan a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Adolfo González Blázquez

Pablo Pedro Mulas Gómez

Rafael Rivera Retamar

1. PALABRAS CLAVE

- Localización
- Wireless
- PDA
- Escáner de red
- Fuerza señal
- Fingerprinting
- K-ésimos vecinos
- Voronoi
- Punto de acceso
- Editor de mapas

2. RESUMEN DEL PROYECTO

2.1. Resumen

El objetivo de este proyecto es desarrollar un sistema software capaz de localizar dispositivos móviles en entornos interiores usando como tecnología de localización redes Wireless 802.11.

Para implementar este sistema, se llevó a cabo una fase de investigación de posibles alternativas de desarrollo, que incluyó el estudio de posibles sistemas operativos y entornos de desarrollo de software para dispositivos móviles, tales como PDAs, ordenadores portátiles, etc. También se investigaron distintos enfoques tecnológicos de localización en interiores, sopesando sobre todo, la facilidad y mínimo coste de implantación de las distintas alternativas, así como su eficacia en distintos entornos.

La segunda fase consistió en la implementación del software que realiza la localización, así como diversas aplicaciones de ayuda, como un editor de mapas, para usarlos en la aplicación principal, un gestor de dispositivos, y un visor para mostrar la información obtenida, y herramientas de escaneo de redes wireless.

Todo el software del proyecto ha sido desarrollado con vistas a futuras extensiones del sistema, de modo que pueda ser aplicado a diferentes ámbitos donde la localización podría ser útil, como hospitales, almacenes, etc.

2.2. Abstract

The scope of this project is to develop a software system able to locate mobile devices in indoor environments using Wireless 802.11 networks.

To implement this system, a phase of investigation of possible alternatives of development was carried out, that included the study of possible operating systems and development environments for mobile devices, such as PDAs, laptops, etc. We also investigated different technological approaches to indoor location, hefting mainly, the facility and minimum cost of implantation of the different alternatives, as well as its effectiveness in different environments.

The second phase consisted of the implementation of location software, as well as diverse helper applications, like a map editor, which produces maps to be used with the main application, a device manager, and a viewer to show the obtained data, and tools for wireless networks scanning.

All the software of the project has been developed thinking of future extensions of the system, so it can be applied to different environments where the indoor location could be useful, like hospitals, warehouses, etc.

3. INTRODUCCIÓN

Actualmente, el uso de tecnologías de localización se ha introducido en la vida diaria de la mayoría de las personas. Los sistemas GPS son cada vez más comunes entre la gente corriente, y cada vez más empresas depositan gran parte de su responsabilidad y eficacia en este tipo de sistemas, como por ejemplo las empresas de transportes, que necesitan tener constancia en tiempo real de donde están sus envíos.

La mayoría de estas aplicaciones están enfocadas a la localización en espacios abiertos, usando satélites. Pero cada vez hay un mayor interés en implantar servicios de localización allá donde los satélites no tienen cobertura, como ocurre por ejemplo en interiores de edificios o en sistemas subterráneos.

La creciente implantación de tecnologías inalámbricas, así como su cada vez más bajo coste, favorece la idea de un sistema de localización basado en este tipo de dispositivos. De entre las opciones disponibles, las redes Wireless 802.11bg son las que tienen mayor implantación hoy en día, y las que tienen un coste más reducido.

Prácticamente todos lo edificios de empresas disponen hoy en día de una red inalámbrica disponible ya instalada, lo que hace la implantación de un servicio de localización aprovechando la infraestructura existente barato y altamente funcional. Hospitales, almacenes, incluso restaurantes se beneficiarían de conocer la posición de sus trabajadores y recursos.

Además, cualquier dispositivo con acceso a una red wireless podría ser localizado en cualquier momento, como ordenadores portátiles, agendas electrónicas (PDA), teléfonos móviles, Tablet PCs, etc. Todos ellos cuentan hoy en día con tarjetas de red wireless, lo que les hace factibles para su uso en sistemas de localización.

Esto quiere decir que cualquier persona que use, por ejemplo, una PDA dentro de un hospital, puede ser localizado en tiempo real, sin usar caros sistemas GPS, y ofrecerle servicios que se beneficien directamente de conocer la posición del usuario.

Por ejemplo, supongamos que un doctor de un hospital está visitando a un paciente. Necesita consultar una radiografía almacenada en la base de datos del hospital, así que usando su PDA se conecta al servidor central, y le pide que le muestre la placa del paciente en la pantalla más cercana a su posición, así que el sistema le localiza dentro del edificio, y se muestra la información pedida en la pantalla del pasillo más cercana.

Se ha investigado en diferentes tipos de estrategias de localización, diferenciadas básicamente por la complejidad de implantación de cada una.

La más sencilla es la localización del dispositivo teniendo en cuenta el tiempo que tarda un paquete en viajar desde el cliente al servidor y su tiempo de vuelta. Esta estrategia se basa en enviar Pings del cliente al servidor. Este es el método más barato y más sencillo de implementar, pero también el más impreciso, y además es el menos realista en la implantación práctica, puesto que las suposiciones teóricas de este modelo son inaceptables en un entorno real.

Otra alternativa es la localización basada en la fuerza de la señal recibida en un momento dado en el dispositivo móvil. Este sistema tiene un grado de error bastante bajo, y un coste de implantación muy reducido, lo que le convierte en un modelo óptimo para entornos donde la localización no sea un factor crítico en cuanto al tiempo, pues este sistema introduce un pequeño retardo en la localización, debido a las limitaciones propias del hardware Wireless.

La otra alternativa estudiada es una modificación de la primera propuesta, basada en el tiempo de respuesta de paquetes de red, pero implementada a nivel hardware, es decir, modificando las tarjetas Wireless para hacer las mediciones de tiempo. Este sistema tiene una precisión mayor en la posición de localización y en el tiempo de respuesta, pero su coste de implantación, debido a las modificaciones hardware necesarias, lo hace menos apropiado para sistemas donde la localización sólo es un complemento.

En este proyecto se desarrollará el segundo modelo, después de haber constatado las limitaciones del primero, al haber intentado implantarlo en un entorno real, y debido a limitaciones de presupuesto y tiempo para implantar la tercera opción. Además, el estudio de las distintas aproximaciones al problema nos condujo a darnos cuenta de los mayores beneficios de sistemas de localización por fuerzas de señal, puesto que el hardware necesario está disponible en la mayoría de entornos, y los algoritmos necesarios para la localización no son costosos en tiempo, ni en requerimientos de cómputo, lo que les hace ideales para ejecutarse en dispositivos con recursos limitados, como PDAs o teléfonos móviles.

Durante el proceso de estudio de estos modelos de localización, investigamos distintas alternativas de sistemas operativos y entornos software de desarrollo en dispositivos móviles, llegando a la conclusión de que, hoy en día, el sistema operativo más adecuado para soportar este tipo de aplicaciones es cualquiera basado en GNU/Linux, y que el entorno software más idóneo es Java, debido a su sencilla portabilidad y rendimiento.

4. JAVA EN PDAS Y DISPOSITIVOS MÓVILES

En la actualidad la mayoría de los dispositivos portátiles, como agendas personales (PDAs), teléfonos móviles, y ordenadores portátiles son capaces de ejecutar aplicaciones Java, independientemente del sistema operativo que ejecuten dichos dispositivos, lo que nos llevo a elegirlo como entorno de desarrollo de software.

Durante el desarrollo del proyecto investigamos en las distintas opciones de máquinas virtuales Java disponibles para sistemas operativos ejecutables en dispositivos portátiles, que explicaremos a continuación.

4.1. Java en Microsoft Windows XP

En el sistema operativo principal de Microsoft la ejecución de Java no supone ningún problema. Existen diversos entornos, libres y no libres, para desarrollar software en este lenguaje.

El entorno elegido fue el Java 2 Standard Edition de Sun, debido a su gratuidad, y a que se ha convertido en un estándar de facto en el mundo del desarrollo de software.

4.2. Java en Microsoft Windows Mobile / Pocket PC

Windows Mobile es el sistema operativo que lleva instalado por defecto la mayoría de las PDAs del mercado. A pesar de ello, el estado actual de las máquinas virtuales Java para el sistema operativo móvil de Microsoft no está tan desarrollado como era de esperar.

Las alternativas libres y gratuitas escasean, siendo generalmente necesario adquirir una licencia para obtener una máquina virtual Java, que proporcione un entorno adecuado para las aplicaciones necesarias del proyecto.

La alternativas que probamos fueron:

- **IBM WebSphere Everyplace Micro Environment**. IBM ofrece un entorno completo de programación para dispositivos móviles, incluyendo un IDE, librerías propias, y una máquina virtual para PDAs. El precio de la versión completa de desarrollo es superior a 500$, mientras que la máquina virtual, llamada **J9**, tiene un coste por licencia de alrededor de 10$.

Este entorno es compatible con Java 2 Micro Edition, y soporta la mayoría de configuraciones estándar, como CDC, MIDP, CLDC, Personal Profile y Foundation Profile. Además, está disponible en varios sistemas operativos, como Windows XP, Windows Mobile, Linux y Palm OS.

Algunos fabricantes, como Palm en sus modelos Tungsten y Treo, ofrecen esta máquina virtual de manera gratuita a sus usuarios.

- **NSICOM CrEme**. Esta máquina virtual corre en la mayoría de las PDAs del mercado, siempre que el sistema operativo sea al menos la versión 2003 de Microsoft Pocket PC. Implementa la versión 1.3 del JDK de Sun, así como algunas configuraciones de J2ME, como la CDC 1.0. La librería gráfica que usa para dibujar la interfaz es una variante ligera de AWT, pero no tiene soporte para SWING, aunque se puede extender con plugins, como por ejemplo, uno para la librería de Eclipse SWT. Su precio es algo más elevado que la opción de IBM.

- Dentro del mundo del software libre, se está activamente desarrollando una alternativa al JDK de Sun, llamada **GNU Classpath**. Basada en esta alternativa libre, y en otras librerías de código abierto, se levanta la máquina virtual **Mysaifu**, que corre en los sistemas operativos Windows Mobile (desde la versión 2003), es compatible con Java 1.4, e implementa AWT así como SWING. Además, es fácilmente extensible debido a su condición de software libre.

- Además, existen otras máquinas virtuales. Algunos fabricantes ofrecen versiones propias del kit de Sun o versiones propietarias y cerradas a su plataforma. Otras máquinas virtuales gratuitas tienen como gran desventaja estar anticuadas y sin soporte de versiones recientes de los estándares Java.

4.3. Java en sistemas GNU/Linux

Al igual que en Windows XP, las alternativas de desarrollo en Java para este sistema operativo son amplias. Desde la alternativa de Sun, en proceso de liberación a código libre GPL, a la implementación libre de la *Free Software Fundation*, GNU Classpath, pasando por *port* de **BlackDown Java**.

Cualquiera de las opciones disponibles es válida para nuestro propósito.

4.4. Java en GNU/Linux para dispositivos móviles como PDAs

En las distintas distribuciones GNU/Linux para PDAs, como son *OpenEmbedded*, *Familiar, Ångström*, las alternativas disponibles son básicamente dos: las implementaciones del *GNU Classpath* de la FSF, o las implementaciones de la alternativa de Sun realizadas por el proyecto *BlackDown*. Como máquinas virtuales las más populares son *SableVM*, *JamVM* y *Kaffe*.

- **GNU Classpath**, corriendo sobre cualquiera de las máquinas virtuales libres ofrece un entorno compatible con J2SE 1.4 y 5.0, implementando prácticamente todas sus clases, y ofreciendo un entorno estable y rápido.

- **BlackDown Java** es prácticamente el *port* oficial del JDK de Sun en Linux. Implementa al 100% la especificación 1.4 de Java, y existen versiones para todos los Linux disponible, en todas las arquitecturas. Debido a la reciente liberación de Java de Sun, es muy probable que este proyecto caiga en el olvido.

En nuestro caso, hemos utilizado principalmente para nuestras pruebas los JDK 1.4, 5.0 y 6.0 de Sun en los ordenadores portátiles (Windows y GNU/Linux); **Mysaifu** y **J9** en Windows Mobile 2003 para la PDA; y **GNU Classpath** y **JamVM** en la PDA usando Familiar Linux.

5. HERRAMIENTAS DE CONFIGURACIÓN DE TARJETAS WIRELESS

El primer modelo de localización, basado en tiempos de respuesta a llamadas ping, nos creó la necesidad de investigar la programación de configuradores para las tarjetas wireless en distintos sistemas operativos.

Esto se debía a que para poder hacer una llamada ping a un punto de acceso, primero debemos conectar la tarjeta wireless a dicho punto, si es que es alcanzable en el lugar donde se encuentre el cliente, como ya vimos en un punto anterior.

Por este motivo, eran necesarias dos herramientas: un escáner de red, capaz de hallar todas las redes accesibles en un punto dado; y un configurador de red para poder conectar a dichas redes. Ambas herramientas tendrían que funcionar en línea de comandos, para poder ser ejecutadas desde un programa Java sin que el usuario tenga que interactuar con ellas.

Además, es necesaria una herramienta de Ping, pero como todos los sistemas operativos disponen de una integrada, no fue necesaria su implementación.

5.1. Herramientas wireless en sistemas GNU/Linux

En los sistemas basadas en GNU/Linux, estas herramientas están disponibles en todas las distribuciones, generalmente instaladas por defecto.

Estas herramientas son:

- *iwlist*: escáner de redes wireless.
- *iwconfig*: configurador de la tarjeta wireless.
- *ping*: herramienta para obtener los tiempos de respuesta.

Con *iwlist* realizaremos el escaneo de red. En concreto, la orden para obtener las redes accesibles es:

```
$ iwlist wlan0 scan
```

Que produce una salida como ésta:

```
$ iwlist wlan0 scan
  wlan0     Scan completed :
  Cell 01 - Address: 00:0F:FF:FF:FF:FF
      ESSID:"WirelessNetwork"
      Protocol:IEEE 802.11bg
      Mode:Master
      Channel:1
      Encryption key:off
      Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 6 Mb/s; 9 Mb/s
                11 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s; 36 Mb/s
                48 Mb/s; 54 Mb/s
      Quality=35/100  Signal level=-79 dBm
      Extra: Last beacon: 224ms ago
```

Para el primer modelo de localización, debemos extraer las direcciones *MAC*, que en la salida de *iwlist* aparecen referidas como *Address*, podemos saber si el punto de acceso es uno de los que forman parte de nuestro sistema, para así conectarnos a él y poder realizar la llamada *ping*.

Para el modelo basado en fuerzas de señal, tendremos que extraer el valor del campo *Quality*, que es la intensidad de la señal recibida desde el punto de acceso.

Para conectarnos a los puntos de acceso, tenemos que hacer uso de la herramienta *iwconfig*. El uso de esta herramienta es sencillo, pues sólo necesitamos pasar como parámetro al interfaz de red que queremos configurar, y la *MAC* y el *ESSID* del punto de acceso al que nos queremos conectar, y suponemos que el demonio *DHCP* se encarga de obtener una IP válida desde el punto de acceso.
Como ejemplo podemos ejecutar:

```
$ iwconfig wlan0 essid “WirelessNetwork” ap 00:0F:FF:FF:FF:FF
```

Esta herramienta sólo la usamos en el modelo de localización por pings, puesto que en el modelo por fuerzas de señal sólo necesitaremos *iwlist*.

A partir de aquí, en el modelo de pings, realizaremos las llamadas al punto de acceso, cuya IP conocemos de antemano, y recuperaremos el tiempo en recibir la respuesta, para posteriormente enviarla al servidor.

El comando sería:

```
$ ping –c 4 ip_punto_acceso
```

Por motivos de seguridad, *iwlist* e *iwconfig* sólo pueden ser ejecutadas por el *superusuario (root)*. Por este motivo, deberemos modificar los permisos de ambos binarios para poder ejecutarlos como usuario normal.
Para ello, como *root* debemos teclear en la consola:

```
# chmod +s /sbin/iwconfig /sbin/iwlist
```

5.2. Herramientas wireless en sistemas Microsoft Windows

A diferencia de los sistemas GNU/Linux, en Windows no disponemos de una herramienta de escaneo o configuración en consola de comando, por eso tuvimos que escribir una herramienta de escaneo y recurrir a una herramienta de configuración de terceros.

Como herramienta de escaneo, desarrollamos una pequeña aplicación de linea de comandos llamada **WifiScanner.exe**, escrita en lenguaje C++, y basada en la librería de código abierto *HerecastLib*.

WifiScanner.exe es bastante sencillo de usar, puesto que no necesita ningún tipo de parámetro al ejecutarlo, puesto que al estar basado en el protocolo *NDIS*, obtiene todos los parámetros de configuración del propio sistema operativo y del driver de la tarjeta de red wireless.

Comentaremos más a fondo esta aplicación más adelante.

Respecto a la aplicación de configuración de la tarjeta de red wireless, el mayor problema que encontramos es la nula documentación de Microsoft sobre como desarrollar estas herramientas.

El auto-configurador de Windows, llamado *Zeroconf* solo se puede usar como herramienta gráfica, no desde consola. Exiten, eso sí, editores de perfiles de *Zeroconf*, distribuidos en formato *Freeware*, como por ejemplo *Zwlancfg*, que podrían ser usados para configurar la red, pero las pruebas realizadas nos dieron como resultado que el *Zeroconf* de Windows acaba ignorando estos perfiles, por lo que estas herramientas son de poco valor.

Por su parte, la mayoría de los fabricantes de tarjetas wireless ofrecen sus propios configuradores de red, pero la única forma de acceder a la documentación de estas tarjetas para poder programar una aplicación “unversal” es firmando contratos de no divulgación con los fabricantes, lo que se sale del ámbito de este proyecto.

Cómo última opción, nos planteamos desarrollar la aplicación basándonos en el protocolo *NDIS*, pero de nuevo, la falta de documentación y de tiempo nos alejó de ello.

En febrero Microsoft presentó su llamado *WirelessToolkit*, que es un entorno desarrollado para Windows Vista, con el que se podrán crear este tipo de aplicaciones sin problemas, e incluso este sistema operativo ya llevará por defecto alguna herramienta de este tipo incluida.

Debido a estos problemas, se decidió que para el modelo de tiempos de respuesta, los sistemas operativos de Microsoft no eran una opción viable, por lo que se decidió usar GNU/Linux.

Para el segundo modelo, cualquier alternativa sería valida, puesto que en Windows con nuestro *WifiScanner.exe* y en GNU/Linux con las herramientas estándares no habría problemas para obtener los datos necesarios.

6. SISTEMAS OPERATIVOS EN PDAS

Debido a los especiales requerimientos software de las tarjetas wireless, comentadas en el punto anterior, nos vimos en la necesidad de probar distintos sistemas operativos en dispositivos portátiles. Además, la decisión de desarrollar nuestras aplicaciones en Java también nos hizo probar como funcionan las diferentes implementaciones de este lenguaje de programación en diferentes entornos, como ya comentamos en el punto 4.

En ordenadores portátiles corrientes, los sistemas operativos no son ningún misterio, puesto que tanto Windows XP como cualquier distribución GNU/Linux son fáciles de instalar y configurar.

Pero en PDAs la situación es diferente. Las versiones recientes de Windows Mobile (antes PocketPC), no cuentan con las herramientas de configuración de tarjetas wireless previamente comentadas, y ni siquiera cuentan con un intérprete de línea de comandos. Así que decidimos investigar como instalar GNU/Linux en una PDA corriente.

A continuación comentaremos todo lo relativo a nuestro software de comunicación en todos los sistemas operativos probados, todos ellos instalados en nuestra PDA de pruebas, una HP iPaq h5450.

6.1. Windows Mobile en una PDA

En nuestra PDA probamos dos versiones de Windows Mobile, la versión 2002 (instalada de serie), y la versión 2003.

Ninguno de estos sistemas operativos ofrece herramientas de configuración de tarjetas wireless en modo consola, ni escáneres de red.

La inexistencia de programas de configuración de las tarjetas fue lo que nos forzó principalmente a probar otros sistemas operativos, mientras que el problema del escáner de red se solventó portando nuestro *WifiScanner.exe* a Windows Mobile, y ejecutarlo con el intérprete de consola que ofrece Microsoft, como descarga adicional, dentro de su paquete de ayuda al desarrollador, llamado *Windows Mobile Developer Power Toys*.

WifiScanner.exe sólo se puede ejecutar en Windows Mobile 2003 y superiores, puesto que a partir de esta versión es cuando se añadió soporte para el protocolo *NDIS*, que es en el que se basa esta herramienta.

El primer modelo de localización (basado en *pings*) no se puede implementar en una PDA con Windows Mobile, debido a la imposibilidad de configurar la tarjeta de red mediante linea de comandos, sin embargo, el segundo modelo, basado en la medición de las fuerzas de las señales recibidas, se podría implementar sin problemas gracias a *WifiScanner.exe*.

6.2. GNU/Linux en una PDA

En la actualidad existen diversas formas de usar GNU/Linux como sistema operativo en una PDA, aunque la mayoría de ellas derivan del proyecto *OpenEmbedded*, que es un entorno de desarrollo orientado a crear sistemas Linux para dispositivos embebidos.

Al igual que en los ordenadores de escritorio, para las PDAs existen varias "distribuciones" Linux, siendo las principales y mas usadas:

- **Familiar Linux**, que está basada en *OpenEmbedded*, y usa el núcleo y las utilidades compiladas por el proyecto *HandHelds.org*.

  Esta distribución se basa en un núcleo de la serie 2.4, y da soporte a las PDAs más comunes de HP, Palm y Zaurus. La instalación de esta distribución no es trivial, pero es sencilla si se siguen los pasos indicados cuidadosamente (ver apéndice 1).

  Existe una rama inestable de la distribución, basada en el núcleo Linux 2.6, que incluye un soporte mejorado para tarjetas inalámbricas, y otras mejoras, pero de momento no es lo suficientemente estable como para uso diario.

  Al igual que las distribuciones para máquinas de escritorio, *Familiar Linux* cuenta con todas las herramientas típicas que necesitamos en el proyecto, como son *iwconfig, iwlist*, etc.

- **Ångström**, que es una distribución más moderna que *Familiar,* basada también en *OpenEmbedded*, aunque ésta se basa en el núcleo 2.6 únicamente. Aunque es una distribución muy avanzada, de momento es bastante inestable para uso normal, aunque el soporte de distintos dispositivos es bueno.

- En los últimos meses, se han producido grandes avances en el soporte Linux en PDAs, sobre todo apoyados por el empuje de Nokia y sus *Internet Tablets, N770* y *N800*,cuyo sistemas operativo es una versión

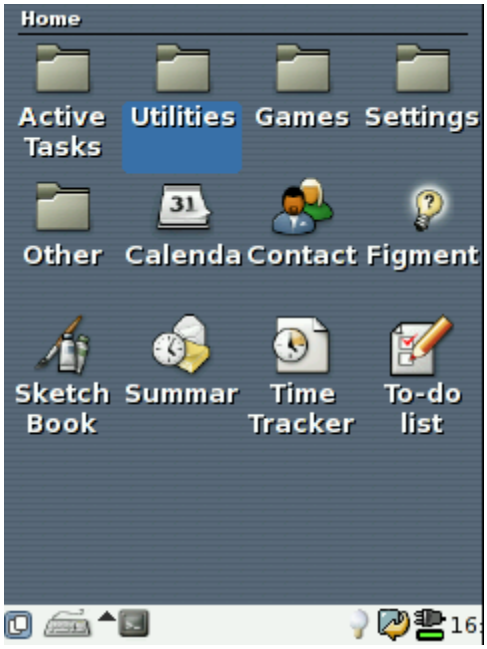
modificada de *Debian*, llamada **Maemo**, que incluye tanto el sistema operativo a bajo nivel, como una serie de aplicaciones y un completo entorno gráfico orientado a PDAs.

A la vista del éxito de esta propuesta, otros grandes del mundo del software libre se están lanzando a desarrollar sus sistemas para dispositivos móviles, como es la distribución *Ubuntu*, que ha anunciado la futura publicación de una versión de su sistema para PDAs.

También hay que tener en cuenta a *OpenMoko*, otra distribución reciente orientada principalmente a teléfonos móviles.

Hay que tener en cuenta que en los últimos 10 meses se ha producido una frenética carrera por avanzar en el soporte Linux, y en general de software libre, en dispositivos móviles, ya sean *PDAs*, *Tablet Pcs*, *Smartphones*, etc. así que el futuro de este tipo de sistemas se presenta muy interesante.

Es importante resaltar que todas las distribuciones Linux comentadas incluyen las típicas aplicaciones gráficas que todo usuario de una PDA necesita (gestión de correo y tareas, citas, internet, etc.)



7. MODELOS DE LOCALIZACIÓN

En la actualidad, los modelos más investigados en la localización usando tecnologías wireless son tres: usando el tiempo de respuesta de una llamada *ping*, es decir, desde la *capa de aplicación*; teniendo en cuenta la fuerza recibida en la tarjeta wireless de las antenas wireless accesibles; y usando el tiempo de respuesta desde la *capa de enlace de datos 802.11*.

En nuestro proyecto experimentamos con los dos primeros modelos, ya que el tercero exige modificaciones hardware de la tarjeta de red del cliente, y esto se salía del propósito de nuestro proyecto.

El primer modelo lo estudiamos debido a que nuestro proyecto es una continuación de uno anterior, que basaba sus estudios teóricos en este sistema.
En los primeros meses de desarrollo del proyecto implementamos este modelo, pero vimos que, primero las restricciones software, y luego las restricciones de infraestructura hardware en entornos reales, hacían de este modelo algo inapropiado para obtener resultados de localización válidos.

El segundo modelo fue en el que obtuvimos los mejores resultados, y en el que hemos basado nuestra aplicación de localización. Este modelo es barato de implementar en los clientes, tiene un reducido coste computacional, y es lo suficientemente preciso como para usarlo en entornos reales, donde la localización no es una cuestión vital, en cuanto a margen de error en distancia.

El tercer modelo es el más exacto en cuanto a localización, aunque su coste de implementación es mayor, pues requiere modificaciones hardware en la tarjeta de red inalámbrica. Debido a este coste, habría que realizar un estudio previo, para ver si la relación coste/precisión es asumible en el entorno deseado.

7.1. Localización usando RTT mediante llamadas Ping

Este sistema se basa en medir el tiempo que tarda un paquete *ICMP*, enviado usando la utilidad *ping*, desde el cliente al punto de acceso y su retorno.

Para localizar al cliente, se necesitan al menos cuatro APs inalámbricos accesibles, para poder ejecutar el algoritmo de triangulación explicado en el proyecto del año pasado.

7.1.1. Proceso de localización

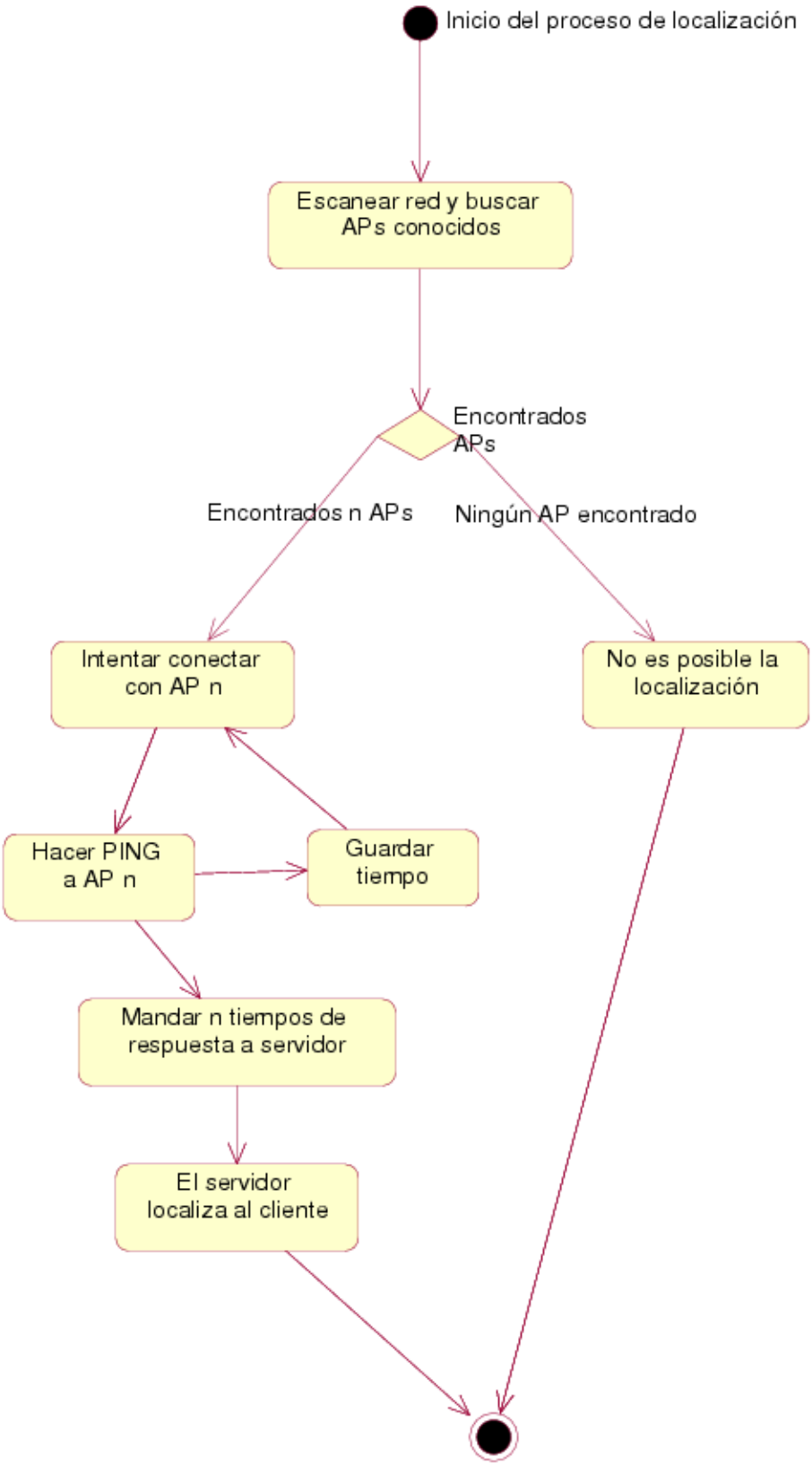
- La primera vez que el cliente entre en el edificio, el cliente se conectará automáticamente al servidor, y obtendrá un listado actualizado de puntos de acceso conocidos, con sus direcciones *MAC*, sus *IPs*, y sus coordenadas en el edificio.

- Después el cliente escaneará la red y obtendrá un listado de las direcciones *MAC* de todos los puntos de acceso inalámbricos accesibles en su posición. Además filtrará los puntos de acceso no conocidos, para sólo trabajar con los que pertenezcan a la lista obtenida desde el servidor.

- El siguiente paso sería conectarse a los puntos de acceso obtenidos. Para ello, buscaremos su *IP* en la lista que facilitada por el servidor, usando la dirección *MAC* hallada en el paso anterior. Una vez conectado a cada punto de acceso, haremos cuatro llamadas *ping* al punto de acceso, y obtendremos la media del tiempo de respuesta de dichas llamadas. Almacenaremos este valor y pasaremos al siguiente punto de acceso.

- Una vez calculados todos los tiempos de acceso a cada punto de acceso, nos debemos conectar de nuevo al servidor, a través de alguno de los puntos de acceso, y le enviaremos los datos obtenidos.

