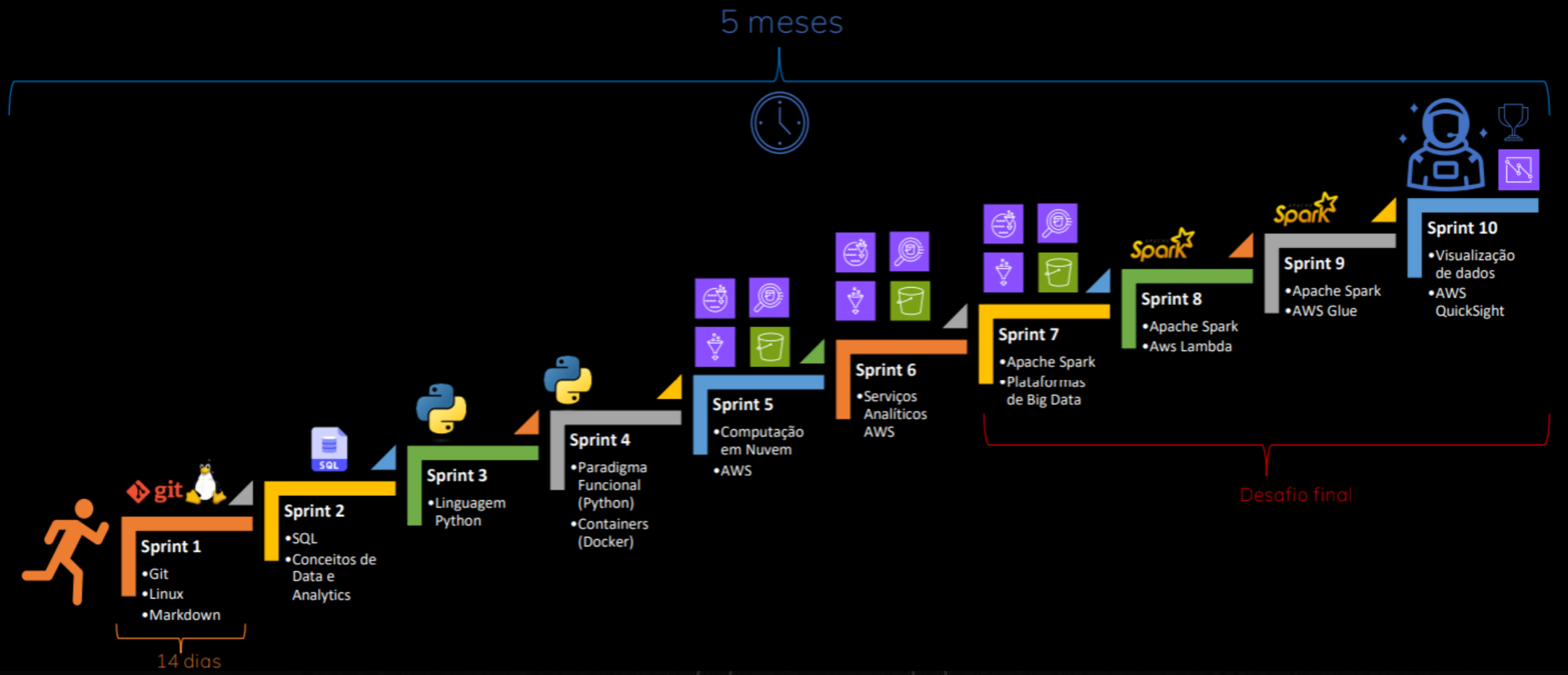
EXPLORANDO O UNIVERSO CINEMATOGRÁFICO:

