Universidad del Bío-Bío
Facultad de Ciencias Empresariales
Departamento de Sistemas de Información

Simulador gráfico de servicios basados en localización que opera sobre una red móvil P2P inalámbrica.

Proyecto de Título para optar al título profesional de
Ingeniería Civil en Informática

Autora
Betty Jane Sepúlveda Zúñiga
Profesor Guía
Patricio Galdames Sepúlveda

RESUMEN

Este proyecto da conformidad a los requisitos exigidos por la Universidad del Bío – Bío en el proceso de titulación de la carrera Ingeniería Civil Informática.

El proyecto titulado "Simulador grafico de servicios basados en localización que opera sobre una red móvil P2P inalámbrica", tiene como objetivo principal la simulación grafica del procedimiento de consultas basadas en la localización y consultas de localización sobre una red p2p móvil.

El simulador desarrollado configura la cantidad de usuario, permite seleccionar un usuario que desee localizar, permite la implementación de movimientos random, entre otras características.

Este simulador fue creado en Java, por lo tanto puede ser ejecutado en diferentes plataformas.

ABSTRACT

This project complies with the requirements demanded by the University of Bío - Bío in the process of titling the Civil Engineering Computer Science degree.

The project entitled "Graphical simulator of location-based services operating over a wireless P2P wireless network", has as main objective the graphic simulation of the procedure of consultations based on location and location queries on a mobile p2p network.

The developed simulator configures the amount of user, allows to select a user that wants to locate, allows the implementation of random movements, among other characteristics.

This simulator was created in Java, so it can be executed on different platforms.

INDICE GENERAL

INDICE DE TABLA

INDICE DE FIGURAS

INTRODUCCIÓN

Los Servicios Basados en la Localización de los usuarios (SBL) son una nueva gama de servicios disponibles para usuarios de redes inalámbricas. Los SBL emplean la ubicación de los usuarios para proporcionarles a estos la información relevante a su localización. Por ejemplo, un conductor con cierto apuro para llegar a su casa desea conocer la densidad de tráfico en alguna zona, como en alguna rotonda, con el fin de evitar una zona congestionada. Otro ejemplo de LBS consiste en localizar al amigo o el lugar de asistencia pública más cercano a mi paradero.

La típica arquitectura de un SBL consiste de uno o más servidores dedicados instalados en la Internet que manejan un mapa de los lugares de interés y/o la ubicación de los usuarios del sistema. Esta última información es obtenida de los mismos usuarios quienes le reportan explícitamente su ubicación o mediante servidores de localización que mediante técnicas de triangulación determinan la ubicación de los usuarios. Sin la menor duda, los SBL han demostrado ser servicios de gran utilidad en un sin número de situaciones especialmente en aquellas de emergencia médica.

Sin embargo, el acceso a la infraestructura que soporta a los SBL pudiera no estar siempre presente o bien simplemente no existir. Por ejemplo, luego de un cataclismo natural como un terremoto o tornado, la infraestructura de acceso a la Internet pública pudiera estar seriamente comprometida o no existir. Otro ejemplo menos severo surge cuando un usuario se encuentra en una zona aislada, en la que acusa la falta de señal para conectase a la red. Por lo tanto, resulta interesante explorar otros tipos de infraestructuras de comunicación alternativas.

Una de ellas corresponde a formar una red de comunicación ad hoc entre los mismos dispositivos móviles de comunicación que emplean estos usuarios. Un red ad hoc móvil, es una infraestructura de comunicación en la que los mismos usuarios de forma colaborativamente ejecutan los servicios de comunicación. Implementar un SBL sobre esta red presenta diversos desafíos de conectividad, dependencia energética, etc.

En este trabajo de titulo se busca desarrollar un software de simulación grafica de SBL que operan sobre una red ad hoc móvil. Esta implementación se

basa en la técnica publicada por Galdames et Al. [1]. El simulador grafico desarrollado presenta diversas características:

- Implementa dos tipos de modelos movimientos de los usuarios: Randow Walk y Random WayPoint
- Implementa dos tipos de servicios: Servicio de resolución de consultas de densidad de tráfico en un área geográfica especifica (S-RMQ) y un Servicio de Localización (SLOC).

Este documento cuenta de 9 capítulos, los cuales están constituidos de acuerdo a la documentación de proyectos de desarrollo de software definido por el departamento de Sistemas de Información de la Universidad del Bío-Bío. El contenido de cada capítulo se describe a continuación:

- **CAPITULO 2: "DEFINICION DEL PROYECTO".** Este capítulo incluye lo relativo al proyecto en sí, los objetivos generales y específicos, la metodología utilizada y la descripción de la técnica utilizada para cumplir tales objetivos.

- **CAPITULO 3: "ANTECEDENTES GENERALES".** Se describe las tecnologías que se ven implicadas en el proyecto, tales como las consultas de Localización y consultas S-RMQ, además como los movimientos a desarrollar el random walk y waypoint, y algunos antecedes importantes con respecto a tecnología que ayudan a desarrollar esta teoría.

- **CAPITULO 4:"FACTIBILIDAD".** se detalla de la disponibilidad de los recursos necesarios para llevar a cabo los objetivos del proyecto.

- **CAPITILO 5: "ESPECIFICACION DE REQUERIMIENTOS DEL SOFTWARE".** Se detalla lo relativo a la aplicación, como sus alcances, objetivos generales y específicos, los requisitos para el correcto funcionamiento de la aplicación y la descripción global del software.

- **CAPITULO 6: "ANALISIS".** En este capítulo se presenta el comportamiento y funcionamiento del software utilizando herramientas de modelado.

- **CAPITULO 7: "SIMULACION".** En este capítulo se presenta el diseño a pequeña escala de la simulación y sus diversas ventanas, como sus funcionamientos.

- **CAPITULO 8: "CONCLUSIONES".** En este capítulo se realiza la contrastación de los objetivos del proyecto y del sistema planteados y alcanzado al final del proyecto. Se incluyen conclusiones generales del proyecto desde los puntos de vista académico y personal.

- **CAPITULO 9: "BIBLIOGRAFIA".** Se indican las fuentes de información utilizadas durante el desarrollo del software.

2. DEFINICIÓN DE PROYECTO

2.1 Objetivo Del Proyecto

2.1.1 Objetivo General

Desarrollar la implementación un simulador grafico que muestre el procesamiento de consultas basadas en la ubicación en una red P2P móvil.

2.1.2 Objetivo Específicos

- ✓ Estudiar el trabajo [1], donde se presenta como se procesa en una red P2P móvil, consultas como S-RMQ.

- ✓ Extender el trabajo [1] para incluir un servicio de localización de usuarios móviles.

- ✓ Estudiar diversos modelos de movimiento de usuarios tales como el random walk o random waypoint.

- ✓ Desarrollar un simulador gráfico de una red P2P que permita seleccionar el modelo de movimiento de los usuarios, el tipo de servicio que se desea simular y simular la transmisión de la información requerida para responder una consulta.

2.2 Ambiente de la Ingeniería de Software

2.2.1 Metodología de desarrollo

La metodología descriptiva cualitativa es la que se utiliza en este proyecto debido a que su meta no se limita a la recolección de datos, sino a la descripción, predicción e identificación subjetiva de la investigación de consultas topológicas fija y móvil.

2.2.2 Herramientas de apoyo al desarrollo de software

Para el desarrollo del proyecto se utilizaran las siguientes herramientas:

NetBeans IDE 8.2: Es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java.

Power Designer 16: Software utilizado para la creación de los distintos diagramas presentados en el informe.

2.3 Definiciones, Siglas y Abreviaciones

2.3.1 Definiciones

- **WIFI:** Mecanismo de conexión de dispositivos electrónicos de forma inalámbrica**.** Los dispositivos habilitados con Wi-Fi pueden conectarse a internet a través de un punto de acceso de red inalámbrica [9].

- **SMARTHPHONES:** Término (en inglés) utilizado para referirse a un teléfono móvil inteligente dentro del presente proyecto. [11]

- **TABLET:** Término (en inglés) utilizado dentro del presente proyecto para referirse a un computador portátil táctil que no requiere ratón ni teclado físico [10]

- **NODOS:** es un punto de intersección, conexión o unión de varios elementos que confluyen en el mismo lugar [12].

- **JAVA:** es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. [13]

2.3.2 Siglas y Abreviaciones

- **AP:** *Access Point*; Punto de Acceso en español. Un dispositivo de red que interconecta equipos de comunicación alámbrica para formar una red inalámbrica que interconecta dispositivos móviles o con tarjetas de red inalámbricas [8].

- **IDE:** Entorno de desarrollo integrado (Integrated development environment), es un programa informático que consiste en un conjunto de herramientas para facilitar la edición de código para uno o varios lenguajes de programación, como también la compilación y depuración del código. [14]

- **P2P:** Peer-to-peer, red punto a punto es una red de computadoras en la que todas actúan como nodos, sin clientes ni servidores, para compartir archivos. [15]

- **LBS:** Location Based Services o LDIS (Location Dependent Information Services) hacen referencia a Servicios Basados en Localización o para algunos autores simplemente servicios de localización. [16]

- **WLAN:** *Wireless Local Area Network*; Red de Área Local Inalámbrica en español. Red inalámbrica local cuyo dispositivos no necesitan estar vinculados a través de cables para conectarse. [17]

- **3G:** es la abreviación de tercera generación de transmisión de voz y datos a través de telefonía móvil mediante UMTS (*Universal Mobile Telecommunications System* o servicio universal de telecomunicaciones móviles). [18]

- **4G:** La 4G está basada completamente en el protocolo IP, siendo un sistema y una red, que se alcanza gracias a la convergencia entre las redes de cable e inalámbricas.[19]

- **S-RMQ:** *Stationary Range Monitoring Query [1],* es un conjunto de nodos móviles que están dentro de una región geográfica definida por el usuario y proporciona actualizaciones en tiempo real cada vez que un nodo cruza sobre el borde de la región. A medida que los nodos se mueven, los resultados de la consulta pueden seguir cambiando.

- **RW:** Random Walk (en español: "Caminata Aleatoria"), uno de los modelos de movimiento estudiados en este proyecto.

- **RWP:** Random Waypoint (en español: "Punto de recorrido Aleatorio"), uno de los modelos de movimiento estudiados en este proyecto.

3. ANTECEDENTES GENERALES

En este capítulo explicaremos los siguientes conceptos que se desea implementare en la simulación

3.1 SBL en redes P2P inalámbricas.

Los servicios basados en la localización o SBL son servicios online que utilizan la ubicación del usuario para ofrecerle la información más relevante de acuerdo al lugar donde este se encuentra. Pertenecen por tanto a la categoría de servicios de información y poseen hoy en día un sinfín de usos. Aquel ejemplo más popular o conocido es el juego PokémonGO que es una realidad aumentada basada en la localización []. Este juego es un servicio de entretención que se ofrece a los usuarios de redes inalámbricas (Wi-Fi[9], 3G[18], 4G[19]), el cual consiste en buscar y capturar personaje de la saga Pokémon escondidos en ubicaciones del mundo real y luchar con ellos, lo que implica desplazarse físicamente por las calles de la ciudad para progresar de nivel. Este juego requiere del uso de un dispositivo móvil, como un Smartphone[11] y Tablet[10],y necesita conocer la posición geográfica del jugador, la que es obtenida por medio del GPS instalado en el teléfono, o de la misma red inalámbrica.

Otras áreas y contextos de uso de los SBL son por ejemplo, una aplicación que sea capaz de confirmarme si un amigo o familiar se encuentra en un lugar en particular (estas aplicaciones son conocidas como *FriendFinder*), ubicar la clínica de salud u hospital más cercano al usuario, o bien la estación de gasolina que se encuentra más cerca, el restaurant de comida China que se encuentra más cerca, etc.

Vamos a clasificar los SBL en 4 tipos de servicios de acuerdo a si los datos de interés (lugares, personas) son móviles o no o bien el usuario que consulta está o no en constante movimiento.

- **Servicio de Monitoreo de Rango Estacionario (S-RMQ):** Este servicio consiste en encontrar un conjunto de usuarios o nodos móviles que están dentro de una región geográfica fija declarada por el usuario. Este servicio constantemente proporciona actualizaciones en tiempo real cada vez que un nuevo usuario/nodo ingresa a la región o bien cuando un nodo sale de ella. Por tanto, a medida que los nodos se mueven, los resultados de la consulta cambiar.

Por ejemplo, el ministerio de Transporte desea saber la densidad de tráfico que circula en una rotonda. Para ello requiere contar la cantidad de vehículos que salen y entran a dicha rotonda. Esto se hace con el fin de saber si es conveniente eliminar dicha rotonda.

- **Servicio de Monitoreo de Rango Móvil (M-RMQ):** Este servicio consiste en monitorear la población móvil dentro de una región móvil definida por el usuario. Se considera que la región está asociada a un nodo móvil, denominado nodo focal. Por tanto cuando el nodo focal se mueve, cambia la ubicación de la región, por tanto la población móvil también pudiera cambiar. Por ejemplo, un usuario quisiera conocer el número de carabineros por cuadra mientras el usuario se mueve.

- **Servicio de monitoreo de KNN estacionario (S-KNNMQ):** Este servicio busca determinarlos K nodos móviles más cercanos a un punto fijo de consulta definido por el usuario. Cada vez que existe un nuevo k-ésimo nodo más cercano, esto es notificado al usuario. Por ejemplo una persona al llegar al terminal de buses desea ubicar el taxi (k=1) más cercano a su paradero.

- **Servicio de supervisión de KNN móvil (M-KNNMQ):** Este es similar al servicio anterior, excepto que el punto de consulta está asociado con algún nodo móvil, llamado nodo focal. Cuando el nodo focal se mueve, el punto de consulta cambia y por tanto pudiera también cambiar el conjunto de los K nodos más cercanos. Por ejemplo durante un viaje aéreo, un avión desea ubicar el avión o helicóptero más cercano a su posición. Resolver esta consulta le permitirá evitar colisiones y accidentes lamentables, ya que este podría corregir su trayectoria o dar algún tipo de aviso.

3.2 Plataforma Genérica para el procesamiento eficiente de Consultas de Monitoreo en una red de sistemas móviles P2P

En esta sección describiremos la plataforma que implementa SBLs sobre

una red adhoc móvil propuesta por [1]. Los autores de este trabajo, consideran a una red ad hoc móvil cualquiera que está formada por un conjunto de nodos móviles, los cuales conocen sus ubicaciones exacta en cada momento. En este articulo se asume que los nodos emplean algún protocolo de comunicación, que emplea la retransmisión de los paquetes, con el objeto de difundir información entre ellos y por tanto no describen el cómo se implementan tales protocolos.

En este trabajo nos centramos en simular el procesamiento de SBL del tipo de S-RMQs y la implementación de un servicio de ubicación de usuarios, que denominamos SLoc.

3.2.1 Servicio S-RMQ

Para acceder a un servicio S-RMQ, consideraremos que un usuario define su consulta en términos de una región geográfica fija de forma rectangular para la cual desea conocer que usuarios han ingresado o salido de ella. También se asume que los nodos tienen las mismas capacidades de comunicación. Se define el rango de transmisión de un nodo (r) , como el radio de cobertura de una transmisión inalámbrica. Por tanto todos los usuarios cuya ubicación cae dentro de este radio, se asume que escuchan sin problemas una transmisión.

La técnica propuesta en [1] requiere primero en particionar la región de servicio (lugar donde se mueven los usuarios) en un conjunto de celdas o dominios (O subdominios), como se muestra a continuación en la Figura 1:

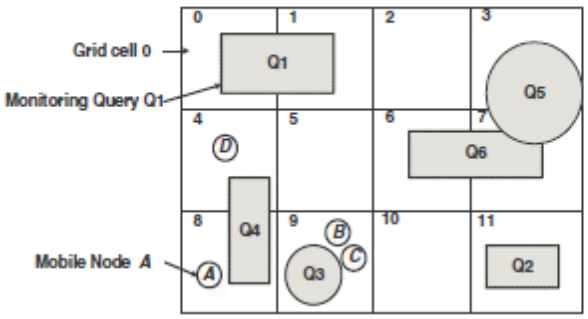

Figura 1: Especificación grafica Consulta S-RMQ[1]

Cada celda se asume que corresponde a un cuadrado cuya diagonal es menor o igual al radio de cobertura de los nodos (r). Por lo tanto, cuando un nodo difunde un mensaje, cubre toda la celda donde reside el nodo. La partición de red se hace conocida a todos los nodos móviles, y cada nodo móvil almacena en un caché las consultas que son relevantes para su celda de origen.

Decimos que una celda es **la celda de origen** de un nodo, si el nodo está dentro de la celda y una consulta es relevante para esta celda la consulta se superpone con la grilla. Una consulta puede ser relevante para más de una celda, ya que su rango puede abarcar varias celdas. Por eso la consulta Q3 sólo es relevante para la celda 9, mientras que Q5 es relevante para las celdas 3 y 7.

Considerando la partición de la red ya explicada, la creación / eliminación de una consulta puede hacerse primero determinando sus celdas relevantes y luego la operación a todos los nodos en estos. Cuando un nodo se mueve dentro de su celda de origen, supervisa su movimiento contra las consultas relevantes del nodo. Cuando detecta que ha cruzado cualquier límite de consulta, notifica al creador de la consulta en consecuencia. Dado que un nodo sabe exactamente cuándo se necesita dicho informe, este enfoque garantiza resultados de consultas en tiempo real. En este esquema, las consultas relevantes de un nodo se replican entre todos los nodos de la celda. Por lo tanto, cuando un nodo se mueve a una nueva celda, puede recuperar las consultas relevantes de la nueva celda desde cualquier otro nodo de la celda.

3.2.2. Servicio de Localización (SLoc)

Este servicio consiste en conocer la ubicación, en términos de la celda, más reciente de un usuario. SLoc también podría permitir la implementación de un servicio de seguimiento de usuarios, si los nodos en la celda hogar de un

usuario almacenaran todos los reportes enviados. La celda hogar de un nodo es aquella celda calculada con alguna función de Hashing (F) que mapea el espacio de identidades de los usuarios al espacio de identidades de las celdas.

Utilizando la misma infraestructura de grillas y celda que se utiliza para procesar consultas S-RMQ, se podría complementar del siguiente modo:

1. Cuando un usuario cruza hacia una nueva celda aprovecha el mensaje de solicitud de consultas relevantes de dicha celda para notificar su identidad a los usuarios existentes en ella.
2. Un nodo residente en la nueva celda que escucha el mensaje del paso 1. envía un geocast a la celda hogar del nuevo nodo. La identidad de la celda hogar se determina mediante una función de Hashing (F).
3. En el caso que no haya nodos residentes, es el nodo retenedor el que se encargara de realizar el paso 2.

3.3 Tecnologías para implementar una red P2P

En la información que se acaba de escribir se mencionan algunas tecnologías que utilizan las consultas mencionadas anteriormente, por eso aunque no las estemos utilizando en la implementación del simulador las mencionaremos para mayor información. Los dispositivos son fundamentales para todo que todos se

comuniquen y funcionen correctamente son los dispositivos móviles hablaremos de ello a continuación de forma breve.

Redes P2P

Una red P2P (Peer-to-peer), también conocida en español como red entre pares, es en la actualidad una de las formas más importantes y populares de compartir todo tipo de material entre usuarios de Internet**,** sin importar la plataforma de software utilizada ni el lugar o momento en que se encuentren.

Básicamente**, las redes P2P son una red que funciona sin necesidad de contar ni con clientes ni con servidores fijos,** lo que le otorga una flexibilidad que de otro modo sería imposible de lograr. Esto se obtiene gracias a que la red trabaja en forma de una serie de nodos que se comportan como iguales entre sí. Esto en pocas palabras **significa que las computadoras o móviles se conectadas a la red P2P actual al mismo tiempo como clientes y servidores** con respecto a las demás conectados.

Otra de las ventajas asociadas a las redes P2P es que las mismas **pueden aprovechar de mejor manera, es decir obtener un mejor provecho y optimización, en el uso del ancho de banda disponible entre los usuarios para el intercambio de archivos,** lo que permite de este modo obtener una mejor performance y rendimiento en las conexiones, lo que se traduce en una mejor velocidad de transferencias, y por lo tanto en una bajada de archivos más rápida

Bluetooth

La tecnología inalámbrica Bluetooth es una tecnología de ondas de radio de corto alcance que opera en la banda de frecuencia de 2,4 GHz (gigahercios), cuyo objetivo es el simplificar las comunicaciones entre dispositivos

informáticos, como computadores portátiles, smartphones, tablets, entre otros. También pretende simplificar la sincronización de datos entre los dispositivos y otros computadores.

La topología de red de Bluetooth es 1:N, donde un dispositivo "maestro" puede conectarse a uno o muchos dispositivos "esclavos". La red formada es llamada una "Piconet", donde el dispositivo con rol maestro es responsable del control de la red [20].

Existen equipos Bluetooth clase 1, 2 y 3. Estas clases se diferencian entre sí solo por el rango de alcance de la comunicación inalámbrica. Los dispositivos clase 1 llegan a 100 metros, los de clase 2 lo hacen a 20 metros, mientras que los dispositivos Bluetooth de tercera clase, poseen apenas un metro de alcance y son los que casi no se usan [21].