- Con los tiempos de acceso, el servidor ejecuta el algoritmo de triangulación, y obtiene la posición del cliente. Para más información sobre este algoritmo, refiérase al proyecto “Software de comunicación de PDAs”, publicado en el año 2006.


Inicio del proceso de localización
Escanear red y buscar APs conocidos
Encontrados APs
Encontrados n APs
Ningún AP encontrado
Intentar conectar con AP n
No es posible la localización
Hacer PING a AP n
Guardar tiempo
Mandar n tiempos de respuesta a servidor
El servidor localiza al cliente

7.1.2. Resultados de la investigación en este modelo

A la vista de los estudios realizados, hemos llegado a la conclusión de que no existe ningún beneficio reseñable que lleve a pensar que la implantación de este sistema sea el idóneo, por las siguientes razones:

Las condiciones hardware que se necesitan para implementar este modelo son prácticamente inasumibles en un entorno real. Esto es debido a que con este sistema, se supone que la llamada *ping* nos da un valor real de tiempo de respuesta, y la experiencia nos dice que este valor, depende de múltiples factores, como por ejemplo, que el punto de acceso sea un terminal "tonto", es decir, que se componga de una antena, conectada por cable a un centro de datos lejano, con lo cual, el camino del paquete *ICMP* enviado por la utilidad *ping* es más largo de lo debido.

Por ejemplo, en el entorno inalámbrico de la Universidad Complutense, las antenas wireless no son activas, y están conectadas por cable al Centro de Proceso de Datos. Lo que significa que al hacer *ping* a uno de los puntos de acceso reconocidos dentro de nuestra facultad, el paquete está viajando hasta la antena, y de ahí a los servidores del Centro de Proceso de Datos, y de ahí de vuelta a nuestro terminal, sin tener porqué llevar el mismo camino, pues las condiciones de los enrutadores pueden haber cambiado.

Esta situación falsearía los datos obtenidos, con lo cual la única manera de implementar este método sería con puntos de acceso activos, que respondan ellos mismos a la llamada *ping*.

Además, habría que tener en cuenta que los paquetes *ICMP* que enviamos tardarían más tiempo en llegar a su destino si la carga en la red es alta. Es decir, en momentos de mucho tráfico de red, los datos obtenidos tampoco serían correctos.

Otra razón es que este modelo, a día de hoy, es únicamente implementable en sistemas GNU/Linux, debido a que no existe una aplicación de configuración de tarjetas de red inalámbricas que sea apropiada para este tipo de tareas, que requiere una interfaz en línea de comandos, y que sea independiente del modelo de tarjeta de red. Se podría desarrollar una aplicación específica para un modelo de tarjeta determinado, siempre teniendo en cuenta que habría que llegar a un acuerdo con fabricante para obtener las especificaciones. Esto aumentaría de manera inasumible los costes de implementación del sistema.

Por último, otro problema sería el tiempo que se tardaría en obtener el punto de localización, puesto que el proceso es largo, ya que, como se ha explicado, debemos escanear la red, y luego conectarnos uno a uno a los puntos de acceso y hacer la llamada *ping*, lo que aumentaría muchísimo el tiempo final del proceso.

7.2. Localización usando fuerzas de señal (Fingerprinting)

El modelo de localización llamado *Fingerprinting* se basa en medir las fuerzas de las señales recibidas de diferentes puntos de acceso en un lugar determinado, y aplicar a esas fuerzas una serie de algoritmos que permiten determinar la posición del cliente.

Se llama *Fingerprinting*, porque la parte principal del método es crear una base de conocimiento con datos de fuerzas tomados en diferentes puntos del entorno en el que se implantará el sistema. Es decir, se crea una matriz con las fuerzas recibidas de cada punto de acceso en una serie de puntos del recinto.

7.2.1. Proceso de localización

- El primer paso es realizar el proceso de *Fingerprinting*. Para ello es necesario recorrer el edificio escaneando en cada punto la red, y almacenando los valores de fuerza de la señal recibidos de cada AP accesible.

  Este sistema tiene una mayor precisión cuanto mayor es la densidad de puntos en los que se han tomado valores. La precisión en metros de este algoritmo depende de la distancia entre los puntos que se han tomado durante el proceso de *Fingerprinting*.

  Este proceso se realiza sólo una vez y sirve para todos los clientes. Sólo sería necesario repetirlo en caso de producirse cambios estructurales en el edificio, o severas modificaciones en la infraestructura de la red inalámbrica, por ejemplo, mover todos los puntos de acceso.

  En nuestro caso, tomamos medidas cada 120 centímetros en los pasillos de la facultad, con lo cual el posible error obtenido era entre 1 y 2 metros. Este proceso nos tomó una media de hora y media para cada planta del edificio.

- Con la base de conocimiento de *Fingerprinting* realizada, la primera vez que el cliente entra en el edificio y se conecta al servidor, se descarga a su dispositivo móvil los datos tomados.

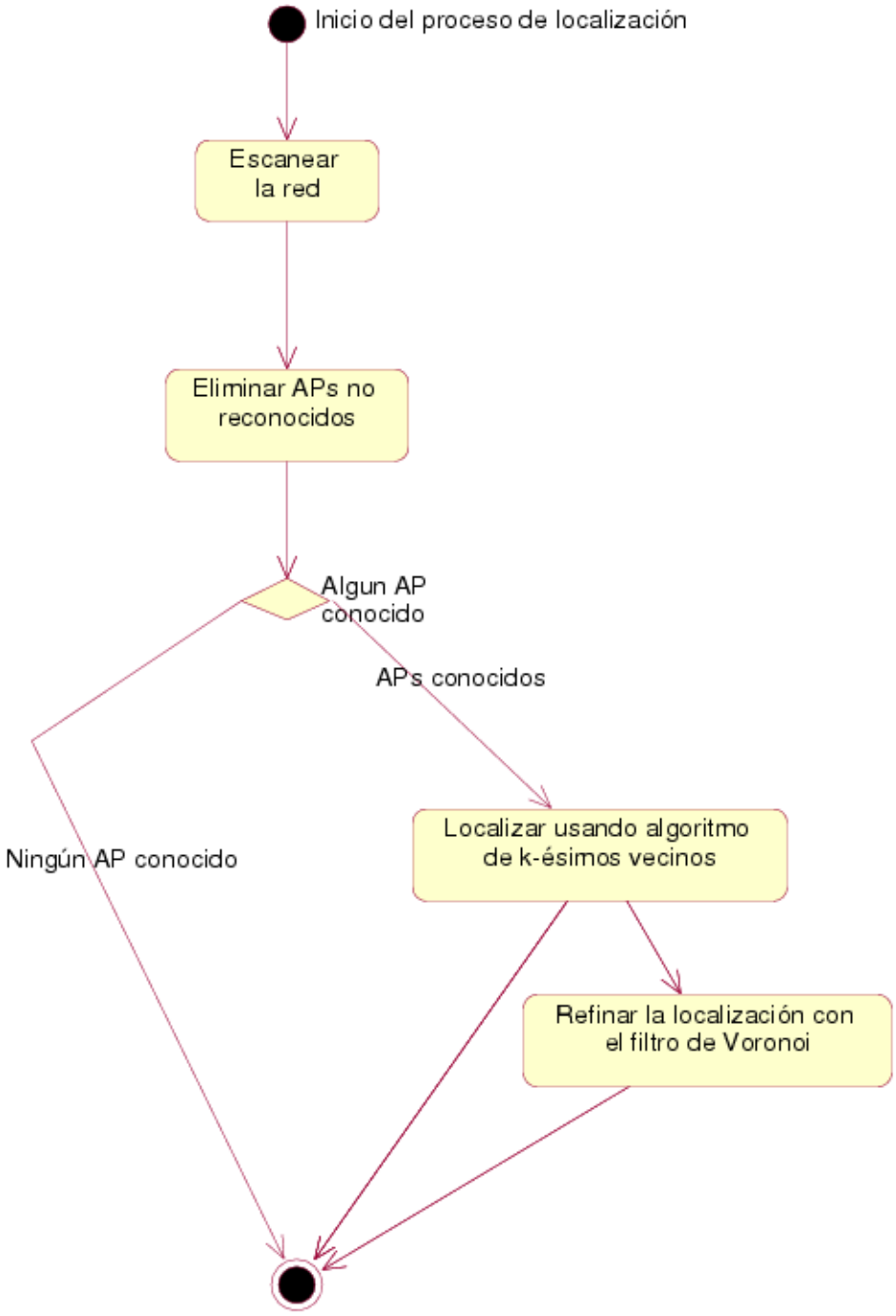
  En el caso de que el dispositivo no tuviese la capacidad computacional necesaria para realizar estas operaciones, la descarga de la base de datos no sería necesaria y sería el servidor el encargado de realizar todos los cálculos.

- En el siguiente paso, el cliente escanearía la red obteniendo la fuerza de la señal de los APs accesibles en ese lugar determinado.

- Después, usando el algoritmo de *k-ésimos vecinos más cercanos*, compararemos los datos obtenidos por el cliente con la base de conocimiento obtenida en el proceso de *Fingerprinting* para hallar el punto donde se encuentra localizado el cliente.

  Una vez más, si el cliente no posee la capacidad computacional suficiente para analizar estos datos, enviará los resultados del escaneo al servidor para que éste realice los cálculos pertinentes.

- El siguiente paso sería aplicar una serie de filtros para refinar el resultado obtenido por el algoritmo anterior, como podría ser un *filtro de partículas* aplicado a un mapa de *Voronoi* o un filtro más simple basado en la restricción de movimientos a los puntos más cercanos.


Inicio del proceso de localización
Escanear la red
Eliminar APs no reconocidos
Algun AP conocido
APs conocidos
Ningún AP conocido
Localizar usando algoritmo de k-ésimos vecinos
Refinar la localización con el filtro de Voronoi

7.2.2. Resultados de la investigación en este modelo

Este modelo, aplicado a los requerimientos de precisión de localización de nuestro proyecto es el más apropiado si tenemos en cuenta la relación *coste de implementación / coste de cálculo / precisión*. No obstante, hemos encontrado algunos inconvenientes dignos de reseñar.

- Las fluctuaciones de la fuerza de la señal de un punto de acceso recibidas en un lugar determinado a lo largo del tiempo, debidas a ondas electromagnéticas, muros, otros dispositivos en funcionamiento, etc., introducen una serie de saltos indeterminados en el proceso de localización. Estos saltos pueden producir que estando el cliente detenido, el sistema detecte que se está desplazando alrededor de un punto. Sin embargo, el impacto de estos saltos se pueden minimizar usando los ya nombrados filtros o con un detector de inercia.

- Puede darse el caso que en varios puntos del edificio se obtengan valores similares de fuerzas de señal para APs accesibles todos ellos desde dichos puntos, lo que provocaría que el algoritmo podría situarnos en cualquiera de esos puntos. Esto se solucionaría, de nuevo, utilizando los filtros mencionados en puntos anteriores.

- En algunas tarjetas de red inalámbricas el tiempo que transcurre entre las actualizaciones de las señales recibidas puede variar. En nuestras pruebas, con una tarjeta *Intel ipw2200* el tiempo de reconfiguración de la tarjeta era de alrededor de 5 segundos. De todas maneras, se podrían optimizar estos valores modificando el *driver* de la tarjeta de red inalámbrica.

- Otro inconveniente es que con este método no se puede trazar correctamente la trayectoria real que sigue un usuario. Debido a los problemas mencionados anteriormente, durante el movimiento del usuario a través del edificio se podrían obtener localizaciones físicamente imposibles como causa de la velocidad de desplazamiento del cliente. Para resolver este problema, sería necesario aplicar alguno de los filtros nombrados.

  Un método sería restringir los movimientos del cliente a los *n* puntos más cercanos a la anterior localización usando para ello un mapa de *Voronoi*. De esta manera, no sería posible desplazarse en dos instantes de tiempo consecutivos a puntos que no sean los indicados por el mapa. También se podría aplicar sobre el mapa de *Voronoi* un *filtro de partículas*, que restringiría los movimientos del usuario solamente al sentido de su marcha.

A pesar de estos inconvenientes, las ventajas que se encuentran en este método son múltiples:

- El algoritmo de localización es lo suficientemente eficiente como para que la capacidad de cómputo requerida sea asumible para dispositivos móviles con poca capacidad de procesamiento, como por ejemplo, PDAs.

- Debido a que la precisión es directamente proporcional al número de puntos de acceso y al número de mediciones tomadas durante el proceso de *Fingerprinting*, se podría ajustar el error medio de la localización hasta el grado que el entorno donde se implante requiera sin requerir un coste en tiempo excesivo para este proceso.

- Otro beneficio es que para aumentar considerablemente el rendimiento del sistema sólo hay que incrementar la densidad de APs del sistema, lo cual, debido al estado del mercado actual, no desembocaría en un desembolso excesivo. Sin embargo, una densidad excesiva de puntos de acceso aumentaría ligeramente el tiempo de respuesta del sistema.

- Con este método no es necesario conocer la localización de los APs del entorno, lo cual evita un estudio de la infraestructura de la red inalámbrica.

- Este modelo está implementado de forma que se ajusta eficazmente a cualquier configuración de red inalámbrica real, no siendo necesaria la modificación de la misma para crear un entorno controlado, donde no se produzcan interferencias que modifiquen el resultado de la aplicación.

- No es necesario ningún tipo especial de requerimientos software para poder ejecutar una aplicación desarrollada siguiendo este modelo, ya que con el software estándar de cualquier sistema operativo y las herramientas creadas es suficiente para un correcto funcionamiento.

7.3. Localización con el tiempo de respuesta desde la *capa de enlace de datos 802.11*

El último modelo propuesto consiste en la medida del tiempo de respuesta de paquetes *802.11*, pero no en la capa de aplicación, como en el primer modelo, sino en la capa de enlace *802.11*.

Para poder tomar las medidas necesarias a este nivel se precisan ciertas modificaciones hardware. Las tarjetas de red inalámbricas cuentan con un reloj con una frecuencia de 44MHz. La modificación necesaria consistiría en agregar un contador al reloj, con sus disparadores de inicio y parada.

Habría que diseñar, además, un módulo que implementase un protocolo de comunicación entre el contador y el software para poder obtener las medidas de tiempo.

El contador sería activado por su disparador al realizar el envío de un datagrama en la *capa de enlace* y desactivado al recibir la respuesta de confirmación *ACK* del punto de acceso, obteniendo, de este modo, el tiempo de respuesta necesario.

Para obtener unos resultados óptimos habría que aplicar una serie de métodos estadísticos, así como algún filtro, en concreto un *filtro de Kalman*. Esto nos permitiría obtener una precisión menor de 1 metro en un tiempo muy reducido.

Este método sería el más preciso teniendo en cuenta que sólo implica una mínima modificación hardware y ninguna en cuanto a la infraestructura de la red inalámbrica existente.

No obstante, dado que no se contempló la implementación de este método, no se procederá a explicar a fondo sus características. Si se desease obtener más información acerca de esta vía de investigación se puede consultar la bibliografía adjunta.

8. ALGORITMOS PARA LA LOCALIZACIÓN

8.1. Algoritmo de k-ésimos vecinos más cercanos (*k-closest neighbors*)

El modelo de localización usando *Fingerprinting*, se basa en hallar la fuerza de las señales recibidas de cada punto de acceso en un lugar determinado. Esas fuerzas se tratarán como distancias métricas, y se les aplicará el algoritmo de k-ésimos vecinos más cercanos (*k-closest neighbors*).

Este algoritmo busca dentro de la base de conocimiento creada en el proceso de *Fingerprinting*, y selecciona los puntos que mejor se ajustan a las señales recibidas en el lugar en el que se encuentra el cliente.

El criterio usado para seleccionar dichos puntos es la menor distancia euclídea de las fuerzas, tomando las fuerzas como distancias métricas, como ya se comentó anteriormente.

Una vez seleccionados los k puntos con la distancia mínima al punto donde estamos, se calculará el baricentro de dichos puntos, y así hallaremos el punto donde se encuentra el cliente.

El algoritmo, detallado en pseudocódigo, sería así:

```
ptos_escaneo = vector de fuerzas obtenidas en el escaneo actual
fp_data      = base de conocimiento creada durante el fingerprinting
fp_points    = vector con las fuerzas obtenidas en cada punto
k-mínimos    = vector de puntos cuya distancia al punto actual es mínima

calcula punto:
      para cada pi ∈ fp_points:
            distancia(ptos_escaneo, pi)
            si distancia es menor almacena el pi en vector de k-mínimos
      pto_buscado = baricentro(k-mínimos)

donde distancia(ptos_escaneo, pi):
```

$$d(p,p_i)=\frac{1}{M}*\sqrt{\sum_{j=1}^{M}\left(RSS_j(x,y)-RSS_j(x_i,y_i)\right)^2}$$

donde M = número de elementos de ptos_escaneo
donde RSS_j = fuerza recibido del AP_j en (x,y)

donde baricentro(k-mínimos):

$$X = \frac{\sum_{j=1}^{k} \frac{1}{dist(Z,Z_i)} * X_j}{\sum_{j=1}^{k} \frac{1}{dist(Z,Z_i)}}$$

donde $X_j \in$ k-mínimos
donde X es el punto donde se encuentra el cliente

Habría que hacer algunas precisiones sobre este algoritmo.

Si al realizar el escaneo de la red se obtienen puntos de acceso que no aparecen en la base de conocimiento creada durante el proceso de *fingerprinting*, se eliminan dichos puntos de acceso y no se tienen en cuenta durante el resto del algoritmo.

En cuanto a la función que halla la distancia del punto en el que se encuentra el cliente y los puntos de la base de datos de *fingerprinting*, hay que reseñar, que si el punto con el que estamos intentando hallar la distancia no contiene valor de fuerza para uno de los puntos de acceso que hemos localizado en nuestro escaneo, vamos a devolver una distancia máxima, de modo que al elegir los mínimos, ese valor se ignorará, puesto que ese punto no puede ser de los buscados.

También tenemos que dejar claro que, durante el proceso de *fingerprinting*, al tomar valores de señal en un punto, tomamos varias medidas en el mismo punto, con lo cual se obtienen varios valores de fuerza del mismo punto de acceso en el mismo punto, así que para que el algoritmo sea eficiente, almacenamos la media de dichas fuerzas, y ese es el valor que se tiene en cuenta al hallar las distancias.

Este algoritmo es lo suficientemente ligero para ser ejecutado con rapidez en cualquier máquina. Por ejemplo, en un ordenador portátil a 1.4GHz, el algoritmo se ejecuta entre 1 y 3 milisegundos, dependiendo del número de puntos de acceso detectados. Esto consigue que el proceso de localización se pueda ejecutar en dispositivos con capacidad computacional reducida, como ya se comentó anteriormente.

Para ver la implementación en código Java de este algoritmo refiérase al apéndice 3.

8.2. Algoritmo de filtrado por caminos

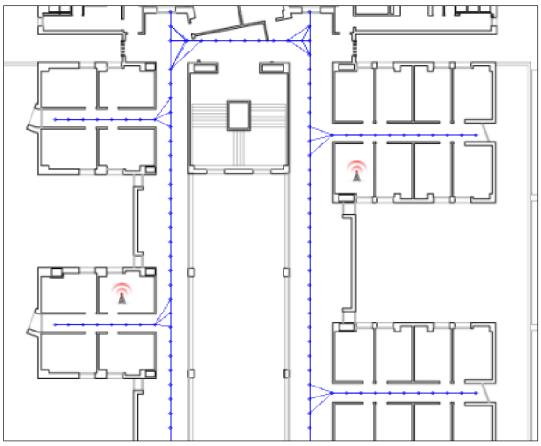
El algoritmo de localización basado en los *k-ésimos vecinos*, a pesar de que suele ser bastante exacto, puede introducir errores bastante grandes esporádicamente, cuando por ejemplo, existen dos puntos opuestos en el edificio en los cuales las fuerzas de las señales de los puntos de acceso que tienen en común son similares. Si el cliente se encuentra en un punto, en el que esas fuerzas son similares a las que recibimos, se puede dar la circunstancia de que el sistema nos localice en un punto muy lejano del que realmente estamos situados.

Por ello se puede introducir en la aplicación un algoritmo que filtre esos errores.

La forma más sencilla es suponer que una persona, en un tiempo determinado, no puede moverse más de una determinada cantidad de metros.

Una forma poco costosa de resolver este problema sería trazando rutas por las que una persona se puede mover por el edificio, usando para ello un mapa de Voronoi, y restringiendo los posibles movimientos del individuo a las posiciones marcadas por este mapa.

Otra forma más sencilla, si cabe, es aprovechar el proceso de *fingerprinting* para trazar dichas rutas. Es decir, usando los puntos del edificio donde se tomaron valores de fuerza, se trazan posibles rutas de movimiento, como se puede observar en los gráficos siguientes.



De este modo, se podrían evitar saltos erróneos en las posiciones localizadas consecutivamente, pues se restringiría el movimiento de la persona a, por ejemplo, 10 puntos accesibles en un tiempo de por ejemplo 5 segundos.

9. APLICACIONES PRÁCTICAS DE LA LOCALIZACIÓN

Los sistemas de localización en interiores pueden ser realmente útiles en ciertos entornos, en los que aparte de la localización propiamente dicha, se pueden ofrecer diversos servicios basados en la posición de una cierta persona o cosa.

Por ejemplo, en un entorno hospitalario, se podrían ofrecer muchos servicios a los pacientes y al personal utilizando este tipo de sistemas.

Supongamos que se ha dotado al personal del hospital de dispositivos móviles con acceso wireless. Una vez superada esta primera premisa, se abre un amplio abanico de posibles funcionalidades a desarrollar.

El primer y más evidente servicio a ofrecer, sería la posibilidad de conocer la posición de los distintos trabajadores del hospital. Sería muy útil disponer de la situación de cada persona en cada momento debido a que ante cualquier tipo de necesidad o emergencia, el conocimiento de dicha localización podría acelerar enormemente la resolución de cualquier problema médico.

Otro beneficio sería, por ejemplo, el poder usar ciertos dispositivos informáticos, basados en su localización respecto a la persona que solicite el servicio. Por ejemplo, un médico necesita una copia impresa de la historio clínica de un paciente, así que dentro de la habitación del paciente, selecciona en su PDA "Imprimir historia del paciente", y el sistema localiza al médico en la habitación, consulta en la base de datos qué paciente está ingresado en dicha habitación, y envía una orden de impresión del expediente de dicho paciente en la impresora más cercana a la posición del doctor. Otro posible caso sería el de mostrar una radiografía de un paciente en la pantalla más cercana, por ejemplo.

También se podría llevar un control estadístico de las zonas del hospital en las que hay más movimiento de personal, y así calibrar si se necesita contratar más personal para ciertas zonas, o si se requieren más dispositivos para ciertas zonas, etc.

Del mismo modo que a las personas, se podría usar este sistema para controlar la posición de instrumental médico, solo añadiendo un pequeño emisor wireless. Como por ejemplo, localizar el carro de paradas más cercano a un doctor.

Del mismo modo, esto se podría extrapolar a otros entornos, como por ejemplo almacenes, restaurantes y hoteles, grandes empresas que necesiten localizar a sus trabajadores, como en los grandes complejos de edificios de ciertas empresas, recintos feriales, etc.

10. SOFTWARE IMPLEMENTADO PARA NUESTRO PROYECTO

10.1. Indoor Location – Aplicación de *fingerprinting* y geolocalización

Indoor Location es la aplicación principal de nuestro proyecto, y sirve tanto para realizar todo el proceso de *fingerprinting* como visor de localización de clientes.

Esta aplicación está totalmente desarrollada sobre Java, usando como librerías gráficas Swing y Awt, lo que convierte a *Indoor Location* en una aplicación 100% multiplataforma.

El flujo de trabajo con esta aplicación sería el siguiente:

1. Crear proyecto nuevo

Al abrir la aplicación por primera vez, se nos pregunta si abrir un fichero de datos ya existente, o crear un proyecto nuevo. Al elegir "C*rear proyecto nuevo*", se nos solicitará seleccionar un fichero de imagen con el mapa de la planta a escanear y los datos de la planta elegida.

2. Realizar el *Fingerprinting*

La aplicación permite, de una forma sencilla, realizar el proceso de *fingerprinting* en el edificio. En el menú "*Vista*" seleccionaremos la opción "*Escanear*". Pulsando sobre el mapa, seleccionaremos la posición de origen para el *fingerprinting*, entonces al pulsar el botón de "*escanear*", representado por una antena, se realizarán tantos escaneos consecutivos como se haya fijado en el campo "*Escaneos*". Los escaneos se realizarán cada 2 segundos.

Para tomar datos en otro punto del mapa, o bien se selecciona una nueva posición pulsando con el ratón sobre el mapa, o se usan los botones de desplazamiento. Estos botones, al pulsarlos, desplazan la posición tantos decímetros como marque el campo *"Paso"*.

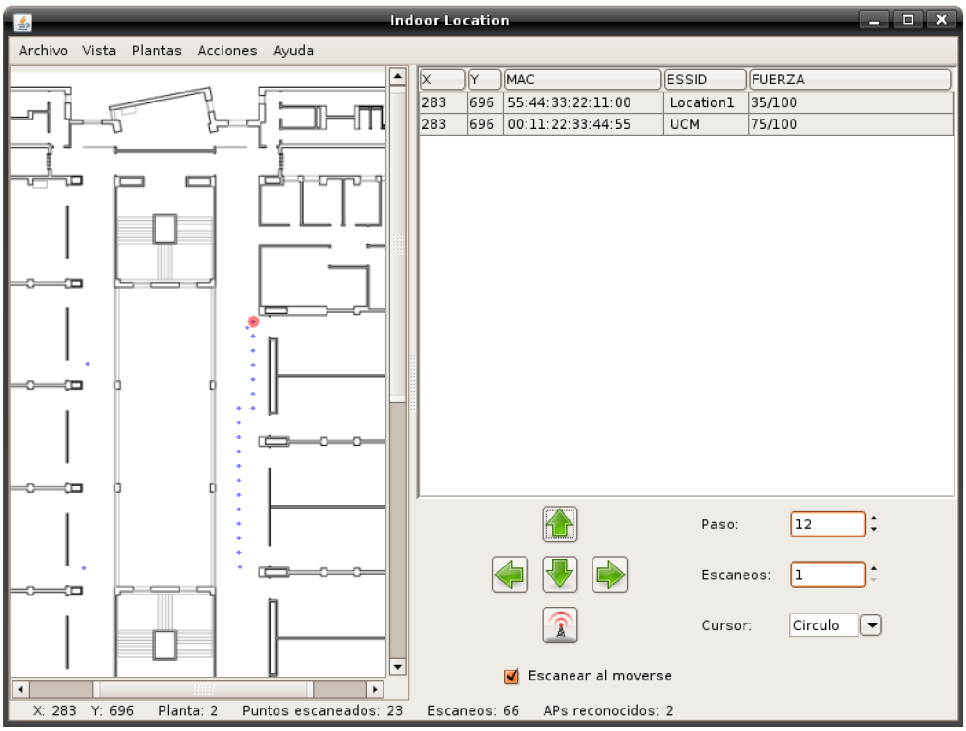
Esto es muy útil si por ejemplo se quieren tomar medidas cada cierto número de metros continuamente. De modo que, por ejemplo, en nuestro caso, hemos tomado medidas cada 120 centímetros. Con lo cual solo teníamos que escanear en un punto, seleccionar la opción de "*Escanear al moverse*", movernos a la nueva posición, y pulsar el botón correspondiente.

Los datos obtenidos se mostrarán en la tabla superior, indicando la posición de escaneo, y los puntos de acceso encontrados, con sus fuerzas, dirección *MAC* y su *ESSID*.

Además, en el mapa de la izquierda, van apareciendo en azul los puntos donde se ha escaneado.

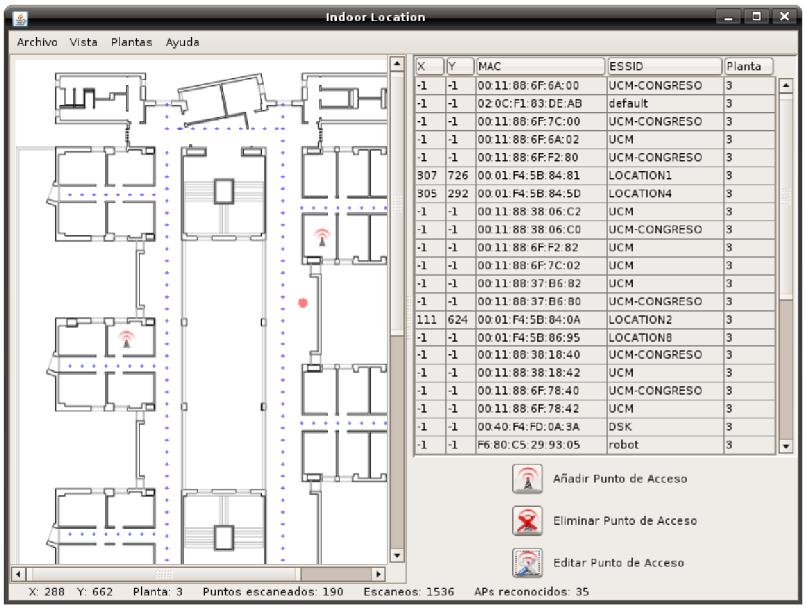
Estos datos se van almacenando de forma preventiva, en un fichero en el directorio de usuario, nombrado por la fecha del escaneo y el nombre de la planta escaneada.

Es importante reseñar que todos los datos del proyecto se pueden guardar en el disco duro usando la opción "*Guardar*" del menú "*Archivo*", incluidos los mapas, y los datos obtenidos en el *fingerprinting*.



3. Introducir datos de los puntos de acceso

El siguiente paso sería pasar a la vista de "*Dispositivos*", y modificar los datos de los puntos de acceso localizados. En este paso sería necesario cambiar la planta donde se encuentra situado el punto de acceso. Los demás cambios no serían necesarios a no ser que los puntos de acceso se hayan introducido después del *fingerprinting*.

