AGRUPAMENTO DE ESCOLAS DO CASTÊLO DA MAIA

Curso Profissional de Gestão e Programação de Sistemas Informáticos

RELATÓRIO FINAL DA FORMAÇÃO EM CONTEXTO DE TRABALHO (ANO II)

ALUNO/A	CRISTIANO REINALDO SANTOS MONTEIRO
ANO / TURMA	12º F
PROFESSOR/A ORIENTADOR/A	LUÍS GONÇALO
ENTIDADE DA FCT	CÂMARA MUNICIPAL DA MAIA
TUTOR	PEDRO PIMENTA

ANO LETIVO 2023 / 2024

AE CASTÊLO DA MAIA, 7 DE MAIO DE 2024

Agradeço à Câmara Municipal da Maia, em especial ao meu tutor, Engenheiro Pedro Pimenta por me ter concedido a oportunidade de realizar este estágio e por me ter permitido adquirir valiosas experiências na área de tratamento de dados e programação. Agradeço também a sua orientação, paciência e ensinamentos, a dedicação e conhecimento foram cruciais para o meu desenvolvimento profissional. Aos técnicos Bruno e Emanuel, fico grato pela disponibilidade para me ajudar e esclarecer as minhas dúvidas. Todas as contribuições foram essenciais para a minha aprendizagem e desenvolvimento ao longo da minha Formação em Contexto de Trabalho.

O apoio da Escola, dos professores, Ricardo Soares e Luís Gonçalo, permitiram igualmente ter esta experiência enriquecedora permitindo-me crescer profissionalmente e pessoalmente.

Sou grato por tudo o que aprendi e pelas pessoas que conheci.

Índice

Introdução

Este relatório apresenta a experiência de FCT realizada na Câmara Municipal da Maia, no âmbito do último ano do curso Técnico de Gestão e Programação de Sistemas Informáticos. Com a duração total de 325 horas, a FCT decorreu entre 19 de fevereiro de 2024 a 23 de abril de 2024 sob a orientação do Engenheiro Pedro Pimenta e do Professor Luís Gonçalo. As atividades realizadas integraram-se na equipa de Departamento de Qualidade de Sistemas de Informação e focaram-se na criação de dashboards para o Power BI e utilização da plataforma OpenDataSoft.

Identificação do aluno

Nome: Cristiano Reinaldo Santos Monteiro
Email: crismm.1106@gmail.com

Identificação do professor orientador da FCT

Nome: Luís Gonçalo
Contacto telefónico: 919976009
Email: luisgoncalo@aecastelomaia.pt

Identificação do tutor da FCT

Nome: Eng.º Pedro Pimenta
Contacto telefónico: 916106292
Email: pedroccpimenta@gmail.com
Função: Consultor

Identificação da entidade da FCT

Designação da entidade: Câmara Municipal da Maia
Morada: Praça Dr. José Vieira de Carvalho, 4474-006 Maia, Portugal

Caracterização da Entidade da FCT

Organização governamental de nível concelhio (Câmara Municipal)
Endereço da Página: https://www.cm-maia.pt
Número de telefone: 229408600
Endereço de email: geral@cm-maia.pt
Latitude e longitude: 41.2332700779865, -8.622672708393615

Resumo

No relatório, descreverei minhas atividades ao longo da Formação em Contexto de Trabalho (FCT). Explorarei as tarefas realizadas, a aplicação dos conhecimentos adquiridos e os desafios enfrentados, assim como as estratégias empregadas para superá-los. Além disso, ressaltarei a significância da FCT em minha trajetória profissional e pessoal.

Finalidades da FCT

A FCT na Câmara Municipal da Maia foi uma etapa essencial do meu percurso educativo, proporcionando-me uma experiência prática relevante. Aprofundei o meu conhecimento em linguagens de programação como Python, HTML e CSS. Melhoria das habilidades sociais como comunicação de problemas, o que está a ser feito, quanto tempo a tarefa demora até ser concluída.

Áreas de formação

Consegui aplicar o meu conhecimento que foi obtido nas aulas de Programação de Sistemas Informáticos, Redes e Comunicação, Sistemas Operativos e estudos em casa. Com os meus professores aprendi a ser mais humilde e mais responsável.

Objetivos gerais

- Aplicar a contextos reais de trabalho os conhecimentos adquiridos nas várias disciplinas que compõem o curso;
- Desenvolver hábitos no domínio da sociabilização, solidariedade, respeito pelos outros e por si próprio;

Objetivos específicos

- Desenvolver, distribuir, instalar e efetuar a manutenção de aplicações informáticas, utilizando ambientes e linguagens de programação procedimentais e visuais;
- Manipular dados retirados de bases de dados;
- Desenvolver, instalar e efetuar a manutenção de sistemas de informação baseados nas tecnologias web.

Cronograma

Atividades / Tarefas		Semanas									
		1º	2º	3º	4º	5º	6º	7º	8º	9º	10º
		19.02 23.02	26.02 01.03	04.03 08.03	11.03 15.03	18.03 22.03	25.03 29.03	01.04 05.04	08.04 12.04	15.09 19.04	22.04 23.04
1	Escolher área das atividades da FCT										
2	Aprender a trabalhar com a plataforma OpenDataSoft										
3	Inserção de cerca de 500 datasets no OpenDataSoft										
4	Criação de propostas de Landing Pages para o OpenDataSoft										
5	Criação de Dashboards no Power BI										
6	Migração de Dashboard HGP para Power BI										
7	Pesquisa sobre a API de automação da OpenDataSoft										
8	Criação de dashboards ISO 37120 no Power BI										

Recursos

Nesta FCT foram utilizadas as seguintes plataformas e APIs fornecidas pela Câmara Municipal da Maia:

https://www.opendatasoft.com/

https://baze.cm-maia.pt/BaZe/api/api4sV3.php/

https://maia.opendatasoft.com/

Recursos de hardware e software:

- Computador com ligação à internet
- Teclado
- Rato
- Secretária
- Windows
- Linux
- Google Docs
- Google Sheets
- Google Chat
- Gmail
- Power BI

Desenvolvimento – opções estratégicas, problemas e soluções encontradas

Atividades desenvolvidas

- Escolher área das atividades da FCT
- Aprender a trabalhar com a plataforma Open DataSoft
- Inserção de cerca de 500 datasets no OpenDataSoft
- Criação de propostas de Landing Pages para o OpenDataSoft
- Criação de Dashboards no Power BI
- Migração de Dashboard HGP para Power BI
- Pesquisa sobre a API de automação da OpenDataSoft
- Criação de dashboards ISO 37120 no Power BI
- Escrita continua de um relatório em Google Docs proposta pelo tutor da FCT

Opções estratégicas / Aplicação de conhecimentos

Para ultrapassar as dificuldades encontradas recorri ao meu colega de estágio, Miguel Medeiros, ao meu tutor, Pedro Pimenta, e ao resto da equipa do departamento. Também utilizei a internet como fonte de referência e pesquisa de forma a solucionar os meus problemas e também os recursos que me foram fornecidos para aquela tarefa.

Aprendizagem da disciplina de Redes e Comunicação foi importante para desenvolver as tarefas propostas em HTML e CSS.

Soluções / Novas aprendizagens

Uma das aprendizagens que mais se reflete é a melhoria da escrita bem como, o que escrever em relatórios, pois é muitas vezes difícil transcrever o que foi realizado. Os conhecimentos na área da programação e análise de dados, foram aprofundados.

A utilização das plataformas como Power BI da Microsoft e OpenDataSoft foram uma nova e grande aprendizagem.

Problemas / Dificuldades no FCT

Compreender o que é necessário e como deveria redigir os relatórios, e ultrapassar a dificuldade em transcrever as tarefas propostas pela equipa.

Desenvolvimento futuro

Impacto

Durante a minha FCT, pude aprimorar várias competências técnicas e especializar-me na minha área de atuação. Além disso, desenvolvi habilidades interpessoais e de comunicação, essenciais para o ambiente profissional. A FCT proporcionou-me ainda a oportunidade de expandir a minha rede de contactos, abrindo portas para futuras oportunidades. A gestão de tempo e a autonomia também foram aspectos que desenvolvi, tornando-me mais eficiente e independente no meu trabalho.

Sugestões

Nada a referir.

Autoavaliação

Durante a minha participação na formação em contexto de trabalho na Câmara Municipal da Maia, posso afirmar que as minhas expectativas foram plenamente atendidas. Ao mesmo tempo, percebi que as necessidades e objetivos da empresa foram igualmente alcançados. A experiência proporcionou-me oportunidades valiosas de aprendizagem e desenvolvimento, contribuindo significativamente para o meu crescimento profissional e pessoal.

Autoavalio-me com 19 valores.

Reflexão final

A FCT na Câmara Municipal da Maia foi uma experiência valiosa na área de análise de dados, com foco nas ferramentas Power BI e OpenDataSoft. Desenvolvi habilidades na recolha, organização, análise e visualização de dados, contribuindo para decisões estratégicas. A experiência com as plataformas evidenciou a importância da análise de dados para otimizar processos em instituições públicas. Estou grato pela oportunidade e acredito que as habilidades adquiridas serão essenciais para o meu futuro profissional.

Assinatura

Cristiano Monteiro

Bibliografia / Webgrafia

Google Maps (https://maps.google.com/)

Website da Câmara Municipal da Maia https://www.cm-maia.pt/

Anexos

Integração e Visualização de Dados

Cristiano Reinaldo Santos Monteiro

Relatório de Estágio

Técnico de Gestão e Programação de Sistemas Informáticos
12º Ano

Câmara Municipal da Maia, 7 de maio de 2024

PESSOAS 2030 PORTUGAL 2030 Cofinanciado pela União Europeia
Os Fundos Europeus mais próximos de si.

Sumário

Introdução

O relatório tem como objetivo relatar o que foi feito durante as 325 horas de estágio, bem como as dificuldades e estratégias adotadas para superá-las.

Início da utilização da OpenDataSoft

Durante a minha utilização da plataforma OpenDataSoft, uma das principais dificuldades que enfrentei foi compreender o seu funcionamento. Inicialmente, o ambiente da plataforma mostrou-se complexo, exigindo um período de adaptação curto para entender a sua interface e funcionalidades.

