Geert Hoogerland

Documentatie en beheerplan

BITT

Business Intelligence Track and Trace

Inhoud

[**Beschrijving BITT** 2](#_Toc144115381)

[Het doel van het project 2](#_Toc144115382)

[**Werking** 3](#_Toc144115383)

[Stap 1: Het extraheren van gegevens 3](#_Toc144115384)

[Stap 2: Het invoegen en verwerken in een MSSQL-database. 4](#_Toc144115385)

[Stap 3: Het plaatsen van de informatie in een rapport 4](#_Toc144115386)

[Werking Python-code voor het filteren van gegevens uit de datalayer (returnReportData) 5](#_Toc144115387)

# **Beschrijving BITT**

Het BITT-project is gericht op het vinden van rapport-sourcebestanden binnen EPZ-servers. Deze bestanden worden geanalyseerd met behulp van Python om daar bepaalde informatie uit te halen. Vervolgens worden deze gegevens opgeslagen in een Microsoft SQL Server database.

## Het doel van het project

Het doel van het project is om een rapport te creëren waarin we kunnen zien wat de wat de locatie is van de rapporten binnen EPZ, en wat de inhoud is van deze rapporten. Door een programma te maken wat elke dag uitgevoerd wordt, kunnen wijzigingen worden bijgehouden in de rapporten. Als er fouten ontstaan in een rapport, kunnen we zo kijken in welke query/query’s het probleem zich bevindt, zonder handmatig alle rapporten te hoeven doorzoeken. Door de inhoud van de rapporten op te halen, kan gekeken worden welke databases/servers betrokken zijn bij de werking van het rapport, en kunnen zo fouten geanalyseerd worden.

Het project is opgebouwd uit verschillende stappen, waaronder het zoeken naar de rapporten, het extraheren van de informatie uit deze bestanden en het opslaan van deze informatie in een database, met als laatste het weergeven in een rapport. Deze stappen worden uitgevoerd door python scripts, stored procedures in SQL Management Studio en een rapport door middel van de Logi omgeving.

# **Werking**

Om de rapporten te vinden, worden er stappen uitgevoerd in zowel Python als SQL. Daarom moet gezocht worden naar bestanden met een ".lgx"-formaat. Deze rapportbestanden bevatten alle gegevens die nodig zijn voor het traceren van de locatie van de databases en tabellen.

## Stap 1: Het extraheren van gegevens

De rapporten bevinden zich in een standaardopbouw van mappen op een server:

* Hoofdmap (root/file\_root): Dit is de bovenliggende map of de hoofdmap van het bestandssysteem. Het kan worden beschouwd als de startlocatie voor het programma.
* Domein A, B, C: Dit zijn submappen binnen de hoofdmap. Elke submap vertegenwoordigt een specifiek domein of een functioneel gebied. Voorbeelden hiervan zijn Inkoop, Workmanagement.
* Specifieke mapstructuur: Binnen elke domeinsubmap is er een specifieke mapstructuur aanwezig. Binnen deze mapstructuur bevinden zich altijd de rapporten. In dit programma is de specifieke mapstructuur als volgt: "\_Definitions\\\_Reports"
* Rapport: Dit zijn de daadwerkelijke bestanden die binnen het domein worden geplaatst. Dit zijn de .lgx-bestanden waaruit gegevens worden geëxtraheerd en verwerkt.

Het schema geeft dus een gestructureerde manier weer om rapportbestanden te organiseren op basis van domeinen en specifieke mapstructuren. Door deze structuur te volgen, kunnen rapporten gemakkelijk worden gelokaliseerd en geïdentificeerd op basis van hun locatie in het bestandssysteem.

De informatie die opgehaald moet worden, bevindt zich in rapporten met een XML-structuur.

De XML-structuur in een rapport volgt de regels van XML (eXtensible Markup Language). XML is een opmaaktaal die wordt gebruikt om gegevens te structureren en te beschrijven, en bestaat uit verschillende elementen. Elk element heeft een tag (naam) en kan inhoud bevatten. De tags worden gebruikt om de elementen te categoriseren.

Het belangrijkste element in de rapporten is de datalayer. In de datalayer bevindt zich de belangrijkste informatie over SQL-query’s en de query’s zelf. Deze SQL-query’s halen informatie op uit databases/servers, waarvan de locatie in beeld moet worden gebracht.

