Fontys

University of Applied Sciences

FINAL REPORT
2015 SOFA GTL
TREEWATCH PROJECT

TREEWATCH

Jan Kerkenhoff
Martijn Bonajo
Max van der Linden
René Karoff
Ron Gebauer

8th January 2016

Document information

Document name:	Final Report
Document owner:	TreeWatch
Company/Organisation:	Fleuren Baarlo
Contact person:	Max van der Linden, Group leader
Date:	8th January 2016
Place:	Fontys University of Applied Science Venlo
Authors:	Jan Kerkenhoff j.kerkenhoff@student.fontys.nl 2197164 Martijn Bonajo m.bonajo@student.fontys.nl 2213297 Max van der Linden max.vanderlinden@student.fontys.nl 2209349 René Karoff r.karoff@student.fontys.nl 2198664 Ron Gebauer r.gebauer@student.fontys.nl 2153294

Acronyms

API Application program interface. 27, *Glossary:* Application program interface

app Application. 2, 3, 5, 6, 14, 16, 18, 22, 23

GUI Graphical User Interface. ii, 8, 23, *Glossary:* Graphical User Interface

KML Keyhole Markup Language. ii, 19, *Glossary:* Keyhole Markup Language

XML Extensible Markup Language. ii, 19, *Glossary:* Extensible Markup Language

Glossary

Application program interface is a software intermediary that makes it possible for application programs to interact with each other and share data. It's often an implementation of REST that exposes a specific software functionality while protecting the rest of the application.. ii, 27

Extensible Markup Language is a markup language to display hierarchic structured data in text files. ii, 19

Geofencing is a feature in a software program that uses the global positioning system (GPS) to define geographical boundaries. A geofence is a virtual barrier.. 19

Graphical User Interface is a user interface that includes graphical elements, such as windows, icons and buttons. ii, 8

Keyhole Markup Language is a markup language based on XML used for geospatial data. ii, 19

map tiles is part of a way maps are displayed digitally, each tile corresponds to a specific area of the map. Ever map consists of multiple tiles that contain information for ever zoom level. 18

Nuget is open source package manager for .Net projects developed by microsoft Homepage . 16

overlay is an way to display additonal data on top of some thing else like a map. Can be a image or a shape or something else.. 16, 19

polygon is a plane figure that is bounded by a finite chain of straight line segments closing in a loop to form a closed chain or circuit. 17

renderer component of Xamarin Forms, responsible to create the actual represenation for the data for the device. 16

Xamarin Forms is a framework developed by Xamarin Inc to rappidly built cross-platfrom apps. It can be compiled to iOS, Android an Windows Phone from a shared code base.. 16, 27

List of Figures

List of Tables

Contents

1 Introduction

This document is about a project called Software Factory which has to be done during the 7th semester at Fontys University of Applied Science Venlo. The project brings Software Engineering and Business Informatics Students together to develop a solution for a real customer. In this case the customer is called Fleuren.

One goal of this project is to let the students act as the SoFa was their own company. This means the group works independent from the lecturers. Only the process coach keeps track of what the students do. In meetings which where scheduled on a weekly base, the process coach got information about the current status and if there were any problems.

1.1 The Customer

Fleuren is a tree nursery using high precision farming to grow fruit trees. After two or three years, depending on the variety, the trees are sold to companies which then harvest the fruits. Precision farming, in case of Fleuren, means that the trees are planted automatically by a tractor in rows of six trees of the same kind, this is then called a block.

1.2 Structure

2 Project definition

Fleuren has given the SoFa TreeWatch group the assignment to create a multi-platform app. The app should provide Fleuren with insight in what has been done to their fields and should give an overview of what still needs to be done to the fields. Using the app Fleuren should then also be able to make long term decisions based on the data shown. For instance when a certain field produces fruit trees of better quality it should be possible to look into the history of that field and compare that history to the history of a field which has produced lower quality trees. By looking at the differences Fleuren should be able to spot patterns in how to improve the quality of its trees.

TreeWatch project group will work on the project for five months. During this time the TreeWatch project group will try to fullfill as much requirements as possible. After the five month period VAA ICT Consultancy will take over and finish the project. The take over means that it will be important for the TreeWatch project group to properly document all the code.

2.1 Problem description

The main problem of Fleuren is that they started to notice that the quality of some fruit trees decreased due to human error. For instance one block of trees would get sprayed twice because something went wrong when registering which blocks have been sprayed. This is an unnecessary waste of resources.

Another problem Fleuren has, is that there is no clear overview of what happens when they change one of the steps within the production process. Therefore it is hard to improve the production and make decisions which affect long term production.

2.2 Scope

This section is about what is in scope of this project and what needs to be done afterwards by the VAA ICT Consultancy after they took over the project.

2.2.1 In Scope

This document is about analysis, design and implementation of a functional prototype of a mobile app running on iOS and Android.

2.2.2 Out of Scope

Not inside of the scope is anything which is needed to provide the app with data from external sources. This means servers or web services are not created during this project.

2.3 Stakeholders

This paragraph contains an overview of the most important stakeholder information of the stakeholders of this project. An overview will also be given to show the importance and influence of each stakeholder. In order to improve communication contact information is added. To clarify who benefits most from the success of the project, numbers have been added. The bigger the number, the higher the importance/influence of the stakeholder.

