



T-404-LOKA LOKAVERKEFNI

MAREL
CLOUD ANALYTICS

Final Report

Students:

Gísli Rafn Guðmundsson
Gunnar Páll Gunnarsson
Jón Reginbald Ívarsson

Teacher:

Hallgrímur Arnalds

Instructor:

Elín Elísabet Torfadóttir

Examiner:

Haukur Kristinsson

May 13, 2016

Contents

1	Preface	4
1.1	About this Document	4
1.2	Marel and Innova	4
1.3	Terms and Definitions	4
2	Cloud Analytics	6
2.1	Introduction	6
2.2	Motivation	6
3	System Design Progression	7
3.1	Initial System Design	7
3.1.1	Reasons for Redesign	7
3.2	System Design 2	7
3.2.1	Reasons for Redesign	8
3.3	System Design 3	8
3.3.1	Reasons for Redesign	9
3.4	System Design 4	9
3.4.1	Reasons for Redesign	9
4	Final System Design	10
4.1	Microsoft Azure	10
4.2	IoT Hub	10
4.3	Stream Analytics	11
4.4	Azure SQL Database	11
4.5	Cloud Analytics Website	11
5	Development	13
5.1	Development Environment	13
5.2	Centralized Version Control System	13
5.3	Continuous Integration	14
5.4	Continuous Deployment	15
5.5	Automation Scripts	15
5.6	Innova Azure Communications	15
5.7	Testing	16
5.7.1	Front-end Unit Tests	16
5.7.2	Back-end Unit Tests	16
5.7.3	End-to-end Tests	16
6	User Interface	18
7	Work Arrangements	19
7.1	Scrum Methodology	19
7.2	Roles & Responsibilities	19
7.2.1	Product Owner	19
7.2.2	Scrum Master	19
7.2.3	Development Team	20
7.3	Facilities	20

8	Risks	21
8.1	Internal Risks	21
8.2	External Risks	21
9	Progress	23
9.1	Total Progress Overview	23
9.2	Time Registration	23
10	Sprint Overview	25
10.1	Themes For Each Sprint	25
10.2	Sprint 0	25
10.2.1	Sprint Planning Notes	26
10.2.2	Sprint Retrospective Notes	26
10.3	Sprint 1	26
10.3.1	Sprint Retrospective Notes	27
10.3.2	What Went Well	28
10.3.3	What Could Have Gone Better	28
10.4	Sprint 2	28
10.4.1	Sprint Retrospective Notes	28
10.4.2	What Went Well	29
10.4.3	What Could Have Gone Better	30
10.5	Sprint 3	30
10.5.1	Sprint Retrospective Notes	30
10.5.2	What Went Well	30
10.5.3	What Could Have Gone Better	31
10.6	Sprint 4	32
10.6.1	Sprint Retrospective Notes	32
10.6.2	What Went Well	32
10.6.3	What Could Have Gone Better	32
10.7	Sprint 5	33
10.7.1	Sprint Retrospective Notes	33
10.7.2	What Went Well	33
10.7.3	What Could Have Gone Better	33
10.8	Sprint 6	34
10.8.1	Sprint Retrospective Notes	34
10.8.2	What Went Well	34
10.8.3	What Could Have Gone Better	35
10.9	Sprint 7	35
10.9.1	Sprint Retrospective Notes	36
10.9.2	What Went Well	36
10.9.3	What Could Have Gone Better	36
10.10	Final Sprint	37
10.10.1	What Went Well	37
11	Conclusion	39
11.1	Challenges	39
11.2	Future	39
11.3	Review from Product Owner	40
	Appendix A: Automation Scripts	41

Appendix B: Code Coverage	45
Appendix C: Wireframes	48

1 Preface

This document describes the system design, implementation and development process of the final project *Cloud Analytics*, at *Reykjavik Univeristy*. The project was done in cooperation with the *Innova* team at *Marel*.

1.1 About this Document

The document outlines the process and design of the system, describing each component used and the reason for it being chosen. It describes the data that is taken into the system, generated by Innova systems, as well as how that data is further relayed into other subsystems.

The development environment, version control, continuous integration, continuous deployment and testing are thoroughly described to give insight into the development process.

The document provides user interface wireframes as well as other front-end designs and reports that governs the way the data is effectively represented. Risk analysis, a progress report and a short description of what work was done during each sprint is also included. The document is concluded on a post-mortem.

1.2 Marel and Innova

Marel is the leading global provider of advanced processing systems and services to the Poultry, Meat and Fish industries. It was founded on March 17, 1983 and is headquartered in Garðabær, Iceland. Marel is comprised of approximately 4,600 employees worldwide, offices and subsidiaries in 30 countries across six continents, and a network of more than 100 agents and distributors.¹

Innova is a product development department within Marel that provides software solutions for Marel's food processing systems. Innova Food Processing Software range from simple device control solutions, to total processing solutions adapted to the individual needs of food processors. Based on modular designs, the solutions are scalable, thereby providing maximum flexibility for food processors, ranging from small operations to large, plant-wide systems.²

1.3 Terms and Definitions

- **AngularJS**

A JavaScript based MVW (model-view-whatever) framework by Google, great for creating front-end or full-stack web applications.

- **Azure - Microsoft Azure**

A cloud platform with relatively easy setup of services.

- **Critical Site - Critical Innova Site**

A critical site is an Innova system which is sending critical error logs.

- **IoT Hub - Microsoft Internet of Things Hub**

A Microsoft service for transferring data to and from the cloud.

¹<http://marel.com/corporate/about-marel/marel-at-a-glance>

²<http://marel.com/innova>

- **Power BI - Microsoft interactive data visualization BI tools**
Microsoft's tool for representing data interactively through beautiful charts.
- **Site - Innova Site**
A site is an Innova food processing system of any type.
- **Spark - Apache Spark for Azure HDInsight**
A processing framework that runs large-scale data analytics applications.
- **SQL - Microsoft Azure SQL Database**
Data-storage service based on SQL Server database technology.
- **Stream Analytics - Azure Stream Analytics**
Real-time stream processing in the Azure cloud.
- **TFS - Microsoft Team Foundation Server**
A service for version control, continuous integration and deployment, project management tools and more.

2 Cloud Analytics

The Cloud Analytics system is comprised of various different services that cooperate to analyze and visualize logs generated by Innova systems worldwide. These logs are processed by the system to give a better understanding of how the Innova production management software is behaving, and how Marel could improve performance and security with minimal downtime for its customers.

2.1 Introduction

The scope of the design is to create a system for processing and visualizing logs generated by Innova systems. Three different kinds of logs are processed; base logs, system program logs and device logs. These logs contain different sets of information, which is processed by the system. As a requirement the Cloud Analytics system was to be implemented on the *Microsoft Azure* platform. The system design is based on service-oriented architecture in which system components provide services to other components, each component of the system is independent and decoupled from the rest.

A combination of *IoT Hub*, *Stream Analytics*, *SQL Server* and other technologies were used to build the system as well as good software engineering practices, such as continuous integration, continuous deployment and extensive testing.

2.2 Motivation

The motivation behind the project is to give Innova Customer Service, a clear picture of when and where errors appear in Innova systems around the world. This allows them to respond quickly to errors, but also see if there are patterns emerging. For instance, if all sites in a specific country start sending errors, while others remain the same, it can be assumed that the problem is on a larger scale than if a single site is sending errors.

At present, problems with Innova systems require customers to contact and communicate over the phone. Customer service employees can then respond accordingly to the problem whether it is remotely connecting to the troubled system or instructing the customer on how to fix the problem.

The new system allows customer services to monitor Innova systems, get notified of emerging problems and possibly fix them before the client has to contact them. This may allow for many problems to be solved near instantly.

3 System Design Progression

This section contains an overview of previous designs for the system and the reasons for the system being redesigned. The system design changed tremendously since work first began. Each iteration of the system was thoroughly researched, in order to ensure the best possible solution for the project.

3.1 Initial System Design

The initial system design relied on the *ELK* stack, which consists of *Elasticsearch*, *Logstash* and *Kibana* - all developed and maintained by the Elastic team. A Logstash shipper was connected to each Innova system and responsible for sending logs to the cloud. A Logstash indexer received the logs, processed them and sent the results to Elasticsearch. Elasticsearch was then used to store the data in an easily searchable manner enabling Kibana to visualize the data fast and reliably. An SQL database was connected to the ELK stack to reliably store the data, and as a backup in case the Elasticsearch database had to be rebuilt or vice versa.

The ELK stack was originally chosen due to Elasticsearch's reputation for being both fast and flexible and because of Kibana's engaging visual representation of data.

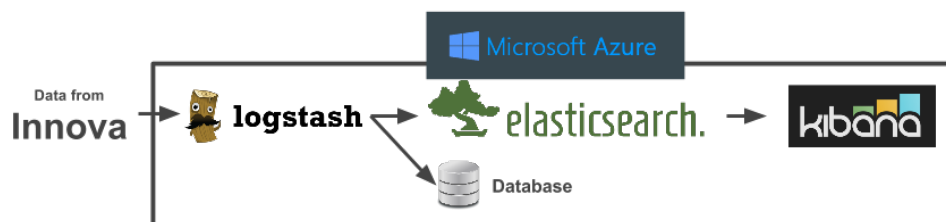


Figure 1: Initial System Diagram

3.1.1 Reasons for Redesign

The main reason for abandoning the ELK stack was lack of support in the Azure environment, as well as encouragement from the Innova team to research Microsoft services available in Azure.

3.2 System Design 2

For the second iteration of the system design, IoT Hub was selected as the best solution for connecting Innova systems to the cloud. IoT Hub offers secure two-way communications which, if needed, could be used to send a message back to a troublesome system. For instance, after a fatal error, a system could be automatically restarted. This functionality is, however, way beyond the scope of this project, and would require careful planning and execution.

HDInsight was researched as a possible better solution for processing and storing data, compared to the previous solution. *Azure HDInsight* is an *Apache Hadoop* distribution powered by the cloud. This means that it handles any amount of data, scaling from terabytes to petabytes on demand. HDInsight includes services like *Spark*, *R*, *HBase* and *Storm* as a full suite of tools.³

³<https://azure.microsoft.com/en-us/services/hdinsight/>

Microsoft's Power BI was chosen for the visualization of the data as it is a powerful feature-rich data manipulation and report authoring tool.

SQL database stayed unchanged from the previous design and would be used for permanent storage.

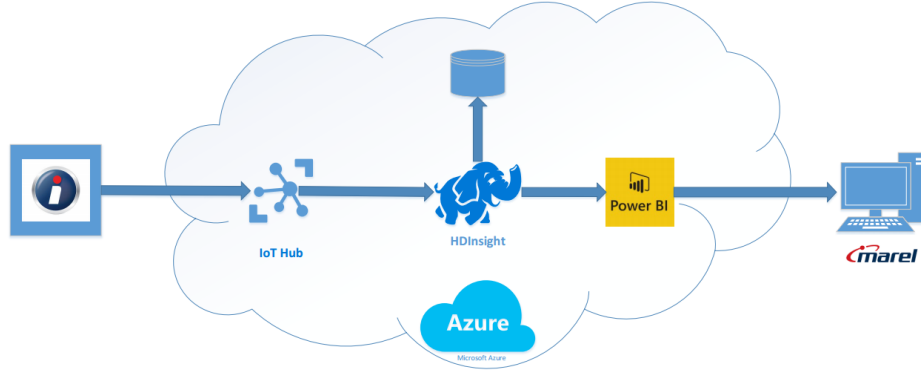


Figure 2: 2nd System Diagram

3.2.1 Reasons for Redesign

Through our research we learned that Spark, one of the tools provided by HDInsight offered faster in-memory data processing, ease of use and has a growing community support. Additionally, connecting Spark directly to Power BI was made trivial by Microsoft when they added Spark as a data connection option in Power BI.

3.3 System Design 3

The inclusion of Spark was the only change in the third system design. Spark would keep most of the data in memory for incredibly fast data processing and retrieval. Power BI would then be directly connected to Spark for visualization of the processed data.

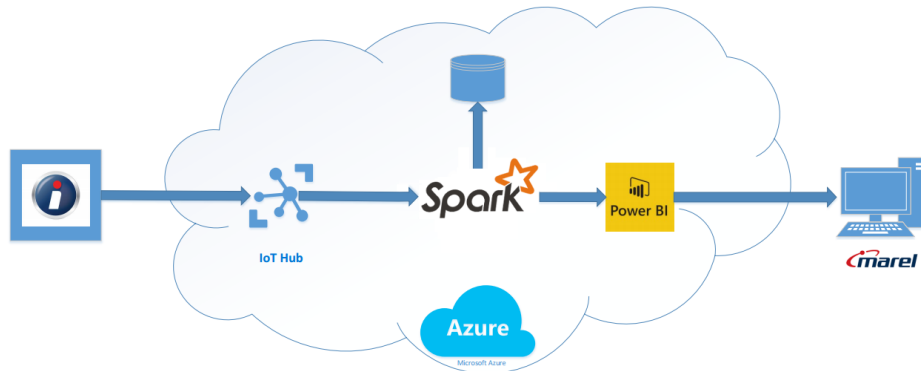


Figure 3: 3rd System Diagram

3.3.1 Reasons for Redesign

Excessive cost of maintaining the system was the main culprit for redesigning the system. Both HDInsight and Spark are big data systems, meant to handle processing and aggregating massive amounts of data (terabytes/petabytes) in a short period of time. The system required expensive resources to be used and caused us to take down and provision the system daily.

3.4 System Design 4

The service that set us on the right track, was Stream Analytics. It is lightweight, cost effective and easier to use service than HDInsight/Spark. Instead of creating complicated processing jobs to work with the data, Stream Analytics is essentially a pipe that streams data in real time from IoT Hub into a SQL database. A streaming job is created using SQL-like language to map the data correctly into the database.

In order to manage registered devices in the system, a web application would be created (see Innova Registry in Figure 4). It would contain basic forms to register, edit or delete a site from the system.

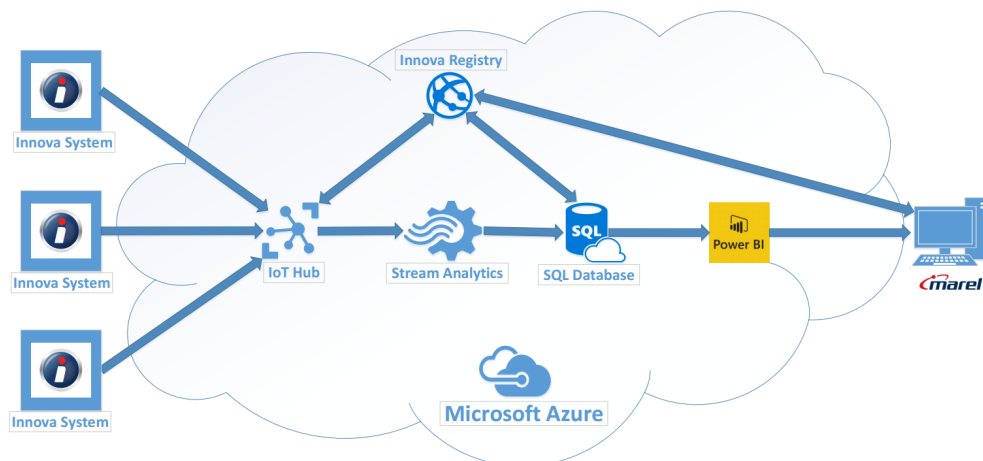


Figure 4: 4th System Diagram

3.4.1 Reasons for Redesign

This version of the system is the one most similar to the final version. Difficulties connecting Power BI to the system in real time and user manipulation of the visualization UI was deemed unnecessary and hence motivated us to redesign the system.

