GPS de Montaña

Adrià Gómez-Acosta

Resumen. La popularidad y la utilización de los GPS ha crecido radicalmente en los últimos años en muchos ámbitos de la vida humana. Uno de ellos es la realización de rutas por la montaña para poder llevar a cabo actividades como rutas en bicicleta, salir a correr o caminatas por la montaña. Para satisfacer esa necesidad, han surgido varias aplicaciones que permiten el manejo de estas rutas pudiendo visualizarlas y compartirlas para poder realizar las actividades deseadas. En este proyecto, se desarrolla una aplicación de escritorio multiplataforma capaz de unificar varias rutas cargadas por el usuario en un único grafo sin nodos ni aristas repetidas ni redundantes, para posteriormente, con un algoritmo de búsqueda, poder generar rutas entre dos puntos seleccionados, con restricciones de distancia y desnivel acumulado y con la posibilidad de añadir paradas intermedias en ella, con la mejor relación entre velocidad de respuesta y precisión de los resultados posible. Además, es posible el guardado de esta ruta para poder introducirla en GPSs de montaña y poder realizarla sin problema, teniendo así un generador de rutas el cual ofrece una infinidad de ellas gracias a todos sus parámetros seleccionables.

Palabras clave. GPS, Senderismo, Rutas, Aplicación de escritorio, Grafo, Algoritmo de búsqueda, Backtracking, Greedy, Dijkstra, Heurística, Interfaz gráfica.

Abstract. The popularity and use of GPS has grown dramatically in recent years in many areas of human life. One of them is the realisation of mountain routes to be able to carry out activities such as cycling routes, jogging or hiking in the mountains. In order to satisfy this need, several applications have emerged that allow the management of these routes, visualising and sharing them in order to be able to carry out the desired activities. In this project, a multiplatform desktop application is developed that is capable of unifying several routes loaded by the user in a single graph without repeated or redundant nodes or edges, in order to subsequently, with a search algorithm, be able to generate routes between two selected points, with restrictions on distance and accumulated slope and with the possibility of adding intermediate stops in it, with the best possible relationship between speed of response and accuracy of the results. In addition, it is possible to save this route to be able to introduce it in mountain GPS and to be able to carry it out without any problem, thus having a route generator which offers an infinity of them thanks to all its selectable parameters.

Index Terms. GPS, Hiking, Routing, Desktop application, Graph, Search algorithm, Backtracking, Greedy, Dijkstra, Heuristics, Graphical interface.

1 Introducción - Contexto del trabajo

En los últimos años la importancia y popularidad de los GPS ha crecido innegablemente, hasta dejar obsoletos y en desuso los mapas en papel.

En el senderismo, excursiones por montaña o salidas en bicicleta, esto también queda reflejado con el diseño y creación de aplicaciones capaces de crear y visualizar rutas por montaña. Una de las más populares es Wikiloc[1], aplicación capaz de ver rutas, diseñarlas y seguirlas tanto para móvil como para ordenador, que también cuenta con la posibilidad de subir rutas propias y descargar rutas de otros usuarios. Strava[2], es otra de las aplicaciones más populares entre corredores y ciclistas, también capaz de ver mapas, preparar rutas y descargar tracks, que, además ofrece la posibilidad de grabar rutas.

En este proyecto se pretende desarrollar una aplicación similar a Wikiloc y Strava, pero con la diferencia de que esta sea capaz de unificar rutas y generar nuevas rutas para poder planificar escapadas por la montaña.

Mas específicamente, el funcionamiento de la aplicación consiste en unificar en un único grafo todas las rutas seleccionadas y subidas por el usuario a la aplicación y mostrar este grafo creado sobre la zona del mapa correspondiente que el usuario cargue. Sobre este grafo, el usuario tiene la posibilidad de escoger un punto de partida de la ruta y punto final de esta, unos criterios para la creación de la ruta como puede ser generar aquella ruta con menos desnivel acumulado o con una longitud concreta y, además seleccionar puntos intermedios en la ruta. Con estos parámetros, la aplicación será capaz de generar la ruta que mejor se adapte a las necesidades del usuario y se mostrará sobre el mapa, con la posibilidad de poder guardarla en un nuevo fichero o incluso guardar el grafo completo.

2 Estado del arte

2.1 Strava

Strava es una red social enfocada a deportistas como pueden ser ciclistas y corredores y una aplicación de seguimiento GPS deportiva.

Los miembros de la red pueden registrar la actividad deportiva y, además es posible obtener mediante la aplicación móvil o en el sitio web, un análisis detallado de los datos como pueden ser el camino tomado, la velocidad

media y máxima, una gráfica de la velocidad y el perfil vertical de la pista, la velocidad media de cada kilómetro, número de sesiones de entrenamiento, kilómetros completados durante un periodo, etc. Estos datos pueden ser comparados con los de otros miembros de la red.

2.2 Wikilok

Wikiloc es una aplicación web híbrida donde se pueden almacenar y compartir rutas al aire libre georreferenciadas y puntos de interés de todo el mundo.

Esta aplicación permite crear tus propias rutas GPS y subirlas, utilizar el móvil de navegador, hacer seguimiento en vivo del track realizado y utilizar rutas de otros usuarios que hayan sido subidas.

2.3 ViewRanger

ViewRanger es una app móvil destinada al senderismo y rutas para hacer deporte que contiene una base de datos de rutas de miles de localizaciones.

Esta aplicación nos permite grabar una ruta que estemos realizando sin tener que seguir indicaciones, creando así nuestras propias excursiones. Pero, también seguir rutas que otros usuarios han subido previamente teniendo acceso además a mucha información sobre la excursión que vamos a realizar, sabiendo que puntos de interés vamos a encontrar, tiempo vamos a emplear, desnivel, dificultad y otra información de interés. La app también nos ofrecerá opciones de rutas próximas disponibles en la zona que nos encontremos, pudiendo delimitar según busquemos una u otra cosa.

