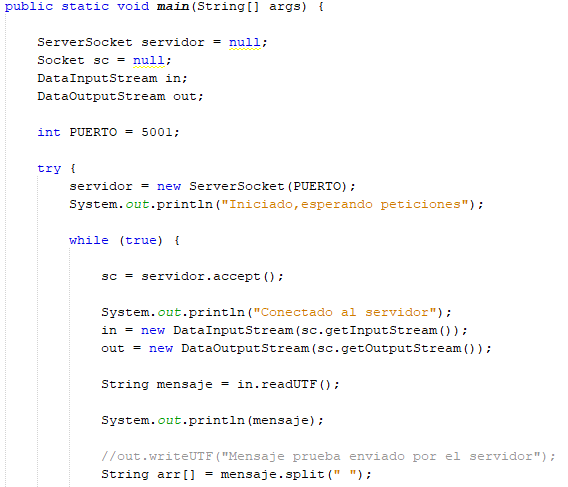
La aplicación de escritorio que hemos desarrollado en Java está compuesta por 2 partes fundamentales:

La parte del modelo cliente-servidor que viene a ejecutarse constantemente en la nube en el servidor de Amazon, que está esperando peticiones de la parte del cliente.

La otra parte es la aplicación propiamente dicha que se comunica con el servidor para pedir información.

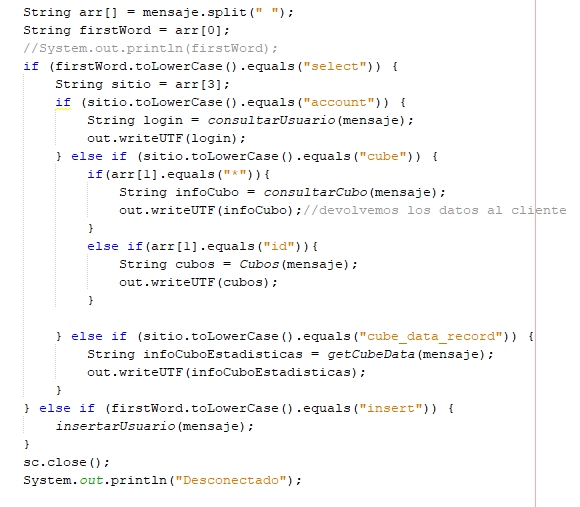
El servidor está en bucle comunicándose por el puerto 5001



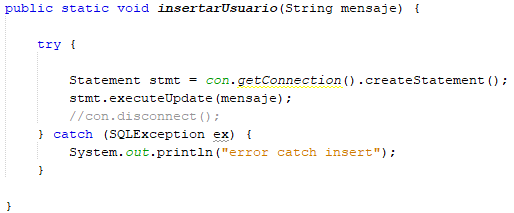
Y al recibir un mensaje despedaza en las palabras que lo componen que estaban separadas por “ ”(espacio). Los mensajes sql son generados en la clase Cliente (que hace de intermediario) dentro de las funciones respectivas a cada necesidad. La primera consulta que se hace está relacionada a la cuenta de usuario para poder acceder. Ya sea un registro nuevo o un login de un usuario ya registrado el mensaje va a contener “account” (nombre de la tabla de la Base de datos). La unica diferencia es que al hacer login debe hacer un “select” pero hay más funciones diferentes que hacen el select y por ello el servidor hace la distinción de antemano gracias a la primera posición del array.

En el caso de select, este puede ser empleado también para los cubos y los registros almacenados entonces obtenemos el elemento al que se refiere de la posición 4 del array.

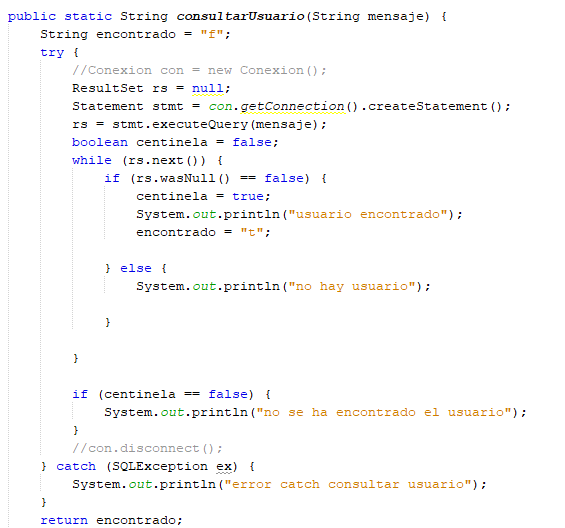
Dependiendo del caso que sea se ejecuta una función u otra.



