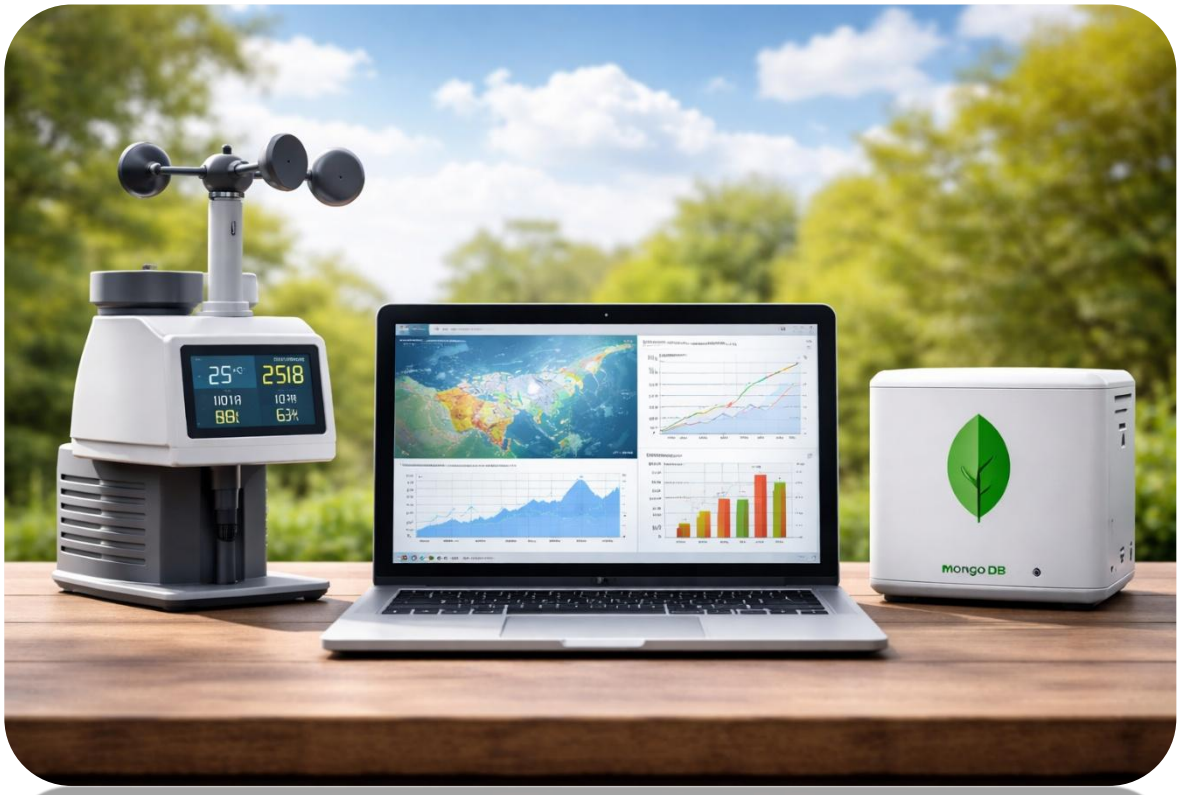
Politécnico de Tomar
Escola Superior de Tecnologia de Abrantes



Predição Meteorológica

https://meteoabrantes.lcerejo.com

Politécnico de Tomar
Escola Superior de Tecnologia de Abrantes

Escola Superior de Tecnologia de Abrantes

Meteo Abrantes:

Real vs Previsto

CTeSP Informática 2º Ano, 1º Semestre

Luís Cerejo, nº 82674

Unidade Curricular: Projeto Integrado

Docente: Paulo Rêgo

Abrantes, 29 de Dezembro de 2025

AGRADECIMENTOS

Gostaria de expressar o meu sincero agradecimento ao Professor Paulo Rêgo pelo acompanhamento, disponibilidade e orientação ao longo do desenvolvimento deste projeto. As sugestões técnicas e conceptuais apresentadas, nomeadamente a introdução e recomendação de aplicações e ferramentas que vieram enriquecer significativamente a solução desenvolvida, foram determinantes para a evolução do projeto. Sem essas indicações, muitas das funcionalidades e abordagens adotadas não teriam sido exploradas. Este contributo revelou-se essencial para o aprofundamento dos meus conhecimentos, tanto a nível técnico como metodológico, resultando num trabalho mais consistente, estruturado e com maior valor tanto académico como prático.

RESUMO

A ideia subjacente ao projeto AbrantesMeteo consistiu no desenvolvimento de uma plataforma integrada para a recolha, previsão e a análise de dados meteorológicos na cidade de Abrantes. Combinou-se dados históricos importados de uma base de dados externa, medições em tempo real e utilização de modelos preditivos (prophet), com visualização centralizada em dashboards produzidos em Power BI. O objetivo principal é demonstrar, em contexto puramente académico, a aplicação prática de tecnologias de IoT, bases de dados NoSQL, modelação preditiva e Business Intelligence, assegurando a atualização automática dos dados, bem como a sua consistência ao longo do tempo.

Na fase inicial, o sistema recolhe os dados meteorológicos históricos e diários através da API Open-Meteo, armazenando-os numa base de dados MongoDB em ambiente Docker (modo local). Estes dados incluem os registos de temperatura mínima, máxima e média, bem como a precipitação diária registada, constituindo a base histórica do projeto desde 1 de janeiro de 2023. A atualização ocorre diariamente de forma automática, garantindo a continuidade temporal da informação.

Foi sobre esta base de dados históricos que se implementou um módulo de previsão meteorológica em Python, recorrendo ao modelo Prophet, que gera previsões diárias para um horizonte temporal definido para 365 dias. As previsões são armazenadas separadamente dos dados reais, permitindo análises comparativas entre valores observados e valores previstos, bem como a avaliação de tendências futuras.

O projeto foi posteriormente expandido com a integração de um Raspberry Pi 5 equipado com um sensor BME280, permitindo a recolha de dados físicos reais de temperatura, humidade e pressão atmosférica. Estas medições locais acrescentam uma componente experimental ao sistema e são igualmente armazenadas localmente em MongoDB, possibilitando visualizações em “tempo real” no Power BI.

Para garantir a atualização automática dos dashboards no Power BI Service, os dados são sincronizados para o MongoDB Atlas e expostos através de Atlas SQL (ODBC), solução compatível com o On-premises Data Gateway. Desta forma, todo o ecossistema utiliza uma arquitetura consistente e robusta, evitando limitações de conectividade direta ao MongoDB.

Em resumo, o projeto AbrantesMeteo apresenta uma solução completa e escalável, que integra recolha automática de dados, previsão estatística e visualização analítica, evidenciando técnicas de engenharia de dados, automação e análise preditiva, com uma abordagem alinhada com práticas profissionais.

Índice

1. Introdução

O projeto AbrantesMeteo é apresentado como um caso de estudo que visa demonstrar uma arquitetura pensada e implementada para a recolha, tratamento e disponibilização de dados, aplicável a qualquer domínio (operações, produção, vendas, etc.) - e não somente o tratamento de dados meteorológicos. O objetivo principal é validação, em contexto académico, de uma cadeia de dados de ponta a ponta: ingestão de fontes heterogéneas, normalização e validação, armazenamento estruturado, criação de métricas e modelos (quando aplicável) e, por fim, consumo em dashboards para análise e decisão.

A componente meteorológica pretende recriar um cenário realista para testar requisitos típicos de um pipeline de informação: automatização, repetibilidade, auditabilidade (logs e rastreio de execuções), separação de responsabilidades (módulos e serviços independentes) e qualidade dos dados (consistência temporal, controlo de duplicados, tratamento de falhas). O que se pretende, mais do que "fazer gráficos e tabelas informativos", é evidenciar um processo operacional robusto e funcional que pode ser reutilizado e escalado para outros contextos.

Ao longo do relatório são descritos os objetivos e requisitos, a arquitetura e tecnologias adotadas, a implementação passo a passo (incluindo organização de ficheiros/serviços), as principais dificuldades técnicas e respetivas correções, os resultados obtidos no Power BI e um conjunto de melhorias futuras orientadas para maior resiliência, escalabilidade e manutenção da solução.

---

2. Objetivos e critérios de sucesso

- Recolher histórico meteorológico (Open-Meteo archive) e armazenar em MongoDB local uma vez por dia.
- Gerar previsões diárias (365 dias) com Prophet, normalizando o output em formato "flat" para análise tabular.
- Sincronizar automaticamente para MongoDB Atlas, assegurando autenticação, segregação de utilizadores e permissões mínimas.
- Expor os dados ao Power BI via Atlas Data Federation/Atlas SQL (ODBC), evitando dependência de drivers "MongoDB nativos" no Power BI.
- Construir um modelo de dados no Power BI com tabela de calendário, medidas DAX e visualizações para comparação real vs. previsto.

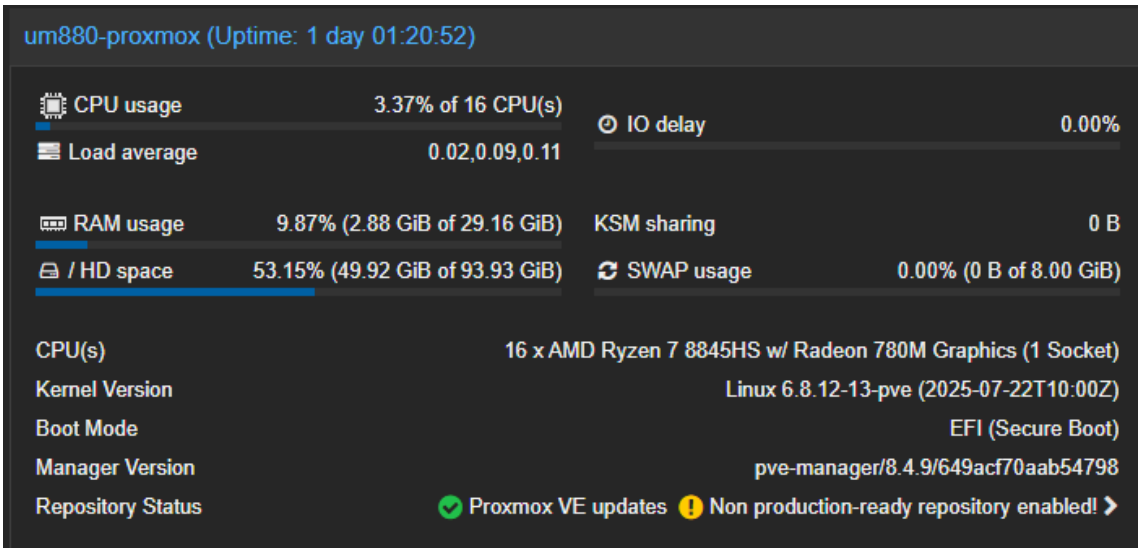
- Garantir atualização diária e consistência do baseline de previsão (forecast_fixo) para comparar ‘o que estava previsto’ com o que aconteceu.

---

3. Ambiente de execução e tecnologias

3.1. Infraestrutura

- Servidor: MiniPC UM880 (host Proxmox).
- Execução: contentor LXC dedicado (“docker-ct”) a correr Docker Engine e Docker Compose.
- Cliente analítico: PC Windows (Power BI Desktop) e, opcionalmente, Power BI Service com On-premises Data Gateway (Personal Mode).



3.2. Arquitetura tecnológica

- MongoDB 7 (container) para armazenamento local e testes/validação.
- MongoDB Atlas (cluster) para persistência cloud e consumo analítico.
- Atlas Data Federation + Atlas SQL para exposição SQL/ODBC.
- Python 3.11 para ingestão, normalização e previsão.
- Prophet (cmdstanpy) para forecasting; requests/pymongo/pandas para integração e transformação.
- Power BI Desktop para modelação, DAX e dashboards; ODBC para ligação ao Atlas SQL.

4. Arquitetura lógica e fluxo de dados

A solução proposta foi projetada como um pipeline modular. A ingestão e a previsão correm no servidor, junto ao armazenamento local, garantindo execução diária e independência do posto de trabalho. A cloud (Atlas) é usada para centralizar e disponibilizar os dados, permitindo que o Power BI aceda de forma consistente (ODBC) e que o refresh seja operacionalizado via Gateway quando necessário.

Fonte (Open-Meteo)
→ Serviço Python (ingestão + Prophet)
→ MongoDB local (abrantes_meteo)
→ Sync (export/import)
→ MongoDB Atlas
→ Data Federation/Atlas SQL
→ ODBC (DSN)
→ Power BI (Modelo + Medidas + Dashboards)

5. Estrutura do projeto no servidor (docker-ct)

A implementação foi organizada numa pasta dedicada dentro do RAID/volume do servidor, assegurando persistência dos dados (mongo-data) e separação entre código, logs e segredos. A estrutura abaixo corresponde à pasta /mnt/storage/abrantes_meteo.

