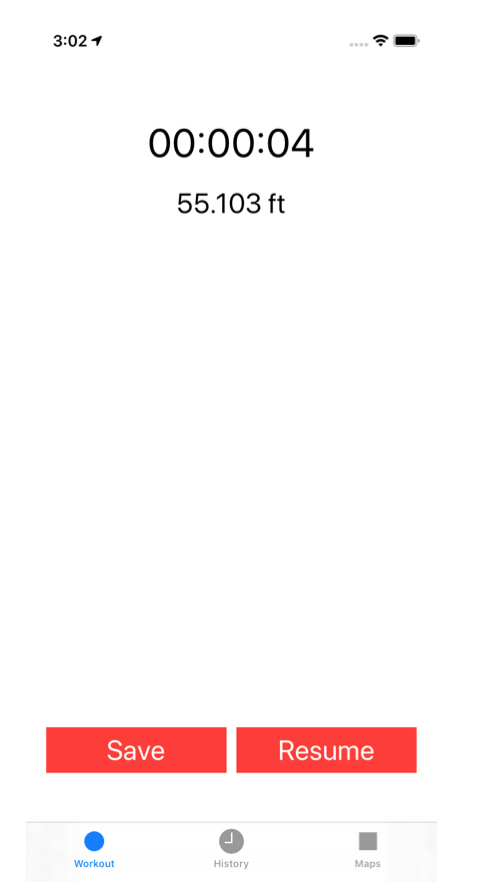
When a user hits the start button, the session (or workout) view will be the following:



To manage a timer, we use the **Timer API**. For the distance covered, we will use the CoreLocation API. These two APIs are well documented and numerous tutorials exists on the web.

When the user hits Resume the session must... resume.

When the user hits Save, the session must be saved (on disk and Healthkit) but this will be the purpose of the next part of the project.

**Saving a session on disk**

To save a session on disk (and later in HealthKit) it would be interesting to model it. For this purpose We could use a struct with 5 fields: startTime (Date), endTime (Date), duration (TimeInterval), locations ([Coordinate]) and distance (Double). The Coordinate type can also be a struct with two fields: latitude and longitude both of Double type.

The saving part will consists in saving the session object in the form of a plist file. To do this We can use the PropertyListEncoder (PropertyListDecoder for loading) class. This one will allow to encode a swift object into a plist file which will be saved in the documentDirectory of the FileManager. To use the decoder/encoder classes, Wer object must implement the Codable protocol: this is the equivalent of the Serialization concept in Java.

We should be careful to not overload the main ViewController.

**Displaying the last session on a map**

Since the session is saved on disk and contains an array of the different locations of the run, We can display its path on map.  
To do this, we use the MKAnnotation class (allowing to dispaly a pin on a map).  
To display a path, We can use the MKPolyline and MKOverlay classes.