TrackIt! (SearchIt!)

Diogo Xavier
diogo.xavier@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2017

Abstract

The popularity of GPS devices has increased in the past years due to several factors like the decrease of their price and size and also its inclusion in mobile devices. For that reason, the number of applications supporting this type of devices has also increased. These applications are used by athletes that want to measure their performance of their outdoor activities. Some of these applications, despite their great utility, still lack some useful functionalities. This work focuses on TrackIt!, and describes the implementation of new functionalities. The principal functionality implemented was the search of activities and courses, with a calendar view of activities and courses. The solutions implemented by this work were tested by usability tests with users. The tests showed that less experienced users have more difficulty in first use, but that with time, that difficulty disappears. In the end of this work we present suggestions for future work and further improvements to the current application.
Keywords: GPS, Sports, Activities, Courses, Search, Calendar

1. Introduction

Popularity of GPS devices has increased in the last years and have had a big impact on our daily life.[1] Due to their evolution, there has been a decrease in their size and price and also the increase of their precision.[2] These devices provide a great range of information, from simple information like position, altitude and time to, depending on the extra sensors present in the device, information like heart rate and body temperature.

These devices have also been embedded in mobile devices like smart-phones and smart-watches that provide a big array of sensors that improve the devices functionalities (Internet, Bluetooth).

For the reasons above, applications have been created on the data generated by these devices. There are various types of applications, available to all platforms, with different objectives like navigation, course planning, analysis of sport performance data, sharing activities, online challenging and augmented reality games, to mention a few.

In this work we will focus on applications that target athletes. These applications provide athletes with means to record their performance during their activities, store activities, share activities, plan future activities or search and compare for past activities.[3]

This work focuses on TrackIt!, an application for amateur and professional athletes that provides several important functionalities. Its main objective is planning outdoor activities, supporting several edition tools. It also supports importing and exporting of several type of GPS files, and supports some of the most well known digital map providers.[4][5][6]

On the other hand, TrackIt! still had some problems and lacking some functionalities. For this reason, in this work, we expanded TrackIt! to make it more competitive in terms of functionalities.

One of the biggest problems with TrackIt! was its search of activities since it's was only possible to search by geographical area. With this work we improved and expanded that functionality by adding search by several criteria. Criteria can range from information like position or time to real world criteria like the track state or to user criteria like difficulty. Using these criteria we wanted to create a complex search with enough criteria to differentiate activities so that it became possible to easily search one activity out dozens or hundreds of activities. With this in mind we planned to expand activities and courses so that they contained new criteria. We also planned to increase the number of supported sports and subsports.

Some sports consider only descents, while others consider ascent or both. Because of this, we wanted to create a search by sport, a search that, depending on the selected sport and subsport, would change the criteria shown.

And lastly, we also wanted to implement a functionality that is present in our daily lives, a calendar view. With a calendar view, it would become easy to search for activities and courses in a view known by everyone, making this view really intuitive for the user.

1.1. Goals

The goals we planed to achieve were:

- Adding new sports and subsports
- Adding new criteria to activities and courses
- Implementation of the search algorithm
- Implementation of the search interface
- Implementation of the calendar view

2. State of the Art

In this section we present the results on our survey on the state of the art on GPS applications, focusing on the most used applications with features similar to those that we wanted to implement, summarizing what can and cannot be done with the existing software and identify its limitations. To do so, we considered four categories: a)search by criteria, b)search by area, c)search by sport and d)calendar view. The applications were selected due to their functionality, popularity and availability and were the following: GPSies[7], Strava[8], MyTourBook[9], Wikiloc[10] and SportsTracker[11].

In our survey we concluded that search by criteria is already common to applications we surveyed, although a lot more complete in the applications that focus on storing and sharing activities and courses like GPSies or Wikiloc. From the applications studied, the only one that didn't contain this functionality is Sports Tracker, due to the fact that its focus is more on the analysis of athletes performance.

Search by area is a functionality that is also really standard in the applications we studied, with only MyTourBook not supporting it due to the fact that is an application focused on planning and editing activities and courses.

Of the applications studied none of them contained the feature of search by sport. In the case of MyTourBook it's normal since it only supports cycling. In the case of the other applications, it seems they tried to create a generalized search for every sport, although in the process some of these applications left behind some important criteria for some sports. In spite of this, we decided to implement a search by sport, since TrackIt! supports a larger number of sports and already contain functionalities that operate differently at times depending on the selected sport.