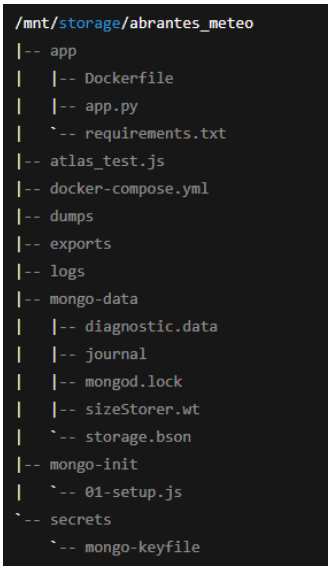


Figura 1 - Estrutura de pastas do projeto no servidor (diretório /mnt/storage/abrantes_meteo).

5.1. Função dos diretórios e ficheiros

a) *docker-compose.yml*

Define os serviços Docker do projeto:

- mongodb (MongoDB 7) com --replSet rs0, --auth e --keyFile (requisito técnico para auth + replicaset).
- mongo_setup (container temporário) que executa mongo-*init/01-setup.js* após o Mongo estar saudável.
- app (Python) que faz backfill, previsão, sync e scheduler.

```yaml
services:
  mongodb:
    image: mongo:7
    container_name: mongodb
    restart: unless-stopped
    ports:
      - "27017:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: "lcerejo"
      MONGO_INITDB_DATABASE: projeto
    volumes:
      - ./mongo-data:/data/db
      - ./secrets/mongo-keyfile:/etc/mongo-keyfile:ro
    command: ["--bind_ip_all","--replSet","rs0","--keyFile","/etc/mongo-keyfile","--auth"]
    healthcheck:
      test: ["CMD-SHELL", "mongosh --quiet --eval 'db.adminCommand({ping:1})' || exit 1"]
      interval: 10s
      timeout: 5s
      retries: 12

  meteo_cron:
    image: python:3.11-slim
    container_name: meteo_cron
    restart: unless-stopped
    depends_on:
      mongodb:
        condition: service_healthy
    working_dir: /app
    volumes:
      - ./scripts:/app
    environment:
      TZ: "Europe/Lisbon"
      LAT: "39.46"
      LON: "-8.2"
      RUN_HOUR_LOCAL: "10"
      LOCAL_URI: "mongodb://app:lcerejo@mongodb:27017/projeto?replicaSet=rs0&authSource=projeto&authMechanism=SCRAM-SHA-256"
    command:
      - /bin/sh
      - -c
      - |
        pip install --no-cache-dir requests pymongo &&
        python /app/fetch_meteo.py --daemon ;
        tail -f /dev/null
```

Figura 2 - Ficheiro docker-compose.yml do projeto (serviços e volumes definidos).

b) *secrets/mongo-keyfile*

Keyfile usado pelo MongoDB para permitir replicaset com autenticação ativa. É gerado localmente com permissões restritas (chmod 400) e montado read-only no container.

c) *mongo-init/01-setup.js*

Script idempotente que:

- inicializa replicaset (rs0) se necessário,
- cria a base abrantes_meteo e o utilizador de aplicação (app) com readWrite,
- cria índices essenciais (ex.: meteo_historico.date único; forecast indexado por run_id + date e por created_at).

d) app/Dockerfile e app/requirements.txt

Constroem a imagem da aplicação:

- Base python:3.11-slim,
- Instala dependências (requests, pymongo, pandas, prophet, cmdstanpy),
- Define app.py como entrypoint.

e) app/app.py

É o núcleo do sistema e implementa:

- **Recolha histórica** via Open-Meteo Archive (por blocos, para estabilidade),
- **Upsert** diário (não duplica dias e mantém consistência),
- **Treino e previsão** com Prophet para as variáveis meteorológicas,
- Escrita do forecast com run_id diário e created_at,
- **Sync para Atlas** (histórico por janela lookback + forecast do run_id atual),
- **Scheduler**: corre uma vez no arranque e depois diariamente na hora definida.

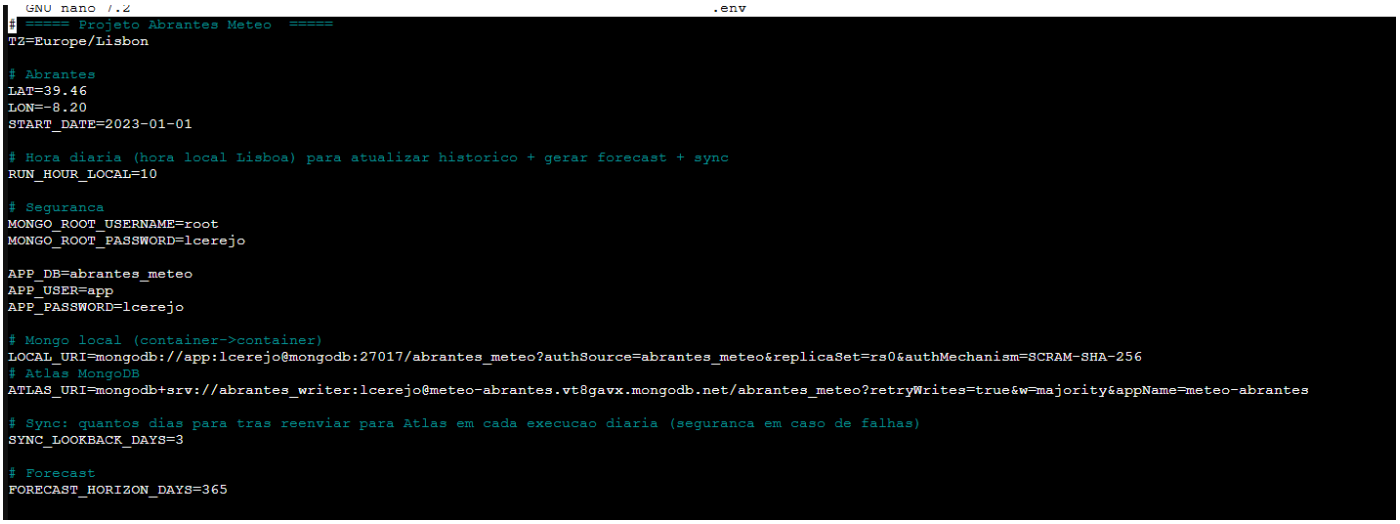


Figura 3 - Variáveis de ambiente no ficheiro .env (fuso horário e hora de execução do scheduler).

f) mongo-data/ (persistência da base de dados local)

Esta pasta é um bind mount para /data/db do container MongoDB, garantindo persistência após reinícios.

Os ficheiros internos (ex.: journal/, sizeStorer.wt, mongod.lock, diagnostic.data, storage.bson) são artefactos normais do motor WiredTiger, e a sua gestão é feita pelo MongoDB.

Esta decisão assegura:

- retenção do histórico e forecasts locais,
- recuperação rápida após falhas de energia,

- independência do Atlas para o funcionamento base (Atlas é consumo/sync).

logs/, exports/, dumps/

Pastas operacionais:

- logs/: evidência operacional (logs de execução e troubleshooting).
- exports/: suporte a exportações para ficheiros (quando necessário para auditoria/partilha).
- dumps/: suporte a backups com mongodump/mongorestore.
  (Em contexto académico, são úteis como "instrumentação" e preparação para operação em produção.)

g) atlas_test.js

Script de teste para validar conectividade e queries no Atlas via mongosh, garantindo que:

- credenciais e URI estão corretos,
- coleções existem e estão acessíveis,
- o ambiente está operacional para o Power BI.

6. Implementação detalhada

Esta secção pretende descrever o processo de implementação de uma forma sequencial. Sempre que foram necessárias decisões de configuração, as mesmas são justificadas tecnicamente, indicando também os problemas encontrados e a respetiva resolução.

6.1. Preparação do diretório e ficheiros base

O ponto de partida foi a criação da pasta de trabalho no servidor e a definição da estrutura mínima (app, mongo-init, secrets, volumes). O objetivo é permitir que todo o ciclo de vida (deploy, restart, upgrades) seja controlado por Docker Compose.

```
# (no docker-ct)
mkdir -p /mnt/storage/abrantes_meteo/{app,mongo-init,secrets,mongo-
data,logs,exports,dumps}
cd /mnt/storage/abrantes_meteo
```

```
env (exemplo)
TZ=Europe/Lisbon
LAT=39.46
```

```
LON=-8.20
START_DATE=2023-01-01
RUN_HOUR=10

# Mongo local
MONGO_DB=abrantes_meteo
MONGO_USER=app
MONGO_PASS=<LOCAL_PASSWORD>
LOCAL_URI=mongodb://app:<LOCAL_PASSWORD>@mongodb:27017/abrantes_meteo?authSource=abrantes_meteo

# Mongo Atlas (writer/reader)
ATLAS_URI=mongodb+srv://abrantes_writer:<ATLAS_PASSWORD>@meteo-abrantes.<cluster>.mongodb.net/abrantes_meteo?retryWrites=true&w=majority&appName=meteo-abrantes
```

6.2. Ficheiros de configuração essenciais

Para manter a solução reprodutível e segura, as credenciais e parâmetros são isolados em variáveis de ambiente (.env) e/ou secrets. No relatório, as passwords são representadas por placeholders – por uma questão de boas práticas é recomendado a não inclusão de segredos em documentos académicos nem em repositórios públicos.

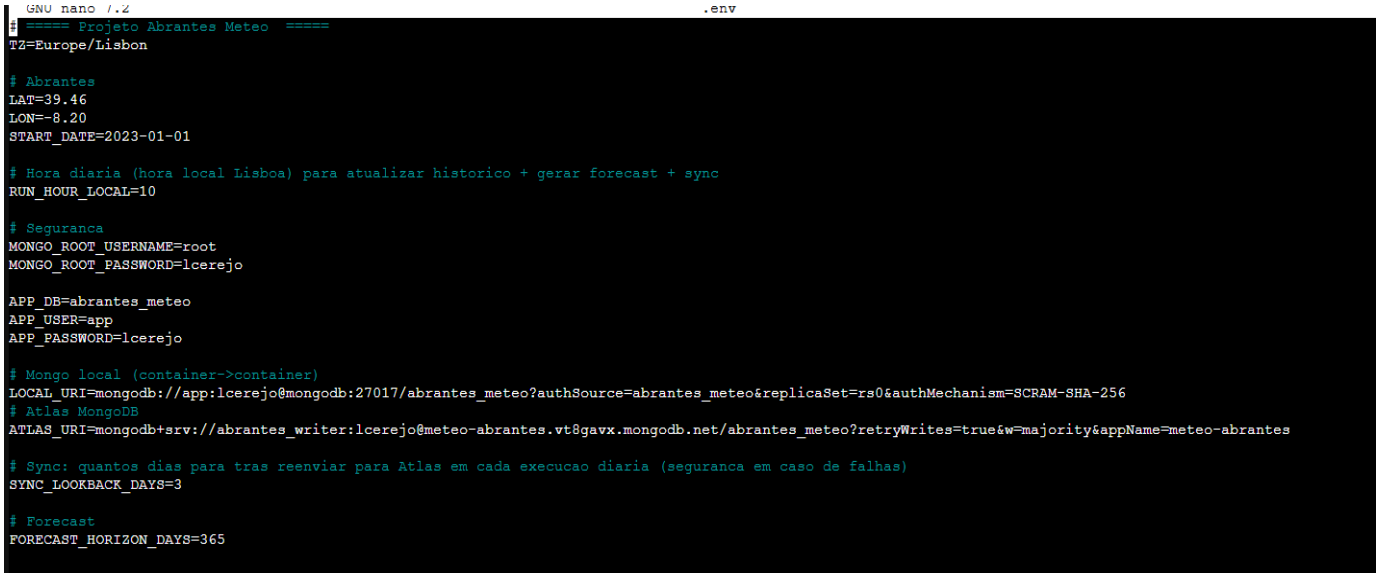


Figura 4 - Variáveis de ambiente no ficheiro .env (fuso horário e hora de execução do scheduler).

