U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

DEMONSTRAÇÃO DO POTENCIAL DOS DIGITAL TWINS DE EDIFÍCIOS EM AMBIENTE EXPERIMENTAL

DANIEL FILIPE COUTO SANTOS

Dissertação submetida para satisfação parcial dos requisitos do grau de

MESTRE EM ENGENHARIA CIVIL — ESPECIALIZAÇÃO EM CONSTRUÇÕES

Orientador: Professor Doutor Nuno Manuel Monteiro Ramos

Coorientador: Doutor Fernando Pedro Fernandes Pereira

JUNHO DE 2023

MESTRADO EM ENGENHARIA CIVIL 2022/2023

DEPARTAMENTO DE ENGENHARIA CIVIL

Tel. +351-22-508 1901

✉ m.ec@fe.up.pt

Editado por

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Rua Dr. Roberto Frias

4200-465 PORTO

Portugal

Tel. +351-22-508 1400

✉ feup@fe.up.pt

http://www.fe.up.pt

Reproduções parciais deste documento serão autorizadas na condição que seja mencionado o Autor e feita referência a *Mestrado em Engenharia Civil - 2022/2023 - Departamento de Engenharia Civil, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2023*.

As opiniões e informações incluídas neste documento representam unicamente o ponto de vista do respetivo Autor, não podendo o Editor aceitar qualquer responsabilidade legal ou outra em relação a erros ou omissões que possam existir.

Este documento foi produzido a partir de versão eletrónica fornecida pelo respetivo Autor.

Aos meus Pais e irmã

O sucesso é a soma de pequenos esforços repetidos dia após dia.

Robert Collier

AGRADECIMENTOS

Nesta fase final de mais um ciclo, gostaria de expressar minha profunda gratidão a todas as pessoas que, de alguma forma, estiveram ao meu lado e me apoiaram ao longo deste percurso, tornando possível a conclusão deste trabalho.

Os meus agradecimentos aos meus pais e irmã que me transmitiram valores que foram fundamentais para a minha formação. Agradeço pelo constante apoio e suporte que me proporcionaram em todas as decisões que tomei até então.

Expresso a minha profunda gratidão para com os meus colegas de curso por todo o apoio e colaboração, juntos enfrentamos desafios, celebramos conquistas e construímos memórias que irei valorizar para sempre.

Muito obrigado a toda a minha família e amigos por estarem sempre do meu lado e apoiarem nos momentos mais difíceis.

Isto não seria possível sem o apoio incansável do meu orientador, Professor Doutor Nuno Ramos, e coorientador, Professor Doutor Pedro Pereira. A generosidade em partilhar conhecimento e dedicação em esclarecer todas as minhas dúvidas ao longo destes meses de trabalho foram essenciais para concluir este projeto.

Agradeço também ao engenheiro Rui Cunha por toda a ajuda prestada que me permitiu superar diversos obstáculos.

RESUMO

Ao longo desta dissertação é abordado de forma abrangente o conceito de Digital Twins aplicado em diversas indústrias, contudo, o principal foco deste trabalho é abordar os Digital Building Twins, que são Digital Twins aplicados à indústria AEC (Arquitetura, Engenharia e Construção). São apresentados exemplos de casos práticos já aplicados que ilustram as funcionalidades destes modelos digitais na otimização da eficiência energética e da qualidade do ambiente interior em edifícios, IEQ.

Um dos principais pontos deste trabalho é a criação de um modelo Digital Building Twin de uma sala da FEUP, com base nos parâmetros ambientais interiores medidos através de um sistema de monitorização. Este sistema de monitorização consiste num conjunto de sensores que medem 12 parâmetros de IEQ, incluindo temperatura, humidade relativa, concentração de CO_2, entre outros. Esses sensores permitem a recolha contínua e precisa de dados relevantes para a análise do ambiente interior. Para a criação do Digital Building Twin, foi concebido um modelo 3D da sala, reunindo este modelo com os dados recolhidos pelo sistema de monitorização foi possível conceber o Digital Building Twin.

Foi desenvolvido um código em Python com a finalidade de automatizar o funcionamento do atuador de janela instalado na sala, que estava ligado ao sistema de monitorização. O atuador era acionado tendo em consideração valores definidos para um parâmetro medido, a concentração de CO_2, e um parâmetro calculado, a pressão parcial do vapor de água, através da temperatura e humidade relativa (parâmetros medidos pelo sistema de monitorização).

Para facilitar a visualização e a interação com o modelo Digital Building Twin foi desenvolvido um Dashboard. Esta interface apresenta o modelo 3D da sala e disponibiliza informações sobre os parâmetros medidos, permitindo ao utilizador a análise dos valores obtidos com os limites definidos como aceitáveis. Esta integração possibilita a visualização dos dados recolhidos em tempo real, assim como o armazenamento histórico dos dados por meio de gráficos e indicadores, bem como dados estatísticos.

Em suma, esta dissertação abrange desde a criação do modelo Digital Building Twin até a implementação de um sistema de monitorização e a sua operabilidade com um atuador de janela, oferecendo uma visão completa e prática do potencial dessas tecnologias para a melhoria dos edifícios, nomeadamente na qualidade do ambiente interior, IEQ.

PALAVRAS-CHAVE: Digital Building Twin, IEQ – Qualidade do ambiente interior, sistema de monitorização, Atuador de janela, Dashboard.

ABSTRACT

Throughout this dissertation the concept of Digital Twins applied in various industries is comprehensively addressed, however, the main focus of this work is to address Digital Building Twins, which are Digital Twins applied to the AEC industry (Architecture, Engineering and Construction). Examples of practical cases already applied are presented that illustrate the functionalities of these digital models in optimizing energy efficiency and indoor environmental quality in buildings, IEQ.

One of the main points of this work is the creation of a Digital Building Twin model of a room at FEUP, based on indoor environmental parameters measured through a monitoring system. This monitoring system consists of a set of sensors that measure 12 IEQ parameters, including temperature, relative humidity, CO_2 concentration, among others. These sensors enable the continuous and accurate collection of relevant data for the analysis of the indoor environment. To create the Digital Building Twin, a 3D model of the room was designed, bringing this model together with the data collected by the monitoring system it was possible to design the Digital Building Twin.

A Python code was developed in order to automate the operation of the window actuator installed in the room, which was connected to the monitoring system. The actuator was triggered taking into account values set for a measured parameter, the CO_2 concentration, and a calculated parameter, the partial pressure of water vapor, through temperature and relative humidity (parameters measured by the monitoring system).

To facilitate visualization and interaction with the Digital Building Twin model, a Dashboard was developed. This interface displays the 3D model of the room and provides information about the measured parameters, allowing the user to analyze the obtained values with the limits defined as acceptable. This integration allows the visualization of the collected data in real time, as well as the historical storage of data through graphs and indicators, and statistical data.

In summary, this dissertation covers from the creation of the Digital Building Twin model to the implementation of a monitoring system and its operability with a window actuator, offering a complete and practical view of the potential of these technologies for the improvement of buildings, namely in indoor environmental quality, IEQ.

KEYWORDS: Digital Building Twin, IEQ - Indoor Environmental Quality, Monitoring System, Window Actuator, Dashboard.

ÍNDICE GERAL

Agradecimentos i
Resumo iii
Abstract v
1.INTRODUÇÃO 1
1.1. Enquadramento e Motivação 1
1.2. Objetivos 1
1.3. Organização e Estrutura 2
2.ESTADO DE ARTE E FUNDAMENTOS 3
2.1. Enquadramento 3
2.2. Digital Twins 3
2.3. Digital Building Twin 4
2.4. Digital Building Twin – casos de estudo 5
2.4.1. Teatro em Norrköping, Suécia 5
2.4.2. Edifício em Larisa, Grécia 6
2.4.3. Departamento de Arquitetura Do Politécnico De Milão 7
2.4.4. Comparação dos métodos utilizados na criação dos Digital Building Twins 8
2.5. Eficiência energética 9
2.6. IEQ – Qualidade do ambiente interior 10
2.6.1. Conforto Térmico 11
2.6.2. Conforto Acústico 12
2.6.3. Conforto Visual 13
2.6.4. QAI - qualidade do ar interior 13
2.7. Software e ferramentas para a criação do Digital Building Twin 14
2.7.1 Revit 15
2.7.2. 3DBI 15
2.7.3. Dynamo 15
2.7.4. Excel 16
2.7.5. MariaDB 16
2.7.6. Python 16
2.7.7. Power BI 16
2.7.7. Mosquitto 17
3.METODOLOGIA 19

3.1. Considerações Iniciais 19
3.2. Digiroom 21
3.3. Modelo 3D 21
3.3.1. Criação do modelo 3D em Revit 21
3.3.2. Inserção do modelo 3D no Power BI 23
3.3.3. Exportação dos dados do modelo 3D através do Dynamo 25
3.4. Sistema de monitorização 29
3.4.1. Inserção dos dados recolhidos pelos sensores no Power BI 31
3.5. Dados no Power BI 31
re3.5.1. Organização dos dados 31
3.5.2. Criação de indicadores de dados em tempo Real 35
3.5.3. Dados Históricos 36
3.5.3.1. Criação de indicadores de valor médio 38
3.5.3.2. Criação de gráficos de linhas 40
3.5.3.3. Criação de gráficos de barras 41
3.5.4. Dados Estatísticos 44
3.6. Criação do Dashboard 45
3.7. Atuador de Janela 46
3.7.1. Controlo da janela através da linha de comandos 47
3.7.2. Controlo da janela através do Python 48
3.7.2.1. Pressão de saturação e pressão parcial do vapor de água 49
3.7.2.2. Descrição do código criado em Python 50
4.RESULTADOS 55
4.1. Atuador de Janela 55
4.1.1. Comparação de dados: Impacto do uso e não uso de atuador 57
4.2. Dashboard 59
5.CONCLUSÕES 65
5.1. Conclusões finais 65
5.2. Desenvolvimentos futuros 66
REFERÊNCIAS BIBLIOGRÁFICAS 67
ANEXOS 72

ÍNDICE DE FIGURAS

Fig. 1 - Componentes de um Digital Twin [6] 4
Fig. 2 - Peso dos parâmetros do IEQ 10
Fig. 3 - Parâmetros da qualidade do ambiente interior 11
Fig. 4 - Visualização dos dados no Revit [25] 12
Fig. 5 - 3DBI no Revit 15
Fig. 6 - Funcionamento do Power BI [38] 17
Fig. 7 - Fluxograma geral das etapas para a criação do Dashboard 20
Fig. 8 - Planta do Digiroom 21
Fig. 9 - Construir – Revit 22
Fig. 10 - Ambiente e Área - Revit 22
Fig. 11 – Vista 3D do modelo Revit 23
Fig. 12 - 3DBI – Exportação do modelo do Revit 24
Fig. 13 - 3DBI – Inserção do modelo 3D no Power BI 25
Fig. 14 - Script em Dynamo para extração dos dados das salas 28
Fig. 15 - Folha Excel com os dados extraídos do Revit 29
Fig. 16 - Ligação do sistema de monitorização ao Power BI 31
Fig. 17 - Organização das tabelas 32
Fig. 18 - Tabela "api_device" 32
Fig. 19 - Tabela "api_component" 33
Fig. 20 - Tabela "api_componentoutput" 33
Fig. 21 - Tabela "api_measurement" 34
Fig. 22 - Power BI - Coluna condicional 35
Fig. 23 - Indicador em tempo real – Concentração de CO_2 35
Fig. 24 - Criação de indicador em tempo real 36
Fig. 25 – Filtragem temporal 37
Fig. 26 - Processo de criação do filtro 37
Fig. 27 – Indicador – Concentração de CO_2 38
Fig. 28 - Processo de criação dos indicadores 39
Fig. 29 - Gráfico de linhas - Concentração de CO_2 40
Fig. 30 - Criação de Gráficos de linhas 41
Fig. 31 - Gráfico de barras 42
Fig. 32 - Criação de gráficos de barras 43
Fig. 33 - Dados estatísticos – Concentração de CO_2 44
Fig. 34 - Criação do Dashboard 45
Fig. 35 - Introdução do tema do Dashboard 46
Fig. 36 - Atuador de janela 47
Fig. 37 - Comando para conectar à janela 47
Fig. 38 - Código atuador de janela completo - Linha de comandos 48
Fig. 39 – Fluxograma com condições para ativar a janela 50
Fig. 40 - Python - Cálculo da pressão parcial do vapor de água 51
Fig. 41 - Python - Definição dos comandos da janela 51
Fig. 42 - Python - Envio dos comandos ao atuador de janela 52
Fig. 43 - Python - Condições para abrir e fechar janela 52
Fig. 44 - Python - Registo da data e hora e criação de ficheiro de texto 52
Fig. 45 - Python - Envio de dados para ficheiro de texto 52
Fig. 46 - Python - Guardar dados numa folha Excel 53