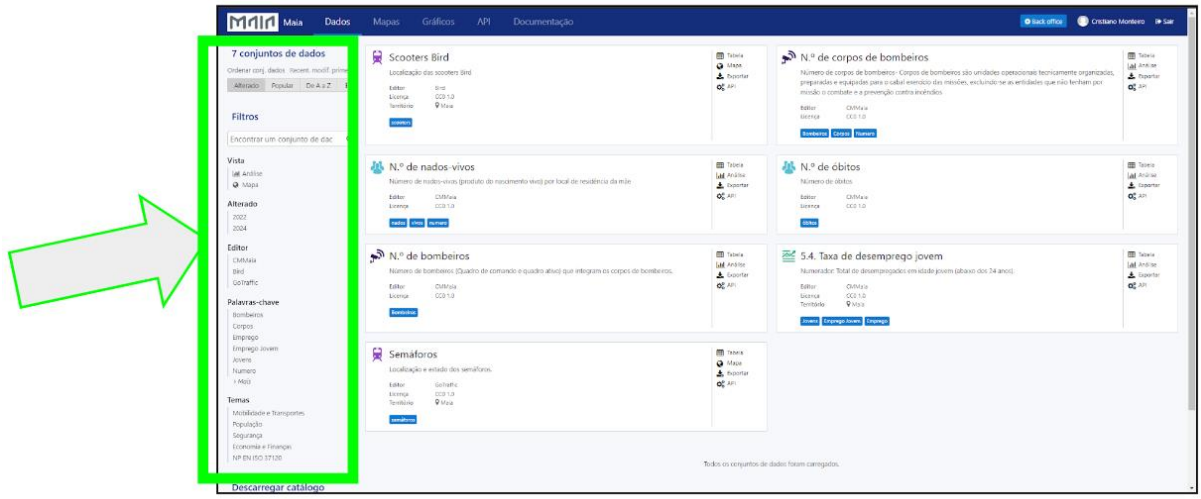


Figura 1. Landing page da plataforma https://maia.opendatasoft.com/

Quando me refiro ao "ambiente" da plataforma, estou a falar do conjunto de características, ferramentas e opções que esta oferece. Das várias secções e possibilidades disponíveis, explorei principalmente a criação e visualização de conjuntos de dados.

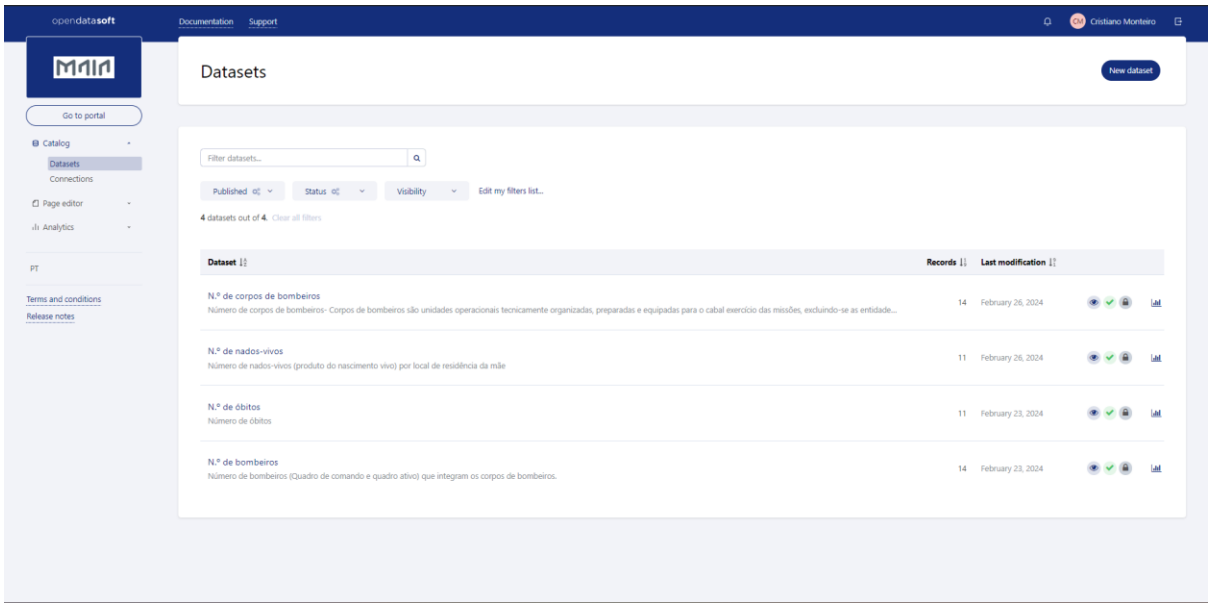


Figura 2. Back-office da plataforma https://maia.opendatasoft.com/

Quando menciono "interface", estou a referir-me à forma como os elementos da plataforma são apresentados e como interajo com eles. As "funcionalidades" são as capacidades e recursos oferecidos pela plataforma, como a importação de dados, a criação de visualizações e a configuração de APIs.

Outra dificuldade, foi uma situação em que estávamos a utilizar uma API inadequada para aceder e manipular os dados.

Figura 3. Funcionalidade de criar um conjunto de dados.

Estratégias de Superação

Para superar as dificuldades iniciais, foi fundamental recorrer a recursos externos. O site da OpenDataSoft Academy (https://academy.opendatasoft.com/) revelou-se especialmente útil, fornecendo orientações detalhadas (- quais –) sobre o funcionamento da plataforma.

Além disso, o acesso ao tutorial fornecido, (https://docs.google.com/document/d/1pbZMFtmCqDPvEpyLS7asnHxCV594iK_8rIskoltQ4ZM/edit#heading=h.n22lht7xmoms), foi essencial para a compreensão passo a passo das funcionalidades da plataforma.

Para superarmos a dificuldade que estávamos a ter por estarmos a aceder de forma inadequada para aceder e manipular os dados, apercebemos que tínhamos de usar a API4s V3 (https://baze.cm-maia.pt/BaZe/api/api4sV3.php) e não a API4s (https://baze.cm-maia.pt/BaZe/api/api4s.php), pois estávamos a ver tutoriais do OpenDataSoft Academy (Tutorial 1, Tutorial 2, Tutorial 3, Tutorial 4, Tutorial 5) e só tínhamos noção que só existia a API4s. A diferença entre os objetos JSON que são retornados pelos 2 endpoints são que a API4s V3 retorna os dados dentro do elemento “data” que faz com que a plataforma OpenDataSoft consiga ler o JSON File.

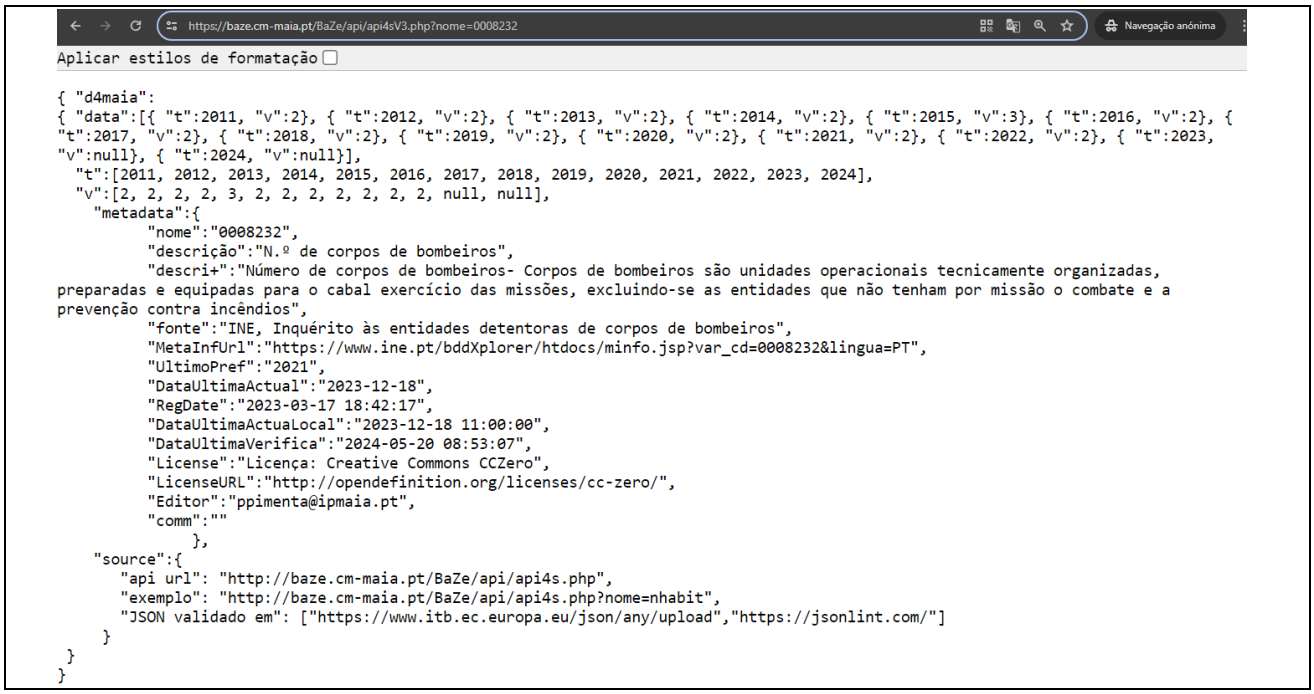


Figura 4. Endpoint da API4s V3.

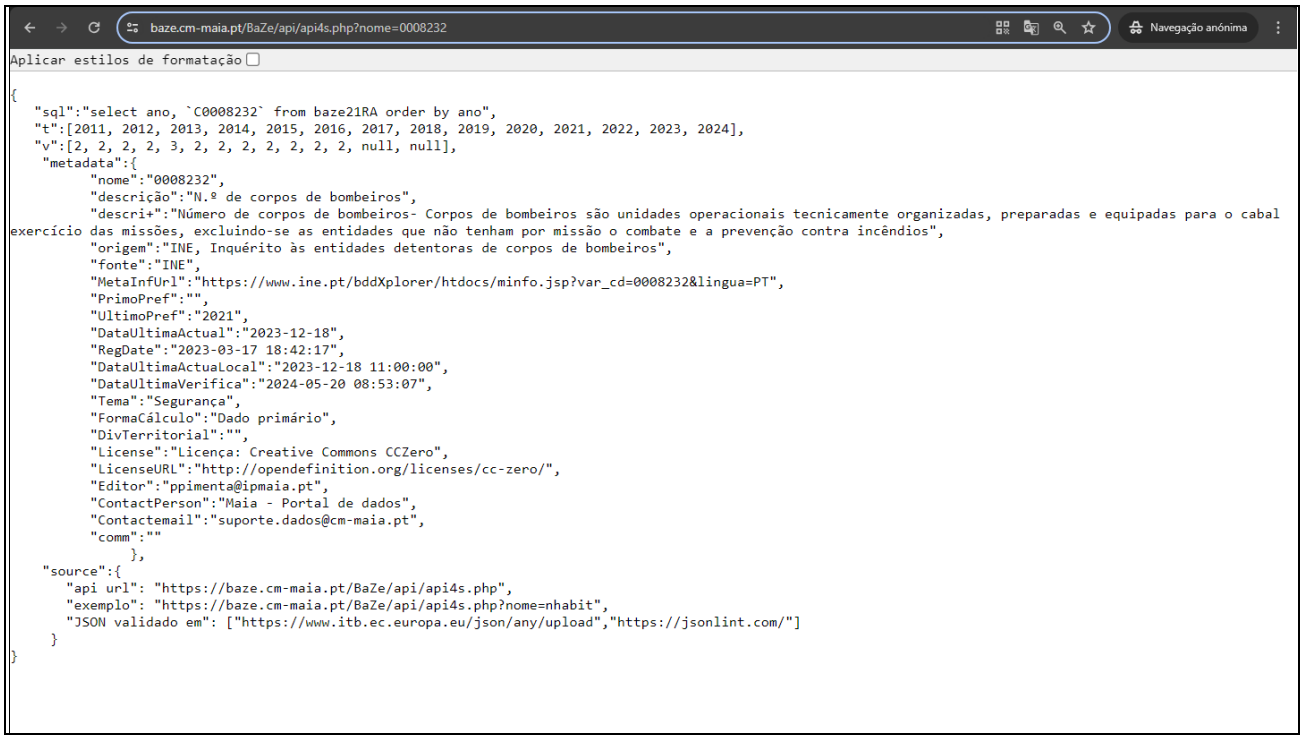


Figura 5. Endpoint da API4s.

