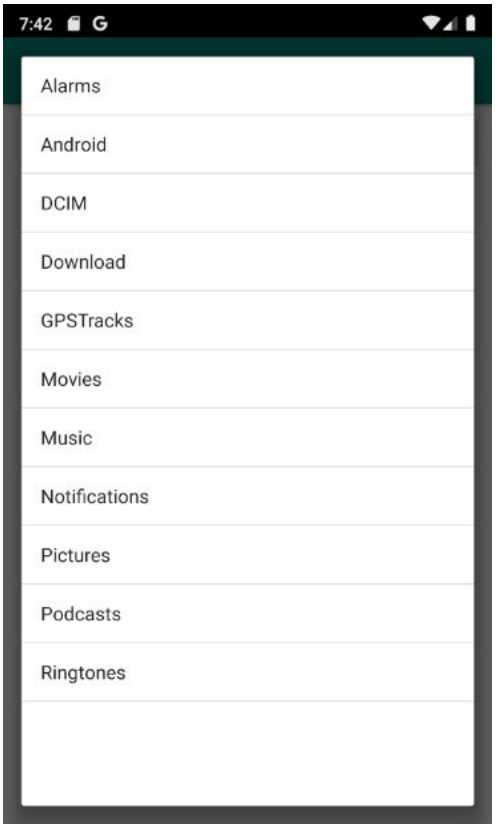
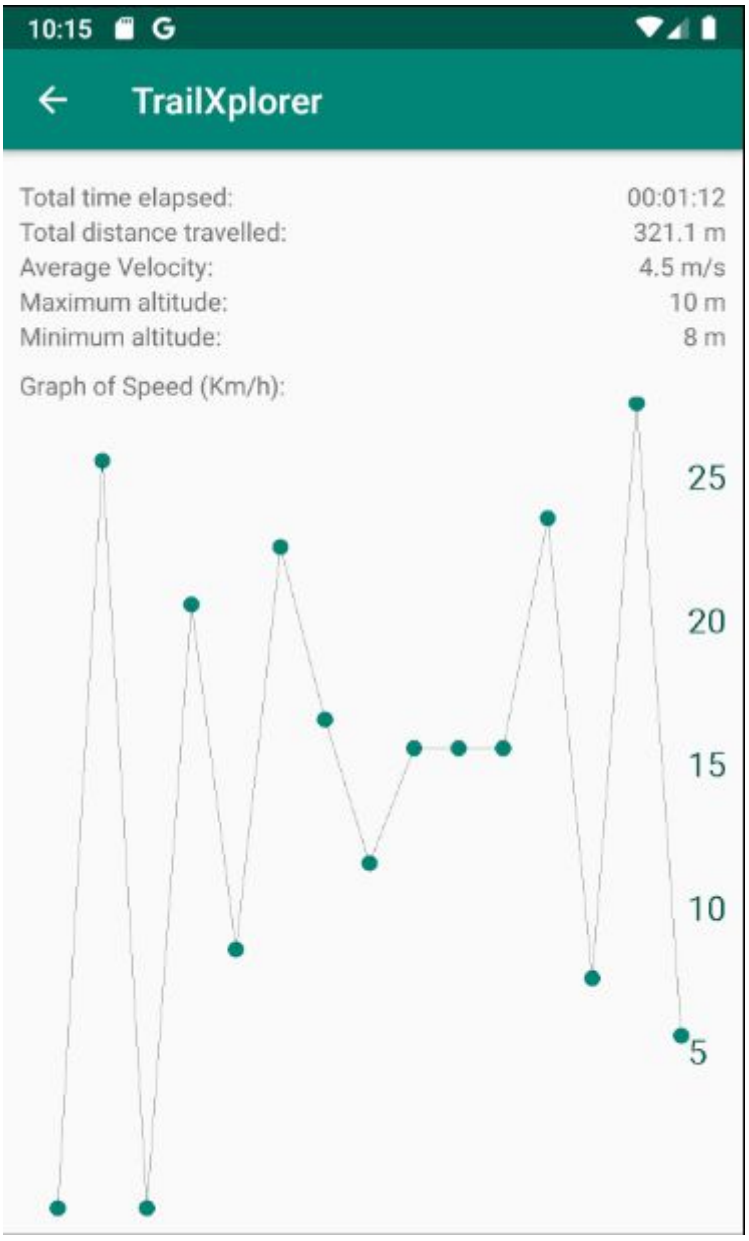
Documentation - TrailXplorer

1. The User Interface



This is the first activity of the application. From this point the user has the possibility to start tracking his gps coordinates, as well as importing a gpx file to be analyzed.
The “start tracking” button is in the center of the activity, given that it is the main feature of the application and should be the first one to be seen.
There is also the inclusion of a timer, that activates upon pressing the start tracking button, to indicate to the user that the app has not frozen, and the time since the beginning of recording.
Now, pressing the import GPX file button will trigger a dialog for the user to select a file, it is our second image here.
The file choosing dialog is very simple and easy to use.

