Tracking sailing regattas with mobile devices

T2_2000 Project Thesis

for the

Bachelor of Science

at Course of Studies Applied Computer Science

at the Cooperative State University Karlsruhe

by

Jan Broß

September 2013

Time of Project	13 Weeks
Student ID, Course	4476648, KA-TINF11B2
Company	SAP AG, Walldorf
Supervisor in the Company	Dr. Axel Uhl
Reviewer	Heinrich Braun

Author's declaration

Unless otherwise indicated in the text or references, or acknowledged above, this thesis is entirely the product of my own scholarly work. This thesis has not been submitted either in whole or part, for a degree at this or any other university or institution. This is to certify that the printed version is equivalent to the submitted electronic one.

Walldorf, September 2013

_____

Jan Broß

Abstract

In the last years SAP Sailing Analytics transformed the way a sailing regatta is experienced by spectators and sailors alike. SAP Sailing Analytics helps to create a better understanding of this traditional sport by the usage of modern technology. Meaningful metrics and visualizations calculated on the basis of wind and Global Positioning System (GPS) tracking data enable the sailor to retroactively understand what makes a boat win and facilitate the observation of a sailing race for a spectator ashore.

So far SAP Sailing Analytics relied on tracking providers that deliver the GPS positions of the sailors as well as the positions of the buoys marking the course. To make SAP Sailing Analytics available to a wider audience smartphones could be used as a supplier of tracking data. This thesis deals with the requirements for a mobile application used for tracking sailing races and the development of a first prototypical implementation.

Contents

1 Introduction

> "SAP Sailing Analytics not only helps make the sport more accessible for the fans. It also helps the sailors optimize their strategies, as the teams and their coaches can use the information to evaluate their performance better than ever before." – *Marcus Baur, three-time European 49er Champion, June 2011*

For a long time, it was hard for spectators and team coaches alike to follow the happenings on the water during a sailing regatta. SAP helps in the context of its sportsponsoring program to make this sport more comprehensible.

Sportsponsoring enables SAP to present itself to a wide audience. Since 2011 SAP has a new concept for sponsoring, which includes that not only the company's name is presented, but also the technology developed by it. Sponsoring is therefore a technology showcase, where SAP is able to show its potential.

1.1 SAP Sailing Analytics

SAP Sailing Analytics is a web application which transform the GPS tracking data, so far delivered by partners, into meaningful metrics and visualizations.

This helps spectators to see what is happening on water almost immediately and also gives crucial information to commentators. Additionally it is also helpful for sailors to retroactively analyze the race and to understand what went wrong and where to improve.

1.2 Motivation

Up to now a tracking company had to be hired to equip all the sailors and marks with GPS trackers. These tracking companies receive the GPS position of their trackers

wirelessly over the mobile network or via ultra-short wave. They then pre-process the data and forward it to the SAP Sailing Analytics.

Therefore tracking a sailing regatta leads to high organizational efforts, high costs and is only available for high profile events. In the future tracking could be done with commodity mobile devices which are already used widely by the general public. This would facilitate the tracking of sailing races tremendously as the Sailing Analytics no longer relies on tracking providers.

By enabling sailors to track themselves using their smartphones SAP Sailing Analytics will reach a much wider audience.

1.3 Task

The goal of this project was to develop an app for a competitor of a sailing regatta. The mobile application should enable the user to register for sailing races, as well as create and administer sailing races. The user of the application should furthermore be able to set his user data e.g. name, boat, boat class and nationality. The administration of a sailing race includes the selection of the competitors, which take part in the administered race. In addition the administrator of a race, which could be the race committee of a sailing regatta or an individual person, should be able to set a course as well as the number of rounds for a race and should also be able to start the race. Having started the race the administrator should see a live overview of events happening during the race. Furthermore he should have the ability to enter information about the current happenings on the water.

During the race the mobile devices of the competitors should track their GPS-Position and send it to the SAP-Sailing Analytics Server, which is then able to calculate a live leaderboard. Special attention has to be paid to the fact that a mobile device is not connected to the internet respectively the mobile network at all times. Therefore all data has to be buffered on the mobile devices and has to be sent as soon as the mobile devices is reconnected to the network.

2 Previous Work

2.1 SAP Sailing Analytics[1]

> "Historically, it has been almost impossible for spectators to follow a sailing race. It is a costly challenge to get telling images of the water and it is a challenge to make the spectators understand the dynamic nature of the game. But with the advances in GPS tracking technology, cutting edge analytics and 3D animations, it is now possible to dive into the action of a sailing race and better understand what makes a boat win." – *Marcus Baur, three-time European 49er Champion, June 2011*

SAP Sailing Analytics not only helps the spectators of a sailing race to better understand what is currently happening on the water, but also enables the coaches and sailors to analyze the sailing race retroactively by looking at meaningful metrics.

In the following subsections different components of the SAP Sailing Analytics are presented.

2.1.1 Race Board

Figure 2.1 shows the default User Interface (UI) of the SAP Sailing Analytics named Race Board. It consists of three components: *TimeSlider*, *Map* and *Leaderboard*.

The *TimeSlider* enables the user to navigate in the already tracked parts of a race. Furthermore there is a replay function, with which the user is able to start a replay of the race starting from the timepoint selected with the *TimeSlider*.

In the *Leaderboard* on the left hand side the current rank of the sailors is displayed. Each sailor is represented by a row in the table. By expanding a column one is able to see details of a race for example the rank of the sailors at certain legs. By further expanding the sub-columns of the expanded column, one is then able to see leg details, for example

[1] cf. [Tes12]

Figure 2.1: Components of the SAP Sailing Analytics Race Board [1]

Total rank	Competitor	Name	R5	R6	L1	L2	L3	∅ Speed [kts]	Distance [m]	Rank Gain	L4	L5	L6	R7	R8	Σ
1	AUS 2621	Robbie LOVIG	1	1	1	1	1	10.35	944	0	1	1	1	~~OCS~~	1	19
3	RSA 2287	Allan LAWRENCE	4	2	2	2	2	9.52	906	0	2	2	2	~~OCS~~	4	40
2	RSA 2004	Blaine DODDS	2	3	4	4	3	9.55	895	1	3	3	3	1	2	39
11	RSA 2645	Michael GOODYER	7	4	3	3	5	8.32	871	2	4	4	4	9	7	126
4	GER 2599	Hauke BOCKELMANN	3	5	8	5	4	9.26	860	1	5	5	5	OCS	~~DNF~~	69
12	GER 180	Markus GORCZYNSKI	9	6	10	8	8	8.61	932	0	8	8	7	5	12	126
9	RSA 2639	Mark WIJTENBURG	6	7	13	12	12	9.48	987	0	12	10	8	6	10	115
8	GBR 2715	Benn GARNHAM	5	8	11	11	11	9.43	982	0	10	9	9	2	8	115

Figure 2.2: Regatta Leaderboard

the average speed of the sailors during this leg or the time distance to the leader. The data displayed after expanding a race column or a leg column is configurable and the user is able to choose from a wide range of interesting statistics including in-race rankings, average speeds, distance travelled, ETA (estimated time of arrival at the next mark rounding), gaps to leader, gains and losses per leg. In figure 2.2 a leaderboard with the third leg of race six expanded is displayed. One is also able to see that it is possible for the sailors to exclude a certain amount of races from the ranking, which is indicated by a strike through the position achieved in the column of the race. This is for example the case in race seven of Robbie Lovig, who excluded the race, because he started on course side (OCS).

The *Map*, which can be seen in figure 2.3, shows the position of the boats and marks at the timepoint chosen via the *TimeSlider*. Furthermore in the upper left corner it displays the wind speed and direction at this timepoint. The black line shown on the

[1] Source: [Tes12]

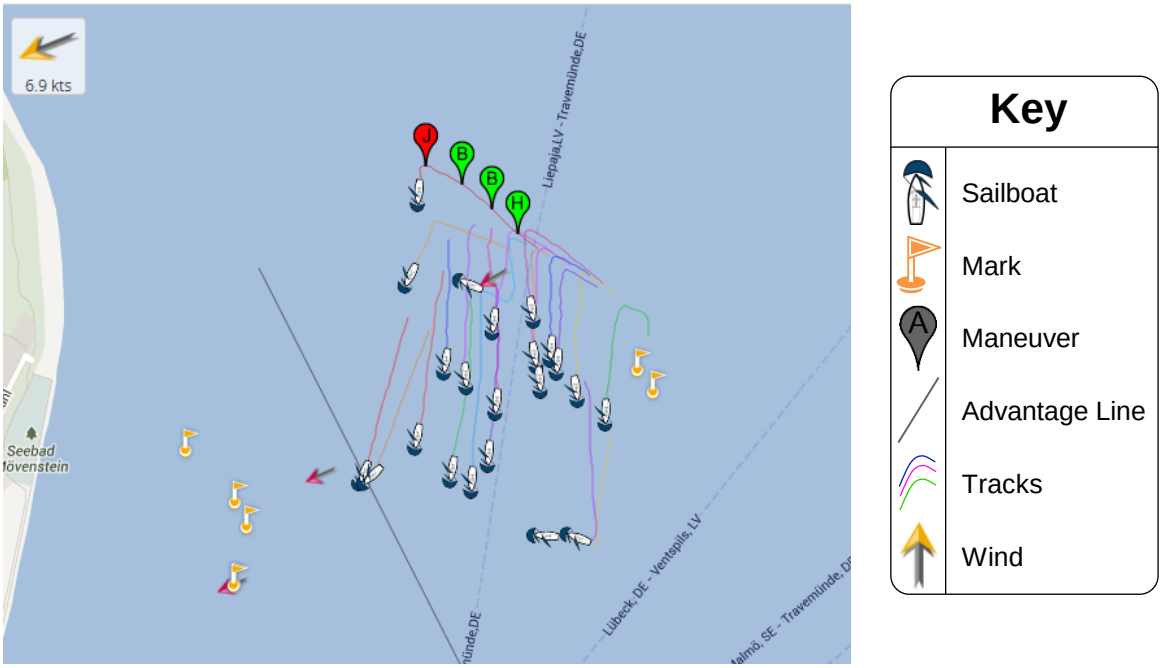


Figure 2.3: Map Visualization of Regatta[1]

map represents the advantage line. It is a tangent to the leading boat perpendicular to the wind direction. The colored lines show the previous positions of the sailboats. When a boat gets selected its maneuvers are displayed as marks on the map. The marks contain a letter, which identifies the maneuver.