Crime em Destaque

Kesley Wilie Costa Santos

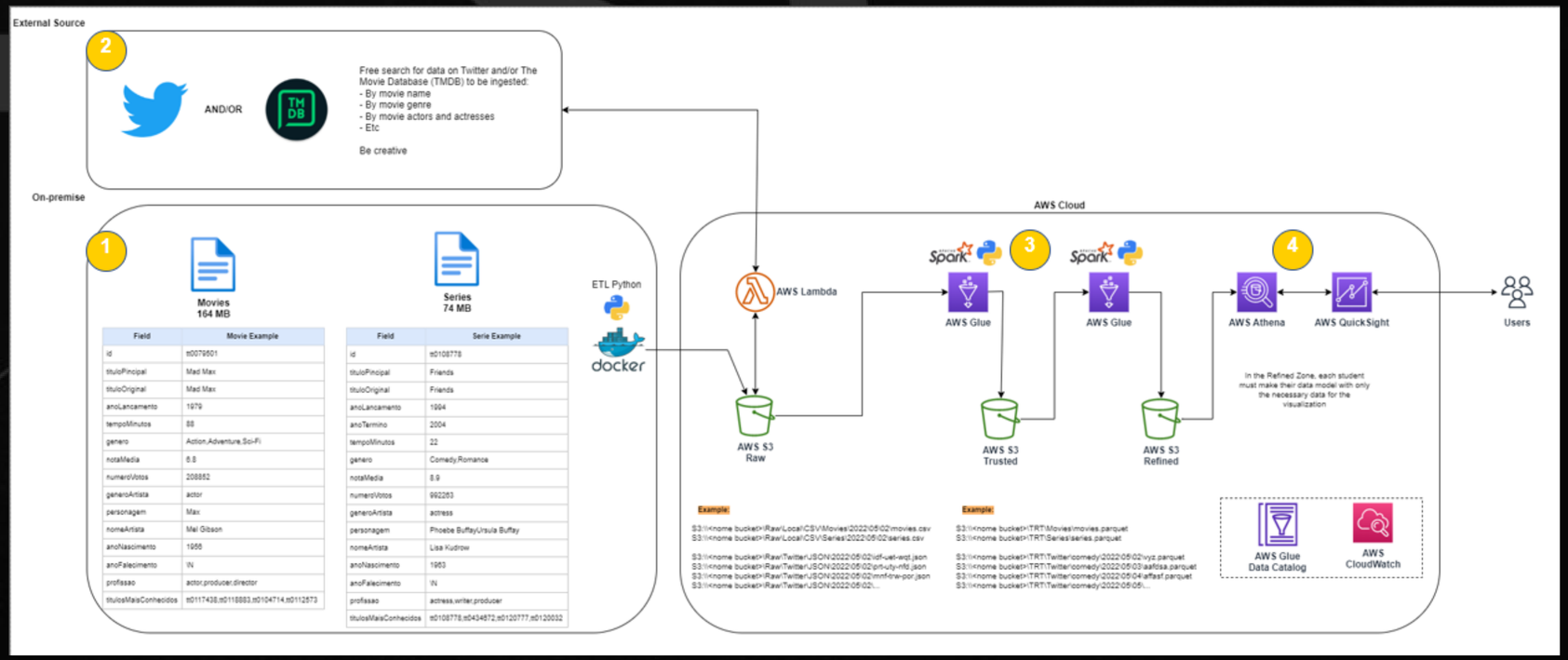
CONTEÚDOS


5 meses
Sprint 1
•Git
•Linux
•Markdown
14 dias
Sprint 2
•SQL
•Conceitos de Data e Analytics
Sprint 3
•Linguagem Python
Sprint 4
•Paradigma Funcional (Python)
•Containers (Docker)
Sprint 5
•Computação em Nuvem
•AWS
Sprint 6
•Serviços Analíticos AWS
Sprint 7
•Apache Spark
•Plataformas de Big Data
Sprint 8
•Apache Spark
•Aws Lambda
Sprint 9
•Apache Spark
•AWS Glue
Sprint 10
•Visualização de dados
•AWS QuickSight
Desafio final

Desafio

Passos:

- Processo ETL
- Coleta de dados (API TMDB)
- Limpeza de dados (TRUSTED)
- Tratamento de dados (REFINED)
- Análise de dados (DASHBOARD)



Dados da API (TMDB)

```python
# Código para obter os dados da API
# Substitua 'sua_chave_de_api' pela sua chave real da API
# Certifique-se de ter instalado a biblioteca 'requests' antes de executar este código

import requests
import json
from pyspark.sql import SparkSession

# Sua chave de API TMDB
api_key = 'sua_chave_de_api'

# Endpoint da API TMDB para os top filmes de crime (exemplo)
endpoint = f'https://api.themoviedb.org/3/discover/movie?api_key={api_key}&with_genres=80&sort_by=popularity.desc'

# Obtendo dados da API
response = requests.get(endpoint)
data = response.json()

# Transformando os dados em um DataFrame Spark
spark = SparkSession.builder.appName("Filmes").getOrCreate()
df_api = spark.createDataFrame([json.dumps(data)])

# Exibindo os dados obtidos
df_api.show(truncate=False)
```

Limpeza de dados (Trusted)

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

# Inicializa a sessão Spark
spark = SparkSession.builder.appName("Filmes").getOrCreate()

# Lista dos anos para os quais você tem arquivos JSON
anos = list(range(2000, 2024))

# Caminho de origem dos arquivos JSON (substitua pelo seu caminho real)
caminho_origem_json = "s3://SEU CAMINHO DE ORIGEM"

