T-404-LOKA, Final Project

Mobile Store Manager

Árni Freyr Þorsteinsson
Björn Halldór Helgason
Guðjón Geir Jónsson
Steinar Ágúst Steinarsson

Spring 2016
B.Sc Computer Science

Instructor: Birgir Kristmannsson
Examiner: Skúli Arnlaugsson

Abstract

In this report we discuss the final project in Computer Science at Reykjavik University titled "Mobile Store Manager" which was done in collaboration with LS Retail. We created a generic solution for developing smart device apps for the clients of LS Retail.

Review

In the end of the year 2015 LS Retail had business requirements from customers and prospects that needed easy access for their employees to certain functionality in their ERP systems. The requirements were different but had a common factor that they would be a fit which was solved with a simple registration on an App on a tablet/smartphone.

We came up with the idea to build a general solution where we could in each case define the interface to the ERP system as well as the GUI and generate an App that would run on tablets and solve the task – e.g. the registration of a new item in Dynamics Ax.

To realize this idea we had a group of HR students that worked on the task from the beginning of the year. As the project was not fully defined it demanded that the team did a lot of research and use newest technology as well as using careful planning to be able to solve the task in the given time.

The work with the group went very well and now 4 months later we have a working solution that has exceeded our expectations of what could be done in this timeframe. We had set the goal on a solution that could be run on one mobile platform but the solution now runs on two platforms. The solution also has more advanced functionality in defining the interfaces as well as defining the UI in the App. We have already taken a prototype of the system to our customers and have received good feedback from them which ensures us that the architecture of the solution is the right one going forward. The work with HR group was a pleasure for us and they demonstrated both skills and professionalism in all of their work.

Ólafur Jónsson
Product Owner
LS Retail

Contents

1 Introduction

LS Retail is a leading developer in software solutions for retail, hospitality and forecourt businesses worldwide. Their solutions are currently used by retailers and restaurateurs across 120 countries. Their main products are built on top of Business Solutions from Microsoft, mainly Dynamics Nav and Dynamics AX.

Today LS Retail offers their clients mobile solutions using their product LS OMNI. Because LS Retails provides services to many clients, whose needs and requirements for a mobile application can be vary different from one another, each implementation and customization can be very time consuming for the developers of LS Retail.
For our final project we set out to create a base solution to a smart device app that can be easily customized and deployed for the clients of LS Retail. The main stakeholders of our project are not the clients of LS Retail but the developers of LS Retail. Our solution provides them with a simplified process of providing mobile services to their clients. The main goal of our project was to offer a framework that would minimize the work needed to supply a mobile solution that is fully adapted to the client's needs.

The services LS Retail provide their customers include, for example, inventory management for retailers and forecourt solutions for petrol stations. Store and petrol station managers can monitor and maintain inventory and status on machinery for petrol stations.

Our approach simplifies this procedure significantly. Various custom operations can be created specifically for each customer such as registering a new product, updating status on machinery or getting a list of available products. Through a web-based dashboard a LS Retail developer can create, edit or delete these operations on the fly which are then published to a client's mobile application on his or her device. We have currently implemented applications for both Windows and Android mobile devices. Future implementations include iOS devices.

2 Mobile Store Manager

2.1 Enterprise Resource Planning - Dynamics AX & Dynamics Nav

Enterprise resource planning (ERP) is a category of business management software, that an organisation can use to collect, store, manage and interpret various types of data, for example product planning, sales and inventory management.

As mentioned before, LS Retail mainly builds their solutions on Microsoft Dynamics AX and Dynamics Nav both of which are highly customizable ERP solutions available in many languages and currencies and offer industry specific functionality. Nav is generally geared for small to midsize companies and AX for large, enterprise organizations.

2.2 Overview

The app is a portal to services offered by LS Retail for the end-user. When a user logs into the app he receives the operations he can execute on the fly.

For the app to be able to communicate with any ERP service, a web server was set up as a middle layer. What this server does is provide authentication against the ERP service it's connected to, send the app the operations a given user has access to and forward these operations to the ERP service when the user executes them.

The developers dashboard is a small web application to simplify the process of creating, editing and deleting operations for the client. When a developer has created an endpoint for an operation in the ERP service, he can then easily specify how that operation is supposed to look inside the app.

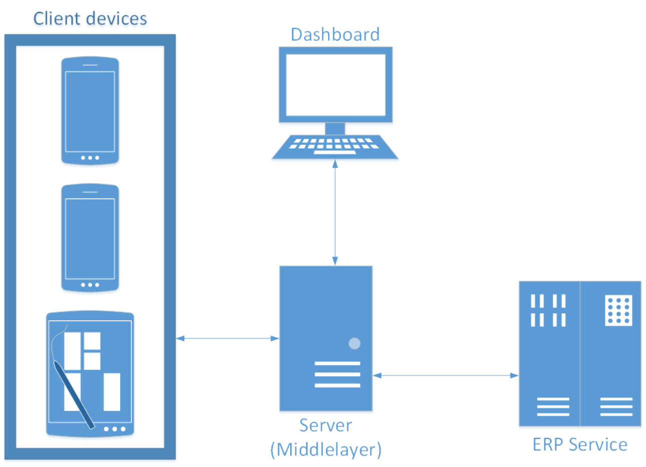


Figure 1: Overview

2.3 Functionality

In this chapter we will go over a general workflow for a ERP developer using our solution. The core feature of creating operation which will then be automatically deployed on mobile devices related to this instance of the solution.

Login

The first task a user would do, apart from creating the correct ERP endpoints which the operation about to be created should talk to, is to login to the dashboard.

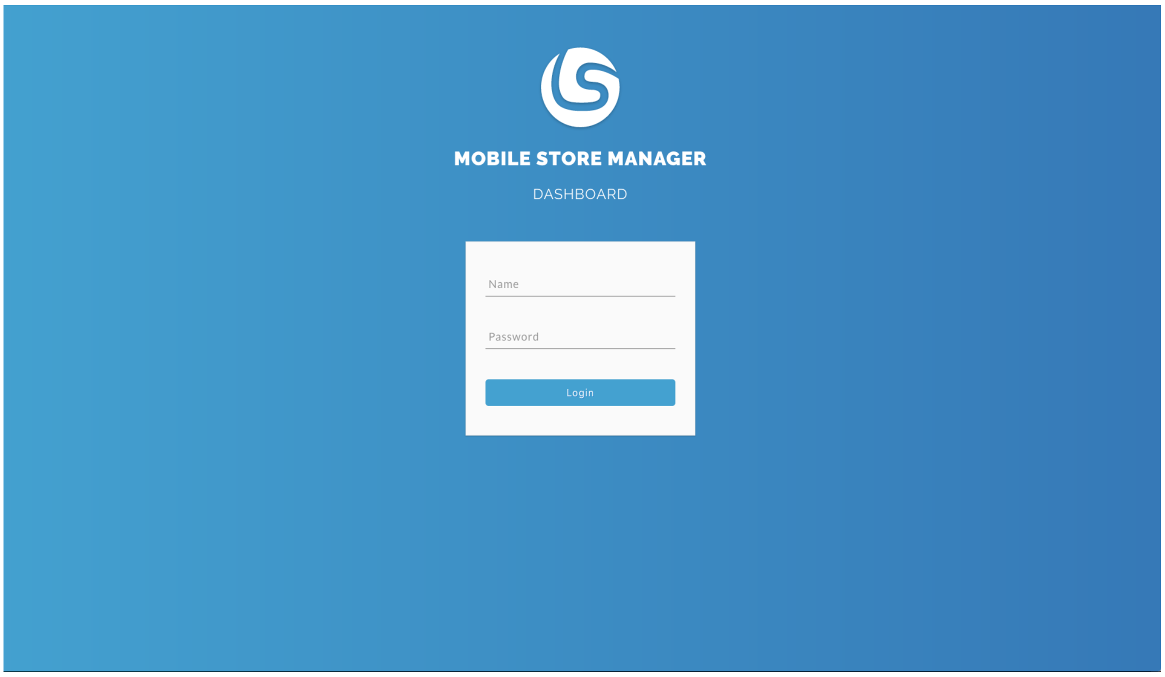
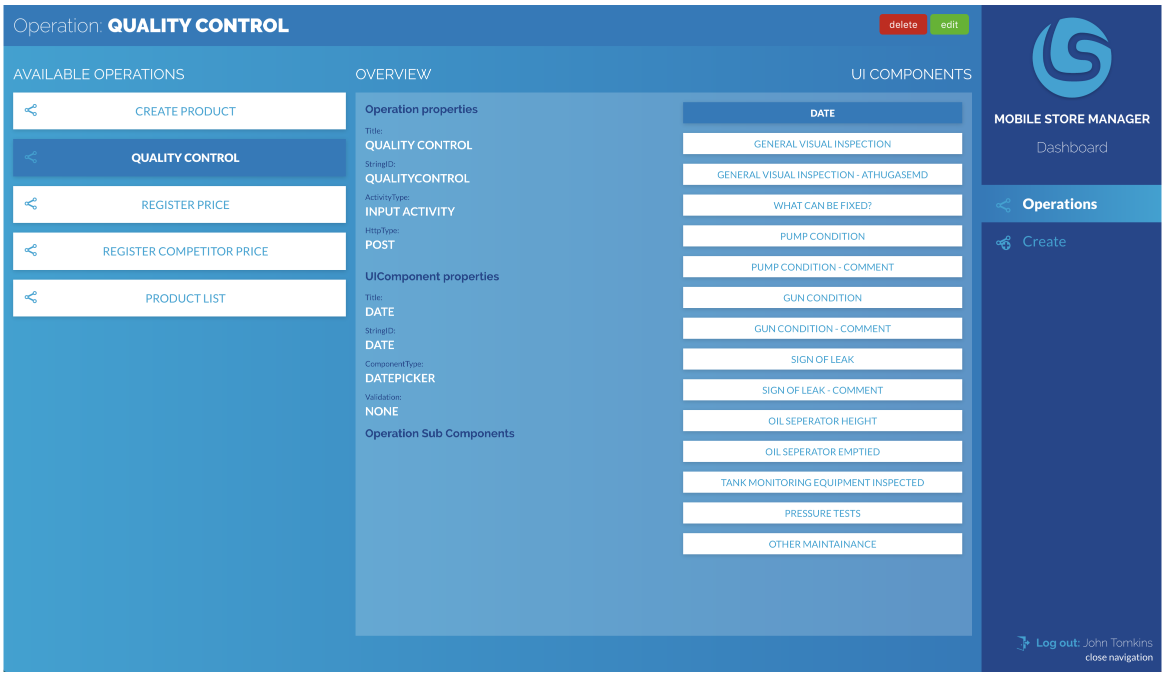