A decision was made to replace Power BI with a full scale web application, that would contain the data visualization and site registry. This means that developers would need to add new visualizations for customer services but more advanced features could also be added.

4 Final System Design

This section contains an overview of the final system design, and describes the platforms and services used in the system's implementation.

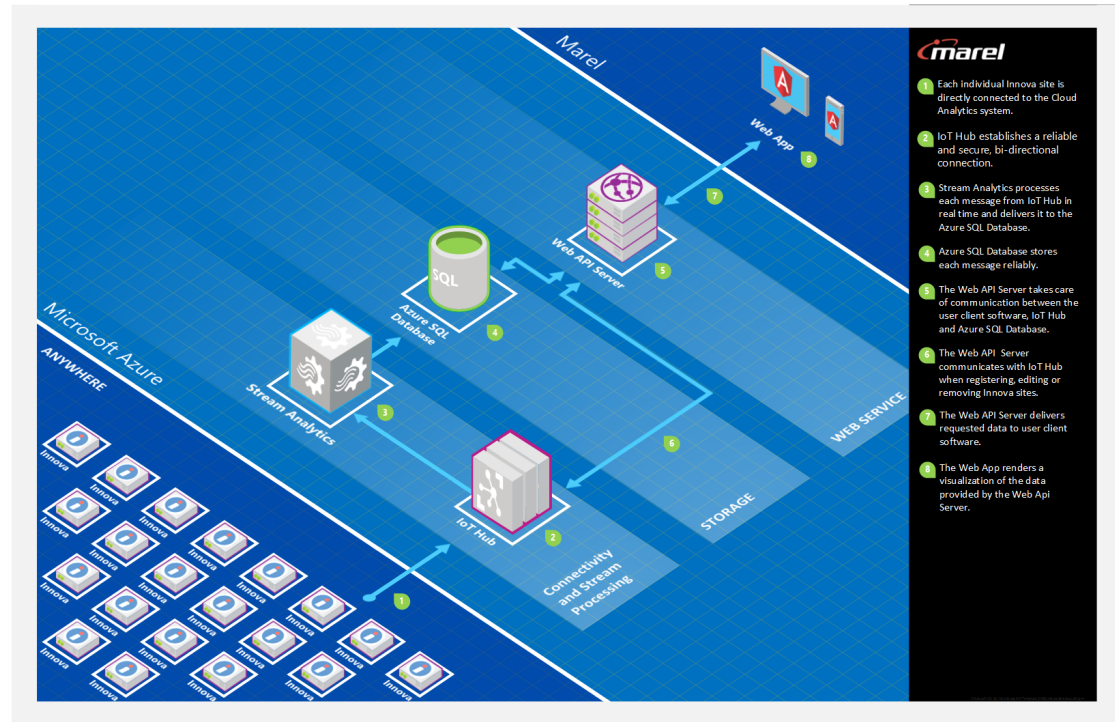


Figure 5: System Diagram

4.1 Microsoft Azure

Microsoft Azure is a cloud computing platform and infrastructure. It is used for building, deploying and managing applications and services through Microsoft hosted data centers. It provides *PaaS* (Platform as a Service) and *IaaS* (Infrastructure as a Service) services, and supports numerous programming languages and tools.⁴

The Cloud Analytics system runs on Microsoft's Azure platform and makes use of the services available on it. This specific platform was set as a requirement by the product owner when work began on the project.

4.2 IoT Hub

IoT Hub is an event processing service available in the Microsoft Azure cloud. It enables event and telemetry influx to the cloud at immense scale, with high reliability and low latency. The IoT Hub enables both device-to-cloud and cloud-to-device messaging, and supports device protocols like *AMQP* (Advanced Message Queuing Protocol), *MQTT* (Message Queuing Telemetry Transport) and *HTTP/1*. It is optimized to support millions of simultaneously connected devices and offers security features like per-device identity and revocable access control.⁵

⁴<https://azure.microsoft.com/en-gb/overview/what-is-azure/>

⁵<https://azure.microsoft.com/en-us/services/iot-hub/>

Each individual Innova system is connected directly to the IoT Hub service using a specially designed communication class. Logs generated by an Innova system is sent to the cloud where IoT Hub adds them to its queue for other services to request.

4.3 Stream Analytics

Azure Stream Analytics is a real-time event processing engine available in the Microsoft Azure cloud. It is able to perform real-time analytics for Internet of Things (IoT) solutions and capable of streaming millions of events per second coming from connected devices or log files, with throughput of up to 1 GB/s. Stream Analytics supports a simple, declarative, SQL-like query model for describing data transformation and therefore makes it easy to set up real-time analytic computations on data streaming for various devices, applications, sensors, and more. Results from the stream can be written to Azure Storage Blobs or Tables, Azure SQL DB, Power BI, and more. It also provides the ability to specify and use reference data, and helps prevent data loss with the ability to maintain state.⁶

Azure Stream Analytics service takes care of processing incoming logs from IoT Hub and storing the processed data in the correct database tables.

4.4 Azure SQL Database

Azure SQL Database is a cloud-based service from Microsoft offering data storage capabilities as a part of the Azure Services Platform. It is based on the market-leading Microsoft SQL Server engine. Azure SQL Database allows users to make relational queries against stored data and delivers predictable performance. Additionally, the service offers scalability with no downtime, business continuity and data protection.⁷

Azure SQL Database stores all processed data from the Stream Analytics services as well as data related to registered Innova sites.

4.5 Cloud Analytics Website

The *Cloud Analytics Website* is built using the *AngularJS* framework by Google. The website uses a restful API server built with *ASP.NET* Web API libraries. It is comprised of a Innova site registry, the main analytics screens and finally, a screen containing all log data for a requested Innova site.

The registry allows users to register new Innova sites, edit existing sites and remove sites from the system in a secure and reliable manner. When a new Innova site is registered or an existing one is edited, the user needs to supply basic information about the system, such as the name of the site and what version of the Innova system is in use. They can also set the exact location of the site using the provided map or click the *Get Location* button if they are located at the site. When registering a site to the system a connection key is generated which is required to establish communications between the two. The key for the desired site can be retrieved by clicking the *Show Key* button.

The analytics screens show where errors are occurring in the world at different zoom levels. The overview (front) page summarizes critical sites into six main regions of the world represented with colored circles. A region can be clicked to reach its sub-regions. A sub-region can then be clicked to see what countries have sites that are producing errors. Finally, if a country is selected, all critical sites (sites with errors) are displayed

⁶<https://azure.microsoft.com/en-us/services/stream-analytics/>

⁷<https://azure.microsoft.com/en-us/services/sql-database/>

for that country. Each site is displayed on a map as a circle and the size indicates how many critical errors have occurred in the last 24 hours. Each screen features a timeline that can be used to filter the data into smaller time segments.

If a site is selected, the user gets a list of all logs for that site - errors or otherwise. The logs can be filtered and ordered to only see what is relevant for the user. Each log can be clicked to get further info about that particular log, such as type or what object caused the error to be logged.

5 Development

The following sections describe the project's development environment, source control, continuous integration, continuous deployment, automation scripts as well as testing practices.

5.1 Development Environment

The solution consists of PowerShell scripts, .NET projects and AngularJS projects. Microsoft Visual Studio is mainly used for developing .NET code and should be exclusively used for any code organization purposes. All files intended for source control should be added to the Visual Studio solution.

Other editors can be used for the development of front-end code or PowerShell scripts, such as Atom or Sublime and PowerShell ISE for PowerShell Scripts. Source Code is kept in a private repository on Visual Studio Team Services. For more details see the attached *Development Manual*.

5.2 Centralized Version Control System

Version control systems help development teams to manage changes that happen within directories or files over time. Because every modification is tracked automatically by the system, developers are able to revert to or check out earlier versions to fix mistakes, minimizing disruption to all team members. *Centralized Version Control System* (CVCS) works in a client and server relationship. A single server that contains all the files, changesets, users, and information, provides access to the latest version. Every code change must be sent and received from this central server, enabling collaboration between developers.

For this project CVCS was integrated using the Microsoft's *Visual Studio Team Services* (VSTS) platform and *Team Foundation Version Control* (TFVC) software.

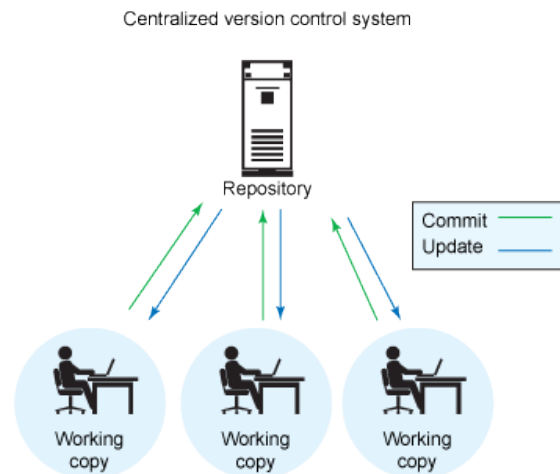


Figure 6: Centralized Version Control System⁸

⁸<http://www.ibm.com/developerworks/library/d-app-centric-ops/>

5.3 Continuous Integration

Continuous Integration (CI) is a development practice that requires software developers to integrate code changes early and often into a shared repository. A build server runs automated unit tests written through the practice of test-driven development to verify the changes. It then reports the results back to the team, allowing them to detect and prevent integration problems. This practice aims to reduce rework and thus reduce cost and time.

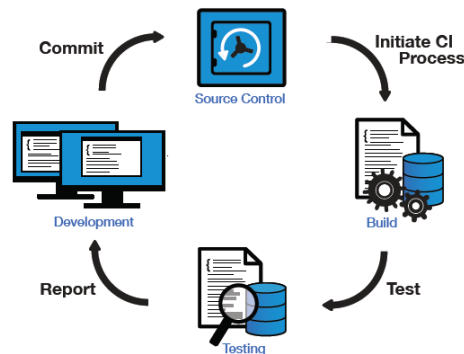


Figure 7: Continuous Integration Cycle⁹

For this project CI was implemented using Microsoft's Visual Studio Team Services platform and an on-premises build server. Code changes are checked into VSTS and the build server runs all tests, sending reports to the team in case of a build failure.

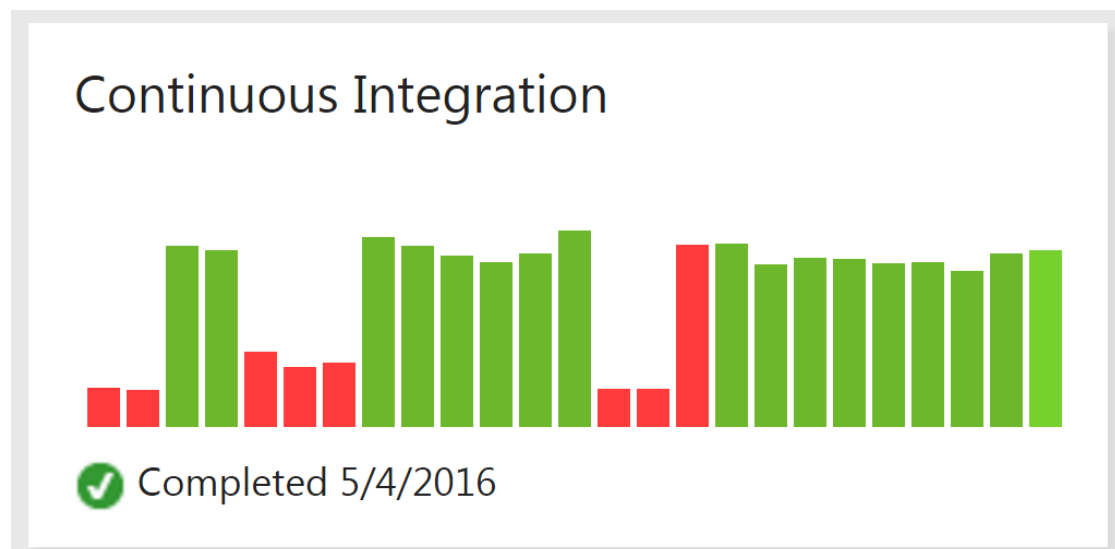


Figure 8: Build Chart

⁹<https://kaizentesting.wordpress.com/2012/08/19/agile-test-automation-is-incomplete-without-continuous-integration/>

5.4 Continuous Deployment

Continuous Delivery is an extension of Continuous Integration. It is a discipline where every changes to the software can be released to production at any time. It aims at building, testing, and releasing software faster and more frequently in a repeatable deployment process. Continuous Delivery makes it possible to continuously adapt software in line with user feedback, shifts in the market and changes to business strategy. *Continuous Deployment* is the next step of Continuous Delivery. Every change that passes the automated tests is deployed to production automatically.

In this project Continuous Deployment is implemented using Microsoft's Visual Studio Team Services platform and an on-premises build server. After a successfully testing and building the software, the on-premises build server packages the software and deploys it to the Azure cloud.

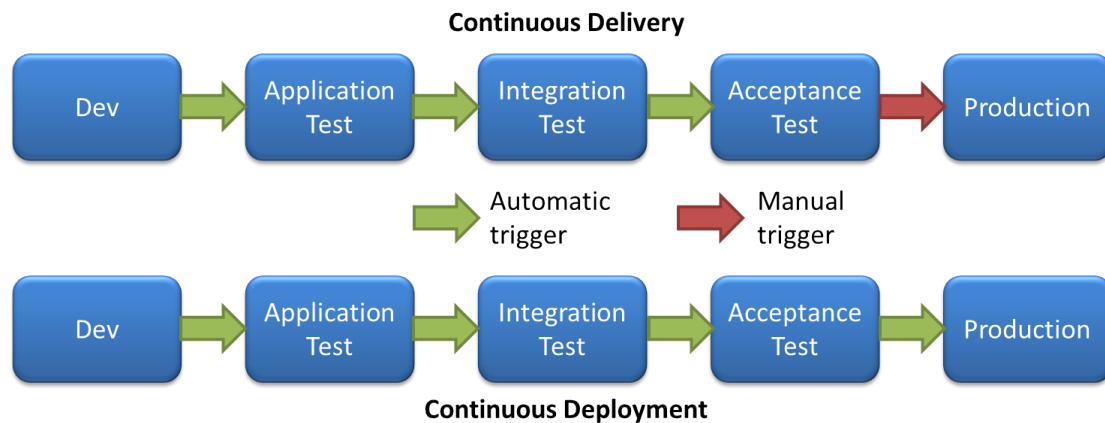


Figure 9: Continuous Delivery / Deployment¹⁰

5.5 Automation Scripts

In order to provision the system as a whole, the team developed a set of automation scripts. The scripts created are used to provision all services used by the system, deploy the user software and tear the whole system down. These scripts are explained in detail in *Appendix A*. How to use these scripts to assemble an instance of the system is described in the attached *Administration Manual*.