Fig. 47 - Janela fechada 55
Fig. 48 - Janela aberta 55
Fig. 49 - Registo de mensagens 56
Fig. 50 - Concentração de CO_2 com o atuador de janela ativo 57
Fig. 51 – Dashboard - Modelo 3D e informações 60
Fig. 52 – Dashboard - Dados em tempo real 60
Fig. 53 – Dashboard - Dados históricos 61
Fig. 54 – Dashboard - Dados históricos (continuação) 62
Fig. 55 – Dashboard - Dados estatísticos 63

ÍNDICE DE TABELAS

Tab. 1 - Comparação de métodos utilizados na criação de Digital Building Twins..... 9
Tab. 2 – Sensores utilizados 29
Tab. 3 - Fórmulas de Cálculo Estatístico..... 44
Tab. 4 - Comandos para atuador de janela - Linha de comandos..... 48

SÍMBOLOS, ACRÓNIMOS E ABREVIATURAS

DEC - Departamento de Engenharia Civil

FEUP – Faculdade de Engenharia da Universidade do Porto

DT – Digital Twin

AEC – Arquitetura, Engenharia e Construção

IEQ – Qualidade do Ambiente Interior

QAI – Qualidade do Ar Interior

BIM - Building Information Modeling

BMS – Building Management System

IoT - Internet of Things

UTA – Unidade de Tratamento de Ar

AVAC - Aquecimento, Ventilação e Ar Condicionado

CCTV – Closed-Circuit Television

CO_2 - Dióxido de carbono

TVOCs – Compostos Orgânicos Voláteis Totais

HCHO – Formaldeído

COV - Compostos Orgânicos Voláteis

Tab – Tabela

Fig - Figura

1. INTRODUÇÃO

1.1. ENQUADRAMENTO E MOTIVAÇÃO

A digitalização e a tecnologia desempenham um papel cada vez mais crucial nos dias de hoje e a indústria da arquitetura, engenharia e construção (AEC) não é exceção. Nesse contexto, os Digital Building Twins emergem como uma abordagem inovadora e promissora para a gestão eficiente de edifícios.

No entanto, a utilização dos Digital Building Twins ainda requer investigação e demonstração mais aprofundadas do seu potencial. Essa necessidade de estudo torna-se especialmente relevante diante dos desafios enfrentados na gestão e operação eficiente de edifícios, como a eficiência energética e a qualidade do ambiente interior, IEQ.

Nesse contexto, esta pesquisa tem como objetivo principal demonstrar o potencial dos Digital Building Twins aplicados a um caso de estudo. Pretende-se explorar a aplicação destes modelos virtuais na melhoria da qualidade do ambiente interior do espaço em estudo.

Desta forma, este estudo procura oferecer uma contribuição significativa para a informação existente, ao explorar e demonstrar empiricamente o potencial dos Digital Building Twins. Acredita-se que os resultados deste estudo possam ser um estímulo para a implementação eficaz dessa tecnologia noutros modelos, impulsionando a eficiência energética, a qualidade do ambiente interior e a gestão sustentável de edifícios num futuro próximo.

1.2. OBJETIVOS

Neste trabalho, procurou-se evidenciar as vantagens e o proveito que pode ser retirado dos modelos de Digital Twins na indústria AEC (arquitetura, engenharia e construção). Embora já existam vários casos de sucesso, tecnologia dos Digital Building Twins está em expansão, revelando-se uma abordagem promissora e em crescimento na indústria AEC.

Assim, o principal objetivo desta dissertação é a criação de um modelo Digital Building Twin representativo de uma sala da FEUP, o Digiroom. Provenientes desta finalidade, foram propostos mais 3 objetivos a alcançar.

O primeiro, a implementação de um sistema de monitorização e tratamento de dados adequados a Digital Building Twins no ambiente experimental do Digiroom.

De seguida, a implementação de um atuador simples operado a partir do Digital Building Twin de forma automática no ambiente experimental do Digiroom.

Por fim, a criação de um Dashboard que possibilite apresentar ao utilizador uma representação do modelo físico da sala e visualizar os dados recolhidos pelo sistema de monitorização.

1.3. ORGANIZAÇÃO E ESTRUTURA

Este documento está organizado em cinco capítulos, cada um com sua estrutura e conteúdo específicos. A organização adotada visa proporcionar uma abordagem clara e coerente do trabalho desenvolvido. No capítulo presente, é apresentada a introdução do trabalho, estabelecendo o contexto e a relevância do tema abordado.

O segundo capítulo é dedicado ao "Estado de arte e fundamentos". Inicialmente, são discutidos os temas mais amplos relacionados ao tema, contextualizando o leitor. À medida que se avança no capítulo, aprofunda-se em tópicos mais específicos, explorando o conhecimento existente na área. Além disso, são apresentados os softwares utilizados no desenvolvimento do caso de estudo, fornecendo uma base sólida para a compreensão do trabalho.

O terceiro capítulo, concentra-se na aplicação dos conceitos teóricos discutidos anteriormente ao caso de estudo em questão. Neste capítulo, descrevem-se em detalhe os processos de criação que conduziram ao resultado final. Explicando minuciosamente o desenvolvimento do Dashboard e a conexão com o atuador de janela. Através dessa descrição, procura-se fornecer clareza sobre como estes elementos foram integrados no estudo.

No quarto capítulo são apresentados os resultados obtidos com a pesquisa realizada. Descreve-se cuidadosamente o modo de atuação da janela e expõe-se o Dashboard final desenvolvido.

Por fim, o quinto capítulo é reservado às "Conclusões". É feito um balanço dos principais pontos abordados ao longo do trabalho, dando resposta aos objetivos propostos. Além disso, apresentam-se propostas de desenvolvimentos futuros, apontando possíveis direções a estudos posteriores.

2. ESTADO DE ARTE E FUNDAMENTOS

2.1. ENQUADRAMENTO

De forma a tornar a investigação relevante, é importante realizar um estudo aprofundado e identificar o tema em questão. Neste capítulo são abordados os principais conceitos e tópicos relacionados com os Digital Building Twins, partindo do geral para o particular, utilizando exemplos.

Inicialmente, é abordada a origem dos Digital Twins e sua aplicação na indústria, logo de seguida é feita uma análise semelhante para os Digital Building Twins, que é a aplicação de Digital Twins na indústria da arquitetura, engenharia e construção, complementando este tema com casos de estudo de projetos reais que foram implementados com sucesso.

Posteriormente é explorada a importância da eficiência energética nos edifícios e de que forma os Digital Building Twins podem contribuir para otimizar o consumo energético, tornando os edifícios mais sustentáveis e eficientes. Um procedimento análogo é aplicado para a qualidade do ambiente interior (IEQ) e de que modo os Digital Building Twins podem auxiliar na monitorização e melhoria dos parâmetros de IEQ.

O capítulo finaliza com a apresentação de softwares e ferramentas disponíveis para a criação e gestão de Digital Building Twins, fornecendo uma visão geral das principais capacidades de cada um deles nesse campo.

2.2. DIGITAL TWINS

Paralelamente com a recente transformação digital em diferentes setores, o conceito de "Digital Twins" é visto como uma das últimas tendências tecnológicas que pode ser aplicada em diversos campos [1]. No entanto, a origem do conceito de Digital Twins não é recente. Foi introduzido pela primeira vez na área da engenharia aeroespacial na década de 1960, quando a NASA (Administração Nacional de Aeronáutica e Espaço) criou dois simuladores para refletir as condições espaciais, a fim de se preparar para a longa sessão de preparação de voo pré-missão no programa Apollo da NASA. Estes simuladores foram usados para salvar a tripulação da missão Apollo 13, simulando alternativas no modelo terrestre antes de serem implementadas [2]. O conceito foi retomado em 2002, quando foi apresentado por Grieves na Universidade de Michigan como parte de um projeto para criar um Centro de Gestão do Ciclo de Vida do Produto. O modelo consiste em três componentes: uma parte real, uma parte virtual e um sistema que permite o fluxo de dados entre ambas as partes [3]. Posteriormente, em 2010, a NASA definiu o termo Digital Twins como “uma simulação multi-física integrada, multi-escala e probabilística de um veículo ou sistema 'as-built', que utiliza os melhores modelos físicos disponíveis, dados de sensores atualizados, histórico da frota, etc., para refletir a vida do seu correspondente voador." [4].

Os Digital Twins são modelos virtuais que representam objetos ou sistemas reais. São criados com base em dados recolhidos por sensores, dispositivos de medição e/ou outras fontes de informação. Um digital twin é a representação virtual exata de um objeto ou sistema físico, incluindo as suas características e comportamentos. Através da utilização de sensores, software e outros dispositivos de monitorização, a interpretação dos dados pode ser utilizada para analisar e monitorizar o desempenho e condição da entidade em tempo real. Também pode ser utilizado para simular os seus comportamentos, permitindo-

nos fazer previsões sobre o seu comportamento no futuro de forma a utilizar este modelo para estudar e otimizar vários aspetos tais como funcionalidade, eficiência e sustentabilidade [4].

2.3. DIGITAL BUILDING TWINS

Os Digital Building Twins são uma evolução tecnológica na indústria da arquitetura, engenharia e construção (AEC) que vem ganhando expressão há pelo menos uma década. São referidos como "digital twins" ou "réplicas digitais" de edifícios e o seu papel é replicar edifícios em tempo real usando sensores [5]. Estes modelos digitais podem refletir todas as informações ao longo do ciclo de vida do edifício, o que ajuda na simulação e controle de edifícios ou construções através da interligação de sensores que por sua vez, recolhem, processam e analisam dados relevantes. Desde 2017 que a utilização de digital twins na indústria AEC tem vindo a ter um crescimento abrupto e tem sido amplamente estudada tanto em ambiente académico quanto industrial [5]. Até ao momento, o conceito mais geral que liga os digital twins com a indústria AEC é "a ligação entre tecnologias sob a forma de um modelo digitalizado de um bem físico, transmitindo dados em pelo menos uma direção, e monitorizando o elemento físico em tempo real" [6][7]. Comparativamente com a indústria aeroespacial os digital twins aplicados à indústria da construção estão ainda numa fase embrionária. Já foram criadas soluções comerciais chamadas digital twins do ativo, além das propostas por investigadores, mas dependem dos fornecedores e não há uniformidade na sua utilização, e/ou implementação técnica. No entanto, a utilização de modelos 3D não garante a simulação do mundo real [8].

Num modelo BIM, a representação de um edifício é definida através das suas características físicas e funcionais usando parâmetros e propriedades. Esta informação é especificada progressivamente durante o ciclo de vida do edifício desde a conceção até ao funcionamento. Assim, um modelo digital twin aplicado à construção surge como uma extensão do modelo BIM, vem alongar este período, até à fase de utilização do edifício, os Digital Building Twins são atualizados com dados em tempo real para fornecer um modelo reativo que atua dinamicamente sobre o ambiente em que se insere, conforme representado na Fig. 1 [9][10]. O modelo reativo é enriquecido com dados em tempo real fornecidos por sensores e dados sintéticos gerados a partir de simuladores. Estes dados ajudam a uma tomada de decisão mais informada e fazem previsões sobre como o edifício irá evoluir ou comportar-se no futuro. O digital building twin também pode ser utilizado para analisar sistemas de edifícios tais como aquecimento, ventilação e ar condicionado (AVAC), eletricidade e canalização, e para monitorizar o consumo de energia do edifício, sustentabilidade e desempenho global [6].

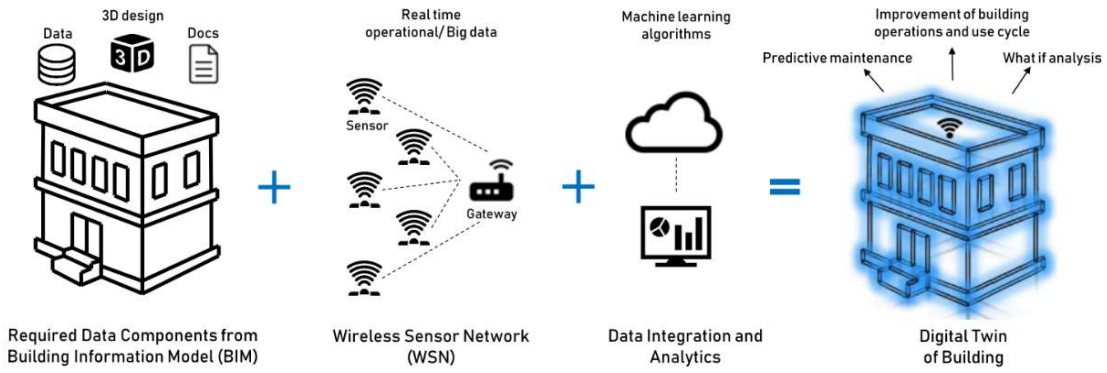


Fig. 1 - Componentes de um Digital Twin [6]

A criação de um Digital Building Twin envolve tipicamente várias etapas:

- Recolha de dados: Envolve a recolha de dados de várias fontes tais como planos de edifícios, dados de sensores e ferramentas de simulação.
- Criação de modelos: Os dados recolhidos são utilizados para criar um modelo digital do edifício, que inclui informação sobre a estrutura, sistemas e equipamento do edifício.
- Simulação e análise: O digital twin pode ser utilizado para simular vários cenários e analisar o desempenho do edifício, o consumo de energia e a sustentabilidade.
- Visualização: Os resultados da simulação e análise podem ser visualizados através de dashboards, gráficos e modelos 3D, para ajudar a compreender e a tomar decisões informadas sobre o funcionamento do edifício [4].