Figure 2: Login screen

After a successful login the user is redirected to our dashboard home screen. In this view the user can see all operations currently available in the system.



Operation properties

Title:
QUALITY CONTROL

StringID:
QUALITYCONTROL

ActivityType:
INPUT ACTIVITY

HttpType:
POST

UIComponent properties

Title:
DATE

StringID:
DATE

ComponentType:
DATEPICKER

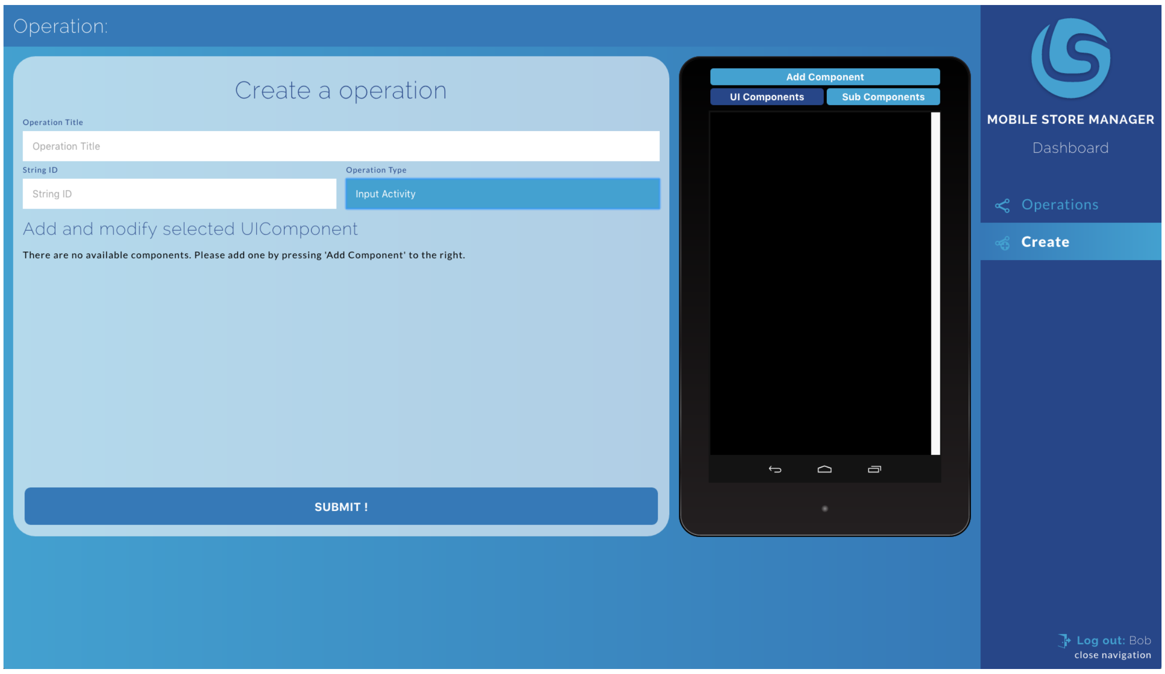
Validation:
NONE

Operation Sub Components

Figure 3: Home screen

Create an operation

When creating a new operation the user should press the Create button located on the right hand side of the screen, in the navigation section. A redirect to the create view occurs.



Operation Title: Operation Title

String ID: String ID

Operation Type: Input Activity

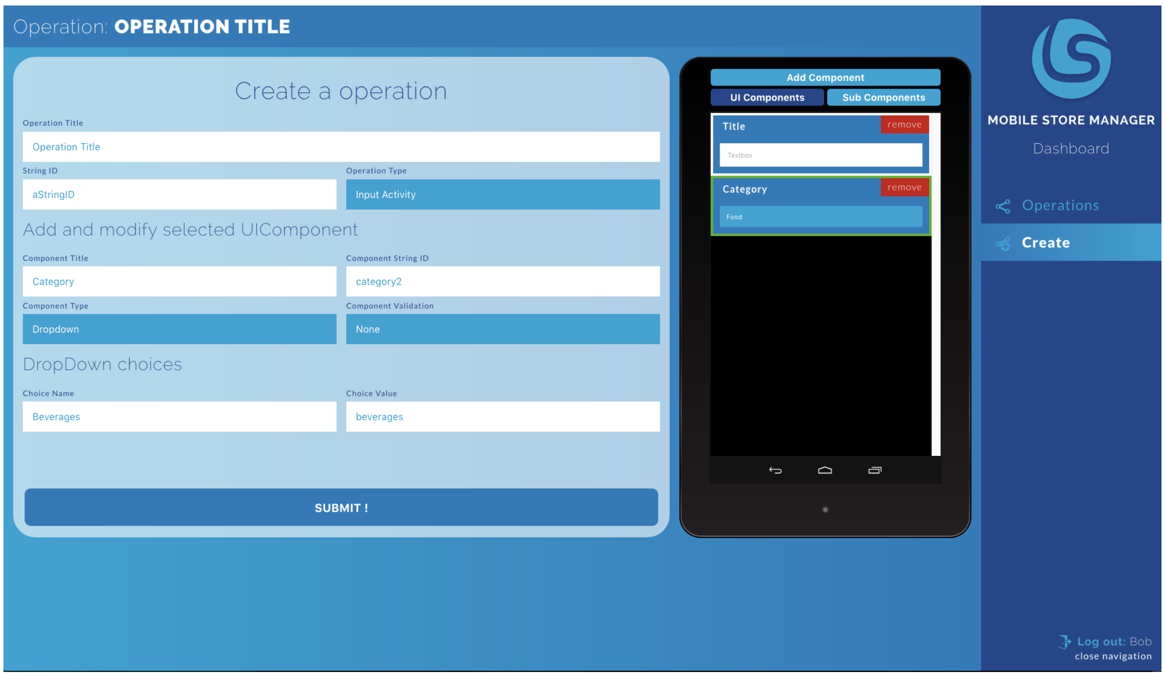
There are no available components. Please add one by pressing 'Add Component' to the right.