Stakeholder	**Important notes**	**Profit from project success**	**Influence on project success**	**Contact information**
TreeWatch group	Main developers	7	9	gtl@fontysvenlo.org
Han Fleuren	Owner of Fleuren Baarlo	8	4	directie@fleuren.nl
Yannick Smedts	Project leader of Fleuren	7	7	planning@fleuren.nl
Randy Wilbrink	Consultant of VAA	5	7	rwilbrink@vaa.com
Jan Jacobs	Coach	3	5	jan.jacobs@fontys.nl

Table 1: List of Stakeholders

3 Planning

This chapter is about the overall planning of the TreeWatch project, which means it contains the roles each group member was elected for. Furthermore the overall planning contains the different project phases, the deliverables and milestones as well as the time-planning.

3.1 Roles

The following rules were assigned to the group members by the group. Every group member was asked to give his preferred role and since there were nor conflicts everyone got the role which he preferred. Each role had it's own responsibilities which are described in this subchapter.

Project Manager
This role was given to Max van der Linden as he studies Business Informatics which is a good basis for this role. The project manager was responsible for the communication with the customer and the planning of the project.

Quality Manager
The Quality Manager role was assigned to René Karoff. The Quality Manager was responsible for defining the quality standards which have to be met by all implemented features e.g. code-coverage in testing or coding style guides. Furthermore he took care of the requirements and that everything implemented matched them.

Scrum Master
The Scrum Master role was chosen by Ron Gebauer. The Scrum Master was had the task to set up the scrum environment and to solve problems. This means that, if there were problems which kept the group or a group member from working on the project he was about to solve it.

Configuration Manager
The Configuration Manager had to make sure that the development environment ran smooth. This means e.g. solve issues with the used IDE or used servers.

Main Software Engineer
The Main Software Engineer was responsible for the implementation. He made sure that coding was done. If someone struggled with an issue too long, he tried to help him out.

3.2 Phases

Since this project was done using scrum, the typical phases as analysis, design and implementation as used in the waterfall methodology do not fit to us. But still the project could be broken down to analysis, design, implementation and documenting sprints. Each sprint took two weeks.

Analysis sprints
During the analysis sprints the project was set up and the user stories were written and validated by the customer.

Design sprints
During the design sprints the first impression of a design was created using mockups.

Implementation sprints
These were used to implement the functional prototype. The implementation sprints were planned as the most time consuming sprints.

Documenting sprint
At the end of the project there will be a documenting sprint, which is used only to document the everything what was done, so the VAA ICT Consultancy is able to work on the application after the project was finished by the SoFa....

After every sprint a sprint-meeting with the customer was scheduled to give an update to him, clarify any misunderstandings and receive possible new ideas from the customer. With this meeting the customer was able to take part at the development of the app and could influence it e.g. when he wanted another feature with a higher priority than others. The customer also could introduce totally new ideas which should be implemented first. This is the main strength of an agile methodology like scrum.

3.3 Deliverables

This section describes the deliverables which had to be handed over to Fontys and/or the customer.

User Stories
The User Stories had to be defined by the TreeWatch group and the customer had to prioritise them.

Personal development plan
This document is about how every group member wants to improve during the

project.

Personal competence plan

The Personal Competence Plan is a reworked personal development plan extended to s.m.a.r.t. details.

Mockup-Design

The Mockup-Design will be created by the TreeWatch group and provided to the customer to give him an idea of how the app could look like.

Functional prototype

The prototype is most time consuming deliverable and milestone.

Handover

This is the document which will be needed to start working on the app directly afterwards we finished the project. It contains what was done and what needs to be done to finalise the application.