# Lista para armazenar os DataFrames lidos de cada ano
dataframes_por_ano = []

# Leitura dos arquivos JSON e armazenamento em uma lista de DataFrames
for ano in anos:
    nome_arquivo = f"filmes({ano})_crime_aclamados(1).json"
    df = spark.read.option("multiline", "true").json(f"{caminho_origem_json}{nome_arquivo}")
    dataframes_por_ano.append(df)

# Concatenação de todos os DataFrames em um único DataFrame
df_final = dataframes_por_ano[0]
for df in dataframes_por_ano[1:]:
    df_final = df_final.union(df)

# Remoção de linhas com valores nulos em 'id_imdb', 'receita', 'total_votos' e 'media_de_votos'
colunas_para_verificar_nulos = ['id_imdb', 'receita', 'total_votos', 'media_de_votos']
df_sem_nulos = df_final.filter(
    (col("id_imdb") != "null") & (col("receita").isNotNull()) & (col("total_votos") != 0) & (col("media_de_votos").isNotNull())
)

# Caminho de destino no Amazon S3
caminho_destino = "s3://SEU CAMINHO DE DESTINO"

# Escreve o DataFrame em formato Parquet no caminho de destino
df_sem_nulos.coalesce(1).write.parquet(caminho_destino, mode="overwrite")

# Exibe os resultados
print("Total de registros após a filtragem:", df_sem_nulos.count())
df_sem_nulos.show()
```

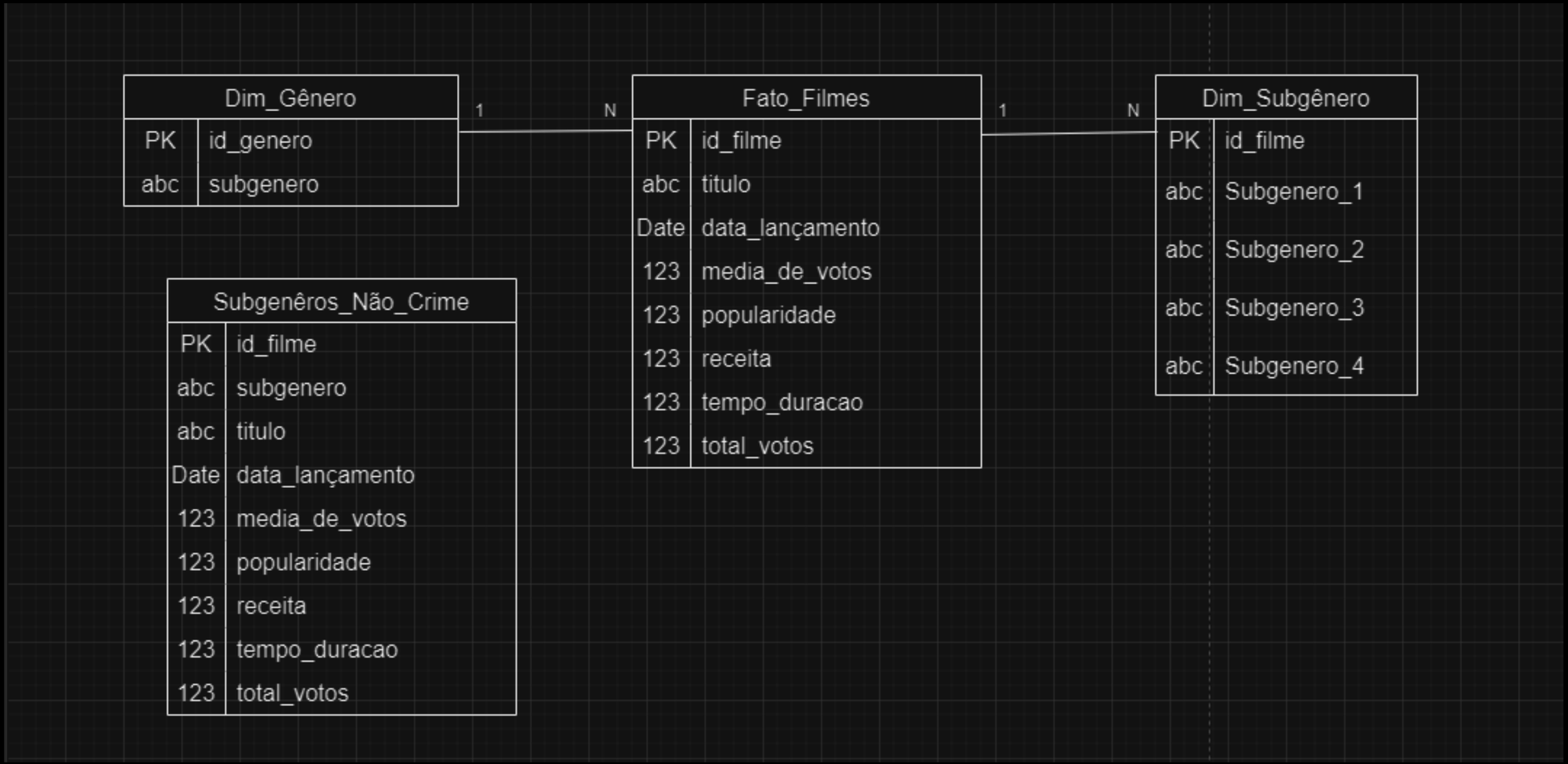
Refinamentos de Dados (Refined)

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode, dense_rank, col, lit, when, collect_list, size
from pyspark.sql.window import Window

# Inicializa a sessão Spark
spark = SparkSession.builder.appName("Filmes").getOrCreate()

# Caminho para a camada Trusted
arquivo_parquet_trusted = "s3://NOME DO SEU CAMINHO"

# Carrega os dados da camada Trusted
df = spark.read.parquet(arquivo_parquet_trusted)

# Explode os gêneros
dataframe_genero_explodido = df.withColumn("genero", explode("generos"))

# Adiciona IDs aos gêneros
dataframe_generos_com_id = dataframe_genero_explodido.withColumn("id_genero", dense_rank().over(Window.orderBy("genero")))

# Cria a dimensão de gênero/subgênero
dim_genero = dataframe_generos_com_id.select("id_genero", col("genero").alias("subgenero")).distinct()

# Cria a dimensão de filme
fato_filmes = df.select("id_filme", "titulo", "data_lancamento", "media_de_votos", "popularidade", "receita", "tempo_duracao(minutos)", "total_votos").distinct()

# Realiza o join entre o dataframe original e a dimensão de gênero para obter os IDs correspondentes
dim_genero = dataframe_generos_com_id.join(
    dim_genero,
    (dataframe_generos_com_id['genero'] == dim_genero['subgenero']) & (dataframe_generos_com_id['id_genero'] == dim_genero['id_genero']),
    'left_outer'
).select(
    dataframe_generos_com_id['id_filme'],
    "subgenero",
    "data_lancamento",
    "media_de_votos",
    "popularidade",
    "receita",
    "tempo_duracao(minutos)",
    "total_votos",
    "titulo"
)
```