Wi-Fi Direct

Wi-Fi Direct, inicialmente llamado Wi-Fi peer-to-peer (P2P), es una tecnología desarrollada por Wi-Fi Alliance que soporta los estándares IEEE 802.11 a/b/g/n/ [22], la cual permite a los dispositivos que cuenten con la tecnología Wi-Fi conectarse directamente sin necesidad de un AP (punto de acceso) que permita la comunicación entre ellos. Los dispositivos que soportan Wi-Fi Direct no tienen necesidad de acceder a internet, los dispositivos simplemente establecen un "Grupo P2P" para comunicarse entre ellos (comunicación Peer-to-Peer). Como se muestra en la figura 2, un dispositivo que pertenece a un Grupo P2P es llamado Group Owner P2P (GO P2P) o cliente P2P.

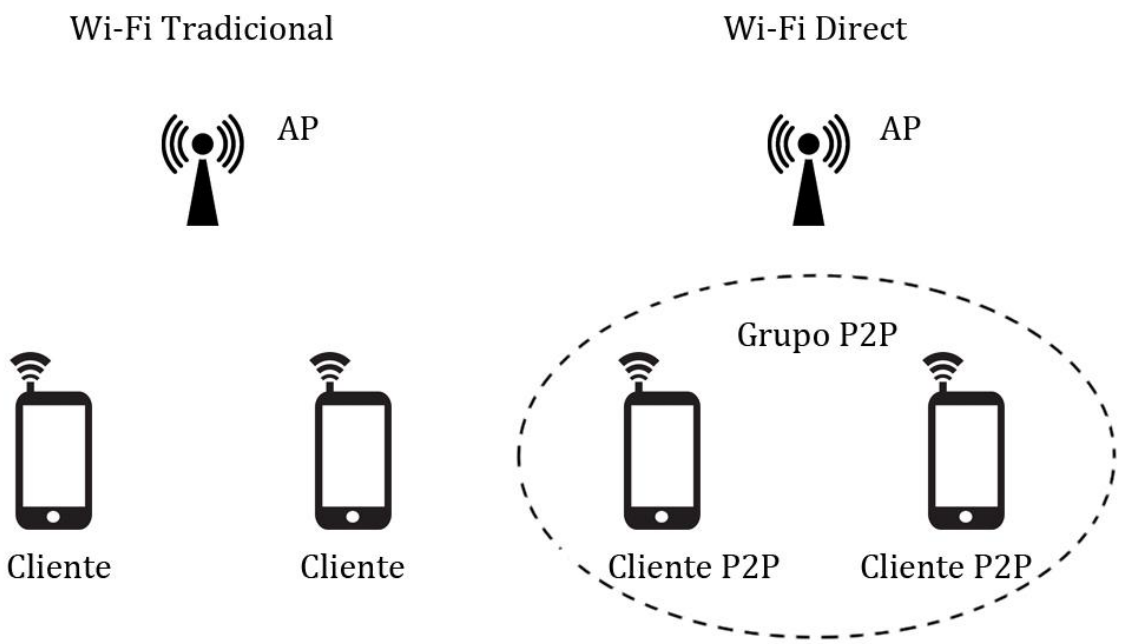

Figura: 2 Conexión Wi-Fi comparada con Wi-Fi Direct

La topología en un Grupo P2P es 1:N, donde un GO P2P (Group Owner P2P), que en este caso actúa similar a un AP (punto de acceso) normal, puede conectarse a múltiples clientes. Es similar a la topología WLAN pero sin la comunicación cliente-cliente a través del GO P2P. Wi-Fi Direct se diferencia de Ad hoc debido a que sus estructuras de red son completamente diferentes, aunque ambos son soluciones peer-to-peer.

Wi-Fi Direct es compatible con dispositivos certificados Wi-Fi. Dado que Wi-Fi Direct usa los mismos estándares IEEE 802.11 que el Wi-Fi tradicional, puede operar a distancias de hasta 200 metros y alcanzar velocidades de transferencia de datos de hasta 250 Mb/s (megabits por segundos).

Actualmente hay cerca de 5000 dispositivos electrónicos certificados con Wi-Fi Direct, que van desde smartphones y tablets a televisores y dispositivos de juegos. Wi-Fi Direct ha sido una característica obligatoria para los nuevos dispositivos Android comenzando con la versión 4.0 "Ice Cream Sandwich". Dispositivos con Android inferior a la versión 4.0 no soportan esta tecnología.

Los dispositivos que usan Wi-Fi Direct, oficialmente denominados "dispositivos P2P", interactúan con otros dispositivos estableciendo Grupos P2P. Un dispositivo en un Grupo P2P asumirá el rol de Group Owner P2P y los otros dispositivos en el grupo serán referidos como clientes P2P [23].

Modo Ad Hoc

Ad hoc [6] es un tipo de red descentralizada que consiste solo de estaciones (clientes), como se puede ver en la figura 2, donde cada cliente se comunica directamente con los demás, sin necesidad de un AP (punto de acceso)[8]. Las configuraciones Ad hoc son comunicaciones punto a punto. Solo los dispositivos dentro de un rango definido pueden comunicarse entre sí para formar una red peer to peer (de igual a igual)[15].

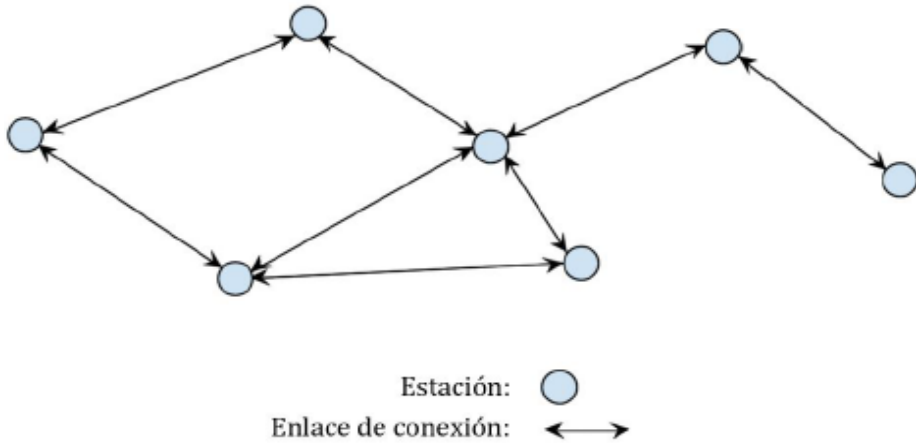

Figura 3: Estructura de una Red Ad Hoc [6]

WI-FI

Es una marca comercial de Wi-Fi Alliance, organización que adopta y certifica los equipos que cumplen con los estándares 802.11 de las redes inalámbricas de área local. Wi-Fi Alliance define Wi-Fi como un dispositivo de red de área local inalámbrica (WLAN) que implementa el estándar 802.11 por el IEEE [5]. Wi-Fi es una metodología de conexión que permite interconectar dispositivos y acceder a Internet sin usar cables ni realizar configuraciones avanzadas, lo que permite una gran simpleza de uso y movilidad. Los dispositivos que poseen esta posibilidad, tales como computadores, computadores portátiles, consolas de videojuegos, smartphones o tablets, entre otros, se pueden conectar a internet a través de un punto de acceso de red inalámbrica [9].

Dispositivos móviles

Un dispositivo móvil inteligente, desde ahora smartphone (contraparte en inglés), es un teléfono con características muy avanzadas, tales como una pantalla táctil de alta resolución, conectividad Wi-Fi, capacidad de navegar por la Web, y la habilidad de aceptar sofisticadas aplicaciones. Es por esto que cada vez se están requiriendo dispositivos con mayores capacidades de almacenamiento, procesamiento y hardware, que permita a los usuarios acceder a los distintos tipos de aplicaciones con una mayor fluidez. Los smartphones hoy en día se asemejan a verdaderos computadores ya que no sirven solo para hablar, sino que son un medio de comunicación, información y diversión para el usuario. Estos smartphones funcionan bajo cualquiera de estos sistemas operativos móviles: iOS, Symbian, BlackBerry OS, Windows Mobile y Android OS [7].

3.4 Modelos de Movimientos

Como se mencionó con anterioridad, en este proyecto se utilizarán un algoritmos de movimiento con la finalidad de representar de una forma óptima y cercana a la realidad el movimiento, en las simulaciones gráficas que se desarrollarán.

El algoritmo a implementar y utilizar para esta simulación es:

- *Random Walk.*

Aunque igual mencionaremos otro movimiento que igual puede ser implementado en algún momento en el simulador, este es:

- *Random Waypoint.*

Random Walk

También conocido como "*Caminata Aleatoria*" o "*Paseo Aleatorio*" es una formalización matemática de la trayectoria que se crea al realizar sucesivos pasos aleatorios. Este concepto es aplicado en una variedad de campos para estudiar diversos fenómenos como el camino que toma una molécula a través de un líquido o gas (Física), el precio de una acción fluctuante (Economía), la posición una persona en un área definida en un determinado periodo de tiempo (Ciencias de la Computación). Es esto último lo que es de interés para este trabajo, debido a que de

esta forma podemos simular el movimiento de un número específico de "usuarios" dentro del área.

Para realizar cada movimiento, se utiliza el principio básico de los algoritmos de este estilo, el cual dice que la posición de un punto o "usuario" en un cierto instante depende sólo de su posición en un instante previo y una variable aleatoria que determina su nueva dirección y longitud de paso.

Para efectos del algoritmo, esto se traduce como lo describimos a continuación:

$$X(i+I) = X(i) + A(I)$$

Donde "X(i)" corresponde a una trayectoria que empieza con X(0), "A" es la variable aleatoria que determinará el siguiente paso y "I" corresponde al intervalo de tiempo que hay entre un paso determinado y el siguiente.

Es importante destacar que, un paseo aleatorio define el movimiento de un usuario de forma libre en el área en que este se mueve, lo cual significa que un usuario puede moverse tomando cualquier dirección y cualquier camino que desee sin importar si este sea un cruce o una vía urbana. Es debido a esto, dado que trabajaremos las simulaciones gráficas en un entorno urbano.

Random Waypoint

Corresponde al segundo modelo de movimiento que hemos seleccionado para realizar las simulaciones gráficas de las técnicas seleccionadas. En el manejo de movilidad (*Mobility Management*), el modelo *Random Waypoint* es un modelo aleatorio para generar el movimiento de usuarios móviles, y permite observar como su localización, velocidad y aceleración cambian con el tiempo. Este modelo es uno de los modelos de movilidad más populares utilizado mayoritariamente con propósitos de diferentes tipos de simulación, sobre todo en el área de las Redes Ad Hoc Móviles, para evaluar protocolos de rutas (*RoutingProtocols*), esto es debido a su amplia difusión y simplicidad.

Una de las características más importantes de este modelo, es que se basa y permite a los nodos o usuarios moverse de forma aleatoria libremente sin restricciones. Esto quiere decir que tanto la velocidad, su localización en un determinado momento y dirección que toman para moverse son todas escogidas de forma aleatoria e independiente de otros usuarios (esto es entonces, un usuario solo depende de sí mismo para obtener sus "variables de movimiento").

El modelo finalmente se basa en lo que describimos a continuación: Cada usuario o nodo comienza su "camino" realizando una pausa de una determinada cantidad de segundos. Luego, este selecciona una nueva posición aleatoria a la que se moverá en el área de simulación y una velocidad aleatoria entre 0 y un máximo pre-definido. El usuario se mueve entonces a ese lugar y pausa de la misma manera que se describió anteriormente, espera una cantidad determinada de segundos y obtiene una nueva posición donde moverse, así como también su velocidad de movimiento. Es así entonces, como esto se repite en un ciclo mientras la simulación esté en ejecución.

Estos modelos han sido la base en la que hemos basado la construcción de las simulaciones gráficas de las dos técnicas de encubrimiento que trabajaremos de ahora en adelante en este proyecto, por lo que es importante tenerlos en cuenta al momento de observar el movimiento de los usuarios en el área de las simulaciones y posteriormente, los resultados obtenidos en estas.

3.5 Trabajos Relacionados

Existe un trabajo de investigación realizado por un profesor de la Universidad del Bío Bío, llamado **"A Generic Platform for Efficient Processing of Spatial Monitoring Queries in Mobile Peer-to-Peer Networks"**, desarrollado por Patricio Galdames Sepúlveda en conjunto de otras personas, el cual implementa un mecanismo de procesamiento de consultas (S-RMQ) [1]

Aroa Carcavilla Sanz, "Sistemas de posicionamiento basados en WiFi", Un sistema de seguimiento y localización consiste básicamente en la combinación de las tecnologías de posicionamiento, para la localización geográfica de las unidades móviles, un medio de comunicación para trasmitir y recibir información entre las unidades móviles y el centro de control, y finalmente un software con capacidad de procesamiento de cartografía. [2]

Ana M. Bernardos Barbolla, Juan A. Besada Portas y José R. Casar Corredera, ETS Ingenieros de Telecomunicación Universidad Politécnica de Madrid, "Tecnologías de localización", Los servicios basados en localización (Location Based Services, LBS), tal y como los entendemos hoy en día, engloban un conjunto de aplicaciones que incorporan a la información de posición otros datos relativos al entorno, con el fin de proporcionar un servicio de valor añadido al usuario [3]

Andrés Jonathan Abeliuk Kimelman, Universidad de Chile, "Arboles de sufijos comprimidos para textos altamente repetitivos", habla de las consultas S-RMQ las cuales pueden ser de gran ayuda para su tema de árboles de sufijo, poniendo varias teorías que las trata de llevar a cabo. [4].

4 FACTIBILIDAD

4.1 Introducción

El estudio de factibilidad permite analizar qué tan viable puede llegar a ser el desarrollo de la aplicación en cuestión, permitiendo tomar la decisión de si es conveniente llevarlo a cabo.

Es muy probable que la realización de un proyecto de la aplicación esté plagado de escasez de recursos y de fechas de entregas no realistas, por lo mismo es necesario evaluar la viabilidad de un proyecto, ya que gracias a éste es posible evitar meses de esfuerzos sin resultados, la perdida de dinero y el bochorno profesional si se reconoce que el sistema es un fracaso, además ayuda a disminuir los riesgos y asegura el valor del trabajo.

En relación con la factibilidad, se refiere a la disponibilidad de los recursos necesarios para llevar a cabo los objetivos del proyecto, la factibilidad se apoya en 3 aspectos básicos que serán desarrollados en el capítulo, estos son:

- Factibilidad técnica.
- Factibilidad operativa.
- Factibilidad económica

4.2 Factibilidad Técnica

Para que el proyecto de la construcción de la aplicación se debe contar con los siguientes elementos de hardware:

Windows.

- ✓ Ram: 128 MB y 64 MB para windows XP (32bits).
- ✓ Espacio en disco: 124 MB.

Mac OS X.

- ✓ Sin restricción.

Linux.

- ✓ Ram: 64MB.
- ✓ Espacio en disco: 58 MB.

Recursos de Software Necesarios:

Nombre: Java Runtime Enviroment
Desarrollador: Oracle.
Versión: 7 update 45 o posterior.

Todos los recursos de software cuentan con licencias gratuitas y se encuentran disponibles en Internet específicamente en la página web de Java. Respecto a los conocimientos del desarrollador, estos abarcarían dominio del lenguaje de programación Java.

4.3 Factibilidad Operativa

El presente proyecto busca implementar una aplicación del manejo de consultas entre usuarios a través de redes P2P. Referente a los impactos positivos, se puede mencionar que se desea lograr que mediante esta simulación sea mejorar la interpretación de información disponible entre los usuarios móviles. Esta solución no solo es aplicable a un grupo determinado de personas, sino a todas aquellas que posean un teléfono móvil y que desee compartir su información.

4.4 Factibilidad Económica

En este estudio se determinan los recursos necesarios para desarrollar el proyecto y los costos en los que se debe incurrir para su fabricación, se realiza una comparación entre los costos en hardware, software y mano de obra con los beneficios que se obtendrán cuando el sistema esté en su fase de explotación.

A continuación se detallan los costos considerados en el desarrollo del proyecto, los cuales corresponden a valores de mercado, obtenidos a través de diversas consultas a personas que trabajan en el área del desarrollo de software.

4.4.1 Costo de desarrollo

Hardware y Software de desarrollo: tiene un costo total de $0, ya que el hardware se encuentra instalado y operativo hace algún tiempo en la computadora del analista desarrollador. Respecto del software son herramientas de libre acceso sin costo. Así su costo final total es de $0.

Encargado del desarrollo: para llevar a cabo el proyecto se requiere de un ingeniero civil informático.

- ✓ Costo de mercado de un analista desarrollador hora/hombre es de $4.166,666.- aprox. Donde $4.167x4x45=750.000.- pesos por mes trabajado.

- ✓ El trabajo se estima en un período de 2 meses y se trabajará 45 horas semanales, lo que se traduce en un total de 360 horas de elaboración del proyecto.
- ✓ El costo total del informático es de $1.500.000. Este costo es logrado por 2 meses trabajados por el informático 750.000x2= 1.500.000.-

El costo total calculado anteriormente, no es considerado, debido a que el desarrollador, es un estudiante que se encuentra realizando su proyecto de título.

4.4.2 Costo de Instalación

Hardware y software del servidor: este costo no es considerado, puesto que lo que se necesita para el desarrollo de la aplicación están en mano del alumno, por lo cual no tiene costo alguno.

4.4.3 Costo de Operación

Hardware y software: para la operación del sistema se requerirá del Hardware y Software detallado anteriormente, el cual se encuentra funcionando por lo que se considera costo de operación $0.

4.4.4 Costo Mantención

Este costo no será considerado, ya que solo es una aplicación de una simulación no se requiere mantención de dicha aplicación. Del análisis de los costos se puede obtener el detalle de la inversión inicial en que se debe incurrir para la puesta en marcha.

Inversión Inicial:

Desarrollo	$1.500.000
Instalación	$0
Operación	$0
Total	**$1.500.00**

Tabla 1: Inversión Inicial

4.4.5 Beneficios Intangibles

Debido a que es solo una aplicación en base a una simulación no pretende mejorar ni reemplazar algún proyecto anterior, solo es posible mencionar los beneficios que generaría la solución propuesta.

Si se llevara a la práctica la implementación de este simulador en Comunicación entre celulares móviles a través de redes P2P, obtendríamos beneficios directos, pero si lo lleváramos a la práctica, que serían:

- Este simulador es una ayuda base para la implementación de otras consultas y técnicas de procesamiento mencionadas anteriormente.

- También hay beneficios pedagógicos, al mostrar cómo funcionan las técnicas implementadas en el simulador, facilitando a los alumnos un mejor entendimiento de ellas.

Por otra parte se obtendrían beneficios intangibles en diferentes escenarios expuesto a continuación:

✓ **Escenario 1: Carretera:** en un accidente automovilístico los usuarios se pondrían avisar que ocurrió un accidente en la carretera, de esta manera podrían cambiar su ruta, tomando alternativas y descongestionando la vía de acceso a dicho accidente a los servicios de salud, bomberos, etc.

✓ **Escenario 2: Sismo:** en un desastre podrían los vecinos de un sector preguntar si se encuentran bien, comunicándose entre sí por medio de esta aplicación y ayudándose sin perder tiempo.

✓ **Escenario 3: Extravió:** a veces los padres se les extravían sus hijos en los mall, calles etc., con esta aplicación podrían mandar un alerta a todos los celulares que se encuentran a sus alrededor haciendo más ágil la búsqueda de la persona extraviada, optimizando el tiempo de búsqueda y poder localizarla más rápido.

✓ **Escenario 4: Entretenimiento:** A veces los turista necesitan encontrar localizaciones como centro de comidas, mall, cines en algún determinado lugar, preguntando por medio de esta aplicación se podría recomendar el mejor lugar que pueda ir, acortando la pérdida de tiempo por buscar algo, incluso que dicho turista se extraviara.

✓ **Escenario 5: Servicio Públicos:** En ocasiones las personas necesitan acudir con urgencia a un Centro de Salud, Carabineros, Bomberos, Municipalidades,

Colegios etc. Con esta aplicación se podría compartir información exacta de estos servicios compartiendo la ubicación de ellos.

5. ESPECIFICACIONES DE REQUERIMIENTOS DE SOFTWARE

5.1 Alcances

Este software se realizó con la finalidad de desarrollar una simulación gráfica de las técnicas seleccionadas en este proyecto de título que permita procesar una consulta S-RMQ (SBL) y consultas de localización (SLOC) entre usuarios móviles, explicada en el capítulo 1 "Definición Proyecto". El desplazamiento se realiza mediante la utilización de los algoritmos de movimientos descritos con anterioridad.

La aplicación permite

- Que busque usuario dentro de su rango para compartir información
- Que en la simulación de una red P2P, dado a los diferentes movimientos de los usuarios (nodos) pueda existir la transmisión de información entre ellos debido a las consultas S-RMQ y las consultas de localización.

La aplicación no permite

- Guardar la información personal de cada dispositivo móvil.
- Una conexión insegura entre dispositivos.
- No recabar información innecesaria y fuera de la red P2P.
- Que un usuario no puede conectarse a más de un dispositivo a la red.
- Que un usuario no puede transmitir información a más de un usuario a la vez.

5.2 Objetivo del Software

5.2.1 Objetivo General

La correcta transferencia de información entre usuarios móviles aplicando las técnicas de consultas y movimientos mencionados anteriormente en una red P2P.

5.2.2 Objetivos Específicos

- La simulación realiza una conexión entre usuario móviles.
- La simulación realiza consultas entre los usuarios móviles
- La simulación realiza el movimientos randomicos requerido con los usuarios móviles.

5.3 Requisitos mínimos del Software

Debido a que es un simulador requiere de condiciones mínimas para su completo funcionamiento, ya sean requisitos de software o hardware. En este artículo veremos los requisitos mínimos que requiere la aplicación para su correcto funcionamiento.