Figure 4: Creating an operation

From here the user chooses which kind of activity he needs, currently only list activity and input activity is supported. After choosing input activity the user gives his operation a title and the correct stringID which he gets from the ERP endpoint. The create operation on related input fields are located in a form on the left side of the screen.

Components represent the input fields for the operation for example a textbox for a title, dropdown for category or datepicker for datetime.

All components appear on the phone on the right side of the screen. They can be selected and edited accordingly, re-order by dragging each component to its correct place.


Operation: **OPERATION TITLE**

Create a operation

Operation Title: Operation Title

String ID: aStringID

Operation Type: Input Activity

Add and modify selected UIComponent

Component Title: Category

Component String ID: category2

Component Type: Dropdown

Component Validation: None

DropDown choices

Choice Name: Beverages

Choice Value: beverages

Figure 5: Adding components to an operation

If an UI-Component needs Sub-Components such as options for a dropdown or checkboxes the user can create subsequent sub-components by pressing the Sub-Component button on top of the phone. By pressing the Add Sub-Component button a new Sub-Component has been created and can be edited on the left.

When the operation is ready the user can submit the operation by pressing the Submit button on the left. If all validations are met the user is redirected to the home screen and can now see the new operation.

Using the operation on a Mobile device

When a new operation has been created a mobile device running our solution can start using the operation by opening the app and input all required fields and submitting.

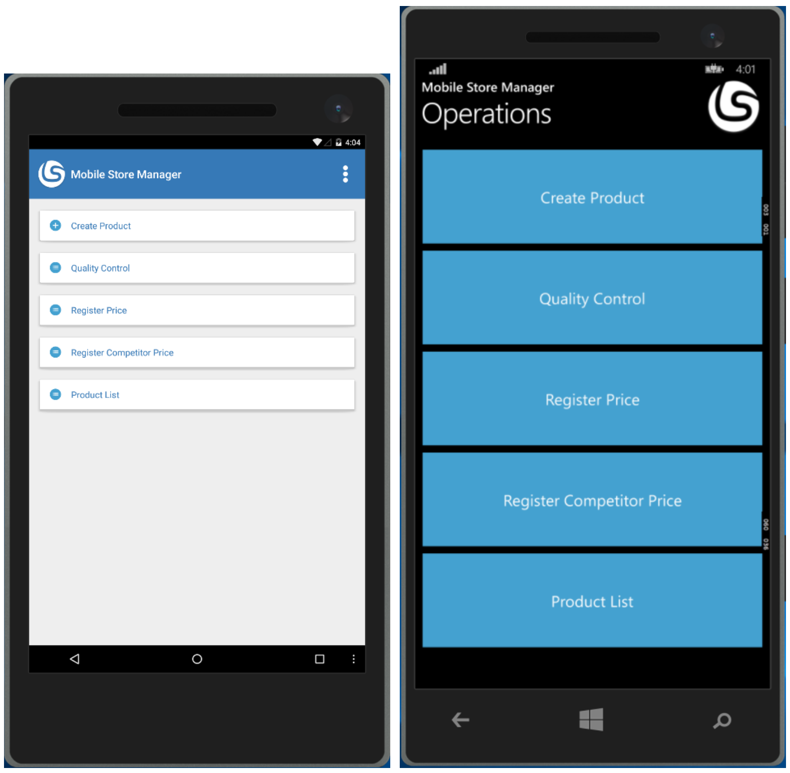


Figure 6: Operations on the apps

3 Design and architecture

3.1 Operations - App configuration file.

As described before, the app receives information from the server regarding what operations are available and how they are structured in what we call a configuration file. Each operation in this configuration file contains information needed to display and execute that information. If the operation requires user input, the operation also contains a list of the necessary input fields and their properties.

3.2 App

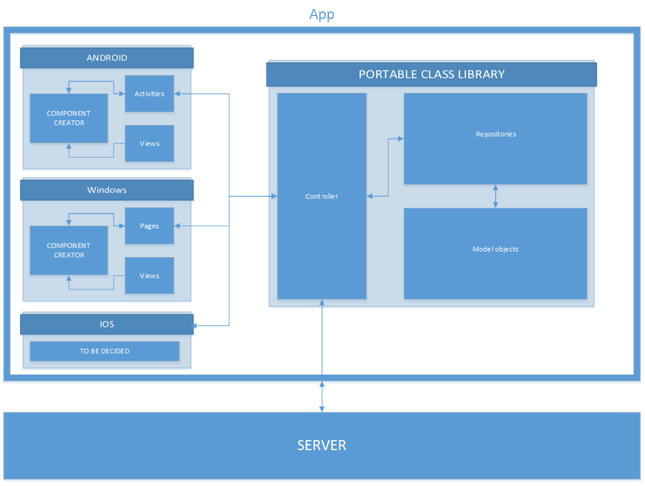


Figure 7: App architecture

For the app, we decided to use Xamarin which is a framework for developing cross-platform mobile apps. This has had some benefits for us, including a shared codebase and being able to use the same programming language for all platforms.

The app solution is split into four separate projects, three for the different mobile operating systems (Android, Windows Phone and iOS) and one portable class library which contains the shared code between each platform.

The portable class library (PCL) consists of Controllers, Repositories and Model Objects. Controllers communicate with the server.

Repositories are where all data is stored from requests that a controller has made to the server, Model objects are models for objects that are the same for every platform. The repositories uses these models to store objects that are acquired from the server.

The Android project consists of Activities, Component Creator and Views. Activities are classes that have similar purpose as Controllers have in the MVC(Model View Controller) pattern. That is, they load up data and render the view. The Component creator takes in a json string and makes a list of components(Views) to be rendered. The windows part is similar, but instead of Activities we have Pages, which are equivalent between platforms. The iOS project was decided to have the lowest priority and due to time constraints was not implemented.

3.3 Server

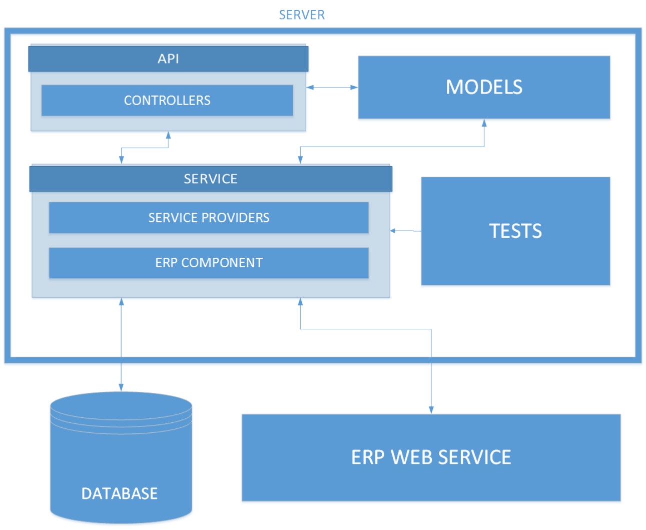


Figure 8: Server architecture

The server solution is broken into four parts, the main API layer, Service layer, Models and Tests.

The API layer handles the routing for the API and serving up the dashboard web app. The controllers in the API layer contain a reference to one or more service provider in the Service layer.

The Service layer contains all of the business logic, the database access and handles communication with an ERP web service. There are two service providers in the Service layer; the OperationServiceProvider that contains all the business logic that is independent of the ERP service the server communicates with, and the AuthenticationServiceProvider which contains business logic for authenticating and authorizing users. ERP specific business logic and communication to an ERP web service is handled in a class that implements an IERPComponent interface. This is to make it as simple as possible to set up an instance of this server for a specific client.