Refinamentos de Dados (Refined)

```python
# Adiciona coluna indicando se é um subgênero (excluindo "Crime")
dim_genero = dim_genero.withColumn("subgenero", when(col("subgenero") != "Crime", col("subgenero")).otherwise(None))

# Agrupa por filme e coleta os subgêneros
dim_subgenero = dim_genero.groupBy("id_filme").agg(
    collect_list("subgenero").alias("subgeneros")
)

# Extraindo até 4 subgêneros distintos para cada filme
for i in range(1, 5):
    dim_subgenero = dim_subgenero.withColumn("Subgenero_" + str(i), when(size("subgeneros") >= i, col("subgeneros")[i - 1]).otherwise(None))

# Inclui os dados dos subgêneros não relacionados a "Crime"
subgeneros_nao_crime = dataframe_genero_explodido.filter(col("genero") != "Crime")
subgeneros_nao_crime = subgeneros_nao_crime.select(
    "id_filme",
    col("genero").alias("subgenero"),  # Renomeia a coluna para subgenero
    "titulo",
    "data_lancamento",
    "media_de_votos",
    "popularidade",
    "receita",
    "tempo_duracao(minutos)",
    "total_votos"
)

# Caminho para a camada Refined
caminho_refined = "s3://SUA SAÍDA"

# Salva as tabelas também em formato CSV
dim_genero.write.csv(caminho_refined + "dim_genero_CSV", header=True, mode="overwrite")
fato_filmes.write.csv(caminho_refined + "fato_filmes_CSV", header=True, mode="overwrite")
dim_subgenero.write.csv(caminho_refined + "dim_subgenero_CSV", header=True, mode="overwrite")
subgeneros_nao_crime.write.csv(caminho_refined + "subgeneros_nao_crime_CSV", header=True, mode="overwrite")

# Exibe os dados
dim_subgenero.show()
dim_genero.show()
fato_filmes.show()
subgeneros_nao_crime.show(truncate=False)
```

Modelo Dimensional



ANÁLISE DE DADOS

DASHBOARD IN QUICKSIGHT

compass.uol

Muito Obrigado