3 OBJETIVOS

El objetivo principal del proyecto consiste en el desarrollo de una aplicación con interfaz gráfica, capaz de unificar rutas cargadas por el usuario en un único grafo y, generar rutas desde un punto de origen a un punto final seleccionados por el usuario, teniendo en cuenta posibles restricciones seleccionadas también por este.

Para un funcionamiento correcto, la aplicación tiene que seguir un flujo de procesos. Este flujo, comienza por mostrar el mapa que el usuario cargue en la aplicación. Seguidamente, el usuario tiene que subir las rutas con las que desee trabajar que correspondan a la sección de mapa seleccionada. Estas rutas, se leen y fusionan creando un grafo único como representación de todas las posibles rutas de la zona. Este grafo, se muestra sobre el mapa y el usuario tiene que escoger un punto de partida y uno de finalización sobre este. Si lo desea, el usuario también puede escoger restricciones para que la ruta se adapte lo máximo posible a sus necesidades como restricciones de longitud, de desnivel acumulado o puntos intermedios en la ruta los cuales se desean visitar. Con estos parámetros, se aplica un algoritmo de búsqueda que encuentra la mejor ruta entre los dos puntos respetando las restricciones seleccionadas. Finalmente se muestra la ruta sobre el mapa y el usuario tiene la posibilidad de guardar la ruta creada.

Con mayor concreción, las acciones que pude realizar el programa son:

- Cargar el mapa que el usuario desee, mostrándolo correctamente y con libertad de movimiento sobre él, es decir, poder hacer zum y desplazarse sobre este. Esta acción tiene una importancia alta en el funcionamiento de la aplicación.
- Cargar rutas seleccionadas por el usuario, unificarlas en un único grafo y mostrarlas correctamente sobre el mapa previamente cargado. Esta acción tiene una importancia muy alta en el funcionamiento de la aplicación.
- Seleccionar un punto de origen y un punto de destino sobre el grafo mostrado. Esta acción tiene una importancia muy alta.
- Seleccionar parámetros de la ruta como una longitud concreta o un desnivel acumulado. Esta acción tiene una importancia media-baja, ya que la aplicación puede funcionar sin ella.
- Seleccionar puntos intermedios que se deseen visitar para que la ruta creada pase por ellos. Esta acción tiene una importancia media-baja, ya que la aplicación puede funcionar sin ella.
- Crear la ruta con un algoritmo de búsqueda respetando todos los parámetros seleccionados por el usuario y mostrarla sobre el mapa. Esta acción tiene una importancia muy alta.
- Guardar la ruta creada por el algoritmo en un fichero nuevo. Esta acción tiene una importancia alta.
- Guardar el grafo creado en un nuevo fichero. Esta acción tiene una importancia baja.
- Cargar mapas de elevaciones para mejorar la precisión de las elevaciones del grafo y poder mostrar un mapa de calor sobre el mapa topográfico cargado. Esta acción tiene una importancia baja.

A continuación, se desglosan los objetivos del proyecto para detallar que metas se pretenden conseguir en cada uno de ellos y su prioridad.

3.1 Lectura de datos

Este objetivo trata de leer y almacenar correctamente en el programa los ficheros de entrada necesarios. Hay 3 tipos de archivos diferentes que leer.

Por una parte, están los archivos que contienen los mapas, los cuales son necesarios leer y tratar adecuadamente para poder visualizar sobre ellos las rutas existentes y la generada por el programa. Estos archivos existen en formato GeoTIFF[3] y ECW[4].

Por otro lado, están los archivos que contienen las rutas que acabaran formando el grafo final. Estos archivos están en formato GPX[5], que es una versión muy parecida

- *E-mail de contacto: adria.gomeza@autonoma.cat*
- *Mención realizada: Computación*
- *Trabajo tutorizado por: Javier Sánchez Pujades*
- *Curso 2021/22*

a los XML[6]. En ellos, se encuentra la posición del punto registrado, la altura y la fecha en la que se registró.

Estos dos primeros tipos de archivos son esenciales para el funcionamiento del programa así que su implementación es crítica.

Por último, están los archivos que corresponden a los mapas de alturas. Con ellos se puede implementar la restricción de elegir el camino con menos elevación acumulada o con mayor desnivel de una forma muy precisa.

Este último tipo de archivo es menos prioritario ya que los archivos GPX ya contienen la elevación de cada punto, aunque esta no pueda ser del todo correcta. Por ello, no se le dará una importancia crítica a su implementación, sino condicional.

3.2 Visualización de datos

Este, consiste en visualizar de forma correcta los datos leídos y almacenados correspondientes a los mapas y las rutas.

En la interfaz gráfica, se tiene que poder mostrar el mapa leído y sobre el ser capaz de dibujar tanto rutas como el grafo generado por todas las rutas leídas.

Este objetivo es esencial en el desarrollo de la aplicación y su implementación es crítica.

3.3 Creación del grafo

En este punto, se trata de generar un grafo único a partir de todas las rutas leídas.

Para generar el grafo, hay que tener en cuenta que no se deben repetir aristas como si fuesen tramos diferentes entre nodos, también la dirección de cada tramo ya que no todos los tramos pueden hacerse en ambos sentidos y por último, buscar que puntos son los mismos en rutas diferentes para evitar nodos repetidos y poder generar un grafo consistente que represente bien la zona.

Este objetivo tiene una prioridad máxima y su desarrollo es crítico para el correcto funcionamiento de la aplicación.

3.4 Selección de parámetros

Este objetivo, consiste en la selección de los datos con los que buscar la ruta deseada en el grafo creado.

Sobre el grafo generado, el usuario tiene que ser capaz de seleccionar un punto de partida de la ruta y un punto donde esta debe finalizar. Además, de unos parámetros seleccionables para poder añadir restricciones a las rutas. Estos parámetros seleccionables pueden ser, ruta más corta, ruta más larga, ruta con menos desnivel, ruta con más desnivel, incluso combinaciones de ellas o seleccionar a partir de que grado de desnivel buscar.

La implementación de la selección de origen y fin es esencial para el funcionamiento básico del programa, en cambio los parámetros seleccionables son menos importantes ya que sin ellos, la función básica de la aplicación funcionaria correctamente por lo que se le da una prioridad condicional

3.5 Generación de la ruta

Este, consiste en generar la ruta a partir del grafo y los parámetros seleccionados por el usuario.