Al tener que insertar usuario se abre conexión y se hace “executeUpdate” con la sentencia sql del mensaje. Es el metodo que se usa al tener que hacer insert,update o delete.



Si el usuario ya está registrado e intentara acceder con su usuario y contraseña :



Donde es buscado en la BD con “execute query”(el estándar para consultas) y arroja un print diciendo que no existe el usuario en caso de no encontrarlo. Esta función lleva return porque devolvemos la respuesta, tanto positiva como negativa, a través del cliente para que la aplicación dé una ventana emergente con un error en caso de no poder hacer login.

Rs es el puntero result set a la BD después de ejecutar la sentencia y vamos mirando todas las filas para ver dónde se encuentra. Si llega hasta la última sin encontrar al usuario el resultado del login resultará en un false.

Para consultar datos sobre los cubos de la BD tenemos las siguientes funciones tanto para estadísticas como para información.

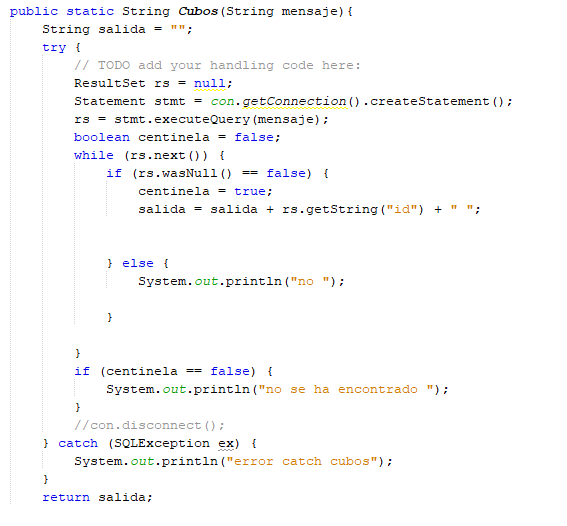
La primera de ellas es necesaria para sacar los datos sobre el cubo. Es usada para Información. Rs.getString() se emplea para ver las columnas de la fila apuntada y devolver sus valores. Todos los valores de devuelven como un único mensaje dividido por comas que se va a emplear para mostrarlos en un ventana.



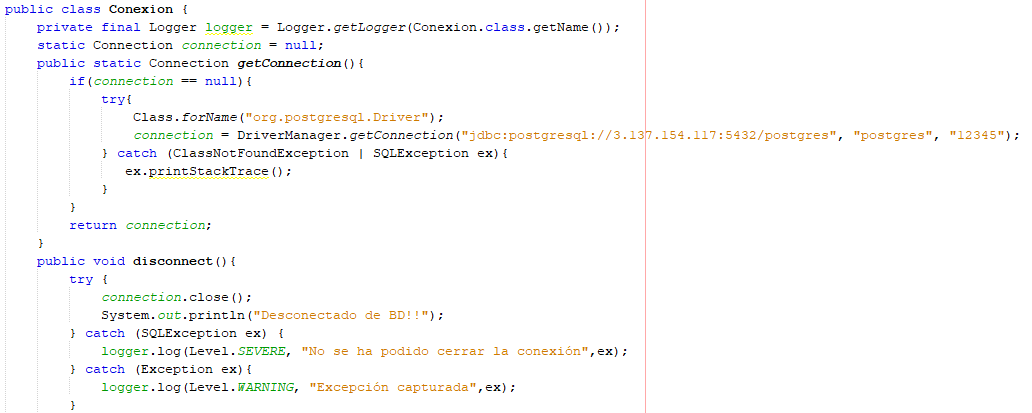
La segunda es empleada para crear la tabla según las tablas de los históricos de la BD de cubos.



Y esta se utiliza para ver todos los cubos actuales para poder trabajar sobre ellos.

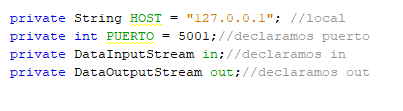


La clase conexión se conecta a nuestra BD por medio del driver de postgresql.

