



“KPI WITH POWERBI”

Streamlining report building with PowerBI and PowerShell

Name: Anders Ingelsten
Student ID: 20095402

Supervisor: Anita Kealy

Module: Project

Course: Higher Diploma in Computer Science

Contents

Table of Figures	4
Abstract	7
1 Introduction	8
1.1 Project Type	8
1.2 The organisation	8
1.3 System Background and Project Scope	8
1.4 Fieldview.....	9
1.5 Smartsheet	10
1.6 Sharepoint/Office365.....	10
1.7 Situation at the time of the project	10
1.8 What were the problems.	11
2 Use Case.....	11
3 Technologies, Tools, and Languages.....	11
3.1 Potential Issues	12
4 Objectives.....	12
5 Methodology	12
5.1 Dataflow	12
5.2 Modelling.....	13
5.3 Feedback Loop	13
6 Technologies.....	14
6.1 PowerBI	14
6.2 SOAP API.....	14
6.3 PowerShell	14
6.4 Power Automate	15
6.5 DAX.....	15
7 Project Plan	16
7.1 Project Planner	16
8 Implementation.....	17
8.1 Phase 1: Research & Training	17
8.2 Phase 2: Smartsheet Data Connector setup.....	20
8.3 Phase 3: FV API setup and Data Persistence	21
8.3.1 Data persistence.....	23

8.4	Phase 4: PowerBI Dashboard – Fieldview Vehicle-check	24
8.5	Phase 5: PowerBI Dashboard – Pipedrive	28
8.6	Phase 6: PowerBI Dashboard – SmartSheet Timesheet	30
8.7	Phase 7: PowerBI Dashboard – Fieldview Timesheet	31
8.8	Phase 8: PowerBI Dashboard – Asset Manager	35
8.9	Phase 9: Automation of Data flows and Power BI Refresh	38
8.9.1	Power Automate – Domain Controller Server	38
8.9.2	Power Automate – Domain Server - Reverting to Task Scheduler	43
8.9.3	Task Scheduler - Webserver	47
8.9.4	PowerBI – Automated Refresh and Publish	48
9	Project Evaluation	53
9.1	Achievements	53
9.2	Knowledge attained.	53
9.3	Future Learning	53
9.4	Future Development Work	54
9.4.1	Review of the current state	54
9.4.2	Adding more data sources	54
9.4.3	Long Term future functionality	55
10	Conclusion	56
11	References	57

Declaration I declare that the work which follows is my own, and that any quotations from any sources (e.g., books, journals, the internet) are clearly identified as such by the use of 'single quotation marks,' for shorter excerpt and identified italics for longer quotations. All quotations and paraphrases are accompanied by (date, author) in the text and a fuller citation is the bibliography. I have not submitted the work represented in this report in any other course of study leading to an academic award.

Student..... Date

Workplace Mentor..... Date

Table of Figures

Figure 1: Overview of existing potential systems available to the project	8
Figure 2: Table of systems	9
Figure 3: Sample view of Fieldview mobile form	9
Figure 4: Sample view of the Fieldview widget reports	9
Figure 5: Sample view of Smartsheet form.....	10
Figure 6: The flow of data from Fieldview to PowerBI	13
Figure 7: Sample representation of the data model in PowerBI.....	13
Figure 8: Sample of DAX language in the formula bar of Power BI	15
Figure 9: Table of milestones	16
Figure 10: Table of phases	16
Figure 11: Microsoft Planner – Chart of Buckets	16
Figure 12: Microsoft Planner - Status of tasks and Buckets	17
Figure 13: Sample of input Parameters	18
Figure 14: SOAP request and response	18
Figure 15: Returned information.	18
Figure 16: Call quota per API token.....	19
Figure 17: Certificate of Completion	19
Figure 18: Get Data view in PowerBI of Online Services	20
Figure 19: Preview of Connected Table ready to be loaded into PowerBI.	20
Figure 20: View of Created API Tokens in Fieldview	21
Figure 21: Code Snippet of the New-WebServiceProxy used to pull data from Fieldview.....	21
Figure 22: Code snippet of foreach loop with an if else statement	22
Figure 23: Screenshot of Sample Data from the GetProjectFormsList() call.	22
Figure 24: Sample Code of writing a file to Sharepoint Online.....	23
Figure 25: Screenshot of successfully writing the data to Sharepoint Online in PowerShell	23
Figure 26: Code of File export Sample.....	24
Figure 27: Export to SharePoint with automated details	24
Figure 28: Form data as exported from FV.	25
Figure 29: DAX code of Count	25
Figure 30: View of Counts.....	25
Figure 31: View of Embed Link.....	26
Figure 32: View of admin message.....	26
Figure 33: View of embed code view.	26
Figure 34: View of Sharepoint Online Embed function.	27
Figure 35: View of Sharepoint Intranet	27
Figure 36: Code snippet of Pipedrive script	28
Figure 37: View of table relationships	29
Figure 38: View of Values per Quarter	29
Figure 39: Report view of logged activities.....	30
Figure 40: View of Smartsheet timesheet table.....	30
Figure 41: View published on Company Intranet.....	30
Figure 42: Drill down view Timesheet	31

Figure 43: Code Snippet of 4 answer queries	31
Figure 44: Code snippet of API call.....	32
Figure 45: Code snippet of export to SharePoint.....	32
Figure 46: Query view in PowerBI	33
Figure 47: Table relationships in PowerBI	33
Figure 48: Summary visualisation of the hours per contract and user	34
Figure 49: Drill down view per contract, month, and person.	34
Figure 50: View of Devart Configuration	35
Figure 51: View of connected Data Source.....	36
Figure 52: View of Tables available from Asset Manager.....	36
Figure 53: Query of the latest service ID	37
Figure 54: Table query of the latest service ID	37
Figure 55: View of the joined table relationship.	37
Figure 56: View of asset types and corresponding table.....	37
Figure 57: Installation of Power Automate desktop on domain controller server.	38
Figure 58: Before implementation of flows	39
Figure 59: Creating the Run PowerShell Script on Power Automate Desktop	39
Figure 60: Adding the script to run in the Flow.	40
Figure 61: View of all the 5 PowerShell scripts that was setup on the Power Automate desktop.	40
Figure 62: Recurrence step on the Power Automate Cloud	41
Figure 63: Running the Power Automate desktop flow on the Power Automate Cloud	41
Figure 64: Connection used for cloud hosted servers to run the PowerShell scripts.	42
Figure 65: Sample of a flow -this running multiple scripts	42
Figure 66: Summary view of the cloud flows.....	42
Figure 67: Initial setup with run history	43
Figure 68: Successful runs of flows	44
Figure 69: View failed flows.....	44
Figure 67: Detailed view of a failed flow	44
Figure 71: Detailed flow error message	45
Figure 72: Turning off all the Cloud Flows	45
Figure 73: Setting up the Task Scheduler	45
Figure 74: Sample view of the export/import XML file of a task	46
Figure 75: Summary view of the 5 PowerShell scripts in the Task Scheduler.....	46
Figure 76: Code snippet of copy code	47
Figure 77: View of Task Scheduler setup to run script every 15 min.	47
Figure 78: View of Scheduled copy task in situ with running history.....	48
Figure 79: PowerBI Datasets view for automated updates.	49
Figure 80: On-Premises data gateway.....	49
Figure 81: Gateway connected.	50
Figure 82 :Authentication of data sources.....	50
Figure 83: Scheduled refresh view	51
Figure 84: Refresh History	51
Figure 85: PowerBI refresh error.....	52
Figure 86: SonicWALL Netextender.....	52

Figure 87: Change of Task Scheduler..... 54

Figure 88: DAX formula to resolve weighted monthly value. 54

Figure 89: PowerBI view of weighted monthly value..... 54

Figure 90: Sample of requested view 55

Abstract

Previously data for KPI reports within the company were pulled manually from locally hosted and remotely hosted cloud-based systems. The data was then processed manually by staff into report visuals.

This project's objective was to streamline and automate this process. By pulling data via system APIs and then making it available to PowerBI, by storing it into the organisation's SharePoint Online as CSV Data, this was achieved.

PowerShell was used as the scripting language to facilitate the data pull and storage. PowerBI, in conjunction with the formula language DAX, was used as the visualisation tool, to display the stored data, as up to date organisational interactive KPI reporting diagrams on the company's intranet. Automation was applied to the data pull and the PowerBI data refresh.

1 Introduction

1.1 Project Type

This was a work-based project, where I improved the reporting process within several business areas. I learned new skills in relation to developing a KPI dashboard using the knowledge acquired from the course modules. The internal company mentor was the company's Chief Financial Officer. This meant that I was able to spend time on this project as part of my day-to-day work.

1.2 The organisation

Company A is an engineering solutions provider operating in Ireland, the UK and Scandinavia. It provides a wide range of services from the Design & Build of Sub-stations to construction of Airside Aviation Infrastructure to Turn-key Wind & Solar Energy Solutions. The company has a turnover of approx. 30million euro and employees approximately one hundred staff.

1.3 System Background and Project Scope

Company A has several systems that hold business information. Finance, Health and Safety and Operations staff query this data on a regular basis to produce business performance reports and to generate KPI (Key Performance Indicators) reports for the different departments. The reports are generated on a weekly or monthly basis but there are also ad hoc reports.

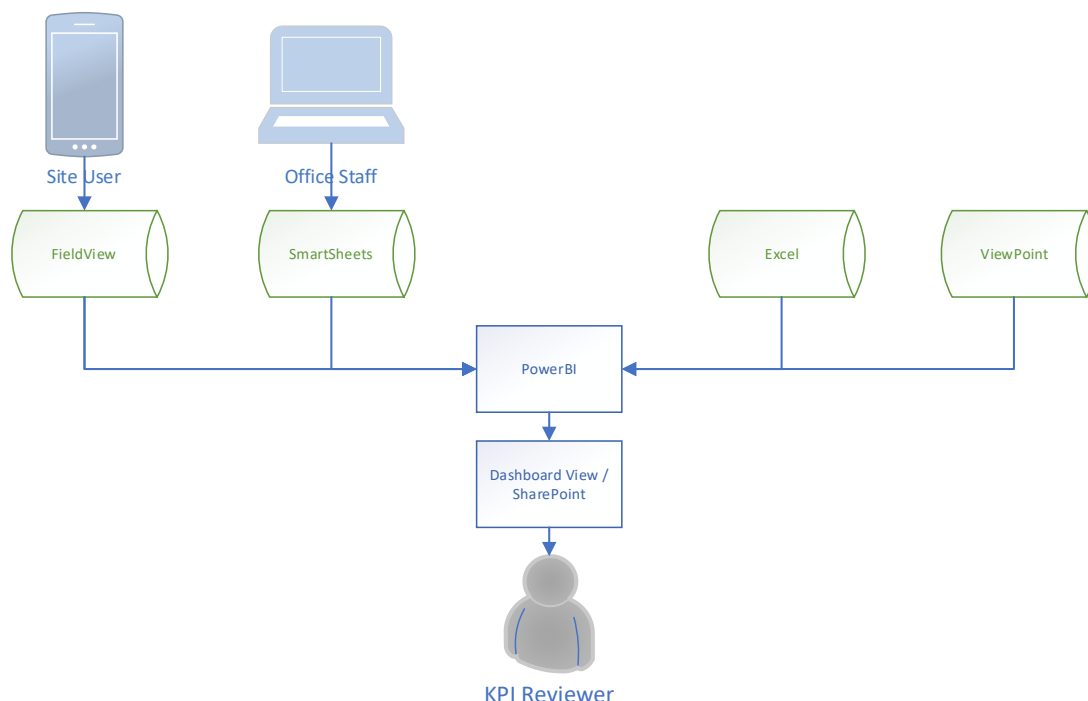


Figure 1: Overview of existing potential systems available to the project

The current systems identified for the initial project scope are:

System	Function	Location
Fieldview	Record QA/QC/Admin forms and timesheets for site staff	Cloud
Smartsheet	Record timesheets for Salaried Staff	Cloud
Office365	SharePoint, Excel, word etc	MS Cloud

Figure 2: Table of systems

1.4 Fieldview

Fieldview is a third-party cloud-based and off-line mobile solution developed by Trimble (Floor et al., n.d.). It is used in Company A to replace paperwork on site. Users are equipped with a mobile device (phone and tablet), where the app has been installed. The users log in and use the application to for snagging tasks and to produce reporting forms. When the mobile device is synced – data is pushed from the device to the cloud hosted database.

The image shows a mobile application interface for a time sheet entry. At the top, it displays 'F215017.36 - Coins TimeSheet - 14/1/2023' along with status icons for 'All Good', 'Distribute', and 'View Report'. Below this, it shows ownership details: 'Owned By: Anders Ingelsten', 'Location: MP-General', and 'Date Raised: 14/1/2023, 06:35'. There are buttons for 'Copy' and 'Opened'. The form includes fields for 'Completed By *', 'Contract *', and 'Date *' (set to 14/1/2023). Each field has a dropdown menu and a set of icons for additional actions like adding attachments or comments.

Figure 3: Sample view of Fieldview mobile form

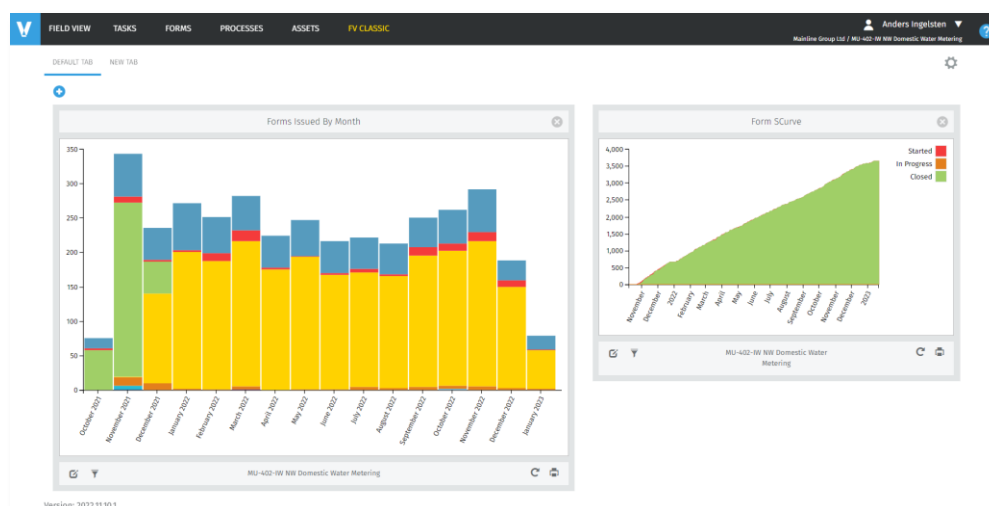


Figure 4: Sample view of the Fieldview widget reports

1.5 Smartsheet

Smartsheet is an online hosted solution that allows organisations to plan, track, automate, and report on work (Smartsheet, 2019). Company A uses the online application to track timesheets for salaried staff.

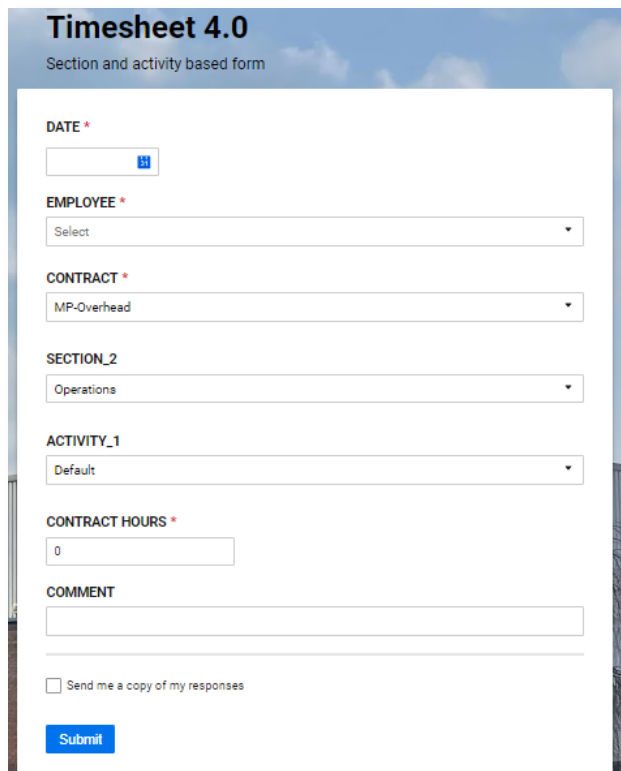


Figure 5: Sample view of Smartsheet form

1.6 Sharepoint/Office365

During the spring of 2022, the organisation fully migrated its whole IT environment to the Sharepoint Online cloud, i.e., the full file repository of the organisation, approx. 2.5 TB of files is now hosted online with 24/7 365 access.

SharePoint Online is a cloud-based SAAS - “software as a service” provided by Microsoft, where organisations and users store and share information and use it collaboratively (Www.microsoft.com, n.d.). All data and software are hosted on Microsoft own servers. Access, storage, and use of software is done by using a subscription model.

1.7 Situation at the time of the project

The company had, over the summer of 2022, revised the KPI process and in September rolled out a new KPI process where departmental stakeholders fill in excel spreadsheet reports with various KPI data. For example, the number of hours spent by employee per contract and other data like work site accident frequency etc. The departmental stakeholders access several systems and pull-down pdf and

excel reports, from where they extract data and then collate it into excel spreadsheets which is then presented to the leadership team. This process is repeated monthly.

The company was aware that this was a time-consuming process and when it's well established, the company was looking to improve the efficiencies in this process. The company had already identified that business intelligence tools like Power BI could be used to display almost real-time dashboard report visuals of the business data. During 2022 there was internal talks about identifying and bringing in business intelligence consultants/developers to do this work during 2023. Within the project, I investigated business intelligence products for example the Microsoft Power BI platform.

1.8 What were the problems.

The two main problems were:

1. Manual input of data by staff into excel spreadsheets was time consuming. This time can be better spent on other business processes and tasks.
2. The time difference between the live situation and the compiled report by staff, lead to delays in business understanding, for example, if work is profitable or not.

The CFO of Company A was therefore looking for a solution that could display KPI information from systems on the internal company SharePoint internet website.

2 Use Case

The primary reason for developing a dashboard pulling data from sources is the time saving. For example, the Fieldview application is structured in such a way, that when a user would like to do a count of reports done in a week, he will have to log in, navigate to the report view, enter a date range, then export to excel, and in excel analyse the data.

A simple calculation shows how a dashboard can provide efficiencies: A user spends 1.5 hour every week to access Fieldview and compile a report of the amount of vehicle reports submitted by staff that week. $1.5 \text{ hours} \times 48 \text{ working weeks} = 72 \text{ Hours}$ which equates to 9 working days per year.

3 Technologies, Tools, and Languages

As this was a work-based project the preferred direction from the company supervisor was to use known technologies that the company had access to or was already using. The two main reasons for this were that it creates greater resilience, and costs are known, for example Company A hosts Windows 2019 servers in the Azure cloud.

List of potential technologies, tools, and languages:

- Office365
- SharePoint
- Power BI Desktop
- Power BI Data Gateway

- Power Query Editor
- SOAP UI
- MS Planner
- Html
- PowerShell
- Sharepoint Online Management Shell
- M language and DAX

3.1 Potential Issues

As the project had access to commercially sensitive and potentially personal data, measures were put in place to minimise exposure of this data. This limited the scope of which systems and data that was incorporated into the project. If it was not feasible to limit the exposure, any personal or corporate sensitive data was redacted from screenshots of views and code.

