Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Integração de IoT e Aprendizagem Automática num Smart Campus para a eficiência energética.

Nuno Manuel Pina Rolo

Orientador(es) | Pedro Salgueiro

Vitor Beires Nogueira

Évora 2023

Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Integração de IoT e Aprendizagem Automática num Smart Campus para a eficiência energética.

Nuno Manuel Pina Rolo

Orientador(es) | Pedro Salgueiro

Vitor Beires Nogueira

Évora 2023

A dissertação foi objeto de apreciação e discussão pública pelo seguinte júri nomeado pelo Diretor da Escola de Ciências e Tecnologia:

Presidente	|	Teresa Gonçalves (Universidade de Évora)
Vogais	|	Luís Rato (Universidade de Évora) (Arguente) Pedro Salgueiro (Universidade de Évora) (Orientador)

Évora 2023

Aos meus pais

Agradecimentos

Agradeço aos meus pais e ao meu irmão, pelo apoio prestado durante o desenvolvimento desta dissertação, pois sem o seu apoio, não seria possível.

Agradeço também aos orientadores, Prof. Pedro Salgueiro e Prof. Vítor Nogueira, pelo seu apoio, motivação, conhecimento, partilha de ideias e disponibilidade, pois sem eles o trabalho desenvolvido nesta dissertação também não seria possível.

Por último, agradeço também ao Prof. Luís Rato e à Prof. Teresa Gonçalves, pois foram fundamentais na investigação, análise e construção de um modelo de predição do consumo energético.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

HTTP Hypertext Transfer Protocol

SSL Secure Sockets Layer

MQTT Message Queuing Telemetry Transport

CoAP Constrained Application Protocol

NGSI Next Generation Service Interface

NGSI-v2 Next Generation Service Interface - Version 2

PEP Proxy Policy Enforcement Point Proxy

JWT Json Web Tokens

LoRa Long Range

GE Generic Enablers

AR Autoregressive

ARX Autoregressive Exogenous

SAR Seasonal Autoregressive

SARX Seasonal Autoregressive Exogenous

ARMA Autoregressive Moving Average

ARMAX Autoregressive Moving Average Exogenous

SARMA Seasonal Autoregressive Moving Average

SARMAX Seasonal Autoregressive Moving Average Exogenous

ARIMA Autoregressive Integrated Moving Average

ARIMAX Autoregressive Integrated Moving Average Exogenous

SARIMA Seasonal Autoregressive Integrated Moving Average

SARIMAX Seasonal Autoregressive Integrated Moving Average Exogenous

SVM Support Vector Machines

SVR Support Vector Regression

MSE Mean Squared Error

MAE Mean Absolute Error

RMSE Root Mean Square Error

REST Representational state transfer

API Application Programming Interface

JSON JavaScript Object Notation

IoT Internet of Things

AMQP Advanced Message Queuing Protocol

Sumário

A IoT está cada vez mais presente nos dias de hoje, pois existe uma grande quantidade de dispositivos que comunicam os seus dados para o exterior. Para que estes dados possam ser tratados, armazenados e analisados, é necessário seguir algumas abordagens de desenvolvimento e usar tecnologias adequadas, tais como, a integração de diferentes tipos de sensores, o seu armazenamento, a segurança do fluxo dos dados, entre outros.

Nesta dissertação é apresentado um sistema de integração de dados IoT, provenientes de diferentes sensores de medição do consumo energético instalados no pólo da Mitra da Universidade de Évora, e de um inversor solar do sistema fotovoltaico instalado no pavilhão Gimnodesportivo da Universidade de Évora, para a medição da produção de energia.

Para além desta integração de dados, é também apresentada uma análise à construção de um modelo de predição do consumo energético para um dos sensores, onde foram experimentadas diferentes abordagens e modelos de predição adequados a dados temporais.

Palavras chave: IoT, Campus inteligente, Aprendizagem Automática

Abstract

Using IoT and Machine Learning in a Smart Campus for Energy Eficiency

IoT is increasingly present nowadays, as there are a large number of devices that communicate their data to the outside world. In order for this data to be processed, stored and analyzed, it is necessary to follow some development approaches and use appropriate technologies, such as the integration of different types of sensors, their storage, the security of the data flow, among others.

This dissertation presents an IoT data integration system, coming from different energy consumption measurement sensors installed at the Mitra pole of the University of Évora, and from a solar inverter of the photovoltaic system installed in the Gimnodesportivo pavilion of the University of Évora, for the measurement of energy production.

In addition to this data integration, an analysis of the construction of an energy consumption prediction model for one of the sensors is also presented, where different approaches and prediction models suitable for temporal data were experimented.

Keywords: IoT, Smart Campus, Machine Learning

1

Introdução

Neste Capítulo é apresentada a Introdução ao trabalho realizado nesta dissertação, relacionado com a integração de dados IoT num Smart Campus, bem como a aplicação de métodos de aprendizagem automática sobre estes dados.

Nos últimos anos, as preocupações ambientais, alterações climáticas, escassez de recursos naturais, entre outras, têm ganho uma relevância maior, pelo que o papel da tecnologia, mais especificamente o IoT, tem um papel importante, ao permitir otimizar recursos e consequentemente o melhoramento da eficiência energética [HMMHZ20].

O trabalho apresentado nesta dissertação, tem como objetivo principal, a investigação e implementação de um sistema de integração de dados IoT num Smart Campus, em particular o da Universidade de Évora. O conceito de Smart Campus refere-se a instituições que implementem dispositivos com capacidade de comunicação, e com isso possibilitar a análise de dados, para melhorar os serviços da instituição, neste caso a Universidade de Évora [sma]. Estes dados são obtidos de diferentes tipos de dispositivos, como os sensores que medem o consumo energético, e os dispositivos que medem a produção de energia. Um outro objetivo deste trabalho, é a construção de um modelo de aprendizagem automática, para predição do consumo energético futuro com base no seu histórico.

Estes dois objetivos, visam otimizar recursos e melhorar a eficiência energética, através da disponibilização de ferramentas de análise de dados.

1.1 Estrutura

A estrutura desta dissertação divide-se em 5 Capítulos, os quais são:

Capítulo 1 - Introdução Neste Capítulo é apresentada a introdução ao trabalho desenvolvido nesta dissertação, bem como é feita uma breve introdução aos seus Capítulos.

Capítulo 2 - Estado da Arte: Neste Capítulo são apresentados diferentes aspetos para o desenvolvimento de um sistema de integração de dados IoT, como diferentes plataformas de desenvolvimento, sistemas de base de dados, plataformas de processamento de dados, protocolos de comunicação, entre outros.

Capítulo 3 - Arquitetura: Neste Capítulo são apresentados todos os componentes que compõem o sistema de integração de dados IoT.

Capítulo 4 - Predição do consumo energético: Neste Capítulo são apresentadas diferentes abordagens de predição do consumo energético, bem como a construção de modelos de aprendizagem automática para a predição.

Capítulo 5 - Conclusão e Trabalho Futuro: Neste Capítulo é apresentada a conclusão e aspetos finais desta dissertação, bem como o trabalho futuro.

2

Estado da Arte

Neste Capítulo é apresentado o estado da arte relacionado com o tema desta dissertação, onde são abordados os seguintes tópicos: plataformas de desenvolvimento, bases de dados, processamento de dados, comunicação, e aprendizagem automática.

Com o evoluir da tecnologia, existem cada vez mais dispositivos com capacidade de comunicação, sendo os sensores de IoT um bom exemplo, pois eles recolhem informação do local onde estão instalados, e transmitem esses dados para poderem ser armazenados, tratados, analisados, entre outros. Esta transmissão de dados de diferentes sensores em diferentes áreas de aplicação, abre um mundo de oportunidades para a gestão, monitorização e análise dos dados, como é o caso do tema deste dissertação, que aborda a integração de dados IoT num Smart Campus para a eficiência energética, sendo o pólo da Mitra da Universidade de Évora utilizado como objeto de estudo.

Neste Capítulo, são apresentados diferentes tópicos relevantes para o desenvolvimento de um sistema de integração de dados IoT e seu processamento. Na Secção 2.1 são apresentadas várias plataformas de desenvolvimento. Na Secção 2.2 são apresentados diferentes sistemas de bases de dados para diferentes propósitos. Na Secção 2.3 são apresentadas algumas plataformas de processamento de dados. Na

Secção 2.4 são apresentados alguns protocolos de comunicação indicados para IoT. Na Secção 2.5 são apresentados alguns conceitos e modelos relacionados com aprendizagem automática para predição de séries temporais.

2.1 Plataformas de Desenvolvimento

Para a construção de um fluxo de dados, desde a origem dos mesmos, até às fases finais de armazenar os dados persistentemente, o processar os dados, aplicar algoritmos de aprendizagem automática, entre outros, existem algumas plataformas que ajudam neste processo, disponibilizando diferentes componentes para os diferentes propósitos neste fluxo. Nesta Secção são analisadas algumas plataformas adequadas a este tipo de dados e fluxos.

2.1.1 Fiware

O Fiware [FIWa] é uma plataforma de código aberto, apoiada pela União Europeia para a transformação digital [Eur], cujo objetivo é facilitar o desenvolvimento de soluções de diferentes áreas relacionadas com IoT, como, por exemplo, soluções energéticas, agroalimentares, industriais, sendo um dos seus focos o desenvolvimento de cidades ou campus inteligentes. O Fiware fornece uma arquitetura onde, cada um dos componentes disponibiliza uma Application Programming Interface (API) para a sua interação, chamados Generic Enablers (GE).

Existem alguns projetos desenvolvidos com esta plataforma em diferentes áreas, como o desenvolvimento do Smart Campus no Instituto Politécnico de Viana do Castelo [MLC21], que consiste em medições inteligentes de dados vindos de sensores, como, por exemplo, medição de água consumida, gestão de produção de energia, e análise da qualidade do ar nos edifícios da instituição. Estes sensores, comunicam os dados de diferentes formas, por WiFi, 3G/4G e LoRaWAN, para o IoT Agent, que é um GE do Fiware, utilizado como ponte entre os dispositivos e o fluxo de dados. Nos dispositivos mencionados, foi utilizado uma abordagem que fornece alguma inteligência ao dispositivo, para, por exemplo, detetar anomalias nos valores lidos pelos sensores de forma imediata.

A plataforma Fiware disponibiliza vários módulos para construir um sistema de processamento de dados IoT desde a entrada de dados até ao seu processamento e armazenamento. Devido à sua arquitetura modular, é possível também integrar um módulo externo na plataforma, através das API que cada módulo disponibiliza. Nas Secções seguinte são apresentados alguns módulos da plataforma, que permitem a receção de dados, o seu processamento, armazenamento, entre outros.

IoT Agent

O IoT Agent é um componente do Fiware, que tem como objetivo a recolha dos dados dos dispositivos a ele conectados e a transmissão desses mesmos dados para o Fiware Orion Context Broker, onde a comunicação é feita através do protocolo de comunicação Next Generation Service Interface - Version 2 (NGSI-v2) utilizado pelo Orion Context Broker.

Existem vários IoT Agents para diferentes protocolos de comunicação, como, por exemplo, MQTT, LoRaWAN, entre outros. [1].

[1] https://github.com/FIWARE/catalogue/blob/master/iot-agents/README.md

Fiware Orion Context Broker

O Fiware Orion Context Broker [2] pode ser considerado o módulo principal da plataforma Fiware, sendo responsável por gerir os dados recebidos e guardar informação de contexto dos elementos envolvidos, numa base de dados NoSQL MongoDB. Esta informação de contexto pode ser, por exemplo, a localização de um sensor e alguns dados referentes ao mesmo. Na informação de contexto que o Orion Context Broker gere, é também guardado o último estado de cada dispositivo, ou seja o último valor lido, que depois pode ser utilizado por outros módulos do Fiware. Este módulo permite também um mecanismo de subscrições, para que quando o estado de um sensor é alterado, notificar um serviço externo, que pode ser, ou não, um serviço do Fiware.

QuantumLeap

O QuantumLeap é um componente do Fiware, que tem como objetivo guardar de forma persistente os dados recolhidos dos dispositivos, ou seja, é um serviço que subscreve as alterações de estado do Orion Context Broker, e a cada alteração guarda numa base de dados. As base de dados que o QuantumLeap suporta para a persistência dos dados, são o CrateDB e o Timescale, sendo que se for necessário guardar noutro sistema de base de dados, é necessário implementar um serviço para subscrever as alterações do Orion Context Broker, e guardar onde for necessário.

O QuamtumLeap permite ainda a ligação a um sistema de cache, por forma a otimizar as consultas à base de dados. O único sistema de cache que suporta é o Redis, que guarda dados em memória numa base de dados não relacional no formato chave-valor de cinco tipos diferentes [Fiwb]. O Redis suporta replicação para melhorar o desempenho de leitura, e suporta também "client-sharding" para melhorar o desempenho de escrita [Car13].

Segurança

A segurança dos sistemas de IoT, e não só, é cada vez mais importante, pelo que o Fiware disponibiliza um conjunto de componentes para proteger o sistema e também para o controlo de acessos para vários utilizadores baseado em permissões.

Estes componentes são:

- **PEP Proxy**, que consiste numa camada intermédia entre o cliente e o servidor, visando proteger o destinatário, sendo que os pedidos têm de passar pelo proxy, que implementa validações de segurança, criando assim uma "barreira" contra acessos indesejados.

  No Fiware, existe uma implementação de um proxy, chamada Fiware Wilma, que funciona em conjunto com outro componente, o Fiware Keyrock, responsável pela gestão dos utilizadores e pelas suas permissões para os recursos do Fiware. Assim, ao realizar pedidos aos GE do Fiware, é necessário passar por uma fase de autenticação no Fiware Keyrock, e com uma chave de acesso, temos então permissão para interagir com os GE do Fiware.

- **Keyrock** O Keyrock, como referido anteriormente, consiste na gestão e segurança de utilizadores da plataforma Fiware, permitindo assim a sua autenticação e utilização dos diferentes recursos a que tiver acesso.

[2] Fiware, Welcome To Orion Context Broker: https://fiware-orion.readthedocs.io/en/master/

De acordo com [TOMdOS+18], em 2018 foram identificados alguns problemas de segurança na plataforma Fiware, mais especificamente no componente IoT Agent, onde a comunicação entre um dispositivo/sensor e o IoT Agent, e depois a comunicação com o Orion Context Broker, não utilizava nenhum mecanismo de segurança, ou seja, existia a exposição de informação sensível e da possibilidade de interação com um dispositivo.

Este mesmo artigo, propôs uma implementação de segurança, onde os pedidos eram encriptados utilizando uma de duas abordagens, TLS para o protocolo TCP, ou DTLS para o protocolo UDP. Com esta alteração, as mensagens transmitidas entre o dispositivo e o IoT Agent, ou vice-versa, são encriptadas, e em cada mensagem, era feita a verificação do certificado ("handshake").

Desempenho

De acordo com um estudo realizado em 2019 [AMSÅ19], foi avaliado o desempenho da plataforma Fiware em diferentes cenários de carga, tendo em consideração também a escalabilidade horizontal e vertical.

A metodologia utilizada nesta avaliação de desempenho, baseou-se em 4 aspetos, que foram eles:

1. Utilização de Elastic Compute Cloud da Amazon, como ambiente de testes, onde foram utilizadas diferentes tipos de máquinas: *m4.large*, *c4.xlarge* e *c4.2xlarge*.

2. Quantificação de desempenho baseado no tempo de registo dos dados à base de dados MongoDB, responsável por guardar todas as atualizações realizados pelos sensores, pela utilização de recursos, como CPU e memória, e pela latência de consultas ao Orion Context Broker.

3. Simulação de dados de sensores IoT, enviando dados para a plataforma com os protocolos de comunicação utilizados em IoT. Esta simulação foi feita através do software Apache JMeter que possibilita a criação de carga na plataforma, e utiliza o modo distribuído para simular grandes quantidades de pedidos em simultâneo.

4. Utilização de protocolos de comunicação utilizados em IoT, que foram MQTT e Constrained Application Protocol (CoAP).

Quanto aos resultados obtidos por estes testes de desempenho, concluíram que com a configuração sem qualquer escalabilidade horizontal e vertical, o componente IoT Agent, a partir dos 1000 pedidos por segundo, começa a dar problemas, sendo este o componente bloqueante de todo o fluxo de dados.

Contudo, ao utilizar escalabilidade vertical, verificou-se uma melhoria de desempenho de cerca de 30%, mas não proporcional às instâncias de computação utilizadas, pois a capacidade dessas instâncias era 4 vezes superiores.

Quanto ao uso de escalabilidade horizontal no componente Orion Context Broker e na base de dados MongoDB, verificou-se que o desempenho foi inferior, fazendo com que o IoT Agent apresentasse problemas mais cedo. A justificação dada para este problema, foi que o componente Orion Context Broker não era responsável pelo problema de desempenho, pois ele não diminuía a pressão da base de dados, sendo que a sua escalabilidade horizontal não ajudou em nada. Quando se utilizou escalabilidade horizontal na base de dados MongoDB, o sistema começou a suportar mais de 3000 pedidos por segundo, mostrando uma melhoria de desempenho considerável.

2.1.2 DeviceHive

O DeviceHive é uma plataforma de código aberto, que fornece alguns componentes de comunicação e bibliotecas para diferentes linguagens de programação, para a integração de dispositivos IoT.

Esta plataforma é baseada em microsserviços que disponibilizam uma API para a interação com os diferentes componentes, e que permite escalabilidade e alta disponibilidade.

A arquitetura utilizada pelo DeviceHive consiste em conectores para diferentes dispositivos com diferentes protocolos de comunicação (MQTT, Representational state transfer (REST), etc). A informação enviada pelos dispositivos é transmitida por Apache Kafka, para que, com plugins personalizados e bibliotecas que o DeviceHive disponibiliza, utilizar esses dados para outros fins. Esta arquitetura utiliza como sistema de base de dados o PostgreSQL, para guardar informação de contexto, em conjunto com o serviço de cache, Hazelcast, que guarda todos os dados temporais durante 2 minutos para que o acesso seja o mais rápido possível.

Para guardar os dados temporais persistentemente, o DeviceHive não disponibiliza nenhum componente em concreto, mas considerando que os dados são transmitidos através do Apache Kafka, é possível criar um plugin personalizado para ler estes dados, e guardá-los na base de dados pretendida.

O DeviceHive disponibiliza também uma interface web para interação com os dispositivos.

Segurança

A segurança desta plataforma é conseguida através de Json Web Tokens (JWT), que contém informação das permissões do utilizador, e os dispositivos a que ele tem acesso. Este JWT é gerado através de uma API que o DeviceHive disponibiliza para a autenticação, validação e revalidação dos JWT.

Toda a comunicação dentro do DeviceHive pode ser configurada para utilizar SSL.

2.1.3 ThingsBoard

O ThingsBoard é uma plataforma IoT para a gestão de projetos nesta área. A arquitetura do ThingsBoard foi desenhada para ser escalável, tolerante a falhas, eficiente e customizável. [3]

O ThingsBoard, à semelhança das outras plataformas já analisadas, também disponibiliza diferentes componentes desde os sensores até a uma fase final dos dados. Existem várias versões, sendo que apenas a versão comunitária é gratuita. A instalação pode ser feita localmente através de vários métodos (Docker, Ubuntu, Windows, etc), sendo que as versões profissionais têm a opção de utilizar a cloud e consequentemente tem custos acrescidos. As principais diferenças entre a versão profissional e a comunitária, é que na versão profissional existem mais integrações, um sistema de agendamento de eventos, entre outros. [4]

Esta plataforma disponibiliza diferentes protocolos de comunicação como entrada de dados, como, por exemplo, MQTT, HTTP, CoAP, entre outros. No caso do MQTT disponibiliza um Gateway, em que o seu objetivo é servir de uma ponte entre os dispositivos e o ThingsBoard, e que permite vários dispositivos enviarem mensagens para o gateway, sendo que a comunicação de entrada no ThingsBoard é apenas uma conexão MQTT.

[3] https://thingsboard.io/docs/reference/
[4] https://thingsboard.io/products/thingsboard-pe/

Para a persistência dos dados, o ThingsBoard disponibiliza diferentes abordagens, uma delas é a utilização de apenas uma base de dados PostgreSQL para guardar toda a informação, e a outra é uma abordagem que utiliza dois sistemas de base de dados para propósitos diferentes, ou seja, PostgreSQL para guardar informação de contexto sobre os dispositivos, entre outros, e Timescale para os dados temporais. [5]

ThingsBoard Core

O ThingsBoard Core é responsável por tratar pedidos REST e subscrições de WebSockets. Este componente é responsável também por guardar informação sobre as sessões de utilizadores ativas, e o estado de conexão dos dispositivos a ele ligados. [3]

ThingsBoard Rule Engine

O ThingsBoard Rule Engine é considerado o componente principal da arquitetura, responsável pelo tratamento de mensagens recebidas. Este componente subscreve os dados recebidos de uma fila, e só após os dados serem processados é que a mensagem é confirmada para continuar no fluxo de dados.