5.6 Innova Azure Communications

A simple class, named *CloudLogger*, was written for the secure and reliable communications between Innova systems and the Azure Cloud. Its main purpose is to streamline the integration of this functionality into the existing Innova code base. The *CloudLogger* class uses the Microsoft recommended AMQP protocol and Azure SDK libraries for the communications. The class follows a few steps when sending a log to the cloud. When a new log is created, it is sent to the class which extracts its content, adds additional properties and then serializes everything into a JSON object. Once this is completed, the log is sent asynchronously to the cloud.

¹⁰<https://notafactoryanymore.com/tag/continuous-deployment/>

When an Innova system has been updated with the CloudLogger update, it is important to configure the system properly to start sending logs to the cloud. The first step is to add the new system to the Innova registry, which is found in the Cloud Analytics website. A primary key will then be generated for the system which is used by the CloudLogger class to initiate a secure communication line with the IoT Hub. The primary key is added to the Innova system through the configuration screen.

5.7 Testing

Software testing is an important part of software development. It provides information to the development team about the quality and risk of failure for the software. Software testing evaluates whether the software meets the predetermined requirements by executing software component or system component. Testing is often automated but can also be done manually.

Unit testing is a software testing method where the smallest individual units of an application are tested. Units are commonly individual functions or procedures, but can also be an entire interface such as a class.

End-to-end testing is a software testing method that is meant to test a system, from its entry point all the way to being displayed on screen (or received by some other end-point). It tests whether the flow of the system, from start to finish, performs as expected. The purpose of carrying out end-to-end tests is to identify system dependencies and to ensure that the right information is passed between various system components.

5.7.1 Front-end Unit Tests

Karma and *Jasmine* are used in combination to design and run unit tests for the front-end of the web application.

Karma is a JavaScript test runner that starts a web server to execute source code against tests for each connected browser. Karma watches for changes in files, and signals the test server to run when changes have been detected. Karma is highly configurable. It is able to generate *code coverage* reports and it is test framework agnostic.

Jasmine is an open source test framework for JavaScript code. It enables developers to write clean and understandable tests. These tests run automatically on the build server as well as on developers' local machine to ensure code quality.

Code coverage for the front-end unit tests can be found in Appendix B.

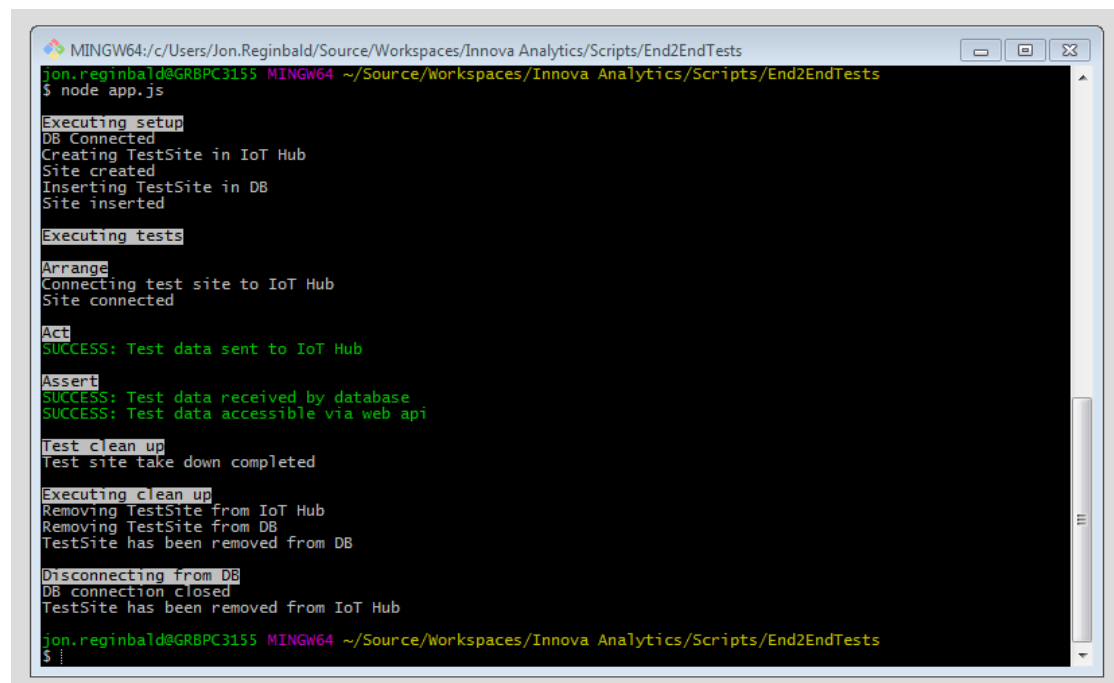
5.7.2 Back-end Unit Tests

Visual Studio's test runner is used in combination with Microsoft's Unit Testing Framework to unit test the back-end code of the web application as well as the CloudLogger class. These tests run automatically on the build server to ensure code quality and are manually started by developers.

Code coverage for the back-end unit tests can also be found in Appendix B.

5.7.3 End-to-end Tests

A *Node.js* console application was created to end-to-end test the system. The application starts by registering a test site with the system. It then generates a log, sends it to the system and verifies that the log is able to travel through the entire system. Developers should run this application after creating a new Cloud Analytics system to ensure that all services are able to communicate and that they work as intended.



```
MINGW64:/c:/Users/Jon.Regimbald/Source/Workspaces/Innova Analytics/Scripts/End2EndTests
jon.regimbald@GRBPC3155 MINGW64 ~/Source/Workspaces/Innova Analytics/Scripts/End2EndTests
$ node app.js
Executing setup
DB Connected
Creating TestSite in IoT Hub
Site created
Inserting TestSite in DB
Site inserted

Executing tests

Arrange
Connecting test site to IoT Hub
Site connected

Act
SUCCESS: Test data sent to IoT Hub

Assert
SUCCESS: Test data received by database
SUCCESS: Test data accessible via web api

Test Clean up
Test site take down completed

Executing Clean up
Removing TestSite from IoT Hub
Removing TestSite from DB
TestSite has been removed from DB

Disconnecting from DB
DB connection closed
TestSite has been removed from IoT Hub

jon.regimbald@GRBPC3155 MINGW64 ~/Source/Workspaces/Innova Analytics/Scripts/End2EndTests
$
```

Figure 10: End-to-end Test

6 User Interface

A number of different user interface experiments were conducted to study the best way of representing the data in a clear way, while still remaining easy and intuitive to navigate. The original idea called for a map with a dot or a circle representing every single site in the system, scaled in relation to the amount of error logs it contained. After a meeting with the product owner the conclusion was made that such a representation would be virtually unusable due to clutter. Instead a dot should represent a region, sub-region or country at different zoom levels respectively. Not until reaching a screen that represents a country can the user see individual sites.

Due to limitations with the current map component, zooming in by using the scroll wheel of a mouse was not possible. Other features were also saved for later, such as writing the amount of sites with errors as a number inside the circles. We will continue to expand on the design after handing in the project to Reykjavík University.

For more detailed instructions on how the user interface works and how to use it, see the attached *User Manual*. Screenshot of the opening page of the web application can be seen in Figure 11.

Wireframes describing the initial designs for the interface can be found in Appendix C.

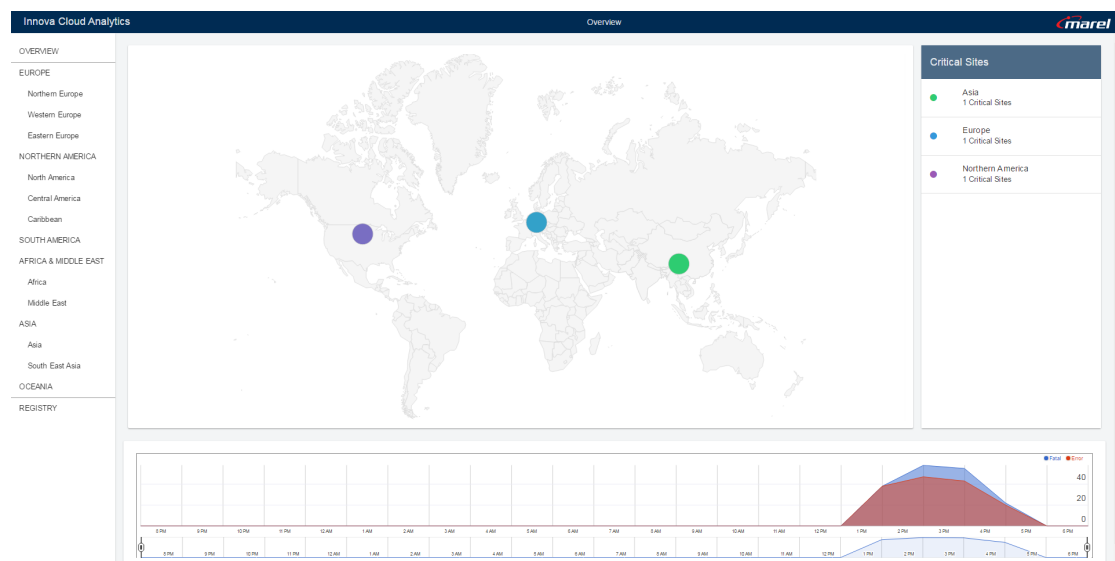


Figure 11: Screenshot of the Overview Screen

7 Work Arrangements

This section discusses the work arrangements for the project. This includes what sort of organizational methodologies were adopted, what roles team members and other persons involved in the project took on, facilities, etc.

7.1 Scrum Methodology

Software development at Marel is focused around *Scrum* and *Kanban* methodology. Each team implements variations of it that suits the current project and the team's capabilities. Keeping with Marel's best practices, on this project, we implemented Scrum software development mixed with Kanban.

The Scrum methodology is an *Agile* software development framework for managing iterative and incremental product development. The product is built in a series of fixed-length iterations called sprints, typically one week or two weeks long. The state of the product is kept potentially shippable and test proven at all times.

Kanban emphasizes on just-in-time delivery and visual process-management. All participants have a full view of the process from task definition to delivery. Visualizing the work flow of the team leads to increased communication and collaboration. Limiting unfinished work in progress reduces the time it takes for individual items to travel through the system. Teams are able to measure their effectiveness by tracking flow, quality, throughput and lead times.

7.2 Roles & Responsibilities

There are three essential roles in the scrum methodology; *Product Owner*, *Scrum Master* and *Team Member*. Generally, only one person is a Product Owner and one is a Scrum Master. A Product Owner and Scrum Master may or may not be in the development team. These core roles are committed to the project and are responsible for the delivery of a finished product.

7.2.1 Product Owner

The Product Owner is responsible for communicating the vision to the development team, understanding the business and market requirements, and prioritizing the work that needs to be done. The Product Owner represents the customer and guides the team toward building the desired end product.

Karl Karlsson is a lead software developer in the product development department at Marel and was the formal representative of the Product Owner during the development of the product.

7.2.2 Scrum Master

The Scrum Master ensures that team members practice the agreed Scrum processes and that each team member is able to perform at the highest level. The Scrum Master acts as a buffer between the team and distracting influences that obstruct the team from achieving its sprint goals. They facilitate key sessions, sprint planning, and daily sprint and retrospective meetings. Encouraging the team to improve, remain creative and productive while making sure the team's success is visible to the Product Owner.

During the course of this project **Jón Reginbald Ívarsson** was the acting Scrum Master based on his prior knowledge with Scrum methodology.

7.2.3 Development Team

The development team is responsible for analyzing, designing, developing and testing. The team is cross-functional, meaning that all of the necessary skills to create a finished product is present within the team. The team determines how much work is to be completed for each sprint using their previous velocity as a guide. The team is responsible for accomplishing confirmed sprint goals, and at the end of each sprint, the team delivers a shippable product increment.

The development team consisted of three computer science undergraduate students from Reykjavík University:

- Gísli Rafn Guðmundsson
- Gunnar Páll Gunnarsson
- Jón Reginbald Ívarsson

7.3 Facilities

Facilities were provided by Marel, and all development work was done at their headquarters. Marel provided the team with three development computers to conduct research and development. An Azure account with monthly credit was provided.

8 Risks

The risk analysis is an integral part of the project's planning process. Understanding what risks lie ahead that could delay or hinder progress of the project enables the team to react beforehand and possibly prevent them from happening. It can also clear responsibility issues, i.e. figure out who is responsible for each risk, as well as realizing what consequences each risk can have on the project. Analyzing these factors can additionally help the team's decision on whether or not to devote the time and resources needed to react to specific risks - since not all risks are equal. Below are the most pressing risks and responses for the project as they were defined by the development team.

8.1 Internal Risks

- Project Delay
Description - Project gets delayed, either because the scope was too large, or because the velocity of the team is too low.
Response - If the project falls far behind schedule, an emergency meeting is held with the project owner and the project re-planned.
- Technical Issues within Marel
Description - General technical issues that may arise from within Marel, e.g. network issues.
Response - Move our operations to HR, while Marel IT fixes the issue.
- Scope of the project increases
Description - The project's scope becomes greater than what was initially expected.
Response - Decrease the scope by removing functionality that has low priority.
- Technology too complicated
Description - One of the technologies is more complicated, or harder to work with, than initially expected, resulting in delays.
Response - Reevaluate whether the technology is essential and if there is an immediate alternative. If so, consider substituting it.

8.2 External Risks

- Licensing
Description - Licensing issues preventing project's deployment.
Response - Find an alternative licensing solution.
- Problem with Azure during development
Description - Microsoft Azure cloud platform or services become inaccessible.
Response - Continue development offline until Azure becomes accessible again.

- Sickness
Description - A team member, or other person vital to the project, can not work due to being sick.
Response - Remaining team members try to work harder for a while. If the problem persists, consider redefining the scope of the project to be manageable for a smaller team.
- Busy times at the university
Description - All team members are full-time students at Reykjavík University. Some parts of the semester are more busy with work unrelated to the project than others, such as during final exams.
Response - Plan sprints during these times accordingly and reduce the amount of work expected to be done.

