Universidad Politécnica de Valencia

Departamento de Sistemas Informáticos y Computación

Máster Universitario en Ingeniería del Software, Métodos Formales y Sistemas de Información

Tesis Final de Máster

Simulador para la recreación de comportamiento de redes 2G, 3G, 4G y WLAN en escenarios sintéticos y realistas

Instituto de Telecomunicaciones y Aplicaciones Multimedia

Autor: Javier García San Juan
Director DSIC: Juan Sánchez Díaz

2013-2014

Agradecimientos

La realización de la tesis final de un máster conlleva el término de una etapa en la que merece la pena echar la vista atrás y recordar todo lo pasado, tanto por el fruto de meses de desarrollo como por la conclusión de un máster como es el de ingeniería del software, métodos formales y sistemas de información a través de un año de duro trabajo, tanto por la dedicación al máster como por el trabajo en el instituto de telecomunicaciones y aplicaciones multimedia (iTEAM) . Por esto, quiero agradecer con estas palabras a todos aquellos que me han ayudado en este camino, a veces duro, pero pleno y satisfactorio.

En primer lugar, reconocer a mis padres el gran esfuerzo realizado en ayudarme de forma moral y económica para mi formación como ser humano y como profesional. Gracias por dedicar gran parte de vuestra vida en guiarme en el camino. A David, mi hermano pequeño, que siempre me ha ayudado con su buen humor en las situaciones difíciles y en general a toda mi familia, que siempre han estado ahí cuando la he necesitado.

También gracias a mi tutor Juan por su orientación y sus consejos durante todo el proyecto. Su disponibilidad y el buen trato tenido siempre conmigo han sido de gran valor para mí.

Gratificar a todos los compañeros que he podido cruzarme en el máster y que han sabido enseñarme valores como la amistad, compañerismo, trabajo en grupo y solidaridad.

Muchas gracias a todos mis amigos por todos aquellos buenos momentos vividos y los que nos quedan por vivir.

Y por supuesto a Virginia, mi novia, por haber estado ahí todos estos años, tanto en los buenos como en los malos momentos, apoyándome en cada momento de mi vida.

Resumen

El objetivo principal de la tesis es la descripción del desarrollo de la parte "front-end" y la estructura de la base de datos de un simulador para la recreación de comportamientos de redes 2G, 3G, 4G y Wlan para poder permitir, entre otras, simular a una compañía telefónica el comportamiento de sus redes en una ciudad determinada y visualizar toda aquella información necesaria para poder desplegar sus redes de forma eficiente.

La interfaz del simulador ha sido desarrollada principalmente bajo el lenguaje Java ya que se trata de un lenguaje multiplataforma, que es el principal objetivo de esta interfaz del simulador, y además esta orientado a objetos, lo cual es una ventaja para los múltiples parámetros que necesita una simulación. Por otro lado, también se ha desarrollado una parte mediante una plataforma web, llamada ExtJS, la cual nos permite simular escenario realistas gracias a su api de Google Maps.

Concretamente, la parte de la interfaz del sistema simulador de escenarios de comportamientos de redes debe ser capaz de realizar los siguientes escenarios en el sistema:

- Gestión de Acceso/Creación de usuarios.
- Visualización de licencias activas.
- Definición y creación de la base de datos del simulador.
- Listado de simulaciones sintéticas y realistas.
- Consulta/Edición de simulaciones.
- Gestión de simulaciones realistas externas mediante la Api de Google Maps.
- Posibilidad de borrado, simulación, clonación y paradas de simulación múltiples mediante la utilización de hilos en Java.
- Visualización de los resultados en gráficos: Eficiencia espectral de celda, Rendimiento, SNR y BLER.
- Gestión de simulaciones realistas internas.
- Posibilidad de exportar los datos a formato csv.
- Visualización de las imágenes resultantes para escenarios realistas.

El trabajo está estructurado como sigue: en el primer capítulo se introduce el objeto del trabajo y su motivación, los capítulos del 2 al 5 se corresponden con las fases de desarrollo de software del Proceso Unificado Rational (RUP), el capítulo 6 esta dedicado al trabajo futuro, mientras que el capítulo 7 se dedica a las conclusiones finales, finalmente, el capítulo 8 contiene las referencias bibliográficas.

Palabras Clave

Simulación de redes, Java, base de datos, ExtJS.

Índice General

Índice de Figuras

1 Introducción

El siguiente documento tiene como objetivo detallar la tesis realizada para el termino del máster en ingeniería del software, métodos formales y sistemas de información, que consiste en el desarrollo de una interfaz gráfica junto a su base de datos para un simulador que recrea el comportamiento de redes 2G, 3G, 4G y WLAN tanto en escenarios sintéticos como realistas, todo ello desarrollado en el instituto de telecomunicaciones y aplicaciones multimedia (iTEAM) colaborando con la empresa SistelNetworks.

El simulador consiste en una aplicación multiplataforma que tiene tres partes básicas, en primer lugar una interfaz desarrollada en Java junto con una api de Google Maps basada en ExtJS para la configuración de simulaciones en escenarios sintéticos y realistas, por otro lado se tiene una compleja base de datos implementada en MySQL donde se guardaran los diferentes datos que se han ido detallando en la interfaz gráfica, finalmente se tiene el "core" desarrollado en C++, o también llamado núcleo, esta parte es donde se simulan las distintas configuraciones cuando el usuario lo indica, utilizando un servidor para poder simular utilizando el máximo de "cores" disponibles o contratados.

Esta arquitectura supone para el usuario la posibilidad de configurar las diferentes simulaciones de forma sencilla e intuitiva, además de poder mandar las distintas simulaciones al servidor de forma que este se encargue de realizar la tarea de gestionar la cola y realizar cada una de ellas, almacenando cada una de los resultados en la base de datos.

Las ventajas de esta arquitectura es que por un lado es el servidor el que gestiona la cola de simulaciones y se encarga de lanzar los distintos procesos y por otro lado que la tarea más lenta, la de realizar las simulaciones, es desarrollada por el servidor, y, por tanto, el usuario tan solo se encarga de la configuración previa de las simulaciones y de la visualización de los resultados tras la simulación.

En lo referente al desarrollo de la memoria y del proyecto se utilizará tecnología orientada a objetos, las fases de análisis y diseño se llevarán a cabo empleando una adaptación de RUP (*Rational Unified Process*) y con la notación del lenguaje unificado de modelado para el modelo de requisitos, diagrama de clases y escenarios.

1.1 Motivación

En primer lugar remarcar que este proyecto ha sido realizado colaborando para la empresa SistelNetworks desde el instituto de telecomunicaciones y aplicaciones multimedia (iTEAM) y ha servido para aumentar mi experiencia en mundo laboral aportando mi conocimiento en estos años universitarios tanto de la ingeniería informática como del máster cursado, además de proporcionar la ayuda con el diseño y desarrollo de la interfaz gráfica y la base de datos del simulador, donde se destaca por encima de todo la funcionalidad, fiabilidad y eficiencia a la hora de la creación del simulador.

Por otro lado, este proyecto (SN4G Simulator) es una parte de la solución que proporciona SistelNetwoks para las operadores de telecomunicaciones, tanto para entornos de interior como de exterior, cubriendo las necesidades de comunicación inalámbrica de empresas y usuarios.

Al mismo tiempo, este sistema facilitara a las operadoras móviles la posibilidad de realizar búsquedas de la ubicación y el diseño de los parámetros óptimos de las distintas redes, de forma sencilla e intuitiva para el usuario.

Además, el desarrollo de este software comercial, me ha permitido utilizar herramientas y técnicas aprendidas de forma directa en el máster estudiado, por lo que tiene una motivación extra el poder utilizar de forma directa los conocimientos aprendidos, como ejemplo, tenemos la utilización de concurrencia a la hora del borrado de múltiples simulaciones mediante hilos en Java, o la gestión de las múltiples simulaciones mediantes hilos en lenguaje Perl, que permite aprovechar los múltiples núcleos del servidor para tener la máquina a su máximo potencial.

Finalmente, cabe destacar la redacción de la memoria, la cual es una gran motivación a la hora de poder combinar en un documento todo el duro trabajo de un año, tanto por parte de mi situación laboral, trabajando en un centro de investigación donde la innovación y el desarrollo están a la orden del día, como por parte del desarrollo de un máster de informática, altamente especializado en desarrollo de software, donde se muestra y se aprende a utilizar las herramientas punteras y sobre todo a saber como y cuando utilizarlas para crear software de calidad.

1.2 Propósito

El principal objetivo del proyecto es la creación de una interfaz gráfica intuitiva para un simulador comercial de redes que sea capaz de gestionar simulaciones de forma eficiente, lanzar ordenes al núcleo del simulador y poder visualizar los resultados mediante gráficos e imágenes a través de dicha interfaz. Esta solución forma parte de un proyecto más grande llamado "LTE Small Cell Solutions" de la empresa SistelNetworks que se encarga de proporcionar servicio 4G, cubriendo las necesidades de comunicación inalámbrica tanto de usuarios como de empresas.

Esta aplicación logra que cualquier usuario pueda realizar una gestión de las simulaciones de una forma rápida y sencilla desde su computador, indicar las peticiones de simulación y poder visualizar los resultados a posteriori. De esta manera podemos decir que hay tres objetivos bien diferenciados:

- Acceso eficiente: Presentar los datos de forma ordenada y optimizada, además de ser lo más interactivo posible para facilitar la creación y edición de las distintas simulaciones.
- Fácil instalación: Posibilidad de realizar la gestión de las simulaciones desde un computador, tan solo teniendo instalado la máquina virtual de Java y haciendo clic en la aplicación.
- FeedBack: Cuando haya finalizado la simulación desde el núcleo del simulador en el servidor este indica a través de una petición la finalización de las simulaciones para que el usuario pueda visualizar los distintos resultados obtenidos, donde se visualiza tanto los gráficos como las distintas imágenes.

La finalidad personal es el poder aportar mi granito de arena en el lugar de trabajo donde he estado obteniendo experiencia para complementar mis años universitarios y poder ayudar tecnológicamente y aportar una solución a parte de un gran proyecto de la empresa SistelNetworks.

Además, la posibilidad de tratar una temática directa en relación con el máster, ya que este proyecto tiene mucha relación con la ingeniería del software y los sistemas de información.

1.3 Estructura del proyecto

La memoria está organizada principalmente en la forma en la que se ha realizado el proyecto, es decir siguiendo una adaptación del proceso (RUP) el cual permite que sea un proceso de desarrollo fundamentalmente iterativo, además de gestionar una administración de requisitos necesarios para la realización de este.

El proceso RUP esta centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (productos tangibles del proceso), que iremos presentando en la memoria según el proceso en el que nos encontremos en cada momento.

Dicho esto, podemos dividir la memoria en 4 capítulos diferenciados por este proceso:

- Fase de inicio: En este capítulo se define y acota el alcance del proyecto con SistelNetworks, además se identifica los riesgos asociados al proyecto y se propone una visión muy general de la arquitectura software, es decir, de la interfaz Java, de su api web de Google Maps con ExtJS y de la gestión de la base de datos con el lenguaje MySQL.

- Fase de elaboración: En este capítulo se orienta al desarrollo de la arquitectura, abarcando mas los flujos de trabajos de requisitos, análisis y diseño.

- Fase de construcción: En este capítulo se describe la construcción de la aplicación y el desarrollo de la api web. Se seleccionan los casos de uso, se refina su diseño y análisis y se procede a su implementación y pruebas.

- Fase de transición: En este capítulo se asegura que el software este disponible para los usuarios finales, se acaban de ajustar errores encontrados en las pruebas de aceptación y se provee del soporte técnico necesario para el uso de la aplicación.

2 Fase de Inicio

En este capítulo se establece el ámbito del proyecto y sus límites, además de poner en manifiesto los casos de uso del sistema, y la muestra general de la arquitectura utilizada para su implementación.

También se indica unas pequeñas estimaciones de costes y tiempos del proyecto que intentaremos seguir durante toda la implementación, y por último se estimaran los posibles riesgos que se pueden dar en el proyecto.

Los productos o "artefactos" que incluiremos en esta fase de inicio del proceso de desarrollo RUP será el modelo de casos de uso, además de la visión del negocio, la cual describe los objetivos y restricciones a alto nivel.

2.1 El proyecto

En este apartado y los siguientes subapartados se describen cuestiones generales acerca del proyecto en curso. En primer lugar, se detalla el contexto del mismo, después se definen los requisitos y por último se marcan las decisiones tomadas en base a las necesidades.

2.1.1 Visión y Análisis del Negocio

Por un lado el instituto donde se ha llevado a cabo el desarrollo de la interfaz del simulador es el de Telecomunicaciones y Aplicaciones Multimedia (iTEAM) en la Universidad Politécnica de Valencia, el cual, esta integrado en la Ciudad Politécnica de la Innovación, el nuevo parque científico de la UPV, donde se desarrollan actividades de investigación, desarrollo e innovación (I+D+I) dentro del área de las tecnologías de la información y las comunicaciones.

iTEAM esta formado por 8 grupos de investigación que aglutinan a más de 100 investigadores, abarcando 5 áreas científicas relacionadas con la Ingeniería de Telecomunicaciones. iTEAM también desarrolla actividades de I+D+I en el Campus de la EPS de Gandía, con personal y laboratorios propios.

iTEAM participa en un gran número de proyectos competitivos de I+D+I en

los ámbitos Autonómico, Nacional y Europeo, a la vez que mantiene un estrecho vínculo y gran número de convenios y proyectos con empresas privadas.

Por otro lado la empresa para la que se ha realizado el proyecto es SistelNetworks, la cual se encarga de ofrecer productos de alto valor añadido y soluciones para la industria inalámbrica, para mejorar la experiencia del usuario y ayudar al crecimiento de la rentabilidad de los servicios.

Se trata de una empresa joven con un equipo de ingenieros cualificados, desarrolladores y expertos técnicos que se basan en ofrecer soluciones innovadoras para los mercados de LTE y NFC. Además, SistelNetworks tiene una amplia experiencia de trabajo con operadores de telecomunicaciones, proveedores de soluciones y distribuidores.

Actualmente, y con todos estos datos entre manos, es lógico pensar que la empresa SistelNetworks haya decido diseñar un producto que de solución a las operadoras de telecomunicaciones, el simulador se visualiza como una herramienta de apoyo a la hora de gestionar todos los posibles elementos de comunicación de una ciudad, de una oficina o del propio hogar de un usuario.

Por otro lado, en el mercado no se dispone de un sistema tan completo de simulación informatizado y automático para la realización de las gestiones de simulaciones de redes de comunicaciones. Una posible referencia de lo que se quiere obtener es la herramienta de simulación Atoll.

Por todo esto, y gracias a la adquisición de un servidor que facilitará las simulaciones de los usuarios mediante el envío y encolado de estas, se decidió como objetivo personal la realización del proyecto final de máster que suprimirá dicha carencia. El objetivo del proyecto es conseguir que abarque la simulación de las redes de comunicación 2G, 3G, 4G y WLAN, con lo que se consigue un simulador completo para abastecer a las operadoras de comunicaciones tanto en las nuevas como en las antiguas comunicaciones.

2.1.2 Modelado de casos de uso

En este diagrama se define una notación gráfica para representar los casos de uso, es decir, el comportamiento del sistema al afrontar una tarea de negocio o un requisito del negocio. Para realizar cualquiera de ellas debe haberse autenticado mediante el protocolo de autenticación CHAP (*Challenge-Handshake Authentication Protocol*) previamente.

Figura 1: Diagrama de descomposición de paquetes

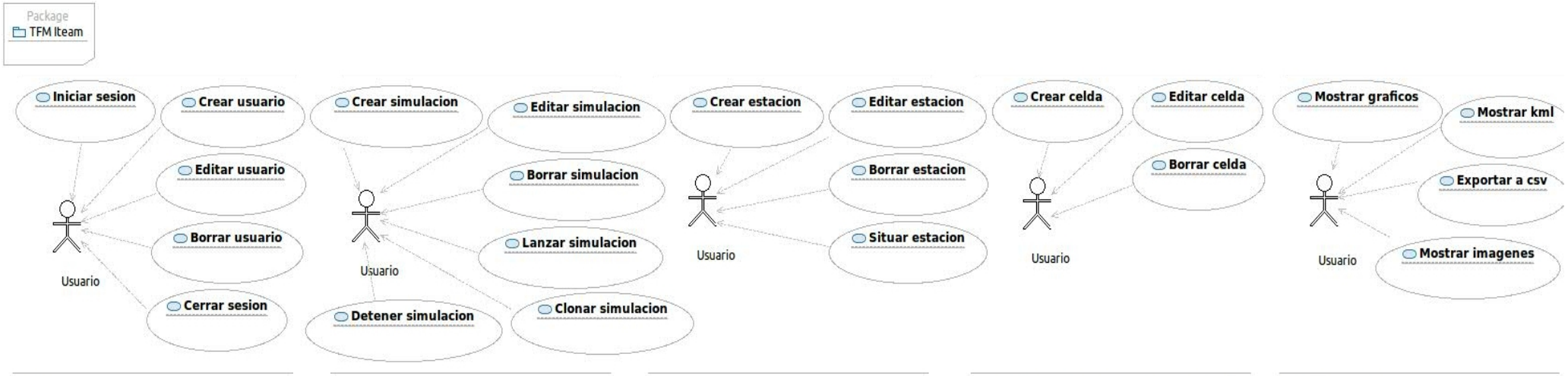

Figura 2: Diagrama de casos de uso

2.1.3 Especificaciones de requisitos

SistelNetworks no disponía de ninguna interfaz para el simulador definida, con lo cual se debe definir tanto las decisiones de la interfaz gráfica de la aplicación como los requisitos funcionales. La finalidad es que la aplicación sea lo más sencilla de utilizar posible para el personal operativo y que el salto a la nueva plataforma para un usuario sea lo más ventajosa e intuitiva posible. Los principales requisitos establecidos para realizar la aplicación son:

- Autenticación: El usuario deberá autenticarse con los datos definidos previamente, mediante un usuario y contraseña.

- Interfaz: Clara y sencilla para el usuario. Con el aspecto similar a una aplicación de escritorio Java.

- Seguridad: Tan solo un usuario con credenciales puede acceder a la aplicación.

- Ampliación: Código adaptable a futuras modificaciones o ampliaciones.

- Interacción con el servidor: Dado que esta diseñado para ser una aplicación de escritorio, dicha aplicación debe poder interactuar con el servidor, pudiendo conectarse a este para acceder a la base de datos y poder lanzar las simulaciones configuradas previamente.

- Gestión de errores: Tratar todos lo errores posibles con diálogos para que el usuario pueda interactuar sin ayuda con la aplicación de escritorio.

- Gestión de simulaciones: Posibilidad de configurar una simulación tanto real y sintética como externa e interna con las distintas posibilidades.

- Filtrado: Mostrar solo las simulaciones que correspondan al usuario en cuestión.

2.1.4 Especificaciones técnicas

La interfaz gráfica del simulador desarrollado cuenta con tres capas básicas para poder realizar las especificaciones de requisitos descritas en el apartado anterior.

En primer lugar, tenemos almacenados los datos de la organización en bases de datos almacenada en varios servidores disponibles en el servidor de SistelNetworks.

Se tiene diseñada en un servidor las diferentes tablas de las que se compone una simulación y las estaciones base que se pueden configurar dentro de esta simulación, además de todas las variables configurables todo ello con MySQL y finalmente se realiza la autenticación con las credenciales de los usuarios contra la base de datos del servidor MySQL donde tenemos el listado del personal accesible.

En segundo lugar, la aplicación cuenta con una conexión mediante SSH (Secure Shell) al servidor de la empresa, donde se conecta a la base de datos tras la autenticación segura.

Por otro lado, el usuario puede realizar gestiones con el núcleo de la aplicación, situada en el servidor de la empresas mediante la conexión SSH, por lo que, puede realizar las ordenes de lanzar una simulación, detenerla...

Finalmente, la aplicación trata la información recibida por la base de datos y los muestra por pantalla para que el usuario autenticado pueda gestionar las simulaciones correspondientes a su usuario, además de ofrecer la posibilidad de la visualización de gráficos almacenados en la base de datos e imágenes creados por el núcleo del simulador.

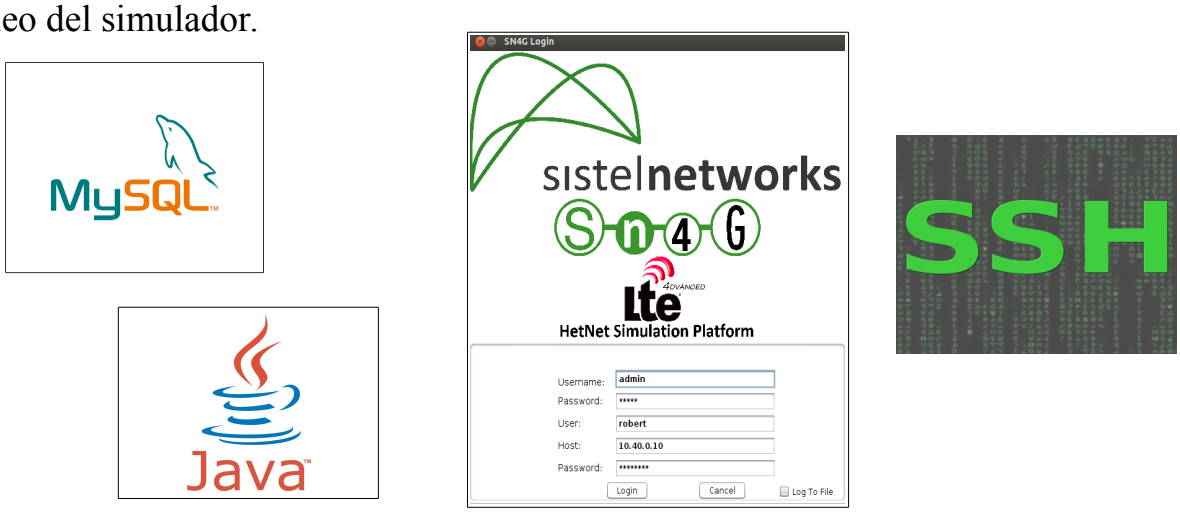

Figura 3: Especificaciones técnicas (Java, MySQL, ExtJS, SSH)

2.1.4.1 Autenticación

SistelNetworks tiene un servidor propio donde se tiene una base de datos en la cual se almacenan los datos de los distintos usuarios y sus credenciales, lo cual permite proporcionar direcciones personalizadas a todos los usuarios. Además es accesible desde Internet, por lo que se puede acceder a el desde cualquier lugar usando el cliente apropiado. Teniendo en cuenta esto, nuestra aplicación se autenticara contra dicho servidor, mediante usuario y contraseña protegida en MD5, por tanto, cualquier usuario operativo poseedor de un usuario y contraseña en el sistema podrá acceder a la aplicación.

2.1.4.2 Base de datos MySQL

SistelNetworks utiliza este sistema gestor de datos MySQL para almacenar mediante tablas entidad-relación las distintas variables de las simulaciones, además de todas las estaciones base de cada simulación. Por otro lado, almacena los distintos usuarios y sus credenciales que pueden utilizar la aplicación, y finalmente, se encarga de guardar los pid de todos las simulaciones que se han ejecutado o se están ejecutando, para que el usuario tenga control sobre estas y pueda detenerlas, matando todas las simulaciones asociadas.

2.1.4.3 Conexión SSH

Debido a que la aplicación se ha diseñado para aplicaciones de escritorio y el servidor esta situado en la empresa SistelNetworks se ha decidido utilizar una conexión SSH (Secure Shell), que es un protocolo y programa que lo implementa y que sirve para acceder a máquinas remotas a través de una red. SSH utiliza técnicas de cifrado que hacen que la información que viaja por el medio de comunicación vaya de manera no legible, evitando que terceras personas puedan descubrir el usuario y contraseña de la conexión ni lo que se escribe durante toda la sesión.

2.1.4.4 Tecnología ExtJS

Para situar las celdas en un escenario real externo se ha decantado por la tecnología ExtJS junto a su api de Google Maps, basada en javascript, dada su versatilidad y la capacidad de la integración de todos los elementos necesarios para dar solución a las necesidades de gestión de esta parte.

ExtJS es una librería Javascript que permite construir aplicaciones complejas en Internet. Esta librería incluye:

- Componentes UI del alto performance y personalizables.
- Modelo de componentes extensibles.
- Un API fácil de usar.
- Licencias Open source y comerciales.

Antes de poder entrar a examinar ExtJS primero tenemos que hablar sobre RIA (Aplicaciones Ricas en Internet). Lo que RIA intenta proveer es una experiencia de usuario muy parecida o igual a la que se tiene en las aplicaciones de escritorio.

Las aplicaciones web tradicionales tienen problemas como la recarga continua de las páginas cada vez que el usuario pide nuevo contenido, o la poca capacidad multimedia, para lo cual se han hecho necesarios plug-ins externos.

Junto con el reto de llevar la experiencia RIA a los usuarios comenzó el debate sobre cual sería el mejor modo de atacar el problema. La historia de los últimos años nos ha traído diversas tecnologías, basadas en Flash (Adobe), Java (Sun), Silverlight (MS). Todas muy interesantes, pero con la desventaja de necesitar algún tipo de extensión en los navegadores que podría no estar presente. Ha sido esta limitante lo que le ha dado la victoria (al menos por el momento) al casi dejado de lado Javascript y la “nueva” tecnología conocida como AJAX.

ExtJS encaja dentro de este esquema como un motor que permite crear aplicaciones RIA mediante Javascript. Si enmarcamos a ExtJS dentro del desarrollo RIA, éste seria el render de la aplicación que controla el cliente y que ese encarga de enviar y obtener información del servicio.

Una de las grandes ventajas de utilizar ExtJS es que nos permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts similar al que provee Java Swing, gracias a esto provee una experiencia consistente sobre cualquier navegador.

Además la ventana flotante que provee ExtJS es excelente por la forma en la que funciona. Al moverla o redimensionarla solo se dibujan los bordes haciendo que el movimiento sea fluido lo cual le da una ventaja tremenda frente a otros.

Usar un motor de render como ExtJS nos permite tener además estos beneficios:

- Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- Comunicación asíncrona. En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se de cuenta.