3.4 Milestones

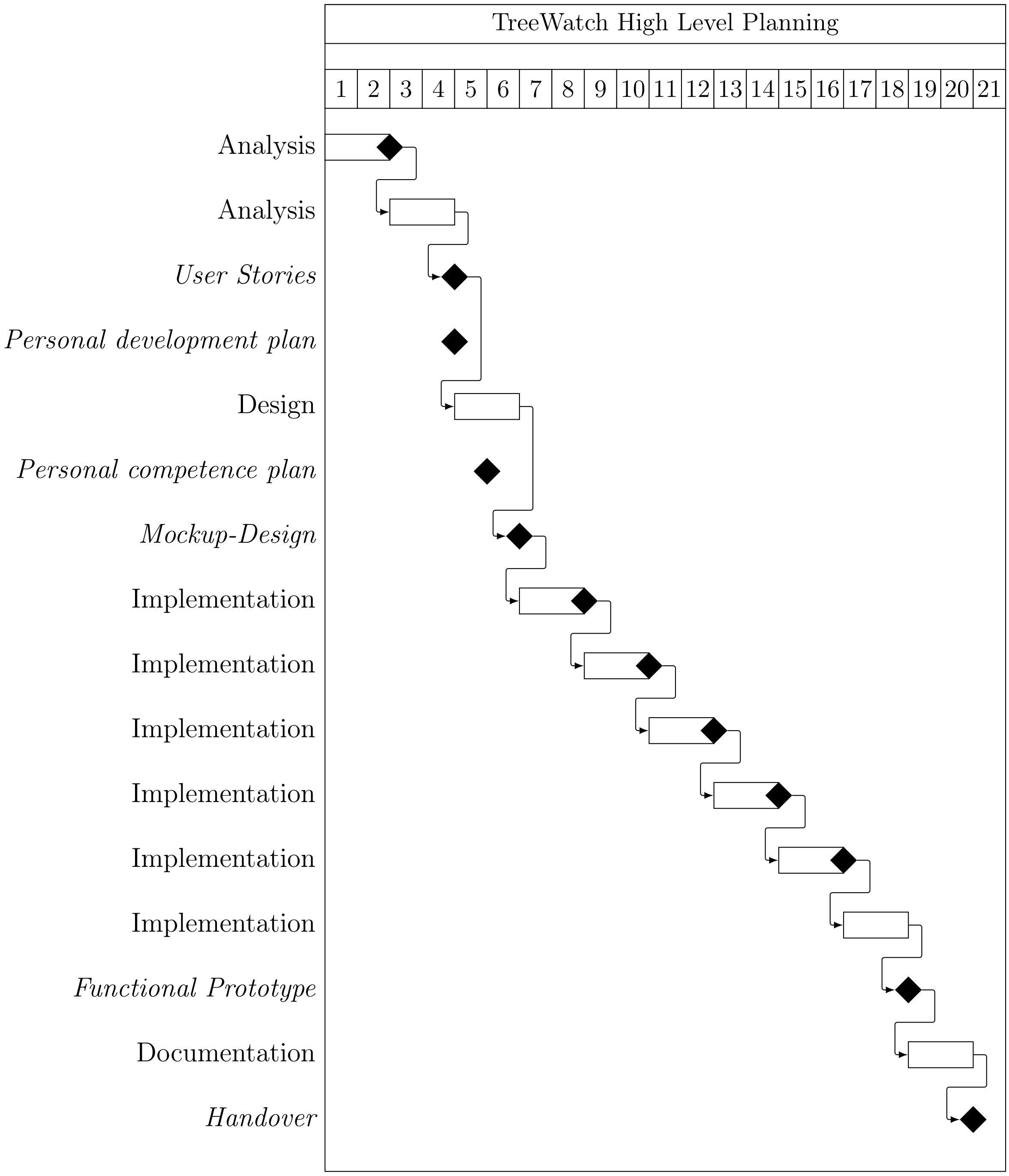


Figure 1: Milestones of the whole project

As shown in the figure 1, every two weeks there is a milestones. These milestones are the sprint meetings in which the customer gets to see the progress. Furthermore all the deliverables described in the chapter before are also milestones.

3.5 Projectplanning

As shown in in figure 1 the SoFa group had 20 weeks of working time, it was planned that one sprint is two weeks long. Furthermore it was planned that the analysis would take two sprints in which the project was set up, the user stories were defined and prioritised. The design phase was the planned to take only one sprint, since after a first design the group was able to start implementing anyways even if there were changes to the Graphical User Interface (GUI). The implementation, which was the main part of the project was planned to take six sprints and the documenting should take one sprint.

3.6 Risk

The following table contains the biggest risks the project will be facing. The severity and probability are defined for every known risk. These are graded on a scale of one to five with one having the lowest severity and five the highest. Also an action is defined which will be executed when a problem occurs. If the severity and probability are multiplied a multiplier is calculated. This multiplier helps determine which risks pose the most danger to the project.

ID	Risk	Severity	Probability	Multiplier	Action
1	Possible conflict of interest	4	4	16	Speak to involved parties to clear the problem and define direction of the project.
2	Xamarin licenses not available	2	4	8	Develop natively for both apps.
3	Customer not satisfied	4	2	8	Talk with the customer and define what the deficits are. Then communicate to VAA what needs to be changed.

3	No extra MacBooks available	2	3	6	Let the group members who do not have a MacBook only program android.

Table 2: Risks

4 Project analysis and prioritization

Content of this chapter is everything that happens in the beginning of the project. What contains talks between us and the company for which the project is. Furthermore, out thinking about the project in the beginning, the analysis of what the user wants to get at the end and final the mockup planning of the menu structure.

4.1 Planning and the desired results of the analysis phase

The analysis phase started on 31st August 2015 and proceeded over a period of 2 milestones that were each 2 weeks long. The end of the stage was therefore the 25th September 2015.

In the first milestone the requirements of the application were examined. From the results various Epics and User Stories have been created, which have been prioritized by our contact person in Fleuren Baarlo and accordingly brought into order.

The second milestone was launched with finding ways of development and the currently used systems. After the systems were determined, various guidelines had to be created to determine how to be developed. Following mock-ups are created to represent the structure of the surface.

4.2 Results and their discussion within the Milestones

The results of the first milestone are in Table 3 of the 5. Section. While the results of the second milestone, which have been expanded in the design phase, can be found in Section 6.

At the beginning of the project a discussion with Yannick Smedts took place, our future contact person within Fleuren Baarlo, about their wishes to the application being created.

During the next few days some questions arose that were answered and discussed during a further meeting, what was extended with a visit to Van Den Borne aardappelen.

Finally, a last talk was organized in Fleuren Baarlo to clarify the recent ambiguities and thus successfully complete the analysis phase.

The talks took place as follows:

31st August 2015 Fontys University of Applied Science Venlo

9th September 2015 Fleuren Baarlo headquarter and on a field

17th September 2015 Van Den Borne aardappelen

25th September 2015 Fleuren Baarlo headquarter

5 Requirements

The requirements definition was started by setting a meeting with Fleuren. Here Fleuren explained what their problem was and what they wanted to see as functionality of the app. Based on that meeting a set of user stories was created. These user stories were then grouped in epics. The epics were then sent back to Fleuren to have them prioritised and checked for completeness.

The checked and prioritised user stories were then put in a table to show which functionalities needed to be implemented first and which ones were optional. The table of user stories can be found below. Within the table the word system is to be defined as the combination of the App and the database conatining the data.

Priority	**ry**
1	*As a user*, I want the ability to overlay the visualizations on top of the field. *As a user*, I want to have my field represented graphically in the application.
2	*As a user*, I want the application to register as much data as possible, automatically. *As a user*, I want my GPS data entered into the system. *As a user*, I want my weather data entered into the system. *As a user*, I want my soil data gathered into the system. *As a user*, I want my system data entered into the system. *As a user*, I want to be able to take pictures of the trees. *As a user*, I want to be able to upload pictures and tag them to a tree inside a certain row/block. *As a user*, I want a system which visualises the available data for the fields/blocks/rows/trees. *As a user*, I want to correlate the available data in different views so that I can see the relations between different datasets.
3	*As a user*, I want to digitise the kwekerij schrift (Nursery Script).