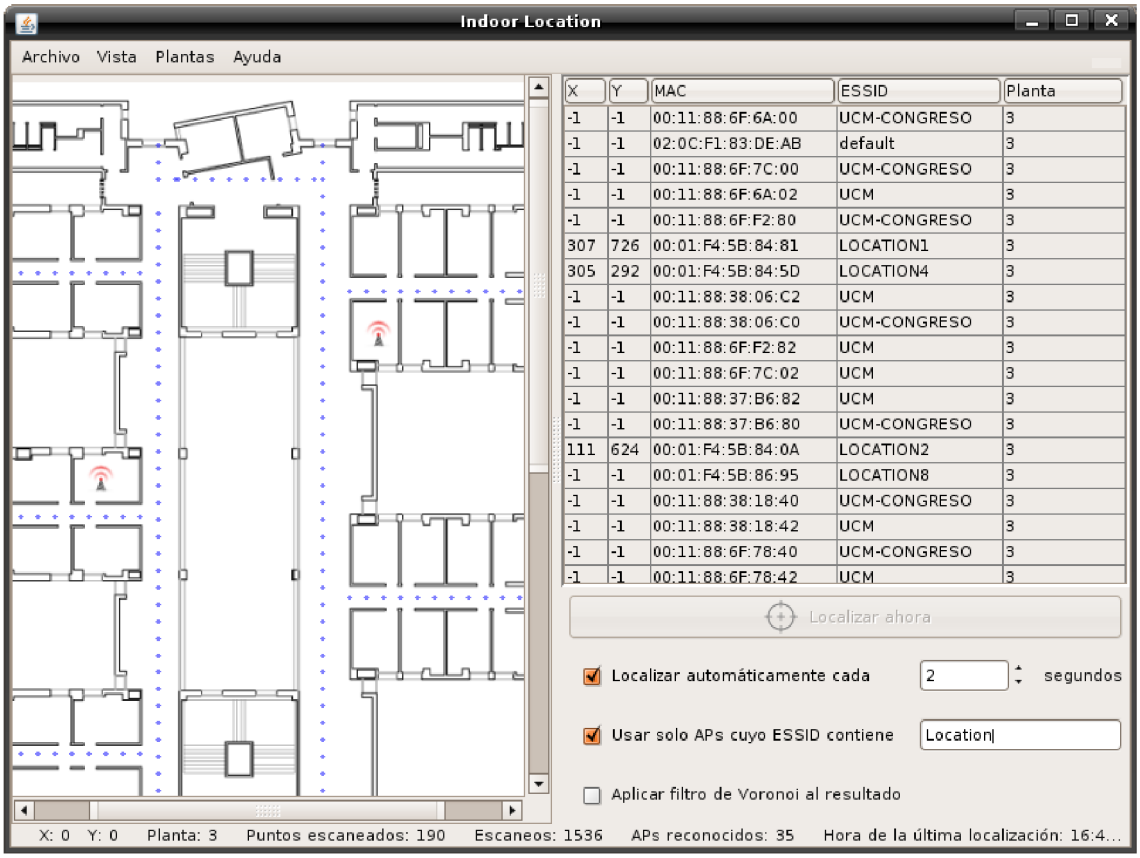


4. Localización

El siguiente paso sería pasar a la vista de "*Localización*", donde se puede ver la posición en el edificio del usuario.

Se puede realizar una localización automática, que se repetirá cada un cierto tiempo definido por el usuario, o se puede forzar la localización en un momento dado pulsando el botón "*Localizar ahora*".

Además, se presentan dos opciones extras. Una para procesar la localización usando sólo los puntos de acceso cuyo *ESSID* contenga una cadena introducida por el usuario; y la otra es para activar el filtrado de la posición usando un mapa de Voronoi.



Al pulsar el botón de "*Localizar ahora*", lo que se ejecuta es el proceso de localización explicado en el punto 1 de esta memoria, es decir, se escanea la red, y se pasan los resultados obtenidos (mostrados en la lista de la parte superior de la interfaz) al algoritmo de *k-ésimos vecinos más cercanos*, y se halla el baricentro para obtener nuestra posición.

Los aspectos internos de esta aplicación se detallarán en profundidad en el apéndice 3 .

10.2. Escáner de red para Windows en modo consola – WifiScanner.exe

Debido a la necesidad de una herramienta de escaneo de red para sistemas operativos Windows basada en línea de comandos, como ya se comentó en el punto 3, desarrollamos una aplicación llamada **WifiScanner.exe**.

Esta aplicación esta desarrollada en C++, y se apoya en la librería de código abierto *HerecastLib*.

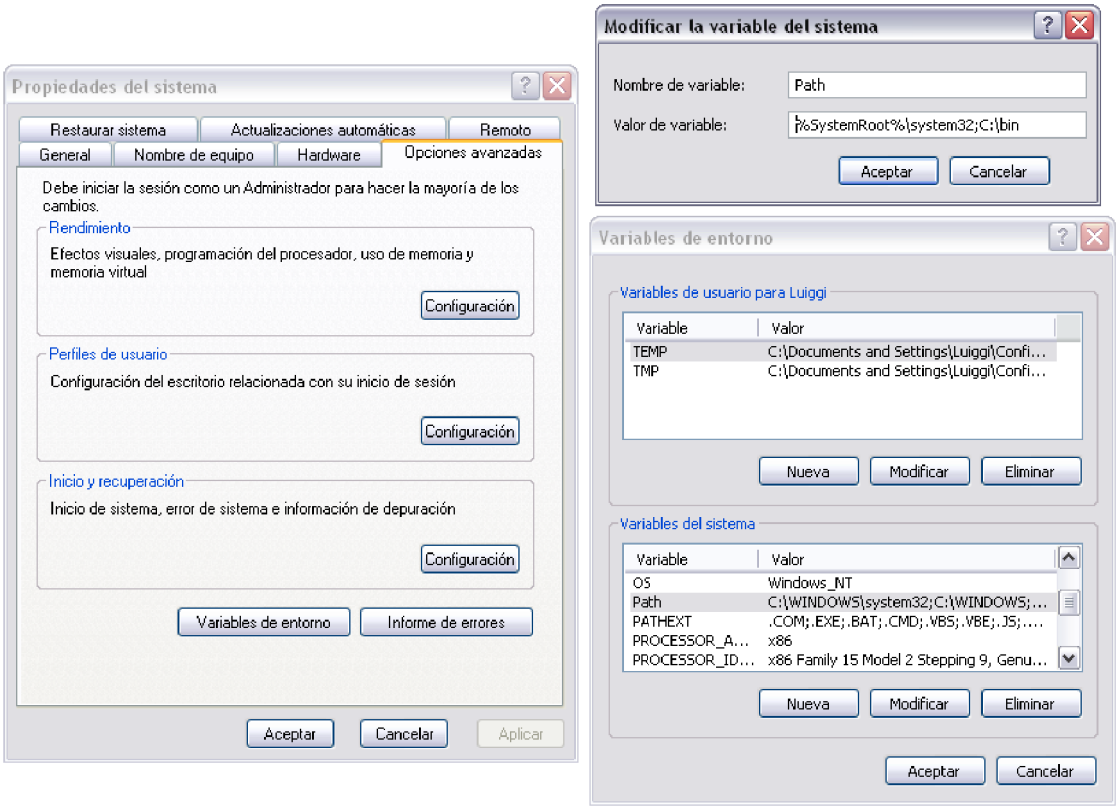
Esta librería ofrece una sencilla interfaz para interactuar con los drivers de tarjetas inalámbricas usando el protocolo *NDIS* (*Network Driver Interface Specification*). De esta forma, se pueden extraer de la tarjeta los puntos de acceso visibles, así como el estado de estos puntos de acceso (su *MAC*, la fuerza con que se recibe su señal, si el punto de acceso está protegido, etc.)

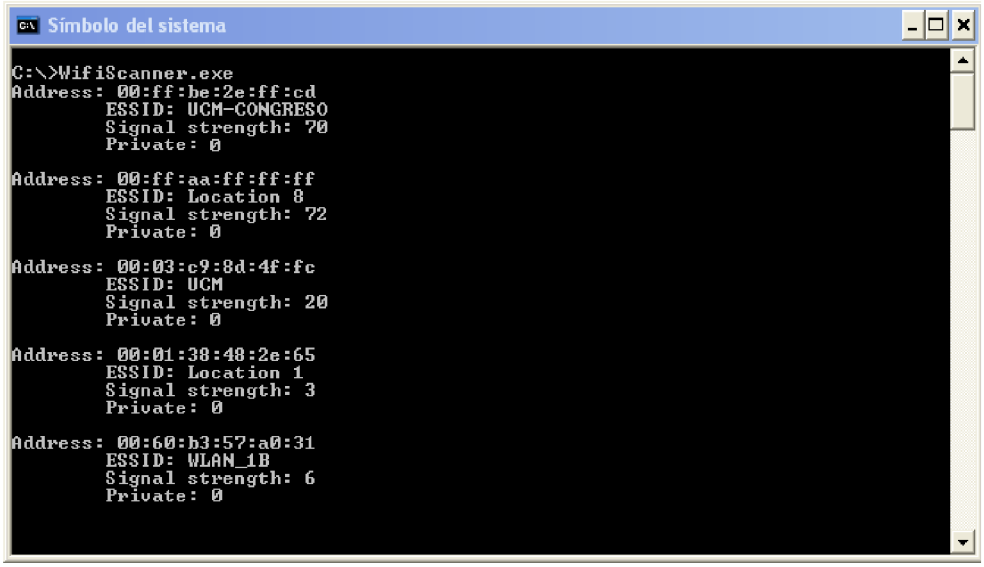
Debido a que la herramienta que necesitábamos es muy sencilla, implementamos *WifiScanner.exe* de la forma más parca posible. Al ejecutarla en línea de comandos, se obtiene un listado de los puntos de acceso visibles, además de la información necesaria para el algoritmo de localización.

La salida estándar de esta herramienta es:

```
C:\WifiScanner.exe
Address: 00:ff:45:de:ff:32
      ESSID: UCM
      Signal strength: 70
      Private: 0

Address: a4:ff:c5:75:de:a1
      ESSID: Location2
      Signal strength: 72
      Private: 0

Address: 00:03:c9:8d:4f:fc
      ESSID: UCM-CONGRESO
      Signal strength: 20
      Private: 0

Address: 00:01:38:48:2e:65
      ESSID: UCM
      Signal strength: 3
      Private: 0

Address: 00:60:b3:57:a0:31
      ESSID: Location3
      Signal strength: 6
      Private: 0
```

Para que esta aplicación funcione correctamente, debemos añadirla al *PATH* de Windows. Es decir, se puede crear, por ejemplo, un directorio llamado *bin* en la raíz del disco duro (*C:\bin*), y después añadiremos ese directorio al PATH (*Mi PC -> Propiedades del Sistema -> Opciones avanzadas -> Variables de entorno -> Variables del sistema -> Path*), y allí añadimos C:\bin a la lista.





10.3. Editor de mapas - Map_Editor

Para este proyecto se han usado unos planos originales de la facultad, para así hacer las pruebas correctamente, viendo la funcionalidad del localizador en un entorno real. Dichos planos fueron facilitados por la facultad, lo que ha originado que no tuviésemos que hacerlos nosotros mismos para poder probar. Pero, a pesar de esto, se ha realizado una aplicación de edición gráfica. Así si alguien está interesado en probar el localizador en base a otro lugar (que no sea la facultad) se puede hacer unos planos de ese sitio en cuestión. La aplicación de la que hablamos es *Map_Editor*.

A continuación vamos a exponer como se ha llevado a cabo la implementación de la aplicación *Map_Editor*, la cual es un editor para poder desarrollar planos en los cuales probar el localizador. Dichos planos podrán estar compuestos de diferentes plantas, las cuáles tendrán diversas habitaciones, en las que se podrán incorporar dispositivos que nos interese poder localizar, como puede ser una impresora o un PC. Además se verá como poder usar las diferentes funciones de las que se dispone (esto se verá en el manual facilitado para el uso del editor).

10.3.1. Implementación de Map_Editor

Uso del *framework*

La aplicación está basada en el uso del *framework* **JHotDraw** (más concretamente en la versión 5.2 del mismo). Dicho *framework* está diseñado para desarrollo de editores gráficos, y se basa en el uso de patrones de diseño (*factory method, decorator, template method*, etc..), los cuales permiten una fácil extensión del núcleo del *framework*.

La gran diferencia del uso de un *framework* al uso de una biblioteca, es que la segunda la llamamos nosotros haciendo una referencia a la misma (y así poder usar todo lo que forma parte de ella); por el contrario el primero nos llama a nosotros, es decir realizamos ciertas extensiones de partes de la estructura del *framework* y lo “enchufamos“ al mismo.

Para más información sobre *JHotDraw* se puede acceder a los siguientes enlaces:

http://www.jhotdraw.org
http://sourceforge.net/projects/jhotdraw

Sí cabe destacar que para la implementación de *Map_Editor* no se ha hecho un uso del *framework* en su función principal; es decir hemos comentado que están hechos para extender de ellos, pero nosotros debido a

las características de nuestra aplicación hemos optado por hacer el editor modificando *JHotDraw*. Algunas clases si han sido extendidas pero la mayoría están modificadas, esto se hizo así por comodidad ya que las características de nuestras figuras, manejadores y algunas otras cambiaban bastante.

Esto es una posible mejora de *Map_Editor*, el adaptarlo de tal manera que sea todo una extensión del *framework* sin tocar el núcleo del mismo.

Figuras, menús y objetos implementados

Map_Editor tiene implementado las siguientes figuras:

- Rectángulos: con ellos podremos hacer habitaciones. Están implementados a partir del básico de *JHotDraw*. Se dibuja completamente cerrado, y tenemos otra figura que permitirá abrir puertas, la cuál se comentará a continuación.

- Puertas: la puerta nos permitirá abrir un rectángulo y así poder completar la realización de la habitación. Está implementado de tal manera que al pinchar cerca de uno de los lados del rectángulo se abre una puerta en esa zona. El proceso consiste en que a la hora de pinchar de la manera expuesta el rectángulo se convierte en una polilínea, la cual quitará ese trozo de un lado del rectángulo para así asemejar una puerta.

De momento sólo está implementada la opción de crear una puerta, como futuras mejoras de la aplicación estaría la opción de crear varias puertas. De momento para tener varias habría que hacer las habitaciones en base a líneas simples.

- Líneas: es la línea básica dada por *JHotDraw*. Con ella podremos crear habitaciones de tal manera que puedan tener una o más puertas.

- Polilínea: se usa la que viene en *JHotDraw* con ciertas modificaciones leves para adaptarla a *Map_Editor*. Con ella también podremos crear habitaciones, tanto directamente o con el paso intermedio de abrir puerta en un rectángulo.

- Texto: se ha dejado respecto a *JHotDraw* la opción de introducir un *textFigure* para poner nombres a objetos, como podría ser nombrar una habitación, un pasillo....

Cabe destacar, que de inicio en las plantas aparece una cuadrícula, la cual se ha creado con un simple relleno de líneas dibujadas directamente en el rectángulo que forma la planta. Está la opción de la cuadrícula se vea o no.

En caso de la existencia de la cuadrícula (cosa que es bastante útil para la realización de un plano), la introducción de figuras se hace de tal manera que se alinea con la cuadrícula; es decir se crea en base a los subcuadrados del rectángulo que forma la planta, aparecerá alineado con esas líneas.

Map_Editor tiene implementado los siguientes objetos:

- Impresora: se introduce una impresora del tamaño que queramos ya que se puede manipular como cualquier figura.

- Pantalla: similar a la anterior.

Se deja abierta la posibilidad de extender la aplicación con la incorporación de todo objeto deseado.

Map_Editor tiene implementado los siguientes menús:

- Archivo: este menú tiene las funciones típicas de toda aplicación: *abrir*, *nuevoMapa* (este al pulsar sobre él hace uso de un asistente el cuál ha sido implementado de tal forma que te pide el tamaño de la planta, tamaño de la cuadrícula (puntos entre cada subcuadrado) y el número de plantas que queremos para nuestro mapa. Todo está implementado correctamente para ver si a la hora de crear uno nuevo hay que guardar el que tenemos abierto, si lo hay, y demás), *guardarComo* (la forma de guardar la explicaremos después), exportar (a varios formatos gráficos como pueden ser *.jpg*, *.gif*, *.png*, etc.), imprimir y salir.

- Editar: este se ha optado por usar el de *JHotDraw* con las opciones de siempre : cortar, pegar, duplicar etc...

- Nueva planta: la cual hace la función de introducir una nueva planta en el mapa.

- Cuadrícula: este menú nos da la opción de ocultar o mostrar la cuadrícula, cambiar su tamaño, o alinear o no respecto a ella.

- Vista: se ha implementado una herramienta "*zoom"* que permite variar el tamaño de toda la planta con sus figuras respectivas. No sólo de la planta que esté visible en ese momento, sino de todas las que forman el mapa.

En torno a los menús, podemos reseñar que a la hora de la ejecución habrá campos de los menús que no estén activos, ya que no tiene sentido que lo estuviesen. Como ejemplo podemos dar los campos del menú "*editar*" (copiar etc..), el zoom, guardar o exportar etc... Esto es así porque son campos que realizan acciones sobre el dibujo.

Aparte, al ejecutar la aplicación, se puede ver que en la parte derecha sale una tabla donde se ven los nombres de las diferentes plantas que forman parte del mapa en cuestión. Además están las opciones para manipular dicha tabla: subir, bajar (cambia la planta visible), duplicar planta, borrar y crear una nueva. Todo esto se ha implementado haciendo uso de *swing*, con diversos botones, tabla, etc...

Otro hecho destacado de la implementación es lo siguiente: una vez se crea un nuevo mapa se abre la ventana donde se dibujará. En ese dibujo aparece de inicio el rectángulo al cual hemos dado tamaño con el asistente; en este es donde se pinta,, las figuras se han implementado de tal forma que al moverlas no podemos salirnos de los límites del cuadrado, que hace las veces de planta. Además a la hora de usar los manejadores de las figuras, ocurre más de los mismos, sólo deja extender esa figura en puntos que estén dentro del rectángulo base.

Vamos a explicar como se dan el proceso de "*guardar*".

Antes de ver como es el texto de lo guardado, vamos a analizar como es el proceso de guardar:

Se crea un fichero "*.draw*" para cada planta, el formato "*.draw*" que usamos es una modificación del que trae por defecto *JHotDraw*. A continuación se crea un fichero "*.map*" (formato creado por nosotros) en el que se almacenan las direcciones de los archivos de cada planta, los cuales están todos en la misma ruta.

Cabe destacar que el nombre de la carpeta es el que especifiquemos en el asistente de guardado. El nombre del ".map" también será el especificado, y el nombre de cada planta será "nombre especificado"_"nombre de la planta".

El “*.map”* constará de lo siguiente:

```
/home/location/prueba/prueba_Planta1.draw  Planta1
```

Como se ve nos da la ruta donde se haya guardado. Se puede apreciar que le hemos dado el nombre “prueba”, así como la carpeta creada con dicho nombre y el de la planta que tiene.

El “*.draw*” correspondiente a la planta será como sigue:

```
CH.ifa.draw.standard.StandardDrawing 18
  CH.ifa.draw.figures.BasicFloorFigure "no_attributes" 0 0 800 800
  CH.ifa.draw.figures.PolyLineFigure 5 280 40 40 40 40 360 280 360 280 200
  CH.ifa.draw.figures.LineFigure 2 440 40 680 40
  CH.ifa.draw.figures.LineFigure 2 680 40 680 360
  CH.ifa.draw.figures.LineFigure 2 680 360 540 360
  CH.ifa.draw.figures.LineFigure 2 440 40 440 100
  CH.ifa.draw.figures.LineFigure 2 440 140 440 280
  CH.ifa.draw.figures.LineFigure 2 440 320 440 360
  CH.ifa.draw.figures.LineFigure 2 440 360 540 360
  CH.ifa.draw.figures.ImageDeviceFigure 100 60 40 20 "/CH/ifa/draw/images/PRINTER.gif"
  CH.ifa.draw.figures.ImageDeviceFigure 220 240 20 20 "/CH/ifa/draw/images/SCREEN.gif"
  CH.ifa.draw.figures.ImageDeviceFigure 80 280 20 20 "/CH/ifa/draw/images/SCREEN.gif"
  CH.ifa.draw.figures.ImageDeviceFigure 80 240 20 20 "/CH/ifa/draw/images/SCREEN.gif"
  CH.ifa.draw.figures.ImageDeviceFigure 220 280 20 20 "/CH/ifa/draw/images/SCREEN.gif"
  CH.ifa.draw.figures.ImageDeviceFigure 580 80 20 20 "/CH/ifa/draw/images/SCREEN.gif"
  CH.ifa.draw.figures.ImageDeviceFigure 580 140 20 20 "/CH/ifa/draw/images/SCREEN.gif"
  CH.ifa.draw.figures.ImageDeviceFigure 580 200 20 20 "/CH/ifa/draw/images/SCREEN.gif"
  CH.ifa.draw.figures.ImageDeviceFigure 580 260 20 20 "/CH/ifa/draw/images/SCREEN.gif"
```

Se puede ver que la primera habitación se ha creado con un rectángulo y abriendo puerta, y la segunda en base a líneas. Luego nos salen todos los dispositivos incorporados.

10.3.2. Posibles mejoras en Map_Editor

Se han ido comentando algunas a lo largo de lo expuesto anteriormente, pero daremos aquí las que consideramos oportunas y relevantes:

- Introducción de los dispositivos que se consideren necesarios. Como dijimos hemos dados dos tipos de dispositivos pero es fácilmente ampliable a todos los que se necesite, dependiendo de para que esté destinada la aplicación (un hospital, una facultad, etc...).

- Mejorar la herramienta para abrir puertas, ya que se ha dicho que de momento sólo se permite crear una puerta en un rectángulo, por lo que para hacer varias se debe crear con líneas.

Así pues una manera sería, por ejemplo, poder convertir una polilínea en otra, o no transformar el rectángulo en polilínea sino cambiar el dibujo del mismo.

- Otra mejora puede ser mejorar la codificación para usar correctamente el *framework*, sin modificarlo. De esta manera habría que hacer extensiones de las clases necesarias (como se ha hecho con algunas), y redefinir los métodos que no nos valen de la manera que están en *JHotDraw*. Por ejemplo las clases de los diferentes tipos de figuras están modificadas, se podría posiblemente extender dichas clases y redefinir lo que hemos modificado, como son los métodos de mover, manipular etc...

11. RESULTADOS DEL PROYECTO

En esta sección se procederá a resumir las metas alcanzadas durante el desarrollo del proyecto, así como posibles ampliaciones y trabajos o estudios futuros. En particular, todo lo relacionado con el editor de mapas, que se incluye como apoyo a la aplicación principal, se encuentra especificado en el punto correspondiente a dicha herramienta en la sección anterior.

Dado que tras un profundo análisis del proyecto realizado el curso anterior por unos compañeros de la facultad y de los distintos modelos de localización existentes actualmente, se decidió continuar por otra rama totalmente distinta de investigación y desarrollo, los primeros objetivos fueron estudiar las distintas alternativas que ofrecía la nueva elección. Esto llevó a realizar un amplio estudio en el campo de los sistemas operativos para los dispositivos móviles (en concreto sobre *PDAs*), así como de los distintos entornos de desarrollo disponibles para cada uno de ellos, para, de este modo, tener la absoluta certeza de que los siguientes pasos a tomar fueran los más certeros.

Tras este primer estudio, reflejado en distintos puntos a lo largo de esta memoria, y una vez elegido el software más apropiado para nuestros intereses, se procedió a la implementación de la aplicación de localización propiamente dicha. La cual fue diseñada para que finalmente pudiera ser ejecutada en distintos sistemas operativos.

La herramienta de localización desarrollada, tal y como se ha precisado en puntos anteriores, utiliza el modelo de localización de *Fingerprinting*, el cual, según nuestras investigaciones y estudios, teóricamente proporcionaba una precisión bastante aceptable teniendo en cuenta los entornos para los cuales está destinado el servicio. Sin embargo, nuestra mayor preocupación era comprobar que dicha precisión era tal y como la habíamos visto planteada en los pocos estudios y artículos disponibles acerca de este tema, ya que, recordemos, este tipo de aplicaciones lleva relativamente poco tiempo en período de investigación y no es sencillo encontrar experimentos que certifiquen todos los estudios teóricos.

Así pues, una vez finalizado el proceso de implementación de la herramienta, se consideró imprescindible establecer un campo de pruebas y analizar los resultados obtenidos en un entorno real.

Gracias a la colaboración de nuestro profesor director del proyecto, conseguimos que el personal del Centro de Proceso de Datos (CPD) de la Universidad Complutense nos facilitara el hardware, en modalidad de préstamo, y el permiso para interferir en la red wireless de la Universidad Complutense de Madrid necesarios para conseguir realizar las pruebas deseadas.

De este modo, se llevó a cabo el experimento y se comprobó que en efecto, los resultados eran los esperados. La media del error obtenida en el proceso de localización oscilaba entre dos y cinco metros como mucho, lo cual, tal y como se ha comentado en secciones anteriores, es asumible. Este error se produce debido a que la señal wireless no es constante, sufre fluctuaciones a causa de interferencias con otros dispositivos, con muros e incluso debido al tráfico de la red en ese instante. En cualquier caso, se observó que, a pesar de estos factores externos, en la mayoría de los casos se obtenía, tras varios intentos, una localización prácticamente exacta.

Estas fluctuaciones en algún momento puntual provocaban que en la localización del dispositivo se dieran "saltos" a posiciones en los que las fuerzas de la señal medidas en el proceso previo eran similares, aunque estos puntos no se encontraran cercanos en el espacio.

Asimismo, se comprobó que cuanto más refinada fuera la toma de datos durante el proceso de *Fingerprinting*, es decir, cuanto mayor número de mediciones se realicen, mejores resultados se obtenían en cuanto a la precisión final del algoritmo.

Este tipo de factores externos, son totalmente incontrolables e impredecibles en un entorno real. Se conseguirían atenuar en un entorno preparado y controlado, pero a la hora de la implantación del sistema se seguirían sin obtener unos resultados precisos al cien por cien, por lo que resulta inútil intentar controlar y adelantarse a dichos factores.

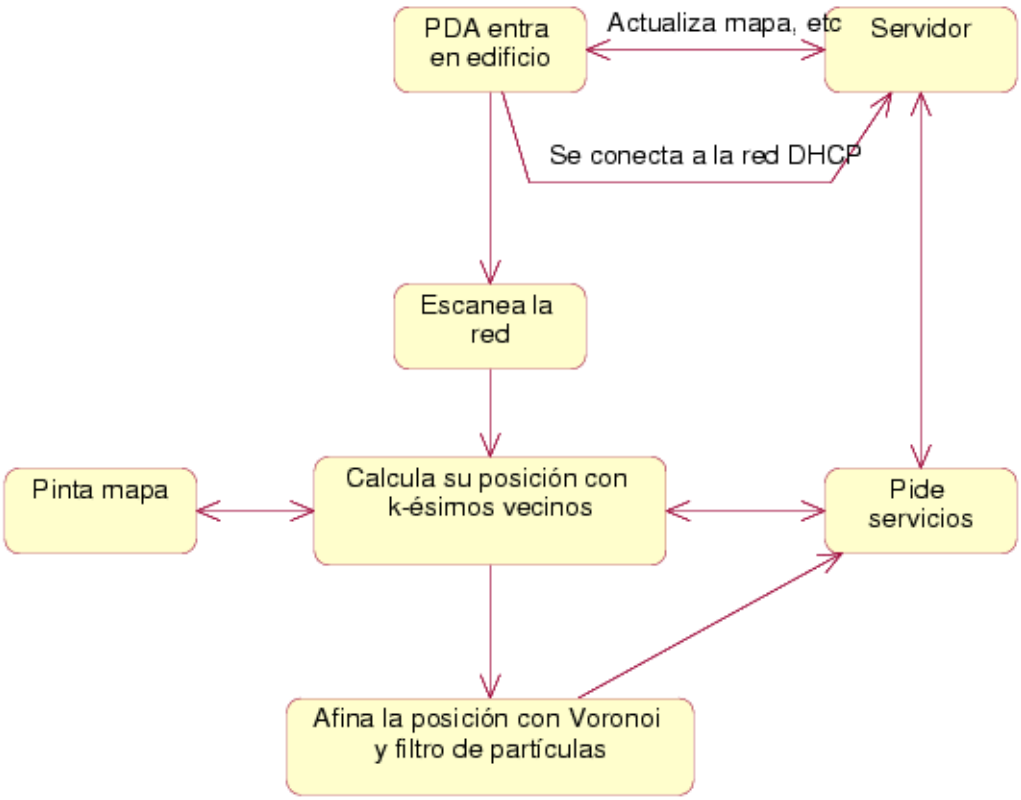
Sin embargo, es posible mejorar dicha precisión hasta conseguir una localización prácticamente exacta. La solución es aplicar una serie de pasos como complemento a nuestro sistema.

Dentro de este tipo de mejoras, se encuentran los procesos descritos en la sección 8 de la memoria, aplicación del algoritmo de *Voronoi* para evitar los saltos entre puntos con fuerzas de señal similares y aplicación del filtro de partículas para tener en cuenta sentidos de dirección del movimiento y aceleraciones del mismo.

La mejora de la precisión, podría ser un posible punto de partida para un posible futuro trabajo a partir de los resultados obtenidos en este proyecto, sin embargo, habría que estudiar el entorno en el que se desearía aplicar este tipo de sistemas y si, realmente, no resulta suficiente el resultado obtenido en la actualidad.

Otras posible vía de desarrollo a continuación del punto en el que se da por terminado nuestro proyecto, sería la de desarrollar una aplicación *cliente-servidor* que pudiera recibir peticiones de distintos usuarios y que les proporcionase diferentes servicios, tales como localización de otros usuarios de la red, caminos mínimos entre distintos puntos del lugar en el que se aplica el sistema, o acceso a los recursos que proporcione el entorno (impresoras, pantallas o demás), por ejemplo.

El esquema de esta aplicación podría ser similar al siguiente:



Esta opción, además, ofrece un amplio abanico de posibilidades a la hora del enfoque, ya que es aplicable a gran variedad de entornos (hospitales, almacenes, comercios, etc.) y por tanto, la cantidad de servicios que se podría ofrecer sería muy amplia.

APÉNDICE 1. Instalar GNU/Linux en una PDA

Vamos a instalar la versión 0.8.4 de Linux Familiar en nuestra iPaq h5450. Lo vamos a hacer desde un sistema operativo Ubuntu, y usando la estación base de la PDA conectada a la red, y a al ordenador vía el puerto serie.

En Ubuntu necesitamos tener instalados el paquete *minicom* (emulador de *módem* serie) y la utilidad *ymodem* (para enviar archivos).

1. Descargar Familiar 0.8.4

Bien, lo primero que tenemos que hacer es descargar el Linux que vamos a meter en la PDA. Para ello nos dirigimos a:

http:familiar.handhelds.org/releases/v0.8.4/install/download.html

Una vez allí, elegimos que versión queremos (stable, v0.8.4), que PDA tenemos (h5400) y que entorno queremos (OPIE, estilo KDE). y pulsamos en *Download*.

Se nos descargará un archivo llamado *bootopie-v0.8.4-h3900.tar*.

2. Copiar Familiar Linux a la PDA

Bien, descomprimimos el archivo bajado en nuestro ordenador, y nos disponemos a copiarlo a la PDA.

Para copiarlo, tenemos varias opciones:

- Enviar la carpeta usando el ActiveSync de Windows
- Enviar la carpeta usando SynCE de Linux
- Enviar la carpeta via Bluetooth
- Copiar la carpeta en una tarjeta SD y meterla en la PDA

Nosotros hemos optado por la opción de la tarjeta.

Pues bien, una vez copiados los archivos a la SD, la metemos en la PDA, y nos vamos a *Inicio -> Programas -> Explorador de archivos*. Allí, en la barra gris de abajo, seleccionamos *Storage Card*, y nos aparecerá lo que tenemos dentro de la SD. Luego nos metemos en la carpeta de Familiar (*bootopie-v0.8.4-h3900*).