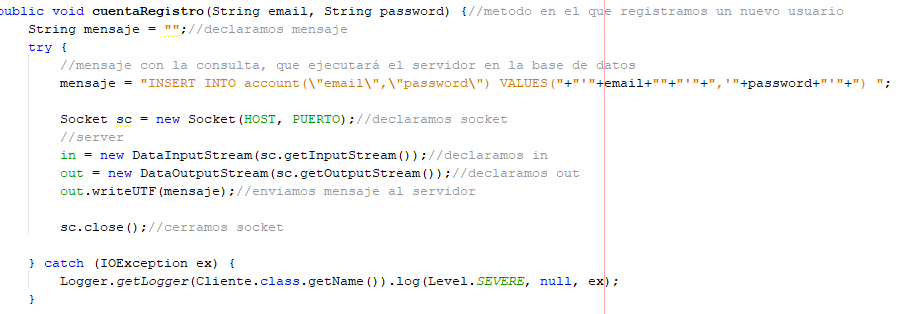


El programa es colocado en el servidor de Amazon.

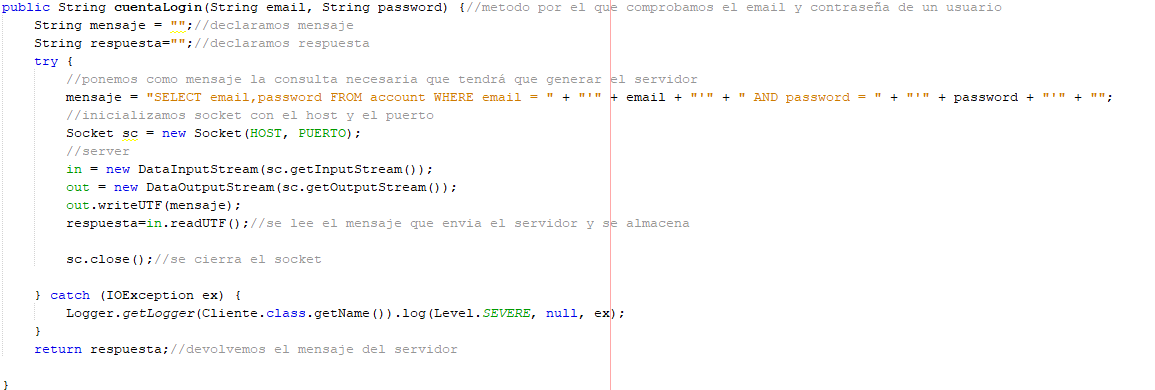
El intermediario (Clase Cliente):



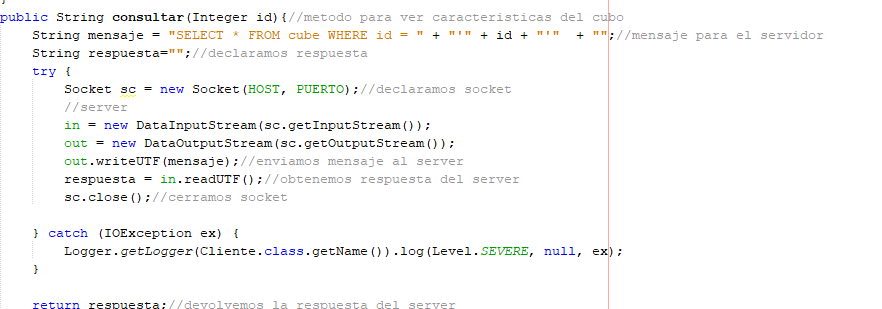
Cuando en la interfaz se pulsa Registro se llama a este método para hacer la conexión con el servidor remoto. Se lleva el email y la contraseña y envía la sentencia.



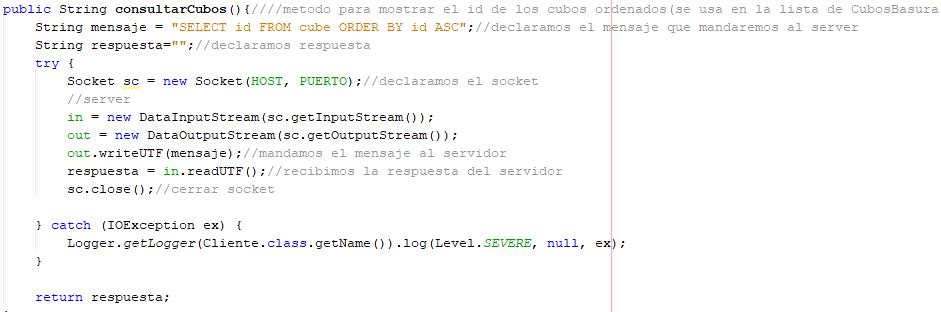
Al desear iniciar sesión se pide acceso con el email y la contraseña proporcionados



