



KPI PROCESS IMPROVEMENT

Final Report

Anders Ingelsten 20095402

Higher Diploma in Science in Computer Science

1 Contents

.....	0
2 Table of Figures	4
3 Summary	6
4 Introduction	7
4.1 Project Type	7
4.2 The organisation	7
4.3 System Background and Project Scope	7
4.4 Fieldview.....	8
4.5 Smartsheet	9
4.6 Sharepoint/Office365.....	9
4.7 Current Situation.....	9
4.8 What are the problems.	10
5 Use Case.....	10
6 Potential Technologies, Tools, and Languages.....	11
6.1 Potential Issues	11
7 Objectives.....	11
8 Methodology	11
8.1 Dataflow	11
8.2 Modelling.....	12
8.3 Feedback Loop	13
8.4 Technologies.....	13
8.4.1 PowerBI.....	13
8.4.2 SOAP API	13
8.4.3 PowerShell	14
8.4.4 Power Automate	14
8.4.5 DAX	14
9 Project Plan	15
10 Project Planner	15
11 Implementation	17
11.1 Phase 1: Research & Training	17
11.2 Phase 2: Smartsheet Data Connector setup.....	19
11.3 Phase 3: FV API setup and Data Persistence	21

11.3.1	Data persistence.....	23
11.4	Phase 4: PowerBI Dashboard – Fieldview Vehicle-check	24
11.5	Phase 5: PowerBI Dashboard – Pipedrive	28
11.6	Phase 6: PowerBI Dashboard – SmartSheet Timesheet.....	30
11.7	Phase 7: PowerBI Dashboard – Fieldview Timesheet	32
11.8	Phase 8: PowerBI Dashboard – Asset Manager	36
11.9	Phase 9: Automation of Data flows and Power BI Refresh	39
11.9.1	TaskScheduler - Webserver	39
11.9.2	PowerAutomate – Domain Controller Server	40
11.9.3	Power Automate – Domain Server - Reverting to Task Scheduler.....	45
11.9.4	PowerBI – Automated Refresh and Publish.....	49
12	Project Evaluation.....	50
12.1	Self-Reflection.....	50
12.2	Knowledge attained.	50
12.3	Achievements	50
12.4	Future Learning.....	50
12.5	Future Work and Development	51
13	References.....	53

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

2 Table of Figures

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

Figure 41: Code snippet of API call.....	33
Figure 42: Code snippet of export to SharePoint.....	33
Figure 43: Query view in PowerBI	34
Figure 44: Table relationships in PowerBI	34
Figure 45: Summary visualisation of the hours per contract and user	35
Figure 46: Drill down view per contract, month, and person.....	35
Figure 47: View of Devart Configuration	36
Figure 48:View of connected Data Source.....	37
Figure 49: View of Tables available from Asset Manager.....	37
Figure 50: Query of the latest service ID	38
Figure 51:Table query of the latest service ID	38
Figure 52:View of the joined table relationship.....	38
Figure 53:View of asset types and corresponding table.....	38
Figure 54: Code snippet of copy code	39
Figure 55: View of Task Scheduler setup to run script every 15 min	39
Figure 56: View of Scheduled copy task in situ with running history.....	40
Figure 57: Installation of Power Automate desktop on domain controller server	41
Figure 58: Before implementation of flows	41
Figure 59: Creating the Run PowerShell Script on Power Automate Desktop	42
Figure 60: Adding the script to run in the Flow	42
Figure 61: View of all the 5 PowerShell scripts that was setup on the Power Automate desktop	43
Figure 62: Recurrence step on the Power Automate Cloud	43
Figure 63: Running the Power Automate desktop flow on the Power Automate Cloud	44
Figure 64: Connection used from Cloud to Hosted Servers to run the Powershell scripts	44
Figure 65: Sample of a flow -this running multiple scripts	44
Figure 66:Summary view of the cloud flows.....	45
Figure 67: Initial setup with run history	46
Figure 68: Successful runs of flows	46
Figure 69: View of several failed flows.....	47
Figure 70: Detailed view of a failed flow	47
Figure 71: Detailed flow error message	47
Figure 72: Turning off all the Cloud Flows	48
Figure 73: Setting up the Task Scheduler	48
Figure 74: Sample view of the export/import XML file of a task	49
Figure 75: Summary view of the 5 Powershell scripts in the TaskScheduler.....	49
Figure 76: Sample of requested view 1	51
Figure 77: Sample of requested view 2	52

3 Summary

Text here

4 Introduction

4.1 Project Type

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

4.2 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 approx. one hundred staff.

4.3 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) metrics for the different departments. The reports are generated on a weekly or monthly basis but there are also ad hoc reports.

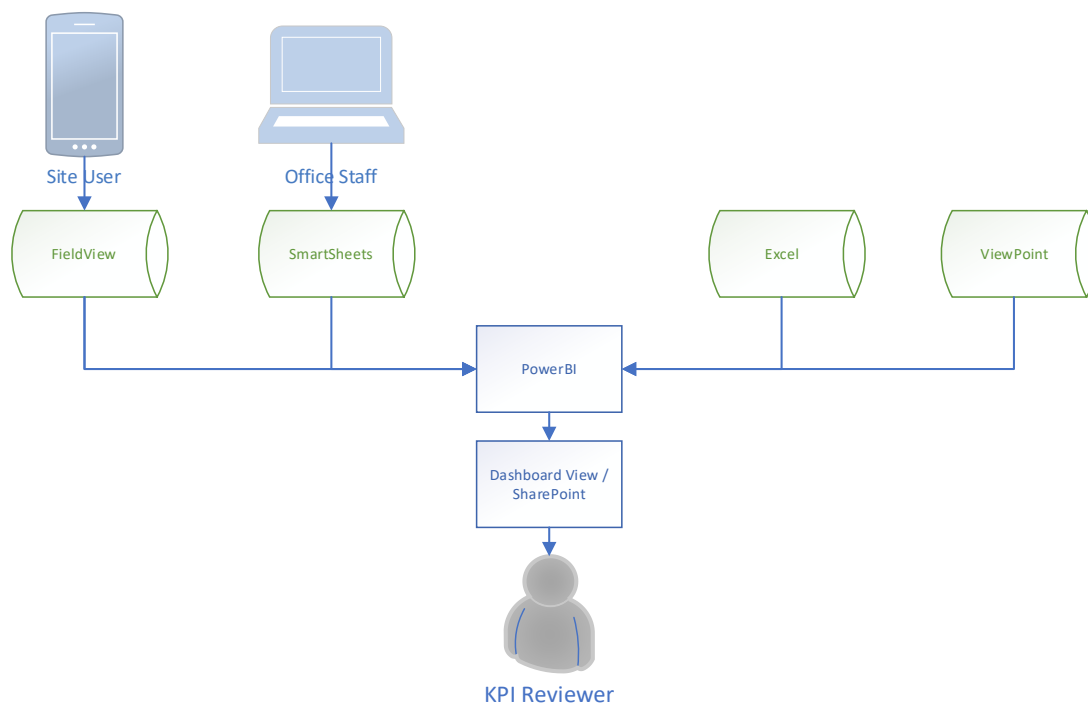


Figure 1: Overview of existing potential systems available to the project

The current systems identified for the initial project scope are:

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

4.4 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 for snagging tasks, forms & permits, project delivery and handover. When the mobile device is synced – data is pushed to the cloud hosted database.

← F215017.36 - Coins TimeSheet - 14/1/2023 All Good Distribute View Report

Owned By: Anders Ingelsten Location: MP-General Date Raised: 14/1/2023, 06:35

Copy Opened

Completed By *

Contract *

Date * 14/1/2023

Figure 2: Sample view of Fieldview mobile form

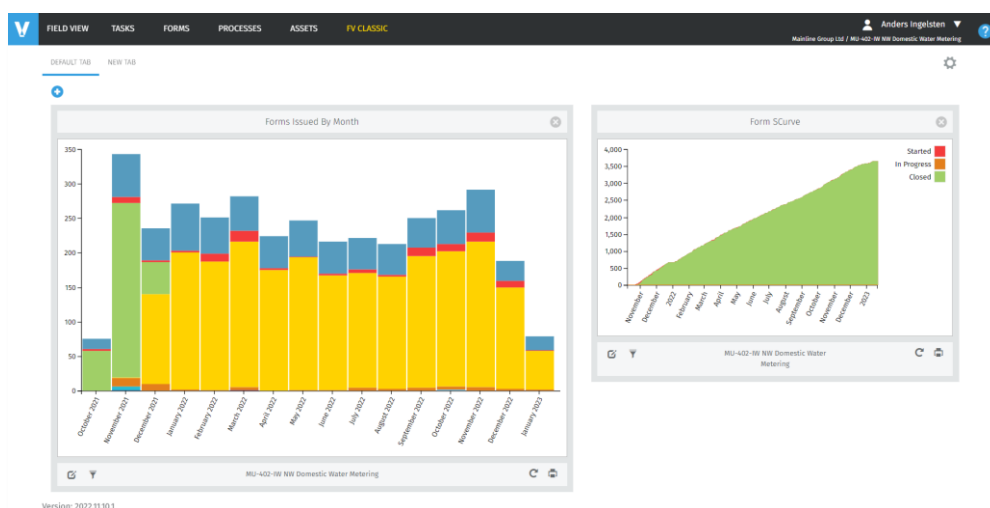


Figure 3: Sample view of the Fieldview widget reports

4.5 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.

Timesheet 4.0
Section and activity based form

DATE *
[Date Picker]

EMPLOYEE *
Select

CONTRACT *
MP-Overhead

SECTION_2
Operations

ACTIVITY_1
Default

CONTRACT HOURS *
0

COMMENT
[Text Area]

☐ Send me a copy of my responses

Submit

Figure 4: Sample view of Smartsheet form

4.6 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 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.

4.7 Current Situation

The company has, 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, 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 is aware that this is a time-consuming process and when it's well established is looking to improve the efficiencies in this process. The company has already identified that business intelligence tools like Power BI can be used to display almost real-time dashboard views of the business data. During 2022 there has been internal talks about identifying and bringing in business intelligence consultants/developers to do this work during 2023. Within the project, I will investigate business intelligence products for example the Microsoft Power BI platform. It's worth noting that the company has fully migrated to Office365/SharePoint during 2022 and is using the SharePoint platform as Intranet and repository for company data.

4.8 What are the problems.

The two main problems are:

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

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

5 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, for example a project manager, would like to access the online portal, view and pull-down information it can ONLY be done on a project-by-project basis. The systems do not have functionality in the user interface for a user to pull down, for example, all vehicle checks across all projects.

This means a user must access one project, access the report view, pull the information down and then access the next project and repeat the process until completed. Currently a user can create widgets that displays summaries but again, this must be done for every user on an individual basis.

A simple calculation will show how a dashboard can provide efficiencies.

Example, a user spends 1.5 hour every week to access every project area and compile a report of the amount of vehicle reports submitted by staff. $1.5 \text{ hours} \times 48 \text{ working weeks} = 72 \text{ Hours}$ which equates to 9 working days per year.

6 Potential Technologies, Tools, and Languages

As this is a work-based project the preferred direction from the leadership team is to use known technologies that are currently in use by the company. The two main reasons for this are that it creates greater resilience, and any costs are known, for example the company already host a Windows 2019 server 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

6.1 Potential Issues

As the project will have access to commercially sensitive and potentially personal data, measures will be put in place to minimise exposure of this data. This may limit the scope of which systems and data will be incorporated into the project. If it's not feasible to limit the exposure, test data will be utilised.

7 Objectives

This project has 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.

8 Methodology

8.1 Dataflow

The data is 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 Azure environment, pull 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.

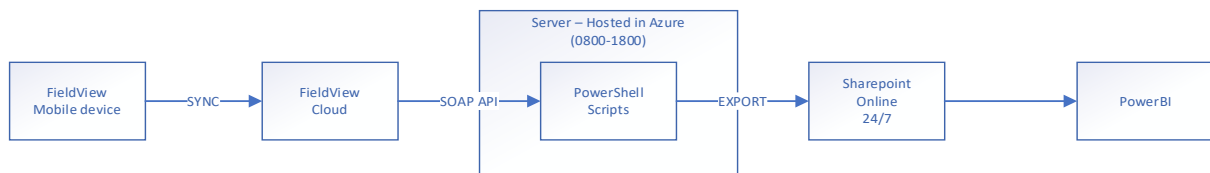


Figure 5: The Flow of Data from Fieldview to PowerBI

8.2 Modelling

Below is a simple representation of modelling of the data in Power BI, it is envisaged that data is 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.