De datalayer bevat de volgende informatie:

1. ID van de datalayer (DatalayerID): Dit is een unieke identificatiecode die aan elke datalayer wordt toegewezen. Het wordt gebruikt om de datalayer te identificeren en te onderscheiden van andere datalagen.
2. De connectie met de server waar de query op wordt uitgevoerd (ConnectionID)
3. Query (Query): Dit is de SQL-query die wordt gebruikt om gegevens op te halen uit een bepaalde database. Het bevat instructies en voorwaarden om de gewenste gegevens te filteren en op te halen.
4. Het object, de database waaruit informatie wordt opgehaald (Object): Dit verwijst naar het specifieke object (bijvoorbeeld een tabel of weergave) in de database waaruit de informatie wordt opgehaald, deze staat binnen de query.

De connectionID bevat indirect de volgende informatie:

1. De database waarmee een verbinding wordt gemaakt, zodat de query kan uitgevoerd worden.

De informatie over deze database bevindt zich in de “Settings”-bestanden. In deze bestanden wordt ingesteld wat de ID is van verschillende tabellen. Om de informatie op te halen uit de Settings-bestanden, is er een functie in python “returnSettingsData.py”

Door middel van python worden deze rapporten vanuit de hoofdmap gelokaliseerd, waarna de gegevens eruit opgehaald worden en in een SQL-tabel geplaatst worden.

De volgende Python-functies zijn aanwezig (zie commentaar in python-bestanden):

1. Een functie voor het lokaliseren van rapport-sourcebestanden binnen een bepaald domein. (locateReports)
2. Een functie voor het halen van connectie-informatie uit “Settings.lgx” bestanden (returnSettingsData)
3. Een functie voor het extraheren van gegevens en SQL-query's uit de datalayer in de bestanden (returnReportData)
4. Een functie voor het plaatsen van deze gegevens in een tijdelijke MSSQL-tabel (saveDataToTable)

## Stap 2: Het invoegen en verwerken in een MSSQL-database.

De informatie die met de python functies is opgehaald, wordt in een tijdelijke MSSQL-tabel (logi.temp\_xml\_data) geplaatst. Hierna gebeurt het volgende:

1. Het object die aangeroepen wordt in de query in een rapport wordt vergeleken met de informatie uit [DWH.CDWH\_FB].INFORMATION\_SCHEMA.COLUMNS . Dit is een tabel waarin zich alle relevante objecten bevinden. Als het object niet overeenkomt met de informatie uit deze tabel, kan het overgeslagen worden
2. De gegevens worden vergeleken met gegevens van eerdere runs
   1. Er wordt vergeleken op basis van nieuwe gegevens
   2. Er wordt vergeleken op basis van verwijderde gegevens
   3. Er wordt vergeleken op basis van gewijzigde gegevens

De rijen worden gevlagd, zodat precies gezien kan worden of gegevens geldig zijn, en welke datum.

1. Alle gegevens worden toegevoegd aan de database/tabel “logi.xml\_data”

## Stap 3: Het plaatsen van de informatie in een rapport

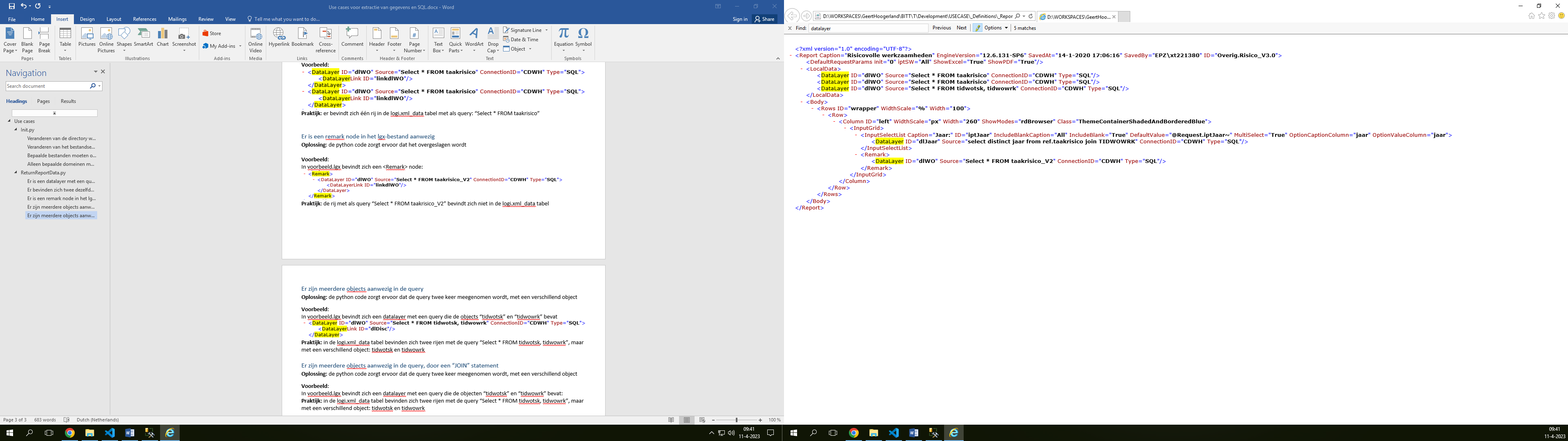
In het rapport TrackAndTrace.lgx wordt de informatie uit de SQL tabel in een rapport weergegeven. Hier kan gefilterd worden op verschillende waarden. Door op de naam van het rapport te klikken, kan een subtabel getoond worden, waarin de informatie binnen het rapport te zien is.