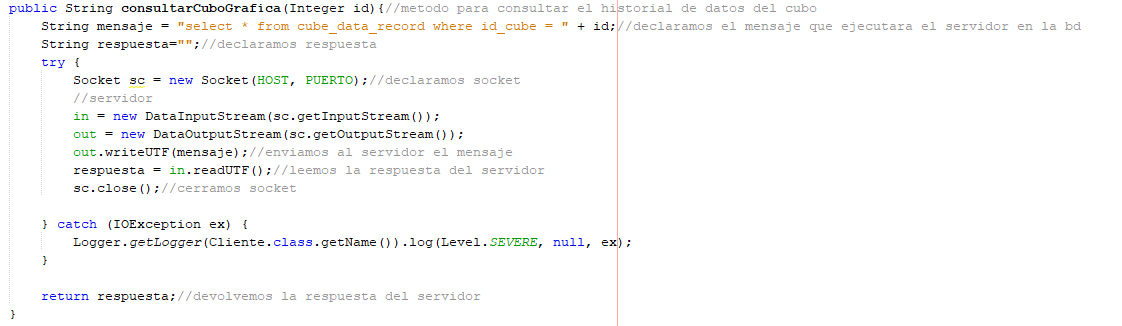
Para sacar información sobre 1 cubo determinado seleccionado el Cliente pide información sobre una id de cubo determinada:



La interfaz que rellena la lista con ids de cubos ordenadamente llama a:



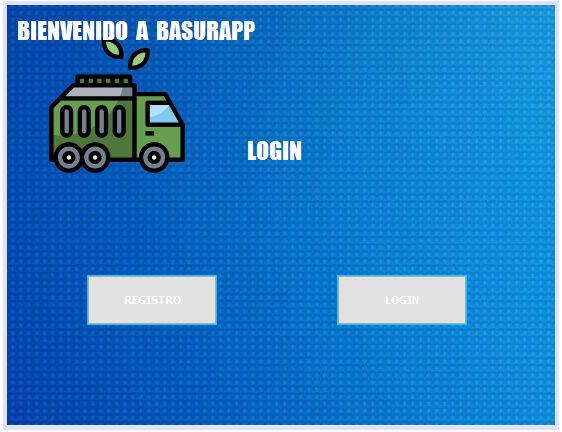
Al intentar graficar y hacer la tabla de estadísticas sobre los registros de un cubo determinado:



Para poder dibujar los cubos en el mapa el servidor devuelve SOLO los cubos activos (sin el flag deleted).

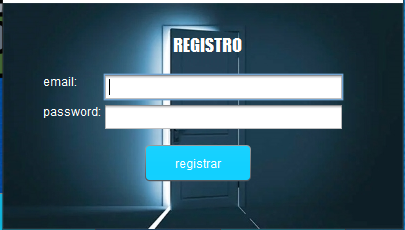


La parte de la interfaz abre una ventana:

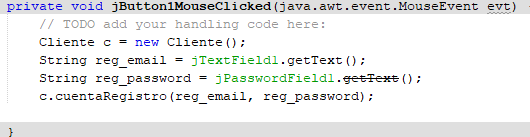


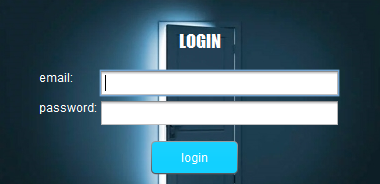
Que solicita al usuario Registrarse si es la primera vez que va a acceder o hacer login si ya dispone de una cuenta.

Ambas opciones crean y abren una nueva ventana



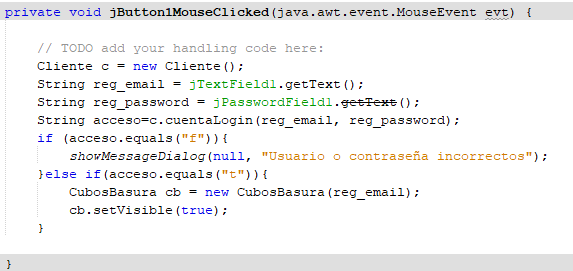
Al introducir los datos y hacer click en el botón se abre una conexión para realizar el registro.