3. Arrancar la utilidad de flasheo del gestor de arranque (BootBlaster)

Bien, aquí apareció el primer problema. Resulta que la ejecución de programas desde la tarjeta SD es muy lenta (o imposible, no lo sé). Así que tuve que copiar dos archivos a la memoria interna de la PDA. En concreto copié desde la SD a Mis Documentos los archivos:

- *BootBlaster3900-2.6.exe*
- *bootldr-pxa-2.21.12.bin.gz*

Bien, una vez copiados, nos vamos a Mis Documentos, y hacemos click en *BootBlaster3900-2.6*. Se nos presentará una pantalla con instrucciones, y abajo un menú con tres opciones, *File - Flash – About.*

4. Hacer una copia de seguridad del *Bootloader* original y de Windows

Lo primero es guardar el sistema original. Para ello seguimos 3 pasos:

- *Flash -> Save Bootldr .gz Format* (guarda el gesto de arranque original)
- *Flash -> Save Wince .gz Format* (guarda el Windows original)

Se nos muestra un par de mensajes de confirmación, y después copiamos los ficheros *saved_bootldr.gz, wince_image.gz y assets.gz* desde Mis Documentos a la tarjeta SD.

5. Instalar el gestor de arranque (*BootLoader*)

Bueno, de nuevo dentro del *BootBlaster*, vamos a cargar el nuevo gestor de arranque en la PDA.

Es muy importante hacer esto con la PDA conectada a la corriente y no desconectarla durante el proceso! Si lo hacemos, dejaremos inservible la iPaq!

Flash -> Program

Ahora se nos abrirá una pantalla donde tenemos que elegir el fichero para cargar. Elegimos el *bootldr-pxa-2.21.12.bin.gz* que está en Mis Documentos.

El proceso de carga durará unos 15 segundos. Si todo ha ido bien, se nos mostrarán una par de diálogos confirmando que todo ha salido correctamente.

De todas maneras, le damos a *Flash -> Verify* para comprobar una vez más que todo ha ido bien. Si algo fallase, no te alarmes y no resetees la PDA, carga de nuevo el *bootloader*, y si sigue fallando, carga el *bootloader* de Windows que has guardado previamente (lo hiciste verdad?).

Bueno, pues el gestor de arranque nuevo ya está cargado. Ahora vamos a instalar Linux.

6. Entrar en el modo *Bootloader*

Primero vamos a poner la PDA en modo *bootloader*, para ello, pulsamos el botón del centro de la PDA, y a la vez metemos el lápiz en el agujerito de abajo para resetear la PDA, que al hacer esto vibrará. Soltamos los botones, y ponemos la PDA en la base. Ya estaremos en modo bootloader, aunque la pantalla no cambia (se queda lo que hubiese antes).

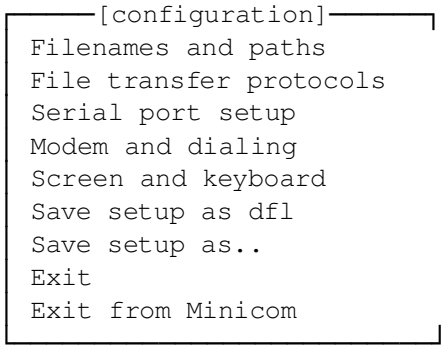
7. Preparar el *Minicom* para cargar Linux

Bueno, necesitamos instalar y configurar el programita minicom para conectar via puerto serie con la PDA. Para ello necesitamos entrar en la terminal de linux como root, y poner el minicom en modo 115200 8N1.

```
$ sudo apt-get install minicom lrzsz
$ su
Password: lo que sea
```

Ahora ejecutamos el *minicom* en idioma inglés...

```
# LANG=C minicom -s
```



Elegimos "*Serial port setup*" y ponemos las opciones como las siguientes

```
A -    Serial Device      : /dev/ttyS0
B - Lockfile Location     : /var/lock
C -   Callin Program      :
D -  Callout Program      :
E -    Bps/Par/Bits       : 115200 8N1
F - Hardware Flow Control : No
G - Software Flow Control : No

   Change which setting?
```

Por último elegimos "*Save setup as dfl*" y después "*Exit*"

```
Initializing Modem
```

Y después aparecerá en pantalla...

```
Welcome to minicom 2.1

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on Nov  5 2005, 15:45:44.

Press CTRL-A Z for help on special keys

boot>
```

8. Cargar el Linux en la PDA

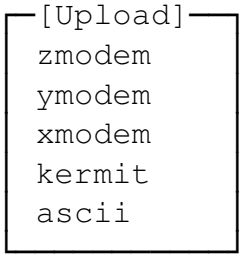
Ahora vamos a mandar el sistema de archivos Linux (comprimido en un archivo llamado "*bootopie-v0.8.4-h3900.jffs2*" a la PDA.

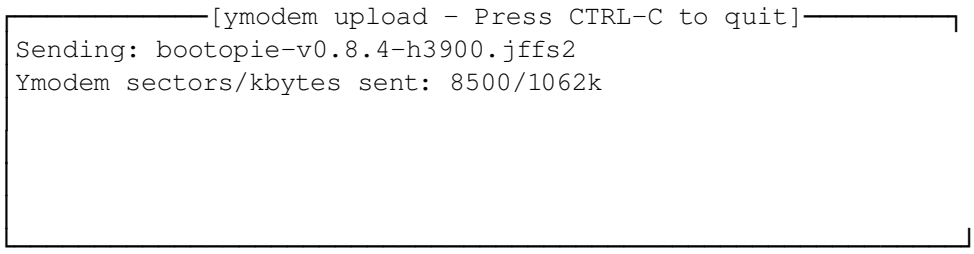
Usaremos las órdenes "*set ymodem 1*" y "*load root*".

```
Welcome to minicom 2.1

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on Nov  5 2005, 15:45:44.

Press CTRL-A Z for help on special keys

boot> set ymodem 1
    setting param <ymodem> to value <1>
```

```
boot> load root
partition root is a jffs2 partition:
 expecting .jffs2 or wince_image.gz.
 After receiving file, will automatically uncompress .gz images
loading flash region root
using ymodem
ready for YMODEM transfer...
```

Ahora pulsamos *Control+A* y después *S*, nos saldrá un menú en el que escogeremos "*YMODEM*", y luego saldrá otro menú para elegir ficheros, y tendremos que elegir el "*bootopie-v0.8.4-h3900.jffs2*", que es la imagen del sistema Linux.





Este proceso durará bastante tiempo, mas de media hora... Paciencia... Cuando termine de cargar, saldrá algo como:

```
Erasing sector 00140000
Erasing sector 00180000
Erasing sector 001C0000
Erasing sector 00200000
.
.
.
addr: 00360000 data: 781590DB
addr: 00370000 data: 642637AE
addr: 00380000 data: E0021985
addr: 00390000 data: 15DA97EC
Erasing sector 00FC0000
writing flash..
```

```
addr: 00100000 data: E0021985
addr: 00110000 data: E3BAD617
addr: 00120000 data: 0FA1F57B
addr: 00130000 data: 9343AEEB
.
.
.
addr: 00600000 data: E0021985
addr: 00610000 data: FFFFFFFF
addr: 00620000 data: FFFFFFFF
addr: 00630000 data: FFFFFFFF
verifying ... formatting ... done.
boot>
```

9. Arrancando la PDA con Linux

Bien, ya solo nos queda arrancar. Usaremos la orden "*boot*".

```
boot> boot
```

La pantalla de pondrá en negro (si no lo estaba ya), y al cabo de unos segundos aparecerá un pequeño pinguino linuxero en la parte superior izquierda de la pantalla. En unos segundos se cargará el entorno gráfico.

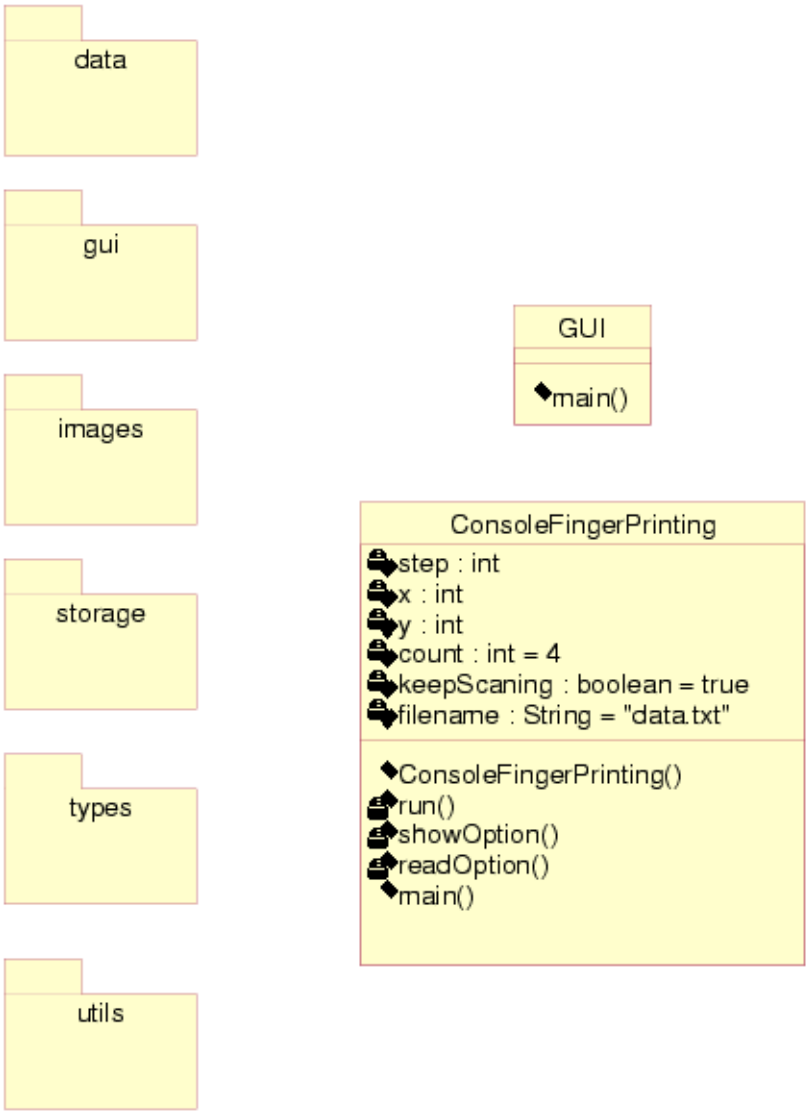
APÉNDICE 2. Diagramas UML de la aplicación Indoor Location

Dado que *Indoor Location* se trata de la aplicación principal y con mayor interés de cuantas se han desarrollado, procederemos a entrar un poco más en detalle en la estructura interna de la misma mostrando los diagramas de clases y de componentes UML.

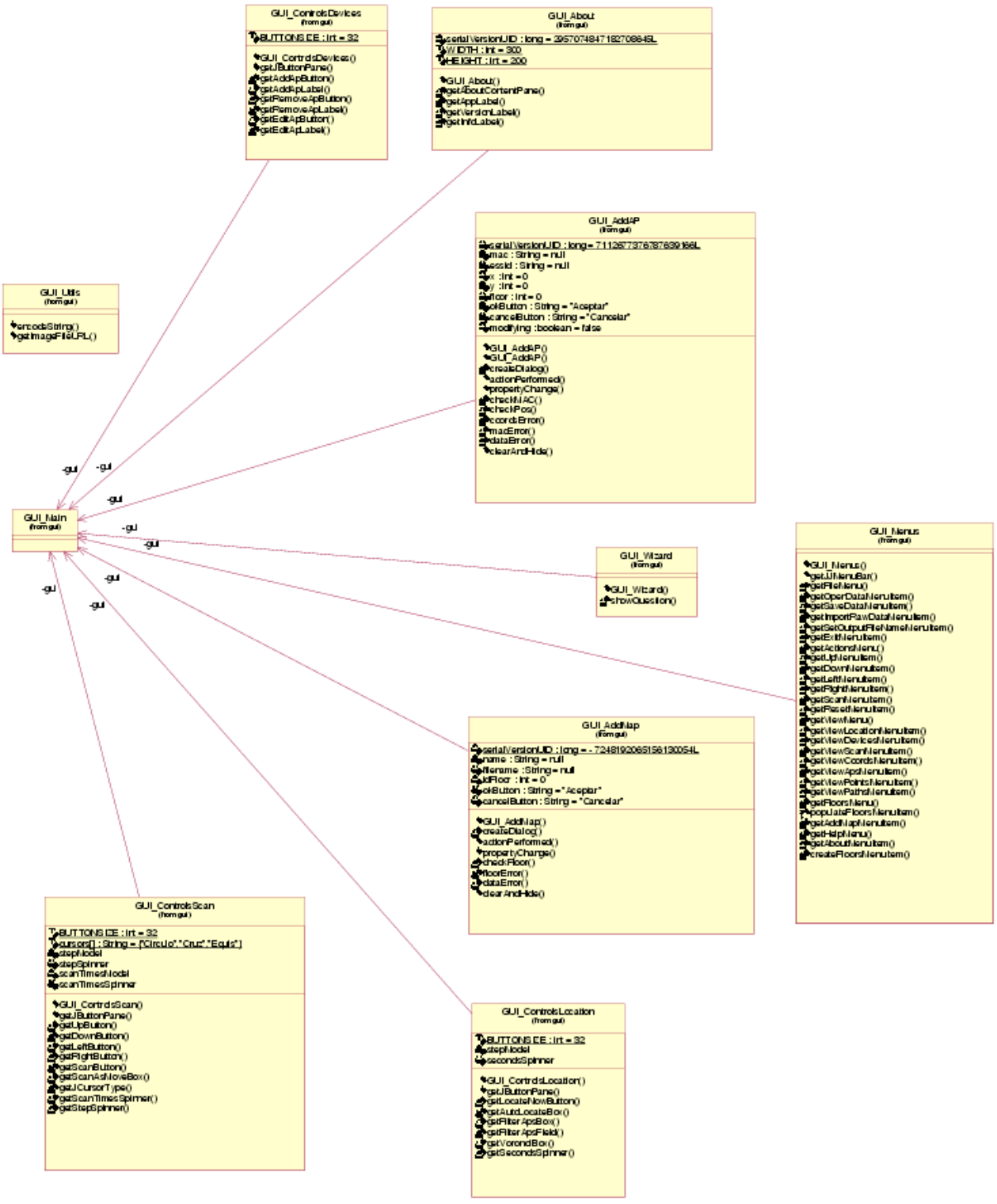
2.1. Diagramas de clases

La aplicación está compuesta por dos paquetes principales, *location* y *wifi*, los cuales, a su vez, tienen nuevos subpaquetes. A continuación se procederá a mostrar cada uno de ellos en detalle.

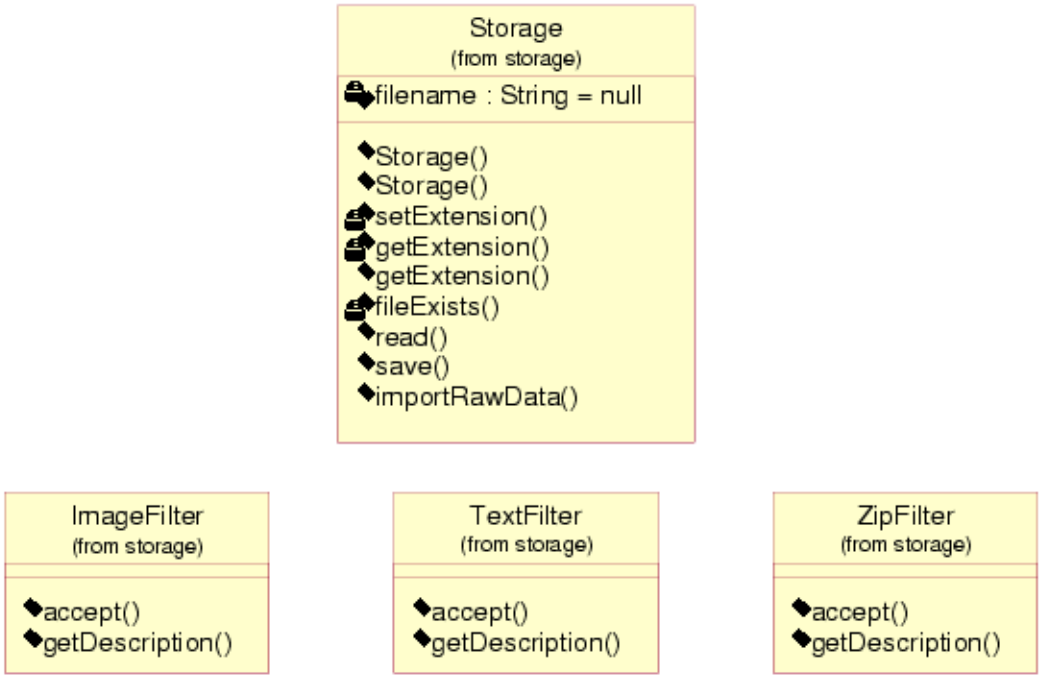
1. Paquete *location*



1.1. Paquete *location.gui*

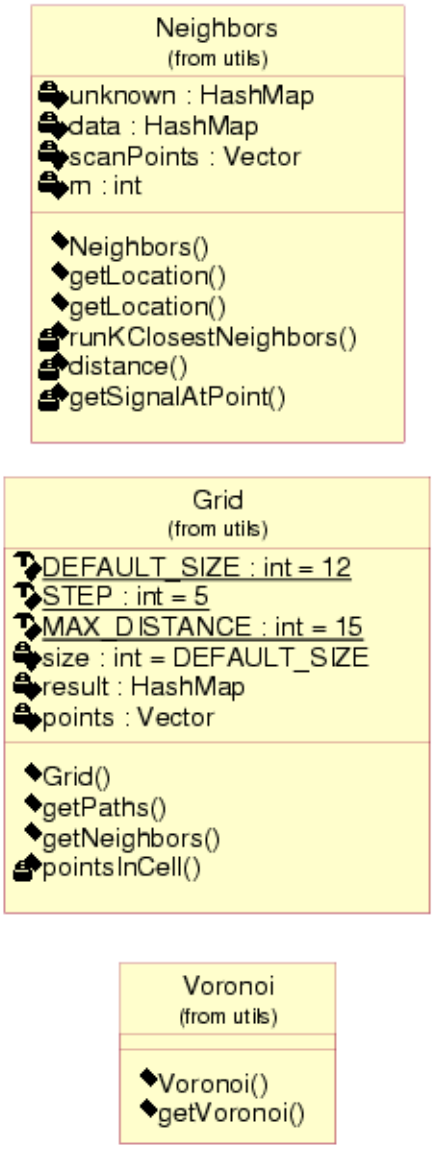


1.2. Paquete *location.storage*



1.3. Paquete *location.types*

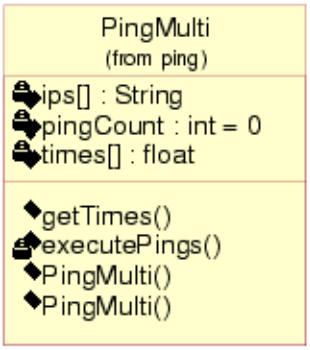
1.4. Paquete *location.utils*

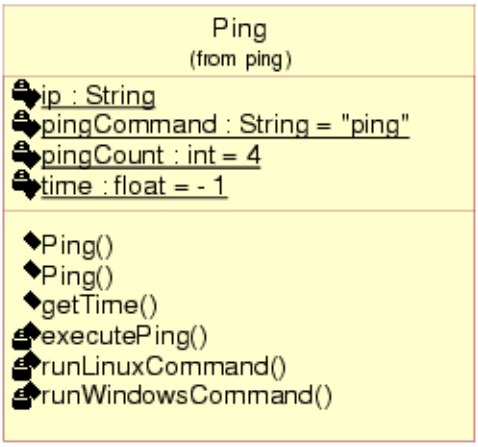


2. Paquete *wifi*

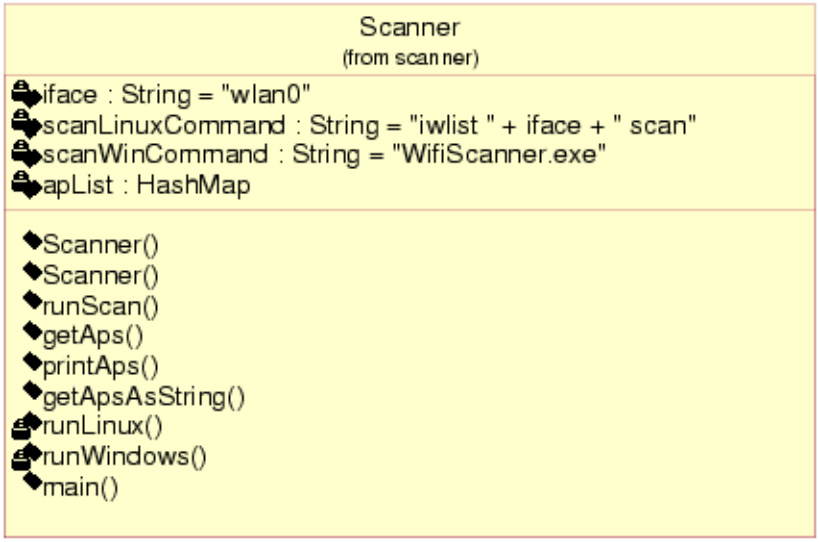
Este paquete se compone de tres subpaquetes, *wifi.ping*, *wifi.scanner* y *wifi.wificonfig*, detallados a continuación.

2.1 Paquete *wifi.ping*

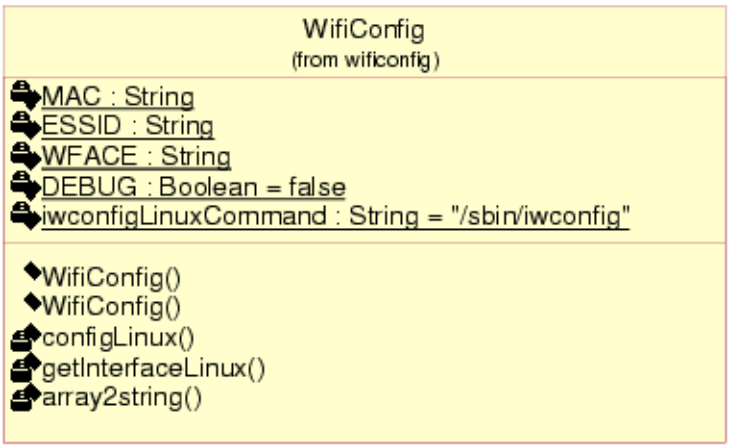




2.2. Paquete *wifi.scanner*

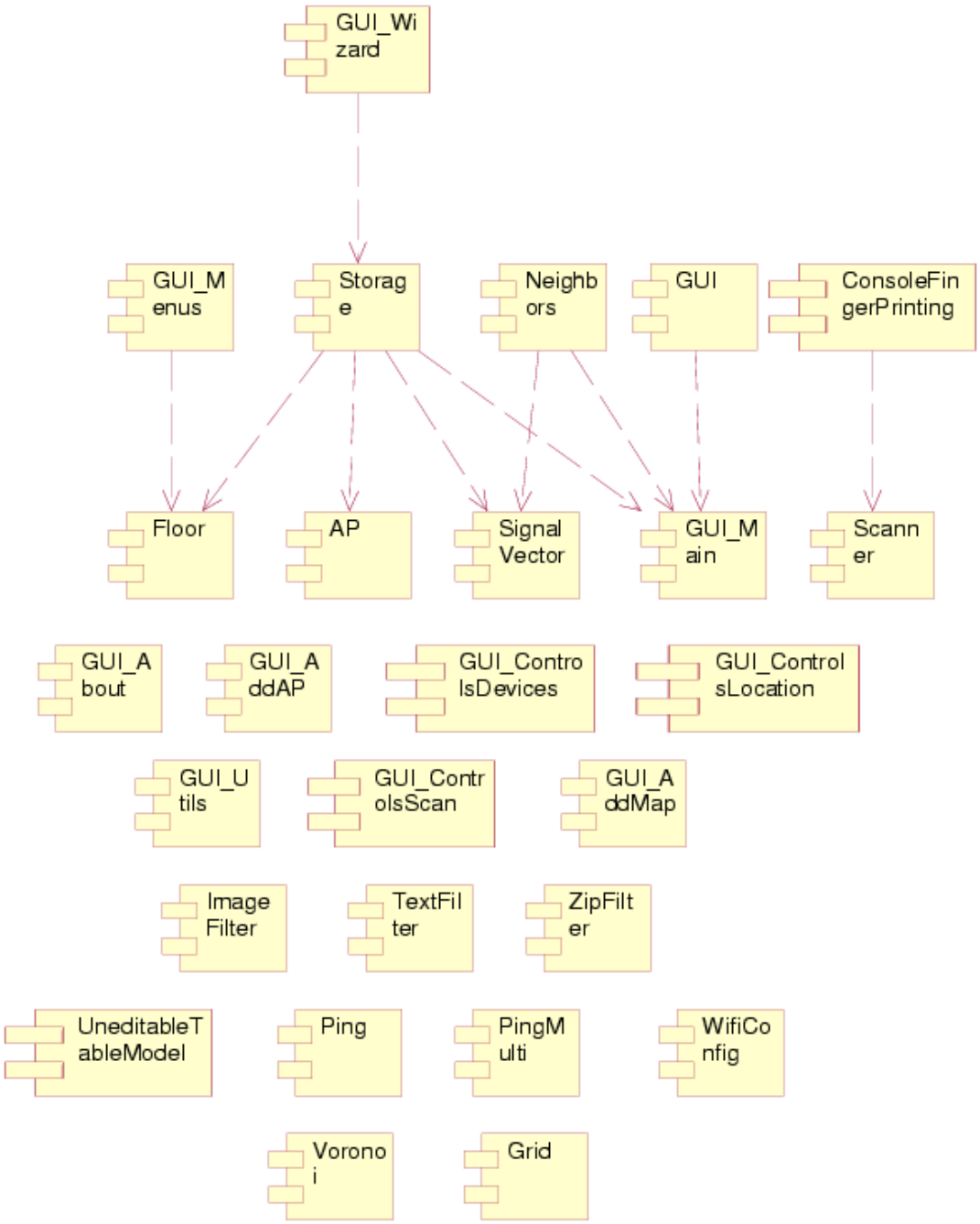


2.3. Paquete *wifi.wificonfig*



2.2. Diagramas de componentes

Una vez detalladas las clases de la aplicación, así como sus atributos y métodos, incluimos un diagrama general con los componentes de la misma, en el que se muestran todas las relaciones existentes entre cada una de las clases.



APÉNDICE 3. Código fuente de la aplicación Indoor Location

Location/src/wifi/ping/Ping.java

```
package wifi.ping;

import java.io.BufferedReader;
import java.io.InputStreamReader;

/**
 * The Class Ping.
 * This class lets you ping a host easily
 */
public class Ping{

	/** The ip. */
	private static String ip;

	/** The ping command. */
	private static String pingCommand = "ping";

	/** How many times the ping should be executed. */
	private static int pingCount = 4;

	/** The average response time. */
	private static float time = -1;

	/**
	 * The Constructor.
	 *
	 * @param ipp the ip to ping
	 */
	public Ping(String ipp) {
		ip = ipp;
		executePing();
	}

	/**
	 * The Constructor.
	 *
	 * @param ipp the ip to ping
	 * @param count times the ping should be executed
	 */
	public Ping(String ipp, int count) {
		ip = ipp;
		pingCount = count;
		executePing();

	}

	/**
	 * Gets the average response time.
	 *
```

```
	 * @return the time
	 */
	public float getTime() {
		return time;
	}

	/**
	 * Execute ping command, depending on os
	 */
	private void executePing() {
		String os = System.getProperty("os.name");
		if (os.contains("Linux")) { runLinuxCommand(); }
		else if (os.contains("Windows")) { runWindowsCommand(); }
	}

	/**
	 * Runs linux ping command.
	 */
	private void runLinuxCommand() {
		try {
			String[] command = {pingCommand, "-c " + pingCount, ip};

			Process p = Runtime.getRuntime().exec(command);
			BufferedReader stdInput = new BufferedReader(new
InputStreamReader(p.getInputStream()));

			String s;
			while ((s = stdInput.readLine()) != null) {
				if (s.contains("rtt")){
			s = s.replaceAll("rtt min/avg/max/mdev = ", "");
			s = s.replaceAll(" ms", "");
			s = s.split("/")[1];
			time = new Float(s);
				}
			}
		} catch (Exception e) {
			System.err.print(e);
		}
	}

	/**
	 * Runs windows ping command.
	 */
	private void runWindowsCommand() {
		try {
			String[] command = {pingCommand, "-n",
Integer.toString(pingCount) , ip};

			Process p = Runtime.getRuntime().exec(command);
			BufferedReader stdInput = new BufferedReader(new
InputStreamReader(p.getInputStream()));

			String s;
			while ((s = stdInput.readLine()) != null) {
				if (s.contains("Media")){
```

```
                              int pos = s.indexOf("Media = ");
                              s = s.substring(pos+8);
                              s = s.replace("ms", "");
                        time = new Float(s);
                              }
                        }
                  } catch (Exception e) {
                        System.err.print(e);
                  }
            }
}
```

Location/src/wifi/ping/PingMulti.java

```
package wifi.ping;

/**
 * The Class PingMulti.
 * This class executes pings to serveral hosts
 */
public class PingMulti {

	/** The ips to ping. */
	private String[] ips;

	/** How many times the ping should be executed. */
	private int pingCount = 0;

	/** The average response times for every ip. */
	private float[] times;

	/**
	 * The Constructor.
	 *
	 * @param ipps the ips to ping
	 */
	public PingMulti(String[] ipps) {
		ips = ipps;
		executePings();
	}

	/**
	 * The Constructor.
	 *
	 * @param ipps the ipps
	 * @param count how many times the ping should be executed
	 */
	public PingMulti(String[] ipps, int count) {
		ips = ipps;
		pingCount = count;
		executePings();
	}

	/**
	 * Gets the average response times for every ip
	 *
	 * @return the times
	 */
	public float[] getTimes() {
		return times;
	}

	/**
	 * Execute pings.
	 */
```

```
	private void executePings() {
		times = new float[ips.length];
		for (int i=0; i<ips.length; i++) {

			// TODO: No me mola nada esta mierda...
			Ping ping;
			if (pingCount == 0 ) { ping = new Ping(ips[i]); }
			else { ping = new Ping(ips[i], pingCount); }
			times[i] = ping.getTime();
		}
	}
}
```