## Werking Python-code voor het filteren van gegevens uit de datalayer (returnReportData)

### Er is een datalayer met een query aanwezig in het bestand

De python code zorgt ervoor dat de gegevens, onder andere de query, uit de datalayer wordt gehaald

**Voorbeeld:**

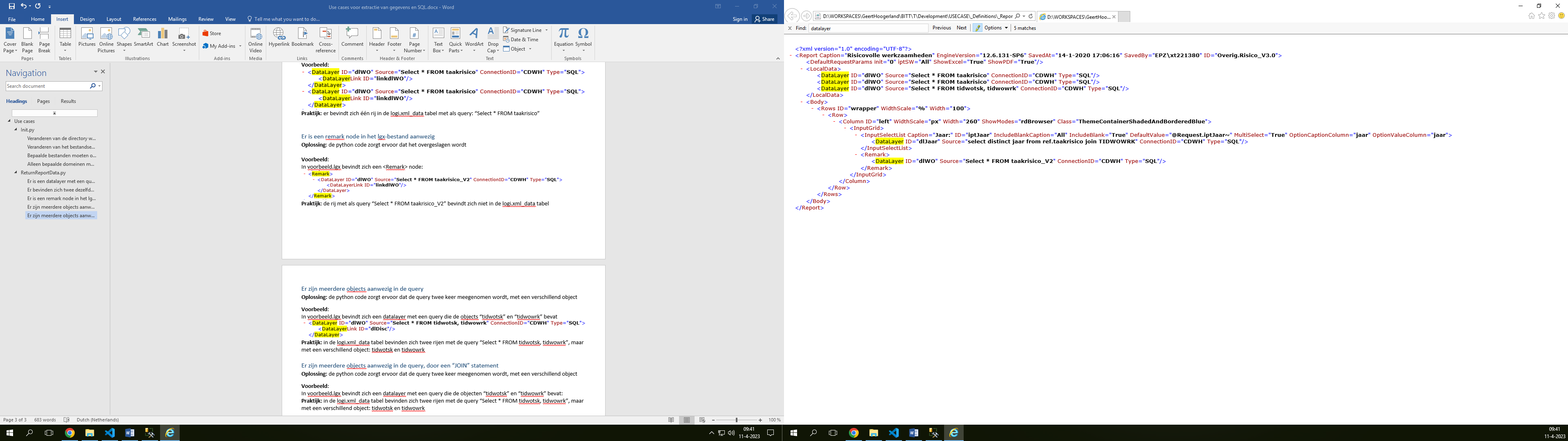


* Er bevindt zich in de logi.xml\_data tabel een rij met als query: “Select \* FROM taakrisico”

### Er bevinden zich twee dezelfde datalayers in het bestand

De python code zorgt ervoor dat de rij met gegevens één keer voorkomt in de logi.xml\_tabel