Con el grafo de rutas de la zona, el punto de origen y de finalización y los posibles parámetros seleccionables, a partir de un algoritmo de búsqueda, se tiene que encontrar la ruta que mejor cumpla con las condiciones del usuario y mostrarla sobre el mapa, con la posibilidad de generar un archivo GPX con ella para su utilización en otras aplicaciones o el simple almacenamiento de esta.

El desarrollo de este objetivo es esencial para la finalización del proyecto.

3.6 Desarrollo de la interfaz de usuario

La aplicación debe tener una interfaz de usuario agradable, intuitiva y sencilla de utilizar, con todo lo necesario para poder realizar los diferentes procesos del programa.

La creación de una buena interfaz es esencial para el proyecto.

3.7 Objetivos adicionales

En este punto, se mencionan objetivos opcionales cuya implementación no es básica para el funcionamiento principal de la aplicación, sino que son funcionalidades extras.

Poder visualizar un mapa de calor sobre el mapa de la zona para poder visualizar las elevaciones y desniveles del terreno.

Poder seleccionar puntos intermedios entre el origen y el fin de la ruta para obligar pasar a la ruta generada por zonas concretas.

Poder generar un archivo GPX a partir del grafo creado para almacenarlo y poder volver a cargarlo en la aplicación cuando se desee.

4 METODOLOGÍA

Para el proyecto se va a utilizar Trello[7], que es un software de administración de proyectos con interfaz web, con un diseño simple e intuitivo de utilizar, con el que se van a organizar las fases del proyecto con sus fechas esperadas de realización y prioridad.

Utilizando Trello, se va a seguir la metodología Agile[8] con la que el proyecto dividido en varias partes va a ir creciendo iterativamente con la realización de pequeños *sprints*. Además, con esta metodología se responde bien a los cambios ya que el proyecto nunca debe envararse en un plan estricto.

5 PLANIFICACIÓN

En este apartado se detallan las fechas de entrega de los *sprints* y las tareas a realizar en cada uno de ellos.

El proyecto se divide en tareas las cuales la mayoría de ellas siguen un desarrollo secuencial donde la finalización de una indica el comienzo de otra. Estas tareas están distribuidas en los diferentes *sprints* de la siguiente forma:

Sprint 1 (semanas 0 - 4)

Precisar y planificar el proyecto

A partir de la realización de una primera reunión con el tutor, se acuerdan los objetivos a realizar, estos se detallan y se crea una planificación del proyecto. Esta tarea se desarrolla en el *sprint* 1.

Sprint 2 (semanas 5 - 9)

Lectura de los datos

En esta tarea, se busca y desarrolla como leer los ficheros GPX de entrada y guardarlos en un grafo sin restricciones, también la forma de leer los mapas topográficos en formato GeoTIFF o ECW y por último como leer los mapas de alturas. Esta tarea se desarrolla en las semanas 5 - 8.

Visualización de los datos

Después de leer todos los datos anteriores, se muestran mediante la interfaz gráfica el mapa topográfico leído y en él se dibuja el grafo sin restricciones (aristas y nodos repetidos) generado por las rutas. De esta forma, ya se conocerá la forma de mostrar cualquier tipo de grafo ya sea una ruta o el grafo único que hay que crear. Esta tarea se desarrolla en la semana 9.

Sprint 3 (semanas 10 - 14)

Generación del grafo

Con todas las rutas leídas, se tiene que crear un grafo único sin nodos ni aristas repetidas y mostrarlo sobre el mapa topográfico visualizado anteriormente. Esta tarea se desarrolla en las semanas 10 - 13.

Selección de puntos de origen y fin

Con el grafo creado y mostrado, el usuario tiene que poder seleccionar en él, un punto de origen de la ruta y un punto donde finalizarla. Esta tarea se desarrolla en la semana 14.

Sprint 4 (semanas 15 - 19)

Selección de parámetros

En la interfaz gráfica, se tienen que mostrar unos parámetros seleccionables por el usuario para indicar condiciones en la creación de la ruta. Esta tarea se desarrolla en la semana 15.

Algoritmo de búsqueda

Con todos los parámetros seleccionados, se tiene que desarrollar un algoritmo de búsqueda de caminos que sea capaz de encontrar la ruta que se adapte mejor a las necesidades del usuario en un tiempo adecuado. Esta tarea se desarrolla en la semana 16 - 19.

Sprint 5 (semanas 20 - 22)

Generación de fichero con ruta

Con la ruta ya encontrada, se tiene que mostrar sobre el mapa en la interfaz gráfica y además generar un archivo GPX con la ruta encontrada. Esta tarea se desarrolla en la semana 20.

Paradas intermedias

Con la funcionalidad básica de la aplicación creada y funcionando adecuadamente, se añade a la selección del punto de origen y punto de finalización de la ruta, puntos intermedios por donde la ruta tiene que pasar. Esta tarea se desarrolla en las semanas 21 – 22.

Generación de fichero con grafo

Con la funcionalidad básica de la aplicación creada y funcionando adecuadamente, se desarrolla la posibilidad de almacenar en un fichero GPX el grafo creado en la lectura de las rutas. Esta tarea se desarrolla en las semanas 21 – 22.

Mapa de calor

Con la funcionalidad básica de la aplicación creada y funcionando adecuadamente, se añade la opción de ver un mapa de calor con las alturas de la zona sobre el mapa topográfico en la interfaz. Esta tarea se desarrolla en las semanas 21 – 22.

Desarrollo de la interfaz

Durante el desarrollo de todas las tareas, se va a ir creando una interfaz gráfica a la cual se le van a ir añadiendo funcionalidades cuando sean necesarias para que el proyecto pueda seguir desarrollándose correctamente. Se desarrolla incrementalmente durante todos los *sprints*.