2.1.4.5 Tecnología Java

Se utiliza la plataforma Java y su librería gráfica Swing para el desarrollo principal de la aplicación gráfica del simulador. Java es un lenguaje de programación orientado a objetos, potente, robusta, segura y, además, una y universal plataforma de programación con una amplia y diversa colección de librerías integradas entre sí que permiten desarrollar aplicaciones informáticas actuales de un elevado grado de complejidad y funcionalidad.

Entre las características del lenguaje de programación Java destacan las siguientes:

- Es un lenguaje orientado a objetos de una baja complejidad.
- Es interpretado: se compila y seguidamente se ejecuta sobre una máquina virtual.
- Las aplicaciones en él desarrolladas son portátiles entre múltiples plataformas.
- Las aplicaciones son robustas ya que la JVM maneja la memoria.
- Permite multihilos, mejorando así las prestaciones y funcionalidad de determinadas aplicaciones.
- Las aplicaciones son adaptables en entornos cambiantes porque dinámicamente pueden descargar software de la red.
- Las aplicaciones son seguras por varios motivos, entre ellos están que la JVM protege contra los virus y aplicaciones hostiles.
- Permite el diseño y desarrollo de sistemas con arquitecturas complejas de programas.

2.1.5 Riesgos asociados

En general para un proyecto informático los riesgos asociados son que el software nunca llegue a funcionar, que no se cumplan los plazos de entrega y que no se cumplan con las funcionalidades esperadas principalmente causado por la alta complejidad o la incertidumbre de cara al comienzo de un proyecto.

Para evitar que estos riesgos se traduzcan en problemas, trabajaremos de forma proactiva, tratando de identificar los problemas potenciales y atajándolos. En nuestro caso, el software es testeado en cada iteración realizada, según el proceso de desarrollo RUP, por tanto evitamos el riesgo de que nuestro software nunca llegue a funcionar.

Por otro lado, para evitar el incumplimiento de plazos hemos realizado un diagrama de Grantt para cumplir cada tarea en la fecha indicada previamente.

Por último, realizamos reuniones con los empleados de la empresa SistelNetworks, el cliente del proyecto, en cada iteración para tratar de adaptar el producto lo máximo posible a sus requisitos.

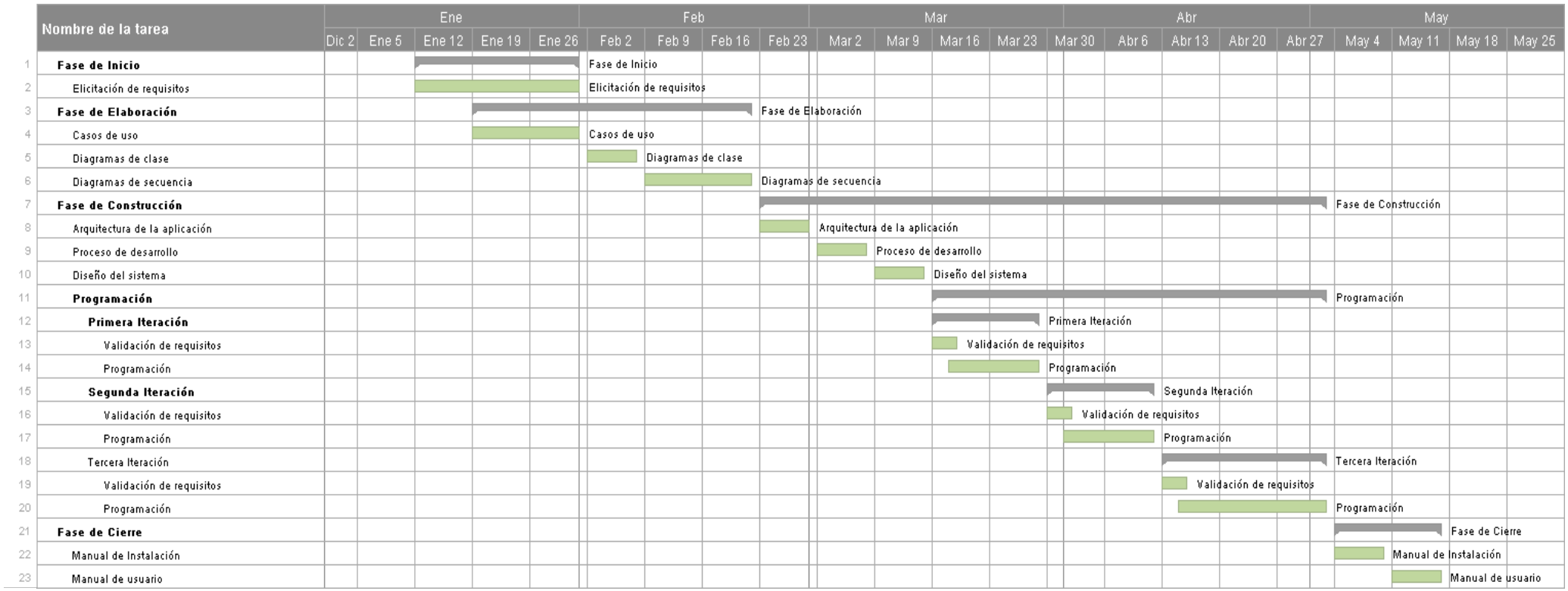

Figura 4: Diagrama de Gantt

3 Fase de elaboración

En este capítulo se analiza el dominio del problema, se establecen los cimientos de la arquitectura, se desarrolla el plan del proyecto y se eliminan los mayores riesgos.

A partir de esta fase se llega al punto de no retorno del proyecto, por tanto esta es la fase en la que se debe indicar si el proyecto es viable para su próximas fases, que serán mucho más costosas y arriesgadas.

La fase de elaboración debe contener los casos de uso críticos identificados en la fase de inicio y demostrarse que se han evitado los riesgos más graves.

3.1 Casos de uso

En este apartado se analiza cada uno de los casos de uso que se presentan en el diagrama descrito en el capítulo anterior. Con esto describiremos de forma breve cada uno de los casos del diagrama, de esta forma obtendremos una visión más completa del proyecto.

Caso de uso: Iniciar Sesión

Actor primario: Usuario.

Actores secundarios: -

Precondicones: -

Postcondicones: El personal operativo queda autenticado en el sistema.

Descripción (Flujo Básico):

1) Indica su nombre de usuario y contraseña.
2) El sistema comprueba su validez comparando las credenciales con las de la base de datos del servidor.
3) El sistema muestra la página principal de la aplicación.

Extensiones (Flujo Alternativo):

2.a) Indica su nombre de usuario y contraseña. El nombre de usuario o la contraseña es incorrecto.

2.a.1) El sistema notifica que el nombre de usuario o la contraseña es incorrecto y lo pide de nuevo.

Caso de uso: Cerrar Sesión

Actor primario: Usuario.

Actores secundarios: -

Precondicones: El usuario se ha autenticado en el sistema.

Postcondicones: El usuario cierra sesión en el sistema.

Descripción (Flujo Básico):

1) El usuario cierra sesión.
2) El sistema sale de la sesión y se muestra de nuevo la autenticación.

Extensiones (Flujo Alternativo): -

Caso de uso: Crear usuario

Actor primario: Usuario.

Actores secundarios: -

Precondicones: El usuario tiene permisos de administrador y se ha autenticado.

Postcondicones: El usuario crea un usuario nuevo.

Descripción (Flujo Básico):

1) El usuario selecciona Users.
2) El sistema muestra los usuarios.
3) El usuario pulsa New.
4) El sistema muestra los datos de usuario.
5) El usuario añade los datos necesarios para rellenar el usuario.
6) El usuario selecciona Save.
7) El sistema valida las entradas del usuario .
8) El sistema añade el usuario en la base de datos.

Extensiones (Flujo Alternativo):

1.a) No hay conexión.

1.a.1) El sistema notifica que no se puede autenticar.

Caso de uso: Editar Usuario

Actor primario: Usuario.

Actores secundarios: -

Precondicones: El usuario tiene permisos de administrador y se ha autenticado.

Postcondicones: El usuario ha editado al usuario correspondiente.

Descripción (Flujo Básico):

1) El usuario selecciona Users.
2) El sistema muestra los usuarios.
3) El usuario elige a un usuario.
4) El usuario hace doble clic para editar dicho usuario.
5) El sistema muestra los datos del usuario.
6) El usuario modifica los datos necesarios.
7) El usuario selecciona confirmar el usuario.
8) El sistema valida los datos del usuario.
9) El sistema edita los datos en la base de datos.

Extensiones (Flujo Alternativo):

6.a) El usuario selecciona cancelar.

6.a.1) El sistema sale de la edición de usuarios sin editar ningún campo.

7.a) Alguna entrada ha sido introducida incorrectamente.

7.a.1) El sistema notifica que la entrada esta vacía o no es correcta.

8.a) No hay conexión.

8.a.1) El sistema notifica que no se puede editar el usuario correspondiente.

Caso de uso: Borrar Usuario

Actor primario: Usuario.

Actores secundarios: -

Precondicones: El usuario tiene permisos de administrador y se ha autenticado.

Postcondicones: El usuario borra el usuario seleccionado.

Descripción (Flujo Básico):

1) El usuario selecciona Users.
2) El sistema muestra los usuarios.
3) El usuario elige a el usuario a borrar.
4) El usuario selecciona borrar el usuario.
5) El sistema muestra la confirmación de borrado.
6) El usuario confirma el borrado.
7) El sistema borra el usuario en la base de datos.

Extensiones (Flujo Alternativo):

5.a) El usuario cancela el borrado.

5.a.1) El sistema sale del borrado sin borrar a el usuario.

6.a) No hay conexión.

6.a.1) El sistema notifica que no se puede borrar el usuario.

Caso de uso: Crear simulación

Actor primario: Usuario.

Actores secundarios: -

Precondicones: El usuario se ha autenticado.

Postcondicones: El usuario crea la simulación.

Descripción (Flujo Básico):

1) El usuario selecciona Synthetic Scenario.
2) El sistema muestra las simulaciones.
3) El usuario selecciona la simulación que desea crear.

4) El usuario añade los datos necesarios para configurar la simulación.
5) El usuario selecciona confirmar.
6) El sistema valida las entradas del usuario.
7) El sistema añade la simulación en la base de datos.

Extensiones (Flujo Alternativo):

1.a) El usuario selecciona Synthetic With Pico Cells.

1.b) El usuario selecciona Real Outdoor Scenario.

1.c) El usuario selecciona Real Indoor Scenario.

6.a) El usuario selecciona cancelar.

6.a.1) El sistema sale del registro sin añadir ninguna simulación.

7.a) No hay conexión.

7.a.1) El sistema notifica que no se puede añadir la simulación correspondiente.

Caso de uso: Editar Simulación

Actor primario: Usuario.

Actores secundarios: -

Precondicones: El usuario se ha autenticado.

Postcondicones: El usuario edita la simulación.

Descripción (Flujo Básico):

1) El usuario selecciona Synthetic Scenario.
2) El sistema muestra las simulaciones.
3) El usuario selecciona editar la simulación.
4) El sistema muestra los datos de edición correspondientes.
5) El usuario modifica los datos necesarios para la simulación.
6) El usuario selecciona confirmar la simulación.
7) El sistema valida las entradas del usuario.
8) El sistema edita los datos la base de datos.

Extensiones (Flujo Alternativo):

1.a) El usuario selecciona Synthetic With Pico Cells.

1.b) El usuario selecciona Real Outdoor Scenario.

1.c) El usuario selecciona Real Indoor Scenario.

6.a) El usuario selecciona cancelar.

6.a.1) El sistema sale de la edición de simulaciones sin editar ningún campo.

7.a) Alguna entrada ha sido introducida incorrectamente.

7.a.1) El sistema notifica que la entrada esta vacía o no es correcta.

8.a) No hay conexión.

8.a.1) El sistema notifica que no se puede editar la simulación correspondiente.

Caso de uso: Borrar Simulación

Actor primario: Usuario.

Actores secundarios: -

Precondicones: El usuario se ha autenticado.

Postcondicones: El usuario borra la simulación.

Descripción (Flujo Básico):

1) El usuario selecciona Synthetic Scenario.
2) El sistema muestra las simulaciones.
3) El usuario elige la simulación a borrar.
4) El usuario selecciona borrar la simulación.
5) El sistema muestra el dialogo de confirmación de borrado.
6) El usuario confirma el borrado.
7) El sistema borra la simulación de la base de datos.

Extensiones (Flujo Alternativo):

1.a) El usuario selecciona Synthetic With Pico Cells.

1.b) El usuario selecciona Real Outdoor Scenario.

1.c) El usuario selecciona Real Indoor Scenario.

5.a) El usuario cancela el borrado.

5.a.1) El sistema sale del borrado sin borrar la simulación.

6.a) No hay conexión.

6.a.1) El sistema notifica que no se puede borrar la simulación.

Caso de uso: Lanzar Simulación

Actor primario: Usuario.

Actores secundarios: -

Precondicones: El usuario se ha autenticado.

Postcondicones: El usuario lanza la simulación.

Descripción (Flujo Básico):

1) El usuario selecciona Synthetic Scenario.
2) El sistema muestra las simulaciones.
3) El usuario elige la simulación a lanzar.
4) El usuario selecciona lanzar la simulación.
5) El sistema muestra la confirmación de lanzado.
6) El usuario confirma el lanzado.
7) El sistema lanza la simulación en el servidor.

Extensiones (Flujo Alternativo):

1.a) El usuario selecciona Synthetic With Pico Cells.

1.b) El usuario selecciona Real Outdoor Scenario.

1.c) El usuario selecciona Real Indoor Scenario.

5.a) El usuario cancela el lanzado.

5.a.1) El sistema sale del lanzado sin lanzar la simulación.

6.a) No hay conexión.

6.a.1) El sistema notifica que no se puede lanzar la simulación.

Caso de uso: Clonar Simulación

Actor primario: Usuario.

Actores secundarios: -

Precondicones: El usuario se ha autenticado.

Postcondicones: El usuario clona la simulación.

Descripción (Flujo Básico):

1) El usuario selecciona Synthetic Scenario.
2) El sistema muestra las simulaciones.
3) El usuario elige la simulación a clonar.
4) El usuario selecciona clonar la simulación.
5) El sistema muestra la confirmación de clonado.
6) El usuario confirma el clonado.
7) El sistema clona la simulación en la base de datos.

Extensiones (Flujo Alternativo):

1.a) El usuario selecciona Synthetic With Pico Cells.

1.b) El usuario selecciona Real Outdoor Scenario.

1.c) El usuario selecciona Real Indoor Scenario.

5.a) El usuario cancela el clonado.

5.a.1) El sistema sale del clonado sin clonar la simulación.

6.a) No hay conexión.

6.a.1) El sistema notifica que no se puede clonar la simulación.

Caso de uso: Detener Simulación

Actor primario: Usuario.

Actores secundarios: -

Precondicones: El usuario se ha autenticado.

Postcondicones: El usuario detiene la simulación.

Descripción (Flujo Básico):

1) El usuario selecciona Synthetic Scenario.
2) El sistema muestra las simulaciones.
3) El usuario elige la simulación a detener.
4) El usuario selecciona detener la simulación.
5) El sistema muestra la confirmación de detención.
6) El usuario confirma la detención.

7) El sistema detiene la simulación de la base de datos.

Extensiones (Flujo Alternativo):

1.a) El usuario selecciona Synthetic With Pico Cells.

1.b) El usuario selecciona Real Outdoor Scenario.

1.c) El usuario selecciona Real Indoor Scenario.

5.a) El usuario cancela la detención.

5.a.1) El sistema sale de la detención sin detener la simulación.

6.a) No hay conexión.

6.a.1) El sistema notifica que no se puede detener la simulación.

A continuación, se tienen los casos de uso tanto de la gestión de las estaciones base como los casos de uso de las diferentes celdas, en estos casos no se va a describir mediante la plantilla descrita anteriormente, porque son bastante similares a los casos de uso de la gestión de las simulaciones, aunque existe una pequeña diferencia y es que previo a la gestión de estaciones base debe haberse creado una simulación, entonces se indica sobre que simulación se quiere gestionar dichas estaciones base, se puede decir, que para una simulación pueden existir muchas estaciones base.

Algo similar ocurre en la gestión de celdas, aunque en este caso, además de existir una simulación, se deben gestionar las celdas dentro de las estaciones base, es decir, para una estación base pueden existir o se pueden crear múltiples celdas, todo ello siempre dentro de la ventana de una estación base, donde esta desarrollada toda la parte de gestión de celdas.

Finalmente, y dada la gran diferencia entre los demás casos de uso, si que se va a detallar toda la parte de gestión de resultados, que consta de cuatro casos de uso, como se puede ver en el diagrama y que se detallaran sus casos de uso a continuación:

- Mostrar gráficos.
- Mostrar kml.
- Exportar a CSV.

- Mostrar imágenes.

Caso de uso: Mostrar Gráficos

Actor primario: Usuario.

Actores secundarios: -

Precondicones: El usuario se ha autenticado y las simulaciones han terminado.

Postcondicones: El sistema muestra los gráficos

Descripción (Flujo Básico):

1) El usuario selecciona Synthetic Scenario.
2) El sistema muestra las simulaciones.
3) El usuario elige las simulaciones.
4) El usuario selecciona mostrar gráficos.
5) El sistema realiza los cálculos de los datos en cada simulación.
6) El sistema muestra los distintos gráficos con los resultados obtenidos.

Extensiones (Flujo Alternativo):

1.a) El usuario selecciona Synthetic With Pico Cells.

1.b) El usuario selecciona Real Outdoor Scenario.

1.c) El usuario selecciona Real Indoor Scenario.

4.a) No hay conexión.

4.a.1) El sistema notifica que no se puede mostrar los gráficos de la simulación.

Caso de uso: Mostrar Kml

Actor primario: Usuario.

Actores secundarios: -

Precondicones: El usuario se ha autenticado y la simulación ha terminado.

Postcondicones: El sistema muestra el kml de la simulación.

Descripción (Flujo Básico):

1) El usuario selecciona Real Outdoor Scenario.
2) El sistema muestra las simulaciones.
3) El usuario elige una simulación.
4) El usuario selecciona mostrar kml.
5) El sistema realiza la conexión al servidor y lanza un navegador.
6) El sistema muestra el kml superpuesto en un mapa.

Extensiones (Flujo Alternativo):

4.a) No hay conexión.

4.a.1) El sistema notifica que no se puede mostrar el kml de la simulación.

Caso de uso: Exportar a CSV

Actor primario: Usuario.

Actores secundarios: -

Precondicones: El usuario se ha autenticado y las simulaciones han terminado.

Postcondicones: El sistema muestra los resultados en formato CSV.

Descripción (Flujo Básico):

1) El usuario selecciona Synthetic Scenario.
2) El sistema muestra las simulaciones.
3) El usuario elige las simulaciones.
4) El usuario selecciona exportar a CSV.
5) El sistema realiza los cálculos de los datos en cada simulación.
6) El sistema exporta los distintos resultado a un fichero con formato CSV.

Extensiones (Flujo Alternativo):

1.a) El usuario selecciona Synthetic With Pico Cells.

1.b) El usuario selecciona Real Outdoor Scenario.

1.c) El usuario selecciona Real Indoor Scenario.

4.a) No hay conexión.

4.a.1) El sistema notifica que no se pueden exportar los resultados.

Caso de uso: Mostrar Imágenes

Actor primario: Usuario.

Actores secundarios: -

Precondicones: El usuario se ha autenticado y las simulaciones han terminado.

Postcondicones: El sistema muestra las imágenes de la simulación.

Descripción (Flujo Básico):

1) El usuario selecciona Synthetic Scenario.
2) El sistema muestra las simulaciones.
3) El usuario elige una simulación.
4) El usuario selecciona mostrar imágenes.
5) El sistema realiza la conexión al servidor y lanza las imágenes asociadas.
6) El sistema muestra las imágenes de cada simulación.

Extensiones (Flujo Alternativo):

1.a) El usuario selecciona Synthetic With Pico Cells.

1.b) El usuario selecciona Real Indoor Scenario.

4.a) No hay conexión.

4.a.1) El sistema notifica que no se puede mostrar el kml de la simulación.

3.2 Diagrama de clases

Una vez definidos los casos de uso, se procede a diseñar las características mediante diagramas UML. En primer lugar, dado que se trata de una aplicación de tratamiento de datos principalmente, lo primero es definir el modelo de datos, es decir las distintas clases de objetos que se necesitan para implementar todas las características que se han tratado en los casos de uso. El diagrama de clases no sólo se realiza en base a las especificaciones dadas por la aplicación sino también se ha orientado en torno a la base de datos donde se almacena toda la información. En este caso concreto, dada la enorme cantidad de atributos, se ha diseñado una versión reducida de diagrama de clases, con las clases principales y algunos de los atributos más importantes.

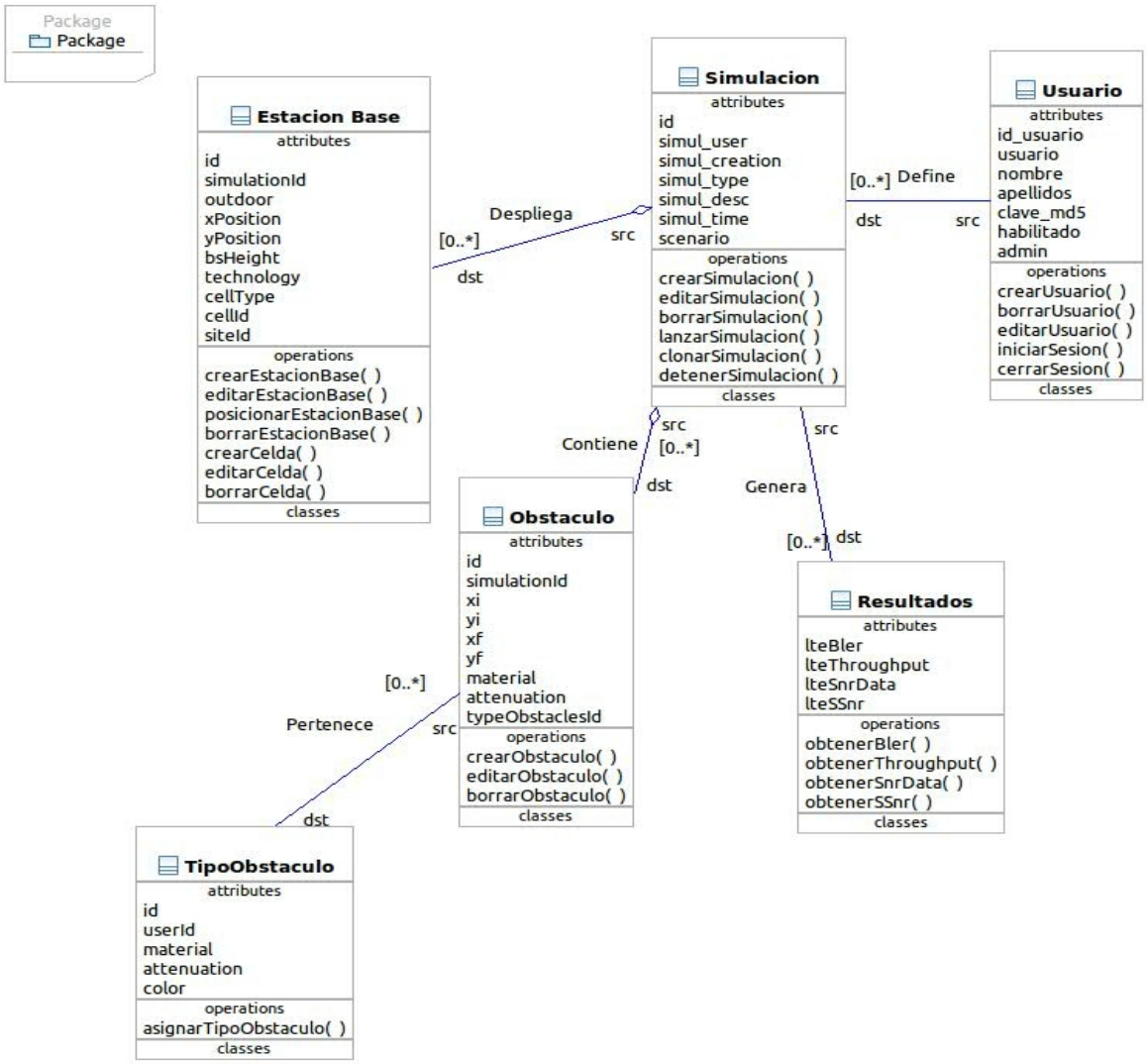

Figura 5: Diagrama de clases

3.3 Diagramas de secuencias

El siguiente paso ha sido la elaboración de los diagramas de secuencias para mostrar la interacción de un conjunto de objetos en una aplicación a través del tiempo modelándose para cada caso de uso. Este contiene detalles de la implementación del escenario, incluyendo objetos y clases que se usan para implementar el escenario, y mensajes intercambiados entre los objetos. Dado que ya tenemos las descripción de los casos de uso como una secuencia de varios pasos, entonces se puede pasar sobre esos pasos para obtener que objetos son necesarios para que se pueda construir dicho diagrama de secuencias.

Para diseñar los diagramas de secuencias se ha utilizado un programa llamado "Quick Sequence Diagram Editor" que consiste en una herramienta de desarrollo construida en Java 5 para generar UML de forma profesional con diagramas de secuencias mediante sencillas lineas de código.

Sus principales ventajas para utilizar esta herramienta, han sido principalmente que puede ser exportado a una imagen para introducirlo de forma sencilla en la memoria del proyecto y que cambia de forma automática mediante la introducción del código .