Los requisitos del Sistema operativo y hardware que se mencionaran a continuación son los que se requiere de forma mínima para su funcionamiento.

5.3.1 Requisitos del Sistema Operativo

Al ser un simulador y aplicación de escritorio desarrollada en el lenguaje de programación Java, requiere de un Sistema Operativo con interfaz gráfica tales como:

Nombre: Windows.
Desarrollador: Microsoft.
Versión:
• Windows 8
• Windows 7
• Windows Vista SP2

Nombre: Mac OS X
Desarrollador: Apple Inc.
Versión:

- Mac basado en Intel que ejecuta Mac OS X 10.7.3 (Lion) o posterior.

Nombre: Linux.
Desarrollador: Varios.
Versión:
• Oracle Linux 5.5+
• Oracle Linux 6.x (32 bits), 6.x (64 bits)
• Red Hat Enterprise Linux 5.5+, 6.x (32 bits), 6.x (64 bits)
• Ubuntu Linux 10.04 y superior

5.3.2 Requisitos de Software

Nombre: Java Runtime Enviroment
Desarrollador: Oracle.
Versión: 7 update 45 o posterior.

5.3.3 Requisitos de Hardware

Los requisitos de hardware hacen referencia a la configuración mínima de los dispositivos básicos que un computador necesita para poder ejecutar la aplicación.

• Windows.

- Ram: 128 MB y 64 MB para windows XP (32bits).
- Espacio en disco: 124 MB.

• Mac OS X.

- Sin restricción.

• Linux.

- Ram: 64MB.
- Espacio en disco: 58 MB.

5.4 Descripción Global del Producto

5.4.1 Interfaz de Usuario

Debido a que el simulador de la aplicación se encuentra codificado en lenguaje de programación Java la interfaces de usuario se diseñó de una forma que fuera sencilla y amigable para el usuario. A través de una ventana principal, se tiene acceso al iniciar la aplicación, generando una ventana en donde se dan a conocer las opciones que tiene la aplicación, comenzando con un botón de Generar Grilla, mostrando los usuarios con sus dispositivos móviles circulando por esta grilla.

5.4.2 Interfaz de hardware

Al ser un simulador, esta aplicación necesita lo básico que conlleva un computador:

- Pantalla: para mostrar los resultados obtenidos al usuario.
- Teclado y mouse: para controlar la simulación e ingresar datos.

5.4.3 Interfaz de Software

La aplicación no necesita el uso de otro software, ni otras interfaces con otros sistemas. La aplicación es autónoma en ese sentido.

5.4.4 Interfaz de comunicación

Debido a que es una aplicación de escritorio, no son requeridos protocolos específicos de internet para su debido funcionamiento.

5.5 Requerimiento Específicos

5.5.1 Requerimientos Funcionales del sistema

ID	NOMBRE	DESCRIPCION
1	Generar Marco	Se genera un marco para comenzar.
2	Cantidad Usuario	El usuario debe selecciona la cantidad de usuario que se necesita.
3	Generar Usuario	Se muestra a los usuarios seleccionados por el usuario principal.
4	Habilitar/Deshabilitar Grilla	Se visualiza la grilla del mapa en caso de estar invisible o se oculta en caso de estar visible
5	Pausar/Reanudar	Se pausa o reanuda el movimiento de los usuarios en el mapa dependiendo se está activo o pausado.
6	Limpiar Grilla	Se limpia la grilla de todo que haya en ella.
7	Localización	Se establece un usuario dentro de todos los que haya el que generara su región de encubrimiento.
8	Seleccionar Consulta	Se define una técnica de consulta, siendo estas las de Localización.
9	Seleccionar Movimiento	Se define una técnica de movimiento Random Walk.
10	Generar Consulta	Se define la selección de la consulta S-RMQ
11	Mostrar Camino	Se muestra el camino al nodo creador de localización.

Tabla 2: Requerimientos funcionales del sistema

5.5.2 Interfaces externas de entrada

Cada interfaz de entrada indica todos los grupos de datos que serán ingresados al sistema, independiente del medio de ingreso.

ID	NOMBRE DEL ÍTEM	DETALLE DE DATOS
1	Cantidad de Usuarios	Número de clientes que realizaran las consultas.

Tabla 3: Interfaces externas de entrada

6. ANALISIS

6.1 Casos de Uso

Se representan los requerimientos funcionales mediante Casos de Uso para cada uno.

6.1.1 Actores

- ✓ **Usuario**

El usuario corresponde a la persona que utilizará el simulador gráfico desarrollado. En el software tiene permisos para ejecutar todas las operaciones disponibles. No requiere de grandes conocimientos técnicos, solo debe estar familiarizado con el entorno gráfico de Java y con el uso de botones y CheckBox.

6.1.2 Descripción Caso de Uso

A continuación, se visualizan las interacciones que tienen los usuarios con la aplicación.

El actor "Usuario" es el que utiliza el software desarrollado. Este es quien ejecuta la aplicación utilizando un computador o notebook y puede ingresar nuevas posiciones para generar las rutas o caminos y visualizar así un área urbana simulada con una cantidad determinada de personas y vehículos. En base al área generada, puede finalmente generar la región de encubrimiento de un usuario que seleccione, dada una técnica que él especifique.

El usuario posee completo acceso a las funcionalidades del simulador sin la necesidad de realizar un log in.

6.1.3 Diagrama del Caso de Uso

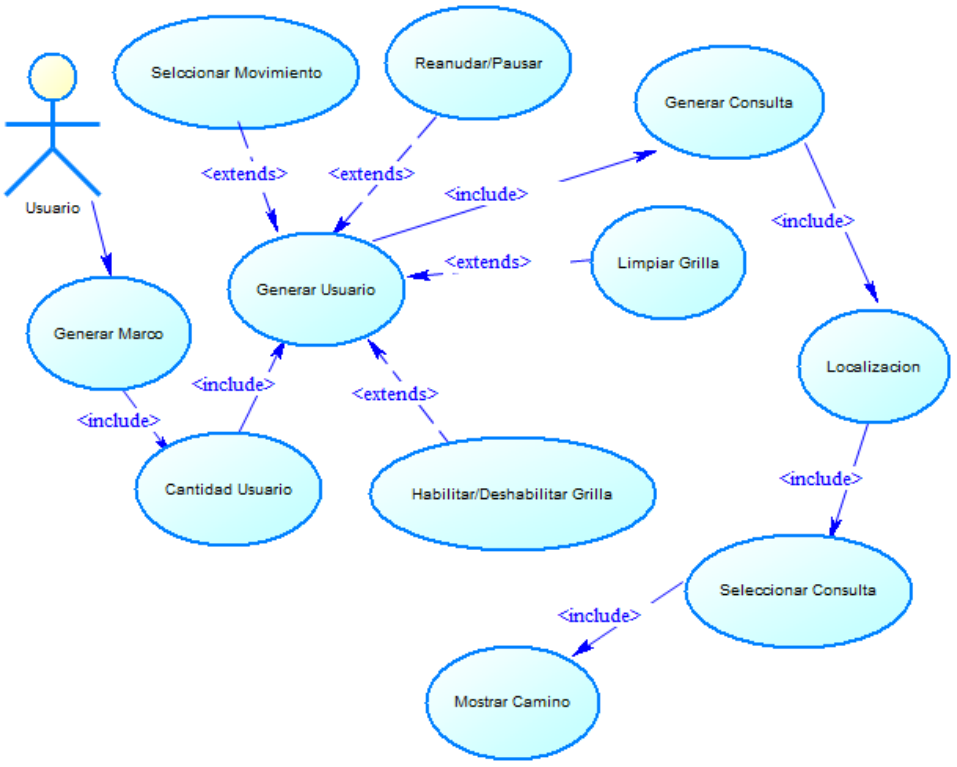

Figura 4: Diagrama de Caso de Uso

6.2 Especificación Casos de Uso.

6.2.1 Caso de uso: <Generar Marco>

- Descripción: Se genera un marco para comenzar.
- Pre-Condiciones: El usuario debe presionar el botón "Generar Marco".
- Flujo de Eventos Basico:

ACTOR	SISTEMA
	1. Se genera un maco para comenzar la aplicación,
2. El usuario debe presionar generar marco	3. Se muestra por medio de una interfaz gráfica.

Tabla 4: Flujo de Evento Basico "Generar Marco"

- Flujos de eventos alternativos:

ACTOR	SISTEMA
	1(a). Se informa que no se ha podido generar el marco volver a intentarlo.

Tabla 5: Flujo de Evento Alternativo "Generar Marco"

6.2.2 Caso de uso: <Cantidad Usuario>

- Descripción: El usuario debe selecciona la cantidad de usuario que se necesita.
- Pre-Condiciones: Se debe poner la cantidad de usuario se debe seleccionar el número que desea.
- Flujo de Eventos Básicos:

ACTOR	SISTEMA
	1. Se debe seleccionar la cantidad de usuario.
2. Se debe presionar el botón cantidad de usuario.	

Tabla 6: Flujo de Evento Basico "Cantidad de Usuario"

6.2.3 Caso de uso <Generar Usuario>

- Descripción: Se muestra a los usuarios seleccionados por el usuario principal.
- Pre-Condiciones: Se debe seleccionar la cantidad de usuario y luego presionar el botón generar usuario.
- Flujo de Eventos Básicos:

ACTOR	SISTEMA
1. El usuario debe seleccionar la cantidad de usuario.	2. El usuario debe seleccionar el botón generar usuario
	3. Se genera los usuarios correspondiente después de haberlos seleccionado

Tabla 7: Flujo de Evento Basico "Generar Usuario"

6.2.4 Caso de uso: <Pausar/Reanudar>

- Descripción: Se pausa o reanuda el movimiento de los usuarios en el mapa dependiendo se está activo o pausado.
- Pre-Condiciones: El usuario debe presionar el botón pausar/reanudar, luego de haberse generado los usuarios.
- Flujo de Eventos Básicos:

ACTOR	SISTEMA
1. El usuario presiona el botón "Pausar/Reanudar"	2. Se pausa el movimiento de los usuario en caso de que este pausado.
	3. Se reanuda el movimiento de los usuario en caso de que este en movimiento.

Tabla 8: Flujo de Evento Basico "Pausar/Reanudar"

6.2.5 Caso de uso: <Habilitar/Deshabilitar grilla>

- Descripción: se visualiza la grilla del mapa en caso de estar invisible o se oculta en caso de estar visible.
- Pre-Condiciones: El usuario debe hacer click en "Habilitar/deshabilitar grilla", luego de haber generado los usuarios.
- Flujo de Eventos Básicos:

ACTOR	SISTEMA
1. El usuario debe hacer click en "habilitar/deshabilitar grilla"	2. Se muestra en pantalla la grilla en caso de que se encuentre oculta.
	3. Se oculta la grilla en caso de que se encuentre visible en pantalla.

Tabla 9: Flujo de Evento Basico "Habilitar/deshabilitar Grilla"

6.2.6 Caso de uso: <Limpiar Grilla>

- Descripción: Se limpia la grilla de todo que haya en ella.
- Pre-Condiciones: El usuario debe presionar el botón "Limpiar Grilla", luego de haber generado los usuarios.
- Flujo de Eventos Básicos:

ACTOR	SISTEMA
1. El usuario debe presionar el botón "Limpiar Grilla".	2. Se limpia toda la grilla si es que esta con usuarios.

Tabla 10: Flujo de Evento Basico "Limpiar Grilla"

- Post-Condiciones: Se limpia la grilla de todos aquellos elementos que se encuentra en ella.

6.2.7 Caso de uso: <Generar Consulta>

- Descripción: Se establece que el usuario genere las consultas que llevas las consultas S-RMQ.
- Pre-Condiciones: El usuario debe seleccionar el botón generar consulta.
- Flujo de Eventos Básicos:

ACTOR	SISTEMA
1. El usuario selecciona el botón generar consulta.	2. Se generan las consultas en un lugar de manera estacionara en la grilla.

Tabla 11: Flujo de Evento Basico "Seleccionar Usuario"

- Flujo de Eventos Alternativo:

ACTOR	SISTEMA
	2(a). Ningún consulta generada

Tabla 12: Flujo de Evento Alternativo "Seleccionar Usuario"

- Post-Condiciones: Se muestra gráficamente mediante el color azul determinado.

6.2.8 Caso de uso: <Localización>

- Descripción: Se define como técnica de consulta..
- Pre-Condiciones: El usuario selecciona la técnica y aparece un nuevo campo, para la selección de consulta.
- Flujo de Eventos Básicos:

ACTOR	SISTEMA
1. El usuario selecciona la técnica de localización.	2. Se define la técnica de consulta en la aplicación.

Tabla 13: Flujo de Evento Basico "Seleccionar Consulta"

- Post-Condiciones: Se establece la técnica de consulta en el campo seleccionado.

6.2.9 Caso de uso <Seleccionar Movimiento>

- Descripción: Se define una técnica de movimiento Random Walk.
- Pre-Condiciones: El usuario selecciona la técnica de movimiento dentro de una lista previa definida en un campo, también debe estar seleccionado las técnicas de consulta antes mencionados.
- Flujo de Eventos Básicos:

ACTOR	SISTEMA
	1. El sistema hace que los usuarios se muevan dependiente de la técnica seleccionada, en este caso el movimiento random walk.

Tabla 14: Flujo de Evento Basico "Seleccionar Movimiento"

6.2.10 Caso de uso <Seleccionar consulta>

- Descripción: Se selecciona la consulta para buscar el camino al nodo creador.
- Pre-Condiciones: El usuario selecciona la consulta dentro de una lista previa definida en un campo, también debe estar seleccionado las consulta antes mencionados.
- Flujo de Eventos Básicos:

ACTOR	SISTEMA
1 El usuario selecciona dentro del campo asignado la consulta que desea.	2. El sistema busca el camino y lo muestra hasta el nodo creador.

Tabla 15: Flujo de Evento Basico "Seleccionar Consulta"

6.2.11 Caso de uso <Mostrar Camino>

- Descripción: Muestra el camino desde el nodo inicial al nodo creador.
- Pre-Condiciones: El usuario selecciona la consulta dentro de una lista previa definida en un campo.

- Flujo de Eventos Básicos:

ACTOR	SISTEMA
2 El usuario selecciona dentro del campo asignado la consulta que desea buscar el camino.	3. El sistema muestra el camino desde el nodo inicial y los nodos que paso hasta el nodo creador.

Tabla 16: Flujo de Evento Basico "Mostrar Camino"

7. SIMULACIONES

Realizando los estudios y análisis correspondientes de las técnicas de localización y las consultas S-RMQ que se han expuesto en este informe, se desarrolló un simulador gráfico, en donde se podrá notar con más claridad el comportamiento de las técnicas. Se ha mencionado en varias oportunidades que fue diseñado en un lenguaje de programación apto llamado Java, siendo esta de escritorio puede ser visualizada en múltiples plataformas de acuerdo a las necesidades del usuario.

Mostraremos algunas imágenes importantes de la aplicación.

Al iniciar la aplicación, lo primero que se observa el botón de Generar Marco, en donde ira puesto los usuarios y los demás botones que necesita la aplicación para su funcionalidad.

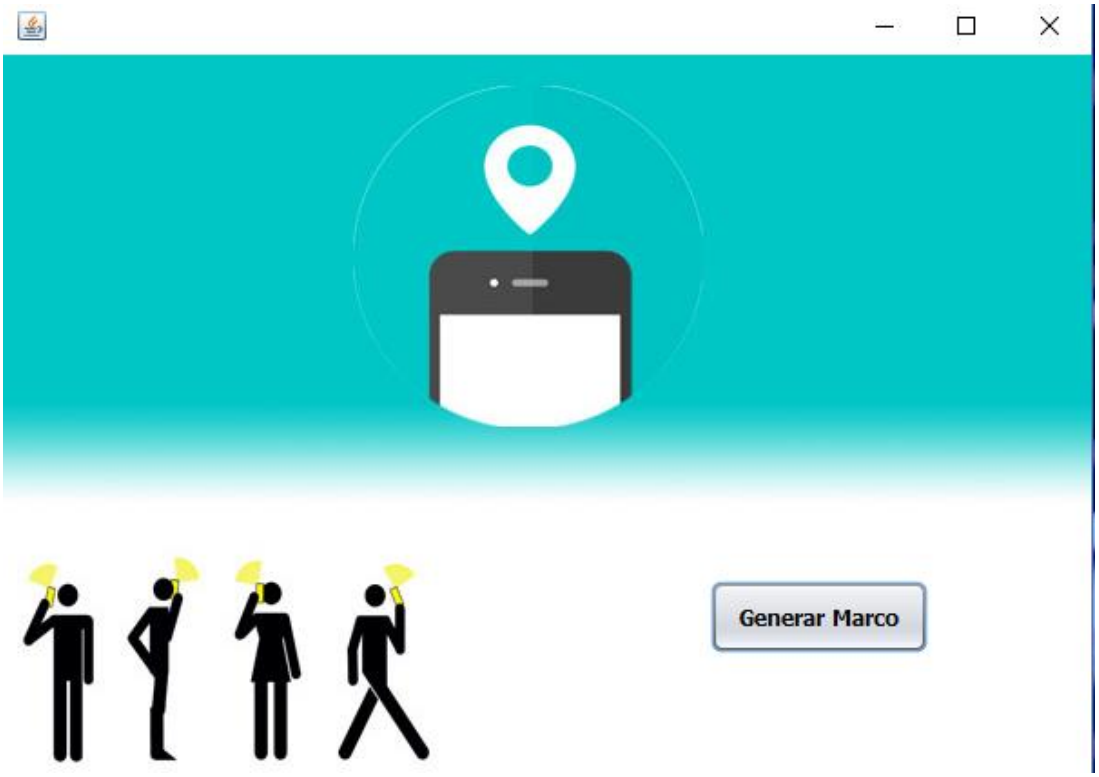

Figura 5: Pantalla de Inicio Generar Marco

Al generar el marco aparece la siguiente ventana con su primer ítem que es Cantidad de usuario empezado en 10 el cual hacer desplegado puede variar.

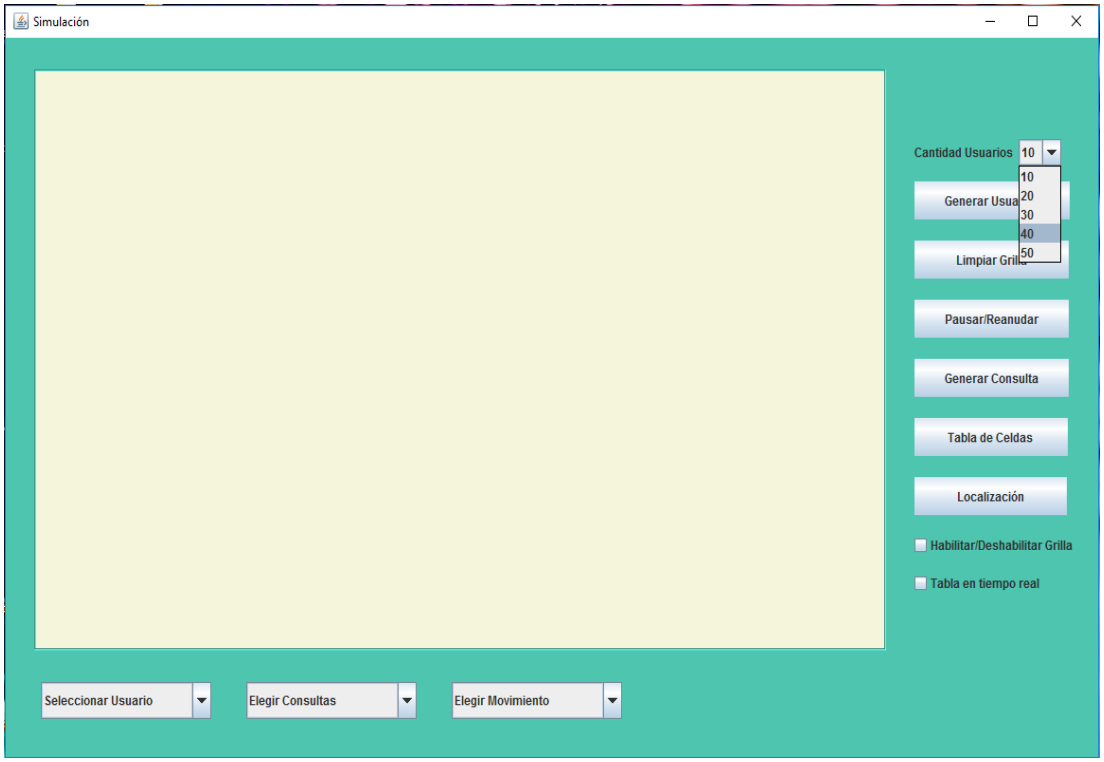

Figura 6: Cantidad de usuario

Al presionar el botón de Generar Usuario apareces los usuario en el panel designado.