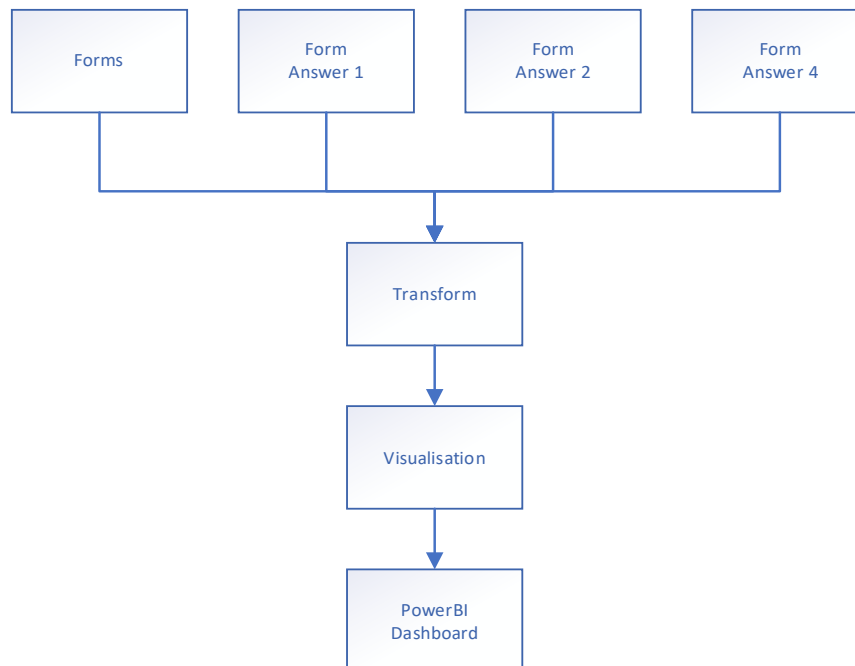


Figure 6: Sample Representation of the data model in PowerBI

8.3 Feedback Loop

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

The company is in a rapid growth spurt as several contracts wer won during 2022/2023 and a current ramp up of activities are in progress. The direction is to use a feedback loop during the phased development process. Envisaged steps 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 initial requirement.
3. Follow up meeting to determine if the result has fit the expected requirements.
4. Publish any changes or improvements of any feedback received.

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

8.4 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 can 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.

8.4.1 PowerBI

PowerBI will be 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.

8.4.2 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.

8.4.3 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.

8.4.4 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.).

8.4.5 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 an important 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.).

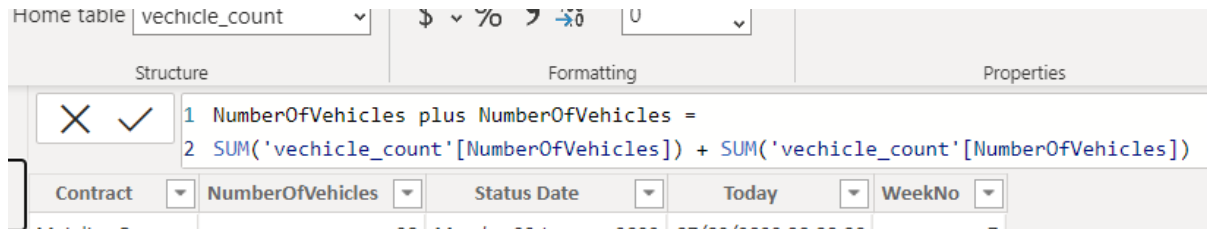


Figure 7: Sample View of DAX language in the formula bar of Power BI

9 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

The project intends to span over the below phased development activities with preliminary due dates. Note the dates and phases may be subject to change as purpose of phase and allocated time is 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

10 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 known as sprints into buckets. Inside the buckets the user would then add tasks into it.

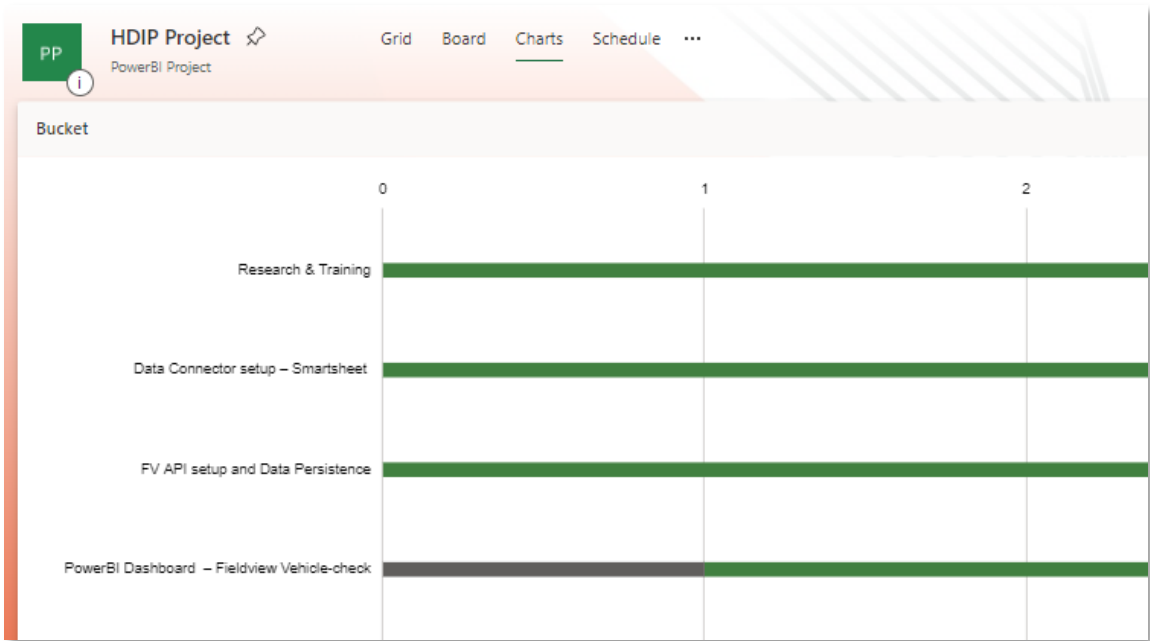


Figure 8: Microsoft Planner – Chart of Buckets

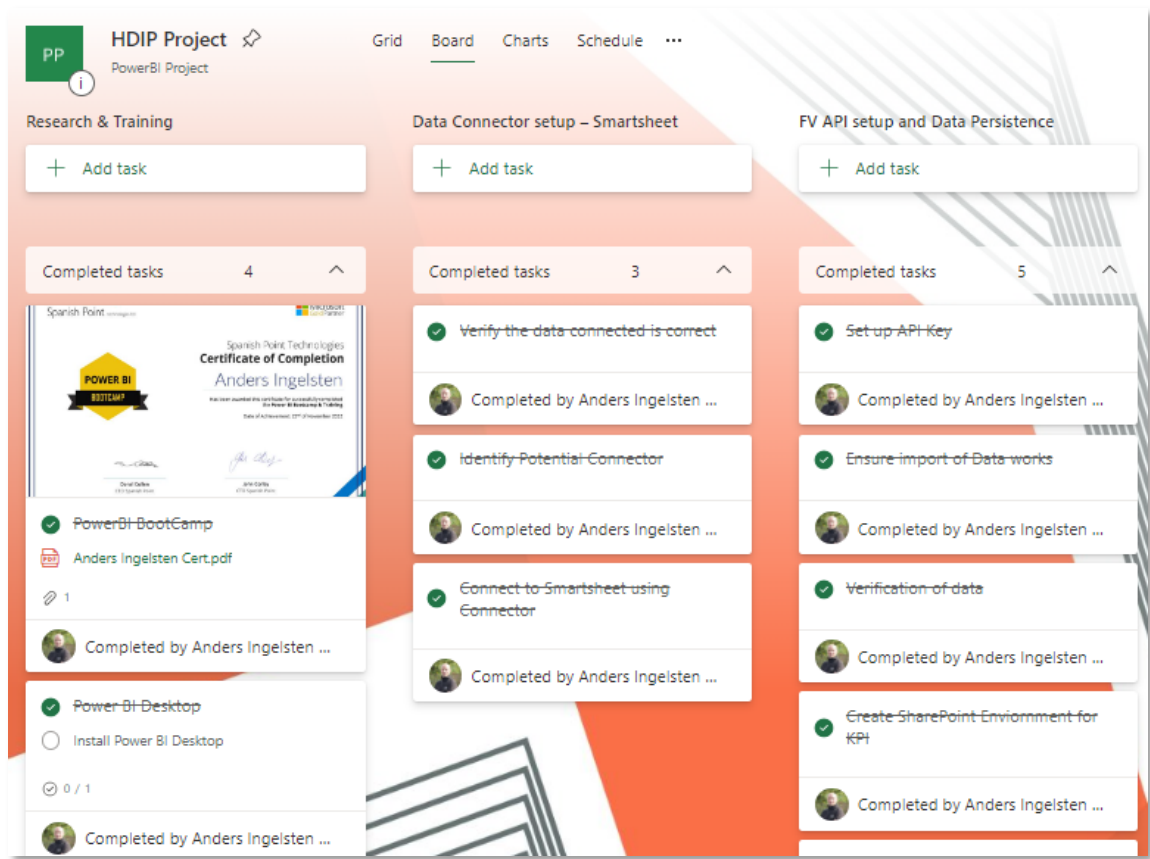


Figure 9: Microsoft Planner - Status of tasks and Buckets

11 Implementation

11.1 Phase 1: Research & Training

The phase of research and training covered three main 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.

1. UK
2. North America
3. 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

2 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 3 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 10: 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 11: 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 12: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 13: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).

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.). Company A has over the years used Spanish Point for various software solutions.

"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.

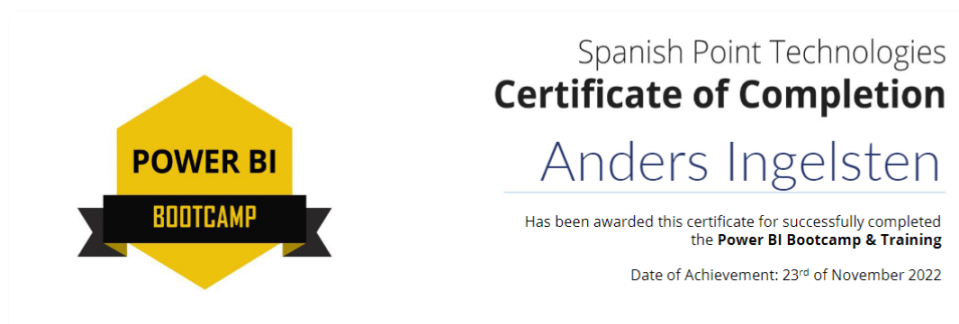


Figure 14: Certificate of Completion

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

11.2 Phase 2: Smartsheet Data Connector setup

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

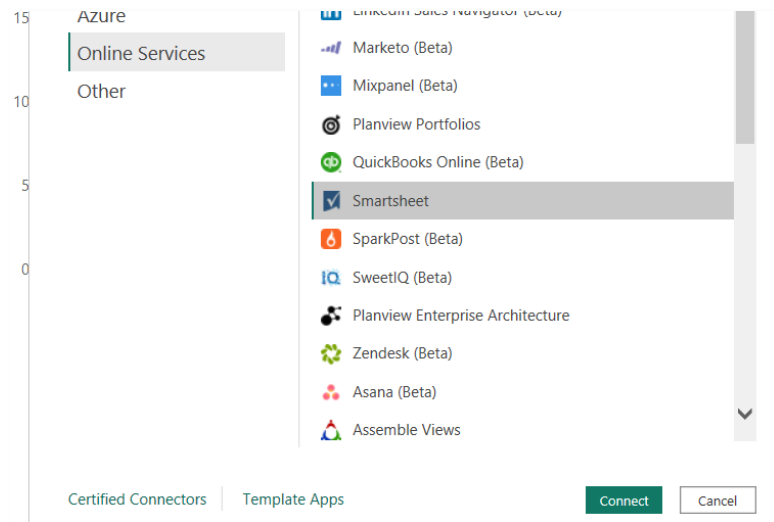


Figure 15: Get Data view in PowerBI of Online Services

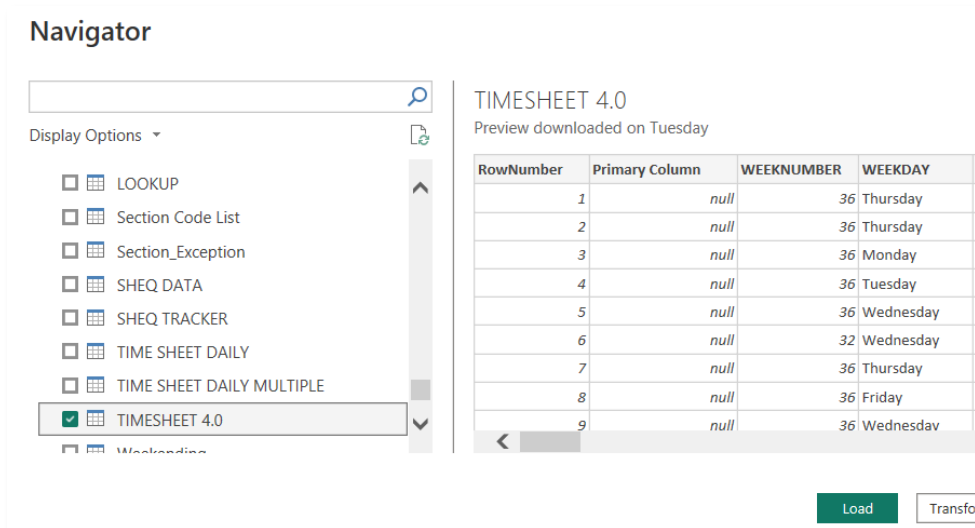


Figure 16: Preview of Connected Table ready to be loaded into PowerBI.