4 Objectives

This project had three main objectives:

- **Ease of access.** By connecting separate data sets, transforming, and cleaning the data into a data model and creating charts or graphs to provide visuals of the data; it will assist users to find insights, within the organisation, of the operational data generated in Fieldview.
- **Real-time information.** To give users in the organisation, an updated almost real-time view of the situation in the company. This should provide the ability to solve problems and identify issues and opportunities.
- **Process improvement.** By streamlining publication and the distribution of the data into dashboards – users interpret published data whenever the underlying dataset is updated. This is instead of compiling reports through a time-consuming process and sharing the data in emails or a shared drive for stakeholders to then review.

5 Methodology

5.1 Dataflow

The data was envisaged to flow in the following way.

1. Users generate data onsite in the Fieldview app.
2. By syncing the device, the data generated in the app is pushed to the Fieldview server environment (Cloud)
3. The PowerShell scripts will then run automated on a server hosted in the Company A's Azure environment, pulling data from the Fieldview cloud.
4. Data is then saved in CSV files by the PowerShell scripts to Sharepoint Online.
5. PowerBI is then connected to the CSV files and refreshed on a regular basis.

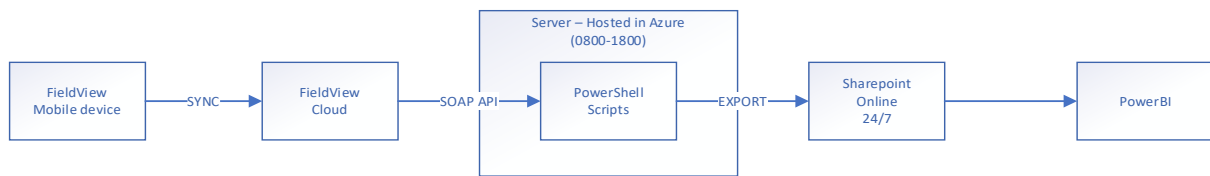


Figure 6: The flow of data from Fieldview to PowerBI

5.2 Modelling

Below is a simple representation of modelling of the data in Power BI, it was envisaged that data was presented in several tables. These tables are then joined pending any queries that they may relate to. These tables and queries will then be the basis for the visualisation.

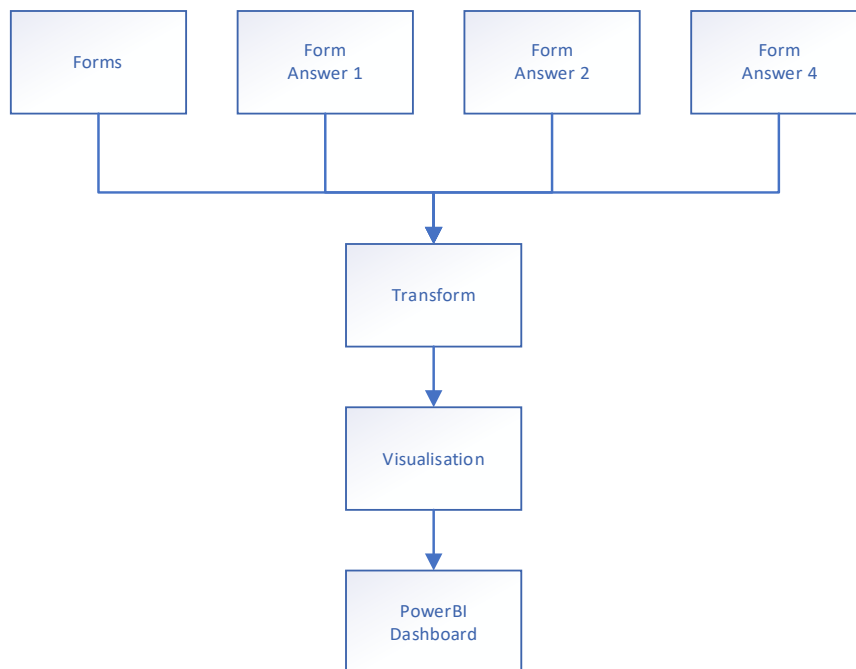


Figure 7: Sample representation of the data model in PowerBI

5.3 Feedback Loop

The senior leadership team and the supervisor were very keen on the KPI Process improvement, its outputs, results of visualisation the data and the automation of the process.

The company was in a rapid growth spurt, as several contracts were won during 2022/2023, and a ramp up of activities were in progress. The direction was to use a feedback loop during the phased development process. Envisaged steps that was taken during the feedback loop:

1. Initial requirement meeting with suggestion by the supervisor of the visualisation of data.
2. Analyse, develop and display data as per the initial requirement.
3. Follow up meeting to determine if the result fit the expected requirements.
4. Publish any changes or improvements of the feedback received.

Step 3 and 4 would be repeated until decision was made to move to next visualisation feature. As the scope of the project and allocated time was controlled by the organisation supervisor's priorities; it's important to note that I had to adhere to my supervisor's direction, which was fine and something to be expected of, when you work in SME environment, where priorities and work allocation change on a day-to-day basis.

6 Technologies

The technologies used in the project were dictated by several limitations set by several factors. For example, the type of API used by Fieldview, scripting language/framework that could very efficiently and quickly generate data to PowerBI and any other technologies linked, researched, or discovered during the project development process. The key drivers for deciding the scripting language were the Fieldview API, SharePoint Online and how quickly and cost effective the solution could be put into productivity. There was no direction from the internal supervisor in relation to which technology was used.

6.1 PowerBI

PowerBI is the tool to display dashboards of KPI views to the stakeholders of the organisation. Developed by Microsoft, PowerBI is an interactive data visualization software product (Microsoft, 2022). It is a collection of software services, apps, and connectors that work together to turn unrelated sources of data into coherent, visually immersive, and interactive insights. Data can be inputted/connected by reading directly from a database, webpage, or structured files such as spreadsheets, CSV, XML, and JSON.

6.2 SOAP API

Fieldview uses SOAP API. SOAP is an acronym for Simple Object Access Protocol (AltexSoft, n.d.), which is a messaging protocol specification for exchanging structured information, i.e. allows for implementation of web services. Normally it uses XML Information Set for its message format, and relies on application layer protocols i.e., Hypertext Transfer Protocol (HTTP). It is worth noting Fieldview uses Hypertext Transfer Protocol Secure (HTTPS) for its API.

SOAP is over two decades old and allows users to pull or push data from a range of operating systems as well as numerous clients to run web services and receive responses over a range of script language and platforms.

6.3 PowerShell

During the research process several scripting languages were considered for the project for example Node JS, PowerShell, or Python.

PowerShell was selected as it was deemed it would cause the least amount of impact on any existing system or servers and could very easily be transferred between machines in the organisations IT

environment. It has the potential of running future API requests inside the SharePoint Online Management Shell.

PowerShell is a command-line shell and scripting language (Sdwheeler, n.d.). It supports variables, functions, branching (if-then-else), loops (while do, for, and foreach) and structured error/exception handling and closures/lambda expression.

6.4 Power Automate

Power Automate is part of Microsoft's Power platform, which is a low-code application environment which allows for data analytics and workflow automation. This is where power automate will be used for the project, automating the flow of getting data from for example the Fieldview API to SharePoint Online.

Power Automate has 2 environments and both will be used in the project.

- The Power Automate Cloud Based service will be used to schedule Power Automate desktop tasks to run on a required basis (powerautomate.microsoft.com, n.d.).
- The Power Automate for desktops application will be installed and used to run PowerShell scripts. The scripting feature allows the user to run blocks of code in a couple of different languages, for example PowerShell, Python, VBScript, and JavaScript (Georgiostrantzias, n.d.).

6.5 DAX

DAX is a formula language used in Power BI for data analysis. DAX is an acronym for Data Analysis Expressions. DAX allows the user to formulate formulas to perform queries and calculations on data in tables stored in Power BI (Minewiskan, n.d.).

Measures are a vital component of Power BI and Dax. A measure is a dynamic calculated formula which results change depending on the content i.e., when the data was last refreshed. Measures are created by using the DAX formula bar in the model designer. (Minewiskan, n.d.).

A formula in a measure can use standard functions such as COUNT or SUM, or you can define your own formula by using the DAX formula bar. It's worth noting that measures can be passed as an argument to other measures (Minewiskan, n.d.).

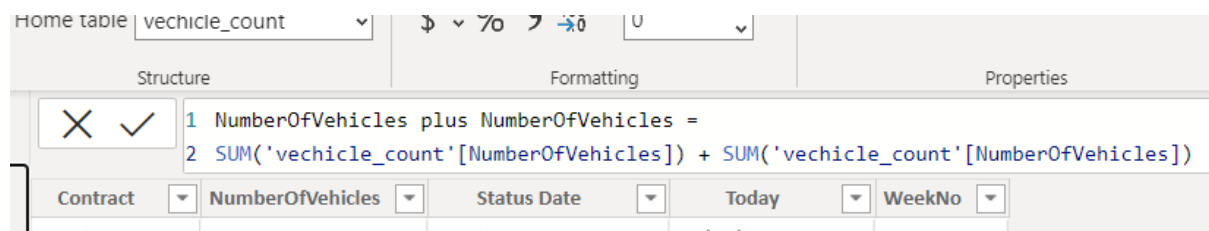


Figure 8: Sample of DAX language in the formula bar of Power BI

7 Project Plan

The milestones to achieve are outlined below.

Milestone	Description	Due Date
Draft Proposal	Initial project proposal and concept	6 th of November
Final Proposal	Articulate project nature and concept	4 th of December 2022
Interim Report	Substantial update progress of the project	12 th of February 2023
Final Project	Final project submission	2 nd of April 2023

Figure 9: Table of milestones

The project intended to span over the below phased development activities with preliminary due dates. It was anticipated that the dates and phases may be subject to change, due to a phase might take less or more time than expected as well as the allocated time and purpose of phase was controlled by the organisation supervisor's priority.

Phase	Preliminary Sprint Due Dates
Research & Training	8 th January 2023
Data Connector setup – Smartsheet	15 th of January 2023
FV API setup and Data Persistence	29 th of January 2023
PowerBI Dashboard – Fieldview Vehicle-check	12 th of February 2023
PowerBI Dashboard – Fieldview Form Count	25 th of February 2023
PowerBI Dashboard – SmartSheet Timesheet	19 th of March 2023
Automation of Data flows and Power BI Refresh	26 th of March 2023
Project Wrap Up	2 nd April 2023

Figure 10: Table of phases

7.1 Project Planner

The Microsoft Planner (www.microsoft.com, n.d.) tool was used for planning and execution of the project. In planner the user can divide development phases also knowns as sprints into buckets. Inside the buckets the user would then add tasks into it.

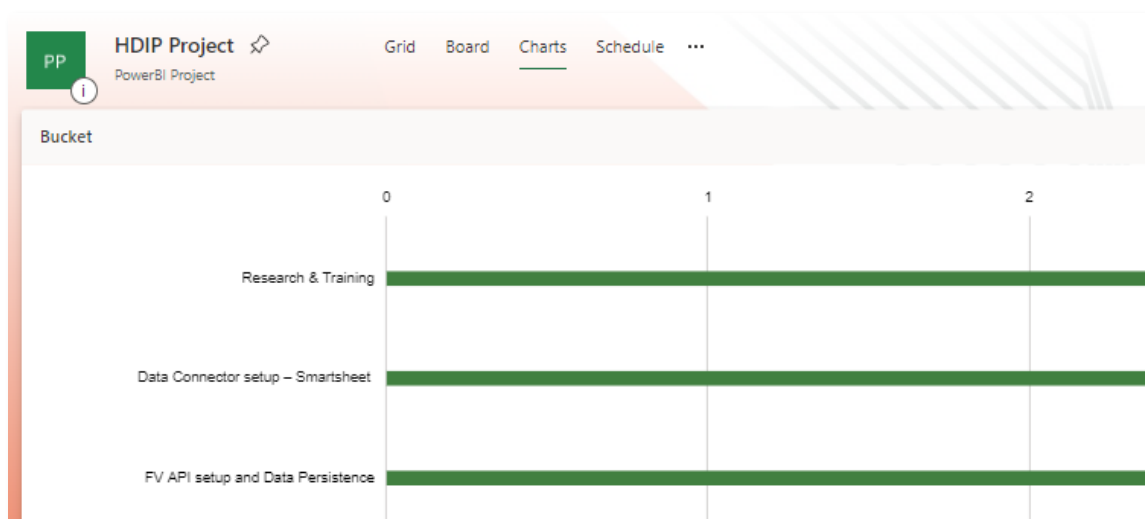


Figure 11: Microsoft Planner – Chart of Buckets

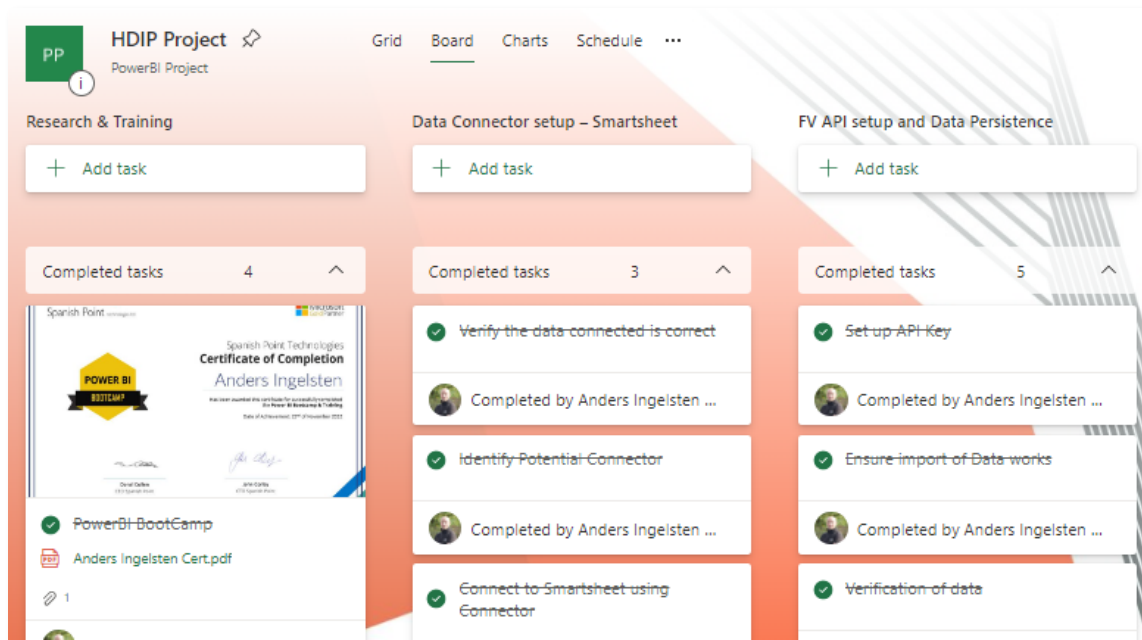


Figure 12: Microsoft Planner - Status of tasks and Buckets

8 Implementation

8.1 Phase 1: Research & Training

The phase of research and training covered three principal areas.

The Fieldview API. The API Documentation of in the Viewpoint Help Section was investigated (Help.viewpoint.com, n.d.), contact was also made to the support desk of Trimble to get direction of how the API was structured. The research revealed the following. Fieldview has 3 Data Centre API regions/URLs these are: UK, North America & Australia New Zealand. The organisations data is stored in the UK region data centre.

Every Region has twelve APIs in two set of six different API (Help.viewpoint.com, n.d.). Grouped by XML or JSON, the APIs are:

- Configurations Services
- Forms Services
- Tasks Services
- Process Services
- Assets Services
- Project Services

Two APIs was explored for the purposes of the project.

- Configurations Services – this API allowed me to identify and call project ids and associated information. For Example, the projectID is required to get Form Information
- Forms Services – this API allowed me to get form information, and individual answers.

Below follows three images of the formsservices API GetQuestionAnswer() command, first the parameters, a view of the SOAP API and then the returned answer information

Parameter	Type	Max Length	Required	Description
apiToken	string	20	Y	Your API security token configured in Field View.
formId	string	20	Y	The unique ID of the form.
questionAlias	string	100	Y	The question alias in the form you wish to retrieve the answer for.

Figure 13: Sample of input Parameters

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```
POST /FieldViewWebServices/WebServices/XML/API_FormsServices.asmx HTTP/1.1
Host: www.priority1.uk.net
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "https://localhost.priority1.uk.net/Priority1WebServices/XML/GetQuestionAnswer"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetQuestionAnswer xmlns="https://localhost.priority1.uk.net/Priority1WebServices/XML">
      <apiToken>string</apiToken>
      <formId>string</formId>
      <questionAlias>string</questionAlias>
    </GetQuestionAnswer>
  </soap:Body>
</soap:Envelope>
```

Figure 14: SOAP request and response