Al introducir los datos y hacer click en el botón se abre una conexión para realizar el login.

La respuesta de posibilidad o la imposibilidad de hacer el login viene dada por el servidor a traves de cliente.



Si se realiza login correctamente se abre una nueva ventana con la que se pueden realizar diferentes acciones sobre los cubos de basura.



La lista de la izquierda se llena de cubos registrados en la BD ordenados por ID. Los cubos vienen por un mensaje separados por un “ ”(espacio). Desde el servidor.

Al pulsar “X”(salir) se pedirá confirmación de acción para no cerrar por equivocación.

Se debe escoger un cubo sobre el que trabajar de la lista de la izquierda.

Al pulsar el botón de Información se abre una ventana en la que podemos ver diferentes datos sobre el cubo:

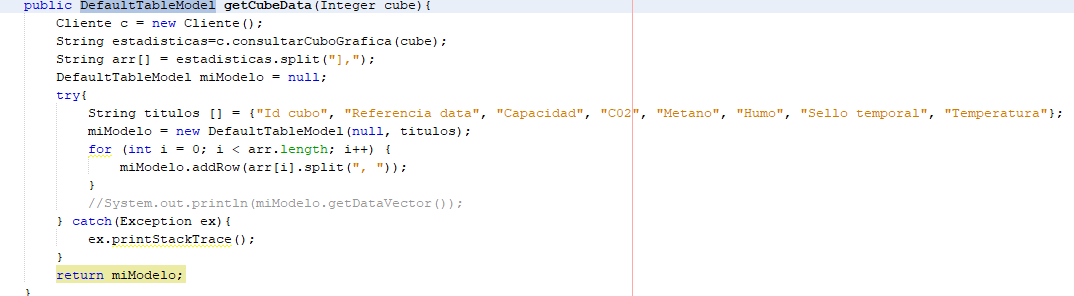
Su id, la capacidad, si sigue activo o fue eliminado hace poco (a través del atributo “deleted”), su localización y el id con el que estada registrada la entrada en cube\_data\_record.

Al pulsar sobre Estadísticas se abre una ventana que tendrá una lista desplegable para ver la gráfica en función de lo seleccionado: capacidad, CO2, metano, humo. Al seleccionar una y pulsar Graficar se representarán los datos registrados en diferentes momentos y un gráfico de barras 3D en función del atributo seleccionado, el número y el sello temporal. Para ello se emplea JFreeChart y ChartPanel de la librería JFree.

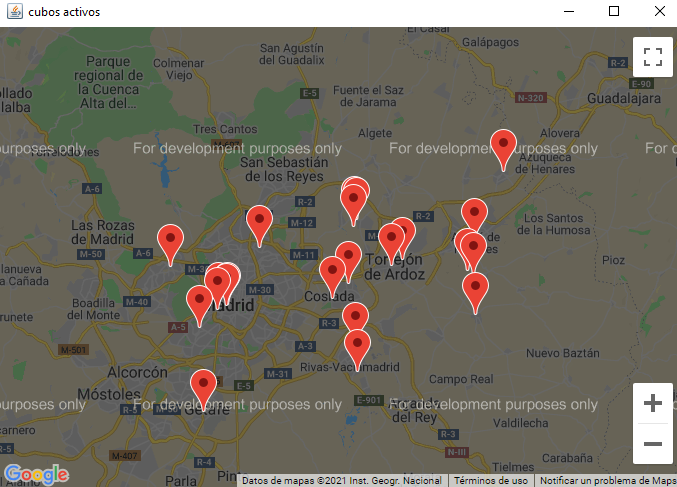


La grafica se coloca en un determinado número de pixeles para poder caber al mismo tiempo que la tabla.

La clase Estadísticas necesita de Graficar que devuelve un DefaultTableModel para la tabla, gracias a una petición sql por la que recibe todos los datos registrados, separados por corchetes debido a que en el servidor los sacamos como arrays.



Al pulsar sobre “mapa” se despliega un mapa con los cubos que están activos actualmente, teniendo en cuenta sus coordenadas para poder marcarlos alrededor del centro del mapa.