6 DESARROLLO

Para desarrollar el proyecto, es necesaria la selección de un lenguaje de programación con el que realizar todo el back-end de la aplicación y un buen IDE de desarrollo que facilite la creación de la interfaz gráfica y la conexión entre back-end y front-end. El lenguaje de programación escogido es C++ ya que es un leguaje compilado, es decir, rápido en su ejecución, con multitud de librerías y mucha documentación para el desarrollo de programas. Como IDE se ha seleccionado QtCreator[9] junto con QtDesignStudio[10] que ofrece muchas herramientas para la creación de interfaces gráficas y para su conexión con el back-end.

6.1 Lectura de los datos

La lectura de los datos de entrada ha sido la primera tarea investigada y desarrollada del proyecto. Se ha cumplido satisfactoriamente gracias a la utilización de librerías que QtCreator ofrece y librerías externas al IDE.

6.1.1 Lectura de rutas

Previamente a la lectura de las rutas, se ha tenido que crear el código capaz de generar un grafo con sus nodos y aristas donde cada nodo sea capaz de guardar la información necesaria leída del fichero.

Las rutas que el programa tiene que leer, son seleccionadas por el usuario, es decir, la aplicación permite al usuario cargar las rutas que el tenga almacenadas en su

ordenador.

Los archivos que contienen las rutas son archivos GPX que son un tipo de archivos XML, algo que facilita las cosas ya que QtCreator cuenta con una librería propia para la lectura de ficheros XML. Esta librería llamada QtXml[11], es capaz de moverse por las diferentes etiquetas del documento, permitiendo acceder a aquellas que contengan la información necesaria para la creación de los nodos. Con esta librería y gracias a una función recursiva, el programa es capaz de recuperar la longitud, latitud y elevación de cada punto registrado, crear un nodo con estos datos y unirlo con el anteriormente leído y el siguiente nodo que se cree. De esta forma se pueden leer tantos archivos GPX como se desee y el grafo ira creciendo, creando nuevos nodos con cada punto registrado y nuevas aristas entre ellos. Esta lectura esta universalizada y es capaz de leer correctamente tanto archivos GPX propios generados por una aplicación, como archivos GPX descargados de internet. Con el algoritmo de lectura actual, el programa también es capaz de leer archivos GPX que contengan varias rutas en el.

6.1.2 Lectura de mapas topográficos

El mapa sobre el que trabajará la aplicación es seleccionado por el usuario de la misma forma que las rutas, permitiendo subir la sección de territorio que el usuario desee.

Para la lectura de mapas topográficos en este proyecto, se utiliza la librería OpenCv[12]. Esta es una librería externa a QtCreator, implementada para el desarrollo de visión por computador. Aun siendo una librería dedicada para la visión por computador, ofrece la posibilidad de lectura de archivos ráster geoespaciales integrando GDAL en él, que es otra librería, esta si, dedicada puramente a desarrollo de programas geoespaciales. Con esta librería, se pueden leer archivos GeoTIFF. Estos mapas se han obtenido de las páginas web del Instituto Geográfico Nacional[13] y del Institut Cartografic i Geológic de Catalunya[14]. Para el desarrollo de la aplicación se opta por los archivos GeoTIFF ya que tienen una resolución muy alta y son idóneos para trabajar con ellos en un planificador de rutas de montaña. Por contra, al tener una resolución tan alta, son bastante pesados cosa que ralentiza el programa. Estos archivos TIFF, se georreferencian gracias a un archivo TFW que los acompaña, donde se encuentran las coordenadas mundo en formato UTM de la esquina superior izquierda de la imagen.

6.2 Visualización de los datos

Qt ofrece librerías para cargar y visualizar datos en la interfaz dinámicamente, de forma que se pueden añadir y eliminar elementos en esta en tiempo de ejecución del programa.

6.2.1 Visualización de rutas

Las rutas leídas y el grafo generado se dibujarán sobre un Canvas[15] en el front-end. Para ello, previamente hay que convertir ese punto que está en coordenadas geográficas en grados decimales, a coordenadas píxel de la imagen y que esta, concuerde con el lugar que le corresponde en el mapa. Para poder hacerlo, se necesita el archivo TFW[16] correspondiente al TIFF, que contiene las coordenadas mundo UTM[17] de la esquina superior izquierda de la imagen y la medida de un pixel de la imagen en el mundo. Como los puntos leídos de los GPX contienen las coordenadas en grados decimales, se tienen que convertir a UTM. A parte, es necesario conocer el escalado de la imagen, ya que para que se ajuste a la pantalla se le aplica uno. Conociendo las coordenadas del punto en UTM, las coordenadas de la esquina superior izquierda, la medida de un píxel y el escalado de la imagen, se utilizan las siguientes fórmulas para conseguir tener el punto bien posicionado en la imagen:

$$x = (xPunto - xMapa)/(xSize * escalado)$$
$$y = (yPunto - yMapa)/(ySize * escalado)$$

Qt, ofrece la posibilidad de emitir *signals*[18] (señales) a la interfaz desde el back-end, enviando así unos datos concretos en un punto de la ejecución que se desee. Gracias a esto, cada vez que los puntos de los extremos de los tramos (aristas) hayan sido convertidos a coordenadas imagen, se emite una señal enviando los puntos al front-end, donde se almacenan en una array. Finalmente, se dibuja sobre el Canvas una recta entre cada par de puntos. Haciendo esto para todas las aristas del grafo, se consigue mostrarlo en el mapa sobre los caminos correspondientes con una gran precisión.

6.2.2 Visualización de mapas topográficos

Con el mapa ya en el programa gracias a la librería OpenCv, se utiliza la librería de Qt QQuickImageProvider[19] que permite coger una imagen cargada en el programa y crear una conexión con la interfaz mostrándola como se desee. Esta librería, llama a una función del back-end que, gracias a un parámetro que se le envía desde el código de la interfaz, la función sabe que parte de código ejecutar. En este caso, para mostrar el mapa inicial se envía "-1". Esto se ha desarrollado así, para poder implementar el zum del mapa. Para el funcionamiento del zum, se envía el parámetro a la función desde el código de la interfaz, pero en este caso, al mover la rueda del ratón hacia delante, el parámetro se incrementa en uno y al mover la rueda hacia atrás se disminuye. El valor de este parámetro se compara con el anteriormente enviado. En caso de ser más grande, querrá decir que el usuario quiere incrementar el zum, si es menor, se querrá disminuir el zum. Por último, este parámetro se guarda para poder compararlo en la siguiente llamada a la función.