2.2 Bases de Dados

Um dos requisitos principais num sistema de integração de dados IoT é o armazenamento dos dados ao longo do tempo, sendo que para guardar os dados lidos pelos sensores, é necessário um sistema de gestão de bases de dados adequado ao seu tipo. Neste caso, é necessária uma base de dados para guardar os dados temporais, e outra para guardar informação relacionada a cada sensor. Nesta Secção, são analisados alguns sistemas de base de dados, adequados para os diferentes tipos de dados.

2.2.1 MongoDB

O MongoDB é uma base de dados NoSQL de código aberto, que permite armazenar dados semi-estruturados e não estruturados onde a sua estrutura é baseada em documentos JavaScript Object Notation (JSON). Este sistema de base de dados, é utilizado pela plataforma Fiware no componente Orion Context Broker para guardar informação de contexto dos dispositivos, pelo que para dados temporais existem melhores opções, apresentadas abaixo.

O MongoDB permite escalabilidade horizontal e vertical, bem como alta disponibilidade, o que se traduz num bom desempenho [Cha19] [Mon].

2.2.2 InfluxDB

O InfluxDB é um sistema de base de dados de séries temporais de código aberto, em que o seu propósito é lidar com grandes volumes de dados temporais. O InfluxDB possui duas versões, uma de código aberto, e outra empresarial com mais funcionalidades. A principal diferença destas duas versões, é a escalabilidade horizontal e a alta disponibilidade, apenas estar disponível na versão empresarial, sendo que na versão de código aberto está limitada apenas a escalabilidade vertical [NYZ17].

[5] https://thingsboard.io/docs/reference/#sql-vs-nosql-vs-hybrid-database-approach

O InfluxDB permite a realização de operações na base de dados através de duas linguagens, Flux e InfluxQL, sendo as duas proprietárias da InfluxData, Inc [Inf].

Em termos de desempenho, com base neste estudo [Crab] de comparação entre o InfluxDB e o CrateDB, podemos concluir que o InfluxDB apresenta melhores resultados quando é utilizado apenas 1 nó, sendo que quando existem vários clientes em simultâneo, o desempenho é inferior em comparação com o CrateDB.

2.2.3 CrateDB

O CrateDB é um sistema de base de dados SQL distribuído de código aberto, que permite armazenar qualquer tipo de dados, ou seja, estruturados em tabelas relacionais, e não estruturados em documentos JSON, sendo um dos seus focos armazenar dados temporais em tempo real [Craa] [Mat].

Os pontos fortes deste sistema de base de dados, é que permite escalabilidade vertical, horizontal e alta disponibilidade, permitindo a adição de mais nós, conforme as necessidades, e a recuperação ao estado normal após uma falha nalgum nó [Craa].

Neste caso, apenas existe a versão de código aberto, que nos permite ter alta disponibilidade e escalabilidade, sem custos adicionais, ao contrário do InfluxDB.

Conforme o estudo realizado pelo CrateDB, para avaliar o desempenho entre diferentes sistemas de bases de dados, TimescaleDB e InfluxDB, num caso de uso IoT, onde eram gerados dados a cada meio segundo provenientes de 5000 sensores, concluíram que dos três sistemas avaliados, o CrateDB foi o que utilizou menos espaço em disco, e foi o que apresentou a execução de uma consulta SQL mais rápido.

2.2.4 Timescale

O Timescale é um sistema de base de dados de séries temporais e relacional de código aberto, que utiliza como base o PostgreSQL, sendo que as suas modificações relativas à base, permitem um melhor desempenho para dados temporais, sendo esse o seu propósito. O Timescale suporta nativamente a linguagem SQL, simplificando o processo de consulta e registo de dados.

2.2.5 Apache Cassandra

O Apache Cassandra é uma base de dados NoSQL distribuída de código aberto, sendo o seu principal objetivo o armazenamento de grandes quantidades de dados. Uma característica importante do Apache Cassandra é que as bases de dados são distribuídas, ou seja, pode executar em mais do que uma máquina, prevenindo assim a perda de dados quando o hardware falha, e possibilita a escalabilidade quando a base de dados está sobre carga.

Para o armazenamento de dados temporais, de acordo com o estudo realizado pela Influx [Inc], foram comparados os sistemas de bases de dados, InfluxDB e Apache Cassandra, através de testes de inserção, compressão e consulta de dados, sendo que o InfluxDB obteve melhor desempenho neles todos.

2.2.6 MySQL

O MySQL é um sistema de base de dados relacional SQL de código aberto, utilizado pelo componente do Fiware, Keyrock Identity Management, por forma a guardar os dados de utilizadores, permissões de acessos, entre outros.

Devido à sua estrutura transacional, este sistema de base de dados é indicado para aplicações que requerem integridade e consistência dos dados [Roy]. Permite ainda a escalabilidade horizontal e vertical, sendo que existe uma versão empresarial, que contém recursos mais avançados de escalabilidade, do que a versão de código aberto [MyS].

2.3 Processamento de Dados

Quando existe entrada de dados num formato específico de cada sensor, é necessário realizar alguma transformação dos mesmos, por forma a uniformizar os dados de diferentes sensores. Para que isto seja possível, é necessária uma abordagem de processamento de dados, não só para a transformação dos dados de origem, mas também para a aplicação de algoritmos de aprendizagem automática em tempo real, ou outro caso de uso.

Nesta Secção, são analisadas algumas tecnologias de processamento de dados.

2.3.1 Apache Kafka

O Apache Kafka, é uma plataforma de código aberto, em que o seu objetivo é a transmissão e processamento de fluxos de dados, com baixa latência, bom desempenho e escalável [Kaf]. Existem alguns conceitos principais ligados ao Apache Kafka, os quais são, produtor, partições, tópicos e consumidor. O produtor pode ser considerado o ponto de entrada no fluxo de dados, sendo ele que envia os dados. Os dados provenientes do produtor podem ser distribuídos em diferentes partições e diferentes tópicos. Em cada partição as mensagens estão ordenadas pela sua posição, e indexadas pela sua data e hora. O consumidor pode ser considerado o ponto final do fluxo de dados, pois é ele que lê os dados guardados nas partições.

KSQL

O KSQL pode ser considerado um complemento à plataforma Apache Kafka, pois permite processar os dados no fluxo do Apache Kafka, de uma forma interativa para, por exemplo, agregar dados, filtragem, associação, entre outros. O KSQL permite esta interação com o fluxo de dados, através de comandos SQL, como, por exemplo, SELECT, WHERE, etc, sendo bastante semelhante a uma base de dados relacional, mas com outro propósito.

Uma funcionalidade interessante do KSQL é que pode ser utilizado para a execução de um modelo de aprendizagem automática [Con19] no fluxo de dados do Apache Kafka, sendo bastante útil para detetar, por exemplo, anomalias em tempo real, ou outras situações.

2.3.2 Apache Spark

O Apache Spark é uma plataforma de código aberto, de processamento de fluxo de dados. O seu objetivo inicial era resolver as limitações do Hadoop MapReduce, onde as operações são feitas em disco, e que no Apache Spark são feitas na memória, resultando num processamento de dados mais rápido. Ele utiliza também uma cache na memória com o objetivo de otimizar o desempenho de algoritmos de aprendizagem automática [AWS]. O Apache Spark, tem ainda um sistema de tolerância a falhas caso aconteça alguma falha na máquina.

O Apache Spark disponibiliza uma API de alto nível, para diferentes linguagens de programação, em que é possível escolher entre Java, Python, R e Scala como linguagem de programação [Spab].

A plataforma Fiware disponibiliza um conector, em que o seu objetivo é criar a ligação entre o Fiware Orion Context Broker e o Apache Spark, para o processamento dos dados em tempo real. Esta ligação permite receber um fluxo de dados com uma frequência de transmissão definida (por exemplo, receber um fluxo de dados a cada 1 minuto) no Apache Spark, e após transformação dos dados, aplicação de algoritmos de aprendizagem automática, entre outros casos de uso, é possível enviar os dados, por exemplo, para o Fiware Orion Context Broker, para a sua persistência.

O trabalho apresentado em [VTACC21], usa o Apache Spark em conjunto com o Apache Kafka para o processamento de dados, com o objetivo de deteção de anomalias, e com base nisso gerar alertas. Outro caso de uso, referido no mesmo artigo, é a predição do tempo de carregamento de baterias com base no histórico de dados.

MLlib

O MLlib é uma biblioteca de aprendizagem automática do Apache Spark, que disponibilizada diferentes ferramentas para a execução de modelos de aprendizagem automática, como, a disponibilização de diferentes algoritmos de aprendizagem automática, como, por exemplo, algoritmos de classificação, regressão, clustering, entre outros, ferramentas para extração de características dos dados, ferramentas para a construção de "pipelines", persistência de modelos, entre outras [Spaa].

2.3.3 Apache Flink

O Apache Flink é uma plataforma de código aberto, de processamento de dados distribuído, mais recente do que o Apache Spark. A sua arquitetura também permite escalabilidade e tolerância a falhas [Mac], para que o processamento dos dados tenha um bom desempenho em função da quantidade de dados [PMOK18].

A principal diferença entre o Apache Flink e o Apache Spark, é que o Apache Flink suporta nativamente o processamento de fluxos de dados (streams) e em lote (batches), enquanto que o Apache Spark, apesar de suportar também o processamento de fluxos de dados, ele utiliza uma abordagem de micro lotes (micro-batching), em que divide o fluxo de dados em pequenos lotes, o que se traduz num desempenho inferior [Mac].

A plataforma Fiware também disponibiliza um componente para a ligação entre o Fiware Orion Context Broker e o Apache Flink, para o processamento de dados em tempo real, tal como acontece com o Apache Spark, sendo que neste caso, a cada evento gerado o Apache Flink recebe de imediato a notificação, em vez de agrupar os dados num determinado espaço temporal.

FlinkML

O Apache Flink, também disponibiliza uma biblioteca de aprendizagem automática (FlinkML), que, em comparação com o MLlib do Apache Spark tem menos funcionalidades e algoritmos disponíveis. Segundo a sua página oficial [6], apenas estão disponíveis os algoritmos "Support Vector Machines (SVM)", "Multiple Linear Regression" e "Optimization Framework" na aprendizagem supervisionada, e o algoritmo "k-Nearest neighbors join" para a aprendizagem não supervisionada, enquanto que na biblioteca MLlib do Apache Spark a lista algoritmos, e a flexibilidade é maior.

2.3.4 Node-RED

O Node-RED é uma ferramenta web de código aberto, cujo objetivo é criar e gerir fluxos de dados, possibilitando assim a conexão entre dispositivos, serviços, entre outros. A criação destes fluxos é feita interativamente, através de diferentes nós que o Node-RED disponibiliza. Estes nós, podem ser a ligação a um dispositivo, uma função a ser executada, o encaminhamento dos dados para um serviço, entre outros. Um caso de uso desta ferramenta, seria o pré-processamento dos dados, ou seja, a transformação dos dados de origem de um dispositivo para um serviço que requer a entrada dos dados num formato específico, como é o caso do componente Fiware IoT Agent, que impõe algumas regras no formato dos dados.

2.4 Comunicação

Na área do IoT, existem diferentes meios e protocolos de comunicação para a transmissão de dados desde os sensores, até ao início do fluxo de dados, e para que estes dados possam ser transmitidos existem também diferentes meios para a comunicação dos dados. Nesta Secção são apresentados alguns protocolos e meios de comunicação.

2.4.1 Meios de Comunicação

Para que os dados possam ser transmitidos, é necessário um meio de comunicação entre os dispositivos e o início do fluxo de dados. Existem diferentes meios de comunicação, onde, de seguida, são apresentados os principais.

Wi-Fi

O Wi-Fi, é uma comunicação sem fios que utiliza ondas de rádio para a transmissão dos dados. Essa transmissão pode ser feita em frequências de 2.4 GHz ou 5 GHz, e utiliza o protocolo de rede IEEE 802.11 que possui vários tipos [Wikb]. Este meio de comunicação é de curto alcance, ou seja, funciona num raio de cerca de 45 metros para a frequência de 2.4GHz e de cerca de 15 metros para a frequência de 5GHz, sendo estes valores muito variáveis, pois dependem de diversos fatores [epb].

[6] https://nightlies.apache.org/flink/flink-docs-release-1.4/dev/libs/ml/

Ethernet

O Ethernet, é uma comunicação com fios que utiliza o protocolo de rede IEEE 802.3, indicada para áreas de rede locais, como, por exemplo, escritórios, escolas, entre outros.

Bluetooth

O Bluetooth, é uma comunicação sem fios, que utiliza ondas de rádio para a transmissão dos dados, e as frequências utilizadas são de curto alcance, variando entre 10 metros e 100 metros [Pes], dependendo da sua versão e das condições. Esta comunicação é também considerada de baixo consumo.

Long Range (LoRa)

O LoRa, é uma comunicação sem fios de rádio frequência que permite comunicações de longo alcance, podendo alcançar uma área de até 100 km^2 [DK18], nas condições ideais, sendo que o seu alcance máximo registado foi de 832 km [Dan]. Esta comunicação fornece também um baixo consumo energético, e é indicada para transferência de poucos dados em curtos intervalos de tempo [Wika]

Rede Móvel

A rede móvel, é também uma comunicação sem fios, e que utiliza ondas rádio, sendo que é distribuída através de antenas da operadora.

Existem várias iterações de rede móvel, sendo que as mais importantes na área deste artigo, terá sido a partir da rede 4G, onde as velocidades de comunicação começaram a ser maiores, sendo neste momento o 5G uma evolução importante para a comunicação, com velocidades, largura de banda e capacidade superiores. As principais diferenças entre as diferentes iterações, são principalmente, velocidade e volume de rede [Eri].

2.4.2 Protocolos de Comunicação

MQTT

O MQTT é um protocolo de comunicação muito utilizado em plataformas IoT, pois é bastante eficiente em termos de recursos. A comunicação consiste no modelo "Publish - Subscribe", em que o produtor de dados envia mensagens num tópico específico, e o consumidor aguarda por mensagens dos produtores que esteja subscrito. Para este protocolo de comunicação MQTT, é necessário um broker MQTT, em que o objetivo é receber as mensagens dos produtores e encaminhá-las para os consumidores que subscrevem o tópico.

Advanced Message Queuing Protocol (AMQP)

O AMQP é um protocolo de comunicação que foi desenhado principalmente para a comunicação entre sistemas, utilizando várias técnicas, como o armazenamento de dados, encaminhamento, entre outros.

As suas principais funcionalidades são a existência de filas de mensagens, fiabilidade, segurança e a troca de mensagens que suporta quatro abordagens: [Nai17]

1. "Direct (point-to-point);
2. "Fanout";
3. "Topic (publish-subscribe)"
4. "Headers (publish-subscribe)"

CoAP

O CoAP é também um protocolo de comunicação indicado para dispositivos com recursos limitados. Este protocolo de comunicação é baseado no modelo REST, pelo que a comunicação é feita por pedidos Hypertext Transfer Protocol (HTTP): GET, PUT, POST e DELETE, semelhante à comunicação com uma REST API. Para a obtenção de dados, por exemplo, de um sensor, o modo de funcionamento é o oposto do MQTT, pois aqui é feito um pedido para ler os dados, enquanto no MQTT, é o próprio sensor que envia os dados.

Ultralight

O Ultralight é um protocolo de comunicação de dispositivos, utilizado pela plataforma Fiware 2.1.1, em que consiste num protocolo simples baseado em texto, permitindo assim uma maior eficácia na transmissão de dados na rede. Este protocolo é indicado para os casos onde os recursos são limitados, principalmente largura de banda e memória do dispositivo. [7]

Modbus TCP

O Modbus TCP é um protocolo de comunicação utilizado principalmente em sistemas de automação industrial, que é baseado no protocolo original Modbus, em que a comunicação era feita por serial. Este protocolo, é uma adaptação do Modbus, que permite que a comunicação seja feita através de rede TCP, num formato próprio de mensagens.

2.5 Aprendizagem Automática

Na área de aprendizagem automática, existem cada vez mais modelos, técnicas e abordagens, para a predição de dados. No caso do IoT é possível utilizar a aprendizagem automática para diferentes propósitos, como, por exemplo, a predição do consumo energético num intervalo futuro, que é um problema de regressão, a deteção de anomalias, que é um problema de classificação, entre outros.

Para a predição de dados temporais existem diferentes abordagens e modelos que se podem seguir, desde os mais simples, como os modelos estatísticos, até a modelos mais complexos, como os de deep learning.

[7] Welcome To The FIWARE IoT Agent For Ultralight 2.0: https://fiware-iotagent-ul.readthedocs.io/en/latest/

2.5.1 Modelos Estatísticos

Os modelos estatísticos baseiam-se em princípios estatísticos que tentam procurar características nos dados temporais, como a sazonalidade, tendências, padrões, entre outros, permitindo assim a predição de dados futuros com base no histórico de dados.

Quando o conjunto de dados temporal é muito dinâmico, e não linear, os modelos estatísticos enfrentam alguns desafios para identificar algum padrão nos dados, sendo que existem melhores opções para lidar com estas situações, como é o caso de deep learning.

Alguns exemplos de modelos de predição estatísticos, são os AR, ARMA, ARIMA. Todos estes modelos referidos, suportam ainda a adição de duas componentes, a sazonalidade, e a de dados exógenos, que consistem em adicionar mais informação ao modelo para lidar melhor com a sazonalidade, e os dados exógenos que permitem a adição de informação relacionada com a série temporal principal.

2.5.2 Modelos SVR

Os modelos SVR têm como principal objetivo a classificação e regressão de dados, mas é possível adaptá-los para lidarem com dados temporais. Para isso é necessário transformar a série temporal num conjunto de atributos, para que o modelo possa identificar padrões e relações numa série temporal.

Tal como os modelos estatísticos, estes também têm dificuldade em encontrar padrões nos dados, quando eles são demasiado dinâmicos e não lineares.

2.6 Conclusão

Neste Capítulo, foi apresentado o estado da arte relacionado com a integração de dados IoT num Smart Campus, onde foram abordados diferentes tópicos relevantes no desenvolvimento de um sistema de integração de dados IoT.

Primeiro, na Secção 2.1, foi apresentado as plataformas de desenvolvimento IoT existentes que visam ajudar no processo de desenvolvimento do sistema.

Depois, para que os dados recolhidos pelos sensores possam ser guardados persistentemente, é necessário escolher um sistema de base de dados adequado, no qual foram analisados diferentes sistemas de bases de dados, na Secção 2.2.

Uma outra tarefa necessária, é o processamento de dados, onde foram analisadas diferentes plataformas de processamento de dados na Secção 2.3.

Por fim, como é necessário a comunicação desde os sensores até ao fluxo dos dados, foram analisados diferentes protocolos, e meios de comunicação, na Secção 2.4.

3

Arquitetura

Neste Capítulo é apresentada a arquitetura do sistema de integração de dados IoT, abordando como os dados entram no sistema, como são processados e armazenados, bem como a integração de um modelo de aprendizagem automática no fluxo de dados.

Neste Capítulo é apresentada a arquitetura do sistema de processamento de dados IoT, desde a sua entrada no fluxo de dados, até à forma como são processados e armazenados. A escolha da arquitetura do sistema, é importante para que os componentes e tecnologias a serem utilizados respondam aos requisitos.

Nas secções seguintes são apresentados os diferentes componentes utilizados na arquitetura do sistema, e abordagens utilizadas. Na Secção 3.1 são apresentadas duas propostas de abordagens de desenvolvimento, de seguida na Secção 3.2 é apresentada a fase inicial de entrada dos dados no sistema, depois na Secção 3.3 é apresentada a forma como os dados são processados após entrarem no sistema, a seguir é apresentada na Secção 3.4 a estrutura como os dados são armazenados, bem como as tecnologias e abordagens utilizadas, depois na Secção 3.6 é apresentado o módulo de predição dos dados, a seguir na Secção 3.7 é apresentada a plataforma escolhida para a visualização dos dados, bem como os dados são obtidos, e por último na Secção 3.8 é apresentada a avaliação do desempenho da plataforma Fiware no fluxo de dados.

3.1 Abordagens de Desenvolvimento

Nesta Secção foram estudadas e analisadas diferentes abordagens de desenvolvimento para um sistema de integração de dados IoT, suas vantagens e desvantagens. Foram analisadas em maior detalhe, duas abordagens, uma que não utiliza nenhuma plataforma de desenvolvimento IoT, e outra com o uso de uma em particular, o Fiware.