For database access we used Unit of Work and Repository pattern. This is implemented by having a generic IRepository interface, where one database table maps to one repository, and an IUnitOfWork interface, which job is to returning repositories for tables and committing changes to the database. Each service provider has access to a UnitOfWork which it uses to get access to the repositories it requires. The Unit of Work and Repository pattern allows us to easily provide mock Unit of Work implementations of the IUnitOfWork interface to the service providers when for unit tests.

All inbound and outbound data objects are defined in the Models project. These object are exposed by the API. Inbound data is received in ViewModel objects and outbound data is sent in DTO (Data Type Object) objects. The Models project also contains Enums that are used throughout the solution.

The Tests project contains unit tests which run against the Service Layer.

3.4 Dashboard

The developers dashboard is a angularjs single page web application, built with the gulp task runner. It's structured so controllers communicate with services and factories, directives manipulate the DOM and one controller per view.

The application uses a REST API so for each API resource a factory is defined and it's functions manipulate that resource. We use a data model that encapsulates a response from resource factories, to do so we created a factory that returns a object prototype that extends response properties and values. These data models have various function that are useful for boolean expression in html. Tests were written with Jasmine test framework and run by Karma test-runner.

For styling we used SASS, a CSS extension that offers better control over the style sheet.

We followed best practices like single responsibility and only directives manipulate the DOM.

3.5 UI Design

Straight from the beginning of the project we created a wireframe prototype of the mobile application. It was constructed in sketch and then made interactive using Invision which is a prototyping solution.

Since we were developing the application for both the Android and Windows platform we decided to maintain a platform specific design for each application. The Android applications design would be based on Google's Material Design standard which is the standard for modern Android applications. The Windows application would as well be based on Microsoft Windows 8/10 design standard.

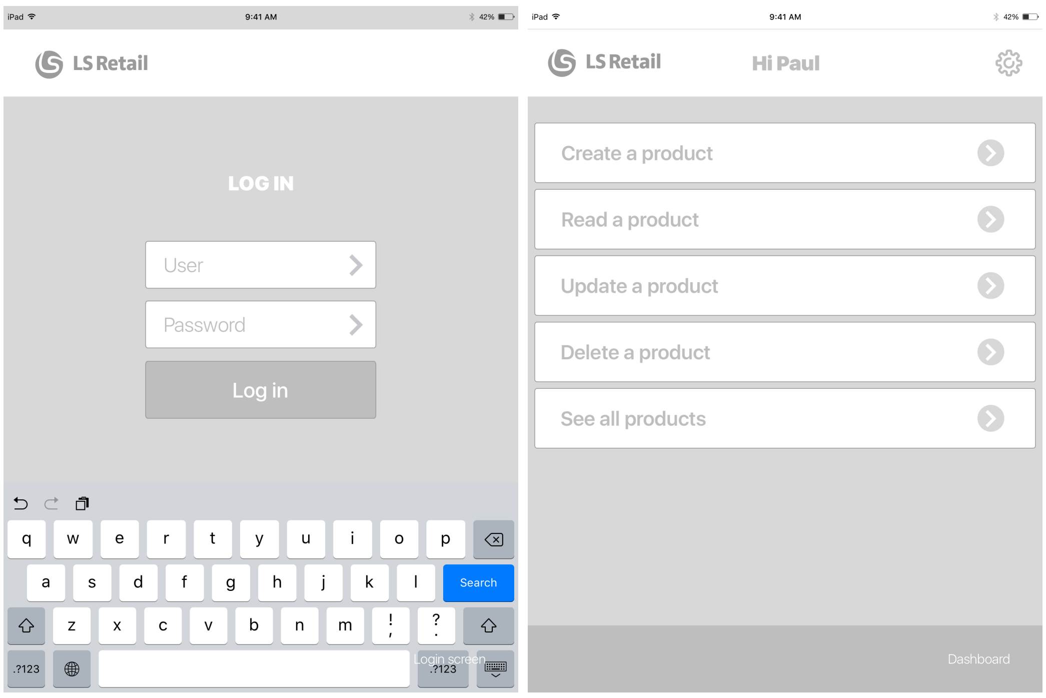


Figure 9: Initial wireframes for the app

4 Project Plan

4.1 Methodology

The methodology that we chose for this project was a relaxed form of Scrum. The main difference to pure Scrum was that we didn't use daily standups, which we didn't find necessary because the small size of the team and our facilities made it easy for everyone to know what everyone else was up to.

The Scrum master for the project was Björn Halldór and the product owner Ólafur Jónsson, who was our main contact in LS Retail.

User stories were put in the product backlog and divided into sections (A-C) and within the backlog they were prioritized by their importance. Story points were allocated to each story according to how difficult the team perceived implementation of them to be.

At the beginning of each sprint a meeting was held where stories were put into the sprint backlog and then broken down into tasks and given time estimates. Each team member chose tasks from the sprint backlog to work on. At the end of each sprint a retrospective meeting was held and tasks that weren't finished were put into the sprint backlog of the next sprint.

We used an on-line service called waffle.io as our scrum board. We connected Waffle.io to our version control which created tasks on a board for every issue created. The whole sprint backlog was put into one column. From that column we moved tasks that were not dependant on implementation of other functionality to the "Ready" column. Then team members either chose or were assigned tasks from that column. Once work on an issue commenced it was moved to the "In progress" column and then the "Done" column once finished.

4.2 Development environments and tools

- Visual Studio 2015 - For developing the app and the middle layer web service.
- Xamarin Studio - For developing the app.
- Microsoft SQL - Contains the operations for the app

- Angularjs - Front end logic for the dashboard
- NUnit - test framework used to test the android app.
- Karma - Test runner for the dashboard.
- Jasmine - Test framework used on the dashboard.
- Gulp - To build the dashboard.
- Git/GitHub - Version control and issue tracking
- Waffle.io - Scrum board that uses GitHub issues
- Slack - Inter-team communication
- Toggl - Time tracker

4.3 Programming rules

- Method parameters and local variables should be in camelCasing. Everything else (classes, properties, methods, etc.) should be in PascalCasing.

  ```
  public void Function(int someParameter) { ... }
  ```

- Declare variables using 'var' if possible.

  ```
  var number = 1;
  ```

- Avoid lines longer than 80 characters
- Indent using a tab character with the "hanging paragraph style".
- Curly braces should be placed in a new line

  ```
  if (somePredicate)
  {
      Func();
  }
  ```

- Comments should start with a double slash and should be in the line above what's being commented with same indentation.

  ```
  // This is a comment
  ```

- Decorate methods and classes with an xml documentation comment

  ```
  /// <summary>
  ///  This class performs an important function.
  /// </summary>
  public class MyClass { ... }
  ```

5 Progress overview

5.1 Sprints

We planned 9 sprints in total for our project. For the first 12 weeks of the semester each sprints duration was 2 weeks. During these sprints we planned on working for 4 full working days or 2 days each week, Monday and Tuesday. The last 3 sprints were only week long but we were working a full working week, Monday to Friday although we worked pretty much every day of the week on the project.

Table 1 shows the sprints. Their starting and end dates and names we gave them for easier communication about each sprint. The names are drawn from fantasy creatures.

Sprint number	Time period	Sprint name
Sprint 0	18. Jan - 31. Jan	Preparation
Sprint 1	1. Feb - 14. Feb	Sprite
Sprint 2	15. Feb - 28. Feb	Kelpie
Sprint 3	29. Feb - 13. Mar	Basilisk
Sprint 4	14. Mar - 27. Mar	Griffin
Sprint 5	28. Mar - 10. Apr	Sphinx
Sprint 6	25. Apr - 1. May	Phoenix
Sprint 7	2. May - 8. May	Thestral
Sprint 8	9. May - 13. May	Balrog

Table 1: Sprint overview

Sprint 0 - Preparation

During this sprint we did mostly preparation and tried to get familiar with the project at hand. We got setup at the LS Retail office and planned the next 15 weeks. Went through user stories from our product owner Ólafur Jónsson.

Sprint 1 - Sprite

This sprint was twofold, we used this sprint for setting up our development environment and IDEs. Created our project backlog and prepared for the first status meeting, finishing the required reports for said meeting (Progress report, Risk Analysis report, Design Report).

After the status meeting however we realized that the scope of our project had not really been structured well enough. Lack of information from LS Retail and a very vague project description resulted in a misdirection for our group. After the status meeting and a meeting with LS Retail we restructured and revamped our project backlog and got back on the right track. The following Sprint burndown chart shows this development.