Location/src/wifi/scanner/Scanner.java

```
package wifi.scanner;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Set;

/**
 * The Class Scanner.
 * This class scans the wifi environment using specific os utils
 */
public class Scanner {

	/** The network wireless interface to scan. */
	private String iface = "wlan0";

	/** The scan linux command. */
	private String scanLinuxCommand = "iwlist " + iface +  " scan";

	/** The scan windows command. */
	private String scanWinCommand = "WifiScanner.exe";

	/** The access points found. */
	private HashMap<String, String[]> apList;


	/**
	 * The Constructor.
	 */
	public Scanner() {
		apList = new HashMap<String, String[]>();
		runScan();
	}

	public Scanner(String iface) {
		this.iface = iface;
		this.scanLinuxCommand = "iwlist " + iface +  " scan";
		apList = new HashMap<String, String[]>();
		runScan();
	}

	/**
	 * Runs the scanner
	 */
	public void runScan() {
		String os = System.getProperty("os.name");
		if (os.contains("Linux")) { runLinux(); }
		else if (os.contains("Windows")) { runWindows(); }
	}
```

```
	/**
	 * Gets the Access Points MACs.
	 *
	 * @return the Access Points MACs as String[]
	 */
	public HashMap<String, String[]> getAps() {
		return apList;
	}

	/**
	 * Print scanner results
	 */
	public void printAps() {
		Set apKeys = apList.keySet();
		Iterator It = apKeys.iterator();
		while (It.hasNext()) {
			String mac = (String)(It.next());
			String[] info = apList.get(mac);
			System.out.println(mac + " -- " + info[0] + " -- " +
info[1]);
		}
	}

	public String getApsAsString() {
		String out = "";
		Set apKeys = apList.keySet();
		Iterator It = apKeys.iterator();
		while (It.hasNext()) {
			String mac = (String)(It.next());
			String[] info = apList.get(mac);
			out += mac + "\t" + info[0] + "\t" + info[1] + "\n";
		}
		return out;
	}


	/**
	 * Runs linux scanner.
	 */
	private void runLinux() {
		try {

			Process p = Runtime.getRuntime().exec(scanLinuxCommand);
			BufferedReader stdInput = new BufferedReader(new
InputStreamReader(p.getInputStream()));

			String s;
			String mac = "";
			String essid = "";
			String signal = "";

			while ((s = stdInput.readLine()) != null) {
				if (s.contains("Address")) {
					// We get the AP MAC
					s = s.toUpperCase();
```

```
                              int pos1 = s.indexOf("ADDRESS: ");
                              mac = s.substring(pos1 + 9);
                        }

                        if (s.contains("ESSID")) {
                              // We get the AP ESSID
                              int pos1 = s.indexOf("ESSID:");
                              s = s.substring(pos1 + 6);
                              essid = s.replaceAll("\"","");
                        }

                        if (s.contains("Quality")) {
                              // We get the signal strength
                              s = s.toUpperCase();
                              int pos1 = s.indexOf("QUALITY=");
                              int pos2 = s.indexOf("SIGNAL LEVEL=");
                              signal = s.substring(pos1 + 8, pos2 - 2);
                        }

                        // Ok, the AP info is read, add it to the list
                        if (mac != "" && essid != "" && signal != "") {

                              String[] info = {essid, signal};

                              // If we've seen this AP before, update it if
the essid is not hidden
                              if (apList.containsKey(mac)) {
                                    String[] info2 = apList.get(mac);
                                    if (info2[0].equals("<hidden>"))
                                          apList.put(mac, info);
                              }
                              else apList.put(mac, info);

                              mac = "";
                              essid = "";
                              signal = "";
                        }
                  }
            } catch (Exception e) {
                  System.err.print(e);
            }
      }

      /**
       * Runs windows scanner.
       */
      private void runWindows() {
            try {

                  Process p = Runtime.getRuntime().exec(scanWinCommand);
                  BufferedReader stdInput = new BufferedReader(new
InputStreamReader(p.getInputStream()));

                  String s;
                  String mac = "";
```

```
                String essid = "";
                String signal = "";

                while ((s = stdInput.readLine()) != null) {
                      if (s.contains("Address")) {
                            // We get the AP MAC
                            s = s.toUpperCase();
                            int pos1 = s.indexOf("ADDRESS: ");
                            mac = s.substring(pos1 + 9);
                      }

                      if (s.contains("ESSID")) {
                            // We get the AP ESSID
                            int pos1 = s.indexOf("ESSID: ");
                            essid = s.substring(pos1 + 7);
                      }

                      if (s.contains("Signal strength")) {
                            // We get the signal strength
                            int pos1 = s.indexOf("Signal strength: ");
                            signal = s.substring(pos1 + 17);
                      }

                      // Ok, the AP info is read, add it to the list
                      if (mac != "" && essid != "" && signal != "") {

                            String[] info = {essid, signal};

                            // If we've seen this AP before, update it if
the essid is not hidden
                            if (apList.containsKey(mac)) {
                                  String[] info2 = apList.get(mac);
                                  if (info2[0].equals(""))
                                        apList.put(mac, info);
                            }
                            else apList.put(mac, info);

                            mac = "";
                            essid = "";
                            signal = "";
                      }
                }
          } catch (Exception e) {
                System.err.print(e);
                System.out.println("Por favor, instale WifiScanner.exe en
el PATH de Windows");
          }
    }


    public static void main(String[] args) {
          Scanner scanner = new Scanner();
          scanner.printAps();
    }
} // End  Class
```

Location/src/wifi/wificonfig/WifiConfig.java

```
package wifi.wificonfig;

import java.io.*;

/**
 * The Class WifiConfig.
 * This class is used to configure the wireless network card.
 */
public class WifiConfig {

	/** The MAC. */
	private static String MAC;

	/** The ESSID. */
	private static String ESSID;

	/** The WIRELESS INTERFACE. */
	private static String WFACE;

	/** The DEBUG. */
	private static Boolean DEBUG = false;

	/** The iwconfig linux command. */
	private static String iwconfigLinuxCommand = "/sbin/iwconfig";
	//private String iwconfigWindowsCommand = "ipconfig";

	/**
	 * The Constructor.
	 *
	 * @param essid the essid to connect to
	 * @param mac the mac to connect to
	 */
	public WifiConfig(String mac, String essid) {
		MAC = mac;
		ESSID = essid;
		WFACE = getInterfaceLinux();

		configLinux();
	}

	/**
	 * The Constructor.
	 *
	 * @param debug do you want debug information?
	 * @param essid the essid to connect to
	 * @param mac the mac to connect to
	 */
	public WifiConfig(String mac, String essid, Boolean debug) {
		MAC = mac;
		ESSID = essid;
		WFACE = getInterfaceLinux();
```

```
		DEBUG = debug;

		configLinux();
	}

	/**
	 * Configure the wireless interface with the new data on linux.
	 */
	private void configLinux() {
		try
		{
			// Enter new ESSID
			String[] command1 = {iwconfigLinuxCommand, WFACE, "essid",
ESSID};
			Process p1 = Runtime.getRuntime().exec(command1);

			// Debug information: command, stdout and errout
			if (DEBUG) {
				System.out.println(array2string(command1));
				BufferedReader stdInput1 = new BufferedReader(new
InputStreamReader(p1.getInputStream()));
				String s1;
				while ((s1 = stdInput1.readLine()) != null) {
					System.out.println(s1);
				}
				BufferedReader errorInput1 = new BufferedReader(new
InputStreamReader(p1.getErrorStream()));
				String e1;
				while ((e1 = errorInput1.readLine()) != null) {
					System.out.println(e1);
				}
			}

			// Enter new MAC
			String[] command2 = {iwconfigLinuxCommand, WFACE, "ap",
MAC};
			Process p2 = Runtime.getRuntime().exec(command2);

			// Debug information: command, stdout and errout
			if (DEBUG) {
				System.out.println(array2string(command2));
				BufferedReader stdInput2 = new BufferedReader(new
InputStreamReader(p2.getInputStream()));
				String s2;
				while ((s2 = stdInput2.readLine()) != null) {
					System.out.println(s2);
				}
				BufferedReader errorInput2 = new BufferedReader(new
InputStreamReader(p2.getErrorStream()));
				String e2;
				while ((e2 = errorInput2.readLine()) != null) {
					System.out.println(e2);
				}
			}
```

```
            } catch (Exception e) {
                  System.err.print(e);
            }
      }

      /**
       * Gets the active wireless interface on linux.
       *
       * @return the active wireless interface
       */
      private static String getInterfaceLinux() {
            String tmp;
            try {
                  BufferedReader input = new BufferedReader(new
FileReader("/proc/net/wireless"));
                  input.readLine();
                  input.readLine();
                  tmp = input.readLine();
                  tmp = tmp.split(":")[0];
                  tmp = tmp.trim();
            }
            catch (IOException e) {
                  tmp = "";
            }
            return tmp;
      }

      /**
       * Convert an array to a string
       *
       * @param a the a
       *
       * @return the string
       */
      private static String array2string(String[] a) {
            String separator = " ";
             StringBuffer result = new StringBuffer();
                if (a.length > 0) {
                    result.append(a[0]);
                    for (int i=1; i<a.length; i++) {
                        result.append(separator);
                        result.append(a[i]);
                    }
                }
                return result.toString();
      }
}
```

Location/src/location/ConsoleFingerPrinting.java

```
package location;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;

import wifi.scanner.Scanner;

/**
 * The Class ConsoleFingerPrinting.
 *
 * @author Adolfo González Blázquez <code@infinicode.org>
 */
public class ConsoleFingerPrinting {

      private int step;
      private int x, y;
      private int count = 4;
      private boolean keepScaning = true;
      private String filename = "data.txt";

      /**
       *
       */
      public ConsoleFingerPrinting() {
            this.step = 120;

            try {
              BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
              String str = "";

              System.out.print("Nombre ficherol > ");
            str = in.readLine();
            if (str != "") {
            filename = str;
            }

            System.out.print("X inicial > ");
            str = in.readLine();
            try {
            x = Integer.parseInt(str);
            } catch (NumberFormatException e) {
            }

            System.out.print("Y inicial > ");
            str = in.readLine();
            try {
            y = Integer.parseInt(str);
```

```
            } catch (NumberFormatException e) {
            }
          } catch (IOException e) {
          }

            try {
                  run();
            } catch (InterruptedException e) {
                  e.printStackTrace();
            }
      }

      private void run()  throws InterruptedException  {


            String aps;

            while (keepScaning) {

                  for (int i = 0; i < count; i++) {
                        Scanner scan = new Scanner();
                        aps = scan.getApsAsString();
                        System.out.println(x + "\t" + y);
                        System.out.println(aps);
                        try {
                          BufferedWriter out = new BufferedWriter(new
FileWriter(filename, true));
                          out.write(x + "\t" + y);
                          out.write("\n");
                          out.write(aps);
                          out.write("\n");
                          out.close();
                      } catch (IOException e) {
                          e.printStackTrace();
                      }
                  }

//                try {
//                      Runtime.getRuntime().exec("sh clear");
//                } catch (IOException e) {
//                      e.printStackTrace();
//                }
                  showOption();
                  readOption();
            }

      }

      private void showOption() {
            System.out.println("(1) Arriba");
            System.out.println("(2) Derecha");
            System.out.println("(3) Abajo");
            System.out.println("(4) Izquierda");
            System.out.println();
            System.out.println("(0) Salir");
```

```
	}

	private boolean readOption() {
		boolean ret = false;
		try {
			BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
			String str = "";
			int op = 9;

			System.out.print("> ");
			str = in.readLine();
			try {
			op = Integer.parseInt(str);
			} catch (NumberFormatException e) {
			//e.printStackTrace();
			}
			switch (op) {
				case 1: y += step; ret = true; break;
				case 2: x += step; ret = true; break;
				case 3: y -= step; ret = true; break;
				case 4: x -= step; ret = true; break;
				case 0: keepScaning = false; break;
					default: break;
				}
			} catch (IOException e) {
			}
		return ret;
	}

	public static void main(String[] args) {
		new ConsoleFingerPrinting();
	}

}
```

Location/src/location/GUI.java

```
package location;

import javax.swing.SwingUtilities;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;

import location.gui.GUI_Main;

public class GUI {



/**********************************************************************
**
       * Launches this application

**********************************************************************
*/
      public static void main(String[] args) {

            try {
                // Set System L&F
              UIManager.setLookAndFeel(
                  UIManager.getSystemLookAndFeelClassName());
           }
           catch (UnsupportedLookAndFeelException e) {
              // handle exception
           }
           catch (ClassNotFoundException e) {
              // handle exception
           }
           catch (InstantiationException e) {
              // handle exception
           }
           catch (IllegalAccessException e) {
              // handle exception
           }

             SwingUtilities.invokeLater(new Runnable() {
                   public void run() {
                         new GUI_Main();
                   }
             });
      }

}
```

Location/src/location/storage/Storage.java

```
package location.storage;


import java.awt.Point;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.HashMap;
import java.util.Vector;
import java.util.zip.ZipEntry;
import java.util.zip.ZipInputStream;
import java.util.zip.ZipOutputStream;

import javax.swing.JOptionPane;

import location.gui.GUI_Main;
import location.types.AP;
import location.types.Floor;
import location.types.SignalVector;

public class Storage {

	private GUI_Main gui = null;
	private String filename = null;

	public Storage(GUI_Main gui) {
		this.gui = gui;
		filename = gui.getOutputFileName();
	}

	public Storage(GUI_Main gui, String filename) {
		this.gui = gui;
		this.filename = filename;
	}

	private void setExtension() {
		if (getExtension() != "zip") {
			int i = filename.lastIndexOf('.');

			if (i > 0 &&  i < filename.length() - 1) {
				filename = filename.substring(0, i);
			}
			filename += ".zip";
		}
	}
```

```
      private String getExtension() {
        String ext = null;

        int i = filename.lastIndexOf('.');

        if (i > 0 &&  i < filename.length() - 1) {
            ext = filename.substring(i+1).toLowerCase();
        }
        return ext;
    }

      public static String getExtension(File f) {
            String name = f.getName();
            String ext = null;

        int i = name.lastIndexOf('.');

        if (i > 0 &&  i < name.length() - 1) {
            ext = name.substring(i+1).toLowerCase();
        }
        return ext;
      }

      private boolean fileExists() {
            File f = new File(filename);

            if ( f.exists()) {
                  int n = JOptionPane.showConfirmDialog(
                              gui,
                              "El fichero " + f.getName() + " ya
existe.\n¿Quiere reemplazarlo?",
                              "El fichero ya existe",
                              JOptionPane.YES_NO_OPTION);

                  if (n == JOptionPane.NO_OPTION)
                        return false;
                  else if (n != JOptionPane.YES_OPTION)
                        return false;
            }
            return true;
      }

      @SuppressWarnings("unchecked")
      public boolean read() {
            try {

                  // Open the zip file
                  FileInputStream f = new FileInputStream(filename);
                  ZipInputStream zin = new ZipInputStream(new
BufferedInputStream(f));

                  // Read Floors
                  zin.getNextEntry();
                  ObjectInputStream ain = new ObjectInputStream(zin);
                  gui.setFloors((HashMap<Integer, Floor>) ain.readObject());
```

```
                // Close readers
                ain.close();
                zin.close();

                return true;

            } catch (Exception e) {
                e.printStackTrace();
                return false;
            }
        }

        public boolean save() {

            // Set the correct extension and check if file already exists
            setExtension();
            fileExists();

            try {

                // Create zip file
                FileOutputStream f = new FileOutputStream(filename);
                ZipOutputStream zout = new ZipOutputStream(new
BufferedOutputStream(f));

                zout.putNextEntry(new ZipEntry("data.dat"));
                ObjectOutputStream aout = new ObjectOutputStream(zout);
                aout.writeObject(gui.getFloors());
                aout.flush();

                // Close writers
                aout.close();
                zout.close();

                return true;

            } catch (Exception e) {
                e.printStackTrace();
                return false;
            }
        }

        public boolean importRawData() {

            int scanCount = 0;
            Vector<Point> points = new Vector<Point>();
            HashMap<String, HashMap<Point, SignalVector>> tabla = new
HashMap<String, HashMap<Point, SignalVector>>();
            Vector<AP> aps = new Vector<AP>();  // {mac, essid}

            try {
              BufferedReader in = new BufferedReader(new
```

```
FileReader(filename));

               String l;
               int x = -99;
               int y = -99;
               String mac = "";
               String essid = "";
               double signal = -99;

                   while ((l = in.readLine()) != null) {
                         String[] aux = l.split("\t");
                         if (aux.length == 2) {
                               // PUNTO
                               x = Integer.parseInt(aux[0]);
                               y = Integer.parseInt(aux[1]);
                               scanCount++;
                               Point p = new Point(x,y);
                               if (!points.contains(p)) points.add(p);
                         }
                         else if (aux.length == 3) {
                               // MAC
                               mac = aux[0];
                               essid = aux[1];
                               signal =
Integer.parseInt(aux[2].split("/")[0]);
                               Point p = new Point(x, y);

                               if (tabla.containsKey(mac)) {
                                     HashMap<Point, SignalVector> v =
(HashMap<Point, SignalVector>)tabla.get(mac);
                                     if ( v.containsKey(p) ) {
//                                         signal = (signal + v.get(p)) / 2;
//                                         v.put(p, signal);
                                           v.get(p).add(signal);
                                     }
                                     else v.put(p, new SignalVector(signal));

                               } else {

                                     HashMap<Point, SignalVector> v = new
HashMap<Point, SignalVector>();
                                     v.put(p, new SignalVector(signal));
                                     tabla.put(mac, v);
                                     AP ap = new AP(mac, essid, -1, -1,
gui.getActiveFloor().getIdFloor());
                                     aps.add(ap);
                               }
                         }
                         else {
                               // NADA
                               x = -99;
                               y = -99;
                               mac = "";
                               signal = -99;
                         }
```

```
                }

            in.close();


            gui.setData(tabla);
            gui.setScanAps(aps);
            gui.setScanPoints(points);
            gui.setScanCount(scanCount);

            return true;

        } catch (Exception e) {
          e.printStackTrace();
          return false;
        }
    }
}
```

Location/src/location/storage/ZipFilter.java

```
package location.storage;

import java.io.File;

import javax.swing.filechooser.FileFilter;



public class ZipFilter extends FileFilter {

    //Accept all directories and zip files
    public boolean accept(File f) {
        if (f.isDirectory()) {
            return true;
        }

        String extension = Storage.getExtension(f);
        if (extension != null) {
            if (extension.equals("zip")) {
                    return true;
            } else {
                return false;
            }
        }

        return false;
    }

    //The description of this filter
    public String getDescription() {
        return "Ficheros Zip";
    }
}
```

Location/src/location/storage/ImageFilter.java

```
package location.storage;

import java.io.File;

import javax.swing.filechooser.FileFilter;



public class ImageFilter extends FileFilter {

    //Accept all directories and all gif, jpg, tiff, or png files.
    public boolean accept(File f) {
        if (f.isDirectory()) {
            return true;
        }

        String extension = Storage.getExtension(f);
        if (extension != null) {
            if (extension.equals("tiff") ||
                extension.equals("tif") ||
                extension.equals("gif") ||
                extension.equals("jpeg") ||
                extension.equals("jpg") ||
                extension.equals("png")) {
                    return true;
            } else {
                return false;
            }
        }

        return false;
    }

    //The description of this filter
    public String getDescription() {
        return "Imagenes";
    }
}
```

Location/src/location/storage/TextFilter.java

```
package location.storage;

import java.io.File;

import javax.swing.filechooser.FileFilter;



public class TextFilter extends FileFilter {

    //Accept all directories and txt files
    public boolean accept(File f) {
        if (f.isDirectory()) {
            return true;
        }

        String extension = Storage.getExtension(f);
        if (extension != null) {
            if (extension.equals("txt") ||
            extension.equals("text")) {
                    return true;
            } else {
                return false;
            }
        }

        return false;
    }

    //The description of this filter
    public String getDescription() {
        return "Ficheros de texto";
    }
}
```

Location/src/location/utils/Voronoi.java

```
package location.utils;

import java.awt.Point;


public class Voronoi {

	private Point point;

	public Voronoi(Point point) {
		this.point = point;
	}

	public Point getVoronoi() {
		return point;
	}
}
```

Location/src/location/utils/Neighbors.java

```
package location.utils;

import java.awt.Point;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Vector;

import location.gui.GUI_Main;
import location.types.SignalVector;

public class Neighbors {

	private HashMap<String, String[]> unknown;                    // The
data received in this place
	private HashMap<String, HashMap<Point, SignalVector>> data; // The
data from location.gui
	private Vector<Point> scanPoints;                                    //
Every point from location.gui
	private int m;
		// Number of aps located from here

	public Neighbors(GUI_Main gui) {
		unknown = new HashMap<String, String[]>();
		data = gui.getData();
		scanPoints = gui.getScanPoints();
		m = 0;
	}

	public Point getLocation(HashMap<String, String[]> unknown) {

		Iterator it = unknown.entrySet().iterator();
		while (it.hasNext()) {
			Map.Entry e = (Map.Entry)it.next();
			String mac = (String)e.getKey();

			// Filter MACs that are not on our database
			if (data.containsKey(mac))
				this.unknown.put(mac, (String[])e.getValue());
		}

		m = this.unknown.size();
		if (m > 0)
			return runKClosestNeighbors();
		else
			return new Point(0,0);
	}

	public Point getLocation(HashMap<String, String[]> unknown, String
filter) {
```

```
            Iterator it = unknown.entrySet().iterator();
            while (it.hasNext()) {
                  Map.Entry e = (Map.Entry)it.next();
                  String mac = (String)e.getKey();
                  String essid = ((String[])e.getValue())[0];

                  // Filter MACs that are not on our database and those
whose ESSID we don't like
                  if (data.containsKey(mac) && essid.contains(filter))
                        this.unknown.put(mac, (String[])e.getValue());
            }

            m = this.unknown.size();
            if (m > 0)
                  return runKClosestNeighbors();
            else
                  return new Point(0,0);
      }



      /*
       * Runs the k-closest neighbors algorithm, and returns the desired
point
       */
      private Point runKClosestNeighbors() {

            double sum1x = 0;
            double sum1y = 0;

            double sum2 = 0;

            double dist;
            double mindist = Integer.MAX_VALUE;
            Point minpoint = null;

            for (int k = 0; k < scanPoints.size(); k++) {
                        Point p = scanPoints.elementAt(k);
                        dist = distance(p);
                        if (dist < mindist) {
                              minpoint = p;
                              mindist = dist;
                  }

                  if (minpoint == null) return new Point(0,0);

                  sum1x += (1 / mindist) * minpoint.getX();
                  sum1y += (1 / mindist) * minpoint.getY();

                  sum2 += 1 / mindist;
            }

            Point location = new Point((int)(sum1x / sum2), (int)(sum1y /
sum2));
```

```
            return location;
      }

      // d(Z, Zi) = 1/M * raiz(SUM<j=1,M> (cuadrado(RSSj(x,y) -
RSSj(xi,yi))))
      private double distance(Point i) {
            double sum = 0;
            double sig;
            double sigv;
            String mac;
            String info[];

            Iterator it = unknown.entrySet().iterator();
            while (it.hasNext()) {
                  Map.Entry e = (Map.Entry)it.next();

                  mac = (String)e.getKey();
                  sigv = getSignalAtPoint(mac, i);

                  if (sigv == Integer.MAX_VALUE-1)
                        return Integer.MAX_VALUE-1;

                  info = (String[])e.getValue();
                  sig = Integer.parseInt(info[1].split("/")[0]);

                  sum += Math.pow((sig - sigv), 2);
            }

            return (1f/m) * Math.sqrt(sum);
      }

      private double getSignalAtPoint(String mac, Point i) {
            HashMap<Point, SignalVector> v = data.get(mac);
            if (v != null) {
                  if (v.containsKey(i))
                        return v.get(i).getMeanValue();
                  else
                        return Integer.MAX_VALUE-1;
            }
            else
                        return Integer.MAX_VALUE-1;
      }
}
```

Location/src/location/utils/Grid.java

```
package location.utils;

import java.awt.Point;
import java.util.HashMap;
import java.util.Vector;

public class Grid {

	static final int DEFAULT_SIZE = 12;
	static final int STEP = 5;
	static final int MAX_DISTANCE = 15;

	private int size = DEFAULT_SIZE;
	private HashMap<Point, Vector<Point>> result;
	private Vector<Point> points;

	public Grid(Vector<Point> points) {
		this.points = points;
		result = new HashMap<Point, Vector<Point>>();
	}

	public HashMap<Point, Vector<Point>> getPaths() {

		if (points.size() < 2) return result;

		for (int i = 0 ; i < points.size(); i++) {
			Point p = points.elementAt(i);
			result.put(p, getNeighbors((int)p.getX(), (int)p.getY()));
		}
		return result;
	}

	public Vector<Point> getNeighbors(int x, int y) {

		Vector<Point> neighbors = new Vector<Point>();

		if (points.size() < 2) return neighbors;

		neighbors.addAll(pointsInCell(x, y + size));
		neighbors.addAll(pointsInCell(x + size, y + size));
		neighbors.addAll(pointsInCell(x + size, y));
		neighbors.addAll(pointsInCell(x + size, y - size));
		neighbors.addAll(pointsInCell(x, y - size));
		neighbors.addAll(pointsInCell(x - size, y - size));
		neighbors.addAll(pointsInCell(x - size, y));
		neighbors.addAll(pointsInCell(x - size, y + size));

		if (neighbors.size() < 2 && size < MAX_DISTANCE) {
			size += STEP;
			neighbors = getNeighbors(x, y);
		}
```

```
		size = DEFAULT_SIZE;

		return neighbors;
	}

	private Vector<Point> pointsInCell(int x, int y) {

		Vector<Point> puntos = new Vector<Point>();
		Point p = new Point(x,y);
		for (int i = 0; i < size; i++)
			for (int j = 0; j < size; j++) {
				p.setLocation(x+i, y+j);
				if (points.contains(p))
					puntos.add(new Point(x+i, y+j));
			}
		return puntos;
	}

}
```

Location/src/location/gui/GUI_ControlsScan.java

```
package location.gui;

import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.KeyEvent;

import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSpinner;
import javax.swing.SpinnerNumberModel;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class GUI_ControlsScan {

	private GUI_Main gui;

	private JPanel jButtonPanel = null;

	static final int BUTTONSIZE = 32;

	// Buttons
	private JButton upButton = null;
	private JButton leftButton = null;
	private JButton downButton = null;
	private JButton rightButton = null;
	private JButton scanButton = null;

	private JCheckBox scanAsMoveBox;

	private SpinnerNumberModel stepModel;
	private JSpinner stepSpinner;

	private SpinnerNumberModel scanTimesModel;
	private JSpinner scanTimesSpinner;


	private JComboBox jCursorType = null;
	static final String[] cursors = { "Circulo", "Cruz", "Equis" };
```

```
public GUI_ControlsScan(GUI_Main gui) {
	this.gui = gui;
}

public JPanel getJButtonPane() {
	if (jButtonPanel == null) {
		jButtonPanel = new JPanel();
		jButtonPanel.setLayout(new GridBagLayout());

		GridBagConstraints c = new GridBagConstraints();
		c.insets = new Insets(5, 5, 5, 5);
		c.fill = GridBagConstraints.HORIZONTAL;

		// Controls
		c.gridx = 1;
		c.gridy = 0;
		jButtonPanel.add(getUpButton(), c);

		c.gridx = 0;
		c.gridy = 1;
		jButtonPanel.add(getLeftButton(), c);

		c.gridx = 1;
		c.gridy = 1;
		jButtonPanel.add(getDownButton(), c);

		c.gridx = 2;
		c.gridy = 1;
		jButtonPanel.add(getRightButton(), c);

		c.gridx = 1;
		c.gridy = 2;
		jButtonPanel.add(getScanButton(), c);


		// Scan as move
		c.gridx = 0;
		c.gridy = 3;
		c.gridwidth = 6;
		jButtonPanel.add(getScanAsMoveBox(), c);
		c.gridwidth = 1;


		// Separator
		c.gridx = 3;
		c.gridy = 0;
		jButtonPanel.add(new JLabel("          "), c);

		// Options
		c.gridx = 4;
		c.gridy = 0;
		jButtonPanel.add(new JLabel("Paso: "), c);
		c.gridx = 5;
		c.gridy = 0;
		jButtonPanel.add(getStepSpinner(), c);
```

```
                c.gridx = 4;
                c.gridy = 1;
                jButtonPanel.add(new JLabel("Escaneos: "), c);
                c.gridx = 5;
                c.gridy = 1;
                jButtonPanel.add(getScanTimesSpinner(), c);

                c.gridx = 4;
                c.gridy = 2;
                jButtonPanel.add(new JLabel("Cursor: "), c);
                c.gridx = 5;
                c.gridy = 2;
                jButtonPanel.add(getJCursorType(), c);
          }
          return jButtonPanel;
    }

    private JButton getUpButton() {
          if (upButton == null) {

                Icon icon = new
ImageIcon(GUI_Utils.getImageFileURL(gui.imagesDir + "up.png"));
                upButton = new JButton(icon);
                upButton.setPreferredSize(new Dimension(BUTTONSIZE,
BUTTONSIZE));
                upButton.setMnemonic(KeyEvent.VK_UP);
                upButton.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            gui.moveCursorUp();
                      }
                });
          }
          return upButton;
    }

    private JButton getDownButton() {
          if (downButton == null) {
                Icon icon = new
ImageIcon(GUI_Utils.getImageFileURL(gui.imagesDir + "down.png"));
                downButton = new JButton(icon);
                downButton.setPreferredSize(new Dimension(BUTTONSIZE,
BUTTONSIZE));
                downButton.setMnemonic(KeyEvent.VK_DOWN);
                downButton.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            gui.moveCursorDown();
                      }
                });
          }
          return downButton;
    }

    private JButton getLeftButton() {
          if (leftButton == null) {
```

```
                  Icon icon = new
ImageIcon(GUI_Utils.getImageFileURL(gui.imagesDir + "left.png"));
                  leftButton = new JButton(icon);
                  leftButton.setPreferredSize(new Dimension(BUTTONSIZE,
BUTTONSIZE));
                  leftButton.setMnemonic(KeyEvent.VK_LEFT);
                  leftButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              gui.moveCursorLeft();
                        }
                  });
            }
            return leftButton;
      }

      private JButton getRightButton() {
            if (rightButton == null) {
                  Icon icon = new
ImageIcon(GUI_Utils.getImageFileURL(gui.imagesDir + "right.png"));
                  rightButton = new JButton(icon);
                  rightButton.setPreferredSize(new Dimension(BUTTONSIZE,
BUTTONSIZE));
                  rightButton.setMnemonic(KeyEvent.VK_RIGHT);
                  rightButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              gui.moveCursorRight();
                        }
                  });
            }
            return rightButton;
      }

      private JButton getScanButton() {
            if (scanButton == null) {
                  Icon icon = new
ImageIcon(GUI_Utils.getImageFileURL(gui.imagesDir + "scan.png"));
                  scanButton = new JButton(icon);
                  scanButton.setPreferredSize(new Dimension(BUTTONSIZE,
BUTTONSIZE));
                  scanButton.setMnemonic(KeyEvent.VK_S);
                  scanButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              gui.runScan();
                        }
                  });
            }
            return scanButton;
      }

      private JSpinner getStepSpinner() {
            if (stepSpinner == null) {
                  stepModel = new SpinnerNumberModel(gui.step / gui.scale,
1, 1000, 1);
                  stepSpinner = new JSpinner(stepModel);
```

```
                  stepSpinner.addChangeListener(new ChangeListener() {
                        public void stateChanged(ChangeEvent e) {
                              gui.step = stepModel.getNumber().intValue() *
gui.scale;
                        }
                  }
                  );
            }
            return stepSpinner;
      }

      private JSpinner getScanTimesSpinner() {
            if (scanTimesSpinner == null) {
                  scanTimesModel = new SpinnerNumberModel(gui.scanTimes, 1,
10, 1);
                  scanTimesSpinner = new JSpinner(scanTimesModel);

                  scanTimesSpinner.addChangeListener(new ChangeListener() {
                        public void stateChanged(ChangeEvent e) {
                              gui.scanTimes =
scanTimesModel.getNumber().intValue();
                        }
                  }
                  );
            }
            return scanTimesSpinner;
      }

      private JCheckBox getScanAsMoveBox() {
            if (scanAsMoveBox == null) {
                  scanAsMoveBox = new JCheckBox(" Escanear al moverse");

                  scanAsMoveBox.addItemListener(new ItemListener() {
                        public void itemStateChanged(ItemEvent e) {
                              gui.scanAsMove =
scanAsMoveBox.getModel().isSelected();
                        }
                  });
            }
            return scanAsMoveBox;
      }


      private JComboBox getJCursorType() {
            if (jCursorType == null) {
                  jCursorType = new JComboBox(cursors);
                  jCursorType.setSelectedIndex(gui.cursorSelected);
                  jCursorType.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              gui.cursorSelected =
jCursorType.getSelectedIndex();
                              gui.updateImage();
                        }
                  });
            }
```

```
            return jCursorType;
        }
}
```