4	*As a user*, I want the application to contain different forms to transmit different information about the actual state of the trees on the field (e.g. brown leaves, thin stems...). *As a user*, I want to be able to see a chronological ordering of events that happened on trees in a certain row/block's history. *As a user*, I want to be able to see where and how long I worked on a field. *As a user*, I want to be able to specify what work I am doing on a field/block/row/tree.
5	*As a user*, I want the application to analyse the data of the trees and give hints and solutions for problems.
6	*As a user*, I want to have a system that manages my customers, their orders and the amount of trees that are growing or are fully grown for them.

Table 3: Priority & User-story

6 Design

This chapter is about the design of the TreeWatch project. The design part is necessary to give the customer an idea of what the application will look like and so that the customer is able to give feedback. The design also gives the group members an idea of what the application should look like when it is finished. This way the group members can work towards a visual goal.

First the user should sign in, so that data is only visible to an authenticated user, as can be seen in figure 2.

The main goal of the application is to show all the fields on a map, with overlays of certain data. This is visualised in figure 4 to figure 6.

Fleuren asked to have what they call "What are you coming to do here" as part of the application. As soon as an user enters a field the app will show a screen with the question "What are you coming to do" as seen in figure 3.

Finally there are three more screens, namely the to do list in figure 7, the history of tasks screen in figure 8 and lastly the settings screen in figure 9.