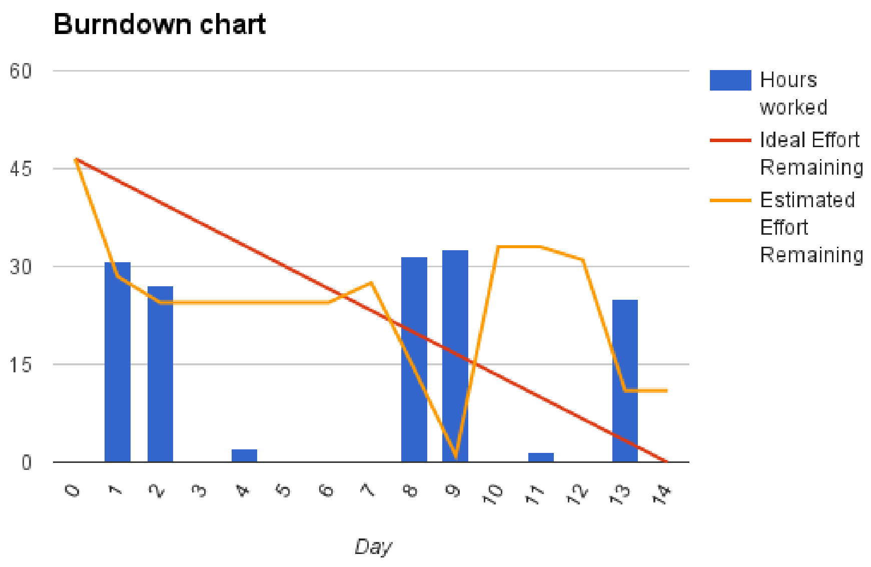


Figure 10: Sprint 1 burndown

Sprint 2 - Kelpie

During this sprint we set up the first version of our web api and implemented it using Continuous Integration through Azure. Scaffolded the Xamarin application. Created UI prototypes for an operation in the App. Finally implemented basic communication between app and web api.

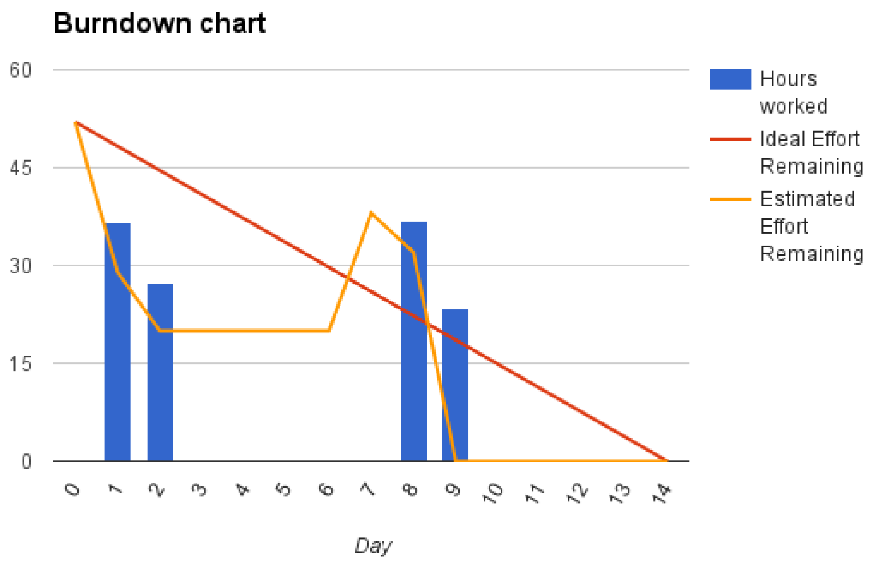


Figure 11: Sprint 2 burndown

Sprint 3 - Basilisk

During this sprint we started the Android implementation. The deliverables for the sprint were that our app should be able to perform an action on the server. An interactive prototype of the android app was created and shown to our product owner at LS Retail who approved the first iteration of said prototype.

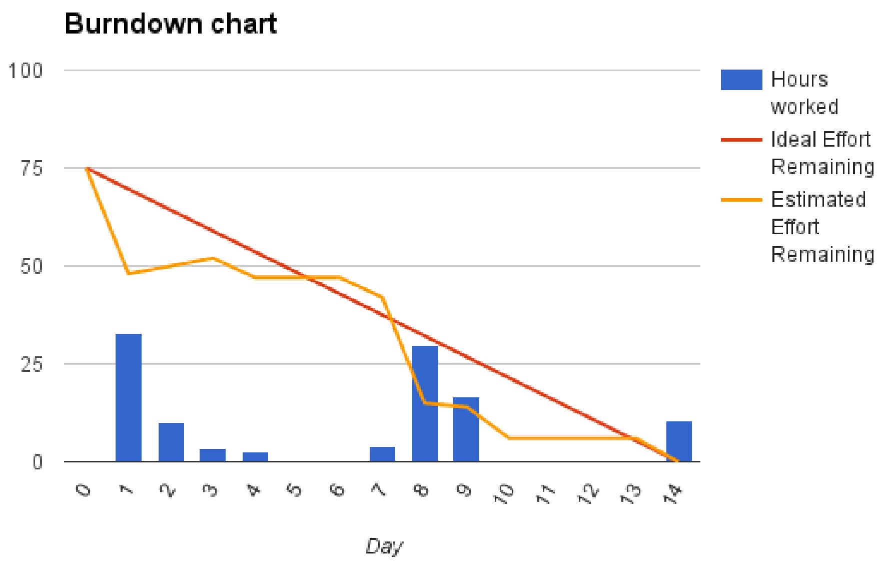


Figure 12: Sprint 3 burndown

Sprint 4 - Griffin

For this sprint we carried on with our implementation for android. Started styling the app according to the aforementioned prototype and prepared for our second status meeting.

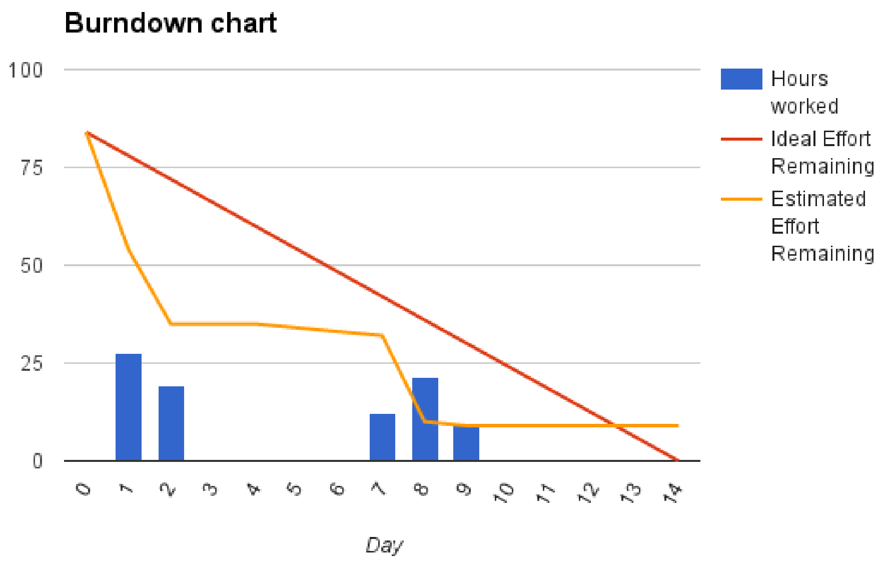


Figure 13: Sprint 4 burndown

Sprint 5 - Sphinx

This sprint was very pivotal for the progression of the project. The android implementation was completed for the most part. The plan for the sprint was to begin implementing the

Windows app but to our surprise we went very far in implementing all core features for the Windows app.