The calculations made by SAP Sailing Analytics can also be visualized in competitor charts. Competitor charts help to compare the same metric of different competitors. The competitors to compare with each other can be selected in the leaderboard and competitor charts have to be activated by enabling the option "Competitor charts". Again the data displayed in the chart is configurable. The user is able to choose from a wide range of metrics. In Figure 2.4 a competitor chart for three competitors is depicted. The data being displayed is the windward distance to the leader of the race over time.

2.1.2 Administration

For administration purposes an additional UI named *Admin Console* exists. It allows users to maintain configuration and regatta data. With the administrative UI it is for example possible to create events, regattas and races.

[1] Source: [Tes12]

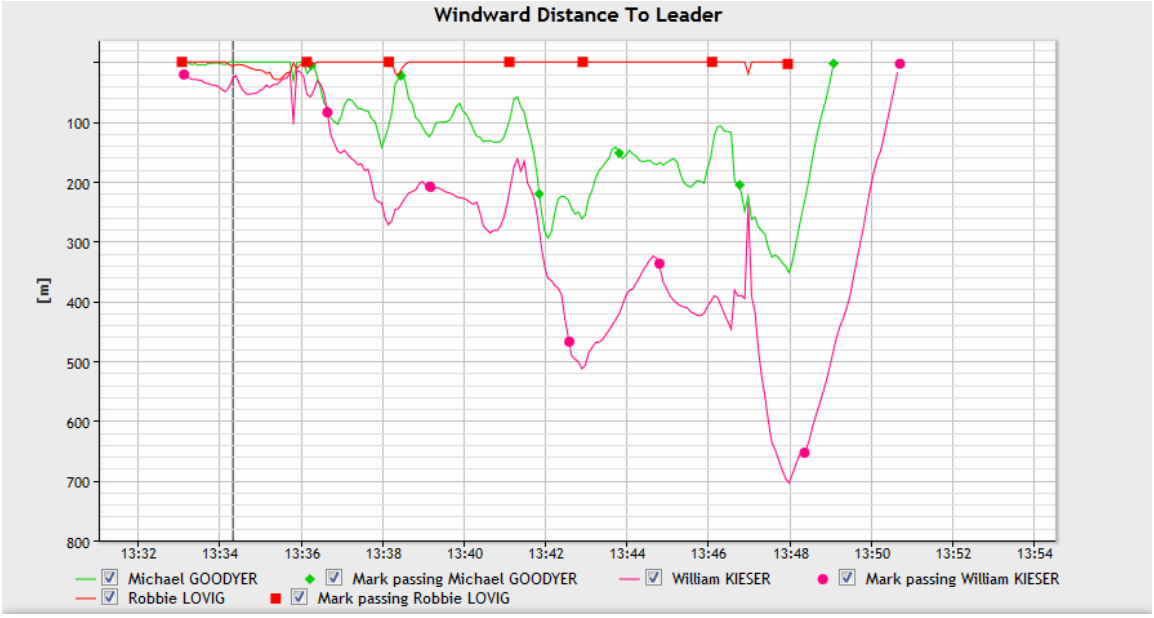


Figure 2.4: Competitor chart displaying the windward distance to the leader

In addition the connection to tracking providers is set up using the *Admin Console*. Races that have been tracked by tracking providers or where tracking is still in progress can be mapped to events in SAP Sailing Analytics.

The import of existing data of past races can also be triggered using the *Admin Console*. Furthermore an input channel for races in progress can be opened. Several connectors for various tracking partners exist, which use a variety of technologies. Some partners push the data in to the SAP Sailing Analytics system, others require to continuously poll for new data. The connectors also fulfill the task of mapping the domain model of the tracking partner to the domain model used by the SAP Sailing Analytics.

Importing past events and live events currently does not make a difference technically.

2.2 Race Committee App

The race committee app is a mobile application specifically designed for the race committee of a sailing regatta. A race committee can be compared to referees in other sports. It consists of several people observing a sailing race prior to the start, until all boats have finished. The race committee is in charge of setting the course prior the race and also in charge of adapting the course during races. The race committee also schedules the start of the race and starts the race according to one of many starting procedures. Starting procedures consist of the flying and removal of flags in fixed

time periods. Furthermore the race committee is responsible of sanctioning improper behavior, like starting to early. It is also empowered to stop, postpone or cancel a race by displaying corresponding flags. A race could for example be postponed or stopped due to changing weather conditions.

The race committee app facilitates and mirrors this process in a mobile application. This helps the race office and the spectators to get an overview in which stages the races are. Furthermore it provides the start time of a race, which is essential for all the analyses made by the SAP Sailing Analytics. Figure 2.5 shows a screenshot of the race committee app, in which a not yet scheduled race has been selected. The race committee is now able to schedule the race by entering a start time or postponing the race. It is also possible to change the course. The available buoys are presented to the user enabling him to set a buoy as a mark or two buoys as a gate. The user is also able to specify how the buoy has to be rounded. It is planned to also provide help for lying out the course, which includes a map of the buoys laid out so far and wind-based estimations where to lay out the other buoys.

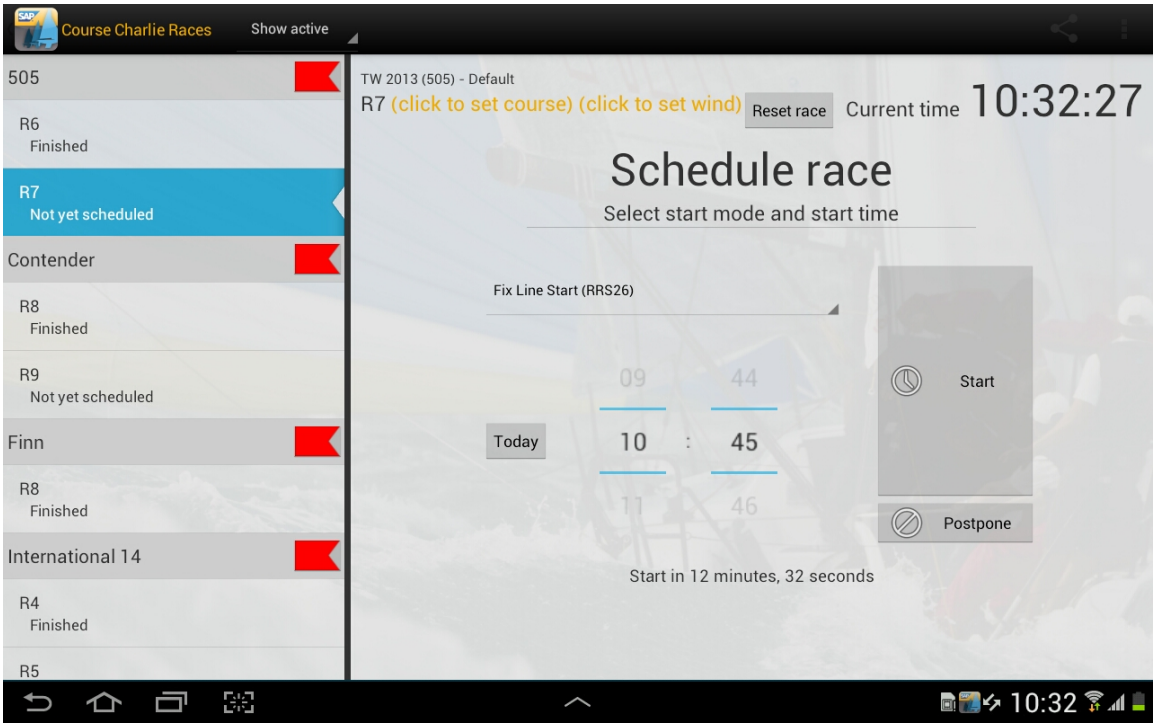


Figure 2.5: UI to set the start time in the race committee app

In figure 2.6 the mobile application's visualization of a race in its start phase is shown. The application displays, which flags are currently flying. The App also displays which flag will be the next to be displayed or removed and when.