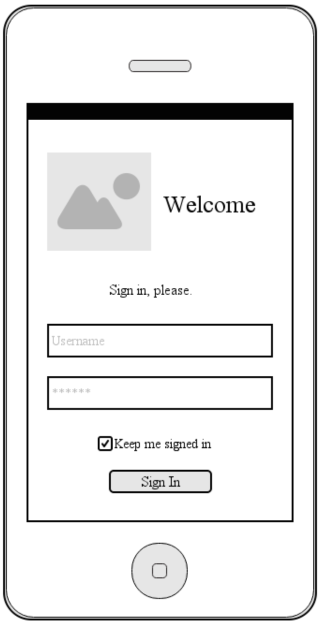


Figure 2: Sign in screen

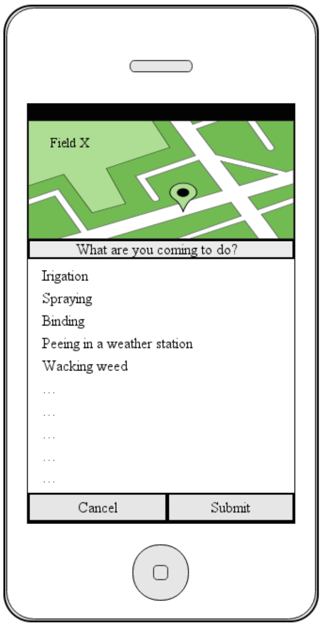


Figure 3: What are you coming to do screen

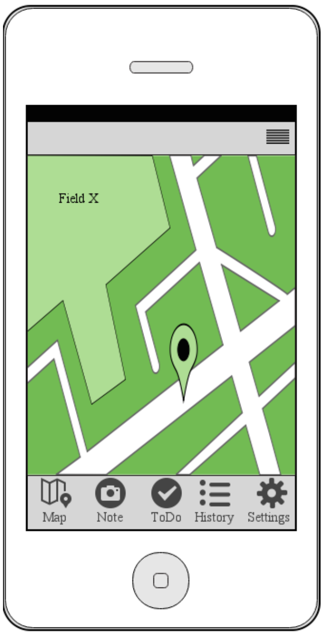


Figure 4: Default screen with the map

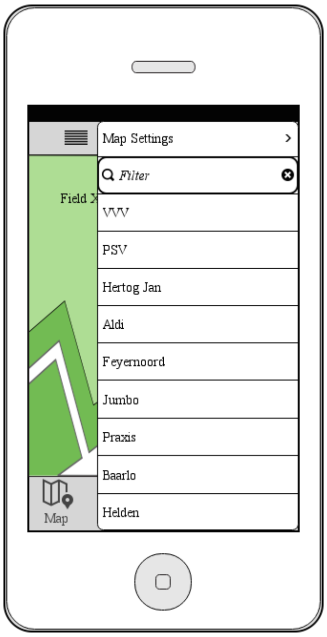


Figure 5: Map screen with menu open

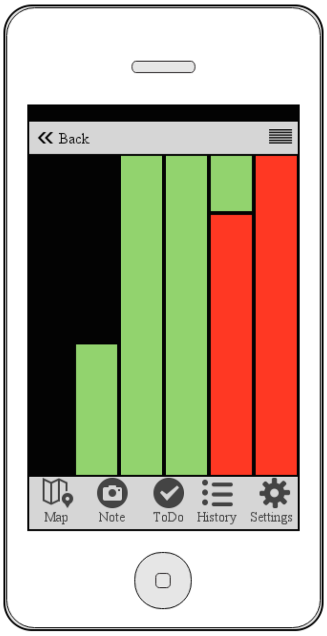
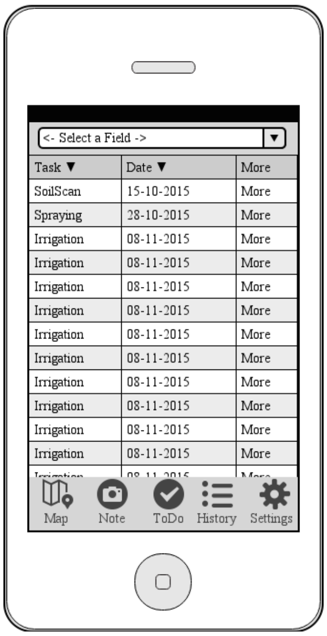
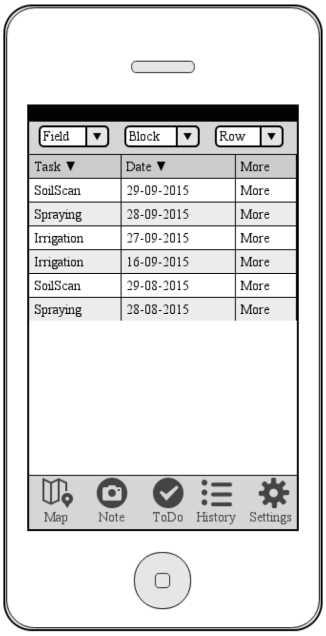


Figure 6: Map screen with an overlay



Task ▼	Date ▼	More
SoilScan	15-10-2015	More
Spraying	28-10-2015	More
Irrigation	08-11-2015	More
Irrigation	08-11-2015	More
Irrigation	08-11-2015	More
Irrigation	08-11-2015	More
Irrigation	08-11-2015	More
Irrigation	08-11-2015	More
Irrigation	08-11-2015	More
Irrigation	08-11-2015	More
Irrigation	08-11-2015	More
Irrigation	08-11-2015	More
Irrigation	08-11-2015	More

Figure 7: Screen where you can see to do list



Task ▼	Date ▼	More
SoilScan	29-09-2015	More
Spraying	28-09-2015	More
Irrigation	27-09-2015	More
Irrigation	16-09-2015	More
SoilScan	29-08-2015	More
Spraying	28-08-2015	More

Figure 8: Screen with the history of tasks

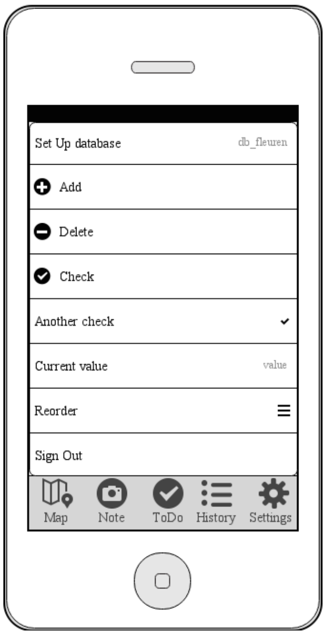


Figure 9: Screen where you can change settings

7 Implementation

This section gives you additional insight on part of the implementation and discusses challenges and problems encountered. Also gives information on the approaches to solve this issues.

7.1 Displaying a Map inside Xamarin Forms

One of the first things that was implemented in the app, after getting the basic structure of the app, was showing a map. The first implementation was really simple and just showed a map, like the default map app on a phone. Afterwards additional functionality was added.

Displaying a map in Xamarin Forms normally is really simple, as you can see on the Xamarin Forms Maps Xamarin-Inc. (2015a) site, using the Xamarin Forms map component is really simple. Just add a Nuget package to the project and then display a map with a few lines of code. This is fine if you don't need to extend it, but in our case we wanted to add additional functionality to it.

After determining that something more powerful was needed, different solutions to display the map were considered. Exiting solutions like OsmSharp SharpSoftware, which provided a good looking map based on open street map, were considered. But they were focused on routing and didn't offer any of the features our app needed like adding custom overlays to a map. Also there are different solutions available that offer extensive server side rendering of maps containing your own data. A popular solution is Mapbox MapBox (2015), but using a server backend for the app was out of scope for the project so a solution was chosen that would work on the device.

The only usable way we found was to implement our own custom map renderer on each platform. This renderer is overriding he default map renderer on each platform and then adds the custom functionality on top of it. This approach also meant that each feature related to the map needed to be implemented two times. Once for android and once for iOS, which mean an increase in development time. This solution worked well until we added extensive data to the map, when performance hits were noticeable.

7.2 Showing Overlays on the Map

After the map was implemented the next challenge we tackled was showing an overlay of the fields on the map. A field consist of two different parts, the outer boundary of the field and also many blocks on the field. The blocks represent six rows of Trees of a specific species. But both are represented on the map as polygons, consisting of the points defining the outer boundaries of the field or block. All blocks of the same species should be easily recognisable so they are shown in the same color.