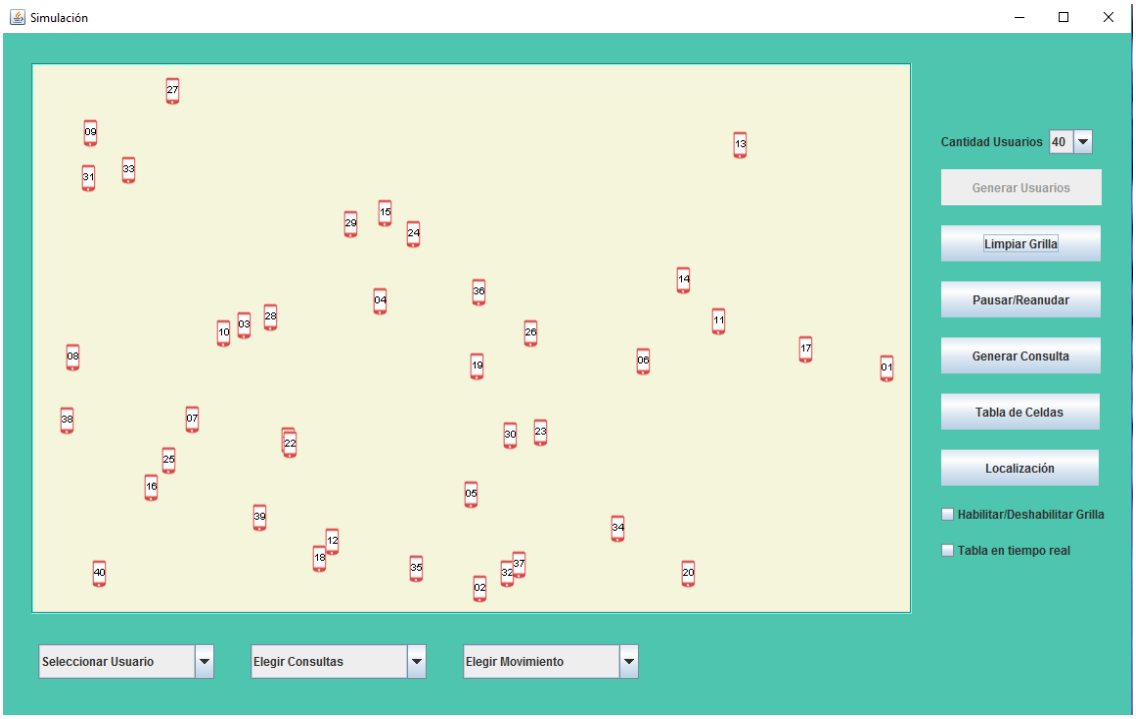

Figura 7: Cantidad de usuario generado

Podemos habilitar y deshabilitar la grilla, para cada ver las posiciones de las consultas que se generaran más adelante.

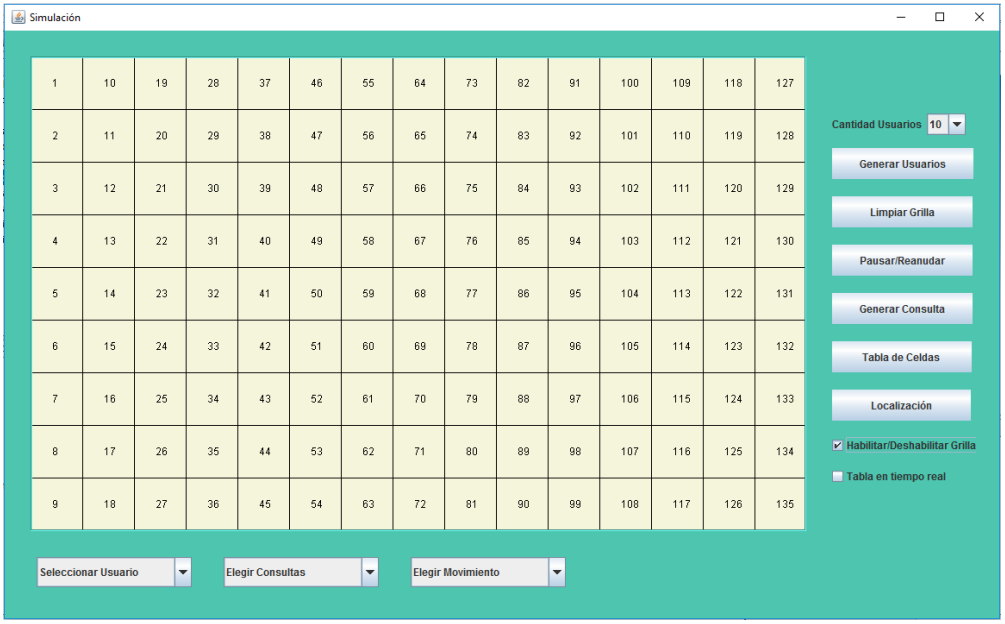

Figura 8: Habilitar/Deshabilitar Grilla

Se genera alguna consulta con un usuario al azar

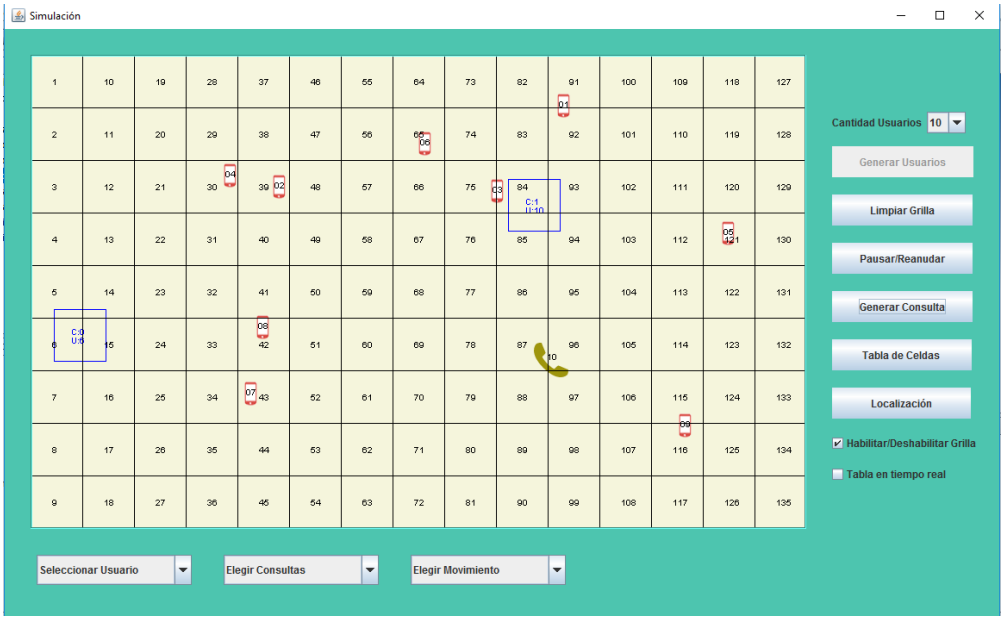

Figura 9: Genera la Consulta

Al presionar el botón de localización, podemos seleccionar una consulta ya creada anteriormente y nos mostrara su camino hasta el llegar al nodo creador.

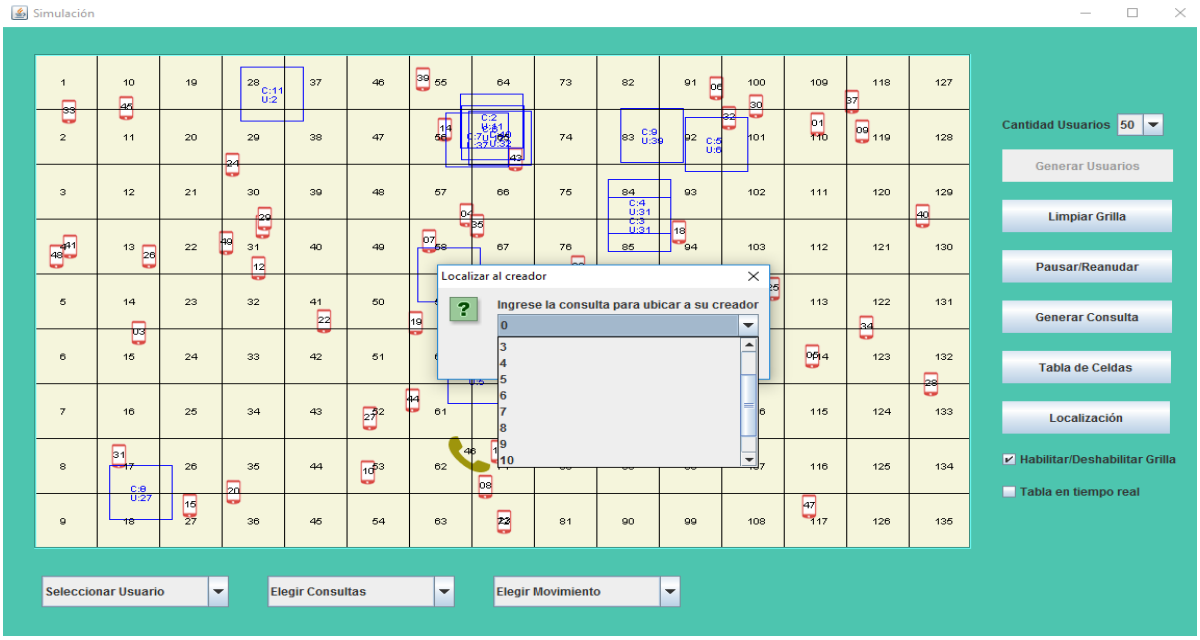

Figura 10: Localización

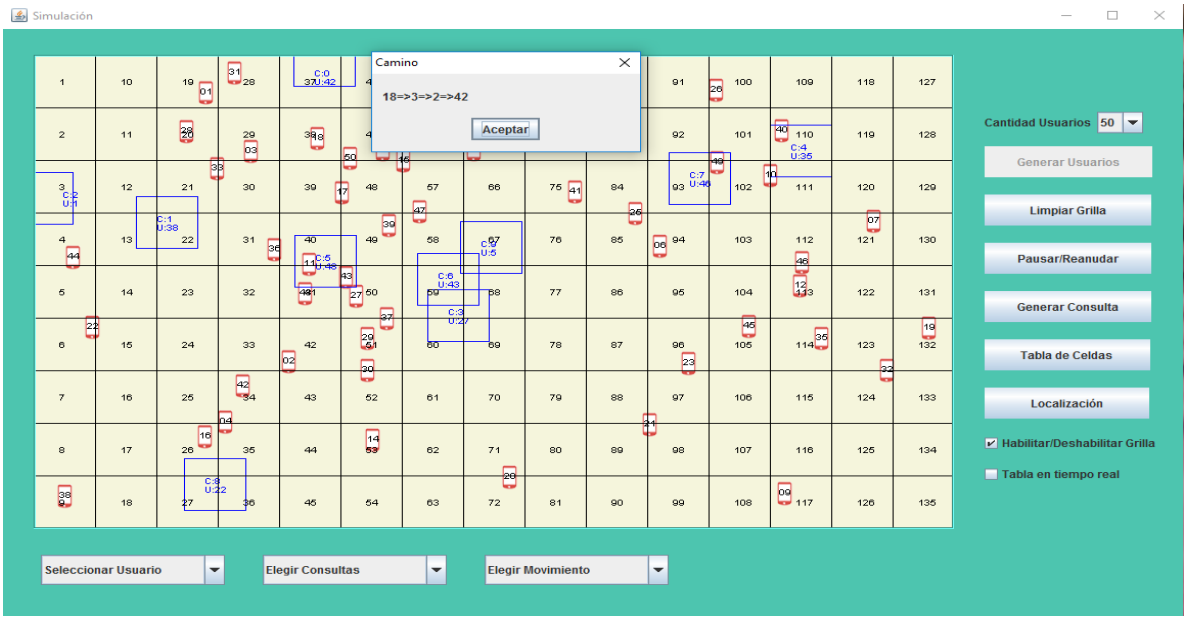

Figura 11: Muestra de Camino

Podemos ver los resultados en la siguiente imagen con una tabla que se genera en tiempo real y no.

Tabla de Celdas

Celda	Consulta {Usuarios}	H Retenedor	H Candidatos	Celda	Consulta {...	H Re...	H Candidat...
1(0,0)	Sin consultas en celda	User 30	Usuario(s): 21[0,0]	28(3,0)	Sin consult...	User...	C vacía
2(0,1)	Sin consultas en celda	User 21	C vacía	29(3,1)	Con(C): 6=...	no R...	C vacía
3(0,2)	Sin consultas en celda	User 21	C vacía	30(3,2)	Con(vlD): 6...	no R...	C vacía
4(0,3)	Sin consultas en celda	User 21	C vacía	31(3,3)	Sin consult...	User 8	Usuario(s):...
5(0,4)	Con(C): 2=>{ 21 9}	User 16	C vacía	32(3,4)	Sin consult...	User...	C vacía
6(0,5)	Con(vll): 2=>{ 21 9}	User 7	C vacía	33(3,5)	Sin consult...	User...	C vacía
7(0,6)	Sin consultas en celda	User 7	C vacía	34(3,6)	Sin consult...	User...	C vacía
8(0,7)	Sin consultas en celda	User 7	C vacía	35(3,7)	Sin consult...	User...	C vacía
9(0,8)	Sin consultas en celda	User 7	C vacía	36(3,8)	Sin consult...	User...	C vacía
10(1,...	Sin consultas en celda	User 30	C vacía	37(4,0)	Sin consult...	User 6	C vacía
11(1,...	Sin consultas en celda	User 30	Usuario(s): 7[1,1] 27[1,1]	38(4,1)	Sin consult...	User 6	C vacía
12(1,...	Sin consultas en celda	User 21	C vacía	39(4,2)	Sin consult...	User 5	C vacía
13(1,...	Sin consultas en celda	User 16	C vacía	40(4,3)	Sin consult...	User 8	Usuario(s):...
14(1,...	Con(vSD): 2=>{ 21 9}	User 16	C vacía	41(4,4)	Sin consult...	User...	Usuario(s):...
15(1,...	Con(vlD): 2=>{ 21 9}	no Retainer	C vacía	42(4,5)	Sin consult...	User...	C vacía
16(1,...	Sin consultas en celda	User 7	C vacía	43(4,6)	Sin consult...	User 3	C vacía
17(1,...	Sin consultas en celda	User 7	C vacía	44(4,7)	Sin consult...	User 3	C vacía
18(1,...	Sin consultas en celda	User 7	C vacía	45(4,8)	Sin consult...	User...	C vacía
19(2,...	Sin consultas en celda	User 30	C vacía	46(5,0)	Sin consult...	User 6	C vacía
20(2,...	Con(vSI): 6=>{}	User 19	C vacía	47(5,1)	Sin consult...	User 6	C vacía
21(2,...	Con(vll): 6=>{}	User 24	Usuario(s): 16[2,2] 30[2,2]	48(5,2)	Con(vSI): 1...	User 5	C vacía
22(2,...	Sin consultas en celda	User 16	C vacía	49(5,3)	Con(C): 1=...	User...	C vacía
23(2,...	Sin consultas en celda	no Retainer	C vacía	50(5,4)	Sin consult...	User...	C vacía
24(2,...	Sin consultas en celda	User 27	C vacía	51(5,5)	Sin consult...	User...	Usuario(s):...
25(2,...	Sin consultas en celda	User 36	C vacía	52(5,6)	Sin consult...	User...	C vacía
26(2,...	Sin consultas en celda	User 36	C vacía	53(5,7)	Sin consult...	User 3	C vacía
27(2,...	Sin consultas en celda	User 36	C vacía	54(5,8)	Sin consult...	User...	C vacía

Figura 12: Tabla de Resultados

8. CONCLUSION

En el proyecto desarrollado se han presentado diversos desafíos por el área de la investigación como aprender a usar un lenguaje de programación llamado Java desde lo básico hasta lo más complejo.

En primer lugar dentro de los aportes de este proyecto fue el estudio y análisis de técnicas de consulta en servicios basados en localización y consultas de localización. Aprendiendo sobre la forma en que se comporta cada técnica permitiendo que los objetivos del proyecto fueron alcanzados sin problema.

Por medio de un estudio de dos modelos el random walk y waypoint se realiza su implementación en el simulador grafico que genera una grilla que posiciona a los usuarios virtuales asignándole una consulta específica.

Cabe notar que la utilización de programación orientada a objeto resulto ser de gran ayuda pero complicado a la vez, el lenguaje de programación JAVA tiene un gran potencial, no solo por la gran cantidad de documentación encontrada también por una sintaxis relativamente simple que permite una mayor fluidez en su codificación. Además una de sus ventajas es el ser una herramienta multiplataforma que permite ejecutar otras plataformas sin tener que modificar el código.

Personalmente el proyecto de primera me costó mucho entender el concepto básico, pero a medida que me junte con mi profesor guía me fue quedando más claro cada vez. Cada día que pasaba al ir investigando me iba entregando más conocimiento y habilidades para enfrentar mis próximos proyectos a futuros tanto personales como laborales.

9. BIBLIOGRAFIA

[1] Galdames P. et al. Kihwan Kim and Ying Cai, “A Generic Platform for Efficient Processing of Spatial Monitoring Queries in Mobile Peer-to-Peer Networks”, In MDM 2010.

[2] Aroa Carcavilla Sanz, “Sistemas de posicionamiento basados en WiFi”, 24 de febrero de 2006.

[3] Ana M. Bernardos Barbolla, Juan A. Besada Portas y José R. Casar Corredera, ETS Ingenieros de Telecomunicación Universidad Politécnica de Madrid, “Tecnologías de localización”, Enero 2005.

[4] Andres Jonathan Abeliuk Kimelman, Arboles de sufijos comprimidos para textos altamente repetitivos, Universidad de Chile, Enero 2012.

[5] Wi-Fi Alliance, "*Who are We*". **Fuente**: http://www.wi-fi.org/who-are-we.

[6] L.M. Feeny, M. Nilsson, "*Investigating the Energy Consumption of a Wireless Network Interface in an Ad hoc Networking Environment*". 2001.

[7] "*What is an smartphone*". **Fuente**:
https://www.techopedia.com/definition/2977/smartphone

[8] "*Punto de acceso inalámbrico"*. **Fuente**:
https://es.wikipedia.org/wiki/Punto_de_acceso_inal%C3%A1mbrico

[9] "*Wifi"*. **Fuente**: https://es.wikipedia.org/wiki/Wifi

[10] "Tableta”(computadora). **Fuente:**
https://es.wikipedia.org/wiki/Tableta_%28computadora%29

[11] “Smartphone”. **Fuente:** http://definicion.de/smartphone/

[12] ”Nodo”. **Fuente:**
https://es.wikipedia.org/wiki/Nodo_(inform%C3%A1tica)

[13] "Java". **Fuente:** https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

[14] "IDE". **Fuente**: https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

[15] "P2P". **Fuente:** https://es.wikipedia.org/wiki/Peer-to-peer

[16] "LBS". **Fuente:** https://es.wikipedia.org/wiki/Servicio_basado_en_localizaci%C3%B3n

[17] "*Definición de WLAN".* **Fuente:** http://definicion.de/wlan/

[18] *"Definición de 3G".* **Fuente:** https://es.wikipedia.org/wiki/Telefon%C3%ADa_m%C3%B3vil_3G

[19] "Definición de 4G". **Fuente:** https://es.wikipedia.org/wiki/Telefon%C3%ADa_m%C3%B3vil_4G

[20] M. Frodigh, P. Johansson, P. Larsson, "*Wireless Ad hoc networking - The art of networking without a network"*. 2000.