3.1.1 Abordagem sem a utilização de plataforma IoT

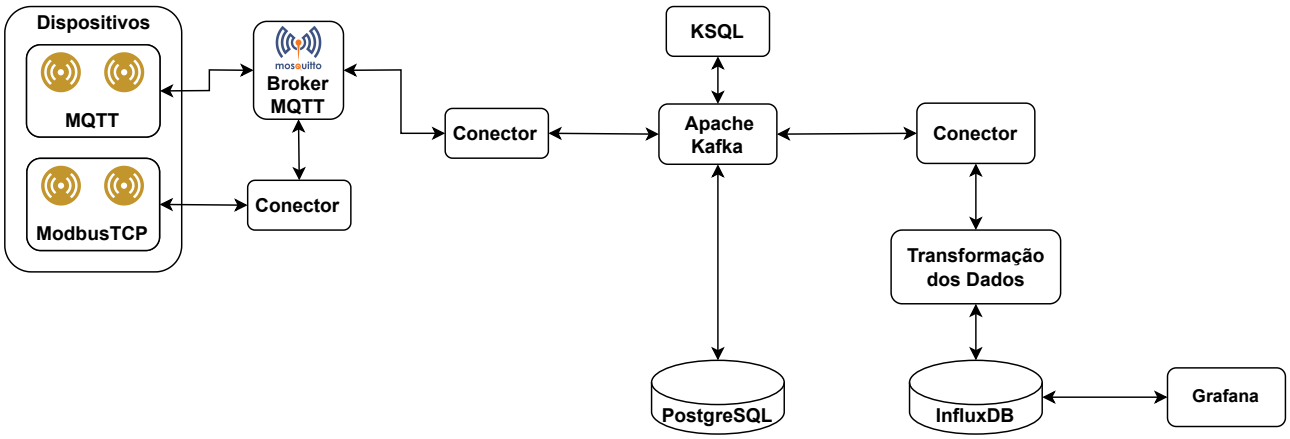


Figura 3.1: Abordagem sem a utilização de plataforma IoT

Nesta abordagem de desenvolvimento representada no diagrama da Fig. 3.1, não é utilizada qualquer plataforma de IoT para a integração dos dispositivos e fluxo dos dados. Inicialmente nesta abordagem é utilizado um broker MQTT para receber os dados dos dispositivos, e no caso de dispositivos que não comunicam por MQTT, é criado um conector para o protocolo de comunicação utilizado, mas a entrada dos dados é sempre feita por MQTT.

De seguida, os dados são enviados para o Apache Kafka, onde pode ser feito algum tratamento dos dados, execução de um modelo de aprendizagem automática em tempo real, entre outros. Neste fluxo, os dados relacionados com os dispositivos são guardados numa base de dados PostgreSQL, e por fim os dados temporais são guardados numa base de dados InfluxDB.

As vantagens desta primeira abordagem, é que cada componente, ao ser criado manualmente, dá-nos a liberdade de implementar o que for necessário, sem a dependência de nenhuma plataforma. Em contrapartida, tendo em conta que os componentes são criados manualmente, o tempo de implementação de alguma funcionalidade pode ser mais demorado, sendo que podem existir falhas de implementação que necessitam de ser testadas.

3.1.2 Abordagem utilizando a plataforma Fiware

Nesta segunda abordagem de desenvolvimento representada no diagrama da Fig. 3.2, é utilizado a plataforma Fiware, que disponibiliza alguns componentes de integração dos dispositivos, fluxo dos dados, e persistência dos mesmos, como foi apresentado na Secção 2.1.1

Na parte inicial, como depende dos dispositivos a serem utilizados, a abordagem é igual à primeira, com o uso de um broker MQTT, que recebe os dados dos dispositivos, e encaminha-os para a plataforma Fiware.

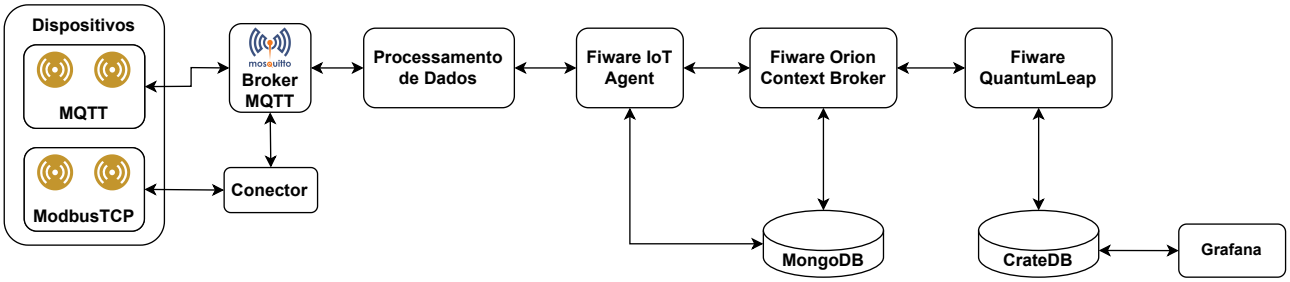


Figura 3.2: Abordagem utilizando a plataforma Fiware

As vantagens de utilizar o Fiware, ou outra plataforma de desenvolvimento de sistemas IoT, é que fornecem um conjunto de componentes fáceis de utilizar, e devidamente testados por uma comunidade de utilizadores, e que por norma, já têm em consideração a segurança, escalabilidade e outros aspetos. Por outro lado, caso seja necessário implementar uma funcionalidade muito específica, pode ser mais complicado o seu desenvolvimento, para corresponder aos requisitos impostos por essa plataforma.

Por forma a ajudar no desenvolvimento deste sistema, esta foi a abordagem escolhida, utilizando a plataforma Fiware como plataforma de desenvolvimento, com os seguintes componentes:

- **Fiware IoT Agent** responsável por receber os dados vindos do broker MQTT;
- **Fiware Orion Context Broker** responsável por gerir os dados presentes no fluxo, bem como as entidades (sensores) previamente registadas;
- **Fiware QuantumLeap** responsável por registar os dados permanentemente numa base de dados CrateDB;

O Fiware foi escolhido, devido à sua ampla variedade de serviços para diferentes propósitos, à sua modularidade, e a possibilidade de integrar componentes externos.

Todos os componentes utilizados na arquitetura são executados pelo Docker, e configurados pelo docker-compose, que é uma ferramenta do Docker para definir, configurar e executar múltiplos serviços, através de um ficheiro de configurações.

3.2 Entrada dos dados

Na área de IoT existem diversos dispositivos com diferentes protocolos de comunicação, sendo necessário escolher uma abordagem que permita a entrada de dados por vários protocolos, por forma a permitir a configuração de qualquer tipo de sensor. Neste trabalho são utilizados dois protocolos de comunicação diferentes, o MQTT e o ModbusTCP. Os sensores que utilizam o protocolo de comunicação MQTT estão instalados no pólo da Mitra da Universidade de Évora, com o objetivo de medirem o consumo energético em vários pontos do pólo. Quanto ao sensor que utiliza o protocolo de comunicação ModbusTCP, está instalado no pavilhão gimnodesportivo da Universidade de Évora, com o objetivo de medir a energia produzida pelos painéis fotovoltaicos instalados.

Por forma a uniformizar os dados enviados por diferentes protocolos de comunicação, foi definido como protocolo principal o MQTT, por ser simples e bastante utilizado na área do IoT [SSH19]. Assim, todos os dispositivos que utilizam outro protocolo de comunicação diferente, é necessária uma tradução para MQTT, como, por exemplo, os sensores que utilizam o ModbusTCP.

Esta uniformização, permite e facilita a entrada dos dados no sistema, porque em componentes posteriores a informação do protocolo de comunicação é abstraída, e caso exista a necessidade de instalar um novo sensor com outro protocolo de comunicação, é apenas necessário implementar um módulo de tradução no início do fluxo de dados, que recebe os dados no protocolo de comunicação do sensor, e os envia para o broker MQTT, e não ficar limitado aos protocolos de comunicação suportados pela plataforma de desenvolvimento utilizada.

3.2.1 Medição do consumo energético

Os sensores utilizados para a medição do consumo energético, foram os "Shelly 3EM" que utilizam o protocolo de comunicação MQTT, e disponibilizam, entre outros, no intervalo de 10 segundos os seguintes dados:

- "current"
- "energy"
- "pf"
- "power"
- "returned_energy"
- "total"
- "total_returned"
- "voltage"

Estes dados são publicados no broker MQTT, pois aqui não há necessidade de fazer qualquer tradução de protocolo de comunicação. Como os sensores utilizados são trifásicos, para cada fase são enviados estes dados, e a identificação da fase é feita a partir do nome do tópico utilizado pelo sensor, onde passa por um processamento do mesmo no fluxo de dados, explicado na Secção 3.3. Na Fig. 3.3 é apresentado a estrutura do tópico utilizado pelos sensores Shelly.

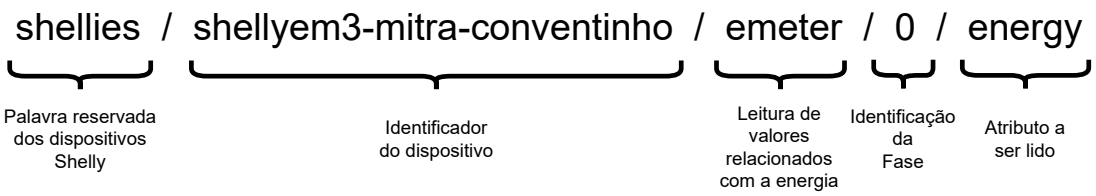


Figura 3.3: Estrutura do tópico utilizado pelos sensores Shelly

3.2.2 Medição da produção de energia

Para a obtenção dos dados de produção de energia pelos painéis fotovoltaicos, foi utilizado o inversor Fronius Symo 10.0-3-M, que utiliza como protocolo de comunicação o ModbusTCP, e disponibiliza, entre outros, os seguintes dados:

- "power"
- "energy_day"
- "energy_year"
- "energy_total"

Ao contrário dos sensores que medem o consumo energético, estes não publicam os dados num broker MQTT, sendo que a abordagem de leitura de dados é ligeiramente diferente.

Os diferentes dados que o inversor produz, são guardados em vários registos no dispositivo, e para a obtenção desses dados, é necessário invocá-lo para a leitura dos registos pretendidos, sendo esta uma comunicação síncrona.

Esta leitura dos registos é feita indicando o valor numérico do registo que desejamos ler, o seu tipo de dados, e a quantidade de bits que é necessário ler, especificada na documentação do dispositivo, para cada informação que o mesmo gera.

Como o dispositivo de medição de produção de energia utiliza o protocolo de comunicação ModbusTCP, é necessário implementar um módulo de tradução de ModbusTCP para MQTT, para seguir a abordagem mencionada acima. Para isso, foi utilizado o módulo Modbus4MQTT [1] de código aberto, em que o objetivo é traduzir os registos lidos por ModbusTCP para MQTT, através de algumas configurações. A configuração deste componente consiste em especificar, para cada registo que desejamos ler, o seu identificador, o seu tipo de dados, o número de bits a serem lidos, qual o nome do tópico em que serão publicados, e a frequência de leitura dos dados.

3.2.3 Fiware IoT Agent

O Fiware IoT Agent, é um componente da plataforma de desenvolvimento Fiware, responsável por receber os dados dos sensores a ele ligados, disponibilizando diferentes formas de obtenção dos mesmos. Neste caso foi escolhido o Fiware IoT Agent JSON com o protocolo de comunicação MQTT, onde o objetivo é subscrever um tópico MQTT, que no seu conteúdo contém um objeto JSON com o valor lido pelo sensor, e os enviar para o próximo componente, o Fiware Orion Context Broker, apresentado na Secção 3.4.2, através de uma API própria do Fiware, pelo protocolo de comunicação Next Generation Service Interface (NGSI) ou NGSI-v2.

Este componente é configurado a partir de pedidos HTTP ao serviço, sendo necessário a criação de um grupo de sensores, e depois a criação dos sensores (identificados como entidades na plataforma Fiware).

Como neste caso temos dois sensores com protocolos de comunicação diferentes, foram criados dois grupos, um para os sensores que comunicam por MQTT, e outro para os sensores que comunicam por ModbusTCP.

A forma como este componente recebe os dados do broker MQTT, tem algumas regras no nome do tópico, ou seja, ele subscreve os dados de um sensor a partir do seguinte tópico "id_grupo_de_dispositivos/id_dispositivo/attrs", onde o "id_grupo_de_dispositivos" corresponde ao identificador dado ao grupo de dispositivos criado, o "id_dispositivo" corresponde ao identificador de cada sensor registado, e o "attrs" que é uma palavra reservada deste componente, que corresponde aos atributos vindos do sensor, e ao fim do tópico.

[1] https://pypi.org/project/modbus4mqtt/

Esta estrutura no nome do tópico não corresponde ao que cada sensor de medição do consumo energético utiliza para comunicar os dados, e visto que os dispositivos apenas permitem a alteração do seu identificador e não permitem a alteração do tópico para onde enviam os dados, é necessário realizar um pré-processamento dos mesmos, que será apresentado na Secção 3.3

3.2.4 Segurança

Os dispositivos utilizados para a medição do consumo energético e para a medição da produção de energia, não suportam a comunicação com Secure Sockets Layer (SSL), apenas suportam a autenticação por nome de utilizador e senha, pelo que foi necessário seguir uma abordagem diferente, para que os dados e a interação com os dispositivos não fosse comprometida.

Como os dispositivos não suportam SSL, foi necessário criar um broker MQTT configurado sem SSL, para os dispositivos conseguirem comunicar os seus dados. A interação dos dispositivos até este broker MQTT, é feita sem SSL, pelo que é importante isolar esta comunicação entre os dispositivos e o broker MQTT, numa VLAN, por forma a não ficar exposta para o exterior, e apenas permitir a comunicação dos dispositivos.

Por forma a assegurar a comunicação segura entre os dispositivos e a plataforma de desenvolvimento Fiware, em que alguns componentes ficam expostos para o exterior, foi utilizado uma abordagem de utilizar outro broker MQTT com configuração SSL, sendo este o que envia os dados para Fiware IoT Agent descrito na Secção 3.2.3. Assim o fluxo de dados, é o seguinte: os dados dos dispositivos são enviados para o broker MQTT sem SSL configurado, depois são processados pelo Node-RED (descrito na Secção 3.3) e por fim, são enviados para o outro broker MQTT configurado com SSL, onde o Fiware IoT Agent irá receber os dados. Na Fig. 3.4 é apresentado um diagrama que representa o fluxo de entrada dos dados com a utilização dos dois brokers MQTT, desde os dispositivos, até à entrada no Fiware IoT Agent.

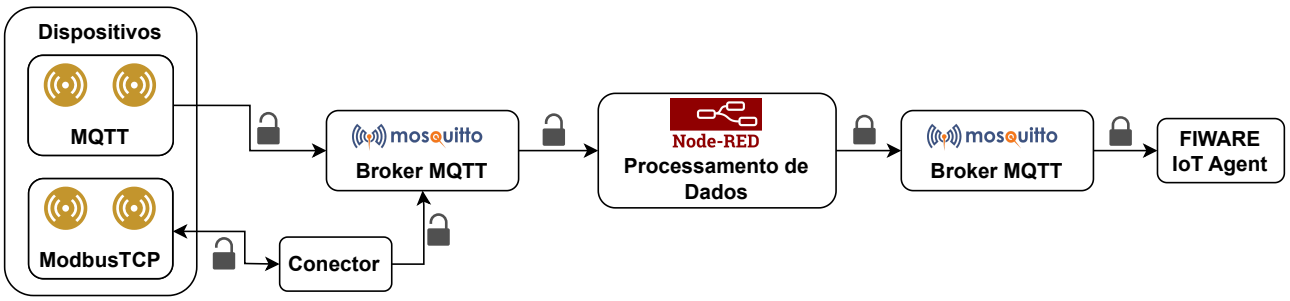


Figura 3.4: Fluxo de entrada dos dados

A utilização de dois brokers MQTT em conjunto com o processamento pelo Node-RED, previne o envio de dados indesejados para o Fiware, ou seja, no broker configurado sem SSL que é utilizado para a comunicação com os dispositivos, se for enviada uma mensagem num tópico que não seja os utilizados pelos sensores, a mesma não é encaminhada para o segundo broker, e consequentemente não entra nos componentes do Fiware, apenas os sensores configurados no Node-RED é que encaminham a informação para o Fiware, sendo que outros tópicos são descartados.

3.2.5 Broker MQTT

O broker MQTT escolhido foi o *mosquitto*, por ser um broker considerado leve, e em simultâneo, com uma boa capacidade de processamento. O *mosquitto* é também considerado eficiente devido a sua leveza, o

que é um aspeto importante na decisão de um broker MQTT. Quanto à segurança, o *mosquitto* suporta a comunicação por SSL, e autenticação através de nome de utilizador e senha [BKPC21].

3.3 Processamento dos dados

A plataforma Fiware, mais especificamente, o componente Fiware IoT Agent, requer que o nome dos tópicos MQTT sigam algumas regras como mencionado na Secção anterior, pelo que é necessário realizar o processamento dos dados na sua entrada para o Fiware. Para isso foi escolhido a plataforma Node-RED [2] para este pré-processamento, por ser uma plataforma fácil de utilizar, com uma abordagem de programação de fluxos em blocos, através de uma página web, e com possibilidade para a execução de código JavaScript.

3.3.1 Node-RED

A forma como são obtidos os dados no Node-RED para o seu processamento, é através de um nó inicial de ligação ao broker MQTT sem SSL, e subscrição de um determinado tópico MQTT ou via "wildcards". Neste caso foram utilizados "wildcards" para obter os dados de todos os dispositivos e todos os seus atributos, sem a necessidade de especificar cada um individualmente.

Quando são recebidos dados a partir do nó de subscrição de tópicos MQTT, eles são processados por código JavaScript, que transforma os dados para a estrutura requerida pelo Fiware IoT Agent, ou seja, é alterado o tópico em que sensor publica os dados, para seguir a regra de "id_grupo_de_sensores/id_sensor/attrs", e o conteúdo da mensagem é um objeto JSON, com a informação proveniente do dispositivo, com a identificação da fase do valor lido, e qual foi o atributo lido. Na Fig. 3.5 é apresentado um diagrama deste processamento de dados, ilustrando a entrada de dados no tópico `/shellies/shelly3em/emeter/0/power` com o valor 1000, e a sua saída no formato JSON desejado.

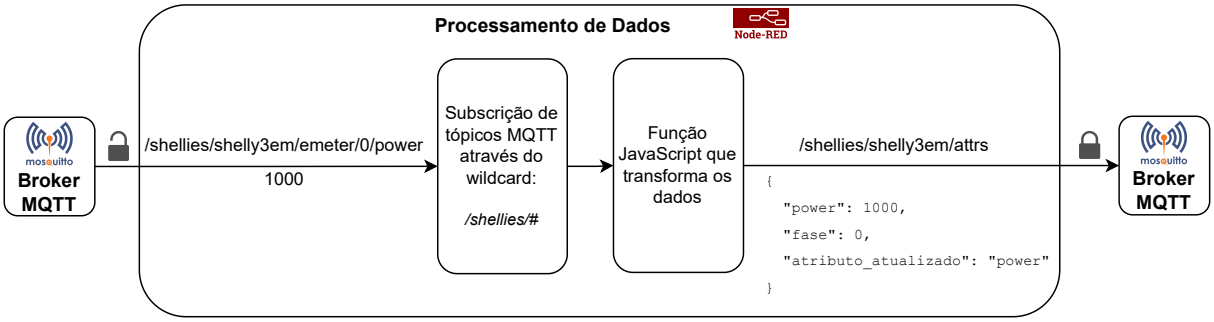


Figura 3.5: Processamento de dados pelo Node-RED

A identificação da fase e de qual atributo foi lido, está presente no nome do tópico, nos últimos dois valores, como, por exemplo: `shellies/sheely3em/emeter/0/power`

Onde o 0 corresponde à fase, e o "power" corresponde ao atributo lido.

Na Listagem 3.1 é apresentado o conteúdo da mensagem completo após estas transformações, tal como mostra a Fig. 3.5.

[2] https://nodered.org/

Listagem 3.1: Conteúdo da mensagem JSON

```
{
    "power": 1000,
    "fase": 0,
    "atributo_atualizado": 'power'
}
```

E o nome do tópico é: `shellies/shelly3em/attrs`

Após esta transformação dos dados e do nome do tópico MQTT, os dados são enviados para o broker MQTT com configuração SSL (como descrito na Secção 3.2.4) através de um segundo nó configurado no Node-RED, que irão ser recebidos posteriormente pelo Fiware IoT Agent no formato correto.

O Node-RED é também utilizado no processo de realização da técnica de *downsampling* dos dados, que consiste no processo de reduzir a granularidade dos dados, resultando numa forma comprimida deles. Um exemplo desta técnica é transformar os dados originais, que dependem da frequência de geração de dados do sensor, em dados de minuto a minuto, horários, diários, entre outros. A função do Node-RED nesta técnica é apenas a execução de uma consulta SQL a cada 1 hora, em que essa consulta agrupa os dados do consumo energético (atributo "total") da última hora, e os insere numa nova tabela da base de dados já agrupados. Este agrupamento é feito através da subtração do último valor do consumo energético com o primeiro valor, resultando no consumo energético total durante 1 hora.

Na Fig. 3.6, é apresentado um diagrama que ilustra esta técnica de *downsampling*.