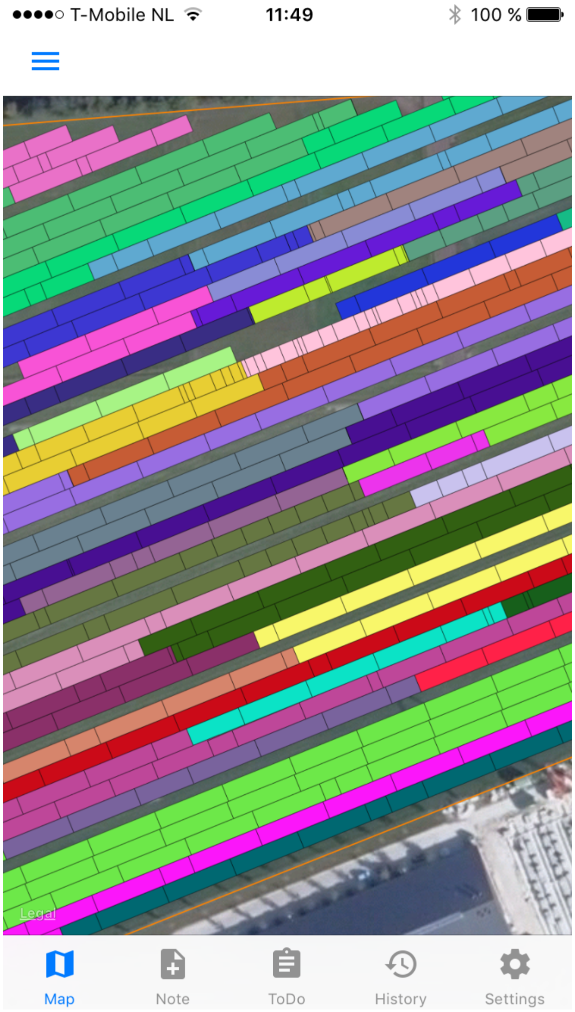


Figure 10: Block overlay of field Grotto

As you can see in figure 10 a field consists of a lot individual blocks, the amount depends on the size of the field but is somewhere in the several hundreds. This created some serious performance issues for the map. The map gets really unresponsive and slow after adding so many polygons to it. To reduce the stress on the map on iOS we added our own polygon implementation that could contain many single polygons. This reduced the work needed by the map enormously, since the polygons for one field are drawn in one single draw and not every polygon on his own. This made the map usable again but also introduce a delay when the polygons need be shown by the map. But this is the only working workaround for the performance problems. Further performance improvements could only be achieved by moving the map drawing to a server and only getting pre rendered map tiles to display from it.

7.3 Creating and Displaying HeatMaps

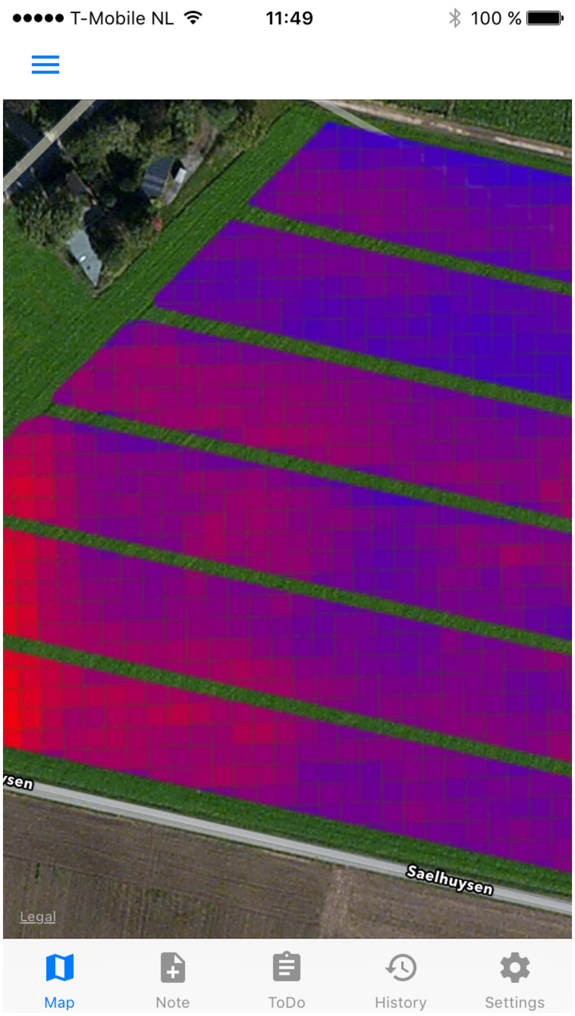


Figure 11: Heatmap inside of the iOS app

Another feature that was implemented for the map was the display of heat maps, these should show data like a soil scan that is not bound to specific blocks but more to specific points on the field.

The first try of implementing heat maps included displaying an image overlays created from the data and putting that on top of the map. For this approach we found a working Objective C Library called LFHeatMap Polak (2015). To use it in our project we created a C# Binding for it. This was really easy to do following the tutorial on the Xamarin site Xamarin-Inc. (2015b). But this created heat maps based on single weighted points. Since we only got heat map data from Fleuren that contained of polygons with data, we looked for another solution.

As you can see in figure 11 we implemented the heat map as just a bunch of polygons. The colors of the polygons were calculated based on their data. The color moves from solid blue for the lowest values to solid red for the highest values. The heat map displayed consists of about 4000 different polygons. The approach therefore is more a proof of concept since the drawing on the map takes several seconds and doesn't even work on android.

7.4 Importing KML files into the app

Fleuren supplied us with various datasets from their fields, to test our implementation with real data, especially the amount of data, we implemented a mechanism to read these files. The data we got was mostly shapefiles, since there is no easy way to read these we converted them to Keyhole Markup Language (KML) files. KML is basically just a version of Extensible Markup Language (XML) which is designed for geospatial data. To convert the files we used

7.5 Geofencing

The Geofencing part of the application is based on the plugin which can be found at `https://github.com/domaven/xamarin-plugins/tree/master/Geofence`. This plugin has most of the functionality that is required in the project, however the plugin is outdated and does't work anymore.
To make the most out of Xamarin, most of the work should be done in the Forms application, this way code is reused for both Android and iOS. The class diagram on the next page shows what classes are in the Forms application.
The IGeofence and IGeofenceStore are interfaces that should be implemented in a platform specific way, because they contain the actual geofences and their platform specific way of working. The IGeofenceStore interface has some methods that can be shared between the platforms, therefor an abstract class BaseGeofenceStore is created, that has these methods already implemented.