Aparte del zum, este código también permite arrastrar la imagen para enfocar la sección del mapa que se desee. Esto se hace de la misma forma, mediante el parámetro, al enviar un -2, se ejecuta la parte del código que permite realizar esta acción.

6.3 Generación del grafo

La generación de un grafo único a partir de todas las rutas leídas es el pilar fundamental de este proyecto ya

que, permite unir las rutas y hace posible la búsqueda de caminos. Para optimizar el rendimiento del programa, este grafo se va creando conforme se van leyendo las rutas de los ficheros GPX, es decir, si al leer un nuevo nodo, este se considera duplicado, se descarta. Primeramente, un nodo se dará por valido si se encuentra a 11 metros o más del anterior, esta distancia es suficiente para conseguir un grafo final con una precisión muy alta sin almacenar nodos innecesarios que apenas aportan información relevante.

Un nodo se considera duplicado si este se encuentra a menos de 11 metros de otro nodo guardado anteriormente en el grafo. Para poder realizar esta comprobación de una forma más fluida, sin la necesidad de comparar este nodo con otros que se puedan encontrar muy alejados, se divide el área de mapa que se encuentra en la pantalla en 100 cuadrantes diferentes. Cada nodo valido, se guarda en el cuadrante que le toca, de esta forma al realizar la comprobación de duplicado, el programa se centra únicamente en la zona afectada. Para realizar una búsqueda más ágil en cada cuadrante, los nodos de cada cuadrante se guardan ordenados de menor a mayor según su coordenada x en formato UTM. Gracias a esto, se hace una comprobación preliminar con las coordenadas x de los nodos y se puede detener la búsqueda si el nodo a comprobar se encuentra a +11 metros de otro ya que, gracias a estar ordenados, los siguientes estarán más alejados y no es necesaria su comprobación. Además de comprobar el cuadrante que le toca, en caso de que el nodo esté a menos de 11 metros de otro cuadrante, este se comprueba de la misma forma.

En caso de dar por valido el nodo, se guarda y se crea una arista del anteriormente leído a este. Si el nodo se considera duplicado, se descarta, se escoge como nodo actual el nodo por el que se ha considerado duplicado, y se crea una arista del anterior a este, en caso de que esta no exista.

Hasta este punto, se crea un grafo bastante fiable, pero con errores. Esto es debido a que los generadores de rutas pueden tener algún error en la geolocalización al crear un punto nuevo, y dos caminos que se muestran como diferentes en el grafo realmente son el mismo camino y deberían unirse. Para solucionarlo, se cuentan los nodos transcurridos desde un nodo duplicado al siguiente nodo duplicado que se encuentre. Si el numero de nodos es menor a un numero K, se analiza esa sección ya que puede ser un camino duplicado, si no, se guarda como un nuevo camino. Para determinar si el camino a analizar es duplicado o no, se mira si existe otro camino que en N o menos nodos pueda llegar del origen al destino del camino. Si existen uno o varios caminos, se calculan las distancias de los diferentes caminos y se comparan con el nuevo. En caso de que la diferencia entre las distancias sea menor a 20 metros, se considera que el camino nuevo es un camino duplicado y se descarta. Por el contrario, si no existe otro camino, o los existentes tienen una longitud superior a 20 metros, este nuevo camino se guarda como valido.

Con todo esto, el programa es capaz de generar un grafo muy bueno sin nodos duplicados, ni caminos redundantes, y todo y ser un algoritmo bastante complejo con muchos pasos, el grafo se crea con un rendimiento alto.

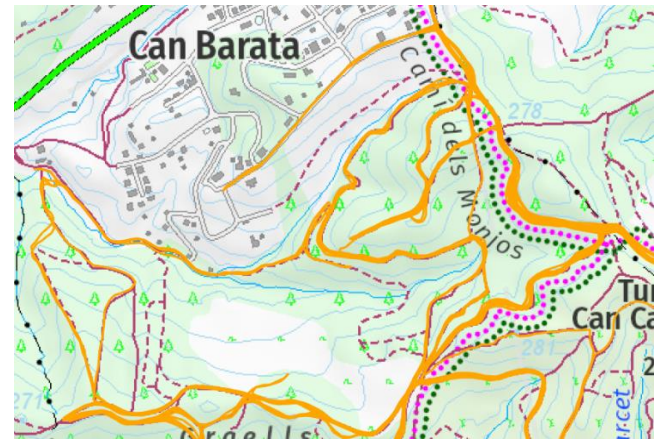

Figura 1: Lectura de rutas sin creación del grafo

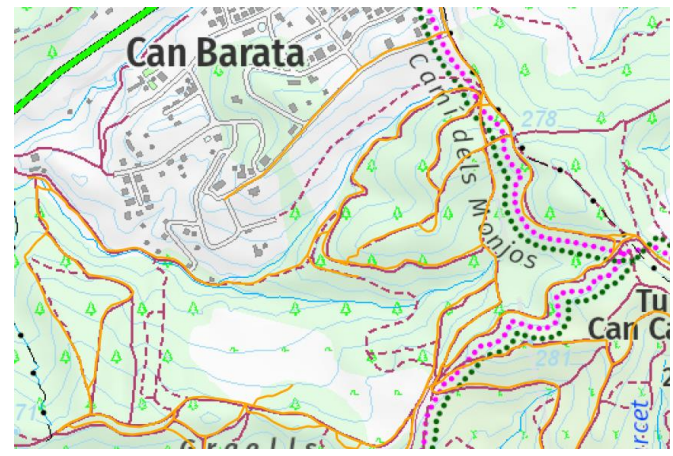

Figura 2: Rutas leídas y grafo creado

6.4 Selección de puntos de origen y fin

El usuario tiene que poder seleccionar un punto de origen y un punto de fin sobre el grafo mostrado por pantalla. Para conseguir un buen programa, se añaden las restricciones de que el usuario no pueda seleccionar el mismo punto como origen y como final y que el usuario únicamente pueda seleccionar puntos pertenecientes al grafo.