[21] "*A Look at the Basics of Bluetooth Technology"*. **Fuente**: https://www.bluetooth.org/pages/basics.aspx

[22] Wi-Fi Alliance, "*FAQ*". **Fuente**:http://www.wi-fi.org/knowledge-center/faq/does-wi-fi-direct-work-on- 80211-abgn

[23] Hughes Systique Corporation, "*Wi-Fi DirectTM*". **Fuente**:hsc.com/Portals/0/Uploads/Articles/WFD_Technology_Whitepaper_v_1.7635035318321315728.pdf

10. ANEXO

10.1 Código Fuentes Simulador Grafico

- **Clase Consulta**

```
package Simulacion;

import java.util.ArrayList;

public class Consulta {
  int ID;
  int idCreador;
  int limite;
  ArrayList <Integer> posicion;
  int consulta;

  public Consulta(){
    this.ID=0;
    this.idCreador=0;
    this.limite=30;
    this.posicion=new ArrayList<>();
    this.consulta=0;
  }

  public Consulta(int newID, int newIdCreador, ArrayList <Integer> newPosicion, int newConsulta){
    this.ID=newID;
    this.idCreador=newIdCreador;
    this.limite=30;
    this.posicion=newPosicion;
    this.consulta=newConsulta;
  }

  public int getID(){
    return ID;
  }

  public int getIdCreador(){
    return idCreador;
  }

  public int getLimite(){
    return limite;
  }

  public ArrayList <Integer> getPosicion(){
    return posicion;
  }
```

```
  public Integer getConsulta(){
    return consulta;
  }

  public void setID(int newID){
    ID=newID;
  }

  public void setIdCreador(int newIdCreador){
   idCreador=newIdCreador;
  }

  public void setPosicion(ArrayList <Integer> newPosicion){
    posicion=newPosicion;
  }

  public void setConsulta(Integer newConsulta){
    consulta=newConsulta;
  }

}
```

- **Clase Grilla**

```
package Simulacion;
import java.util.ArrayList;
public class Grilla {
  ArrayList <Consulta> listaConsultas;
  ArrayList <Integer> IDNodos; //nodos que están en la grilla actualmente
  int ID;
  ArrayList <Double> limitesX;
  ArrayList <Double> limitesY;
  ArrayList <Nodo> home; //lista de nodos que parten desde esta celda, posición se actualiza
a medida de que el nodo cambia de celda.

  public Grilla() {

    this.listaConsultas = new ArrayList<>();
    this.IDNodos = new ArrayList<>();
    this.limitesX = new ArrayList<>();
    this.limitesY = new ArrayList<>();
    this.ID=0;
    this.home=new ArrayList();
  }
  public Grilla(ArrayList <Consulta> newConsultas, ArrayList<Integer> newIDNodos,
ArrayList<Double> newLimitesX, ArrayList<Double> newLimitesY, int newID){
    this.listaConsultas = newConsultas;
    this.IDNodos=newIDNodos;
    this.limitesX = newLimitesX;
```

```
        this.limitesY = newLimitesY;
        this.ID=newID;
        this.home=new ArrayList();
    }

    public ArrayList<Consulta> getListaConsultas(){
        return listaConsultas;
    }

    public ArrayList<Integer> getIDNodos(){
        return IDNodos;
    }

    public ArrayList<Double> getLimitesX(){
        return limitesX;
    }

    public ArrayList<Double> getLimitesY(){
        return limitesY;
    }

    public int getID(){
        return ID;
    }

    public void setListaConsultas(ArrayList <Consulta> newConsultas){
        listaConsultas = newConsultas;
    }

    public void setIDNodos(ArrayList<Integer> newIDNodos){
        IDNodos=newIDNodos;
    }

    public void setLimitesX(ArrayList<Double> newLimitesX){
        limitesX=newLimitesX;
    }

    public void setLimitesY(ArrayList<Double> newLimitesY){
        limitesY=newLimitesY;
    }

    public void setID(int newID){
        ID=newID;
    }
}
```

- **Clase MarcoCentral**

```
package Simulacion;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;


public class MarcoCentral {

   JFrame marco = new JFrame("Simulación");
   boolean Pausar=false;
   //panel de botones a la derecha
   JPanel panelbtnDer = new JPanel();
   //panel de botones inferior
   JPanel panelbtnInf = new JPanel();
   //panel de usuarios
   Panel panel = new Panel();

   //botones a la derecha
   JLabel cantidadUsuarios = new JLabel("Cantidad Usuarios");
   JComboBox numUsuarios = new JComboBox();
   static JButton generarUsuarios = new JButton( "Generar Usuarios" );
   JCheckBox habGrilla = new JCheckBox("Habilitar/Deshabilitar Grilla");
   JButton limpiarGrilla = new JButton( "Limpiar Grilla" );
   JButton pauReanudar = new JButton( "Pausar/Reanudar" );
   JButton generarConsulta = new JButton("Generar Consulta");
   JButton mostrarTabla = new JButton("Tabla de Consultas");
   JButton mostrarTabla2 = new JButton("Tabla de Celdas");
   JButton location = new JButton("Localización");
   JCheckBox autoTable = new JCheckBox("Tabla en tiempo real");

   //botones de la parte inferior
   JComboBox selUsuario = new JComboBox();
   JComboBox consultas = new JComboBox();
   JComboBox movimiento = new JComboBox();
```

```
  MarcoCentral(){
    marco.setSize(1170,720);

    selUsuario.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Seleccionar Usuario", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30","31","32","33","34","35","36","37","38","39","40", "41","42","43","44","45","46","47","48","49","50"}));
    consultas.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Elegir Consultas", "Localización", "S-RMQ" }));
    movimiento.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Elegir Movimiento", "Random Walk", "Random Waypoint" }));
    numUsuarios.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "10", "20", "30","40","50" }));

    marco.getContentPane().setLayout(null);

    //color paneles
    panelbtnInf.setBackground(new Color(77,197,175));
    panelbtnDer.setBackground(new Color(77,197,175));
    habGrilla.setBackground(new java.awt.Color(77,197,175));
    habGrilla.addActionListener(
    new ActionListener(){
      @Override
      public void actionPerformed(ActionEvent ae){
        if(habGrilla.isSelected()){
          panel.seleccionado=true;
        }else{
          panel.seleccionado=false;
        }
      }
    }
    );

    autoTable.setBackground(new java.awt.Color(77,197,175));
    autoTable.addActionListener(
    new ActionListener(){
      @Override
      public void actionPerformed(ActionEvent ae){
        if(autoTable.isSelected()){
          panel.auto=true;
        }else{
          panel.auto=false;
        }
      }
    }
    );

    selUsuario.addActionListener(new ActionListener(){
      @Override
```

```
        public void actionPerformed(ActionEvent ae){
          if(selUsuario.getSelectedItem()!="Seleccionar
Usuario"&&selUsuario.getSelectedItem()!="Deseleccionar"){
            panel.usuarioSeleccionado(Integer.parseInt((String)
selUsuario.getSelectedItem()));
          }else{
            panel.usuarioSeleccionado(1000);
          }
          /*if (selUsuario.getSelectedItem()!="Deseleccionar") {
            panel.usuarioSeleccionado(10000);
          }*/
        }
      });

      //pone los botones del panel inferior horizontalmente con un espacio de 97
      panelbtnInf.setLayout(new FlowLayout(0, 37, 0));

      //tamaño de comboboxes inferior
      selUsuario.setPreferredSize(new Dimension(180,35));
      consultas.setPreferredSize(new Dimension(180,35));
      movimiento.setPreferredSize(new Dimension(180,35));

      //botones inferior añadidos al panel inferior
      panelbtnInf.add(selUsuario);
      panelbtnInf.add(consultas);
      panelbtnInf.add(movimiento);

      //setBounds(x,y,ancho,alto) tamaño paneles
      panel.setBounds(30, 30, 900, 550);
      panelbtnInf.setBounds(0,610,700,35);
      panelbtnDer.setBounds(960,100,220,500);

      //layout para poner botones de la derecha verticalmente
      panelbtnDer.setLayout(new BoxLayout(panelbtnDer,BoxLayout.Y_AXIS));

      //propiedades combobox cantidad de usuarios
      numUsuarios.setBounds(1070,96,45,25);

      //añade botones al panel de botones a la derecha
      panelbtnDer.add(cantidadUsuarios);

      //setBorder(arriba, izq, abajo, der) crea bordes a los botones
      cantidadUsuarios.setBorder(BorderFactory.createEmptyBorder(0,0,20,0));
      panelbtnDer.add(generarUsuarios);
      generarUsuarios.setBorder(BorderFactory.createEmptyBorder(10,32,10,32));
      panelbtnDer.add(Box.createRigidArea(new Dimension(0,20)));
      panelbtnDer.add(limpiarGrilla);
      panelbtnDer.add(Box.createRigidArea(new Dimension(0,20)));//espacio entre botones
      limpiarGrilla.setBorder(BorderFactory.createEmptyBorder(10,44,10,44));
      panelbtnDer.add(pauReanudar);
```

```
    pauReanudar.setBorder(BorderFactory.createEmptyBorder(10,32,10,32));
    panelbtnDer.add(Box.createRigidArea(new Dimension(0,20)));//espacio entre botones
    panelbtnDer.add(generarConsulta);
    generarConsulta.setBorder(BorderFactory.createEmptyBorder(10, 32, 10, 32));
    //panelbtnDer.add(Box.createRigidArea(new Dimension(0,20)));
    //panelbtnDer.add(mostrarTabla);
    mostrarTabla.setBorder(BorderFactory.createEmptyBorder(10, 27, 10, 27));
    panelbtnDer.add(Box.createRigidArea(new Dimension(0,20)));
    panelbtnDer.add(mostrarTabla2);
    mostrarTabla2.setBorder(BorderFactory.createEmptyBorder(10, 35, 10, 37));
    panelbtnDer.add(Box.createRigidArea(new Dimension(0,20)));
    panelbtnDer.add(location);
    location.setBorder(BorderFactory.createEmptyBorder(10, 45, 10, 45));
    generarUsuarios.addActionListener(new Acciones());
    limpiarGrilla.addActionListener(new Acciones());
    generarConsulta.addActionListener(new Acciones());
    pauReanudar.addActionListener(new Acciones());
    mostrarTabla.addActionListener(new Acciones());
    mostrarTabla2.addActionListener(new Acciones());
    location.addActionListener(new Acciones());
    //generarUsuarios.setBorder(BorderFactory.createEmptyBorder(10,32,10,32));
    panelbtnDer.add(habGrilla);
    habGrilla.setBorder(BorderFactory.createEmptyBorder(20,0,10,0));
    panelbtnDer.add(autoTable);
    autoTable.setBorder(BorderFactory.createEmptyBorder(10, 0, 10, 0));
    //añade paneles al marcocentral
    marco.add(numUsuarios);
    marco.add(panel);
    marco.add(panelbtnInf);
    marco.add(panelbtnDer);
    marco.getContentPane().setBackground(new Color(77,197,175));
    marco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    marco.setLocationRelativeTo(null);
    marco.setVisible(true);
}

class Acciones implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent ae) {
      //si se presionó el botón Añadir


      if(ae.getSource() == generarUsuarios){
        generarUsuarios.setEnabled(false);
        //cantidad de usuarios seleccionados de string a int
        int cantUsers = Integer.parseInt((String)numUsuarios.getSelectedItem());

        for(int i=0; i<cantUsers;i++){
           try {
```

```
                //obtiene el diametro y se le pasa como argumento a añadirPelota
                panel.añadirUsuario(40, (i+1));

              } catch (NumberFormatException e) {
                System.out.println(e);
                JOptionPane.showMessageDialog(null,"INTRODUZCA UN VALOR
VÁLIDO","Error",JOptionPane.ERROR_MESSAGE);
                 }
               }
             }
           if(ae.getSource() == limpiarGrilla){
             panel.limpiarGrilla();

           }
           if (ae.getSource()==generarConsulta) {
             panel.generarConsulta(Integer.parseInt((String)numUsuarios.getSelectedItem()));
           }

           if (ae.getSource()== pauReanudar){
              if (Pausar) {
             Pausar=false;
             panel.pauReanudar(Pausar);
           }else{
                 Pausar=true;
             panel.pauReanudar(Pausar);
              }

       }
           if (ae.getSource()==mostrarTabla) {
             panel.tabla();
           }
           if (ae.getSource()==mostrarTabla2) {
             panel.tabla2();
           }
           if (ae.getSource()==location) {
             panel.jOptionPaneLocation();
           }
           }
       }
     }
```

- **MessengerListener**

```
package Simulacion;

import java.util.ArrayList;

public class MessageListener {

   public boolean SearchNode(Nodo Cell, ArrayList<ArrayList<Grilla>> matriz, int x, int y){
      //pregunta si la posición del nodo está dentro de los límites de la celda, si es así, devuelve
true.
      if
((matriz.get(x).get(y).getLimitesX().get(0)<Cell.getMyPos().get(0)&&matriz.get(x).get(y).get
LimitesX().get(1)>=Cell.getMyPos().get(0))&&(matriz.get(x).get(y).getLimitesY().get(0)<Cell
.getMyPos().get(1)&&matriz.get(x).get(y).getLimitesY().get(1)>=Cell.getMyPos().get(1))) {
            return true;
      }else{
         //recorre la lista de consultas que tiene la celda, si la id del nodo aparece en la lista de
id que posee la celda de la matriz, retorna true.
         for (int j = 0; j < matriz.get(x).get(y).getListaConsultas().size(); j++) {
            if
(matriz.get(x).get(y).getListaConsultas().get(j).getID()==Cell.getMyRetains().get(j).getID()) {
               return true;
            }
         }
      }
      return false;
   }
}
```

- **Clase Nodo**

```
package Simulacion;

import java.util.ArrayList;

//Clase para hacer con las variables para S-RMQ, nota todo debe ir dentro de la clase
//principal
public class Nodo {
 ArrayList<Grilla> myRetains;
 Grilla myCache;
 Usuario usuario;
 int myID;
 ArrayList<Integer> myPos;
 ArrayList<Integer> myPosAnterior;
 int myCell; //celda de partida del nodo
 ArrayList<String> myQueries;// lista de consulta relevantes para myCell
 //ArrayList<String> myRetains; //lista de celdas en las que el nodo actualmente esta
detenido // y sus consultas relevantes
 ArrayList<Integer> misRespuestas;
```

```
 boolean esCreador;
 ArrayList<Queries> consulCreadas;
//constructor
  public Nodo(){
     this.myID=0;
     this.myPos=new ArrayList<>();
     this.myPosAnterior=new ArrayList<>();
     this.myCell=0;
     this.myQueries=new ArrayList<>();
     this.myCache=new Grilla();
     //this.myRetains=null;
     this.myRetains=new ArrayList<>();
     this.misRespuestas=new ArrayList<>();
     this.esCreador=false;
     consulCreadas=new ArrayList<>();
  }

  public Nodo(int nuevaID, ArrayList<Integer> newPos,int nuevaCell, ArrayList<String>
nuevaQ, ArrayList <Grilla> newMyRetains, ArrayList<Integer> nuevasRespuestas, boolean
siono){
     this.myID=nuevaID;
     this.myPos=newPos;
     this.myPosAnterior=new ArrayList<>();
     this.myCell=nuevaCell;
     this.myQueries=nuevaQ;
     this.myCache=null;
     //this.myRetains=nuevoR;
     this.myRetains=newMyRetains;
     this.misRespuestas=nuevasRespuestas;
      this.esCreador=siono;
      consulCreadas=new ArrayList<>();
  }

  //get es obtener y para ingresar set
  public ArrayList<Grilla> getMyRetains(){
     return myRetains;
  }
  public Grilla getMyCache(){
     return myCache;
  }
  public Usuario getUsuario(){
     return usuario;
  }
  public ArrayList<Integer> getMyPos(){
     return myPos;
  }
  public ArrayList<Integer> getMyPosAnterior(){
     return myPosAnterior;
  }
  public int getMyId(){
```

```
    return myID;
}
public int getMyCell(){
    return myCell;
}
public ArrayList<String> getMyQueries(){
     return myQueries;
}
/*public ArrayList<String> getMyRetains(){
     return myRetains;
}*/
public ArrayList<Integer> getMisRespuestas(){
     return misRespuestas;
}

public boolean getEsCreador(){
     return esCreador;
}

public ArrayList<Queries> getConsulCreadas(){
  return consulCreadas;
}

public void setMyRetains(ArrayList<Grilla> newMyRetains){
  for (int i = 0; i < newMyRetains.size(); i++) {
    myRetains.add(newMyRetains.get(i));
  }
}
public void setMyCache(Grilla newMyCache){
  myCache=newMyCache;
}
public void setUsuario(Usuario newUsuario){
  usuario=newUsuario;
}
public void setMyId(int NuevaId ){
  myID=NuevaId;
}
public void setMyPos(ArrayList<Integer> newPos){
  for (int i = 0; i < newPos.size(); i++) {
    myPos.add(newPos.get(i));
  }
}
public void setMyPosAnterior(ArrayList<Integer> newPosAnterior){
  for (int i = 0; i < newPosAnterior.size(); i++) {
    myPosAnterior.add(newPosAnterior.get(i));
  }
}
public void setMyCell(int NuevaCell){
 myCell=NuevaCell;
}
```

```
  public void setMyQueries(ArrayList<String> NuevaQueries){
   for (int i = 0; i < NuevaQueries.size(); i++) {
      myQueries.add(NuevaQueries.get(i));
    }
  }
  /*public void setMyRetains(ArrayList<String> NuevaRetains){
    for (int i = 0; i < NuevaRetains.size(); i++) {
      myRetains.add(NuevaRetains.get(i));
    }
  }*/
  public void setMisRespuestas(ArrayList<Integer> NuevasRespuestas){
    for (int i = 0; i < NuevasRespuestas.size(); i++) {
      misRespuestas.add(NuevasRespuestas.get(i));
    }
  }

  public void setEsCreador(boolean siono){
    esCreador=siono;
  }

  public void setConsulCreadas(ArrayList<Queries> newConsulCreadas){
    consulCreadas.clear();
    for (int i = 0; i < newConsulCreadas.size(); i++) {
      consulCreadas.add(newConsulCreadas.get(i));
    }
  }

  public void addConsulCreadas (Queries newConsulCreada){
    consulCreadas.add(newConsulCreada);
  }
}
```

- **Clase Panel**

```
package Simulacion;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.geom.Line2D;
import java.util.ArrayList;
import java.util.Random;
import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JComboBox;
import javax.swing.JComponent;
import javax.swing.JOptionPane;
```

```
public class Panel extends JComponent{
   RegionMonitor RM = new RegionMonitor();
   location L = new location();
   int usuarioSel=10000;
   static int anchura = 900;//anchura, altura panel
   public static int altura = 550;
   static Image imagenUser;
   static Image imagenUserSel;
   Usuario usuario;
   boolean seleccionado=false, pausado=false, auto=false;
   Nodo usuarioCreador = new Nodo();
   int pos=-1;
   int auxSum=0;

   ArrayList <Nodo> conjuntoUsuarios = new ArrayList<>();
   ArrayList<Consulta> consultas = new ArrayList<>();

   Panel(){
      ImageIcon img = new ImageIcon(this.getClass().getResource("/Imagenes/user.png"));
      ImageIcon imgSel = new
ImageIcon(this.getClass().getResource("/Imagenes/phone.png"));
      imagenUser = img.getImage();
      imagenUserSel = imgSel.getImage();
      //tamaño panel
      setPreferredSize(new Dimension(anchura, altura));
      //Bordear el panel, enmarcarlo
      setBorder(BorderFactory.createEtchedBorder());
      //inicializar la variable
      }

   public void paintComponent(Graphics gg){
      wrapArray mix=new wrapArray();
      Graphics2D g = (Graphics2D) gg;
      super.paintComponent(g);
      g.setColor(new Color(245,245,220));//color panel
            //0->fila inicial, 0->columna inicial
      g.fillRect(0,0, getWidth(),getHeight());//pintar el panel

      try {
         //recorre el arraylist de
         for(int i=0;i < conjuntoUsuarios.size();i++){
            ArrayList<Integer> auxPos;

            //pintar cada pelota del array
            if (i==usuarioSel) {
               //auxPos=usuario.pintarUsuario(g, (i+1), true, conjuntoUsuarios, consultas);
               auxPos=conjuntoUsuarios.get(i).usuario.pintarUsuario(g, (i+1), true,
conjuntoUsuarios, consultas);
```

```
            }else{
              //auxPos=usuario.pintarUsuario(g, (i+1), false, conjuntoUsuarios, consultas);
              auxPos=conjuntoUsuarios.get(i).usuario.pintarUsuario(g,            (i+1),            false,
conjuntoUsuarios, consultas);
            }
            if (pos!=-1) {
              consultas=RM.devolverConsultas();
            }
            if (!conjuntoUsuarios.get(i).getMyPos().isEmpty()) {
              conjuntoUsuarios.get(i).getMyPosAnterior().clear();

conjuntoUsuarios.get(i).getMyPosAnterior().add(conjuntoUsuarios.get(i).getMyPos().get(0))
;

conjuntoUsuarios.get(i).getMyPosAnterior().add(conjuntoUsuarios.get(i).getMyPos().get(1))
;
            }
            conjuntoUsuarios.get(i).getMyPos().clear();
            conjuntoUsuarios.get(i).getMyPos().add(auxPos.get(0));
            conjuntoUsuarios.get(i).getMyPos().add(auxPos.get(1));
            conjuntoUsuarios=RM.usuario(conjuntoUsuarios, i, pos);
            if (auto&&i==conjuntoUsuarios.size()-1) {
             // tabla();
              tabla2();
            }
            pos=-1;
          }
          Thread.sleep(300);//hilo que mueve los usuarios con los movimientos random walk
        } catch (Exception e) {

System.out.println("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!");
          System.out.println("Error al pintar usuario");
          System.out.println("Mensaje: "+e);

System.out.println("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!");
        }
        String iterator;
        int cont;
        if(seleccionado){
            Line2D.Double shape;
          g.setColor(Color.black);
          for (int i = 0; i < 14; i++) {
            shape = new Line2D.Double(60*(i+1), 0, 60*(i+1), 550);
            g.draw(shape);
          }
          for (int i = 0; i < 8; i++) {
            shape = new Line2D.Double(0, 61.11112*(i+1), 900, 61.11112*(i+1));
            g.draw(shape);
```

```
        }
        cont=0;
        for (int i = 0; i < 15; i++) {
          for (int j = 0; j < 9; j++) {
            cont++;
            iterator=Integer.toString(cont);
            g.drawString(iterator, (int)(i*60.0+25), (int)(j*61.11112+35));
          }
        }
      }
        if (!pausado) {
        repaint();//repintar el panel
//        auxSum++;
//          if (auxSum==1) {
//            tabla();
//            auxSum=0;
//          }

      }

    }

    void usuarioSeleccionado(int usuario){
      usuarioSel=usuario;
    }

    void añadirUsuario(int diametro, int ID){
      //añade al array cada pelota con el diametro indicado
      Nodo auxNodo=new Nodo();
      ArrayList<Integer>newPos=new ArrayList<>();

      auxNodo.usuario=new Usuario(diametro); //diámetro de la imagen
      auxNodo.setMyId(ID); //id del usuario
      newPos.add(auxNodo.usuario.columna); //posición x inicial
      newPos.add(auxNodo.usuario.fila); //posición y inicial
      auxNodo.setMyPos(newPos);
      conjuntoUsuarios.add(auxNodo);
      RM.llenarMatriz(ID, newPos);
    }

    void generarConsulta(int cantUsuarios){
      if (!conjuntoUsuarios.isEmpty()) {
        pos=(int) (Math.random()*cantUsuarios);
        usuarioSeleccionado(pos);
        //conjuntoUsuarios.get(pos).usuario.pintarConsulta(pos);
      }
    }

    void limpiarGrilla(){
      System.out.println(conjuntoUsuarios.size());
```

```
      try {

        if(!conjuntoUsuarios.isEmpty()){
          conjuntoUsuarios.removeAll(conjuntoUsuarios);
        }
        MarcoCentral.generarUsuarios.setEnabled(true);
      } catch (Exception e) {
        System.out.println("catch");
      }
    }

    void pauReanudar(boolean Pausar){
      if (Pausar) {
        pausado=true;
      }else{
        pausado=false;
        repaint();
      }

}
    void tabla(){
      RM.mostrarDatosConsulta(conjuntoUsuarios);
    }
    void tabla2(){
      RM.mostrarDatosTabla(conjuntoUsuarios);
    }

    void jOptionPaneLocation(){
      int x, idCreador, posU=0, posC=0;
      double distancia, distMin=100000000;
      int[] posConsulta=new int[2];
      ArrayList<Integer> recorrido= new ArrayList<>();
      ArrayList<Nodo> candidatos=new ArrayList<>();
      if(consultas.isEmpty()){
        JOptionPane.showMessageDialog(null, "No hay consultas para mostrar");
      }else{
        x=consultas.size();
        Integer [] Consultas = new Integer[x];
        for (int i = 0; i < consultas.size(); i++) {
            Consultas[i]=i;
        }
        pausado=true;
        Integer seleccion;
        seleccion =(int)JOptionPane.showInputDialog(null, "Ingrese la consulta para ubicar a
su creador", "Localizar al creador", JOptionPane.QUESTION_MESSAGE, null, Consultas,
Consultas[0]);
        System.out.println("selección: "+seleccion);
        idCreador=consultas.get(seleccion).getIdCreador();
        posConsulta[0]=consultas.get(seleccion).getPosicion().get(0);
```

```
        posConsulta[1]=consultas.get(seleccion).getPosicion().get(1);
        for (int i = 0; i < conjuntoUsuarios.size(); i++) {
          if (conjuntoUsuarios.get(i).getMyId()!=idCreador) {
            distancia=Math.sqrt(Math.pow((posConsulta[0]-
conjuntoUsuarios.get(i).getMyPos().get(0)), 2)+Math.pow((posConsulta[1]-
conjuntoUsuarios.get(i).getMyPos().get(1)), 2));
            if (distancia<distMin) {
              distMin=distancia;
              posU=i;
            }
          }else{
            posC=i;
          }
        }
        recorrido.add(conjuntoUsuarios.get(posU).getMyId());
        recorrido=L.mapeo(recorrido, conjuntoUsuarios.get(posU), conjuntoUsuarios,
conjuntoUsuarios.get(posC).getMyId(), candidatos);
        System.out.println("Localizando al usuario...");
        for (int i = 0; i < recorrido.size(); i++) {
          System.out.println("-> ID: "+recorrido.get(i));
        }
        if (recorrido.size()>1) {
          String camino="";
          for (int i = 0; i < recorrido.size(); i++) {
            if (i==0) {
              camino=""+recorrido.get(i);
            }else{
              camino=camino+"=>"+recorrido.get(i);
            }

          }
          JOptionPane.showMessageDialog(null, camino, "Camino",
JOptionPane.DEFAULT_OPTION, null);
        }else{
          JOptionPane.showMessageDialog(null, "El nodo creador está demasiado lejos para
ser ubicado", "Error", JOptionPane.DEFAULT_OPTION, null);
        }

      }
    }
}
```