Location/src/location/gui/GUI_Wizard.java

```
package location.gui;

import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

import location.storage.Storage;

public class GUI_Wizard {

	private GUI_Main gui = null;

	public GUI_Wizard(GUI_Main gui) {
		this.gui = gui;
		boolean defaultProject = showQuestion();

		if (defaultProject) {
			// Default project clicked

			if (!new Storage(gui).read()) {
				// Can't read default project
				JOptionPane.showMessageDialog(null,
						"No se pudo leer el proyecto
predeterminado.\n" +
						GUI_Utils.encodeString("Se creará un
proyecto nuevo."),
						"Error!",
						JOptionPane.ERROR_MESSAGE);

				// Try to create a new map
				if (!gui.addMap()) {
					// Cancel clicked on select map image
					gui.wizardRunning = false;
					gui.setVisible(true);
				}
			}
			else {
				// Default project read correctly
				gui.wizardRunning = false;
			}
		}
		else {
			// New project clicked

			if (!gui.addMap())  {
				// Cancel clicked on select map image
				gui.wizardRunning = false;
				gui.setVisible(true);
			}
		}
	}
```

```
	private boolean showQuestion() {
		Object[] options = { "Abrir fichero predeterminado", "Crear
nuevo proyecto" };
		Icon icon = new
ImageIcon(GUI_Utils.getImageFileURL(gui.imagesDir + "scan.png"));

		int n = JOptionPane.showOptionDialog(
				null,
				"Bienvenido a " + gui.appName,
				"Bienvenido a " + gui.appName,
				JOptionPane.YES_NO_OPTION,
				JOptionPane.QUESTION_MESSAGE,
				icon,
				options,
				options[0]);

		if (n == JOptionPane.NO_OPTION)
			return false;		// No clicked
		else if (n != JOptionPane.YES_OPTION)
			return false;		// Window closed
		else return true;	// OK clicked
	}

}
```

Location/src/location/gui/GUI_About.java

```
package location.gui;

import java.awt.Component;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Point;

import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.SwingConstants;


public class GUI_About extends JDialog {

	private static final long serialVersionUID = 2957074847182708645L;

	private GUI_Main gui;

	static final int WIDTH = 300;
	static final int HEIGHT = 200;

	// About dialog
	private JPanel aboutContentPane = null;
	private JLabel infoLabel = null;
	private JLabel appLabel = null;
	private JLabel versionLabel = null;

	public GUI_About(GUI_Main gui) {
		super(gui, true);
		this.gui = gui;
		setTitle("Sobre " + gui.appName);
		setContentPane(getAboutContentPane());
		setResizable(false);
		pack();
		Point loc = gui.getLocation();
		loc.translate(gui.getWidth()/2 - WIDTH/2, gui.getHeight()/2 -
HEIGHT/2);
		setLocation(loc);
		setVisible(true);
	}

	private JPanel getAboutContentPane() {
		if (aboutContentPane == null) {
			aboutContentPane = new JPanel();
```

```
aboutContentPane.setBorder(BorderFactory.createEmptyBorder(20,50,20,50));
                  aboutContentPane.setLayout(new BoxLayout(aboutContentPane,
BoxLayout.PAGE_AXIS));

                  aboutContentPane.add(getAppLabel());
                  aboutContentPane.add(getVersionLabel());
                  aboutContentPane.add(Box.createRigidArea(new Dimension(0,
20)));
                  aboutContentPane.add(getInfoLabel());
            }
            return aboutContentPane;
      }

      private JLabel getAppLabel() {
            if (appLabel == null) {
                  Icon icon = new
ImageIcon(GUI_Utils.getImageFileURL(gui.imagesDir + "scan.png"));
                  appLabel = new JLabel(gui.appName, icon,
SwingConstants.CENTER);
                  appLabel.setFont(new Font("Sans Serif", Font.BOLD, 20));
                  appLabel.setHorizontalAlignment(SwingConstants.CENTER);
                  appLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
            }
            return appLabel;
      }

      private JLabel getVersionLabel() {
            if (versionLabel == null) {
                  versionLabel = new JLabel();
                  versionLabel.setText(GUI_Utils.encodeString("Versión ") +
gui.appVersion);

      versionLabel.setHorizontalAlignment(SwingConstants.CENTER);
                  versionLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
            }
            return versionLabel;
      }

      private JLabel getInfoLabel() {
            if (infoLabel == null) {
                  infoLabel = new JLabel();
                  String text = "<html>" +
GUI_Utils.encodeString(gui.appDescription) + "<br><br>" +
                                          "Escrito por:<br>\n" +
                                          "  " +
GUI_Utils.encodeString("Adolfo González Blázquez") + "<br>" +
                                          "  " +
GUI_Utils.encodeString("Pablo Mulas Gómez") + "<br>" +
                                          "  " +
GUI_Utils.encodeString("Rafael Rivera Retamar") + "<br>" +
                                          "</html>";
                  infoLabel.setText(text);
                  infoLabel.setHorizontalAlignment(SwingConstants.CENTER);
                  infoLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
```

```
            }
            return infoLabel;
        }
}
```

Location/src/location/gui/GUI_Utils.java

```
package location.gui;

import java.io.UnsupportedEncodingException;
import java.net.URL;
import java.nio.charset.Charset;

public class GUI_Utils {

	public static String encodeString(String string) {
		try {
			return new
String(string.getBytes(Charset.defaultCharset().name()), "UTF-8");
		} catch (UnsupportedEncodingException e) {
			e.printStackTrace();
			return null;
		}
	}

	public static URL getImageFileURL(String filename) {
		URL url = GUI_Utils.class.getResource(filename);
		return url;
	}
}
```

Location/src/location/gui/GUI_Main.java

```
package location.gui;

import java.awt.AlphaComposite;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.RenderingHints;
import java.awt.event.ActionEvent;
import java.awt.event.ComponentEvent;
import java.awt.event.ComponentListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.awt.geom.Rectangle2D;
import java.awt.image.BufferedImage;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Set;
import java.util.TimeZone;
import java.util.Vector;

import javax.swing.AbstractAction;
import javax.swing.Action;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenuBar;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JSplitPane;
import javax.swing.JTable;
import javax.swing.Timer;
import javax.swing.table.DefaultTableModel;

import location.storage.ImageFilter;
import location.storage.Storage;
import location.storage.TextFilter;
import location.storage.ZipFilter;
import location.types.AP;
import location.types.Floor;
import location.types.SignalVector;
```

```
import location.types.UneditableTableModel;
import location.utils.Grid;
import location.utils.Neighbors;
import location.utils.Voronoi;
import wifi.scanner.Scanner;

/**
 * @author Adolfo González Blázquez <code@infinicode.org>
 *
 */
public class GUI_Main extends JFrame implements ComponentListener {

      private static final long serialVersionUID = 2822648979256595838L;

      // Constants
    static final int MIN_WIDTH = 800;
    static final int MIN_HEIGHT = 600;
    protected final int scale = 10;
    protected final String appName = "Indoor Location";
    protected String appVersion = "1.0";
    protected String appDescription = "Geolocalización y Fingerprinting";
    protected final String imagesDir = "/location/images/";


      // Panels
      private JPanel mainContentPane = null;                // Main panel
      private JPanel rightPane = null;                      // Right panel
      private JScrollPane imageScrollPane = null;           // Scroll for
the image
      private JPanel mapImagePane = null;                   // Image
      private JScrollPane rightTableScroll = null;    // Scroll for the
table
      private JSplitPane jSplitPane = null;                 // Image + Table
      private JLabel statusBar = null;                      // Satus bar for
information


      // Map image
      private JLabel map = null;
                  // Label where the image is embeded
      protected ImageIcon image;
                  // The image itself
      static final String home = System.getProperty("user.home");       //
User home dir
      protected String outputFileName = home + "/DatosProyecto.zip";
            // Default output filename
      protected String rawDataFileName = null;


      // Tables
      private JTable rightTable = null;
      private UneditableTableModel scanTableModel = null;
      protected UneditableTableModel apsTableModel = null;
```

```
    // Other vars
      protected int x = 0;
      protected int y = 0;
    protected int step = 120;
    protected int scanTimes = 1;                // How many consecutives
times we want to scan
    protected int scanCount = 0;                // How many times we've
scanned the network
    protected int scanSleep = 3;                // Seconds to sleep before
next scan
    protected int locateTime = 2;
    protected boolean locateAuto = false;
    protected boolean locateFilterEnabled = false;
    protected String locateFilterString = "";
    protected boolean locateVoronoi = false;
    protected boolean scanAsMove = false; // Scan when moving
    protected int cursorSelected = 0;                               // The
selected cursor { "Circulo", "Cruz", "Equis" };
      protected boolean showCoordsOnMap = false;
      protected boolean showAps = true;
      protected boolean showPoints = true;
      protected boolean showPaths = false;

      protected int viewMode = 0;         // 0: Location  1: Devices  2:
Scan

      protected Timer timer = null;
      protected boolean wizardRunning = false;

      protected boolean addingAp = false;
      protected HashMap<String, HashMap<Point, SignalVector>> data = null;
      protected Vector<Point> scanPoints = null;
      protected Vector<AP> scanAps = null;
      protected HashMap<Point, Vector<Point>> scanPaths = null;

      protected int floorCount = 0;
      protected HashMap<Integer, Floor> floors = null;
      protected Floor activeFloor = null;

      // The control buttons
    private GUI_ControlsScan controls = null;
    private GUI_ControlsDevices devices = null;
    private GUI_ControlsLocation locationcontrols = null;
    private GUI_Menus menus = null;
    protected GUI_AddAP addAPDialog;


    /*
     * The main constructor
     */
      public GUI_Main() {



            // Main data structure
```

```
            floors = new HashMap<Integer, Floor>();

            wizardRunning = true;
            new GUI_Wizard(this);


            // Create the GUI
            controls = new GUI_ControlsScan(this);
            devices = new GUI_ControlsDevices(this);
            locationcontrols = new GUI_ControlsLocation(this);
            menus = new GUI_Menus(this);

            setTitle(appName);
            if (!wizardRunning) setVisible(true);
            addComponentListener(this);
            setMinimumSize(new Dimension(MIN_WIDTH,MIN_HEIGHT));
            setSize(MIN_WIDTH, MIN_HEIGHT);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setExtendedState(getExtendedState() | MAXIMIZED_BOTH);

            setJMenuBar(menus.getJJMenuBar());
            setContentPane(getMainContentPane());


            // Filename for raw data
            Calendar cal = new GregorianCalendar();
            rawDataFileName = home + "/" + cal.get(Calendar.DAY_OF_MONTH) +
"_" +
                                    cal.get(Calendar.MONTH) + "_" +
                                    cal.get(Calendar.YEAR) + "_";

            // Set timer for auto location every second
            Action autoLocateAction = new AbstractAction() {
                  private static final long serialVersionUID =
-7681683898617701 94L;
                  public void actionPerformed(ActionEvent e) {
                  locateAuto();
                }
            };
            timer = new Timer(locateTime * 1000, autoLocateAction);
            timer.start();


            // Update GUI
            updateImage();
            updateMenus();
            updateStatusBar();
            updateApsTable();
      }



/*************************************************************************
**
```

```
	 * GUI_Main and panels stuff

**************************************************************************
*/

	// The main panel
	private JPanel getMainContentPane() {
		if (mainContentPane == null) {
			mainContentPane = new JPanel();
			mainContentPane.setLayout(new BorderLayout());
			mainContentPane.add(getMainSplitPane(),
BorderLayout.CENTER);
			mainContentPane.add(getStatusBar(), BorderLayout.SOUTH);
		}
		return mainContentPane;
	}

	// The screen is divided in two
	private JSplitPane getMainSplitPane() {
		if (jSplitPane == null) {
			jSplitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,

	getImageScrollPane(),
								getRightPane());
			jSplitPane.setDividerLocation(this.getWidth()/2);
		}
		return jSplitPane;
	}

	// ScrollPane for the map image
	private JScrollPane getImageScrollPane() {
		if (imageScrollPane == null) {
			imageScrollPane = new JScrollPane();
			imageScrollPane.getViewport().add(getMapImagePane());
		}
		return imageScrollPane;

	}

	// Image panel inside ImageScrollPane
	private JPanel getMapImagePane() {
		if (mapImagePane == null) {
			mapImagePane = new JPanel();

			image = modifyImage();

			map = new JLabel();
			map.setIcon(image);
			mapImagePane.add(map);
			map.addMouseListener(new MouseListener() {

				public void mouseClicked(MouseEvent e) {
					if (viewMode == 0) return;
					x = e.getX();
					y = image.getIconHeight() - e.getY();
```

```
                              if (addingAp)  {

      addAPDialog.xField.setText(String.valueOf(x));

      addAPDialog.yField.setText(String.valueOf(y));
                              }
                              updateImage();
                              updateStatusBar();
                        }
                        public void mouseEntered(MouseEvent e) {  }
                        public void mouseExited(MouseEvent e) {}
                        public void mousePressed(MouseEvent e) {}
                        public void mouseReleased(MouseEvent e) {}
                  });
                  map.addMouseMotionListener(new MouseMotionListener() {
                        public void mouseDragged(MouseEvent e) {
                              if (viewMode == 0) return;
                              x = e.getX();
                              y = image.getIconHeight() - e.getY();
                              if (addingAp)  {

      addAPDialog.xField.setText(String.valueOf(x));

      addAPDialog.yField.setText(String.valueOf(y));
                              }
                              updateImage();
                              updateStatusBar();
                        }
                        public void mouseMoved(MouseEvent e) {}
                  });
            }
            return mapImagePane;
      }

      // The right pane
      private JPanel getRightPane() {
            if (rightPane == null) {
                  rightPane = new JPanel();
                  rightPane.setLayout(new BorderLayout());
                  rightPane.add(getRightTableScroll(), BorderLayout.CENTER);
                  rightPane.add(locationcontrols.getJButtonPane(),
BorderLayout.SOUTH);
            }
            return rightPane;
      }

      // ScrollPan for the Aps table
      private JScrollPane getRightTableScroll() {
            if (rightTableScroll == null) {
                  rightTableScroll = new JScrollPane();
                  rightTableScroll.getViewport().add(getRightTable());
            }
            return rightTableScroll;
      }
```

```
	// The right side table
	private JTable getRightTable() {
		if (rightTable == null) {
			getScanTableModel();
			getApsTableModel();
			rightTable = new JTable(scanTableModel);
		}
		return rightTable;
	}

	// The Scan table
	private DefaultTableModel getScanTableModel() {
		if (scanTableModel == null) {
			scanTableModel = new UneditableTableModel();
			scanTableModel.addColumn("X");
			scanTableModel.addColumn("Y");
			scanTableModel.addColumn("MAC");
			scanTableModel.addColumn("ESSID");
			scanTableModel.addColumn("FUERZA");
		}
		return scanTableModel;
	}

	// The APs table
	public DefaultTableModel getApsTableModel() {
		if (apsTableModel == null) {
			apsTableModel = new UneditableTableModel();
			apsTableModel.addColumn("X");
			apsTableModel.addColumn("Y");
			apsTableModel.addColumn("MAC");
			apsTableModel.addColumn("ESSID");
			apsTableModel.addColumn("Planta");
		}
		return apsTableModel;
	}

	// Status bar
	private JLabel getStatusBar() {
		if (statusBar == null) {
			statusBar = new JLabel();
			statusBar.setPreferredSize(new Dimension(100, 16));
			updateStatusBar();
		}
		return statusBar;
	}



/**************************************************************************
**
	 * Application methods

**************************************************************************
*/
```

```
	protected void moveCursorUp() {
		y += step / scale;
		updateImage();
		if (scanAsMove) runScan();
		updateStatusBar();
	}

	protected void moveCursorDown() {
		y -= step / scale;
		updateImage();
		if (scanAsMove) runScan();
		updateStatusBar();
	}

	protected void moveCursorLeft() {
		x -= step / scale;
		updateImage();
		if (scanAsMove) runScan();
		updateStatusBar();
	}

	protected void moveCursorRight() {
		x += step / scale;
		updateImage();
		if (scanAsMove) runScan();
		updateStatusBar();
	}

	protected void runScan() {

		HashMap apList;

		int count = 4;
		String aps = "";


		for (int j = 0; j < scanTimes; j++) {

			scanCount++;

			// Scan the network n times...
			for (int i = 0; i < count; i++) {
				Scanner scan = new Scanner();
				apList = scan.getAps();
				Set apKeys = apList.keySet();
				Iterator It = apKeys.iterator();
				aps = scan.getApsAsString();
				try {
				  BufferedWriter out = new BufferedWriter(
							new FileWriter(rawDataFileName +
activeFloor.getName().replaceAll(" ", "_") + ".txt", true));
					out.write(x + "\t" + y);
					out.write("\n");
					    while (It.hasNext()) {
						String mac = (String)(It.next());
```

```
                                    String[] info = (String[])
apList.get(mac);
                                    String essid = info[0];
                                    double signal =
Integer.parseInt(info[1].split("/")[0]);

                                    out.write(mac + "\t" + essid + "\t" +
info[1] + "\n");


                                    // Add data to models
                                    Point p = new Point(x, y);
                                    if (!scanPoints.contains(p))
scanPoints.add(p);
                                    Grid grid = new Grid(scanPoints);
                                    scanPaths.put(p, grid.getNeighbors(x,
y));


                                    if (data.containsKey(mac)) {
                                          HashMap<Point, SignalVector> v =
(HashMap<Point, SignalVector>)data.get(mac);
                                          if ( v.containsKey(p) ) {
//                                              signal = (signal + v.get(p))
/ 2;
//
                                              v.put(p, signal);
                                              v.get(p).add(signal);
                                          }
                                          else v.put(p, new
SignalVector(signal));

                                    } else {

                                          HashMap<Point, SignalVector> v =
new HashMap<Point, SignalVector>();
                                          v.put(p, new
SignalVector(signal));
                                          data.put(mac, v);
                                          AP ap = new AP(mac, essid, -1, -1,
getActiveFloor().getIdFloor());
                                          scanAps.add(ap);
                                    }
                              }
                            out.write("\n");
                            out.close();
                        } catch (IOException e) {
                          e.printStackTrace();
                        }
                  }

                  if (scanTimes > 1 && j < scanTimes-1) {
                        try {
                              Thread.sleep(scanSleep * 1000);
                        } catch (InterruptedException e) {
                              e.printStackTrace();
                        }
                  }
```

```
        }

      // Clean the table
      while (scanTableModel.getRowCount() > 0)
        scanTableModel.removeRow(0);

        // Show last results
      String[] res = aps.split("\n");
      for (int j = 0; j < res.length; j++) {
        Object[] row = new Object[5];
        row[0] = x;
        row[1] = y;
        String[] info = res[j].split("\t");
        row[2] = info[0];
        row[3] = info[1];
        row[4] = info[2];
        scanTableModel.addRow(row);
      }

      updateApsTable();
      updateImage();
        updateStatusBar();
    }

    protected void addAP(String mac, String essid, int x, int y, int
floor) {
    Object[] row = new Object[5];
    row[0] = x;
    row[1] = y;
    row[2] = mac;
    row[3] = essid;
    row[4] = floor;
    apsTableModel.addRow(row);
    AP ap = new AP(mac, essid, x, y, floor);
    scanAps.add(ap);
    }

    protected void removeAP() {
          int row = rightTable.getSelectedRow();
          scanAps.remove(row);
          updateApsTable();
          updateImage();
    }

    protected void modifyAPGet() {
          int row = rightTable.getSelectedRow();
          if (row > -1) {
                int x = (Integer)apsTableModel.getValueAt(row, 0);
                int y = (Integer)apsTableModel.getValueAt(row, 1);
                String mac = (String) apsTableModel.getValueAt(row, 2);
                String essid = (String) apsTableModel.getValueAt(row, 3);
                int floor = (Integer)apsTableModel.getValueAt(row, 4);
                new GUI_AddAP(this, mac, essid, x, y, floor);
          }
    }
```

```
	protected void modifyAPSet(String mac, String essid, int x, int y, int
floor) {
		Iterator it = scanAps.iterator();
		AP ap = null;
		while (it.hasNext()) {
			ap = (AP)it.next();
			if (ap.getMac().equals(mac)) break;
		}

		if (ap == null) return;

		ap.setMac(mac);
		ap.setEssid(essid);
		ap.setX(x);
		ap.setY(y);
		ap.setFloor(floor);
		updateApsTable();
		updateImage();
	}


	private void printAps(Graphics2D g2d) {

		if (scanAps == null) return;
		AlphaComposite alpha =
AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 1f);
		g2d.setComposite(alpha);
		ImageIcon ap = new ImageIcon(GUI_Utils.getImageFileURL(imagesDir
+ "antenna.png"));

		for (int i = 0; i < scanAps.size(); i++) {
			int x = scanAps.get(i).getX() - (ap.getIconWidth()/2);
			int y = image.getIconHeight() - scanAps.get(i).getY() -
(ap.getIconHeight()/2);
			if (x > 0 && x < image.getIconWidth() && y > 0 && y <
image.getIconHeight())
				g2d.drawImage(ap.getImage(), x, y, null);
		}

		alpha = AlphaComposite.getInstance(AlphaComposite.SRC_OVER,
0.5f);
		g2d.setComposite(alpha);
	}

	private void printPaths(Graphics2D g2d) {

		if (scanPaths.size() == 0) {
			scanPaths = new Grid(scanPoints).getPaths();
			if (scanPaths.size() == 0) return;
		}

		g2d.setColor(Color.blue);

		for (int i = 0; i < scanPoints.size(); i++) {
```

```
                  Point p = scanPoints.elementAt(i);
                  Vector<Point> ps = scanPaths.get(p);

                  for (int j = 0; j < ps.size(); j++) {
                        Point pp = ps.elementAt(j);
                        g2d.drawLine((int)p.getX(), image.getIconHeight() -
(int)p.getY(), (int)pp.getX(), image.getIconHeight() - (int)pp.getY());
                  }
            }

            g2d.setColor(Color.red);
      }

      private void printPoints(Graphics2D g2d) {

            if (scanPoints == null) return;

            g2d.setColor(Color.blue);

            for (int i = 0; i < scanPoints.size(); i++) {
                  Point p = (Point)scanPoints.get(i);
                  int x = (int) p.getX();
                  int y = image.getIconHeight() - (int) p.getY();
                  g2d.fillOval(x-2, y-2, 4, 4);
            }

            g2d.setColor(Color.red);
      }

      private ImageIcon modifyImage() {

            // Create image
            if (activeFloor == null) return null;
            else image = new
ImageIcon(activeFloor.getMapImage().getImage());

            BufferedImage bufferedImage = new
BufferedImage(image.getIconWidth(),

            image.getIconHeight(),

            BufferedImage.TYPE_INT_RGB);
            Graphics2D g2d = (Graphics2D) bufferedImage.getGraphics();
            g2d.drawImage(image.getImage(), 0, 0, null);

            //Create an alpha composite of 50%
            AlphaComposite alpha =
AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 0.5f);
            g2d.setComposite(alpha);
            g2d.setColor(Color.red);

            // Display APs, scanPoints and Paths (if needed)
            switch (viewMode) {
                  case 0:           // Location
                        if (showAps) printAps(g2d);
```

```
                        if (showPoints) printPoints(g2d);
                        if (showPaths) printPaths(g2d);
                        break;
                  case 1:           // Devices
                        printAps(g2d);
                        if (showPoints) printPoints(g2d);
                        if (showPaths) printPaths(g2d);
                        break;
                  case 2:           // Scan
                        if (showAps) printAps(g2d);
                        printPoints(g2d);
                        if (showPaths) printPaths(g2d);
                        break;
            }


            // Show coords on map
            if (showCoordsOnMap) {
                  g2d.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,

RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
                  g2d.setFont(new Font("Arial", Font.BOLD, 30));
                  String watermark = "X: " + x + " / Y: " + y;
                  FontMetrics fontMetrics = g2d.getFontMetrics();
                  Rectangle2D rect = fontMetrics.getStringBounds(watermark,
g2d);
                  g2d.drawString(watermark, (image.getIconWidth() - (int)
rect.getWidth()) / 2,

      (image.getIconHeight() - (int) rect.getHeight()) / 2);
            }

            // Draw cursor
            if (x != 0 && y != 0) {
                  if (viewMode == 0) g2d.setColor(Color.blue);
                  int ny = image.getIconHeight() - y;
                  switch (cursorSelected) {
                        case 0:     // Circulo
                              g2d.fillOval(x-5, ny - 5, 10, 10);
                              break;
                        case 1: // Cruz
                              g2d.drawLine(x-5, ny, x+5, ny);
                              g2d.drawLine(x, ny-5, x, ny+5);
                              break;
                        case 2: // Equis
                              g2d.drawLine(x-5, ny-5, x+5, ny+5);
                              g2d.drawLine(x-5, ny+5, x+5, ny-5);
                              break;
                        default:
                              g2d.fillOval(x-5, ny - 5, 10, 10);
                              break;
                  }
            }

            //Free graphic resources
```

```
            g2d.dispose();

            image.setImage(bufferedImage);

            return image;
      }

      protected boolean addMap() {

            JFileChooser openDialog = new JFileChooser();
            openDialog.addChoosableFileFilter(new ImageFilter());
            openDialog.setAcceptAllFileFilterUsed(false);
            openDialog.setDialogTitle("Seleccione fichero de mapa...");
        if (openDialog.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
{
            new GUI_AddMap(this,
openDialog.getSelectedFile().getAbsolutePath());
            return true;
        }
        else {
            return false;
        }
      }

      protected void newMap(String name, String filename, int idFloor) {
            Floor floor = new Floor(name, filename, idFloor);
            floors.put(idFloor, floor);
            floorCount++;
            if (activeFloor == null) setActiveFloor(floor);
            updateMenus();
      }

      protected void setActiveFloor(int idFloor) {
            activeFloor = floors.get(idFloor);
            data = activeFloor.getData();
            scanPoints = activeFloor.getScanPoints();
            scanAps = activeFloor.getScanAps();
            scanPaths = activeFloor.getScanPaths();
            scanCount = activeFloor.getScanCount();
      }

      protected void changeActiveMap(int idFloor) {
            setActiveFloor(idFloor);
            updateApsTable();
            updateImage();
            updateStatusBar();
      }

      protected boolean openData() {

            JFileChooser openDialog = new JFileChooser();
            openDialog.addChoosableFileFilter(new ZipFilter());
            openDialog.setAcceptAllFileFilterUsed(false);
            openDialog.setDialogTitle("Abrir fichero de datos...");
        if (openDialog.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
```

```
{
            String dataPath =
openDialog.getSelectedFile().getAbsolutePath();
            resetFloor();

            Storage storage = new Storage(this, dataPath);
            if (storage.read()) {
            Grid grid = new Grid(scanPoints);
                scanPaths = grid.getPaths();

                  updateImage();
                  updateMenus();
                  updateApsTable();
                  updateStatusBar();

                return true;
            }
            else {
            setStatusBar("Error abriendo datos");
            return false;
            }
        }
        else {
            return false;
        }
      }

      @SuppressWarnings("unchecked")
      protected boolean importRawData() {
            JFileChooser openDialog = new JFileChooser();
            openDialog.addChoosableFileFilter(new TextFilter());
            openDialog.setAcceptAllFileFilterUsed(false);
            openDialog.setDialogTitle("Abrir fichero de datos...");
        if (openDialog.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
{
            String dataPath =
openDialog.getSelectedFile().getAbsolutePath();
            resetFloor();

            Storage storage = new Storage(this, dataPath);

            if (storage.importRawData()) {
            Grid grid = new Grid(scanPoints);
                scanPaths = grid.getPaths();

                updateApsTable();
                updateImage();
                updateStatusBar();
                return true;

            } else {
            setStatusBar("Error importando datos en bruto");
            return false;
            }
        }
```

```
        else {
            return false;
        }
      }

      protected boolean setOutputFileName() {
            JFileChooser saveDialog = new JFileChooser();
        saveDialog.setDialogTitle("Seleccionar fichero de salida...");
            if (saveDialog.showSaveDialog(this) ==
JFileChooser.APPROVE_OPTION) {
                  outputFileName =
saveDialog.getSelectedFile().getAbsolutePath();
            return true;
        }
            else {
                  return false;
            }
      }

      protected boolean saveData() {
            JFileChooser saveDialog = new JFileChooser();
            saveDialog.addChoosableFileFilter(new ZipFilter());
            saveDialog.setAcceptAllFileFilterUsed(false);
        saveDialog.setDialogTitle("Seleccionar fichero de salida...");
            if (saveDialog.showSaveDialog(this) ==
JFileChooser.APPROVE_OPTION) {
                  outputFileName =
saveDialog.getSelectedFile().getAbsolutePath();
                  Storage storage = new Storage(this, outputFileName);

                  boolean r = storage.save();
                  if (r) setStatusBar("Datos guardados con exito");
                  else setStatusBar("Datos no guardados!");

            return true;
        }
            else {
                  return false;
            }
      }

      protected void resetFloor() {
            x = 0;
            y = 0;
            scanCount = 0;

          // Clean the table
          while (scanTableModel.getRowCount() > 0)
            scanTableModel.removeRow(0);

          String name = activeFloor.getName();
          String mapFileName = activeFloor.getMapFileName();
            ImageIcon mapImage = activeFloor.getMapImage();
            int idFloor = activeFloor.getIdFloor();
```

```
            floors.remove(idFloor);
            activeFloor = new Floor(name, mapFileName, mapImage, idFloor);
            floors.put(idFloor, activeFloor);
            setActiveFloor(activeFloor);

        updateImage();
            updateStatusBar();
    }

    private void updateApsTable() {
        if (scanAps == null) return;

            // Clean the table
        while (apsTableModel.getRowCount() > 0)
            apsTableModel.removeRow(0);

      for (int i = 0; i < scanAps.size(); i++) {
            AP ap = scanAps.elementAt(i);
            Object[] row = new Object[5];
            row[0] = ap.getX();
            row[1] = ap.getY();
            row[2] = ap.getMac();
            row[3] = ap.getEssid();
            row[4] = ap.getFloor();
            apsTableModel.addRow(row);
      }
    }

    protected void updateImage() {
        // Reload the image
            image = modifyImage();
            map.setIcon(image);
            mapImagePane.repaint();
    }

    private void updateStatusBar() {
            if (activeFloor == null) return;
            statusBar.setText("     X: " + x + "   Y: " + y +
                              "     Planta: " +
activeFloor.getIdFloor() +
                              "     Puntos escaneados: " +
scanPoints.size() +
                              "     Escaneos: " + scanCount +
                              "     APs reconocidos: " +
scanAps.size()
                              );
    }

    private void updateStatusBar(String text) {
            if (activeFloor == null) return;
            statusBar.setText("     X: " + x + "   Y: " + y +
                              "     Planta: " +
activeFloor.getIdFloor() +
                              "     Puntos escaneados: " +
scanPoints.size() +
```

```
                                        "     Escaneos: " + scanCount +
                                        "     APs reconocidos: " +
scanAps.size() +
                                        "     " + text
                                        );
      }

      private void updateMenus() {
            menus.populateFloorsMenuItem();
      }

      protected void setStatusBar(String text) {
            statusBar.setText(text);
      }

      protected void locationMode() {
            viewMode = 0;
            x = 0;
            y = 0;
            rightTable.setModel(getScanTableModel());
            rightPane.remove(1);
            rightPane.add(locationcontrols.getJButtonPane(),
BorderLayout.SOUTH);
            rightPane.validate();
            rightPane.repaint();
            toggleActionsMenu(false);
            updateImage();
            updateStatusBar();
      }

      protected void devicesMode() {
            viewMode = 1;
            rightTable.setModel(getApsTableModel());
            rightPane.remove(1);
            rightPane.add(devices.getJButtonPane(), BorderLayout.SOUTH);
            rightPane.validate();
            rightPane.repaint();
            toggleActionsMenu(false);
            updateImage();
            updateStatusBar();
      }

      protected void scanMode() {
            viewMode = 2;
            rightTable.setModel(getScanTableModel());
            rightPane.remove(1);
            rightPane.add(controls.getJButtonPane(), BorderLayout.SOUTH);
            rightPane.repaint();
            rightPane.validate();
            toggleActionsMenu(true);
            updateImage();
            updateStatusBar();
      }

      private void toggleActionsMenu(boolean state) {
```

```
            JMenuBar menu = menus.getJJMenuBar();
            menu.getComponent(3).setEnabled(state);
            menu.getComponent(3).setVisible(state);
      }

      protected void locateNow() {

            if (viewMode != 0) return;

            Neighbors n = new Neighbors(this);
            Scanner scanner = new Scanner();
            Point p;

            java.util.Date inicio = new java.util.Date();

            // ESSID Filter
            if (locateFilterEnabled && !locateFilterString.equals("")) {
                  System.out.println("Filtrando: " + locateFilterString);
                  p = n.getLocation(scanner.getAps(), locateFilterString);
            }
            else
                  p = n.getLocation(scanner.getAps());

            // Voronoi Filter
            if (locateVoronoi) {
                  System.out.println("Posicion pre-Voronoi: " +
(int)p.getX() + " / " + (int)p.getY());
                  Voronoi voronoi = new Voronoi(p);
                  p = voronoi.getVoronoi();
            }

            // Print information on console
            java.util.Date fin = new java.util.Date();
            long ttotal = fin.getTime() - inicio.getTime();
            System.out.println("Posicion: " + (int)p.getX() + " / " +
(int)p.getY());
            System.out.println("Tiempo total: " + ttotal/1000f + "
segundos");
            System.out.println();

            // Move the point
            x = (int) p.getX();
            y = (int) p.getY();

          // Clean the table
          while (scanTableModel.getRowCount() > 0)
            scanTableModel.removeRow(0);

            // Show last results
          String[] res = scanner.getApsAsString().split("\n");
          for (int j = 0; j < res.length; j++) {
            Object[] row = new Object[5];
            row[0] = x;
            row[1] = y;
            String[] info = res[j].split("\t");
```

```
            row[2] = info[0];
            row[3] = info[1];
            row[4] = info[2];
            scanTableModel.addRow(row);
          }

          updateImage();
          updateStatusBar(GUI_Utils.encodeString("Hora de la última
localización: ") + getTime() +
                                "     Tiempo empleado: " + ttotal/1000f + "
segundos");
      }

      private void locateAuto() {
            if (viewMode == 0 && locateAuto) {
                  locateNow();
            }
      }

      private String getTime() {
            Calendar cal = Calendar.getInstance(TimeZone.getDefault());
            //String DATE_FORMAT = "yyyy-MM-dd HH:mm:ss";
            String DATE_FORMAT = "HH:mm:ss";
          java.text.SimpleDateFormat sdf = new
java.text.SimpleDateFormat(DATE_FORMAT);
          sdf.setTimeZone(TimeZone.getDefault());
          return  sdf.format(cal.getTime());
      }



/**************************************************************************
**
       * ComponentListener methods

**************************************************************************
*/
      public void componentResized(ComponentEvent e) {

            // Avoid window too small
            int width = getWidth();
        int height = getHeight();
        boolean resize = false;

        if (width < MIN_WIDTH) {
            resize = true;
            width = MIN_WIDTH;
        }
            if (height < MIN_HEIGHT) {
                  resize = true;
                  height = MIN_HEIGHT;
            }
            if (resize) {
                  setSize(width, height);
            }
```

```
	}
	public void componentHidden(ComponentEvent e) {}
	public void componentMoved(ComponentEvent e) {}
	public void componentShown(ComponentEvent e) {}




/**************************************************************************
**
	 * Accessors and mutators

**************************************************************************
*/
	public HashMap<String, HashMap<Point, SignalVector>> getData() {
		return data;
	}

	public void setData(HashMap<String, HashMap<Point, SignalVector>>
data) {
		this.data = data;
		this.activeFloor.setData(data);
	}

	public Vector<AP> getScanAps() {
		return scanAps;
	}

	public void setScanAps(Vector<AP> scanAps) {
		this.scanAps = scanAps;
		this.activeFloor.setScanAps(scanAps);
	}

	public int getScanCount() {
		return scanCount;
	}

	public void setScanCount(int scanCount) {
		this.scanCount = scanCount;
		this.activeFloor.setScanCount(scanCount);
	}

	public HashMap<Point, Vector<Point>> getScanPaths() {
		return scanPaths;
	}

	public void setScanPaths(HashMap<Point, Vector<Point>> scanPaths) {
		this.scanPaths = scanPaths;
		this.activeFloor.setScanPaths(scanPaths);
	}

	public Vector<Point> getScanPoints() {
```

```
		return scanPoints;
	}

	public void setScanPoints(Vector<Point> scanPoints) {
		this.scanPoints = scanPoints;
		this.activeFloor.setScanPoints(scanPoints);
	}

	public void setScanTableModel(UneditableTableModel scanTableModel) {
		this.scanTableModel = scanTableModel;
	}

	public void setApsTableModel(UneditableTableModel apsTableModel) {
		this.apsTableModel = apsTableModel;
	}

	public String getOutputFileName() {
		return outputFileName;
	}

	public void setOutputFileName(String outputFileName) {
		this.outputFileName = outputFileName;
	}

	public HashMap<Integer, Floor> getFloors() {
		return floors;
	}

	public void setFloors(HashMap<Integer, Floor> floors) {
		this.floors = floors;

		// Set first floor active
		activeFloor =
floors.get(this.floors.entrySet().iterator().next().getKey());
		data = activeFloor.getData();
		scanPoints = activeFloor.getScanPoints();
		scanAps = activeFloor.getScanAps();
		scanPaths = activeFloor.getScanPaths();
		scanCount = activeFloor.getScanCount();
	}



	public Floor getActiveFloor() {
		return activeFloor;
	}



	public void setActiveFloor(Floor activeFloor) {
		this.activeFloor = activeFloor;
		data = activeFloor.getData();
		scanPoints = activeFloor.getScanPoints();
		scanAps = activeFloor.getScanAps();
		scanPaths = activeFloor.getScanPaths();
```

```
            scanCount = activeFloor.getScanCount();
        }
    }
```