```
<FormAnswerResponse>
  <Status>
    SUCCESS
    <Message>Success.</Message>
  </Status>
  <FormAnswerInformation>
    <FormAnswerID></FormAnswerID>
    <FormTemplateID></FormTemplateID>
    <QuestionType></QuestionType>
    <DataType></DataType>
    <Question></Question>
    <Answer></Answer>
    <AnsweredBy></AnsweredBy>
    <AnsweredDateTime></AnsweredDateTime>
    <HasActions></HasActions>
    <HasImages></HasImages>
    <HasComments></HasComments>
    <HasDocuments></HasDocuments>
  </FormAnswerInformation>
</FormAnswerResponse>
```

Figure 15: Returned information.

Time was spent understanding which API's information had to be pulled, stored, and passed on to other API's. One important feature noted in the research period was the API Call quota. The call quota is the number of points an API token can spend within a minute. An API token in Fieldview has a maximum quota of 120 points allocated. This means you are limited to the number of calls that can be made per minute. For example, the `GetProjectFormsList()` has a point quota of 10, so this would mean that associated token to the API can only process 12 calls in one minute.

API Token	Maximum Quota (/min)	Remaining Quota	Last Reset
AAAA-BBBB-CCCC-DDDD	120	32	2010-09-21 15:49:43.833

Figure 16: Call quota per API token

The research also showed that the Fieldview API would not easily return any data using the built-in data connectors in PowerBI and that the route of developing scripting of an API had to be taken. This is in line with the purpose of the project i.e., to showcase skills learned from the HDIP course.

It's worth noting that there exist third party apps like ZappySys ODBC Power Pack, who integrate SOAP API with PowerBI. However, this not a free software and annual subscription is approx. \$650 per desktop install (ZappySys, 2018) and therefore this was not progressed withing the project.

PowerBI. To get an understanding of what capabilities PowerBI has and its functionality I signed up for Spanish Point Technologies "Dashboard in a day" course. Spanish Point Technologies is a software company and a Gold Certified Microsoft partner. Spanish Point specialises in Azure, Microsoft 365, SharePoint, Dynamics 365, PowerApps & Power Automate, Power BI and SQL Server solutions (Technologies, n.d.). "Dashboard in a day" is a free full day workshop covering the capabilities of PowerBI through an instructor led online course. The goal of the course is to better understand how to:

- Connect to, import, and transform data from a variety of sources.
- Define business rules and KPIs.
- Explore data with powerful visualization tools.
- Build stunning reports.
- Share dashboards with their team and business partners and publish them to the web.

The power BI Bootcamp and training was completed on the 23rd of November 2022.



Figure 17: Certificate of Completion

In addition to the training, I also conducted various research online to find out how to do certain assorted items, anything from PowerShell Scripting to SharePoint Online solutions and Power BI queries.

8.2 Phase 2: Smartsheet Data Connector setup

Next item to resolve was connecting Smartsheet's to PowerBI. Smartsheet provides a Data Connector, which is part of PowerBI built in connectors. Following steps were completed in this Phase.

- 1. Set up of user credentials in SmartSheet.
- 2. Ensure user had access to relevant tables in SmartSheet.
- 3. Connect Smartsheet to PowerBI using the built-in get Data feature in Power BI
- 4. Authenticate connection in PowerBI with Smartsheet user credentials.
- 5. Load and transform the relevant table from Smartsheet into PowerBI

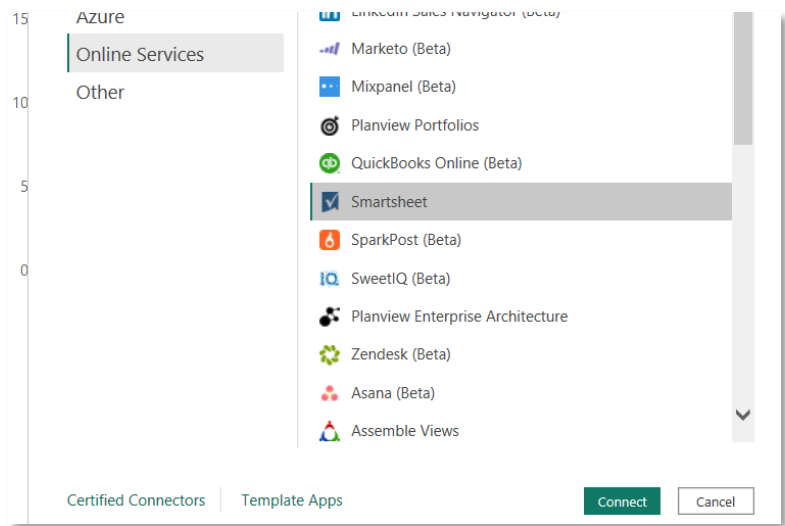


Figure 18: Get Data view in PowerBI of Online Services

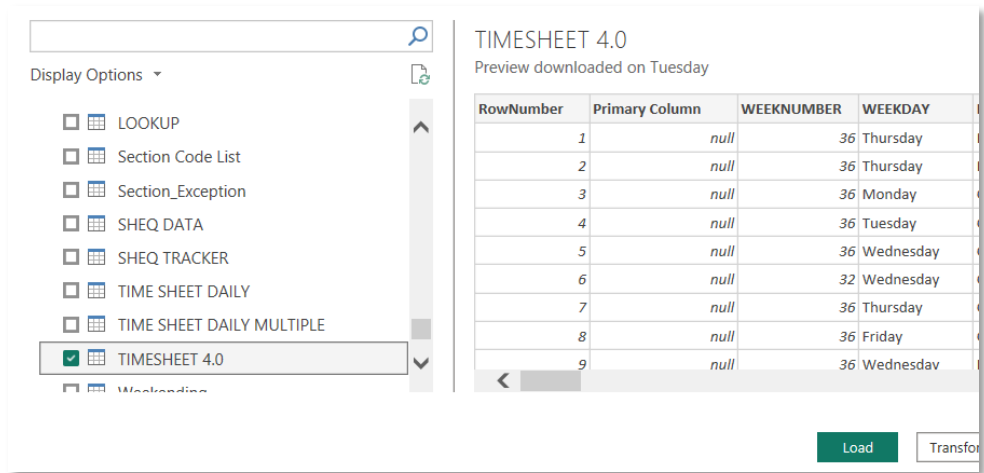


Figure 19: Preview of Connected Table ready to be loaded into PowerBI.

8.3 Phase 3: FV API setup and Data Persistence

The goal of this phase was to be able to reliably pull information from the Fieldview API to the organisation's hosted Data repository i.e., SharePoint Online.

To be able to even access the API, the first thing I had to do to generate API keys inside Fieldview. The API key can be set on Group Organisation and Company level. A Company is a child of an Organisation. All keys were set up on Group Level.

API Token Name	Business Unit	Login Name	E-mail	API Token
All_Form_IDs	M...	ar...	ai...	71... F2
All_Forms	M...	ar...	ai...	E7... FB3
All_Forms_UpTo2Months	M...	ar...	ai...	F1... 95814

Figure 20: View of Created API Tokens in Fieldview

The SOAP API relies on discreet calls for every interaction with the host. These calls also need to be structured inside a SOAP API envelope. This meant I needed to be able to use a scripting language compatible with SharePoint online and which has the capabilities to execute general programming tasks, for example for each loop and if statements.

By utilising PowerShell and its module New-WebServiceProxy I managed to achieve this. The PowerShell New-WebServiceProxy will download the API's WSDL and use it to generate types for the proxy's interface, data contracts and headers. PowerShell also allows you to import variables from a file and export variables to csv files. Tasks can then be scheduled to run at certain intervals, but this will be resolved in a later phase.

```
<#
Fieldview API 1 - Lst of all forms up to 3months old.
This API pulls all project id's and pull all forms up to 3 months old by modified data
When pull is completed the data is exported to Sharepoint Online
#>

$ApiToken = Get-Content C:\Users\aingelsten\scripts\api_id.txt

Write-Output "Connecting to API"

$FVApiConfig = New-WebServiceProxy -Uri "https://www.priority1.uk.net/FieldViewWebServices/WebServices/XML/API_ConfigurationServices.asmx?WSDL"

$FVApiForms = New-WebServiceProxy -Uri "https://www.priority1.uk.net/FieldViewWebServices/WebServices/XML/API_FormsServices.asmx?WSDL"

Write-Output "Getting Project ID's"

$FVApiConfig.GetProjects($ApiToken, $null, $null, 1, 0, 100).ProjectInformation.chiltnodes.id

$id = $FVApiConfig.GetProjects($ApiToken, $null, $null, 1, 0, 100).ProjectInformation.chiltnodes.id
```

Figure 21: Code Snippet of the New-WebServiceProxy used to pull data from Fieldview.

The New-WebServiceProxy sets up a proxy object (Sdwheeler, n.d.), which allows for interaction with the Fieldview SOAP API and by utilising PowerShell capabilities of running foreach loops, if statements, I could connect and store all the project ids, and then loop them back into the next call which would in this phase retrieve form information. Every API Token has a call quota per minute (help.viewpoint.com, n.d.). Due to this fact I generated several API keys so I could generate multiple calls. I also added delays in the loop, to avoid exceeding the call quota.

```
foreach ($projectid in $projectids)
{
    Write-Output $projectid

    $FVApiForms.GetProjectFormsList($apiToken, $projectid, $null, 0, $datefrom, $dateto, $null, $null, $null, $null, $null, $null).ProjectFormsListInformation.chiltnodes
    $formsList = $FVApiForms.GetProjectFormsList($apiToken, $projectid, $null, 0, $datefrom, $dateto, $null, $null, $null, $null, $null, $null).ProjectFormsListInformation.chiltnodes

    if ($formsList -eq $null)
    {
        Write-Output "*****NOTHING FOUND*****"
    }
    else
    {
        Write-Output "Adding Projectid"

        $formsList | Add-Member -MemberType NoteProperty -Name "ProjectId" -Value $projectid

        Write-Output "Writing to file"

        $formsList | Export-Csv -Path c:\Users\aingelsten\scripts\formslist.csv -append -NoTypeInformation

        Write-Output "Data written to file"
    }

    Start-Sleep -Seconds 10
}
```

Figure 22: Code snippet of foreach loop with an if else statement

```
FormID : F268753.52
FormTemplateLinkID : 14680656
Deleted : false
FormType : Operations
FormName : MP-194 Site Diary
FormTitle : 2022-11-25
CreatedDate : 2022-11-25T12:29:28
OwnedBy :
OwnedByOrganisation : Ltd
IssuedToOrganisation :
Status : Completed and signed off
StatusColour : #009900
StatusDate : 2022-11-25T12:48:12
Location : MP-194 Cork Airport Substation
OpenTasks : 0
ClosedTasks : 0
FormExpiryDate : FormExpiryDate
OverDue : false
Complete : true
Closed : true
ParentFormID :
LastModified : 2022-11-25T13:07:17
LastModifiedOnServer : 2022-11-25T13:07:17
ClosedBy :
FormTemplateID : 15738621
ParentProcessTaskID :

Adding Projectid
Writing to file
Data written to file
23754
*****NOTHING FOUND*****
```

Figure 23: Screenshot of Sample Data from the GetProjectFormsList() call.

8.3.1 Data persistence

The main factor that decided which type of method for data persistence was going to be used in the project was availability and cost. The organisation has access to hosted servers, but these are operational only during office hours and increasing the availability would lead to increased cost.

For the data to be persistent it must write to non-volatile storage. Data persistence was achieved in the project by saving the data to a file and then exported to Sharepoint Online (Rajack, 2018). By saving a file with the same name to the same location, SharePoint online just saves an updated version of the file. A CSV file stored in SharePoint Online allows for connection to PowerBI by its built in Web connector.

```
Write-Output "Exporting to SharePoint"

#Configuration of Sharepoint Variables
$SiteURL = "https://typetecmg.sharepoint.com/sites/[redacted]"
$SourceFilePath = "c:\Users\aingelsten\scripts\formslist.csv"
$DestinationPath = "Kpi_Data" #Site Relative Path of the Library

#Connect to PnP Online using Weblogin
Connect-PnPOnline -Url $SiteURL -UseWebLogin

#Powershell pnp to upload file to sharepoint online
Add-PnPFile -Path $SourceFilePath -Folder $DestinationPath

Write-Output "Process Completed"
```

Figure 24: Sample Code of writing a file to Sharepoint Online

```
InformationRightsManagementSettings : Microsoft.SharePoint.Client.InformationRightsManagementFileSettings
ImEnabled                          : False
Length                             : 0
Level                               : Published
LinkingUri                          : https://typetecmg.sharepoint.com/sites/I       e/kpi_Data/formslist.csv?d=wfa70      d2a1
LinkingUri                          : https://typetecmg.sharepoint.com/sites/I       e/kpi_Data/formslist.csv?d=wfa70      d2a1
ListItemAllFields                   : Microsoft.SharePoint.Client.ListItem
LockedByUser                        : Microsoft.SharePoint.Client.User
MajorVersion                        : 1
MinorVersion                        : 0
ModifiedBy                          : Microsoft.SharePoint.Client.User
Name                                : formslist.csv
PageRenderType                     :
Properties                          : Microsoft.SharePoint.Client.PropertyValues
ServerRedirectedUrl                 :
ServerRelativePath                  :
ServerRelativeUrl                   : /sites/1 /Kpi_Data/formslist.csv
SiteId                              :
TimeCreated                         : 22/01/2023 10:20:18
TimeLastModified                   : 22/01/2023 10:32:45
Title                               :
UIVersion                           : $12
UIVersionLabel                     : 1.0
UniqueId                           : fa70bd20-9476-4f24-9567-03ae372bd2a1
VersionEvents                       :
```

Figure 25: Screenshot of successfully writing the data to Sharepoint Online in PowerShell

A	B	C	D	E	F	G	H
ProjectId	FormID	FormTemplateLinkID	Deleted	FormType	FormName	FormTitle	CreatedDate
24250	F1.1801134	12252321	FALSE	SHE Forms	S53Site Safety Audit Form		2023-01-06T12:00:1
24250	F1.1809082	16080373	FALSE	Electrical	E20 HV-LV Ducting Cleaning / Proving Report	Yes	2023-01-10T15:00:4
27385	F1.1823210	16854981	FALSE	Administration	207-Coins TimeSheet	001OSM - Prelims	2023-01-16T12:37:5
24250	F1.1826163	16055566	FALSE	Operations	MP-192 Kilroot Site Diary	16/01/2023	2023-01-17T10:18:4
24250	F1.1826236	16055566	FALSE	Operations	MP-192 Kilroot Site Diary	17/01/2023	2023-01-17T10:31:1
24250	F1.1826305	12252321	FALSE	SHE Forms	S53Site Safety Audit Form		2023-01-17T10:45:5
21549	F1.1842734	16750904	FALSE	Assets	Asset Transfer		2023-01-23T14:48:0
24250	F1.1852233	16080373	FALSE	Electrical	E20 HV-LV Ducting Cleaning / Proving Report	Yes	2023-01-26T10:30:1
24250	F1.1852234	16080373	FALSE	Electrical	E20 HV-LV Ducting Cleaning / Proving Report	Yes	2023-01-26T10:30:2
24250	F1.1853322	16080307	FALSE	Electrical	E21 HV Cable Installation Record Sheet v2		2023-01-26T13:53:1
24250	F1.1853332	16079838	FALSE	Electrical	E23 ESB HV Cable Insulation Resistance Test Certificate		2023-01-26T13:54:5
24250	F1.1854386	15866705	FALSE	Operations	Jointing Quality Check		2023-01-26T17:04:4
24250	F1.1871316	16080307	FALSE	Electrical	E21 HV Cable Installation Record Sheet v2		2023-02-02T10:04:5
27385	F1.1885615	16854981	FALSE	Administration	207-Coins TimeSheet	001OSM - Prelims	2023-02-08T08:22:5
26811	F1.1886631	16088592	FALSE	ITP	ITP 010 INSTALLATION OF MV CABLE		2023-02-08T11:42:5
21549	F1.1886871	16088592	FALSE	ITP	ITP 010 INSTALLATION OF MV CABLE		2023-02-08T12:32:1
20698	F1.1887418	12031524	FALSE	Administration	Company Vehicle Check List	Ford Transit Custc	2023-02-08T14:23:3
21206	F1.1887436	12465271	FALSE	Administration	Coins TimeSheet		2023-02-08T14:30:5

Figure 28: Form data as exported from FV.

I analysed the imported data, and to create the desired outcome I created a Measure in DAX scripting that provides a count per form name, weeknumber, location and status like below. In this case the form is called "Company Vehicle Check List". It is worth noting I had to use the IF(ISBLANK()) feature to avoid the card displaying the value "BLANK" (Rad, 2020).

```

1 Actual MP = IF(ISBLANK(
2 COUNTX(
3     FILTER(
4         formslist,
5         formslist[FormName] = "Company Vehicle Check List"
6         && formslist[WeekNumber] = vehicle_count[Week] && LEFT(formslist[Location], 3) = "MP-" && formslist[Status] = "Completed and signed off"
7     ),
8     formslist[FormName]
9 ), 0)

```

Figure 29: DAX code of Count

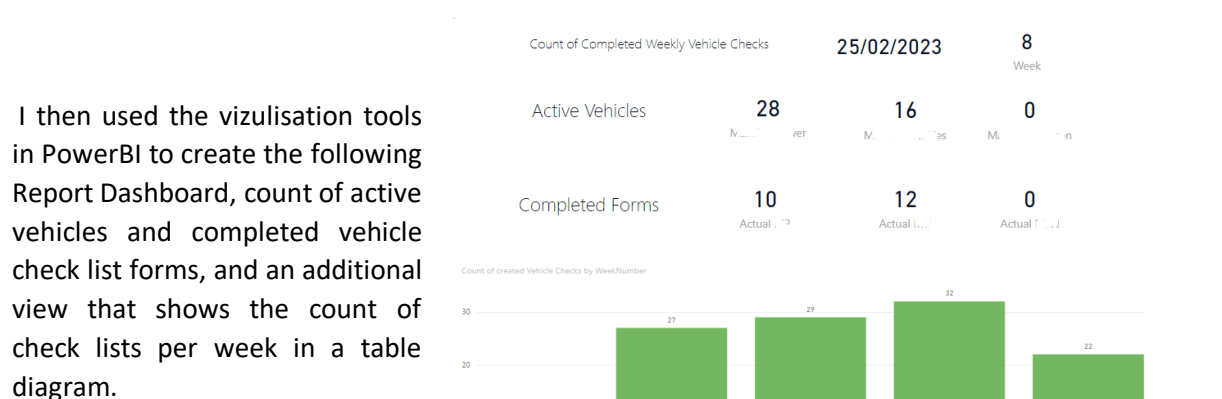


Figure 30: View of Counts

The next task for this phase was to make it visible for the end user through the Company's SharePoint Intranet. I had 2 options.

- Publish an Embed Link in SharePoint – this requires every individual user to have a PowerBI license.
- Or to generate Embedded Html code, so it can be published to multiple users.

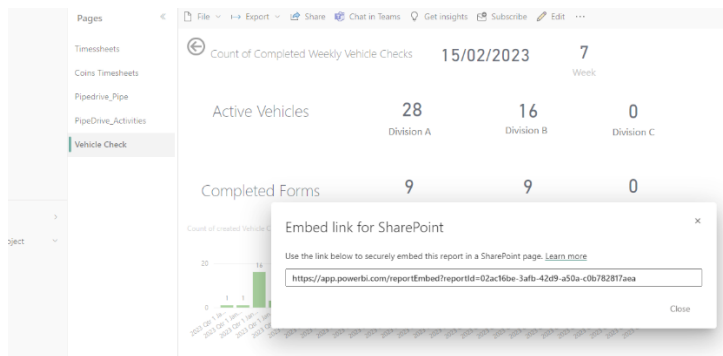


Figure 31: View of Embed Link

I choose the latter, which required me to purchase PowerBI Pro license and I had to make sure the default admin settings allowed the user to create embedded code creation.

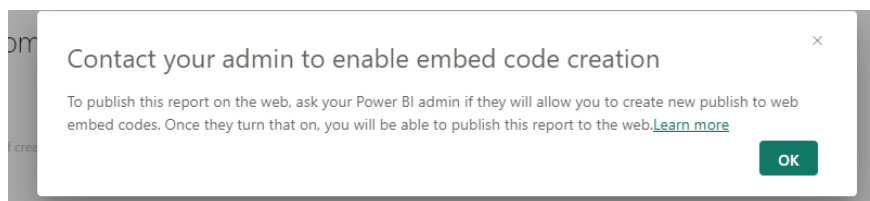


Figure 32: View of admin message

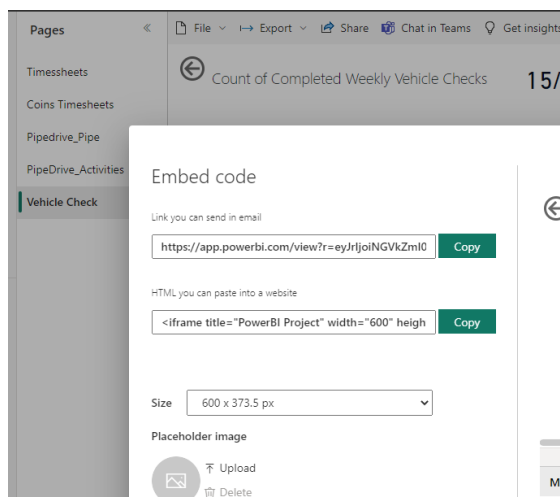


Figure 33: View of embed code view.

The embedded code was then published in SharePoint using the Embed code function.

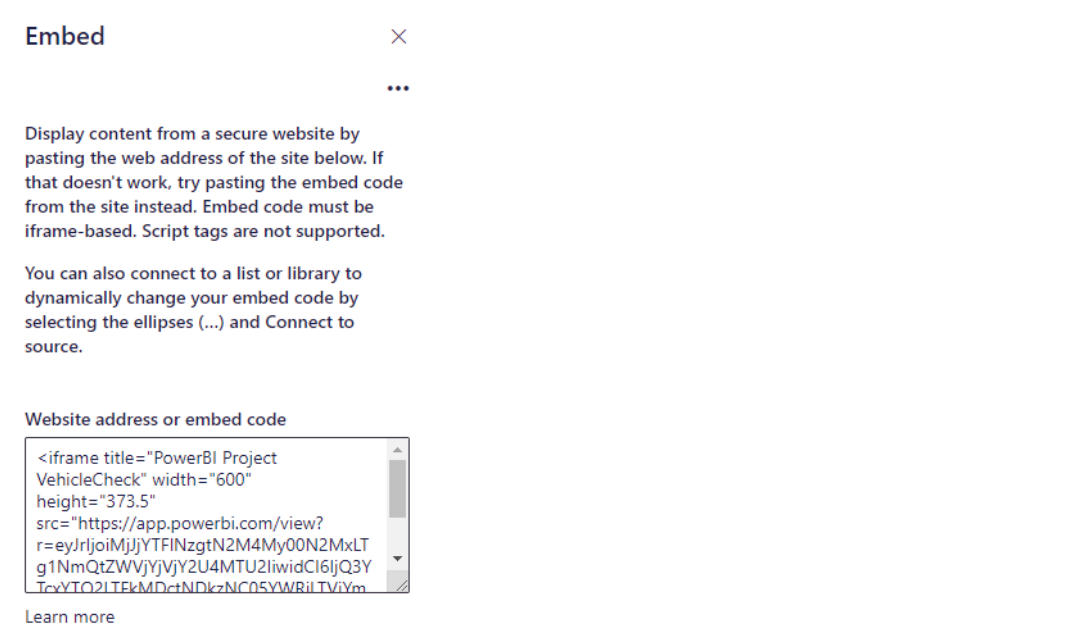


Figure 34: View of Sharepoint Online Embed function.

End result of the published view embedded in the Plant Transport page.

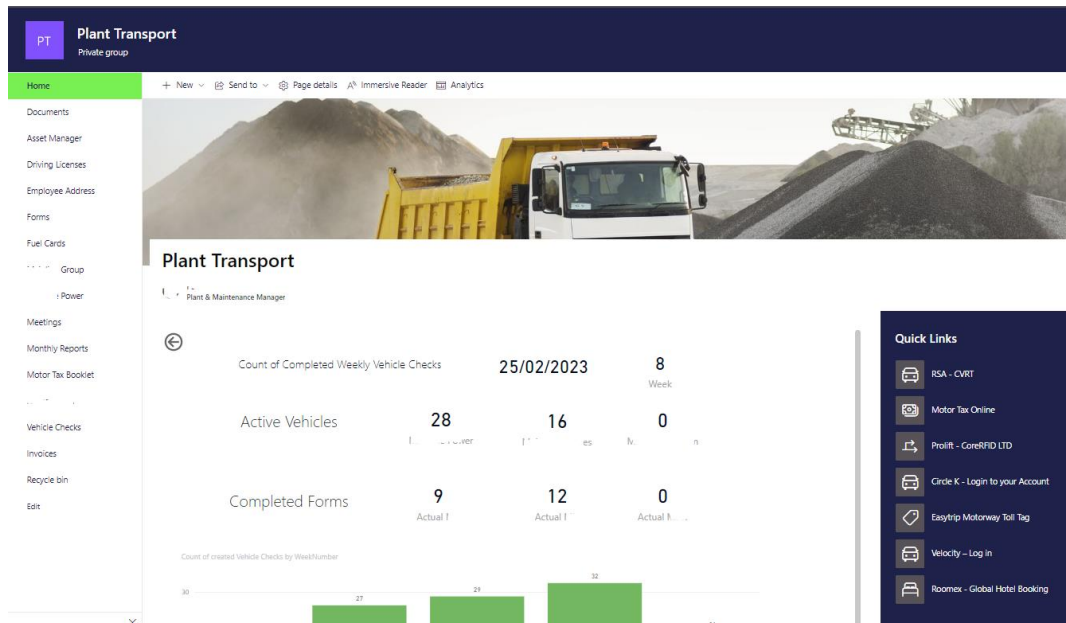


Figure 35: View of Sharepoint Intranet

8.5 Phase 5: PowerBI Dashboard – Pipedrive

At the project planning phase, I planned after the vehicle check phase to develop a dashboard that would display form counts per project. An active project has several forms that are required to be filled in on a regular basis, for example, daily site diary's, weekly project reports, monthly safety audits etc.

However, the commercial team requested that I would instead create a dashboard which would display the current business development pipeline. Displaying a summarised contract value per quarter start period.

Pipedrive is a CRM (Customer Relation Management) system that allows the user to track business development, e.g., potential leads, their value and tracking interactions like phone calls and meetings (Pipedrive Inc / Pipedrive OÜ, 2018).

I therefore investigated the Pipedrive application and its API capabilities. I found the Pipedrive API documentation (developers.pipedrive.com, n.d.) and a GIT repository in relation to using PowerShell (Seidlm, 2021), connecting, and retrieving data from the Pipedrive API. Effectively using PowerShell's Invoke-RestMethod which sends HTTP and HTTPS requests to Representational State Transfer (REST) web services which then returns data in a rich and structured format (sdwheeler, n.d.).

Together with the Business Development manager we added a custom contract start date and data was entered so the report could be generated in Power BI.