The CrossGeofence class is responsible for creating the actual platform specific implementation of the IGeofence interface. This is done by using DependencyService to get the actual platform specific implementation.
The platform specific code calls the methods in the CrossGeofenceListener class inside the Forms project, this way state changes can be handled in a universal way, instead of platform specific.

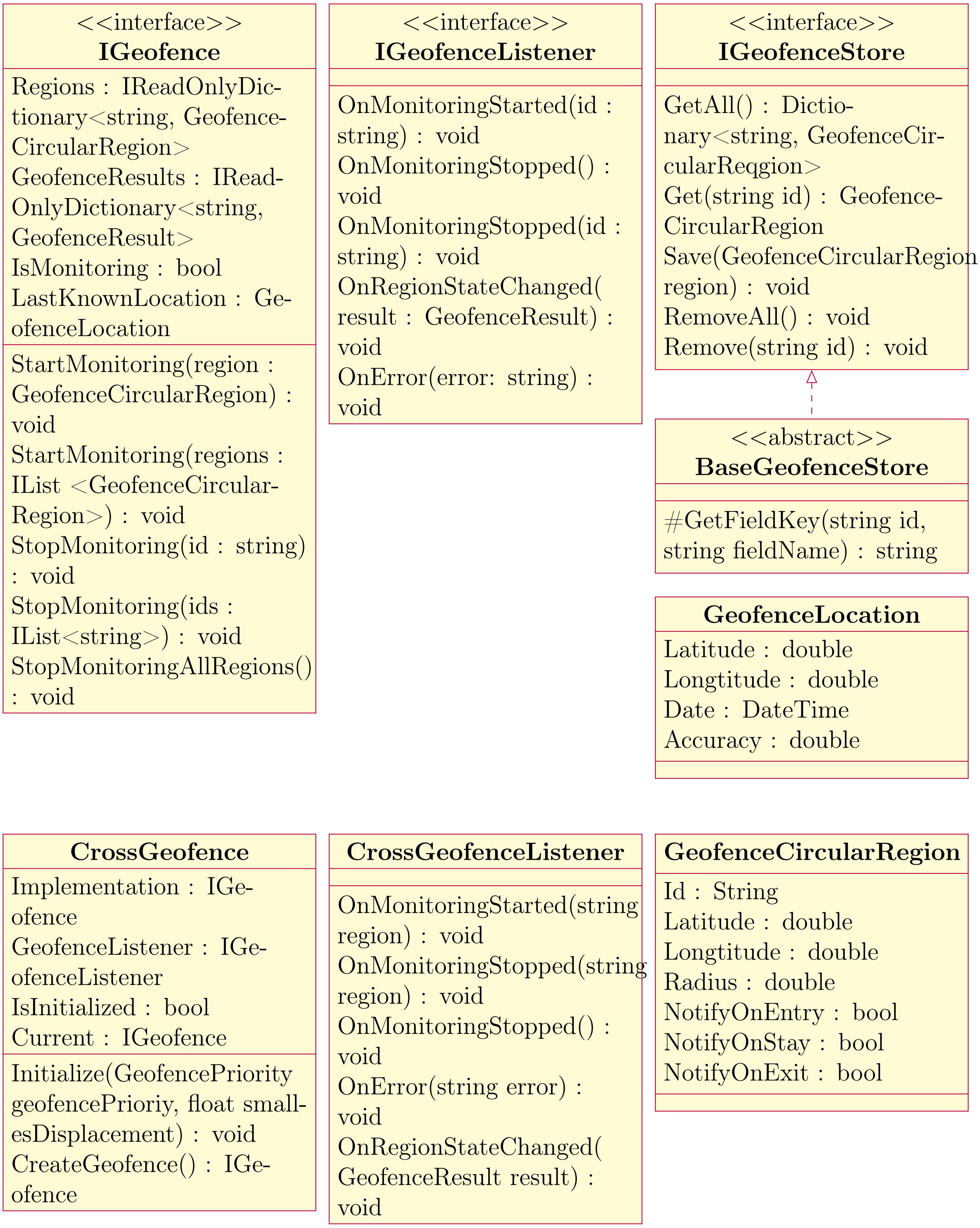

<<interface>>
IGeofence
Regions : IReadOnlyDictionary<string, GeofenceCircularRegion>
GeofenceResults : IReadOnlyDictionary<string, GeofenceResult>
IsMonitoring : bool
LastKnownLocation : GeofenceLocation
StartMonitoring(region : GeofenceCircularRegion) : void
StartMonitoring(regions : IList <GeofenceCircularRegion>) : void
StopMonitoring(id : string) : void
StopMonitoring(ids : IList<string>) : void
StopMonitoringAllRegions() : void
<<interface>>
IGeofenceListener
OnMonitoringStarted(id : string) : void
OnMonitoringStopped() : void
OnMonitoringStopped(id : string) : void
OnRegionStateChanged(result : GeofenceResult) : void
OnError(error: string) : void
<<interface>>
IGeofenceStore
GetAll() : Dictionary<string, GeofenceCircularReqgion>
Get(string id) : GeofenceCircularRegion
Save(GeofenceCircularRegion region) : void
RemoveAll() : void
Remove(string id) : void
<<abstract>>
BaseGeofenceStore
#GetFieldKey(string id, string fieldName) : string
GeofenceLocation
Latitude : double
Longtitude : double
Date : DateTime
Accuracy : double
CrossGeofence
Implementation : IGeofence
GeofenceListener : IGeofenceListener
IsInitialized : bool
Current : IGeofence
Initialize(GeofencePriority geofencePrioriy, float smallesDisplacement) : void
CreateGeofence() : IGeofence
CrossGeofenceListener
OnMonitoringStarted(string region) : void
OnMonitoringStopped(string region) : void
OnMonitoringStopped() : void
OnError(string error) : void
OnRegionStateChanged(GeofenceResult result) : void
GeofenceCircularRegion
Id : String
Latitude : double
Longtitude : double
Radius : double
NotifyOnEntry : bool
NotifyOnStay : bool
NotifyOnExit : bool

Figure 12: Geofencing class diagram

8 Quality management

This chapter describes the necessary information needed to manage the project quality from the project planning to the delivery to the customer. Within this document the quality policies, procedures, roles, responsibilities and authorities are defined.