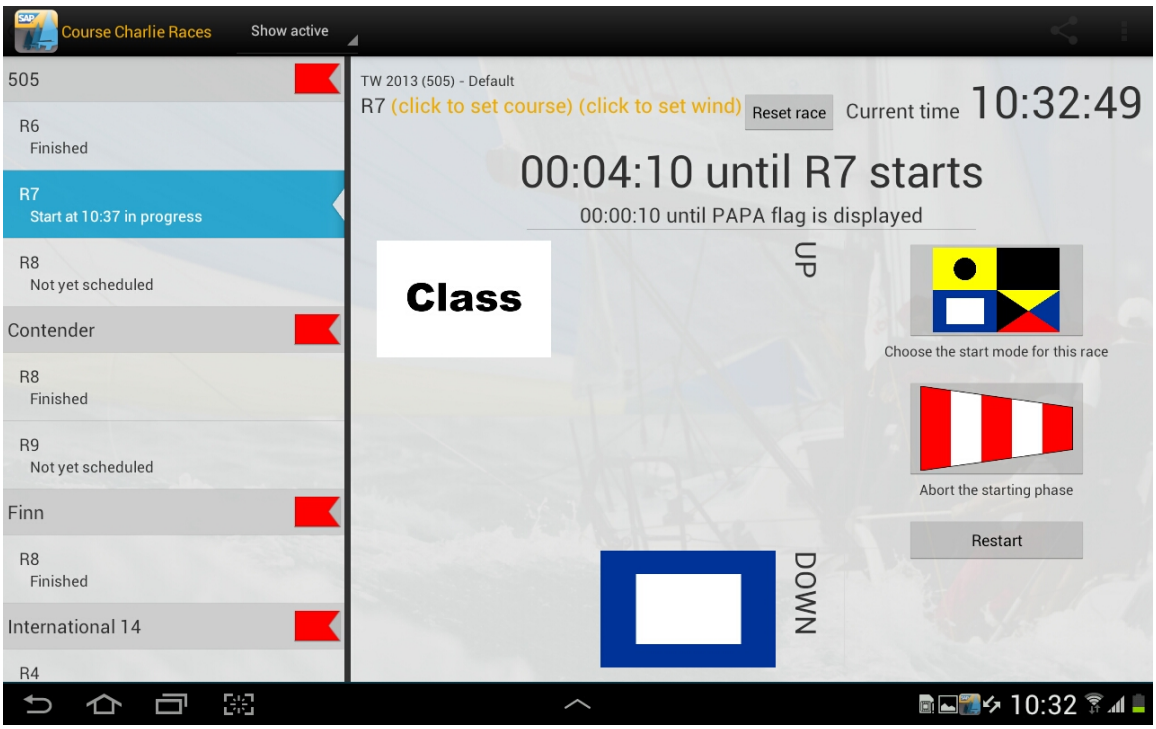


Figure 2.6: Start phase of a sailing race as displayed by the race committee app

After the start phase is over the race committee app transits to another view. This view is displayed in figure 2.7. The user of the app is now able to set recalls for individual and all sailors. The race committee is also able to set the beginning of the finish phase and to abort the race.

After declaring that the race is in its finish phase, the user can end the race by providing an end time. Finally the user is able to take pictures from the results and notes of the race committee and send these pictures to the race office via e-Mail for further processing. When the start sequence for the Extreme Sailing Series is used, the user of the application is also able to enter the finishing time of each competitor.

The race committee app saves all the operations belonging to a certain race in the so called RaceLog. This race log is written to a file on the mobile device and a background process repeatedly tries to send the operations, named RaceLogEvents, to the SAP Sailing Server. Once sending of a given event is successful, this event is deleted from the file stored on the mobile device. This process is used due to the fact that mobile devices are not continuously connected to the internet. Therefore sending data for example via an Hypertext Transfer Protocol (HTTP) Post directly when an event occurs or a user triggers an action is not possible, as in that moment there could be no connection to the mobile network and therefore not to the SAP Sailing Server.

Figure 2.7: Visualization of a race in progress

3 Requirements Analysis

The following two sections present the requirements for a mobile tracking application used by sailors. The requirements are grouped in functional and non-functional requirements.

3.1 Functional Requirements

As already mentioned in the task description (section 1.3) the mobile tracking app is designed for competitors and administrators of a sailing race alike. The Unified Modeling Language (UML) use case diagram in figure 3.1 provides an overview of the functional requirements for such an app.

The users of the application can be divided into two groups. The first group being the competitors of a sailing race and the second group the race committee respectively the administrator of a race. For smaller regattas it is possible that the person administering the race is also taking part in the race himself. A possible workflow of the mobile application is depicted in 3.2. It shows the activities a user is able to conduct, which will be described in more detail in the following sections.

3.1.1 Entering User Information

Each user should be able to enter information about himself, like for example his name and sailor number as well as his nationality, boat class and team. This information is used in the leaderboard of the SAP Sailing Analytics. Previously competitor data was delivered by tracking partners. As there are no tracking providers in the context of smartphone tracking, the user of the tracking app has to provide the relevant information himself. For now the user should be able to create a `Competitor`-Object in the SAP Sailing Analytics System and subsequently use this `Competitor` to compete in a race. Furthermore there has to be a mapping between the GPS data produced by the device being used for tracking and the corresponding competitor. In the future there

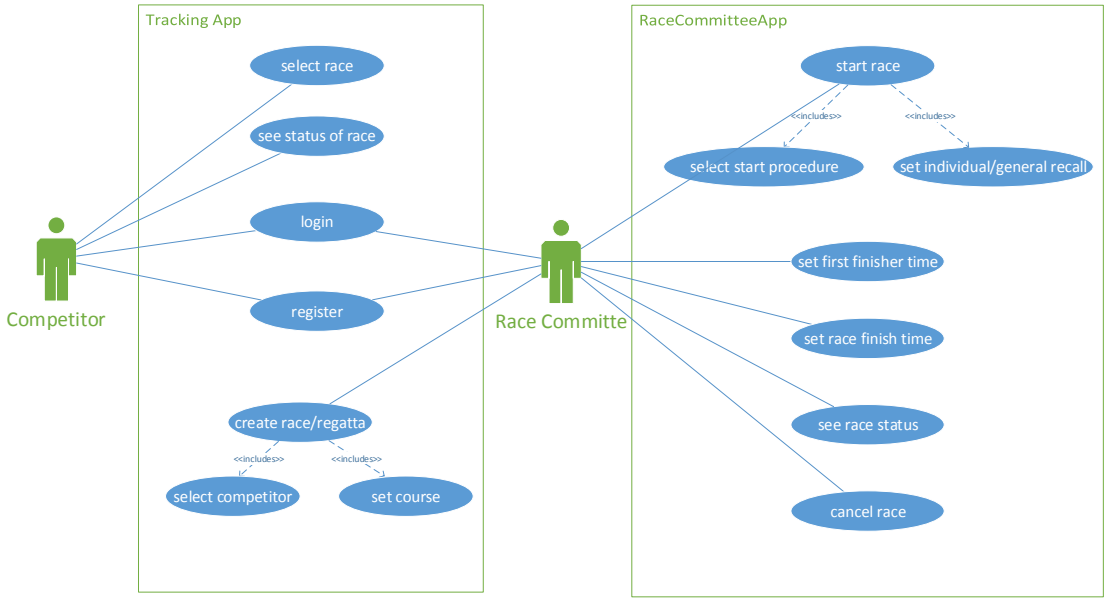


Figure 3.1: Overview of the use cases of the tracking and race committee app

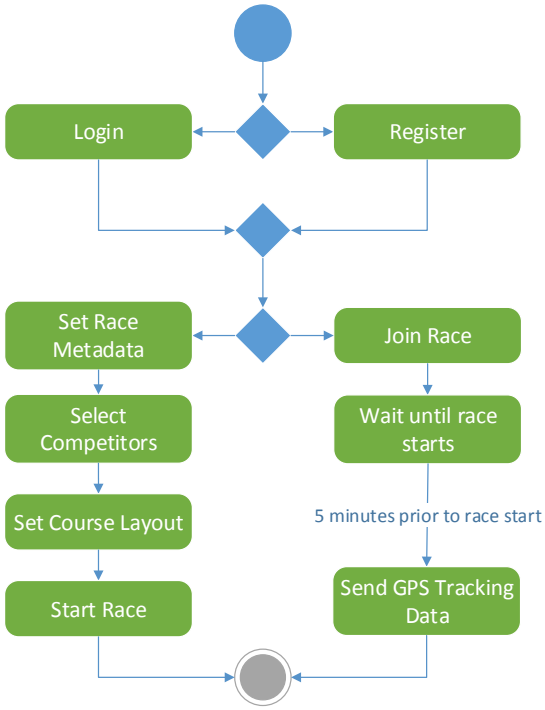


Figure 3.2: UML Activity Diagram of the Tracking App

will have to be user management integrated into the SAP Sailing Server (see section 5.1).