El programa funciona de forma que cuando el usuario hace clic sobre el mapa, se busca el nodo más cercano a este píxel clicado. Si este nodo encontrado está a una distancia menor a N pixeles del punto sobre el que se ha clicado, se marca como seleccionado el píxel perteneciente al nodo más cercano. De esta forma se crea un magnetismo alrededor de cada nodo ya que será detectado siempre que se seleccione un punto que se encuentre a menos de N pixeles en cualquier dirección, y no únicamente si se clica exactamente en él. En caso de no cumplir la restricción, no se selecciona ningún punto del mapa. Cuando un punto es seleccionado, se muestran por pantalla las coordenadas geográficas en grados decimales de este y una chincheta sobre el mapa de forma muy precisa sobre el nodo del grafo.

6.5 Selección de parámetros

A parte de seleccionar un origen y un fin para la generación de una ruta, al usuario se le permite seleccionar una distancia y un desnivel acumulado para que la ruta generada se acerque lo máximo posible a estos parámetros. Esto se permite hacer gracias a dos barras deslizables que se encuentran en la interfaz. Estas barras inicialmente se encuentran en 0 y no permiten seleccionar otra distancia, pero al cargar las rutas, el rango cambia y se permite seleccionar un valor entre 0 y la suma de todas las aristas del grafo, que es el valor máximo que se puede llegar a encontrar en ese grafo, tanto para la distancia como para el desnivel. Finalmente, el valor seleccionado para cada parámetro se pasa al algoritmo de búsqueda para que este trabaje con ellos.

6.6 Paradas intermedias

Además de los parámetros, al usuario también se le permite activar la función de seleccionar paradas intermedias, que de la misma forma que para la selección del origen y el final de la ruta, permite seleccionar puntos sobre el grafo que se guardan como paradas o puntos por los cuales la ruta generada tiene que pasar en algún momento del recorrido. Esta selección no cuenta con un número máximo de paradas fijado, pero cuantas mas paradas seleccionadas, más difícil le será al algoritmo de búsqueda dar con la ruta deseada. Las paradas seleccionadas tampoco tienen un orden determinado, sino que el algoritmo de búsqueda decidirá su orden en función de la ruta encontrada. La selección de puntos cuenta con el mismo magnetismo que par el origen y el final, las únicas diferencias son que no se muestra la latitud ni longitud del punto seleccionado por pantalla y que sobre el mapa esta vez se mostrará una chincheta amarilla sobre el punto seleccionado.

6.7 Algoritmo de búsqueda

Con un origen y un final de la ruta seleccionados y con todos los parámetros y paradas intermedias que el usuario desee, el programa tiene que ser capaz de encontrar una ruta que se adapte lo máximo posible a las necesidades del usuario, y en un tiempo de búsqueda razonable. Para ello, se van a utilizar Dijkstra[20], Greedy[21] y Backtracking[22].

Antes de iniciar la búsqueda, para que los algoritmos se ejecuten de la forma más rápida posible y con la menor complejidad, se crea una versión simplificada del grafo creado en la lectura de las rutas. Este grafo contine únicamente aquellos nodos que correspondan a bifurcaciones y aquellos seleccionados como origen, fin o paradas intermedias. De esta forma se eliminan todos aquellos nodos que no aportan información relevante a la búsqueda ya que pertenecen a la unión entre dos nodos relevantes y únicamente tienen la función de crear el tramo del punto A al punto B. Por lo tanto, se pueden sustituir por una arista la cual contiene la suma de todas las distancias y desniveles de las aristas existentes entre los dos nodos que si son relevantes. Con esto se consigue reducir mucho el numero de nodos y aristas sobre los cuales buscar el mejor camino, cosa que reduce el tiempo de búsqueda significativamente.

Con el grafo simplificado ya creado, en caso de que el usuario no haya seleccionado ningún parámetro de distancia ni desnivel, únicamente se utilizarán Dijkstra y Greedy que en conjunto y teniendo en cuenta tanto la distancia de cada arista como el desnivel de estas, generan una ruta desde el origen hasta el fin pasando por todas las paradas que hayan sido seleccionadas de una forma muy rápida. Esta ruta en muchos casos será aquella con menor dificultad en cuanto a desnivel y distancia, pero no siempre ya que Greedy no es capaz de dar siempre con la mejor solución, sino que se centra mas en la optimización de la generación de la ruta, cosa que es idónea en este caso ya que el usuario no selecciona una distancia ni un desnivel deseado, sino que únicamente se genere una ruta que pase por los puntos seleccionados.

En cambio, si el usuario selecciona una distancia deseada, un desnivel deseado o ambas cosas a la vez, el algoritmo utilizado es Backtracking con la ayuda de Dijkstra, Greedy y diferentes heurísticas para agilizar la búsqueda. En este caso, el algoritmo funciona de forma que antes de ejecutar el Backtracking, se ejecuta Dijkstra junto a Greedy de la misma forma que si el usuario no hubiese seleccionado ningún parámetro. Gracias a esto, Backtracking no empezará su búsqueda desde cero, sino que parte con una ruta base que en la mayoría de los casos no será la ruta final, pero sirve para podar caminos en la búsqueda y agilizarla. Con esta ruta ya encontrada, se marcan como mejor distancia y desnivel encontrados hasta el momento los correspondientes a esta. Para agilizar aún más la búsqueda, se utiliza una heurística con la que se ordenan las aristas de cada nodo en función de los parámetros seleccionados y la ruta encontrada con Greedy, es decir, si Greedy ha encontrado una ruta de longitud L en X aristas, con esto se hace una aproximación de lo que medirá cada arista en proporción a la distancia seleccionada, y las aristas de cada nodo se ordenarán en función de este factor calculado. Con todo esto, se inicia Backtracking, con el que se busca una ruta mejor a la encontrada actualmente, en la que no se pueden repetir aristas, es decir, si se ha ido desde un punto A a un punto B, no se puede volver a pasar por esta arista en el mismo sentido, pero si se puede pasar por la reversa a esta para poder visitar caminos sin salida y poder generar rutas más completas.