Calendar view is a functionality that is present in all the applications focused on social networking. Applications to store and share activities and courses don't provide this kind of information.

3. Architecture

In this section we present the prototype TrackIt! before this work.

TrackIt! is really complete in terms of functionalities, due to the fact that this is the fourth iteration of this application. Although, its architecture is something that has remained constant since the first iteration.

Its architecture is modular, and it's divided between a front-end module and a back-end module.

The front-end module manages the TrackIt! interfaces and views while the back-end module is responsible by all the algorithms necessary for its operation and the database. This provides a separation between the two modules, creating a clear separation between interfaces and data processing.

3.1. Front-End

As stated previously, the front-end module is responsible by all TrackIt! interfaces and views. The most important views are: 1)Document View, 2)Map View, 3)Chart View, 4)Summary View, 5)Data View and 6)Message View.

Document View is the view responsible for presenting the user with its activities and courses. It's hierarchical and its organized by folders. Its two major folders are Workspace and Library.

The Map View, the most important view in TrackIt!, displays the map and therefore activities, courses and waypoints and is where all course edition takes place. This view supports several digital map providers like GoogleMaps[1], OpenStreetsMaps[2] and BingMaps[3] to name a few.

Chart View is the view that presents activities and courses information in the form of graphics, relating the several data of an activity or a course.

The Summary View presents all the information about a TrackIt! course or activity.

Data View displays a table with all data about the selected TrackIt! course or activity.

And, last but not least, Message View displays messages about the operation of TrackIt! (errors, operation results and event processing).

3.2. Back-End

As said before, the back-end is responsible by the operation of TrackIt! and all its algorithms. It contains several algorithms for many operations like import and export of GPS files in various formats. It also supports adding media to an activity and possesses an algorithm that suggests media geographic location.

[1]GoogleMaps: https://maps.google.pt/ seen in 3/11/17

[2]OpenStreetMaps: https://www.openstreetmap.org/ seen in 3/11/17

[3]BingMaps: https://www.bing.com/maps seen in 3/11/17

Since the main goal of TrackIt! is to plan outdoor activities, it has algorithms to create and edit courses and it contains also operations like splitting, joining and inversion of courses.

It has also useful algorithms like identification of climbs and descents, setting of pace, pause detection, consolidation, course simplification and segmentation. It also supports undo and redo of the operations previously mentioned.

As mentioned before, the back-end also controls a SQLite[4] database. This database previously contained three tables important for this work, GPSFiles, Sport and SubSport.

The GPSFiles table is the table that has information about the activities and courses imported to TrackIt!. Before this work it only consisted of eight columns: Filepath, Name, Sport, SubSport, MinLongitude, MaxLongitude, MinLatitude and MaxLatitude. Because of this, it was only possible to search activities by area, creating a bounding box using the minimum and maximum longitude and latitude.[12]

Naturally, the sport and subsport tables store sports and subsports data.

TrackIt! already contained an area search, but, unfortunately this search had some problems. The biggest problem with this search was that it would import the whole document if one of the activities or courses were in the selected location. This of course is not its desired functionality, since it imports file that are not in the selected location.

3.3. Solution

We started by approaching the support to implement the desired functionalities, the search and the calendar view.

First was the search. To implement a search, the most important thing is to satisfy the user, and provide criteria that they need the most when performing a search. With that in mind, we started by inquiring users using an online form on the Google Forms[5] platform to identify the most important criteria to users. We inquired around 180 persons. From the survey, we found that by order of preference the four most voted criteria was the following: 1)total distance, 2)difficulty, 3)location and 4)sport.

Therefore, we decided to create four different types of search: a simple search, an advanced search, a search by sport and an area search. In the following we explain how.

The simple search was supposed to consider only the most important criteria for the most users. With this approach we wanted to create a fast search for situations when the user doesn't need many of criteria.

Then there is the advanced search. The advanced search was created to have all criteria used in the simple search plus criteria that wasn't used previously. With this we wanted to create a more complex search, that would contain the additional criteria like temporal, total descent and ascent and state of the course.

Next the search by sport. Search by sport intended to be a search where the criteria presented depend on the selected sport. With this we wanted to create a specific search for every sport supported by TrackIt!.