3.1.2 Competing in a Race

The user who wants to take part in the race should see a selection screen of races being in their pre-race phase. Races in the pre-race phase are open for competitors to register. Ideally this selection screen would be sorted or filtered according to the distance of a race's location to the user itself to facilitate finding the correct race. After having registered for a race, the user waits for the administrator of that race to schedule a start time and to start the race. If the race administrator decides to exclude this competitor from the race, the user should be notified. In addition the race state, meaning for example the start time if set, should be displayed meanwhile in the app of the competitor. Several minutes prior to the start the App should start to acquire the sailor's GPS position, due to the fact that getting a first GPS-fix can take several minutes. During the race the sailor's GPS position is transmitted to the SAP Sailing Analytics server in a consistent time interval.

3.1.3 Creating a Race

The administrator of a race wants an easy way to create new races. As the application should be used on very small training regattas as well as on big professional sailing events an easy option to just create one or a few races as well as more detailed options allowing to create complete events have to be made available. Creating events includes creating regattas and assigning regattas to leaderboards and choosing from a scoring system, on which the overall ranking of a regatta is based. Furthermore it should be configurable how many series are going to be sailed in each regatta as well as how many races per series and whether the races are divided into different fleets. Races get divided into different fleets, if there are too many boats to sail at once. Those individual races with a distinct subset of competitors still count as one actual race.

In addition the administrator of a race wants to be able to select the competitors taking part in his race, being able to exclude certain competitors. Having selected the competitors, he or she should be able to set the course and the start time. If the administrator is not taking part in the race himself all the functions from the already existent race committee app should be available to the user. If the administrator of the race is taking part in the race, he or she should be able to set the time of the first finisher and the race finish time after the race is finished.

Setting the course is a crucial operation, as most of the calculations being done by the SAP Sailing Analytics rely on the course. A course consists of multiple marks, which

could be buoys or boats. These marks identify a part of the course for example a point on the start or the finish line. In the past the course layout as well as the mark positions and mark passing times were imported from the tracking partners. Due to the fact that there is no tracking provider in the context of smartphone tracking, the mark passings have to be identified by the SAP Sailing Analytics. While a future goal is to implement a detection algorithm, which is able to calculate the marks from the GPS-tracks of multiple boats, for now a UI option has to be provided, which enables the administrator of a race to set the marks by pinging the GPS coordinates of them. This functionality could look like displayed in figure 3.3. Before being able to set the position of the marks the user has to choose a course layout. After having chosen the course layout the administrator has to set each mark's position. A map can be slid in from the right side to give the user an overview of the marks already pinged. After having pinged all the marks the user gets presented the entrance screen of the race committee app, which allows him to schedule a start time for the race. He or she is then able to use all the functionalities already implemented in the race committee app. These have already been described in section 2.2.

3.2 Non Functional Requirements

3.2.1 Semi-Connectedness

As already mentioned in section 1.3 special attention had to be paid to the fact that mobile device are not connected to the network continuously. Therefore the data has to be buffered in some way and sent to the server as soon as the connection is restored.

For this issue two options are discussed in the following sections: enhancing the existent sending mechanism of the racelog, which was already mentioned in section 2.2, or using CouchDB on serverside and Couchbase Mobile on client side.

Enhancing the RaceLog Sending Mechanism

RaceLogEvents represent an action made by the race committee for example scheduling a race start and are created by the race committee app. For these race log events a special sending mechanism that already pays attention to semi-connectedness related issues is implemented in the race committee app. This is done in a rather simple way by writing serialized RaceLogEvents with additional metadata, like the Uniform Resource Locator (URL) of the servlet the event should be send to, to a file. A background task,

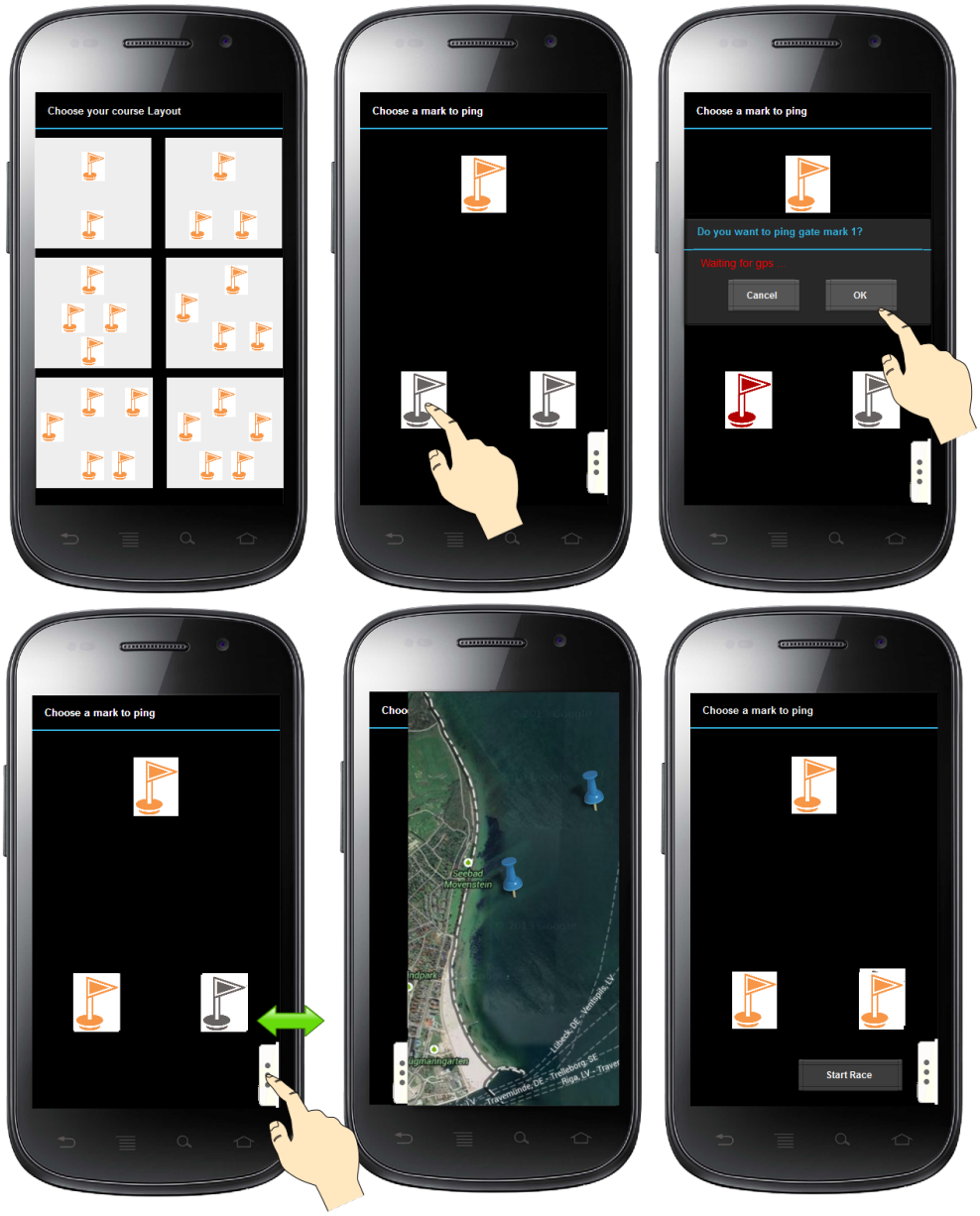


Figure 3.3: UI Mockup of the ping marks feature

named service in android, then repeatedly reads the events from the file and tries to send them to the corresponding servlet using an AsyncTask. AsyncTasks are part of the android framework and ought to be used for short running background operations.[1]

[1] cf. [Med13, p. 121 ff.]

Using CouchDB and Couchbase Mobile

Another possibility would have been to use Apache CouchDB and CouchBase Mobile[1]. CouchDB is a document oriented database which has a built-in replication mechanism.[2] The documents are stored in the JSON format. The replication mechanism of the mobile version of CouchDB, called Couchbase Mobile, takes care of semi-connectedness related issues.

Evaluation

Using CouchDb and CouchBase Mobile to avoid having to deal with semi-connectedness related issues on our own seems a legitimate approach. But using CouchDB and CouchBase Mobile would lead to a more complicated serverside landscape. As a lot of students are involved in the development of SAP Sailing Analytics, the team members change rapidly. Therefore it would take valuable time for new project members to get familiar with a more complex architecture. Furthermore it was expected to be more work than building upon the existent racelog sending mechanism, which already works well.

3.2.2 Facilitate the integration of additional sensors

Besides the actual tracking data a lot of other sensor data could be relevant for the calculations made by the SAP Sailing Analytics. The integration of additional sensors integrated in a smartphone or hooked up to a smartphone is not planned for the first prototype, but is an interesting area for future development and will also be discussed in section 5.5.

While the integration itself is not being implemented, the mobile application should be designed to be easily enhanceable for different types of sensors.

[1] formerly known as TouchDB
[2] cf. [Fou13]

4 Smartphone Tracking App

> "This is still a vision, it will still take a while. But where professional tracking systems are used today, in future smart-phones could be used. One can well imagine that at some point in the future, a club regatta will be tracked by registering smart-phones with SAP Sailing Analytics, so that after going out onto the water and racing, the sailors can watch their own race back in the club-house." – *Stefan Lacher, Head of Technology SAP Sponsorships, July 2012*

4.1 Insourcing vs. Partnering

During the process of planning the development of the Tracking App partnering was taken into consideration as a german university is working on a similar mobile application.