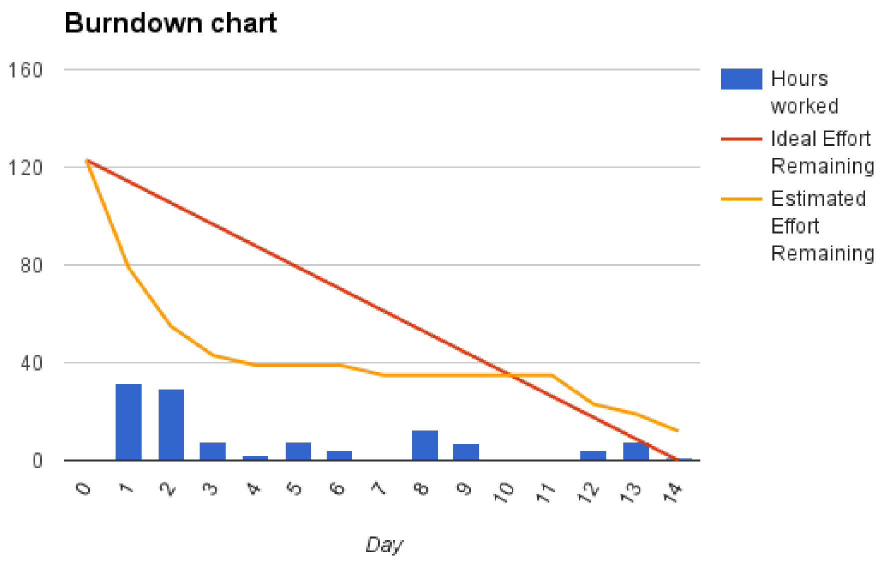


Figure 14: Sprint 5 burndown

Sprint 6 - Phoenix

The first sprint after 2 weeks off because of final exams. This sprints duration as mentioned before was 1 week or five working days. The deliverables for the sprint where finishing touches to the Android application. Look and feel, toast messages for success and error messages, client side and server side validation of input fields.

We also added to our product backlog a web based dashboard to view, create, edit or delete operations for the app. The core features for the dashboard were implemented during this sprint.

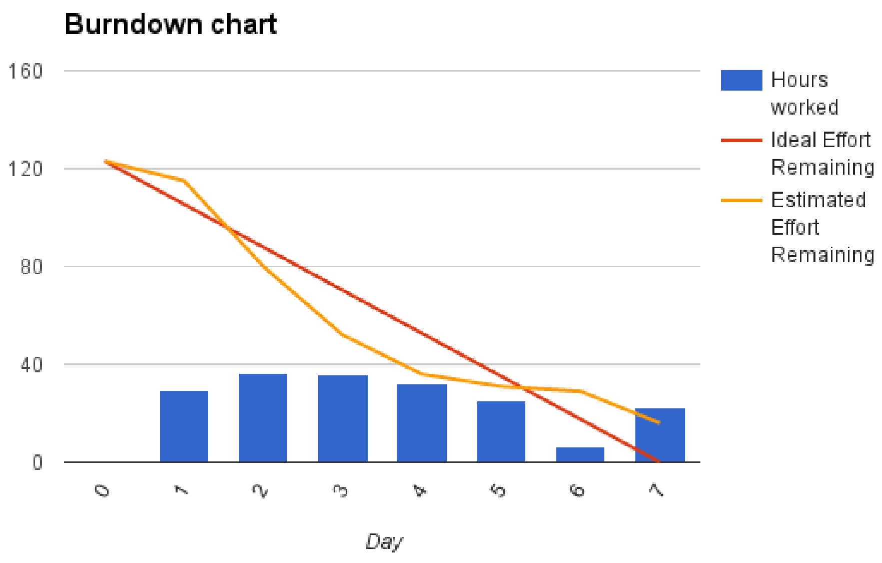


Figure 15: Sprint 6 burndown

Sprint 7 - Thestral

During this sprint the main focus was on finishing all reports including this report and preparation for the last status meeting.

The dashboard was styled and remaining features such as edit and delete an operation were finished. The Windows app went through some finishing touches. Validation and styling applied.

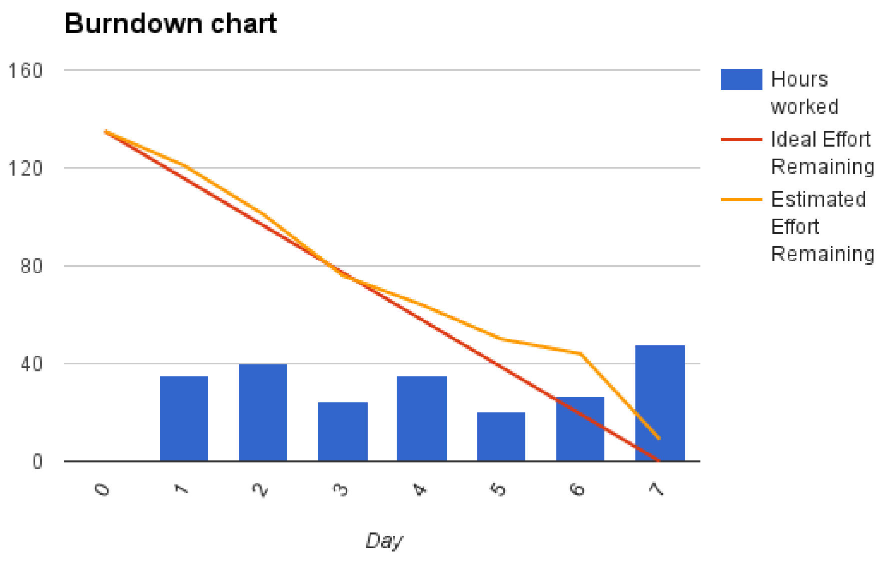


Figure 16: Sprint 7 burndown

Sprint 8 - Balrog

This was our final sprint and its main deliverables were to finalize all written reports and do code clean up. A feature freeze was conducted apart from one final task which was to finish authentication with our Web API. The dashboard and mobile platforms can now login and authenticate with our API.

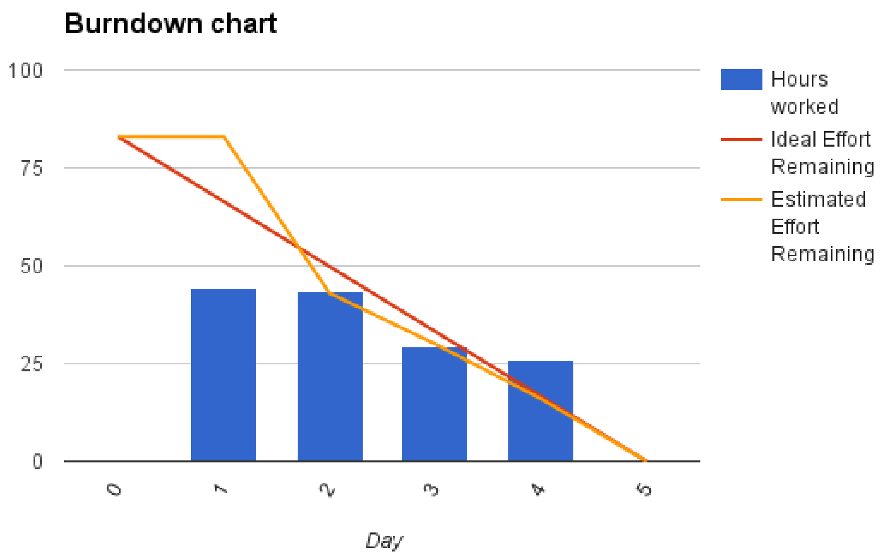


Figure 17: Sprint 8 burndown

5.2 Product burndown

When we started creating the product backlog the scope of the project had not been fully developed. The responsibility lay with both the LS Retail team as well as our own to decide and delimit the scope of the project. In sprint 1, Sprite, we got the scope finalized.

Since we had little experience with both LS Retails software, Dynamics AX, NAV and the Xamarin Cross-Platform Mobile framework, initial planning and assigning tasks the correct story point weight turned out to be a bit off in both directions. We had tendencies to overestimate and underestimate story points. Throughout the process we started to get familiar with the tools at hand and story point estimation got much better.