The last search is the area search. As we said earlier, this search was already implemented on TrackIt!, but had some erratic problems since it imported a whole file when only one activity or course was in the selected area. This problem wasn't of the search itself, but of TrackIt! since it didn't contain a selective activity or course import method. To change this we had to create a method that would import only the selected activities or courses from any document.

To present the search result we selected Document View, namely the Collection folder.

To create the search interface we use the API used in the rest of TrackIt!, Swing.

The Calendar View is a different way to search for activities using a view known by everyone. To make it easy for a user to search an activity or course by navigating through years and months. Normally, days that contained any activity or course would be highlighted, signaling that that day contained an activity or course.

To implement this new functionalities and to make it possible to search for different criteria we would have to expand the existing database. With this in mind, new columns were added to the GPSFiles table. Those columns are the following:

- Course difficulty
- Total Course Distance
- Course state
- Total Ascent/Descent
- Starting/Ending Time
- Course Type (Circular, Not circular)

We also wanted to add support for new sports and subsports. The sports and subsports we wanted to add are presented in table 1:

4. Implementation

In the previous section we presented our solution. In this section we will present how those solutions were implemented. We will start with the tools used, and then present how we implemented the

[4]SQLite: https://www.sqlite.org/, seen in 3/11/17
[5]https://docs.google.com/forms/, seen in 3/11/17

Sports	Subsports
Driving	ATV, Leisure, Track
Golf	Challenge, Leisure
Ciclismo	Mountain, Road, Track
Kaiake	Open Water, Whitewater
Motorcycling	Motocross, ATV, Track
Swimming	Lap Swimming, Open Water
Snowshoeing	Backcountry, Trail
Caminhada	Casual Walking, Road, Speed Walking, Trail
Sailing	Challenge, Leisure

Table 1: Sports and Subsports that were added on TrackIt!

solutions, starting with the modifications made to the database, followed by the creation of the search interface and then concluding with the creation of the calendar view.

4.1. Tools

The new functionalities were implemented using Oracle Standard Edition of the Java Platform, version 8[6]. For the development of this work we used the IDE Eclipse[7], version Mars.2, launch 4.5.2, available for the operating system Windows 10 64 bits. As mentioned earlier, we created the interfaces using the Java GUI, Swing and some third party libraries, to build some of the interfaces: RangedSlider and JCalendar.

We also used Apache Maven[8], to build the application, and to manage dependencies and libraries.

As mentioned previously we used the database management system SQLite.

4.2. Solutions

We start with the modifications made to the database. Due to the fact that this work focuses on search, the database was modificated in a major way.

The principal modification was the GPSFiles table. We increased the number of columns in the table, so that we could search using a big number of criteria. The criteria added are presented in table 2. In the following paragraph we will explain each of the columns that were added.

The StartTime and EndTime columns represent the dates and times when an activity or course started and ended, respectively. TotalTime is the total time of an activity or course including pauses while MovingTime is the total time excluding pauses. Distance is naturally the total distance of an activity or course. Ascent and Descent are the total ascent and descent respectively. Maximum altitude and minimum altitude are presented in the

GPSFiles
StartTime
EndTime
TotalTime
MovingTime
Distance
Ascent
Descent
MaxAltitude
MinAltitude
TrackState
TrackDifficulty
CircularPath

Table 2: Columns in GPSFiles table

Condition
Unknown
Unkept
Bad
Acceptable
Good
Very Good

Difficulty
Unknown
Easy
Moderate
Difficult
Very Difficult
Experts Only

Table 3: New Values of Difficulty and Condition

MaxAltitude and MinAltitude columns. Column TrackState is the state of the course while TrackDifficulty its difficulty. Lastly, we have the CircularPath column that signals if the course is circular or not.

From all criteria added above, almost all of them where already present internally in TrackIt!, except Condition and Difficulty. To do this we created two new classes that contained the values presented in tables in 3.

In previous versions of TrackIt!, it was only possible to add sport and subsport to an activity or course. In this version, with the addition of these two new parameters that have to be inserted by the user, we created a new dialog. With this new dialog, the user can now add the difficulty and condition of an activity or course, and also add sport and subsport as before.

We also added a new way of updating the database version. This is possible due to a new table, the DBVersion. This table contains the database version number, its creation date, and the sport catalog date. If the database version is not the most recent, TrackIt! will run a method that updates the database.