Location/src/location/gui/GUI_ControlsLocation.java

```
package location.gui;

import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.KeyEvent;

import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSpinner;
import javax.swing.JTextField;
import javax.swing.SpinnerNumberModel;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;

public class GUI_ControlsLocation {

	private GUI_Main gui;

	private JPanel jButtonPanel = null;

	static final int BUTTONSIZE = 32;

	// Buttons
	private JButton locateNowButton = null;
	private SpinnerNumberModel stepModel;
	private JSpinner secondsSpinner;
	private JCheckBox autoLocateBox = null;
	private JCheckBox filterApsBox = null;
	private JCheckBox voronoiBox = null;
	private JTextField filterApsField = null;

	// Constructor
	public GUI_ControlsLocation(GUI_Main gui) {
		this.gui = gui;
	}

	public JPanel getJButtonPane() {
		if (jButtonPanel == null) {
			jButtonPanel = new JPanel();
```

```
                jButtonPanel.setLayout(new GridBagLayout());

                GridBagConstraints c = new GridBagConstraints();
                c.insets = new Insets(5, 5, 5, 5);
                c.fill = GridBagConstraints.HORIZONTAL;

                c.gridwidth = 3;
                c.gridx = 0;
                c.gridy = 0;
                jButtonPanel.add(getLocateNowButton(), c);
                c.gridwidth = 1;

                c.gridx = 0;
                c.gridy = 1;
                jButtonPanel.add(getAutoLocateBox(), c);

                c.gridx = 1;
                c.gridy = 1;
                jButtonPanel.add(getSecondsSpinner(), c);

                c.gridx = 2;
                c.gridy = 1;
                jButtonPanel.add(new JLabel("segundos"), c);

                c.gridx = 0;
                c.gridy = 2;
                jButtonPanel.add(getFilterApsBox(), c);

                c.gridx = 1;
                c.gridy = 2;
                c.gridwidth = 2;
                jButtonPanel.add(getFilterApsField(), c);
                c.gridwidth = 1;

                c.gridx = 0;
                c.gridy = 3;
                c.gridwidth = 3;
                jButtonPanel.add(getVoronoiBox(), c);
                c.gridwidth = 1;

          }
          return jButtonPanel;
     }

     private JButton getLocateNowButton() {
          if (locateNowButton == null) {
                Icon icon = new
ImageIcon(GUI_Utils.getImageFileURL(gui.imagesDir + "location.png"));
                locateNowButton = new JButton("Localizar ahora", icon);
                locateNowButton.setPreferredSize(new Dimension(BUTTONSIZE,
BUTTONSIZE));
                locateNowButton.setMnemonic(KeyEvent.VK_UP);
                locateNowButton.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            gui.locateNow();
```

```
                        }
                  });
            }
            return locateNowButton;
      }

      private JCheckBox getAutoLocateBox() {
            if (autoLocateBox == null) {
                  autoLocateBox = new JCheckBox(GUI_Utils.encodeString("
Localizar automáticamente cada"), gui.locateAuto);
                  autoLocateBox.addItemListener(new ItemListener() {
                        public void itemStateChanged(ItemEvent e) {
                              gui.locateAuto =
autoLocateBox.getModel().isSelected();
                              locateNowButton.setEnabled(!gui.locateAuto);
                        }
                  });
            }
            return autoLocateBox;
      }

      private JSpinner getSecondsSpinner() {
            if (secondsSpinner == null) {
                  stepModel = new SpinnerNumberModel(gui.locateTime, 1,
1000, 1);
                  secondsSpinner = new JSpinner(stepModel);

                  secondsSpinner.addChangeListener(new ChangeListener() {
                        public void stateChanged(ChangeEvent e) {
                              gui.locateTime =
stepModel.getNumber().intValue();
                              gui.timer.setDelay(gui.locateTime * 1000);
                        }
                  }
                  );
            }
            return secondsSpinner;
      }

      private JCheckBox getFilterApsBox() {
            if (filterApsBox == null) {
                  filterApsBox = new JCheckBox(" Usar solo APs cuyo ESSID
contiene", gui.locateFilterEnabled);
                  filterApsBox.addItemListener(new ItemListener() {
                        public void itemStateChanged(ItemEvent e) {
                              gui.locateFilterEnabled =
filterApsBox.getModel().isSelected();
                        }
                  });
            }
            return filterApsBox;
      }

      private JTextField getFilterApsField() {
            if (filterApsField == null) {
```

```
                  filterApsField = new JTextField();
                  filterApsField.getDocument().addDocumentListener(new
DocumentListener() {
                        public void changedUpdate(DocumentEvent e) {
                              gui.locateFilterString =
filterApsField.getText();
                        }
                        public void insertUpdate(DocumentEvent e) {
                              gui.locateFilterString =
filterApsField.getText();
                        }
                        public void removeUpdate(DocumentEvent e) {
                              gui.locateFilterString =
filterApsField.getText();
                        }
                  });
            }
            return filterApsField;
      }

      private JCheckBox getVoronoiBox() {
            if (voronoiBox == null) {
                  voronoiBox = new JCheckBox(" Aplicar filtro de Voronoi al
resultado", gui.locateVoronoi);
                  voronoiBox.addItemListener(new ItemListener() {
                        public void itemStateChanged(ItemEvent e) {
                              gui.locateVoronoi =
voronoiBox.getModel().isSelected();
                        }
                  });
            }
            return voronoiBox;
      }
}
```

Location/src/location/gui/GUI_Menus.java

```
/**
 *
 */
package location.gui;

import java.awt.Event;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.util.Iterator;
import java.util.Map;

import javax.swing.ButtonGroup;
import javax.swing.JCheckBoxMenuItem;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JRadioButtonMenuItem;
import javax.swing.KeyStroke;

import location.types.Floor;


/**
 * @author fito
 *
 */
public class GUI_Menus {

	private GUI_Main gui;

	private JMenuBar jJMenuBar = null;

	private JMenu fileMenu = null;
	private JMenu viewMenu = null;
	private JMenu actionMenu = null;
	private JMenu floorsMenu = null;
	private JMenu helpMenu = null;

	private JMenuItem openDataMenuItem = null;
	private JMenuItem saveDataMenuItem = null;
	private JMenuItem importRawDataMenuItem = null;
	private JMenuItem setOutputFileNameMenuItem = null;
	private JMenuItem exitMenuItem = null;

	private JRadioButtonMenuItem viewScanMenuItem = null;
	private JRadioButtonMenuItem viewDevicesMenuItem = null;
	private JRadioButtonMenuItem viewLocationMenuItem = null;
	private JCheckBoxMenuItem viewCoordsMenuItem = null;
	private JCheckBoxMenuItem viewApsMenuItem = null;
	private JCheckBoxMenuItem viewPointsMenuItem = null;
```

```
	private JCheckBoxMenuItem viewPathsMenuItem = null;

	private JMenuItem upMenuItem = null;
	private JMenuItem downMenuItem = null;
	private JMenuItem leftMenuItem = null;
	private JMenuItem rightMenuItem = null;
	private JMenuItem scanMenuItem = null;
	private JMenuItem resetMenuItem = null;

	private JMenuItem addMapMenuItem = null;
	private ButtonGroup floorsGroup = new ButtonGroup();
	private JRadioButtonMenuItem rbMenuItem;

	private JMenuItem aboutMenuItem = null;

	/*
	 * The constructor
	 */
	public GUI_Menus(GUI_Main gui) {
		this.gui = gui;
	}

	/*
	 * Get the menu
	 */
	public JMenuBar getJJMenuBar() {
		if (jJMenuBar == null) {
			jJMenuBar = new JMenuBar();
			jJMenuBar.add(getFileMenu());
			jJMenuBar.add(getViewMenu());
			jJMenuBar.add(getFloorsMenu());
			jJMenuBar.add(getActionsMenu());
			jJMenuBar.add(getHelpMenu());
		}
		return jJMenuBar;
	}



/*************************************************************************
**
	 * File menu

*************************************************************************
*/
	private JMenu getFileMenu() {
		if (fileMenu == null) {
			fileMenu = new JMenu();
			fileMenu.setText("Archivo");
			fileMenu.add(getOpenDataMenuItem());
			fileMenu.add(getSaveDataMenuItem());
			fileMenu.addSeparator();
			fileMenu.add(getImportRawDataMenuItem());
			fileMenu.add(getSetOutputFileNameMenuItem());
```

```
                fileMenu.addSeparator();
                fileMenu.add(getExitMenuItem());
          }
          return fileMenu;
    }

    private JMenuItem getOpenDataMenuItem() {
          if (openDataMenuItem == null) {
                openDataMenuItem = new JMenuItem();
                openDataMenuItem.setText("Abrir datos...");


openDataMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O,Event.C
TRL_MASK, true));
                openDataMenuItem.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            gui.openData();
                      }
                });
          }
          return openDataMenuItem;
    }

    private JMenuItem getSaveDataMenuItem() {
          if (saveDataMenuItem == null) {
                saveDataMenuItem = new JMenuItem();
                saveDataMenuItem.setText("Guardar datos...");


saveDataMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S,Event.C
TRL_MASK, true));
                saveDataMenuItem.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            gui.saveData();
                      }
                });
          }
          return saveDataMenuItem;
    }

    private JMenuItem getImportRawDataMenuItem() {
          if (importRawDataMenuItem == null) {
                importRawDataMenuItem = new JMenuItem();
                importRawDataMenuItem.setText("Importar datos en
bruto...");


importRawDataMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_I,Ev
ent.CTRL_MASK, true));
                importRawDataMenuItem.addActionListener(new
ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            gui.importRawData();
                      }
                });
```

```
			}
			return importRawDataMenuItem;
		}

		private JMenuItem getSetOutputFileNameMenuItem() {
			if (setOutputFileNameMenuItem == null) {
				setOutputFileNameMenuItem = new JMenuItem();
				setOutputFileNameMenuItem.setText("Seleccionar archivo de
salida");


//setOutputFileNameMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.V
K_S,Event.CTRL_MASK, true));
				setOutputFileNameMenuItem.addActionListener(new
ActionListener() {
					public void actionPerformed(ActionEvent e) {
						gui.setOutputFileName();
						gui.scanCount = 0;
					}
				});
			}
			return setOutputFileNameMenuItem;
		}


		private JMenuItem getExitMenuItem() {
			if (exitMenuItem == null) {
				exitMenuItem = new JMenuItem();
				exitMenuItem.setText("Salir");


exitMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Q,Event.CTRL_
MASK, true));
				exitMenuItem.addActionListener(new ActionListener() {
					public void actionPerformed(ActionEvent e) {
						System.exit(0);
					}
				});
			}
			return exitMenuItem;
		}




/**************************************************************************
**
		 * Actions menu

**************************************************************************
*/
		private JMenu getActionsMenu() {
			if (actionMenu == null) {
				actionMenu = new JMenu();
				actionMenu.setText("Acciones");
```

```
                actionMenu.setVisible(false);
                actionMenu.add(getUpMenuItem());
                actionMenu.add(getDownMenuItem());
                actionMenu.add(getLeftMenuItem());
                actionMenu.add(getRightMenuItem());
                actionMenu.addSeparator();
                actionMenu.add(getScanMenuItem());
                actionMenu.addSeparator();
                actionMenu.add(getResetMenuItem());
          }
          return actionMenu;
    }

    private JMenuItem getUpMenuItem() {
          if (upMenuItem == null) {
                upMenuItem = new JMenuItem();
                upMenuItem.setText("Arriba");

    upMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_UP,
Event.ALT_MASK, true));
                upMenuItem.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            gui.moveCursorUp();
                      }
                });
          }
          return upMenuItem;
    }

    private JMenuItem getDownMenuItem() {
          if (downMenuItem == null) {
                downMenuItem = new JMenuItem();
                downMenuItem.setText("Abajo");

    downMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_DOWN,
Event.ALT_MASK, true));
                downMenuItem.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            gui.moveCursorDown();
                      }
                });
          }
          return downMenuItem;
    }

    private JMenuItem getLeftMenuItem() {
          if (leftMenuItem == null) {
                leftMenuItem = new JMenuItem();
                leftMenuItem.setText("Izquierda");

    leftMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_LEFT,
Event.ALT_MASK, true));
                leftMenuItem.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            gui.moveCursorLeft();
```

```
                        }
                  });
            }
            return leftMenuItem;
      }

      private JMenuItem getRightMenuItem() {
            if (rightMenuItem == null) {
                  rightMenuItem = new JMenuItem();
                  rightMenuItem.setText("Derecha");

      rightMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_RIGHT,
Event.ALT_MASK, true));
                  rightMenuItem.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              gui.moveCursorRight();
                        }
                  });
            }
            return rightMenuItem;
      }

      private JMenuItem getScanMenuItem() {
            if (scanMenuItem == null) {
                  scanMenuItem = new JMenuItem();
                  scanMenuItem.setText("Escanear");

      scanMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S,
Event.ALT_MASK, true));
                  scanMenuItem.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              gui.runScan();
                        }
                  });
            }
            return scanMenuItem;
      }

      private JMenuItem getResetMenuItem() {
            if (resetMenuItem == null) {
                  resetMenuItem = new JMenuItem();
                  resetMenuItem.setText("Resetear datos de la planta
actual");
                  resetMenuItem.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              gui.resetFloor();
                        }
                  });
            }
            return resetMenuItem;
      }
```

```
/*****************************************************************************
**
       * View menu

*****************************************************************************
*/
      private JMenu getViewMenu() {
            if (viewMenu == null) {
                  viewMenu = new JMenu();
                  viewMenu.setText("Vista");

                  ButtonGroup viewGroup = new ButtonGroup();

                  JRadioButtonMenuItem rbMenuItem;

                  rbMenuItem = getViewLocationMenuItem();
                  viewGroup.add(rbMenuItem);
                  viewMenu.add(rbMenuItem);

                  rbMenuItem = getViewDevicesMenuItem();
                  viewGroup.add(rbMenuItem);
                  viewMenu.add(rbMenuItem);

                  rbMenuItem = getViewScanMenuItem();
                  viewGroup.add(rbMenuItem);
                  viewMenu.add(rbMenuItem);

                  viewMenu.addSeparator();
                  viewMenu.add(getViewCoordsMenuItem());
                  viewMenu.add(getViewApsMenuItem());
                  viewMenu.add(getViewPointsMenuItem());
                  viewMenu.add(getViewPathsMenuItem());
            }
            return viewMenu;
      }

      private JRadioButtonMenuItem getViewLocationMenuItem() {
            if (viewLocationMenuItem == null) {
                  viewLocationMenuItem = new JRadioButtonMenuItem();

      viewLocationMenuItem.setText(GUI_Utils.encodeString("Localización"));


viewLocationMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_1,Eve
nt.CTRL_MASK, true));
                  viewLocationMenuItem.setSelected(true);
                  viewLocationMenuItem.addActionListener(new
ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              gui.locationMode();
                        }
                  });
            }
            return viewLocationMenuItem;
      }
```

```
	private JRadioButtonMenuItem getViewDevicesMenuItem() {
		if (viewDevicesMenuItem == null) {
			viewDevicesMenuItem = new JRadioButtonMenuItem();
			viewDevicesMenuItem.setText("Dispositivos");


viewDevicesMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_2,Even
t.CTRL_MASK, true));
			viewDevicesMenuItem.setSelected(true);
			viewDevicesMenuItem.addActionListener(new ActionListener()
{
				public void actionPerformed(ActionEvent e) {
					gui.devicesMode();
				}
			});
		}
		return viewDevicesMenuItem;
	}

	private JRadioButtonMenuItem getViewScanMenuItem() {
		if (viewScanMenuItem == null) {
			viewScanMenuItem = new JRadioButtonMenuItem();
			viewScanMenuItem.setText("Escaneo");


viewScanMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_3,Event.C
TRL_MASK, true));
			viewScanMenuItem.setSelected(true);
			viewScanMenuItem.addActionListener(new ActionListener() {
				public void actionPerformed(ActionEvent e) {
					gui.scanMode();
				}
			});
		}
		return viewScanMenuItem;
	}

	private JCheckBoxMenuItem getViewCoordsMenuItem() {
		if (viewCoordsMenuItem == null) {
			viewCoordsMenuItem = new JCheckBoxMenuItem("Ver
coordenadas en mapa", gui.showCoordsOnMap);
			viewCoordsMenuItem.addActionListener(new ActionListener()
{
				public void actionPerformed(ActionEvent e) {
					gui.showCoordsOnMap =
viewCoordsMenuItem.getState();
					gui.updateImage();
				}
			});
		}
		return viewCoordsMenuItem;
	}

	private JCheckBoxMenuItem getViewApsMenuItem() {
```

```
            if (viewApsMenuItem == null) {
                  viewApsMenuItem = new JCheckBoxMenuItem("Ver APs en mapa",
gui.showAps);
                  viewApsMenuItem.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              gui.showAps = viewApsMenuItem.getState();
                              gui.updateImage();
                        }
                  });
            }
            return viewApsMenuItem;
      }

      private JCheckBoxMenuItem getViewPointsMenuItem() {
            if (viewPointsMenuItem == null) {
                  viewPointsMenuItem = new JCheckBoxMenuItem("Ver puntos de
escaneo", gui.showPoints);
                  viewPointsMenuItem.addActionListener(new ActionListener()
{
                        public void actionPerformed(ActionEvent e) {
                              gui.showPoints =
viewPointsMenuItem.getState();
                              gui.updateImage();
                        }
                  });
            }
            return viewPointsMenuItem;
      }

      private JCheckBoxMenuItem getViewPathsMenuItem() {
            if (viewPathsMenuItem == null) {
                  viewPathsMenuItem = new JCheckBoxMenuItem("Ver ruta en
mapa", gui.showPaths);
                  viewPathsMenuItem.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              gui.showPaths = viewPathsMenuItem.getState();
                              gui.updateImage();
                        }
                  });
            }
            return viewPathsMenuItem;
      }



/**************************************************************************
**
       * Floors menu

**************************************************************************
*/
      private JMenu getFloorsMenu() {
            if (floorsMenu == null) {
                  floorsMenu = new JMenu();
                  floorsMenu.setText("Plantas");
```

```
                populateFloorsMenuItem();
            }
            return floorsMenu;
      }

      protected void populateFloorsMenuItem() {

            if (floorsMenu == null) return;
            floorsMenu.removeAll();

            floorsMenu.add(getAddMapMenuItem());
            floorsMenu.addSeparator();

            Iterator it = gui.floors.entrySet().iterator();
            while (it.hasNext()) {
                  Map.Entry e = (Map.Entry)it.next();
                  rbMenuItem =
createFloorsMenuItem(((Floor)e.getValue()).getName(), (Integer)e.getKey());
                  floorsGroup.add(rbMenuItem);
                  floorsMenu.add(rbMenuItem);
                  if ((Integer)e.getKey() == gui.activeFloor.getIdFloor())
                        rbMenuItem.setSelected(true);
            }
      }

      private JRadioButtonMenuItem createFloorsMenuItem(String name, final
int idFloor) {
            JRadioButtonMenuItem floorMenuItem = new JRadioButtonMenuItem();
            floorMenuItem.setText(name);
            floorMenuItem.setSelected(true);
            floorMenuItem.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        gui.changeActiveMap(idFloor);
                  }
            });
            return floorMenuItem;
      }

      private JMenuItem getAddMapMenuItem() {
            if (addMapMenuItem == null) {
                  addMapMenuItem = new JMenuItem();
                  addMapMenuItem.setText(GUI_Utils.encodeString("Añadir
nueva planta..."));


addMapMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_N,Event.CTR
L_MASK, true));
                  addMapMenuItem.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              gui.addMap();
                        }
                  });
            }
            return addMapMenuItem;
      }
```

```
/**************************************************************************
**
       * Help menu

**************************************************************************
*/
      private JMenu getHelpMenu() {
            if (helpMenu == null) {
                  helpMenu = new JMenu();
                  helpMenu.setText("Ayuda");
                  helpMenu.add(getAboutMenuItem());
            }
            return helpMenu;
      }

      private JMenuItem getAboutMenuItem() {
            if (aboutMenuItem == null) {
                  aboutMenuItem = new JMenuItem();
                  aboutMenuItem.setText("Sobre " + gui.appName +"...");
                  aboutMenuItem.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              new GUI_About(gui);
                        }
                  });
            }
            return aboutMenuItem;
      }
}
```

Location/src/location/gui/GUI_AddMap.java

```
package location.gui;

import java.awt.Dimension;
import java.awt.Point;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

/* 1.4 example used by DialogDemo.java. */
class GUI_AddMap extends JDialog
                    implements ActionListener,
                               PropertyChangeListener {

      private static final long serialVersionUID = -7248192065156130054L;

      private String name = null;
      private String filename = null;
      private int idFloor = 0;

      protected JTextField nameField;
      protected JTextField filenameField;
      protected JTextField idFloorField;

    private JOptionPane optionPane;

    private String okButton = "Aceptar";
    private String cancelButton = "Cancelar";

    private GUI_Main gui;


    /** Creates the reusable dialog. */
    public GUI_AddMap(GUI_Main gui, String filename) {

      // Adding a new Map
        super(gui, false);
        this.gui = gui;
        this.filename = filename;

        nameField = new JTextField(10);
        filenameField = new JTextField(10);
```

```
        idFloorField = new JTextField(10);

        filenameField.setText(filename);
        if (gui != null)
            idFloorField.setText((String.valueOf(gui.floorCount + 1)));

        createDialog();
    }

    private void createDialog() {

        setTitle(GUI_Utils.encodeString("Añadir mapa de planta"));

        //Create an array of the text and components to be displayed.
        String info = "Por favor, elija un nombre para el mapa,\n" +
                  GUI_Utils.encodeString("un fichero de imagen y el número
de la planta.\n\n");
        String name = "Nombre de la planta:";
        String filename = "Fichero de la planta:";
        String idFloors = GUI_Utils.encodeString("Número de planta:");

        Object[] array = { info,
                                name, nameField,
                                filename, filenameField,
                                idFloors, idFloorField};

        Object[] options = {okButton, cancelButton};

        //Create the JOptionPane.
        optionPane = new JOptionPane(array,
                                     JOptionPane.QUESTION_MESSAGE,
                                     JOptionPane.YES_NO_OPTION,
                                     null,
                                     options,
                                     options[0]);

        //Make this dialog display it.
        setContentPane(optionPane);
        pack();
        if (gui == null) {
              Point loc = gui.getLocation();
                  loc.translate(gui.getWidth()/2 - this.getWidth()/2,
gui.getHeight()/2 - this.getHeight()/2);
                  setLocation(loc);
        } else {
            Dimension ss = Toolkit.getDefaultToolkit().getScreenSize();
            int x = (int)(ss.getWidth()/2 - this.getWidth()/2);
            int y = (int)(ss.getHeight()/2 - this.getHeight()/2);
            setLocation(new Point(x,y));
        }
        setVisible(true);

        //Handle window closing correctly.
        setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
        addWindowListener(new WindowAdapter() {
```

```
                public void windowClosing(WindowEvent we) {
                    optionPane.setValue(new
Integer(JOptionPane.CLOSED_OPTION));
                }
            });

            //Ensure the text field always gets the first focus.
            addComponentListener(new ComponentAdapter() {
                public void componentShown(ComponentEvent ce) {
                nameField.requestFocusInWindow();
                }
            });

            //Register an event handler that puts the text into the option pane.
            nameField.addActionListener(this);
            filenameField.addActionListener(this);
            idFloorField.addActionListener(this);

            //Register an event handler that reacts to option pane state
changes.
            optionPane.addPropertyChangeListener(this);
        }

        /** This method handles events for the text field. */
        public void actionPerformed(ActionEvent e) {
            optionPane.setValue(okButton);
        }

        /** This method reacts to state changes in the option pane. */
        public void propertyChange(PropertyChangeEvent e) {
                String prop = e.getPropertyName();

                if (isVisible()
                            && (e.getSource() == optionPane)
                            && (JOptionPane.VALUE_PROPERTY.equals(prop) ||

          JOptionPane.INPUT_VALUE_PROPERTY.equals(prop))) {
                        Object value = optionPane.getValue();

                        if (value == JOptionPane.UNINITIALIZED_VALUE) {
                              return;
                        }

                        optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

                        if (okButton.equals(value)) {

                              name = nameField.getText();
                              filename = filenameField.getText();
                              try {
                                    idFloor =
Integer.parseInt(idFloorField.getText());
                                    if (!checkFloor(idFloor))
                                          floorError();
                              } catch (NumberFormatException error) {
```

```
                        floorError();
                    }
                    if (!name.equals("") && !filename.equals("")) {

                        // Add the floor
                        gui.newMap(name, filename, idFloor);
                        clearAndHide();
                        gui.updateImage();

                    } else {
                        dataError();
                    }
                } else { //user closed dialog or clicked cancel
                    clearAndHide();
                }
            }
      }

    // Check if floor id is available
    private boolean checkFloor(int floor) {
      if (gui.floors.containsKey((Integer)floor))
            return false;
      else
            return true;
    }

    // Floor not correct
    private void floorError() {

            idFloorField.selectAll();
            JOptionPane.showMessageDialog(GUI_AddMap.this,
                    GUI_Utils.encodeString("El número de planta no es
válido."), "Aceptar",
                    JOptionPane.ERROR_MESSAGE);
            name = null;
            filename = null;
            idFloor = 0;
            idFloorField.requestFocusInWindow();
    }

    // Nothing typed
    private void dataError() {

            nameField.selectAll();
            JOptionPane.showMessageDialog(GUI_AddMap.this,
                    "Por favor, introduzca los datos necesarios.",
                    "Aceptar", JOptionPane.ERROR_MESSAGE);
            name = null;
            filename = null;
            idFloor = 0;
            nameField.requestFocusInWindow();
    }

    // Clears the dialog and hides it
    public void clearAndHide() {
```

```
        nameField.setText(null);
        filenameField.setText(null);
        idFloorField = null;
            name = null;
            filename = null;
            idFloor = 0;
        setVisible(false);
        gui.addingAp = false;

        if (gui.wizardRunning) {
            gui.wizardRunning = false;
                  gui.setVisible(true);
        }
    }
}
```