Partnering comes with a lot of advantages. One of them being not having to do the work yourself and therefore being able to focus on your main tasks. Furthermore another advantage of partnering is that you need less workers.

The downside of partnering is that one has less influence on the design, focus and feature set of the application. Furthermore the partner might not have an as good domain know-how as the Sailing Analytics core team acquired over the years. He might also not be able to carry out quick fixes for problems or adaptions when something on serverside gets changed. It would also lead to a considerable communication expenditure as the partner is not located in the same building or city.

4.2 Architecture

The Tracking App is based on the racecommittee app as the tracking app already fulfills the needs of a racecommittee, which wants to administer a running race. The duties of a race committee are described in section 2.2. Therefore a lot of code and existent

functionality could be reused. Unfortunately the racecommittee app is build exclusively for tablets and must therefore be adapted to fit on smaller screens, as described in section 5.4.

The Tracking Application is a fragment based application. "A Fragment represents a behavior or a portion of user interface in an Activity."[Inc13] Fragments encapsulate reusable behavior and layouts. Multiple Fragments can be combined to form the layout of an activity.

In the SAP Tracking App the layout of each activity contains one `FrameLayout` in which different Fragments can be loaded. Figure 4.1 gives an overview of the application's activities and the fragments being used in each activity as well as the workflow of the mobile application.

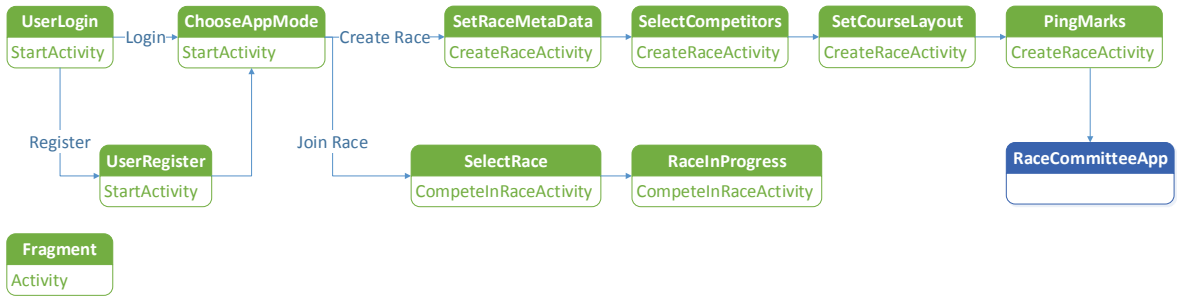


Figure 4.1: Overview of Fragments and Activities

Most of the Fragments used in the tracking app can be divided into two groups. Their purpose is either to provide the user with a UI, where you are able to enter data, which is then send to the server. This is the case for registering a user, where the data of the new user is send to the server and also for creating a race, where the data of a new race is send to the server, as well as for pinging marks. The other purpose that is suited is selecting one or multiple elements from a list. This is the case for the selection of competitors, which is being done by the administrator of a race and for the selection of a race, which is being done by a user who wants to compete in a race.

The second group of Fragments, which displays a list of options to select for the user inherits from an abstract class which provides access to a `DataManager`. With a `DataManager` it is possible to access a `DataStore` or to trigger a `DataLoader`, to load the required Data, which then sends the loaded Data to a `DataHandler` which in turn saves the data in a `DataStore`.

4.2.1 Communication

Communication Channels

Depending on the data to be send the app uses different communication channels. Figure 4.2 provides an architecture overview and depicts the different communication channels used by the tracking application.

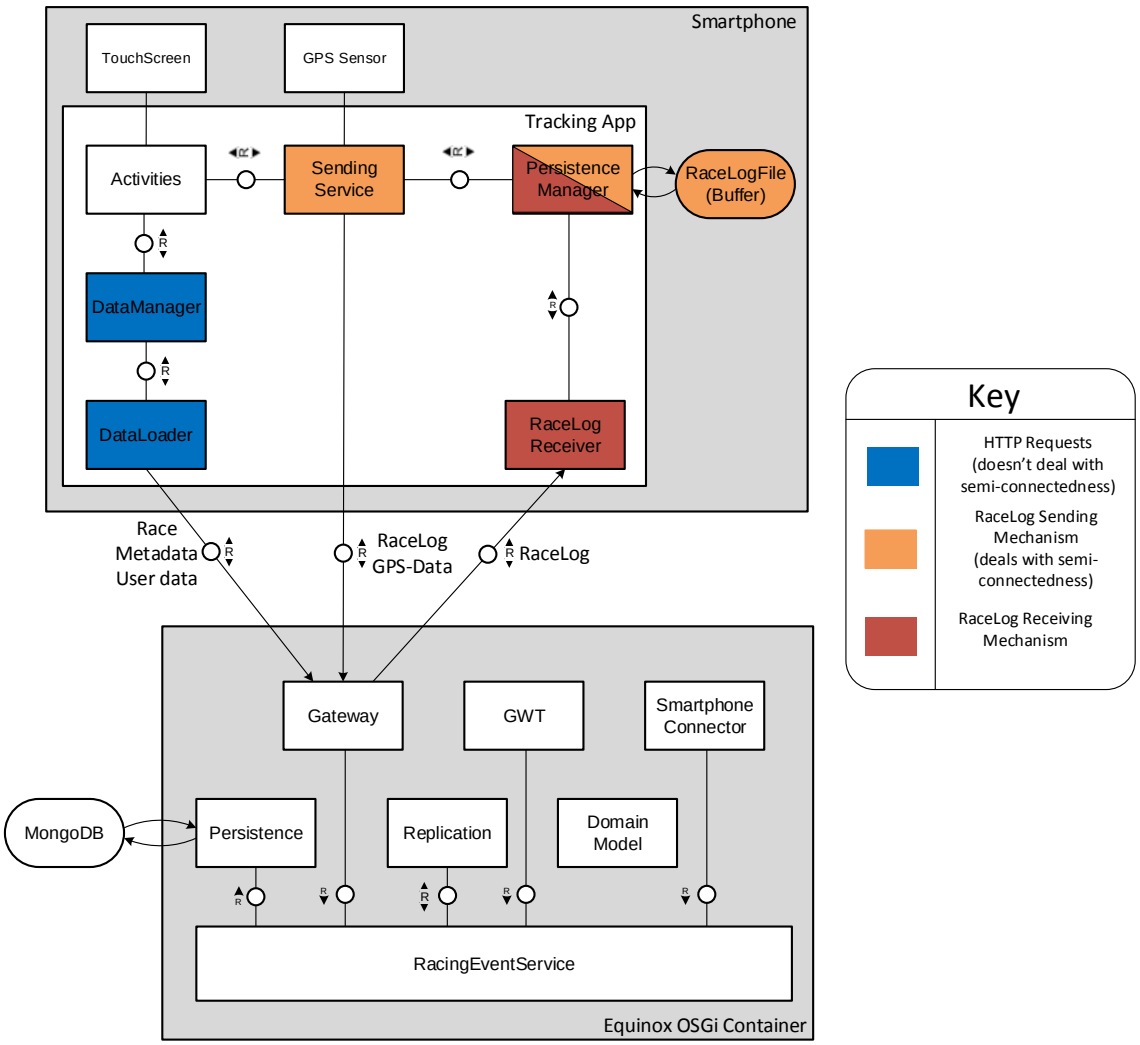


Figure 4.2: Architecture Overview

Servlets: Everything that is not directly related to a specific Race (or rather RaceLog) is handled via `POST` and `GET` servlets, where the data is described as JSON. Examples are: Creating a Race, managing Competitors. It makes no sense to use a sending mechanism that deals with semi-connectedness for certain tasks. For example when the user wants to create a race, he should get immediate feedback whether

the creation succeeded or failed as the next steps, for example the selection of competitors or the registration of competitors only work if the creation succeeded.

RaceLog: All the "master data" communication concerning one race in particular uses the existing RaceLog-mechanism, which already deals with semi-connectedness, persistence and replication. Examples for this are: adding competitors to a race, mapping tracking devices to competitors, starting a race, defining the course layout. The RaceLog has been enhanced, so that it get's automatically synchronized between all the devices that are registered for a race.

Other: The actual tracking data (perhaps also additional data: wind etc.) also has to be transferred. On client side as described in section 3.2.1 we want to use the communication mechanism of the race log, which already deals with semi-connectedness related issues.

Communication during a Race

Figure 4.5 depicts the communication between the tracking app, the app's backend and the SAP Sailing Analytics server during the creation of a race. To create a trackable race the race passes through several phases. This process is displayed in figure 4.3.

To create a race in pre-race phase a HTTP POST with the leaderboard, to which the race to be created should belong to, the race's name and the boat class competing in the race is sent to the app's backend.

The competitors are then able to register for this race by using a RaceLogEvent called `RaceLogPersistentCompetitorRegisteredEvent`. The event includes information about the competitor to register as well as an identifier called `SmartphoneIdentifier`. This identifier later helps to map the incoming GPS fixes to the corresponding competitor. Every competitor that registered himself using the described RaceLogEvent will later be included in the `RaceDefinition`, as soon as the race is created.