Last but not least, we also improved TrackIt! persistence. In previous versions, persistence was only made using table GPSFiles, that only saved references to documents that previously had been registered. With the addition of a new table named

[6]Java SE 8: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html seen in 5/11/17

[7]Eclipse: https://www.eclipse.org/ seen in 5/11/17

[8]Maven: https://maven.apache.org/ seen in 5/11/17

Desktop, we were able to create persistence in the Document View. The columns of table Desktop are presented on table 4. This table saves documents present in Document View when TrackIt! is closed so that when TrackIt! is started again, those document are restored to Document View. To do so, this table saves basic information to locate the files to be imported and other information like activity or course name, sport and subsport and its type (activity or course). There are also two columns: inWorkspace and Loaded. The inWorkspace column determines the location of the document in Document View, Workspace or Collection while the Loaded column determines what files are actually going to be read at TrackIt! start.

Desktop
DocumentName
Filepath
TrackName
isActivity
Sport
SubSport
inWorkspace
Loaded

Table 4: New table Desktop and respective columns

Because we added the new criteria to the database, now we needed to create the queries. But this creates a problem, since the number of criteria that we will receive may vary. For this reason, we wanted to create a single query with all criteria that is selected by the user. This way, we only had to query the database through one query, improving efficiency.

To achieve this, we started by using a characteristic from Java Strings, that of being able to handle the null value. We created a new class, SearchObject, that contains all search criteria in the form of Strings.

Then we created a new class named DBSearch that receives the new class SearchObject objects. DBSearch detects what parameters of SearchObject are not null and creates a query that is then sent to the database that returns the search results.

To present the results, we wanted to use the Document View, more properly the folder Collection. But this presented the problem that we mentioned earlier, the problem that TrackIt! didn't had the ability to import only a specific activity or course, and was only able to import full documents. So, we created a new method that receives the file name and the activities and courses to import and imports only those mentioned.

The search interface was created using the Swing Java library. For this interface we used as a layout the GridBagLayout[9] due to its flexibility and complexity. This layout uses a grid to locate components on the interface.

The search interface is available in three different ways: from the menu Edit selecting Search, from the right mouse button in the folder Collection in Document View and selecting Search and, last but not least, using the Search by Area icon. After a successful search it is possible to Refine Search. We achieved this using the Singleton design pattern. With this pattern there is only one instance of the interface, and in the succeeding searches we only have two choices: New Search and Refine Search. When the user requests a new search, the interface only has to be shown and a reset method is executed, returning the interface to its original state. When there's the need to refine a previous search, the interface only has to be shown, since the interface contains the values that were used previously.

Since we wanted to create three kinds of search, we used a JTabbedPane[10] to build tabs for each one of the search types: Simple, Advanced and By Sport. The interface was created with the necessary size to contain all the components of the biggest search. It also contains three buttons panel, Search, Cancel and Reset. The Search button will send the selected criteria to the search algorithm, while the Cancel button will discard the dialogue. The Reset button will make the interface return to its initial state.

Another functionality that we wanted to provide in our interface was the persistency between simple and advanced search. This means that if a criteria is selected in the simple search it would be propagated to the advanced search, and vice versa.

To create the interface we also used many different components. We started by using JLabels[11] to create a little description of each criteria. We then used the following components to build the interface: JTextField[12], RangedSlider[13], JComboBox[14] and JDayChooser[15].

JTextFields are used for text inserting purposes, to insert Location, Range and Coordinates values. Coordinates can be inserted in three different ways which we will now explain.

[9] GridBagLayout: https://docs.oracle.com/javase/7/docs/api/java/awt/GridBagLayout.html seen in 6/11/17

[10] JTabbedPane: https://docs.oracle.com/javase/7/docs/api/javax/swing/JTabbedPane.html seen in 6/11/17

[11] JLabel: https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html seen in 6/11/17

[12] JTextField: https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html seen in 6/11/17

[13] RangedSlider: https://github.com/ernieyu/Swing-range-slider seen in 6/11/17

[14] JComboBoxes: https://docs.oracle.com/javase/8/docs/api/javax/swing/JComboBox.html seen in 6/11/17

[15] JDayChooser: https://github.com/empeeoh/JCalendar/blob/master/src/com/toedter/calendar/JDayChooser.java seen in 6/11/17

$$newLatitude = latitude \pm \frac{radius}{rEarth} \times \frac{180}{\pi}$$

$$newLongitude = longitude \pm \frac{radius}{rEarth} \times \frac{\frac{180}{pi}}{\cos(latitude \times \frac{\pi}{180})}$$

Figure 1: Expressions to add Area to Coordinates (rEarth = Earth Radius)

The first consists of using the Location and Range JTextFields. This provides the user with a geocoding functionality using Google Maps Geolocation API that returns the coordinates of the inserted Location. But, depending on the value inserted in the range JTextField this functionality will work in two different ways. First, if the range is zero, it will return the bounding box of the inserted location. Oh the other hand, if the value inserted in range is different than zero, the API will only return the inserted location center coordinates and then add the range value using the following expressions presented in 1, creating a new bounding box.