At the highest level quality management involves planning, doing, checking, and acting to improve project quality standards. Project quality management is therefore split into three categories: quality planning, quality assurance and quality control.

8.1 Organization and Responsibilities

Name	Role	Quality Responsibility
Max van der Linden	Project Manager	External communication, Auditing
Martijn Bonajo	Configuration Manager	Infrastructure, source files, Software engineering documentation, auditing
Ron Gebauer	Scrum Master	Scrum planning, auditing
Rene Karoff	Quality Manager	Ensure use of quality guidelines, auditing
Jan Kerkenhoff	Main Engineer	Main Code Master, auditing

Table 4: Group roles

8.2 Quality Planning

Since this project is about programming a mobile business app, it is highly important that the system has a minimum of bugs/errors. A good documentation is necessary too, not only because this app will be developed by another SoFa group after our semester is over, but the customer should understand the system which is developed by us.

8.2.1 Define Project Quality

To ensure to quality of the written code it is mandatory to develop every module test-driven, therefore unit-testing is introduced. The written documents will be checked for its grammar, writing style and content.

8.2.2 Measure Project Quality

Most of the code of this app was tested, but since we have a lot of GUI related code which we weren't able to run unit tests with, we were not able to achieve the former stated goal of a code coverage of 90%.

Furthermore the GUI was tested using the Xamarin Testcloud which was included in our Academic license which represents the Xamarin Business license. Since we had the Business license, we were allowed to use the Xamarin Testcloud for one hour per month. But since we had 5 licenses this expanded to 5 hours per month.

In the following table it is described who reviewed which document and with what rank it was approved.

Document name	Author
User Stories	Max van der
Personal development plan Personal competence plan Mockup Design	Group work
Functional Prototype	Handover

The code of the app will be tested. Therefore it is a goal to get at least 90% code coverage, better 100%. Each evaluation criterion of the written documents will be ranked from “–“ to “++” (“–“ too bad – “++” excellent).

8.3 Quality Assurance

Since the code is tested, this will show the group if the quality goal is achieved. Furthermore the written documents will be audited by at least one group member and the project quality manager.

8.3.1 Analyze Project Quality

The tool to measure the code coverage **(which still needs to be defined)** will show the programmer which code is still uncovered, and the writer of a document will receive feedback of the auditing persons, so he can improve on his writing too.

8.3.2 Improve Project Quality

As stated in the project management plan, proper requirements were created to increase the project quality. This was done during two sprints, where the customer received the user stories, prioritized them and gave feedback to the group in the first sprint. During the 2nd sprint the user stories were reworked due to the remarks the customer gave to the group.

8.4 Quality Control

At the end of the project each deliverable will be audited again, by every team member to ensure the project quality.

9 Conclusion

The TreeWatch app has come a long way but still has a long way to go. A lot of the basic functionality has been implemented. The app can already be used to visualize where all the fields are located on the map and is also able to show overlays such as the block data and biomass.

VAA ICT Consultancy will need to continue the development of the app in order to make it practically usable. This means that a web based database needs to be added which contains all the info. For now the information about the fields is still stored locally on the device. Furthermore also the history and todo parts of the app should be implemented.

10 Advice

Because of performance issues that occur when the app has to calculate the overlays of the fields and the blocks, it is recomended that the server should draw all the polygons and then send it as an image to the phone. Then the phone should be able to overlay this image on the correct position. This would greatly improve the speed of the app.

11 Reflection

11.1 Max van der Linden

As the project leader of the SoFa TreeWatch group I learned a lot about what it means to lead a project. This meant that i had to make sure that everyone follows the rules and sticks to the deadlines. One of the tasks of the project leader was also to be the main contacting point to and from the customer. From this i learned a lot about communicating in a professional way.

All in all in my opinion a good and useful product form which a lot was learned.

11.2 Martijn Bonajo

I learned a lot from this project. For me it was the first time creating a multiplatform application. This meant learning how to work with Xamarin, using MVVM on the Xamarin Forms and programming to platform specific Application program interface (API)s.
As configuration manager I was responsible for making sure everyone could run all the software that was needed during the project.

11.3 Rene Karoff

11.4 Jan Kerkenhoff

11.5 Ron Gebauer

References

Xamarin-Inc. (2015) Xamarin forms maps developer guides. [Online]. Available: https://developer.xamarin.com/guides/cross-platform/xamarin-forms/user-interface/map/ Accessed: 2015-12-18

SharpSoftware. Osmsharp website. [Online]. Available: http://www.osmsharp.com/

MapBox. (2015) Mapbox website. [Online]. Available: https://www.mapbox.com/ Accessed: 2015-12-18

G. Polak. (2015, may) Lfheatmaps heatmap generator library. [Online]. Available: https://github.com/gpolak/LFHeatMap Accessed: 2015-12-18

Xamarin-Inc. (2015) Walkthrough: Binding an objective-c library. [Online]. Available: https://developer.xamarin.com/guides/ios/advanced_topics/binding_objective-c/Walkthrough_Binding_objective-c_library/ Accessed: 2015-12-18