9 Progress

9.1 Total Progress Overview

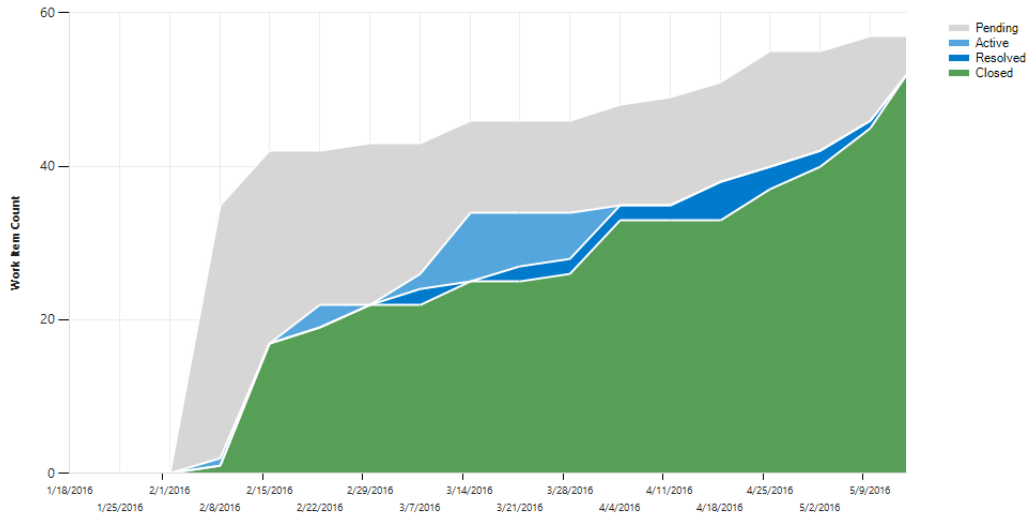


Figure 12: Summary of the total work item count

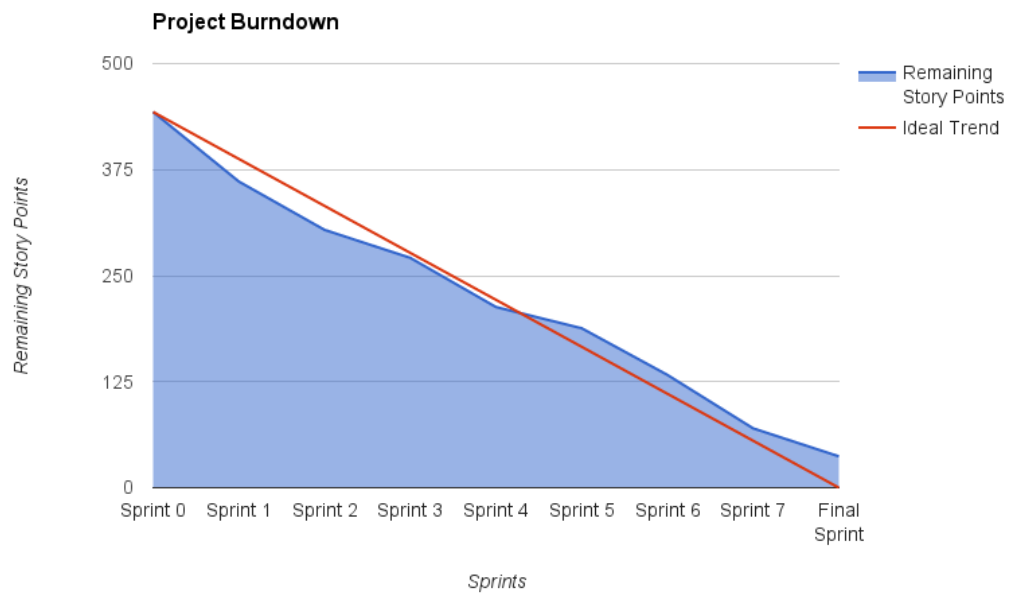


Figure 13: Project Burndown

9.2 Time Registration

The development team maintains a work journal which keeps track of the teams working hours. Each team member is responsible for logging their own work hours. The hours are

then tallied together to create a time report, showing total hours for each team member, on each given day and sprint. Additionally, two charts are generated; one for each team member's time partitioning and one as a summary of total work hours spent on the project. For closer inspection and up-to-date journal, please visit:

<https://goo.gl/91SKOa>

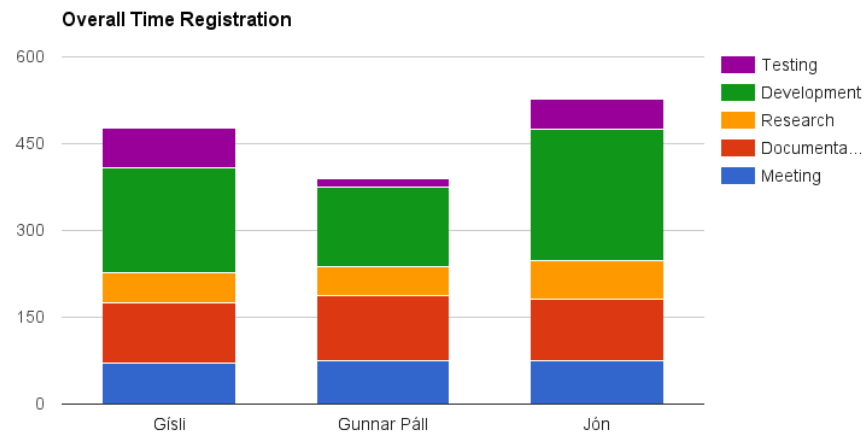


Figure 14: Overall time registration per team member

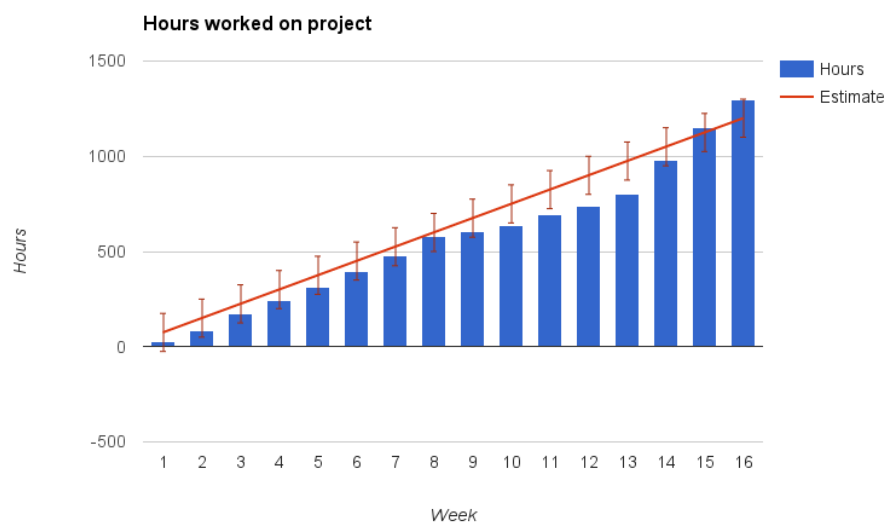


Figure 15: Summary of the total hours spent on the project

10 Sprint Overview

The duration of each sprint is two weeks long, except for sprint 0 and the final sprint, which are one week long each. Every sprint starts on a Monday with sprint planning, and ends on a Sunday with sprint review and retrospective analysis (usually performed on the following Monday).

During the planning meeting the team picks the highest priority features and turns the high-level user stories of the product backlog into detailed tasks. The team commits to the tasks and sets a sprint goal.

The purpose of a retrospective meeting is to assess how well the sprint went; what went well during the sprint and what could have gone better. The teams productivity and level of cooperation is observed, in order to improve for upcoming sprints. Any problems or weaknesses are also taken into consideration to avoid their reappearance in the future.

Daily sprint meetings were short stand-up meetings held every weekday where daily goals, challenges and issues were discussed. *Visual Studio Team Services* was used as the project management tool. Product backlogs and burndown charts are accessible through this link:

<https://goo.gl/5cAcRw>

Some sprints' burndown chart may not reach zero. This was usually due to the sprint ending on a Sunday, making us unable to close the stories when we came to work on Monday.

10.1 Themes For Each Sprint

Each sprint coheres to a theme which describes what kind of work should have been done at that time.

- **Sprint 0:** The Planning Sprint
In this sprint team members mostly worked on planning and logistics.
- **Sprint 1** (a.k.a. Sprint 00): The Research Sprint
This sprint was meant for research work, where team members researched the platform being used and decided what services should be used for the project.
- **Sprint 2-4:** Design and Setup Sprints
These two sprints were used for designing and setting up the system, and making sure that all the components worked well together.
- **Sprint 5-7:** Development Sprints
The bulk of the system development took place during these sprints.
- **Sprint 8:** Final Sprint
The last sprint was meant for finalizing the project, adding any features that still needed to be added and tying up loose ends.

10.2 Sprint 0

Sprint 0 was not a full sprint, and the first and only task for the team was to discuss and negotiate with Marel about the project. The backlog for the sprint and the burndown chart can be seen below. Because we changed our project management tool from *Google Spreadsheets* to *Visual Studio Team Services* in sprint 1, the burndown chart appears to be empty.

Title	Description	Story Points
Marel meetings	As a developer, I want to meet with the staff of Marel so that I get a better understanding of the project.	2

Table 1: Sprint 0 - Backlog

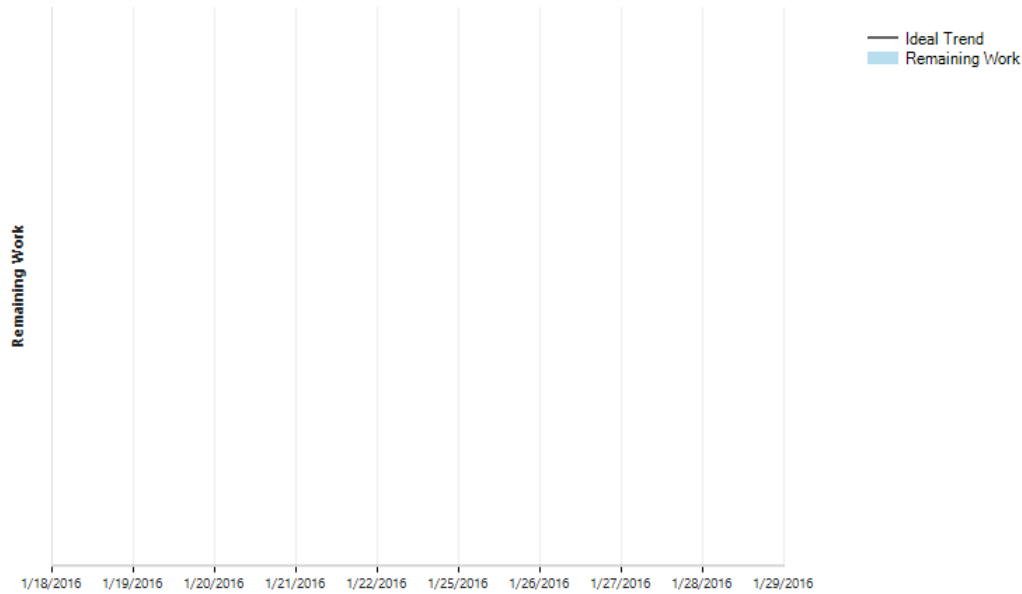


Figure 16: Sprint 0 - Burndown Chart

10.2.1 Sprint Planning Notes

The purpose of the first sprint planning meeting was to define the sprint goal, estimate the team's velocity, decide which stories to include in the sprint, create tasks for each story and assign each task to a team member to indicate who is responsible for the execution of that task.

10.2.2 Sprint Retrospective Notes

The first sprint went as expected. The team got up to speed on what the project was and its purpose.

10.3 Sprint 1

The research sprint. Our main goal was to research technologies that would enable us to reach the desired goal of the project. Finding good project management tools and learn how to use them. In this sprint we decided to use the project management tool *Visual Studio Team Services*. A large amount of time went into implementing Scrum and learning basic actions within the management tool. In the burndown chart below, a large spike can be seen which can be explained by the fact that the management tool was introduced in the later stages of the sprint.

Title	Description	Story Points
Research Azure	As a developer, I want to research Azure so that I know how to develop for the platform.	8
Facilities and Equipment	As a developer, I want to have facilities and equipment set up at Marel so that development is possible.	5
Research the ELK stack	As a developer, I want to research the ELK stack so that I know that it is a viable option for the project.	8
Risk Analysis Report	As a developer, I want to write a risk analysis report so that I can better understand the risks involved with the project.	5
System design draft report	As a developer, I want to draft a system design document so that I fully understand the structure of the solution.	8
Status Summary Report 1	As a developer, I want to write the first status summary report so that the reader can easily track our progress.	5
Product Backlog v1	As a developer, I want to write the first version of the product backlog so that development is able to start.	8
Research IoT Hub	As a developer, I want to research IoT hub so that I understand how it works.	5
First status meeting	As a developer, I want to attend the first status meeting so that the instructor and examiner get a status update.	2
Name for the team	As a developer, I want to find a name for the team so that we can setup accounts at Marel.	3
System diagram	As a developer, I want to design the product's system diagram so that I understand the basic structure of the system.	8
Product Owner + Team meeting	As the scrum master, I want the product owner to meet the team so that we can discuss the project.	5
Work Arrangement Report	As a team member, I want to write a work arrangement report so that the project guidelines are clearly defined.	3
Student presentation	As a team member, I want to present to my peers my project so that I get some experience before the status meetings.	3
Instructor meeting 2	As a team member, I want to meet with the instructor so that I am prepared for status meeting 1.	1
Initial HDInsight research	As a developer, I want to do some initial research on HDInsight so that I know that the team should commit to it.	5

Table 2: Sprint 1 - Backlog

10.3.1 Sprint Retrospective Notes

The first official sprint was used for research. Some planning aspects could perhaps have gone better, such as scheduling work on other projects, in between work on this particular project. This disruption was, however, to be expected, as all team members are involved in other projects during this semester at the university.

The team had initially planned to use a different set of services, but those plans were promptly changed in search of different ones, more fitting to the project. Research on the biggest new element, HDInsight, was extended into sprint number two.

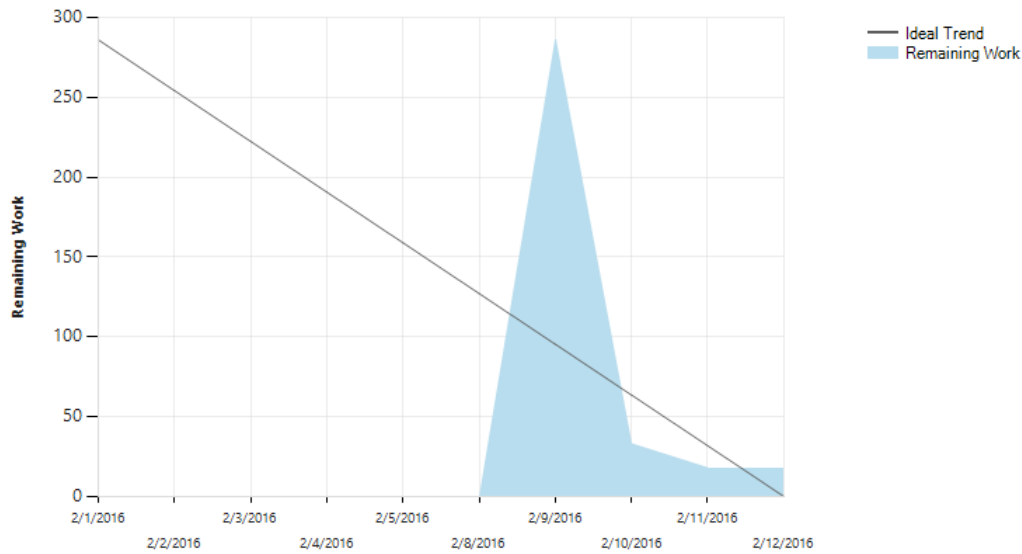


Figure 17: Sprint 1 - Burndown Chart

10.3.2 What Went Well

- Research
Research went very well and equipped the team with the necessary knowledge to start development, using the chosen services.
- Teamwork
We found that the team worked well together and the team's dynamic was productive.

10.3.3 What Could Have Gone Better

- Service change
Changes in the chosen Azure services delayed the work of the team, but adjustments were made easy by addressing the problem early.
- Scrum
Setting up a product backlog and other Scrum related reports took more time than initially anticipated.

10.4 Sprint 2

The first design and setup sprint. At the beginning of the sprint the daily operation was moved from Reykjavík University to Marel's headquarters, as our facilities became ready for use. We continued designing the system, but except for trying out some Azure services, nothing was actually built or coded during the sprint.