- **Clase Queries**

```
package Simulacion;

import java.util.ArrayList;


public class Queries {
   int idConsulta;
   ArrayList<Nodo> nodos;

   public Queries(){
      this.idConsulta=-1;
      this.nodos=new ArrayList<>();
   }

   public Queries(int newIdConsulta){
      this.idConsulta=newIdConsulta;
      this.nodos=new ArrayList<>();
   }

   public int getIdConsulta(){
      return idConsulta;
   }

   public void setIdConsulta(int newIdConsulta){
      idConsulta=newIdConsulta;
   }

   public ArrayList<Nodo> getNodos(){
      return nodos;
   }

   public void setNodos(ArrayList<Nodo> newNodos){
      nodos.clear();
      for (int i = 0; i < newNodos.size(); i++) {
         nodos.add(newNodos.get(i));
      }
   }

   public void addNodos(Nodo newNodo){
      nodos.add(newNodo);
   }
}
```

- **Clase RegionMonitor**

```
package Simulacion;

import java.util.ArrayList;


public class RegionMonitor {
    ArrayList<Consulta> consultas=new ArrayList<>();
    ArrayList<ArrayList<Grilla>> matriz=new ArrayList<>();
    MessageListener ML = new MessageListener();
    tabla T=new tabla();
    tablaCelda TC=new tablaCelda();

  public RegionMonitor(){
    ArrayList<Grilla> auxMatriz=new ArrayList<>();
    Grilla auxGrilla;
    int valor=0;
    for (int i = 0; i < 15; i++) {
      for (int j = 0; j < 9; j++) {
        auxGrilla=new Grilla();
        valor++;
        auxGrilla.setID(valor);
        auxGrilla.getLimitesX().add(i*60.0);
        auxGrilla.getLimitesX().add((i*60.0)+59);
        auxGrilla.getLimitesY().add(j*61.1);
        auxGrilla.getLimitesY().add((j*61.1)+60.1);
        auxMatriz.add(auxGrilla);

      }
      matriz.add(new ArrayList(auxMatriz));
      auxMatriz.clear();
    }
  }

  ArrayList<Consulta> devolverConsultas(){
    return consultas;
  }

  public void llenarMatriz(int idUsuario, ArrayList<Integer> coords){
    int x=(int)Math.floor(coords.get(0)/60);
    int y=(int)Math.floor(coords.get(1)/61.11112);
    matriz.get(x).get(y).getIDNodos().add(idUsuario);
  }

  public void mostrarDatosConsulta(ArrayList<Nodo> conjuntoUsuarios){
      T.ingresarDatos(consultas, conjuntoUsuarios, matriz);
  }

  public void mostrarDatosTabla(ArrayList<Nodo> conjuntoUsuarios){
```

```
        TC.mostrarDatos(matriz, conjuntoUsuarios, consultas);
    }

    public int cargarVertices(Consulta nuevaConsulta, int limite, int xy){
        int vertice;
        if (xy==0) {
            vertice=(int)Math.floor((nuevaConsulta.getPosicion().get(0)+limite)/60);
            if (vertice<0||vertice>14) {
                vertice=(int)Math.floor((nuevaConsulta.getPosicion().get(0))/60);
            }
        }else{
            vertice=(int)Math.floor((nuevaConsulta.getPosicion().get(1)+limite)/61.11112);
            if (vertice<0||vertice>8) {
                vertice=(int)Math.floor((nuevaConsulta.getPosicion().get(1))/61.11112);
            }
        }
        return vertice;
    }

    public ArrayList<Nodo> consultasUsuarios(ArrayList<Nodo> conjuntoUsuarios, int i, Consulta
nuevaConsulta, int id){

        if (conjuntoUsuarios.get(i).getMyCache().getListaConsultas().isEmpty()) {
            conjuntoUsuarios.get(i).getMyCache().setID(id);
            conjuntoUsuarios.get(i).getMyCache().getListaConsultas().add(nuevaConsulta);
        }else{
            conjuntoUsuarios.get(i).getMyCache().setID(id);
            for (int j = 0; j < conjuntoUsuarios.get(i).getMyCache().getListaConsultas().size(); j++) {
                if (conjuntoUsuarios.get(i).getMyCache().getID()!=id) {
                    System.out.println("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! NO ES EL ID CORRESPONDIENTE
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
                }
                if
(conjuntoUsuarios.get(i).getMyCache().getListaConsultas().get(j).getID()==nuevaConsulta.getID())
{
                    break;
                }
                if (j==conjuntoUsuarios.get(i).getMyCache().getListaConsultas().size()-1) {
                    conjuntoUsuarios.get(i).getMyCache().getListaConsultas().add(nuevaConsulta);
                }
            }
        }

        return conjuntoUsuarios;
    }

    public ArrayList<Nodo> calcularDistancias(ArrayList<Nodo> conjuntoUsuarios, int[] arista, int
posUser, Consulta nuevaConsulta, ArrayList <Consulta> auxLista, int id){

        boolean breakDoble=false;
```

```
    double dist1, dist2;
        for (int i = 0; i < conjuntoUsuarios.size(); i++) {
          for (int j = 0; j < conjuntoUsuarios.get(i).getMyRetains().size(); j++) {
            if (conjuntoUsuarios.get(i).getMyRetains().get(j).getID()==id) {
              //distancia desde el usuario i hasta la mitad de la celda consultada
              dist1=Math.sqrt(Math.pow(((arista[0]*60+(60/2))-
conjuntoUsuarios.get(i).getMyPos().get(0)),    2)+Math.pow(((arista[1]*61.11112+(61.11112/2))-
conjuntoUsuarios.get(i).getMyPos().get(1)), 2));
              //distancia del creador de la consulta a la mitad de la celda consultada
              dist2=Math.sqrt(Math.pow(((arista[0]*60+(60/2))-
conjuntoUsuarios.get(posUser).getMyPos().get(0)),
2)+Math.pow(((arista[1]*61.11112+(61.11112/2))-
conjuntoUsuarios.get(posUser).getMyPos().get(1)), 2));
              if (dist1>dist2) {

conjuntoUsuarios.get(i).getMyRetains().get(j).getListaConsultas().add(nuevaConsulta);

conjuntoUsuarios.get(posUser).getMyRetains().add(conjuntoUsuarios.get(i).getMyRetains().get(j))
;
                 conjuntoUsuarios.get(i).getMyRetains().remove(j);
              }else{

conjuntoUsuarios.get(i).getMyRetains().get(j).getListaConsultas().add(nuevaConsulta);
              }
              breakDoble=true;
              break;
            }
          }
          if (breakDoble) {
            break;
          }
          if (i==conjuntoUsuarios.size()-1) {
            if (conjuntoUsuarios.get(posUser).getMyRetains().isEmpty()) {
              Grilla aGrilla = new Grilla(auxLista,matriz.get(arista[0]).get(arista[1]).getIDNodos(),
matriz.get(arista[0]).get(arista[1]).getLimitesX(),matriz.get(arista[0]).get(arista[1]).getLimitesY(),
matriz.get(arista[0]).get(arista[1]).getID());
              conjuntoUsuarios.get(posUser).getMyRetains().add(aGrilla);
              System.out.println("El usuario "+posUser+" se ha vuelto retenedor de la celda "+id);
            }
            else{
              for (int j = 0; j < conjuntoUsuarios.get(posUser).getMyRetains().size(); j++) {
                if (conjuntoUsuarios.get(posUser).getMyRetains().get(j).getID()==id) {

conjuntoUsuarios.get(posUser).getMyRetains().get(j).getListaConsultas().add(nuevaConsulta);
                  break;
                }
                if (j==conjuntoUsuarios.get(posUser).getMyRetains().size()-1) {
                  Grilla aGrilla = new
Grilla(auxLista,matriz.get(arista[0]).get(arista[1]).getIDNodos(),
```

```
matriz.get(arista[0]).get(arista[1]).getLimitesX(),matriz.get(arista[0]).get(arista[1]).getLimitesY(),
matriz.get(arista[0]).get(arista[1]).getID());
                    conjuntoUsuarios.get(posUser).getMyRetains().add(aGrilla);
                    System.out.println("El usuario "+posUser+" se ha vuelto retenedor de la celda
"+id);
                }
              }
            }
          }
        }


    return conjuntoUsuarios;
  }

  public ArrayList<Nodo> ingresarConsulta(int posUser, ArrayList<Nodo> conjuntoUsuarios){
    Consulta nuevaConsulta;
    int idConsulta;
    int idUser=conjuntoUsuarios.get(posUser).getMyId();
    int xActual, yActual;
    ArrayList<Integer> pos=new ArrayList<>();
    pos.add(conjuntoUsuarios.get(posUser).getMyPos().get(0));
    pos.add(conjuntoUsuarios.get(posUser).getMyPos().get(1));
    int cantUsuarios=0;
    Queries newConsulCreada;
    ArrayList <Consulta> auxLista = new ArrayList<>();
    xActual=(int)Math.floor((conjuntoUsuarios.get(posUser).getMyPos().get(0))/60.0);
    yActual=(int)Math.floor((conjuntoUsuarios.get(posUser).getMyPos().get(1))/61.11112);
    int xUser;
    int yUser;
    int idCeldaActual=matriz.get(xActual).get(yActual).getID();
    if (consultas.isEmpty()) {
      idConsulta=0;
    }else{
      idConsulta=consultas.size();
    }
    nuevaConsulta=new Consulta(idConsulta,idUser ,pos, cantUsuarios);
    consultas.add(nuevaConsulta);
    newConsulCreada=new Queries(idConsulta);
    conjuntoUsuarios.get(posUser).addConsulCreadas(newConsulCreada);
    auxLista.add(nuevaConsulta);
    //Grilla auxGrilla = new Grilla(auxLista,matriz.get(xActual).get(yActual).getIDNodos(),
matriz.get(xActual).get(yActual).getLimitesX(),matriz.get(xActual).get(yActual).getLimitesY(),
matriz.get(xActual).get(yActual).getID());
    if (consultas.size()>0) {
      int[] iInf=new int[2], iSup=new int[2], dInf=new int[2], dSup=new int[2];
      int limiteC=nuevaConsulta.getLimite();
      iInf[0]=cargarVertices(nuevaConsulta, -limiteC, 0);
      iInf[1]=cargarVertices(nuevaConsulta, limiteC, 1);
      iSup[0]=cargarVertices(nuevaConsulta, -limiteC, 0);
```

```
        iSup[1]=cargarVertices(nuevaConsulta, -limiteC, 1);
        dInf[0]=cargarVertices(nuevaConsulta, limiteC, 0);
        dInf[1]=cargarVertices(nuevaConsulta, limiteC, 1);
        dSup[0]=cargarVertices(nuevaConsulta, limiteC, 0);
        dSup[1]=cargarVertices(nuevaConsulta, -limiteC, 1);

        int id1=matriz.get(iInf[0]).get(iInf[1]).getID(), id2=matriz.get(dInf[0]).get(dInf[1]).getID(),
id3=matriz.get(iSup[0]).get(iSup[1]).getID(), id4=matriz.get(dSup[0]).get(dSup[1]).getID();
        int cont1=0, cont2=0, cont3=0, cont4=0;
        boolean u1=false, u2=false, u3=false, u4=false;
        System.out.println("valor de las ids:");
        System.out.println("idCeldaActual: "+idCeldaActual);
        System.out.println("id1: "+id1);
        System.out.println("id2: "+id2);
        System.out.println("id3: "+id3);
        System.out.println("id4: "+id4);
        for (int i = 0; i < conjuntoUsuarios.size(); i++) {
          if (i!=posUser) {
            xUser=(int)Math.floor((conjuntoUsuarios.get(i).getMyPos().get(0))/60.0);
            yUser=(int)Math.floor((conjuntoUsuarios.get(i).getMyPos().get(1))/61.11112);
            if (matriz.get(xUser).get(yUser).getID()==idCeldaActual) {
              conjuntoUsuarios=consultasUsuarios(conjuntoUsuarios, i, nuevaConsulta,
idCeldaActual);
                System.out.println("Ingreso la nueva consulta en el usuario "+i+" que está en la celda
"+idCeldaActual);
                  System.out.println("Las consultas del usuario son: ");
                  for (int j = 0; j < conjuntoUsuarios.get(i).getMyCache().getListaConsultas().size(); j++)
{
                    System.out.println("-
>"+conjuntoUsuarios.get(i).getMyCache().getListaConsultas().get(j).getID());
                  }
            }
            if (idCeldaActual!=id1) {
              cont1++;
              if (matriz.get(xUser).get(yUser).getID()==id1) {
                u1=true;
                conjuntoUsuarios=consultasUsuarios(conjuntoUsuarios, i, nuevaConsulta, id1);
                System.out.println("Ingreso la nueva consulta en el usuario "+i+" que está en la
celda"+id1);
                System.out.println("Las consultas del usuario son: ");
                for (int j = 0; j < conjuntoUsuarios.get(i).getMyCache().getListaConsultas().size(); j++)
{
                  System.out.println("-
>"+conjuntoUsuarios.get(i).getMyCache().getListaConsultas().get(j).getID());
                }

              }
            }
            if (idCeldaActual!=id2 && id2!=id1) {
              cont2++;
```

```
                if (matriz.get(xUser).get(yUser).getID()==id2) {
                  u2=true;
                  conjuntoUsuarios=consultasUsuarios(conjuntoUsuarios, i, nuevaConsulta, id2);
                  System.out.println("Ingreso la nueva consulta en el usuario "+i+" que está en la
celda"+id2);
                  System.out.println("Las consultas del usuario son: ");
                  for (int j = 0; j < conjuntoUsuarios.get(i).getMyCache().getListaConsultas().size(); j++)
{
                    System.out.println("-
>"+conjuntoUsuarios.get(i).getMyCache().getListaConsultas().get(j).getID());
                  }
                }
              }
              if (idCeldaActual!=id3 && id3!=id2 && id3!=id1) {
                cont3++;
                if (matriz.get(xUser).get(yUser).getID()==id3) {
                  u3=true;
                  conjuntoUsuarios=consultasUsuarios(conjuntoUsuarios, i, nuevaConsulta, id3);
                  System.out.println("Ingreso la nueva consulta en el usuario "+i+" que está en la
celda"+id3);
                  System.out.println("Las consultas del usuario son: ");
                  for (int j = 0; j < conjuntoUsuarios.get(i).getMyCache().getListaConsultas().size(); j++)
{
                    System.out.println("-
>"+conjuntoUsuarios.get(i).getMyCache().getListaConsultas().get(j).getID());
                  }
                }
              }
              if (idCeldaActual!=id4 && id4!=id3 && id4!=id2 && id4!=id1) {
                cont4++;
                if (matriz.get(xUser).get(yUser).getID()==id4) {
                  u4=true;
                  conjuntoUsuarios=consultasUsuarios(conjuntoUsuarios, i, nuevaConsulta, id4);
                  System.out.println("Ingreso la nueva consulta en el usuario "+i+" que está en la
celda"+id4);
                  System.out.println("Las consultas del usuario son: ");
                  for (int j = 0; j < conjuntoUsuarios.get(i).getMyCache().getListaConsultas().size(); j++)
{
                    System.out.println("-
>"+conjuntoUsuarios.get(i).getMyCache().getListaConsultas().get(j).getID());
                  }
                }
              }
            }
            if (i==conjuntoUsuarios.size()-1) {
              conjuntoUsuarios.get(posUser).getMyCache().setID(idCeldaActual);
              conjuntoUsuarios.get(posUser).getMyCache().getListaConsultas().add(nuevaConsulta);
              //conjuntoUsuarios.get(posUser).setMyCache(auxGrilla);
              System.out.println("mi cache actual es:");
              System.out.println("->ID celda: "+conjuntoUsuarios.get(posUser).getMyCache().getID());
```

```
            for (int j = 0; j < conjuntoUsuarios.get(posUser).getMyCache().getListaConsultas().size();
j++) {
                System.out.println("--
>"+conjuntoUsuarios.get(posUser).getMyCache().getListaConsultas().get(j).getID());
            }

          }
        }
        //este if corresponde a las coordenadas de iInf
        if (cont1>0&&u1==false) {
          System.out.println("inicio la etapa de la esquina inferior izquierda");
          conjuntoUsuarios=calcularDistancias(conjuntoUsuarios, iInf, posUser, nuevaConsulta,
auxLista, id1);
        }
        //este if corresponde a dInf o id2
        if (cont2>0&&u2==false) {
          System.out.println("inicio la etapa de la esquina inferior derecha");
          conjuntoUsuarios=calcularDistancias(conjuntoUsuarios, dInf, posUser, nuevaConsulta,
auxLista, id2);
        }
        //este if corresponde a id3 o iSup
        if (cont3>0&&u3==false) {
          System.out.println("inicio la etapa de la esquina superior izquierda");
          conjuntoUsuarios=calcularDistancias(conjuntoUsuarios, iSup, posUser, nuevaConsulta,
auxLista, id3);
        }
        // este if corresponde a id4 o dSup
        if (cont4>0&&u4==false) {
          System.out.println("inicio la etapa de la esquina superior derecha");
          conjuntoUsuarios=calcularDistancias(conjuntoUsuarios, dSup, posUser, nuevaConsulta,
auxLista, id4);
        }
      }
      System.out.println("Consulta creada\n-----------------------------------");
      System.out.println("id de la consulta: "+idConsulta);
      System.out.println("Datos de la consulta:");
      System.out.println("- ID Usuario: "+idUser);
      System.out.println("- Posición del usuario: ["+pos.get(0)+","+pos.get(1)+"]");
      System.out.println("La cantidad de consultas en el usuario es:
"+conjuntoUsuarios.get(posUser).getConsulCreadas().size());
      System.out.println("\n-------------------------------------------------\n");

      return conjuntoUsuarios;
    }


    public ArrayList <Nodo> usuario(ArrayList <Nodo> conjuntoUsuarios, int posCU, int
posConsulta){
      int x=0, y=1, i, superiorX, superiorY, inferiorX, inferiorY, j;
```

```
    if (posConsulta!=-1) {
      conjuntoUsuarios=ingresarConsulta(posConsulta, conjuntoUsuarios);
    }

    //ArrayList <Integer>  superiorIzquierda = new ArrayList<>(), superiorDerecha = new
ArrayList<>(), inferiorIzquierda = new ArrayList<>(), inferiorDerecha = new ArrayList<>(),
esquina = new ArrayList<>();
    if(!consultas.isEmpty()){
      //recorre la lista de consultas buscando si el usuario está dentro de los límites de las
consultas existentes.
      i=0;
      while(i<consultas.size()){
        superiorX=consultas.get(i).getPosicion().get(x)+consultas.get(i).getLimite();
        inferiorX=consultas.get(i).getPosicion().get(x)-consultas.get(i).getLimite();
        superiorY=consultas.get(i).getPosicion().get(y)+consultas.get(i).getLimite();
        inferiorY=consultas.get(i).getPosicion().get(y)-consultas.get(i).getLimite();
        j=consultas.get(i).getIdCreador()-1;
        if
((conjuntoUsuarios.get(posCU).getMyPos().get(x)<(superiorX))&&(conjuntoUsuarios.get(posCU).g
etMyPos().get(x)>=(inferiorX))&&(conjuntoUsuarios.get(posCU).getMyPos().get(y)<(superiorY))&
&(conjuntoUsuarios.get(posCU).getMyPos().get(y)>=(inferiorY))) {
          for (int k = 0; k < conjuntoUsuarios.get(j).getConsulCreadas().size(); k++) {
            if
(conjuntoUsuarios.get(j).getConsulCreadas().get(k).getIdConsulta()==consultas.get(i).getID()) {
              if (conjuntoUsuarios.get(j).getConsulCreadas().get(k).getNodos().isEmpty()) {

conjuntoUsuarios.get(j).getConsulCreadas().get(k).addNodos(conjuntoUsuarios.get(posCU));
              }else{
                for (int l = 0; l <
conjuntoUsuarios.get(j).getConsulCreadas().get(k).getNodos().size(); l++) {
                  if
(conjuntoUsuarios.get(j).getConsulCreadas().get(k).getNodos().get(l).getMyId()==conjuntoUsuario
s.get(posCU).getMyId()) {
                    break;
                  }
                  if (l==conjuntoUsuarios.get(j).getConsulCreadas().get(k).getNodos().size()-1) {

conjuntoUsuarios.get(j).getConsulCreadas().get(k).addNodos(conjuntoUsuarios.get(posCU));
                  }
                }
              }
              break;
            }
          }
        }else{
          for (int k = 0; k < conjuntoUsuarios.get(j).getConsulCreadas().size(); k++) {
            if
(conjuntoUsuarios.get(j).getConsulCreadas().get(k).getIdConsulta()==consultas.get(i).getID()) {
```

```
                for (int l = 0; l < conjuntoUsuarios.get(j).getConsulCreadas().get(k).getNodos().size();
l++) {
                  if
(conjuntoUsuarios.get(j).getConsulCreadas().get(k).getNodos().get(l).getMyId()==conjuntoUsuario
s.get(posCU).getMyId()) {
                    conjuntoUsuarios.get(j).getConsulCreadas().get(k).getNodos().remove(l);
                    break;
                  }
                }
                break;
              }
            }
          }
          i++;
        }
      }
      conjuntoUsuarios=mensajes(conjuntoUsuarios, posCU);
      return conjuntoUsuarios;
    }

    public ArrayList <Nodo> mensajes(ArrayList <Nodo> conjuntoUsuarios, int posCU){
      Grilla cacheAnterior= new Grilla();
      ArrayList<Nodo> candidateListAnterior = new ArrayList<>();
      int xActual, yActual, xAnterior, yAnterior, xUserConj, yUserConj, idCeldaActual;
      boolean existeAnterior=false, freno=false;
      //encuentra x e y de la celda actual en donde está y la celda anterior
      xActual=(int)Math.floor((conjuntoUsuarios.get(posCU).getMyPos().get(0))/60.0);
      yActual=(int)Math.floor((conjuntoUsuarios.get(posCU).getMyPos().get(1))/61.11112);
      xAnterior=(int)Math.floor((conjuntoUsuarios.get(posCU).getMyPosAnterior().get(0))/60.0);

yAnterior=(int)Math.floor((conjuntoUsuarios.get(posCU).getMyPosAnterior().get(1))/61.11112);
      idCeldaActual=matriz.get(xActual).get(yActual).getID();
      //si la celda en la que está ahora no es igual a la que estaba antes, entonces se borra a sí mismo
de esa celda y se ingresa en la celda que está ahora (mensajes "good bye" y "hello"
respectivamente).
      if(xActual!=xAnterior||yActual!=yAnterior){
        cacheAnterior.setID(conjuntoUsuarios.get(posCU).getMyCache().getID());
        cacheAnterior.setIDNodos(conjuntoUsuarios.get(posCU).getMyCache().getIDNodos());
        cacheAnterior.setLimitesX(conjuntoUsuarios.get(posCU).getMyCache().getLimitesX());
        cacheAnterior.setLimitesY(conjuntoUsuarios.get(posCU).getMyCache().getLimitesY());

cacheAnterior.setListaConsultas(conjuntoUsuarios.get(posCU).getMyCache().getListaConsultas());
        System.out.println("Prueba 1-0. ID celda en cacheAnterior: "+cacheAnterior.getID()+" y
posee "+cacheAnterior.getListaConsultas().size()+" consultas");
        for (int i = 0; i < cacheAnterior.getListaConsultas().size(); i++) {
          System.out.println("La consulta "+(i+1)+" de este cache posee las coordenadas
["+cacheAnterior.getListaConsultas().get(i).getPosicion().get(0)+","+cacheAnterior.getListaConsult
as().get(i).getPosicion().get(1)+"]");
        }
```

```
        //parte en la cual un nodo entra a la nueva celda y pide que los usuarios existentes le compartan el caché
        for (int i = 0; i < conjuntoUsuarios.size(); i++) {
          if (i!=posCU) {
            xUserConj=(int)Math.floor((conjuntoUsuarios.get(i).getMyPos().get(0))/60.0);
            yUserConj=(int)Math.floor((conjuntoUsuarios.get(i).getMyPos().get(1))/61.11112);
            if (xActual==xUserConj&&yActual==yUserConj) {
              conjuntoUsuarios.get(posCU).setMyCache(conjuntoUsuarios.get(i).getMyCache());
              System.out.println("Prueba 1-1. ID celda en "+posCU+": "+conjuntoUsuarios.get(posCU).getMyCache().getID()+" y posee "+cacheAnterior.getListaConsultas().size()+" consultas");
                for (int j = 0; j < conjuntoUsuarios.get(posCU).getMyCache().getListaConsultas().size(); j++) {
                System.out.println("Prueba 1-2. Consultas en la celda: "+conjuntoUsuarios.get(posCU).getMyCache().getListaConsultas().get(j).getID());

              }
            break;
            }
            //si no hay ningún usuario en la celda, entonces pide el cache al usuario retenedor, preguntando a través de todos los usuarios por el retaining host de la celda
            if (i==conjuntoUsuarios.size()-1) {
              for (int j = 0; j < conjuntoUsuarios.size(); j++) {
                for (int k = 0; k < conjuntoUsuarios.get(j).getMyRetains().size(); k++) {
                  if (conjuntoUsuarios.get(j).getMyRetains().get(k).getID()==idCeldaActual) {
                    //si encuentra al usuario retenedor, este le entrega los datos de la celda al caché del usuario que consulta y luego borra los datos de su lista de retenedores

conjuntoUsuarios.get(posCU).setMyCache(conjuntoUsuarios.get(j).getMyRetains().get(k));
                    System.out.println("El usuario "+j+" entrega información de su retención al usuario "+posCU+" ya que este entró a la celda "+idCeldaActual+"["+xActual+"("+conjuntoUsuarios.get(posCU).getMyPos().get(0)+")"+","+yActual+"("+conjuntoUsuarios.get(posCU).getMyPos().get(1)+")"+"]");
                    System.out.println("El host retenedor tenía la siguiente información:");
                    System.out.println("N° retenciones antes: "+conjuntoUsuarios.get(j).getMyRetains().size());
                    System.out.println("Datos de la retención:");
                    System.out.println("ID Celda: "+conjuntoUsuarios.get(j).getMyRetains().get(k).getID());
                    System.out.println("Datos consultas:");
                    for (int l = 0; l < conjuntoUsuarios.get(j).getMyRetains().get(k).getListaConsultas().size(); l++) {
                      System.out.println("-> Consulta N° "+conjuntoUsuarios.get(j).getMyRetains().get(k).getListaConsultas().get(l).getID());
                    }
                    System.out.println("Fin Consultas");
                    System.out.println("----------------------------------------------------------------");
                    conjuntoUsuarios.get(j).getMyRetains().remove(k);
                    System.out.println("N° Retenciones después: "+conjuntoUsuarios.get(j).getMyRetains().size());
```

```
                    System.out.println("Datos del caché del usuario que tomó la retención:");
                    System.out.println("ID de celda:
"+conjuntoUsuarios.get(posCU).getMyCache().getID());
                    System.out.println("Datos consultas:");
                    for (int l = 0; l < conjuntoUsuarios.get(posCU).getMyCache().getListaConsultas().size(); l++) {
                      System.out.println("-> Consulta N°
"+conjuntoUsuarios.get(posCU).getMyCache().getListaConsultas().get(l).getID());
                    }
                    System.out.println("Fin Consultas");
                    freno=true;
                    break;
                  }
                }
                if(freno){
                  break;
                }
              }
            }
          }
        }
        for (int i = 0; i < conjuntoUsuarios.size(); i++) {
        if (i!=posCU) {
          existeAnterior=ML.SearchNode(conjuntoUsuarios.get(i), matriz, xAnterior, yAnterior);
        }
        if (existeAnterior) {
          candidateListAnterior.add(conjuntoUsuarios.get(i));
          existeAnterior=false;
        }
      }
        if (!matriz.get(xAnterior).get(yAnterior).getIDNodos().isEmpty()) {
          for (int i = 0; i < matriz.get(xAnterior).get(yAnterior).getIDNodos().size(); i++) {
            if
(conjuntoUsuarios.get(posCU).getMyId()==matriz.get(xAnterior).get(yAnterior).getIDNodos().get(
i)) {
              matriz.get(xAnterior).get(yAnterior).getIDNodos().remove(i); //Goodbye: Sale de la
lista de nodos de la celda anterior
              break;
            }
          }
        }
        matriz.get(xActual).get(yActual).getIDNodos().add(conjuntoUsuarios.get(posCU).getMyId());
//Hello: Entra en la lista de nodos de la celda actual
        if (!consultas.isEmpty()) {
          if
(candidateListAnterior.isEmpty()&&!conjuntoUsuarios.get(posCU).getMyCache().getListaConsulta
s().isEmpty()) {
            conjuntoUsuarios.get(posCU).getMyRetains().add(cacheAnterior);

conjuntoUsuarios.get(posCU).getMyCache().setID(matriz.get(xActual).get(yActual).getID());
```

```
            //conjuntoUsuarios.get(posCU).getMyCache().getListaConsultas().clear();
          }else{

conjuntoUsuarios.get(posCU).getMyCache().setID(matriz.get(xActual).get(yActual).getID());
            //conjuntoUsuarios.get(posCU).getMyCache().getListaConsultas().clear();
          }
          if (!conjuntoUsuarios.get(posCU).getMyRetains().isEmpty()) {
            int xMyRetain, yMyRetain, xMyCell, yMyCell, xNewCell, yNewCell, xPixel, yPixel,
posUsuario=-1, i=0;
            double distanciaAMyCell, distanciaANewCell, distancia, masCerca;

xMyCell=(int)Math.floor((conjuntoUsuarios.get(posCU).getMyPosAnterior().get(0)/60.0));

yMyCell=(int)Math.floor((conjuntoUsuarios.get(posCU).getMyPosAnterior().get(1)/61.11112));
            xNewCell=(int)Math.floor((conjuntoUsuarios.get(posCU).getMyPos().get(0)/60.0));
            yNewCell=(int)Math.floor((conjuntoUsuarios.get(posCU).getMyPos().get(1)/61.11112));
            while(i<conjuntoUsuarios.get(posCU).getMyRetains().size()){
              //System.out.println("valor de i: "+i+" y valor de getMyRetains:
"+conjuntoUsuarios.get(posCU).getMyRetains().size());
              xMyRetain=(conjuntoUsuarios.get(posCU).getMyRetains().get(i).getID()-1)/9;
              if (conjuntoUsuarios.get(posCU).getMyRetains().get(i).getID()%9==0) {
                yMyRetain=8;
              }else{
                yMyRetain=(conjuntoUsuarios.get(posCU).getMyRetains().get(i).getID()%9)-1;
              }

              distanciaAMyCell=Math.sqrt(Math.pow((xMyCell-xMyRetain),  2)+Math.pow((yMyCell-
yMyRetain), 2));
              distanciaANewCell=Math.sqrt(Math.pow((xNewCell-xMyRetain),
2)+Math.pow((yNewCell-yMyRetain), 2));
              xPixel=(int)Math.round(((xMyRetain)*60.0)+(60.0/2));
              yPixel=(int)Math.round(((yMyRetain)*61.11112)+(61.11112/2));
              if (distanciaAMyCell<distanciaANewCell) {
                masCerca=1000000;
                for (int j = 0; j < conjuntoUsuarios.size(); j++) {
                  distancia=Math.sqrt(Math.pow((xPixel-
conjuntoUsuarios.get(j).getMyPos().get(0)),2)+Math.pow((yPixel-
conjuntoUsuarios.get(j).getMyPos().get(1)),2));
                  if (distancia<masCerca) {
                    masCerca=distancia;
                    posUsuario=j;
                  }
                }
                if (posUsuario!=posCU) {

conjuntoUsuarios.get(posUsuario).getMyRetains().add(conjuntoUsuarios.get(posCU).getMyRetain
s().get(i));
                  conjuntoUsuarios.get(posCU).getMyRetains().remove(i);
                }else{
                  i++;
```

```
                }

              }else{
                i++;
              }
            }
          }
        }
      }
      return conjuntoUsuarios;
    }
}
```