6.2.1. docker-compose.yml (estrutura de referência)

```
services:
  mongodb:
    image: mongo:7
    container_name: abrantes_meteo_mongodb
    command: ["mongod", "--bind_ip_all"]
    ports:
      - "27018:27017"
    volumes:
      - ./mongo-data:/data/db
      - ./mongo-init:/docker-entrypoint-initdb.d:ro
    environment:
      - MONGO_INITDB_ROOT_USERNAME=root
      - MONGO_INITDB_ROOT_PASSWORD=<ROOT_PASSWORD>
    healthcheck:
      test: ["CMD", "mongosh", "--quiet",
"mongodb://root:<ROOT_PASSWORD>@localhost:27017/admin", "--eval", "db.runCommand({
ping: 1 })"]
      interval: 10s
      timeout: 5s
      retries: 12
    restart: unless-stopped
  app:
    build: ./app
    container_name: abrantes_meteo_app
    environment:
      - TZ=${TZ}
      - LAT=${LAT}
      - LON=${LON}
      - START_DATE=${START_DATE}
      - RUN_HOUR=${RUN_HOUR}
      - LOCAL_URI=${LOCAL_URI}
      - ATLAS_URI=${ATLAS_URI}
      - MONGO_DB=${MONGO_DB}
    depends_on:
      mongodb:
        condition: service_healthy
    volumes:
      - ./logs:/app/logs
      - ./exports:/app/exports
      - ./dumps:/app/dumps
    restart: unless-stopped
```

6.2.2.mongo-init/01-setup.js (referência)

```
// Criar base de dados e utilizador de aplicação
db = db.getSiblingDB("abrantes_meteo");

db.createUser({
  user: "app",
  pwd: "<LOCAL_PASSWORD>",
  roles: [
    { role: "readWrite", db: "abrantes_meteo" }
  ]
});

// Índices mínimos para performance e consistência
db.meteo_historico.createIndex({ date: 1 }, { unique: true });
db.meteo_forecast_flat.createIndex({ run_id: 1, date: 1 }, { unique: true });
db.meteo_forecast_flat.createIndex({ created_at: -1 });
```

6.2.3.app/Dockerfile e requirements.txt (referência)

```
# app/Dockerfile
FROM python:3.11-slim

ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1

WORKDIR /app

# Dependências do Prophet/cmdstanpy (mínimo operacional)
RUN apt-get update && apt-get install -y --no-install-recommends     build-essential
gcc g++     && rm -rf /var/lib/apt/lists/*

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY app.py /app/app.py

CMD ["python", "/app/app.py"]
```

```
# app/requirements.txt (exemplo)
requests==2.32.3
pymongo==4.6.3
pandas==2.2.2
numpy==2.0.1
prophet==1.1.6
cmdstanpy==1.2.4
python-dateutil==2.9.0.post0
pytz==2024.1
schedule==1.2.2
```

6.3. Execução e validação do MongoDB local

Após a criação dos ficheiros, o deploy é efetuado com Docker Compose. O objetivo nesta fase é confirmar:

- saúde do MongoDB;
- execução inicial do backfill;
- criação do forecast;
- escrita nas coleções.

A validação “em linha de comandos” foi feita com mongosh, contando documentos e inspecionando o último registo em cada coleção. Este passo é crítico para separar problemas de ingestão de problemas de visualização (Power BI).

```
docker exec -it abrantes_meteo_mongodb mongosh -u root -p "<ROOT_PASSWORD>" --
authenticationDatabase admin --quiet --eval '
db=db.getSiblingDB("abrantes_meteo");
print("HIST=", db.meteo_historico.countDocuments());
print("FORE=", db.meteo_forecast_flat.countDocuments());
printjson(db.meteo_historico.find().sort({date:-1}).limit(1).toArray()[0]);
printjson(db.meteo_forecast_flat.find().sort({created_at:-
1}).limit(1).toArray()[0]);'
```

6.4. Sincronização para MongoDB Atlas

Com o armazenamento local validado, foi criado no MongoDB Atlas um projeto e um cluster para alojar a base de dados em cloud e viabilizar o consumo analítico.

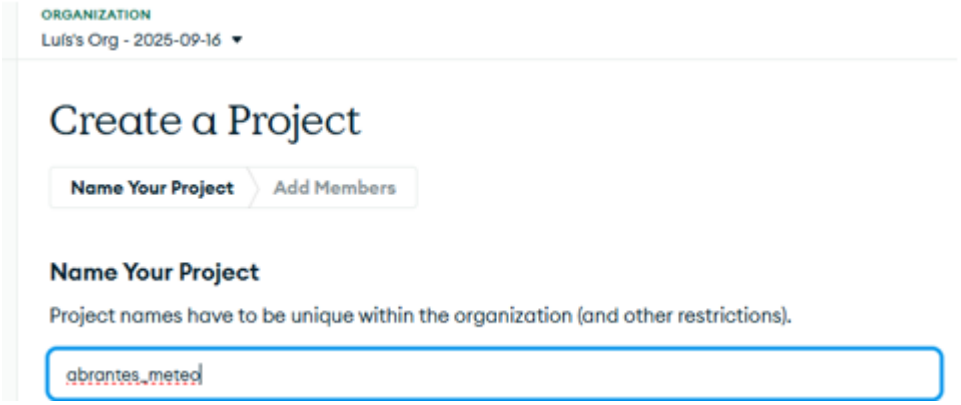


Figura 5 - Criação do projeto no MongoDB Atlas (console Atlas).

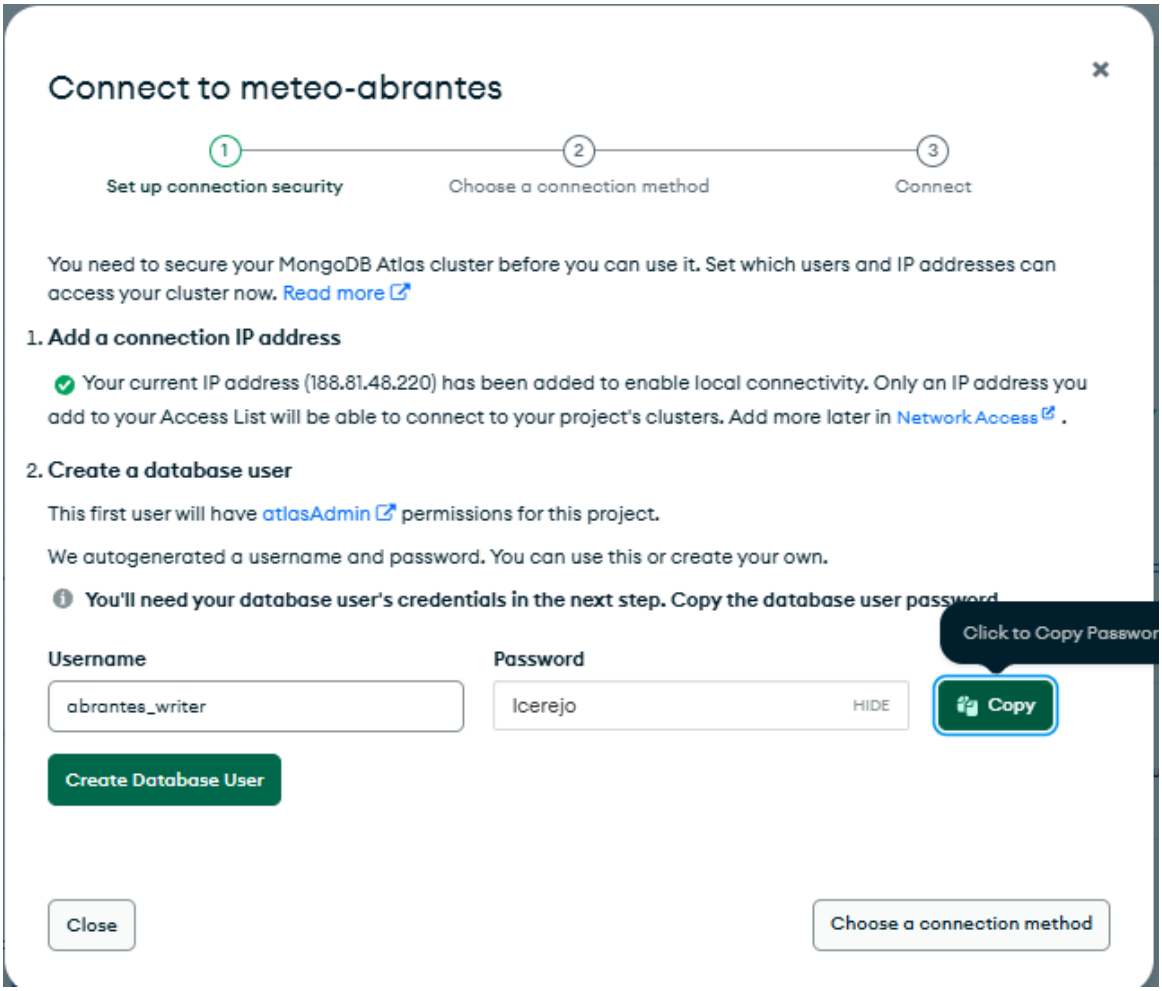


Figura 6 - Configuração de acesso ao cluster no Atlas (IP Access List e utilizador de base de dados).

Após a criação do cluster e a respetiva configuração de acesso (IP Access List), foram definidos dois utilizadores com permissões distintas, aplicando o princípio do menor privilégio:

- **abrantes_writer**: permissões de escrita na base abrantes_meteo, utilizado pelos processos de ingestão/sincronização.
- **abrantes_reader**: permissões de leitura na base abrantes_meteo, utilizado para consumo analítico (ex.: Atlas SQL/ODBC e Power BI).

Nota: após inclusão do Raspberry foram criados mais dois utilizadores.

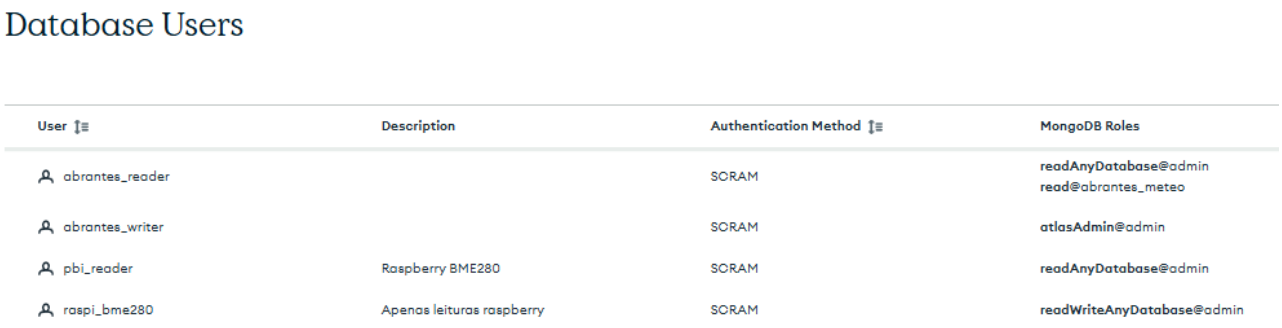


Figura 7 - Gestão de utilizadores no Atlas (Database Users) para acesso controlado ao cluster.

Concluída a configuração, o ATLAS_URI foi parametrizado no ficheiro .env e a conectividade foi validada via mongosh, confirmando-se a disponibilidade e consistência dos dados sincronizados

6.4.1. Resolução de dificuldades típicas (Atlas)

Durante a configuração do Atlas e dos testes de ligação surgiram problemas recorrentes, destacando-se três: autenticação (bad auth), execução do mongosh com variáveis de ambiente mal exportadas, e restrições de rede. A abordagem usada foi sempre a mesma: isolar o problema, reproduzi-lo com um comando mínimo e corrigir uma variável de cada vez.

Erro “Command aggregate requires authentication”: causado por parâmetros/URI incompletos (ou variáveis não carregadas). Correção: carregar. env corretamente (set -a; . ./.env; set +a) e usar a URI completa.

Erro “bad auth : authentication failed”: causado por password errada ou utilizador sem papel na base. Correção: confirmar Database Access, role read@abrantes_meteo e copiar a password correta.

Erro “getaddrinfo ENOTFOUND …mongodb.net”: tentativa de ligação ‘--host’ em vez de URI mongodb+srv (DNS SRV). Correção: usar mongodb+srv URI (recomendado para Atlas).

6.5. Exposição ao Power BI via Data Federation e Atlas SQL (ODBC)

O Power BI não consome MongoDB 'raw' de forma nativa com o mesmo nível de maturidade que consome fontes SQL. Por isso, a estratégia adotada foi usar Atlas Data Federation para criar uma Virtual Database e, em cima dela, Atlas SQL, gerando um schema tabular para consulta via ODBC.