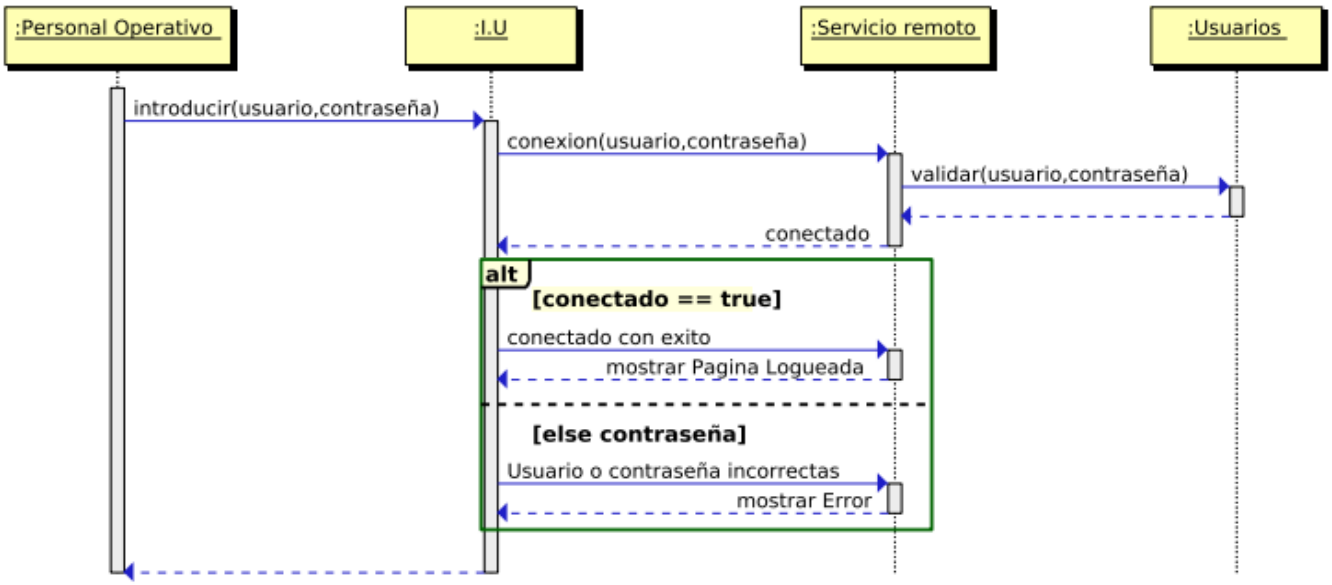

Figura 6: Iniciar sesión

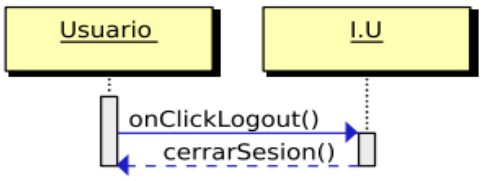

Figura 7: Cerrar sesión

Figura 8: Crear Usuario

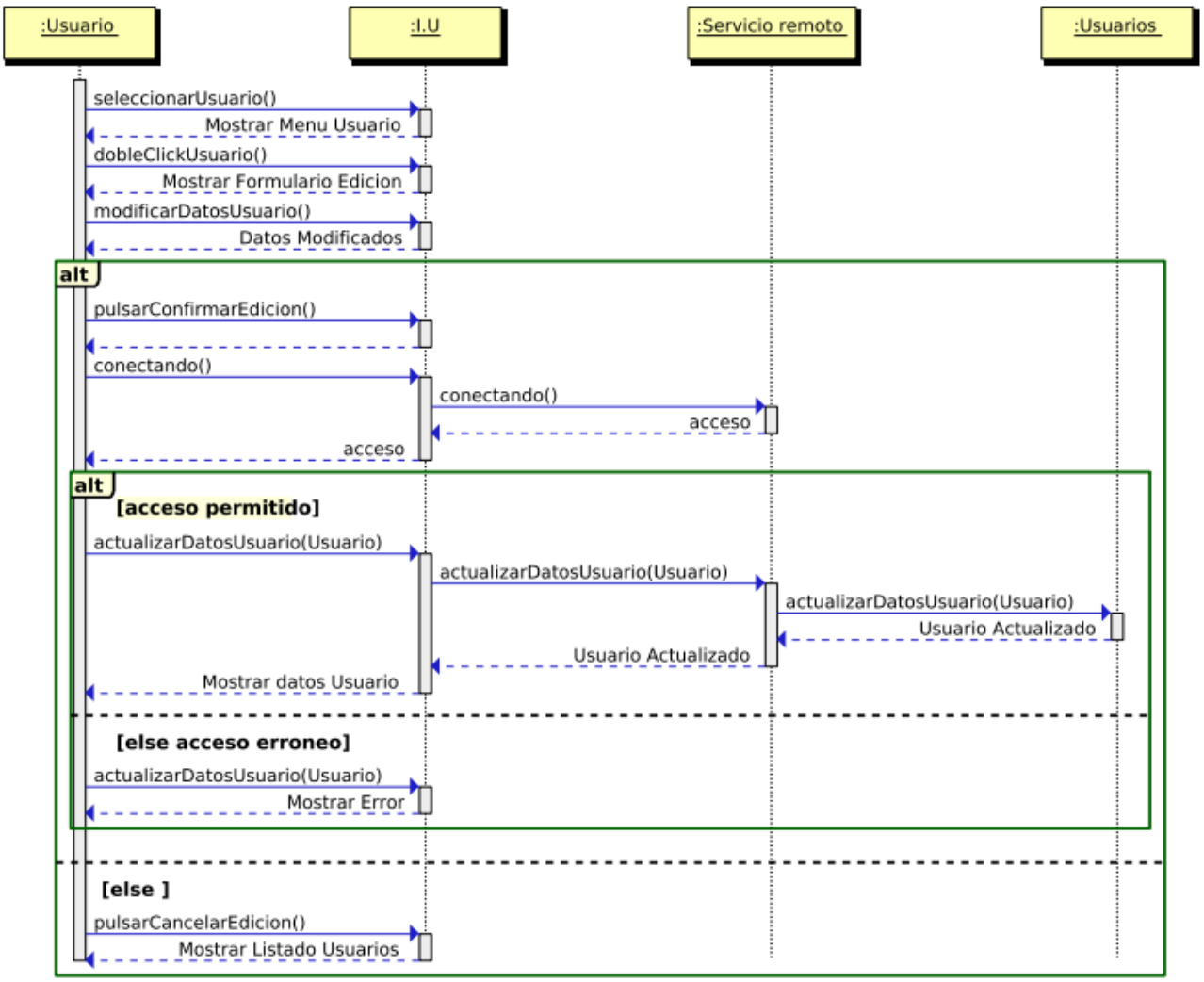

Figura 9: Editar Usuario

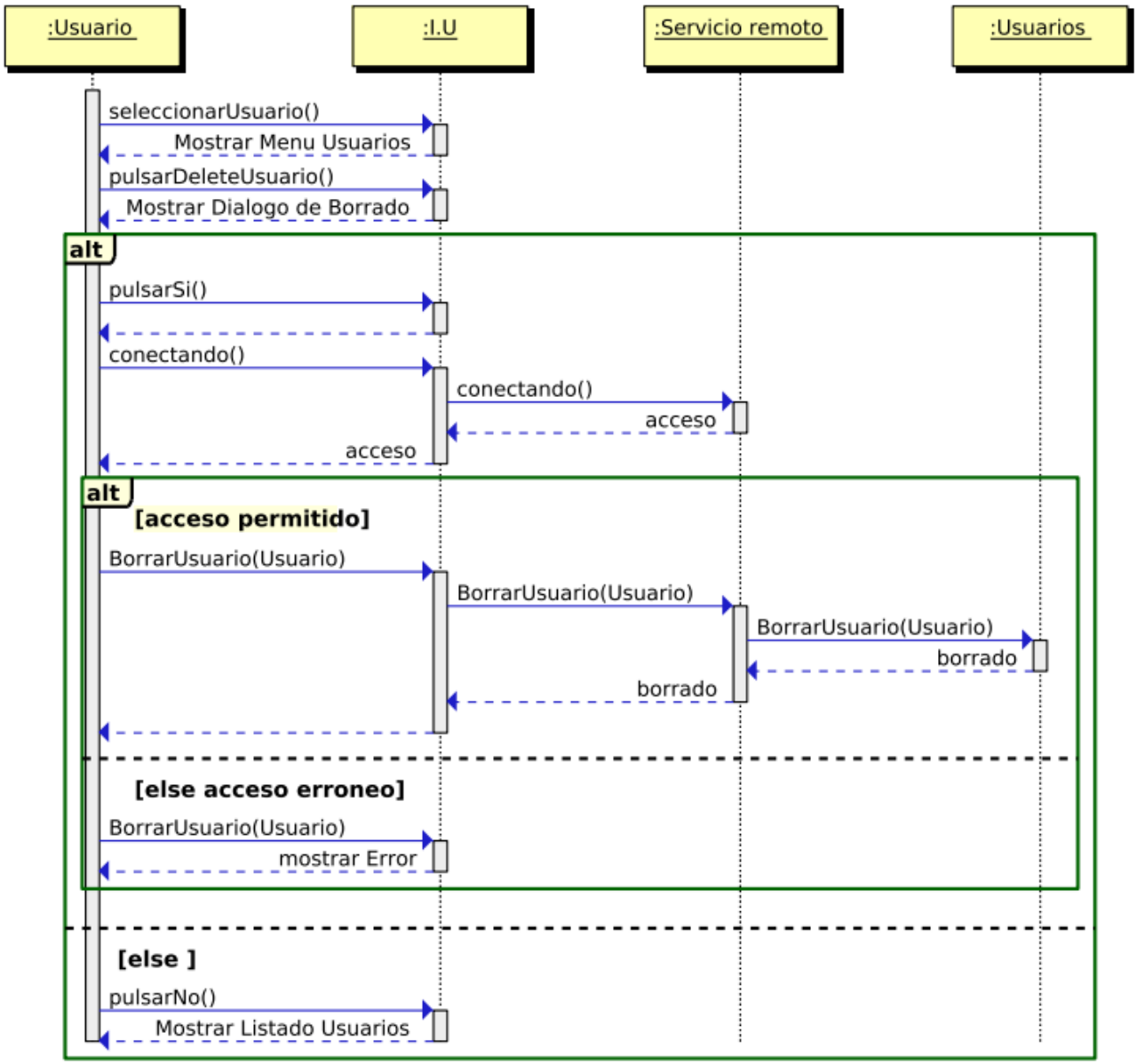

Figura 10: Borrar Usuario

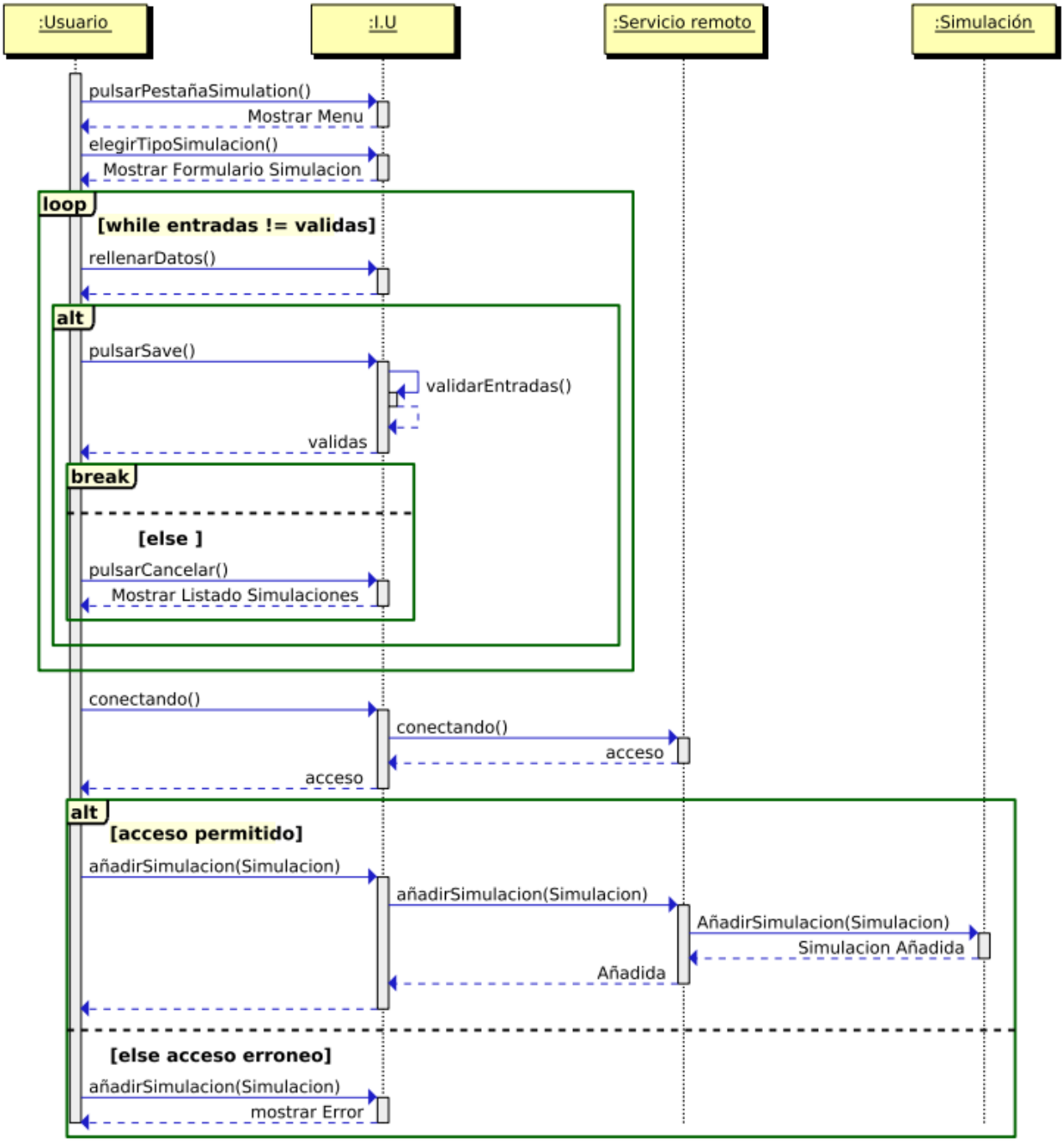

Figura 11: Crear Simulación

Figura 12: Editar Simulación

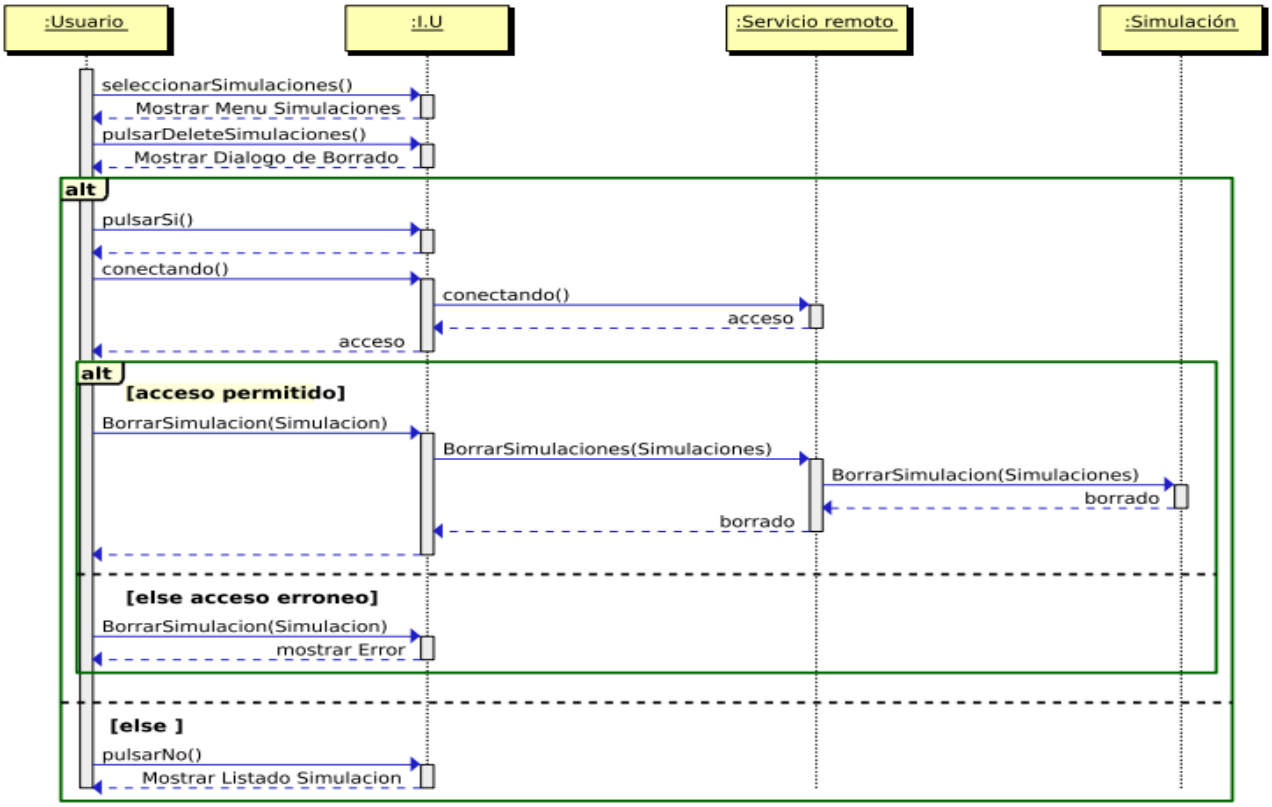

Figura 13: Borrar Simulaciones

Figura 14: Lanzar Simulaciones

Figura 15: Clonar Simulaciones

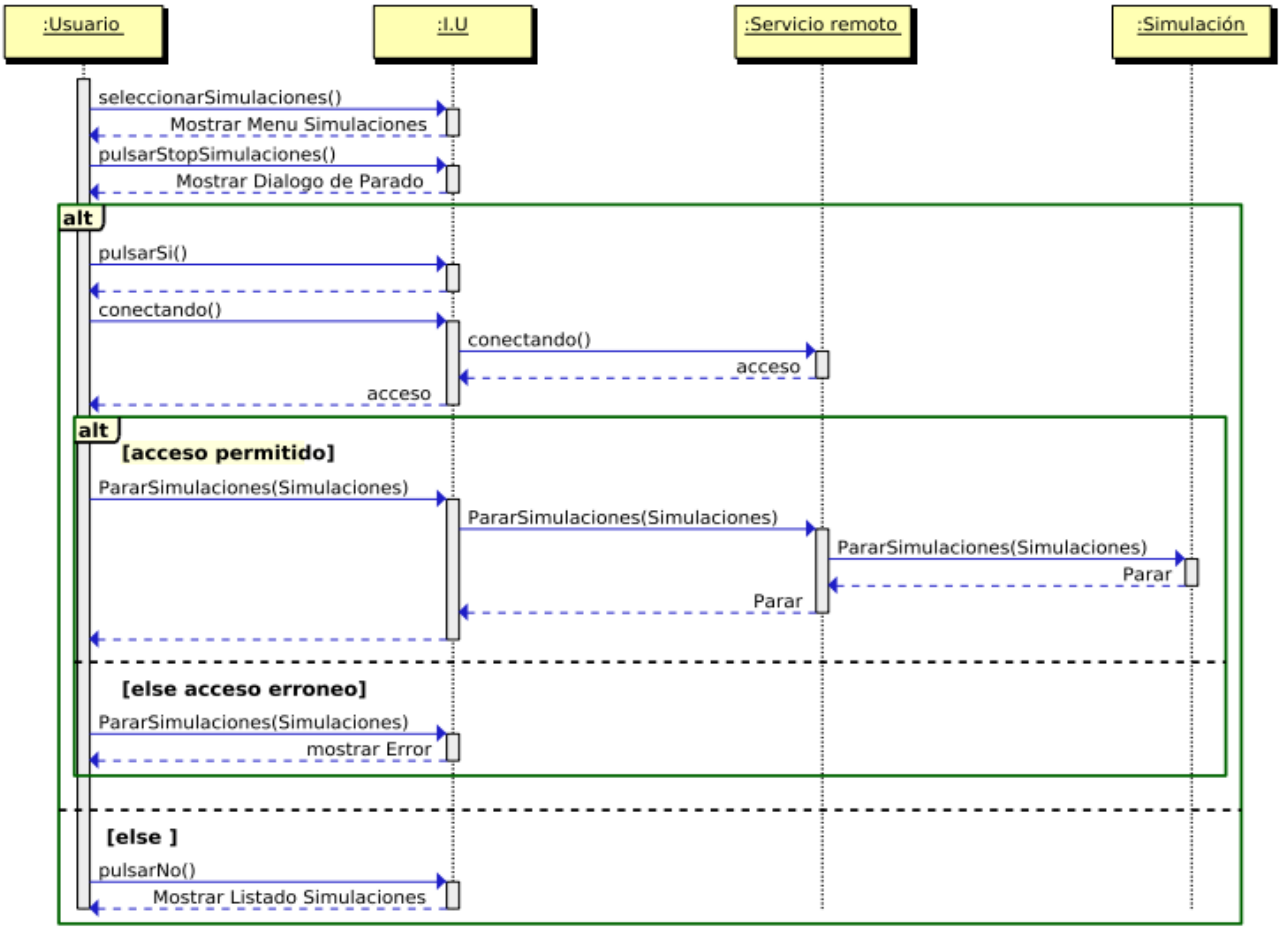

Figura 16: Detener Simulaciones

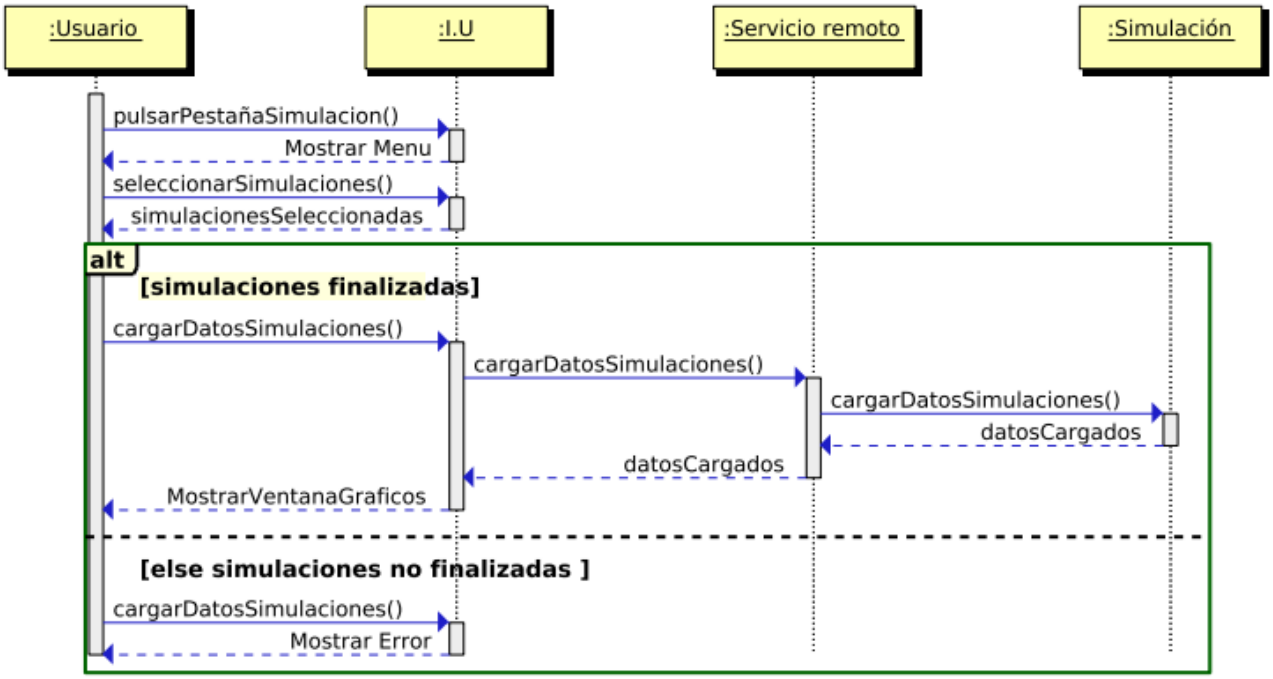

Figura 17: Mostrar Gráficos

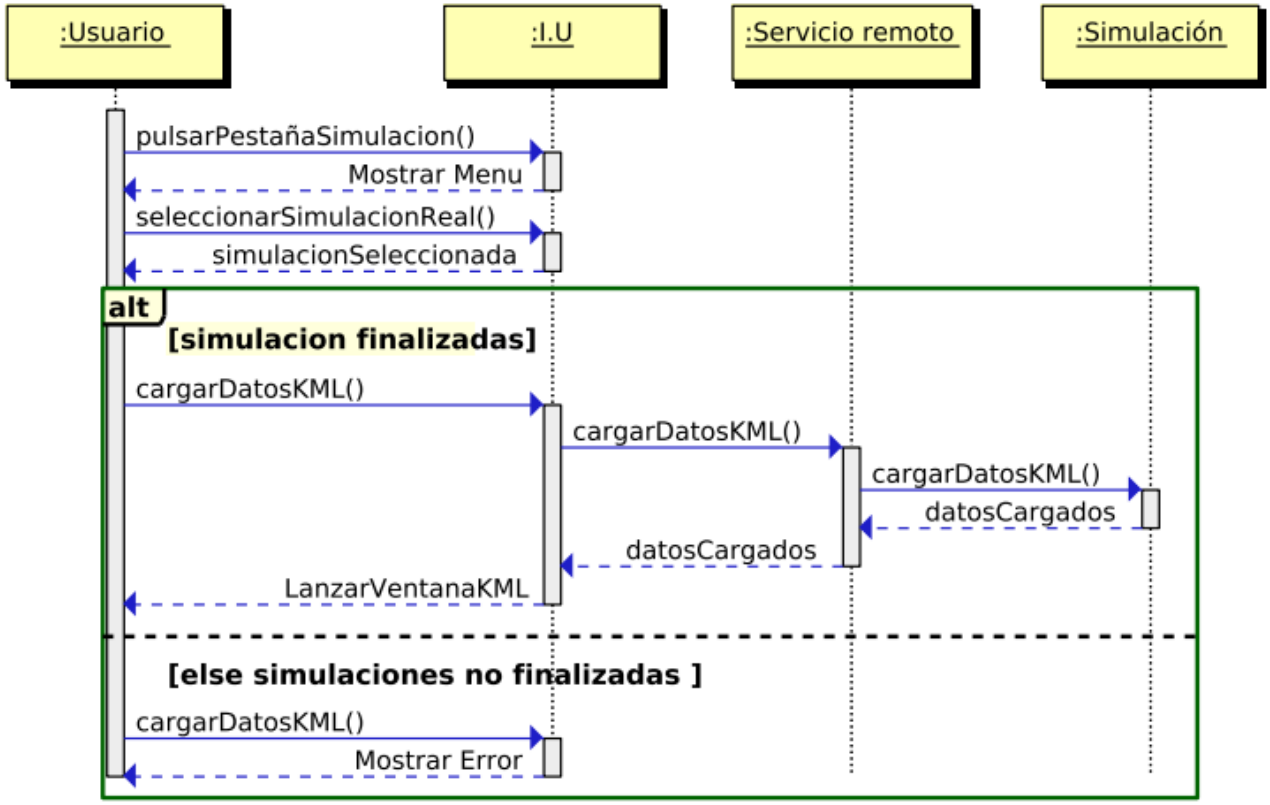

Figura 18: Mostrar KML

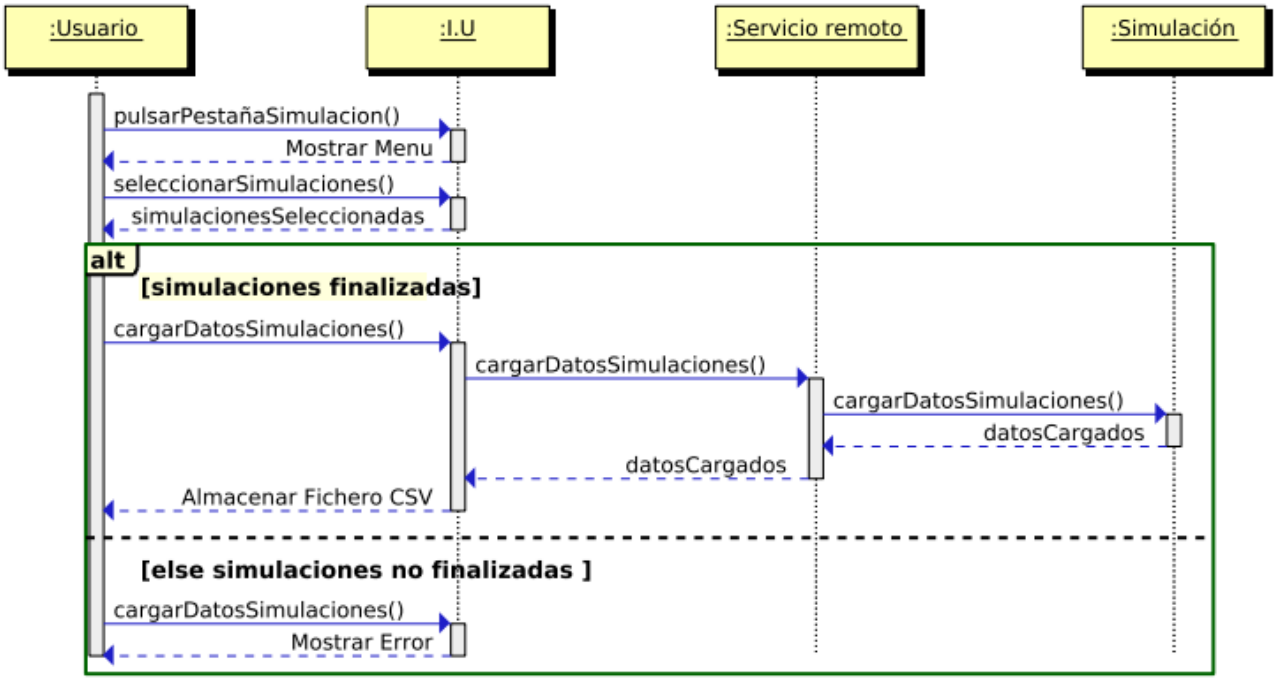

Figura 19: Exportar a CSV

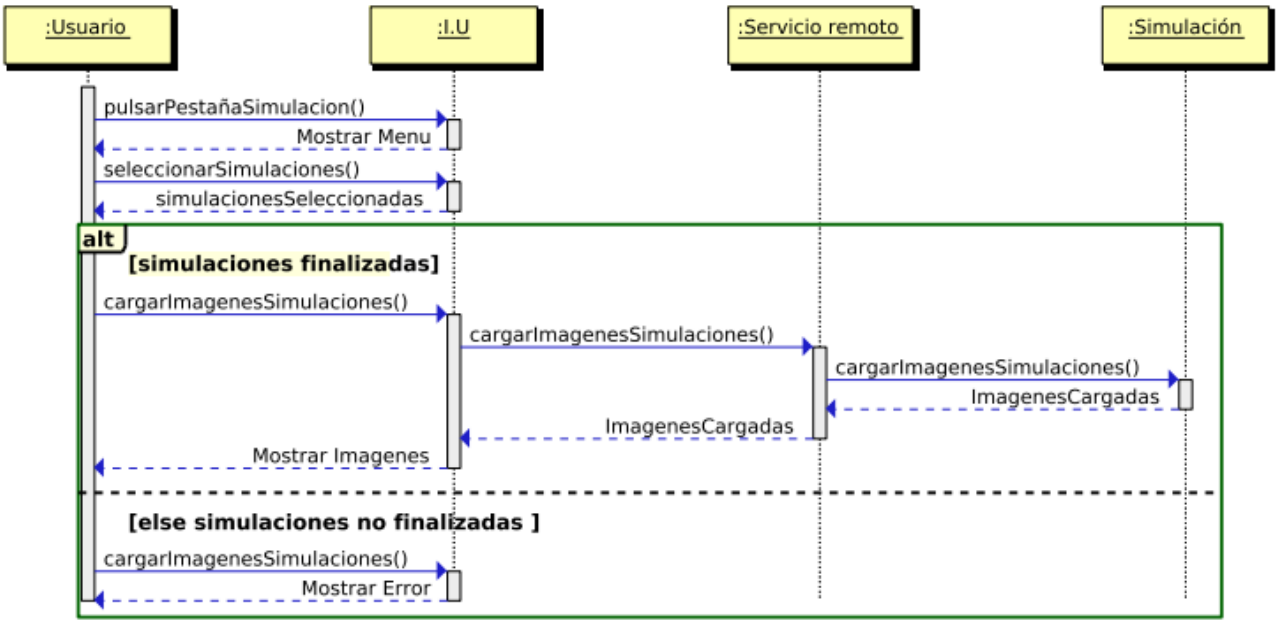

Figura 20: Mostrar Imágenes

4 Fase de Construcción

En la siguiente fase se pretende alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase y siguiendo con el proceso de implementación RUP acabaremos de implementar los artefactos necesarios, como el diagrama de componentes y de despliegue para modelar las distintas partes del sistema y el modelo relacional para la base de datos. Además los integraremos y testearemos obteniendo una versión beta del producto que se pueda poner en manos de los usuarios.

Por otra parte, cabe decir que en la fase de construcción se ha llevado a cabo un proceso iterativo, implementado bajo el sistema operativo Java por parte del cliente e implementado en ExtJs por parte de la Api web de Google Maps.

En primer lugar, el cliente se ha desarrollado bajo la versión 7 de Java bajo el entorno de desarrollo Eclipse y con el Plug-in gráfico de Swing.

En segundo lugar, la api web ExtJS se ha implementado principalmente bajo el lenguaje de programación JavaScript con el framework ExtJS, además de no utilizar ningún entorno de desarrollo.

La aplicación cliente desarrollada principalmente se encarga de gestionar las simulaciones y todas sus variables, además de la posibilidad de añadir estaciones base y las distintas celdas, autenticado previamente, obteniendo los datos de la base de datos MySQL, y que como otras funcionalidades tienen las de almacenar y gestionar los datos del usuario con dicha base de datos que esta situada en la empresa SistelNetworks.

Para esta fase, y una vez tenidos los conceptos básicos claros se ha programado tanto la aplicación cliente como la api de Google Maps en el framework ExtJS siguiendo el estilo de programación por capas.

4.1 Arquitectura de la aplicación

La programación por capas, se puede definir como una arquitectura cliente-servidor donde el objetivo primordial es la separación lógica de negocio de la lógica de diseño, es decir, separando la capa de datos con la capa de presentación al usuario.

En nuestro caso utilizaremos la programación en tres capas, la más utilizada, donde cada aplicación se divide en tres capas bien diferenciadas:

- Capa de Presentación: Es la que ve el usuario, presentando la aplicación, le comunica la información y captura la información que el usuario le indica realizando una validación previa, es decir, en nuestro proyecto será la interfaz gráfica de la aplicación.
- Capa de Negocio: Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Esta capa se comunica con la capa de presentación para recibir las solicitudes y presentar los resultados y con la capa de datos para solicitar al gestor almacenar o recuperar datos de él.
- Capa de Datos: Es donde residen los datos y es la encargada de acceder a los mismo. En el proyecto estará formado por el servicio que accede a la base de datos cuando necesita información, para realizar almacenamiento de datos o para actualización de datos.

Por otra parte, y para finalizar con el apartado de la arquitectura en la que se ha basado dicha programación, cabe destacar que la arquitectura de la solución es por tres capas y dos niveles. Las tres capas han sido detalladas previamente, pero los dos niveles serán necesarios porque la solución reside en dos componentes, por un lado tenemos la presentación mas la lógica que reside en la aplicación de escritorio donde se utilizará la aplicación y, por otro lado, tenemos la lógica mas los datos de la base de datos MySQL que se implementa en un servidor en SistelNetworks y que gracias a toda la interconexión existente se podrán comunicar y realizar las peticiones pertinentes.

Finalmente se puede observar que la principal ventaja para este tipo de arquitectura de programación es la independencia que existe entre niveles, y que permite hacer cambios en un nivel, sin tener que afectar al resto.

4.2 Proceso de Desarrollo

La implementación de la aplicación ha sido desarrollada de forma iterativa y creciente siguiendo el framework (entorno de trabajo) RUP (Rational Unified Process). La idea principal es desarrollar la aplicación de forma incremental, sacando ventaja de lo que se ha aprendido a lo largo del desarrollo e incrementándolo en forma de versiones posteriores o entregables. Dicho aprendizaje se puede obtener por dos caminos, mientras se está desarrollando la iteración o cuando se prueba dicha versión con el cliente, obteniendo así los requisitos para la siguiente versión. Los pasos para tener éxito en este proceso de desarrollo son comenzar con una implementación simple de los requerimientos e iterativamente mejorar la funcionalidad en las distintas versiones hasta que el sistema completo esté implementado.

El proceso en sí mismo consiste en una etapa de inicialización donde se crea la versión del sistema y un producto con el que el usuario pueda interactuar con el proceso, ofreciendo una muestra de los aspectos del problema y obteniendo una solución simple. Para guiar el proceso de iteración se crea una lista de control de proyecto, que contendrá un historial de todas las tareas realizadas y futuras.

Además contiene una etapa de iteración donde involucra el rediseño e implementación de las tareas de la lista de control de proyecto. La meta del diseño es ser simple y modular para soportar el rediseño de la etapa. El análisis de una iteración se basa principalmente en la retroalimentación con el cliente y en las funcionalidades a introducir de la lista de control de proyecto.

En nuestra aplicación se desarrollaron tres iteraciones básicas y bien diferenciadas para abordar el proyecto de la creación de una interfaz gráfica de escritorio para gestionar las simulaciones de redes móviles:

- Primera Iteración: Diseño y construcción de la api de Google Maps con el framework de ExtJS para que se ejecute en un navegador cuando la aplicación de escritorio Java lo requiera, lo que conlleva que esta api necesita tener las principales funciones (CRUD), es decir, crear, obtener, actualizar y borrar las distintas clases que necesitemos para el posicionamiento de estaciones base en escenarios realistas. Introducción a la programación con ExtJS, api de Google Maps y Java. Además se realiza

la conexión entre los distintos sistemas que utilizaremos para realizar dicho proyecto, es decir, la conexión entre el cliente Java y la api de Google Maps para poder realizar el posicionamiento de estaciones base en escenarios realistas, además de la conexión SSH hacia el servidor donde esta almacenada la base de datos y el core del simulador, el cual, nos será útil para obtener toda la información necesaria para las simulaciones.