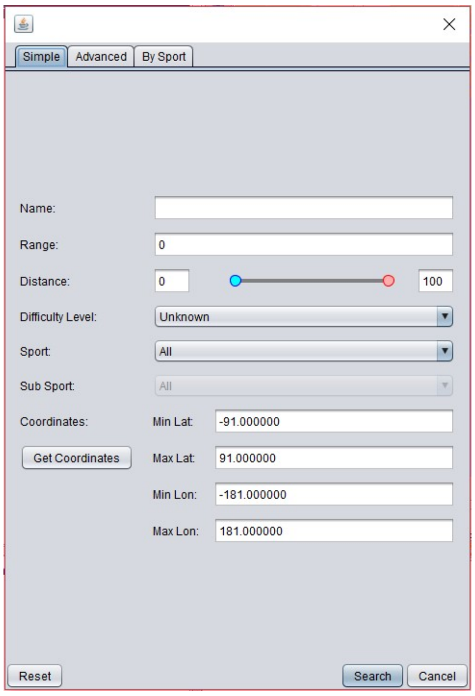
The other way of obtaining coordinates is using the Get Coordinates button in the interface that works in a similar way to Area Search. When this button is pressed, the interface disappears and the user can select the desired area in the Map View. After this, the interface reappears and the selected area coordinates are shown. The other way is using the area search icon that operates in a similar way except that the interface only appears after the area is selected.

We also used a component named RangedSlider. This component is a third party library, due to fact that the basic Java JSlider doesn't allow the selection of intervals. But because this slider doesn't show the selected values, we added two JTextFields. These JTextFields can also change the default minimum and maximum values of the slider and be used to select the desired values. This components were used to select Distance, Ascent and Descent.

The next component we used was JComboBox that provides a drop down menu for the selection of criteria. This component is used for the following criteria: Difficulty Level, Sport, SubSport, Course Condition and Circuit Type.

The last component used was a JDayChooser that was also obtained online in a third party library named JCalendar[16]. We decided to use this library because it contains several components that were used in this work. The JDayChooser component is used to select start and end dates, and provides an icon that opens a calendar to select the desired dates. If both dates are the same, the search will be performed only in that selected day.

[16]JCalendar: https://toedter.com/jcalendar/ seen in 6/11/17



Name:
Range: 0
Distance: 0 100
Difficulty Level: Unknown
Sport: All
Sub Sport: All
Coordinates: Min Lat: -91.000000
Max Lat: 91.000000
Min Lon: -181.000000
Max Lon: 181.000000

Figure 2: Simple Search Interface

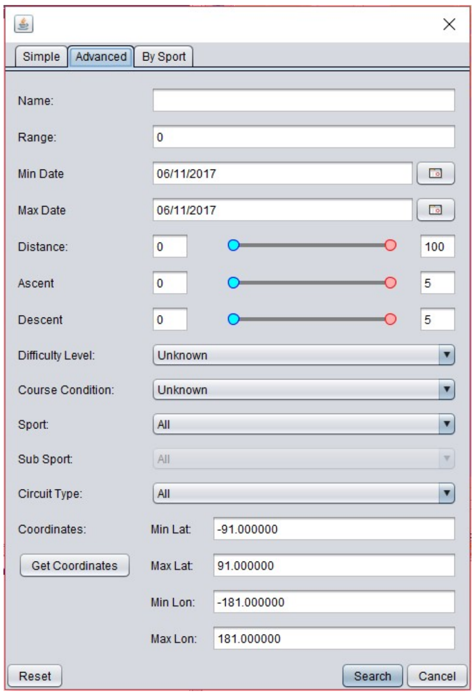
Lastly, we created a search by sport, a search where the criteria presented depends on the selected sport and subsport. To achieve this, we grouped sports that had similar characteristics. We created the four different groups show on table 5, that presents the criteria for each group. After creating each group, we used a CardLayout[17] to create one card for each group, and inserted the layout in the bottom part of the interface. This layout allows to present different cards and criteria depending on the sport and subsport selected.