The race administrator is now able to choose the competitors taking part in the race. If competitors registered, which the administrator does not want to take part in the race, he is able to unregister them by using the RaceLogEvent `RaceLogPersistentCompetitorUnregisterEvent`.

After having selected the competitors the administrator is able to set the course layout. The course layout is set by defining how many marks and gates are used to identify the course. The server gets notified of the course layout selection by a RaceLogEvent

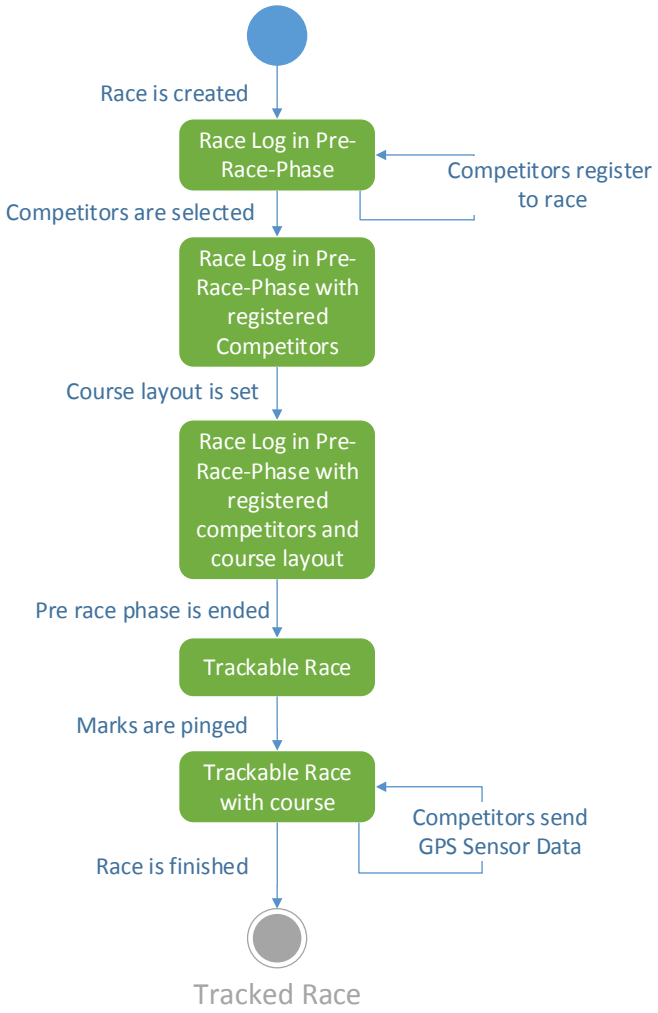


Figure 4.3: State Diagram of a race

called `RaceLogCourseDefinitionChangedEvent`, which includes all marks with their corresponding name.

Subsequently the race can be transferred to a trackable race, where no additional competitors can be added, the boat class is fixed and tracking may begin. The administrator of a race is now able to set the marks position by sending an HTTP-Post to the backend of the application, which includes the marks Id and its position. This would ideally also benefit from the RaceLogs underlying sending mechanism, to ensure the correct handling of semi-connectedness.

After having pinged the marks, a start-time in the racecommittee-app can be set, which leads to a `StartTimeEvent` in the RaceLog. Due to the fact that the RaceLog is synchronized across all competitors, the competitors are able to listen for a `StartTimeEvent`. Now that the competitors know the start time they can start acquiring and sending their

GPS position five minutes prior to the start of the race. The GPS fixes are send using the sending mechanism of the RaceLog to deal with semi-connectedness, but are not written into the RaceLog.

As mentioned in the non-functional requirements section (see section 3.2.2) the application should be easily enhanceable to support various sensors. For marine-related sensor equipment the National Marine Electronics Association (NMEA) 0183 standard is a de jure standard. It defines a protocol, originally intended for serial data buses in nautical environments.[1]

The NMEA standard specifies so called sentences for different sensors. In figure 4.4 an example for an NMEA sentence used for GPS positions is depicted.This sentence, which is the recommended minimum for GPS transit data, is used for the transmission of the boat position during a race.

By using the sentence formats specified by the NMEA it will be easier to add additional sensors, as the app's backend is already familiar with the NMEA standard and can be held generic enough to support all the sensors, which use NMEA 0183 sentences. Furthermore it is much more likely that this Application Programming Interface (API) will be reused by other clients.

A shortcoming of the NMEA sentence format is that it one is not able to identify the device, which sent a sentence by looking solely at the sentence. As the GPS data has to be mapped to competitors it must be identifiable which competitor send the data. Therefore the NMEA sentence is encapsulated in a JSON format and meta-data is added. Listing 4.1 shows a JSON message containing two GPS fixes. Besides the actual GPS data the message contains a unixtime-timestamp and the International Mobile Equipment Identity (IMEI)-number of the device used for sending the message.

```
{"imei": "12345678",
  "data": [
    {"unixtime": 2394820480284,
     "nmea"     :  "$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A"
    },
    {"unixtime": 2394820480287,
     "nmea"     :  "$GPRMC,123519,A,4807.042,N,01131.015,E,022.6,084.5,230394,003.1,W*60"
    }
  ]
}
```

Listing 4.1: Json Message containing two GPS-Fixes in the NMEA GPRMC format

[1] cf. [Nat08]

```
$GP RMC,220516,A,5133.82N,00042.24W,173.8,...,*70<CR><LF>
 1  2     3    4      5         6       7         8
```

Field	Name	Description
1	Talker ID	Identifies the type of device sending the sentence.
2	Sentence Type	Every sentence type has a defined number of fields with defined lengths.
3	Time	Universal Time Coordinated (UTC) timestamp in the format *hhmmss* (*h* hours, *m* minutes, *s* seconds).
4	Status	A for OK and V for warnings.
5	Latitude	Latitude in the format *ddmm.mm,h* (*d* degrees, *m* minutes, *h* hemisphere).
6	Longitude	Longitude in the format *dddmm.mmmm,h* (*d* degrees, *m* minutes, *h* hemisphere).
7	Speed	Speed over ground in knots
8	Checksum	Checksum over the sentence up to this point.

Figure 4.4: NMEA 0183 Recommended Minimum Sentence[2]

[2] cf. [Bad01] and [Tes12]