Por otra parte, se comienza con la implementación de la aplicación de forma inicial, tan solo la consulta de la información obtenida de la base de datos para poder observar que los datos obtenidos son los correctos en cada caso. Finalmente en esta primera iteración, se realiza un primer boceto del diseño de la interfaz gráfica de la aplicación Java de escritorio, con lo que ya podremos realizar la implementación de la aplicación en iteraciones posteriores.

- Segunda iteración: En primer lugar y tras la aceptación del diseño del cliente se realiza la implementación final de la interfaz gráfica de la aplicación, tanto la parte desarrollada con Java como la api de Google Maps. Por otro lado, en el cliente se implementa una primera versión de la gestión de simulaciones y estaciones base en la aplicación eso conlleva las funciones básicas (CRUD) que se implementan para la interfaz. En esta iteración, también, realizamos la implementación de la gestión de usuarios, con la que se podrá dar de alta o borrar usuarios para el posterior acceso a la interfaz del simulador. A continuación se parametrizan las variables de la base de datos en hibernate, para obtener una interfaz más eficiente. Por otro lado, en la parte de obtención de resultados, se implementa la parte de visualización de gráficos. Y por último, se acuerda con el cliente, realizar una autenticación, por lo que se decide realizar un pequeño modulo de autenticación que consistirá en un algoritmo de autenticación con encriptación MD5, además de la gestión de errores en dicho login y su interfaz gráfica correspondiente.

- Tercera iteración: En esta iteración realizamos la implementación de la gestión de los obstáculos, que se incluyen en las simulaciones de tipo realistas en zonas interiores haciendo referencia, por ejemplo, a paredes dentro de un hogar, y por último, la gestión de tipos de obstáculos, donde cada usuario puede definir el material y la atenuación de cada uno de los obstáculos que va a introducir en el escenario. A continuación realizamos el diseño e implementación de la visualización de resultados que faltan, es

decir, de las imágenes, kml, exportar los resultados a ficheros .csv... Finalmente, se implementa una serie de servicios adicionales para hacer más eficiente la interfaz, utilizar hilos para el lanzamiento de múltiples simulaciones, visualización de las imágenes en el computador del usuario y que están almacenadas en el servidor del simulador.

Para realizar el proyecto se ha utilizado en entorno de trabajo Eclipse con el plugin de Swing, además de un editor de texto para desarrollar el modulo ExtJS de Google Maps.

4.3 Diseño del Sistema

En este apartado se define los componentes, módulos y datos del sistema para satisfacer los requerimientos definidos previamente. Además el diseño de sistema es la primera fase en la cual se selecciona la aproximación básica para resolver el problema, se decide la estructura y el estilo global.

Para continuar con el diseño orientado a objetos se realiza un diagrama de componentes que representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos.

4.3.1 Diagrama de Componentes

El diagrama de Componentes es un tipo de diagrama UML que se diseña para obtener una visión estática y dinámica de los componentes físicos y sus dependencias.

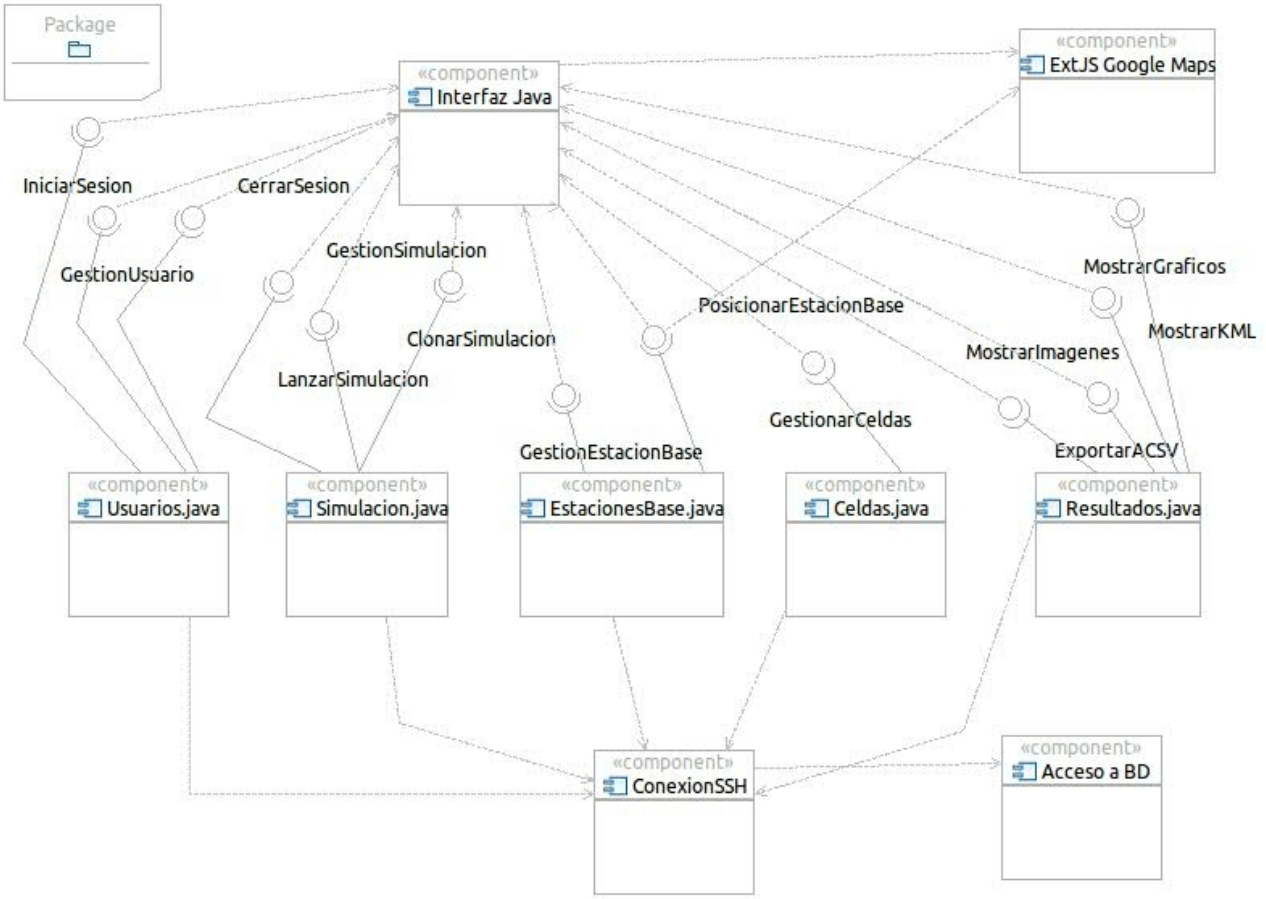

Figura 21: Diagrama de componentes

En el diagrama de componentes diseñado se han implementado interfaces, dado que ofrece la ventaja de romper la dependencia directa entre componentes, en este caso entre las clases java y la interfaz de escritorio Java. Una interfaz contiene una o varias operaciones y se utiliza para especificar los servicios de una clase o de un componente, además se conecta al "component" que la implementa a través de una relación de realización, y al componente que utiliza sus servicios con una dependencia.

4.3.2 Diagrama de Despliegue

El diagrama de despliegue es un tipo de diagrama UML que utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes.

Para el diseño del diagrama de despliegue utilizamos nodos, componentes y asociaciones con el que se representa la topología del hardware sobre el que se ejecuta el sistema.

En el proyecto se ha implementado un diagrama de despliegue para modelar el sistema cliente-servidor, los cuales son un extremo del espectro de los sistemas distribuidos y requieren la toma de decisiones sobre la conectividad de red de los clientes a los servidores y sobre la distribución física de los componentes software del sistema a través de los nodos.

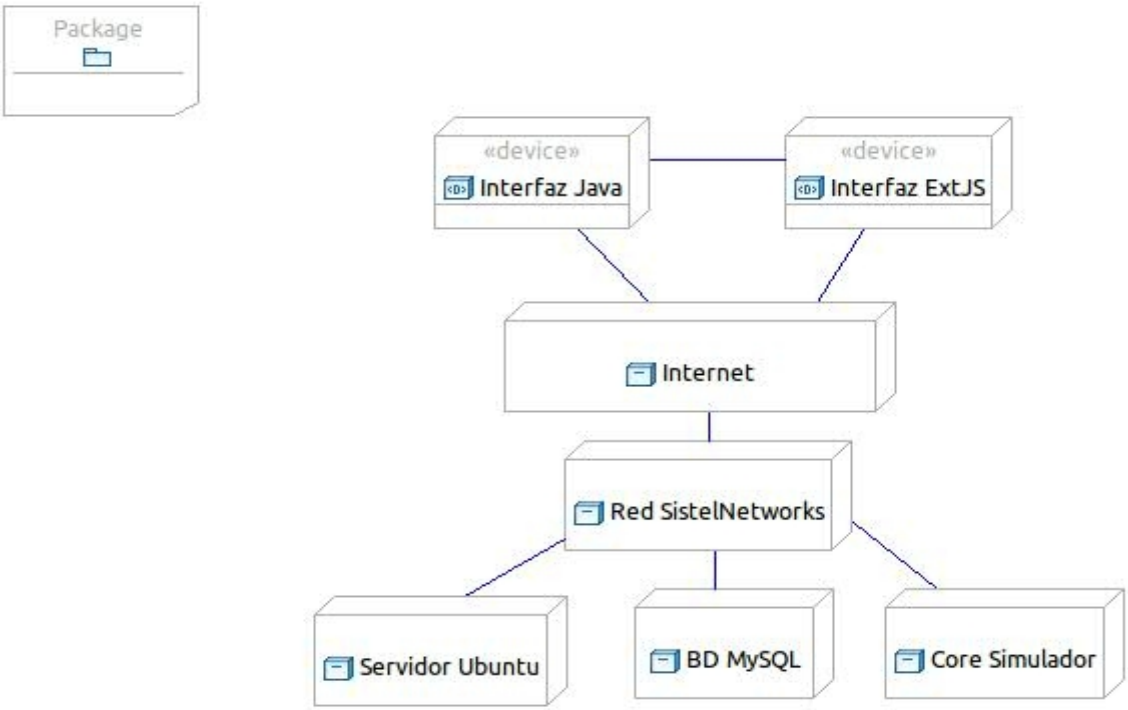

Figura 22: Diagrama de despliegue

En el diagrama de despliegue diseñado se ha implementado la arquitectura hardware en la que se trabajará. Por un lado, tendremos la aplicación Java de escritorio y por otro lado la api implementada con ExtJS para la parte de Google Maps. La aplicación Java se conectará a Internet y via SSH tendrá acceso a la red de SistelNetworks para poder acceder al servidor, al core del simulador situado en este servidor y a la base de datos MySQL.

4.4 Programación de la Aplicación

En el siguiente apartado se va a exponer todo el desarrollo y las tecnologías utilizadas para la elaboración de este proyecto analizando los lenguajes y el entorno de desarrollo que se han utilizado para implementar el sistema de mantenimiento, tanto por parte del servidor como por parte del cliente.

Todo esto se detallará profundizando en las iteraciones que se han explicado anteriormente escribiendo la memoria tal como se ha realizado el proyecto, es decir, siguiendo el entorno de trabajo RUP.

4.4.1 Primera Iteración

En esta primera iteración se han analizado todas las tecnologías que se han elegido para implementar la interfaz del simulador y todos los servicios asociados, tales como el entorno de desarrollo Eclipse, la conexión segura SSH, Java, el framework ExtJS, la api de Google Maps...

Seguidamente, se ha detallado toda la implementación de las conexiones, además de las principales funciones de la API de Google Maps para ExtJS para poder realizar el posicionamiento de las estaciones base para escenarios realistas.

Finalmente, se realiza un pequeño boceto del diseño de la interfaz de escritorio Java, para mostrársela al cliente, y que este de su aprobación.

4.4.1.1 Lenguaje de Programación Java

Para el desarrollo de la aplicación de escritorio, se ha decantado por utilizar el lenguaje multiplataforma Java, por las ventajas que se indican en el capítulo de especificaciones técnicas.

4.4.1.1.1 Entorno de desarrollo Eclipse

Para el desarrollo de la aplicación de escritorio, es decir, la interfaz Java, se ha optado por utilizar el entorno de desarrollo Eclipse. Este entorno es una plataforma de desarrollo open-source y multiplataforma basada en Java. Esta plataforma ha sido usada para desarrollar entornos de desarrollo integrados (IDE) y que, incluso, han llegado a ser utilizados como componentes del mismo Eclipse.

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el creado Eclipse Modeling Project, que cubre casi todas las áreas de Model Driven Engineering y con el cual, se ha desarrollado todos los artefactos detallados anteriormente, como los diagramas de componentes, diagramas de despliegue...

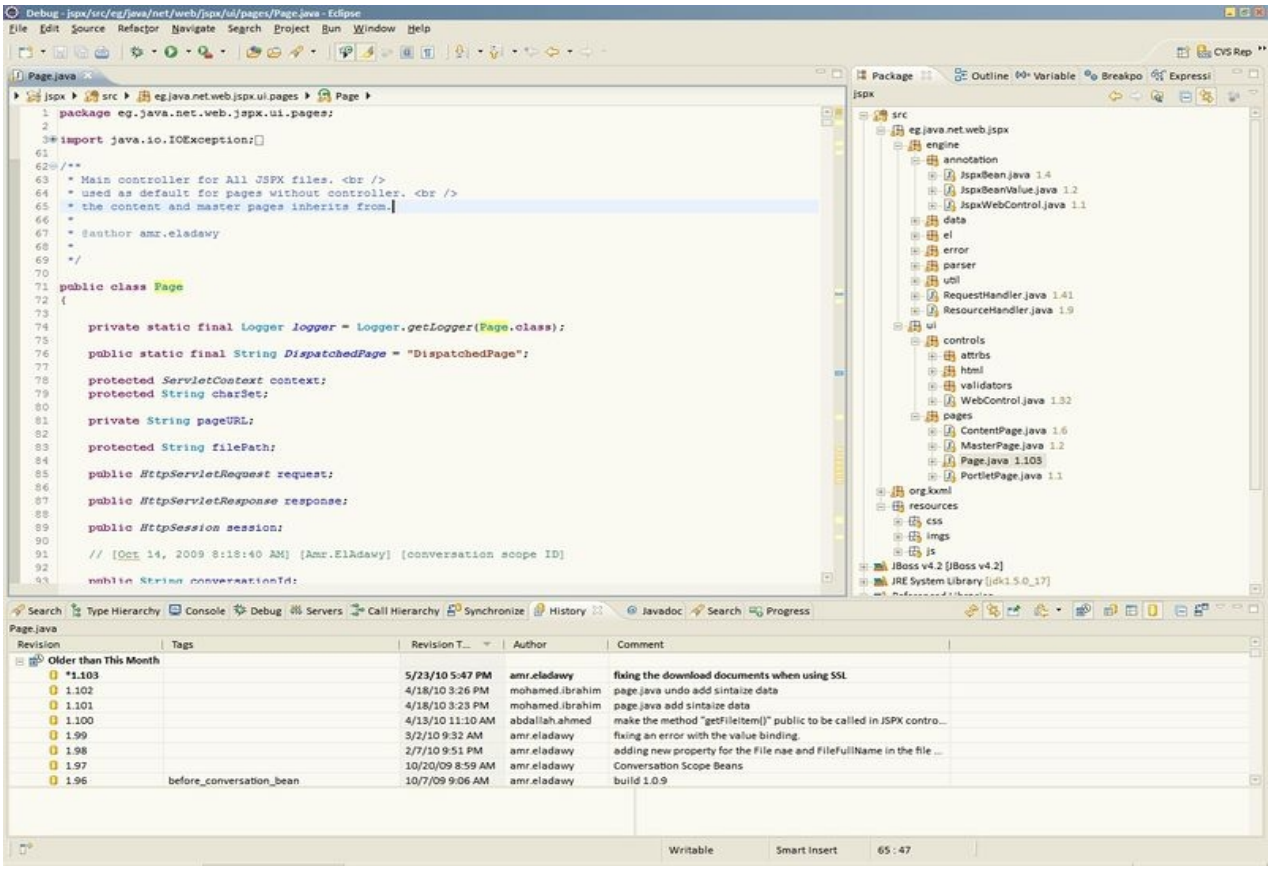

Figura 23: Entorno de desarrollo Eclipse

En el caso del proyecto, se utiliza la versión "Helios" 2.0 de Eclipse for Java Developers, en un entorno Linux. Tras la descarga del software desde http://www.eclipse.org/downloads se puede ejecutar de forma sencilla y en primer lugar nos solicita en que directorio queremos situar nuestro proyecto.