11.3 Phase 3: FV API setup and Data Persistence

The focus 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.

The first thing I did was to generate API keys inside Fieldview, 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 17: View of Created API Tokens in Fieldview

The SOAP protocol does not have to have any readily available plugins and add-on and 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. 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 18: 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.). Therefore I generated several API keys so I can 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.childnodes

    $formsList = $FVApiForms.GetProjectFormsList($apiToken, $projectId, $null, 0, $datefrom, $dateTo, $null, $null, $null, $null, $null, $null).ProjectFormsListInformation.childnodes

    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\jaingelsten\scripts\formslist.csv -append -NoTypeInfoInformation

        Write-Output "Data written to file"
    }

    Start-Sleep -Seconds 10
}
}
```

Figure 19: 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 : Mainline Power 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*****
22755

```

Figure 20: Screenshot of Sample Data from the `GetProjectFormsList()` call.

11.3.1 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 none of these are operational 24 hours a day, 7 days a week.

Increasing availability would lead to increased cost and for the data to be persistent it must write to non-volatile storage. Data persistence was achieved by saving the data to a file, after duplications were removed, it was then exported to Sharepoint Online (Rajack, 2018). By saving a file with the same name to the same location, SharePoint online just saves a new version of the file. A CSV file stored in SharePoint Online allows very easy connection to PowerBI by its built in Web connector.

```
Write-Output "Exporting to SharePoint"

#Configuration of Sharepoint Variables
$SiteURL = "https://typetecmg.sharepoint.com/sites/ITMainline"
$SourceFilePath = "c:\Users\angelsten\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 21: Sample Code of writing a file to Sharepoint Online

```
InformationRightsManagementSettings : Microsoft.SharePoint.Client.InformationRightsManagementFileSettings
IrmEnabled                          : False
Length                              : 0
Level                               : Published
LinkingUri                          : https://typetecmg.sharepoint.com/sites/ITMainline/Kpi_Data/formslist.csv?d=wfa70bd2094764f24956703ae372bd2a1
LinkingUrl                          : https://typetecmg.sharepoint.com/sites/ITMainline/Kpi_Data/formslist.csv?d=wfa70bd2094764f24956703ae372bd2a1
ListItemAllFields                   : Microsoft.SharePoint.Client.ListItem
LockedByUser                        : Microsoft.SharePoint.Client.User
MajorVersion                       : 1
MinorVersion                       : 0
ModifiedBy                          : Microsoft.SharePoint.Client.User
Name                                : formslist.csv
PageRenderType                     : Microsoft.SharePoint.Client.PropertyValues
Properties                          :
ServerRedirectedUrl                 :
ServerRelativePath                  :
ServerRelativeUrl                   : /sites/ITMainline/Kpi_Data/formslist.csv
SiteId                              :
TimeCreated                         : 22/01/2023 10:20:18
TimeLastModified                   : 22/01/2023 10:32:45
Title                              :
UIVersion                          : 512
UIVersionLabel                     : 1.0
UniqueId                           : fa70bd20-9476-4f24-9567-03ae372bd2a1
VersionIds                          :
```

Figure 22: Screenshot of successfully writing the data to Sharepoint Online in PowerShell

11.4 Phase 4: PowerBI Dashboard – Fieldview Vehicle-check

Phase 4 is the first phase that generated a Power BI dashboard result for the end company user. Employers who utilise company vehicles should have a process in place to conduct regular vehicle checks, to ensure that vehicles are in good order and safe to use (Health and Safety Authority, n.d.).

Company has a fleet of 44 vehicles and every user of a vehicle are required to do a weekly vehicle check. The requirement from the Plant Manager was to get a count of completed forms per week and division.

For this purpose I utilised the GetProjectFormsList API call, which gets all forms per project and timeframe. As company A have multiple projects in Fieldview I had to loop the API call through all Projects and also do an if statement as not all projects will have Fieldview forms in them during the timeframe requested

```
#Getting data by created date
$VApiForms.GetProjectFormsList($ApiToken, $projectId, $null, 0, $datefrom, $dateTo, $null, $null, $null, $null, $null, $null).ProjectFormListInformation.childnodes

$formList = $VApiForms.GetProjectFormsList($ApiToken, $projectId, $null, 0, $datefrom, $dateTo, $null, $null, $null, $null, $null, $null).ProjectFormListInformation.childnodes

#If statement for if data is null or not
if ($null -eq $formList)
{
    Write-Output "*****NOTHING FOUND*****"
}

else
{
    Write-Output "Adding ProjectId"

    #Adds the project id to form as an identifier
    $formList | Add-Member -MemberType NoteProperty -Name "ProjectId" -Value $projectId

    Write-Output "Writing to file"

    #Write result to file
    $formList | Export-Csv -Path c:\Users\aingelsten\scripts\All_Formslist.csv -append -NoTypeInformation

    Write-Output "Data written to file"
}
```

Figure 23: Code of File export Sample

Data generated via the API call is then appended to a CSV file, which is checked and cleaned for duplication and then exported to SharePoint Online.

```
#Export to Sharepoint
#Configuration of Sharepoint Variables
$SourceFilePath = "c:\Users\angelsten\scripts\All_Formslist.csv"
$DestinationPath = "Kpi_Data" #Site Relative Path of the Library
$ClientId = "[REDACTED]"
$ClientSecret = "T[REDACTED]g="

#Site collection URL
$SiteURL = "https://t[REDACTED].sharepoint.com/sites/[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
```

Figure 24: Export to SharePoint with automated details

I added the SharePoint data into PowerBI by using the Get Data Web function which takes the SharePoint URL of the CSV file. When the file is updated in SharePoint I can refresh the data in PowerBI by utilising the refresh function.

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
21206	F1.1887438	12031524	FALSE	Administration	Company Vehicle Check List	Ford Ranger Wildt	2023-02-08T14:31:0
21549	F1.1899188	14789567	FALSE	Operations	Pre Site Visit Report		2023-02-13T14:54:1

Figure 25: Form data as exported from FV

I analysed the imported data and to create the desired out come 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 26: DAX code of Count

I then used the visualization tools in PowerBI to create the following Report Dashboard, count of active vehicles and completed vehicle check list forms, and an additional view that shows the count of check lists per week in a table diagram.

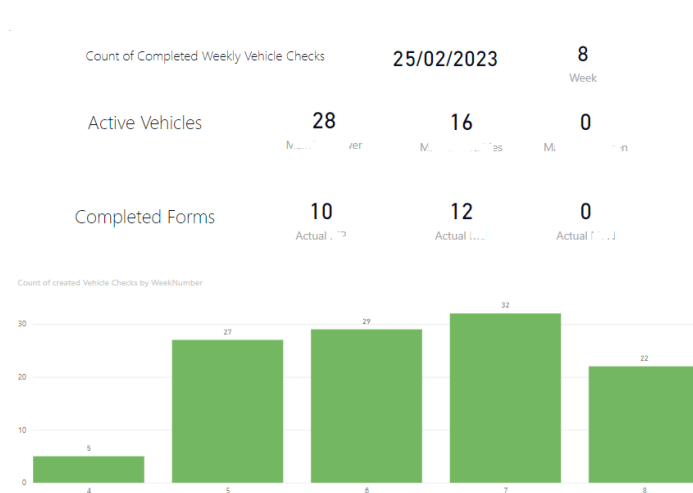


Figure 27: View of Counts

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.

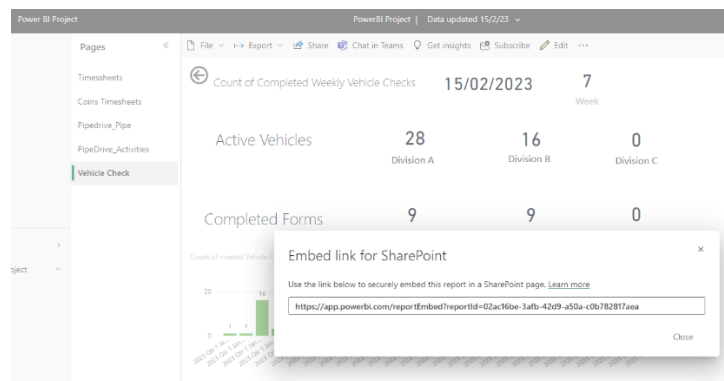


Figure 28: View of Embed Link

- Or to generate Embedded Html code, so it can be published to multi users.

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.

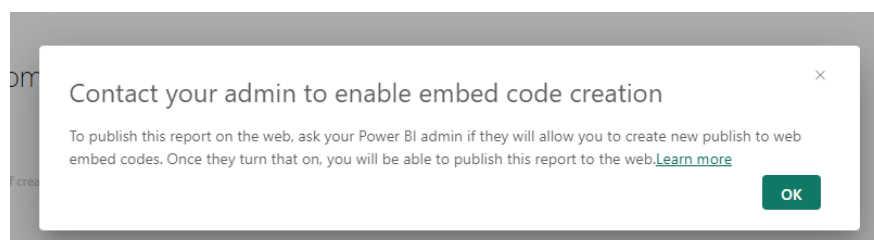


Figure 29: View of admin message

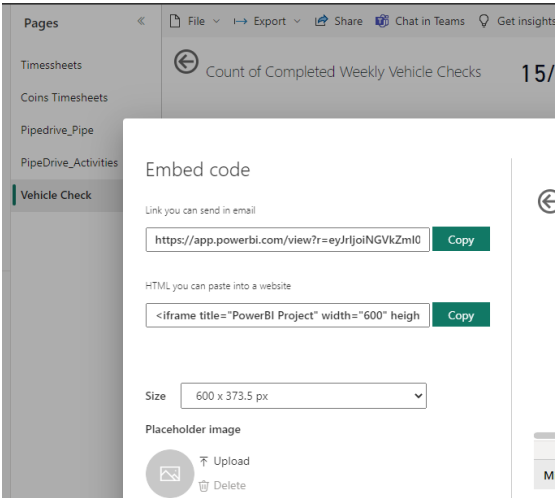


Figure 30: View of embed code view

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

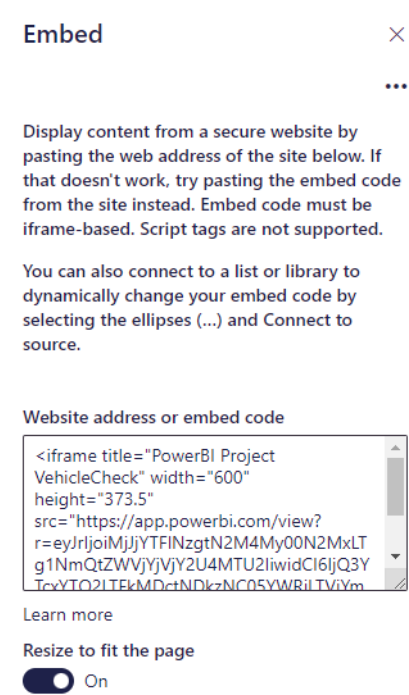


Figure 31: View of Sharepoint Online Embed function

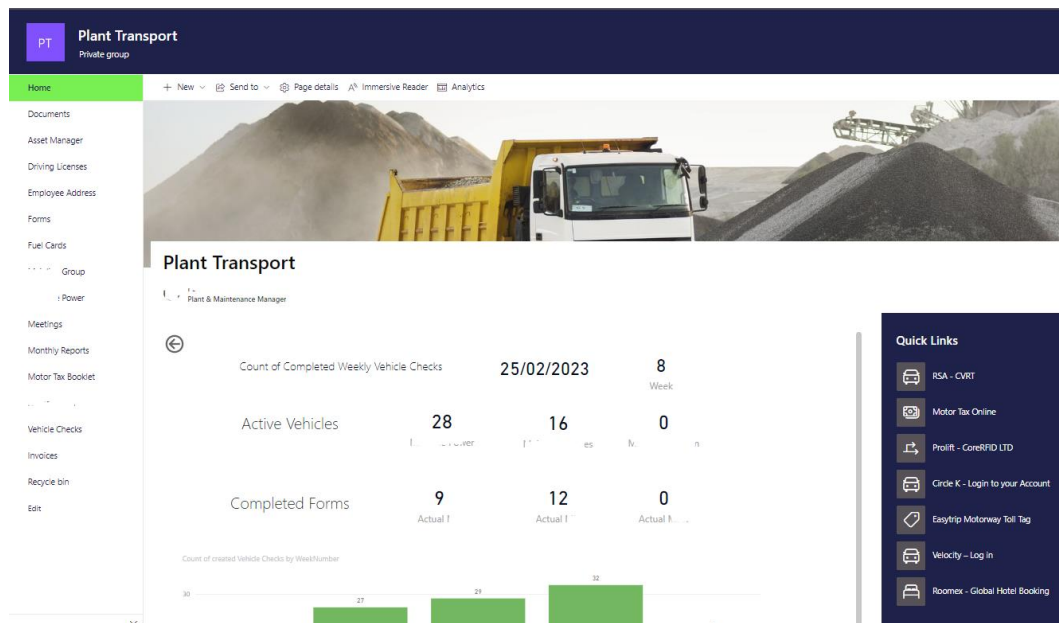


Figure 32: View of Sharepoint Intranet

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

11.5 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.

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.