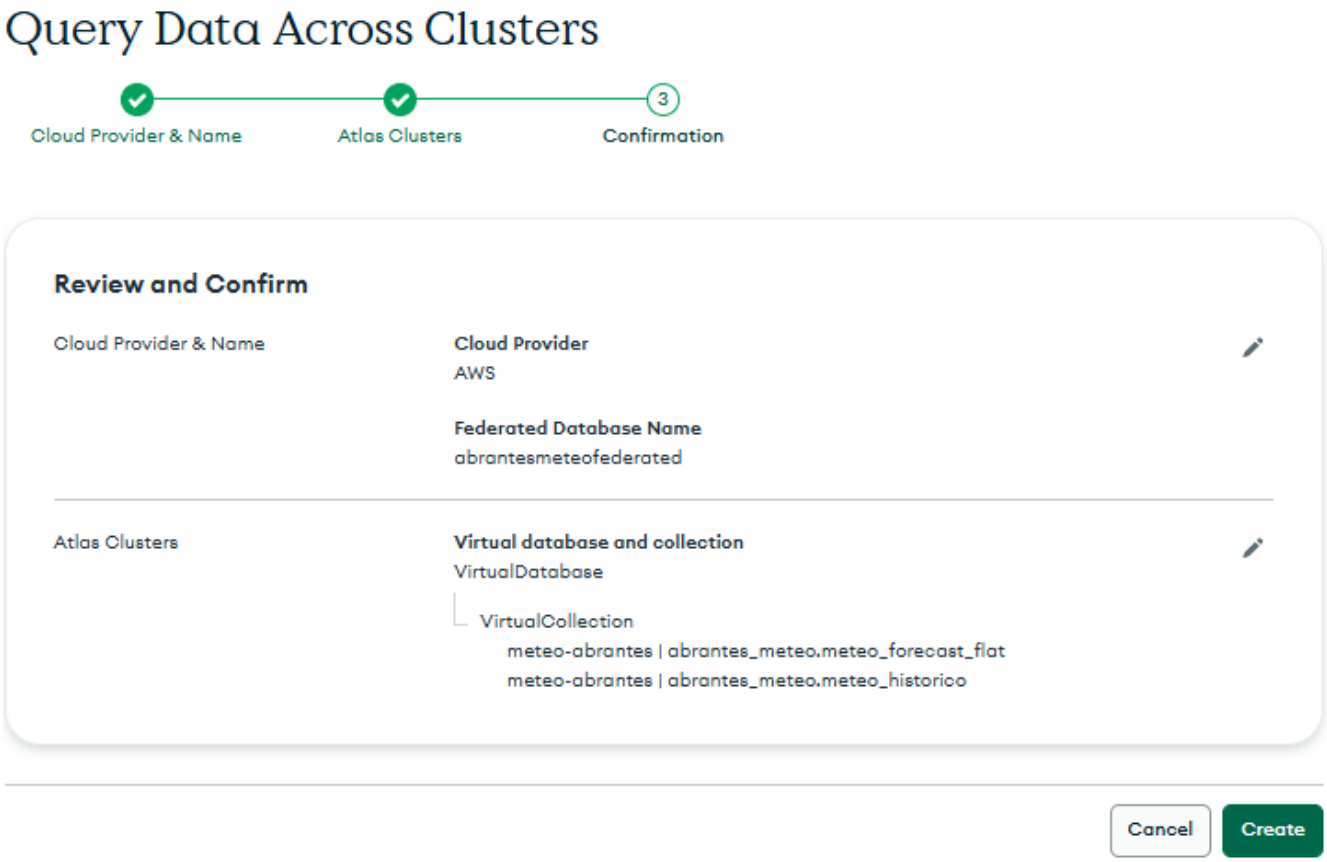


Figura 8 - Criação da Federated Database Instance (Atlas Data Federation) para expor dados do cluster.

6.5.1. Mapeamento de coleções na Federated Database

No ecrã de configuração (Update a Federated Database Instance), foram mapeadas duas coleções do cluster Atlas (meteo-abrantes): meteo_historico e meteo_forecast_flat. O nome da Virtual Database foi normalizado para 'abrantesmeteo', evitando espaços e inconsistências.

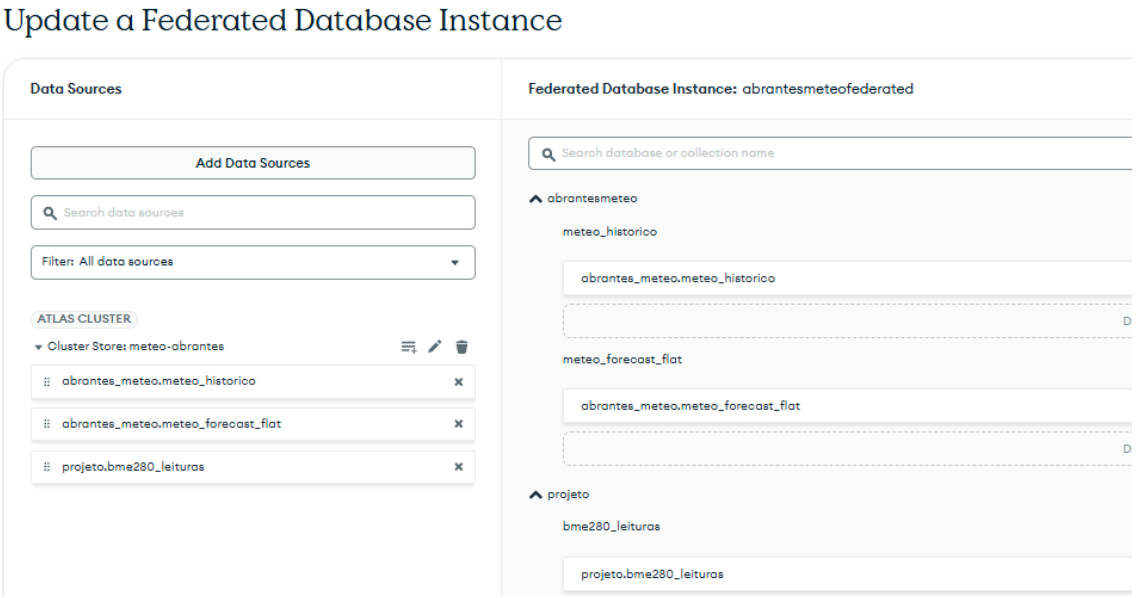


Figura 9 - Mapeamento das coleções do Atlas para a Federated Database Instance (Data Sources / Collections).

6.5.2. Geração do schema SQL (resolução do erro “no visible columns”)

Um ponto crítico foi o erro apresentado no Power BI: “The table has no visible columns and cannot be queried”. Este erro não era do Power BI; era do lado Atlas SQL, que ainda não tinha um schema SQL gerado/visível para as coleções. A correção foi aceder a “Manage SQL Schemas” na instância federada e gerar os schemas por defeito.

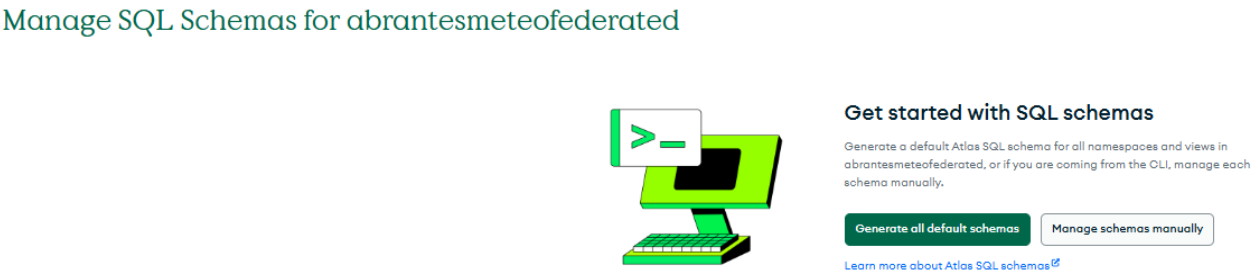


Figura 10 - Ecrã ‘Manage SQL Schemas’ da instância federada — opção para gerar schemas por defeito.

Passos executados:

1. Atlas → Data Federation → selecionar a instância (abrantesmeteofederated).
2. Abrir “Manage SQL Schemas”.
3. Selecionar “Generate all default schemas”.
4. Aguardar a conclusão e confirmar que as tabelas/colunas passam a estar expostas no Atlas SQL.

6.5.3. Configuração do driver ODBC (DSN)

No Windows, foi criado um DSN ODBC com o “MongoDB Atlas SQL ODBC Driver”. Este driver é o correto (não é o ODBC do SQL Server nem o driver genérico). A URI usada é a fornecida no Atlas (Connect → Atlas SQL → Power BI Connector).

Figura 11 - Configuração do DSN ODBC para Atlas SQL/ODBC (nome do Data Source).

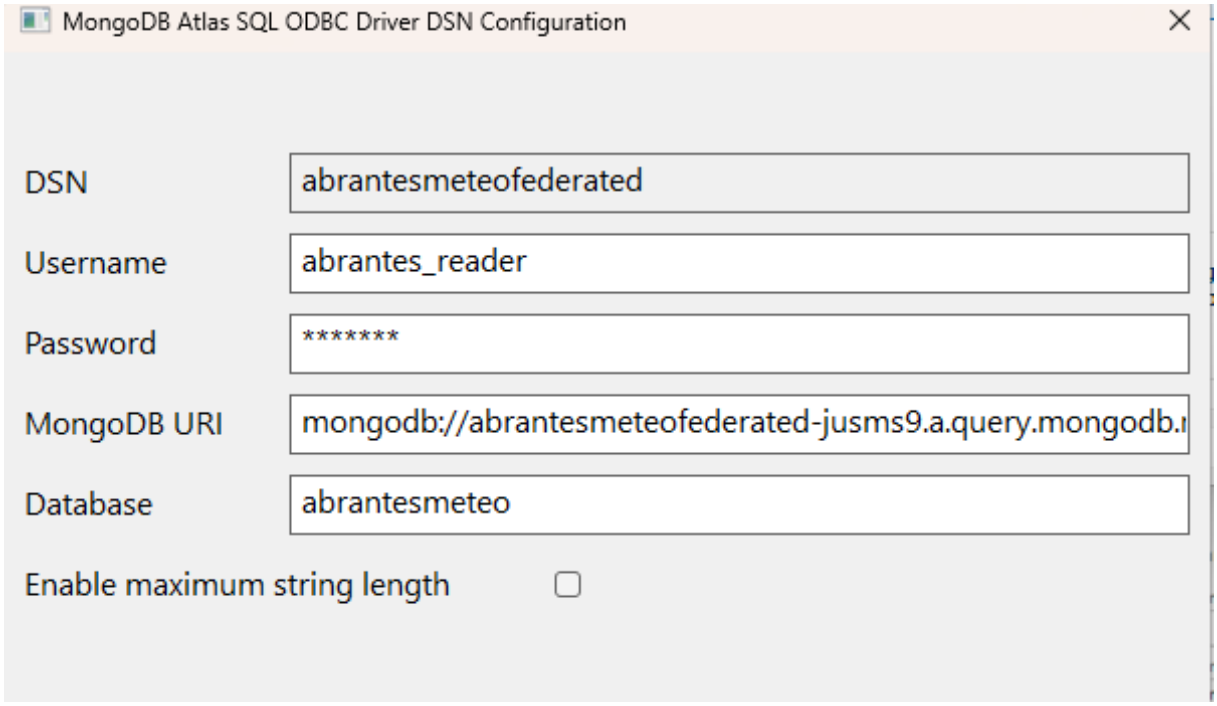


Figura 12 - Validação do DSN ODBC (teste de ligação bem-sucedido ao endpoint Atlas SQL).

6.6. Modelação no Power BI

Com a ligação operacional, o foco passa para a modelação: tipagem de dados, tabela calendário, relacionamentos e medidas DAX. A opção foi manter as tabelas físicas (meteo_historico e meteo_forecast_flat) e criar uma tabela derivada (forecast_fixo) para fixar o baseline de previsão.

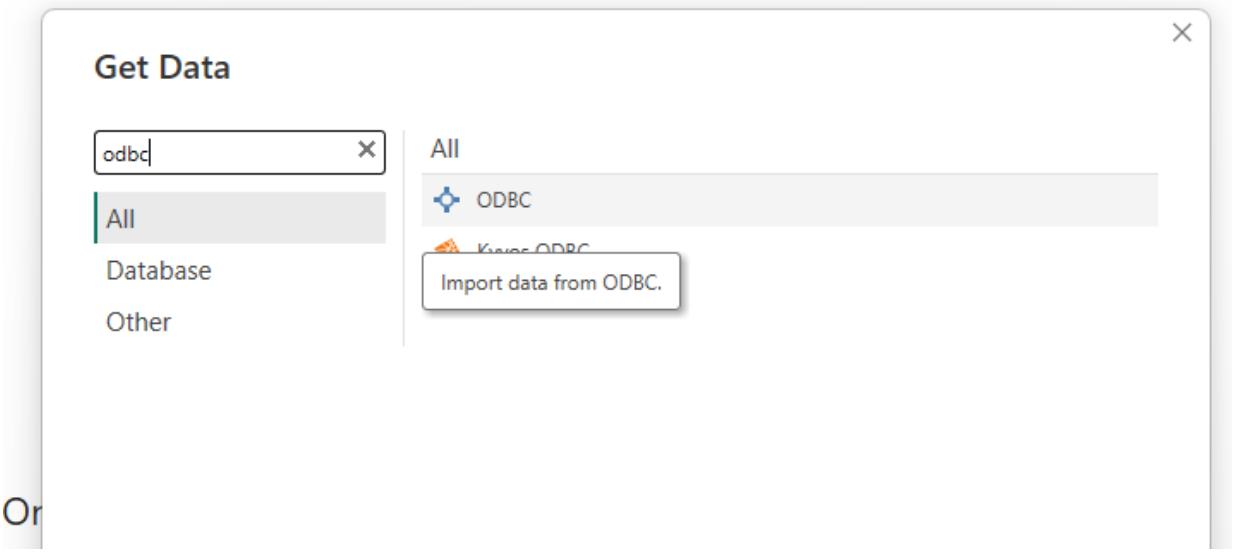


Figura 13 - Power BI Desktop: importação de dados através de ODBC (Atlas SQL).

6.6.1. Tabela calendário

A tabela calendário é um elemento central para garantir filtragem temporal consistente e para permitir que Real e Previsto sejam comparados no mesmo eixo. A recomendação é ter uma única coluna Date (sem hora) e relacioná-la, em relação 1:* , com as colunas date das tabelas de factos.

```
Calendario =
ADDCOLUMNS (
    CALENDAR ( DATE(2023,1,1), DATE(2027,12,31) ),
    "Ano", YEAR ( [Date] ),
    "Mês", FORMAT ( [Date], "MMMM" ),
    "MêsNum", MONTH ( [Date] ),
    "Dia", DAY ( [Date] )
)
```

6.6.2. Relações (Model View)

Calendario[Date] (1)→ meteo_historico[date] (*) - filtro unidirecional.