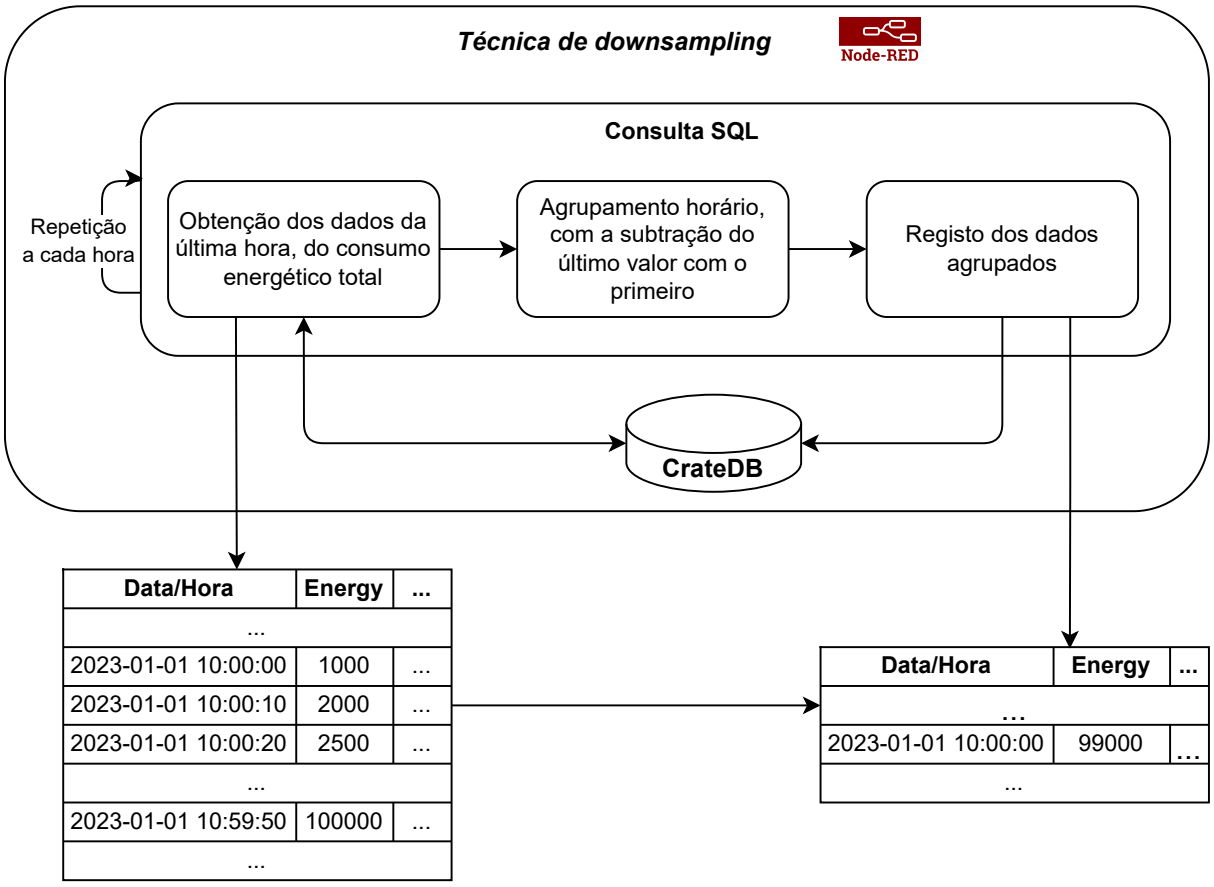


Figura 3.6: Técnica de *downsampling*

Esta técnica é relevante, por permitir que a execução de consultas de dados agrupados seja mais eficiente, ou seja, caso seja necessário obter os dados numa frequência horária, é feita a consulta à tabela de base de dados responsável por armazenar o resultado desta técnica, em vez de percorrer todos os dados numa granularidade pequena, que iria demorar mais tempo e consumir mais recursos.

Por exemplo, para obter os dados agregados de forma horária das últimas 6 horas de um sensor com frequência de 10 segundos, sem a utilização desta técnica, a consulta SQL teria de percorrer 2160 registos ($\frac{3600}{10} * 6$) para calcular os dados agregados. Com a utilização desta técnica de *downsampling*, quando feita de forma horária, a consulta SQL apenas percorre 6 registos, e não existe a necessidade de realizar o cálculo no momento da consulta SQL, pois ele já foi efetuado anteriormente.

3.4 Armazenamento dos dados

O armazenamento dos dados é um componente importante no processamento de dados IoT, pois com o histórico dos atributos dos sensores a serem registados ao longo do tempo, permite realizar análises aos dados, deteção de anomalias, modelos de predição, entre outros.

O armazenamento dos dados é divido em duas partes, uma relacionada com os dados de contexto, ou seja, dados relacionados com os dispositivos utilizados e com a plataforma Fiware, e outra parte relacionada com os dados temporais, ou seja, com o histórico de todos os valores medidos pelos dispositivos.

Para o armazenamento dos dados de contexto e da informação relacionada com o Fiware, é utilizado o sistema de base de dados MongoDB, que é um requisito da plataforma Fiware, e que é partilhado pelos componentes Fiware IoT Agent JSON e Fiware Orion Context Broker. Para os dados temporais foi escolhido o CrateDB devido à sua capacidade e facilidade de escalabilidade horizontal, e é utilizado pelo componente Fiware QuantumLeap.

3.4.1 Fiware IoT Agent

Como referido na Secção 3.2.3, este componente recebe pedidos HTTP para registar a informação dos dispositivos, entre outros, e utiliza como sistema de base de dados o MongoDB, partilhada pelo componente Fiware Orion Context Broker descrito a seguir na Secção 3.4.2.

3.4.2 Fiware Orion Context Broker

O Fiware Orion Context Broker é o componente do Fiware responsável pela gestão de toda a informação de contexto e interações com o fluxo de dados, ou seja, informação sobre os dispositivos registados, o mecanismo de atualizações, registo de subscrições, pesquisas, entre outros.

Ele utiliza o modelo de dados NGSI-v2, para a representação dos dados, em que utiliza o sistema de base de dados MongoDB para os armazenar.

Pode ser considerado o componente principal do fluxo de dados, pois é ele que recebe os dados, armazena-os, e os processa em tempo real.

Na Fig. 3.7 é apresentado um diagrama que representa as ligações deste componente, com o componente Fiware IoT Agent e base de dados MongoDB.

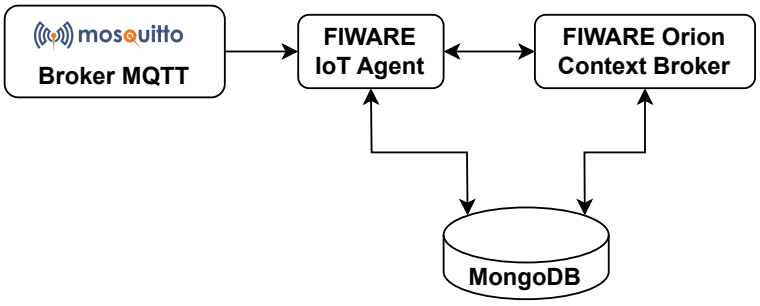


Figura 3.7: Diagrama da interação do Fiware Orion Context Broker

Atualizações

O Fiware Orion Context Broker, disponibiliza um mecanismo de atualizações às entidades, que consiste em atualizar os valores dos seus atributos, assim, quando um sensor envia novos dados, e quando chega a este componente, é feita uma atualização do atributo enviado para o valor correspondente, e a cada atualização o valor do atributo é alterado para o novo valor, ficando os atributos de uma entidade sempre com o último valor enviado pelo dispositivo. A cada atualização está sempre associado uma data e hora do registo.

Subscrição

Outro mecanismo que o Fiware Orion Context Broker disponibiliza, é o de subscrições, que permite a cada atualização de um atributo de uma entidade, a geração de uma notificação para um serviço HTTP definido na criação da subscrição, através de um pedido HTTP ao componente.

3.4.3 Fiware QuantumLeap

O componente Fiware QuantumLeap é responsável por subscrever os dados vindos do Fiware Orion Context Broker através do mecanismo de subscrição descrito na Secção 3.4.2, com o objetivo de armazenar os dados permanentemente numa base de dados temporal. Este componente disponibiliza duas alternativas para o sistema de base de dados temporal, TimescaleDB ou CrateDB, pelo que foi escolhido o CrateDB devido à sua capacidade e facilidade de escalabilidade horizontal, sendo um dos seus pontos fortes.

Os dados são organizados com base no grupo de sensores criado no Fiware Orion Context Broker, ou seja, para cada grupo existe uma tabela de base de dados independente. Quanto aos dados que são registados, estes podem ser configurados na criação da subscrição ao Fiware Orion Context Broker, podendo especificar os atributos que desejamos das entidades para serem armazenados permanentemente, juntamente com os metadados da data e hora do registo.

Cada atributo corresponde a uma coluna da tabela da base de dados, pelo que é necessário distinguir qual coluna foi atualizada em cada registo, sendo que o atributo com o nome "atributo_atualizado" anteriormente mencionado na Secção 3.3.1, é utilizado para esta identificação.

Devido ao modelo de subscrição do Fiware Orion Context Broker, é possível alterar o sistema de base de dados onde é armazenado o histórico, através da criação de um script que subscreva as alterações e as registe no sistema de base de dados pretendido, como, por exemplo, no InfluxDB.

Na Fig. 3.8 é apresentado a continuação do diagrama mencionado acima na Fig. 3.7, com a introdução

do componente Fiware QuantumLeap e respetivas ligações, e na Fig. 3.9 é apresentado um diagrama de sequência desde os dispositivos até ao seu armazenamento persistente na base de dados CrateDB.

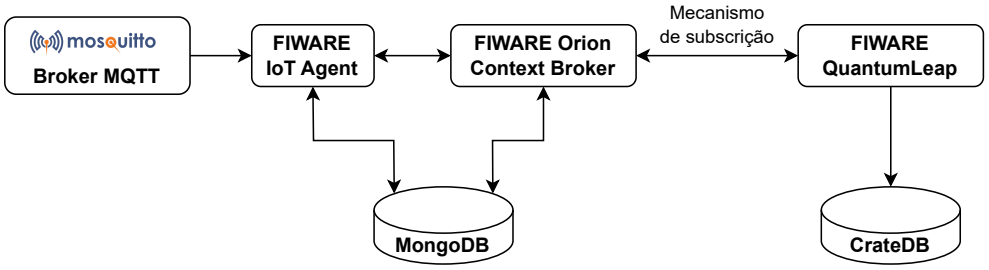


Figura 3.8: Diagrama de fluxo desde os dispositivos até ao seu armazenamento

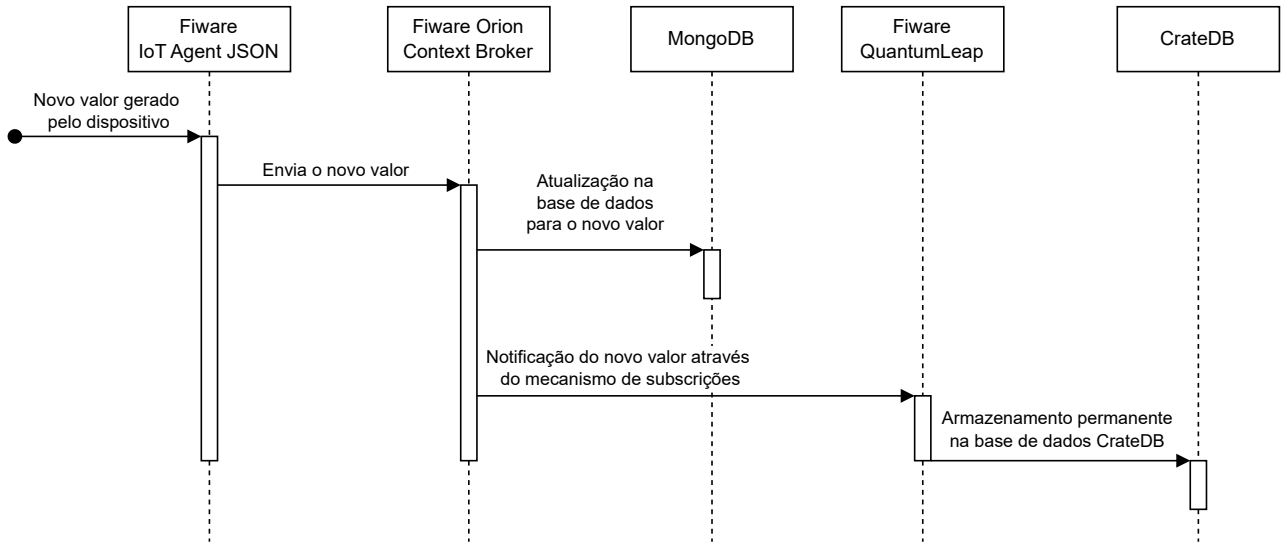


Figura 3.9: Diagrama de sequência para armazenar permanentemente um novo valor gerado por um dispositivo

3.5 Segurança

A segurança do sistema é outro aspeto importante a ter em consideração, por forma a prevenir acessos indesejados, bem como a adulteração dos dados. O Fiware disponibiliza componentes que permitem tornar o sistema mais seguro, sendo eles, o Keyrock e o Wilma PEP Proxy, que serão apresentados a seguir nas secções 3.5.1 e 3.5.2 respetivamente.

Para além destes componentes auxiliares de segurança, foram também configurados com SSL os seguintes componentes:

- Broker MQTT
- Fiware IoT Agent JSON
- Fiware Orion Context Broker
- Fiware Wilma PEP Proxy

- Fiware Keyrock

Quanto ao componente Fiware QuantumLeap, não suporta comunicação por SSL na comunicação com o Fiware Orion Context Broker, nem com o sistema de base de dados CrateDB.

Quanto ao sistema de base de dados MongoDB, o mesmo suporta a configuração com SSL, mas não foi possível configurá-lo com SSL, pois foi encontrado um problema na comunicação entre o componente Fiware Iot Agent JSON e o MongoDB que impossibilita a sua comunicação. Este problema foi reportado no repositório Github do componente [3], estando neste momento em fase de resolução [4].

3.5.1 Keyrock

O Keyrock é um componente do Fiware, que é responsável pela gestão de utilizadores e respetivos acessos ao sistema. Através deste componente, é possível criar diferentes utilizadores, para possibilitar o acesso aos recursos do Fiware.

Para cada utilizador aceder aos recursos do Fiware, como, por exemplo, à API do Fiware Orion Context Broker, é necessário a sua autenticação neste componente, que retorna um *token* de autenticação, para a sua utilização no componente Wilma PEP Proxy descrito de seguida na Secção 3.5.2.

Este componente permite ainda a geração de *tokens* permanentes, que são úteis para os serviços do Fiware que necessitam de comunicar entre eles, ou seja, para o Fiware IoT Agent, é gerado um *token* permanente, para que ele consiga enviar os dados para o Fiware Orion Context Broker.

3.5.2 Wilma PEP Proxy

O Wilma PEP Proxy é um componente do Fiware, onde o seu objetivo, é proteger recursos, mediante um proxy que fica situado entre o cliente e o recurso, prevenindo assim a comunicação direta do cliente ao recurso.

Este componente, quando é feito um pedido por parte de um cliente, valida se o *token* de autenticação é válido, junto do componente Keyrock descrito anteriormente na Secção 3.5.1, que verifica se aquele *token* tem permissões para aceder ao recurso desejado.

Este componente é utilizado para proteger os serviços Fiware IoT Agent e o Fiware Orion Context Broker, para que qualquer interação necessária, seja efetuada mediante autenticação.

Na Fig. 3.10 é apresentado a integração destes componentes de segurança, Fiware Wilma PEP Proxy e Fiware Keyrock, com os componentes já existentes.

[3] https://github.com/telefonicaid/iotagent-json/issues/739
[4] https://github.com/telefonicaid/iotagent-node-lib/pull/1511/commits/7277fda31212dba7563c347de21244918ee3c074

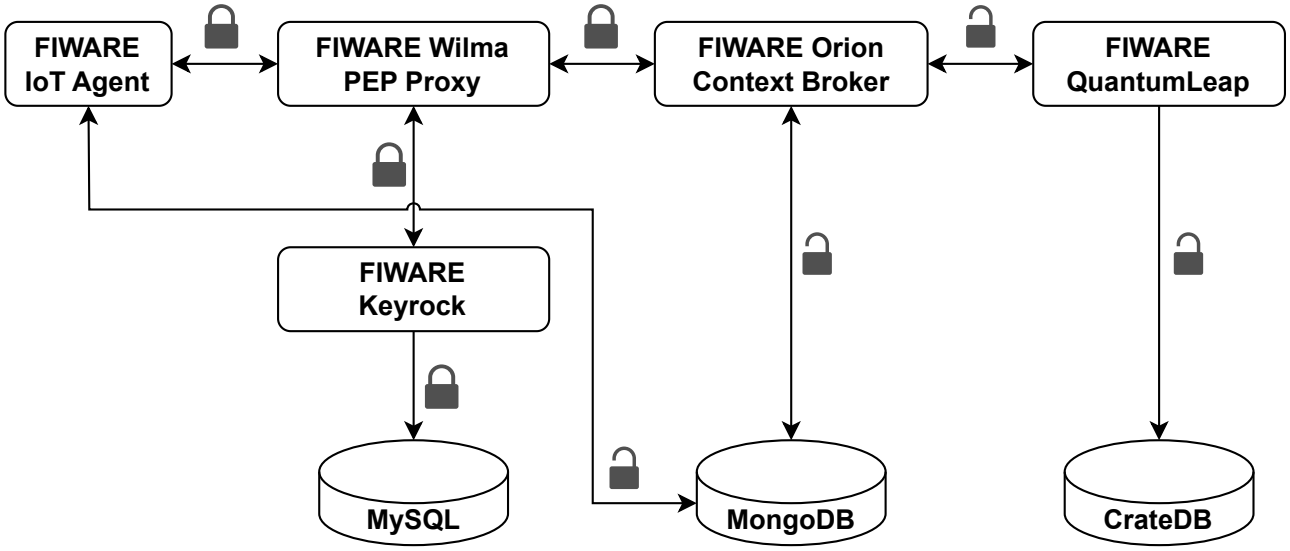


Figura 3.10: Diagrama com os componentes de segurança

3.6 Predição dos dados

A predição dos dados provenientes dos sensores, é outro módulo deste sistema, em que o seu objetivo é integrar um modelo de aprendizagem automática no fluxo de dados. Para esta integração, foi utilizado o Apache Spark, pois o objetivo é o processamento de pequenos conjuntos de dados, e porque a plataforma Fiware utilizada, disponibiliza um conector para o mesmo. O objetivo desta integração é o Apache Spark receber os dados dos sensores, que os processa a cada hora, para realizar a predição das próximas 24h, como descrito no Capítulo 4.

3.6.1 Integração com o Fiware

A integração do Apache Spark com o Fiware, foi realizada através do conector que o Fiware disponibiliza, que permite ao Apache Spark receber os dados do Fiware Orion Context Broker quando eles são recebidos no fluxo. Para esta comunicação, é utilizado o mecanismo de subscrições do Fiware Context Orion Broker, onde a cada alteração do valor de um atributo do sensor, é enviada uma notificação a um serviço HTTP, que neste caso é o Apache Spark. O Apache Spark foi configurado com um nó principal identificado como "master", e outro nó identificado como "worker", onde o "master" é responsável por receber os dados provenientes do Fiware Orion Context Broker, e os distribui para o "worker", responsável pelo processamento dos dados. Podem existir vários nós "worker", conforme as necessidades, sendo o "master" a gerir a carga de trabalho entre os vários "workers" a ele conectados.

O Fiware disponibiliza duas formas de interação com o Apache Spark, uma em que é utilizado a linguagem de programação Scala para realizar o processamento dos dados, e outra em que é utilizado Python, sendo que para ambas existe o conector para a integração com o Fiware. Foi escolhido o Python como linguagem de programação para este processamento de dados, devido às bibliotecas utilizadas para a construção do modelo de aprendizagem automática, serem apropriadas para Python.

3.6.2 Processamento dos dados no Apache Spark

O processamento dos dados no Apache Spark, é feito a cada hora, por forma a corresponder às abordagens utilizadas na construção do modelo de aprendizagem automática, em que a predição é feita a cada hora. Assim, a cada hora, é feito o cálculo do consumo energético da última hora, ou seja, é subtraído o último valor registado do atributo "total" do sensor pelo seu primeiro valor, onde este atributo representa o consumo energético total registado pelo dispositivo. Após este cálculo referente ao consumo real do dispositivo, é adicionado o seu resultado ao modelo construído.

Depois desta atualização dos dados do modelo com os últimos dados observados, é possível realizar a predição das próximas 24h, sendo que com esta abordagem as previsões correspondentes às primeiras horas da predição, têm uma confiança melhor, do que o valor previsto passado 24h.

Após a obtenção dos dados previstos para as próximas 24h, é necessário guardá-los, pelo que, o conector que o Fiware disponibiliza, permite que estes dados sejam enviados para o Fiware Orion Context Broker, para seguirem o fluxo normal de persistência de dados.

Na Fig. 3.11 é apresentado um diagrama de sequência com a integração do Apache Spark e a plataforma Fiware.