Com o auxílio dos materiais e do tutorial, conseguimos produzir 4 conjuntos de dados, cada um acompanhado de uma tabela para análise subsequente.

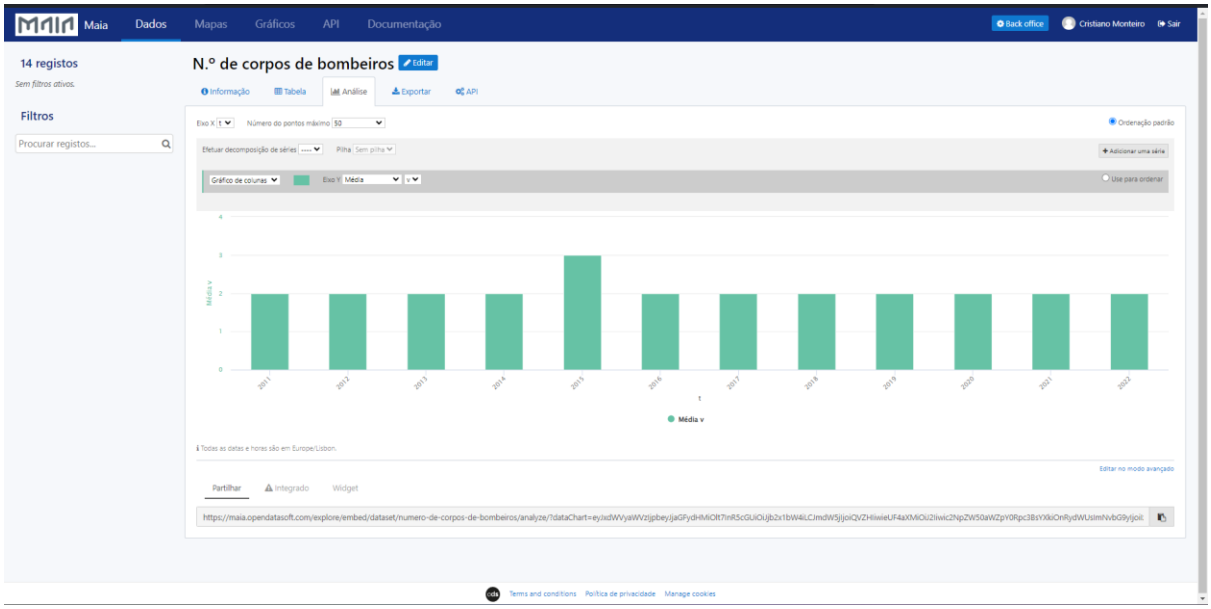


Figura 6. Conjunto de dados criado (Nº de corpos de bombeiros).

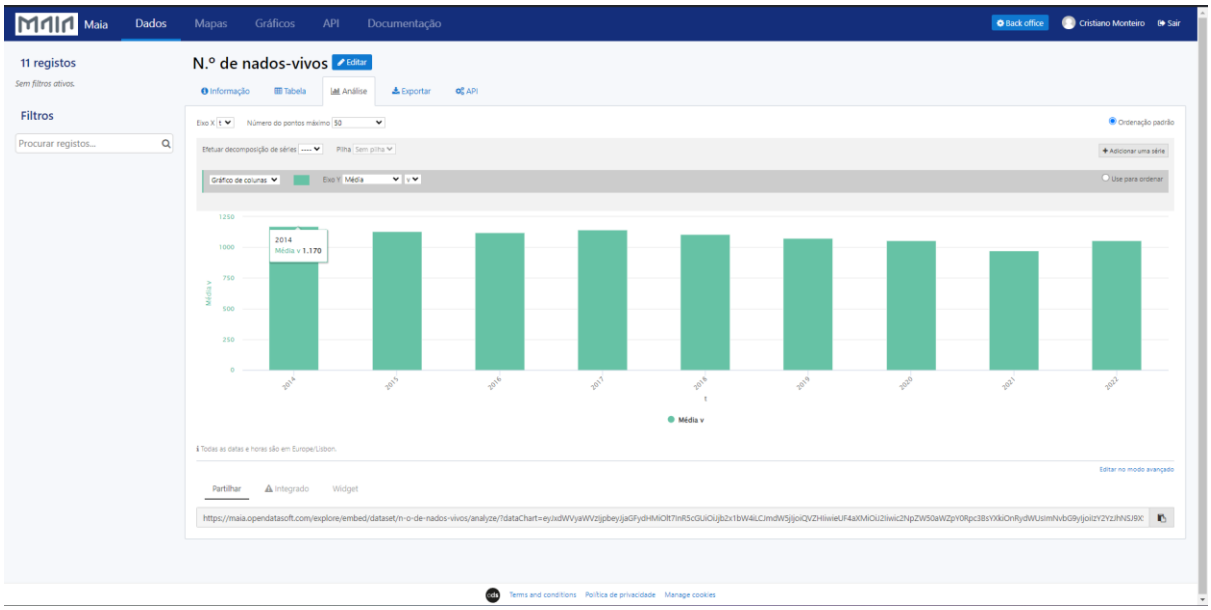


Figura 7. Conjunto de dados criado (Nº de nados-vivos).

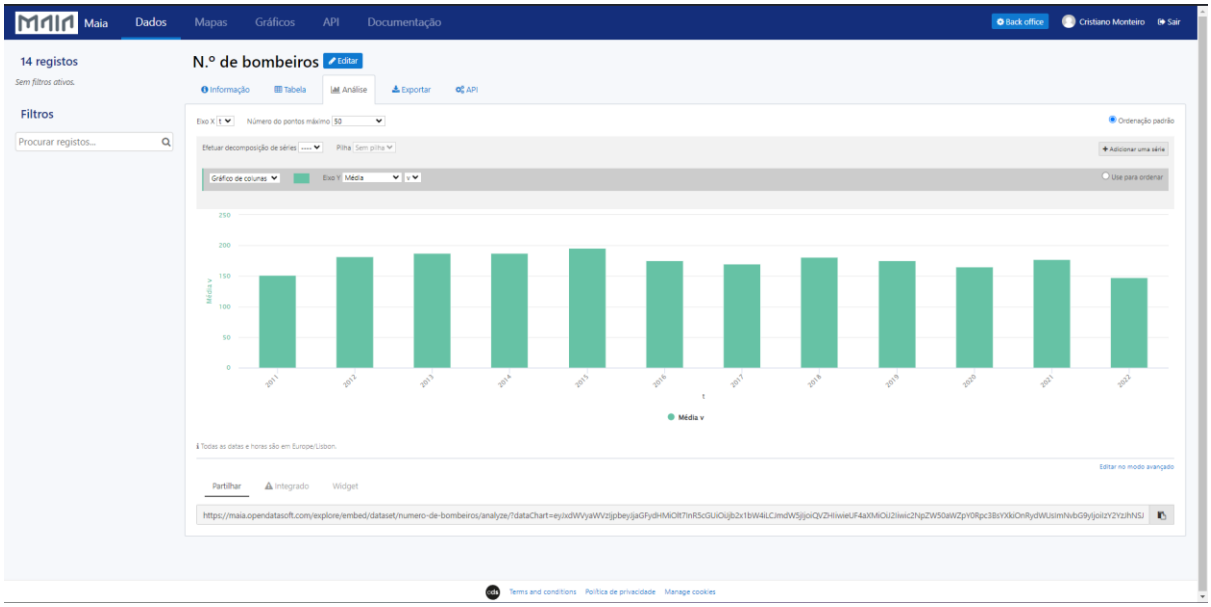


Figura 8. Conjunto de dados criado (Nº de bombeiros).

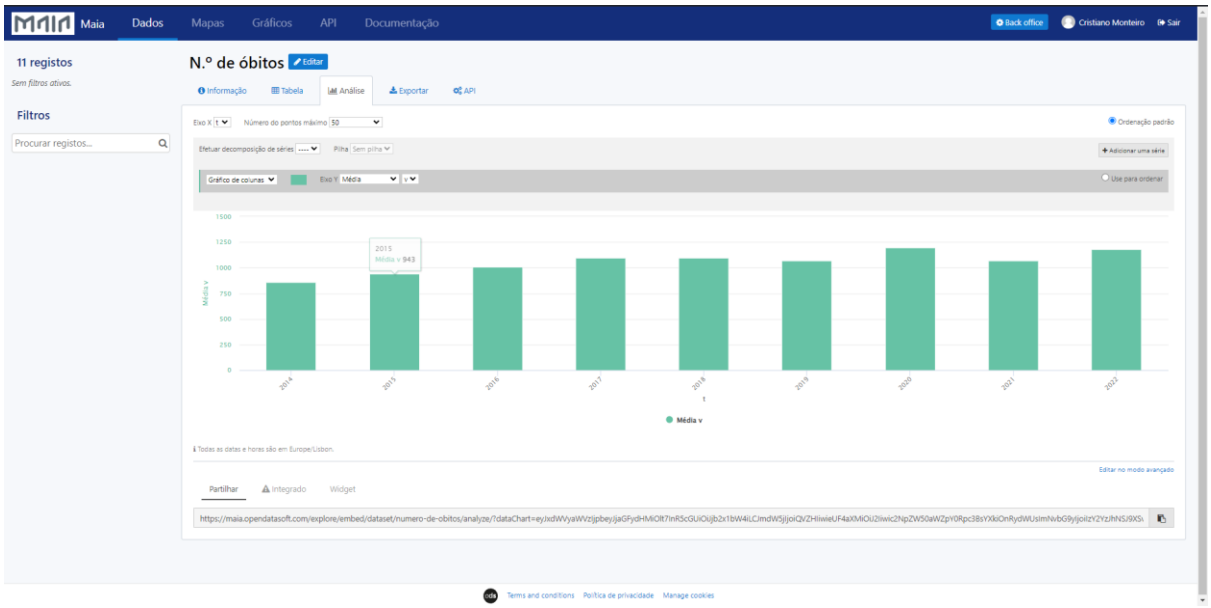


Figura 9. Conjunto de dados criado (Nº de óbitos).

Inserir datasets referentes à ISO 37120

Na reunião semanal da equipa de dados do município, foi-me atribuído o prazo para realizar a tarefa de inserir os conjuntos de dados relativos à ISO 37120 na plataforma OpenDataSoft até 26-03-2024.