2.4. DIGITAL BUILDING TWIN – CASOS DE ESTUDO

Feita a contextualização dos Digital Building Twins, serão agora apresentados três exemplos de aplicação destes modelos em casos de estudo. Todos eles foram criados com propósitos diferentes, o primeiro, tem o objetivo de analisar o impacto dos ocupantes no ambiente interior de um edifício histórico e a sua conservação. O segundo, está ligado à vertente da eficiência energética e o terceiro foca-se na monotorização da ocupação, de forma a gerir as instalações do edifício.

2.4.1. TEATRO EM NORRKÖPING, SUÉCIA

Este estudo de caso foca-se num teatro localizado no centro da cidade de Norrköping, Suécia, onde foi criado um Digital Building Twin. Este teatro histórico foi construído em 1908 no estilo Art Nouveau e é agora um edifício protegido pela legislação histórica nacional. O teatro é usado para espetáculos e tem capacidade para 600 pessoas sentadas. O estudo em causa poderá ser uma inspiração para a transformação digital de edifícios históricos semelhantes, como teatros e igrejas, que têm grandes espaços interiores abertos e atraiam muitas pessoas durante as atividades [11].

A análise de dados neste estudo serve dois propósitos. O primeiro é investigar o impacto dos ocupantes no ambiente interior durante os espetáculos realizados no Teatro da Cidade e a relação entre as mudanças nos diferentes parâmetros ambientais. O segundo é inspecionar as flutuações da humidade relativa interior ao longo do ano civil para ver se há riscos que possam prejudicar a conservação do edifício [11].

Para recolher dados ambientais próximos dos valores médios da área dos lugares da plateia do salão, foi colocada uma caixa de sensores sob a cerca do segundo andar da bancada, perto do centro do salão. Foram utilizados cinco dispositivos de deteção para medir seis parâmetros ambientais interiores:

- DHT22 - mede temperatura e a humidade relativa;
- MH-Z16 - mede a concentração de CO_2;
- PPD42NS - deteta partículas com diâmetro superior a 1 μm;
- MIKOE-1630 - deteta vários gases que afetam a qualidade do ar, tais como amoníaco (NH3), óxidos de azoto (NOx), benzeno, e CO_2;
- Sensor Grove-Piezo - mede a flexibilidade, vibração, impacto e tato [11].

A recolha de dados começou em março de 2021 e continuará até o final de 2023. Os dados são recolhidos a cada 15 segundos para seis parâmetros ambientais: temperatura, humidade relativa, concentração de CO_2, concentração de poeira, concentração de gás nocivo e vibração. A colocação de mais caixas de sensores poderia fornecer uma melhor compreensão da distribuição espacial dos parâmetros ambientais,

mas a localização atual é representativa e os dados devem aproximar-se dos valores médios da área de maior concentração de ocupantes [11].

Os dados ambientais são recolhidos mais rapidamente do que o esperado, e por isso, o suficiente para detetar quaisquer alterações em tempo real. Foi utilizado o Azure Digital Twins [12] para implementar o Digital Building Twin. Os modelos de edifícios históricos são definidos usando DTDL (Digital Twin Definition Language) e a ontologia RealEstateCore [11]. O conhecimento obtido a partir do Digital Building Twin forneceu informações para melhorar tanto a conservação do património como o conforto humano do edifício. A análise de dados neste estudo tem dois objetivos. O primeiro é examinar como os ocupantes afetam o ambiente interno durante a realização de espetáculos no Teatro, e como as alterações nos diferentes parâmetros ambientais se relacionam. O segundo objetivo é analisar as variações da humidade relativa do ar dentro do teatro ao longo de todo o ano, com o intuito de identificar potenciais riscos que possam comprometer a conservação do espaço [11]. Como forma de visualização foi desenvolvido o Minerva utilizando o Dash Plotly e o Dash Bootstrap Components. A interface do utilizador, Minerva, suporta a visualização em tempo real e dados recolhidos historicamente, bem como a partilha desses dados [11]. Para trabalhos futuros, as condições ambientais interiores monitorizadas poderiam ser utilizadas para estimar os níveis de ocupação e para analisar outros parâmetros, como o consumo de energia, a fim de modelar os comportamentos energéticos do edifício histórico e reduzir os custos operacionais [11].

2.4.2. Edifício em Larisa, Grécia

Este estudo descreve como a tecnologia BIM foi transformada em Digital Building Twins, através de um caso de estudo realizado num prédio de dois andares na cidade de Larisa, no norte da Grécia. O edifício tem 40 pequenos apartamentos e uma área total de 1.746 m². A área de teste abrange uma área maior que inclui um jardim com vegetação e árvores. Em cada unidade habitacional, foi instalado um sensor que regista o consumo total de eletricidade em kilowatt-hora (kWh). Além disso, foi colocado um sensor na caldeira de água quente do edifício para medir a temperatura da água em graus Celsius (°C). Foram também instalados três sensores na área circundante do edifício para medir a temperatura ambiente, a humidade relativa e a intensidade do ar [13].

O software Autodesk Revit 2022 foi utilizado para produzir o modelo de informações de construção (BIM) que representasse o edifício. Esse modelo foi criado em LoD4 (Level of Detail), o que significa que descreve detalhadamente as características reais tanto internas quanto externas do edifício [13]. A transição do modelo BIM para o modelo Digital Building Twin é realizada usando o software Unreal Engine 4.27, que permite o desenvolvimento de soluções em tempo real para o setor de arquitetura, engenharia e construção (AEC). Para incorporar o BIM no Unreal Engine, é utilizado o Datasmith. O Datasmith é um add-on que estabelece uma conexão entre o software Revit e o Unreal Engine, permitindo a sincronização dos dois programas. Isso significa que qualquer alteração feita no BIM dentro do ambiente Revit é automaticamente transferida para o ambiente Unreal Engine e vice-versa [13].

Para visualizar as imagens captadas em tempo real pelas câmaras instaladas no edifício, é possível selecionar/clicar no elemento correspondente na interface do programa do DT. Se um microfone estiver devidamente instalado, o som captado será transmitido juntamente com a imagem. Além disso, se um software adequado estiver instalado num dispositivo informático no edifício, é possível aceder e gerenciá-lo remotamente. Consequentemente, a gestão de todos os dispositivos inteligentes do edifício que podem ser ligados e controlados através de um computador local. A única limitação é a conexão

desses dispositivos à Internet, que permite a transferência dos dados necessários para o ambiente virtual do DT [13].

Para visualizar os dados dos sensores e interagir com o modelo, foi necessário criar entidades físicas dos sensores no ambiente virtual. Depois de criados os modelos dos sensores, é possível definir as ações, forma e duração em que os dados serão exibidos na interface virtual, através da geração de um Evento Gráfico dos nós do projeto no software Unreal Engine. Ao selecionar o sensor desejado no modelo virtual, os dados correspondentes são exibidos no ecrã [13].

A partir da aplicação prática, foi possível constatar que a proposta permite a criação do DT de um edifício, permitindo a inclusão, representação e simulação de dados em tempo real. O DT desenvolvido pode conter uma grande quantidade de dados numéricos e informações recebidas por sensores, câmaras ou outros sistemas informáticos instalados no edifício. Além disso, é possível aceder e controlar remotamente os dispositivos inteligentes do edifício, permitindo o controle e modificação das operações de construção em tempo real por meio do ambiente virtual do DT [13].

2.4.3. Departamento de Arquitetura Do Politécnico De Milão

O Departamento de Arquitetura do Politécnico de Milão, Itália, foi o edifício sobre o qual foram implementadas campanhas de calibração e teste de um sistema de sensores baseados em câmaras IoT para monitorizar a ocupação como parte de um projeto de investigação em curso com o objetivo de definir um sistema de gestão de edifícios (BMS) para a gestão de instalações baseado num Digital Building Twin orientado para a ocupação. O edifício possui quatro andares e inclui escritórios administrativos, espaços de pesquisa e escritórios para universitários, totalizando 4300 m² de área bruta. Antes deste estudo, o edifício nunca havia sido monitorizado, o que significa que os dados sobre os padrões reais de ocupação, bem como informações sobre as atividades de limpeza e manutenção, não eram analisados ou otimizados [14].

Antes de implementar o sistema de sensores baseados em IoT para monitorizar o edifício, foi realizado um estudo preliminar para identificar as áreas críticas a serem monitorizadas e otimizar o número, posição e orientação dos sensores. Isso permitiu reduzir os custos de implementação e garantir um planeamento adequado do sistema de sensores. As análises preliminares incluíram:

- análise das funções dos espaços;
- análise da geometria dos espaços;
- simulação da localização e orientação dos sensores;
- análise dos sistemas elétricos, de dados e de comunicação.

Após a conclusão dessas análises, o sistema de sensores foi instalado e a primeira coleta de dados foi realizada durante um período de três meses. Este período foi considerado representativo, pois é o período mínimo necessário para identificar todas as atividades realizadas pelos utilizadores do departamento. Foram utilizados sensores baseados em câmaras devido à sua grande precisão e à capacidade de realizar outras análises, como a monitorização de segurança e proteção. Isso permite que novas funcionalidades sejam implementadas no sistema no futuro, os dados recolhidos foram analisados para identificar questões e falhas [14].

O foco deste estudo está na monotorização da ocupação, que faz parte de um projeto de pesquisa com a intenção de garantir a gestão de instalações baseada num Digital Building Twin direcionado para a ocupação. De acordo com os dados recolhidos, verificou-se que muitas pessoas entravam ou saíam de uma sala num curto espaço de tempo, o que poderia afetar a precisão do sistema de deteção. Para resolver este problema, foram implementadas rotinas automáticas no software para mitigar dados incorretos e

melhorar a precisão do sistema. Essas rotinas automáticas também ajudaram a evitar comportamentos enganadores dos utilizadores, como ficar em frente à porta ao abrir o escritório ou falar em frente à entrada de uma sala [14].

Foi utilizado um modelo BIM previamente criado, sobre o qual não existem informações acerca do software em que o modelo foi produzido. O objetivo do projeto é aprimorar a fase operacional do edifício utilizando técnicas avançadas de monitorização e análise de dados. Por outras palavras, o estudo procura entender como as pessoas usam o edifício para melhorar a sua eficiência e desempenho [14].

Os movimentos do utilizador podem ser rastreados em tempo real e visualizados na plataforma online SophyAI, sem armazenar imagens ou gravações de vídeo reais. A monitorização é feita de forma anónima para preservar a privacidade dos ocupantes [14].

Este estudo faz parte de um projeto de investigação em curso que visa desenvolver um Sistema de Gestão de Edifícios (BMS) para a gestão de instalações, com foco nas atividades de ocupação e limpeza em edifícios de escritórios, baseado num Digital Building Twin orientado para a ocupação. O objetivo do BMS é garantir uma melhor gestão do espaço, organização e limpeza, detetando níveis reais de ocupação e necessidades relacionadas com atividades de limpeza. As vantagens incluem a otimização de custos e a utilização do espaço, bem como atividades e contratos de limpeza personalizados [14].

2.4.4. Comparação dos métodos utilizados na criação dos Digital Building Twins

A criação de Digital Building Twins não se restringe a uma única abordagem, uma vez que existem diversas áreas a serem estudadas. Além disso, há múltiplas formas de recolher dados e uma variedade de softwares disponíveis que permitem a criação, simulação, análise e visualização desses modelos. Na Tab. 1 é feita uma análise comparativa dos métodos utilizados nas várias etapas da criação dos exemplos de Digital Building Twins abordados anteriormente.