The three search interfaces are presented in figures 2, 3 and 4.

Types of Sport	Criteria
Sports that ascents and descents are important	Ascent, Descent, Condition and Difficulty
Sports that only descents are important	Descent, Difficulty and Condition
Sports practiced in flat surfaces	Condition
Aquatic Sports	Difficulty

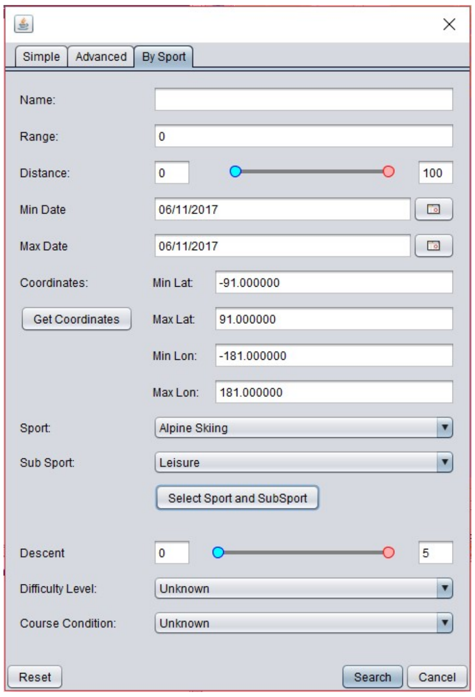
Table 5: Different criteria for each group of sports

[17]CardLayout: https://docs.oracle.com/javase/7/docs/api/java/awt/CardLayout.html seen in 6/11/17



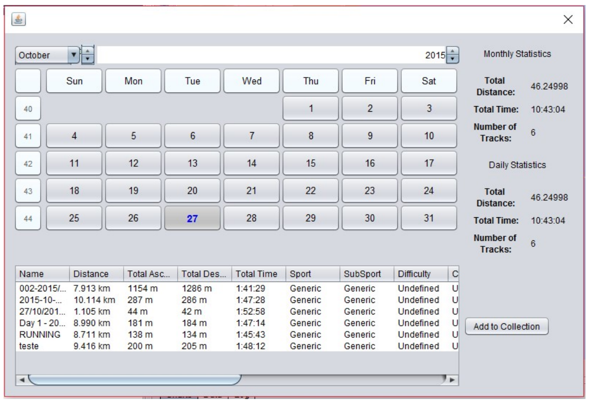
Name:		
Range:	0	
Min Date	06/11/2017	
Max Date	06/11/2017	
Distance:	0	100
Ascent	0	5
Descent	0	5
Difficulty Level:	Unknown	
Course Condition:	Unknown	
Sport:	All	
Sub Sport:	All	
Circuit Type:	All	
Coordinates:	Min Lat:	-91.000000
	Max Lat:	91.000000
	Min Lon:	-181.000000
	Max Lon:	181.000000

Figure 3: Advanced Search Interface



Name:		
Range:	0	
Distance:	0	100
Min Date	06/11/2017	
Max Date	06/11/2017	
Coordinates:	Min Lat:	-91.000000
	Max Lat:	91.000000
	Min Lon:	-181.000000
	Max Lon:	181.000000
Sport:	Alpine Skiing	
Sub Sport:	Leisure	
Descent	0	5
Difficulty Level:	Unknown	
Course Condition:	Unknown	

Figure 4: By Sport Search Interface, with the specific Ski criteria



	Sun	Mon	Tue	Wed	Thu	Fri	Sat
40					1	2	3
41	4	5	6	7	8	9	10
42	11	12	13	14	15	16	17
43	18	19	20	21	22	23	24
44	25	26	27	28	29	30	31

Monthly Statistics

Total Distance:	46.24998
Total Time:	10:43:04
Number of Tracks:	6

Daily Statistics

Total Distance:	46.24998
Total Time:	10:43:04
Number of Tracks:	6

Name	Distance	Total Asc...	Total Des...	Total Time	Sport	SubSport	Difficulty	C
002-2015/...	7.913 km	1154 m	1286 m	1:41:29	Generic	Generic	Undefined	U
2015-10-...	10.114 km	287 m	286 m	1:47:28	Generic	Generic	Undefined	U
27/10/201...	1.105 km	44 m	42 m	1:52:58	Generic	Generic	Undefined	U
Day 1 - 20...	8.990 km	181 m	184 m	1:47:14	Generic	Generic	Undefined	U
RUNNING	8.711 km	138 m	134 m	1:45:43	Generic	Generic	Undefined	U
teste	9.416 km	200 m	205 m	1:48:12	Generic	Generic	Undefined	U

Figure 5: Calendar Interface

Last but not least, is the Calendar View that was created using the Java library Swing and the layout MigLayout[18]. We decided to use this layout due to the fact that this interface contains a reduced number of components and because of its versatility and its ease of use.