Iniciei a introdução manual de alguns dados na plataforma, tal como já havia feito anteriormente. Face à considerável quantidade de informação a inserir - cerca de 500 conjuntos de dados -, ocorreu-me a ideia de automatizar este processo. Assim, avancei com a criação de um script para me auxiliar nesta tarefa.

Antes de proceder com a automatização deste processo, pesquisei informações sobre as APIs oferecidas pela plataforma. Encontrei, em https://help.opendatasoft.com/apis/ods-automation-v1/#tag/Datasets/operation/create-dataset, orientações sobre como criar um dataset através da API REST.

Utilizei o Visual Studio Code (https://code.visualstudio.com/) como editor de código e a linguagem Python para a sua elaboração. A escolha desta linguagem foi deliberada, uma vez que a maioria da equipa possui conhecimentos em Python. Deste modo, garantia que qualquer membro da equipa poderia realizar eventuais alterações no script no futuro e, se necessário, poderia contar com a sua ajuda.

Utilizei também as documentações da API disponibilizadas pela OpenDataSoft (https://help.opendatasoft.com/apis/ods-automation-v1/ e https://maia.opendatasoft.com/api/explore/v2.1/console/). Mais adiante, descobri que o município ainda não tinha uma licença que permitisse essa utilização, pois exigia um plano mais abrangente.

A lógica deste script é simples, ele vai buscar os datasets registados na plataforma OpenDataSoft (https://maia.opendatasoft.com/explore/?sort=modified) e vai buscar também os dados da API4s V3 (https://baze.cm-maia.pt/BaZe/api/api4sV3.php).

```python
# Obtém o JSON de qualquer URL
def get_json_from_api(url: str, headers=None):
    try:
        resp = requests.get(url, headers=headers)

        if resp.status_code == 200:
            return resp.json()
        else:
            print("Failed to fetch data from API. Status code:", resp.status_code)
            return None
    except requests.exceptions.RequestException as e:
        print("Error:", e)
        return None

```

Figura 10. Função para obter o JSON de qualquer URL.

A função realiza uma tentativa (try) e utiliza a biblioteca requests para obter uma resposta a partir da URL fornecida. Caso o código de estado (status code) retornado seja igual a 200, a função procede à devolução do JSON correspondente à referida URL. Caso contrário, é devolvida uma mensagem de erro indicando a ocorrência de um problema, acompanhada do código de estado associado ao referido erro.

```python
import requests
import re
import time
```

Figura 11. Bibliotecas usadas.

```python
# Obtém os datasets criados e faz uma array com os códigos dos datasets criados
def get_registered_datasets():
    ods_dataset_codes = []
    ods_datasets = get_json_from_api(
        "https://maia.opendatasoft.com/api/automation/v1.0/datasets?limit=700",
        ods_headers,
    )

    for data in ods_datasets:
        title = data["dataset"]["metas"]["default"]["title"]
        reference = data["dataset"]["metas"]["default"]["references"]

        if reference is not None:
            parts = reference.split("=")

            if len(parts) >= 1:
                code = parts[1]
                ods_dataset_codes.append(code)
            else:
                print("Não é possivel identificar o código '=':", reference)
        else:
            print(title, "não tem referência por isso foi avançado.")

    return ods_dataset_codes
```

Figura 12. Função para obter os códigos dos datasets já criados.

Nesta função, foi criado um array denominado "ods_dataset_codes" com o intuito de armazenar os códigos obtidos dos datasets. Posteriormente, são obtidos os datasets retornados pela API da OpenDataSoft, seguido de um loop (for) para obter a referência do código da API v3 e armazená-lo no array "ods_dataset_codes". O propósito desta função é preservar os códigos dos datasets previamente criados, permitindo uma verificação

posterior durante a criação de novos datasets no código. Desta forma, é possível determinar se um dataset com o mesmo código já foi criado anteriormente, evitando duplicações.

A estrutura do JSON retornado pelo endpoint foi retirada da documentação da API, visto que não pude testar tudo.

```
84    # Obtém os datasets registados na plataforma
85    ods_registered_datasets = get_registered_datasets()
```

Figura 13. Variável onde chamo a função anteriormente explicada.

```
87  # Obtém os dados da API BaZe
88  api_data = get_json_from_api(maia_api_url)
89  create_dataset_url = "https://maia.opendatasoft.com/api/automation/v1.0/datasets/"
90  created_datasets = []
91
92  if api_data:
93      # Se obter os dados da API BaZe faz um loop de 1 em 1 segundo para não sobrecarregar as APIs
94      for data in api_data["o"]:
95          # time.sleep(1)
96
97          code = data[0]
98          theme = data[7]
99
100         if code == "Código":
101             continue
102
103         if code in ods_registered_datasets:
104             continue
105
106         if code in created_datasets:
107             continue
108
109         print("\nA criar o dataset " + code + " com o tema " + theme)
110
111         code_api_url = maia_api_url + "?nome=" + code
112         code_data = get_json_from_api(code_api_url)["d4maia"]
113
114         new_dataset_title = code_data["metadata"]["descrição"]
115         dataset_id = convert_to_slug(new_dataset_title)
116
```

Figura 14. Parte do código onde começo a criar os datasets.

Nesta parte do código vou então buscar os dados da API do município (https://baze.cm-maia.pt/BaZe/api/api4sV3.php/) e se houver resposta faço um **for loop** e guardo o código e o tema de cada série em duas variáveis chamadas **code** e **theme**. Verifico caso o código se chame "Código" avanço para a próxima iteração do loop, de seguida caso já exista um dataset com esse código também avanço, e para uma verificação adicional, no momento em que corro o programa, verifico se já foi criado algum dataset com aquele código para evitar duplicações em runtime.

O resto ainda não foi testado, pois como já referi no início deste tópico, no momento em que fiz o programa o município não tem licenças para funcionar com a API de automação, mas em teoria funciona.

```
117 >       new_dataset_data = {···
181
182         if post_dataset(create_dataset_url):
183             print("Dataset com o id " + dataset_id + " foi criado.")
184
185         publish_dataset_url = (
186             "https://maia.opendatasoft.com/api/automation/v1.0/datasets/"
187             + dataset_id
188             + "/publish/"
189         )
190
191         if post_dataset(publish_dataset_url):
192             print("Dataset com o id " + dataset_id + " foi publicado.")
193
194         created_datasets.append(code)
195
196     print("\n\nForam criados " + str(len(created_datasets)) + " datasets com sucesso!")
197
```

Figura 15. Resto do código do programa.

Nos anexos, disponibilizarei o link para download do programa mencionado anteriormente, no qual o código está completamente documentado.

No final de tudo tivemos (Eu e o Miguel) que inserir por volta de 500 dados no Open Data Soft à mão.

Criação de uma Landing Page no OpenDataSoft

Para criar uma página de entrada para o site https://maia.opendatasoft.com/ naveguei até ao botão "Back-office".

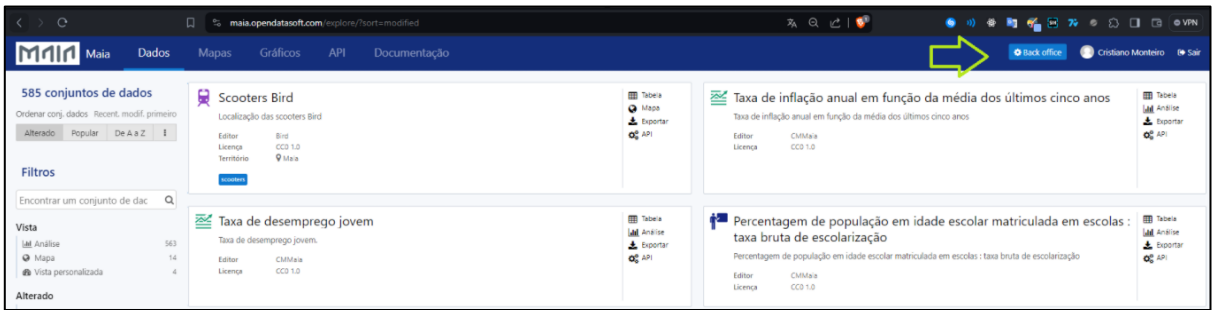


Figura 16. Botão que acede o back-office da plataforma OpenDataSoft.

Acedemos de seguida ao Page Editor => Code editor.

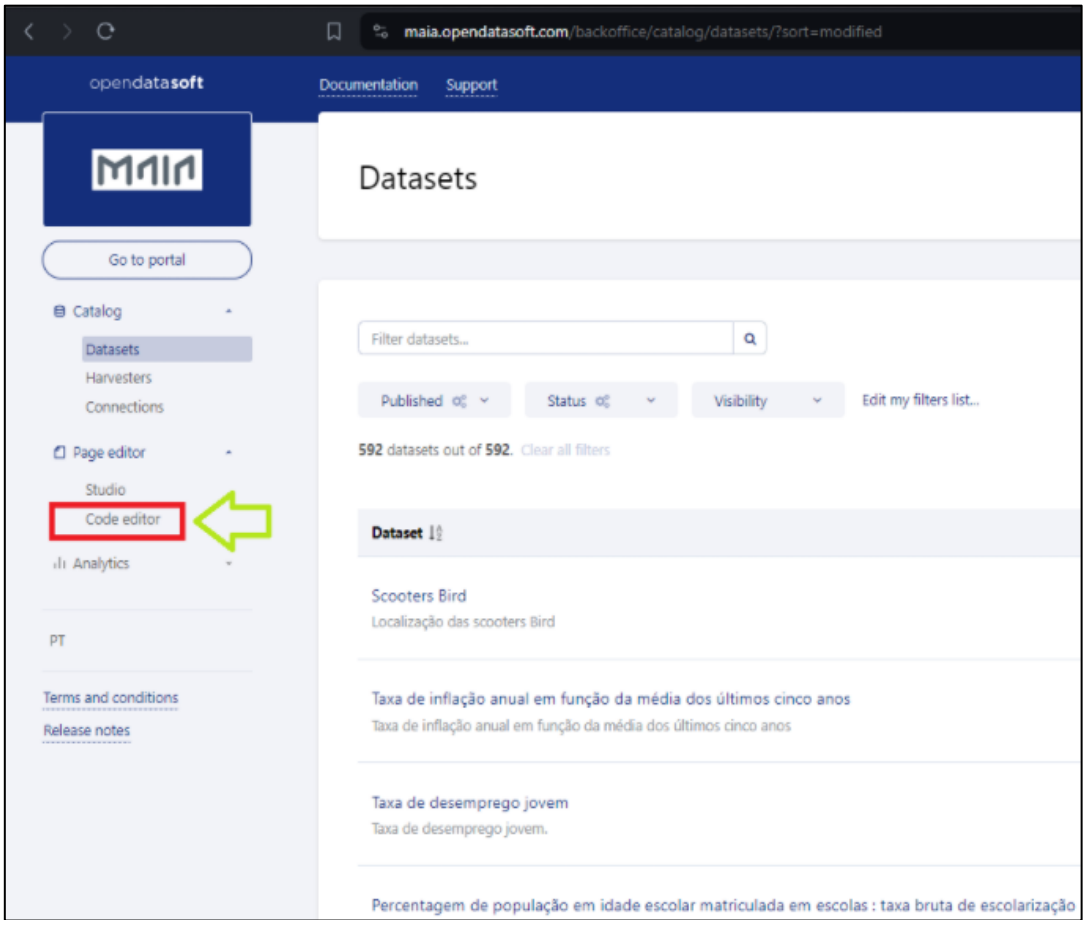


Figura 17. Botão que acede ao Code Editor.