Tab. 1- Comparação de métodos utilizados na criação de Digital Building Twins

	Recolha de dados	Criação de modelos	Simulação e análise	Visualização
Teatro em Norrköping, Suécia	Sensor de temperatura e humidade relativa Sensor de concentração de CO_2 Sensor de deteção de partículas com diâmetro superior a 1 µm Sensor de deteção de vários gases Sensor de medição da flexibilidade, vibração, impacto e tato	Azure Digital Twins Explorer	Azure Digital Twins	Minerva utilizando o Dash Plotly e o Dash Bootstrap Components
Edifício em Larisa, Grécia	Sensor de consumo total de eletricidade Sensor de temperatura da água	Autodesk Revit 2022	Unreal Engine 4.27	Unreal Engine Datasmith add-on
Departamento de Arquitetura do Politécnico de Milão, Itália	Sensores baseados em câmaras IoT	Modelo BIM	Sem informação sobre o software onde foi criado	SophyAI

2.5. EFICIÊNCIA ENERGÉTICA

A eficiência energética em edifícios pode ser definida como o uso eficiente da energia para atender às necessidades dos utilizadores de um edifício, minimizando o consumo de energia e, consequentemente, reduzindo os custos associados [15]. Tem sido dada uma grande ênfase à eficiência energética nos edifícios, já que estes são responsáveis por 40% do consumo de energia na Europa e cerca de 75% dos edifícios são ineficientes do ponto de vista energético. Além disso, é importante aplicar medidas de eficiência energética através da promoção de tecnologias inteligentes e digitais de consumo de energia, incentivando o uso de equipamentos eficientes e fornecendo informações aos consumidores para que eles possam usar a energia de maneira mais eficiente [16].

Num estudo realizado na Universidade de Agder em Grimstad, na Noruega, foi feito um controlo na unidade de tratamento de ar (UTA), que é responsável por regular o ar dentro do edifício, especialmente no que diz respeito ao aquecimento, ventilação e AVAC. Como forma de monitorizar a UTA foram instalados sensores de temperatura, pressão e caudal, estes sensores estavam instalados na UTA e os dados eram enviados para o modelo BIM, desta forma foi possível analisar as falhas operacionais do sistema, tendo resultado na diminuição do custo anual de energia de vários milhões de euros devido à correção desses problemas [17].

Outro estudo realizado no âmbito da eficiência energética, na universidade de engenharia de Daegu, na Coreia do Sul, consistiu na análise da eficiência do edifício no que diz respeito à iluminação. Além de conter informações sobre a arquitetura do edifício, a base de dados armazena informações sobre o sistema, incluindo detalhes como a quantidade de luzes, as características e especificações das luzes e o consumo de energia. Uma das salas da universidade foi monitorizada através de um sistema de vigilância, CCTV (Closed-Circuit Television), a partir do qual se desenvolveu a probabilidade de ocupação das salas, assim, foi possível otimizar o tempo em que as luzes estavam acesas desnecessariamente. Outro aspeto abordado foi a intensidade da luz, foram realizadas medições através de um sensor de fluxo luminoso, em que se verificou que em todas as salas o valor medido era superior ao necessário, desta forma foi possível regular a intensidade das luzes de modo a reduzir significativamente o consumo de energia da universidade [18].

2.6. IEQ – QUALIDADE DO AMBIENTE INTERIOR

A Qualidade do Ambiente Interior, designada em inglês Indoor Environmental Quality (IEQ) é um conceito que se aplica ao interior dos edifícios e descreve as condições nele presentes. Abrange tanto as condições térmicas, acústicas e luminosas, como a Qualidade do Ar Interior (QAI). Com base em 14 sistemas de certificação de edifícios verdes, foram analisados os parâmetros que utilizavam para avaliar o IEQ, verificou-se a contribuição média dos 4 parâmetros abrangidos pelo IEQ, que resultou no peso de 34% à QAI, 27% ao conforto térmico, 22% ao conforto visual e 17% ao conforto acústico, conforme ilustrado na Fig. 2 [19].

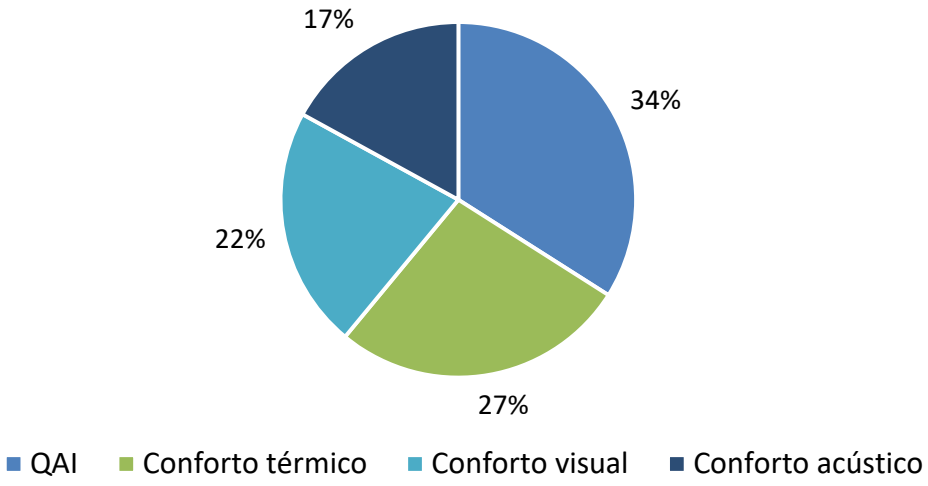


Fig. 2 - Peso dos parâmetros do IEQ

A perceção dos ocupantes sobre essas condições ambientais pode afetar o seu conforto, saúde e bem-estar, assim como a sua produtividade. Embora exista uma relação entre a qualidade do ambiente interior e a satisfação dos ocupantes, ainda não é completamente compreendida. A avaliação da Qualidade do Ambiente Interior pode ser feita através de medições objetivas, como a quantidade de poluentes, temperatura do ar, níveis de ruído e iluminação, ou por meio de questionários subjetivos realizados aos ocupantes. Em estudos de grande escala, as medições objetivas são geralmente limitadas por causa do seu custo elevado. Já os questionários são mais económicos, acessíveis, fáceis de serem realizados e podem ser aplicados simultaneamente em muitos edifícios. Os utilizadores dos edifícios são

considerados a "melhor fonte de informação no que respeita às suas necessidades e exigências de conforto" [20][21].

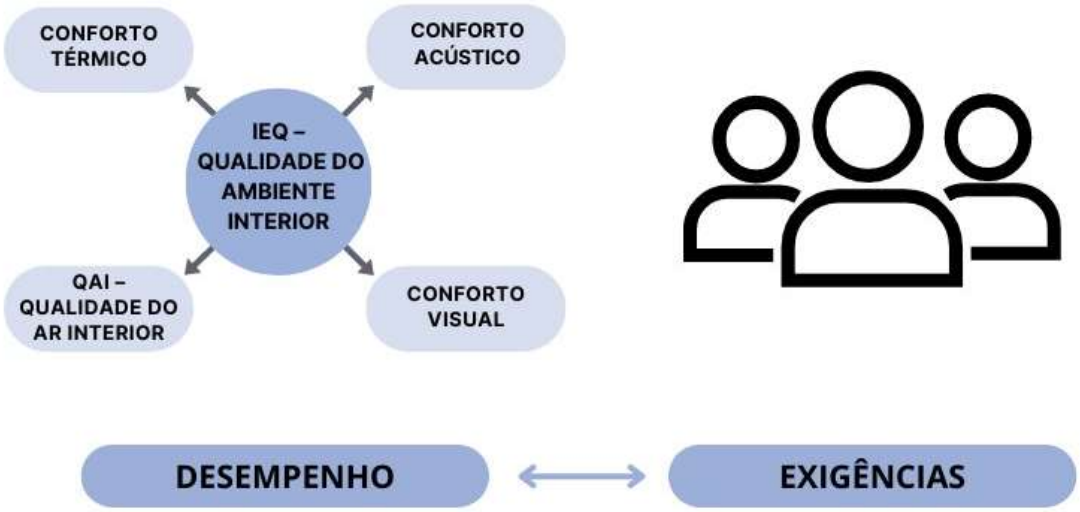


Fig. 3 - Parâmetros da qualidade do ambiente interior

2.6.1. Conforto Térmico

A qualidade do ambiente interior é considerada um dos principais fatores que influenciam a produtividade, conforto, saúde e bem-estar dos ocupantes de um espaço [22], e o conforto térmico é um dos aspetos mais importantes nesse contexto.

O conforto térmico é uma sensação subjetiva relacionada com a temperatura e depende de fatores físicos, fisiológicos e psicológicos. Os fatores físicos dizem respeito às trocas de calor entre o corpo e o ambiente, os fatores fisiológicos envolvem as adaptações do organismo às condições térmicas específicas e os fatores psicológicos estão relacionados à perceção de cada indivíduo. Os estudos sobre conforto térmico procuram compreender as condições ideais para ambientes que sejam adequadas às atividades humanas, assim como desenvolver métodos e princípios para analisar detalhadamente o desempenho térmico de um ambiente [23].

O conforto térmico é alcançado quando uma pessoa não sente nem calor, nem frio, o que requer um equilíbrio entre a temperatura do corpo humano e do ambiente circundante. No entanto, não é apenas a conservação da temperatura do corpo dentro de um intervalo aceitável que garante o conforto térmico. Este fator é resultado da combinação de seis parâmetros principais, são eles [24]:

- Atividade desempenhada;
- Isolamento térmico das roupas utilizadas;
- Temperatura do ar;
- Temperatura radiante média;
- Velocidade do ar;
- Humidade relativa.

Num estudo realizado no Canadá, foi desenvolvido um Digital Building Twin de um dos edifícios da Universidade de Manitoba, no Canada. Parte deste estudo incidiu no controlo do conforto térmico, realizando medições de temperatura através de um sensor. Foi considero que a temperatura ambiente de

referência estava compreendida entre os 21ºC e os 23ºC. O modelo digital, criado no Revit, apresentava o modelo do edifício e permitia visualizar os valores dos parâmetros de temperatura, Fig. 4 [25].



Fig. 4 - Visualização dos dados no Revit [25]

As leituras foram registadas para posterior análise através de um gráfico temporal. Além disso, foi feito um tratamento histórico dos dados, ao examinar os dados de uma sala, constatou-se que em aproximadamente 10% do tempo a temperatura do ar estava abaixo do intervalo de referência estabelecido, quando a sala estava ocupada. Além disso, durante as horas desocupadas, cerca de 7% do tempo registou-se uma temperatura acima do intervalo de referência [25].

Num outro estudo, realizado na Universidade Nacional de Singapura, foram recolhidos dados de sensores instalados na Universidade, que mediram a temperatura e humidade relativa. Além disso, foram obtidos dados a partir de um smartwatch, como o ritmo cardíaco, a temperatura corporal e a localização da pessoa. Esses dados foram utilizados para criar modelos que avaliaram o conforto individual e analisaram como as pessoas se sentiam durante a transição entre diferentes espaços [26].

2.6.2. Conforto Acústico

Assim como o conforto térmico, o conforto acústico pode definir-se como um estado de espírito que reflete satisfação com o ambiente que envolve a pessoa, um ambiente de comodidade e bem-estar. O conforto acústico é a sensação que se obtém num espaço interior quando os níveis de ruído são considerados confortáveis, seja o ruído proveniente do exterior do espaço ou gerado internamente. O conceito de conforto acústico é subjetivo e está intimamente ligado aos hábitos e práticas culturais das pessoas em determinado ambiente [27].

A qualidade do ambiente interior tem como um dos objetivos garantir o conforto acústico, visando proporcionar bem-estar aos utilizadores dos espaços. O desconforto acústico pode ter efeitos prejudiciais na capacidade de concentração, causar desconforto, dificultar a comunicação e em casos extremos pode ser prejudicial à saúde do ser humano [28].

Sendo o conforto acústico um dos parâmetros menos valorizados do IEQ, a sua aplicabilidade nos Digital Building Twins é quase nula, contudo foi desenvolvido um estudo em Barcelona que avançou algumas possibilidades de integração deste parâmetro nos Digital Building Twins através de medições

como o tempo de reverberação e nível de pressão sonora, permitindo realizar a monitorização deste parâmetro [29].

2.6.3. CONFORTO VISUAL

O conforto visual em edifícios refere-se à qualidade do ambiente visual dentro de um espaço construído, como uma casa, escritório, escola, hospital ou qualquer outro tipo de edifício. É um aspeto importante para a qualidade do ambiente interior, que procura criar um ambiente visualmente agradável e saudável para as pessoas que o utilizam.

A iluminação adequada é um dos principais fatores que afetam o conforto visual em edifícios. É importante que o espaço seja bem iluminado, com luz adequada para as atividades realizadas no local. A luz natural, proveniente de janelas ou claraboias, é considerada a melhor opção, pois oferece uma iluminação equilibrada e de maior qualidade comparada com a luz artificial. Além disso, aquando da instalação da iluminação artificial é importante que esta seja projetada corretamente, com uma distribuição uniforme da luz e evitando-se ofuscamento ou sombras incómodas [28].

Assim como o conforto acústico, o conforto visual é um parâmetro que não tem sido alvo de muita aplicação prática no ramo dos Digital Building Twins, contudo, um estudo numa Universidade de Barcelona que estudou todos os parâmetros de IEQ, abordou este tema aliado à eficiência energética, baseando a sua análise na recolha de dados através de sensores, obtendo os níveis de iluminação total e os níveis de iluminação artificial dos espaços. Fazendo uma comparação com o nível recomendado de iluminação para cada espaço, permitia que de forma automática fossem ligadas ou desligadas luzes de forma a manter os níveis de iluminação desejados [29].