```
# 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\aingelsten\scripts\Deals_pipe.csv -NoTypeInformation -Append
        Write-Output "****EXPORT****"
        Start-Sleep -Seconds 0.5
    }
    else {
        Write-Output "****LOOP ENDS****"
        Break
    }
}
}
```

Figure 33: Code snippet of Pipedrive script

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

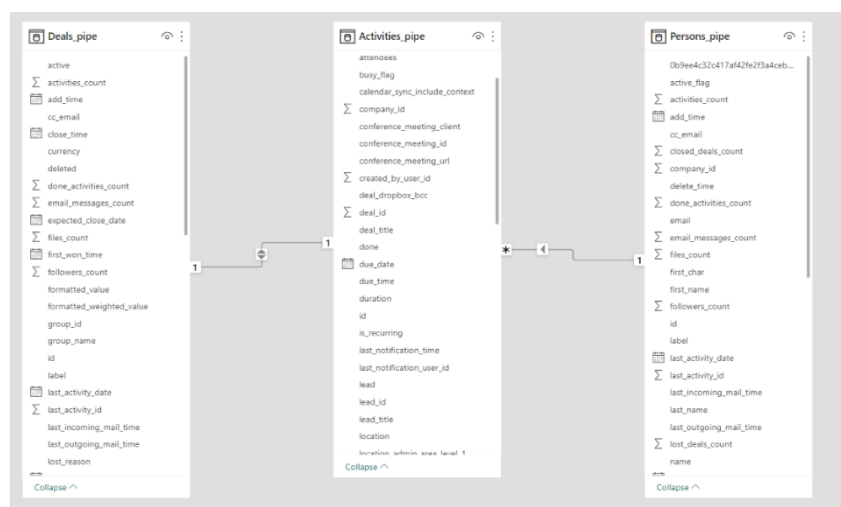


Figure 34: View of table relationships

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.

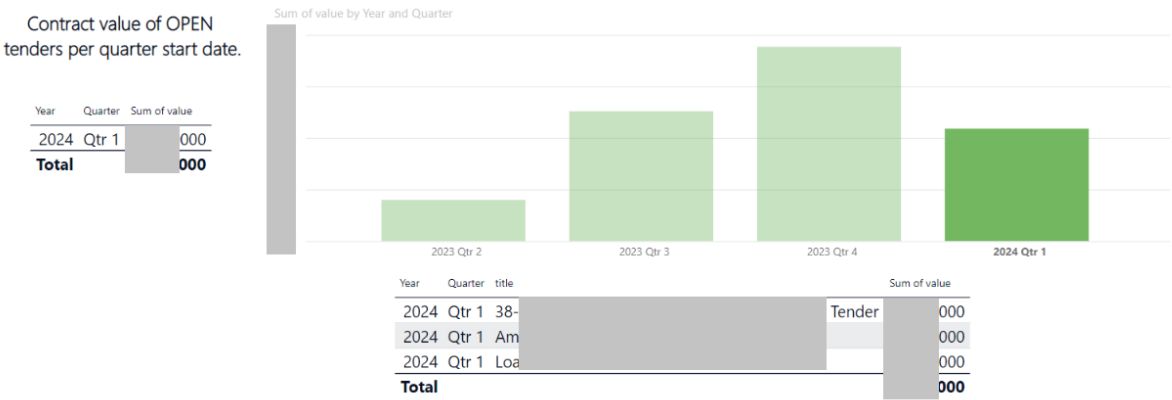


Figure 35: View of Values per Quarter

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

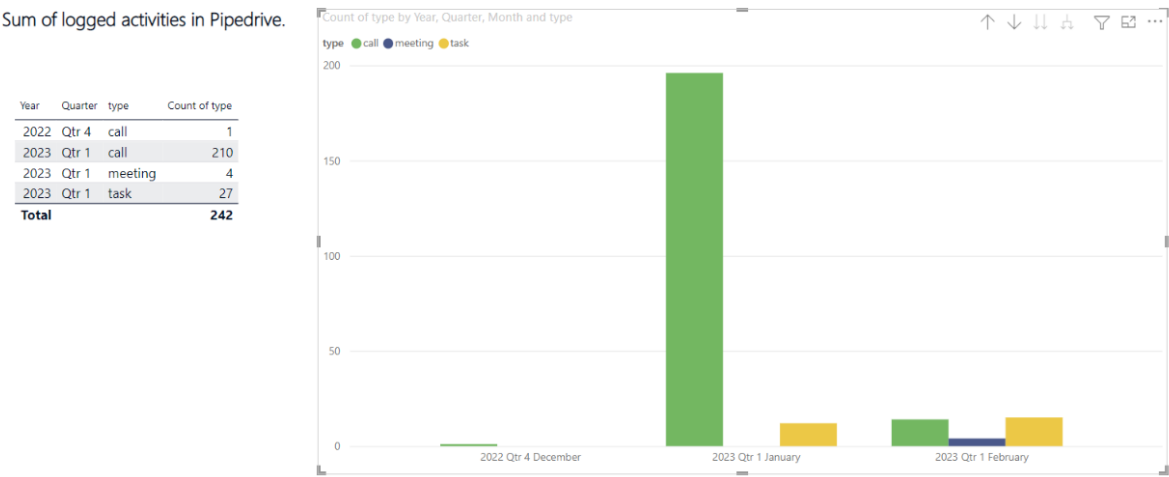


Figure 36: Report view of logged activities

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

11.6 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 could import 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		44	5	Tuesday 3 January 2023	on	9	L22
2	Tuesday	M		41	5	Tuesday 3 January 2023	on	10	L22
2	Wednesday	M		41	5	Wednesday 4 January 2023	on	9	L22
2	Tuesday	P		8	5	Tuesday 3 January 2023	on	2.5	L36
2	Thursday	C		6	5	Thursday 5 January 2023	on	8.5	L36
2	Thursday	M		41	5	Thursday 5 January 2023	on	9	L22
2	Friday	M		41	5	Friday 6 January 2023	on	8.5	L22
2	Wednesday	M		44	5	Wednesday 4 January 2023	on	9	L22
2	Thursday	M		44	5	Thursday 5 January 2023	on	6	L22
2	Thursday	M		44	5	Thursday 5 January 2023	on	4	L22

Figure 37: 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



Figure 38: 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.

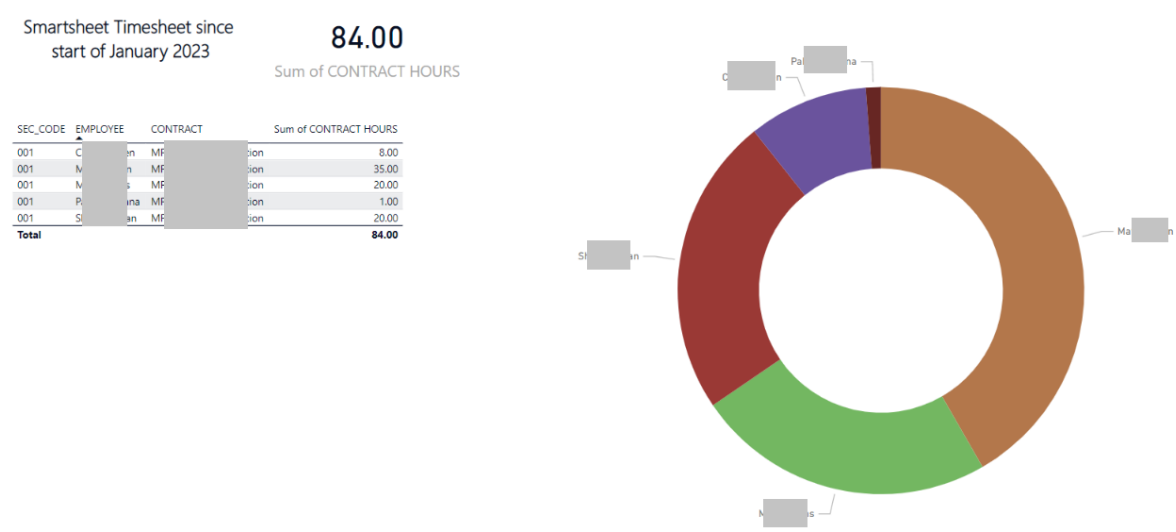


Figure 39: Drill down view Timesheet

11.7 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.asmx?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 40: 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 41: 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-PnPOne -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

Write-Output "Process Answers Completed"

```

Figure 42: Code snippet of export to SharePoint

After the 4 answers from Fieldview had been imported into PowerBI, all the 4 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. The query generated an answer for every form that has a question alias with the word "date". Note there are currently over 120 active forms in the Fieldview system. 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.

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

Figure 43: Query view in PowerBI

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

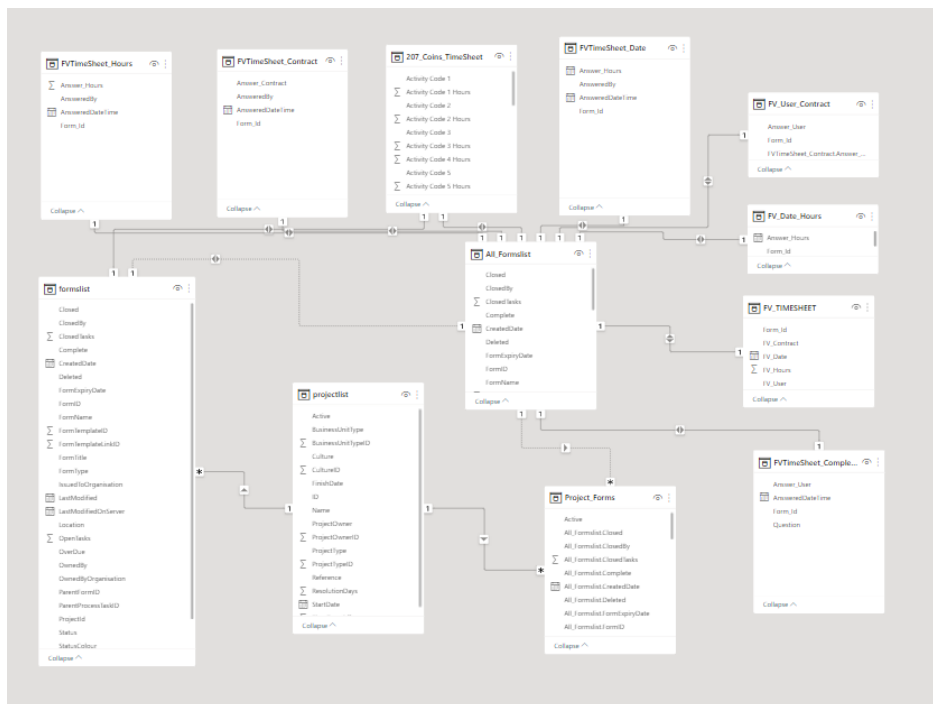


Figure 44: Table relationships in PowerBI

The visualisation requested is 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 allow for drill down per person.