Figura 18. Botão “Create Page”.

E depois clicamos no botão “Create Page”, que nos levará a esta página:

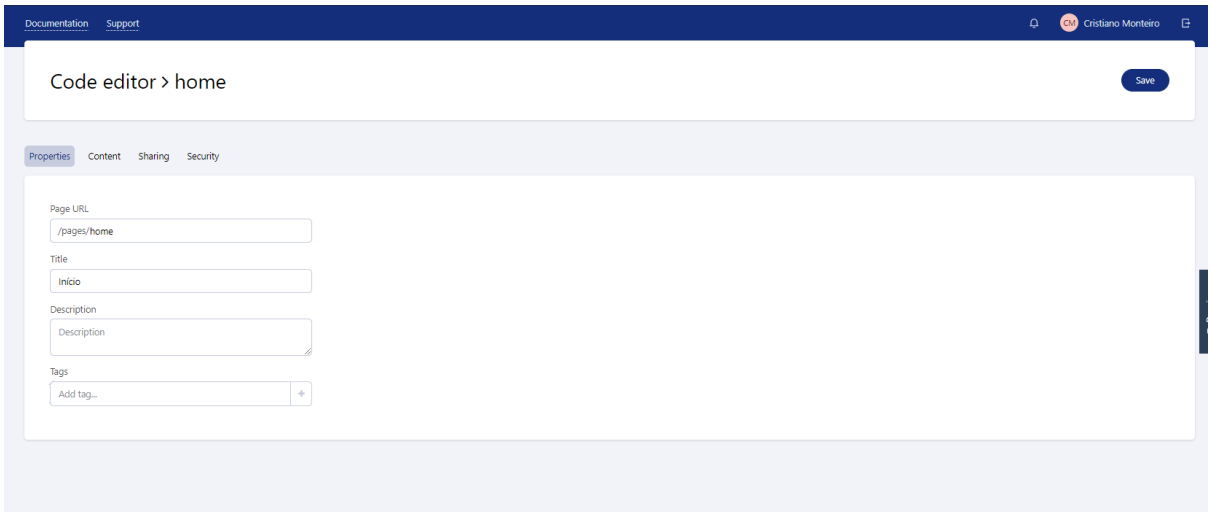


Figura 19. Página de propriedades do code editor.

Onde eu inseri o URL **/pages/home** e o título “Início”. E de seguida cliquei no botão Content e acedi a esta página onde utilizei um template Homepage - Dataset themes.

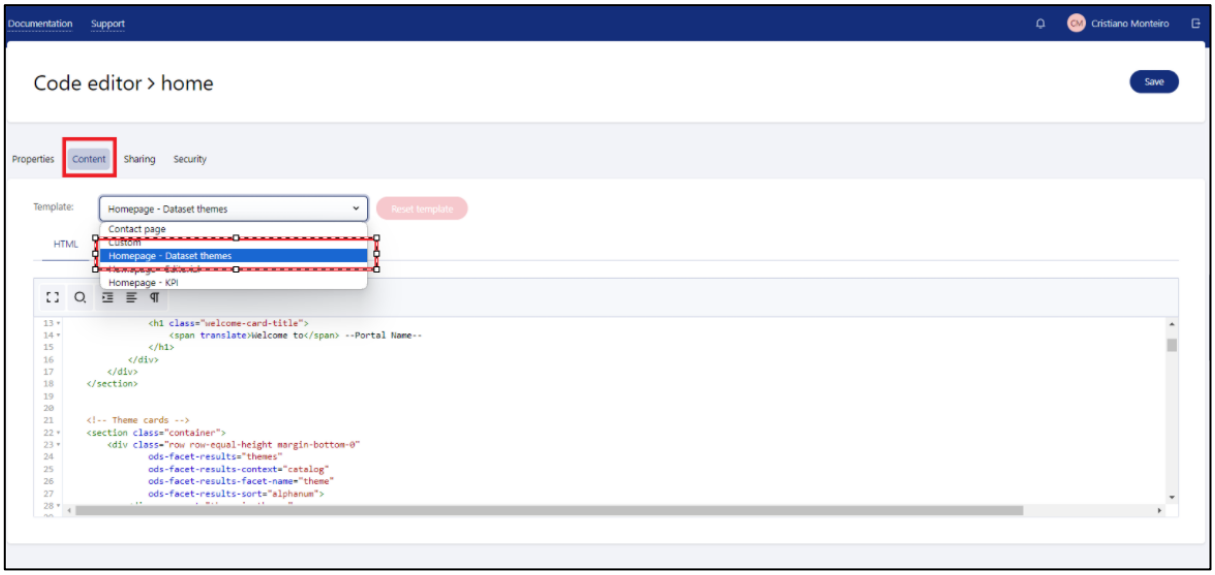


Figura 20. Escolher o tema do site.

Configurar dashboards no Power BI

Foi-nos atribuída a tarefa de "Configurar painéis no Power BI (indicadores internos) com base nos dados do datalake". Foram-nos fornecidos dois ficheiros para a realização da referida tarefa: a **lista de indicadores** (https://docs.google.com/spreadsheets/d/1z3maHwPsdSyR4cEc9PT5ICL74DttsP2NKbL_Ibei_Vk/edit#gid=1204078709) e o **REOT 2022,** (https://docs.google.com/spreadsheets/d/1M9WreQUC_uQePQXHEperAJ7mEBgouTdR9B0T1P55F5Y/edit#gid=1823075864).

Figura 21. Template da câmara municipal da Maia para o Power BI.

Tínhamos os ficheiros já com tudo carregado pelo Emanuel, e só tínhamos de preencher a informação clicando nas caixas dos dados para cada painel em alguns casos como o contador usamos apenas os dados da série em si mas o resto usamos o Ano e o nome da série e para o painel no canto inferior esquerdo usamos o Ano, nome da série a Meta Anual e a Meta de Superação.

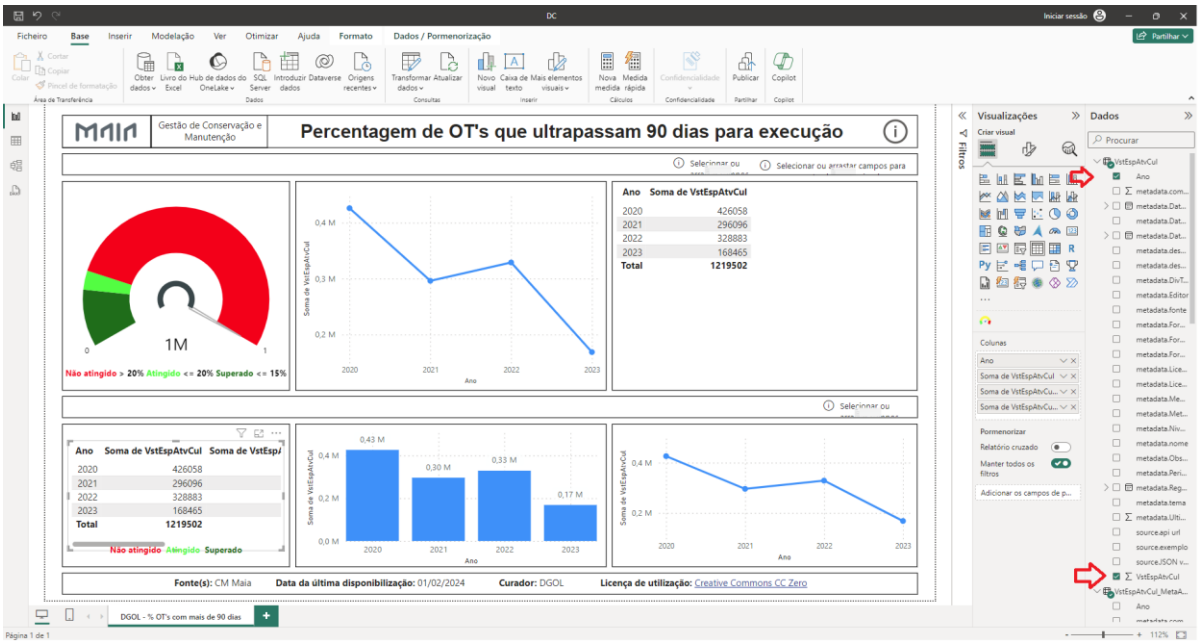
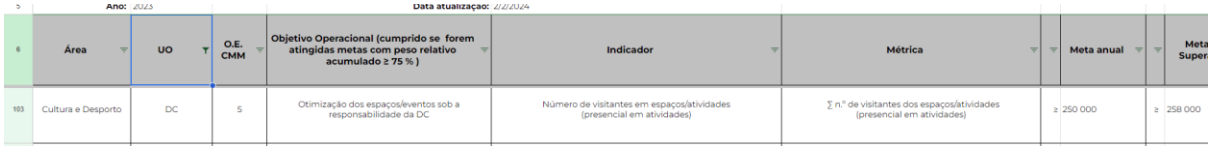


Figura 22. Informação já carregada nos painéis de visualização.

Após isso, o que fazemos é pesquisar na lista de indicadores para obter o resto dos dados como o título, objetivo, fórmula de cálculo etc.



	Área	UO	O.E. CMM	Objetivo Operacional (cumprido se forem atingidas metas com peso relativo acumulado ≥ 75 %)	Indicador	Métrica	Meta anual	Meta Super
103	Cultura e Desporto	DC	5	Otimização dos espaços/eventos sob a responsabilidade da DC	Número de visitantes em espaços/atividades (presencial em atividades)	∑ n.º de visitantes dos espaços/atividades (presencial em atividades)	≥ 250 000	≥ 258 000

Figura 23. Lista de indicadores.