Calendario[Date] (1) → forecast_fixo[date] (*) - filtro unidirecional.

Não é necessário relacionar meteo_historico diretamente com forecast_fixo; a comparação é feita via medidas.

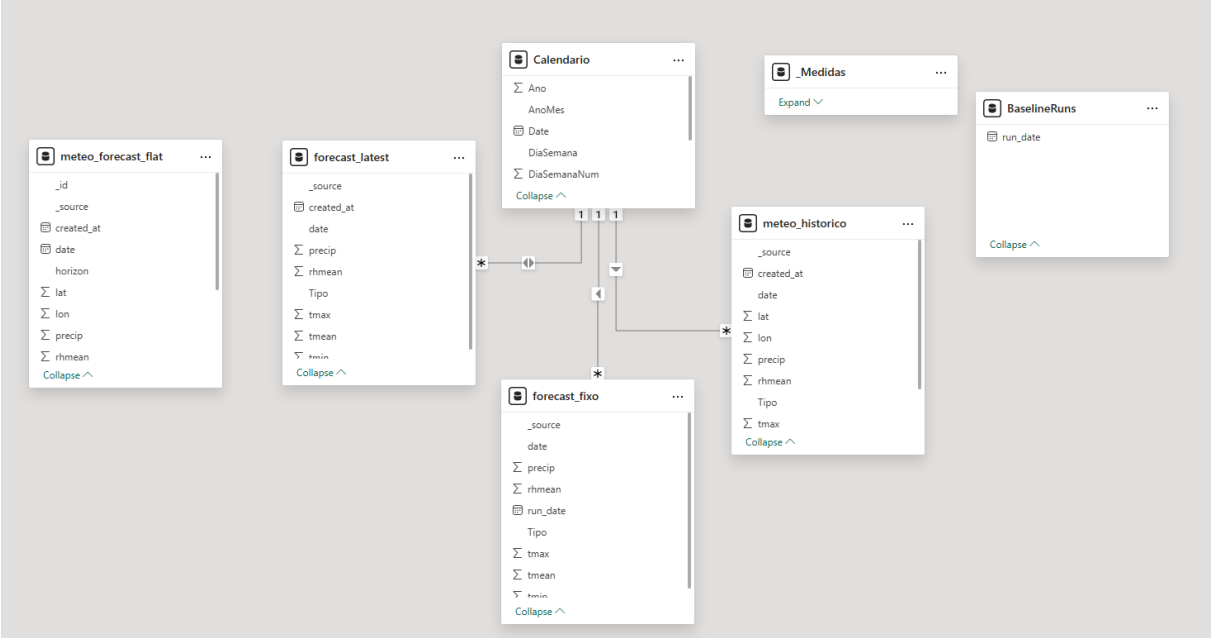


Figura 14 - Power BI: vista de modelo com tabelas de Real e Previsão para comparação.

6.6.3. forecast_fixo: baseline de previsão

A previsão diária é recalculada a cada execução, pelo que o 'forecast do dia 1' de hoje não é igual ao 'forecast do mesmo dia' amanhã. Para análise comparativa séria (o que estava previsto vs. o que aconteceu), é necessário 'congelar' uma execução (run_date) e usá-la como baseline. A tabela forecast_fixo foi criada no Power Query filtrando a meteo_forecast_flat por um run_date escolhido (baseline) e mantendo-o fixo, dessa forma os dados obtidos de forma preditiva, passam a ser "histórico" para comparação posterior.

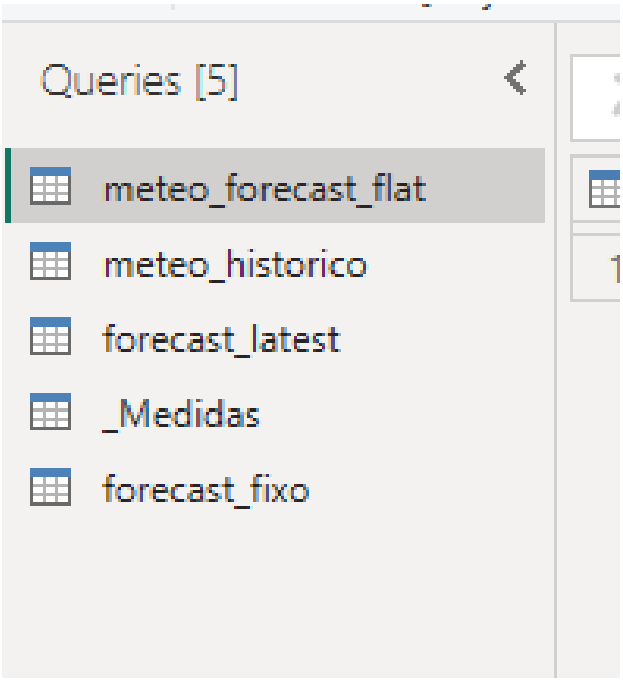


Figura 15 - Power Query: queries finais (tabelas base e tabelas derivadas para análise).

Tipos de dados recomendados (forecast_fixo):

Esta tabela é fundamental estar bem desenhada, pois é um clone da tabela forecast original e deve, para além de fixar os valores passados, receber os novos dados e assim poder fazer as análises comparativas. Neste projeto as colunas foram configuradas da seguinte forma:

date: Date (sem hora) - chave temporal.

run_date: Date - data da execução usada como baseline (constante na tabela).

created_at: Date/Time (opcional para auditoria) - útil se se pretender rastrear o momento exato da geração.

_source e Tipo: Text - identificadores de origem (prophet) e natureza (Previsto).

tmin, tmean, tmax, precip, rhmean, lat, lon: Decimal Number (ou Fixed decimal, se se quiser controlo de precisão).

_id, horizon, run_id: manter apenas se forem necessários; para análise diária básica, podem ser removidos no Power Query.

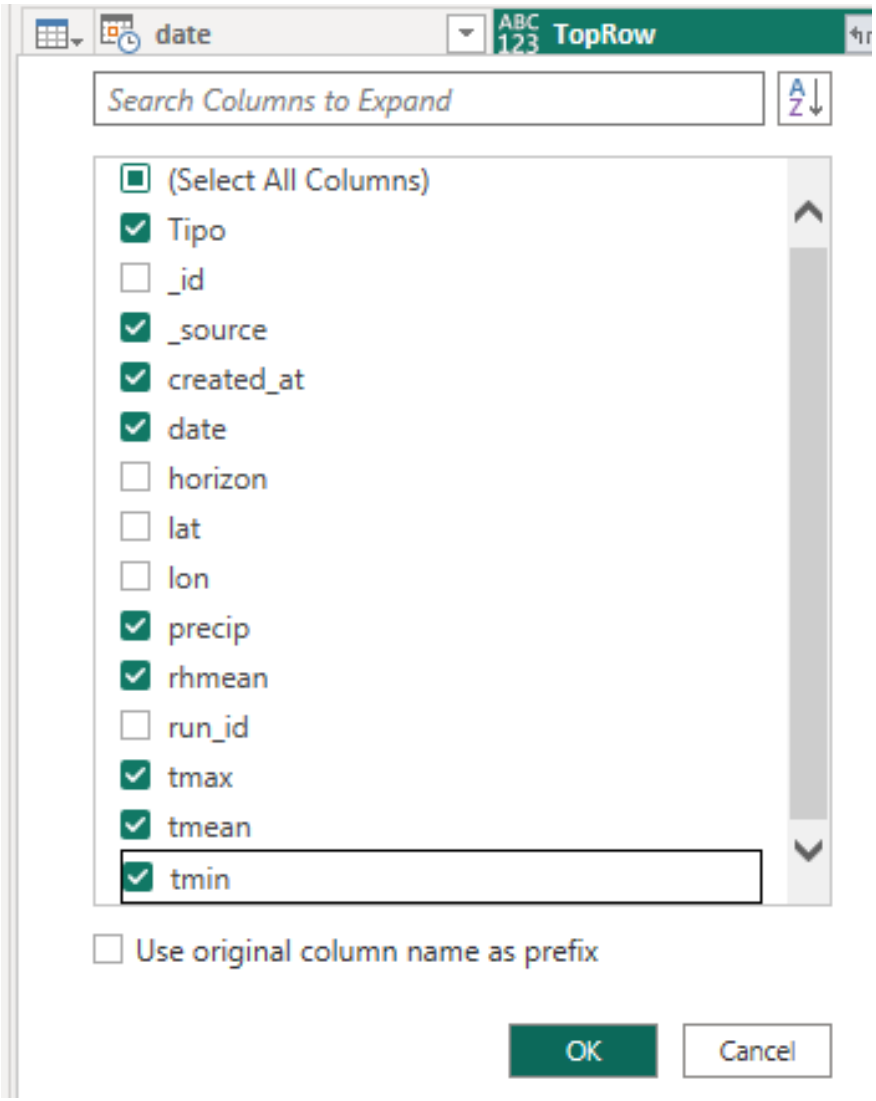


Figura 16 - Power Query: expansão de colunas ao normalizar registos (Expand record).

6.6.4. Medidas DAX para comparação (exemplo: Tmax, Tmin e Precip)

As medidas são criadas individualmente (uma a uma) no Power BI: Modelagem → Nova medida. A lógica base é: uma medida para o Real, outra para o Previsto (baseline), e uma medida de diferença.

```
-- Tmax
Real_Tmax =
AVERAGE ( meteo_historico[tmax] )

Prev_Tmax =
AVERAGE ( forecast_fixo[tmax] )
```

```
Dif_Tmax =
[Real_Tmax] - [Prev_Tmax]
```

```
-- Tmin
Real_Tmin =
AVERAGE ( meteo_historico[tmin] )

Prev_Tmin =
AVERAGE ( forecast_fixo[tmin] )

Dif_Tmin =
[Real_Tmin] - [Prev_Tmin]
```

```
-- Precipitação
Real_Precip =
AVERAGE ( meteo_historico[precip] )

Prev_Precip =
AVERAGE ( forecast_fixo[precip] )

Dif_Precip =
[Real_Precip] - [Prev_Precip]
```

Nota operacional:

Não é necessário aplicar filtros dentro das medidas para 'run_date = baseline', porque forecast_fixo já está filtrada no Power Query. Este desenho evita erros comuns no DAX (ex.: usar uma expressão True/False inválida como filtro de tabela). Isto levou a alfuns erros em fases iniciais pois não devolvia valores de forma lógica, ou pelo menos, na lógica pretendida.

6.7. Integração de Raspberry Pi 5 + sensor BME280

Este subcapítulo documenta a extensão do sistema Abrantes Meteo com um Raspberry Pi 5 equipado com um sensor BME280, permitindo recolher temperatura, humidade relativa e pressão atmosférica e disponibilizá-las no Power BI como uma tabela independente. A integração foi desenhada para ser simples, resiliente (execução autónoma via cron) e compatível com o modelo existente (MongoDB Atlas + ODBC + Power BI Service com Gateway).

6.7.1. Arquitetura e fluxo de dados

Raspberry Pi 5, lê o sensor BME280 via I2C.
O Raspberry executa um script Python que insere leituras na coleção “projeto.bme280_leituras” no MongoDB Atlas (via Atlas SQL / Data Federation).
No PC Desktop, o Power BI Desktop liga via ODBC (DSN do Atlas SQL Interface) e consome a tabela.
O dataset é publicado no Power BI Service e é atualizado com o On-premises data gateway (personal mode). Do mesmo modo que que foi efetuado para os dados históricos e forecast. Mas visto
Nota: as leituras do BME280 não são misturadas com “meteo_historico” / “meteo_forecast_flat”. Isto evita confusões de granularidade (diário vs. quase-real) e permite construir visuais dedicados (cartão de “temperatura atual”, gráfico temporal, etc.).

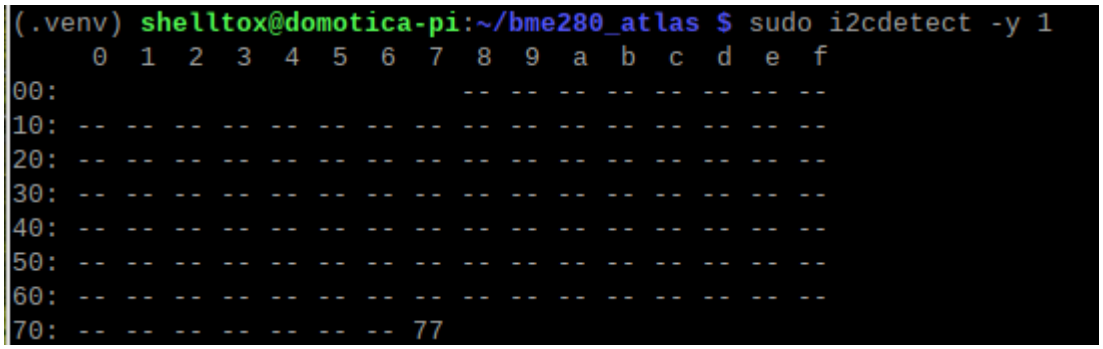


Figura 17 - Confirmação de endereço do sensor no Raspberry 0x77

6.7.2. Preparação do hardware e I2C no Raspberry Pi 5

Ligação típica do BME280 ao Raspberry Pi (I2C):
VCC → 3.3V (Pin 1).
GND → GND (Pin 6).
SDA → GPIO2 / SDA1 (Pin 3).
SCL → GPIO3 / SCL1 (Pin 5).