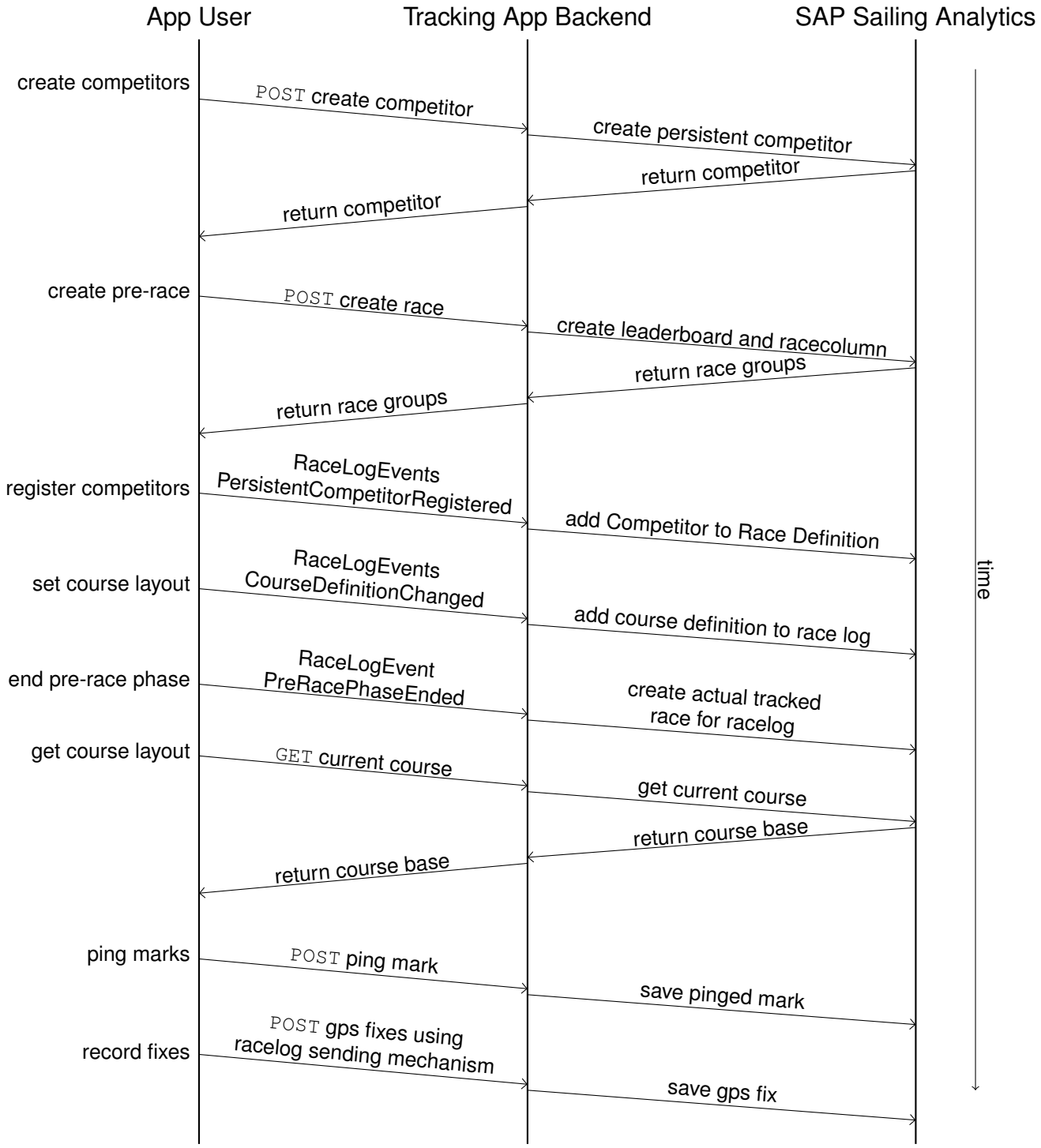


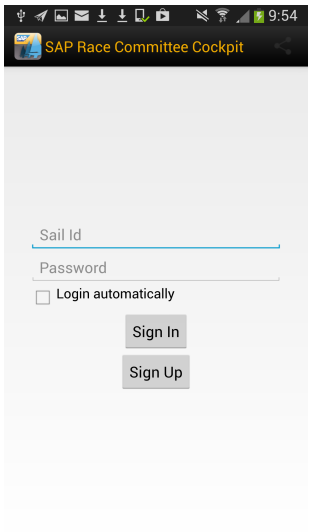
Figure 4.5: Typical communication between the App, its backend and the SAP Sailing Analytics server during a race

4.3 Implementation

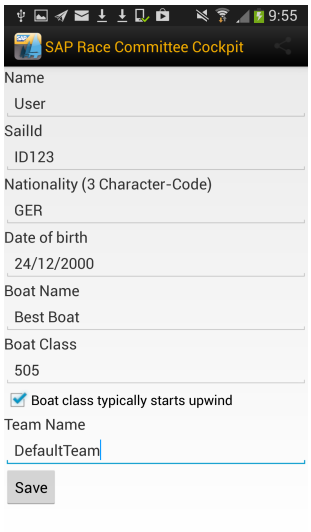
During this practical phase a prototypical implementation has been built, which is the first breakthrough in smartphone tracking for SAP. With the prototype it is already possible to create simple races and to register for races, which have been created by other users. After having joined the race the user is asked to enable GPS, if not already done so, and tracking is started. The GPS positions are sent using the RaceLog sending mechanism, which deals with semi-connectedness. A red symbol is shown in the upper right corner, when there is data buffered, which has not yet been send to the backend due to connectivity issues. By pressing on the symbol the user is able to get more information about the current sending status. Alternatively the GPS Positions can be written to a file. In the future an option could be provided to start recording the GPS positions to a file without having selected a race. This could be useful if the administrator is not able to create a race due to bad or no mobile network connectivity. The race could then be created afterwards and the sailors could upload the file, which contains their GPS positions during the race.

The administrator of a race is able to pick the competitors from a list, which displays the competitors that registered for the race. Currently a function to unregister a competitor from a race is missing in the backend, but it will be implemented promptly. Furthermore the administrator of a race is able to set a course layout by stating how many marks and buoys are on the course and is subsequently able to ping the marks position. After having pinged the marks, the administrator is able to start the race. As of now the functions of the race committee app to administer a race cannot be reused, as a different type of leaderboard is used in the backend system. The change of the leaderboard will be a minor change in the backend of the App.

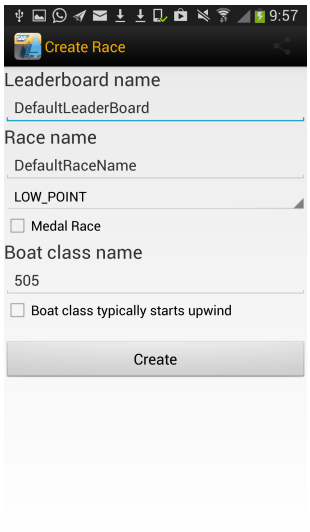
The functions already available in the prototype are shown in figure 4.6.



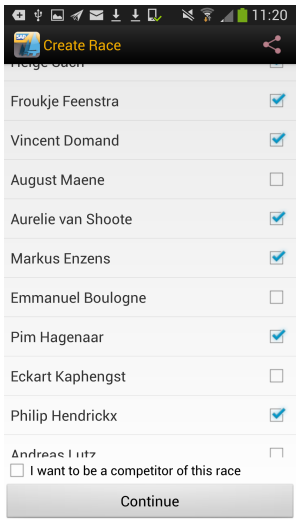
(a) Login



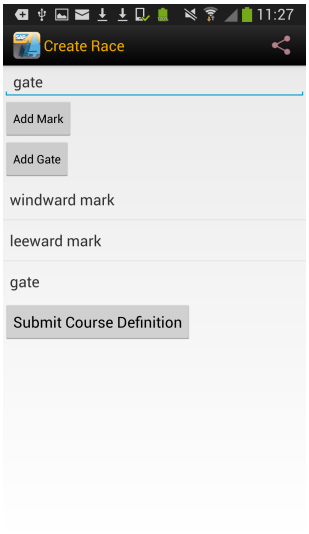
(b) Register



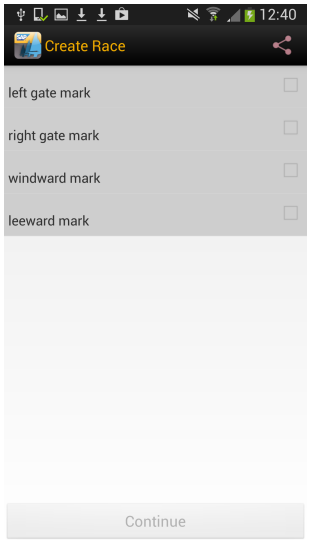
(c) Create race



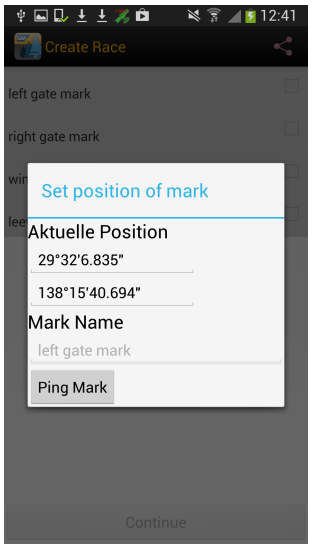
(d) Competitor selection



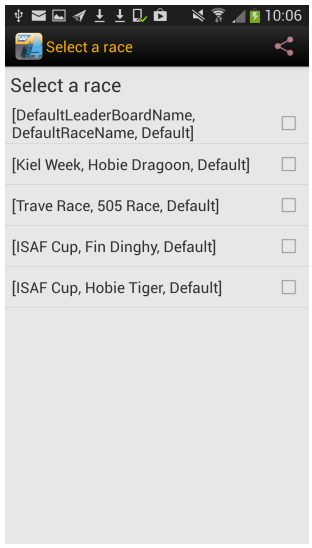
(e) Set course layout



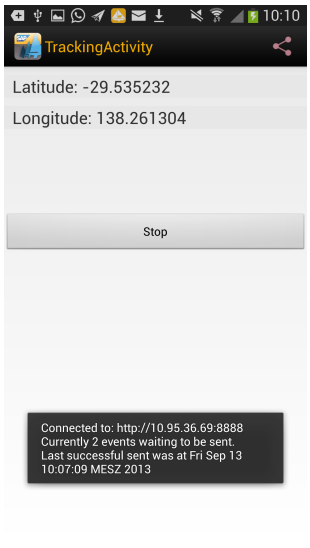
(f) Select mark to set position



(g) Set mark position



(h) Join race



(i) Tracking

Figure 4.6: Several screenshots showing the current functions of the tracking app

5 Future Work

For a fully fledged mobile tracking integration in SAP Sailing Analytics still a lot of work has to be done. The following sections provide an overview of the work that still has to be done and provides an outlook what could be done in the future.

5.1 User Management

So far there is no User Management built into the SAP Sailing Analytics. A User and role management is required for multiple reasons.[1]

Firstly for saving user-related data. This would include the users sailor number, boat type, name and other user related settings. Allowing a user to create an account for the SAP Sailing Analytics facilitates the usage of the mobile tracking application, as the user data does not have to be entered for every race or when a different mobile device is used. The user could just enter his data once and log in from every device, that has the mobile tracking application installed.

Secondly having users and roles it would be able to offer special services and functions only to a restricted audience.

Lastly, it would allow to remember user specific settings for views, such as the leader board, race board and may also help configuring a "portal" with widgets in a personalized way.

5.2 Tracking App

As of now the tracking app presented in this thesis is just a prototypical implementation and a first breakthrough on which further development can be build on. A lot of functions, which are necessary or useful for a tracking app or only implemented

[1] cf. ([Uhl13])

prototypically, in a simplified way or are even missing. The user is for example able to create a leaderboard with a single race, but not multiple leaderboards, which belonging to an event, and contain several race series.