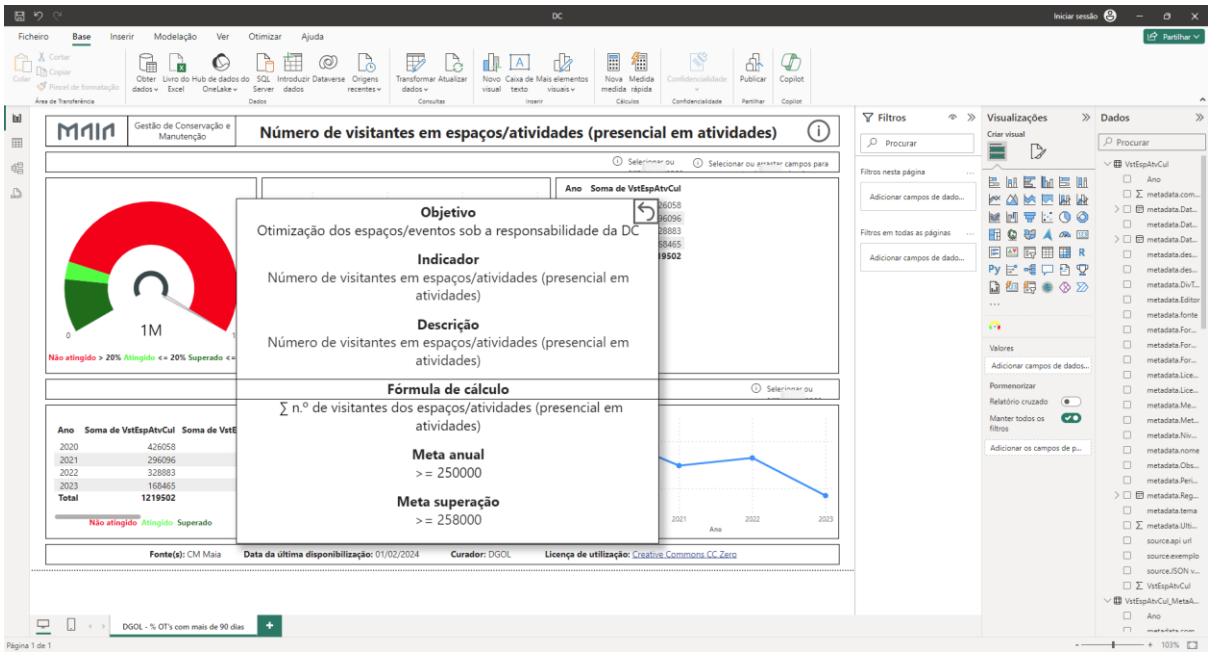


Figura 24. Dados já mudados no Power BI.

Aproveitamos e mudamos o nome das colunas clicando 2 vezes em cima desta caixa e mudamos isso para o nome do indicador em todos os painéis.

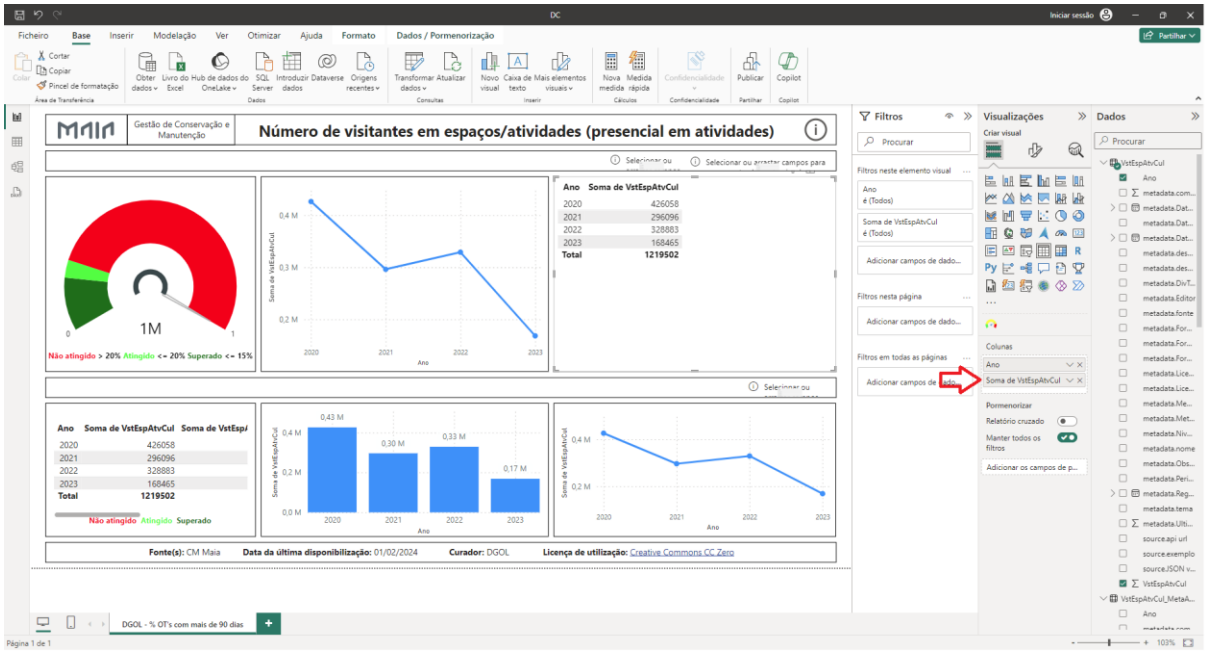


Figura 25. Onde mudamos o nome das colunas.

Agora para finalizarmos temos de editar a consulta no painel inferior esquerdo.

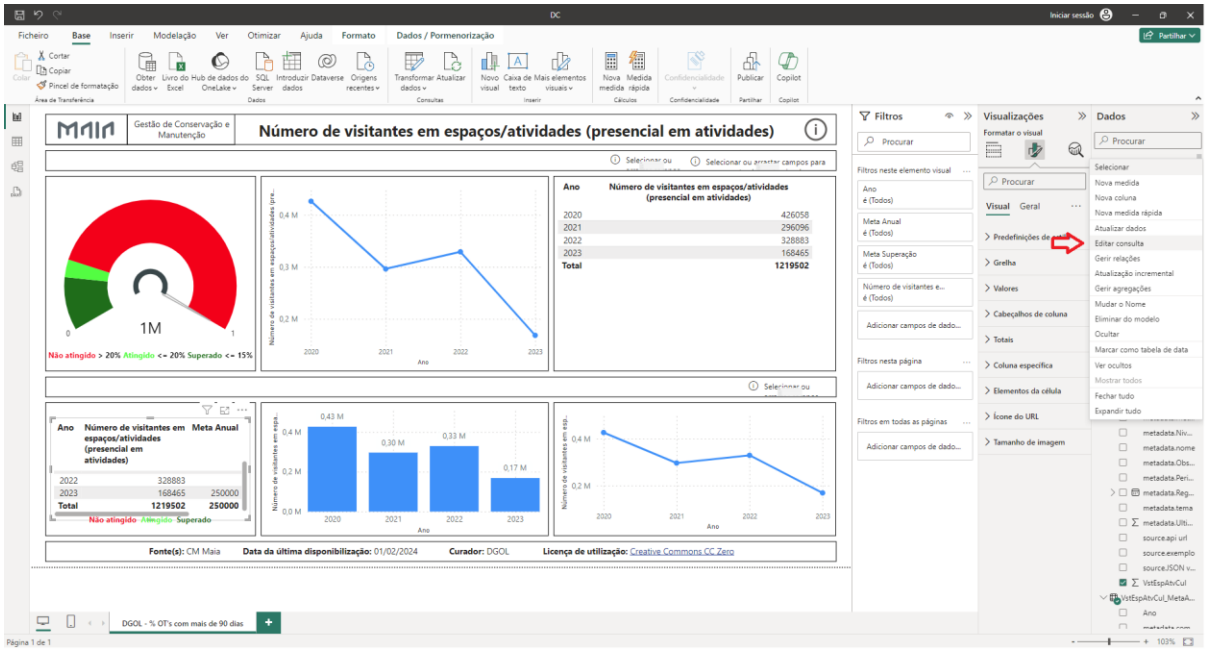


Figura 26. Demonstração de onde é se edita a consulta.

Que irá abrir esta janela onde temos de escolher a opção **Adicionar Coluna de Indíce** e de seguida **Personalizado.**

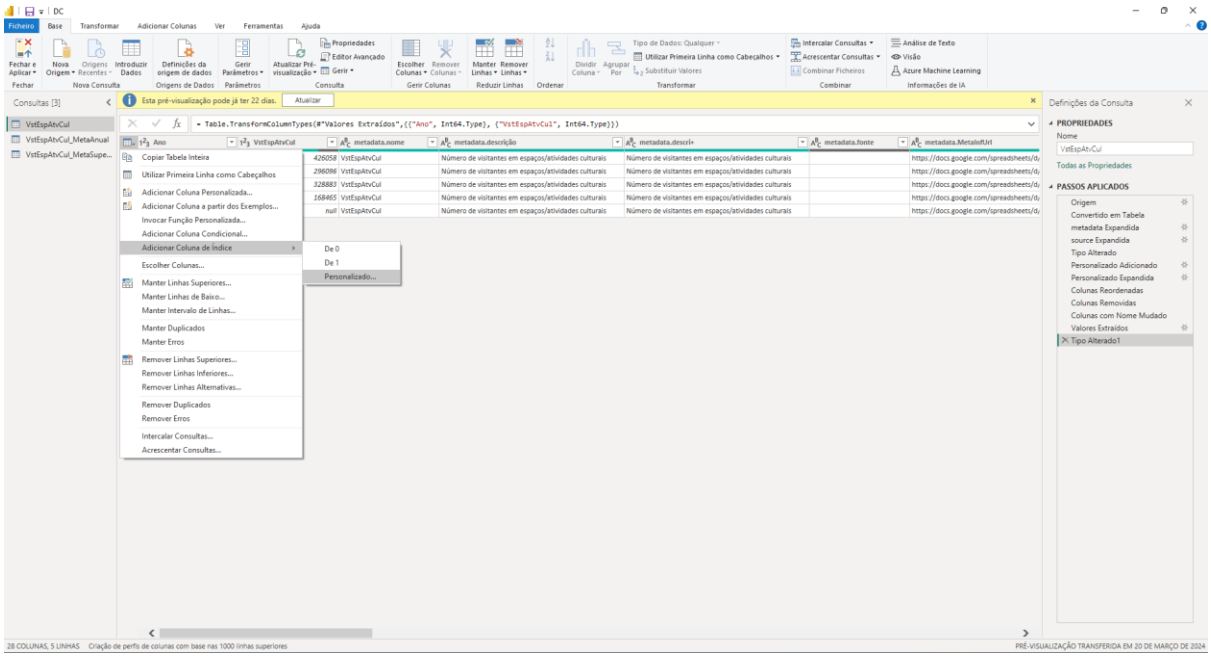


Figura 27. Edição da consulta.

Na caixa que irá aparecer basta preencher com dois 0.

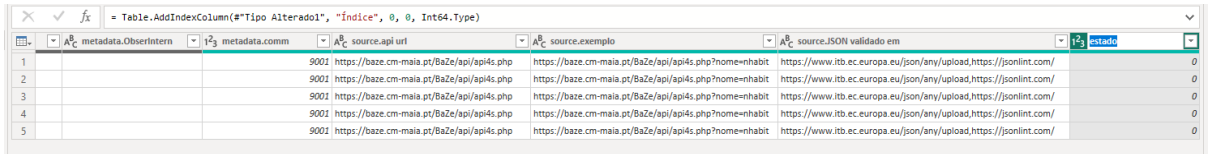


Figura 28. Edição da consulta já realizada.