10.4.1 Sprint Retrospective Notes

Like Sprint 1, the sprint was a success and everything that the group had set out to do was completed. Version control was implemented using the same Visual Studio Team Services platform our project management tool is based on. We settled in comfortably at

Title	Description	Story Points
Research HDInsight	As a developer, I want to research HDInsight so that I know that it is a good solution for the project.	20
Azure account	As a developer, I want to have access to the Microsoft Azure Platform so that I can start developing a final solution.	3
Version Control	As a developer, I want to setup version control on Marel's Team Foundation Server so that development of the system can start.	13
Setup IoT Hub	As a developer, I want to setup IoT hub so that Azure is able to receive messages from Innova systems.	13
Create a fake log generator	As a developer, I want to create a fake log generator so that I can see how the system works.	8

Table 3: Sprint 2 - Backlog

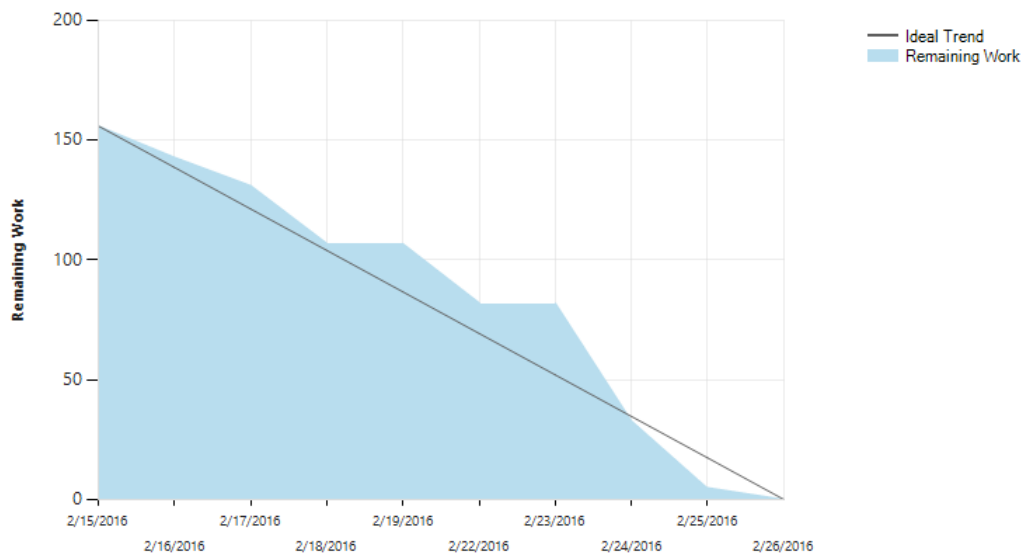


Figure 18: Sprint 2 - Burndown Chart

Marel and continued research at our own pace. The official research phase of the project was officially concluded at the end of the sprint.

10.4.2 What Went Well

- **Equipment Setup**
Setting up development environment on the computers provided by Marel went well. The computers were properly set up early on in the sprint.
- **Version Control**
Version control was set up at the beginning of the sprint without issues.
- **Research**
Research continued to go well.

10.4.3 What Could Have Gone Better

- **Slow Start**
The transition from working at RU and moving to Marel took some time, and regaining the pace at which we were working was gradual. However, this did not affect us much as the bulk of the research was still being done away from Marel and no actual coding was disrupted.
- **Getting Full Access to Resources**
Some time went into providing us with access to internal and external resources such as Marel accounts and a Microsoft Azure account. The wait was still quite short considering the size of Marel as an organization.

10.5 Sprint 3

Sprint 3 was the 2nd *setup and design* sprint. We had by now gotten access to Azure and began experimenting with IoT Hub and HDInsight Spark. Most of the work done in the sprint went into connecting the two together, and to some extent, to connect Power BI to Spark. Continuous Integration/Delivery was also a big focus during the sprint. Unanticipated, we decided to rethink our service pipe line in the middle of the sprint, as reflected upon in the sprint's retrospective notes.

Title	Description	Story Points
Setup Spark on HDInsight	As a developer I want to setup Spark on HDInsight so that Azure is able process messages from Innova systems.	13
Continuous Delivery	As a developer, I want to have continuous delivery so that I ensure that the software can be reliably released at any time.	20
Stream Analytics	As a developer, I want to process events from IoT Hub and insert them into SQL Database.	13

Table 4: Sprint 3 - Backlog

10.5.1 Sprint Retrospective Notes

The sprint went well enough, but we ran into some problems on the way. Firstly, one team member was sick for a few days during mid-sprint. Thankfully, we anticipated it in our risk analysis. We also tried our best to connect IoT Hub and Spark together, which did not go over as well as we had hoped. At the end of the sprint some frustration set in, and after discussing the reasons for selecting Spark we decided to call an emergency meeting. We came to the conclusion that it would be best, at least for the time being, to put Spark on the sidelines and try out a different method using *Stream Analytics* in conjunction with an SQL database. Stream Analytics worked like charm and we got the data across to the database with out much trouble.

10.5.2 What Went Well

- **Stream Analytics Setup**
Setting up the new service went well, even with so little time left of the sprint.

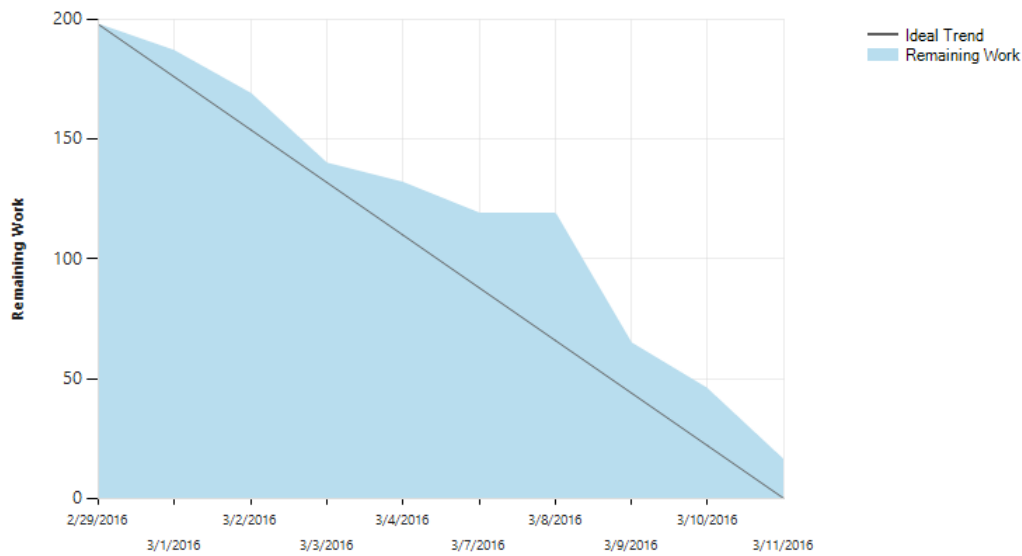


Figure 19: Sprint 3 - Burndown Chart

- Continuous Integration/Delivery
Setting up of CI and CD went well and all scripts were made ready for real world provision/deployment.

10.5.3 What Could Have Gone Better

- Spark
Spark was very hard to tame and finally we steered away from it.
- Sick team member
An unavoidable set-back, but properly reacted to by the team.
- Build Resources Depleted
The "build minutes" of our TFS system ran out, making us unable to do continuous integration for a while. Luckily a member of Marels development department began deploying a build agent that fixed this particular problem.

10.6 Sprint 4

With the 4th official sprint over, we now managed to get data all the way through the system, starting at one or more simulated devices and to being displayed in Power BI. Most of the time went into adjusting build scripts and designing the Power BI interface. The first version of a library to enable real Innova devices to be used in the system was also made. Finally, preparations were made for the second status meeting.

10.6.1 Sprint Retrospective Notes

The sprint went altogether well, although a lot of the work that went into Power BI was ultimately scrapped as we moved on to making our own web application.

10.6.2 What Went Well

- Innova Library

Although it is not yet used, a library was written that will make Innova sites able to send real world data to the system.

10.6.3 What Could Have Gone Better

- Power BI Scrapped

Quite a bit of work went into setting up and configuring Power BI, which will not be used in the final product.

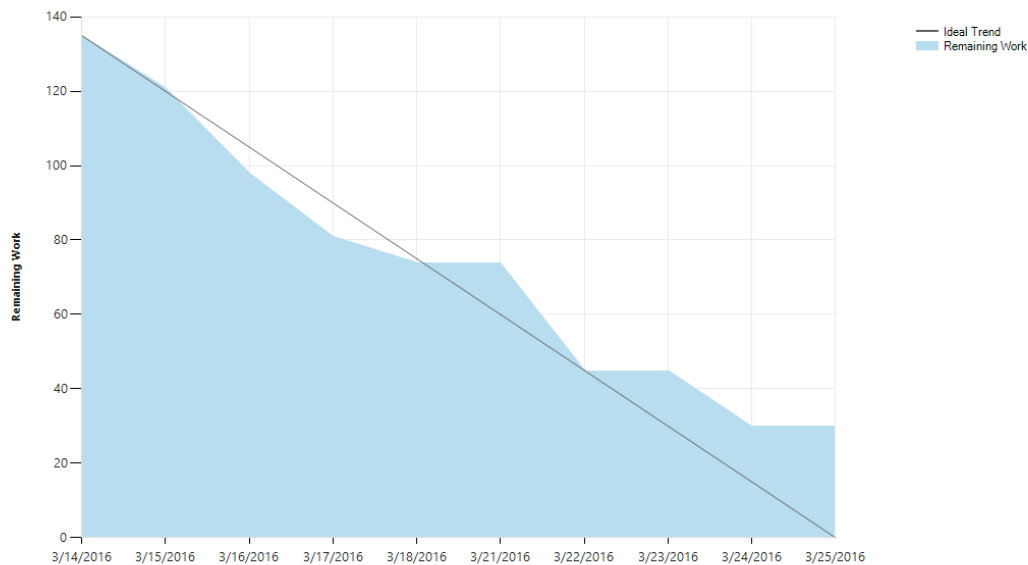


Figure 20: Sprint 4 - Burndown Chart

Title	Description	Story Points
Setup Power BI	As a developer, I want to setup Power BI so that I am able to visualize the processed data from Innova systems.	5
Development Manual	As a developer, I want to write a development manual so that new developers are able to quickly setup a development environment.	5
Continuous Integration	As a developer, I want to have continuous integration to prevent integration problems and automate tests.	13
Power BI look	As a person in customer services, I want to be able to see where on a map error occurs and a timeline so that I can assess if errors are region specific and when they happen.	8
Second status meeting	As a team member, I want to attend the second status meeting so that the instructor and examiner get a status update.	2
New project name	As a team member, I want to find a new project name.	3
Design Report version 2	As a developer, I want to write the second version of the design report so that the readers will get a clear picture of the system.	8
Slideshow for status meeting 2	As a team member, I want to create a slideshow for the second status meeting so that I can argue for the continuation of the project development.	3
Library for innova azure communication	As a developer, I want to have a library for communications between innova and azure so that logs can be easily sent to the cloud	8
Status Report 2	As a developer, I want to write the second status report so that the examiner gets a clear picture of our current progress.	3

Table 5: Sprint 4 - Backlog

10.7 Sprint 5

The sprint was used to fine tune some build scripts and to start work on the site registry part of the web application.

10.7.1 Sprint Retrospective Notes

After several very successful sprints, this sprint had the fewest logged hours of any full-length sprints. All group members had final projects in other courses to hand in, as it took place during the last two weeks before the final exams. Much of the time was therefore spent on work, unrelated to the project.

10.7.2 What Went Well

- Registry Web Application
We finally started work on the web app after deciding to move away from Power BI.

10.7.3 What Could Have Gone Better

- Few Work Hours
Not many work hours were logged during the sprint. Ideally, they should have been around 150, but only reached 85. This was, however, to be expected during a busy time at the university.

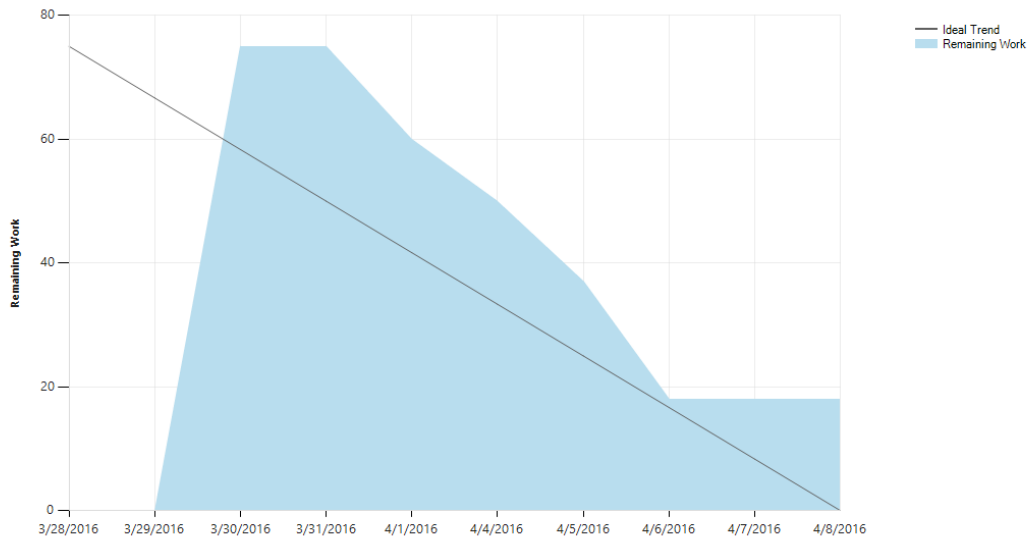


Figure 21: Sprint 5 - Burndown Chart

Title	Description	Story Points
Create Innova registry website	As person in customer services, I want to have a website so that I can add, edit, remove and monitor Innova systems.	20
Connection Strings	As a developer, I would like to automatically insert the IoT Hub connection string into the web app when I am provisioning the system so that I do not have to store it locally.	5
Registry Screen (Moved)	As a person in customer services, I would like to have a dedicated screen for managing Innova systems so that I can add, edit or remove Innova systems.	

Table 6: Sprint 5 - Backlog

10.8 Sprint 6

Further development of the web application was done in sprint 6. The registry screen was finished, as well as the first version of the opening (overview) screen.

10.8.1 Sprint Retrospective Notes

As in sprint 5, not a great deal of work hours were logged. This sprint took place during the final exams and all team members had to take some time off to study, to varying degrees. Not all team members had to take the same number of exams, and one member had to practically remove himself from the project for the duration of the sprint. The other two team members were, however, able to do considerable work in between exams and a surprising amount of work got done.

10.8.2 What Went Well

- Good Amount of Work in Spite of Exams

The remaining team members managed to cover well for the missing one and got

a considerable amount of work done, having in mind that the sprint clashed with final exams.

10.8.3 What Could Have Gone Better

- One Team Member Short
One team member could only work one day during the sprint due to final exams.