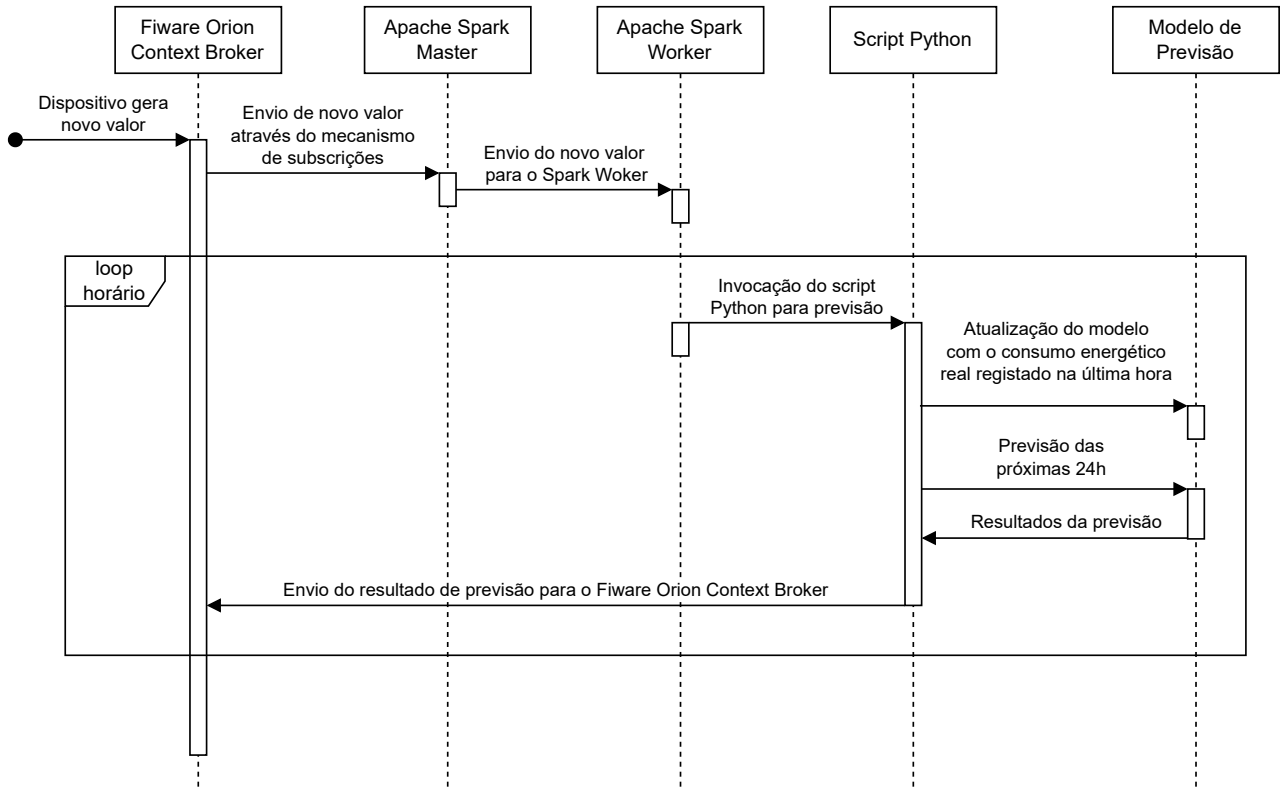


Figura 3.11: Diagrama de sequência da integração do Apache Spark com o Fiware

3.7 Visualização dos dados

A visualização dos dados é também um componente importante, pois através da visualização organizada dos dados, é possível extrair informação de forma rápida e intuitivamente, para que com base na análise dos dados, possam ser tomadas decisões.

3.7.1 Grafana

Como plataforma de visualização dos dados foi utilizado o Grafana [5], pois disponibiliza diferentes formas de visualização importantes para a análise dos dados, e disponibiliza também os *plugins* necessários para a obtenção dos dados ao CrateDB, e ao Fiware Orion Context Broker através de pedidos REST à API do componente.

Foram criados diferentes painéis para a visualização dos dados por cada sensor instalado, que corresponde a um local específico do pólo da Mitra da Universidade de Évora. Os painéis referentes aos sensores de medição do consumo de energia, contém os seguintes dados:

- Potência (por fase) (últimas 24h)
- Potência (total) (últimas 24h)
- Energia horária (por fase) (últimas 24h)
- Energia horária (total) (últimas 24h)
- Total de Energia
- Potência Atual (por fase)
- Potência Atual (total)
- Voltagem (por fase) (últimas 24h)
- Energia diária (total) (últimos 7 dias)

Na Fig. 3.12 é apresentado um *printscreen* do painel de visualização do consumo energético.

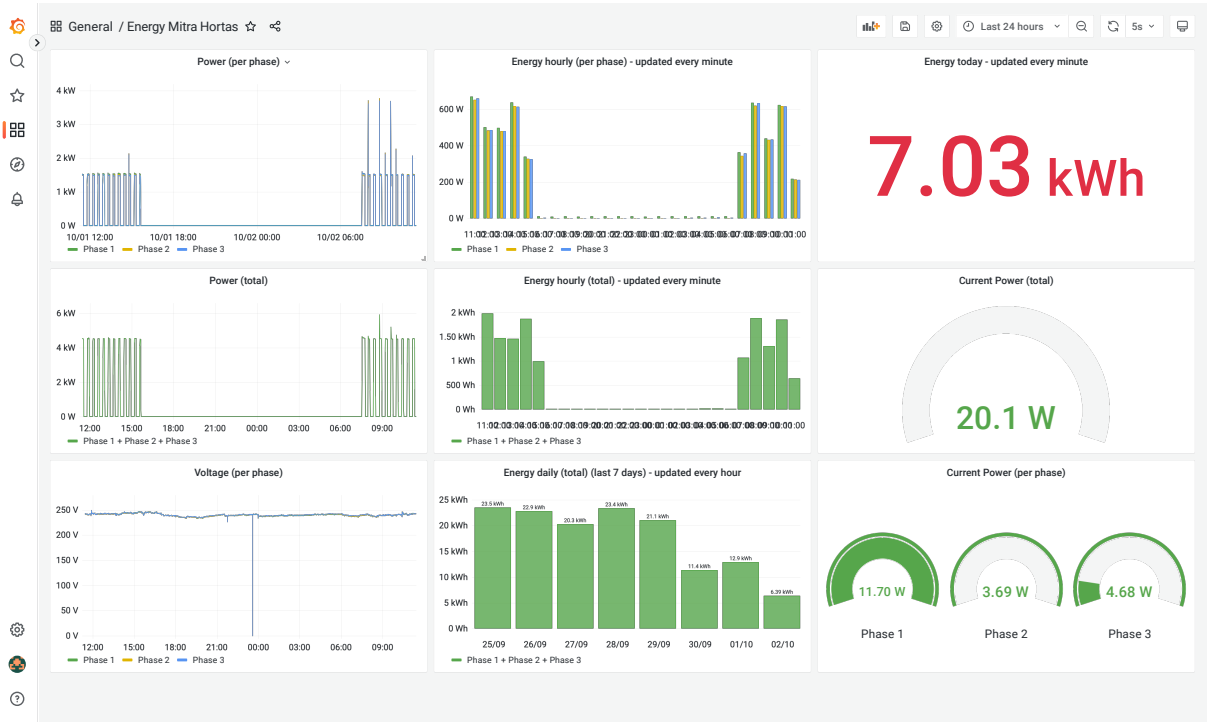


Figura 3.12: Painel de visualização dos dados do consumo energético no Grafana

[5] https://grafana.com

Quanto ao painel de visualização referentes ao dispositivo de medição da produção de energia, contém os seguintes dados:

- Produção de Energia (tempo real)
- Energia Produzida (por hora)
- Energia Produzida (por dia)

3.8 Desempenho da plataforma Fiware

Nesta Secção, foi avaliado o desempenho da plataforma Fiware utilizando a ferramenta Apache JMeter para a realização dos testes, pois fornece diferentes componentes que ajudam a configurar ao detalhe os testes a serem realizados e que permite simular cargas de trabalho intensas.

A realização destes testes, é de elevada importância, pois com eles é possível identificar problemas do sistema quando é colocado sobre carga, identificação de pontos de falha no sistema e identificação de componentes que contribuam para um pior desempenho do sistema.

3.8.1 Configuração

A configuração utilizada para a execução destes testes de desempenho, foi em termos de *hardware* a seguinte:

- **Computador:** Lenovo ThinkBook 14
- **CPU:** Intel i5-1135G7
- **Memória RAM:** 16GB DDR4
- **Disco:** 512GB NVME

E em termos de *software* a seguinte:

- **Sistema Operativo:** Ubuntu 22.04
- **Docker:** 20.10.21
- **Fiware Orion Context Broker:** 3.7.0
- **Fiware QuantumLeap:** 0.7.5
- **MongoDB:** 6.0.4
- **CrateDB:** 5.2.3

Na configuração destes testes de desempenho, foram seguidas as recomendações por parte do Apache JMeter [6], que foram, principalmente, o uso de uma segunda máquina para realizar os testes, a execução do *script* através da linha de comandos e utilização da última versão do Apache JMeter.

Quanto à máquina responsável por realizar os testes de desempenho, em termos de *hardware* foi a seguinte:

[6] https://jmeter.apache.org/usermanual/best-practices.html

- **Computador:** MacBook Pro
- **CPU:** Apple M1
- **Memória RAM:** 16GB
- **Disco:** 512GB NVME

E em termos de *software*, a seguinte:

- **Sistema Operativo:** MacOS Ventura 13.5.1
- **Apache JMeter:** 5.6.2
- **Versão do Java:** 17

Ainda referente à configuração, foram criadas 1000 entidades no Fiware Orion Context Broker, através da API do mesmo, com um atributo para registar o consumo energético. Estas entidades visam simular sensores reais, armazenando dados semelhantes.

Quanto à configuração dos componentes do Fiware, foram utilizadas as configurações recomendadas pelo Fiware por forma a obter um melhor desempenho[7], onde as principais foram:

- Versão do MongoDB superior a 4.4;
- Criação de "Indexs" no sistema de base de dados MongoDB;
- Utilização do modo de notificação *threadpool*;
- Nível dos "Logs" em ERROR ou WARN;
- Execução dos pedidos de forma concorrente;

3.8.2 Plano de Teste

No Apache JMeter foi criado um plano de teste, que consiste em enviar pedidos consecutivos ao componente Fiware IoT Agent, componente responsável por receber os dados dos sensores, durante 1 minuto, através da REST API que o mesmo disponibiliza, com o objetivo de alterar o estado de cada entidade, ou seja, o valor do consumo energético para o qual a entidade tinha sido configurada.

No Apache JMeter foi criado um grupo de dispositivos chamado de *Thread Group*, em que o seu objetivo é simular diferentes sensores a enviarem novos dados em simultâneo, para gerar uma carga elevada na plataforma, e assim avaliar como o sistema responde nestas situações.

Por fim, foram recolhidos alguns dados do comportamento do sistema perante o nível de carga imposto pelos testes, onde se obtém, principalmente, o número de pedidos por segundo, e o tempo de resposta.

O número de sensores a gerar dados em simultâneo, foi configurado como uma variável no Apache JMeter, sendo que os testes foram executados para diferentes números de sensores, que foram eles: 1, 5, 10, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900 e 1000.

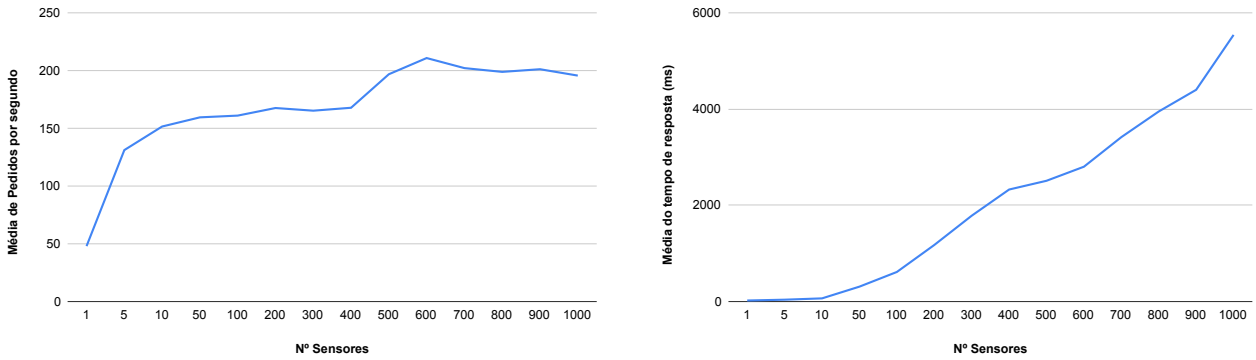
[7] https://fiware-orion.readthedocs.io/en/3.8.1/admin/perf_tuning.html

3.8.3 Resultados

Na Fig. 3.13a é apresentado a média de pedidos por segundo com base no número de sensores, onde é possível verificar que a partir de 50 sensores a enviarem dados em simultâneo, o número de pedidos com que o sistema consegue lidar começa a baixar lentamente.

Quanto à Fig. 3.13b, é apresentado a média do tempo de resposta com base no número de sensores, onde é possível verificar um aumento significativo a partir dos 50 sensores em simultâneo.

Na Tabela 3.1 é possível analisar os resultados obtidos do Apache JMeter, pelo número de sensores.



(a) Média de pedidos por segundo por número de sensores (b) Média do tempo de resposta por número de sensores

Figura 3.13: Gráficos dos resultados dos testes de desempenho

Nº de Sensores	Tempo Médio de Resposta (milissegundos)	Nº de Pedidos por segundo	Nº Total de Pedidos	Rácio Tempo Médio/-Número de Sensores
1	20,51	47,96	2869,00	20,51
5	37,54	131,16	7951,00	7,50
10	64,92	151,57	9218,00	6,49
50	308,97	159,51	9719,00	6,17
100	613,97	161,02	9817,00	6,13
200	1175,02	167,59	10295,00	5,87
300	1783,15	165,28	10175,00	5,94
400	2327,51	167,82	10598,00	5,81
500	2509,68	196,85	12253,00	5,01
600	2802,15	210,87	13248,00	4,67
700	3417,00	202,16	12721,00	4,88
800	3950,15	198,89	12719,00	4,93
900	4400,97	201,19	12890,00	4,89
1000	5542,00	195,68	12450,00	5,54

Tabela 3.1: Tabela de resultados dos testes de desempenho

Na Figura 3.14, é possível verificar o rácio entre o tempo médio de resposta pelo número de sensores, onde é possível concluir, que o desempenho é proporcional ao número de sensores.

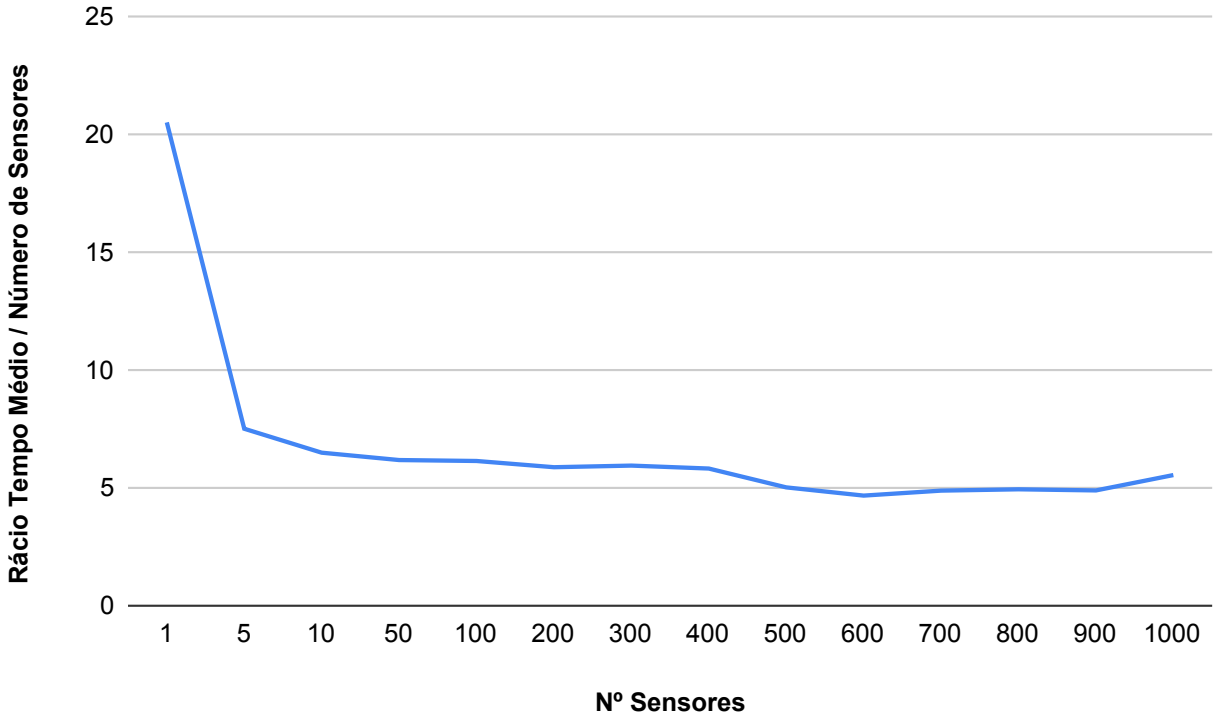


Figura 3.14: Rácio do tempo médio pelo número de Sensores

3.8.4 Avaliação dos resultados

Com base nos resultados acima mencionados, é possível observar que, com a configuração utilizada, a partir de 100 sensores o sistema começa a apresentar tempos de resposta um pouco mais elevados, sinal de que o sistema começou a ficar sobrecarregado.

Nos testes de desempenho foi possível observar que o componente Fiware QuantumLeap, responsável por armazenar os dados persistentemente na base de dados CrateDB, não registava todos os dados enviados, sendo este um problema conhecido pela equipa de desenvolvimento do componente. [8].

Por forma a mitigar este problema deste componente, foi alterado o limite de tempo dos pedidos entre o Fiware Orion Context Broker, e o Fiware QuantumLeap, passando do valor padrão de 5 segundos, para 10 segundos, ou seja, com esta alteração, apenas existe problema de persistência dos dados caso a duração de um pedido demore mais de 10 segundos, o qual não foi observado nos testes realizados, sendo o valor máximo de tempo de resposta que obtivemos nos testes acima descritos, de 5542 ms, mas caso esta alteração não fosse efetuada, iria existir problemas de persistência de dados no teste com 1000 sensores.

Outra solução para este problema seria utilizar uma abordagem de escalabilidade, em que seria implementado um balanceador de carga no componente Fiware QuantumLeap, em que eram criadas diferentes instâncias do componente, cada uma com uma base de dados diferente. Com esta solução é possível obter um melhor desempenho, pois a carga é distribuída por diferentes instâncias do componente, evitando a sobrecarga apenas de uma instância.

[8] https://github.com/orchestracities/ngsi-timeseries-api/issues/722

3.9 Conclusão

Neste Capítulo foi apresentado a arquitetura do sistema de IoT, onde foi explicado todo o fluxo de dados desde a entrada dos dados, até ao seu armazenamento, e também o processamento de dados de predição.

A plataforma de desenvolvimento Fiware escolhida, ajudou significativamente no desenvolvimento, ao disponibilizar diferentes componentes para diferentes propósitos, mas em simultâneo, foi desafiador trabalhar com o Fiware devido às especificidades de cada componente.

Na Fig. 3.15, é apresentado o diagrama do fluxo de dados apresentado neste Capítulo onde é possível ver todas as interações que cada componente faz.

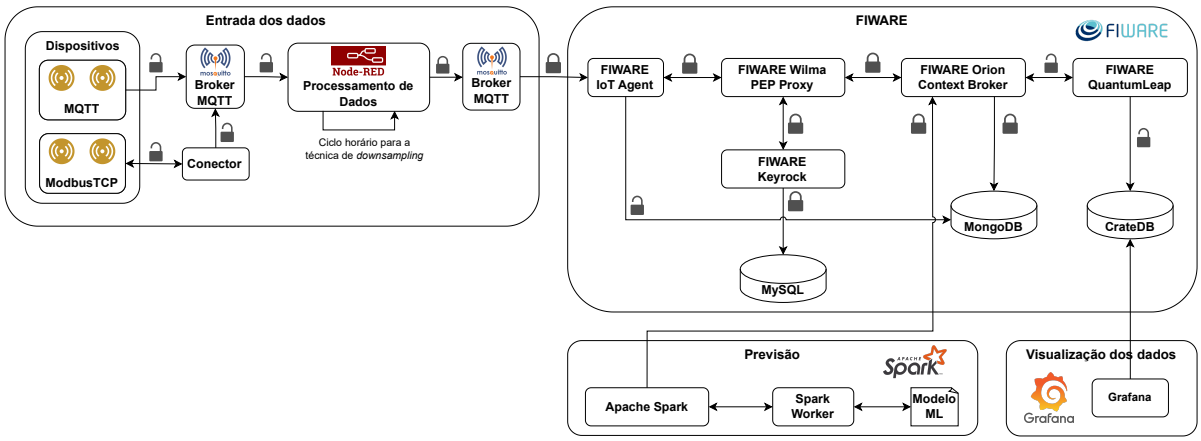


Figura 3.15: Diagrama completo do sistema

4

Predição do consumo energético

Neste Capítulo são apresentadas as diferentes abordagens utilizadas para a construção de um modelo de predição do consumo energético, onde foram utilizados os modelos de aprendizagem automática da família ARIMA e SVR.

A predição de valores futuros com base no histórico de dados, permite num Smart Campus, uma melhor eficiência energética, e também uma melhor gestão da rede elétrica. [AEG18]

Prever o consumo energético tem alguns desafios, pois existem diversos fatores que influenciam o consumo energético de um Smart Campus, como a sua infraestrutura física, equipamento instalado, condições meteorológicas, entre outros [AEG18].