```
# Get All Deals
$num = 0
for ($num;; $num+=99)
{
    $Url_Deals=$Pipedrive_BaseURL+"deals?start=$num&api_token=$PipeDriveAPI"

    $Result_Deals=Invoke-RestMethod -uri $URL_Deals -Method GET

    $Result_Deals.data
    Write-Output "Page number" $num

    if ($null -ne $Result_Deals.data)
    {
        $Result_Deals.data | Export-Csv -Path C:\Users\angelsten\scripts\Deals_pipe.csv -NoTypeInformation -Append
        Write-Output "****EXPORT****"
        Start-Sleep -Seconds 0.5
    }
    else {

        Write-Output "****LOOP ENDS****"
        Break
    }
}
```

Figure 36: Code snippet of Pipedrive script

After that I created a PowerShell script that pulled all deals, activities, and persons. Due to the data being paginated I had to do a loop to retrieve all data.

The data was then exported to Sharepoint online, imported to PowerBI and relationship setup between the 3 tables.

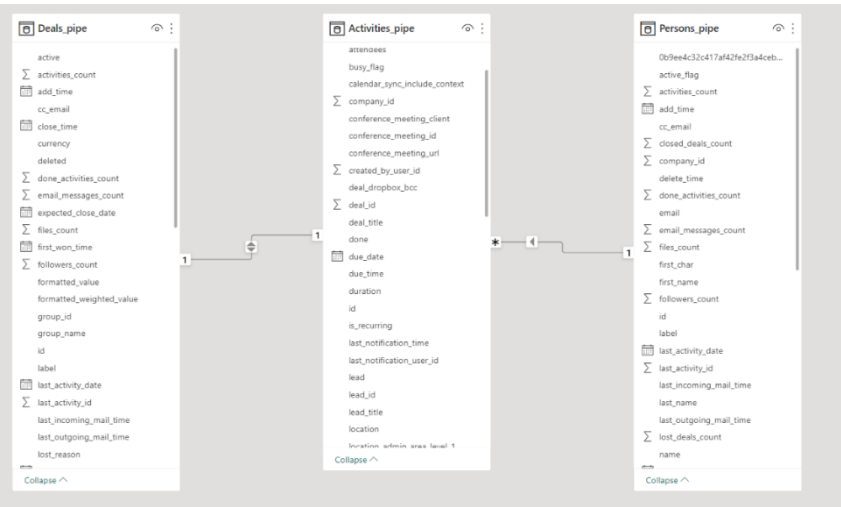


Figure 37: View of table relationships

From this position I generated a report in Power BI that displays the contract value of open tenders with expected quarterly start dates as per below. Note data has been redacted due to they being commercially sensitive.

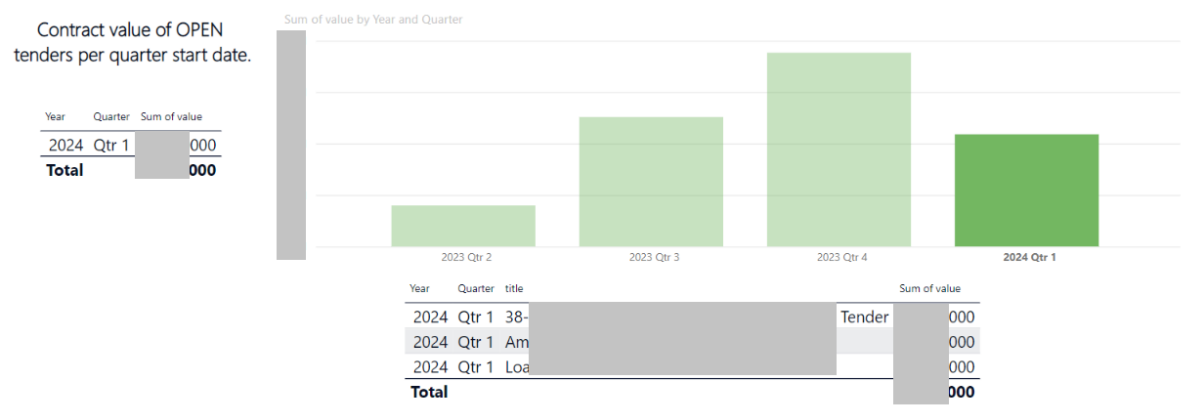


Figure 38: View of Values per Quarter

As a bonus I also generated a report of the logged activities conducted by the business development manager since the inception of Pipedrive

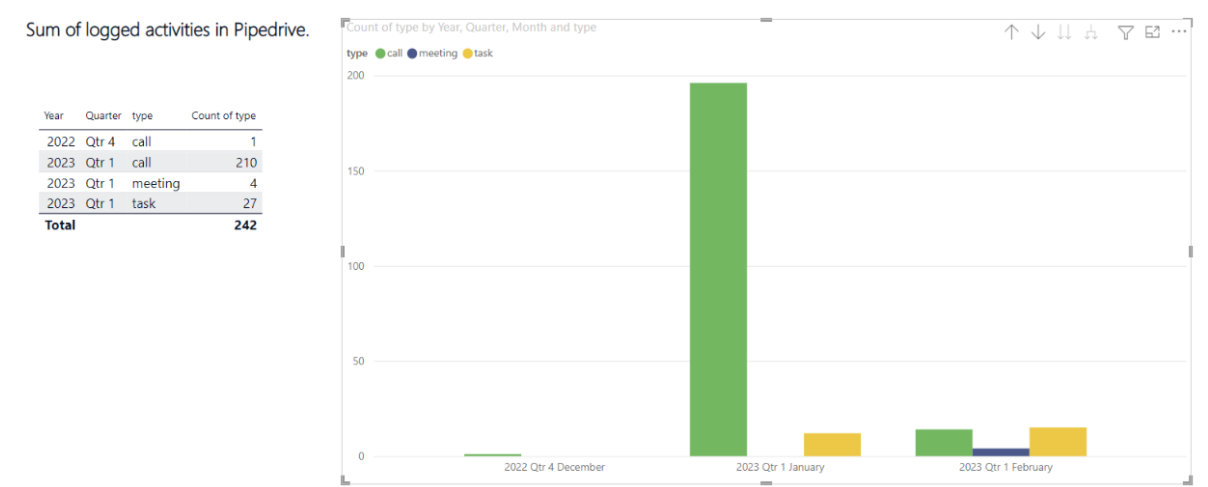


Figure 39: Report view of logged activities

The data for the Pipedrive views must at a minimum be updated monthly as per the KPI dates.

8.6 Phase 6: PowerBI Dashboard – SmartSheet Timesheet

Upon completion of the Pipedrive dashboard, my company supervisor asked me to prioritise a dashboard that displayed a summarised view of hours allocated per contract by user. Utilising the Smartsheet connector as rolled out during Phase 2, I imported the Timesheet table for office staff.

IBER	WEEKDAY	EMPLOYEE	EMPLOYED ON PAYROLL	EMPLOYEE NUMBER	COINS COMPANY NUMBER	DATE	CONTRACT	CONTRACT HOURS	EM
2	Tuesday	M	in		4	5	Tuesday 3 January 2023	M	9 L22
2	Tuesday	M	in		1	5	Tuesday 3 January 2023	M	10 L22
2	Wednesday	M	in		1	5	Wednesday 4 January 2023	M	9 L22
2	Tuesday	F	ana		8	5	Tuesday 3 January 2023	M	2.5 L36
2	Thursday	C	e		6	5	Thursday 5 January 2023	M	8.5 L36
2	Thursday	M	in		1	5	Thursday 5 January 2023	M	9 L22

Figure 40: View of Smartsheet timesheet table

After that I generated a Donut chart of the summary of hours per contract. I also generated a corresponding table view and a card of the total sum of contracts. View have been blurred to hide identifiable personal or commercial data.



Figure 41: View published on Company Intranet

As per requirements from the company supervisor, the visual can be drilled down to contract, section code and employee level.

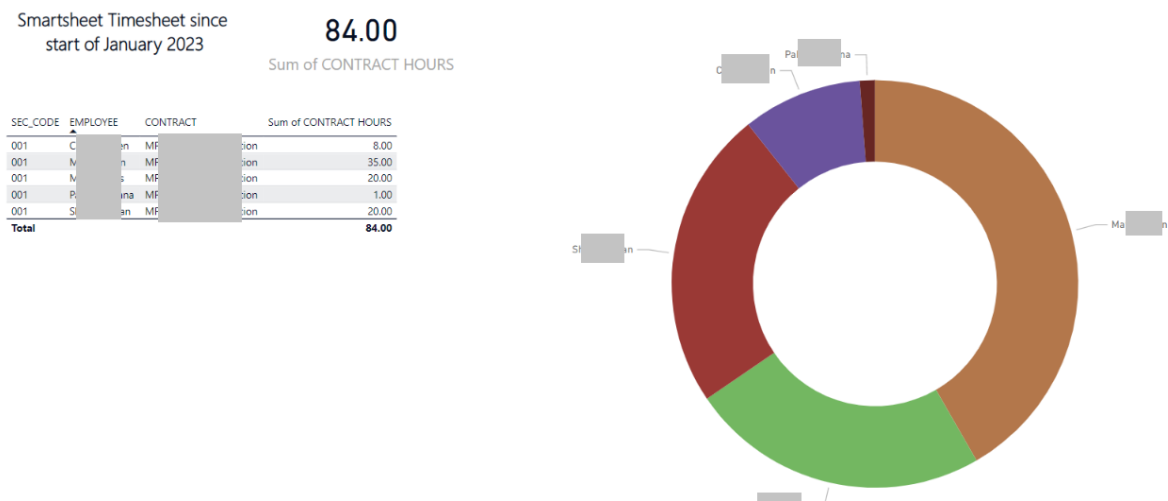


Figure 42: Drill down view Timesheet

8.7 Phase 7: PowerBI Dashboard – Fieldview Timesheet

The next priority set by the company supervisor was to create a dashboard which displayed a summarized view of hours generated by site staff using the Fieldview application.

For this I had to use a different Fieldview API called GetQuestionAnswer. This API takes in the APIToken, FormID and the QuestionAlias and provides the answer and meta data of the answer.

The minimum required data was employee name, contract, date, and hours, so I had to create four get requests and due to the API has an API quota of 10 I quickly realised to run an effective query I had to generate an API Token per API call.

```
$apiToken1 = Get-Content C:\Users\aingelsten\scripts\api_answer1.txt
$apiToken2 = Get-Content C:\Users\aingelsten\scripts\api_answer2.txt
$apiToken3 = Get-Content C:\Users\aingelsten\scripts\api_answer3.txt
$apiToken4 = Get-Content C:\Users\aingelsten\scripts\api_answer4.txt

$FVForms = New-WebServiceProxy -Uri "https://www.priority1.uk.net/FieldViewWebServices/WebServices/XML/API_FormsServices.aspx?WSDL"

$answer = "FVTimeSheet_Completed_By.csv"
$answer1 = "FVTimeSheet_Contract.csv"
$answer2 = "FVTimeSheet_Date.csv"
$answer3 = "FVTimeSheet_Hours.csv"
$questionAlias = "TS_Completed_By"
$questionAlias1 = "TS_Contract"
$questionAlias2 = "TS_Date_Timesheet"
$questionAlias3 = "TS_Hour_Calculation"
```

Figure 43: Code Snippet of 4 answer queries


```

$FormsQuestionAnswer = $FVForms.GetQuestionAnswer($apiToken1, $FormID, $questionAlias).FormAnswerInformation.childnodes
$FormsQuestionAnswer | Add-Member -MemberType NoteProperty -Name "Form_Id" -Value $FormID
$FormsQuestionAnswer | Export-Csv -Path C:\Users\aingelsten\scripts\$answer -Append -NoTypeInformation
Write-Output $FormsQuestionAnswer

```

Figure 44: Code snippet of API call

From here I ensured the script generated four CSV files, one per answer and the script exported the files into SharePoint Online.

```

#Configuration of Sharepoint Variables
$SiteURL = "https://[redacted].sharepoint.com/sites/[redacted]"
$SourceFilePath = "c:\Users\aingelsten\scripts\$answer"
$SourceFilePath1 = "c:\Users\aingelsten\scripts\$answer1"
$SourceFilePath2 = "c:\Users\aingelsten\scripts\$answer2"
$SourceFilePath3 = "c:\Users\aingelsten\scripts\$answer3"
$DestinationPath = "Kpi_Data" #Site Relative Path of the Library
$ClientId = "[redacted]"
$ClientSecret = "[redacted]"

#Connect to SharePoint Online with ClientId and ClientSecret
Connect-PnPOnline -Url $SiteURL -ClientId $ClientId -ClientSecret $ClientSecret