Our principal focus on this interface was the calendar component. For this we used the library previously mentioned, JCalendar, that contained a calendar component that was perfect for our use. The JCalendar also contains two JComboBoxes to select month and year, and that update the calendar to the selected year and month. Another characteristic of the JCalendar is the ability to support different locales. This is a good characteristic since TrackIt! currently supports english and portuguese, using different locales that present different starting week days.

The functionality of this interface is fairly simple: when the interface is initiated it presents the current month and year. The user can select the desired month and year and, if a day contains an activity or course that day will be highlighted in blue. If a day with an activity or course is selected, the activity and courses of that day are presented in the table bellow the calendar. The user can then select the activities and courses that he wants to import to Collection, and once he does, he just needs to press the button Add To Collection to add the chosen activities and courses to the Collection folder.

We also wanted to provide users with informations about the selected month and day. With this end in mind, we created two JLabels that show total distance, total time and number of courses or activities for the selected month and day.

In figure 5, we present the Calendar View.

[18]MigLayout: http://www.miglayout.com/ seen in 6/11/17

5. Results

Next we address how we evaluated our solution. We started by creating tests to determine if users can use the new functionalities.

We created four tests, one for each type of search interface and one for the calendar view to see if the new interfaces are intuitive and easy to use. Each test included some derived interfaces functionalities like the geolocation feature, the propagation between simple and advanced search, the refinement feature, getting coordinates by selecting an area feature and saving documents to the database.

The tests were evaluated by five users with different levels of experience with this type of applications. Two of them were beginners, in other words, they never had used any of these applications previously. Two of them had an intermediary level of experience, since they rarely use this kind of applications. The last user was an expert user that uses this kind of applications on a weekly basis.

The tests were performed on a PC, with an Intel i7-6700HQ processor, 16GB of RAM DDR4 with 2400MHz and a graphics card NVidia GeForce 970. The operating system was Windows 10 with 1.8.0_73 version of Java installed.

Before performing the tests users were briefly briefed about TrackIt!, and were told that if they had any questions they were welcome to ask them.

5.1. Conclusions

The principal objective of the test was to observe how users would interact with the new functionalities and to see if they could complete all the tasks.

Although the users felt some problems, they all completed the tasks at hand. The problems encountered by the users are presented in the next.

The first problem was due to the location of the functionalities in the menus. Users thought that the Calendar View and search interface located in the Edit menu was not very intuitive. To solve this problem, a new menu group with a more appropriate name should be added to the menus. But, since the Calendar View and the search interface are located in the same menu, after the first test, they didn't felt that problem since they already knew where those functionalities were located. All the users used the interfaces created with ease in the remaining tests.

Another problem was saving files to the database. Users couldn't find how to save documents. Although it's possible to save documents when TrackIt! closes, at runtime this is only possible through the Document View. This could be changed with a way to save documents through the File menu, where it would be possible to save the current document or select one from the current list of documents in the Document View.

In addition, one more problem was experienced only by the less experienced user while performing an advanced search. This user didn't use the calendar component to select a date and chose to insert all the dates by hand. To show the users that this feature is included, we could increase the size of the icon that provides this functionality.

Last but not least, all users, except the more experienced, didn't use the functionality to obtain coordinates by selecting an area in the Map View. However they used the new geolocation functionality, showing that this functionality is really valuable for less experienced users, in providing coordinates in a fast and easy to use way.

6. Concluding Notes

In this work we implemented functionalities lacking in TrackIt!, an application focused on amateur and professional athletes that use GPS devices to record their performance. The application also provides support for planning outdoor activities.

The number of this kind of applications has been increasing in the last years due to the evolution of GPS devices and its consequential decrease in price and size and also due to the inclusion of these devices in mobile devices like smart phones and smart watches.