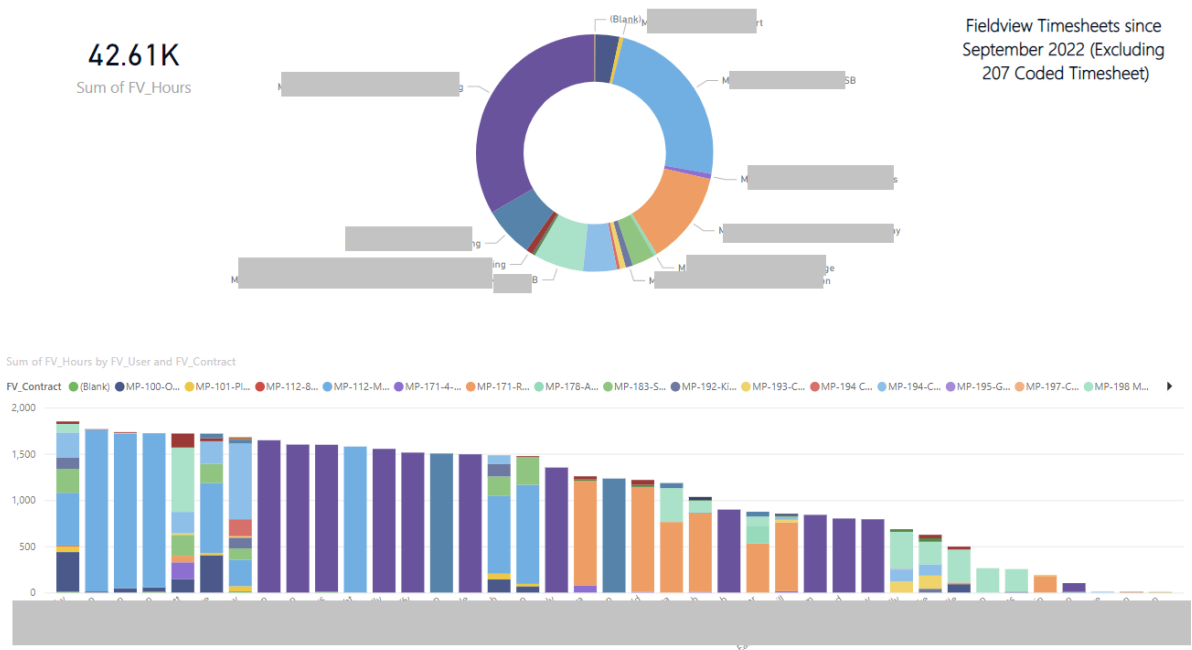


Figure 45: Summary visualisation of the hours per contract and user

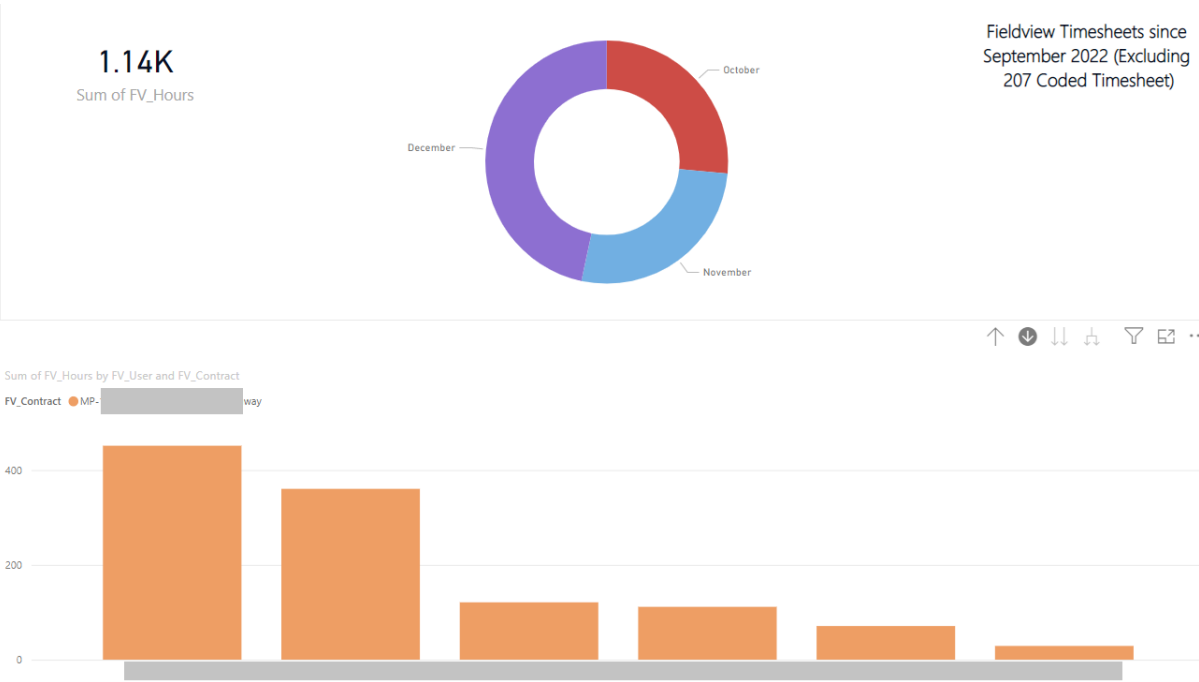


Figure 46: Drill down view per contract, month, and person.

11.8 Phase 8: PowerBI Dashboard – Asset Manager

This phase was added as I was asked by the Company's project supervisor 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 more easily access upcoming service and certifications renewal dates as part of his KPI requirements.

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 I found a software company called Devart who 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.

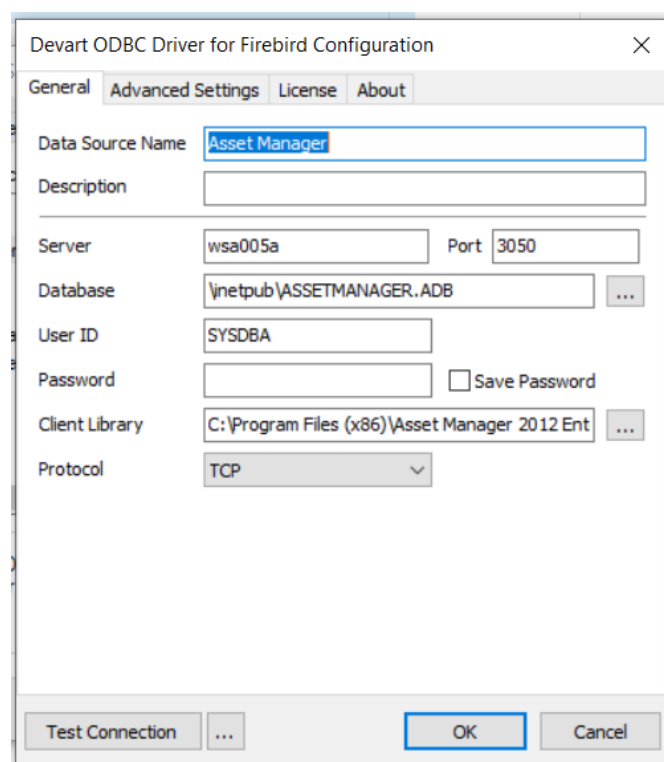


Figure 47: View of Devart Configuration

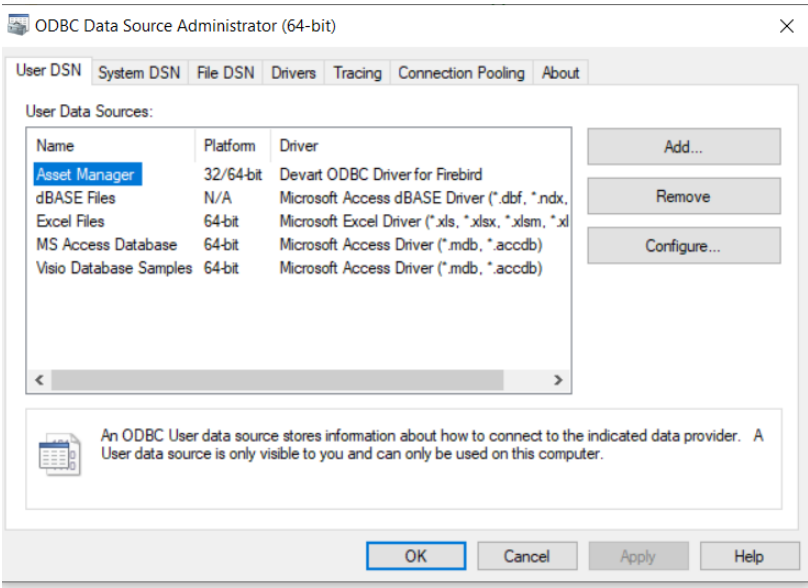


Figure 48:View of connected Data Source

Navigator

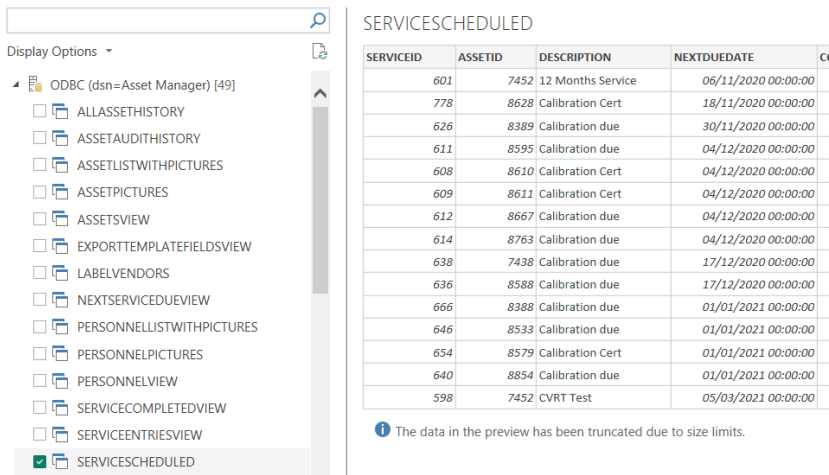


Figure 49: View of Tables available from Asset Manager.

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.

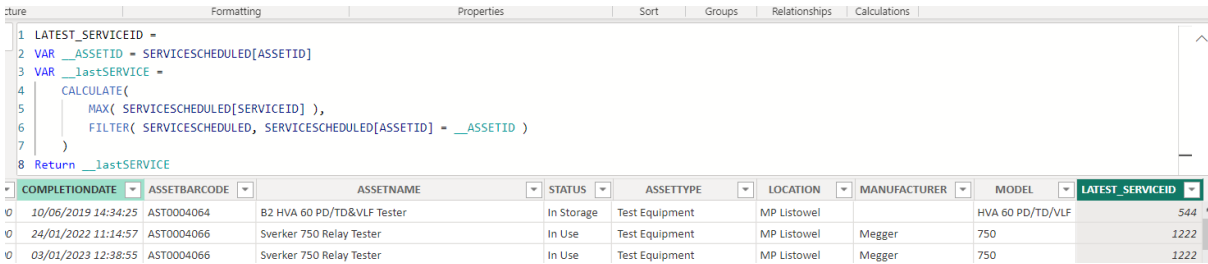


Figure 50: Query of the latest service ID

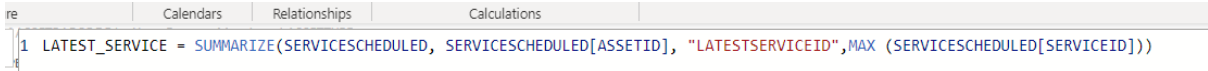


Figure 51:Table query of the latest service ID

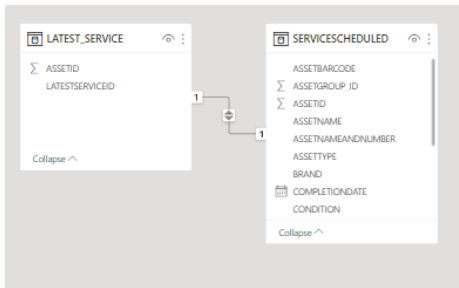


Figure 52:View of the joined table relationship.

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

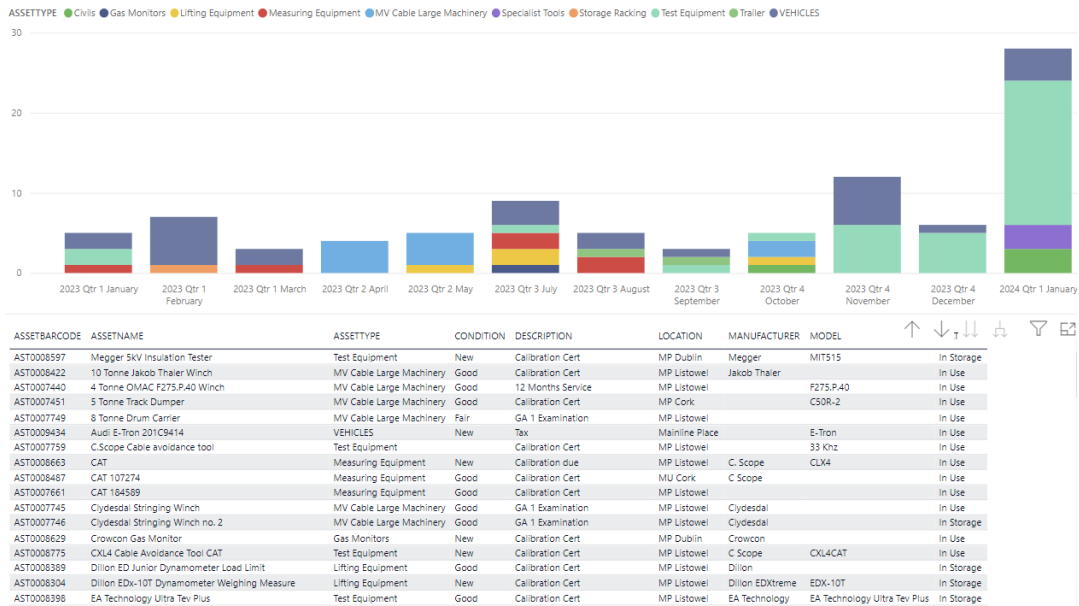


Figure 53:View of asset types and corresponding table.