Após a ligação física, o I2C deve estar ativo no sistema operativo. A forma mais direta é via “raspi-config” (Interface Options → I2C → Enable). Alternativamente, pode-se garantir no “/boot/config.txt” com “dtparam=i2c_arm=on”.

Comandos de validação (no Raspberry):

```
sudo apt update
sudo apt install -y i2c-tools
lsmod | grep i2c || true
sudo i2cdetect -y 1
```

6.7.3. Projeto Python no Raspberry (venv, dependências e ficheiro .env)

Foi criada uma pasta dedicada no Raspberry (ex.: "/home/shelltox/bme280_atlas") com ambiente virtual Python 3.11 e dependências mínimas. O objetivo é evitar conflitos com pacotes globais do sistema.

Criação do ambiente e instalação de dependências (no Raspberry):

```
cd /home/shelltox
mkdir -p bme280_atlas && cd bme280_atlas
python3 -m venv .venv
source .venv/bin/activate
pip install --upgrade pip
# Bibliotecas Adafruit (I2C + BME280) e MongoDB
pip install pymongo dnspython adafruit-blinka adafruit-circuitpython-bme280
```

Configuração de variáveis de ambiente (.env). Este ficheiro concentra a ligação ao Atlas e os nomes da base de dados/coleção, evitando hardcode no código Python.
Configuração de variáveis de ambiente (.env). Este ficheiro concentra a ligação ao Atlas e os nomes da base de dados/coleção, evitando hardcode no código Python.

```
cat > .env <<'EOF'
# String de ligação copiada do Atlas (Data Federation / MongoDB Driver).
ATLAS_URI="mongodb://<db_user>:<db_pass>@<host>.a.query.mongodb.net/?ssl=true&authSource=a
dmin&appName=<appName>"

# Destino no Atlas
DB_NAME="projeto"
COL_NAME="bme280_leituras"
```

```
# Sensor
I2C_BUS="1"
I2C_ADDR="0x77"
EOF

# Carregar variáveis na sessão atual
set -a; source .env; set +a
```

Boas práticas aplicadas durante a resolução de erros de encoding: garantir locale UTF-8 e ficheiros em UTF-8. Em sistemas com locale “C” (ASCII), scripts e here-docs com acentos podem falhar com erros do tipo “Non-UTF-8 code …”. Uma mitigação simples é definir “LANG=C.UTF-8” e “LC_ALL=C.UTF-8” no ambiente de execução.

6.7.4. Script de recolha e envio para o Atlas

O script “sensor_to_atlas.py” efetua:
(1) leitura do BME280 via I2C;
(2) normalização de campos;
(3) inserção de um documento no Atlas. Os campos usados no documento seguem uma estrutura estável para facilitar a inferência de schema no Atlas SQL Interface e a importação no Power BI.

Estrutura típica do documento inserido:

```
{
  ts: <datetime UTC>,
  source: "raspi-bme280",
  host: "domotica-pi",
  temp_c: <float>,
  hum_rel: <float>,
  press_hpa: <float>
}
```

Verificação rápida após execução (confirmar que existe escrita no Atlas):

```
source .venv/bin/activate
set -a; source .env; set +a
python sensor_to_atlas.py
```

```
python - <<'PY'
import os
from pymongo import MongoClient
atlas=os.getenv('ATLAS_URI'); dbn=os.getenv('DB_NAME'); col=os.getenv('COL_NAME')
cli=MongoClient(atlas, serverSelectionTimeoutMS=8000)
c=cli[dbn][col]
last=list(c.find().sort([('ts',-1)]).limit(1))
print(last[0] if last else None)
cli.close()
PY
```

6.7.5. Automatização no Raspberry (cron: 5 leituras diárias)

Para garantir recolhas regulares sem intervenção manual, foi configurado um job de cron no utilizador “shelltox”. Neste cenário, são feitas 5 leituras por dia (07:00, 11:00, 15:00, 19:00 e 23:00).

Exemplo de crontab (shelltox):

```
crontab -e

# 5 execuções diárias
0 7,11,15,19,23 * * * /home/shelltox/bme280_atlas/run_once.sh >>
/home/shelltox/bme280_atlas/cron.log 2>&1
```

Teste manual do script wrapper e criação do ficheiro de log:

```
/home/shelltox/bme280_atlas/run_once.sh
tail -n 50 /home/shelltox/bme280_atlas/cron.log
systemctl status cron --no-pager
```

6.7.6. Atlas: permissões, coleção e schema para Power BI (Atlas SQL Interface)

Para o Power BI consumir esta coleção via ODBC (Atlas SQL Interface), a coleção tem de ter schema visível. Sem schema, o conector pode devolver o erro “The table has no visible columns and cannot be queried”.

Passos no MongoDB Atlas (via UI):

- Confirmar que a coleção existe e tem documentos: Cluster → Collections → DB “projeto” → collection “bme280_leituras”.
- Criar/usar um Database User com permissões mínimas (ex.: “read” no DB/collection, se for apenas leitura do Power BI; “readWrite” apenas para o Raspberry).
- Em Data Federation / SQL Interface: garantir que a base de dados e coleção estão mapeadas e expostas ao endpoint SQL/ODBC.
- Gerar/definir o schema da coleção para que os campos fiquem bem identificados e “visíveis” para consulta via ODBC.

Schema recomendado para “projeto.bme280_leituras” (tipos): “ts” (datetime), “source” (string), “host” (string), “temp_c” (double), “hum_rel” (double), “press_hpa” (double). Após aplicar o schema, a tabela passa a ser carregável no Power BI e desaparece os erros de colunas não visíveis.

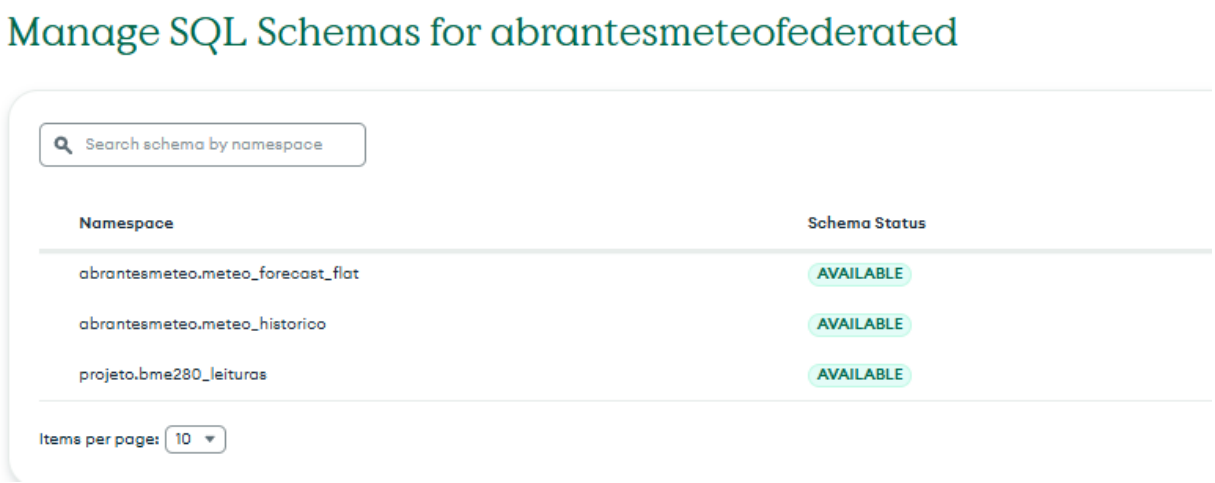


Figura 18 - SQL Schemas (Atlas Data Federation): namespaces disponíveis para consulta (ODBC/Power BI).

6.7.7. Power BI Desktop: importar a tabela e criar o scroller

1. **Importação**: Power BI Desktop → Get Data → ODBC → selecionar o DSN do Atlas (ex.: "abrantesmeteofederated"). No Navigator, escolher a base de dados "projeto" e a tabela "bme280_leituras" e carregar.
2. **Query "última leitura":** no Power Query Editor, duplicar "bme280_leituras" e chamar "bme280_latest" → ordenar por "ts" (descendente) → Keep Rows → Keep Top Rows → 1. Esta abordagem permite um cartão estável (sem DAX complexo) para mostrar a temperatura atual.
3. **Visual do valor atual:** inserir um Card (ou Card (New)) com o campo "temp_c" da query "bme280_latest". Formatar título (ex.: "Temperatura Atual - °C"), unidades e casas decimais.
4. **Scroller/zoom temporal:** criar um Line chart com "ts" no eixo X e "temp_c" em Values. Em Format → X-axis → ativar "Zoom slider" (funciona como scroller) para navegar em janelas temporais (ex.: últimas 24h, 7 dias, etc.).
5. **Publicação e atualização:** publicar o relatório no Power BI Service e configurar atualização programada usando o On-premises data gateway (personal mode) instalado no PC Desktop. O gateway é necessário porque o datasource é ODBC local, apesar de os dados estarem no Atlas.

Observação operacional: para um painel "quase em tempo real", o mais simples foi manter o cron no Raspberry (várias inserções/dia) e fazer refresh no Power BI Service com periodicidade compatível com o gateway e com as restrições do Power BI (ex.: de hora a hora). Se for necessário atualizar minuto-a-minuto, isso já entra em arquitetura de streaming (fora do âmbito do protótipo atual). Optei por manter o cron a correr no Raspberry 5 vezes ao dia e alinhar as atualizações do gateway para 1 hora após serem atualizadas via cron. E, assim, evitam-se alguns dissabores relacionados com diferenças horárias, já que o MongoDB Atlas, na versão gratuita, não disponibiliza o fuso horário de Portugal como opção.

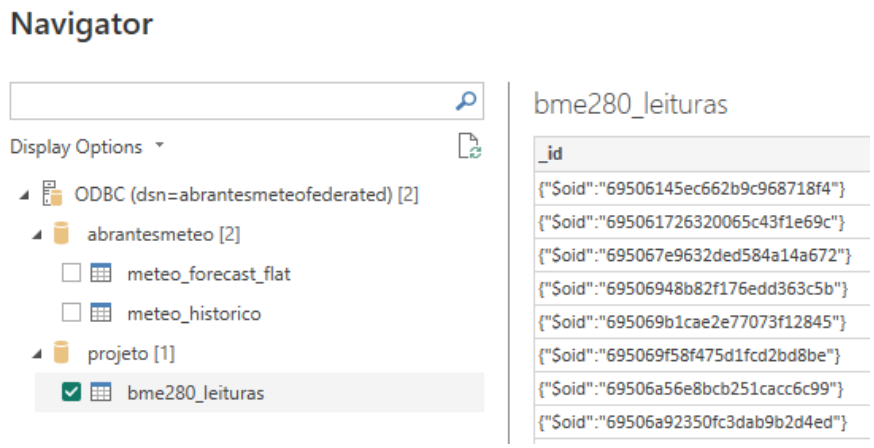


Figura 19 - Importação da Coleção "projeto.bme280_leituras" do MongoDB Atlas (pré-visualização de documentos e campos).

7. Dificuldades & Resolução

Um dos aspetos de mais relevo no projeto foi a resolução de problemas reais de integração. A seguir descrevem-se os incidentes mais relevantes, a causa provável e a ação corretiva aplicada.
Codificação UTF-8 no container Python: erros de execução quando o ficheiro não estava em UTF-8.

- Resolução: garantir gravação em UTF-8 e evitar caracteres inválidos.

Autenticação no Atlas (bad auth): erro por password/role incorretos.

- Resolução: recriar/confirmar utilizador no Database Access e validar com mongosh via mongodb+srv.

Power BI sem colunas visíveis: schema SQL não gerado no Atlas SQL.

- Resolução: Generate all default schemas em Manage SQL Schemas.

Colunas duplicadas (date e date.1) após expandir registos no Power Query:

- Resolução: remover date.1 e mudar rtipo de dados date como Date.

Erros de ligação ODBC/driver:

- Resolução por seleção do driver Atlas SQL ODBC e URI correta do Power BI Connector.