#Powershell pnp to upload file to sharepoint online
Add-PnPFile -Path $SourceFilePath -Folder $DestinationPath
Add-PnPFile -Path $SourceFilePath1 -Folder $DestinationPath
Add-PnPFile -Path $SourceFilePath2 -Folder $DestinationPath
Add-PnPFile -Path $SourceFilePath3 -Folder $DestinationPath

```

Figure 45: Code snippet of export to SharePoint

After the four answers from Fieldview had been imported into PowerBI, all the four tables were joined by their common identifier: the FormID.

After analysing the data and outputs I noticed an issue in relation to amount of generated answer data for the question date. I had too many answers relating to date. The reason for this was that the PowerShell script brought back every answer from every form that had a question alias with the word "date" in it. And as there were currently over 120 active forms in the Fieldview system I was getting too many answers back. After realising this, I renamed the Timesheet alias in Fieldview to "TS_Date_Timesheet". As a precaution I also updated the aliases for the other three questions. This resolved the issue.

To be able to generate the required visualisations, I proceeded to use the PowerBI modelling tool where I created a User/Contract table, a Date/Hours table and from these tables I generated a combined Timesheet table that allowed me to generate the visualisation in Power Bi.

Form_Id	FV_User	FV_Contract	FV_Date	FV_Hours
F207746.157			Friday 29 April 2022	8.5
F207746.159			Thursday 5 May 2022	8.5
F207746.160			Friday 6 May 2022	8.5
F207746.162			Monday 9 May 2022	8.5
F207746.163			Tuesday 10 May 2022	8.5
F207746.164			Wednesday 11 May 2022	8.5
F207746.165			Thursday 12 May 2022	8.5

Figure 46: Query view in PowerBI

See view below of all the Fieldview tables and their relationships in PowerBI.

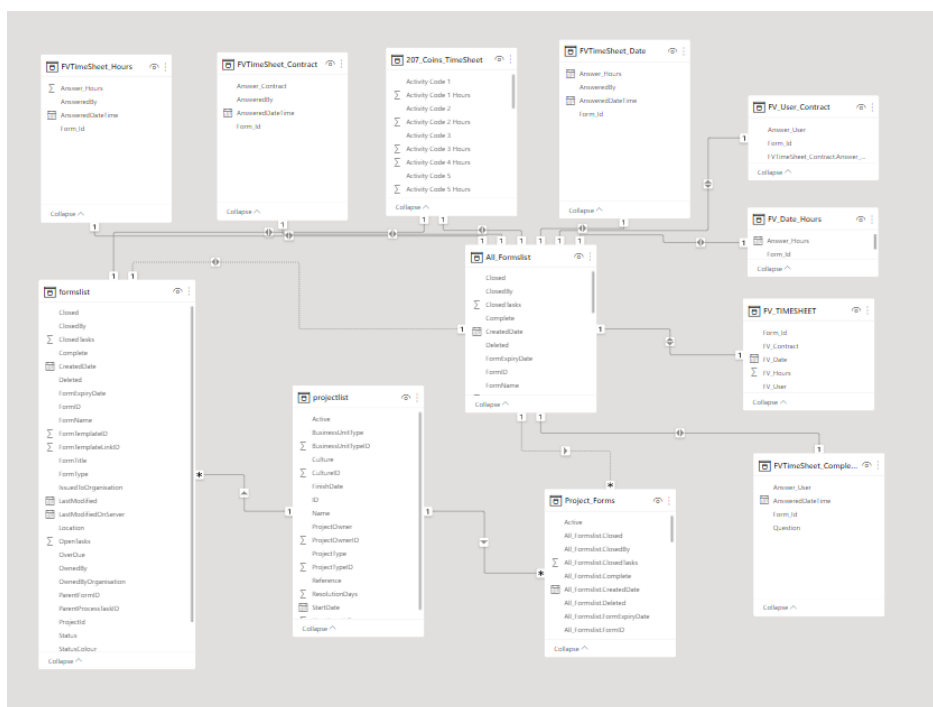


Figure 47: Table relationships in PowerBI

The visualisation requested was the following, a card with total summary of all hours, a donut chart displaying the breakdown of summarised hours per contact and a stacked column chart per employee. The views allowed for drill down per person.

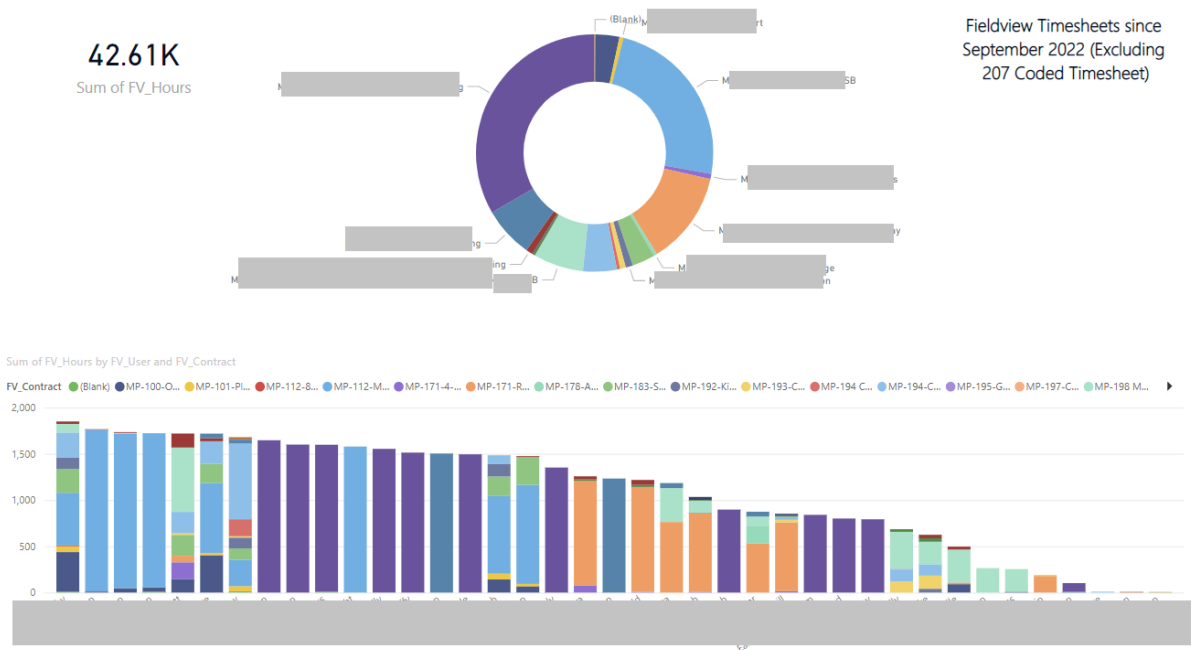


Figure 48: Summary visualisation of the hours per contract and user

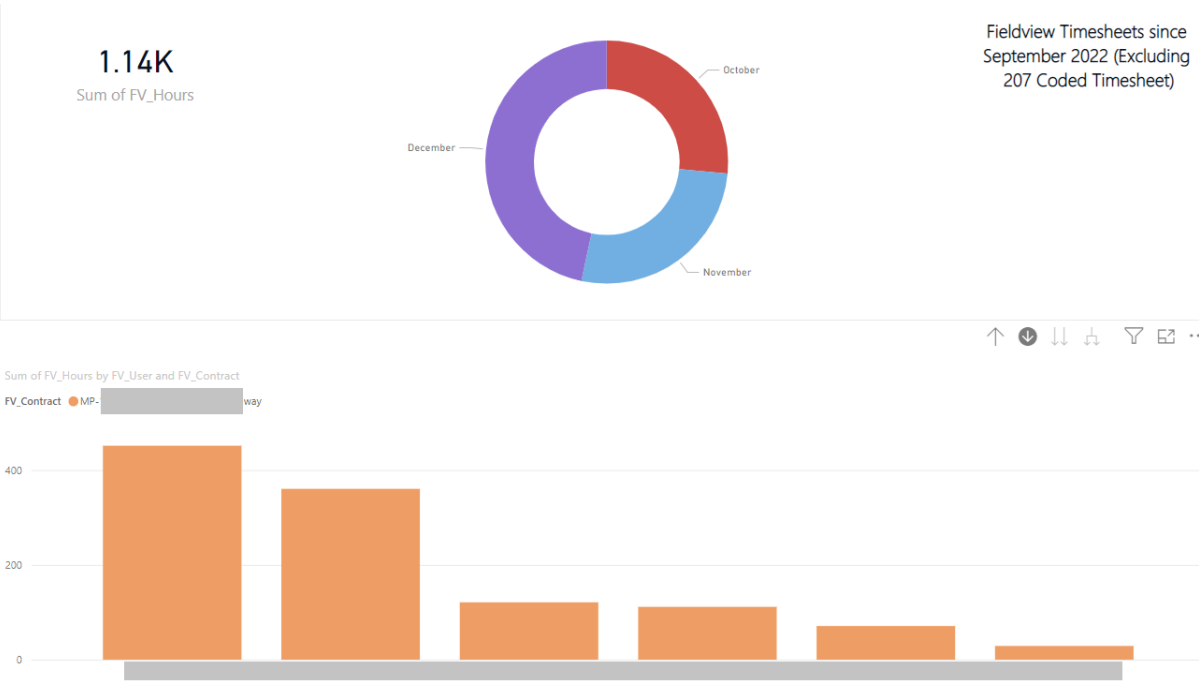


Figure 49: Drill down view per contract, month, and person.

8.8 Phase 8: PowerBI Dashboard – Asset Manager

This phase was also an addition, as the Company's project supervisor asked me to investigate if I could connect Asset Manager and display's KPI report visualisations for the company's plant manager.

Asset Manager is a 3rd party software developed by Kaizen Software Solutions and is utilised by the company to track all fixed and plant assets (Www.kzsoftware.com, n.d.). The system holds asset details, value but also service and certification records. The plant manager needed to be able to access upcoming service and certifications renewal dates more easily as part of his KPI requirements.

The Asset Manager backend database is of type Firebird. It is an open-source SQL relational database management system that supports Linux, Microsoft Windows, macOS and other Unix platforms Home. (n.d.).

PowerBI has no built in connector to Asset Manager, but after a bit of research, a software company called Devart was found. Devart provides an ODBC driver that allows Power BI to connect to Firebird via ODBC (Docs.devart.com, n.d.).

I then progressed and installed the driver, and configured the connection, and used the Power BI ODBC connection to bring in the information, see screenshot samples below.

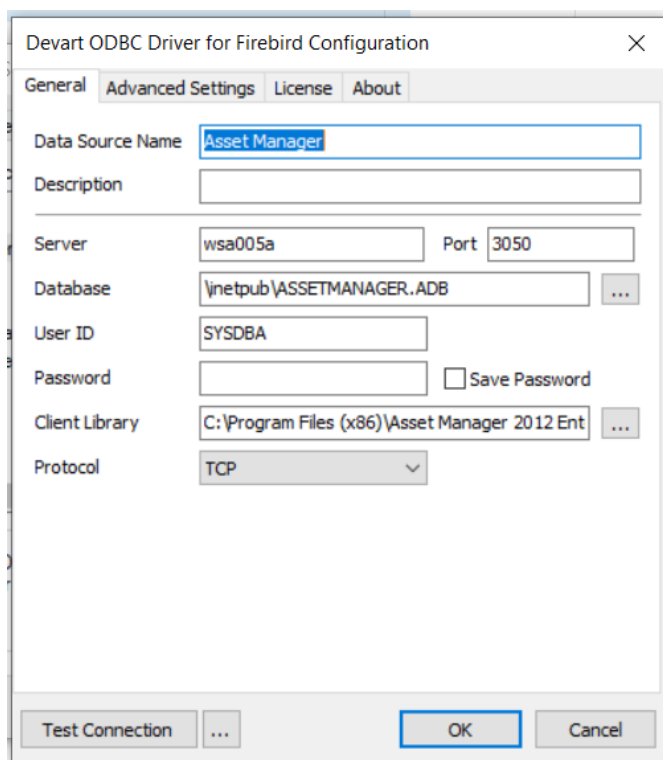


Figure 50: View of Devart Configuration

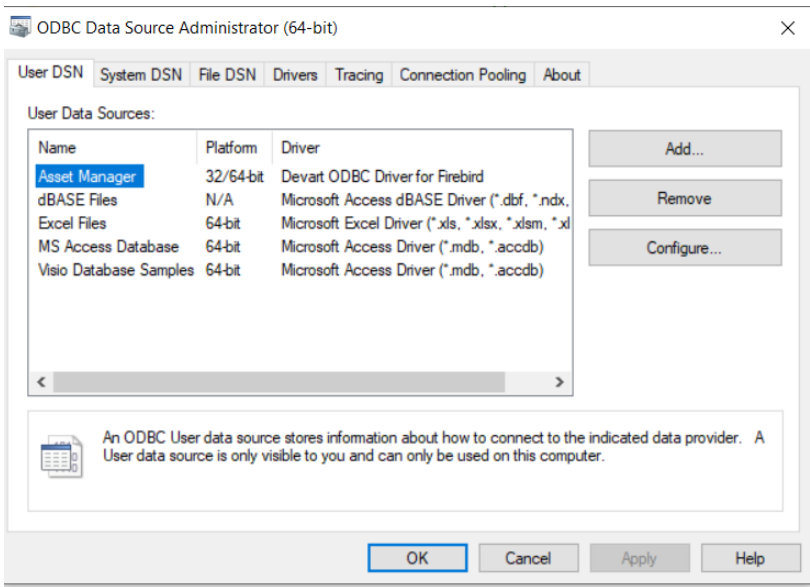


Figure 51: View of connected Data Source

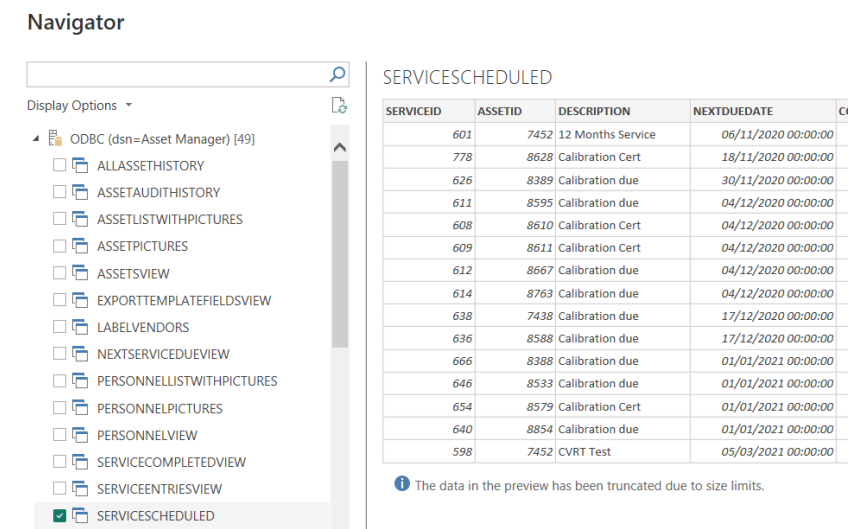


Figure 52: View of Tables available from Asset Manager.

From here I brought in the Servicesheduled table. After analysing the data, I noticed that the table has every service record done for every plant item. The unique identifier for data is the Service ID. To be able to provide a view of upcoming service dates for the plant manager, I generated a DAX query table of the latest Service ID per plant item. From here I then did a join with the existing table and could then generate the relevant views, see screenshots below.

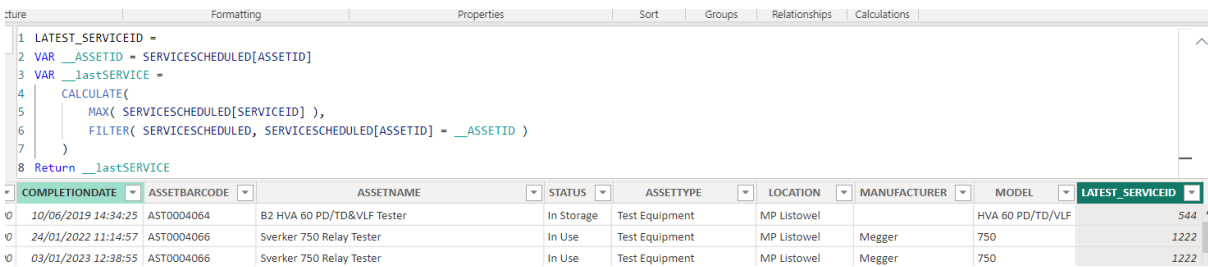


Figure 53: Query of the latest service ID

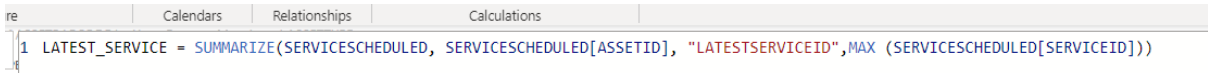


Figure 54: Table query of the latest service ID

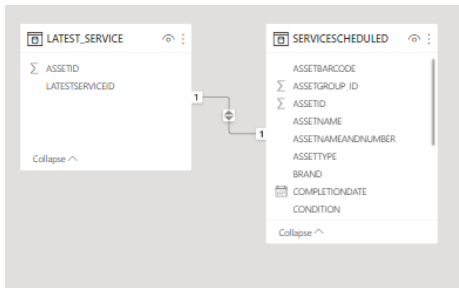


Figure 55: View of the joined table relationship.

The visualization displayed a stacked column table with upcoming plant items, and by utilising the drill down the user could see a list of the Plant items to be serviced per quarter.

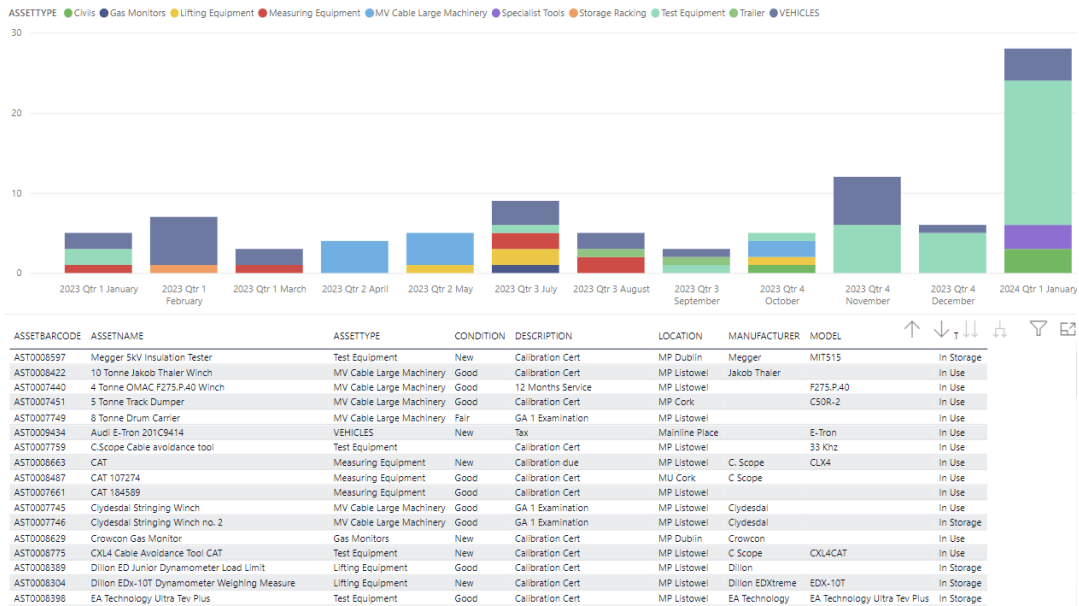


Figure 56: View of asset types and corresponding table.

8.9 Phase 9: Automation of Data flows and Power BI Refresh

The company supervisor required the data to be updated automatically, up to this point in the project I had run all data refresh manually up to this point.

8.9.1 Power Automate – Domain Controller Server

To automate the PipeDrive and Fieldview API scripts I decided to use Power Automate and to use the Domain Controller server, as it has the highest availability in the organisation. The first thing I did was to change the scripts so they can run on the Domain Controller environment. The second thing was to do a test run of the scripts. I could see the data updated locally but the data would not update in SharePoint. The error log showed the following “error Connect-PnPOnline: The term 'Connect-PnPOnline' was not recognized as the name of a cmdlet.” I researched the error and by running the “Install-Module SharePointPnPPowerShellOnline” in PowerShell the issue was resolved.

The method used, using PowerShell script with Power Automate, was the following:

- Install Power Automate Desktop on the device where I want to run the script.