Backtracking dará por valida una ruta cuando se llegue al nodo destino, habiendo pasado por todas las paradas intermedias y si la diferencia entre la distancia o desnivel de la ruta actual y la distancia o desnivel seleccionados por el usuario es menor a la diferencia de la mejor ruta encontrada hasta el momento. Si la diferencia entre los parámetros seleccionados y la ruta encontrada es menor a 1, se detiene la búsqueda y se devuelve esta ruta como la mejor encontrada ya que se considera que cumple completamente con los parámetros seleccionados. La poda de caminos se realiza de forma que, en el caso de la distancia, se le suma a la distancia del camino encontrado hasta el momento la distancia euclidiana del nodo actual hasta el nodo final. En caso de que la diferencia entre la distancia calculada y la deseada sea mayor a el valor ab-

soluto de la diferencia entre la mejor encontrada hasta el momento y la deseada, se poda el camino. De la misma manera sea hace para el desnivel con la única diferencia de que en vez de la distancia euclidiana entre el nodo actual y el final, se utiliza el desnivel entre nodo actual y nodo final. En el peor de los casos, el grafo creado puede hacer que Backtracking no encuentre una solución de forma rápida o que no acabe encontrado una solución que se adapte a los parámetros, cosa que hará que su ejecución no acabe hasta completar todos los posibles caminos existentes. Para un grafo grande, esto es inviable ya que la ruta encontrada se mostraría en un tiempo de ejecución demasiado elevado. Para solucionarlo, se ha establecido un tiempo de búsqueda máximo. Es decir, se irán buscando soluciones y la solución ira siendo cada vez mejor, pero en el caso de que hayan transcurrido 30 segundos desde el inicio de la búsqueda, se devolverá la mejor solución encontrada hasta el momento y se detendrá la búsqueda.

Por último, se reconstruye el camino encontrado ya que este ha sido generado con el grafo simplificado, y para su utilización es necesario convertir la ruta a su equivalente en el grafo completo.

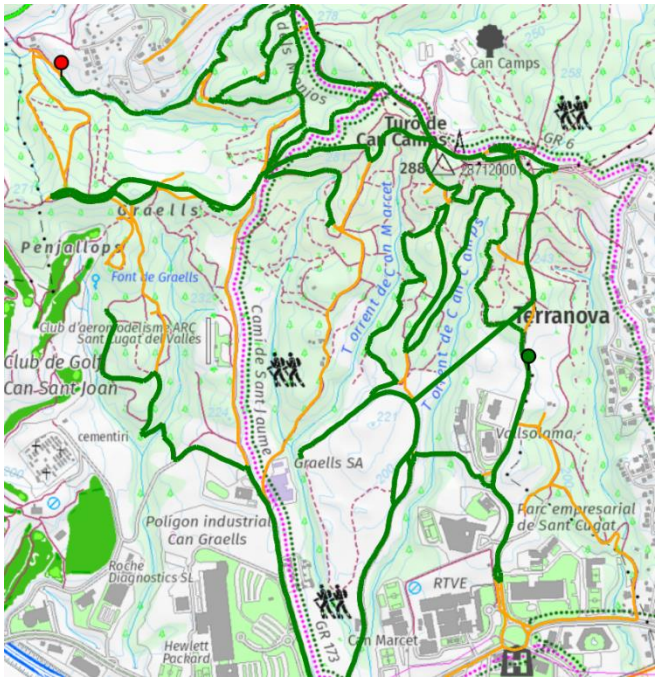

Figura 3: Ruta generada por la aplicación. Buscada con 28.5km de distancia y 1950m de desnivel acumulado deseados. Resultado con 28.512km de distancia y 2421.1m de desnivel tras 30s de búsqueda.

6.8 Generación de fichero con ruta

Al tener la ruta que mejor se adapte a las necesidades del usuario ya creada, el programa tiene que ofrecer la posibilidad de guardarla en un fichero GPX para su utilización en esta misma aplicación o para utilizarla en GPS de montaña y poder realizarla sin problemas. Esta ruta se guarda en formato XML utilizando la librería propia de qt QtXml, con la estructura necesaria para una correcta lectura en otros dispositivos, almacenando todos los puntos de esta, con su latitud, longitud y elevación.

6.9 Generación de fichero con grafo

El programa ofrece la posibilidad de guardar el grafo generado en la lectura de las rutas en un fichero GPX para utilizarlo posteriormente o fusionarlo con mas rutas en el futuro. Para su almacenamiento se utiliza la misma librería y estructura que para guardar la ruta generada, con la diferencia de que, al ser un grafo, el archivo GPX contendrá un conjunto de rutas las cuales lo forman. Estas rutas son cada camino existente entre cada par de bifurcaciones con conexión entre sí. De esta forma, se consiguen guardar múltiples rutas independientes en e archivo que, gracias al algoritmo de lectura de rutas, el programa es capaz de unir y así reconstruir el grafo guardado.

6.10 Desarrollo de la interfaz

QtCreator ofrece un entorno de desarrollo de interfaces llamado QtDesignStudio. Este entorno permite crear interfaces graficas en formato QML con el que se puede modelar la interfaz a gusto del desarrollador y ofrece herramientas para su conexión con el back-end.

La principal librería de Qt utilizada es QObject[23], que permite crear una clase, la cual se puede acceder desde la interfaz y puede realizar cambios en ella. Tiene la función de controlador creando la conexión entre back-end y fornt-end.

Se ha conseguido desarrollar una interfaz sencilla, con los elementos necesarios para realizar cada acción e intuitiva ya que su utilización no es nada complicada. La interfaz se compone de un mapa el cual carga el usuario y que ocupa toda la pantalla para que se pueda ver bien y con todo detalle. Un cuadro con los datos de los puntos seleccionados como origen y como fin, y también contiene los datos de distancia y desnivel de la ruta generada por el algoritmo. A parte, también se han introducido dos barras deslizantes para poder seleccionar la distancia y el desnivel acumulado que se desee de forma muy sencilla. Por último, existen 9 botos, cada uno enlazado a una acción de la aplicación, las cuales son: cargar mapa, cargar rutas, eliminar rutas, guardar la ruta creada, guardar el grafo creado, añadir paradas intermedias, eliminar paradas intermedias, reelegir los puntos seleccionados sobre el mapa y por último buscar la ruta.