Esta predição de valores futuros, pode ser feita por modelos de aprendizagem automática utilizando diversas abordagens e técnicas, sendo que neste caso, foram experimentados diferentes modelos e abordagens, como, os modelos estatísticos da família ARIMA, modelos SVR, abordagem de predição multi-passo direta e recursiva.

Os dados para a predição, foram obtidos do sensor de consumo energético do pólo da Mitra da Universidade de Évora, desde o dia 7 de outubro de 2022 até ao dia 20 de abril de 2023, com intervalo horário, que são apresentados em mais detalhe na Secção 4.1.

Outros dados importantes para a predição de valores futuros, são aqueles que podem estar relacionados ao consumo energético, sendo que foram obtidos no mesmo intervalo do que os anteriores, os dados da estação meteorológica da Universidade de Évora instalados no pólo da Mitra, próxima da instalação dos sensores, que também são apresentados em mais detalhe na Secção 4.1.

Por forma a enriquecer o conjunto de dados relativos ao consumo energético, foram também adicionados alguns dados que podem contribuir para um melhor modelo de predição, que foram:

- Dia da semana;
- Valor indicativo de dia útil;
- Hora;

Com base na quantidade de dados obtidos das diferentes fontes, cerca de oito meses, foi definido que a predição do consumo energético seria no máximo de 24h, tendo sido experimentadas diferentes abordagens de predição multi-passo, que são apresentadas na Secção 4.2.

4.1 Dados

O conjunto de dados utilizado para a construção de um modelo de predição do consumo energético, foi obtido de duas fontes, de um sensor de consumo energético instalado no pólo da Mitra da Universidade de Évora, e da estação meteorológica também instalada no pólo da Mitra. Outros dados foram adicionados, como: dia da semana, valor indicativo de dia útil, e hora. Todos estes dados têm uma frequência horária que corresponde, no caso dos dados do consumo energético, ao consumo total por hora, e no caso dos dados meteorológicos, à média de valores registados por hora.

Na Tabela 4.1, é possível ver quais foram os dados obtidos, a sua unidade, bem como a sua fonte.

Dados	**Fonte**	**Unidade**	**Intervalo**
Temperatura	Estação Meteorológica	Graus Celsius (C)	$[-1.02, 34.15]$
Humidade	Estação Meteorológica	Percentagem (%)	$[12.39, 99.2]$
Velocidade do Vento	Estação Meteorológica	Metros por segundo (m/s)	$[0.408, 9.28]$
Precipitação	Estação Meteorológica	Milímetro (mm)	$[0, 9.9]$
Radiação Solar	Estação Meteorológica	Watt por metro quadrado (W/m2)	$[0, 981]$
Dia da Semana	Calendário	Valor numérico	$\{0, 1...5, 6\}$
Dia útil	Calendário	Booleano	$\{0, 1\}$
Hora	Calendário	Valor numérico	$\{0, 1...22, 23\}$
Consumo energético	Sensor de consumo energético	Quilowatt-hora (kW/h)	$[128.1, 24590.2]$

Tabela 4.1: Detalhes do conjunto de dados

Após a obtenção destes dados, foi necessário realizar um processamento dos mesmos, para eliminar valores nulos, discrepantes, e adicionar dados em falta. Quanto aos valores nulos e discrepantes foram substituídos pela média dos valores adjacentes, por exemplo, se existisse um valor discrepante ou nulo na hora 01/01/2023 10:00:00, o mesmo era substituído pela média dos valores de 01-01-2023 09:00:00 e 01/01/2023 11:00:00. O mesmo procedimento foi feito, quando existia falta de dados no conjunto obtido.

No gráfico apresentado na Fig. 4.1, é possível ver a relação que cada atributo tem com os dados do consumo energético, sendo que, dos dados obtidos da estação meteorológica, o mais relevante é a temperatura, e dos restantes dados adicionados, o mais relevante é a hora. Com base nesta análise é possível verificar que a relação entre os dados da precipitação, radiação solar e velocidade do vento, com o consumo energético, é baixa, podendo ser irrelevantes para a construção de um modelo de predição do consumo energético, ou até serem prejudiciais, sendo que os mesmos foram descartados.

O gráfico da Fig. 4.1 foi construído através da biblioteca *pandas*, sendo que o índice apresentado foi construído automaticamente pela biblioteca. Este índice representa a correlação linear entre os atributos, sendo ele variável entre -1 e 1, em que valores próximos de -1 têm uma forte correlação negativa, valores próximos de 1 têm uma forte correlação positiva, e valores próximos de 0 significa que não existe nenhuma correlação linear entre eles [pan]

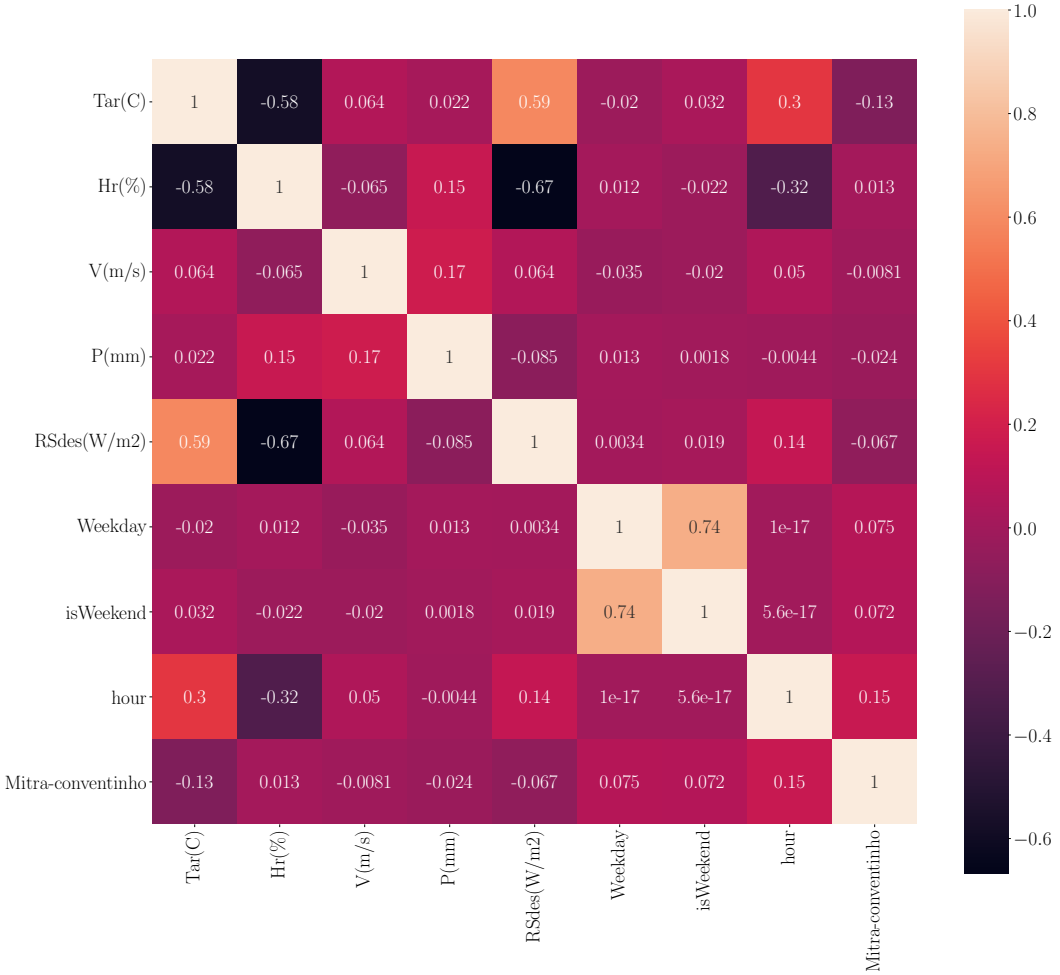


Figura 4.1: *Heatmap* dos atributos

No gráfico apresentado na Fig. 4.2 é apresentado um excerto de 7 dias do conjunto de dados do consumo energético.

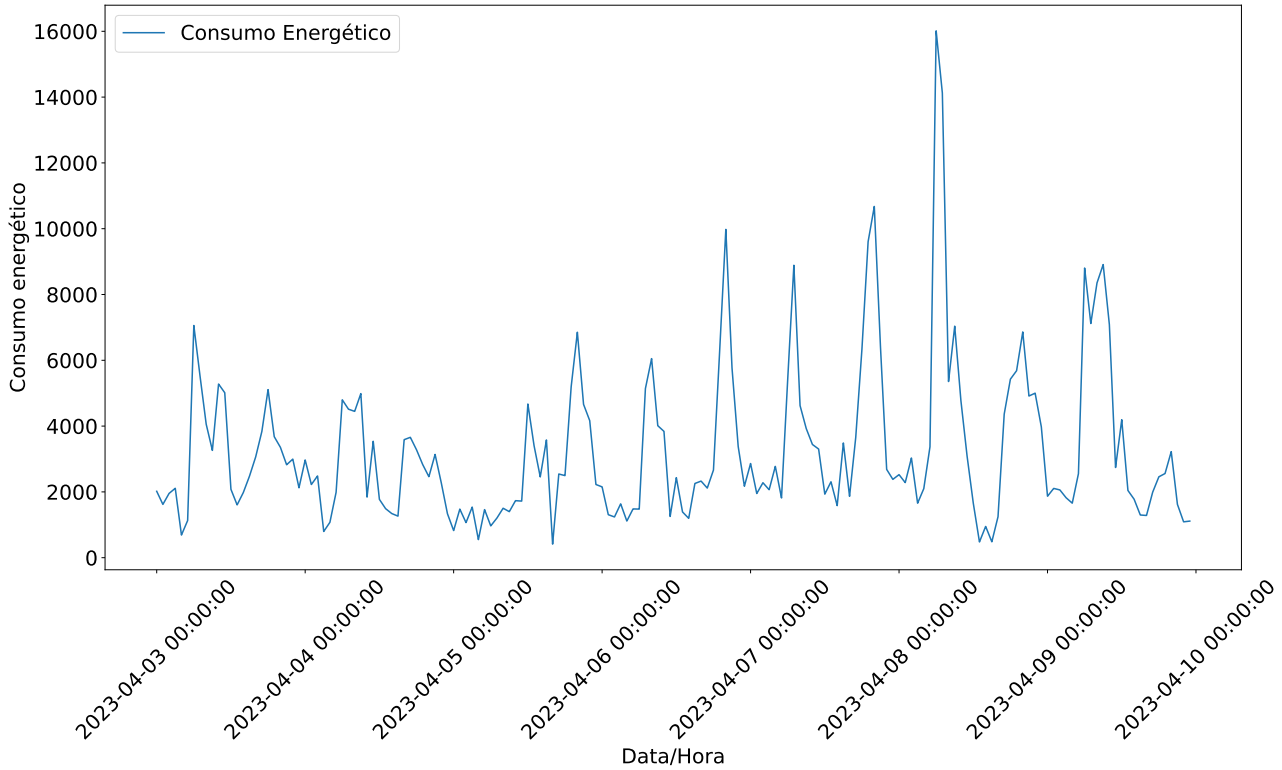


Figura 4.2: Excerto de 7 dias do conjunto de dados do consumo energético

Por forma a identificar a sazonalidade da série temporal referente ao consumo energético, foi utilizado o método de decomposição, resultando em três componentes: 1) tendência; 2) sazonalidade e 3) resíduo. Na componente sazonalidade, é possível identificar padrões recorrentes ao longo do tempo, permitindo assim a identificação da sazonalidade da série temporal de 24h. Na Fig. 4.3, é possível ver um excerto do resultado da componente sazonalidade, onde é visível que a sazonalidade é de 24h, na Fig. 4.4 é apresentada a componente tendência onde não é possível extrair nenhuma informação relevante, e por último, na Fig. 4.5 é apresentado a componente resíduo.

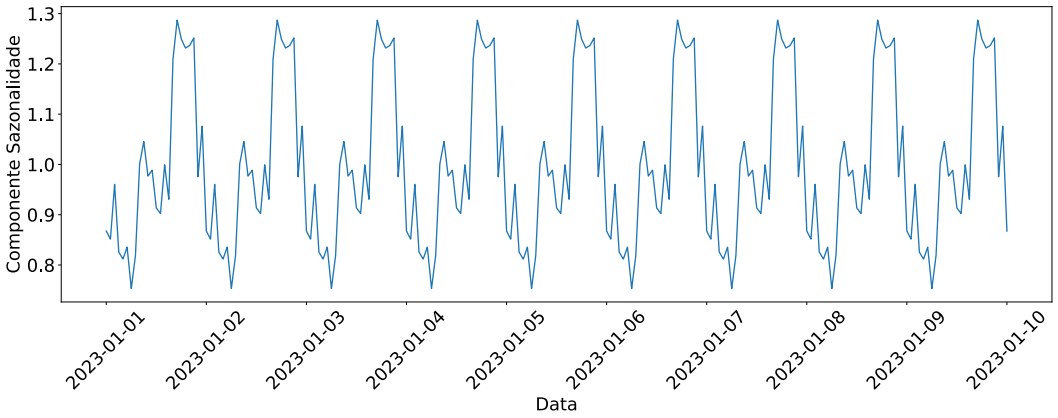


Figura 4.3: Componente Sazonalidade

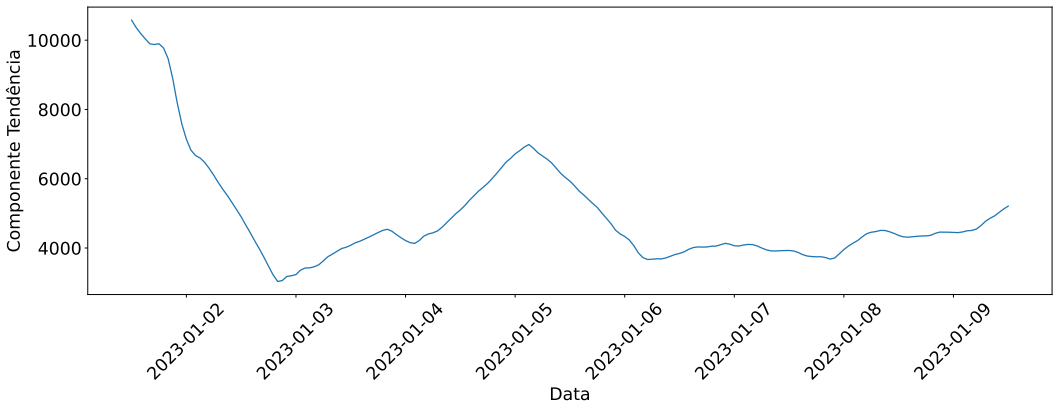


Figura 4.4: Componente Tendência

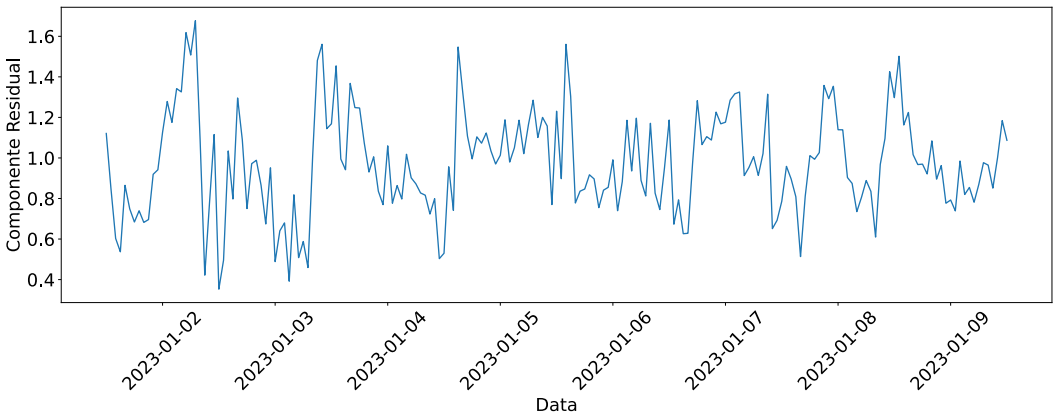


Figura 4.5: Componente Residual

4.2 Abordagens para construção de modelos de predição do consumo energético

Os métodos utilizados e experimentados para construir um modelo de predição do consumo energético, foram principalmente os modelos estatísticos da família ARIMA, e os modelos SVR, ambos seguindo a mesma abordagem de predição multi-passo recursiva, para a predição de 24h.

Para a construção dos modelos estatísticos ARIMA foi utilizada a biblioteca statsmodels [1] de código aberto, e para os modelos SVR a biblioteca skforecast [2] também de código aberto, sendo que internamente utiliza a biblioteca scikit-learn [3].

Foi também experimentada outra abordagem de predição que foi a multi-passo direta, em que consiste em criar um modelo para cada passo, ou seja, neste caso para prever as próximas 24h, iria criar 24 modelos diferentes, cada um para a predição de cada passo no futuro. Para esta abordagem, apenas foi possível experimentar com os modelos SVR utilizando a biblioteca *skforecast*, pelo que para os modelos da família ARIMA, a biblioteca não permite a predição multi-passo direta.

[1] https://www.statsmodels.org/stable/index.html
[2] https://skforecast.org/
[3] https://scikit-learn.org/stable/

4.2.1 Predição Multi-Passo Recursiva

A abordagem multi-passo recursiva consiste na acumulação dos resultados das previsões como entrada para a predição dos seguintes, ou seja, na primeira iteração da predição é previsto o valor da próxima hora $(T+1)$ com base nos dados reais disponíveis ($[T-n, T-1]$, sendo n a quantidade de dados), para a predição da segunda hora $(T+2)$ o resultado da predição de $T+1$ é utilizado para prever o valor de $T+2$, e assim consecutivamente, conforme apresentado na Fig. 4.6, onde os erros da predição vão acumulando, sendo que o primeiro valor previsto tem um erro menor, do que o último valor previsto [Bro].

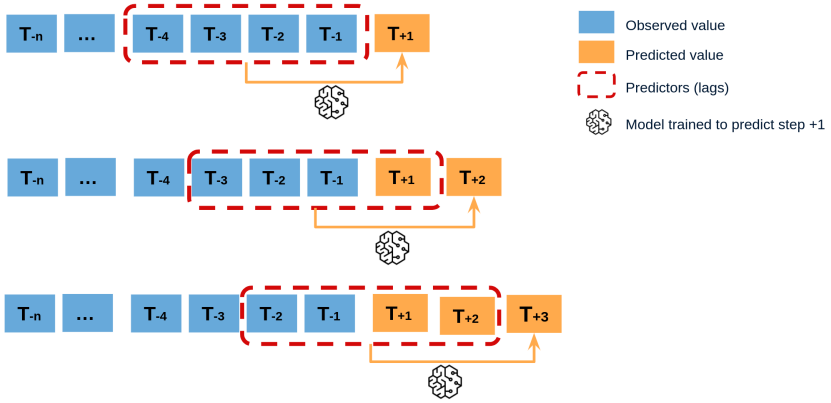


Figura 4.6: Diagrama da abordagem Multi-Passo Recursiva [4]

4.2.2 Predição Multi-Passo Direta

A abordagem multi-passo direta consiste em construir *N* modelos para prever *N* passos no futuro, ou seja, para prever 24h, é necessário construir 24 modelos, em que cada um deles é responsável por prever cada passo. Conforme apresentado na Fig. 4.7, o modelo 1 faz a predição para o valor de $T+1$, o modelo 2 faz a predição para o valor $T+2$, e assim consecutivamente, sendo que, com esta abordagem os erros da predição não acumulam, e não existe dependência dos valores anteriormente previstos [Bro].

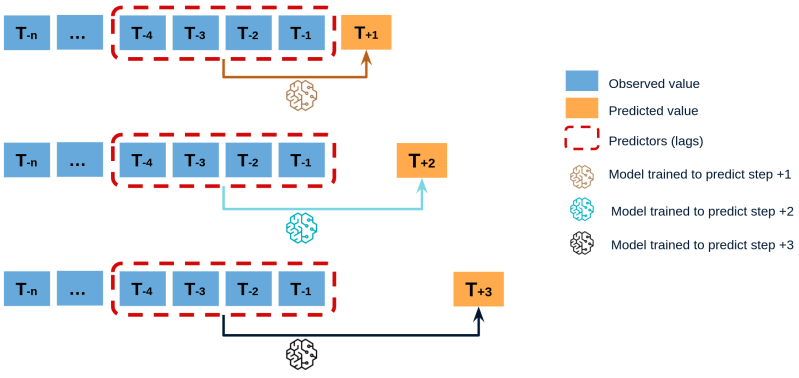


Figura 4.7: Diagrama abordagem Multi-Passo Direta [5]

[4] https://skforecast.org/0.9.1/introduction-forecasting/introduction-forecasting#recursive-multi-step-forecasting
[5] https://skforecast.org/0.9.1/introduction-forecasting/introduction-forecasting#direct-multi-step-forecasting

4.3 Modelos

Nesta Secção são apresentados os modelos testados para a predição do consumo energético, onde na Secção 4.3.1 são apresentados os modelos estatísticos e na Secção 4.3.2 o modelo SVR.