11.9 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. For this I decided to use two separate tools, the Task Scheduler and Power Automate.

11.9.1 TaskScheduler - 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 54: 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 was found from the Spiceworks website (Inc, n.d.).

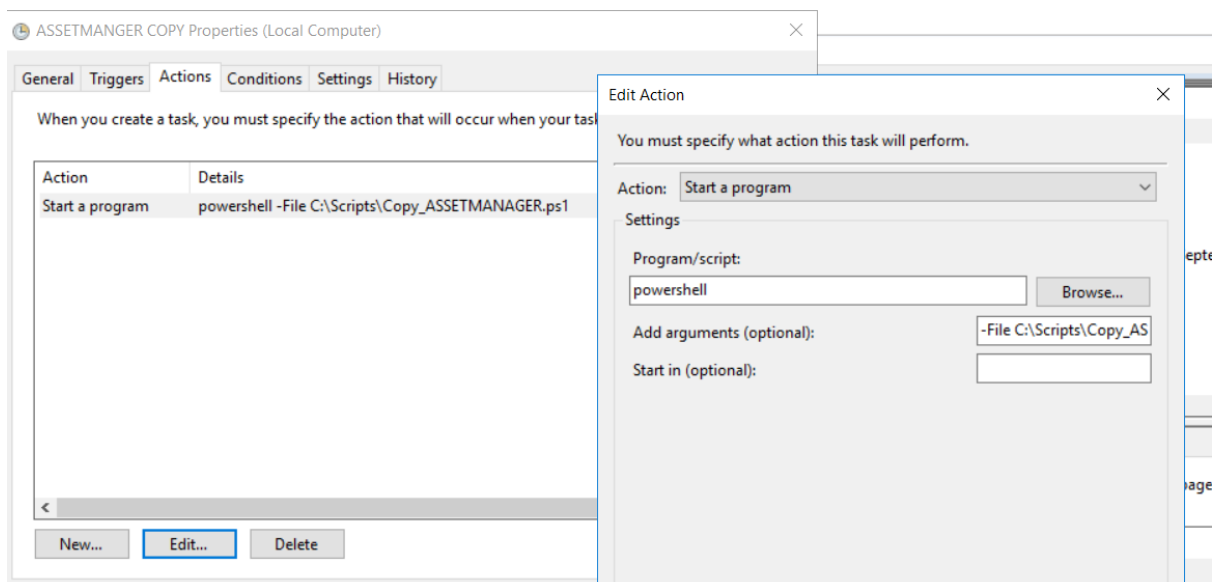


Figure 55: View of Task Scheduler setup to run script every 15 min

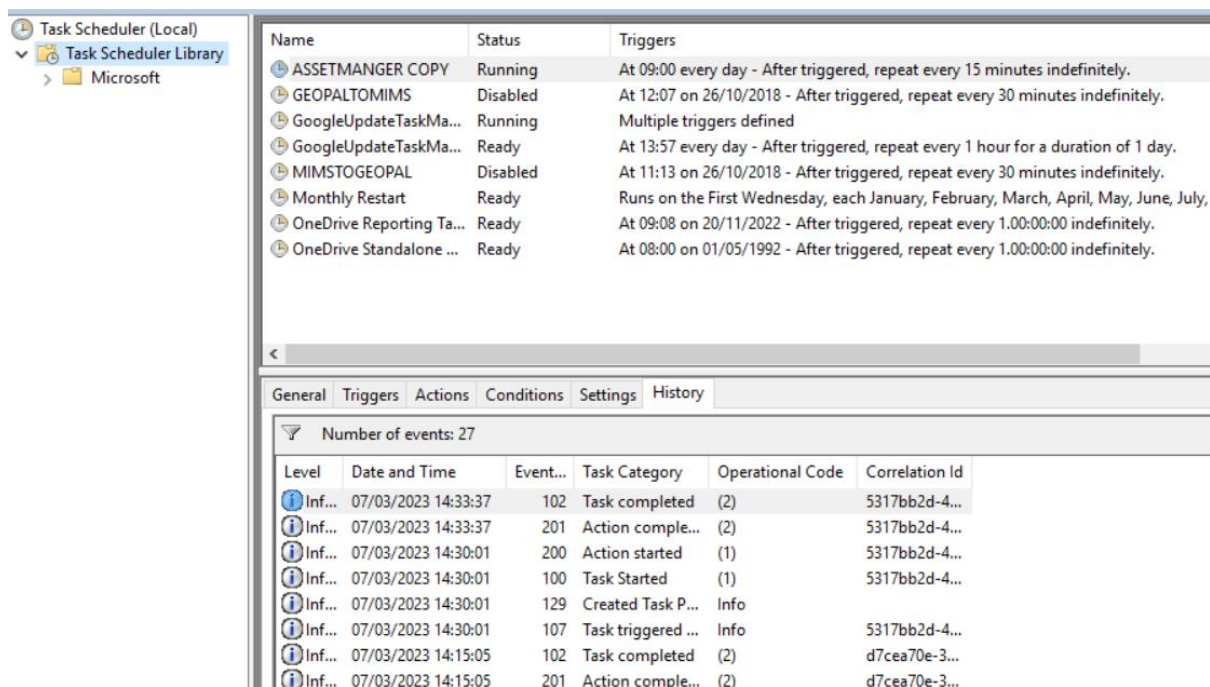


Figure 56: View of Scheduled copy task in situ with running history

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

11.9.2 PowerAutomate – Domain Controller Server

I then moved on to automate the PipeDrive and Fieldview API scripts. For this I decided to use Power Automate and to use the Domain Controller server, as it is on 24/7. 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 being updated locally but the data would not update in SharePoint. The error log showed the following “error Connect-PnPOnline : The term 'Connect-PnPOnline' is 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.