- **Clase Usuario**

```
package Simulacion;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.Line2D;
import java.util.ArrayList;

public class Usuario{

    int columna;//guarda posición primera columna de la pelota
    int fila;//guarda posición primera fila primera pelota
    int direccionColumna = 1;//guarda el incremento o decremento del usuario
    int direccionFila = 1;//guarda el incremento o decremento de fila del usuario
    int diametro;//guarda el diámetro  elegido por el usuario
    int random;
    ArrayList<Integer> puntoX=new ArrayList<>();
    ArrayList<Integer> puntoY=new ArrayList<>();
    ArrayList<Integer> ids=new ArrayList<>();
    ArrayList<Integer> idConsultas=new ArrayList<>();

    Usuario(int diametro){

      this.diametro = diametro;
      this.columna = (int) (Math.random()*800);
      this.fila = (int) (Math.random()*500);



    }

    void calcularDireccionUsuario(){
      //crea número random del 0 al 3 para mover usuario
```

```
        random =(int)(Math.random()*4);
        if(random==0) fila=fila+20;
        if(random==1) fila=fila-20;
        if(random==2) columna=columna+20;
        if(random==3) columna=columna-20;
        /*
        para que no se salga de lo bordes del panel
        se aumentan o disminuyen la fila o columna para
        que se mantenga en el margen
        */
        //para que no se salga de la parte superior
        if(columna<0) columna=columna+20;
        //para que no se salga de la parte izquierda
        if(fila<0) fila=fila+20;
        //para que no se salga del lado derecho
        if(columna + diametro > Panel.anchura) columna=columna-20;
        //para que no se salga de la parte de abajo del panel
        if(fila + diametro > Panel.altura) fila=fila-20;

        //random walk

}
    void pintarConsulta(int id){
        ids.add(id);
        if (idConsultas.isEmpty()) {
            idConsultas.add(0);
        }else{
            idConsultas.add(idConsultas.size());
        }
    }


    ArrayList<Integer> pintarUsuario(Graphics g, int i, boolean usuarioSel, ArrayList<Nodo>
conjuntoUsuarios, ArrayList<Consulta> consultas){
        ArrayList<Integer> auxPos=new ArrayList<>();
        int x,y;
        calcularDireccionUsuario();
        auxPos.add(columna);
        auxPos.add(fila);
        //cada celular cambia de color
        if (usuarioSel) {
            g.drawImage(Panel.imagenUserSel, columna, fila,null);
        }else{
            g.drawImage(Panel.imagenUser, columna, fila,null);
        }

       String iterator=Integer.toString(i), idConsulta, idUsuario;//
        if (i<10) {//este i esta aumentado uno, por eso cuando cambie la imagen por otra color
            //debo poner i-1; pasarle al if del accionPerfomed uno personalizado. pasarle el I
            // para que sepa cual es
```

```
 iterator="0"+iterator;
}

g.setColor(Color.black);
g.setFont(new Font("Console", Font.PLAIN, 10));
g.drawString(iterator, columna+15, fila+20);
g.setColor(Color.blue);
if (!consultas.isEmpty()) {
   for (int j = 0; j < consultas.size(); j++) {
      x=consultas.get(j).getPosicion().get(0);
      y=consultas.get(j).getPosicion().get(1);
      idUsuario="U:"+Integer.toString(consultas.get(j).getIdCreador());
      idConsulta="C:"+Integer.toString(consultas.get(j).getID());

      if (x>40&&x<860&&y>40&&y<510) {
         g.drawString(idConsulta, x-10, y);
         g.drawString(idUsuario, x-10, y+10);
      }else{
         if (x<=40&&y>40&&y<510) {
            g.drawString(idConsulta, x+20, y);
            g.drawString(idUsuario, x+20, y+10);
         }
         if (x>=860&&y>40&&y<510) {
            g.drawString(idConsulta, x-25, y);
            g.drawString(idUsuario, x-25, y+10);
         }
         if (y<=40&&x>40&&x<860) {
            g.drawString(idConsulta, x-10, y+19);
            g.drawString(idUsuario, x-10, y+29);
         }
         if (y>=510&&x>40&&x<860) {
            g.drawString(idConsulta, x-10, y-29);
            g.drawString(idUsuario, x-10, y-19);
         }
         if (x<=40&&y<=40) {
            g.drawString(idConsulta, x+20, y+19);
            g.drawString(idUsuario, x+20, y+29);
         }
         if (x<=40&&y>=510) {
            g.drawString(idConsulta, x+20, y-29);
            g.drawString(idUsuario, x+20, y-19);
         }
         if (x>=860&&y<=40) {
            g.drawString(idConsulta, x-25, y+19);
            g.drawString(idUsuario, x-25, y+29);
         }
         if (x>=860&&y>=510) {
            g.drawString(idConsulta, x-25, y-29);
            g.drawString(idUsuario, x-25, y-19);
         }
```

```
            }
            g.drawRoundRect(x-30, y-30, 60, 60, 0, 0);
          }
        }

        /*if (!ids.isEmpty()) {


          for (int j = 0; j < ids.size(); j++) {
            for (int k = 0; k < conjuntoUsuarios.size(); k++) {
              if (conjuntoUsuarios.get(k).getMyId()==ids.get(j)) {
                if (puntoX.isEmpty()) {
                  puntoX.add(conjuntoUsuarios.get(k).getMyPos().get(0));
                  puntoY.add(conjuntoUsuarios.get(k).getMyPos().get(1));
                }else{
                  if (puntoX.size()-1<j) {
                    puntoX.add(conjuntoUsuarios.get(k).getMyPos().get(0));
                    puntoY.add(conjuntoUsuarios.get(k).getMyPos().get(1));
                  }
                }
                idUsuario="U: "+Integer.toString(ids.get(j));
                idConsulta="C: "+Integer.toString(idConsultas.get(j));
                g.drawString(idConsulta, puntoX.get(j), puntoY.get(j));
                g.drawString(idUsuario, puntoX.get(j), puntoY.get(j)+10);
                g.drawRoundRect(puntoX.get(j)-35, puntoY.get(j)-30, 60, 60, 0, 0);
              }
            }
          }
        }*/
        return auxPos;
      }
    }
```

- **Clase VentanaPrincipal**

```
package Simulacion;


public class VentanaPrincipal extends javax.swing.JFrame {

  public VentanaPrincipal() {
    initComponents();
    this.setLocation(300, 150);
    this.setVisible(true);
  }


  @SuppressWarnings("unchecked")
  // <editor-fold defaultstate="collapsed" desc="Generated Code">
```

```
    private void initComponents() {

        jButton1 = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setMinimumSize(new java.awt.Dimension(600, 400));
        setPreferredSize(new java.awt.Dimension(600, 400));
        getContentPane().setLayout(null);

        jButton1.setText("Generar Marco");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton1);
        jButton1.setBounds(300, 260, 160, 50);

        jLabel1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/Imagenes/ventana.jpg"))); // NOI18N
        getContentPane().add(jLabel1);
        jLabel1.setBounds(0, -50, 630, 420);

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

      this.dispose();//cierra la ventana
      MarcoCentral m = new MarcoCentral();
      //m.setVisible(true);

    }

    public static void main(String[] args) {
        new VentanaPrincipal();
    }
//generar marco Jbutton1 Jlabel imagen
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration
}
```