2.6.4. QAI - QUALIDADE DO AR INTERIOR

Ao contrário do que acontece com outros recursos, o ar não é escasso, pelo contrário, a sua vulnerabilidade reside principalmente no seu risco de deterioração, dessa forma, a qualidade do ar interior é uma preocupação de saúde pública e profissional, uma vez que estamos diariamente expostos a poluentes atmosféricos, dado que passamos uma grande parte do nosso tempo em ambientes interiores [30]. A qualidade do ar interior pode ser afetada por numerosos fatores, desde os poluentes do exterior que chegam ao interior através da ventilação até aos materiais de construção, mobiliário, e atividades levadas a cabo pelos ocupantes do edifício exposição a poluentes atmosféricos tem estado ligada a uma panóplia de efeitos adversos sobre a nossa saúde, bem-estar e desempenho [30], [31].

Atualmente, as diretrizes de projeto estabelecidas em nível nacional para construções residenciais geralmente exigem uma quantidade mínima de trocas de ar por hora e incentivam a melhoria na vedação do edifício. Em contrapartida, a procura pela eficiência energética tem levado a uma redução na ventilação, o que pode resultar numa qualidade inadequada do ar interior (QAI). Em muitos casos, a redução das trocas de ar por hora tem como consequência direta a diminuição da qualidade do ar interior, o que pode afetar de forma negativa a saúde dos ocupantes [31].

A avaliação da qualidade do ar interior (QAI) pode envolver a medição de diversos parâmetros, dependendo das necessidades específicas e dos objetivos da avaliação. Alguns dos parâmetros comuns medidos para avaliar a QAI são [31]:

- Concentração de dióxido de carbono (CO_2): O CO_2 é um indicador comum da ventilação adequada em ambientes internos, uma vez que é produzido pela respiração humana e sua

concentração aumenta significativamente com o aumento da ocupação e naturalmente em ambientes mal ventilados.

- Partículas suspensas no ar (PM1, PM2,5, PM10): As partículas finas podem ter origem tanto de fontes externas como internas. Nas áreas urbanas, os veículos e as atividades econômicas são as principais fontes de origem externa. Já em ambientes internos, a literatura aponta para uma variedade de fatores, como mobiliário, ambientadores, perfumes, desodorizantes e cosméticos, combustão de combustíveis domésticos, atividades de cozinha (emissões de fumo e partículas de vapor de água) e fumo derivado do tabaco. É importante mencionar que a abertura de janelas geralmente está diretamente relacionada com um aumento destes níveis. No entanto, a ressuspensão de partículas transportadas pelo ar é a principal contribuição para os níveis internos de partículas suspensas no ar. Devido à relação entre os níveis internos e externos, a relação entre as partículas internas e externas é frequentemente utilizada como indicador.

- Compostos orgânicos voláteis totais (TVOCs): Os TVOCs são emitidos por diversos materiais de construção, mobiliário, produtos de limpeza e outras fontes internas, e sua concentração pode afetar a qualidade do ar interior. O aumento da ventilação é um procedimento comum para fazer diminuir a concentração dos compostos orgânicos voláteis.

- A temperatura e humidade relativa do ar interior são importantes para o conforto térmico dos ocupantes, mas também para a QAI pois podem influenciar a proliferação de fungos e ácaros.

- Ventilação: A taxa de renovação do ar, expressa em trocas de ar por hora, é um parâmetro importante para avaliar a eficiência da ventilação num ambiente interno [31].

Um estudo realizado na Universidade de Brescia, em Itália, descreve a criação de um Digital Building Twin debruçado na qualidade ar interior do próprio edifício, através da recolha de dados como a concentração de CO_2, temperatura e humidade relativa. As informações recolhidas pelos sensores eram utilizadas para otimizar os horários diários de aquecimento, ventilação e AVAC. Reunindo os dados destes três parâmetros registados foi possível calcular uma condição de conforto numa escala de 1 a 27. Esta escala representa todos os estados possíveis da condição interior, sendo 1 a melhor condição e 27 a pior [32].

2.7. SOFTWARE E FERRAMENTAS PARA A CRIAÇÃO DO DIGITAL BUILDING TWIN

Após terem sido estudados vários casos de sucesso na criação de Digital Building Twins foi analisado o software por eles utilizado. Não há um procedimento único para a criação de Digital Building Twins existindo diferentes ferramentas aplicáveis.

Com base nos casos de estudo, foram analisadas diversas ferramentas para a criação do modelo que permitissem alcançar o resultado desejado de forma a otimizar o trabalho tendo em consideração a operabilidade dos mesmos, assim como a viabilidade financeira para garantir a acessibilidade no âmbito académico. Seguidamente serão apresentadas as ferramentas utilizadas na criação dos Digital Building Twin. Estas ferramentas e softwares foram utilizadas de forma aberta tendo em conta o desenvolvimento em meio académico.

2.7.1 REVIT

O Revit é um software de modelação de informações da construção (BIM) desenvolvido pela Autodesk. É utilizado pela indústria da arquitetura, engenharia e construção para criar modelos 3D precisos de edifícios e estruturas. O Revit permite que os utilizadores criem desenhos de construção, modelos detalhados, documentação e visualizações, bem como análises de desempenho e simulações.

A principal característica que se destaca no BIM, e naturalmente no Revit, é que este software permite criar modelos 3D de um projeto ou construção, com características específicas que vão além das representações geométricas oferecidas por softwares CAD. Desse modo, no contexto dos Digital Building Twins este software é bastante útil na representação digital do modelo real. Além disso, é possível extrair informações úteis destes modelos, tornando a ferramenta mais completa e eficiente para profissionais da indústria AEC [33].

2.7.2. 3DBI

O 3DBI é um add-on que serve de transição entre o software de modelação, como o Revit, e a plataforma de visualização dos modelos, Microsoft Power BI, Fig. 5. Essa integração possibilita que se obtenha uma clara perceção do modelo BIM, sem prejudicar a sua visão espacial. A partir dessa união, através de alguns cliques é possível responder a perguntas quantitativas como o volume total de paredes de um determinado material, ou a área total de um tipo de compartimento numa zona específica.

Além disso esta ferramenta permite a obtenção de diversas fontes de dados externas, como folhas de Excel e respetiva inserção no modelo BIM. Deste modo, é possível cruzar uma infinidade de informações. Uma das funcionalidades mais convenientes no âmbito nos Digital Building Twins é a capacidade de exportar compartimentos e espaços como objetos 3D separados [34].

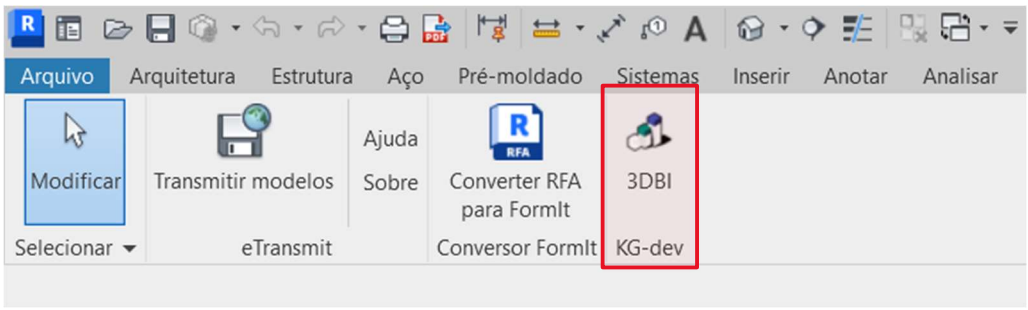


Fig. 5 - 3DBI no Revit

2.7.3. DYNAMO

O Dynamo é uma plataforma de software de código livre para design computacional e modelação de informações de construção (BIM). Através de uma interface visual descomplicada permite a automação de tarefas repetitivas e interações com o modelo Revit, tornando a execução de algumas operações mais céleres.

Para criar uma rotina em Dynamo, é necessário criar uma sequência de tarefas que podem ser executadas automaticamente ou manualmente. Cada tarefa pode ser um comando do Revit, uma operação matemática ou qualquer outra ação. Essas tarefas podem incluir alterar os valores dos parâmetros dos elementos, selecionar objetos, comparar valores, ler e até importar e exportar dados em folhas de Excel e criar ou excluir objetos no modelo Revit. A interoperabilidade entre o Revit e o Excel é uma das

principais vantagens do Dynamo, no ramo dos Digital Building Twins, permitindo organizar diversas informações do modelo [35].

2.7.4. EXCEL

O Microsoft Excel é um software de folhas de cálculo desenvolvido pela Microsoft. Permite aos utilizadores criarem, operarem e analisarem dados. É amplamente utilizado em diversos setores, incluindo a indústria AEC. Com a sua interface intuitiva e recursos avançados de cálculo, o Excel é uma ferramenta poderosa para a criação de gráficos, tabelas e relatórios. O Excel é compatível com outros softwares da Microsoft, como o Power BI, que será abordado à posteriori, permitindo assim que os utilizadores criem documentos mais complexos integrando os dados das folhas de cálculo [36].

2.7.5. MARIADB

O MariaDB é um sistema de banco de dados relacional de código aberto em nuvem que pode ser usado para armazenar e gerir dados num servidor. Este sistema é bastante útil para estabelecer a ligação entre os dados recolhidos no modelo real e o modelo digital, enviando esses dados para a plataforma de análise e tratamento dos mesmos [37].

2.7.6. PYTHON

O Python é uma linguagem de programação de alto nível, conhecida pela sua legibilidade e simplicidade. Uma das principais vantagens do Python é sua extensa biblioteca padrão, que oferece uma variedade de ferramentas e funcionalidades, o que permite desenvolver projetos de forma mais rápida e eficiente. O Python é bastante utilizado no campo da ciência de dados, e foi nesse contexto que foi utilizado. Através do Python é possível desenvolver códigos essenciais num modelo de um Digital Building Twin, através da linhas de comando pode ser feita a ligação entre uma base de dados e dispositivos passiveis de serem comandados à distância, isto é, mediante os comandos previamente programados são enviadas ordens aos dispositivos, estes comandos podem ser programados de forma que os dispositivos sejam acionados de forma automática, tendo por base os parâmetros medidos [37].

2.7.7. POWER BI

O Microsoft Power BI é uma ferramenta de análise de dados e criação de relatórios que oferece recursos avançados para transformar dados brutos em informações significativas e insights acionáveis. Este software é baseado em três componentes principais: o Power BI Desktop, o Power BI Service (mais conhecido por Power BI online) e o Power BI Mobile. O Power BI Desktop é a interface de criação de relatórios, onde os utilizadores podem criar painéis interativos, gráficos e relatórios. O Power BI Service é uma plataforma de nuvem que permite a partilha, a colaboração e a publicação de relatórios, criando desta forma um Dashboard. O Power BI Mobile é uma aplicação que permite aceder aos relatórios, painéis e Dashboards em dispositivos móveis, tornando a análise de dados acessível em qualquer lugar, Fig. 6.

O editor de consulta do Power BI Desktop, o Editor do Power Query, permite aos utilizadores conectarem-se a diversas fontes de dados, como folhas em Excel, arquivos CSV e fontes de dados na nuvem. Uma vez conectado, os utilizadores podem transformar e limpar os dados antes de carregá-los no modelo de dados do Power BI Desktop. Essa capacidade de conexão flexível permite que os

utilizadores acedam e combinem dados de várias fontes numa única visualização de relatório, possibilitando uma análise global dos dados.

A criação de relatórios no Power BI é altamente interativa e visualmente atraente. O Power BI oferece uma ampla variedade de tipos de visualização, incluindo gráficos de barras, gráficos circulares, mapas, tabelas, medidores e muito mais.

No que diz respeito aos Digital Building Twins, este software permite trabalhar uma série de dados a serem estudados e apresentá-los da forma desejada, criando assim a forma de visualização do modelo [38].