---

8. Resultados finais obtidos

No final, foi possível demonstrar:

- dados históricos carregados com sucesso;
- previsão diária gerada e persistida (365 dias por execução);
- sincronização para Atlas e exposição SQL/ODBC;
- dashboards no Power BI com comparação entre Real e Previsto, cada previsão tem um identificador de execução (run_id/run_date), e o previsto pode ser congelado para comparação à *posteriori.*

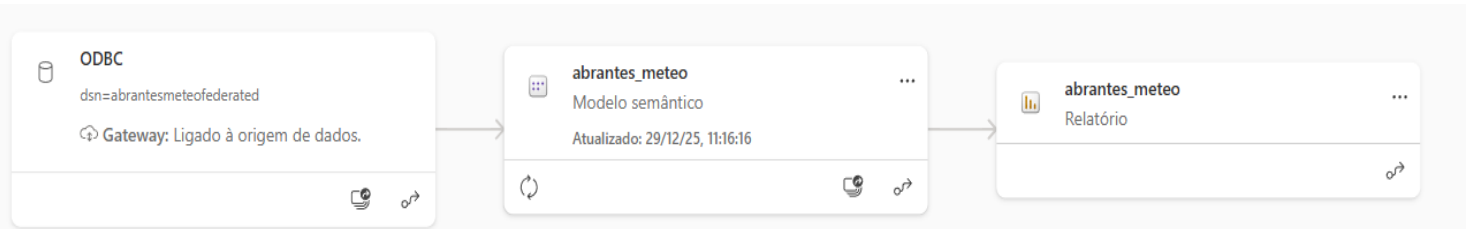


Figura 20 - Pipeline de publicação no Power BI Service: origem ODBC (Atlas SQL) com Gateway, dataset (modelo semântico) e relatório publicado.

9. Melhorias futuras e visão de evolução

Este projeto tem várias vias claras de evolução - e o mais relevante é que a arquitetura não é "meteorologia-dependente": pode ser reaproveitada para qualquer cenário de ingestão → armazenamento → transformação → previsão/analítica → visualização, como vendas, stocks, consumo energético, manutenção preventiva, tráfego, qualidade industrial, KPIs financeiros ou comportamento de utilizadores.

Principais melhorias e extensões (replicáveis noutras análises):

- Automatizar a criação de um forecast_fixo no servidor (coleção dedicada) para eliminar dependências do Power Query na fixação da base (garante consistência e repetibilidade do histórico de previsões).
- Persistir todas as execuções de forecast com run_id e gerar automaticamente métricas de desempenho (ex.: MAE/RMSE) por horizonte (D+1, D+7, D+30, etc.), criando uma base sólida para auditoria e melhoria contínua do modelo.
- Implementar observabilidade e alertas (Prometheus/Grafana e/ou logs estruturados com níveis e correlação por run_id) para detetar falhas, degradação de performance, atrasos de execução e anomalias operacionais.
- Reforçar gestão de segredos e segurança (Docker Secrets ou HashiCorp Vault) com rotação de credenciais do Atlas, princípio do menor privilégio e segregação de acessos por serviço.
- Adicionar camadas de qualidade de dados antes de persistir: validações, normalização, deteção de outliers, regras de consistência temporal e thresholds por variável (isto aplica-se tanto a sensores como a ERP/CRM/BI).
- Escalar para múltiplas fontes e múltiplas "entidades" (multi-cidade, multi-loja, multi-linha de produto, etc.) com parametrização e partição (por lat/lon, por entity_id, por período), mantendo custos e performance controlados.

Nota operacional crítica (não implementada, mas relevante):

Um ponto importante que ficou fora deste projeto foi garantir um ambiente Windows 24/7 em VM no servidor para suportar atualizações contínuas quando a camada de atualização depende de componentes Windows (ex.: gateways, refreshers, conectores específicos). A solução aplicada foi instalar e executar no PC e calendarizar atualizações em janelas horárias em que o equipamento supostamente está ligado. Para maturidade de produção, o ideal é mover essa dependência para infraestrutura sempre disponível (VM/serviço dedicado), reduzindo risco de falhas por indisponibilidade do posto local.

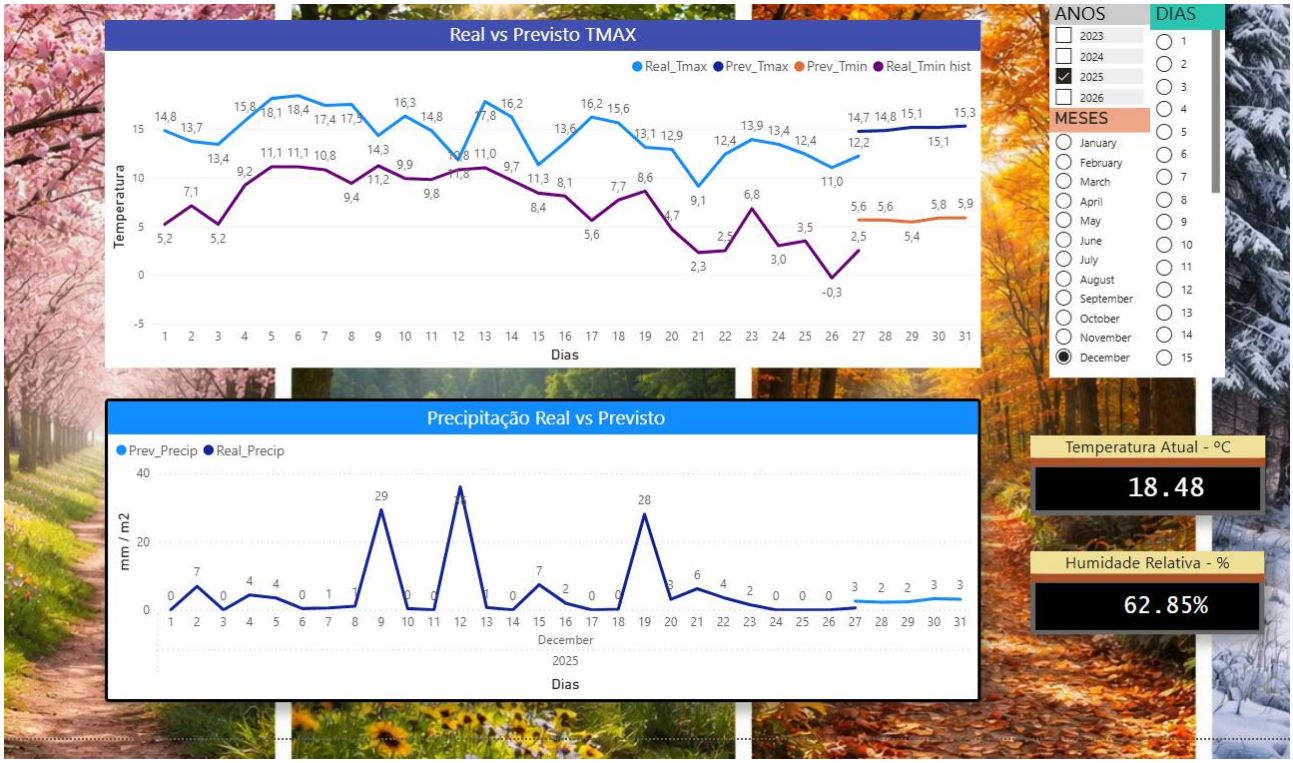


Figura 21 - Visual no Power BI: Cartões "Temperatura Atual, Humidade Relativa, Real vs Previsto em Temperatura e Precipitação".

10. Conclusão

O AbrantesMeteo resultou numa implementação operacional e coerente de um pipeline de dados completo, desde a recolha automática até à análise em Power BI, com armazenamento em MongoDB e previsão em Python com Prophet. A separação entre os dados reais e as previsões, bem como a definição de um baseline de forecast, permitiu que fosse possível efetuar comparações fiáveis e repetíveis ao longo do tempo, evitando leituras enviesadas por recalibrações diárias.

A opção por expor os dados via MongoDB Atlas e Atlas SQL (ODBC) foi determinante para garantir compatibilidade e estabilidade na sua utilização no Power BI, reduzindo dessa forma problemas típicos de integração com fontes NoSQL. A extensão com Raspberry Pi 5 e sensor BME280 adicionou uma componente experimental de relevo e também uma camada de complexidade, que apesar de à partida parecer de fácil integração se tenha revelado desafiante q.b., mas permitiu demonstrar a integração de medições físicas reais no mesmo ecossistema de dados.

No seu conjunto, o projeto permite validar uma abordagem modular e escalável, facilmente adaptável a outros contextos além da meteorologia (ex.: planeamento produção, vendas, stocks, energia ou manutenção), e evidencia competências práticas de integração de sistemas, automação e análise preditiva.

11. Glossário

API — Interface de programação que permite obter dados de um serviço externo (ex.: meteorologia) de forma automática.

Atlas (MongoDB Atlas) — Plataforma cloud gerida pela MongoDB para alojar bases de dados e disponibilizar serviços associados.

Atlas Data Federation — Camada do Atlas que permite virtualizar dados e expô-los como base virtual para consulta.

Atlas SQL (SQL Interface) — Interface SQL do Atlas (sobre Data Federation) para consultar dados MongoDB em formato tabular.

Backfill — Carregamento retroativo de histórico (ex.: desde 01-01-2023) para construir a base inicial de dados.

Baseline — Referência fixa usada para comparação (ex.: "previsão congelada" vs. "real observado").

BME280 — Sensor ambiental (temperatura, humidade e pressão atmosférica) usado no Raspberry Pi.

CMDSTANPY — Motor/biblioteca usada pelo Prophet para executar modelos (dependência do forecasting).

Coleção (MongoDB Collection) — Estrutura onde ficam guardados documentos (equivalente aproximado a "tabela" em SQL).

Container (Docker) — Unidade isolada que executa uma aplicação com dependências, de forma reprodutível.

Cron — Agendador no Linux para executar tarefas automaticamente em horários definidos.

DASHBOARD (Power BI) — Conjunto de visuais/indicadores para análise e acompanhamento.

Dataset / Modelo semântico (Power BI) — Camada de dados publicada no Power BI Service com modelo, relações e medidas.

DAX — Linguagem do Power BI para criar medidas/cálculos (ex.: Real vs. Previsto e diferenças).

Docker — Plataforma para construir e executar serviços em containers.

Docker Compose — Ficheiro/forma de orquestrar vários contentores e volumes (serviços do projeto).

Documento (MongoDB Document) — Registo em formato JSON/BSON (equivalente aproximado a "linha" em SQL).

DSN (ODBC Data Source Name) — Nome/configuração local do conector ODBC (host, credenciais, base, etc.).

Forecast — Previsão gerada pelo modelo para datas futuras (no projeto, horizonte de 365 dias).

Gateway (On-premises Data Gateway) — Componente do Power BI Service que permite refrescar fontes locais (ex.: ODBC/DSN).

I2C — Protocolo de comunicação usado para ligar o sensor BME280 ao Raspberry Pi.

Keyfile (MongoDB) — Ficheiro usado para autenticação interna do replica set quando a autenticação está ativa.

LXC — Tipo de contentor (a nível de sistema) usado no Proxmox para isolar o ambiente "docker-ct".

Mongosh — Shell/cliente de linha de comandos para ligar ao MongoDB e validar queries/contagens.

MongoDB — Base de dados NoSQL orientada a documentos, usada para persistência local e em cloud.

ODBC — Standard de conectividade para aceder a dados via drivers, comum em ferramentas analíticas (Power BI).

Open-Meteo — Fonte/API de dados meteorológicos usada para histórico e atualização diária.

Power BI Desktop — Ferramenta local para modelação, Power Query, DAX e criação do relatório.

Power BI Service — Plataforma cloud onde o relatório/dataset é publicado e atualizado (refresh).

Power Query — Motor de transformação/limpeza de dados no Power BI (tipos, remoções, tabelas derivadas).

Proxmox — Hypervisor/plataforma do servidor (host) onde corre o contentor LXC.

Replica Set — Modo do MongoDB que suporta replicação e é frequentemente necessário para certas configurações com auth.

run_id / run_date — Identificador/data da execução diária do forecast, usado para rastreio e consistência analítica.

Schema (SQL Schema) — Definição de colunas/tipos expostos via Atlas SQL; sem isto podem surgir tabelas "sem colunas visíveis".

Scheduler — Rotina que agenda a execução diária do pipeline (ingestão + forecast + sync).

Timezone / TZ — Configuração do fuso horário (ex.: Europe/Lisbon) para garantir execução e datas coerentes.

UTC — Referência de tempo padrão usada frequentemente em timestamps (evita ambiguidades de fusos horários).

WiredTiger — Motor de armazenamento do MongoDB (gera ficheiros internos como journal e metadata no volume).