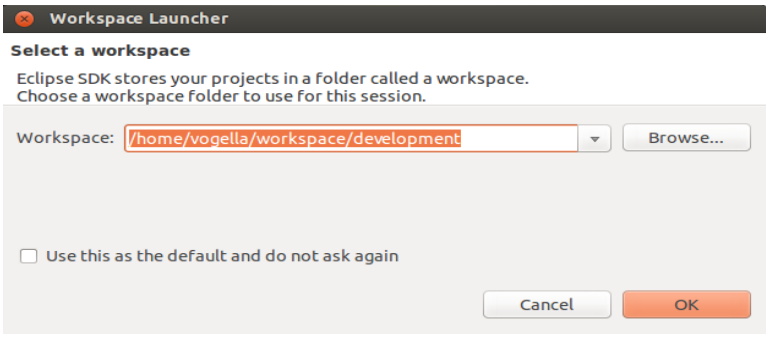

Figura 24: Selección del espacio de trabajo

Eclipse permite organizar los ficheros en forma de proyecto. Para crear el proyecto Java se procede del siguiente modo:

1. Seleccionando en el menú File → New → Project.

2. Aparece una ventana y se selecciona sobre Java → Java project.

3. Se rellenan los datos del proyecto y se pulsa sobre Finish.

4. Eclipse automáticamente abre la perspectiva Java y se crea un directorio especificado con dos ficheros .project y .classpath que contienen toda la información relativa al proyecto.

5. Tras esto, estamos listos para comenzar a crear las clases asociadas y las funciones que detallaran el proyecto completo.

4.4.1.1.2 Introducción a Swing

Para el desarrollo de la aplicación de escritorio, es decir, la interfaz Java, se ha optado por utilizar el paquete Swing, el cual, es parte de la JFC (Java Fundation Classes) en la plataforma Java, esta provee de facilidades para ayudar a construir interfaces de escritorio abarcando botones, tablas, marcos, etc...

Swing existe desde la JDK 1.1 y hereda todo el manejo de eventos de la antigua AWT (Abstract Window Toolkit).

Una aplicación Swing básica se construye mezclando componentes con las siguientes reglas:

- Debe existir, al menos, un contenedor de alto nivel, que provee el soporte que las componentes Swing necesitan para el pintado y el manejo de eventos.

- Otras componentes colgando de dicho contenedor (otros contenedores o componentes simples)

El contenedor de alto nivel (top-level container) puede ser:

- Jframe: Ventana independiente.
- Japplet: Un applet.
- Diálogos: Ventanas de interacción con el usuario.

Swing permite configurar y cambiar el aspecto y la forma de interacción de una aplicación "Look & Feel" de la siguiente forma:

UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName())

Por otro lado, para determinar de qué modo esas componentes se organizarán visualmente se utiliza el concepto Layout. Existen diversos Layout o formas de organizar los componentes dentro del contenedor:

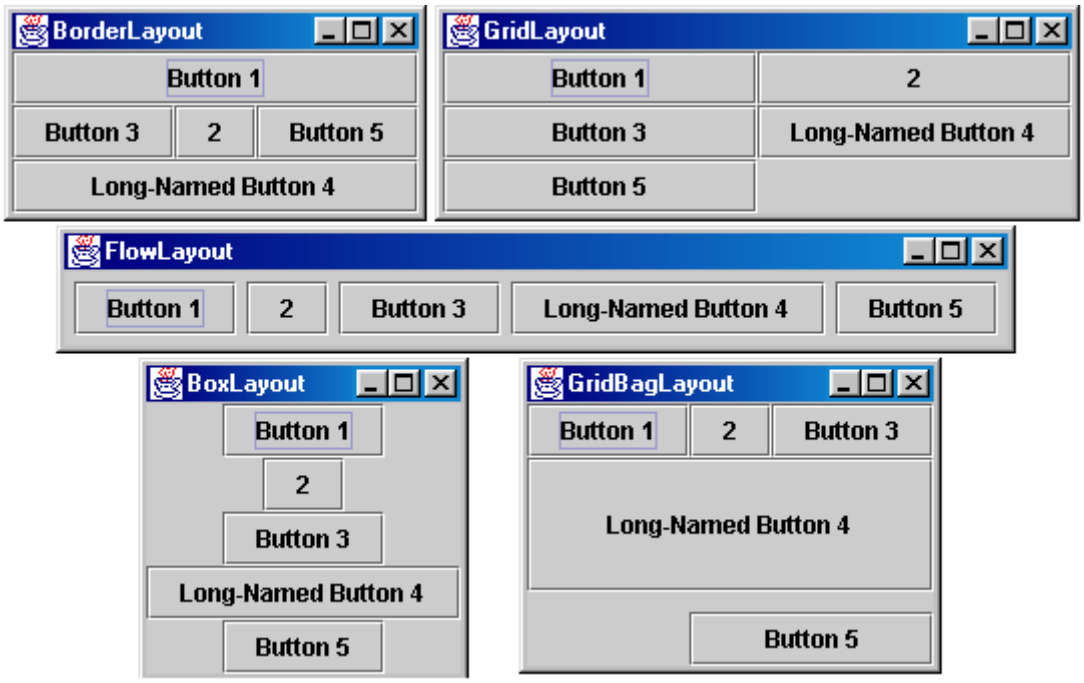

Figura 25: Layouts en Java

Además cada vez que el usuario interactúa con la aplicación se producen eventos, para que un componente reaccione frente a un evento, se debe proveer de un "listener" con al menos, un método que se ejecutará cuando un usuario realice una acción en particular, por ejemplo al pulsar un botón:

```
JButton boton=new JButton("Accept");
    boton.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent e){
          System.out.println("Clic the button");
```

```
        }
    });
```

4.4.1.1.3 Estructura de un proyecto Java

Tras la creación de un proyecto Java se genera automáticamente la estructura de carpetas para la generación de la aplicación. Esta estructura será común a cualquier aplicación Java, en nuestro caso, a partir de esta estructura se expande según las necesidades del proyecto (añadiendo carpetas, librerías...), pero siempre siguiendo la programación en tres capas:

- DevelMapsJv/src/ → Contiene todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, conexión SSH, conexión con la base de datos, clases, manejo de errores, etc. En nuestro caso seguiremos la estructura de las tres capas dentro de este directorio:

- controller → En este paquete se crea una clase para cada dato a gestionar. Su función es la de gestionar las distintas clases que se deben crear, mediante la definición de todas sus variables y las distintas funciones. Además tiene una clase Global.java, donde tenemos definidas variables globales.

- DAOmodel → En esta parte se han definido todas las funciones para el acceso a datos mediante hibernate, lo cual nos facilitara las tareas de bases de datos, dado que hibernate tiene persistencia de datos, y por tanto no tendremos que hacer las consultas de forma tan manual, si no que ya tenemos la clase definida y, por tanto, podemos trabajar sobre ella en las distintas consultas que se requieran.

- mapeos → Los distintos mapeos en XML para poder trabajar con hibernate.

- model → En este directorio, se almacenan todas aquellas clases secundarias y que se pueden abstraer para utilizarlas en varias clases principales.

- view → Aquí se almacenan toda la parte de la interfaz gráfica, este directorio esta compuesto por varios directorios:

  - view.real → Se almacena la vista de la configuración de una simulación realista.

  - view.realindoor → Se almacena la vista de la configuración de una simulación realista interna.

  - view.realpico → Se almacena la vista de la configuración de una simulación realista con pico celdas.

- view.synthetic → Se almacena la vista de la configuración de una simulación sintética.

- DevelMapsJv/jfreechart/ → Contiene la librería de gráficos JfreeChart, de la cual se comentara en el siguiente apartado.

- Finalmente, se deben incluir todo el resto de librerías necesarias, como por ejemplo la mysql-connector-java, para conexiones a base de datos MySQL... Estas librerías, a diferencia de jfreechart, se debe incluir haciendo clic derecho sobre el proyecto → Java Build Path → y en la pestaña Libraries se pueden adjuntar el resto de librerías.

4.4.1.1.4 Gráficos Java – Librería JfreeChart

Para el proyecto es necesario procesar los datos obtenidos y tras una serie de cálculos, poder representarlos de forma gráfica, para ello, tras una investigación exhaustiva se ha decidido utilizar la librería de gráficos JfreeChart, esta librería es 100% desarrollada bajo el lenguaje de Java, y es bastante sencilla de utilizar y de incluir en Java gráficos bastantes útiles para nuestro proyecto, dado que necesitamos observar los datos tras las simulaciones configuradas de una manera entendible. Sus principales características son:

- Dispone de una API bien documentada y además soporta un amplio rango de gráficos.

- Soporta muchos tipos de salida, sobretodo y los más importantes para nuestro proyecto, componentes Swing.

- Finalmente, JfreeChart es open source, por lo tanto se puede utilizar para nuestro proyecto sin ningún coste adicional.

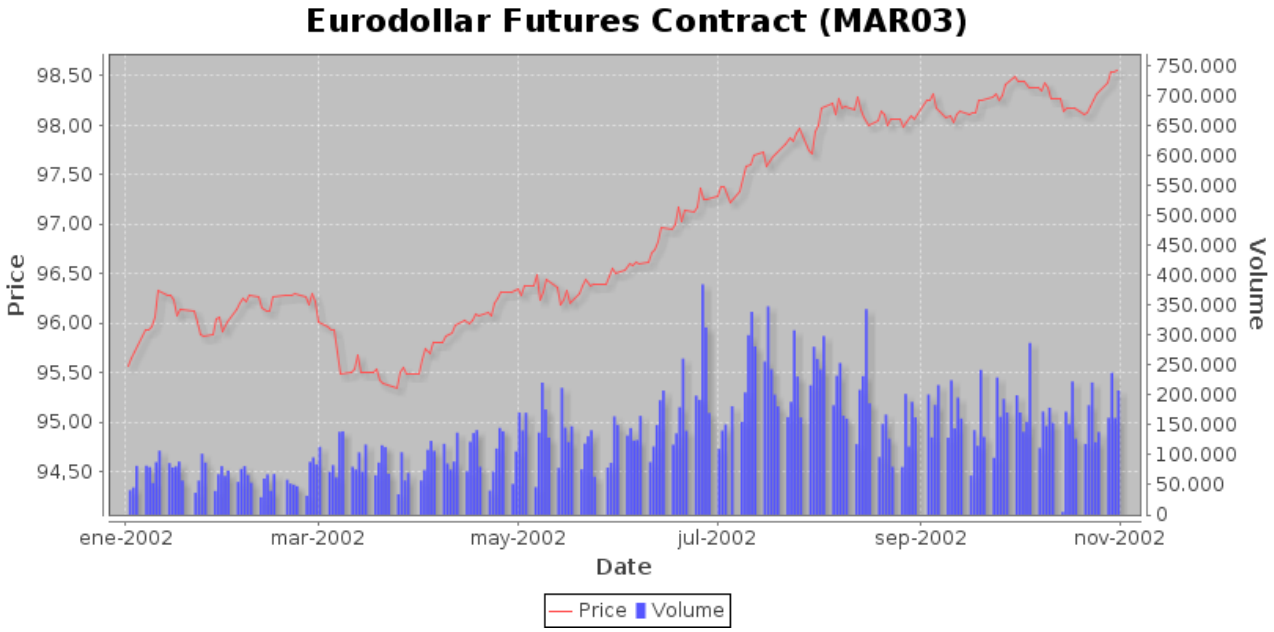

Figura 26: Gráficos JFreeChart

4.4.1.2 Framework ExtJS

Es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas con posibilidad de un "look & field" similar a la de una aplicación de escritorio usando tecnologías como AJAX, DHTML y DOM, la cual, ha sido desarrollada por la empresa Sencha. Entre sus principales funcionalidades o componentes (widgets) son:

- Campos para fechas.
- Campos numéricos.
- Combos.
- Radiobuttons y checkboxes.
- Editor HTML.
- Elementos de datos (con modos de sólo lectura, datos ordenables, columnas que se pueden bloquear y arrastrar, etc.).
- Árbol de datos.
- Pestañas.
- Barra de herramientas.
- Menús al estilo de Windows.
- Paneles divisibles en secciones.
- Sliders.
- Gráficos

4.4.1.2.1 Propósito de uso de ExtJS

El propósito de uso de ExtJS con su API de Google Maps ha sido para situar estaciones base, dado que se requiere de un mapa, en el proyecto, se ha decidido utilizar porque que ya se tenia experiencia en esta framework y era lo más viable respecto al tiempo del proyecto.

4.4.1.2.2 Introducción ExtJS

Para el posicionamiento de estaciones base en escenarios realistas se ha decantado por la tecnología ExtJS, basada en javascript, dada su versatilidad y la capacidad de la integración de la api de Google Maps necesaria para dar solución a las necesidades de gestión de estaciones base.

PHP: PATRÓN MVC (Model View Controller)

/lib/FrontController.php : Todas las peticiones pasan por esta clase, se tiene la posibilidad de controlar de forma centralizada ciertos aspectos de la aplicación.

El método FrontController::main() espera dos parámetros para saber qué petición de qué controlador hay que ejecutar, además de comprobar que el usuario se haya validado previamente:

• control : Este parámetro permite obtener el controlador que se está invocando. Por ejemplo, control = EstacionBase implica la existencia de un controlador EstacionBase Controller definido en un fichero EstacionBase Controller.php alojado en el directorio php/controller/

• acción : Este parámetro determina qué método se quiere invocar. El controlador se encargara de obtener y validar los parámetros de entrada, además de devolver la respuesta que solicita el cliente.

ExtJS: PATRÓN MVC (Model View Controller)

Se ha programado mediante el patrón MVC (model view controller) definido en ExtJS, utilizando PHP en la parte que comunica entre la base de datos “MySQL” y el framework desarrollado en ExtJS:

– Model: Colección de campos y datos Json que sirven para interaccionar con

PHP, y este a su vez gestiona dichos datos con la BBDD.

– View: Se refiere a la parte visual del framework: grids, trees, vistas...

– Controller: Parte de código que realiza las funciones de las vistas: rendering, creación de modelos...proporcionados por el simulador.

Esta arquitectura tiene varias ventajas, dado que trata de proporcionar una consistencia en todo el framework:

• Cada aplicación funciona de la misma manera.

• Es fácil compartir código entre aplicaciones.

/js/app.js : Es el punto de entrada a la aplicación, este método crea una instancia Ext.app.Aplication y se inicia el método de puesta en marcha tan pronto como la página este lista. En este apartado, se realiza la creación del espacio de nombre 'SNF' para la aplicación, la realización de la llamada de todos los controladores y la creación del contenedor Viewport, donde se gestiona la colocación de cada panel.

/js/md5-min.js : Se utiliza esta librería para la gestión de la clave del usuario en md5.

4.4.1.2.3 ExtJS y la Api de Google Maps

Se pretende integrar el módulo de Google Maps para representar las estaciones base en un escenario realista. Para la cual, se debe incluir un archivo que se descargara de los ejemplo de ExtJS y se colocara dentro de la librería extjs para su futura utilización. Una vez se ha hecho esto, no hay más que incluir el mapa dentro de un contenedor de extjs, en el ejemplo, dentro de una ventana:

```
Ext.onReady(function(){
    var mapwin;
     mapwin = Ext.create('Ext.window.Window', {
          autoShow: true,
          layout: 'fit',
          title: 'GMap Window Example',
          closeAction: 'hide',
          width:450,
          height:450,
          items: {
                  xtype: 'gmappanel',
           }
    });
 });
```

Gracias a esta API de Google Maps, se puede configurar tal como si estuviéramos dentro de Google Maps, se pueden añadir markers, que utilizaremos para el posicionamiento de las estaciones base, se pueden añadir rectángulos, que añadiremos para indicar el rango a simular...

Figura 27: Google Maps en ExtJS

Como se puede observar en la imagen, se tienen 5 estaciones base, con distintos números de celda, cada una representada por una flecha de color y los grados de inclinación. Por otro lado, se tiene un rectángulo que representa el área de simulación a la cual ara referencia a la hora de la representación de imágenes y gráficos.

4.4.1.2.4 ExtJS y MySQL

PHP: COMUNICACIÓN (MySQL – ExtJS)

En este punto se detalla la funcionalidad de la parte desarrollada en php, que conecta la base de dato de gestión del simulador con el framework ExtJS,

- index.php : Define el acceso de la aplicación. Contiene referencias al fichero

app.js donde están todas las referencias a los ficheros Javascript, además referencias a la guía de estilos css utilizada por ExtJS. Por otro lado incluye una referencia a "config.php" que se detalla a continuación.

- php/config.php : Define las constantes de las rutas absolutas de los directorios PHP, además de la ruta de la carga "automática" de paneles de navegación y módulos. También se encarga de definir los parámetros de conexión a la BD.

- authenticate.php: Este servicio se utiliza para realizar la validación del usuario.

– framework.php : Define un alias de la clase que implementa el patrón FrontController.php y que se detalla más adelante.

- logout.php : Realizar la desconexión de un usuario.

PHP: CONEXIÓN A BASE DE DATOS (MySQL)

Se ha utilizado una extensión de objetos de datos de PHP (PDO) que definen una interfaz ligera para poder acceder a bases de datos en php. Se debe utilizar un controlador de PDO especifico de la base de datos para tener acceso a un servidor de base de datos.

/lib/ModelBase.php : Aquí se gestionan las conexiones base de datos.

/lib/DBManager.php : Se encarga de realizar la apertura de la base de datos MySQL, mediante el controlador PDO, que se encarga de gestionar los datos de la interfaz.

4.4.1.3 Conexión SSH – Librería JSch

Tras un periodo de investigación y de revisar distintas opciones, se optó por utilizar la librería Jsch, esta es una pequeña biblioteca de Java que provee facilidades para conexiones por medio de canales seguros SSH.

JSch se caracteriza por utilizar los conceptos de "sesión" y "canal", la sesión se puede decir que será el medio de comunicación seguro con el servidor remoto. En cambio, el "canal", es una vía de comunicación que nos permite acceder a toda la funcionalidad en el host que deseamos acceder.

Jsch incluye por defecto varios canales para realizar ejecución de comandos remotos, consola remota, FTP seguro, conexiones TCP-IP, entre otras.

Para su utilización en la aplicación de escritorio Java, en primer lugar, se debe descargar Jsch de su página web y se debe agregar a su Java Build Path importando

así dicha librería, como se indica en capítulos anteriores.

Tras su importación lo primero es crear un objeto del tipo Jsch. Este objeto sirve para crear sesiones y manejar todas las credenciales de autenticación.

```
public class ConexionSSH {
  private JSch jsch;
  public ConexionSSH() {
    jsch = new JSch();
  }
  public static void main(String args[]) {
    new ConexionSSH();
  }
}
```

Antes de conectar la sesión, se debe especificar las "credenciales" de autenticación. SSH se puede identificar por medio de combinación de usuario/password o por medio de pares de llaves pública/privada. En el proyecto, tan solo utilizaremos el par usuario/password.

Por tanto, se debe crear una nueva sesión especificando el nombre de usuario, y el host al que se desea conectarse. Finalmente se le indica la clave a utilizar.

```
public class ConexionSSH {
  private JSch jsch;
  public ConexionSSH() {
    jsch = new JSch();
    // Es necesario capturar JSchException
    try {
      // NO realizar revision estricta de llaves
      JSch.setConfig("StrictHostKeyChecking", "no");
      // Creamos la nueva sesión SSH
      Session sesion = jsch.getSession("usuario","host");
      // Establecemos la clave
```

```
      sesion.setPassword("su_clave");
    } catch(JSchException e) {
      System.out.println("Error de JSCH. Mensaje: "+e.getMessage());
    }
  }
}
```

4.4.1.4 Conexión SSH – Librería Jsch - MySQL

Para poder conectarnos a la base de datos del servidor, se debe hacer un portForwarding, es decir la creación de un túnel en un determinado puerto para poder conectar con las base de datos MySQL del servidor de SistelNetworks.

```
public class ConexionSSH {
  private JSch jsch;
  public ConexionSSH() {
    jsch = new JSch();
    // Es necesario capturar JSchException
    try {
      // NO realizar revision estricta de llaves
      JSch.setConfig("StrictHostKeyChecking", "no");
      // Creamos la nueva sesión SSH
      Session sesion = jsch.getSession("usuario","host");
      // Establecemos la clave
      sesion.setPassword("su_clave");
    } catch(JSchException e) {
      System.out.println("Error de JSCH. Mensaje: "+e.getMessage());
    }
    int assigned_port = sesion.setPortForwardinL(lport, rhost, rport);
    try {
```

```
    // Conexión con la base de datos del servidor
    Connection  unaConexion = DriverManager.getConnection(“
       jdbc:mysql://localhost”+lport+”/nombreBaseDatos”, “user”, “pass”)
  } catch(SQLException a) {
    System.out.println("Error de conexión MySQL. Mensaje: "+a.getMessage());
  }
 }
}
```

Con este código tenemos acceso a la base de datos como si se estuviera trabajando en local, y se pueden hacer consultas para la gestión de las simulaciones.

4.4.1.5 Listado Simulaciones

Una vez se ha explicado todos los componentes que se van a utilizar en el proyecto, ya se es capaz de definir un listado de consulta de simulaciones básico, mediante el adaptador “AbstractTableModel” que nos genera una tabla que se puede configurar y añadir los campos que se necesiten para cada listado y con la conexión entre la base de datos vía SSH, y a través de la interfaz de escritorio Java que se ha explicado anteriormente, se obtiene el siguiente resultado:

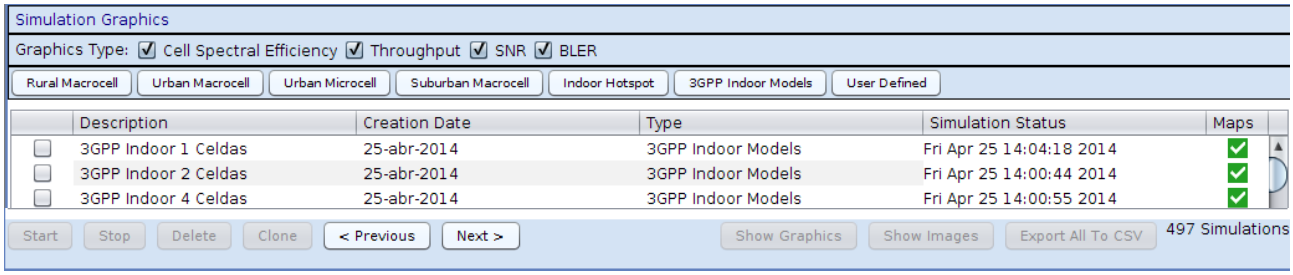

Figura 28: Listado de simulaciones

4.4.2 Segunda Iteración

En esta segunda iteración se han realizado diferentes reuniones con el cliente, para obtener una aceptación de la interfaz final.

Además se ha desarrollado la programación de la gestión de simulaciones y estaciones base. Finalmente se realiza la gestión de acceso (Autenticación con encriptación MD5) y la gestión de los usuarios.

4.4.2.1 Gestión de acceso/usuarios

La gestión de usuarios esta reservada para derechos de administrador, por lo que tan solo aparecerá en el panel de navegación, cuando un usuario tenga dichos derechos, tendrá acceso al panel del sistema, y desde ahí se pulsa sobre "Users" donde se puede acceder al listado de usuarios que tienen acceso a la aplicación. Esta acción da lugar a la llamada de la función donde se obtienen todos los usuarios que existen en la tabla correspondiente de la base de datos.

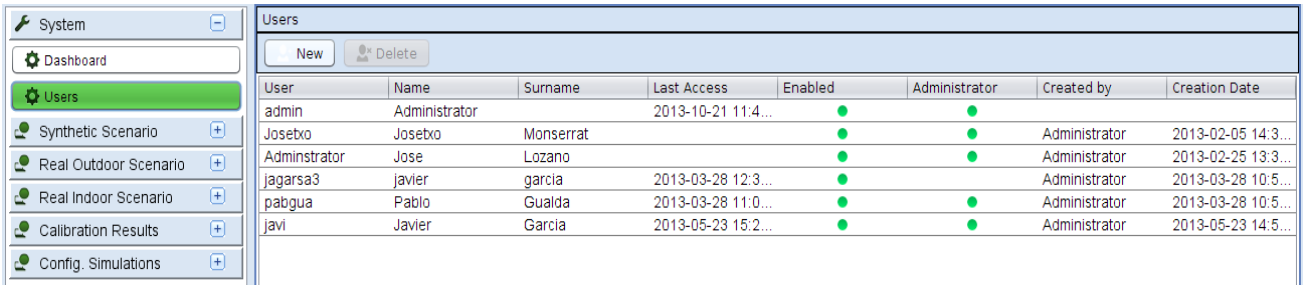

Figura 29: Gestión de usuarios

4.4.2.1.1 Carga de datos de la BD "MySQL"

Dado que la carga de datos de Java a la BD "MySQL" se desarrolla de forma similar, se comentará en esta sección de forma genérica.