- **Clase Location**

```
package Simulacion;

import java.util.ArrayList;

public class location {
  public location(){

  }
  //recorrido nunca tiene que ser vacío, porque el primer valor con el que tiene que venir recorrido, es el valor del nodo inicial
  public ArrayList<Integer> mapeo(ArrayList<Integer> recorrido, Nodo nodoInicial, ArrayList<Nodo> conjuntoUsuarios, int nodoFinal, ArrayList<Nodo> candidatos){
    double dist, distMax=(Math.sqrt(Math.pow(60,2)+Math.pow(61.11112,2)))*3;
     ArrayList<Nodo> candAux=new ArrayList<>();
     System.out.println("La ID del nodo inicial es: "+nodoInicial.getMyId());
     System.out.println("La ID del nodo final es: "+nodoFinal);
     for (int i = 0; i < recorrido.size(); i++) {
       System.out.println("Recorrido "+i+": "+recorrido.get(i));
     }
       for (int i = 0; i < conjuntoUsuarios.size(); i++) {
         dist=Math.sqrt(Math.pow((nodoInicial.getMyPos().get(0)-conjuntoUsuarios.get(i).getMyPos().get(0)), 2)+Math.pow((nodoInicial.getMyPos().get(1)-conjuntoUsuarios.get(i).getMyPos().get(1)), 2));
          if (dist<=distMax) {
            if (nodoFinal==conjuntoUsuarios.get(i).getMyId()) {
              if (recorrido.get(recorrido.size()-1)!=conjuntoUsuarios.get(i).getMyId()) {
                recorrido.add(conjuntoUsuarios.get(i).getMyId());
              }
              return recorrido;
            }else{
              for (int j = 0; j < recorrido.size(); j++) {
                if (recorrido.get(j)==conjuntoUsuarios.get(i).getMyId()) {
                  break;
                }
                if (j==recorrido.size()-1) {
                  if (candidatos.isEmpty()) {
                    candidatos.add(conjuntoUsuarios.get(i));
                  }else{
                    for (int k = 0; k < candidatos.size(); k++) {
                      if (conjuntoUsuarios.get(i).getMyId()==candidatos.get(k).getMyId()) {
                        break;
                      }
                      if (k==candidatos.size()-1) {
                        candAux.add(conjuntoUsuarios.get(i));
                      }
                    }
                  }
```

```
                    }
                  }
                }
              }
            }
            if (!candAux.isEmpty()) {
              for (int i = 0; i < candAux.size(); i++) {
                recorrido.add(candAux.get(i).getMyId());
                if (i==0) {
                  for (int j = 0; j < candAux.size(); j++) {
                    candidatos.add(candAux.get(j));
                  }
                }
                recorrido=mapeo(recorrido, candAux.get(i), conjuntoUsuarios, nodoFinal, candidatos);
                if (recorrido.get(recorrido.size()-1)!=nodoFinal) {
                  System.out.println("Camino erróneo, remuevo el nodo");
                  recorrido.remove(recorrido.size()-1);
                }else{
                  break;
                }
              }
            }
          return recorrido;
        }

          //TTL siempre debe venir con 1 como primer valor
        public int radar(Nodo nodoInicial, ArrayList<Nodo> conjuntoUsuarios, Nodo nodoFinal, int TTL){
          int infTTL=TTL-1;
          double dist, distLim=Math.sqrt(Math.pow(60,2)+Math.pow(61.11112,2));
          for (int i = 0; i < conjuntoUsuarios.size(); i++) {
            if (conjuntoUsuarios.get(i).getMyId()!=nodoInicial.getMyId()) {
              dist=Math.sqrt(Math.pow((nodoInicial.getMyPos().get(0)-
conjuntoUsuarios.get(i).getMyPos().get(0)),          2)+Math.pow((nodoInicial.getMyPos().get(1)-
conjuntoUsuarios.get(i).getMyPos().get(1)), 2));
              if ((dist<=distLim*TTL)&&(dist>distLim*infTTL)) {
                if (nodoFinal.getMyId()==conjuntoUsuarios.get(i).getMyId()) {
                  return TTL;
                }
              }
            }
            if (i==conjuntoUsuarios.size()-1) {
              TTL=radar(nodoInicial, conjuntoUsuarios, nodoFinal, (TTL+1));
            }
          }
          return TTL;
        }
}
```

- **Clase Tabla**

```
package Simulacion;

import java.util.ArrayList;
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import static javax.swing.JTable.AUTO_RESIZE_OFF;
import javax.swing.table.DefaultTableModel;
public class tabla {
   DefaultTableModel model;
   JDialog tabla;
   JTable datos;
   Object [] objeto=new Object[5];
   String col[] = {"ID C","Creador","Usuarios dentro de consulta","Retaining Host", "Celda y vértices"};
   JScrollPane scrollTabla;
   public tabla(){
      tabla=new JDialog();
      tabla.setVisible(false);
      tabla.setBounds(0, 0, 700, 300);
      tabla.setDefaultCloseOperation(JDialog.HIDE_ON_CLOSE);
      tabla.setTitle("Tabla de Consultas");
      model  = new DefaultTableModel(col, 0);
      datos=new JTable(model);
      datos.setAutoResizeMode(AUTO_RESIZE_OFF);
      datos.getColumn(col[0]).setPreferredWidth(35);
      datos.getColumn(col[1]).setPreferredWidth(80);
      datos.getColumn(col[2]).setPreferredWidth(200);
      datos.getColumn(col[3]).setPreferredWidth(120);
      datos.getColumn(col[4]).setPreferredWidth(240);
      scrollTabla = new JScrollPane();

scrollTabla.setVerticalScrollBarPolicy(javax.swing.JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
      scrollTabla.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
      scrollTabla.setViewportView(datos);
      scrollTabla.setBounds(20, 350, 440, 654);
      tabla.add(scrollTabla);
   }
   public void ingresarDatos(ArrayList<Consulta> consultas, ArrayList<Nodo> conjuntoUsuarios, ArrayList<ArrayList<Grilla>> matriz){
      int[] iInf=new int[2], iSup=new int[2], dInf=new int[2], dSup=new int[2];
      int limiteC;
      model.setRowCount(0);
      for (int i = 0; i < consultas.size(); i++) {

         limiteC=consultas.get(i).getLimite();
```

```
        objeto[0]=consultas.get(i).getID();
        objeto[1]="Nodo "+(consultas.get(i).getIdCreador());
        objeto[2]="Nodos";

        for (int j = 0; j < conjuntoUsuarios.size(); j++) {
          if (consultas.get(i).getIdCreador()==conjuntoUsuarios.get(j).getMyId()) {
            for (int k = 0; k < conjuntoUsuarios.get(j).getConsulCreadas().size(); k++) {
              if
(conjuntoUsuarios.get(j).getConsulCreadas().get(k).getIdConsulta()==consultas.get(i).getID()) {
                for (int l = 0; l < conjuntoUsuarios.get(j).getConsulCreadas().get(k).getNodos().size();
l++) {
                  objeto[2]=objeto[2]+"
"+conjuntoUsuarios.get(j).getConsulCreadas().get(k).getNodos().get(l).getMyId();
                }
              }
            }
          }
        }
        if (objeto[2]=="Nodos") {
          objeto[2]="No hay usuarios";
        }
        for (int j = 0; j < conjuntoUsuarios.size(); j++) {
          for (int k = 0; k < conjuntoUsuarios.get(j).getMyRetains().size(); k++) {
            if (conjuntoUsuarios.get(j).getMyRetains().get(k).getID()==consultas.get(i).getID()) {
              objeto[3]="Usuario "+conjuntoUsuarios.get(j).getMyId();
            }
          }
        }
        int x=(int)Math.floor(consultas.get(i).getPosicion().get(0)/60.0);
        int y=(int)Math.floor(consultas.get(i).getPosicion().get(1)/61.11112);
        iInf[0]=cargarVertices(consultas.get(i), -limiteC, 0);
        iInf[1]=cargarVertices(consultas.get(i), -limiteC, 1);
        iSup[0]=cargarVertices(consultas.get(i), -limiteC, 0);
        iSup[1]=cargarVertices(consultas.get(i), limiteC, 1);
        dInf[0]=cargarVertices(consultas.get(i), limiteC, 0);
        dInf[1]=cargarVertices(consultas.get(i), -limiteC, 1);
        dSup[0]=cargarVertices(consultas.get(i), limiteC, 0);
        dSup[1]=cargarVertices(consultas.get(i), limiteC, 1);
        objeto[4]="ID              "+matriz.get(x).get(y).getID()+"              ("+x+","+y+")              Vid:
{"+matriz.get(iInf[0]).get(iInf[1]).getID()+","+matriz.get(iSup[0]).get(iSup[1]).getID()+","+matriz.g
et(dInf[0]).get(dInf[1]).getID()+","+matriz.get(dSup[0]).get(dSup[1]).getID()+"}" ;
        model.addRow(objeto);
        objeto[0]=null;
        objeto[1]=null;
        objeto[2]=null;
        objeto[3]=null;
        objeto[4]=null;
      }

      datos.repaint();
```

```
    tabla.setVisible(true);
  }

  public int cargarVertices(Consulta nuevaConsulta, int limite, int xy){
    int vertice;
    if (xy==0) {
      vertice=(int)Math.floor((nuevaConsulta.getPosicion().get(0)+limite)/60);
      if (vertice<0||vertice>14) {
        vertice=(int)Math.floor((nuevaConsulta.getPosicion().get(0))/60);
      }
    }else{
      vertice=(int)Math.floor((nuevaConsulta.getPosicion().get(1)+limite)/61.11112);
      if (vertice<0||vertice>8) {
        vertice=(int)Math.floor((nuevaConsulta.getPosicion().get(1))/61.11112);
      }
    }
    return vertice;
  }

}
```

- **Clase TablaCelda**

```
package Simulacion;

import java.util.ArrayList;
import javax.swing.JDialog;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import static javax.swing.JTable.AUTO_RESIZE_OFF;
import javax.swing.table.DefaultTableModel;

public class tablaCelda {
  DefaultTableModel model;
  JDialog tabla;
  JTable datos;
  Object [] objeto=new Object[20];
  String col[] = {"Celda","Consulta {Usuarios}","H Retenedor","H Candidatos","Celda","Consulta {Usuarios}","H Retenedor [Pos]","H Candidatos","Celda","Consulta {Usuarios}","H Retenedor","H Candidatos","Celda","Consulta {Usuarios}","H Retenedor","H Candidatos","Celda","Consulta {Usuarios}","H Retenedor","H Candidatos"};
  JScrollPane scrollTabla;

  public tablaCelda(){
    tabla=new JDialog();
    tabla.setVisible(false);
    tabla.setBounds(60, 0, 1200, 400);
    tabla.setDefaultCloseOperation(JDialog.HIDE_ON_CLOSE);
    tabla.setTitle("Tabla de Celdas");
    model  = new DefaultTableModel(col, 0);
```

```
    datos=new JTable(model);
    datos.setAutoResizeMode(AUTO_RESIZE_OFF);
    for (int i = 0; i < 15; i++) {
      if (i==0||i%3==0) {
        datos.getColumn(col[i]).setPreferredWidth(40);
      }
      if ((i%3)-1==0) {
        datos.getColumn(col[i]).setPreferredWidth(140);
      }
      if ((i%3)-2==0) {
        datos.getColumn(col[i]).setPreferredWidth(200);
      }
    }
    scrollTabla = new JScrollPane();

scrollTabla.setVerticalScrollBarPolicy(javax.swing.JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDE
D);
    scrollTabla.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
    scrollTabla.setViewportView(datos);
    scrollTabla.setBounds(750, 350, 440, 654);
    tabla.add(scrollTabla);
  }

  public int cargarVertices(Consulta nuevaConsulta, int limite, int xy){
    int vertice;
    if (xy==0) {
      vertice=(int)Math.floor((nuevaConsulta.getPosicion().get(0)+limite)/60);
      if (vertice<0||vertice>14) {
        vertice=(int)Math.floor((nuevaConsulta.getPosicion().get(0))/60);
      }
    }else{
      vertice=(int)Math.floor((nuevaConsulta.getPosicion().get(1)+limite)/61.11112);
      if (vertice<0||vertice>8) {
        vertice=(int)Math.floor((nuevaConsulta.getPosicion().get(1))/61.11112);
      }
    }
    return vertice;
  }

  public int retornarID(int [] coordenadas){
    int resultado;

    resultado=(coordenadas[0]*9)+coordenadas[1]+1;
    return resultado;
  }

  public void procesarDatos(ArrayList<ArrayList<Grilla>> matriz, ArrayList<Nodo>
conjuntoUsuarios, int partida, int celda, ArrayList<Consulta> consultas){
    int xCelda=-1, yCelda=-1, idMatriz=-1, xUser, yUser, limiteC, idVInf1, idVInf2, idVSup1, idVSup2,
idCeldaCon, superiorX, superiorY, inferiorX, inferiorY;
```

```
    int [] vConsulta= new int[2], vSup1= new int[2], vSup2= new int[2], vInf1= new int[2], vInf2=
new int[2], conCon=new int[5];
    boolean paraFor, noHay=true;
    for (int i = 0; i < 5; i++) {
      conCon[i]=0;
    }
        paraFor=false;
        for (int k = 0; k < matriz.size(); k++) {
          for (int l = 0; l < matriz.get(k).size(); l++) {
            if (celda==matriz.get(k).get(l).getID()) {
              xCelda=(matriz.get(k).get(l).getID()-1)/9;
              if (matriz.get(k).get(l).getID()%9==0) {
                yCelda=8;
              }else{
                yCelda=(matriz.get(k).get(l).getID()%9)-1;
              }
              idMatriz=celda;
              paraFor=true;
              break;
            }
          }
          if (paraFor) {
            paraFor=false;
            break;
          }
          if (k==matriz.size()-1) {
            System.out.println("no se encontró la id de la matriz buscada");
          }
        }

        objeto[partida]=idMatriz+"("+xCelda+","+yCelda+")";
        objeto[partida+1]="";
        if (!consultas.isEmpty()) {
          limiteC=consultas.get(0).getLimite();
          for (int i = 0; i < consultas.size(); i++) {
            superiorX=consultas.get(i).getPosicion().get(0)+consultas.get(i).getLimite();
            inferiorX=consultas.get(i).getPosicion().get(0)-consultas.get(i).getLimite();
            superiorY=consultas.get(i).getPosicion().get(1)+consultas.get(i).getLimite();
            inferiorY=consultas.get(i).getPosicion().get(1)-consultas.get(i).getLimite();
            vConsulta[0]=cargarVertices(consultas.get(i), 0, 0);
            vConsulta[1]=cargarVertices(consultas.get(i), 0, 1);
            vInf1[0]=cargarVertices(consultas.get(i), -limiteC, 0);
            vInf1[1]=cargarVertices(consultas.get(i), limiteC, 1);
            vSup1[0]=cargarVertices(consultas.get(i), -limiteC, 0);
            vSup1[1]=cargarVertices(consultas.get(i), -limiteC, 1);
            vInf2[0]=cargarVertices(consultas.get(i), limiteC, 0);
            vInf2[1]=cargarVertices(consultas.get(i), limiteC, 1);
            vSup2[0]=cargarVertices(consultas.get(i), limiteC, 0);
            vSup2[1]=cargarVertices(consultas.get(i), -limiteC, 1);
            idCeldaCon=retornarID(vConsulta);
```

```
            idVInf1=retornarID(vInf1);
            idVInf2=retornarID(vInf2);
            idVSup1=retornarID(vSup1);
            idVSup2=retornarID(vSup2);
            if (idCeldaCon==idMatriz) {
              if (conCon[0]==0) {
                objeto[partida+1]="Con(C): "+consultas.get(i).getID()+"=>{";
              }else{
                objeto[partida+1]=objeto[partida+1]+", Con(C): "+consultas.get(i).getID()+"=>{";
              }
              conCon[0]++;
            }
            if (idVInf1!=idCeldaCon && idVInf1==idMatriz) {
              if (conCon[1]==0) {
                objeto[partida+1]="Con(vII): "+consultas.get(i).getID()+"=>{";
              }else{
                objeto[partida+1]=objeto[partida+1]+", Con(vII): "+consultas.get(i).getID()+"=>{";
              }
              conCon[1]++;
            }
            if (idVInf2!=idCeldaCon && idVInf2!=idVInf1 && idVInf2==idMatriz) {
              if (conCon[2]==0) {
                objeto[partida+1]="Con(vID): "+consultas.get(i).getID()+"=>{";
              }else{
                objeto[partida+1]=objeto[partida+1]+", Con(vID): "+consultas.get(i).getID()+"=>{";
              }
              conCon[2]++;
            }
            if (idVSup1!=idCeldaCon && idVSup1!= idVInf2 && idVSup1!=idVInf1 &&
idVSup1==idMatriz) {
              if (conCon[3]==0) {
                objeto[partida+1]="Con(vSI): "+consultas.get(i).getID()+"=>{";
              }else{
                objeto[partida+1]=objeto[partida+1]+", Con(vSI): "+consultas.get(i).getID()+"=>{";
              }
              conCon[3]++;
            }
            if (idVSup2!=idCeldaCon && idVSup2!=idVSup1 && idVSup2!= idVInf2 &&
idVSup2!=idVInf1 && idVSup2==idMatriz) {
              if (conCon[4]==0) {
                objeto[partida+1]="Con(vSD): "+consultas.get(i).getID()+"=>{";
              }else{
                objeto[partida+1]=objeto[partida+1]+", Con(vSD): "+consultas.get(i).getID()+"=>{";
              }
              conCon[4]++;
            }
            for (int j = 0; j < conjuntoUsuarios.size(); j++) {
              if
((conjuntoUsuarios.get(j).getMyPos().get(0)<(superiorX))&&(conjuntoUsuarios.get(j).getMyPos().
```

```
get(0)>=(inferiorX))&&(conjuntoUsuarios.get(j).getMyPos().get(1)<(superiorY))&&(conjuntoUsua
rios.get(j).getMyPos().get(1)>=(inferiorY))){
                    objeto[partida+1]=objeto[partida+1]+" "+j;
                  }
                }
              }//fin del for de consultas}
              objeto[partida+1]=objeto[partida+1]+"}";
              if  (conCon[0]==0&&conCon[1]==0&&conCon[2]==0&&conCon[3]==0&&conCon[4]==0)
{
                objeto[partida+1]="Sin consultas en celda";
              }
            }//fin del if consulta != empty();
            else{
              objeto[partida+1]="Sin consultas creadas";
            }
            objeto[partida+2]="User";
            objeto[partida+3]="Usuario(s):";
            System.out.println("----------------------------------------------------------------");
            for (int k = 0; k < conjuntoUsuarios.size(); k++) {
              for (int l = 0; l < conjuntoUsuarios.get(k).getMyRetains().size(); l++) {
                if (conjuntoUsuarios.get(k).getMyRetains().get(l).getID()==idMatriz) {
                  objeto[partida+2]=objeto[partida+2]+" "+(conjuntoUsuarios.get(k).getMyId());
                  System.out.println("tamaño                                                    retaining:
"+conjuntoUsuarios.get(k).getMyRetains().size());
                  System.out.println("ID matriz: "+idMatriz);
                  System.out.println("ID usuario: "+conjuntoUsuarios.get(k).getMyId());
                  paraFor=true;
                  break;
                }
              }
              if (paraFor) {
                break;
              }
            }
            System.out.println("----------------------------------------------------------------");
            for (int i = 0; i < conjuntoUsuarios.size(); i++) {
              xUser=(int)Math.floor(conjuntoUsuarios.get(i).getMyPos().get(0)/60);
              yUser=(int)Math.floor(conjuntoUsuarios.get(i).getMyPos().get(0)/61.11112);
              if (xCelda==xUser && yCelda==yUser) {
                objeto[partida+3]=objeto[partida+3]+"
"+conjuntoUsuarios.get(i).getMyId()+"["+xUser+","+yUser+"]";
                noHay=false;
              }
            }
            if (!paraFor) {
              objeto[partida+2]="no Retainer";
            }
            if(noHay){
              objeto[partida+3]="C vacía";
            }
```

```
  }

  public void mostrarDatos(ArrayList<ArrayList<Grilla>> matriz, ArrayList<Nodo>
conjuntoUsuarios, ArrayList<Consulta> consultas){
    model.setRowCount(0);
    int celda=0;
    if (!consultas.isEmpty()) {
      System.out.println("Cantidad de consultas: "+consultas.size());
    }else{
      System.out.println("No tiene consultas creadas");
    }
    for (int i = 0; i < 27; i++) {
        celda++;
        procesarDatos(matriz, conjuntoUsuarios, 0, celda, consultas);
        procesarDatos(matriz, conjuntoUsuarios, 4, celda+(27*1), consultas);
        procesarDatos(matriz, conjuntoUsuarios, 8, celda+(27*2), consultas);
        procesarDatos(matriz, conjuntoUsuarios, 12, celda+(27*3), consultas);
        procesarDatos(matriz, conjuntoUsuarios, 16, celda+(27*4), consultas);
        model.addRow(objeto);
        objeto=new Object[20];
    }
    datos.repaint();
    tabla.setVisible(true);
  }

}
```

- **Consulta wrapArray**

```
package Simulacion;

import java.util.ArrayList;

public class wrapArray {
  ArrayList<Consulta> consultas;
  ArrayList<Nodo> conjuntoUsuarios;

  public wrapArray(){
    this.consultas=new ArrayList<>();
    this.conjuntoUsuarios= new ArrayList<>();
  }

  public wrapArray( ArrayList<Consulta> newConsultas, ArrayList<Nodo>
newConjuntoUsuarios){
    this.consultas=newConsultas;
    this.conjuntoUsuarios= newConjuntoUsuarios;
  }
```

```
  public void setConsultas(ArrayList<Consulta> newConsultas){
    this.consultas=newConsultas;
  }

  public void setConjuntoUsuarios(ArrayList<Nodo> newConjuntoUsuarios){
    this.conjuntoUsuarios= newConjuntoUsuarios;
  }

  public ArrayList<Consulta> getConsultas(){

    return consultas;
  }

  public ArrayList<Nodo> getConjuntoUsuarios(){

    return conjuntoUsuarios;
  }
}
```