A parte de los elementos visuales, existen funciones en JavaScript (soportadas por qml) para poder realizar las acciones correctamente y que la conexión con el back-end sea factible.

7 Conclusiones

La aplicación desarrollada es capaz de leer un mapa en formato GeoTIFF y mostrarlo por pantalla, haciendo posible también la navegación sobre este mediante zum y arrastrar. Con el mapa cargado, el usuario puede iniciar la lectura de rutas presionando el botón correspondiente, con el que también se lanza la ejecución de la creación del grafo. Con todas las rutas leídas y el grafo creado, la aplicación es capaz de mostrarlo por pantalla sobre el mapa con una precisión muy alta marcando los caminos exactos

que se han recorrido en las diferentes rutas cargadas. Seguidamente, la aplicación permite al usuario seleccionar un punto de origen y un punto de fin sobre el grafo mostrado, mostrando una chincheta sobre los puntos seleccionados y la latitud y longitud correspondiente a cada uno. Además, permite al usuario seleccionar puntos intermedios por los cuales desea que pase la ruta mostrándose una chincheta amarilla sobre el punto seleccionado. También, añade la posibilidad de seleccionar una distancia y un desnivel acumulado que se desee. Con todos los parámetros deseados seleccionados, el usuario puede lanzar la búsqueda de la ruta la cual, dependiendo de los parámetros, ejecutará el algoritmo necesario para ello. La ruta encontrada se muestra por pantalla, mostrando también su distancia de origen a fin y el desnivel acumulado de toda ella. Por último, la aplicación también permite al usuario guardar la ruta creada para su utilización en otros dispositivos, y la posibilidad de guardar el grafo creado en la lectura de las rutas para su futura utilización en la aplicación.

Con todo esto, se ha conseguido desarrollar una aplicación muy completa la cual realiza prácticamente todas las acciones esperadas. Estas acciones, además se ejecutan con un muy buen rendimiento siendo capaz de generar grafos en muy pocos segundos con una gran precisión y encontrando rutas que se adaptan con gran exactitud a las necesidades del usuario en un tiempo no superior a 30 segundos en todos los casos. Todo y esto, se le pueden realizar mejoras ya que los algoritmos no son perfectos y tanto la generación del grafo como la búsqueda de las rutas tienen pequeños errores solucionables o simplemente mejoras en el rendimiento.

BIBLIOGRAFIA

[1] Web oficial Wikiloc. Recuperado el 5 de octubre de 2021, de: https://es.wikiloc.com

[2] Web oficial Strava. Recuperado el 5 de octubre de 2021, de: https://www.strava.com

[3] GeoTIFF Wikipedia. Recuperado el 6 de octubre de 2021, de: https://en.wikipedia.org/wiki/GeoTIFF

[4] Extension de archivo ECW. Recuperado el 6 de octubre de 2021, de: https://www.file-extension.info/es/format/ecw

[5] GPX Wikipedia. Recuperado el 6 de octubre de 2021, de: https://es.wikipedia.org/wiki/GPX

[6] XML Wikipedia. Recuperado el 6 de octubre de 2021, de: https://es.wikipedia.org/wiki/Extensible_Markup_Language

[7] Web oficial Trello. Recuperado el 8 de octubre de 2021, de: https://trello.com

[8] Metodología Agile. Recuperado el 8 de octubre de 2021, de: https://www.santaluciaimpulsa.es/metodologia-agile-que-es-para-que-sirve/

[9] Web oficial QtCreator. Recuperado el 19 de octubre de 2021, de: https://www.qt.io/product/development-tools

[10] Web oficial QtDesignStudio. Recuperado el 19 de octubre de 2021, de: https://www.qt.io/design

[11] Tutorial QtXml. Recuperado el 23 de octubre de 2021, de: https://www.bogotobogo.com/Qt/Qt5_QtXML_QDomDocument_QDomElement.php

[12] Web oficial de OpenCv. Recuperado el 7 de noviembre de 2021, de: https://opencv.org/

[13] Web oficial Instituto Geográfico Nacional. Recuperado el 2 de noviembre de 2021, de: http://www.ign.es/web/ign/portal

[14] Web oficial Institut Cartografic i Geológic de Catalunya. Recuperado el 2 de noviembre de 2021, de: https://www.icgc.cat/

[15] Documentación canvas QML, web oficial Qt. Recuperado el 12 de noviembre de 2021, de: https://doc.qt.io/qt-5.15/qml-qtquick-canvas.html

[16] World file Wikipedia. Recuperado el 14 de noviembre de 2021, de: https://en.wikipedia.org/wiki/World_file

[17] UTM Wikipedia. Recuperado el 14 de noviembre de 2021, de: https://es.wikipedia.org/wiki/Sistema_de_coordenadas_universal_transversal_de_Mercator

[18] Signals and slots, web oficial Qt. Recuperado el 18 de noviembre de 2021, de: https://doc.qt.io/qt-5/signalsandslots.html

[19] Web oficial librería QQuickImageProvider. Recuperado el 9 de noviembre de 2021, de: https://doc.qt.io/qt-5/qquickimageprovider.html

[20] Dijkstra Wikipedia. Recuperado el 16 de enero de 2022, de: https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra

[21] Greedy Wikipedia. Recuperado el 16 de enero de 2022, de: https://es.wikipedia.org/wiki/Algoritmo_voraz

[22] Backtracking Wikipedia. Recuperado el 26 de enero de 2022, de: https://es.wikipedia.org/wiki/Vuelta_atr%C3%A1s

[23] Web oficial librería QObject. Recuperado el 4 de noviembre de 2021, de: https://doc.qt.io/qt-5/qobject.html