- Create a flow with Power Automate Desktop
 - a. Add the PowerShell module to the flow.
 - b. Copy and paste the script into the PowerShell module.
- Create a cloud flow with PowerShell Online
 - a. Add a Re-occurrence task.
 - b. Add a Run a Power Automate Desktop flow.
 - c. Save and run the flow in “Unattended Mode.”

A prerequisite to this is you have to login with the same Microsoft User account for both Power Automate Desktop and Power Automate Cloud. Below follows screenshots of the process.

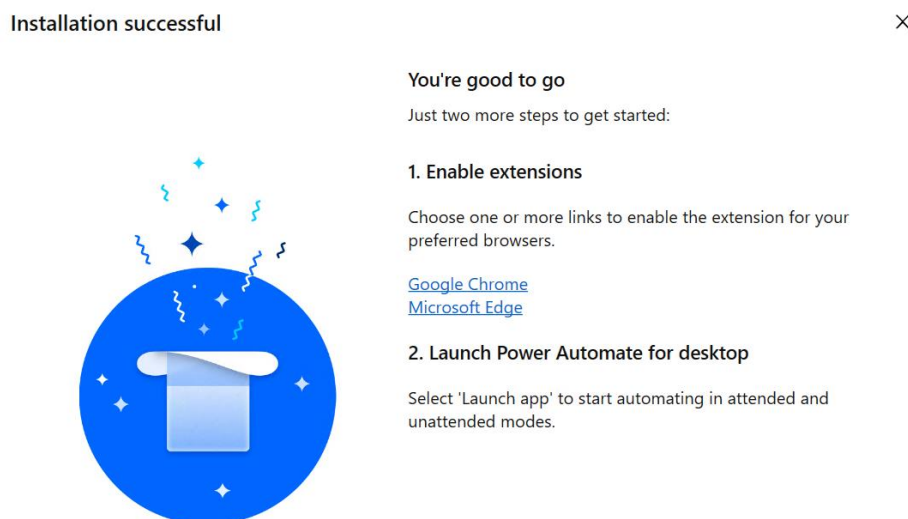


Figure 57: Installation of Power Automate desktop on domain controller server.

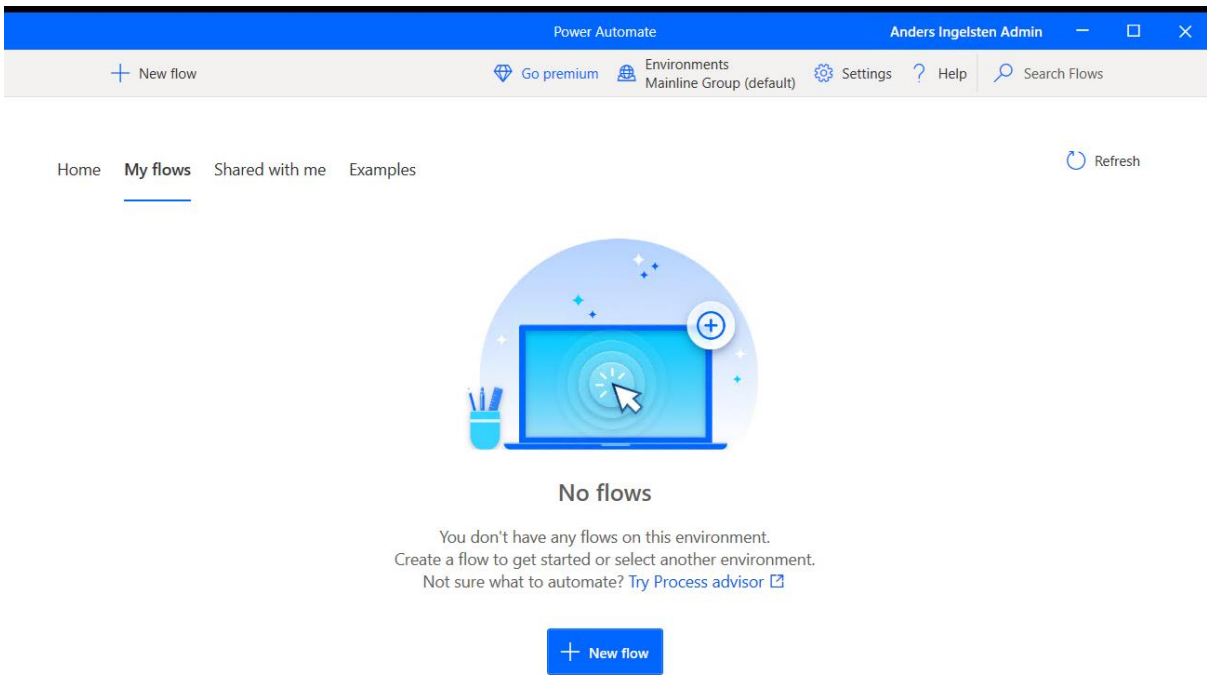


Figure 58: Before implementation of flows

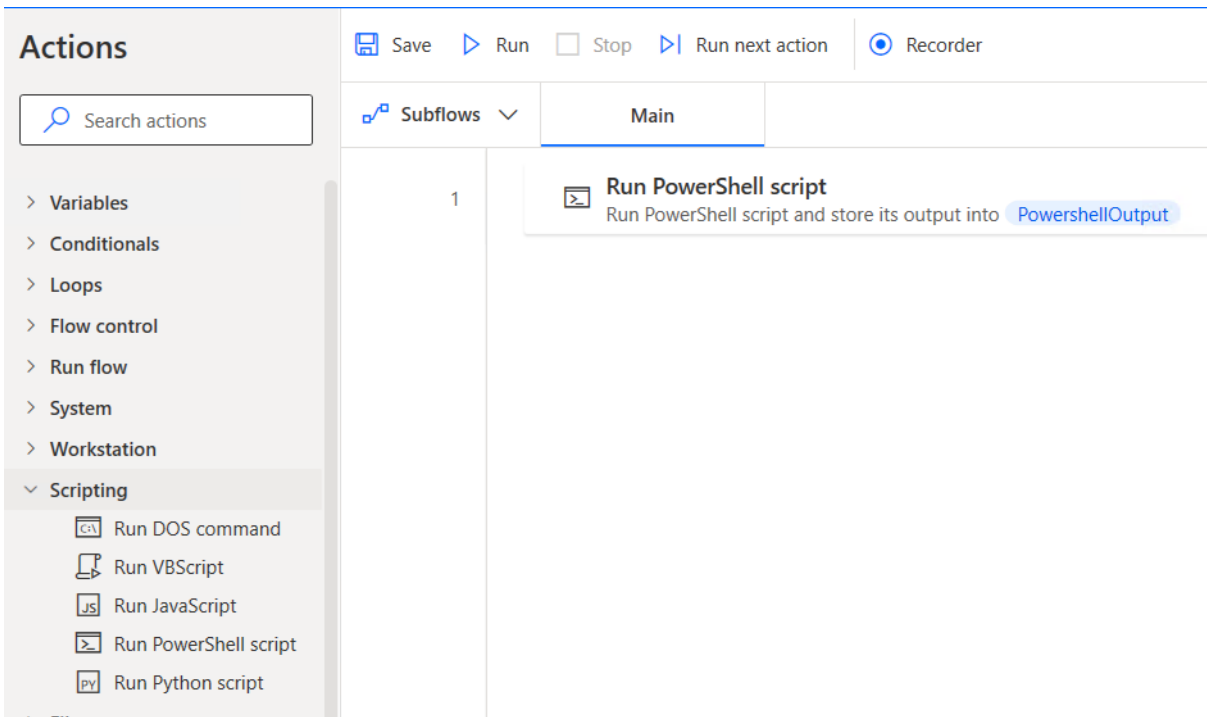


Figure 59: Creating the Run PowerShell Script on Power Automate Desktop

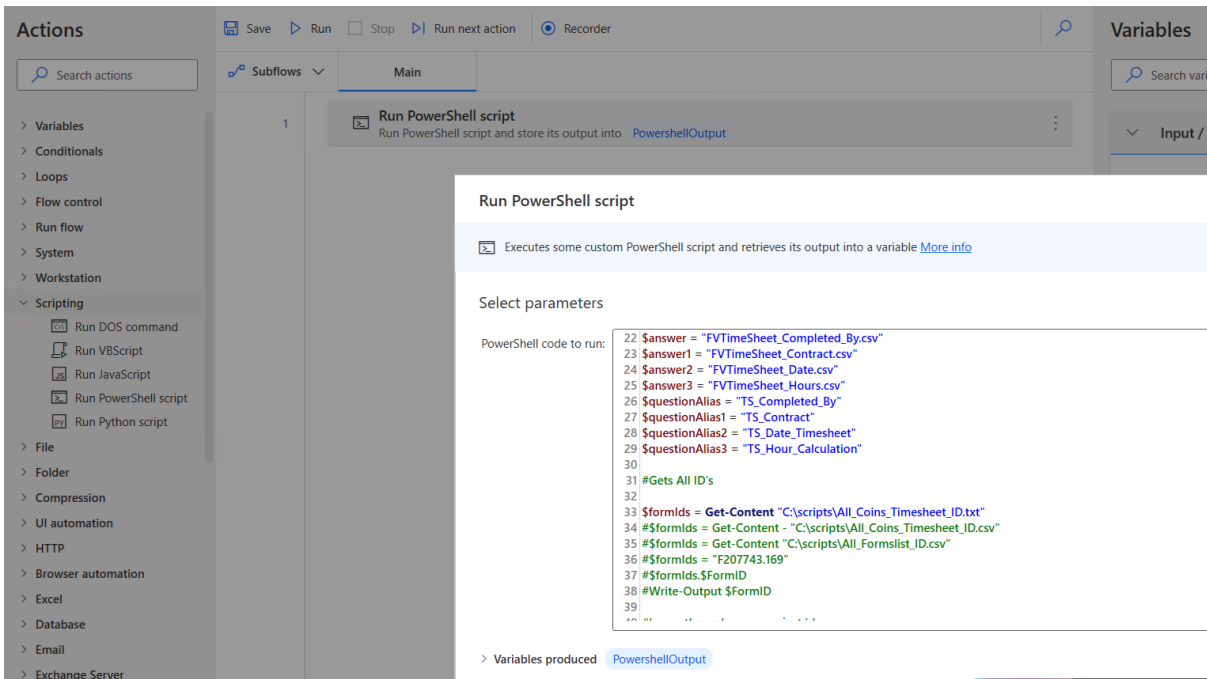


Figure 60: Adding the script to run in the Flow.

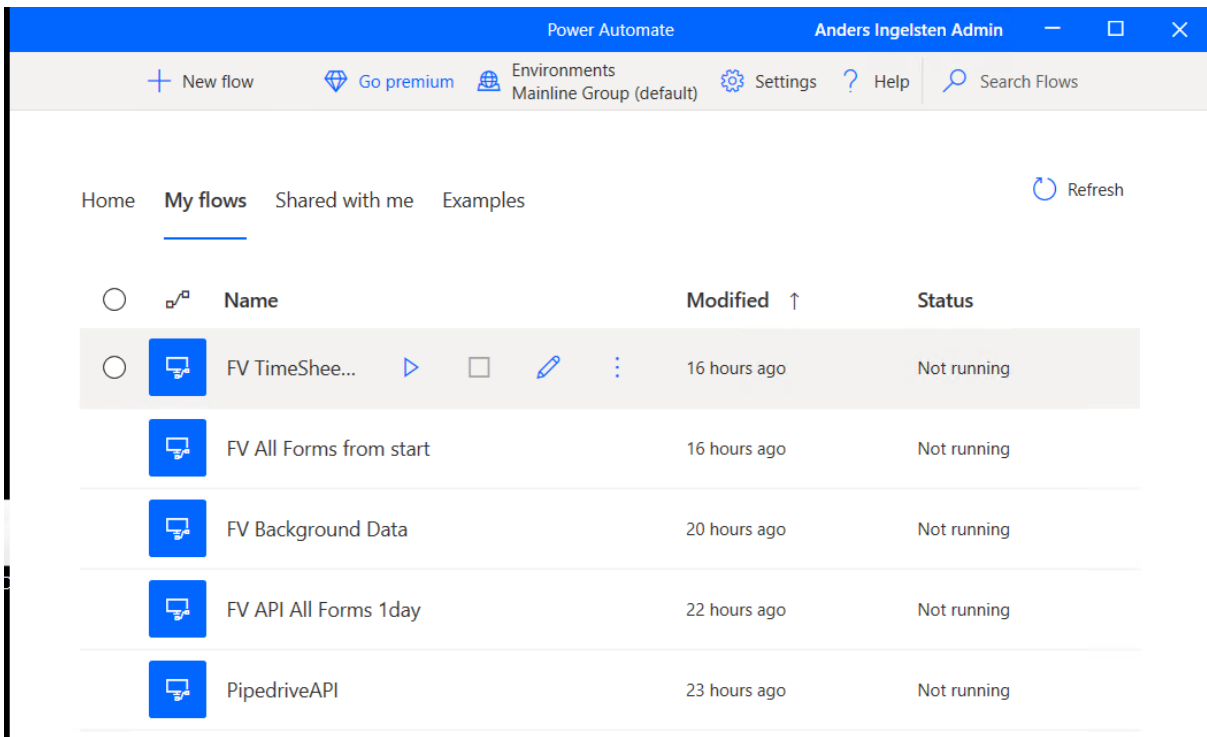


Figure 61: View of all the 5 PowerShell scripts that was setup on the Power Automate desktop.

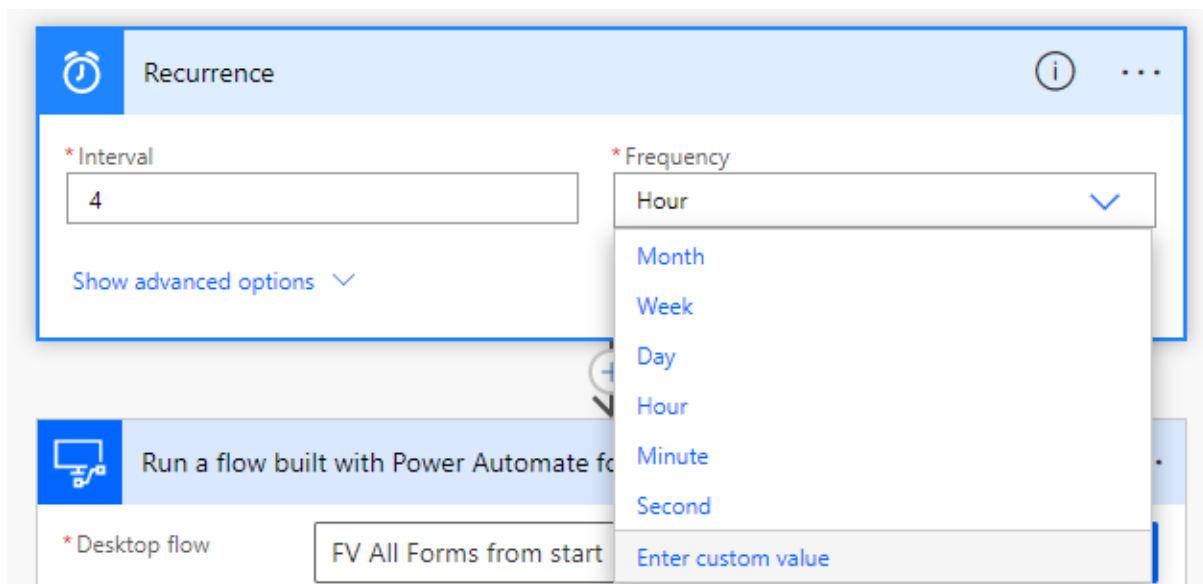


Figure 62: Recurrence step on the Power Automate Cloud

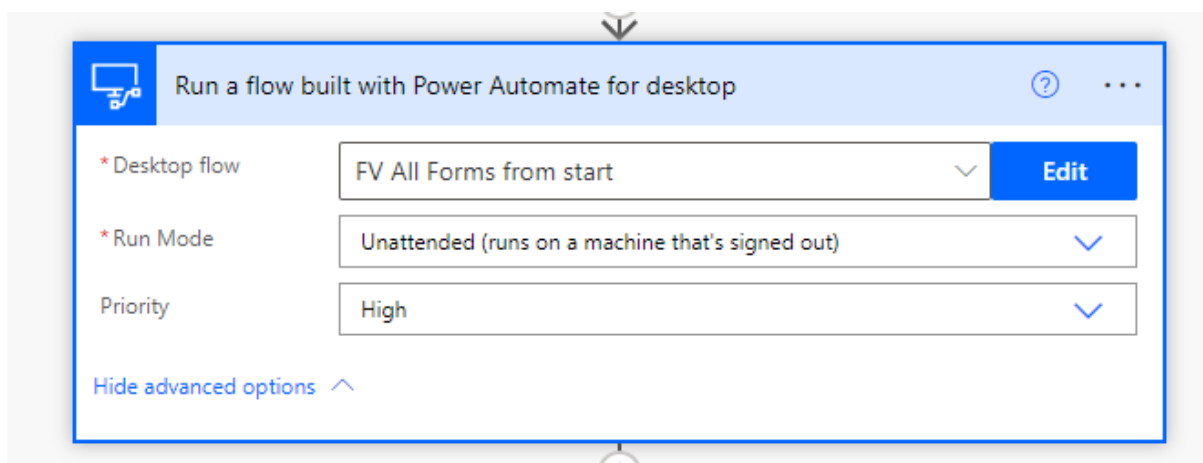


Figure 63: Running the Power Automate desktop flow on the Power Automate Cloud

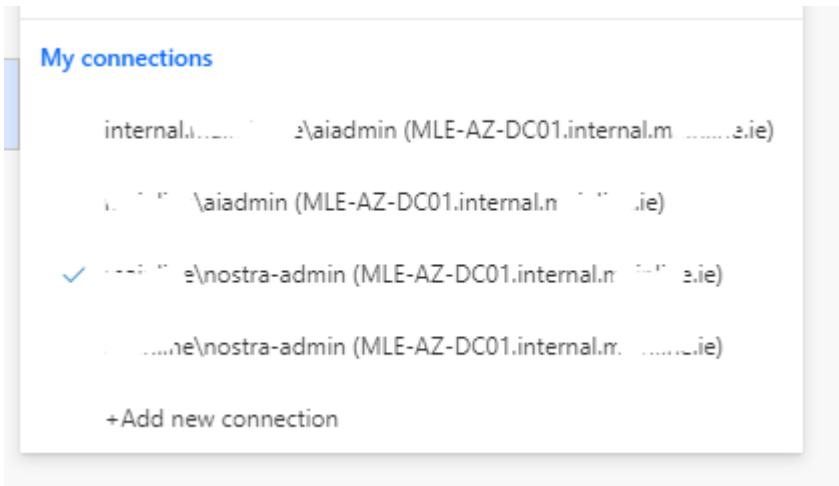


Figure 64: Connection used for cloud hosted servers to run the PowerShell scripts.

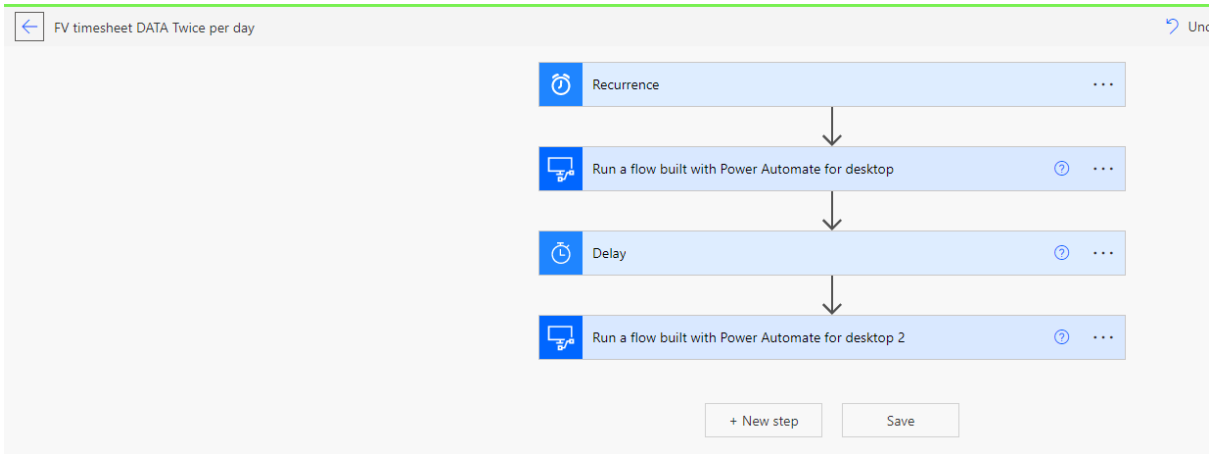


Figure 65: Sample of a flow -this running multiple scripts

Flows			
<div>Cloud flows</div> <div>Desktop flows</div> <div>Shared with me</div>			
	Name	Modified	Type
	FV timesheet DATA Twice per day	16 h ago	Scheduled
	FV Background DATA per day	20 h ago	Scheduled
	FV API forms Every 15min	22 h ago	Scheduled
	PipeDrive Every 15min	22 h ago	Scheduled

Figure 66: Summary view of the cloud flows


8.9.2 Power Automate – Domain Server - Reverting to Task Scheduler

After I completed the flows, I started monitoring to see if data was updated to the SharePoint online stored files. Initial results were fine, and I left the process run for approx. 24 hours. Reviewing the flows and the data after 24 hours I noticed that the data was not updated and that all flows were failing. By reviewing the fault message in Power Automate the errors were due to a user was logged into the machine and the flows would not run when the machine was attended.

Since the Domain Controller is a Server that may be accessed by several users i.e., Sys Admins who may not log off from the machine etc, I had to change the method used for this process. I decided to utilise Task Scheduler which is a Microsoft windows tool that allows for actions to be run on a schedule and defined conditions.

The Task Scheduler is a tool included in the Windows OS that allows predefined actions to be automatically executed whenever a certain set of conditions is met (www.computerhope.com, n.d.). The tool will run regardless of if user is logged on to the machine or not.

Below follows screenshots of the error discovery process and the rectifying of the issue. All initial ran fine.

Flows > FV API forms Every 15min 

Details
[Edit](#)

Flow
FV API forms Every 15min

Owner
Anders Ingelsten Admin


Status
On

Created
Mar 8, 09:51 AM

Modified
Mar 8, 09:58 AM

Type
Scheduled

Plan
This flow runs on owner's plan

28-day run history 
[Edit columns](#)
[All runs](#)

Start	Duration	Status
Mar 8, 11:30 AM (7 min ago)	00:04:41	Succeeded

Figure 67: Initial setup with run history

Mar 8, 06:00 PM (14 h ago)	00:35:26	Succeeded
Mar 8, 04:02 PM (16 h ago)	00:42:01	Succeeded

Figure 68: Successful runs of flows

View of all failed flow history.

28-day run history ⓘ			Edit columns	All runs
Start	Duration	Status		
Mar 9, 07:59 AM (11 min ago)	00:00:13	Failed		
Mar 9, 07:45 AM (26 min ago)	00:00:15	Failed		
Mar 9, 07:30 AM (41 min ago)	00:00:24	Failed		
Mar 9, 07:14 AM (56 min ago)	00:00:16	Failed		
Mar 9, 07:00 AM (1 h ago)	00:00:12	Failed		
Mar 9, 06:44 AM (1 h ago)	00:00:13	Failed		
Mar 9, 06:29 AM (1 h ago)	00:00:23	Failed		
Mar 9, 06:15 AM (1 h ago)	00:00:12	Failed		
Mar 9, 06:00 AM (2 h ago)	00:00:12	Failed		

Figure 69: View failed flows

PipeDrive Every 15min • Ran at 3/9/2023 7:59:59 AM

Resubmit Cancel Edit Help

Flow run failed.

Recurrence0s

Run a flow built with Power Automate for desktop13s

Error Details

Start timeMar 9, 07:59 AM (6 min ago)Duration00:00:13

Error

ActionRun_a_flow_built_with_Power_Automate_for_deskfailed

Error Details

No machine able to run the desktop flow has been found. Aborting execution. Error encountered when connecting to machines: There is a user session on the target machine. Cannot execute unattended desktop flow. (MLE-AZ-DC01.internal.mainline.ie)

Documentation

Learn more about Desktop flow machine /topic/no-machine-able-to-run-the-desktop-f

How to fix

To make this flow work, inspect the inputs to this action and ensure they would provide the correct inputs.

Figure 70: Detailed view of a failed flow

Error Details

No machine able to run the desktop flow has been found. Aborting execution. Error encountered when connecting to machines: There is a user session on the target machine. Cannot execute unattended desktop flow. (MLE-AZ-

Figure 71: Detailed flow error message





	Name	Modified	Type
	FV timesheet DATA Twice per day	16 h ago	Scheduled
	FV Background DATA per day	20 h ago	Scheduled
	FV API forms Every 15min	22 h ago	Scheduled
	PipeDrive Every 15min	22 h ago	Scheduled

Figure 72: Turning off all the Cloud Flows

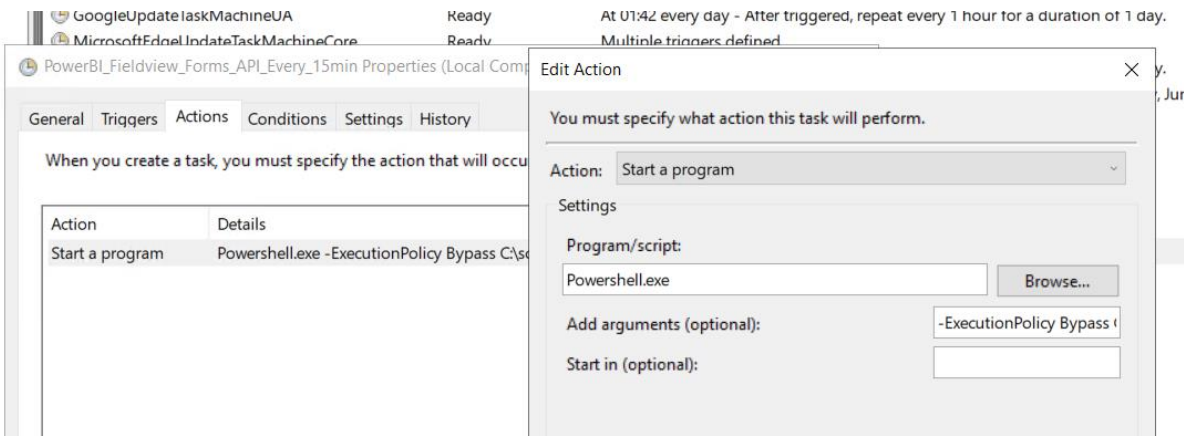


Figure 73: Setting up the Task Scheduler

I decided to have individual tasks per script so when I need to update a script, I will not impact the rest of the updates. To speed up the task creation process this process I used the export and import function available in Task Scheduler. The task was exported to an XML file. The same file was then imported, and relevant details changed as per the individual script.