Que nos irá criar uma nova coluna que renomearemos para “estado”. Após isso, ainda com a coluna selecionada clicamos em “**Transformar**” e “**Substituir valores**” onde iremos colocar **0** e **null**.

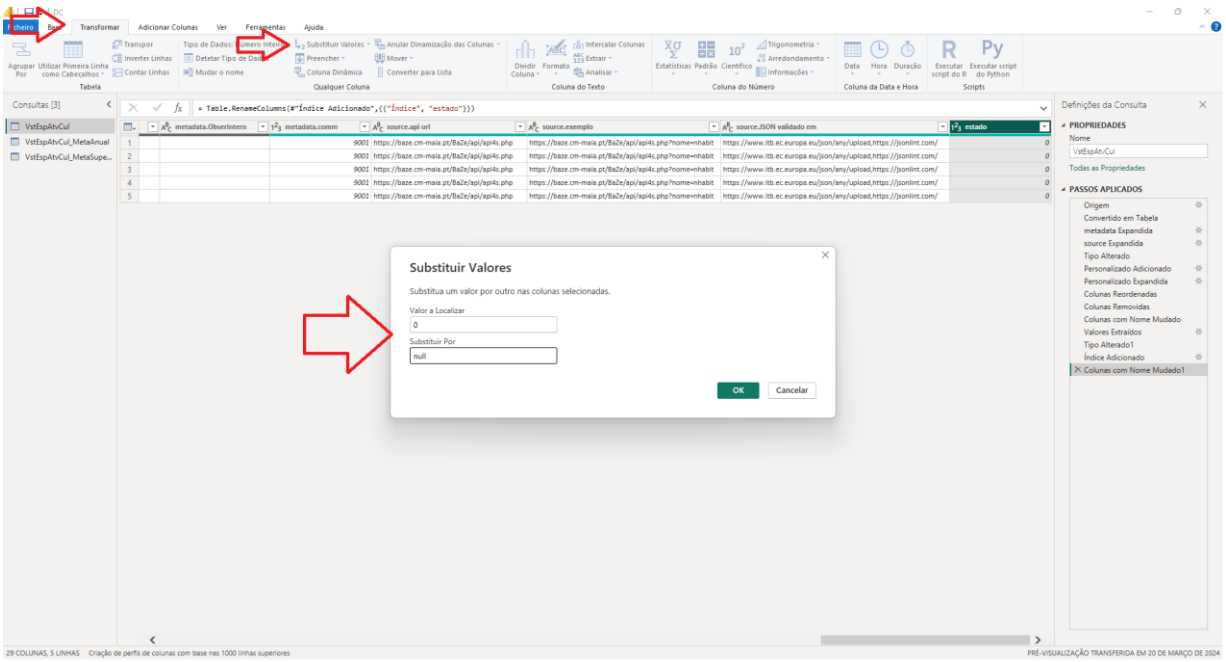


Figura 29. Substituição dos valores na consulta.

Após isso apenas clicamos em “**Fechar e Aplicar**”

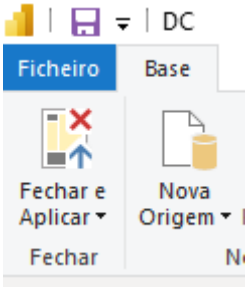


Figura 30. Botão de fechar e aplicar.

Para elaborar o último gráfico, executamos o mesmo procedimento anterior, porém, desta vez, a sequência é: Ano, Meta Anual, Meta Superação, CACptRgd e estado.

Para ativar o estado, precisamos habilitar a cor de fundo e configurá-la com as condições apropriadas conforme as metas. Quando estiver ativado, clicamos em "fx".

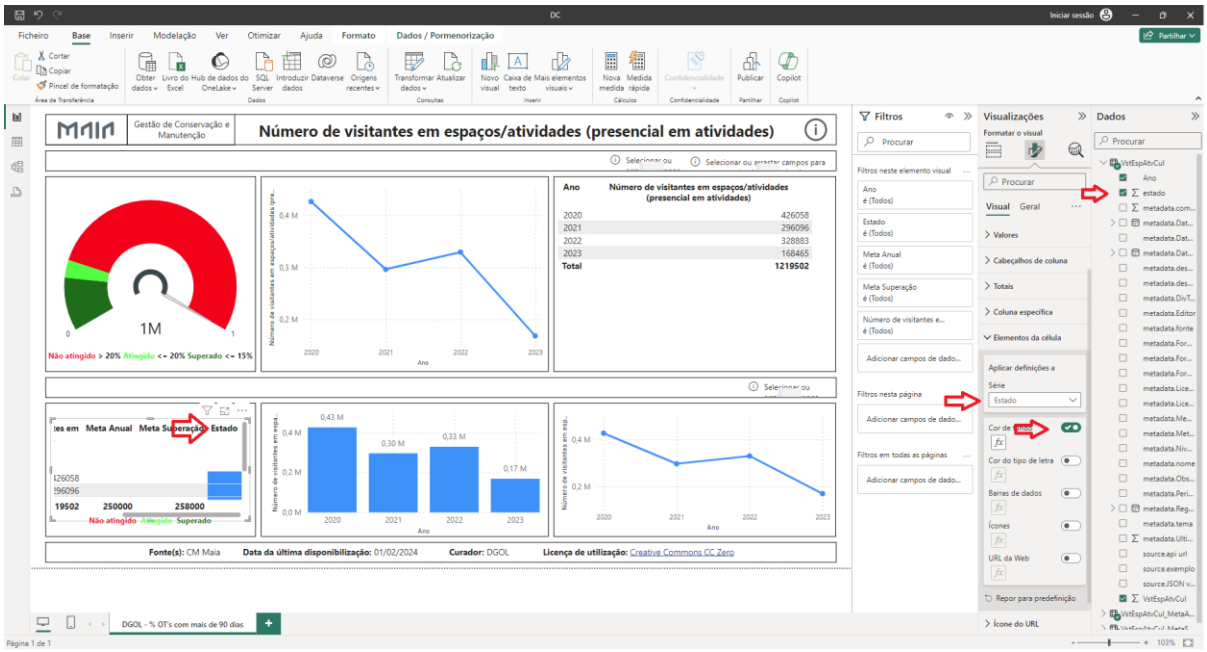


Figura 31. Ativação do estado e cor de fundo do estado.

Na janela que se abre, inserimos as regras de acordo com as metas na lista de indicadores. As cores usadas são: #FF0000 #00FF00 #006E06

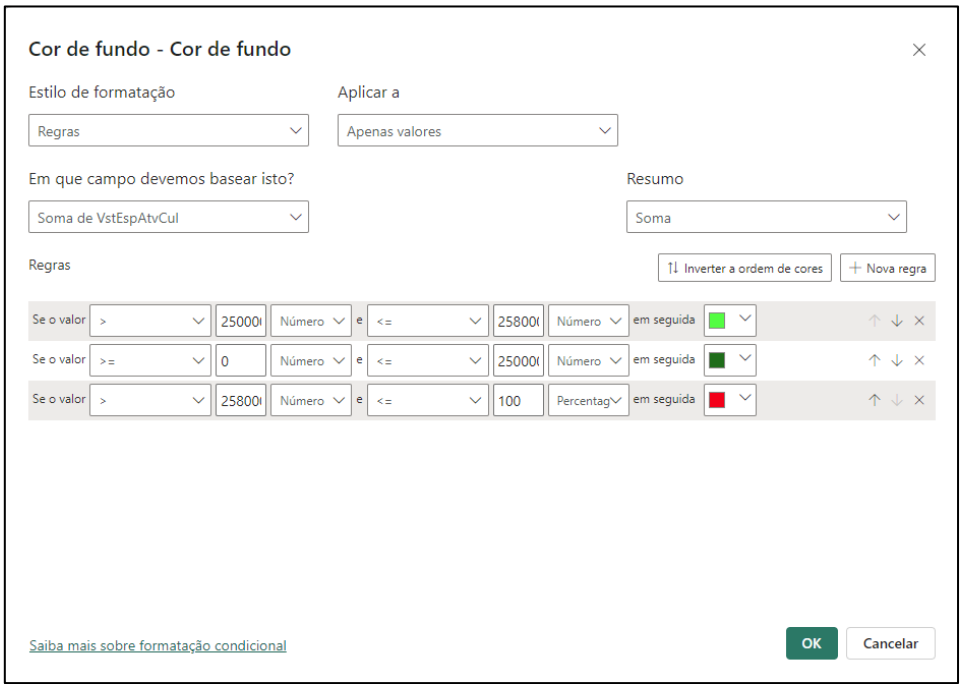


Figura 32. Janela da cor de fundo.

Porquê o Power BI?

O Power BI destaca-se como uma ferramenta de eleição para a criação de dashboards por diversas razões essenciais. A sua capacidade de integrar-se facilmente com várias fontes de dados, oferecer visualizações intuitivas e personalizáveis, e proporcionar análises avançadas garante uma compreensão profunda dos dados. Com atualizações em tempo real, colaboração simplificada e opções flexíveis de implementação, o Power BI apresenta-se como uma escolha ideal para transformar dados em insights acionáveis, fundamentais.

Pesquisa sobre a API de automação da ODS

Comecei a pesquisar sobre a API da automação da Open Data Soft https://help.opendatasoft.com/apis/ods-automation-v1/, visto que obtivemos uma licença de teste para usar essa API e comecei assim a criar um script para atualizar os metadados de uma especifica série e o script começa assim:

```
import requests
import datetime
# from email.mime.text import MIMEText
# from email.mime.multipart import MIMEMultipart
# import smtplib, ssl

maia_api_url = "https://baze.cm-maia.pt/BaZe/api/api4s.php"
ods_api_key = "chave chave chave chave"

ods_headers = {
  "content-type": "application/json",
  "Authorization": "apikey {}".format(ods_api_key),
}

checkable_metadata = [
  {
    'ods': "nome",
    'baze': "nome"
  },
  {
    'ods': "descri",
    'baze': "descrição"
  },
  {
    'ods': "descriplus",
    'baze': "descri+"
  },
  {
    'ods': "tema",
    'baze': "Tema"
  },
  {
    'ods': "formacalculo",
    'baze': "FormaCálculo"
  },
  # {
  #   'ods': "medidaanalise",
  #   'baze': "FormaCálculo"
  # },
  # {
  #   'ods': "origem",
  #   'baze': "origem"
  # },
  # {
  #   'ods': "suporte",
  #   'baze': "suporte"
```

```
# },
{
  'ods': "fonte",
  'baze': "fonte"
},
# {
#   'ods': "tipodefonte",
#   'baze': "fonte"
# },
{
  'ods': "editor",
  'baze': "Editor"
},
{
  'ods': "pcnome",
  'baze': "custom",
  'custom': "Maia - Portal de dados"
},
{
  'ods': "pcemail",
  'baze': "custom",
  'custom': "suporte.dados@cm-maia.pt"
},
# {
#   'ods': "divterrit",
#   'baze': "divterrit",
# },
# {
#   'ods': "divterritgeourl",
#   'baze': "divterritgeourl",
# },
# {
#   'ods': "periodactual",
#   'baze': "periodactual",
# },
{
  'ods': "dataultimaactualocal",
  'baze': "DataUltimaActuaLocal"
},
# {
#   'ods': "primapref",
#   'baze': "primapref",
# },
# {
#   'ods': "ultimopref",
#   'baze': "ultimopref",
# },
# {
#   'ods': "nivelacesso",
#   'baze': "nivelacesso",
# },
{
  'ods': "licenca",
```

```
    'baze': "License"
  },
  {
    'ods': "licencaurl",
    'baze': "LicenseURL"
  },
  {
    'ods': "comm",
    'baze': "comm"
  },
]
```

A variável **checkable_metadata** contém alguns comentários visto que na altura em que escrevo isto não encontrei correspondência para os parâmetros do BaZe com os da Open Data Soft e decidi esperar pela reunião para tirar essas dúvidas.