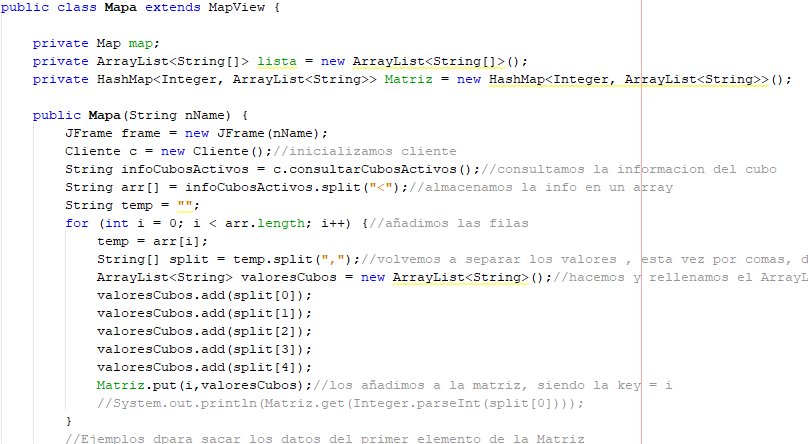
Los backgrounds de las ventanas se hicieron con unas imágenes, incluidas en el paquete, a través de una técnica de agregado de icono a un JLabel.

Para ello se ha creado la clase Mapa que crea un JFrame que más adelante se describirá como es empleado para poner el mapa.

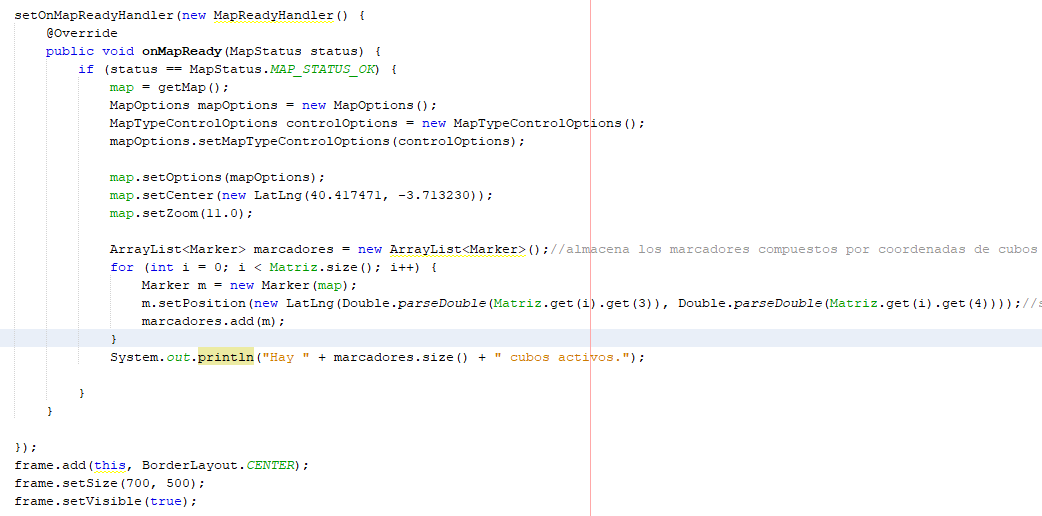
Se hace petición para obtener los cubos que están activos actualmente y todos sus datos aunque de momento solo se van a necesitar algunos.

Las filas de la BD vienen separadas por el carácter “<” y por ello tenemos que hacer un split para dividirlas y manejarlas mejor.

Estos valores se meten a un ArrayList para poderlos tener todos y ordenados para cada cubo separados otra vez esta vez por “,” para obtener los 5 datos del cubo. Estos datos se meten en la matriz.



Para el mapa se emplea la librería GSON que no nos obliga a usar una API de pago como es obligatorio en estas fechas. Aun así, Google ya tampoco le permite la funcionalidad completa dejando el mapa muy oscuro y a falta de algunas funcionalidades.



Se inicializa el mapa y se establece su centro:

40.417471, -3.713230 (Palacio Real de Madrid)

Para poder abrir el mapa automáticamente apuntando a ese sitio.

Se establece un zoom inicial de 11 que puede ser modificado con “+””-” en el mapa.

Una vez dibujado el mapa GPS creamos un “for” para que haga las veces necesarias (número de cubos activos) marcadores con las coordenadas de los cubos activos traídos. Al mapa se le dan unos tamaños predefinidos.