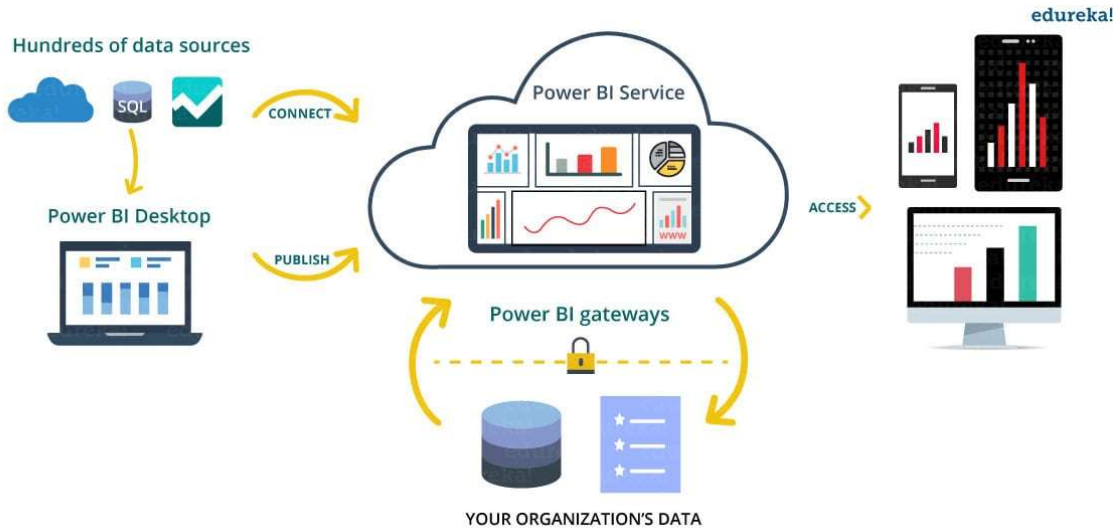


Fig. 6 - Funcionamento do Power BI [39]

2.7.7. MOSQUITTO

O Mosquitto é um software que utiliza o protocolo MQTT (Message Queuing Telemetry Transport) que permite a troca de mensagens entre dispositivos conectados a uma rede. A utilização deste software no ramo dos Digital Building Twins é bastante útil, permitindo receber dados de sensores, controlar dispositivos e emitir alertas [40].

3. METODOLOGIA

3.1. CONSIDERAÇÕES INICIAIS

Conforme descrito anteriormente, um dos focos desta dissertação é o desenvolvimento de um Digital Building Twin, para que isso seja possível é necessário o manuseamento de vários softwares e ferramentas, neste caso foram utilizados Revit, 3DBI, Dynamo, Excel, Maria DB, Python, Mosquitto e Power BI.

O caso de estudo foi previamente definido em conjunto com o orientador e coorientador para que fosse possível implementar na prática o desenvolvimento de um Digital Building Twin com enfase na eficiência energética e qualidade do ambiente interior, que são parâmetros de relevância para os ocupantes do espaço. O estudo debruça-se numa sala existente na Faculdade de Engenharia da Universidade do Porto, localizada no edifício H, denominada Digiroom, que posteriormente será apresentado com o detalhe necessário.

Um dos objetivos desta dissertação passou por compreender de que forma é possível desenvolver um Digital Twin na indústria AEC. Por essa razão, no capítulo precedente foi feita uma abordagem aos conceitos primordiais de Digital Twins, Digital Building Twins, Eficiência Energética e IEQ-Qualidade do Ambiente Interior, complementados com exemplos de aplicação já realizados e devidamente testados. Exemplos estes que serviram de referência para o desenvolvimento do Digital Building Twin do Digiroom.

O modelo desenvolvido, na ótica do utilizador consiste num Dashboard que reúne um conjunto de informações relevantes, indicadores que reproduzem em tempo real as propriedades medidas pelo sensor existente no Digiroom, fazendo com que seja possível saber em tempo real e à distância as condições interiores do espaço. O Dashboard conta também com o modelo 3D da sala para que o utilizador consiga identificar facilmente o espaço, ver a sua configuração e ter uma perceção aproximada da realidade de como é a sala que pretende monitorizar. Além disso, o Dashboard conta com um conjunto de medidores e gráficos de consulta histórica que indicam cada um dos parâmetros enviados pelo sensor. Estes gráficos, permitem uma filtragem anual, mensal e até mesmo diária para que se possa estudar cada parâmetro individualmente e de forma precisa. Finaliza com um conjunto de dados estatísticos e com gráficos representativos da conjugação de vários parâmetros.

O fluxograma apresentado na Fig. 7 representa de uma forma global as etapas percorridas no percurso efetuado para atingir o objetivo desta dissertação, a criação de um Dashboard para o Digiroom.

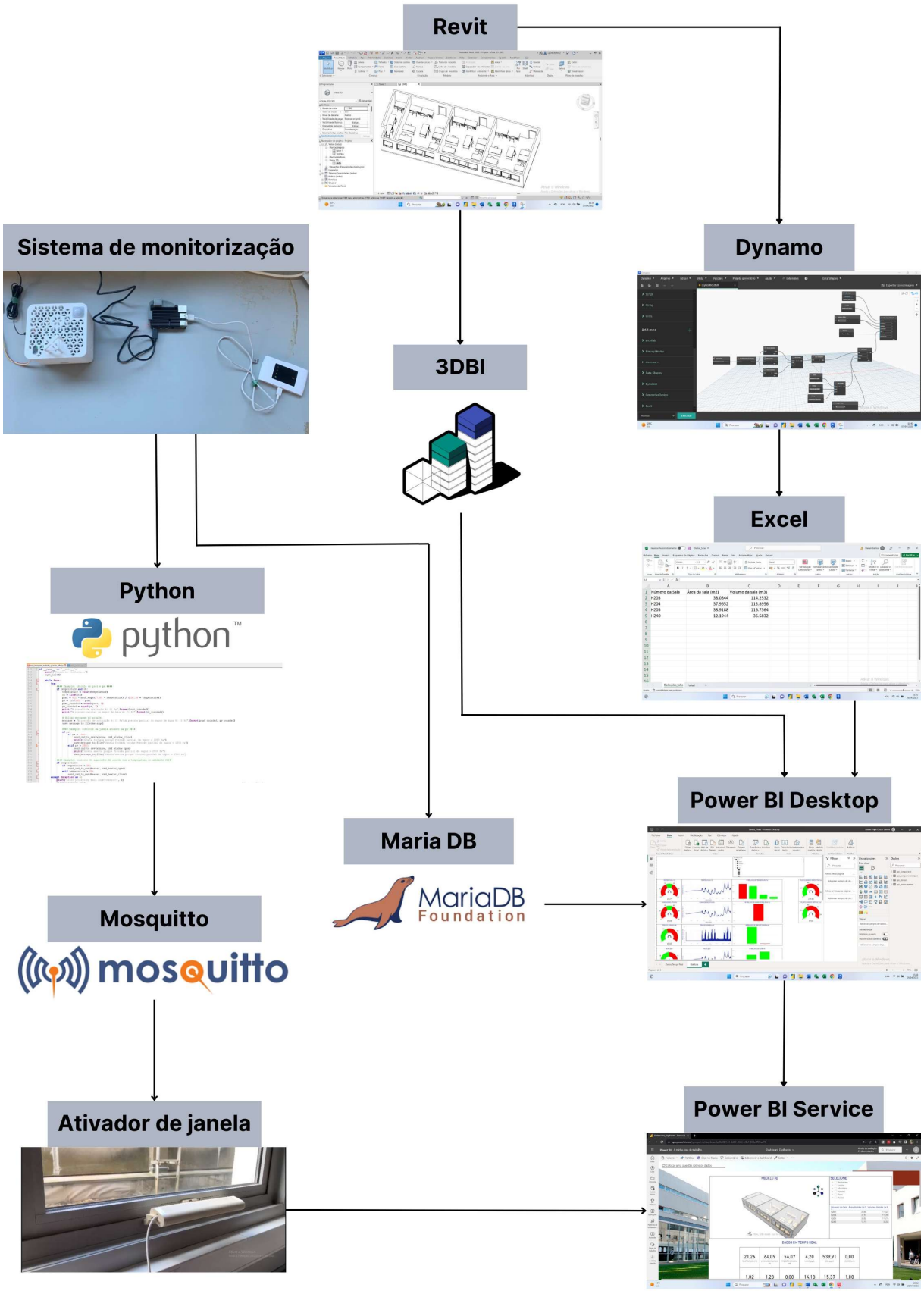


Fig. 7 - Fluxograma geral das etapas para a criação do Dashboard

3.2. DIGIROOM

O Digiroom é uma sala localizada no segundo piso do bloco H da Faculdade de Engenharia da Universidade do Porto, que está designada por H204, a sala tem aproximadamente 38 m², conta com 4 janelas com aproximadamente 1,3m x 1,3m, numa das janelas foi instalado um atuador que permite abrir e fechar a janela de forma automática. A sala tem uma porta de entrada voltada para o corredor do edifício e está diretamente ligada à sala adjacente através de uma porta, o Digiroom tem várias mesas e cadeiras que são utilizadas para fins educacionais e experimentais.

Nesta sala foi além de um sensor que avalia diversos parâmetros ambientais úteis na avaliação da qualidade do ambiente interior (IEQ), foram também implementados outros equipamentos: impressora 3D, torno, CNC, serra circular e serra de cinta, que para o presente caso de estudo estes equipamentos não foram utilizados.

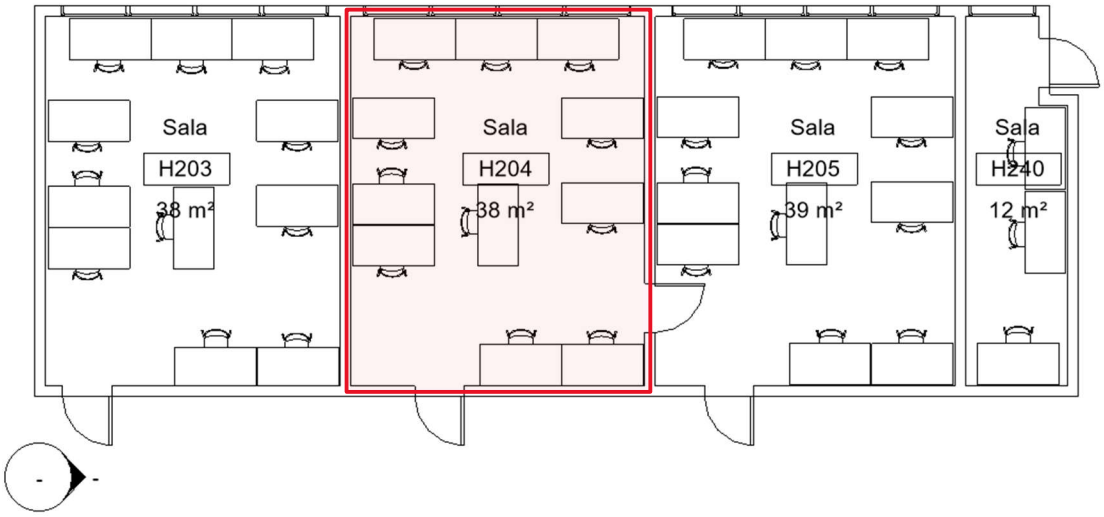


Fig. 8 - Planta do Digiroom

3.3. MODELO 3D

3.3.1. CRIAÇÃO DO MODELO 3D EM REVIT

De forma a enriquecer o Dashboard e para que o utilizador consiga interpretar o local a que se referre foi desenvolvido um modelo do Digiroom em Revit, além desta sala foram também modeladas mais 3 salas próximas do local em estudo.

A criação do modelo em Revit tem uma índole mais visual, deste modo para todos os componentes modelados não foi tido em consideração os materiais efetivamente utilizados na construção da sala. Deste modo as características das paredes, pisos, portas e janelas são genéricas uma vez que não estaria em estudo a relação dos materiais com as propriedades medidas na sala. Contudo foram respeitadas as dimensões de todos os elementos, assim como as áreas e volumes das salas.

A primeira etapa na fase de modelação foi a criação dos elementos base, isto é, paredes e piso, sempre garantindo as distâncias entre paredes o mais semelhante à realidade. Seguidamente foram incorporadas no modelo as portas e as janelas. Estes procedimentos foram realizados através da aba “Arquitetura”, onde é possível aceder a diversas subcategorias, uma das mais utilizadas e fundamental para desenvolver qualquer projeto é a subcategoria “Construir”, com ela é possível fazer, paredes, pisos, portas, janelas, entre outros componentes. Todos estes elementos são editáveis, desta forma o utilizador pode modelar

com as medidas que desejar e atribuir determinadas caracteristicas aos materiais e elementos de forma a poder reproduzir o edifício de acordo com os seus atributos e especifícações Fig. 9.

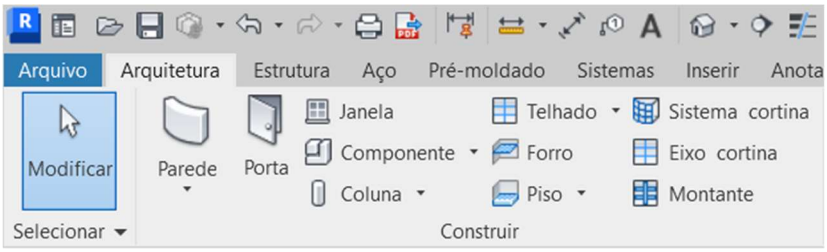


Fig. 9 - Construir – Revit

Finalizada a delimitação dos espaços pelos elementos anteriormente abordados, foi necessário atribuir efetivamente a designação de um espaço, caso contrário o Revit não faria essa distinção. Na aba "Arquitetura", na subcategoria "Ambiente e Área", existe uma funcionalidade que permite definir ambientes, isto é, separar as diversas divisões e/ou espaços que podem existir num projeto, assim, cada compartimento terá um nome, uma área e um volume associado. Isto permite por si só atribuir informações pertinentes sobre os espaços a serem estudados Fig. 10.