12. Referências online

MongoDB. *Set Up and Query Data Federation – Atlas*. [Em linha]. Disponível em: https://www.mongodb.com/docs/atlas/data-federation/. [Consultado em: 19-10-2025]. Set Up and Query Data Federation - Atlas - MongoDB Docs

MongoDB. *Connect with ODBC Driver – Atlas (SQL Interface / Data Federation)*. [Em linha]. Disponível em: https://www.mongodb.com/docs/atlas/data-federation/query/sql/drivers/odbc/connect/. [Consultado em: 19-10-2025]. Connect with ODBC Driver - Atlas - MongoDB Docs

MongoDB. *Configure Database Users – Atlas (Database Access / Roles)*. [Em linha]. Disponível em: https://www.mongodb.com/docs/atlas/security-add-mongodb-users/. [Consultado em: 19-10-2025]. Configure Database Users - Atlas - MongoDB Docs

Microsoft. *MongoDB Atlas SQL interface connector (Power Query / Power BI)*. [Em linha]. Disponível em: https://learn.microsoft.com/en-us/power-query/connectors/mongodb-atlas-sql-interface. [Consultado em: 07-11-2025]. MongoDB Atlas SQL interface connector - Power Query | Microsoft Learn

Microsoft. *Use a Personal Gateway in Power BI (on-premises data gateway — personal mode)*. [Em linha]. Disponível em: https://learn.microsoft.com/en-us/power-bi/connect-data/service-gateway-personal-mode. [Consultado em: 07-11-2025]. Use a Personal Gateway in Power BI - Power BI | Microsoft Learn

Prophet (Meta). *Quick Start — Prophet documentation*. [Em linha]. Disponível em: https://facebook.github.io/prophet/docs/quick_start.html. [Consultado em: 14-12-2025]. Quick Start | Prophet

Adafruit. *Adafruit BME280 Library — Documentation (CircuitPython)*. [Em linha]. Disponível em: https://docs.circuitpython.org/projects/bme280/en/latest/. [Consultado em: 28-12-2025]. Introduction — Adafruit BME280 Library 1.0 documentation

13. Anexos

Organograma Inicial após 1ª revisão

CTeSP INF - UC PROJETO INTEGRADO 2025/26			Semana actual:	1			
PROJETO ForecastLab[ABT]			INICIO	FINAL	% CUMPRIMENTO	DIAS RESTANTES	NOTAS / OBSERVAÇÕES
			#####	16-dez-25	31%	-15	O projeto ForecastLabABT visa criar uma plataforma integrada para a cidade de Abrantes que combina dados meteorológicos históricos (10 anos), recolha em tempo real através de websites públicos e de um sensor local, e a construção de modelos preditivos de curto prazo (7–10 dias), sazonais (1–3 meses) e de longo prazo (10 anos), permitindo comparar previsões com observações reais, identificar desvios e disponibilizar resultados, acessíveis local e remotamente, apoiando decisões de planeamento com base em informação meteorológica fiável e visualmente clara.
OBJETIVOS	FERRAMENTAS	RESULTADO ESPERADO					
00 — Definição do Projeto - Propôr projeto a desenvolver durante o	**Documento de requisitos, Proxmox, D**	**Âmbito, objetivos e arquitetura defi**	#####	30-set-25	100%	OK	
Definir âmbito, objetivos e entregáveis	Word/Excel	Documento inicial			100%	OK	
Preparar arquitetura técnica (UM880, Pi, Power BI)	Proxmox, Raspberry Pi, Power BI	Infraestrutura planeada			100%	OK	
Criar repositório inicial e diretórios no servidor	Docker	Estrutura base criada			100%	OK	
01 - API & Monitorização	**Flask, Docker, Cloudflare**	**API pública segura**	1-out-25	14-out-25	33%	-78	
Configurar segurança básica (token + Cloudflare Tunnel) - criar dominio	Cloudflare, HTTPS	Acesso remoto seguro			100%	OK	.json onde inclua historico 2 anos, acrescentar dados diarios, juntar 5 localidades de Portugal, dados estatisticos, tendências e previsões a futuro
Criar API Flask (criar serviço web com páginas que devolvem dados necessários (histórico, tempo real, etc)	Flask, Python	API funcional			100%	OK	
Criar página de teste para garantir que a API e base de dados estão a funcionar	Flask	Monitorização simples			0%		
Validar tempos de resposta e fiabilidade	cURL, logs	Testes OK			0%		
02 — Carregar dados Tempo Real (3×/dia)	**Python, Docker, Raspberry Pi**	**Dados contínuos em tempo real**	#####	29-out-25	50%	-63	
Criar script para recolha de dados públicos	Python, cron	Inserção 3×/dia em Mongo			100%	OK	
Configurar Raspberry Pi 5 com sensor BME280 para recolha de dados em tempo rea localmente	Raspberry Pi, Python BME280	Leituras locais 3×/dia			0%		
Criar jobs automáticos em Docker com scheduler	Docker, cron	Pipelines automatizados			0%		
Validar inserção de dados em MongoDB, consultas simples para confirmação de dados	MongoDB	Dados confirmados			100%	OK	
03 — Dados & ETL Histórico	**Python, Pandas, MongoDB**	**Histórico 10 anos carregado**	#####	12-nov-25	33%	-49	
Recolher datasets meteorológicos (IPMA, NOAA, ECMWF)	APIs públicas, Python	Ficheiros CSV/JSON			100%		
Normalizar formatos e converter para MongoDB	Python ETL, MongoDB	Dados uniformizados			0%		
Criar índices por timestamp para otimizar consultas	MongoDB	Consultas rápidas			0%		
04 — Modelos Preditivos	**Python (Prophet, ARIMA), Pandas**	**Modelos operacionais**	#####	26-nov-25	0%	-35	
Modelo curto prazo (7–10 dias)	Prophet/ARIMA	Forecast diário			0%		
Previsões sazonais (1–3 meses, percentis/anomalias)	Python, estatística	Seasonal outlook			0%		
Climatologia/tendências (10 anos)	Python, MongoDB	Séries climatológicas			0%		
Validar resultados com métricas	Python, Scikit-learn	Relatório de precisão			0%		
05 — Dashboards Power BI	**Power BI, MongoDB**	**Dashboards interativos**	#####	4-dez-25	0%	-27	
Criar dashboard de histórico (10 anos)	Power BI	Série temporal completa			0%		
Criar dashboard de tempo real (web + sensor)	Power BI	Leituras atualizadas			0%		
Criar dashboard de previsões (curto prazo + sazonal + clima)	Power BI	Visualização preditiva			0%		
Testar exportações (Excel/PDF)	Power BI	Funcionalidade validada			0%		
06 — Qualidade & Relatório Final	**Word, PowerPoint, Power BI**	**Projeto entregue**	5-dez-25	16-dez-25	0%	-15	
Rever métricas de precisão e ingestão	Python, MongoDB	Dados e previsões validadas			0%		dia 13 entrega e dia 16 defesa
Redigir relatório académico com anexos técnicos	Word	Relatório em PDF			0%		
Preparar apresentação e demo remota (UM880 acessível)	PowerPoint, Cloudflare	Ensaio final			0%		
Ensaiar defesa e validação final	Apresentação ao docente	Projeto aprovado			0%		

Organograma projeto final (revisto e ajustado à realidade para comparação com a previsão inicial)

CTeSP INF - UC PROJETO INTEGRADO 2025/26				Semana actual:	1			
PROJETO abrantes_meteo	**AÇÕES**			**INICIO**	**FINAL**	**% CUMPRIMENTO**	**DIAS RESTANTES**	**NOTAS / OBSERVAÇÕES**
				16-set-25	6-jan-26	95%	8	Inicialmente projetei um sistema que serviria para prever as necessidades de produção e custos de uma organização. Com base na análise de dados históricos e previsionais, permitiria construir modelos preditivos para estimar necessidades de produção, recolheria e compararia essa informação com dados em tempo real, por forma a contruir uma estrutura de informação de apoio aos gestores na tomada de decisões com dashboards e alertas.
OBJETIVOS	**TAREFAS**	**FERRAMENTAS**	**RESULTADO ESPERADO**					
O0 — Enquadramento e Arquitetura do Projeto				**16-set-25**	**19-set-25**	**100%**	**OK**	
Definir requisitos, objetivo e arquitetura (Docker -> Atlas -> Power BI -> Web)	Âmbito e objetivos do sistema; modelo de dados (Real/Previsto); periodicidade (recolha de dados); cadeia de consumo (Atlas SQL/ODBC -> Power BI) e critérios de validação. Ferramentas: MS Word/diagramas, MongoDB Atlas, Power BI. Resultado esperado: arquitetura validada e plano de execução.			16-set-25	19-set-25	100%	OK	Definição do projeto final e ferramentas a utilizar para obter o resultado pretendido
O1 — Infraestrutura Local (Docker-ct) e MongoDB				**22-set-25**	**10-out-25**	**100%**	**OK**	
Criar estrutura do projeto e ficheiros base (.env, pastas, keyfile)	Criar pastas e subpastas na doker - Resultado esperado: infraestrutura local reprodutível, com volumes persistentes e Mongo saudável.	Debian Linux; Docker Compose; OpenSSL; MongoDB 7	Infraestrutura local operacional (MongoDB com replicaset/auth) e pronta para ingestão.	22-set-25	26-set-25	100%	OK	Estrutura criada no docker-ct, na pasta /mnt/storage/abrantes_meteo, num servidor local com Proxmox a correr uma versão de Linux Debian, onde correm os scripts necessários para a ingestaõ de dados inicias bem como as atualizações diárias planificadas
Configurar docker-compose (MongoDB 7 + setup + app) e volumes persistentes				29-set-25	3-out-25	100%	OK	
Deploy local, healthcheck e resolução de falhas (keyfile/permissões/replicaset)				6-out-25	10-out-25	100%	OK	
O2 — Ingestão, Normalização e Previsão (Python + Prophet)				**13-out-25**	**7-nov-25**	**100%**	**OK**	
Implementar ingestão diária (Open-Meteo) e upsert em meteo_historico	Desenvolver app/app.py: importar histórico (desde 01-01-2023), uniformizar campos, gerar previsões (365 dias) e guardar por execução (run_id).	Python 3.11; requests; pandas; pymongo; Prophet/cmdstanpy	Coleções localmente preenchidas (histórico + forecast) e prontas para sincronizar.	13-out-25	17-out-25	100%	OK	
Implementar previsão (Prophet) 365 dias e persistência em meteo_forecast_flat				20-out-25	31-out-25	100%	OK	
Agendar execução diária, logs e correções (encoding/dependências)				3-nov-25	7-nov-25	100%	OK	
O3 — Persistência Cloud (Atlas) e Sincronização				**10-nov-25**	**14-nov-25**	**100%**	**OK**	
Criar cluster no MongoDB Atlas e configurar acesso (Network + Users)	Configurar MongoDB Atlas: criar cluster e utilizadores, permitir IP, ligar via ATLAS_URI (mongosh) e sincronizar histórico + previsões.	MongoDB Atlas; mongosh; Docker logs	Persistência cloud operacional (meteo_historico e meteo_forecast_flat) com sync diário.	10-nov-25	10-nov-25	100%	OK	Criação de cluster e coleções. Importação dos dados do mongo local para Atlas, criação de utilizadores. Paralelamente instalação do Power Bi Connector e driver Mongo Atlas ODBC, por forma a configurar o DSN e assim lifar ao Power BI
Configurar ATLAS_URI no .env e sincronização (lookback + run_id)				13-nov-25	13-nov-25	100%	OK	
Validar integridade (contagens e últimos registos) local vs Atlas				13-nov-25	14-nov-25	100%	OK	
O4 — Exposição SQL/ODBC (Data Federation + Atlas SQL)				**17-nov-25**	**21-nov-25**	**100%**	**OK**	
Criar Data Federation e mapear coleções do cluster	Criar Data Federation, mapear coleções e gerar SQL Schemas; configurar ODBC/DSN (VirtualDatabase) e validar ligação no Power BI	Atlas Data Federation; Atlas SQL; ODBC Driver	Tabelas expostas via SQL/ODBC com colunas e tipos reconhecidos pelo Power BI.	17-nov-25	21-nov-25	100%	OK	
Gerar SQL Schemas (Atlas SQL) para expor colunas ao ODBC/Power BI				17-nov-25	21-nov-25	100%	OK	
Configurar driver ODBC + DSN e validar consulta no Power BI Navigator				17-nov-25	21-nov-25	100%	OK	
O5 — Power BI (Modelo, DAX, Dashboards e Publicação Web)				**24-nov-25**	**26-dez-25**	**100%**	**OK**	
Importar dados no Power BI via ODBC e normalizar tipos no Power Query	Power BI: importar via ODBC, tratar no Power Query, criar medidas DAX (Real vs Previsto), publicar e configurar refresh + embed público.	Power BI; Power Query; DAX; ODBC; Gateway; HTML/CSS/JS	Dashboards (Real vs Previsto) publicados e prontos para consulta/refresh.	24-nov-25	26-dez-25	100%	OK	Importar coleção do Atlas e normalizar os tabelas no power query, criação de medidas paar usar nos dashboards
Modelar (calendário e relações) e criar medidas DAX (Real vs Previsto)				24-nov-25	26-dez-25	100%	OK	
Dashboards, publicação, refresh e integração web (embed Power BI)				24-nov-25	26-dez-25	100%	OK	
O6 — Validação Final, Documentação e Defesa				**27-dez-25**	**3-jan-26**	**67%**	5	
Testes end-to-end e verificação do scheduler/refresh	Validação final; consolidar documentação/anexos e preparar apresentação/demonstração.	Docker; mongosh; MS Word/PDF;Browser	Entrega final: pipeline operacional + documentação completa + defesa preparada.	27-dez-25	29-dez-25	100%	OK	A apresentação será feita via browser numa página criada para o efeito e onde estão incluidos os dashboards do power bi publicados via web e sem restrições de login
Relatório académico e anexos (ficheiros, prints e evidências)				29-dez-25	30-dez-25	100%	OK	
Apresentação e defesa (website + demonstração)				3-jan-26	6-jan-26	0%	NÃO INICIADA	

Politécnico de Tomar
Escola Superior de Tecnologia de Abrantes

www.ipt.pt