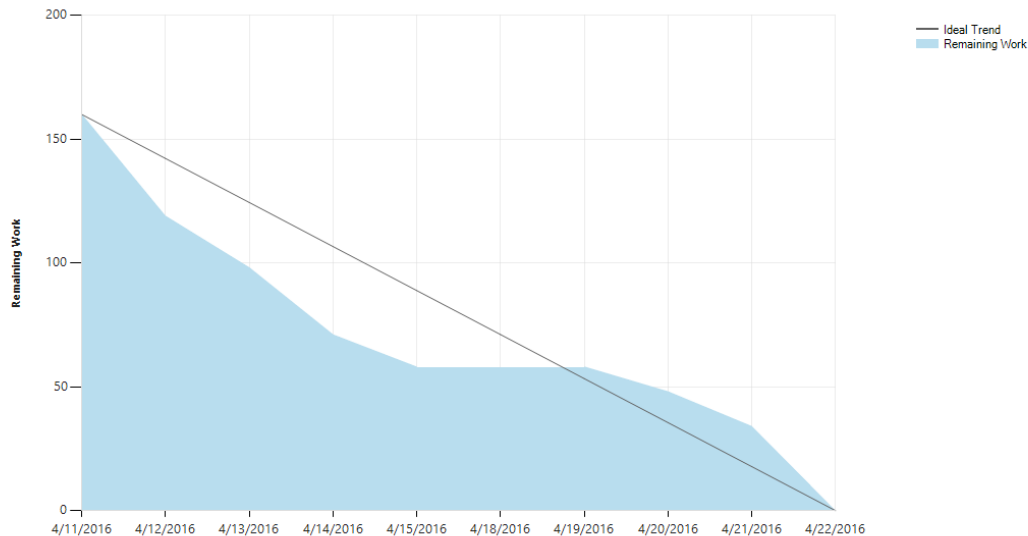


Figure 22: Sprint 6 - Burndown Chart

Title	Description	Story Points
Registry Screen	As a person in customer services, I want to have a registry screen so that I can register, edit and remove Innova systems.	20
Overview Screen	As a person in customer services, I want to have an Overview Screen so that I can monitor Innova systems.	20
Deployment Script for Web App	As a developer, I want to create a deployment script for the web app so that I can implement continuous deployment.	15

Table 7: Sprint 6 - Backlog

10.9 Sprint 7

A lot of work went into the web app during this sprint. It was essentially finished, with only minor tweaks to be made. A lot of testing was done, as well as documentation, to prepare for the final hand in. Feature freeze was set to be on May 8th, the last day of the sprint. This means that no new features should be added or any major components changed after this date.

10.9.1 Sprint Retrospective Notes

The sprint was, by far, the most successful one from the beginning with over twice the amount of work done than any other sprint (and four times as many as the least effective sprint). The web application has seen tremendous progress and has gone from only being able to register sites to showing all relevant data from these sites.

10.9.2 What Went Well

- Work Hour Record!
By far the most logged hours of any sprint.
- All Planned Work Finished
All work on the web application and other peripherals was concluded as planned.
No further features are to be added before handing in.

10.9.3 What Could Have Gone Better

Nothing, the team was very satisfied with the sprint.

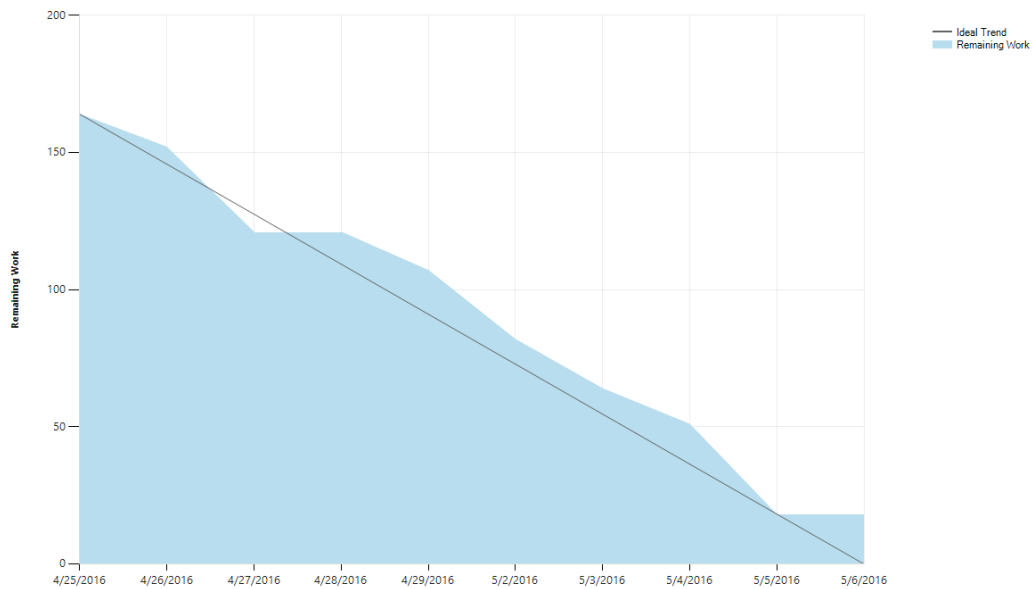


Figure 23: Sprint 7 - Burndown Chart

Title	Description	Story Points
Final report v1	As a team member, I want to write the first version of the final report so that readers fully understand the project and our decisions.	13
Final status meeting	As a team member, I want to attend the final status meeting so that the instructor and examiner get a status update.	5
Drill Down Screens	As a person in customer services, I want to be able to drill down into region, sub region and country so that I get only the information I want.	20
Finish Overview Screen	As a developer, I want to finish developing the overview screen so that customer services are able to see an overview of the world	20
Finish Registry Screen	As a developer, I want to finish developing the registry screen so that customer services are able to register new innova systems to our system	5

Table 8: Sprint 7 - Backlog

10.10 Final Sprint

The sprint was only used for some final cleaning up, testing and fixing. Some minor details were changed or added to the documentation in reflection of the final status meeting. Code freeze was on May 11th, meaning that no code was to be changed after that date.

10.10.1 What Went Well

- Feature Freeze and Code Freeze

Having a set time for feature freeze and code freeze helped us to put the project in perspective and define it as a finished product.

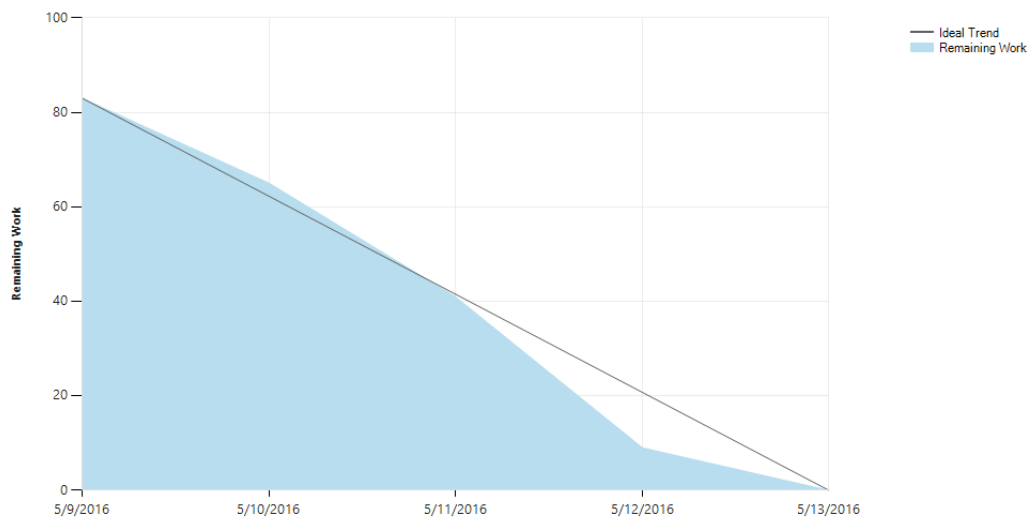


Figure 24: Final Sprint - Burndown Chart

Title	Description	Story Points
User Manual	As a future user, I want to have an user manual so that I can easily navigate and understand the full capabilities of the system.	3
Test The System	As a developer, I want to setup basic tests for the system so that I know that it works correctly.	8
Final Presentation	As a team member, I want to have an open presentation to present my project so that the public get an understanding of how important this project is.	3
Final Report	As a team member, I want to write the final report so that readers fully understand the project and our decisions.	8
Refactor Code	As a developer, I want to clean up the code base so that when development continues all unnecessary code has been removed.	5
Development Manual	As a future developer, I want to have an development manual so that I can easily and quickly start developing features for the project.	3
Administration Manual	As a future administrator, I want to have an administration manual so that I can easily provision and maintain a cloud analytics system.	3

Table 9: Final Sprint - Backlog

11 Conclusion

The project has been a great challenge for all of us. We faced problems we never knew existed and we learned how to solve and avoid many of these problems. Our project involved developing a system that consists of components we had no prior knowledge of. We have broadened our horizon by trying out new solutions and dismissing the ones that were unsuitable. It has been an intensive and extensive learning process that will be useful in our future work as computer scientists. At Marel and Innova development department, we were embraced with a wonderful company spirit that made it easy for us to settle in. They provided us with an excellent professional and technical support that was essential for our project.

Our instructor has been a source of inspiration and support. Her assistance was important when defining the content of the project and confining its boundaries. The status meetings have been invaluable as the instructor has provided us with an outsider's perspective of the project, and assessment of the extent to which our progress was satisfactory.

11.1 Challenges

As already mentioned, we have faced great challenges throughout the development process of our system. The first and perhaps biggest challenge was to choose which tools and services to use. This was an on-going consideration as we tried out many different combinations before making our final choices. We had little or no prior knowledge of any of the services that we included in the various versions of the system design. When we finally settled down on a good system design, we still had to learn how to use the components of the system effectively.

11.2 Future

After handing in the project results as a bachelor degree project to Reykjavík University, we intend to develop our system further. We will do this as employees at Marel for the next 4 months. It is our intention to make the system suited for a commercial release.

We are very excited to continue our work on our system, and we have already many ideas about its future capabilities.

11.3 Review from Product Owner

„Á undanförunum mánuðum hafa Jón Reginbald Ívarsson, Gísli Rafn Guðmundsson og Gunnar Páll Gunnarsson unnið að lokaverkefni hjá Marel. Verkefnið er rannsóknar og þróunartengt sem hefur falist í því að rannsaka hvernig hægt sé með öruggum hætti að flytja ýmsar lykilupplýsingar úr Innova framleiðslustýringarkerfum sem eru í notkun víðsvegar um heim allan í eitt miðlægt kerfi. Þessar upplýsingar veita mikilvægar upplýsingar um ástand kerfanna ásamt því að veita innsýn í hvernig kerfin eru notuð og hvaða hlutar kerfanna eru í raun notaðir. Að lokinni ýtarlegri rannsókn var kerfið hannað og útfærður hugbúnaður fyrir sendingar úr Innova kerfum ásamt móttöku á gögnum og frekari úrvinnslu í miðlægu kerfi. Viðmót var einnig hannað og þróað sem sýnir m.a. á greinagóðan hátt ef alvarlegt ástand kemur upp í einstaka Innova kerfi hvar sem er í heiminum. Þetta veitir Marel einstakt samkeppnisforskot í formi aukinnar þjónustu fyrir viðskiptavinum Marel.

Verkefninu hefur fylgt ýmsar mjög krefjandi áskoranir og hefur hópurinn leyst þær með afbrigðum vel. Á öllum stigum verkefnisins hefur hópurinn unnið af mikilli fagmennsku og stundað öguð vinnubrögð. Áætlanagerð, hönnun, skjölun og útfærsla verkefnisins hefur öll verið til mikillar fyrirmyndar og hefur verið mjög ánægulegt og þægilegt að vinna með hópnun.

Verkefnið uppfyllir allar kröfur og væntingar sem Marel gerði til verkefnisins auk þess að hafa fært sönnur fyrir því að þessi lausn býr yfir ótal fleiri spennandi og áhugaverðum möguleikum sem verða þróaðir í nánustu framtíð.“

Karl Karlsson

Vöruþróun Innova hugbúnaðarlausna.

Appendix A: Automation Scripts

Script overview

A list of the scripts written to provision the Azure services along with a short description. The scripts are listed in the order they are executed. File extensions and prepends (the name of the service in each case) are omitted to save space.

Azure Rm

ServicePrincipal	Used to create a service principal, which is essentially a user for the build script to use when provisioning. Only needs to be run once and the data from it used in the Login script.
Login	Logs into Azure using the information provided by the service principal script. This is done to avoid log-in window popping up each time a script is executed.

Provision / Delete All

Provision	Simply runs all currently used provisioning scripts. All services should be correctly named and configured using parameters set at the top of the script. It correctly connects together services that need access keys by extracting the key from the output of a script that provides it.
Delete	Tears down all services one by one and deletes the resource group.

Resource Group

Provision	Creates a new resource group. Nothing is done if the resource group already exists
Deletion	Deletes the resource group. WARNING: When the resource group is deleted, every service that belongs to it is also deleted. This script should therefore be run last, and is not a preferred method of deprovisioning the system.

IoTHub

Provision	Provisions and deploys an IoTHub using Azure's Resource-GroupDeployment, which allows us to deploy, using a JSON template with custom defined parameters.
Deletion	Tears down the IoTHub created by the previous script. At the time of writing the script is broken and not used in the delete all script. The IoTHub is now torn down by deleting the resource group instead.
CraftConnString	When the entire system is provisioned connection keys for IoTHub are extracted. For these keys to be useful this script is run to craft them into a connection string that is useful for the web app.

SQL Database

Provision	Provisions and deploys both an SQL server and a database. It also creates a firewall rule to allow user access.
CreateTable	Creates a single table using the schema at the specified path (relative to the 'SQL Database' directory in 'Includes' at the powershell script root) on the desired SQL database. Can be run multiple times to generate more tables.
Deletion	Tears down the SQL database and server specified.

Stream Analytics

Provision	Provisions and deploys a Stream Analytics job onto Azure. The path to a transformation query (relative to the 'Stream Analytics' directory at the project root) should also be specified, containing the query used to link the chosen services.
AddInput	Adds a new input to the job. This input should be an IoT Hub. The script then generates a JSON file that is used to create the input request (since these files can not have parameters). A Stream Analytics job can have multiple inputs.
AddOutput	Adds a new output to the job. This output should be an SQL server. The script then generates a JSON file that is used to create the output request (since these files can not have parameters). A stream Analytics job can have multiple outputs.
Start	Starts a provided job. Is run in the Provision.All script to avoid having to manually start the job.
Stop	Stops a provided job. Is run in the Delete.All script to minimize unnecessary errors.
Deletion	Tears down a Stream Analytics job.

Web Application

Provision	Provisions the web app. The build server then handles deploying the newest version of the app to the system provided that no tests fail.
Dependencies	Installs all dependencies using Npm and Bower.
Build	Builds the front-end of the web app using Gulp.
Test	Runs front-end unit tests using Karma.
Deletion	Tears down the web app.

Miscellaneous and Deployment

AddConnString	Adds the IoTHub connection string, created by the Craft-ConnString script to the Azure web app. This way the app can access the string directly while deployed to Azure as an environment variable.
NpmInstallGlob	Installs packages that need to be installed with global flag using Npm.
PublishPackage	A workaround script that is needed by the build server to be able to extract the web server correctly when deploying.