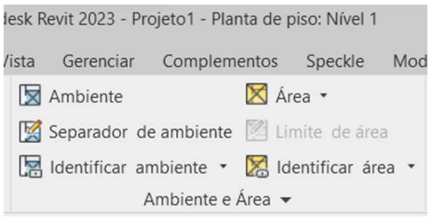


Fig. 10 - Ambiente e Área - Revit

Concluída a modelação dos espaços restava mobilar os cenários, para isso, foram importados alguns objetos simples que permitem a melhor interpretação do espaço, o mobiliário exportado foram cadeiras e mesas e foram colocadas nas salas conforme estão dispostas na realidade. Estes tipos de objetos podem ser incorporados nos projetos de diversas formas, neste caso de estudo foi utilizada a plataforma BIMobject, que permite que os utilizadores procurem, selecionem e transfiram objetos BIM para serem inseridos nos modelos Revit. Na Fig. 11 está apresentada a vista 3D do projeto realizado no Revit. Nesta imagem é visível o mobiliário e a delimitação das salas.

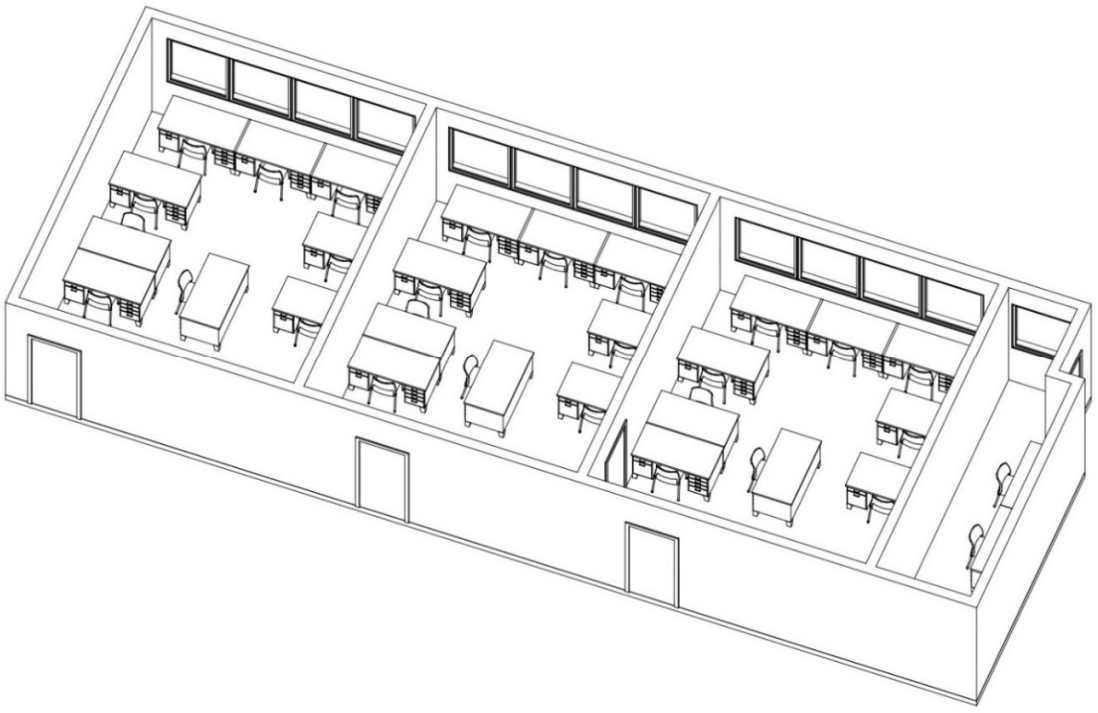


Fig. 11 – Vista 3D do modelo Revit

3.3.2. INSERÇÃO DO MODELO 3D NO POWER BI

Após a criação do modelo Revit foi necessário inseri-lo no relatório criado no Power BI para que este se tornasse visível e interativo, esta transição foi realizada através do 3DBI, um add-on compatível com o Revit já referido anteriormente.

Utilizando o add-on previamente instalado no Revit, foi aberta uma janela que diz respeito à exportação dos dados do modelo, foram selecionados todos os parâmetros e de seguida surgiu uma nova janela com as informações de todos os dados que viriam a ser exportados, Fig. 12.

Desta forma foi criado um ficheiro no formato .json e 3 tabelas com toda a informação necessária para que se tornasse possível a visualização do modelo 3D no Power BI.

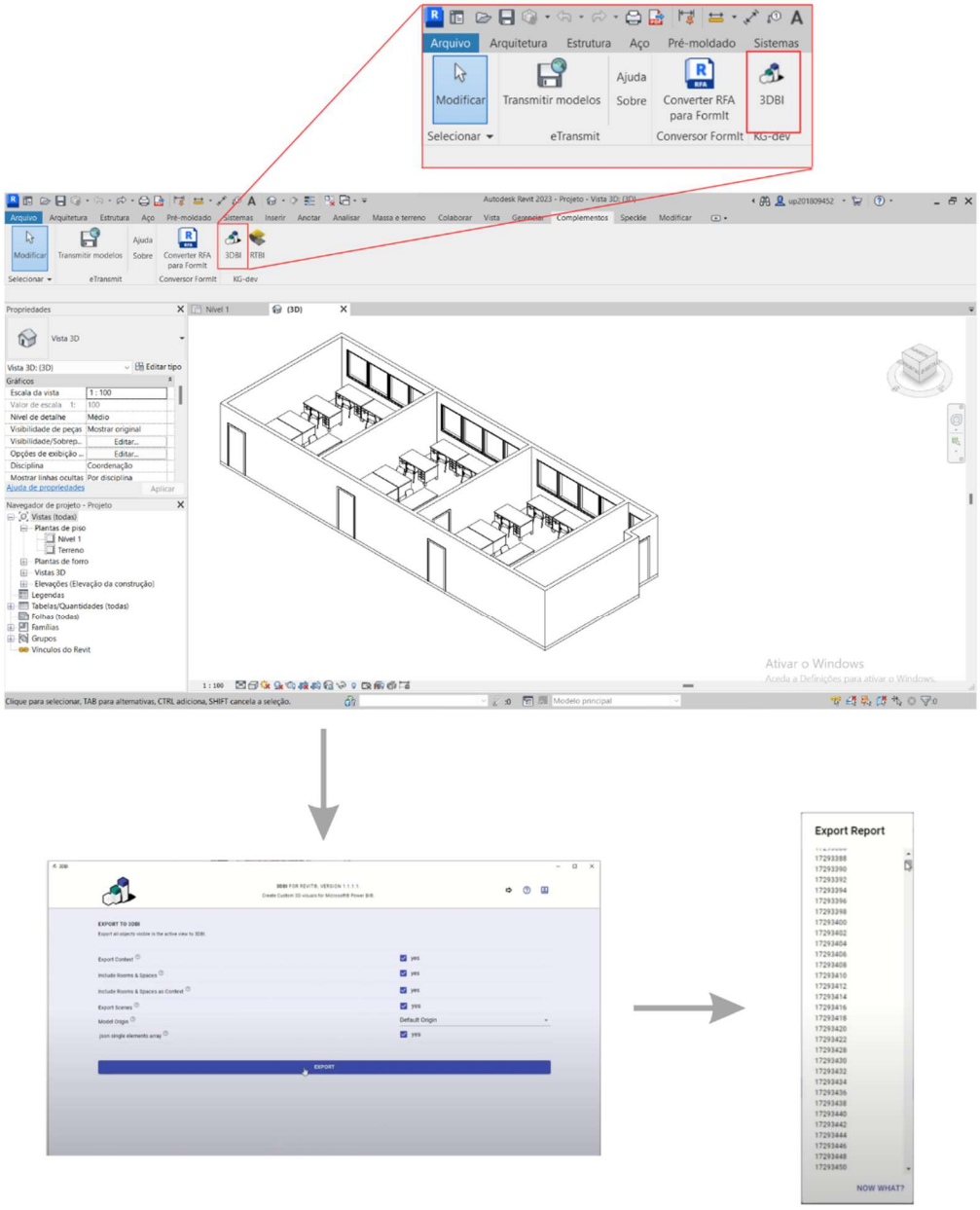


Fig. 12 - 3DBI – Exportação do modelo do Revit

Para inserir o modelo 3D no Power BI através do 3DBI foi necessário que este elemento visual fosse inserido, previamente, no Power BI, visto que por defeito não consta no painel de visualização.

No campo de leitura, foi adicionada unicamente a coluna “id” presente na tabela “elements”, visto que esta coluna continha todos os elementos presentes no modelo. Seguidamente, ao selecionar “INSERT MODEL” na aba destinada ao modelo, foi possível selecionar o modelo criado no Revit.

Desta forma, o modelo estaria pronto para ser observado no Power BI, permitindo que o utilizador selecionasse os vários elementos projetados, ampliasse e reduzisse o modelo e mudasse os ângulos de visualização de forma ajustar o campo de visão.

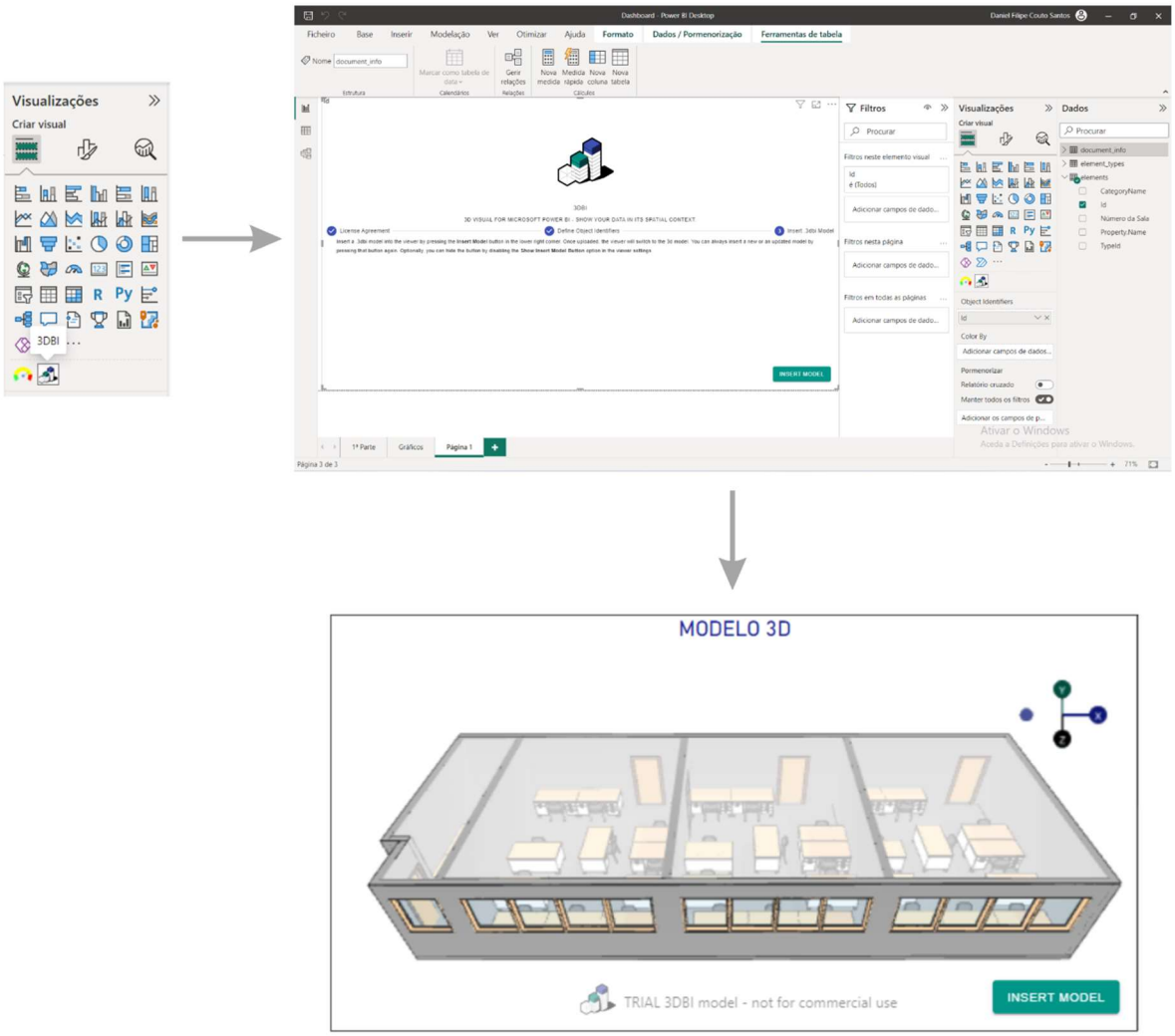


Fig. 13 - 3DBI – Inserção do modelo 3D no Power BI

3.3.3. EXPORTAÇÃO DOS DADOS DO MODELO 3D ATRAVÉS DO DYNAMO

O Dynamo permite a exportação de dados de um modelo Revit e que essa informação seja guardada num ficheiro Excel. Como já foi referido, no modelo Revit foram criadas divisões, cada uma com nome, área e volume associado, e esses mesmos dados foram exportados para uma folha de Excel utilizando o Dynamo.