Para realizar dicha carga, en primer lugar se debe realizar la conexión SSH mediante la librería Jsch (comentada en apartados anteriores).

Una vez hecho esto, seguidamente se generan los ficheros necesarios para

realizar un parseado de todas las variables mediante Hibernate.

Hibernate, como la definen sus autores, es una herramienta de mapeo objeto/relacional para ambientes Java. Además no solo se encarga del mapeo de clases Java a tablas de la base de datos (y de regreso), sino que también maneja los queries y recuperación de datos, lo que puede reducir de forma significativa el tiempo de desarrollo que de otra forma gastaríamos manejando los datos de forma manual con SQL y JDBC, encargándose de esta forma de alrededor del 95% de las tareas comunes relacionadas con la persistencia de datos, manejando todos los problemas relativos con la base de datos particular con la que estemos trabajando, de forma transparente para nosotros como desarrolladores. Entonces, si cambiamos el manejador de base de datos no será necesario que modifiquemos todo el SQL que ya teníamos para adaptarse al SQL que maneja la nueva base de datos. Solo será necesario modificar una línea en un archivo de configuración de Hibernate, y este se encargará del resto.

Dado que se trata de un proyecto de gran tamaño y con múltiples variables para la configuración de las distintas simulaciones, se decidió utilizar el parseado de hibernate para obtener una mayor eficiencia de tiempo. Se detalla un ejemplo sencillo del funcionamiento para la aplicación en el proyecto.

En primer lugar, se debe crear una biblioteca con casi todos los jar de hibernate, en concreto:

- hibernate3.jar

  y del directorio de jars requeridos:

- antlr-2.7.6.jar
- commons-collections-3.1.jar
- dom4j-1.6.1.jar
- javassist-3.4.GA.jar
- jta-1.1.jar

El siguiente paso es crear las diferentes clases cuyas instancias serán almacenadas como fila en una tabla de la base de datos, es decir, las entidades. Un ejemplo sencillo de estación base sería:

```
public class EstacionBase implements Serializable
{
   private long id;
   private String nombre;
```

```
private String xPosicion;
private String yPosicion;

public EstacionBase()
{
}

public EstacionBase(String nombre, String xPosicion, String yPosicion)
{
   this.nombre = nombre;
   this. xPosicion = xPosicion;
   this. yPosicion = yPosicion;
}

public String getXPosicion ()
{
   return xPosicion;
}

public void setXPosicion (String xPosicion)
{
   this. xPosicion = xPosicion;
}

public long getId()
{
   return id;
}

private void setId(long id)
{
   this.id = id;
}

public String getNombre()
{
   return nombre;
}

public void setNombre(String nombre)
```

```
    {
        this.nombre = nombre;
    }

    public String getYPosicion ()
    {
        return yPosicion;
    }

    public void setYPosicion (String yPosicion)
    {
        this. yPosicion = yPosicion;
    }
}
```

Se ha utilizado la convención de nombres JavaBeans, con esto podemos empaquetar las clases y utilizarlas en otros proyectos o aplicaciones.

Por otro lado, el siguiente paso es crear un archivo de mapeo, es decir, un fichero xml que contendrá el mapeo de la clase "EstacionBase" con la tabla "BaseStationLocation" de la base de datos, con el nombre EstacionBase.hbm, el IDE se encargará de agregarle el .xml al final del nombre del fichero. El fichero de mapeo quedará de la forma siguiente:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <class name="hibernatenormal.EstacionBase" table="BaseStationLocation">
        <id name="id" column="ID">
            <generator class="identity" />
        </id>
        <property name="nombre" type="string" column="NOMBRE" />
        <property name="xPosicion" />
        <property name="yPosicion" />
    </class>
</hibernate-mapping>
```

Una vez definidos todos los mapeos de cada tabla que corresponde con cada

clase de la base de datos se debe configurar Hibernate, indicandole información de la conexión,...

Para ello generamos un fichero llamado hibernate.cfg.xml donde se borra todo el contenido generado por el IDE, y se añaden las siguiente lineas:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
</hibernate-configuration>
```

Para el proyecto, se necesitan el drive de conexión “MySQL”, el lugar “url” donde se ha conectado la base de datos, el usuario y contraseña de estas, el pool de conexiones posibles, y a continuación el lugar donde se encuentran todos los archivos de mapeos creados anteriormente.

Finalmente, se deben crear los DAO de cada clase, en esta sección se inicializa el atributo SessionFactory, para asegurarnos de que solo existe una instancia hibernate en la aplicación. Además de la incialización de esta variable, se deben incluir aquí, todas las operaciones que se realizaran en la base de datos: insertar, actualizar, borrar, leer...pero esta vez no hace falta incluir todas y cada una de las variables de la base de datos, gracias a hibernate y su parseo de datos.

4.4.2.2 Gestión de simulaciones

La gestión de simulaciones esta dividida en la interfaz en cuatro pestañas diferenciadas, dependiendo de cada tipo de simulación que deseamos configurar.

En primer lugar se han diseñado los escenarios sintéticos, los cuales se utilizan para escenarios interiores y exteriores pero preconfigurados, es decir, con la mayoría de variables definidas por defecto, con lo cual, el usuario tan solo tiene la posibilidad de configurar unas pocas variables, estos escenarios sintéticos se han definidos tanto para ciudades como para pueblos o barrios y tienen las celdas ya configuradas y posicionadas, con lo que limita mucho la variabilidad de la simulación, pero es un buen comienzo, para obtener unos datos previos.

Seguidamente, se ha desarrollado una pestaña para los escenarios sintéticos

pero con celdas de tipo pico, donde el usuario puede realizar una configuración mucho más amplia de la simulación, pero en este caso, las estaciones base son de tipo pico, y ya están configuradas y posicionadas, por lo que también limita la variabilidad de la simulación y la variabilidad de la obtención de las imágenes de la SNR y Througphut.

Por otro lado, se ha desarrollado otra pestaña para los escenarios exteriores de tipo realista, en este caso tenemos la posibilidad de realizar una configuración amplia de la simulación, del mismo estilo que los escenarios sintéticos con celdas de tipo pico, además una vez configuradas las simulaciones se pueden añadir y situar las celdas en el mapa desarrollado con el plugin de Google Maps en el framework ExtJS . Una vez situadas, posicionadas y configuradas todas las celdas necesarias para una simulación, se dibuja un rectángulo, de forma que se restringe la simulación al área seleccionada por dicho rectángulo.

Finalmente, como última pestaña de simulaciones se ha diseñado los escenarios interiores de tipo realistas. En este caso, también se tiene una configuración de las simulaciones mucho más amplia, además una vez se han configurado las simulaciones se introduce una imagen del escenario interior que se quiere simular, por ejemplo, una sala de oficinas. Una vez se ha guardado la imagen en la base de datos, se pasa al posicionamiento y configuración de todas las celdas, dentro de dicha imagen, que es la que se va a pasar a simular.

4.4.2.3 Autenticación MD5

La aplicación de escritorio del simulador realiza una autenticación previa para evitar el acceso de intrusos, esta autenticación se realiza de una forma sencilla pero efectiva, mediante un método que recibe una cadena (String) y la transforma mediante encriptación con el algoritmo MD5, una vez hecho esto, se comprueba con la base de datos si el código de la cadena en MD5 corresponde con la clave almacenada en la base de datos en MD5, en concreto el código utilizado es el siguiente:

```
MessageDigest md = MessageDigest.getInstance("MD5");
md.update(clave.getBytes());

byte byteData[] = md.digest();

// convert the byte to hex format method 1
```

```
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < byteData.length; i++) {
                sb.append(Integer.toString((byteData[i] & 0xff) + 0x100, 16)
                                .substring(1));
        }

        ConnectionDAO conexion = new ConnectionDAO();
        Connection unaConexion = conexion.getConnection();
        if(unaConexion == null){
                return 3;
        } else {
                Statement instruccionSQL = unaConexion.createStatement();
                ResultSet resultadosConsulta = instruccionSQL
                                .executeQuery("SELECT * FROM sn4g_usuarios WHERE
usuario="'+ usuario + "' AND clave_md5 ='" + sb + "'");
                if (resultadosConsulta.first()){
                        if(resultadosConsulta.getInt("habilitado") == 1){
                                conexion.closeConnection();
                                conexion = null;
                                return 1; // usuario validado correctamente
                        } else {
                                conexion.closeConnection();
                                conexion = null;
                                return 2;
                        }
                } else {
                        conexion.closeConnection();
                        conexion = null;
                        return 2; // usuario validado incorrectamente
                }
        }
} catch (Exception e) {
        e.printStackTrace();
        return 2;
}
```

4.4.2.4 Visualización de resultados gráficos

En este capítulo se detalla una de las partes de los resultados, concretamente, los resultados gráficos, donde podremos hacer una comparativa con otras simulaciones, y ver todos los resultados que se han seleccionado para la simulación concreta, de forma especifica los resultados que se pueden obtener en cualquier simulación son: SNR, Cell Spectral Efficiency, Cell Edge user Spectral Efficiency, Througput Per Cell/User y BLER.

Para obtener dichos resultados, en primer lugar, se debe ejecutar la simulación, y a continuación, una vez esta ha finalizado, se tendrá activo el botón de resultados, para poder visualizar todos los gráficos asociados, además se podrán seleccionar varias simulaciones para comparar resultados entre ellas.

Cuando se pulsa el botón de mostrar gráficos, se ejecutan una serie de hilos simultáneos, que se comentará su gestión en la tercera iteración del proyecto, uno por cada gráfico a mostrar y mientras se va llenando una barra de proceso (conforme van finalizando cada uno de los hilos). Cuando todos los hilos han finalizado, la barra de progreso desaparece, quedando una ventana con todos los gráficos asociados a las simulaciones.

Estos hilos, realizan una llamada a la base de datos para obtener los resultados en bruto, y después se produce un calculo de los valores que se mostrarán en los gráficos:

```
Statement instruccionSQL = null;
ResultSet rs = null;
try {
        instruccionSQL = unaConexion.createStatement();
} catch (SQLException e) {
        e.printStackTrace();
}
try {
        String sql = "SELECT id, LTE_RBs, simul_desc, numberOfSeeds,
simulationScenarioId FROM sn4g_synthetic ";
        for (int i = 0; i < Global.SimIdSelIndoor.size(); i++) {
```

```
                    if (i == 0)
                        sql += " WHERE simulationScenarioId = " +
Global.SimIdSelIndoor.get(i);
                    else
                        sql += " OR simulationScenarioId = " +
Global.SimIdSelIndoor.get(i);
                }
                rs = instruccionSQL.executeQuery(sql);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            ArrayList<Graphics> listaceuse = new ArrayList<Graphics>();
            try {
                while (rs.next()) {
                    Graphics syntsimul = new Graphics();
                    syntsimul.setSimul_desc(rs.getString("simul_desc"));
                    // Select and order all the throughput of the simulations
                    // needed.
                    ResultSet rs2 = null;
                    try {
                        instruccionSQL = unaConexion.createStatement();
                    } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                    String sql2 = "SELECT SUM(throughput) AS sumThroughput,
MAX( scheduledTime) AS scheduled, COUNT( DISTINCT userId) AS users FROM
lteThroughput WHERE simulationId="+ rs.getInt("simulationScenarioId");
                    rs2 = instruccionSQL.executeQuery(sql2);
                    if (rs2.first()) {
                        double lteuser = rs2.getDouble("sumThroughput");
                        lteuser = lteuser / rs2.getDouble("users");
                        lteuser = lteuser / rs.getDouble("numberOfSeeds");
                        syntsimul.setLteuser(lteuser);
                    }
                    listaceuse.add(syntsimul);
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
```

```
return listaceuse;
```

Los gráficos que se muestran por pantalla son de tipo gráficos de barras e histogramas principalmente:

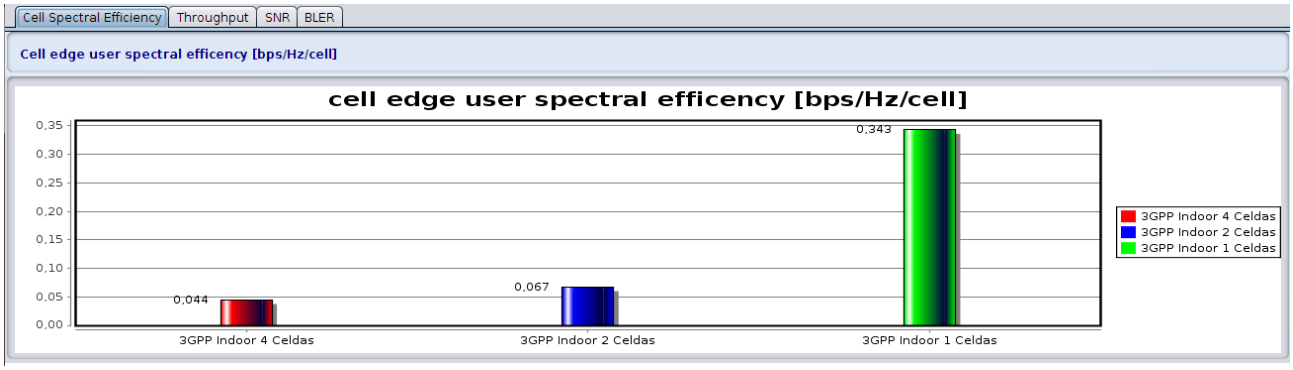

Figura 30: Eficiencia espectral del usuario al borde de la celda

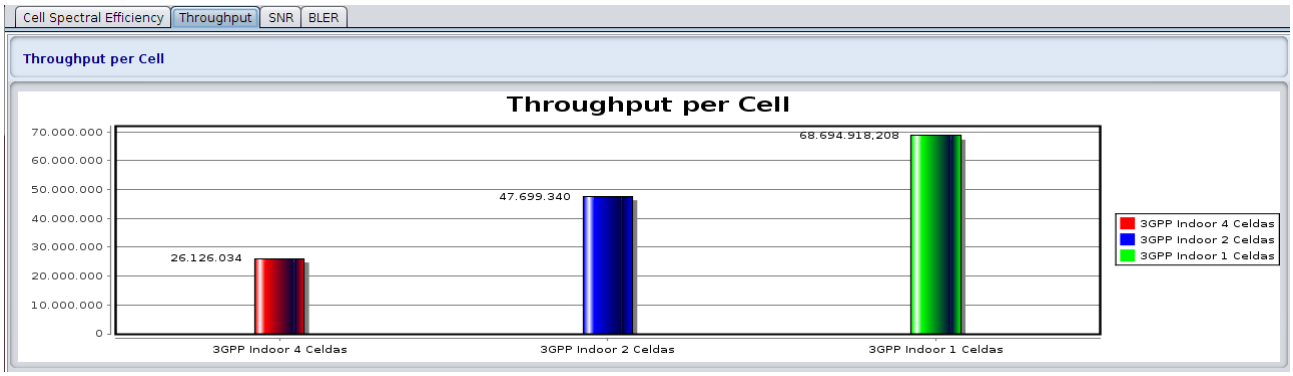

Figura 31: Rendimiento por celda

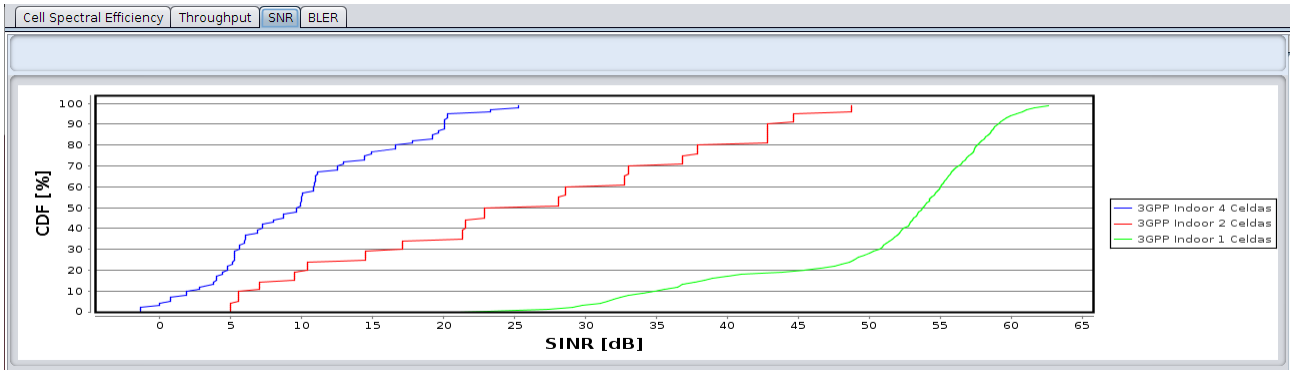

Figura 32: SNR

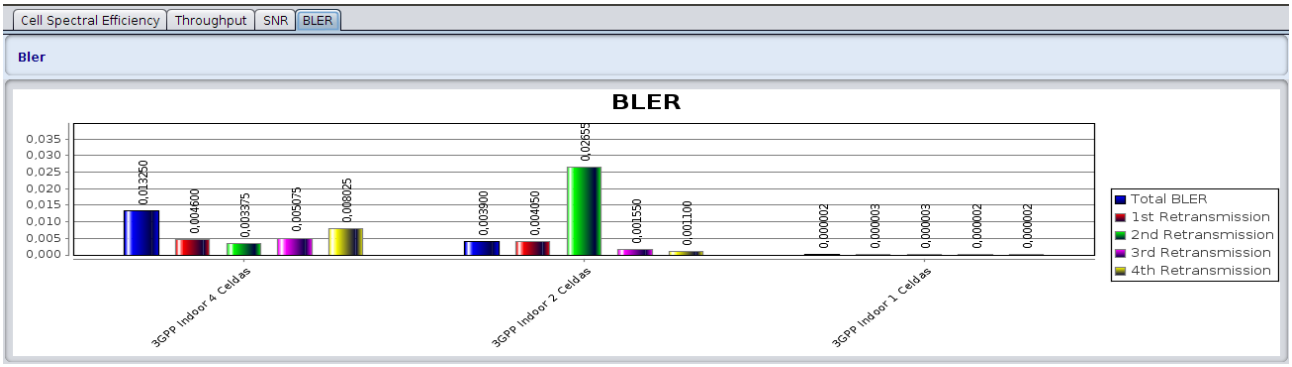

Figura 33: BLER

4.4.3 Tercera Iteración

Finalmente se completa el proyecto con una última iteración, donde se procederá con los detalles finales que se establecieron con el cliente de forma que se termine la aplicación y se puede pasar a la fase final de cierre o transición.

4.4.3.1 Gestión de hilos

Los Hilos o los "Threads" en Java, son básicamente una forma de poder ejecutar varios procesos simultáneamente en nuestros programas en Java.

Usualmente para poder utilizarlos tenemos que crear clases que extienden a la clase Thread, y reescribir el método principal "run()", el cual es el que se va a ejecutar principalmente al iniciar un hilo, thread o nuevo proceso en java.

En el proyecto desarrollado, se han utilizado los hilos para gestionar de forma eficiente acciones donde se requieren varios procesos a la vez, como por ejemplo, el lanzamiento de múltiples simulaciones, el borrado múltiple o la clonación de varias simulaciones de forma simultanea...

Por otro lado, se va a detallar un ejemplo de creación de un hilo, cuando se seleccionan varias simulaciones y se pulsa el botón de borrado, se activara:

```
public void actionPerformed(ActionEvent e) {
    String message = "Do you want to delete multiple selected simulation?";
    int answer = JOptionPane.showConfirmDialog(getRootPane(),message);
    if (answer == JOptionPane.YES_OPTION) {
```

```
            new ThreadMultiDelete().start();
        }
}
```

realizando el borrado de todas las simulaciones en segundo plano:

```
class ThreadMultiDelete extends Thread {

        public ThreadMultiDelete() {
                super();
        }

        public void run() {
                SyntheticDAO syntheticdao = new SyntheticDAO();
                updateTableSimuls();
                List<Integer> SimIdSelSimAux = Global.SimIdSel;
                Global.SimIdSel = new ArrayList<Integer>();
                for (int i = 0; i < SimIdSelSimAux.size(); i++) {
                        syntheticdao.deleteSynthetic(SimIdSelSimAux.get(i));
                }
        }
}
```

4.4.3.2 Visualización de imágenes

Una de las últimas peticiones o requisitos del cliente fue la posibilidad de obtener imágenes de los diferentes mapas donde se ha realizado la simulación, tanto de realista como de sintético. Estos mapas nos indican el SNR y Throughput del mapa donde se realizó dicha simulación, y sirve, entre otras cosas, para obtener información de la eficiencia donde situar las estaciones base.

Estas imágenes son generadas por el core del simulador y se almacenan en una carpeta que identifica la simulación dentro del servidor, por tanto, la interfaz debe encargarse de leerla.

La decisión tomada fue de leer las imágenes la conexión SSH comentada en capítulos anteriores, además de imágenes también se generan ficheros en kml para los escenarios realistas en el exterior, los cuales, deben leerse en la api de Google Maps en el framework de ExtJS.

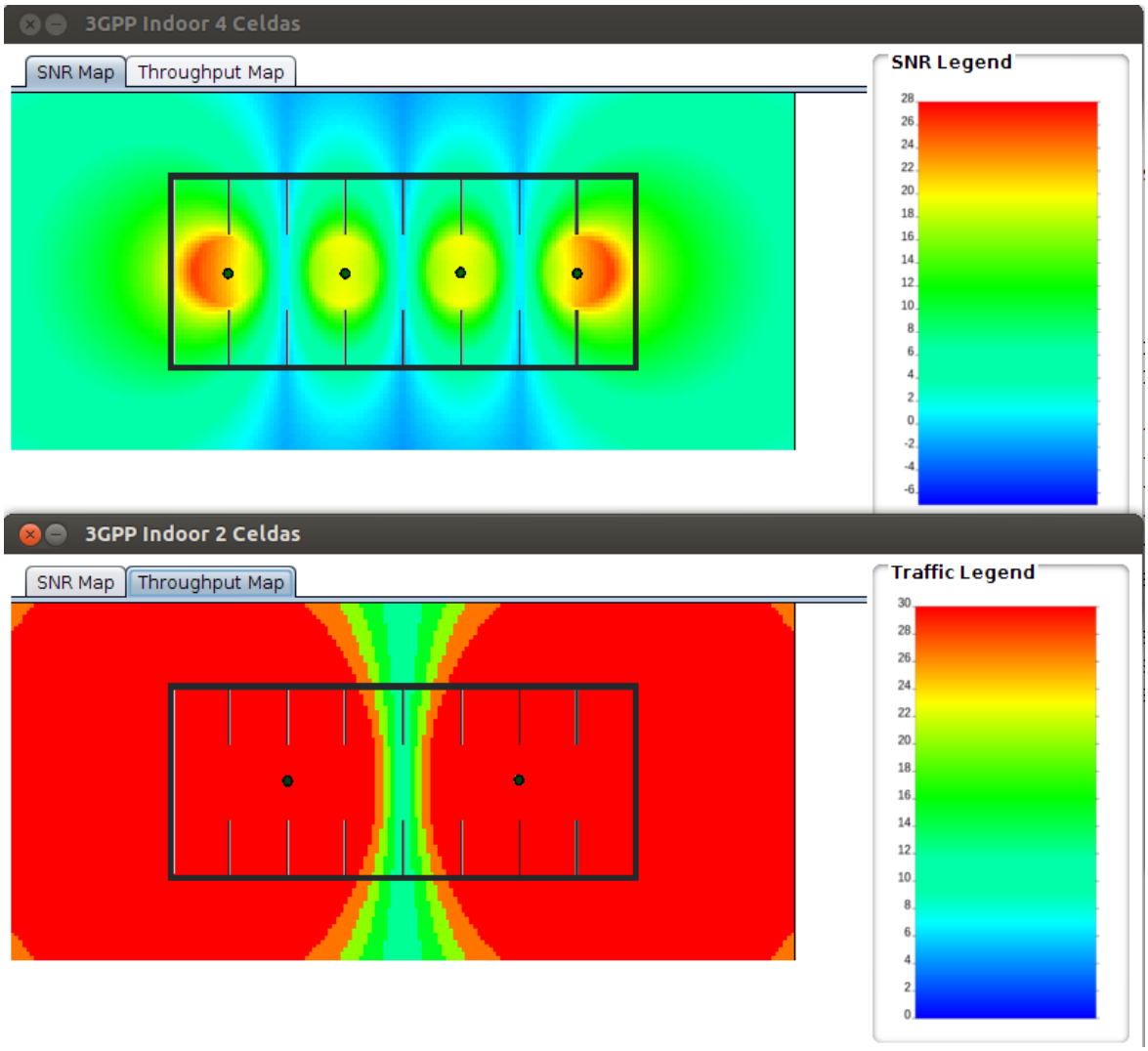

Figura 34: Visualización imágenes sintéticas

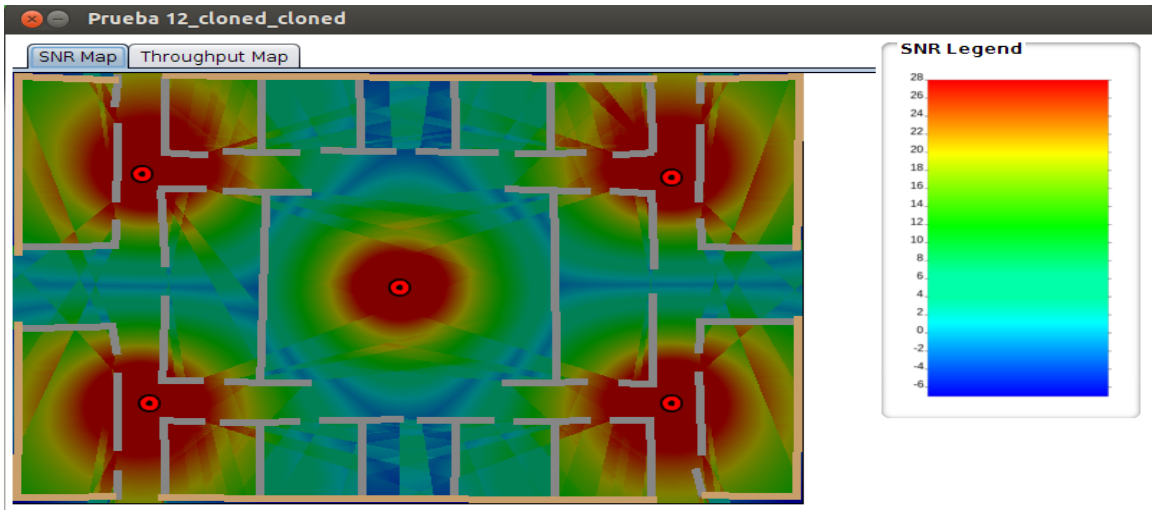

Figura 35: Visualización imágenes realistas interiores

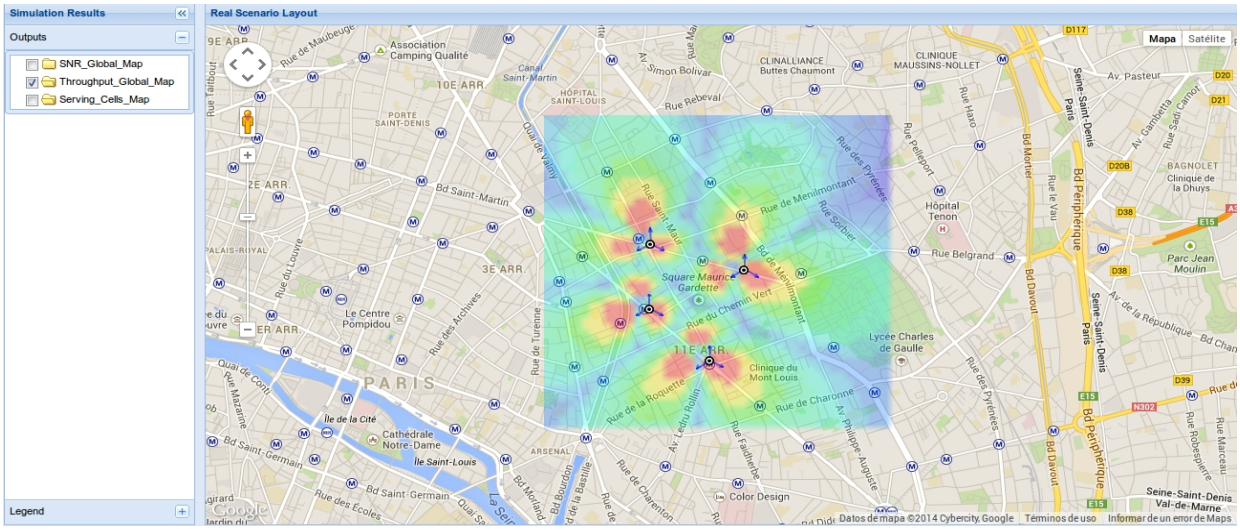

Figura 36: Visualización imágenes realistas exteriores

4.4.3.3 Exportación de resultados

En este capítulo se detalla la necesidad de obtener los resultados en otro formato que sea más tratable por herramientas informáticas, y por lo tanto otro formato que no sea tan visual como los gráficos comentados en capítulos anteriores.