Appendix B: Code Coverage

Introduction

The following sections depict and explain the code coverage reached for front-end unit tests, firstly, and back-end unit tests secondly.

Code coverage for the front-end unit tests

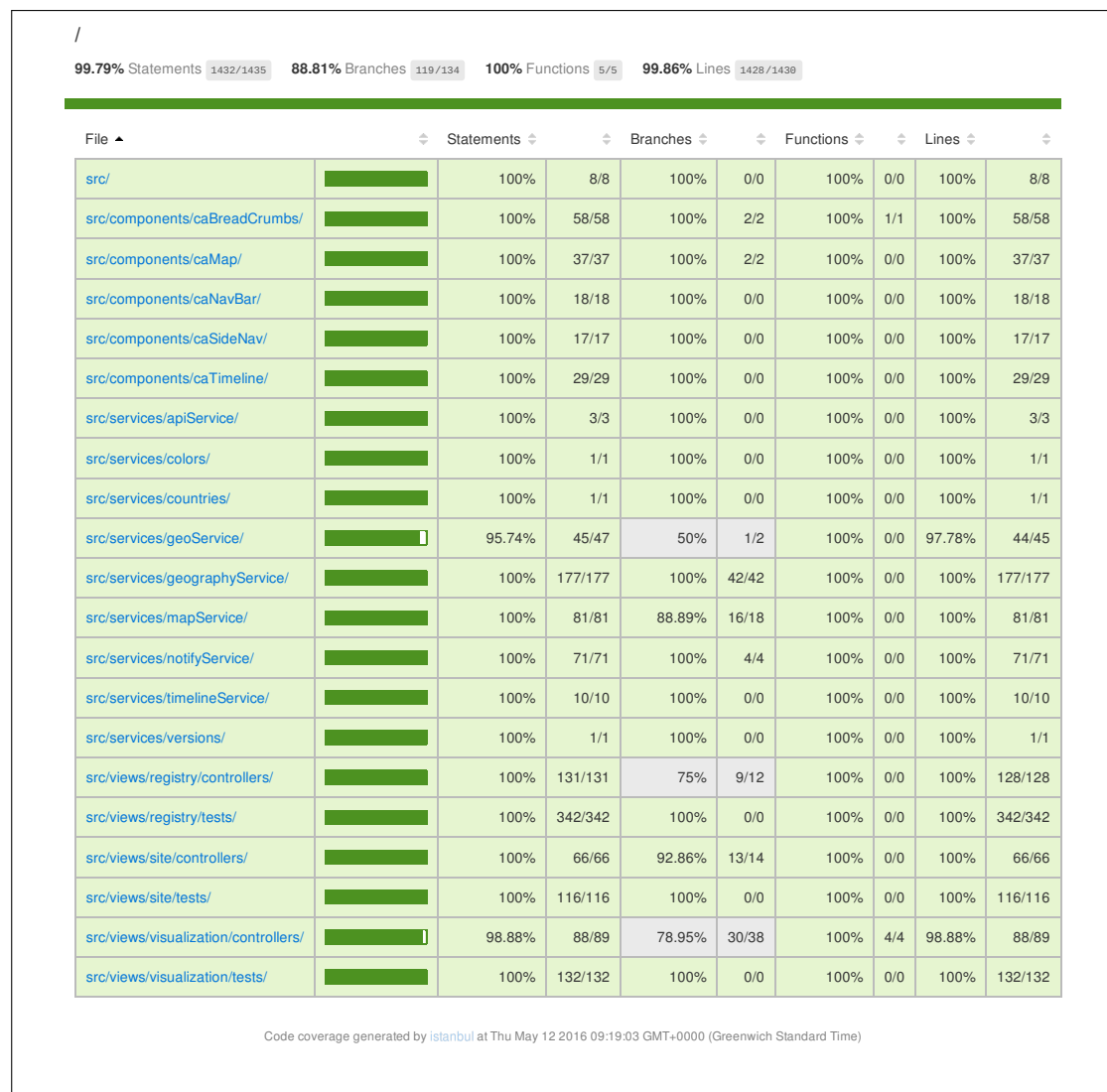


Figure B.1: Front-end Code Coverage

Appendix B: Code Coverage

Code coverage for the back-end unit tests

Summary

Generated on:	11/05/2016 - 13:25:00
Parser:	VisualStudioParser
Assemblies:	2
Classes:	4
Files:	4
Covered lines:	663
Uncovered lines:	4
Coverable lines:	667
Total lines:	667
Line coverage:	99.0%

Assemblies

▼ Name	▼ Covered	▼ Uncovered	▼ Coverable	▼ Total	▼ Line coverage	▼ Branch coverage
— cloudlogger.dll	69	1	70	157	98.5%	
— MareI.CloudLogger	69	1	70	157	98.5%	
MareI.CloudLogger.CloudLogger	69	1	70	157	98.5%	
— innovaregistry.services.dll	594	3	597	936	99.4%	
— InnovaRegistry.Services.Services	594	3	597	936	99.4%	
InnovaRegistry.Services.Services.AnalyticsServiceProvider	316	0	316	428	100%	
InnovaRegistry.Services.Services.RegistryServiceProvider	123	3	126	261	97.6%	
InnovaRegistry.Services.Services.SiteServiceProvider	155	0	155	247	100%	

Generated by: ReportGenerator 2.4.5.0

11/05/2016 - 13:25:01

Figure B.2: Back-end Code Coverage

Appendix C: Wireframes

Wireframes

Wireframes are initial user interface designs for the system. They should describe what the interface of the web application should look like in a clear and concise way, which can then be used as a blueprint to create the final interface. However, they are not a final version of the user interface and the final design may differ from the wireframes.

The attached User Manual shows screenshots of the final product.