Additionally there will have to be a testing phase with a limited number of users. When the app gets rolled out to a huge amount of users, which could possibly be via Google's Play Store[1], a sever-landscape that is easily enhanceable to serve the workload of a big and rapidly growing user base has to be provided.

5.3 Server landscape

As there will likely bee a rapidly growing number of users for the mobile tracking app and therefore for the SAP Sailing Analytics a scalable server infrastructure has to be provided. Furthermore the replication function of the SAP Sailing Analytics has to be enhanced.

5.4 Smartphone Version of the Racecommittee App

The racecommittee app is build exclusively for ten inch tablets and is therefore not usable on a smartphone. As the tracking app is an enhancement of the racecommittee app and should mainly be used on smarpthones, the UI of the racecommittee app has to be adapted to support different screen sizes. Supporting different screen sizes with one app is possible by the usage of so called fragments. Fragments are reusable UI components

5.5 Integration of Additional Sensors

Smartphones could deliever a lot of useful information besides GPS-data. A lot of sensors are already included in a smartphone itself, like for example a sensor for acceleration and a compass. Furthermore other sensors can be hooked up to smartphones. This is for example the case for wind sensors.

Leveraging this sensors data would lead to new and more accurate analyses made by the SAP Sailing Analytics and would therefore generate additional value.

[1] Google Play Store is the main source of applications for android and is preinstalled on every android phone

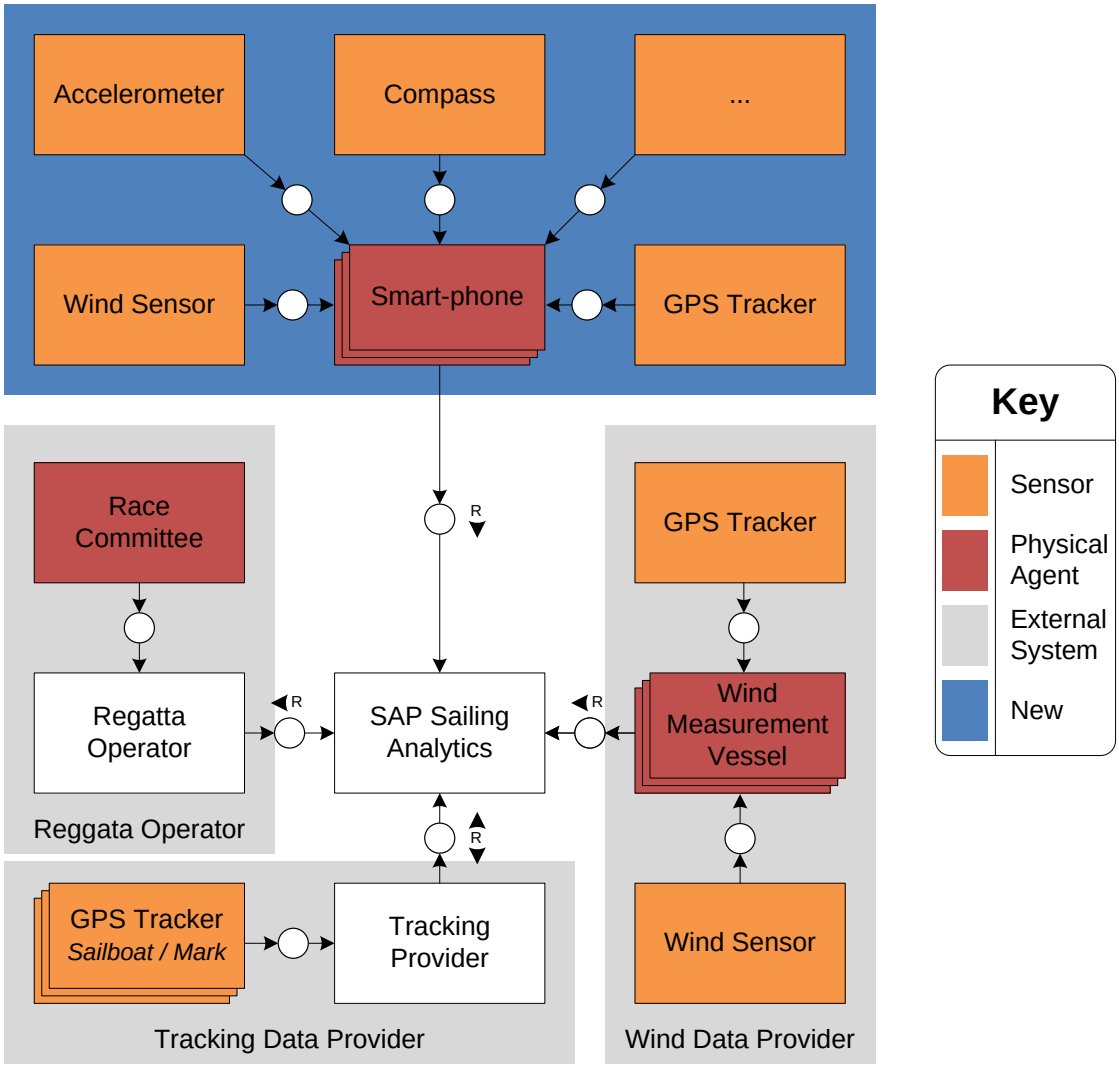


Figure 5.1: Integration of smartphones into SAP Sailing Analytics[1]

5.6 Dynamic Power Consumption Adaption

Due to the fact that mobile devices rely on batteries as their power source power consumption is an important topic. Especially acquiring and sending GPS-Fixes in small intervals leads to a high power consumption of the mobile app. In the past tracking devices with low or empty batteries could be exchanged during two races or multiple tracking devices were put on a boat to have redundancy.

For smartphones it would be beneficial to have a dynamic power management build into the App. This power management should be able to adapt the power consumption

[1] Source: [Tes12]

of the App by reducing the number of GPS-Fixes being acquired in a certain time period or by buffering a couple of GPS-Fixes and sending them as one request. This would lead to less dropouts, because of low batteries and would allow to track longer sailing races.

6 Conclusion

The prototype built during this practical phase is a first step towards a mobile tracking application, which empowers sailors to track their races in an easy way. As of now the application fulfills the most basic requirements for a mobile tracking application and is therefore a good starting point for further enhancements. So far it is for example only possible to create a regatta with a single race. In the future more complex race creation options will have to be provided to support bigger and more complex events.

Enabling the sailors to track and thus analyze their sailing races in an easy way will have a huge impact on the way sailing is experienced. The computations being done by the SAP Sailing Analytics will help to evaluate the race and to get a better understanding of what makes a boat win.

As already mentioned the results presented in this thesis are just laying a foundation and a lot of work still has to be done. Not only the mobile application has to be enhanced to be able to carry out a testing period with a limited number of users, also the server landscape and the server itself has to be improved to handle the vast amount of expected users.

List of Figures

Bibliography

[Bad01] Glenn Baddeley. Gps - nmea sentence information, 2001. Link: http://aprs.gids.nl/nmea/#rmc.

[Fou13] Apache Software Foundation. Apache couch db. Link: http://en.wikipedia.org/wiki/JSON, March 2013.

[Inc13] Google Inc. Android developers reference - fragments. Link: http://developer.android.com/guide/components/fragments.html, September 2013.

[Med13] Zigurd R. Mednieks, editor. *Android-Programmierung : [Java-Programmierung für moderne mobile Endgeräte].* OReilly, Beijing, 2. edition, 2013.

[Nat08] National Marine Electronics Association. Nmea 0183 standard. Link: http://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp, 2008.

[Tes12] Fredrik Teschke. Server side integration of mobile devices into sap sailing analytics, 2012. Link: http://wiki.sapsailing.com/doc/theses/20130210_Teschke_Server-side_Integration_Mobile_Devices.pdf.

[Uhl13] Dr. Axel Uhl. Project planning - user management. Link: http://wiki.sapsailing.com/wiki/planning/usermanagement, May 2013.

[Wik13a] Wikipedia. Oauth. Link: http://en.wikipedia.org/wiki/JSON, March 2013.

[Wik13b] Wikipedia. Oauth. Link: http://en.wikipedia.org/wiki/Unix_time, March 2013.

Acronyms

API Application Programming Interface.

GPS Global Positioning System.

HTTP Hypertext Transfer Protocol.

IMEI International Mobile Equipment Identity.

NMEA National Marine Electronics Association.

SAP Originally: Systemanalysis and Programdevelopment, since 1976: Systems, Applications and Products in Data Processing.

UI User Interface.

UML Unified Modeling Language.

URL Uniform Resource Locator.

UTC Universal Time Coordinated.

Glossary

JSON

JSON or JavaScript Object Notation, is a text-based open standard designed for human-readable data interchange. It is derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for many languages. [Wik13a].

unixtime

Unix time, or POSIX time, is a system for describing instances in time, defined as the number of seconds that have elapsed since midnight Coordinated Universal Time (UTC), 1 January 1970, not counting leap seconds. It is used widely in Unix-like and many other operating systems and file formats. It is neither a linear representation of time nor a true representation of UTC. [Wik13b].