4.3.1 Modelos Estatísticos

Os modelos construídos da família ARIMA, foram obtidos a partir das diferentes componentes destes modelos, ou seja, foi inicialmente construído um modelo mais simples, os AR, depois os ARMA um pouco mais complexos, e por fim os ARIMA, sendo que para cada um destes modelos, foi também construído um modelo que continham as variáveis exógenas (X), e outro com a informação da sazonalidade (S), ou seja, foram construídos os seguintes modelos:

- AR
- Autoregressive Exogenous (ARX)
- Seasonal Autoregressive (SAR)
- Seasonal Autoregressive Exogenous (SARX)
- ARMA
- Autoregressive Moving Average Exogenous (ARMAX)
- Seasonal Autoregressive Moving Average (SARMA)
- Seasonal Autoregressive Moving Average Exogenous (SARMAX)
- ARIMA
- Autoregressive Integrated Moving Average Exogenous (ARIMAX)
- Seasonal Autoregressive Integrated Moving Average (SARIMA)
- SARIMAX

Os hiperparâmetros destes modelos são identificados pelas letras "p", "d" e "q", sendo que a letra "p" significa a ordem do modelo auto-regressivo, "d" o grau de diferenciação e o "q" a ordem do modelo de média móvel. Quando é utilizado a sazonalidade são também definidos estes hiperparâmetros mais o "S" que significa o número de passos do passado a ter em conta, ou seja, para um modelo com sazonalidade são definidos os hiperparâmetros "p", "d", "q" e os "P", "D", "Q" e "S", referentes à componente da sazonalidade.

Para a obtenção destes hiperparâmetros, foi utilizada a técnica de *Grid Search* por forma a obter os hiperparâmetros ótimos para o conjunto de dados, sendo que foi utilizada a biblioteca *pmdarima* de código aberto para a realização desta técnica. Foi necessário definir o valor mínimo e máximo de cada hiperparâmetros do modelo, ou seja, do "p", "d" e "q", e do "P", "D" e "Q" para a sazonalidade, sendo que, das várias experimentações realizadas, os valores apresentados na Tabela 4.2, foram os que apresentaram melhores resultados para os hiperparâmetros do modelo. Para além destes hiperparâmetros, foi também definido que a sazonalidade da série temporal é de 24h. Quanto à medida de desempenho utilizada nesta técnica de *Grid Search*, foi a medida padrão Mean Squared Error (MSE).

Hiperparâmetros	Mínimo	Máximo
p	2	7
d	NA	7
q	1	4
P	1	7
D	NA	2
Q	1	3

Tabela 4.2: Hiperparâmetros *Grid Search* para os modelos da família ARIMA

4.3.2 Modelo SVR

Outro modelo que foi construído, foi o SVR, que consiste em encontrar um hiperplano que minimize a diferença entre as previsões do modelo e os valores reais, ao mesmo tempo que tenta maximizar a margem entre os pontos de dados e o hiperplano, ou seja, a sua distância [Set].

Para a construção de um modelo SVR foram experimentadas diferentes combinações dos seus hiperparâmetros. Os principais hiperparâmetros para a construção de um modelo SVR são:

C: O C é considerado um hiperparâmetro de regularização do modelo, ou seja, um valor de C maior permite ao modelo ajustar-se mais aos dados com uma margem mais pequena, podendo resultar num problema de *overfitting*, enquanto um valor de C pequeno, resulta numa margem maior, podendo resultar em *underfitting* [Pat].

gamma: O *gamma* controla a influência individual dos pontos de dados na definição da função de decisão. Um valor de *gamma* alto significa uma maior influência de cada ponto dos dados, que resulta num modelo mais sensível aos detalhes dos dados, e um valor de *gamma* baixo significa uma menor influência, que resulta num modelo menos sensível aos detalhes dos dados [Pat].

kernel: Existem diferentes *kernel* possíveis, como, por exemplo, linear, *radial basis function*, polinomial, entre outros, que consistem numa função matemática, que transforma os dados originais num espaço de dimensão superior [Tea].

Para a obtenção destes hiperparâmetros, foi também utilizada a técnica de *Grid Search* com o conjunto de dados disponível, que consiste em testar várias combinações de hiperparâmetros pré-definidos, por forma a encontrar a melhor combinação que otimizem o desempenho do modelo [Sha]. A medida de desempenho utilizada para a avaliação do desempenho dos modelos, foi a medida padrão R^2 [SL] Para a execução desta técnica foi definida uma lista de valores para cada um dos hiperparâmetros, que foi a seguinte

- **C**

  0.1, 1, 10, 100, 1000, 5000, 10000, 20000, 30000, 40000, 50000, 100000

- ***gamma***

  0.0001, 0.001, 0.01, 0,1, 1

- ***kernel***

  'rbf', 'linear', 'poly'

Outro aspeto a ter em conta para a construção de um modelo SVR para a predição de uma série temporal, é a alteração do conjunto de dados, pois é necessário transformar os dados temporais num conjunto de atributos. O conjunto de atributos construído, contém as variáveis mencionadas acima, ou seja, do consumo energético, da temperatura do ar, da indicação de dia útil, do dia da semana, e da hora. Para o modelo ter informação do passado para prever o futuro, é adicionado para cada variável os seus valores das 24 horas anteriores, onde, para cada registo do conjunto de dados existem os dados de desfasamento de cada variável. Esta transformação dos dados para a construção de um modelo SVR, é feita pela biblioteca escolhida, *skforecast*, sendo apenas necessário a indicação dos hiperparâmetros do modelo, e a quantidade de dados do passado a ter em conta.

4.4 Resultados

Para testar os modelos de predição construídos, foram utilizadas duas abordagens, uma em que o conjunto de teste correspondia às últimas 24h do conjunto de dados, e outra onde o conjunto de teste correspondia a 24h intercaladas no conjunto de dados. Foram seguidas estas duas abordagens por forma a obtermos resultados mais realistas, e não apenas das últimas 24h do conjunto, em que os resultados podiam ser enviesados devido ao conjunto de teste poder coincidir num dia atípico.

As divisões do conjunto de dados original, para estas duas abordagens, foram as seguintes:

- **Abordagem no final** Corresponde à abordagem onde o conjunto de teste é o final do conjunto de dados. A divisão dos dados nesta abordagem foi a seguinte:

  Treino: De 07/10/2022 00:00:00 até 19/04/2023 23:00:00

  Teste: De 20/04/2023 00:00:00 até 20/04/2023 23:00:00

- **Abordagem variável** Corresponde à abordagem onde o conjunto de teste é variável no conjunto de dados, e no final é feita a média das medidas de desempenho utilizadas para todas as divisões. As divisões dos dados realizadas nesta abordagem, são apresentadas na Tabela 4.3

Conjunto de Treino	Conjunto de Teste	
	De	Até
01/12/2022 00:00:00	01/12/2022 01:00:00	02/12/2022 00:00:00
08/12/2022 00:00:00	08/12/2022 01:00:00	09/12/2022 00:00:00
16/12/2022 00:00:00	16/12/2022 01:00:00	17/12/2022 00:00:00
24/12/2022 00:00:00	24/12/2022 01:00:00	25/12/2022 00:00:00
01/01/2023 00:00:00	01/01/2023 01:00:00	02/01/2023 00:00:00
08/01/2023 00:00:00	08/01/2023 01:00:00	09/01/2023 00:00:00
16/01/2023 00:00:00	16/01/2023 01:00:00	17/01/2023 00:00:00
24/2023/01 00:00:00	24/01/2023 01:00:00	25/01/2023 00:00:00
01/02/2023 00:00:00	01/02/2023 01:00:00	02/02/2023 00:00:00
08/02/2023 00:00:00	08/02/2023 01:00:00	09/02/2023 00:00:00
16/02/2023 00:00:00	16/02/2023 01:00:00	17/02/2023 00:00:00
24/02/2023 00:00:00	24/02/2023 01:00:00	25/02/2023 00:00:00
01/03/2023 00:00:00	01/03/2023 01:00:00	01/03/2023 00:00:00
08/03/2023 00:00:00	08/03/2023 01:00:00	09/03/2023 00:00:00
16/03/2023 00:00:00	16/03/2023 01:00:00	17/03/2023 00:00:00
24/03/2023 00:00:00	24/03/2023 01:00:00	25/03/2023 00:00:00
01/04/2023 00:00:00	01/04/2023 01:00:00	02/04/2023 00:00:00
10/04/2023 00:00:00	10/04/2023 01:00:00	11/04/2023 00:00:00
19/04/2023 23:00:00	20/04/2023 00:00:00	20/04/2023 23:00:00

Tabela 4.3: Divisões dos dados na abordagem variável

Como forma de medição do erro dos modelos construídos, foram utilizadas três medidas de desempenho, MSE, Mean Absolute Error (MAE) e Root Mean Square Error (RMSE), onde a MSE é uma medida que penaliza de forma mais significativa erros maiores, tornando-a mais sensível a valores discrepantes. Quanto à medida de desempenho RMSE, ela está relacionada com a MSE, e fornece uma medida mais interpretável, devido à sua unidade ser igual à dos dados originais, o que facilita na compreensão do desempenho do modelo. Por fim, quanto à medida MAE, ela calcula a média dos erros absolutos, o que a torna menos sensível a valores discrepantes.

4.4.1 Modelos da família ARIMA

Na Tabela 4.4 são apresentados os resultados dos modelos da família ARIMA, com as duas abordagens de teste mencionadas acima, utilizando a biblioteca *statsmodels*.

Modelo	Hiperparâmetros	Abordagem no final			Abordagem variável		
		RMSE	MSE	MAE	RMSE	MSE	MAE
AR	p=6, d=0, q=0	1.56×10^3	2.44×10^6	1.33×10^3	3.16×10^3	1.43×10^7	2.39×10^3
ARX	p=6, d=0, q=0	1.55×10^3	2.40×10^6	1.35×10^3	2.94×10^3	1.32×10^7	2.28×10^3
SAR	p=6, d=0, q=0 P=3, D=0, Q=0, S=24	1.31×10^3	1.71×10^6	1.08×10^3	2.98×10^3	1.29×10^7	2.26×10^3
SARX	p=6, d=0, q=0 P=3, D=0, Q=0, S=24	1.61×10^3	2.59×10^6	1.43×10^3	2.84×10^3	1.20×10^7	2.14×10^3
ARMA	p=5, d=0, q=2	1.62×10^3	2.61×10^6	1.42×10^3	2.73×10^3	1.13×10^7	2.13×10^3
ARMAX	p=5, d=0, q=2	2.10×10^3	4.39×10^6	1.86×10^3	2.74×10^3	1.07×10^7	2.16×10^3
SARMA	p=5, d=0, q=2 P=2, D=0, Q=1, S=24	1.54×10^3	2.38×10^6	1.25×10^3	2.44×10^3	9.43×10^6	1.85×10^3
SARMAX	p=5, d=0, q=2 P=2, D=0, Q=1, S=24	1.96×10^3	3.86×10^6	1.65×10^3	2.45×10^3	8.96×10^6	1.88×10^3
ARIMA	p=5, d=1, q=2	1.74×10^3	3.02×10^6	1.53×10^3	2.62×10^3	1.05×10^7	2.10×10^3
ARIMAX	p=5, d=1, q=2	1.66×10^3	2.74×10^6	1.46×10^3	2.59×10^3	1.04×10^7	2.05×10^3
SARIMA	p=5, d=1, q=2 P=2, D=1, Q=1, S=24	1.21×10^3	1.47×10^6	1.03×10^3	2.54×10^3	1.01×10^7	1.91×10^3
SARIMAX	p=5, d=1, q=2 P=2, D=1, Q=1, S=24	1.13×10^3	1.28×10^6	9.86×10^2	2.61×10^3	1.06×10^7	1.99×10^3

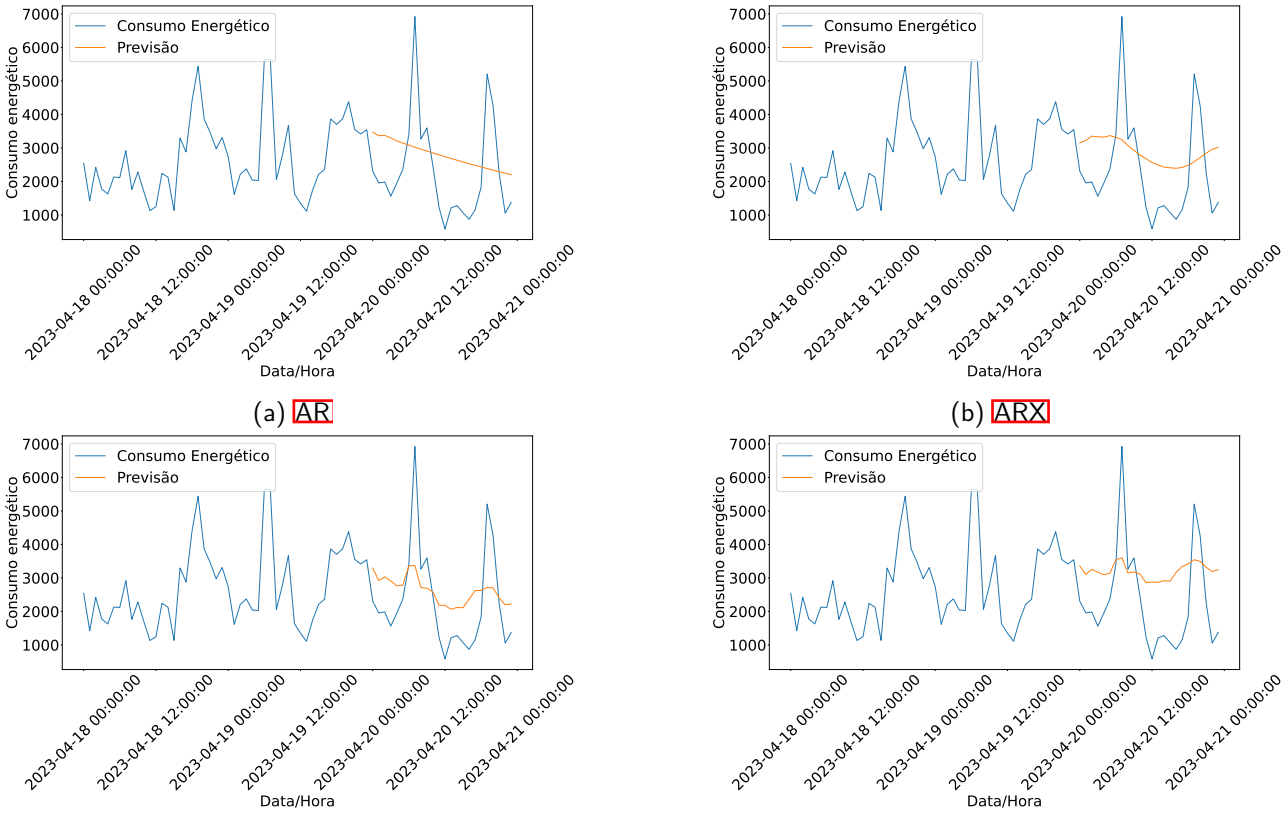
Tabela 4.4: Resultados modelos da família ARIMA

A biblioteca *skforecast* também foi experimentada para os modelos da família ARIMA com a abordagem multi-passo recursiva, por forma a comparar resultados de diferentes bibliotecas, sendo os resultados do modelo SARIMAX, apresentados na Tabela 4.5.

Modelo	Hiperparâmetros	Abordagem no final			Abordagem variável		
		RMSE	MSE	MAE	RMSE	MSE	MAE
AR	p=6, d=0, q=0	2.03×10^3	4.11×10^6	1.83×10^3	2.94×10^3	1.27×10^7	2.28×10^3
ARX	p=6, d=0, q=0	1.64×10^3	2.69×10^6	1.47×10^3	2.83×10^3	1.17×10^7	2.17×10^3
SAR	p=6, d=0, q=0 P=3, D=0, Q=0, S=24	1.85×10^3	3.42×10^6	1.64×10^3	2.70×10^3	1.09×10^7	2.07×10^3
SARX	p=6, d=0, q=0 P=3, D=0, Q=0, S=24	1.70×10^3	2.89×10^6	1.49×10^3	2.61×10^3	9.66×10^6	2.19×10^3
ARMA	p=5, d=0, q=2	2.21×10^3	4.90×10^6	2.00×10^3	2.88×10^3	1.22×10^7	2.25×10^3
ARMAX	p=5, d=0, q=2	2.25×10^3	5.06×10^6	1.96×10^3	2.63×10^3	1.04×10^7	2.09×10^3
SARMA	p=5, d=0, q=2 P=2, D=0, Q=1, S=24	1.75×10^3	3.06×10^6	1.58×10^3	2.72×10^3	1.11×10^7	2.04×10^3
SARMAX	p=5, d=0, q=2 P=2, D=0, Q=1, S=24	2.20×10^3	4.83×10^6	1.95×10^3	2.67×10^3	1.01×10^7	2.05×10^3
ARIMA	p=5, d=1, q=2	1.77×10^3	3.12×10^6	1.56×10^3	2.64×10^3	1.05×10^7	2.12×10^3
ARIMAX	p=5, d=1, q=2	1.61×10^3	2.59×10^6	1.41×10^3	2.63×10^3	1.07×10^7	2.09×10^3
SARIMA	p=5, d=1, q=2 P=2, D=1, Q=1, S=24	1.46×10^3	2.12×10^6	1.17×10^3	2.59×10^3	1.06×10^7	2.01×10^3
SARIMAX	p=5, d=1, q=2 P=2, D=1, Q=1, S=24	1.50×10^3	2.25×10^6	1.19×10^3	2.64×10^3	1.07×10^7	2.04×10^3

Tabela 4.5: Resultados SARIMAX com abordagem multi-passo recursiva utilizando a biblioteca *skforecast*

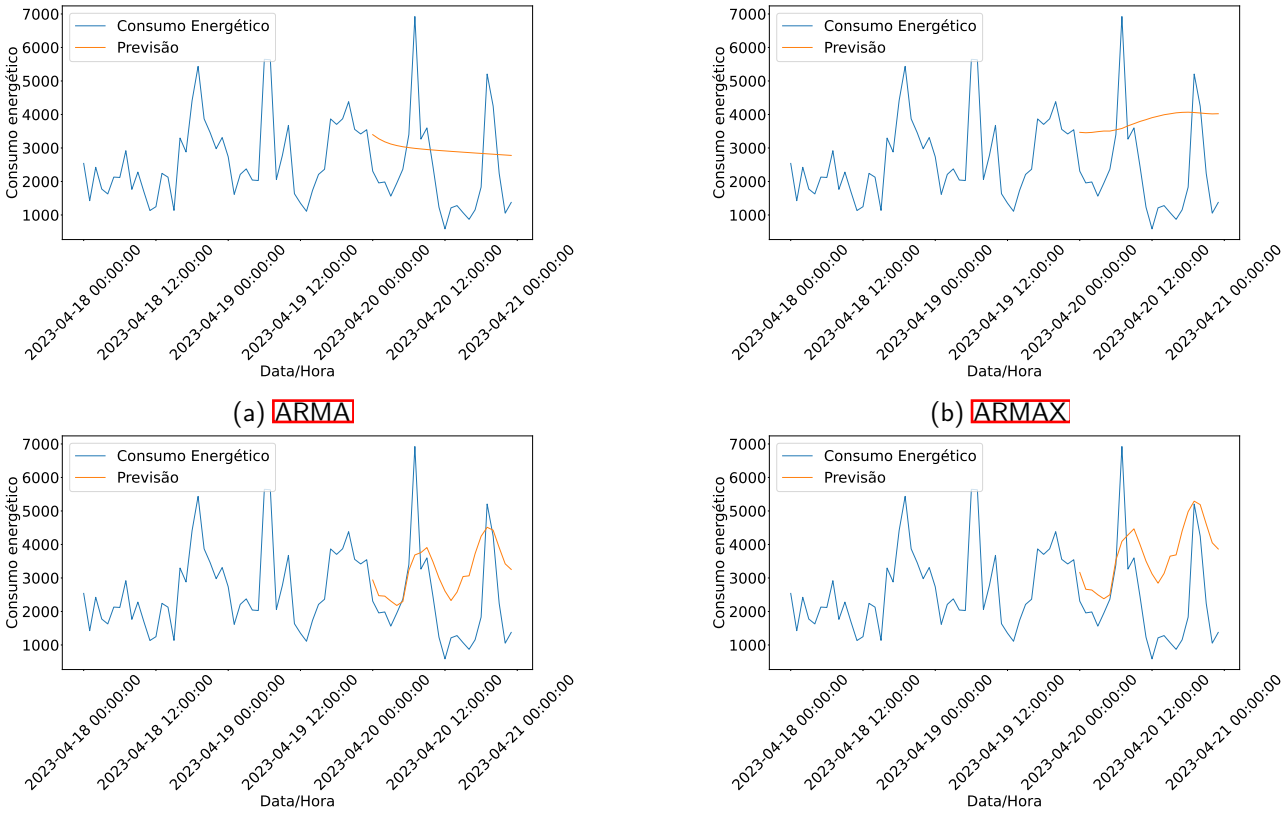
De seguida é apresentado o resultado gráfico dos modelos construídos da família ARIMA, onde na Fig. 4.8 é apresentado os resultados dos modelos AR, na Fig. 4.9 dos modelos ARMA, e na Fig. 4.10 dos modelos ARIMA. Em todos estes resultados gráficos, está presente o modelo base, com variáveis exógenas e com a componente de sazonalidade.