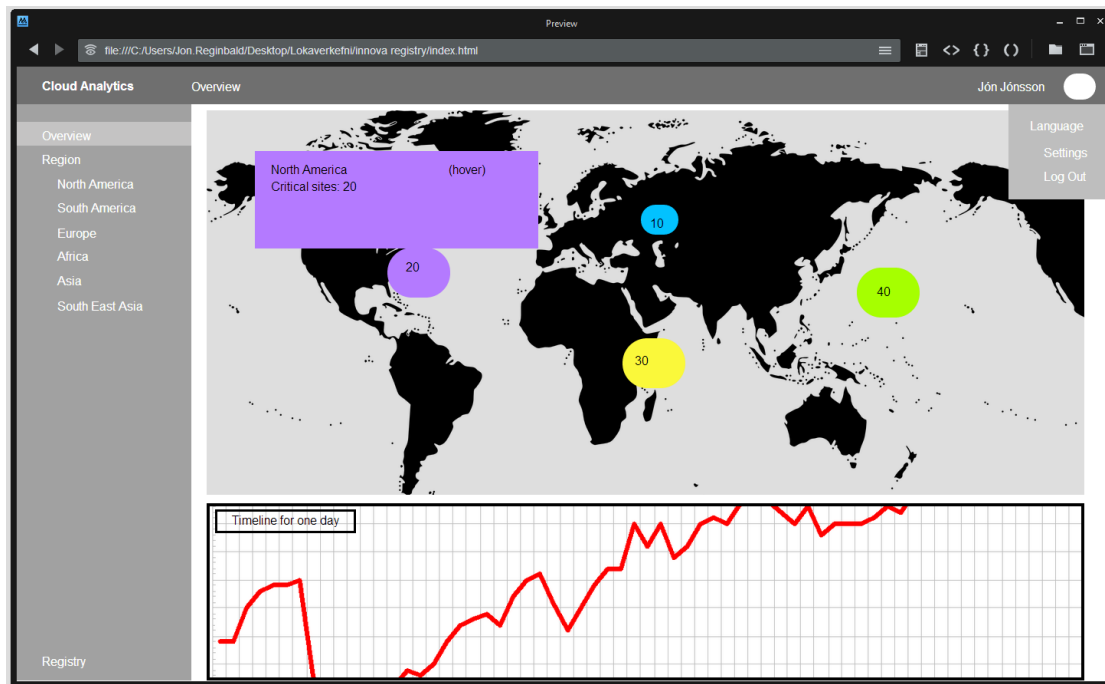


Figure C.1: Overview Screen

Appendix C: Wireframes

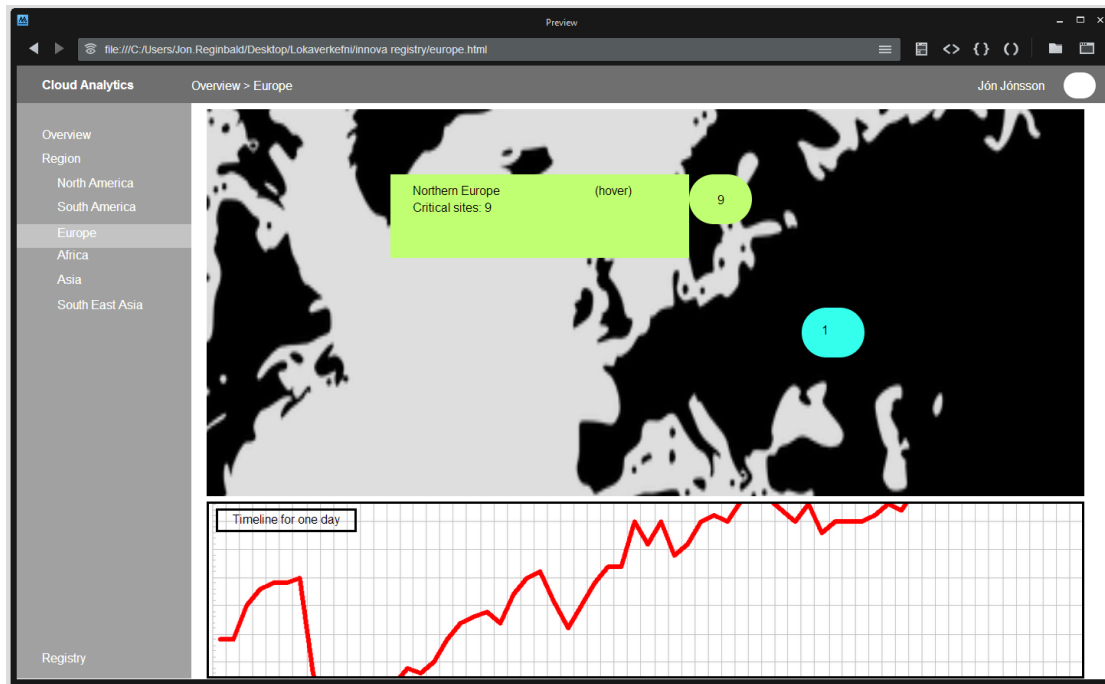


Figure C.2: Region Screen

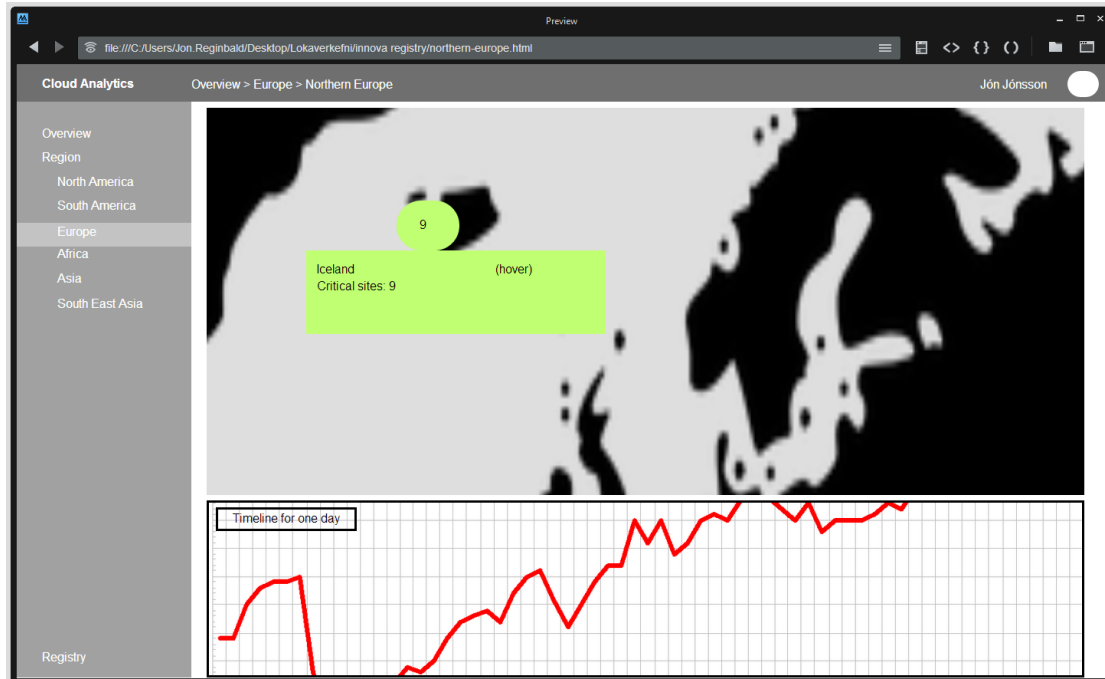


Figure C.3: Sub Region Screen

Appendix C: Wireframes

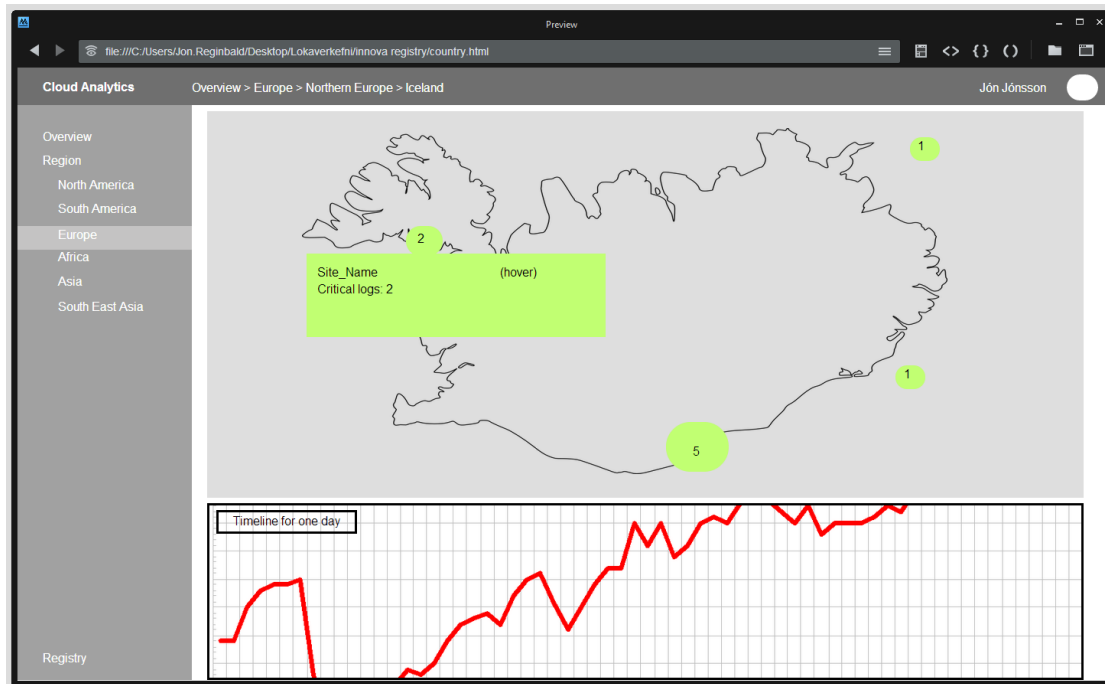


Figure C.4: Country Screen

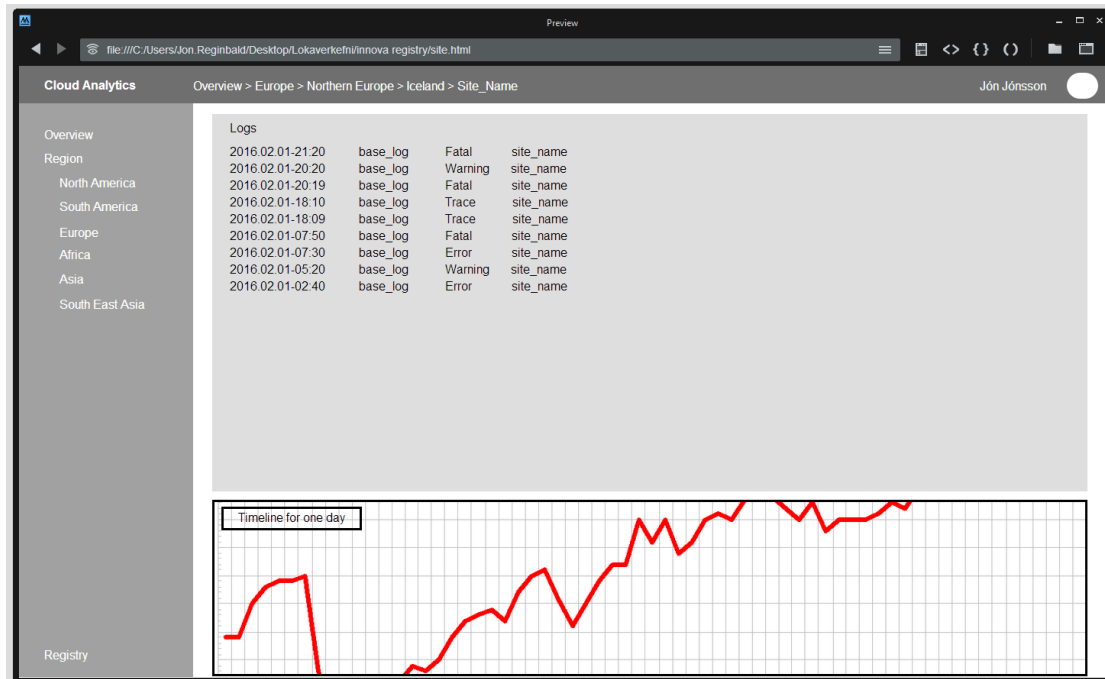


Figure C.5: Site Screen

Appendix C: Wireframes

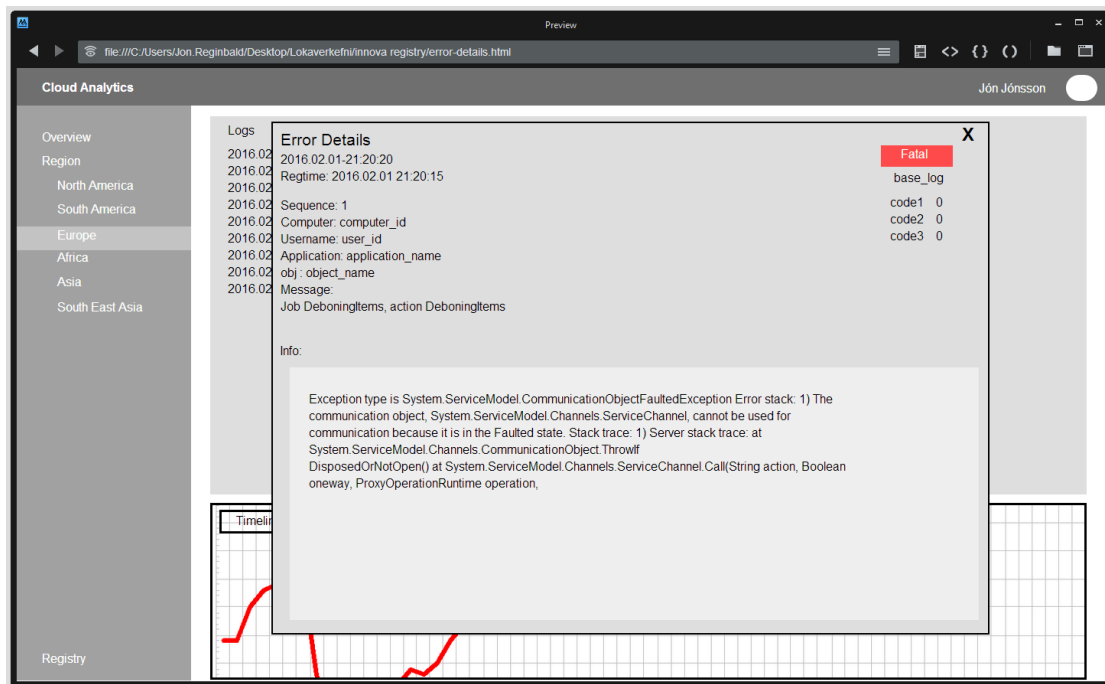


Figure C.6: Error Details Dialog

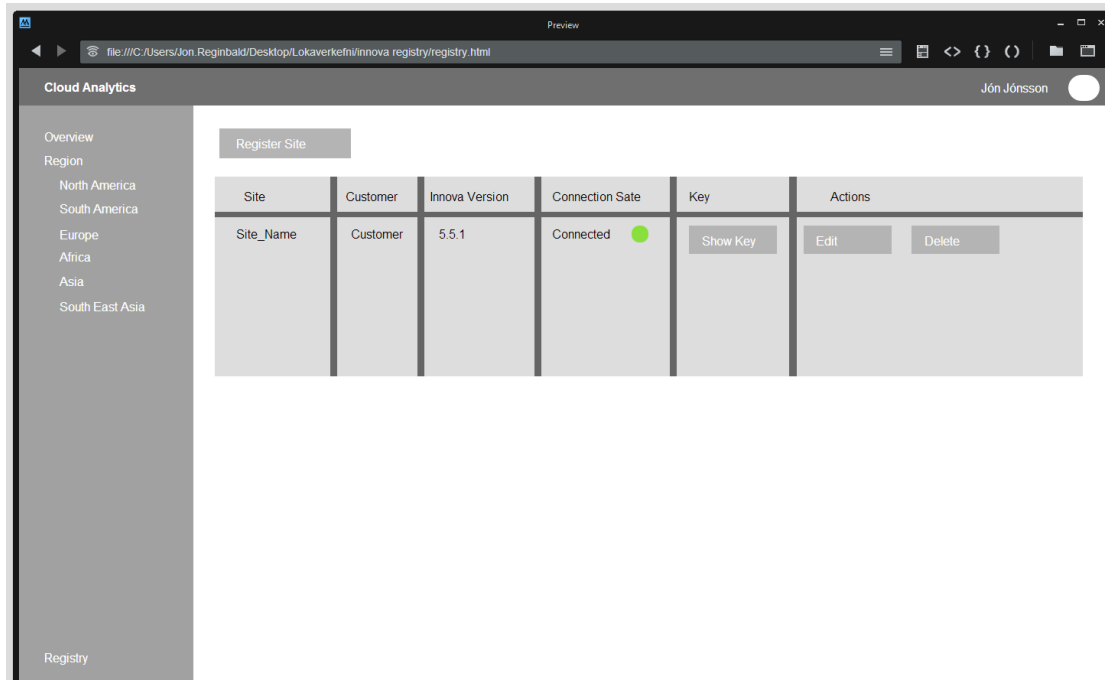


Figure C.7: Registry Screen

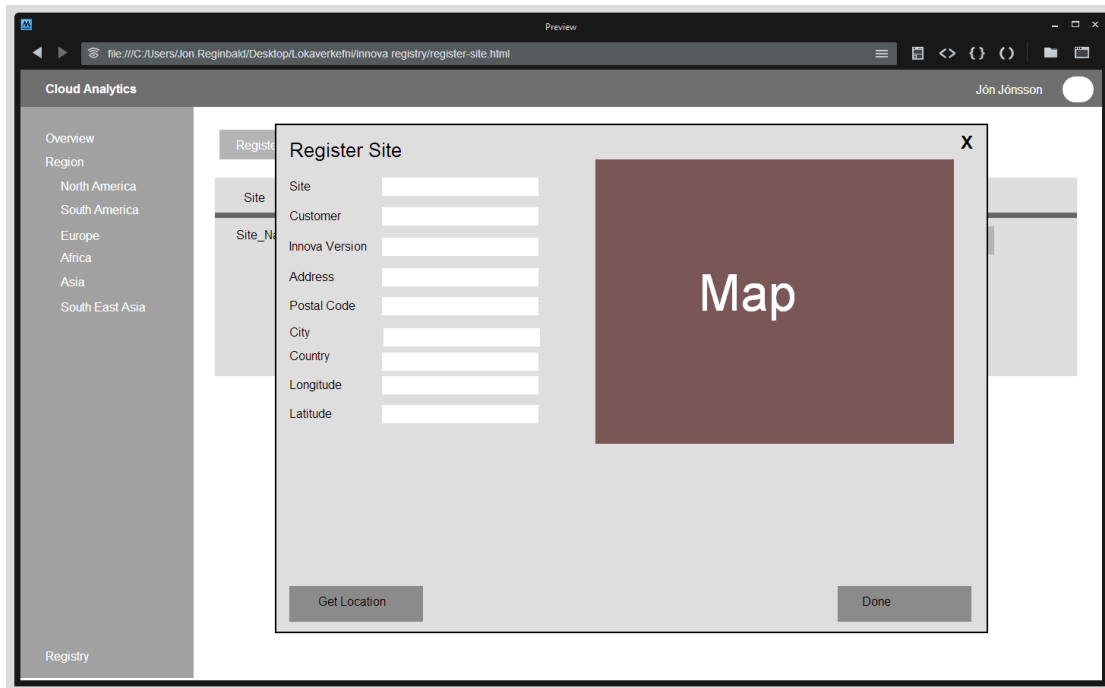


Figure C.8: Register Site Dialog

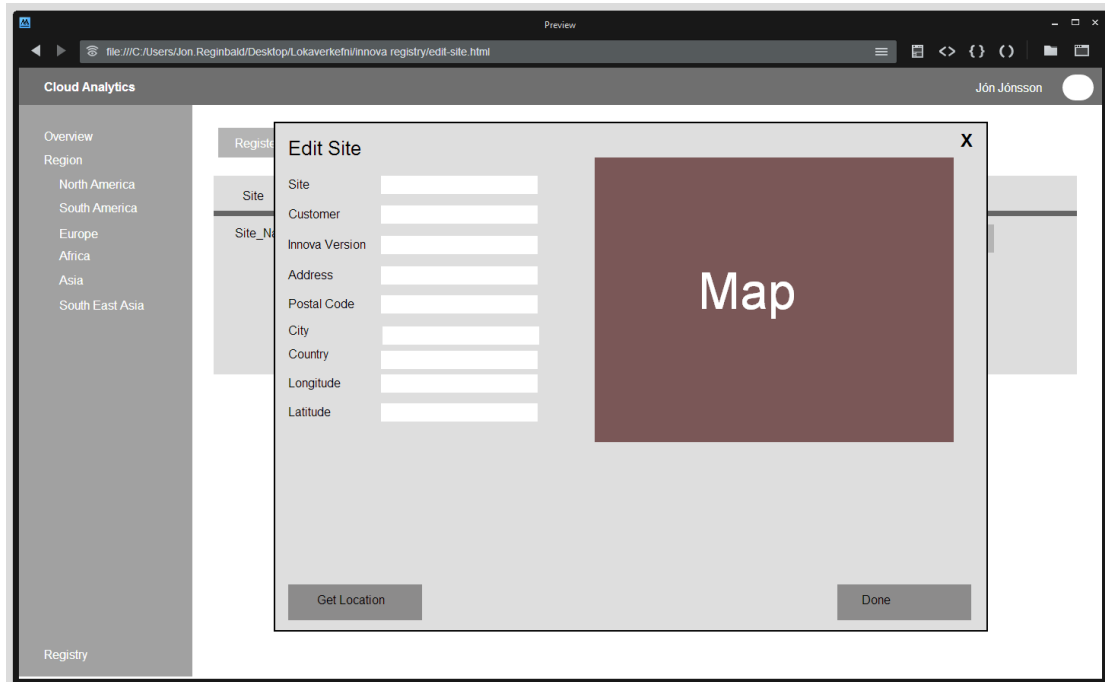


Figure C.9: Edit Site Dialog

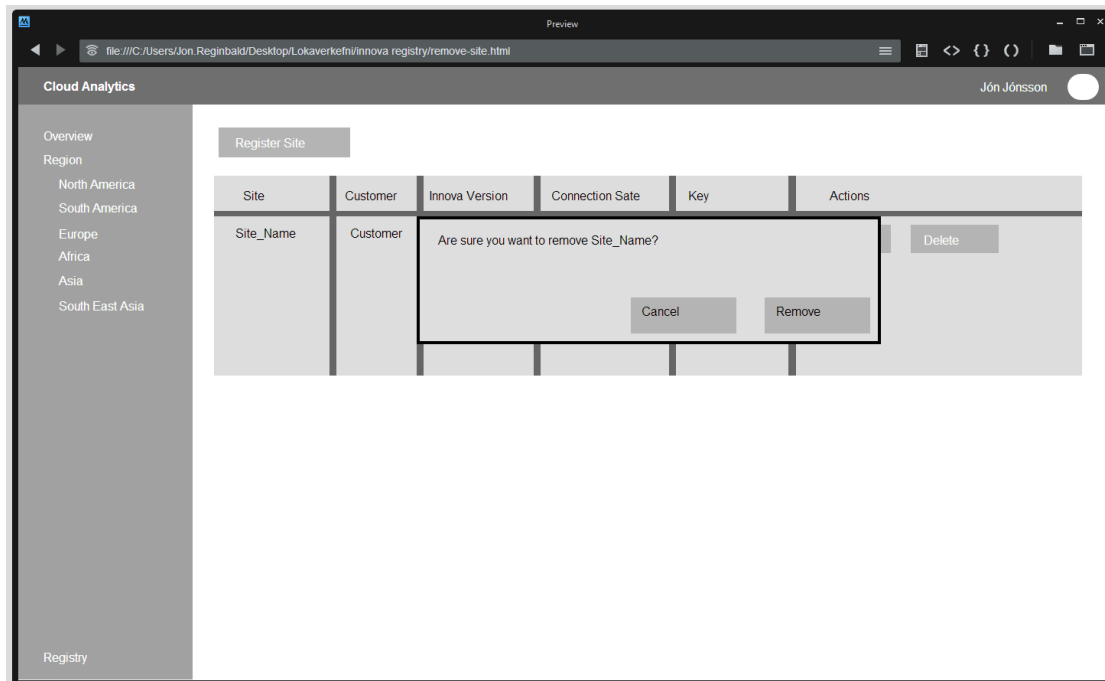


Figure C.10: Remove Site Dialog