Por lo tanto, se necesito obtener los datos en formato csv, el cual, se pueden tratar mediante hojas de cálculo o abrir directamente con un editor de texto, además existen múltiples “parsers” para tratar los datos escritos en este formato.

	A	B	C	D	E	F	G
1	Simulation Name	Cell edge user spectral efficiency [bps/Hz]	Cell spectral efficiency [bps/Hz/cell]	Throughput per Cell	Throughput per User	Averaged SINR	Data transmissions averaged SINR
2	3GPP Indoor 4 Celdas	0.0443332	1.3063017	2.6126034E7	2612603.4	15.418710371416076	15.415729083208404
3							
4	Simulation Name	Total BLER	1st Retransmission	2nd Retransmission	3rd Retransmission	4th Retransmission	
5	3GPP Indoor 4 Celdas	0.013250000000000001	0.004600000000000001	0.0033750000000000004	0.0050750000000000004	0.0080250000000000001	
6							
7		normalized user throughput [bps/Hz]			SINR [dB]		
8	Percentile	3GPP Indoor 4 Celdas		Percentile	3GPP Indoor 4 Celdas		
9	0	0.0260624		0	-1.3300995467447891		
10	1	0.0260624		1	-1.3300995467447891		
11	2	0.0260624		2	-1.3300995467447891		
12	3	0.0443332		3	0.007192994010254053		
13	4	0.0443332		4	0.007192994010254053		
14	5	0.0451176		5	0.760199835442207		
15	6	0.0451176		6	0.760199835442207		
16	7	0.0451176		7	0.760199835442207		
17	8	0.0508736		8	1.8698538389119004		
18	9	0.0508736		9	1.8698538389119004		
19	10	0.0508736		10	1.8698538389119004		
20	11	0.0558112		11	2.8268635879611863		
21	12	0.0558112		12	2.8268635879611863		
22	13	0.056154		13	3.7736927570411267		
23	14	0.056154		14	3.7736927570411267		
24	15	0.0583864		15	3.969471645810778		
25	16	0.0583864		16	3.969471645810778		
26	17	0.0583864		17	3.969471645810778		
27	18	0.0595444		18	4.440827641491362		
28	19	0.0595444		19	4.440827641491362		
29	20	0.0619752		20	4.756623864876426		
30	21	0.0619752		21	4.756623864876426		
31	22	0.0619752		22	4.756623864876426		
32	23	0.0639184		23	5.095140403102175		
33	24	0.0639184		24	5.095140403102175		
34	25	0.0661656		25	5.23677989834336		
35	26	0.0661656		26	5.23677989834336		
36	27	0.0661656		27	5.23677989834336		
37	28	0.0663868		28	5.241262834636423		
38	29	0.0663868		29	5.241262834636423		
39	30	0.0709144		30	5.611747616832935		
40	31	0.0709144		31	5.611747616832935		
41	32	0.0709144		32	5.611747616832935		
42	33	0.072634		33	5.9657159347716995		
43	34	0.072634		34	5.9657159347716995		
44	35	0.07615		35	6.065994175769967		
45	36	0.07615		36	6.065994175769967		
46	37	0.07615		37	6.065994175769967		
47	38	0.0776196		38	6.866710346457464		
48	39	0.0776196		39	6.866710346457464		
49	40	0.0793968		40	7.270382529140815		
50	41	0.0793968		41	7.270382529140815		
51	42	0.0793968		42	7.270382529140815		
52	43	0.0810412		43	8.036212097151056		
53	44	0.0810412		44	8.036212097151056		
54	45	0.0822256		45	8.741737702631113		
55	46	0.0822256		46	8.741737702631113		
56	47	0.0822256		47	8.741737702631113		
57	48	0.0856612		48	9.664344955764063		
58	49	0.0856612		49	9.664344955764063		
59	50	0.0856612		50	9.664344955764063		

Figura 37: Datos en formato csv

4.4.3.4 Gestión Obstáculos

Finalmente, en la tercera iteración se requiere la necesidad de generar obstáculos en las simulaciones realistas internas, es obvio pensar, que en una sala de oficinas o en una casa, hay elementos que debilitan la cobertura de los elementos que deseamos simular.

Por todo esto, una vez se ha cargado una imagen de la simulación realista interna en la base de datos, se procede a la gestión de las estaciones base, una vez estamos en este punto, se debe fijar la escala, es decir, la relación pixels/metros de la imagen.

A continuación, una vez definida dicha escala, hay dos opciones, en primer lugar se pueden añadir, posicionar y configurar las diferentes estaciones base que se deseen simular, y por otro lado, se pueden añadir obstáculos, además de poder definir qué tipo de obstáculos es y su atenuación, por ejemplo, el ladrillo con una atenuación de 23 dB y que en la interfaz gráfica tenga el color rojo, para una mejor diferenciación entre otros obstáculos.

5 Fase de Cierre

El propósito de esta fase es asegurar que el software esté disponible para los usuarios finales, ajustar los errores y defectos encontrados en las pruebas de aceptación, capacitar a los usuarios y proveer el soporte técnico necesario. Se debe verificar que el producto cumpla con las especificaciones entregadas en el proyecto.

5.1 Manual de Instalación

A continuación se procede a la descripción de la fase de generación de un ejecutable y su instalación en el ordenador del cliente.

En primer lugar, la aplicación debe tener una clase con un método main para generar el fichero ejecutable .jar, además se debe disponer de una configuración de ejecución en nuestro IDE (Eclipse) para esa clase.

A continuación, se debe exportar como un "Runnable Jar File" mediante la opción de "Export" del menú "File", seguidamente se solicita la configuración de ejecución que queremos utilizar, en nuestro caso, será la autenticación del usuario, y, finalmente, como nuestra aplicación incluye otras librerías se selecciona la opción "Extract required libraries into generated JAR

Todo lo que tiene que hacer un usuario para que funcione la interfaz gráfica en su ordenador.

Además cabe decir que los requisitos recomendados para utilizar la interfaz gráfica en uno ordenador personal son las siguientes:

RECOMMENDED HARDWARE (IN-HOUSE SOFTWARE)	
Processor	1 Core (2,3GHz) per Simulation
RAM	1 GB per Simulation (2GB optimal)
Supported OS	Windows and Linux

Figura 38: Requisitos mínimos

Gracias a que el software esta desarrollado en Java, la interfaz gráfica es multiplataforma, por tanto soporta tanto Windows como Linux, con el único requisito de que se debe descargar e instalar Java en cualquiera de los sistemas operativos desde la web oficial de Java.

5.2 Manual de Usuario

Tras la generación del .jar y la instalación de Java en el ordenador personal del usuario, el cliente tan solo necesita unas directrices para utilizar el sistema de mantenimiento de forma fluida.

5.2.1 Selección de la Aplicación

En primer lugar, se puede ejecutar la aplicación mediante dos opciones, desde consola con el comando: java -jar "nombreAplicacion", o haciendo clic derecho en la aplicación y abriendo la aplicación con el Java instalador en el ordenador del usuario.

5.2.2 Autenticación de la Aplicación

Primeramente aparecerá un menú de autenticación de la aplicación, donde se debe indicar el usuario y contraseña para poder acceder al menú principal. Estas credenciales deben estar previamente almacenadas en la base de datos.

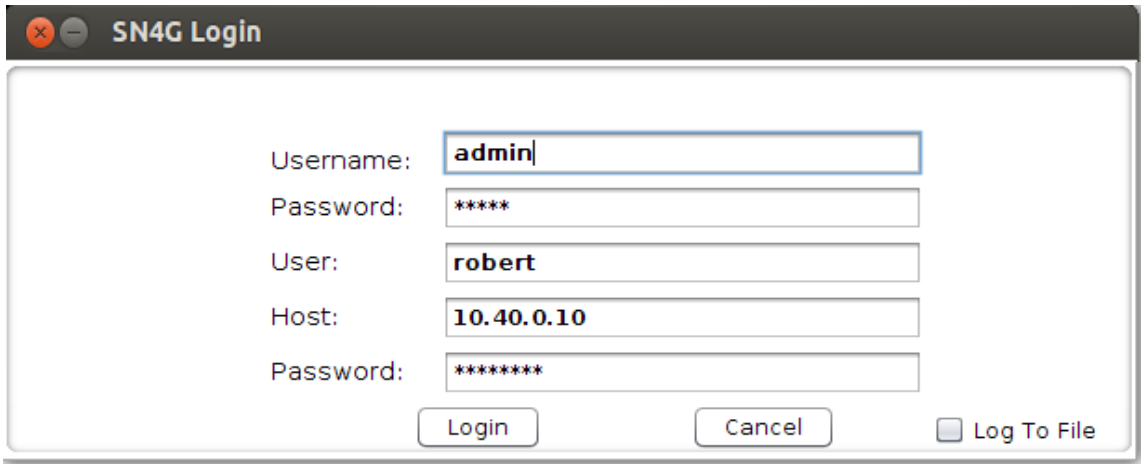

Figura 39: Autenticación

Cuando se haya introducido estos datos y al pulsar sobre el botón de “Login” pueden ocurrir tres estados:

• Mostrar un mensaje de “Write a User and/or Password”, cuando el usuario no ha introducido ningún dato en los apartados correspondientes.

• Mostrar un mensaje de “Access Failure”, cuando el usuario no ha introducido los datos correctamente o no tiene permisos para acceder a la aplicación.

• Acceso con éxito cuando el usuario se ha autenticado con éxito mediante la autenticación con el algoritmo MD5 comprobando si existe dicho usuario y contraseña en la base de datos correspondiente.

5.2.3 Pantalla Principal

Esta pantalla se divide en dos columnas, a la izquierda el menú fijo con las diferentes pestañas para poder navegar en la aplicación de escritorio, y a la derecha el panel que ira cambiando, dependiendo de que pestaña elijamos en cada momento. En la pantalla principal, aparece datos del sistema, licencias...

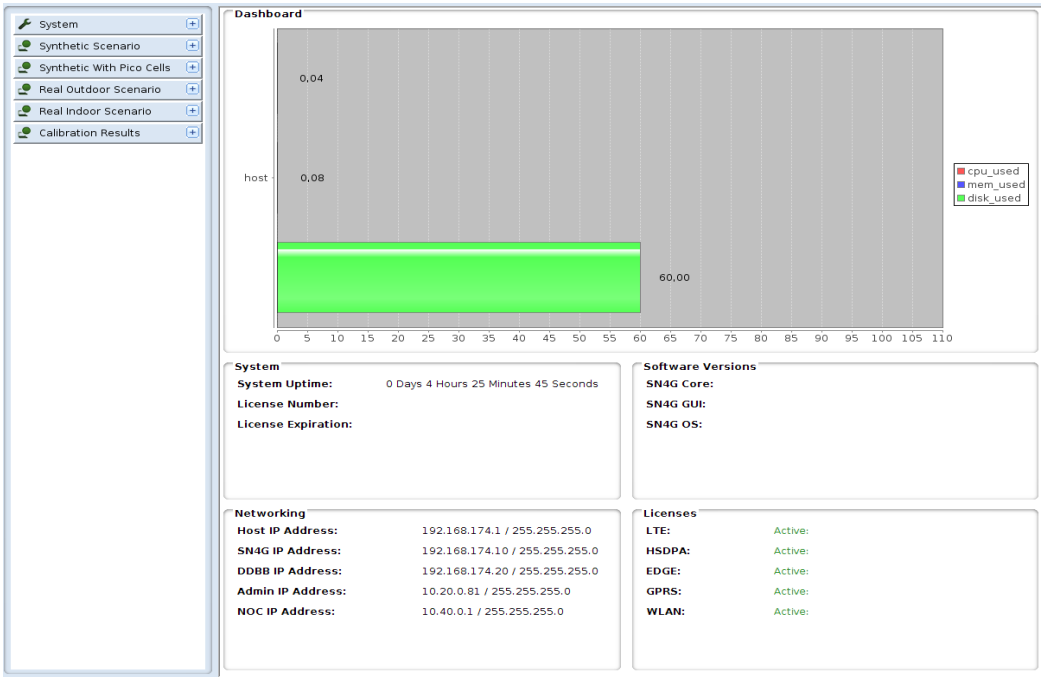

Figura 40: Pantalla principal

5.2.4 Gestión de usuarios

Para acceder a esta pantalla, se debe tener permisos de administración, como se comentó anteriormente, y pulsar sobre la pestaña "System" > "Users", entonces aparece un listado con los usuarios definidos y se puede crear (pulsando en New), actualizar (doble clic en el usuario) y borrar (seleccionando el usuario y pulsando Delete) desde este menú mediante los botones de forma intuitiva:

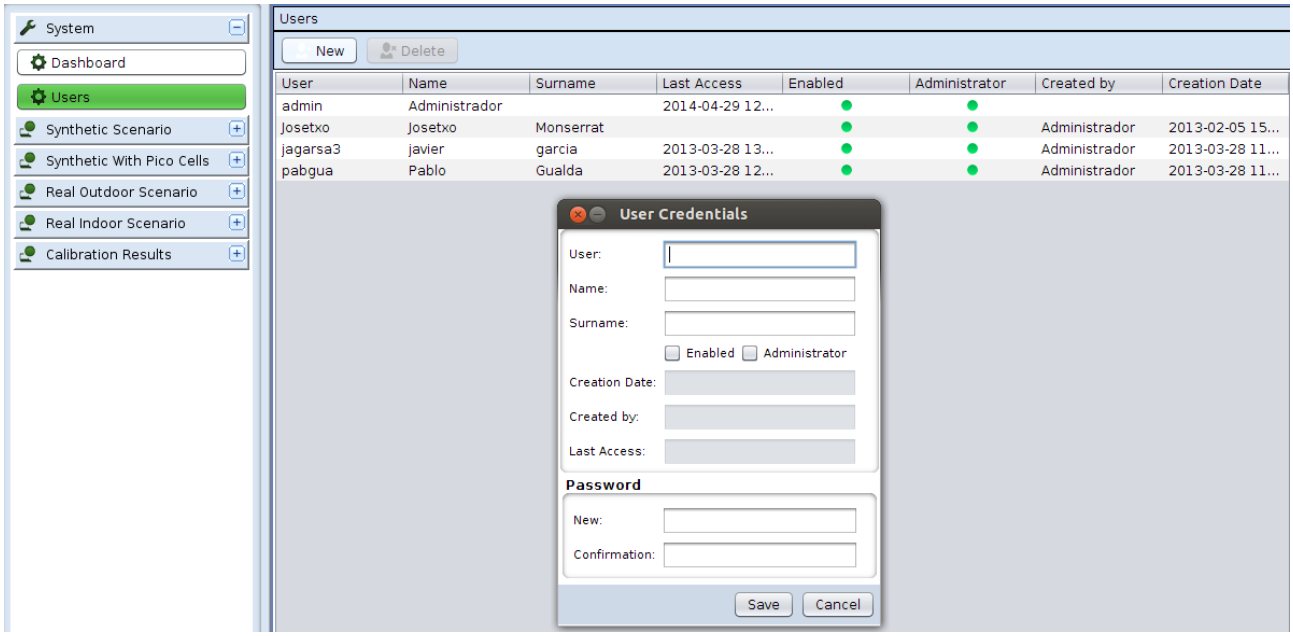

Figura 41: Gestión de usuarios

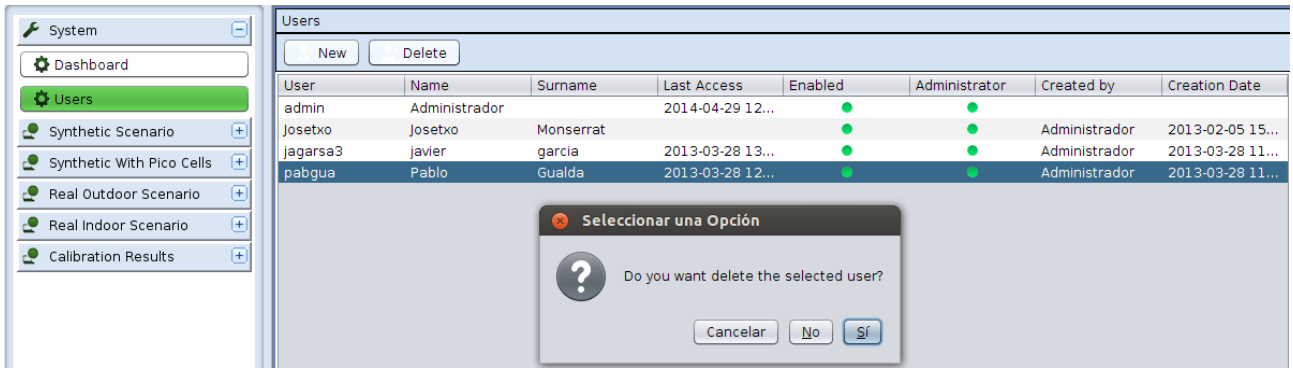

Figura 42: Dialogo de borrado de usuarios

5.2.5 Gestión de simulaciones

A continuación, desde el menú principal se puede acceder a la gestión de simulaciones, dependiendo de que simulación se desee seleccionar. Si se quiere utilizar una simulación sintética se accederá a "Synthetic Scenario" > "Simulation Config & Results" y tendremos las diferentes opciones como se puede observar en la figura:

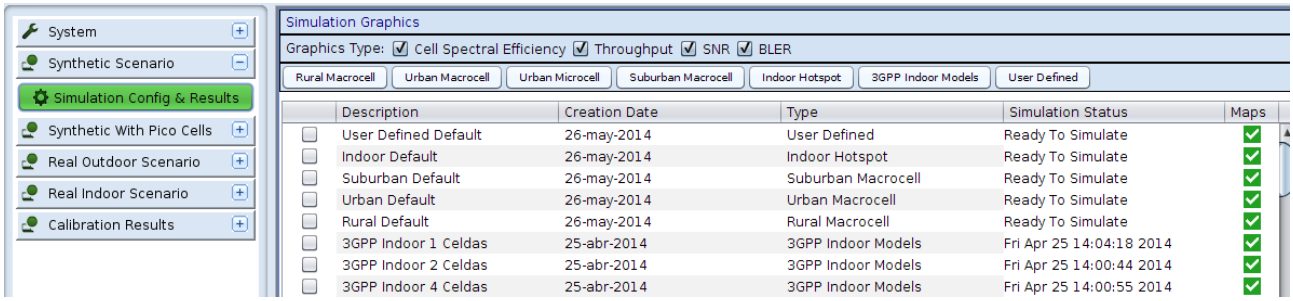

Figura 43: Gestión de simulaciones

Además si se quiere crear una simulación, se debe elegir la simulación adecuada y hacer clic en el botón, entonces aparecerá una ventana para la configuración:

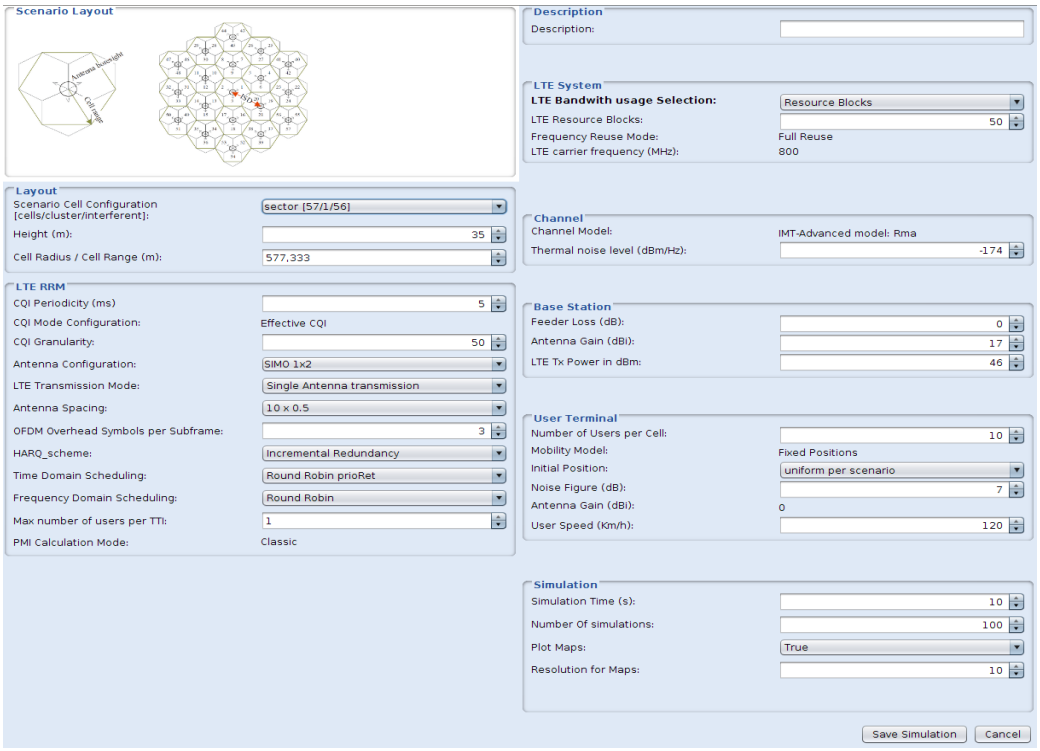

Figura 44: Formulario configuración simulaciones

También se pueden editar simulaciones, haciendo doble clic sobre la simulación elegida, o borrar una simulación, haciendo clic sobre ella y pulsando el botón Delete (también habilitada la opción de borrado, clonación, lanzamiento y parada múltiple debido a los hilos en Java, del mismo modo).

Por otro lado, se tienen dos opciones para la gestión de simulaciones ya creadas, por un lado haciendo clic con el botón derecho, aparece un menú con las opciones habilitadas para la simulación, o como se indica en el párrafo superior, haciendo clic en el botón indicado como se puede observar en la figura:

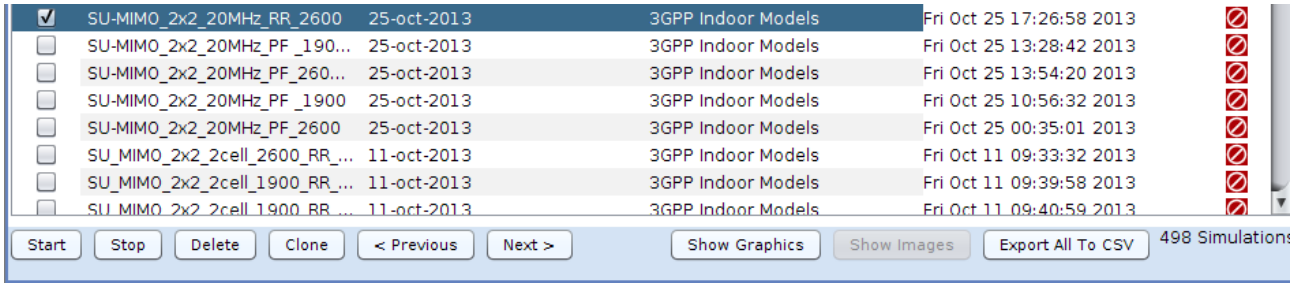

Figura 45: Botones gestión simulaciones

Si se quiere utilizar una simulación sintética con celdas de tipo Pico se accederá a "Synthetic With Pico Cells" > "Simulation Config & Results" y tendremos las diferentes opciones como se puede observar en la figura:

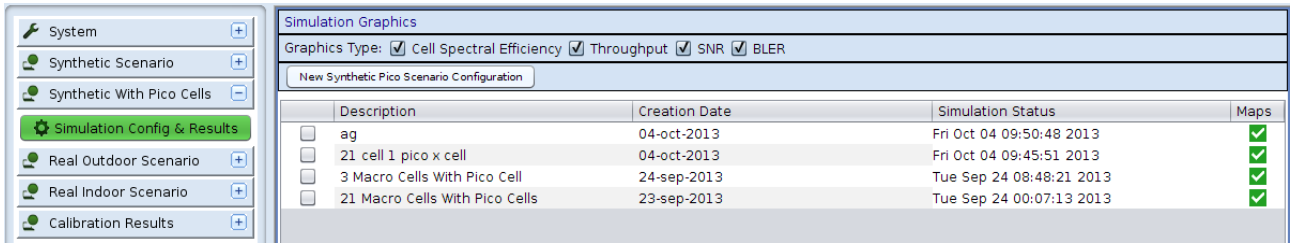

Figura 46: Listado simulaciones con celdas pico

Este caso, tiene las opciones similares a la gestión de las simulaciones sintéticas, por tanto, no se va a detallar de forma más extensa.