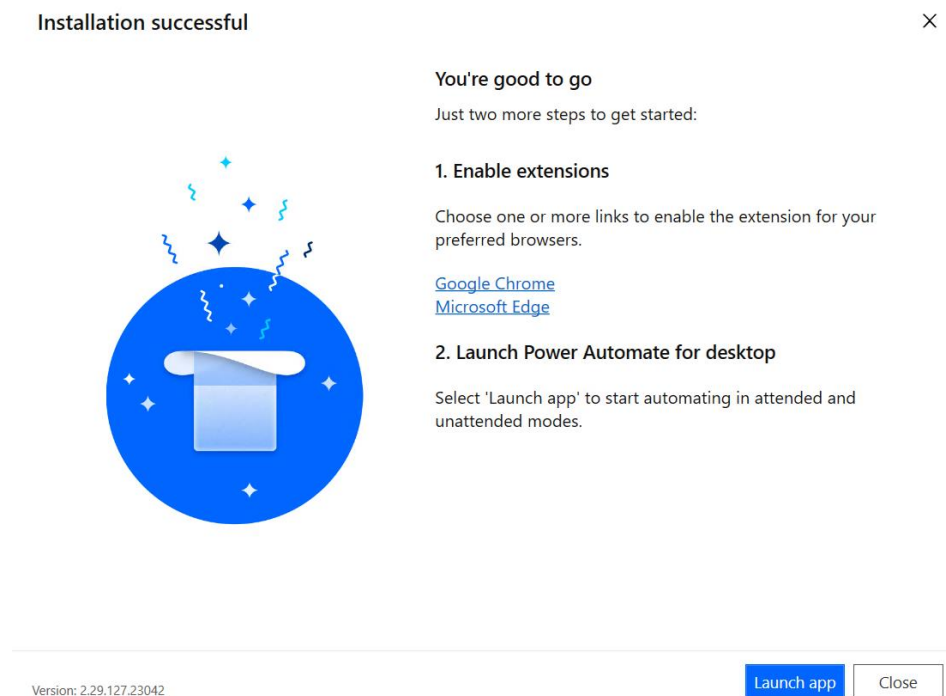


Figure 57: Installation of Power Automate desktop on domain controller server

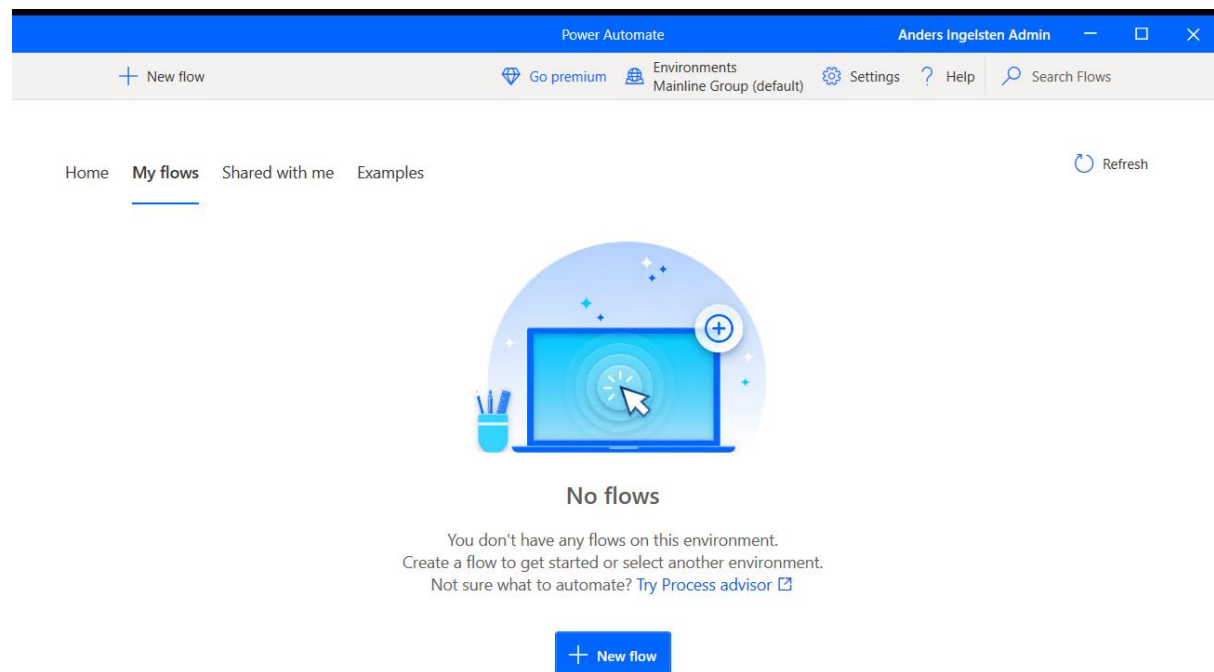


Figure 58: Before implementation of flows

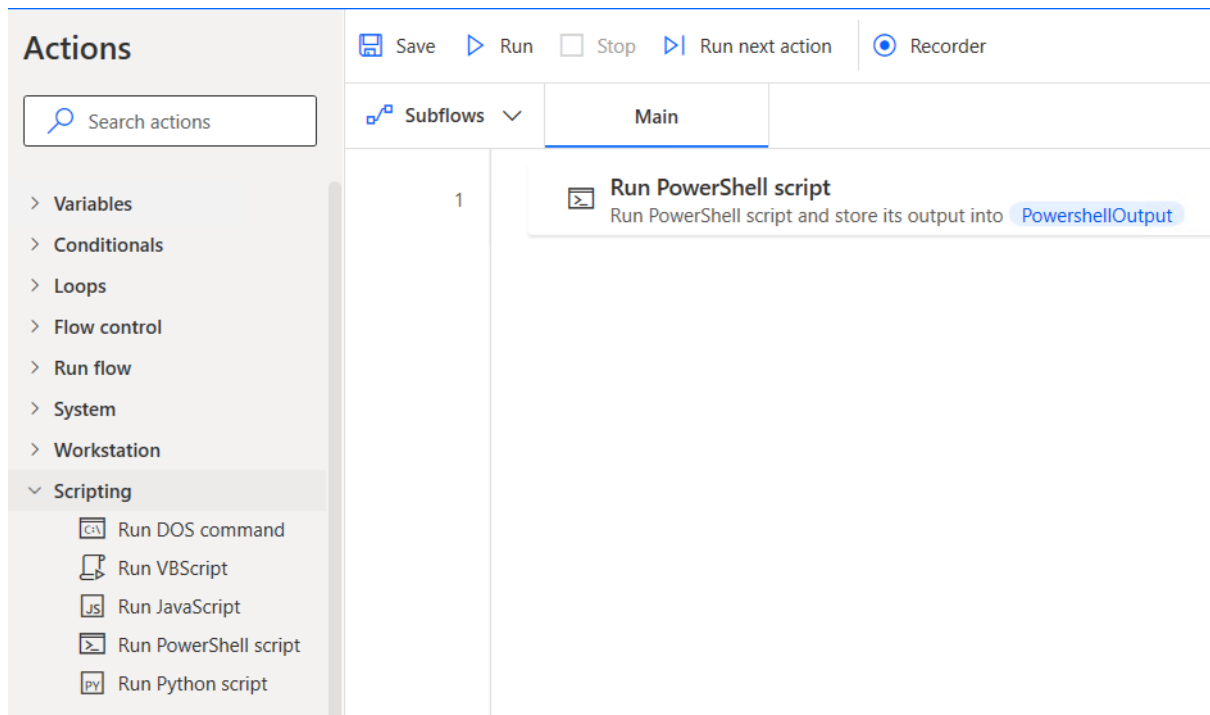


Figure 59: Creating the Run PowerShell Script on Power Automate Desktop

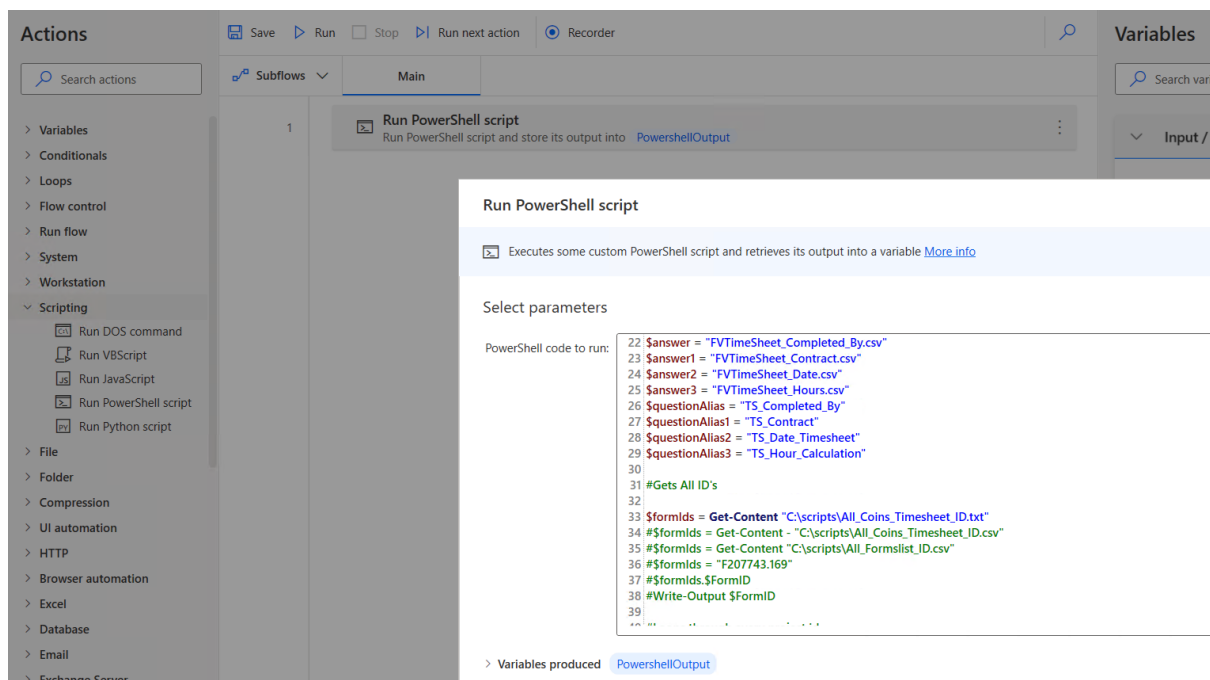


Figure 60: Adding the script to run in the Flow

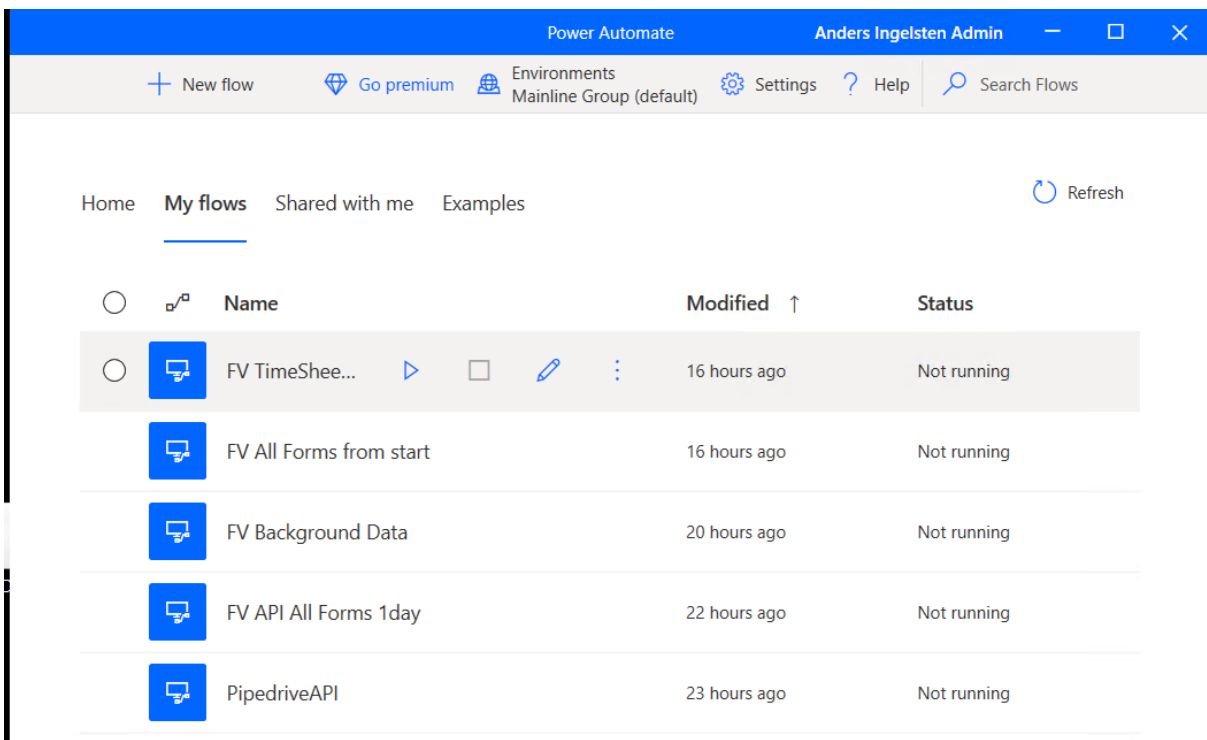


Figure 61: View of all the 5 PowerShell scripts that was setup on the Power Automate desktop

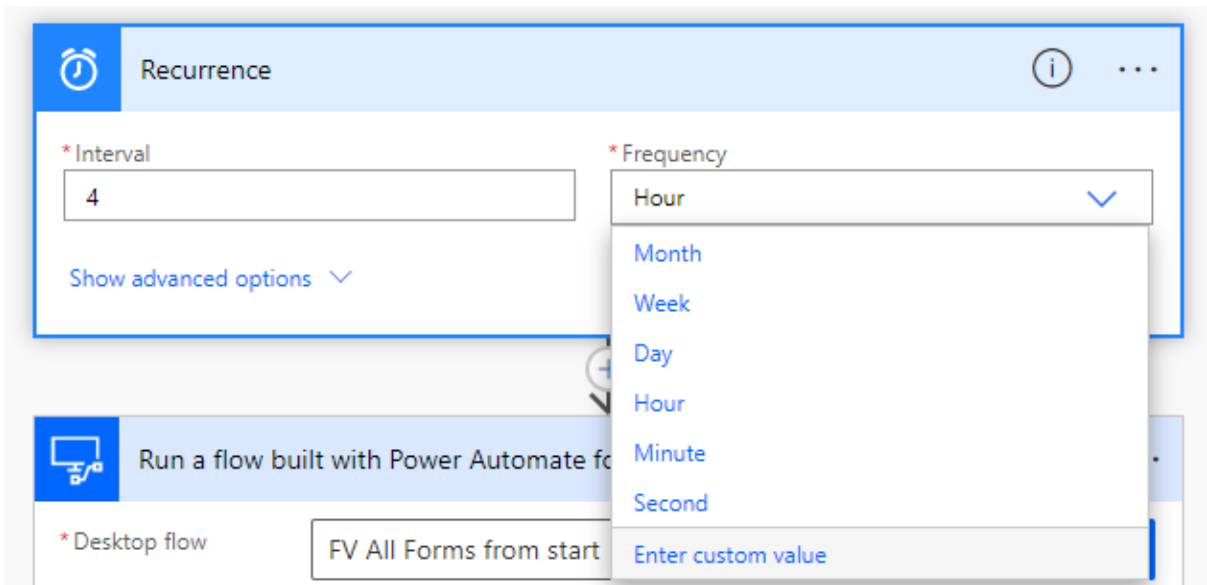


Figure 62: Recurrence step on the Power Automate Cloud

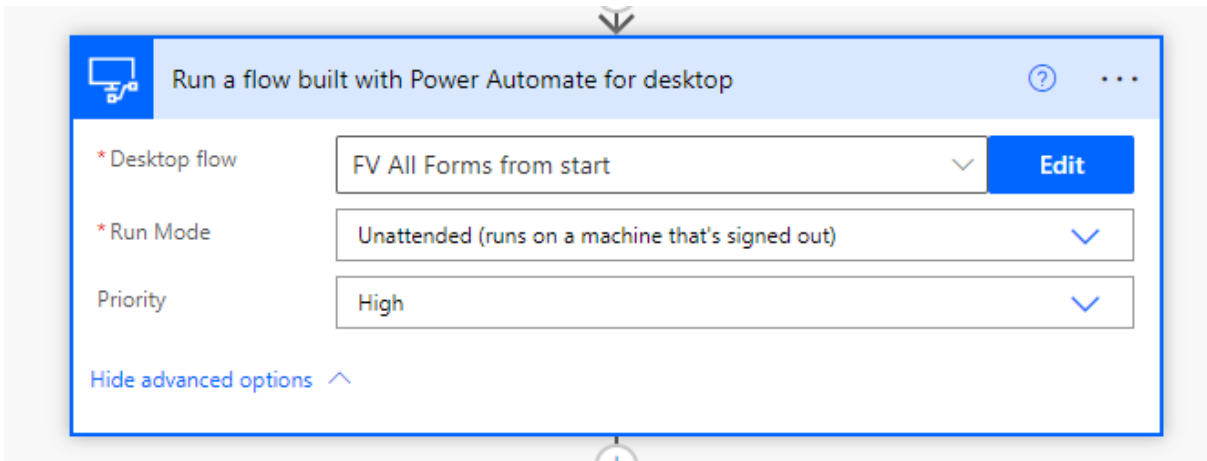


Figure 63: Running the Power Automate desktop flow on the Power Automate Cloud

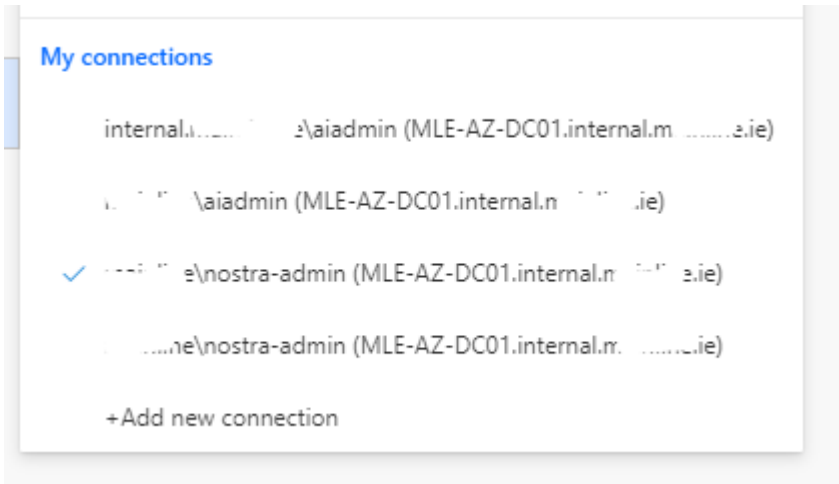


Figure 64: Connection used from Cloud to Hosted Servers to run the Powershell scripts