**Voorbeeld:**

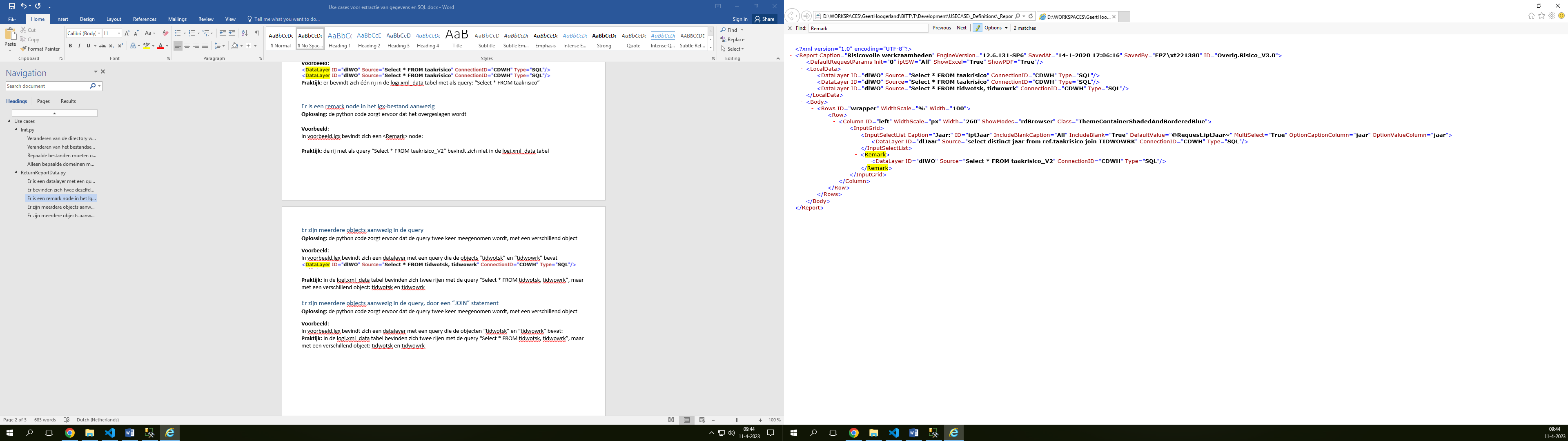


* Er bevindt zich één rij in de logi.xml\_data tabel met als query: “Select \* FROM taakrisico”

### Er is een “Remark” node in het .lgx-bestand aanwezig

De python code zorgt ervoor dat het overgeslagen wordt

**Voorbeeld**:



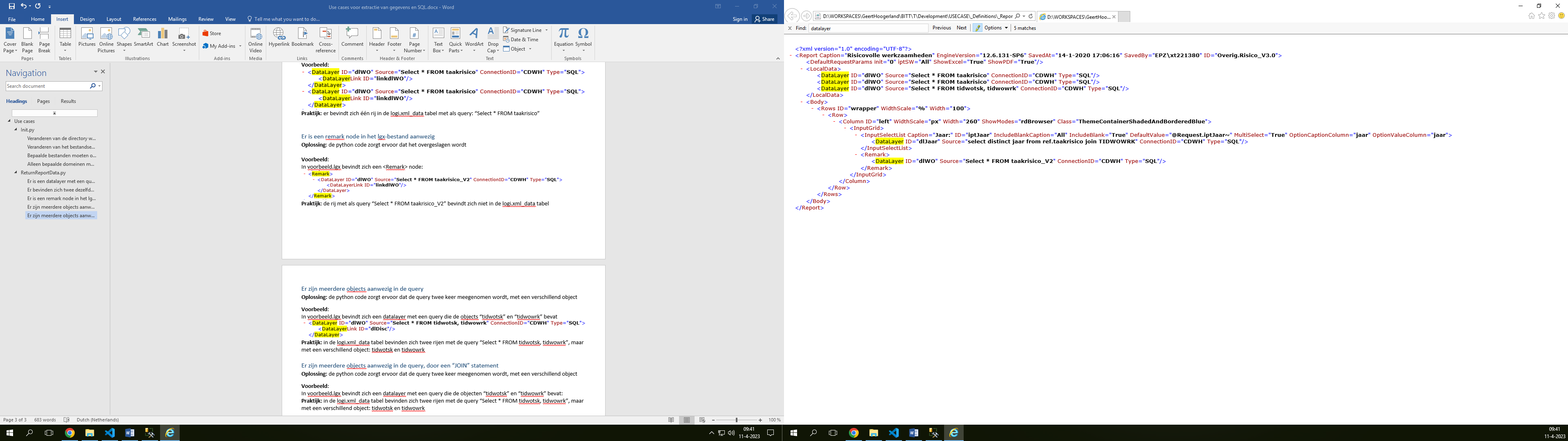
* De rij met als query “Select \* FROM taakrisico\_V2” bevindt zich niet in de logi.xml\_data tabel

### Er zijn meerdere objects aanwezig in de query

De python code zorgt ervoor dat de query twee keer meegenomen wordt, met een verschillend object

**Voorbeeld**:

Er bevindt zich een datalayer met een query die de objects “tidwotsk” en “tidwowrk” bevat