```

- <Principals>
  - <Principal id="Author">
    <UserId>S-1-5-21-2889858440-467955338-1439349030-1103</UserId>
    <LogonType>InteractiveToken</LogonType>
    <RunLevel>LeastPrivilege</RunLevel>
  </Principal>
</Principals>
- <Settings>
  <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
  <DisallowStartIfOnBatteries>true</DisallowStartIfOnBatteries>
  <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
  <AllowHardTerminate>true</AllowHardTerminate>
  <StartWhenAvailable>false</StartWhenAvailable>
  <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
  - <IdleSettings>
    <StopOnIdleEnd>true</StopOnIdleEnd>
    <RestartOnIdle>false</RestartOnIdle>
  </IdleSettings>
  <AllowStartOnDemand>true</AllowStartOnDemand>
  <Enabled>true</Enabled>
  <Hidden>false</Hidden>
  <RunOnlyIfIdle>false</RunOnlyIfIdle>
  <WakeToRun>false</WakeToRun>
  <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
  <Priority>7</Priority>
</Settings>
- <Actions Context="Author">
  - <Exec>
    <Command>Powershell.exe</Command>
    <Arguments>-ExecutionPolicy Bypass C:\scripts\Automation_All_Forms_From_Start.ps1 -RunType $true</Arguments>
  </Exec>
</Actions>
</Task>

```

Figure 74: Sample view of the export/import XML file of a task

Name	Status	Triggers
GoogleUpdateTaskMachineCore	Ready	Multiple triggers defined
GoogleUpdateTaskMachineUA	Ready	At 01:42 every day - After triggered, repeat every 1 hour fc
MicrosoftEdgeUpdateTaskMachineCore	Ready	Multiple triggers defined
MicrosoftEdgeUpdateTaskMachineUA	Ready	At 00:56 every day - After triggered, repeat every 1 hour fc
Monthly Restart	Ready	Runs on the First, Third Wednesday, each January, Februa
OneDrive Reporting Task-S-1-5-21-2889858440-...	Ready	At 03:45 on 21/02/2023 - After triggered, repeat every 1.00:
OneDrive Standalone Update Task-S-1-5-21-288...	Ready	At 02:00 on 01/05/1992 - After triggered, repeat every 1.00:
PowerBI_Fieldview_AllForms_Every_4hours	Ready	At 04:00 every day - After triggered, repeat every 1 hour ir
PowerBI_Fieldview_AllFVTimehseet_Answers_Ever...	Ready	At 04:50 every day - After triggered, repeat every 1 hour ir
PowerBI_Fieldview_Background_Data_One_Aday	Ready	At 06:16 every day
PowerBI_Fieldview_Forms_API_Every_15min	Ready	At 09:16 every day - After triggered, repeat every 15 minut
PowerBI_PipeDriveAPI_Every_Hour	Ready	At 09:30 every day - After triggered, repeat every 1 hour ir
ShadowCopyVolume{6d558d97-0000-0000-0000-...	Ready	Multiple triggers defined
User_Feed_Synchronization-{75F72BE1-B213-41B...	Ready	At 11:35 every day - Trigger expires at 08/03/2033 11:35:30.
User_Feed_Synchronization-{EEC70044-AEA6-4F...	Ready	At 15:51 every day - Trigger expires at 15/12/2030 15:51:46.
WindowsAdminCenterConfigurationCheck	Ready	Multiple triggers defined

Figure 75: Summary view of the 5 PowerShell scripts in the Task Scheduler

8.9.3 Task Scheduler - Webserver

The Asset Manager database is situated on the Company's web server. The web server is only available from 08.00 to 1800 Monday to Friday. So, for the process of an automated copy, I wrote PowerShell copy script as per below.

```
Copy_AssetManager.ps1 > ...
1  <#
2  AssetManager copy script - to be run to location for PowerBI to access
3  #>
4
5  # This is the file to be copied
6  $source = "C:\KZSoftware\ASSETMANAGER.ADB"
7
8  # Destination for file
9  $dest = "C:\inetpub"
10
11 # Copy of file
12 Copy-Item $source -Destination $dest -Recurse -force
```

Figure 76: Code snippet of copy code

The next thing I did was to open Task Scheduler and create a scheduled task which starts PowerShell and runs the above scripts. Inspiration for this came from the Spiceworks website (Inc, n.d.).

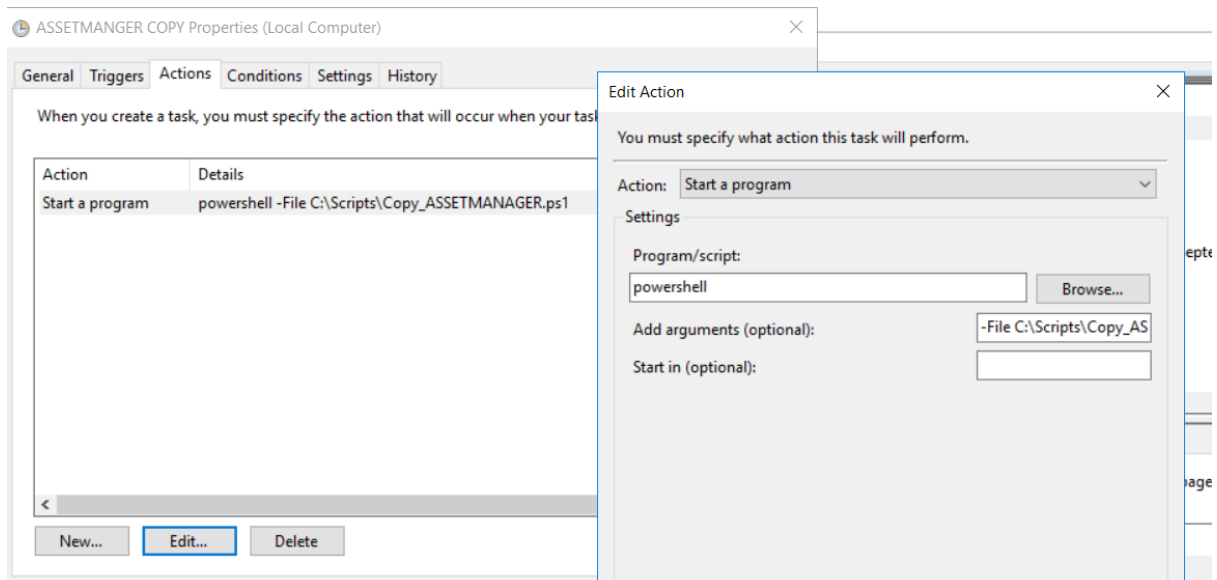


Figure 77: View of Task Scheduler setup to run script every 15 min.

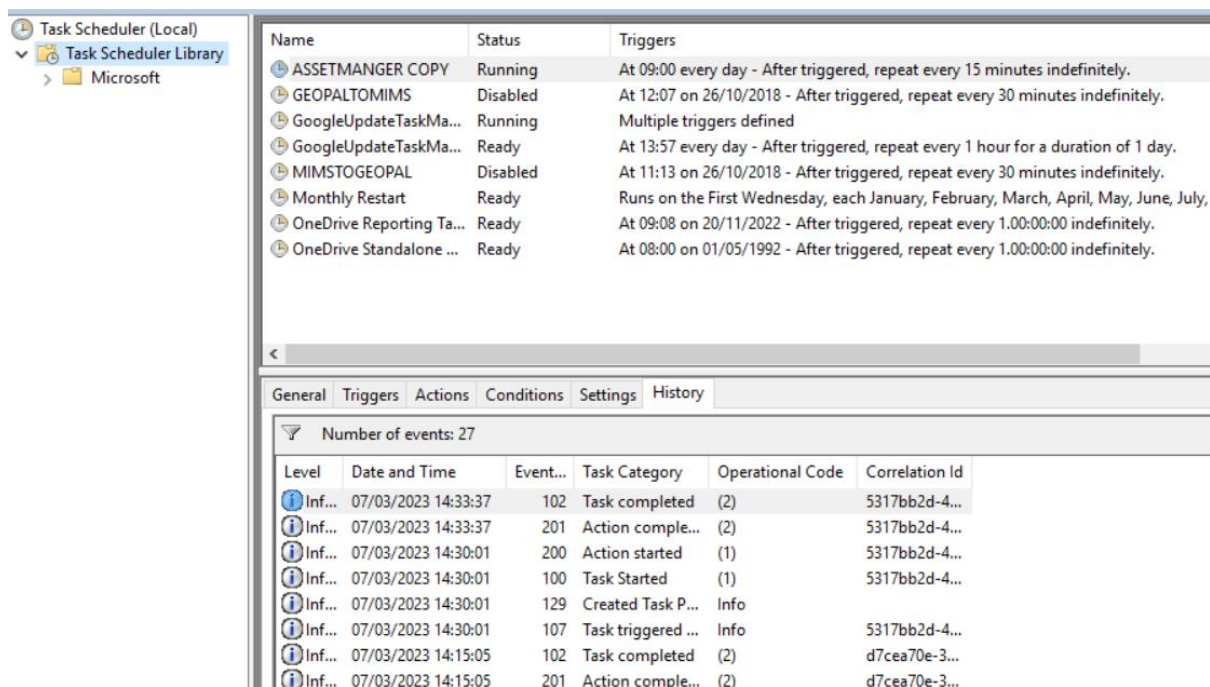


Figure 78: View of Scheduled copy task in situ with running history

I updated data in Asset Manager, waited an appropriate amount of time (30min) and refreshed PowerBI to ensure this process was working.


8.9.4 PowerBI – Automated Refresh and Publish

After I had automated the running of the scripts the next requirement to fulfil was the automation of the PowerBI dataset and views. PowerBI has 2 interfaces, the desktop version and the online version. After researching the matter, the method suitable for the project was to schedule the refresh using functionality in the online version (Davidiseminger, n.d.).

The following was executed, I identified the datasets, installed, and configured an on-premises data gateway on my company domain joined laptop to ensure connectivity between the PowerBI and the data sets. OAuth2 was used as the authentication method and privacy level was set to organizational. The refresh was set to happen during weekdays between business hours. Below follows screenshots of the process.

Dashboards **Datasets** Workbooks Reports Dataflows App

Settings for PowerBI Project

[View dataset](#) 

This dataset has been configured by [aigelsten@i...e.ie](#).
[Refresh history](#)

- ▷ Dataset description
- ▷ Gateway connection
- ▷ Data source credentials
- ▷ Parameters
- ⌵ Scheduled refresh
 - Keep your data up to date**
 - Configure a data refresh schedule to import data from the data source into the dataset. [Learn more](#)

Figure 79: PowerBI Datasets view for automated updates.

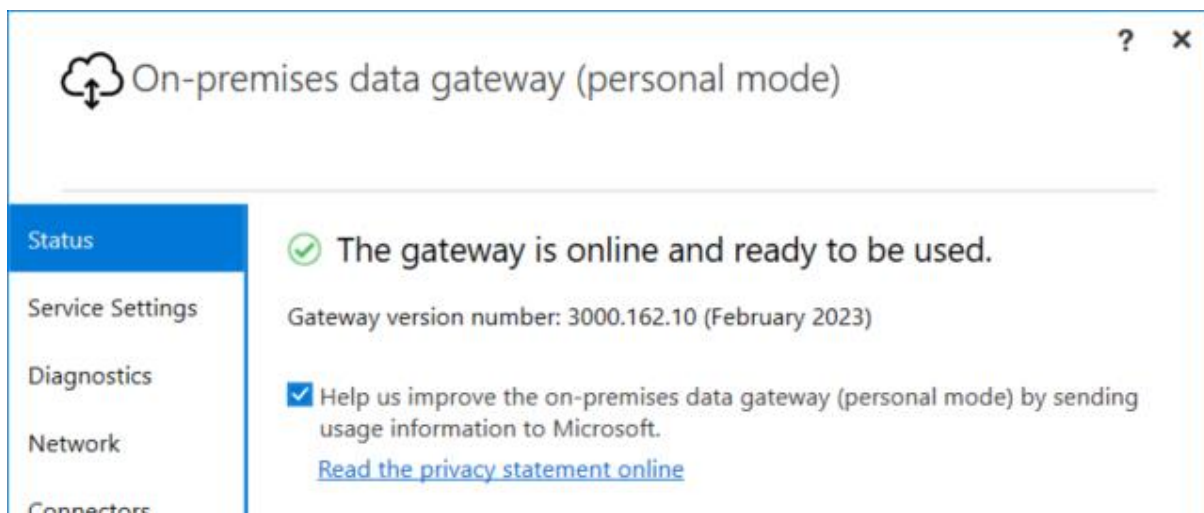


Figure 80: On-Premises data gateway

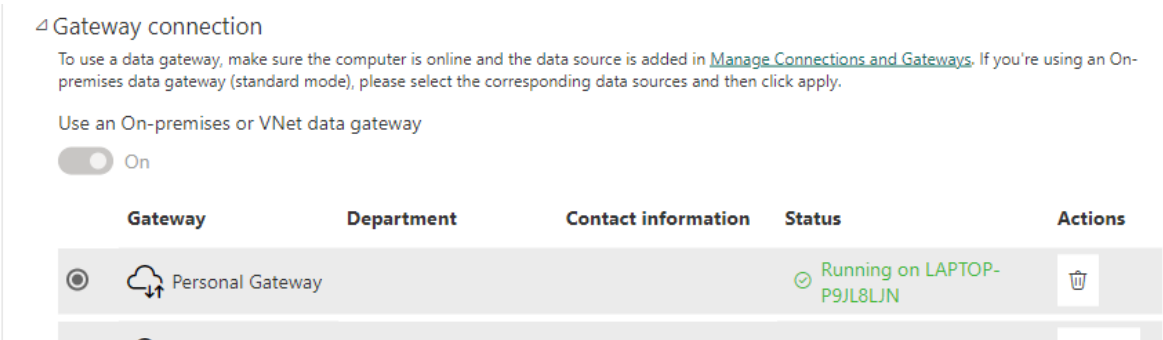


Figure 81: Gateway connected.

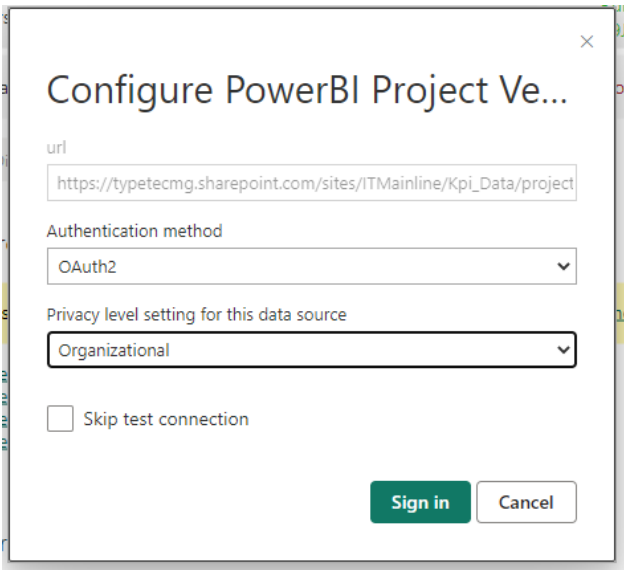


Figure 82 :Authentication of data sources

⌵

Scheduled refresh

Keep your data up to date

Configure a data refresh schedule to import data from the data source into the dataset. [Learn more](#)

On

Refresh frequency

Weekly

Time zone

(UTC) Dublin, Edinburgh, Lisbon, Lon

☐

Sunday

☒

Monday

☒

Tuesday

☒

Wednesday

☒

Thursday

☒

Friday

☐

Saturday

Time

9

00

AM

×

11

00

AM

×

1

00

PM

×

3

00

PM

×

5

00

PM

×

Figure 83: Scheduled refresh view

Refresh history

Scheduled OneDrive

Details	Type	Start	End	Status	Message
	Scheduled	16/3/2023, 17:06:19	16/3/2023, 17:06:29	Completed	
	Scheduled	16/3/2023, 15:08:01	16/3/2023, 15:08:14	Completed	
	Scheduled	16/3/2023, 13:08:04	16/3/2023, 13:08:23	Completed	
	Scheduled	16/3/2023, 11:08:01	16/3/2023, 11:08:14	Completed	
	Scheduled	16/3/2023, 09:08:01	16/3/2023, 09:08:21	Completed	

Figure 84: Refresh History

The implementation of the refresh was successful, and all my PowerBI dashboards updated successfully as per set schedule. One minor thing I noticed was that the refresh doesn't run exactly on the set time, my assumption here was that they are queued in the Microsoft cloud thus the 5–8-minute delay.

A while after I had implemented the phase I had to go to site to assist with a printing issue, this meant I had to bring my laptop with me. During the call out I received an email with an error notification, see below.

An error occurred while processing the data in the dataset.

Refresh failed:

PowerBI Project VehicleCheck has failed to refresh.

Failure details: The last refresh attempt failed because of an internal service error. This is usually a transient issue. If you try again later and still see this message, contact support.