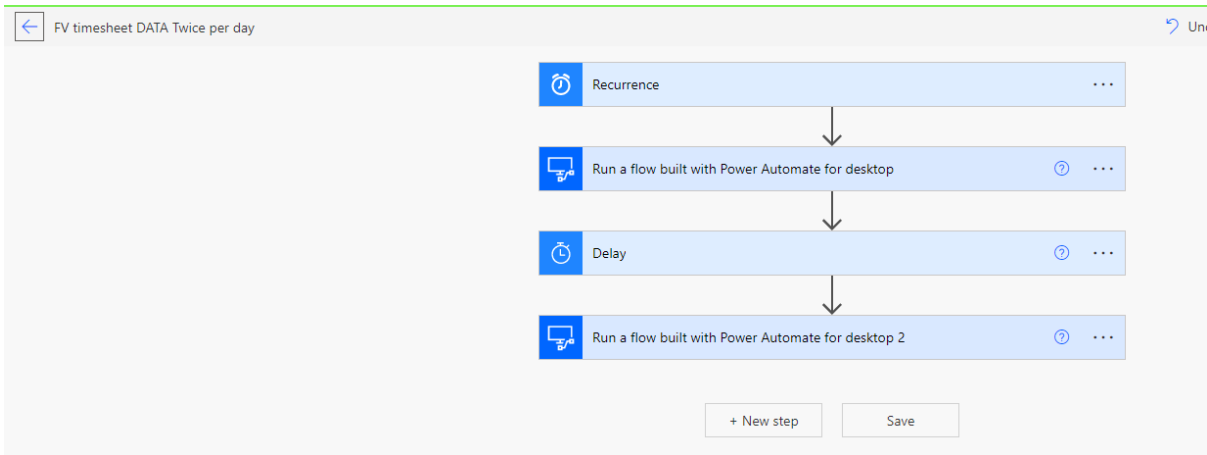


Figure 65: Sample of a flow -this running multiple scripts

Flows

Cloud flows Desktop flows Shared with me










	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

11.9.3 Power Automate – Domain Server - Reverting to Task Scheduler

After I completed the flows, I started generating some test data and monitoring if real 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 had not been updated and that all flows were failing. By reviewing the fault message in Power Automate the errors were due to a user who 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 instead as it will run regardless of user is on 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
Mar 8, 11:15 AM (22 min ago)	00:00:13	Failed

Figure 67: Inital 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

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 of several failed flows

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

[Resubmit](#) [Cancel](#) [Edit](#) [Help](#)

Flow run failed.

Recurrence

0s

Run a flow built with Power Automate for desktop

13s

Error Details

Start time

Mar 9, 07:59 AM (6 min ago)

Duration

00:00:13

Error

Action

Run a flow built with Power Automate for desktop failed

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-DC01.internal.mainline.ie)

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

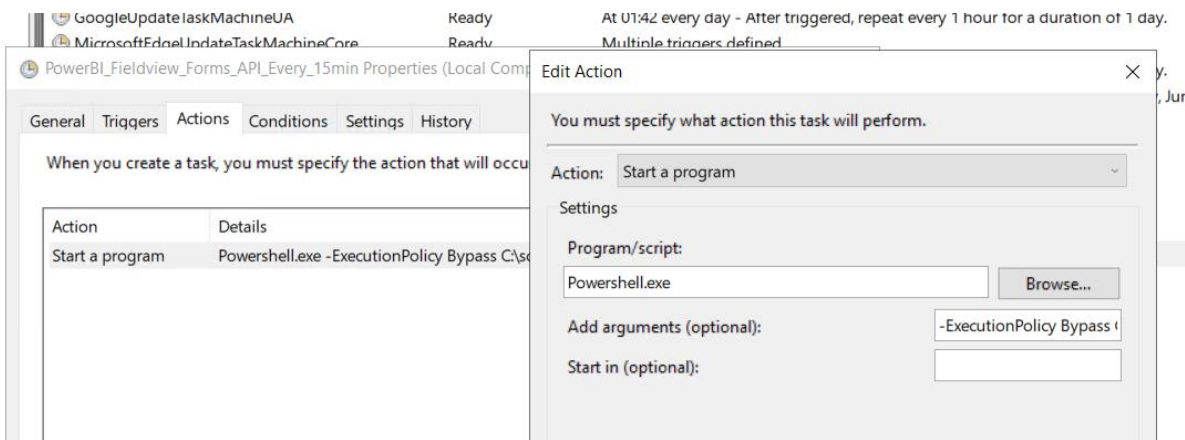


Figure 73: Setting up the Task Scheduler

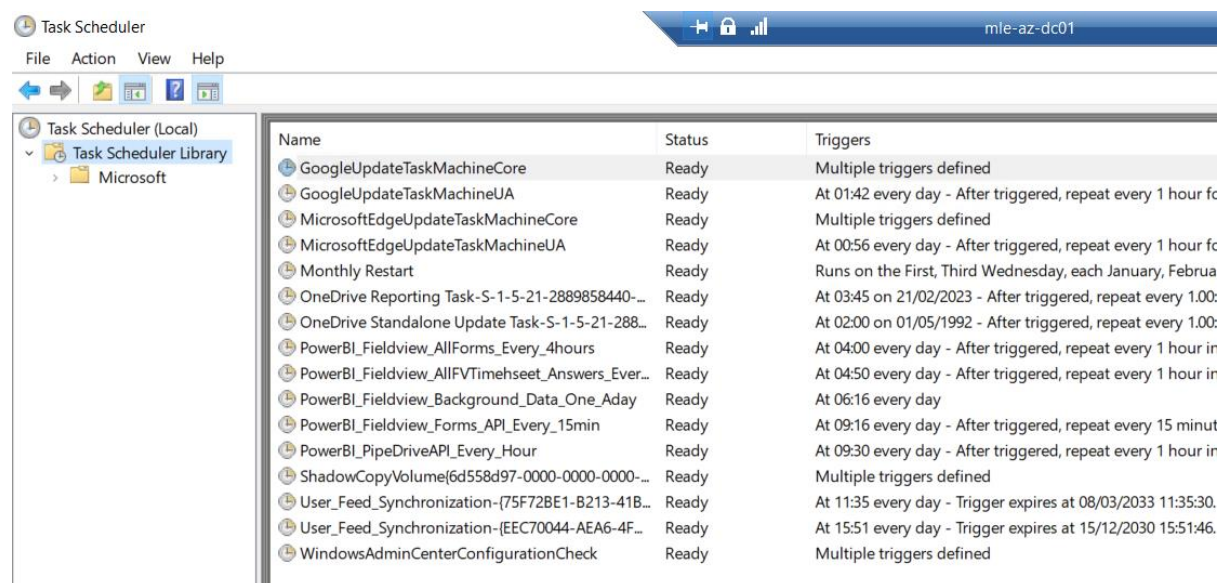
I decided to have individual tasks per script so I if/when I need to updated/fix a script I wont impact the rest of the updates. To speed up this process I used the export and import function available in Task Scheduler. The task is export to XML and this is also the format for importing the task then I updated the schedule and arguments i.e. which 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 TaskScheduler

[scripting - How to use a config file \(ini, conf,...\) with a PowerShell Script? - Server Fault](#)

11.9.4 PowerBI – Automated Refresh and Publish

12 Project Evaluation

12.1 Self-Reflection

The initial showing of the iteration of a published dashboard of real company timesheet data to the company's CFO went very well. The eyes lit up, the penny dropped, the feedback was immensely positive and new requirements were expressed. *"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 emotions 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 learned the most and where the acquired knowledge from the course assisted me in the best practise.

Furthermore, the expression "garbage in, garbage out" was very 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 or "deleted" data and how to apply the appropriate business logic. It's worth noting that in Fieldview you cannot delete data, it is only labelled deleted with a flag of True or False.

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

12.2 Knowledge attained.

12.3 Achievements

12.4 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 Data verse business layer

12.5 Future Work and Development

In relation to future work for the project, I have identified two very clear directions:

The first one is to gather more data and publishing more business data analytics for more 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 I'm paraphrasing: *"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 being in the utilities/construction business, it would be of great 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 Table on View Point 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"

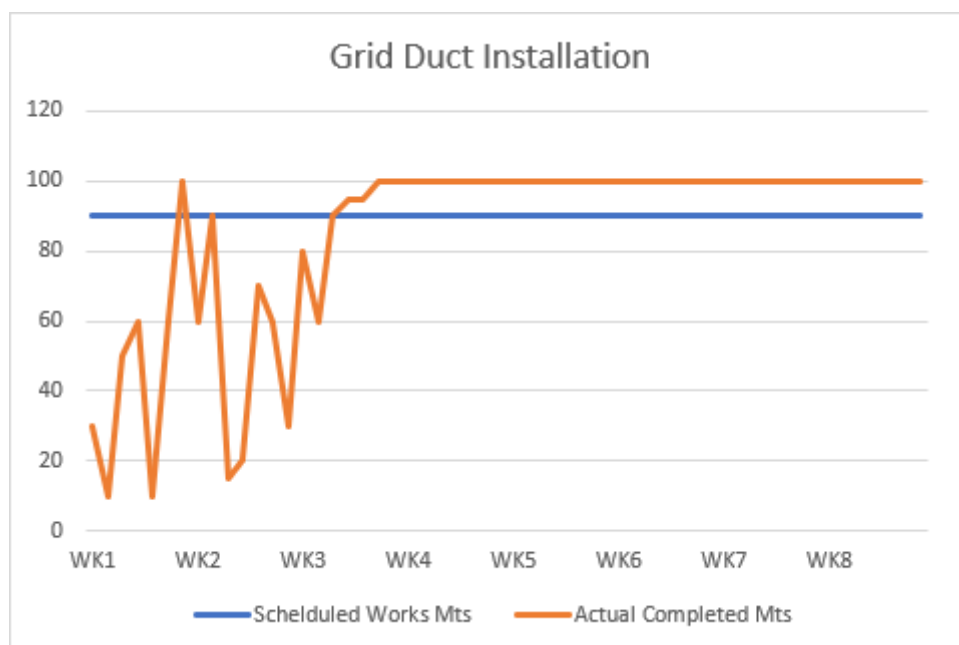


Figure 76: Sample of requested view 1

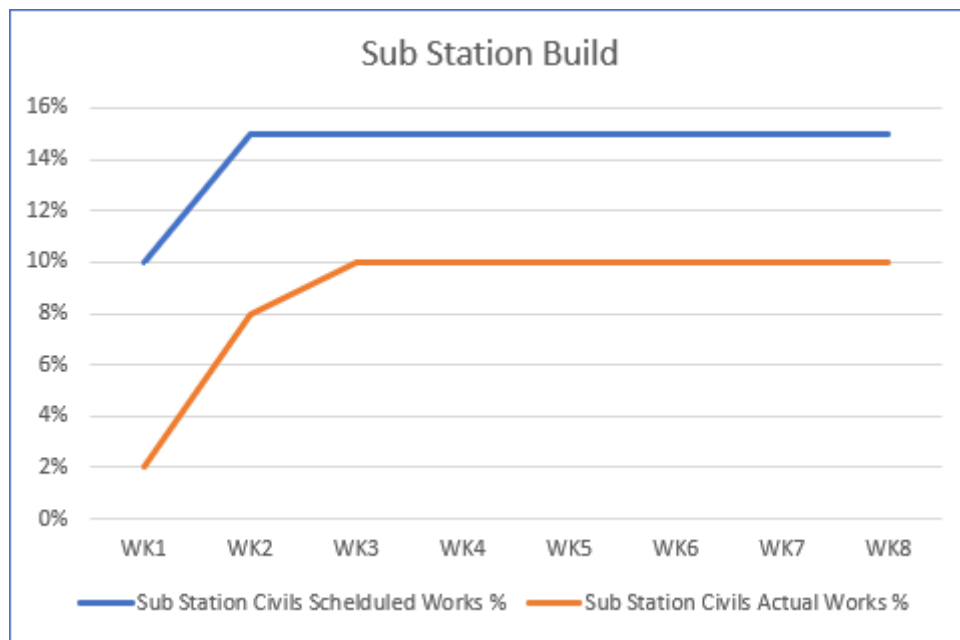


Figure 77: Sample of requested view 2

Secondly, during the project, I have identified three areas of future development from a technical point of view, and these are areas that will be continuously developed.

- Development in relation to business logic and rules around them, for example just simple changes or updates when it comes to automation and updating periods.
- Saving data as CSV files SharePoint online might be a solution for the immediate future in relation to Data persistence. Overtime other solutions will be looked at, for example SharePoint Lists, MS Dataverse and Azure SQL Database.
- Further additions of features, in relation to data queries of existing systems or systems that have not yet has been logged in to the power BI platform. For example, the ERP system and the HR management system.

13 References

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

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>.

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

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

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/>.

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

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

georgiostrantzias (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>.

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

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>.

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

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>.

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

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>.

www.youtube.com. (n.d.). Power Automate Desktop | | How to schedule Desktop Flows? [online] Available at: <https://www.youtube.com/watch?v=IJ112737JWU>.

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

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

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].

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].

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].

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].

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].

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

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].

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].