Priority	Total story points	Finished story points	% Finished
A	184	184	100%
B	39.5	39.5	100%
C	46	22	48%

Table 2: Overview of story points

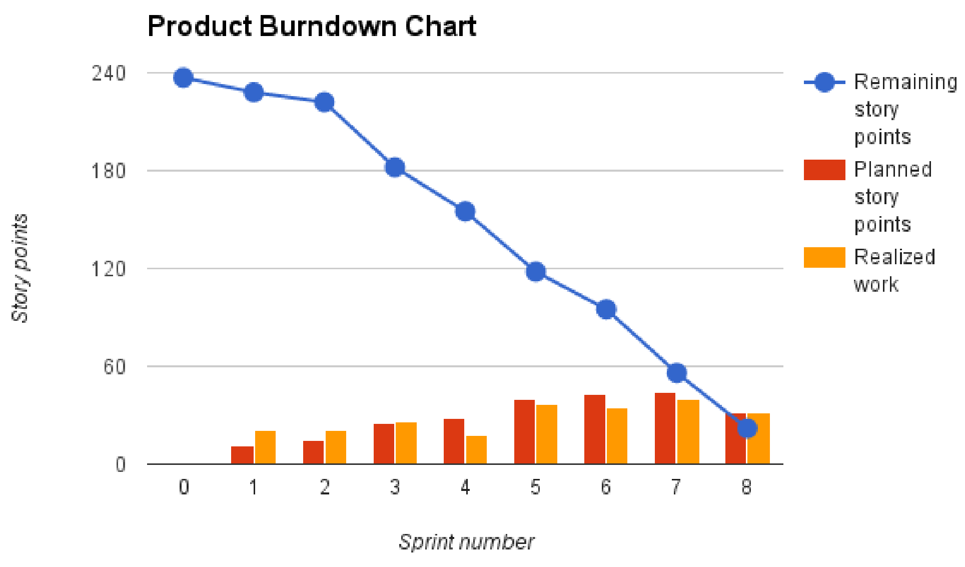


Figure 18: Product burndown

5.3 Total work hours

Sprint	Árni	Björn	Guðjón	Steinar	Total by sprint
Preparation	30:12:00	25:21:00	35:11:00	32:27:00	123:12:00
Sprite	45:00:00	33:47:00	32:28:00	42:00:00	155:15:00
Kelpie	23:56	25:14:00	28:35:00	35:21:00	113:08:00
Basilisk	37:06:00	22:45	21:16	34:26:00	115:35:00
Griffin	30:53:00	26:26:00	24:44:00	33:04:00	115:08:00
Sphinx	28:00:00	29:15:00	21:20	23:17	101:53:00
Phoenix	44:58:00	47:30:00	44:58:00	43:17:00	180:44:00
Thestral	68:21:00	58:43:00	57:28:00	48:11:00	232:43:00
Balrog	40:41:00	38:50:00	41:11:00	26:45:00	145:59:00
Total	**353:09:00**	**307:51:00**	**309:15:00**	**327:06:00**	**1305:23:00**

Table 3: Work hours

6 Usability testing

The usability happy path we focused on was the creation of an operation within the web dashboard and then recognizing the results in mobile app. It did not make sense to test usability on the app itself since it might never look the same between instances of it. We conducted two test sessions in the Phoenix and Thestral sprint.

Method

We predefined tasks that use most of our features on the dashboard as user stories.

Each test session conducted by two members of the team. One member was the test administrator and the other recorded the time. Talking to participant while completing a task was forbidden unless he/she asked for help and the test administrator was the only one who was allowed to answer. After each task we asked the participant a couple of questions about his experience to get a better idea what needs improvement.

Participants were given instructions to complete the tasks since the overall dashboard is complex and the actual user's intended for the product will be taught how to use it.

The tasks users were given were:

Task 1: Create Time Logger operation.

Precondition: The endpoint 'TimeLogger' exists within users ERP system.

You realize you need a time tracking operation in your app. You want to be able to log your time from your phone.

#	Action
1	Create an Operation
2	Change the Operation type to Input Activity and title to 'Time Logger' and string ID to TimeLogger.
3	Add two components.
4	Change the first component title and component string ID to 'Title'.
5	Change the first Component Validation to required.
6	Select the second component.
7	Change the second component Title and string ID to 'Datetime'.
8	Change the Component Type of the second component to datetimepicker.
9	Change the Component Validation to required.
10	Submit the result.
11	Open the app and confirm you created the operation.

Table 4: Task 1 actions

Task 2: Edit the Time Logger operation.

Precondition: Task 1 was performed successfully, ERP endpoint was changed.

You realize your time tracking operation was not complete. You need to log the time you started and when you ended. But your previous implementation only offered one DateTime picker. You need two.

#	Action
1	Edit TimeLogger on frontpage.
2	Change the title and string ID of the second component to ‘Start’.
3	Add one more component.
4	Change the title and string ID of the third component to ‘End’.
5	Change the third components type to datetimepicker.
6	Change the third components validation to required.
7	Submit result
8	Open the app and confirm your modified operation.

Table 5: Task 2 actions

Task 3: Delete the Time Logger operation.

Precondition: Task 1 was performed successfully.

You realize no longer need the TimeLogger operation since your job has punch in cards now and you find it easier to just punch in and out. Therefore you no longer need your operation. It was a good run.

#	Action
1	Select TimeLogger on frontpage.
2	Press the delete button.
3	Open the app and confirm TimeLogger is not there.

Table 6: Task 3 actions

Usability Test 1.

Conducted on the 1st of may in the meeting room Esja at LS Retail. Table below lists information about each user who participated in session.

All participants came from different background and had no experience using the product.

Gender	Age	Profession
male	25	Construction worker
female	25	Electrician
male	23	Web developer
male	29	Computer Science student

Table 7: Test 1 participants

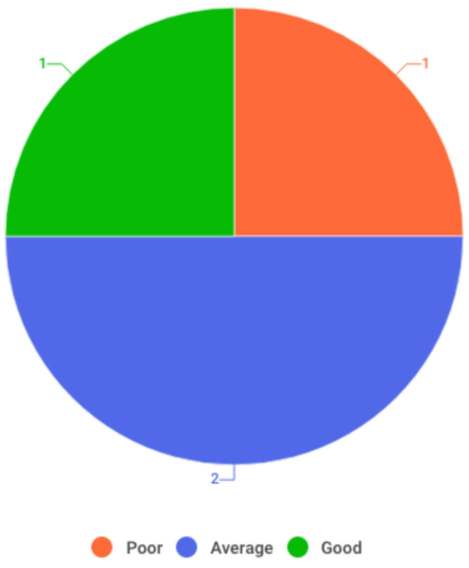
Test 1 results

Results

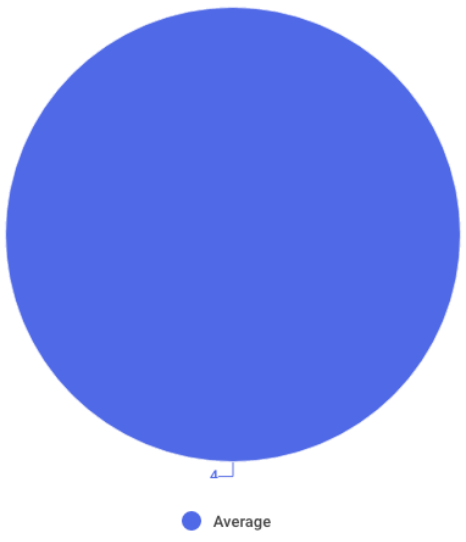
Task	Average Time	Finished Task	Needed Assistance
1	236,75 sec	0	4
2	143,25 sec	1	3
3	27 sec	4	0

Table 8: Test 1 results

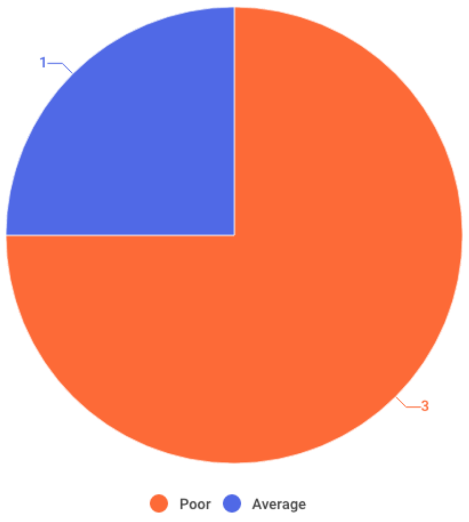
Questions 1-5 scale where numbers correlate to Bad, Poor, Average, Good, Excellent.