Location/src/location/gui/GUI_ControlsDevices.java

```
package location.gui;

import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;

import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class GUI_ControlsDevices {

	private GUI_Main gui;

	private JPanel jButtonPanel = null;

	static final int BUTTONSIZE = 32;

	// Buttons and labels
	private JButton addApButton = null;
	private JLabel addApLabel = null;
	private JButton removeApButton = null;
	private JLabel removeApLabel = null;
	private JButton editApButton = null;
	private JLabel editApLabel = null;


	public GUI_ControlsDevices(GUI_Main gui) {
		this.gui = gui;
	}

	public JPanel getJButtonPane() {
		if (jButtonPanel == null) {
			jButtonPanel = new JPanel();
			jButtonPanel.setLayout(new GridBagLayout());

			GridBagConstraints c = new GridBagConstraints();
			c.insets = new Insets(5, 5, 5, 5);
			c.fill = GridBagConstraints.HORIZONTAL;

			// Controls
			c.gridx = 0;
			c.gridy = 0;
			jButtonPanel.add(getAddApButton(), c);
```

```
                c.gridx = 1;
                c.gridy = 0;
                jButtonPanel.add(getAddApLabel(), c);

                c.gridx = 0;
                c.gridy = 1;
                jButtonPanel.add(getRemoveApButton(), c);

                c.gridx = 1;
                c.gridy = 1;
                jButtonPanel.add(getRemoveApLabel(), c);

                c.gridx = 0;
                c.gridy = 2;
                jButtonPanel.add(getEditApButton(), c);

                c.gridx = 1;
                c.gridy = 2;
                jButtonPanel.add(getEditApLabel(), c);
          }
          return jButtonPanel;
    }

    private JButton getAddApButton() {
          if (addApButton == null) {
                Icon icon = new
ImageIcon(GUI_Utils.getImageFileURL(gui.imagesDir + "scan.png"));
                addApButton = new JButton(icon);
                addApButton.setPreferredSize(new Dimension(BUTTONSIZE,
BUTTONSIZE));
                addApButton.setMnemonic(KeyEvent.VK_A);
                addApButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                      new GUI_AddAP(gui);
                }
          });
          }
          return addApButton;
    }

    private JLabel getAddApLabel() {
          if (addApLabel == null) {
                addApLabel = new JLabel(GUI_Utils.encodeString("Añadir
Punto de Acceso"));
          }
          return addApLabel;
    }

    private JButton getRemoveApButton() {
          if (removeApButton == null) {
                Icon icon = new
ImageIcon(GUI_Utils.getImageFileURL(gui.imagesDir + "scan_no.png"));
                removeApButton = new JButton(icon);
                removeApButton.setPreferredSize(new Dimension(BUTTONSIZE,
BUTTONSIZE));
```

```
                removeApButton.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            gui.removeAP();
                      }
                });
          }
          return removeApButton;
    }

    private JLabel getRemoveApLabel() {
          if (removeApLabel == null) {
                removeApLabel = new JLabel("Eliminar Punto de Acceso");
          }
          return removeApLabel;
    }

    private JButton getEditApButton() {
          if (editApButton == null) {
                Icon icon = new
ImageIcon(GUI_Utils.getImageFileURL(gui.imagesDir + "scan_edit.png"));
                editApButton = new JButton(icon);
                editApButton.setPreferredSize(new Dimension(BUTTONSIZE,
BUTTONSIZE));
                editApButton.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            gui.modifyAPGet();
                      }
                });
          }
          return editApButton;
    }

    private JLabel getEditApLabel() {
          if (editApLabel == null) {
                editApLabel = new JLabel("Editar Punto de Acceso");
          }
          return editApLabel;
    }
}
```

Location/src/location/gui/GUI_AddAP.java

```
package location.gui;

import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

class GUI_AddAP extends JDialog
                    implements ActionListener,
                               PropertyChangeListener {

      private static final long serialVersionUID = 7112677376787639166L;

      private String mac = null;
      private String essid = null;
      private int x = 0;
      private int y = 0;
      private int floor = 0;
      protected JTextField macField;
      protected JTextField essidField;
      protected JTextField xField;
      protected JTextField yField;
      protected JTextField floorField;

    private JOptionPane optionPane;

    private String okButton = "Aceptar";
    private String cancelButton = "Cancelar";

    private GUI_Main gui;

    private boolean modifying = false;


    /** Creates the reusable dialog. */
    public GUI_AddAP(GUI_Main gui) {

      // Creating a new Access Point
```

```
        super(gui, false);
        this.gui = gui;
        gui.addingAp = true;
        gui.addAPDialog = this;

        macField = new JTextField(10);
        essidField = new JTextField(10);
        xField = new JTextField(10);
        yField = new JTextField(10);
        floorField = new JTextField(10);

        xField.setText(String.valueOf(gui.x));
        yField.setText(String.valueOf(gui.y));
        floorField.setText(String.valueOf(gui.activeFloor.getIdFloor()));

        createDialog();
    }

    public GUI_AddAP(GUI_Main gui, String mac, String essid, int x, int y,
int floor) {

      // We're modifying an Access Point

        super(gui, false);
        this.gui = gui;
        gui.addingAp = true;
        gui.addAPDialog = this;
        modifying = true;

        macField = new JTextField(10);
        essidField = new JTextField(10);
        xField = new JTextField(10);
        yField = new JTextField(10);
        floorField = new JTextField(10);

        macField.setText(mac);
        essidField.setText(essid);
        xField.setText(String.valueOf(x));
        yField.setText(String.valueOf(y));
        floorField.setText(String.valueOf(gui.activeFloor.getIdFloor()));

        createDialog();
    }

    private void createDialog() {

        setTitle(GUI_Utils.encodeString("Añadir Punto de Acceso"));

        //Create an array of the text and components to be displayed.
        String info = "Por favor, introduzca la MAC y el ESSID\n" +
                          "del Punto de Acceso, y seleccione un\n" +
                          "punto en el mapa para añadirlo.\n\n";
        info = GUI_Utils.encodeString(info);
        String mac = "MAC del Punto de Acceso:";
```

```
        String essid = "ESSID  del Punto de Acceso:";
        String posx = "Posición en el eje X:";
        String posy = "Posición en el eje Y";
        posx = GUI_Utils.encodeString(posx);
        posy = GUI_Utils.encodeString(posy);
        String planta = "Planta";
        Object[] array = { info,
                                mac, macField,
                                essid, essidField,
                                posx, xField,
                                posy, yField,
                                planta, floorField};

        //Create an array specifying the number of dialog buttons
        //and their text.
        Object[] options = {okButton, cancelButton};

        Icon icon = new ImageIcon(GUI_Utils.getImageFileURL(gui.imagesDir +
"scan.png"));

        //Create the JOptionPane.
        optionPane = new JOptionPane(array,
                                     JOptionPane.QUESTION_MESSAGE,
                                     JOptionPane.YES_NO_OPTION,
                                     icon,
                                     options,
                                     options[0]);

        //Make this dialog display it.
        setContentPane(optionPane);
        pack();
        Point loc = gui.getLocation();
            loc.translate(gui.getWidth()/2 - this.getWidth()/2,
gui.getHeight()/2 - this.getHeight()/2);
            setLocation(loc);
        setVisible(true);

        //Handle window closing correctly.
        setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
        addWindowListener(new WindowAdapter() {
                public void windowClosing(WindowEvent we) {
                /*
                 * Instead of directly closing the window,
                 * we're going to change the JOptionPane's
                 * value property.
                 */
                    optionPane.setValue(new
Integer(JOptionPane.CLOSED_OPTION));
            }
        });

        //Ensure the text field always gets the first focus.
        addComponentListener(new ComponentAdapter() {
            public void componentShown(ComponentEvent ce) {
                macField.requestFocusInWindow();
```

```
            }
        });

        //Register an event handler that puts the text into the option pane.
        macField.addActionListener(this);
        essidField.addActionListener(this);

        //Register an event handler that reacts to option pane state
changes.
        optionPane.addPropertyChangeListener(this);
    }

    /** This method handles events for the text field. */
    public void actionPerformed(ActionEvent e) {
        optionPane.setValue(okButton);
    }

    /** This method reacts to state changes in the option pane. */
    public void propertyChange(PropertyChangeEvent e) {
            String prop = e.getPropertyName();

            if (isVisible()
                        && (e.getSource() == optionPane)
                        && (JOptionPane.VALUE_PROPERTY.equals(prop) ||
JOptionPane.INPUT_VALUE_PROPERTY
                                        .equals(prop))) {
                  Object value = optionPane.getValue();

                  if (value == JOptionPane.UNINITIALIZED_VALUE) {
                        //ignore reset
                        return;
                  }

                  //Reset the JOptionPane's value.
                  //If you don't do this, then if the user
                  //presses the same button next time, no
                  //property change event will be fired.
                  optionPane.setValue(JOptionPane.UNINITIALIZED_VALUE);

                  if (okButton.equals(value)) {

                        mac = macField.getText().toUpperCase();
                        essid = essidField.getText();
                        try {
                              x = Integer.parseInt(xField.getText());
                              y = Integer.parseInt(yField.getText());
                              floor =
Integer.parseInt(floorField.getText());
                        } catch (NumberFormatException error) {
                              coordsError();
                        }
                        if (!mac.equals("") && !essid.equals("") &&
checkPos(x, y)) {
                              if (checkMAC(mac)) {
                                    // Everything correct, add it to the
```

```
table
                                    if (!modifying)
                                          // Create new AP
                                          gui.addAP(mac, essid, x, y,
floor);
                                    else
                                          // Modifying AP
                                          gui.modifyAPSet(mac, essid, x, y,
floor);
                                    clearAndHide();
                                    gui.updateImage();
                              } else {
                                    // MAC not correct
                                    macError();
                              }
                        } else {
                              if (checkPos(x, y)) {
                                    // Nothing typed
                                    dataError();
                              }
                              else {
                                    // Position not correct
                                    coordsError();
                              }
                        }
                  } else { //user closed dialog or clicked cancel
                        mac = null;
                        essid = null;
                        x = 0;
                        y = 0;
                        floor = 0;
                        clearAndHide();
                  }
            }
      }

    // Check correct MAC format
    private boolean checkMAC(String mac) {
      Pattern p = Pattern.compile("^[0-9a-fA-F]{2}:[0-9a-fA-F]{2}:[0-9a-fA-
F]{2}:[0-9a-fA-F]{2}:[0-9a-fA-F]{2}:[0-9a-fA-F]{2}$");
      Matcher m = p.matcher(mac);
      return m.find();
    }

    // Check if position is inside map
    private boolean checkPos(int x, int y) {
      if (x < 0 || y < 0 || x > gui.image.getIconWidth() || y >
gui.image.getIconHeight())
            return false;
      else
            return true;
    }

    // Position not correct
    private void coordsError() {
```

```
            macField.selectAll();
            JOptionPane.showMessageDialog(GUI_AddAP.this,
                        GUI_Utils.encodeString("Las coordenadas no son
válidas."), "Aceptar",
                        JOptionPane.ERROR_MESSAGE);
            mac = null;
            essid = null;
            x = 0;
            y = 0;
            floor = 0;
            xField.requestFocusInWindow();
    }

    // MAC format not correct
    private void macError() {
            macField.selectAll();
            JOptionPane.showMessageDialog(GUI_AddAP.this,
                        "La MAC introducida no es correcta.",
                        "Aceptar", JOptionPane.ERROR_MESSAGE);
            mac = null;
            essid = null;
            x = 0;
            y = 0;
            floor = 0;
            macField.requestFocusInWindow();
    }

    // Nothing typed
    private void dataError() {

            xField.selectAll();
            JOptionPane.showMessageDialog(GUI_AddAP.this,
                        "Por favor, introduzca los datos necesarios.",
                        "Aceptar", JOptionPane.ERROR_MESSAGE);
            mac = null;
            essid = null;
            x = 0;
            y = 0;
            floor = 0;
            macField.requestFocusInWindow();
    }

    /** This method clears the dialog and hides it. */
    public void clearAndHide() {
        macField.setText(null);
        essidField.setText(null);
        mac = null;
      essid = null;
        setVisible(false);
        gui.addingAp = false;
    }
}
```

Location/src/location/types/AP.java

```
package location.types;

import java.awt.Point;
import java.io.Serializable;


public class AP implements Comparable, Serializable {

	private static final long serialVersionUID = -302128733987236478L;

	private String mac, essid;
	private Point point;
	private int floor;

	public AP(String mac, String essid) {
		this.mac = mac;
		this.essid = essid;
		this.point = new Point(-1, -1);
		this.floor = 0;
	}

	public AP(String mac, String essid, Point point, int floor) {
		this.mac = mac;
		this.essid = essid;
		this.point = point;
		this.floor = floor;
	}

	public AP(String mac, String essid, int x, int y, int floor) {
		this.mac = mac;
		this.essid = essid;
		this.point = new Point(x, y);
		this.floor = floor;
	}

	public int compareTo(Object o) throws ClassCastException {
		if (!(o instanceof AP))
			throw new ClassCastException("An AP object expected.");

		AP ap = (AP)o;

		if (mac.compareTo(ap.getMac()) != 0)
			return mac.compareTo(ap.getMac());
		else if (essid.compareTo(ap.getEssid()) != 0)
			return essid.compareTo(ap.getEssid());
		else if (getX()- ap.getX() != 0)
			return getX() - ap.getX();
		else if (getY() - ap.getY() != 0)
			return getY() - ap.getY();
		else if (floor - ap.getFloor() != 0)
			return floor - ap.getFloor();
```

```
		else return 0;
	}

	public String getEssid() {
		return essid;
	}

	public void setEssid(String essid) {
		this.essid = essid;
	}

	public int getFloor() {
		return floor;
	}

	public void setFloor(int floor) {
		this.floor = floor;
	}

	public String getMac() {
		return mac;
	}

	public void setMac(String mac) {
		this.mac = mac;
	}

	public int getX() {
		return (int)point.getX();
	}

	public void setX(int x) {
		this.point.setLocation(x, point.getY());
	}

	public int getY() {
		return (int)point.getY();
	}

	public void setY(int y) {
		this.point.setLocation(point.getX(), y);
	}

	public Point getPoint() {
		return point;
	}

	public void setPoint(Point point) {
		this.point = point;
	}
}
```

Location/src/location/types/Floor.java

```
package location.types;

import java.awt.Point;
import java.io.Serializable;
import java.util.HashMap;
import java.util.Vector;

import javax.swing.ImageIcon;

public class Floor implements Serializable {

	private static final long serialVersionUID = -7517807854139671257L;

	protected HashMap<String, HashMap<Point, SignalVector>> data = null;
	protected Vector<Point> scanPoints = null;
	protected Vector<AP> scanAps = null;
	protected HashMap<Point, Vector<Point>> scanPaths = null;
	protected int scanCount;
	protected String name = null;
	protected String mapFileName = null;
	protected ImageIcon mapImage = null;
	protected int idFloor;

	public Floor(String name, String mapFileName, int idFloor) {
		data = new HashMap<String, HashMap<Point, SignalVector>>();
		scanPoints = new Vector<Point>();
		scanAps = new Vector<AP>();
		scanPaths = new HashMap<Point, Vector<Point>>();

		this.scanCount = 0;
		this.name = name;
		this.mapFileName = mapFileName;
		this.mapImage = new ImageIcon(mapFileName);
		this.idFloor = idFloor;
	}

	public Floor(String name, String mapFileName, ImageIcon mapImage, int
idFloor) {
		data = new HashMap<String, HashMap<Point, SignalVector>>();
		scanPoints = new Vector<Point>();
		scanAps = new Vector<AP>();
		scanPaths = new HashMap<Point, Vector<Point>>();

		this.scanCount = 0;
		this.name = name;
		this.mapFileName = mapFileName;
		this.mapImage = mapImage;
		this.idFloor = idFloor;
	}

	public ImageIcon getMapImage() {
```

```
		return mapImage;
	}

	public void setMapImage(ImageIcon mapImage) {
		this.mapImage = mapImage;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public void setIdFloor(int idFloor) {
		this.idFloor = idFloor;
	}

	public HashMap<String, HashMap<Point, SignalVector>> getData() {
		return data;
	}

	public void setData(HashMap<String, HashMap<Point, SignalVector>>
data) {
		this.data = data;
	}

	public int getIdFloor() {
		return idFloor;
	}

	public void setIdPlanta(int idPlanta) {
		this.idFloor = idPlanta;
	}

	public String getMapFileName() {
		return mapFileName;
	}

	public void setMapFileName(String mapFileName) {
		this.mapFileName = mapFileName;
	}

	public Vector<AP> getScanAps() {
		return scanAps;
	}

	public void setScanAps(Vector<AP> scanAps) {
		this.scanAps = scanAps;
	}

	public int getScanCount() {
		return scanCount;
	}
```

```
	public void setScanCount(int scanCount) {
		this.scanCount = scanCount;
	}

	public HashMap<Point, Vector<Point>> getScanPaths() {
		return scanPaths;
	}

	public void setScanPaths(HashMap<Point, Vector<Point>> scanPaths) {
		this.scanPaths = scanPaths;
	}

	public Vector<Point> getScanPoints() {
		return scanPoints;
	}

	public void setScanPoints(Vector<Point> scanPoints) {
		this.scanPoints = scanPoints;
	}

}
```

Location/src/location/types/SignalVector.java

```
package location.types;

import java.io.Serializable;
import java.util.Vector;

public class SignalVector implements Serializable {
	private static final long serialVersionUID = 7278309722357410086L;
	private double meanValue;
	private Vector<Double> values;

	public SignalVector() {
		this.meanValue = 0;
		//this.meanValue = Integer.MAX_VALUE;
		this.values = new Vector<Double>();
	}

	public SignalVector(double value) {
		this();
		this.add(value);
	}

	public void add(double value) {
		this.values.add(value);
		this.meanValue = calculateMeanValue();
		//if (value < this.meanValue) this.meanValue = value;
	}

	private double calculateMeanValue() {
		double sum = 0;
		for (int i = 0; i < values.size(); i++) {
			sum += values.elementAt(i);
		}
		return sum / values.size();
	}

	public double getMeanValue() {
		return meanValue;
	}

	public void setMeanValue(double meanValue) {
		this.meanValue = meanValue;
	}

	public Vector<Double> getValues() {
		return values;
	}

	public void setValues(Vector<Double> values) {
		this.values = values;
	}
}
```

Location/src/location/types/UneditableTableModel.java

```
package location.types;

import javax.swing.table.DefaultTableModel;

/*
 * This is a DefaultTableModel with uneditable cells.
 */

public class UneditableTableModel extends DefaultTableModel {

	private static final long serialVersionUID = 83990223218552452 46L;

	public boolean isCellEditable(int row, int col) {
		return false;
	}
}
```

GLOSARIO

- **802.11**: El protocolo *IEEE* 802.11 o *WI-FI* es un estándar de protocolo de comunicaciones del *IEEE* que define el uso de los dos niveles más bajos de la arquitectura *OSI* (capas física y de enlace de datos), especificando sus normas de funcionamiento en una *WLAN*. En general, los protocolos de la rama 802.x definen la tecnología de redes de área local.

- **ACK (*ACKNOWLEDGEMENT*)**: en comunicaciones entre computadores, es un mensaje que se envía para confirmar que un mensaje o un conjunto de mensajes han llegado. Si el terminal de destino tiene capacidad para detectar errores, el significado de ACK es "ha llegado y además ha llegado correctamente".

- **DHCP (*Dynamic Host Configuration Protocol*)**: es un protocolo de red que permite a los nodos de una red IP obtener sus parámetros de configuración automáticamente. Se trata de un protocolo de tipo cliente/servidor en el que generalmente un servidor posee una lista de direcciones IP dinámicas y las va asignando a los clientes conforme estas van estando libres, sabiendo en todo momento quien ha estado en posesión de esa IP, cuanto tiempo la ha tenido, a quien se la ha asignado después.

- **Driver**: programa informático que permite al sistema operativo interactuar con un periférico, haciendo una abstracción del hardware y proporcionando una interfaz para usarlo. Se puede esquematizar como un manual de instrucciones que le indica cómo debe controlar y comunicarse con un dispositivo en particular. Por tanto, es una pieza esencial, sin la cual no se podría usar el hardware.

- **ESSID (*Extended Service Set IDentifier*)**: es un código incluido en todos los paquetes de una red inalámbrica (*Wi-Fi*) para identificarlos como parte de esa red. El código consiste en un máximo de 32 caracteres alfanuméricos. Todos los dispositivos inalámbricos que intentan comunicarse entre sí deben compartir el mismo ESSID. A menudo al ESSID se le conoce como nombre de la red.

- **Framework**: estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros softwares para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

- **GNU/Linux**: el sistema operativo que utiliza el *kernel Linux* en conjunto con las aplicaciones de sistema creadas por el proyecto GNU y de varios otros proyectos/grupos de software.

- **GPS (*Global Positioning System*)**: es un Sistema Global de Navegación por Satélite (*GNSS*) el cual permite determinar en todo el mundo la posición de un objeto, una persona, un vehículo o una nave, con una precisión hasta de centímetros usando GPS diferencial, aunque lo habitual son unos pocos metros.

- **ICMP (*Internet Control Message Protocol*)**: subprotocolo de diagnóstico y notificación de errores del Protocolo de Internet (IP). Como tal, se usa para enviar mensajes de error, indicando por ejemplo que un servicio determinado no está disponible o que un *router* o *host* no puede ser localizado.

- **IP, dirección**: número que identifica de manera lógica y jerárquica a una interfaz de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo IP, que corresponde al nivel de red o nivel 3 del modelo de referencia *OSI*. Dicho número no se ha de confundir con la dirección MAC que es un número hexadecimal fijo que es asignado a la tarjeta o dispositivo de red por el fabricante, mientras que la dirección IP se puede cambiar.

- **IP, protocolo (*Internet Protocol*)**: protocolo orientado de datos, usado tanto por el origen como por el destino para la comunicación de estos a través de una red (Internet) de paquetes conmutados**.**

  Los datos en una red que se basa en IP son enviados en bloques conocidos como paquetes o datagramas. En particular, en IP no se necesita ninguna configuración antes de que un equipo intente enviar paquetes a otro con el que no se había comunicado antes.

- **MAC (*Media Access Control address*)**: en redes de computadoras la dirección MAC es un identificador hexadecimal de 48 bits que se corresponde de forma única con una tarjeta o interfaz de red. Es individual, cada dispositivo tiene su propia dirección MAC determinada y configurada por el *IEEE* (los primeros 24 bits) y el fabricante (los últimos 24 bits). La mayoría de los protocolos que trabajan en la capa 2 del modelo OSI usan una de las tres numeraciones manejadas por el *IEEE*: MAC-48, EUI-48, y EUI-64 las cuales han sido diseñadas para ser identificadores globalmente únicos. No todos los protocolos de comunicación usan direcciones MAC, y no todos los protocolos requieren identificadores globalmente únicos.

- **NDIS (Network Driver Interface Specification)**: estándar que permite convivir a los múltiples protocolos de transporte con los diversos adaptadores de red. NDIS permite a los componentes de los protocolos ser independientes de la tarjeta de red

- **PDA (*Personal Digital Assistant*)**: es un computador de mano originalmente diseñado como agenda electrónica (calendario, lista de contactos, bloc de notas y recordatorios) con un sistema de reconocimiento de escritura.

- **Ping**: Se trata de una utilidad que comprueba el estado de la conexión con uno o varios equipos remotos por medio de los paquetes de solicitud de eco y de respuesta de eco (definidos en el protocolo de red *ICMP*) para determinar si un sistema *IP* específico es accesible en una red. Es útil para diagnosticar los errores en redes o enrutadores *IP*.

- **Punto de acceso [AP] (*Access Point*)**: en redes de computadoras es un dispositivo que interconecta dispositivos de comunicación inalámbrica para formar una red inalámbrica. Los puntos de acceso inalámbricos tienen direcciones *IP* asignadas, para poder ser configurados.

- **RTT (*Round Trip Time*)**: se aplica en el mundo de las telecomunicaciones y redes informáticas al tiempo que tarda un paquete enviado desde un emisor en volver a este mismo emisor habiendo pasado por el receptor de destino.

- **Tablet PC**: es un ordenador a medio camino entre una computadora portátil y un *PDA*, en el que se puede escribir a través de una pantalla táctil. Un usuario puede utilizar un "lápiz" para trabajar con el ordenador sin necesidad de teclado o ratón.

- **Wireless**: referido a las telecomunicaciones, se aplica el término inalámbrico (inglés *wireless*, sin cables) al tipo de comunicación en la que no se utiliza un medio de propagación físico, sino se utiliza la modulación de ondas electromagnéticas, las cuales se propagan por el espacio sin un medio físico que comunique cada uno de los extremos de la transmisión.

REFERENCIAS

[1] Chin-Liang Wang, Yih-Shyh Chiou y Sheng-Cheng Yeh, "An indoor location scheme based on wireless local area networks", en *Consumer Communications and Networking Conference*, 2005.

[2] Frederic Evennou y Francois Marx , "Advanced Integration of WiFi and Inertial Navigation Systems for Indoor Mobile Positioning ", en *EURASIP Journal on Applied Signal Processing*, 2006.

[3] F. Barceló, F. Evennou, L. de Nardis y P. Tomé, "Advances in indoor Location", en *LIAISON - ISHTAR Workshop*, 2006.

[4] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon y Tim Clapp, " A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking ", en *IEEE Transactions on Signal Processing,* 2001.

[5] Frédéric Evennou y François Marx, "Improving positioning capabilities for indoor environments with WiFi ", en *IST Summit*, 2005.

[6] Marc Ciurana, Francisco Barceló y Sebastiano Cugno, "Indoor Tracking in WLAN Location with TOA Measurements ", en *The 4-th ACM International Workshop on Mobility Management and Wireless Access*, pp. 121-125, Oct. 2006.

[7] Lin Liao, Dieter Fox, Jeffrey Hightower, Henry Kautz y Dirk Schulz, "Voronoi Tracking: Location Estimation Using Sparse and Noisy Sensor Data", en *Proceedings of the IEEE International Conference on Robots and Systems,* 2003.

[8] David Sánchez, Sergio Afonso, Elsa M. Macías, Member IAENG y Álvaro Suárez, "Devices Location in 802.11 Infrastructure Networks using Triangulation ", IMECS 2006.

[9] Eiman Elnahrawy, Xiaoyan Li y Richard P. Martin, "The Limits of Localization Using Signal Strength: A Comparative Study ", en *Sensor and Ad Hoc Communications and Networks*, 2004.

[10] Srdjan Capkun, Maher Hamdi y Jean-Pierre Hubaux, "GPS-free positioning in mobile ad hoc networks", en *34th Annual Hawaii International Conference on System Sciences*, 2001.

[11] Dhruv Pandya, Ravi Jain, E. Lupu, "Indoor location estimation using multiple wireless technologies", en *Personal, Indoor and Mobile Radio Communications*, 2003

[12] P. Prasithsangaree, P. Krishnamurthy y P.K. Chrysanthis, “On indoor position location with wireless lans”, en *13th IEEE Int'l Symposium on Personal, Indoor, and Mobile Radio Communications*, 2002.

[13] Yi-Chao Chena, Ji-Rung Chianga, Hao-hua Chua,b, Polly Huangb y Arvin Wen Tsui, “Sensor-Assisted Wi-Fi Indoor Location System for Adapting to Environmental Dynamics ”, en *International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, 2005.

[14] Daryl Wilding-McBride, “Java Development on PDAs: Building Applications for Pocket PC and Palm Devices”, Addison-Wesley Professional, 2003.

[15] Yuanli Wang, “Building Java Applications on PDA”, 2002.

[16] Handhelds.org – Información sobre dispositivos móviles y Linux
http://www.handhelds.org

[17] OpenEmbedded – Metadistribución Linux para PDAs
http://www.openembedded.org/

[18] Distribución Familiar Linux para PDAs
http://familiar.handhelds.org/

[19] Distribución Angström Linux para PDAs
http://www.angstrom-distribution.org/

[20] Maemo – Entorno de desarrollo Linux para *Internet Tablets*
http://maemo.org/

[21] OpenMoko – Plataforma de desarrollo Linux para teléfonos
http://www.openmoko.org/

[22] Entornos gráficos para PDAs GPE y OPIE
http://gpe.linuxtogo.org/ http://opie.handhelds.org/

[23] Software de localización
http://www.herecast.com/

[24] Configurador de perfiles inalámbricos para Windows XP
http://www.engl.co.uk/products/zwlancfg/

[25] Framework de desarrollo gráfico en Java
http://www.jhotdraw.org/

[26] WebSphere Everyplace Micro Environment y J9 JVM
http://www-306.ibm.com/software/wireless/weme/

[27] Java on PocketPC (Unofficial FAQ)
http://blog.vikdavid.com/2004/12/java_on_pocketp.html

[28] Mysaifu JVM
http://www2s.biglobe.ne.jp/~dat/java/project/jvm/index_en.html

[29] NSICOM CrEme
http://nsicom.com/Default.aspx?tabid=138

AGRADECIMIENTOS

Nos gustaría dar las gracias a las siguientes personas por la ayuda prestada durante el desarrollo de este proyecto:

Manuel Ortega Ortíz de Apodaca

Diego Gachet Páez

Luis Couto Cortegoso

Andrew Hines

Daniel Gómez González

Manuel Hernández Urrea

María Teresa Hortalá Gómez

Antonio Gavilanes Franco

Javier Resano Ezcaray

Carmen Rodrigo Martín

Rubén Fuentes Fernández

Guillermo Jiménez Díaz

Familiares y amigos