Due to the fact that this is the fourth major revision of TrackIt!, the application already possesses a great number of functionalities useful for the user. It also supports a large number of digital map providers and tools for planning and editing of activities and courses.

The functionalities implemented in this work were:

- Introduction of new sports and subsports
- Expansion of the database to support new criteria
- Persistence between sessions in the Document View
- Search Algorithm
- Search Interface
- Calendar View search and interface

By introducing new sports, we wanted to improve the number of supported sports and subsports and also enlarge the number of sports that could be searched.

The database went through a lot of changes during this work. We started by inserting new criteria in the database so we could use them to perform a search. We also created a new table to provide persistence in the Document View. With this table the Document View documents are saved when

TrackIt! is closed, and reloaded when TrackIt! is started again.

To implement the search of activities and courses we had to create an algorithm and a search interface.

We started by creating a search algorithm to create a custom query for the desired search. With this approach, the query only contains the criteria selected by the user.

We created a search interface using the Java API Swing. Criteria presented in each interface were chosen by the users using an online survey that inquired the most important criteria for them. With the results of this survey we decided to create three types of search: simple, advanced and by sport. The simple search provides a fast way to search for activities and courses with only the criteria that users found more relevant. Advanced search contains criteria present in the simple search plus the remaining criteria that are less often used. Lastly, the search by sport, provides only specific criteria for the selected sports. Due to the fact that TrackIt! supports a great number of sports and subsports, we though that this kind of search would be beneficial for the user.

We also implemented a calendar view search. This view was implemented using Java API Swing and the third party library JCalendar. The calendar view presents activities and courses for selected month and year, making it possible to import a specific activity or course to the Document View. It also presents information about the selected month and day.

In addition, we also improved some of TrackIt! functionalities, namely the area search by creating a new method that only imports selected activities or courses from a document, something that wasn't possible before this work.

The solutions implemented were submitted to usability tests with five users so that we could find problems with the interface. The problems found with the application derived from the menu location of the functionalities, since users didn't find their location intuitive. We also found that users with less experience have more difficulty performing the tasks at first times.

6.1. Future Work

Although the functionalities implemented in this work function correctly, there's still ways to improve the application and add new functionalities. These are some suggestions for future work:

- New item in group File that saves the current document, maybe giving the chance to select the whole document or a specific activity or course
- Add points of interest, thereby making possible to mark using the Map View locations like monuments for example
- Statistical adjustment of sport and subsport values using past activities
- User profiles, that depending on the selected profile, change sport and subsport values
- Addition of filters in the calendar view that present only activities or courses that correspond to that filter. For example, selecting a sport as a filter so that the calendar only presents activities or courses of that selected sport
- Connecting to social networks providing ways to share any activity or course. It would also become possible to challenge new users

Acknowledgements
I would like to thank my family and friends for their support, your support helped me in a major way during this work. I would also like to thank Professor João Brisson for all his help and advice.

References

[1] J M Brisson Lopes. Visualising and Processing GPS Enhanced Outdoor Activities Data. 2016.

[2] Sameer Kumar and Kevin B Moore. The Evolution of Global Positioning System (GPS) Technology. II(1), 2002.

[3] Julen Castellano and David Casamichana. Deporte con dispositivos de posicionamiento global (GPS): Aplicaciones y limitaciones. *Revista de Psicologia del Deporte*, 23(2):355–364, 2014.

[4] Henrique Guimarães Malheiro. Track it! (May), 2014.

[5] Pedro André Dominguez Alves Gomes. Track It (Again). 2015.

[6] Miguel Alexandre Sequeira Pernas. TrackIt (Better!). 2016.

[7] GPSies. GPSies. 2006. http://www.gpsies.com, [Online; accessed 22-September-2017].

[8] Strava Inc. Strava. 2009. http://www.strava.com, [Online; accessed 22-September-2017].

[9] MyTourbook Contributors. MyTourBook. 2007. http://mytourbook.sourceforge.net/mytourbook/, [Online; accessed 22-September-2017].

[10] Wikiloc. Wikiloc. 2006. https://www.wikiloc.com/, [Online; accessed 22-September-20]17.

[11] Sports Tracker. Sports Tracker. 2004. http://www.sports-tracker.com/, [Online; accessed 22-September-2017].

[12] Douglas R. Caldwell. Unlocking the Mysteries of the Bounding Box. 2005.