Si se quiere utilizar una simulación real en un escenario en el exterior se accederá a "Real Outdoor Scenario" > "Simulation Config & Results" y tendremos las diferentes opciones visibles en la figura siguiente.

En este listado se puede observar un elemento diferente, y es que también se listan las diferentes celdas que se añaden en cada simulación, pulsando sobre esta. Esto es debido porque en estos casos, cada simulación puede tener un número de celdas diferentes y localizadas en puntos diferentes.

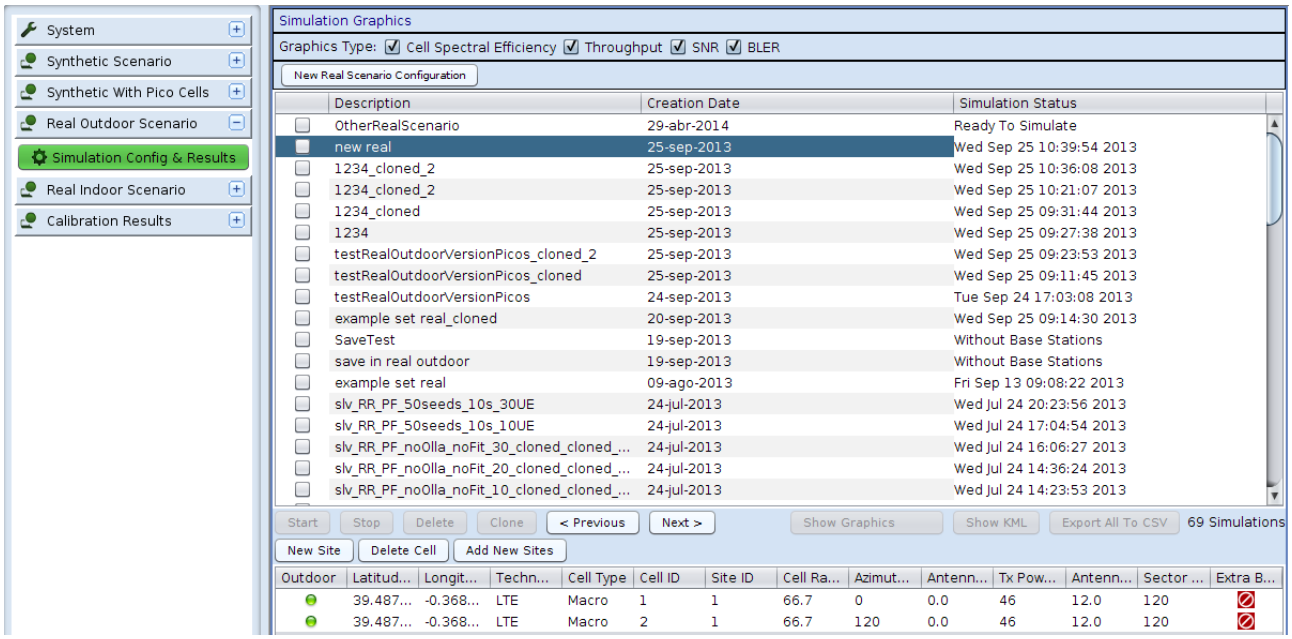

Figura 47: Listado simulaciones realistas externas

Una vez se ha creado la simulación, para añadir y posicionar las celdas, se debe hacer clic en la simulación correspondiente y sobre el botón Add New Sites, entonces el navegador nos muestra la api Google Maps en el framework ExtJS en el cual se realizará toda la parte de gestión de estaciones base:

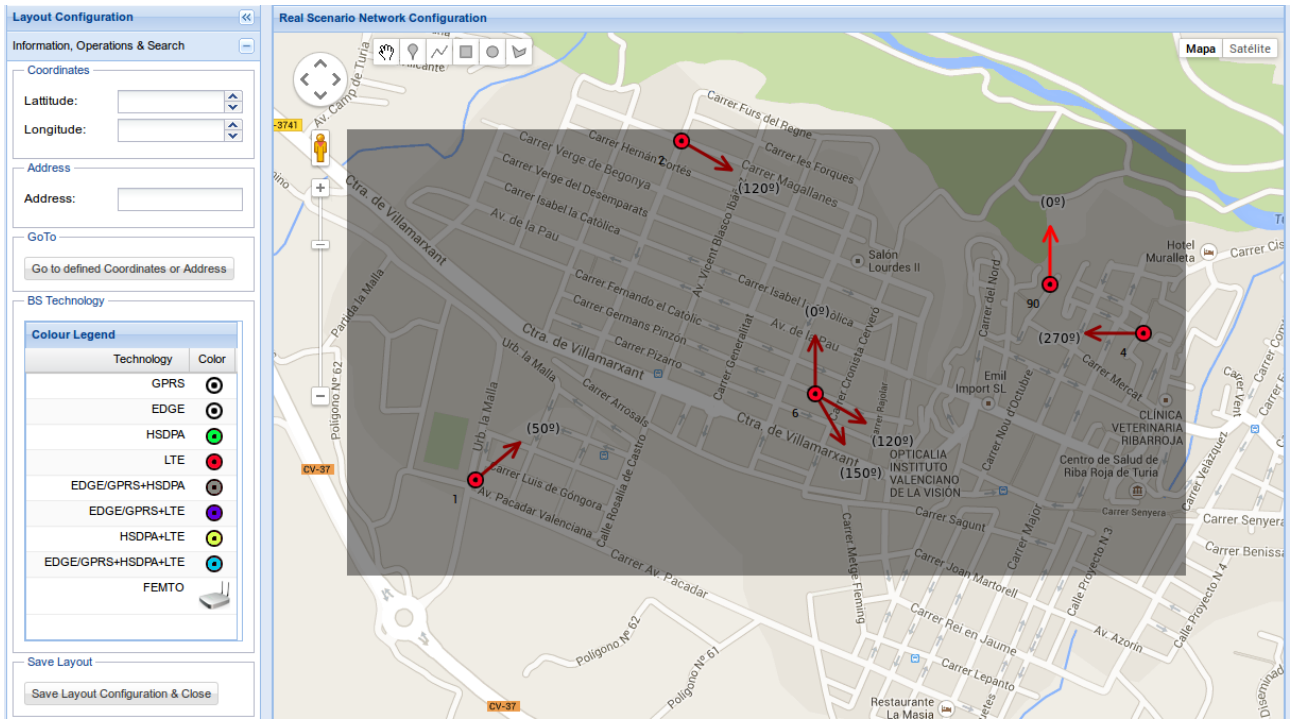

Figura 48: Gestión de estaciones base externas

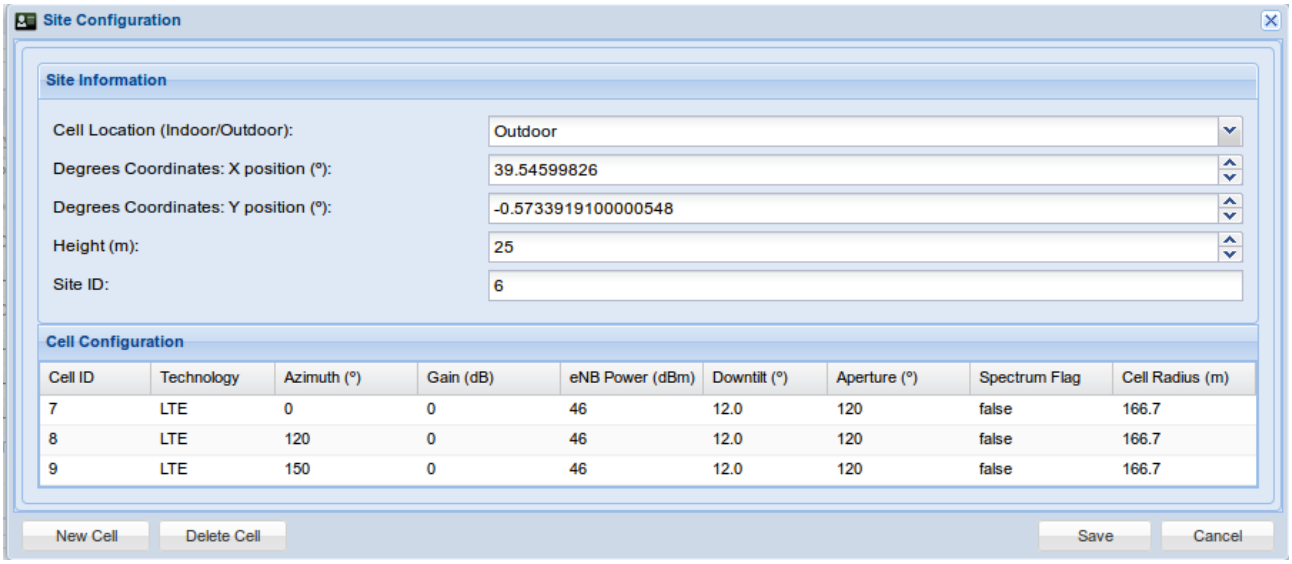

Figura 49: Formulario configuración estaciones base

Una vez añadidas todas las estaciones base, se realiza un rectángulo para acotar el área a simular, como se puede observar en la figura anterior, para finalmente ejecutar dicha simulación.

Si se quiere utilizar una simulación real en un escenario en el interior se accederá a "Real Indoor Scenario" > "Simulation Config & Results" y tendremos las diferentes opciones visibles en la figura siguiente. En este caso también se puede observar el listado de estaciones base de cada simulación.

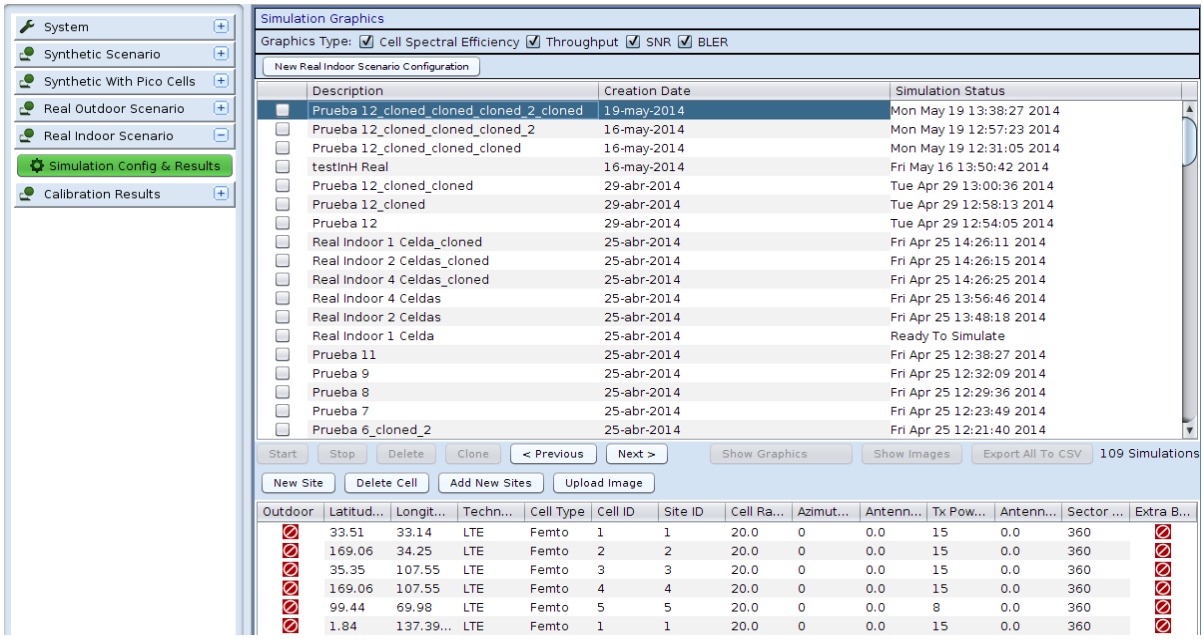

Figura 50: Listado de simulaciones realistas interiores

Una vez se ha creado la simulación, para añadir y posicionar las celdas, se debe subir una imagen de la sala interior que se quiere simular y situar todas las estaciones base y obstáculos en ella:

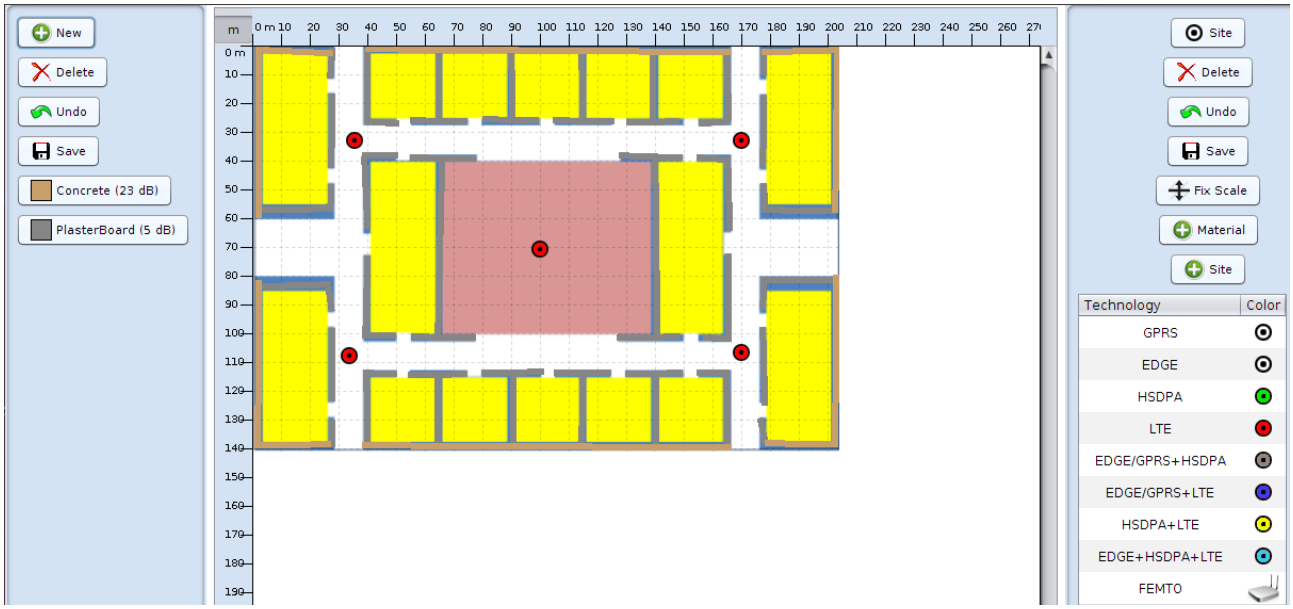

Figura 51: Configuración estaciones base interiores

Las estaciones base se configuraran de forma similar a las simulaciones externas realistas, con un ventana en modo de formulario haciendo clic en cada estación base, y los obstáculos se definirán haciendo clic en cada obstáculo y añadiéndolo en la imagen de la simulación.

5.2.6 Visualización de gráficos

Una vez configuradas las simulaciones, se procede a su lanzamiento mediante cualquiera de las dos opciones comentadas anteriormente, haciendo clic derecho en la simulación o seleccionando la simulación y pulsando el botón “Start”:

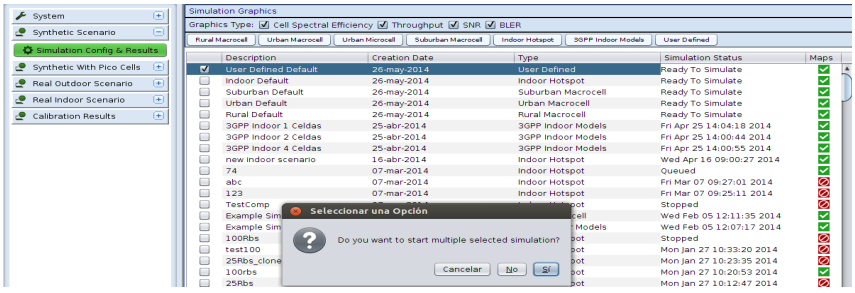

Figura 52: Dialogo de lanzamiento de simulaciones

Una vez lanzada la simulación, aparece una barra de progreso que indicará el tiempo que falta para finalizar la simulación:

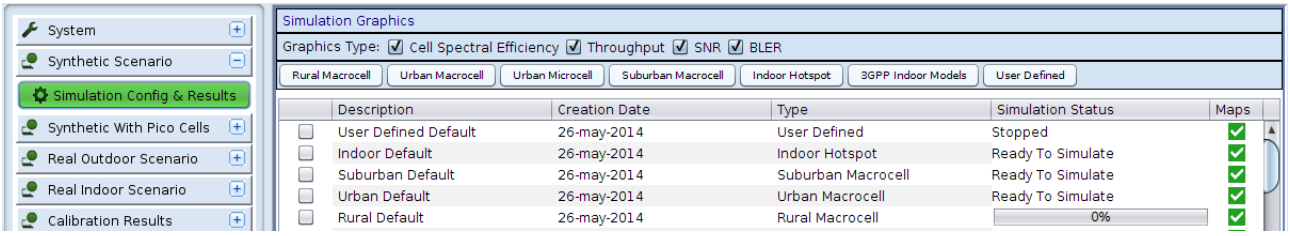

Figura 53: Barra de progreso

Una vez ha finalizado la simulación, se incluirá la fecha de finalización en el campo "Simulation Status" para tener un indicador de lo que ha tardado en simular, y por otro lado, cuando se seleccione dicha simulación ya se habilitarán los botones de resultados, concretamente interesan los gráficos:

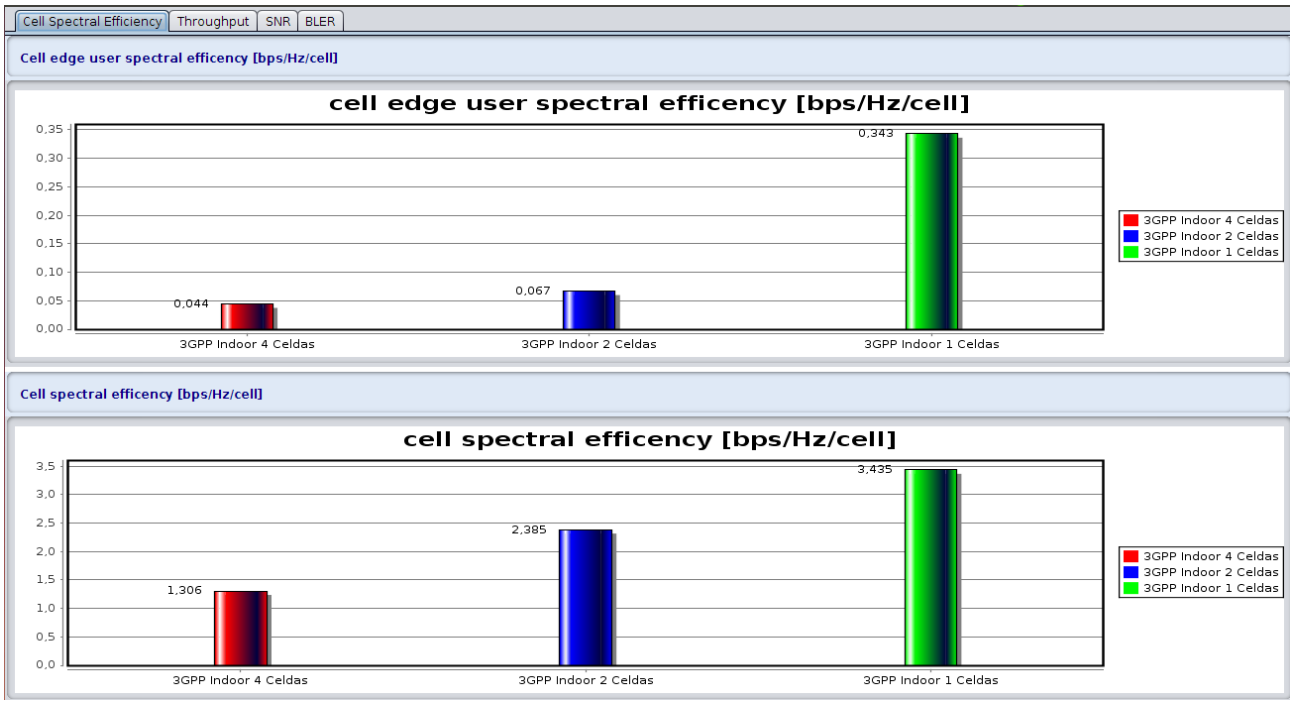

Figura 54: Visualización de gráficos

5.2.7 Visualización de imágenes

Finalmente, también se pueden visualizar las imágenes que se generan después de la simulación, como se pueden observar las figuras del capítulo 4.4.3.2 de la tesis.

6 Trabajo Futuro

Tanto la aplicación como la api de Google Maps en el framework ExtJS están preparados y programados para facilitar el diseño e implementación de nuevas funcionalidades en un futuro para el simulador en la empresa SistelNetworks.

Este sistema de simulación es el esqueleto de un sistema que se incrementará en el futuro según las necesidades de los diferentes clientes y operadoras de la empresa, con lo que se pretende diseñar para informatizar toda esta gestión y que además sea de fluida e intuitiva.

Una posible e inmediata ampliación para el simulador es la de cambiar la parte de configuración de las simulaciones de forma más eficiente, es decir, que la configuración de simulaciones esté dividida por partes y cuando un usuario vaya creando cada parte, pueda reutilizarla en otras simulaciones donde esa parte sea idéntica, de esta forma se esta siendo más eficiente tanto en el almacenamiento en base de datos como en la interfaz gráfica.

Por otro lado, otra ampliación con respecto a las simulaciones podría ser la de tener la posibilidad, depende del dinero que quiera poner el cliente, utilizar varios servidores para realizar las simulaciones, y todos esto se pueda configurar y gestionar desde la interfaz gráfica.

Además, una ampliación significativa sería la de crear una aplicación web que tenga la misma funcionalidad que la interfaz gráfica de escritorio, ya que con la aplicación web se podría utilizar dicha interfaz vía móvil o desde cualquier ordenador. Además se podría almacenar todos los datos, tanto las bases de datos como las imágenes, kml... generados en el “cloud” de la empresa, o contratar una tercera empresa que se encargue de gestionar el “cloud” y pagar por lo que se gasta, siendo así muchos más eficiente y no teniendo que comprar toda la infraestructura, para quizas luego, no tener el éxito esperado.

Finalmente, se puede realizar una ampliación para desarrollar servicios web, en lugar de tener que realizar conexiones SSH, dado que esto es bastante más seguro y accesible.

7 Conclusión

En conclusión, este proyecto realizado durante todo este tiempo en el Instituto de Telecomunicaciones y Aplicaciones Multimedia de Valencia ha servido de una continuación en el mercado laboral utilizando los conceptos aprendidos en la Universidad Politécnica de Valencia, concretamente en el Máster de métodos formales, ingeniería del software y sistemas de información. Además ha sido muy gratificante poder colaborar con la empresa SistelNetworks para realizar la interfaz gráfica del simulador SN4G.

En primer lugar, puntualizar que todos los objetivos que se han desarrollado en la fase de inicio han sido llevados a cabo con éxito, gracias a la motivación de que el sistema ha sido requerido por el mismo SEO de la empresa SistelNetworks y sobretodo a que el proyecto era de cierta importancia debido a que se quiere comercializar de forma inmediata.

Por otro lado, este proyecto resulta innovador para la empresa, aunque existen algunos software de pago, se requería una aplicación mucho más detallada, que simule con una alta eficiencia y efectividad y con una fuerte carga de investigación a las espaldas y testing acerca de todos los resultados obtenidos.

El proyecto también requiere de cierto personal, sobretodo trabajadores de telefonías móviles, que pueda gestionar las simulaciones con esta interfaz de escritorio, por tanto, ha sido muy complaciente el poder incrementar los puestos de trabajo y todo ello gracias a un software que ayudara a las compañías móviles a reducir su enorme despliegue de estaciones base, dado que este simulador se encarga de indicarle a dichas compañías cuales son los puntos más eficiente para colocar las estaciones base y con que configuración para maximizar la cobertura y el servicio para los usuarios que tengan contratados.

Finalmente, desde el punto de vista personal, he podido aprender a programar en un lenguaje multiplataforma utilizando una metodología iterativa e incremental como es (RUP) y plasmándolo en la memoria del proyecto, además de estar en contacto constante con el cliente y la empresa colaboradora, realizando reuniones y obteniendo requisitos de forma constante para la aplicación a través de ellas.

8 Bibliografía

[1] iTEAM, http://www.iteam.upv.es/

[2] SistelNetworks: http://sistelnetworks.com/

[3] Dan Pilone, Neil Pitman, UML 2.0 in a Nutshell. Ed: O'Reilly, June 2005

[4] Hans - Erik Eriksson, UML 2.0 Toolkit. Ed: Wiley 2004.

[5] Desarrollo Iterativo: http://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente accedido en Mayo 2014

[6] Desarrollo CRUD: http://es.wikipedia.org/wiki/CRUD accedido en Marzo 2014

[7] Sencha Ext JS: http://www.sencha.com/products/extjs/ accedido en Febrero 2014

[8] Java Look&Field accedido en Marzo 2014:
http://www.java2s.com/Product/Java/Swing/Look-And-Feel-LaF.htm

[9] Hibernate accedido en Mayo 2014:
http://www.javatutoriales.com/2009/05/hibernate-parte-1-persistiendo-objetos.html

[10] Jsch: http://www.jcraft.com/jsch/ accedido en Abril 2014

[11] JfreeChart: http://www.jfree.org/jfreechart/ accedido en Febrero 2014

[12] Google Maps con ExtJS accedido en Marzo 2014:
http://www.sencha.com/blog/integrating-google-maps-api-with-ext-js/