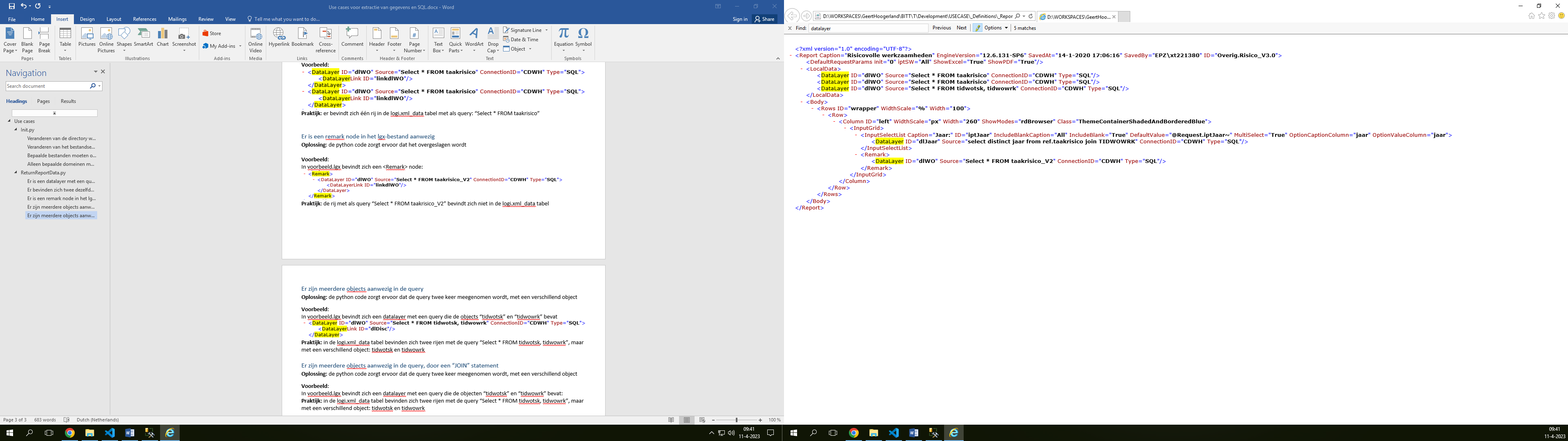
* In de logi.xml\_data tabel bevinden zich twee rijen met de query “Select \* FROM tidwotsk, tidwowrk”, met een verschillend object: tidwotsk en tidwowrk

### Er zijn meerdere objects aanwezig in de query, door een “JOIN” statement

De python code zorgt ervoor dat de query twee keer meegenomen wordt, met een verschillend object

**Voorbeeld**:

Er bevindt zich een datalayer met een query die de objecten “tidwotsk” en “tidwowrk” bevat:



* In de logi.xml\_data tabel bevinden zich twee rijen met de query “Select \* FROM tidwotsk, tidwowrk”, met een verschillend object: tidwotsk en tidwowrk

# **Beheer**

## Locaties van projectbestanden

* Het BiTT-project bevindt zich in de map “D:\Workspaces\GeertHoogerland\BITT\O”
  + De python-bestanden bevinden zich in \src
  + Het run.bat-bestand om door middel van pip de bibliotheken te installeren en het programma uit te voeren.
  + De logbestanden bevinden zich in \src\logfiles
* De documentatie en use cases bevinden zich in \docs

## Uitvoeren van het programma

* Om het programma initieel uit te voeren kan het bestand main.py handmatig gerund worden met als parameter ‘--init’. Dit kan gedaan worden in de command line. Bij de initiële run worden alle tabellen gemaakt wanneer niet aanwezig en alle tabellen leeggehaald.
* Voor het programma uitgevoerd wordt, moeten de variabelen in het init.py bestand worden aangepast. In dit bestand is de uitleg aanwezig van de variabelen. Dit is ook te vinden in de use cases.
* Om het programma uit te voeren zijn de volgende dingen vereist:
  + Toegang tot het uitlezen van bestanden in de gewenste server(s)
  + Toegang tot python
  + Toegang tot “pip” om bibliotheken te kunnen installeren indien nodig
* Er moet een gebruikersaccount zijn met rechten om elke dag main.py uit te voeren via de task scheduler.
* De task scheduler moet het volgende doen:
  + Een taak uitvoeren met:
    - Als gebruiker het systeemaccount
    - Uitvoeren afgezien van of de user ingelogd is of niet
  + Als trigger een tijdsschema:
    - Elke dag uitgevoerd worden op een bepaalde tijd
    - Stop de taak als het langer dan een uur duurt
  + Als actie het uitvoeren van het bestand main.py (dmv python: python main.py)