(a) AR (b) ARX

(c) SAR (d) SARX

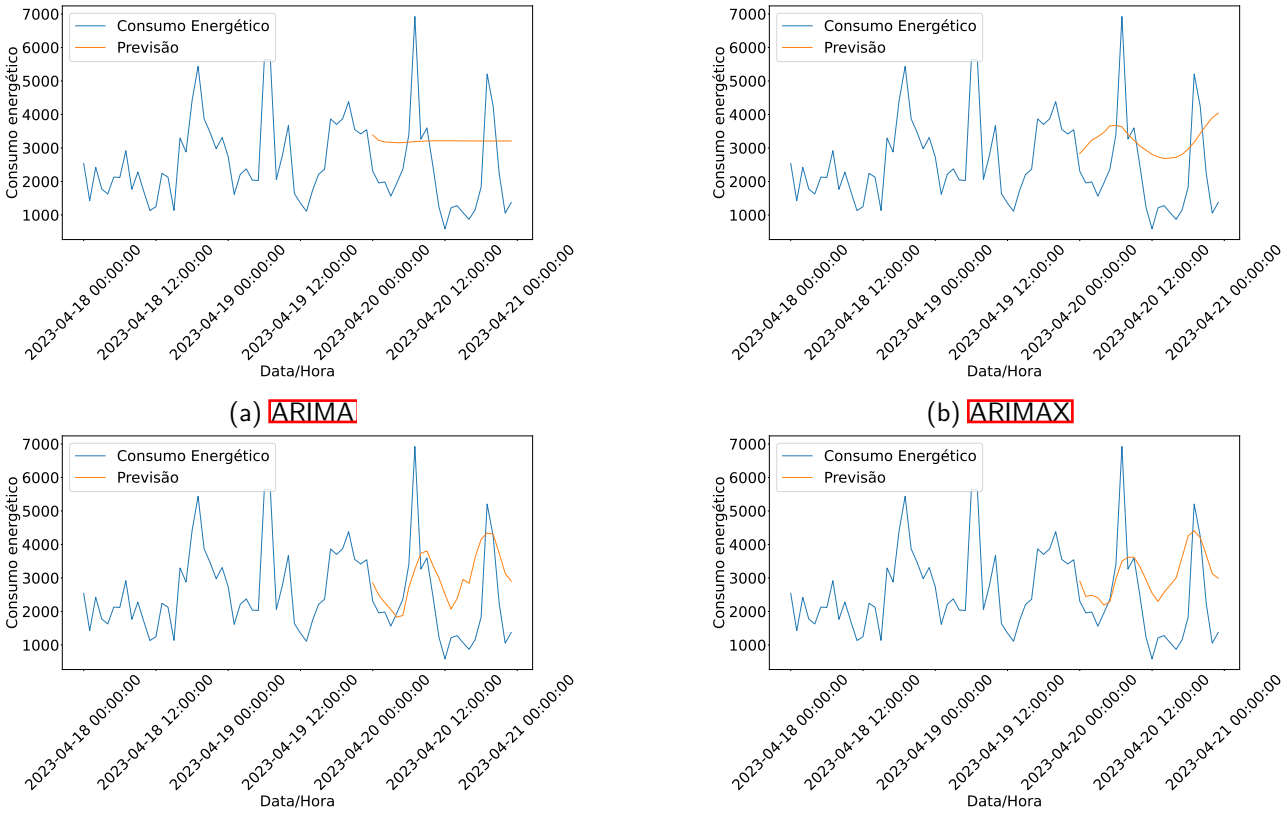
Figura 4.8: Resultado gráfico dos modelos AR construídos



(a) ARMA (b) ARMAX

(c) SARMA (d) SARMAX

Figura 4.9: Resultado gráfico dos modelos ARMA construídos



(a) ARIMA (b) ARIMAX

(c) SARIMA (d) SARIMAX

Figura 4.10: Resultado gráfico dos modelos ARIMA construídos

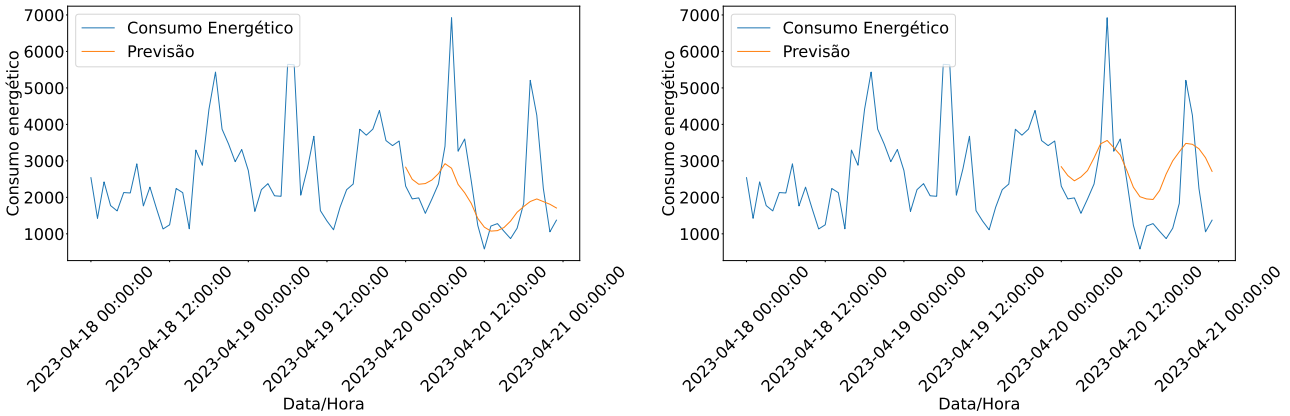
4.4.2 SVR

Na Tabela 4.6, são apresentados os resultados das métricas dos modelos SVR construídos, utilizando a biblioteca *skforecast*, com as duas abordagens multi-passo, recursiva e direta.

Abordagem	Hiperparâmetros	Abordagem no final			Abordagem variável		
		RMSE	MSE	MAE	RMSE	MSE	MAE
Recursiva	C=100, kernel='rbf'	1.30×10^3	1.68×10^6	8.31×10^2	4.13×10^3	2.50×10^7	3.37×10^3
Direta	C=100, kernel='rbf'	1.28×10^3	1.64×10^6	1.06×10^3	3.57×10^3	1.95×10^7	2.91×10^3

Tabela 4.6: Resultados dos modelos SVR

Na Fig. 4.11 são apresentados os resultados gráficos de ambos os modelo SVR construídos, onde na Fig. 4.11a é apresentado o gráfico do modelo SVR com abordagem multi-passo recursiva, e na Fig. 4.11b com a abordagem multi-passo direta.



(a) Multi-passo Recursiva

(b) Multi-passo Direta

Figura 4.11: Resultado gráfico dos modelos SVR construídos

4.5 Conclusão

Por forma a concluir o melhor modelo de predição, foi dada mais importância às métricas de desempenho MAE e RMSE, porque a MAE como pode não revelar a escala proporcional de um erro, a RMSE penaliza os erros maiores, complementando-se entre si, sendo que quanto maior a diferença entre o RMSE e MAE, pior o modelo de predição construído [Vau].

Quanto às duas abordagens de teste utilizadas, no final e variável no conjunto de dados, é dada mais importância à abordagem de teste variável no conjunto, por permitir uma análise mais realista, devido a ter sido executada em vários períodos.

Com base nos resultados apresentados na Secção 4.4, o modelo SARMA com os hiperparâmetros p=5, d=0, q=2, P=2, D=0, Q=1, S=24, na abordagem de teste variável no conjunto de dados, foi o que obteve o melhor resultado em ambas as métricas RMSE e MAE.

Comparando aos resultados obtidos pela abordagem de teste no final do conjunto de dados, os resultados diferem um pouco, onde de forma generalizada, os resultados são melhores nesta abordagem, sendo expectável visto terem apenas em consideração um período de teste. Nesta abordagem, o modelo que obteve melhor desempenho foi o SARIMAX com os hiperparâmetros p=5, d=1, q=2, P=2, D=1, Q=1, S=24 na métrica de desempenho RMSE, sendo que na MAE, o modelo SVR com os hiperparâmetros C=100, kernel='rbf' foi o que obteve o melhor resultado.

Nos resultados de todos os modelos da família ARIMA, é possível verificar que a componente de sazonalidade tem um impacto positivo no desempenho do modelo, e quanto às variáveis exógenas, verificou-se que nem sempre têm um impacto positivo no modelo, podendo a causa ser da qualidade dos dados e da não existência de uma relação muito forte entre os atributos. Outra componente que não teve um impacto positivo no desempenho do modelo foi a *Integrated*, onde é possível verificar que entre os gráficos da Fig. 4.9 e 4.10, a diferença é pouco percetível.

Quanto aos modelos SVR construídos com as duas abordagens multi-passo, direta e recursiva, é possível concluir que a multi-passo direta obteve melhores resultados em todas as métricas de desempenho na abordagem de teste variável no conjunto de dados, sendo que para a abordagem de teste no final do conjunto, apenas a métrica MAE é superior, sendo que se pode concluir, que a abordagem de predição multi-passo direta apresenta melhores resultados relativamente à abordagem multi-passo recursiva.

5

Conclusão e Trabalho Futuro

Neste Capítulo são apresentadas as conclusões do trabalho realizado no âmbito desta dissertação, bem como o trabalho futuro para melhorar todo o sistema apresentado.

5.1 Conclusão

Os dados provenientes de sensores IoT têm-se tornado cada vez mais relevantes, ao permitirem uma análise em tempo real, a construção de modelos de predição, a deteção de anomalias, a criação de automatismos, entre outros casos de uso, sendo que, através destes dados, é possível tomar decisões mais informadas, otimizar recursos, melhorar a eficiência energética, entre outros.

No trabalho apresentado nesta dissertação é apresentado a construção de um sistema de integração de dados IoT provenientes de sensores de consumo energético instalados em vários pontos do pólo da Mitra de Universidade de Évora, bem como da produção de energia pelo pavilhão gimnodesportivo.

Esta integração é dividida principalmente em quatro fases, a primeira em que são obtidos os dados dos sensores e como são uniformizados independentemente do protocolo de comunicação utilizado por cada

dispositivo, tendo em consideração também aspetos de segurança; a segunda fase, pelo seu processamento e armazenamento através de uma plataforma de desenvolvimento indicada para o processamento de dados IoT, neste caso a plataforma Fiware; a terceira fase, pela predição de dados do consumo energético através de modelos de aprendizagem automática; e por último, na quarta fase, a apresentação dos dados de forma organizada num *dashboard* interativo.

Os objetivos para esta dissertação foram, na maioria, alcançados, tendo sido realizada a implementação de um sistema que obtém os dados de diferentes tipos de sensores, os processa, e os armazena num sistema de gestão de base de dados apropriado. Esta integração apresenta um desempenho apropriado para o caso de uso do Smart Campus da Universidade de Évora, mas com o evoluir do número de dispositivos instalados, é necessário melhorar o seu desempenho, pois, como apresentado na Secção 3.8, a partir de 100 dispositivos os tempos de resposta começam a ser mais elevados. Outro aspeto importante que não foi totalmente conseguido, foi a segurança de todos os componentes utilizados, devido a alguns problemas com a plataforma Fiware utilizada.

Quanto à construção de um modelo de predição do consumo energético, os resultados obtidos não corresponderam às expetativas, sendo que uma das causas, foi a quantidade de dados disponível para treino ser de apenas de cerca de oito meses, revelando-se insuficiente para um modelo de predição. Apesar disso, foi possível perceber que a melhor abordagem seria a construção de um modelo da família ARIMA, sendo o modelo SARMAX o que apresentou melhores resultados para os dados existentes, como apresentado na Secção 4.4.

5.2 Trabalho Futuro

Nesta Secção são apresentadas algumas propostas para o trabalho futuro referentes à arquitetura do sistema, bem como aos modelos de predição utilizando aprendizagem automática, por forma a melhorar este desenvolvimento inicial, e a implementar novas funcionalidades interessantes.

5.2.1 Arquitetura do sistema

- Investigação e implementação de mecanismos de segurança em todos os componentes utilizados;
- Implementação da técnica de *downsampling* para mais períodos, minuto a minuto, diários, mensais, entre outros;
- Implementação de alarmes em tempo real, com base nos dados dos dispositivos;
- Implementação do treino contínuo do modelo de aprendizagem automática ao longo do tempo, e utilização de uma plataforma para a gestão do ciclo de vida de cada modelo;
- Investigação e melhoramento do desempenho do fluxo de dados, com mais dispositivos instalados;

5.2.2 Modelos de Predição

- Melhoramento do modelo de predição de dados;
- Avaliação de *Deep Learning* para a construção de um modelo de predição de dados do consumo energético;
- Implementação de um detetor de anomalias, através de um modelo de aprendizagem automática;

- Construção de um modelo de predição para os dados relacionados com a produção de energia pelos painéis fotovoltaicos;

Bibliografia

[AEG18] Kadir Amasyali and Nora M. El-Gohary. A review of data-driven building energy consumption prediction studies. *Renewable and Sustainable Energy Reviews*, 81:1192–1205, 2018.

[AMSÅ19] Victor Araujo, Karan Mitra, Saguna Saguna, and Christer Åhlund. Performance evaluation of fiware: A cloud-based iot platform for smart cities. *Journal of Parallel and Distributed Computing*, 132:250–261, 2019.

[AWS] AWS. Introduction to apache spark. https://aws.amazon.com/pt/big-data/what-is-spark/.

[BKPC21] Melvin Bender, Erkin Kirdan, Marc-Oliver Pahl, and Georg Carle. Open-source mqtt evaluation. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–4. IEEE, 2021.

[Bro] Jason Brownlee. 4 strategies for multi-step time series forecasting. https://machinelearningmastery.com/multi-step-time-series-forecasting/.

[Car13] Josiah Carlson. *Redis in action*. Simon and Schuster, 2013.

[Cha19] Anjali Chauhan. A review on various aspects of mongodb databases. *International Journal of Engineering Research & Technology (IJERT)*, 8(05):90–92, 2019.

[Con19] Confluet.io. Machine learning with python, jupyter, ksql and tensorflow. https://www.confluent.io/blog/machine-learning-with-python-jupyter-ksql-tensorflow/, Fevereiro 2019.

[Craa] CrateDB. What is cratedb. https://crate.io/products/cratedb?utm_referrer=https%3A%2F%2Fcrate.io%2F.

[Crab] Inc Crate.IO. Cratedb vs influxdb: Benchmark. https://crate.io/resources/white-papers/lp-wp-timeseries.

[Dan] Piotr Daniłowski. What is the real range of lora? https://yosensi.io/posts/what_is_the_real_range_of_lora/.

[DK18] Shilpa Devalal and A. Karthikeyan. Lora technology - an overview. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 284–290, 2018.

[DPDLP18] Lucio Tommaso De Paolis, Valerio De Luca, and Roberto Paiano. Sensor data collection and analytics with thingsboard and spark streaming. In *2018 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS)*, pages 1–6, 2018.

[epb] epb. How far will your wi-fi signal reach? https://epb.com/get-connected/gig-internet/how-far-will-your-wi-fi-signal-reach/.

[Eri] Ericsson. 5g vs 4g. https://www.ericsson.com/en/5g/5g-vs-4g.

[Eur] Comissão Europeia. Fiware – a european success story. https://digital-strategy.ec.europa.eu/en/news/fiware-european-success-story.

[FIWa] FIWARE. About fiware. https://www.fiware.org/about-us/.

[Fiwb] Fiware. Quantumleap. https://quantumleap.readthedocs.io/en/latest/.

[Fiwc] Fiware. Welcome to orion context broker. https://fiware-orion.readthedocs.io/en/master/.

[Fiwd] Fiware. Welcome to the fiware iot agent for ultralight 2.0. https://fiware-iotagent-ul.readthedocs.io/en/latest/.

[HMMHZ20] Naser Hossein Motlagh, Mahsa Mohammadrezaei, Julian Hunt, and Behnam Zakeri. Internet of things (iot) and the energy sector. *Energies*, 13(2):494, 2020.

[Inc] Influx Inc. Influxdb tops cassandra in time series data & metrics benchmark. https://www.influxdata.com/blog/influxdb-vs-cassandra-time-series/.

[Inf] Inc. InfluxData. Query data in influxdb. https://docs.influxdata.com/influxdb/v2.7/query-data/.

[Kaf] Apache Kafka. Apache kafka. https://kafka.apache.org/.

[Mac] Macrometa. Apache spark vs flink. https://www.macrometa.com/event-stream-processing/spark-vs-flink.

[Mat] Matej Matkuliak. Ingesting with cratedb. https://crate.io/blog/ingesting-with-cratedb.

[MLC21] Pedro Martins, Sérgio I Lopes, and António Curado. Designing a fiware-based smart campus with iot edge-enabled intelligence. In *World Conference on Information Systems and Technologies*, pages 557–569. Springer, 2021.

[Mon] Inc. MongoDB. A guide to horizontal vs vertical scaling. https://www.mongodb.com/basics/horizontal-vs-vertical-scaling.

[MyS] MySQL. Mysql enterprise scalability. https://www.mysql.com/products/enterprise/scalability.html.

[Nai17] Nitin Naik. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–7, 2017.

[NYZ17] Syeda Noor Zehra Naqvi, Sofia Yfantidou, and Esteban Zimányi. Time series databases and influxdb. *Studienarbeit, Université Libre de Bruxelles*, 12, 2017.

[pan] Pandas - data correlations. https://www.w3schools.com/python/pandas/pandas_correlations.asp.

[Pat] Savan Patel. Chapter 2 : Svm (support vector machine) — theory. https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72.

[Pes] Akash Peshin. What is the range of bluetooth and how can it be extended? https://www.scienceabc.com/innovation/what-is-the-range-of-bluetooth-and-how-can-it-be-extended.html.

[PMOK18] Dr. Yusuf Perwej, Husamuddin Mohammed, Majzoob Omer, and Bedine Kerim. A comprehend the apache flink in big data environments. *IOSR Journal of Computer Engineering (IOSR-JCE), e-ISSN: 2278-0661,p-ISSN: 2278-8727,www.iosrjournals.org*, Volume 20:Page 48 – 58, 03 2018.

[Roy] Dipta Roy. Mysql and its use cases. https://byte.builders/blog/post/mysql-and-its-use-cases/.

[Set] Alakh Sethi. Support vector regression tutorial for machine learning. https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/.

[Sha] Rahul Shah. Tune hyperparameters with gridsearchcv. https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/.

[SL] Scikit-Learn. Specifying an objective metric. https://scikit-learn.org/stable/modules/grid_search.html#specifying-an-objective-metric.

[sma] What is a smart campus? https://www.smartcampusbcit.ca/what-is-a-smart-campus.

[Spaa] Apache Spark. Machine learning library (mllib) guide. https://spark.apache.org/docs/latest/ml-guide.html.

[Spab] Apache Spark. Spark overview. https://spark.apache.org/docs/2.2.0/.

[SSH19] Yoshiko Sueda, Mizuki Sato, and Kazuo Hasuike. Evaluation of message protocols for iot. In *2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD)*, pages 172–175. IEEE, 2019.

[Tea] Towards AI Editorial Team. Support vector machine (svm) introduction — machine learning. https://vitalflux.com/svm-rbf-kernel-parameters-code-sample/.

[Thia] ThingsBoard. Thingsboard architecture. https://thingsboard.io/docs/reference/.

[Thib] ThingsBoard. Thingsboard databases. https://thingsboard.io/docs/reference/#sql-vs-nosql-vs-hybrid-database-approach.

[TOMdOS+18] Caio Thomás Oliveira, Rodrigo Moreira, Flávio de Oliveira Silva, Rodrigo Sanches Miani, and Pedro Frosi Rosa. Improving security on iot applications based on the fiware platform. In *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, pages 686–693, 2018.

[Vau] Mystery Vault. A guide to different evaluation metrics for time series forecasting models. https://analyticsindiamag.com/a-guide-to-different-evaluation-metrics-for-time-series-forecasting-models/.

[VTACC21] Washington Velasquez, Luis Tobar-Andrade, and Ivan Cedeno-Campoverde. Monitoring and data processing architecture using the fiware platform for a renewable energy systems. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 1383–1387, 2021.

[Wika] Wikipedia. Lora. https://pt.wikipedia.org/wiki/LoRa.

[Wikb] Wikipedia. Wi-fi. https://pt.wikipedia.org/wiki/Wi-Fi.

UNIVERSIDADE DE ÉVORA
INSTITUTO DE INVESTIGAÇÃO
E FORMAÇÃO AVANÇADA

Contactos:
Universidade de Évora
Instituto de Investigação e Formação Avançada — IIFA
Palácio do Vimioso | Largo Marquês de Marialva, Apart. 94
7002 - 554 Évora | Portugal
Tel: (+351) 266 706 581
Fax: (+351) 266 744 677
email: iifa@uevora.pt