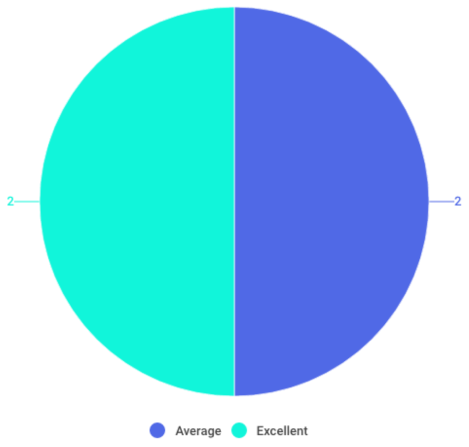
(a) How hard was it to navigate to the 'Create' menu?



(b) How was your overall experience of creating and submitting an Operation?



(a) Was it easy to find the edit button?



(b) Was it easy to edit the operation?

Analysis

Failed steps in first task:

- Step 1. All testers failed to find the Create view and had to be pointed out where it was.
- Step 3. Majority of testers needed assistance to find the add component button.
- Step 6. All testers failed to select the second component.

Failed steps in second task.

- Step 1. Majority of testers needed assistance finding the edit button.

No failed steps in third task.

Discussion

Speculations on results made it clear we had to redesign our interface and the following was not user friendly:

- Navigation design not too clear.
- Adding a component proved difficult to most.
- Selecting a component to edit needed to be more obvious
- Edit and delete button on front page not user friendly. Users already knew where the delete button was since it was next to edit button.

A new design and its implementation was set as a task in Thestral sprint and was to be completed for Usability Test 2.

Usability Test 2

Conducted 6th of may in the meeting room Esja at LS Retail. All participants came from different background and had no experience using the product.

Gender	Age	Profession
male	23	Computer Science Student
female	58	Chief Nurse
male	59	Technician

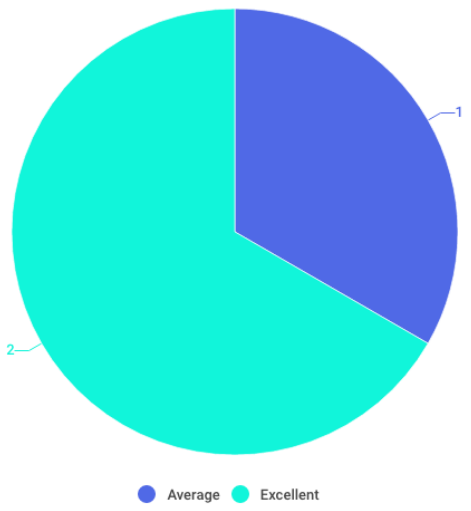
Table 9: Test 2 participants

Test 2 results

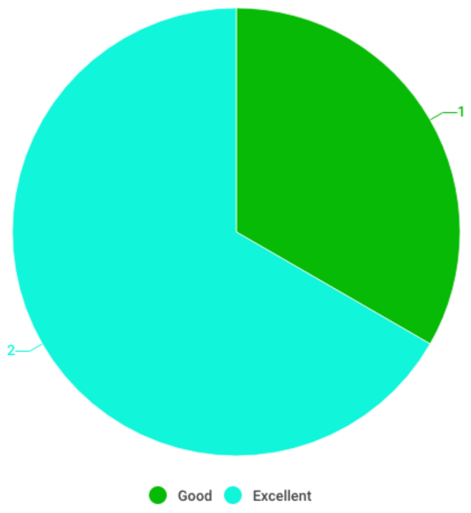
Task	Average Time	Finished Task	Needed Assistance
1	291 sec	2	1
2	148,33 sec	2	1
3	28.5 sec	3	3

Table 10: Test 2 results

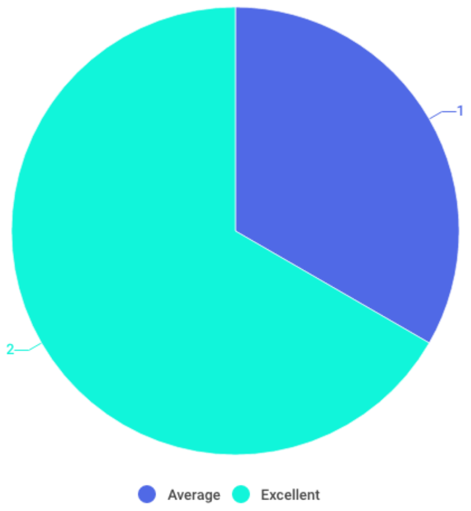
Questions 1-5 scale where numbers correlate to Bad, Poor, Average, Good, Excellent.



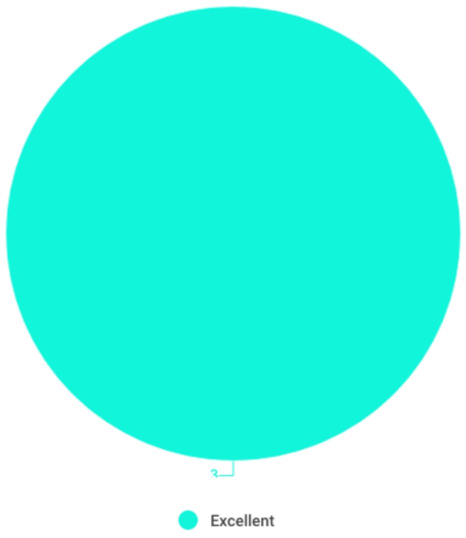
(a) How hard was it to navigate to the 'Create' menu?



(b) How was your overall experience of creating and submitting an Operation?



(a) Was it easy to find the edit button?



(b) Was it easy to edit the operation?

Analysis

Failed steps in first task:

- Step 1. One tester failed to find the Create view and had to be pointed out where it was.
- Step 3. One tester needed assistance finding the add component button.
- Step 4. One tester needed to be pointed out where to modify the component
- Step 6. One tester failed to select the second component.

Failed steps in second task:

- Step 1. One tester needed assistance finding the edit button.

No failed steps in third task.

Discussion

Compared to the previous testing the application seems to have improved and become more user friendly. Users varied more in this session. Some participants affected the finish time of each task due to not being as familiar to computers as others. It was expected to take longer time. None the less these testers finished most of their tasks and the improvements are considered a success.

7 Future Works

The product that we are handing off to LS Retail is a framework for their development team that is ready to adapt to their client's needs.

We would have liked to implement an iOS version but, like was stated before, did not have the time. We also would have liked to implement a client version of the dashboard with the same functionality as the apps so LS Retail could provide a web app as well as a native solution to their client's.

8 Conclusion

At the very beginning of our project we researched a couple of ways to develop cross-platform solutions since the solution needed to run on several platforms. We discussed multiple ways between ourselves but also we wanted to adapt to LS Retail methods so future work would be as easy as possible. Many ideas we're on the table and among them was Xamarin. We liked the idea behind it and wanted to use it but none of us had experience with it. To our surprise the first time we had a meeting with our representatives at LS Retail they revealed to us that the in house app team, LS OMNI, uses Xamarin for their development. So it was decided, Xamarin it is.

During the first few weeks of the project we hit a small bump because the vagueness of the project definition caused us to misunderstand the main goal. This misunderstanding was cleared up early enough that minimal work was wasted.

Overall the teamwork went great. Everyone provided their own speciality to the project, Björn with his experience in UI/UX design, Árni and Steinar taking point on the app front and Guðjón handling the web api.

We learned a lot during the course of this project, as it was the first time any of us implemented anything as large. Developing an app was also an enjoyable challenge given our inexperience with that branch of software development. We are very happy with both the final product and our experience during it development.

LS Retail vill continue this product. They have informed us that there is much interest for this idea among their customers and they are also happy with our final product.

We are very thankful to LS Retail for giving us a unique assignment which is, all in all, a very good idea. They provided us with facilities in their office, computer screens and testing equipment, and were accommodating when we needed help from their mobile team.

We would also thank our instructor, Birgir Kristmannsson, for lending his expertise and providing us with guidance.