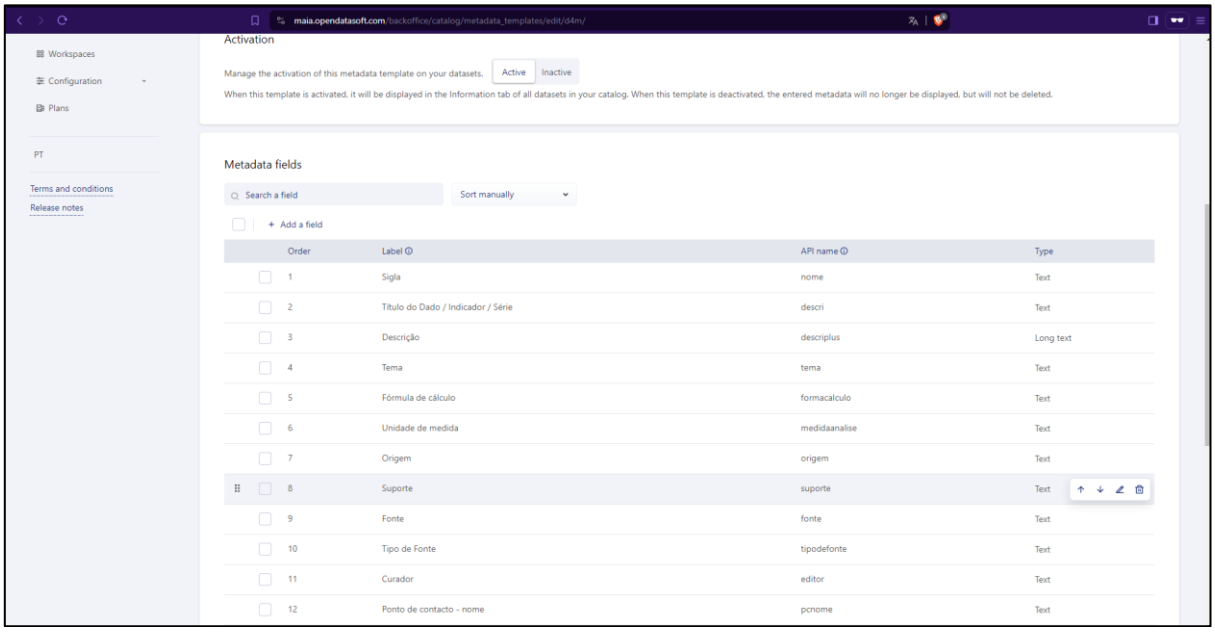


Figura 33. Alguns campos de um template de metadados do Open Data Soft.

Eu peço ao utilizador o código da série que ele pretende atualizar e depois faço um request à API do Baze Maia e para verificar se a série existe verifico se há o parâmetro **sql** no dicionário que o request retorna, se não existir dá uma mensagem de erro.

Depois dou um request à API de automação para obter todos os datasets existentes, porque não consigo procurar pelo nome da série, apenas pelo uid que não tenho como saber qual é, seria melhor se o soubesse claro.

Após isso faço um loop pelos results desse request e verifico se há metadados e se como consequência houver faço faço um split na referência para verificar se o nome da série é igual ao split da referência e guarda os dados desse dataset numa variável..

A referência de um dataset é assim https://baze.cm-maia.pt/BaZe/api/api4sV3.php?nome=AlMat7AnoPubMasc e o que o programa faz é obter o código a seguir ao "**=**".

E agora verifico se de facto encontrou um dataset se não ele retorna um erro a dizer que a série não está registada no Open Data Soft. Para sincronizar os dados utilizo uma função chamada **update_metadata** onde passo como parâmetros os dados da API Baze e os dados do dataset da Open Data Soft para executar uma comparação.

```
dataset_code = input("Código da série: ")
maia_query_url = f'{maia_api_url}?nome={dataset_code}'

maia_api_response = get_json_from_api(maia_query_url)

if 'sql' not in maia_api_response:
  raise ValueError("Código de série inválido!")

print("A procurar...")
ods_catalog_datasets =
get_json_from_api("https://maia.opendatasoft.com/api/automation/v1.0/datasets?limit=700", ods_headers)
dataset_data = {}

for data in ods_catalog_datasets['results']:
  if 'metadata' in data and 'default' in data['metadata'] and 'references' in data['metadata']['default']:
    reference = data['metadata']['default']['references'].get('value')
    if reference is not None and "=" in reference:
      parts = reference.split("=")
      if len(parts) >= 2 and parts[1] == dataset_code:
        dataset_data = data
        print("Encontrado!")
        break

if 'dataset_id' in dataset_data:
  resp = update_metadata(dataset_data, maia_api_response["metadata"])

  if resp.status_code == 200:
    print("Metadados atualizados com sucesso!")
  else:
    print(f"Erro ao atualizar metadata: {resp.status_code}")
else:
  print("Esta série não está registada no Open Data Soft!")
```

Anexo - Código da script metadados data lake -> opendatasoft

```
import requests
import re
import time

maia_api_url = "https://baze.cm-maia.pt/BaZe/api/api4sV3.php"
ods_api_key = "chave chave chave chave"

ods_headers = {
    "content-type": "application/json",
    "Authorization": "Apikey {}".format(ods_api_key),
}


# Obtém o JSON de qualquer URL
def get_json_from_api(url: str, headers=None):
    try:
        resp = requests.get(url, headers=headers)

        if resp.status_code == 200:
            return resp.json()
        else:
            print("Failed to fetch data from API. Status code:", resp.status_code)
            return None
    except requests.exceptions.RequestException as e:
        print("Error:", e)
        return None
# Obtém os datasets criados e faz uma array com os códigos dos datasets criados
def get_registered_datasets():
    ods_dataset_codes = []
    ods_datasets = get_json_from_api(
        "https://maia.opendatasoft.com/api/automation/v1.0/datasets?limit=700",
        ods_headers,
    )

    for data in ods_datasets:
title = data["dataset"]["metas"]["default"]["title"]
        reference = data["dataset"]["metas"]["default"]["references"]

        if reference is not None:
            parts = reference.split("=")

            if len(parts) >= 1:
                code = parts[1]
                ods_dataset_codes.append(code)
            else:
                print("Não é possivel identificar o código '=':", reference)
        else:
            print(title, "não tem referência por isso foi avançado.")
```

```
    return ods_dataset_codes


# Função para criar um id para cada dataset, converte um titulo em um id sem caracteres especiais, usado na URL
def convert_to_slug(string: str):
    # Substitui caracteres especiais por hífens
    slug = re.sub(r"[^a-zA-Z0-9]", "-", string)
    # Converte para minúsculas
    slug = slug.lower()
    # Remove hífens duplicados
    slug = re.sub(r"-+", "-", slug)
    # Remove hífens no início e no fim
    slug = slug.strip("-")
    return slug


def post_dataset(url: str, params=None):
    if params == None:
        return False

    try:
        resp = requests.post(url, headers=ods_headers, params=params)

        if resp.status_code == 200 or resp.status_code == 201:
            return True
        else:
            print("Failed to post dataset. Status code:", resp.status_code)
            return False
    except requests.exceptions.RequestException as e:
        print("Error:", e)
        return False

# Obtém os datasets registados na plataforma
ods_registered_datasets = get_registered_datasets()

# Obtém os dados da API BaZe
api_data = get_json_from_api(maia_api_url)
create_dataset_url = "https://maia.opendatasoft.com/api/automation/v1.0/datasets/"
created_datasets = []

if api_data:
    # Se obter os dados da API BaZe faz um loop de 1 em 1 segundo para não sobrecarregar as APIs
    for data in api_data["o"]:
        # time.sleep(1)

        code = data[0]
        theme = data[7]

        if code == "Código":
            continue
```

```
if code in ods_registered_datasets:
    continue

if code in created_datasets:
    continue

print("\nA criar o dataset " + code + " com o tema " + theme)

code_api_url = maia_api_url + "?nome=" + code
code_data = get_json_from_api(code_api_url)["d4maia"]

new_dataset_title = code_data["metadata"]["descrição"]
dataset_id = convert_to_slug(new_dataset_title)

new_dataset_data = {
    "dataset_id": dataset_id,
    "is_restricted": True,
    "metadata": {
        "default": {
            "title": {
                "value": new_dataset_title,
            },
            "description": {
                "value": code_data["metadata"]["descri+"],
            },
            "keyword": {
                "value": [],
            },
            "modified": {
                "value": "2023-01-01T00:00:00Z",
            },
            "modified_updates_on_metadata_change": {
                "value": True,
            },
            "modified_updates_on_data_change": {
                "value": True,
            },
            "language": {
                "value": "pt",
                "remote_value": "pt",
                "override_remote_value": False,
            },
            "timezone": {
                "value": "Europe/Lisbon",
                "remote_value": "Europe/Lisbon",
                "override_remote_value": False,
            },
            "publisher": {
                "value": "CMMaia",
                "remote_value": "CMMaia",
                "override_remote_value": False,
            },
            "references": {
```

```
                "value": code_api_url,
                "remote_value": code_api_url,
                "override_remote_value": False,
            },
            "attributions": {
                "value": ["Creative Commons CCZero"],
                "remote_value": ["Creative Commons CCZero"],
                "override_remote_value": False,
            },
        },
        "internal": {
            "theme_id": {
                "value": [theme],
                "remote_value": [theme],
                "override_remote_value": False,
            },
        },
    },
    "default_security": {
        "is_data_visible": True,
        "visible_fields": [],
        "filter_query": "",
        "api_calls_quota": {"unit": "month", "limit": 12000},
    },
}

if post_dataset(create_dataset_url):
    print("Dataset com o id " + dataset_id + " foi criado.")

publish_dataset_url = (
    "https://maia.opendatasoft.com/api/automation/v1.0/datasets/"
    + dataset_id
    + "/publish/"
)

if post_dataset(publish_dataset_url):
    print("Dataset com o id " + dataset_id + " foi publicado.")

created_datasets.append(code)

print("\n\nForam criados " + str(len(created_datasets)) + " datasets com sucesso!")
```