```
{
  "error": {
    "code": "DM_GWPipeline_Gateway_MashupDataAccessError",
    "pbi.error": {
      "code": "DM_GWPipeline_Gateway_MashupDataAccessError",
      "parameters": {},
      "details": [
        {
          "code": "DM_ErrorDetailNameCode_UnderlyingErrorCode",
          "detail": {
            "type": 1,
            "value": "-2147467259"
          }
        },
        {
          "code": "DM_ErrorDetailNameCode_UnderlyingErrorMessage",
          "detail": {
            "type": 1,
            "value": "<ccon>ODBC: ERROR [08001] [Devart][ODBC][Firebird]Unable to complete network request to host \\wsa005a\\.rFailed to locate host machine.rThe specified name was not found in the hosts file or Domain Name Services.</ccon>"
          }
        },
        {
          "code": "DM_ErrorDetailNameCode_UnderlyingHResult",
          "detail": {
            "type": 1,
            "value": "-2147467259"
          }
        },
        {
          "code": "Microsoft.Data.Mashup.ValueError.DataSourceKind",
          "detail": {
            "type": 1,
            "value": "Odbc"
          }
        },
        {
          "code": "Microsoft.Data.Mashup.ValueError.DataSourcePath",
          "detail": {
            "type": 1,
            "value": "<ccon>dsn=Asset Manager</ccon>"
          }
        },
        {
          "code": "Microsoft.Data.Mashup.ValueError.OdbcErrors",
          "detail": {
            "type": 1,
            "value": "#table({\\SQLState\\, \\NativeError\\, \\Message\\, {}})"
          }
        },
        {
          "code": "Microsoft.Data.Mashup.ValueError.Reason",
          "detail": {
            "type": 1,
            "value": "DataSource.Error"
          }
        }
      ]
    },
    "exceptionCulprit": 1
  }
}
Table: SERVICESCHEDULED.
```

Figure 85: PowerBI refresh error.

After analysing the error, this line *“Unable to complete network request to host \\wsa005a\\.rFailed to locate host machine”* had the key to the answer – the issue was due to the laptop was not connected to the Company’s domain - the data gateway could not connect to webserver “wsa005a “. I connected to the domain using the company’s VPN client, SonicWALL Netextender, and I could then run a successful PowerBI data refresh.



Figure 86: SonicWALL Netextender.

With the PowerBI data refresh completed and functioning, I had achieved what I was asked to do by the company supervisor, and this concluded the implementation phase within the envisaged scope of the project.

9 Project Evaluation

9.1 Achievements

With the project I believe I achieved the objectives and solved the two main problems I set out to resolve.

Objectives achieved and problems solved.

- Ease of access; a greater number of staff can now see the information published in dashboard on the company's intranet.
- Real-time information: users are not waiting days to review the data.
- Process improvement: less time spent by staff to collect and produce reports.
- Less manual input of data and time spent on this process.

9.2 Knowledge attained.

I found that the subject matter within this project was natural extensions of the topics covered in the different modules of the Higher Diploma in Computer Science. More importantly it provided me a base from where I have the confidence to do research and apply solutions suitable for the project. It also allowed me to see and resolve errors more easily. To summarize the project has provided the knowledge and skills to

- Setup, generate and publish report views in PowerBI.
- Pull in data sets and model their relationships by common identifiers.
- Use built in as well as third party connectors in PowerBI.
- Use PowerShell to pull data over API's.
- Use DAX formulae language to create bespoke data columns and measures in PowerBI.
- Automating the running of PowerShell scripts and PowerBI data sets.
- Project Manage an internal IT project from start to finish.
- Gather company requirements, research, and implement solutions for the same.
- Communication and leadership skills in a project manager capacity
- Greater knowledge of PowerBI and its limitations as well as possibilities.

9.3 Future Learning

I have identified I need further education and self-development in the following areas.

- The PowerBI platform and its Data Queries
- DAX scripting language
- Sharepoint online management shell
- Microsoft Dataverse business layer

9.4 Future Development Work

I can foresee the following future development work in relation to the projects. I have categorised them into three categories.

9.4.1 Review of the current state

As the data and requests will grow, I anticipate iteration and tweaking of PowerShell scripts, review of the task scheduler refresh schedule, and the PowerBI refresh schedule, view changes in PowerBI and DAX scripts updates. My assumption is that this is something that I will be continuously looking at in my role at the company. Below are examples screenshots of minor changes that I have done after its implementation phase.

OneDrive Standalone updat...	Ready	At 02:00 on 01/05/1992 - After triggered, repeat every 1.000000 indefinitely.
PowerBI_Fieldview_AllForms...	Ready	At 18:00 every Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday of every week, starting 09/03/2023
PowerBI_Fieldview_AllFVTim...	Ready	At 04:50 every day - After triggered, repeat every 1 hour indefinitely.
PowerBI_Fieldview_Backgrou...	Ready	At 06:16 every day
PowerBI_Fieldview_Forms_A...	Running	At 09:16 every day - After triggered, repeat every 15 minutes indefinitely.
PowerBI_PipeDriveAPI_Every...	Running	At 16:30 every Monday, Tuesday, Wednesday, Thursday, Friday of every week, starting 09/03/2023
ShadowCopyVolume6d558d...	Ready	Multiple triggers defined

Figure 87: Change of Task Scheduler

```

1 Total_Value = CALCULATE (
2     SUM( Deals_pipe[monthly_value] ),
3     FILTER (
4         Deals_pipe,
5         Deals_pipe[Start_Date] <= EARLIER ( Calender_Months[Month] )
6         && 'Deals_pipe'[End_Date] >= EARLIER ( Calender_Months[Month] )
7     )
8 )

```

Figure 88: DAX formula to resolve weighted monthly value.

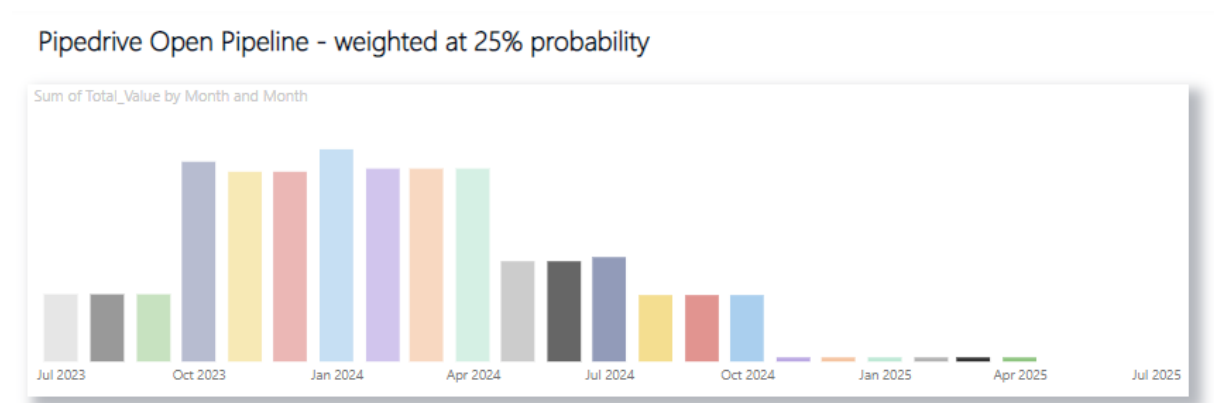


Figure 89: PowerBI view of weighted monthly value

9.4.2 Adding more data sources

As the project is now established in the Company, I anticipate I will be asked to gather more data sources and publish more business data analytics for stakeholders within the company. This was made

explicitly clear to me when during one of our regular project meetings my supervisor turned to me and said: *“Best thing you can do now, is to shut up and not to tell anyone else about this and remember look after our team first.”*

The meaning of this was twofold, one was not allowing the project scope to creep, and secondly if I would have allowed the rest of the organisation know of the capabilities we had in hand, there was a chance I would get inundated with data query request from all parts of the organisation.

Company A is in the utilities business. It would be of terrific value for a project manager, contract manager or the commercial team to know the real progress of a project, for example a duct excavation on target as per the project plan. Below follows a typical request received during the project development process by a site manager.

“Hi Anders,

Not sure if it's possible, maybe it's there, I was wondering if it were possible to create a View where a person could just fill in a Tab showing a Weekly Progress Report say on the Meterage of Grid Duct Trenching / Installation, that may run an excel file in the background which would show for instance the agreed amount per week which is a fixed point & the other the actual, Or do we have something else set up, as it would be great to track, say cable installed, joints made off etc”

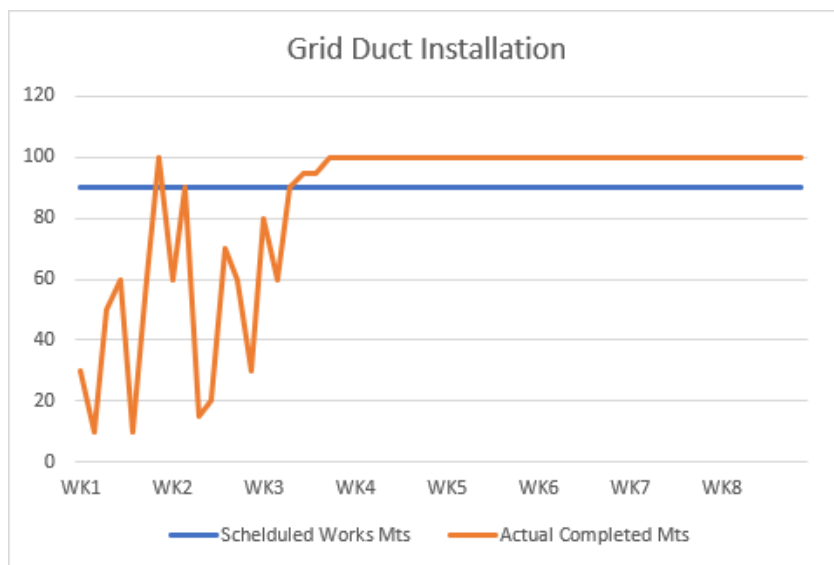


Figure 90: Sample of requested view

9.4.3 Long Term future functionality

In relation to Data persistence, storing data as CSV files in SharePoint online might be a solution for the immediate future. Overtime other solutions will be reviewed, for example SharePoint Lists, MS Dataverse and Azure SQL Database can possibly be in scope. In addition, adding systems that have not yet considered to be on the power BI platform. For example, the Company's ERP system and the HR management system.

10 Conclusion

I have successfully connected Smartsheet, Asset manager as data sources. Researched and created PowerShell code that could access and pull data over APIs from Pipedrive and Fieldview. Subsequently the same code stored CSV files onto Sharepoint Online as data sources to PowerBI. The data was presented in several report visuals on dashboards which was published on the company's intranet. Finally, the data update and refresh were automated and running on schedule.

I remember well the initial review of the first iteration of a published dashboard of real company timesheet data to the company's CFO went beyond my expectations. The person's eyes lit up, the penny had dropped, the feedback was immensely positive and new requirements were expressed immediately. *"Can we show this per month per person instead, and what about this, can we compare with previous month?"*

From a personal view, on many occasions during this project several thoughts kept re-occurring, these self-reflecting thoughts spanned across various topics but to highlight the most descriptive ones:

"I couldn't have done this 2 years ago".

"It works! - look at all that data coming in."

"Is this right? Is the tool and methods I used, correct and stable?"

Being the sole responsible person for the project, (the company supervisor provided no direction in relation to preferred techniques or methods), I had to look at the surrounding conditions, environment, and cost, then decide and proceed down a chosen path, deal with problems encountered and solve them on the go. This is where I believe I have learned the most and where the acquired knowledge from the course assisted me in applying best practise.

Furthermore, the expression "garbage in, garbage out" was truly relevant in this project. Both from the perspective of understanding the data and that the reports displayed in Power BI were true. To also seeing samples of uncomplete and erroneous data and how to apply the appropriate business logic.

In the end, I enjoyed the entire process of exploration, research, implementation and mostly seeing the result which is now benefiting the reporting process of the Company.

11 References

AltexSoft. (n.d.). What is SOAP: Formats, Protocols, Message Structure, and How SOAP is Different from REST. [online] Available at: <https://www.altexsoft.com/blog/engineering/what-is-soap-formats-protocols-message-structure-and-how-soap-is-different-from-rest/> [Accessed 27 Mar 2023].

www.computerhope.com. (n.d.). What is Task Scheduler? [online] Available at: <https://www.computerhope.com/jargon/t/tasksched.htm> [Accessed 29 Mar. 2023].

Davidiseminger (n.d.). Configure scheduled refresh - Power BI. [online] learn.microsoft.com. Available at: <https://learn.microsoft.com/en-us/power-bi/connect-data/refresh-scheduled-refresh> [Accessed 17 Mar. 2023].

Developers.pipedrive.com. (n.d.). Pipedrive API v1 Reference. [online] Available at: <https://developers.pipedrive.com/docs/api/v1> [Accessed 19 Feb. 2023].

Docs.devart.com. (n.d.). Connecting Power BI to Firebird via ODBC Driver. [online] Available at: <https://docs.devart.com/odbc/firebird/powerbi.htm> [Accessed 4 Mar. 2023].

Floor, T.V.U.O. 4th, Square, C., Tyne, F.S.N.U. and Kingdom +44.0800.048.8152, N. 3PJ U. (n.d.). *Cloud-Based Field Construction Software*. [online] Trimble Viewpoint. Available at: <https://www.viewpoint.com/en-gb/products/viewpoint-field-view> [Accessed 27 Mar 2023].

Georgiostrantzis (n.d.). Scripting actions reference - Power Automate. [online] learn.microsoft.com. Available at: <https://learn.microsoft.com/en-us/power-automate/desktop-flows/actions-reference/scripting> [Accessed 4 Mar. 2023].

Health and Safety Authority. (n.d.). Managing Driving for Work Information Sheet. [online] Available at: https://www.hsa.ie/eng/vehicles_at_work/driving_for_work/employer_responsibilities/ [Accessed 25 Feb. 2023].

Help.viewpoint.com. (n.d.). Viewpoint Help. [online] Available at: <https://help.viewpoint.com/en/viewpoint-field-view/field-view/api-documentation/apis> [Accessed 4 Mar. 2023].

Home. (n.d.). Firebird: The true open source database for Windows, Linux, Mac OS X and more. [online] Available at: <https://firebirdsql.org/en/start/> [Accessed 20 Feb. 2023].

Inc, S. (n.d.). Run PowerShell Scripts from Task Scheduler. [online] community.spiceworks.com. Available at: https://community.spiceworks.com/how_to/17736-run-powershell-scripts-from-task-scheduler [Accessed 11 Mar. 2023].

www.kzsoftware.com. (n.d.). Asset Management Software - A Fixed Assets Register. [online] Available at: <https://www.kzsoftware.com/products/asset-management-software/> [Accessed 4 Mar. 2023].

www.microsoft.com. (n.d.). SharePoint, Team Collaboration Software Tools. [online] Available at: <https://www.microsoft.com/en-ie/microsoft-365/sharepoint/collaboration>[Accessed 27 Mar 2023].

www.microsoft.com. (n.d.). Task Management Kanban Solution for Teams | Microsoft Planner. [online] Available at: <https://www.microsoft.com/en-ie/microsoft-365/business/task-management-software>[Accessed 27 Mar 2023].

Microsoft (2022). Data Visualisation | Microsoft Power BI. [online] powerbi.microsoft.com. Available at: <https://powerbi.microsoft.com/en-gb/>[Accessed 27 Mar 2023].

Minewiskan (n.d.). Data Analysis Expressions (DAX) Reference - DAX. [online] learn.microsoft.com. Available at: <https://learn.microsoft.com/en-us/dax/>[Accessed 27 Mar 2023].

Pipedrive Inc / Pipedrive OÜ (2018). Sales CRM & Pipeline Management Software. [online] Pipedrive. Available at: <https://www.pipedrive.com/>[Accessed 27 Mar 2023].

Powerautomate.microsoft.com. (n.d.). Power Automate | Microsoft Power Platform. [online] Available at: <https://powerautomate.microsoft.com/en-gb/>[Accessed 27 Mar 2023].

Rad, R. (2020). Replace BLANK with Zero in Power BI Visuals Such as Card. [online] RADACAD. Available at: <https://radacad.com/replace-blank-with-zero-in-power-bi-visuals-such-as-card> [Accessed 25 Feb. 2023].

Rajack, S. (2018). How to Connect to SharePoint Online using PnP PowerShell? [online] SharePoint Diary. Available at: <https://www.sharepointdiary.com/2018/03/connect-to-sharepoint-online-using-pnp-powershell.html>[Accessed 27 Mar 2023].

Sdwheeler (n.d.). Invoke-RestMethod (Microsoft.PowerShell.Utility) - PowerShell. [online] learn.microsoft.com. Available at: <https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/invoke-restmethod?view=powershell-7.3> [Accessed 19 Feb. 2023].

Sdwheeler (n.d.). PowerShell Documentation - PowerShell. [online] learn.microsoft.com. Available at: <https://learn.microsoft.com/en-us/powershell/>[Accessed 27 Mar 2023].

Seidlm (2021). PIPEDRIVE-PowerShell/1-Basic Example.ps1 at main · Seidlm/PIPEDRIVE-PowerShell. [online] GitHub. Available at: <https://github.com/Seidlm/PIPEDRIVE-PowerShell/blob/main/1-Basic%20Example.ps1> [Accessed 19 Feb. 2023].

Smartsheet (2019). Smartsheet: Less Talk, More Action. [online] Smartsheet. Available at: <https://www.smartsheet.com/>[Accessed 27 Mar 2023].

Technologies, S.P. (n.d.). Spanish Point Technologies. [online] Available at: <https://www.spanishpoint.ie>[Accessed 27 Mar 2023].

www.youtube.com. (n.d.). Power Automate Desktop || How to schedule Desktop Flows? [online] Available at: <https://www.youtube.com/watch?v=IJ1I2737JWU>[Accessed 27 Mar 2023].

ZappySys (2018). Calling SOAP API in Power BI (Read XML Web Service data). [online] ZappySys Blog. Available at: <https://zappysys.com/blog/call-soap-api-power-bi-read-xml-web-service-data>[Accessed 27 Mar 2023].