Importing a file or ending the tracking via the app will take the user to a new activity. This activity is used in showcasing the results given by the application.



That is the second activity, which is used to showcase the analysis of the tracking, as well as the analysis of a GPX file that would have been provided.
There are two parts to this activity.
Up top are the statistics, and below is a graph of the speed of the user at each point that was recorded.
This approach is minimalist, the screen is not too cluttered, and the information presented will be easily understandable.

2. The Code

   Now let us inspect one by one the different methods that are in use in the application.

   a. MainActivity.java

`protected void onCreate`
This method will initialize the different views of the activity, and will also create the directory where GPX files will be stored.
`private void selectFile()`
This method is used to display the file chooser and then access the stats.
It is called by the Import button.
`private void switchToResults()`
This method is used when the application is done getting location data. It will shutdown the location manager and finish the writing of the GPX file and then switch to the results activity.
`private void startRecording()`
This method is used to start the recording of gps coordinates. It will start the requesting of coordinates and change the text on the button.
`private boolean checkLocationPermission()`
This method is used to check if the device allows the application to use lcoation data, if not it will prompt the user to enable it.
`private void initializeLocationListener(`
This method is used to initialize the service that will provide locations. It also defines what the app will do when a location is received.
`private void createGpxFile()`
This method will create a new GPX file in the directory, and name it according to the current date.
It will also start the writing to the file.
`public File initializeGpsDirectory()`
This method is called by OnCreate(), and will create the directory where GPX files will be stored if it is not present already.
`private boolean isExternalStorageWritable()`
This method is used to check if the app can access external storage. It is called before every write or read.

b. arraygpx.java

This class is static, and contains the information about the current series of location coordinates, in a form of an array, and accessible by other classes of the app.

`public static void addlocationArray(Location location)`
This method adds a Location object to the array.
`public static void setlocationArray(List<Location> val)`
This method is a setter for the array.
`public static List<Location> getlocationArray()`
This method is a getter for the array.

c. results.java

This class is used in combination with the results activity, it contains the methods that are used to create the statistics.

`public void onBackPressed()`

This method overrides the back functionality, so that it loads back the first activity. It is there as a fix to a bug.

`private Float getAverageSpeed()`

This method returns the average speed of the series of locations.

`private double getTimeTaken()`

This method returns the total time of the tracking.

`private float getTotalDistance()`

This method returns the total distance travelled during the tracking.

`private void getAltitudes()`

This method updates both the MinAltitude and MaxAltitude variables.

d. gpxWriter.java

This class is used to write to the GPX file during tracking.

`public void startWriting()`

This method will start the writing of a new GPX file. It will write all the mandatory tags and prepare the file to receive location info.

`public void writeNewLocation(Location location)`

This method will write a new <trkpt> entry to the file, using the data in the Location object.

`public void finishWriting()`

This method will close all the tags and finish the file.

`public static String toPrettyString(String xml)`

This method is used to have a better formatting of the GPX file.

e. gpxParser.java

This class is used by the import function, it generate a new array of location when fed a GPX file.

`public static void parse(String path)`

The method will generate the array of locations, using the path to a GPX file as an input. It uses the DocumentBuilder class.

e. graphView.java

This class is used to generate the graph that is displaying the speed on the results screen.

`private void init()`

This method will initialize the variables needed to draw the graph, such as paints.

`protected void onDraw(Canvas canvas)`

This is the OnDraw method, it draws the graph.

`private void generateArrays()`

This method will generate the array containing the speed at each position and the one containing the elapsed time at each position.

`private void calcPositions()`

This method, from the array, calculate the position of the points that have to be displayed. It takes into account the size of the screen, so that it scales intelligently.

`private void drawGraduations(Canvas canvas)`

This method will dynamically draw the graduations on the side of the graph.
It adapts to the size of the screen.

`private void drawLine(Canvas canvas)`

This method will draw the line of the graph, as well as each of the points.

f. FileChooser.java

This class is used to display the file choosing dialog, when importing a file.

`FileChooser setFileListener(FileSelectedListener fileListener)`

This is a setter for the filelistener, which states what to do with the file once found.

`FileChooser(Activity activity)`

This is the main method that generates the dialog.

`void showDialog()`

This is the method to show the dialog.

`private void refresh(File path)`

THis method is used to sort, filter and display the files in the corresponding path.