O script desenvolvido no Dynamo, apresentado na Fig. 14 foi criado usando vários nós disponíveis no software. Previamente foi criada um ficheiro Excel com o nome "Dados_Salas". Seguem os passos realizados para a criação deste script:

- Introdução do nó "ExportExcel";
  - Este nó é o cerne do processo da exportação dos dados;
- Introdução o nó "File Path";
  - Este nó é responsável por procurar o ficheiro Excel onde os dados serão armazenados;
- Fazer a ligação entre o nó "File Path" e o "filePath" presente no nó "ExportExcel";
  - Esta ligação serve para informar onde será guardado e lido o script;

- No nó "File Path" foi aberto o ficheiro Excel previamente criado "Dados_Salas";
    - Serve para o Dynamo reconhecer o local onde será guardado o script;
- Introdução do nó "String" e atribuição do nome da folha "Dados_das_salas";
    - Este nó indica o nome da folha de Excel a utilizar;
- Fazer a ligação entre o nó "String" e o "sheetName" presente no nó "ExportExcel";
    - Fornece a indicação ao Dynamo de qual a folha onde será guardada a informação;
- Introdução do nó "Slider" e atribuição do valor "0";
    - Este nó serve para atribuir uma variável numérica;
- Fazer a ligação do nó "Slider" com "startRow" e "startCol" presente no nó "ExportExcel";
    - Esta ligação indica que os dados exportados serão apresentados a partir da primeira linha e primeira coluna;
- Introdução do nó "Boolean" e escolha da opção "True";
    - Este serve para atribuir "True" ou "False"
- Fazer a ligação do nó "Boolean" com "overWrite" presente no nó "ExportExcel";
    - Serve para indicar ao Dynamo se pode ou não gravar sobre um ficheiro existente, neste caso foi dada permissão;
- Introdução do nó "Categories" e escolha da opção "Ambientes";
    - Este nó permite identificar a categoria dos elementos do Revit;
- Introdução do nó "All Elements of Category";
    - Este nó agrupa todos os elementos de uma determinada categoria;
- Fazer a ligação do nó "Categories" com o nó "All Elements of Category";
    - Esta ligação permite a seleção de todos os "Ambientes" criados no Revit;
- Introdução dos nós "Room Number", "Room Area" e "Room Volume";
    - Estes nós selecionam as informações número, área e volume, respetivamente;
- Fazer a ligação do nó "All Elements of Category" com os nós "Room Number", "Room Area" e "Room Volume";
    - Esta ligação permite que os nós acedam às informações;
- Introdução do nó "List Create" e adição de 2 itens, de modo a ficar com 3;
    - Este nó permite criar listas com os dados fornecidos;
- Fazer a ligação dos nós "Room Number", "Room Area" e "Room Volume" com os itens do nó "List Create";
    - Esta ligação cria 3 listas com as propriedades que foram consultadas;
- Introdução do nó "List Transpose";
    - Este nó troca linhas por colunas;
- Fazer a ligação do nó "List Create" com o nó "List Transpose";
    - Esta ligação faz inverter as linhas pelas colunas ficando cada lista com 3 propriedades (número, área e volume), cada lista corresponde a uma sala;
- Introdução de um novo nó "List Create" e adição de 2 itens, de modo a ficar com 3;
    - Este nó permite criar listas com os dados fornecidos;
- Introdução de 3 novos nós "String" e atribuição dos nomes "Número da Sala", "Área da Sala (m2)" e "Volume da Sala (m3)";
    - Estes string serão utilizados para definir os títulos de cada coluna;
- Fazer a ligação dos nós "Número da Sala", "Área da Sala (m2)" e "Volume da Sala (m3)" com os itens do nó "List Create";
    - Esta ligação cria 3 listas com as propriedades definidas;
- Introdução do nó "List Insert";
    - Este nó permite a adição de novas listas;

- Fazer ligação do nó “List Transpose” e o “list” presente no nó “List Insert”;
    - Esta ligação indica quais os dados que serão apresentados na lista;
- Fazer a ligação do nó “List Create” e o “element” presente no nó “List Insert”;
    - Esta ligação é responsável pela atribuição dos títulos às listas
- Introdução do nó “Slider” e atribuição do valor “0”;
    - Este nó serve para atribuir uma variável numérica;
- Fazer a ligação do nó “Slider” e o “index” presente no nó “List Insert”;
    - Serve para dar a indicação de onde irá iniciar a lista;
- Fazer a ligação do nó “List Insert” e o “data” presente no nó “ExportExcel”.
    - Serve para enviar os dados para o Excel.

Após a execução do script descrito ficam automaticamente guardados os dados na folha Excel conforme a Fig. 15, este processo é alterado de forma automática caso sejam alterações no modelo Revit, automatizando assim o processo de introdução deste tipo de dados na folha Excel.

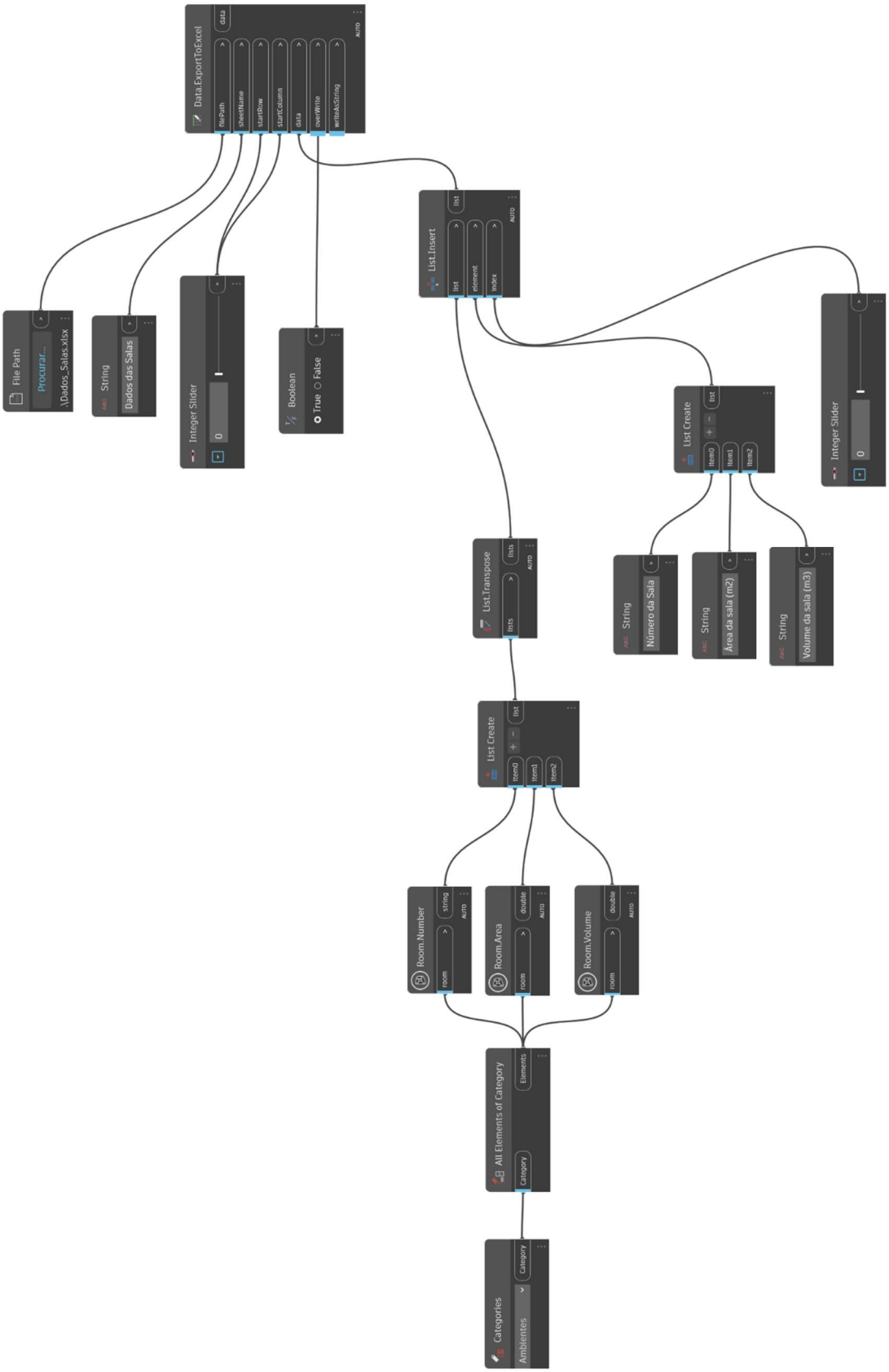


Fig. 14 - Script em Dynamo para extração dos dados das salas

	A	B	C
1	Número da Sala	Área da sala (m2)	Volume da sala (m3)
2	H203	38.0844	114.2532
3	H204	37.9652	113.8956
4	H205	38.9188	116.7564
5	H240	12.1944	36.5832

Fig. 15 - Folha Excel com os dados extraídos do Revit

3.4. SISTEMA DE MONITORIZAÇÃO

A recolha dos parâmetros medidos no Digiroom foi feita através de um sistema de monitorização, composto por 10 sensores, medindo 12 parâmetros de IEQ, o sistema de monitorização envia dados para uma Raspberry, esta funciona como um servidor, armazenando todos os dados numa base de dados que estão ligados a uma rede privada (EP_base) que se destina ao uso exclusivo do Digiroom. O Maria DB faz a ponte entre a Raspberry e o Power BI, as informações que estão armazenadas na base de dados chegam ao Power BI através desta ligação. O Mosquitto obtém os dados da Raspberry e o Python faz a ligação ao Mosquitto para ler esses mesmos dados, permitindo executar ações no atuador de janela. Todos os sensores recolhem os dados a cada 30 segundos, com exceção do medidor de concentração de partículas, que recolhe dados a cada 60 segundos, Tab. 2.

Tab. 2 – Sensores utilizados

Nome do sensor	Parâmetro medido	Intervalo de medição
SHT85	Temperatura (ºC)	30 segundos
BME680	Humidade relativa (%)	30 segundos
SLS-BTA	Nível de pressão sonora (dB)	30 segundos
SFA30	Concentração de HCHO (ppb)	30 segundos
K30	Concentração de CO_2 (ppm)	30 segundos
Wind	Velocidade do vento (m/s)	30 segundos
TSL2591_V	Fluxo luminoso vertical (lux)	30 segundos
TSL2591_H	Fluxo luminoso horizontal (lux)	30 segundos
PMS1003	Concentração de partículas inferiores a 1µm, PM_1 (µg/m³)	60 segundos
	Concentração de partículas inferiores a 2.5 µm, $PM_{2,5}$ (µg/m³)	60 segundos
	Concentração de partículas inferiores a 10 µm, PM_{10} (µg/m³)	60 segundos
PIR	Movimento (valor binário)	30 segundos

Para avaliar cada um dos parâmetros medidos foi feita uma pesquisa para obter os valores mínimos e/ou máximos recomendados e/ou regulamentares. Que foram posteriormente utilizados como referências no caso de estudo deste trabalho.

Tendo em consideração o Decreto-Lei nº 243/86 de 20 de agosto, que estipula que em ambientes de trabalho, assim como as instalações comuns, devem ser proporcionadas boas condições de temperatura e humidade relativa de modo a não comprometer o bem-estar e saúde dos ocupantes, acrescentando ainda que o intervalo de valores aceitáveis para a temperatura deva oscilar entre 18°C e 22°C, podendo, contudo, atingir os 25°C [41]. A mesma legislação, para as mesmas condições de utilização, estabelece um intervalo de valores para a humidade relativa considerando que esta deve permanecer no intervalo de 50% a 70% [41].

Conforme estabelecido pelo Regulamento dos Requisitos Acústicos dos Edifícios, sendo o Digiroom uma sala de aula destinada a fins de investigação, deve ser mantido o nível de pressão sonora abaixo dos 60 dB [42].

A regulamentação relativa à qualidade do ar interior nos edifícios é estabelecida pela Portaria n.º 138-G/2021. Conforme a legislação, é determinado que a concentração de partículas com diâmetro inferior a 10 μm não deve exceder os 50 μg/m3, enquanto a concentração de partículas inferiores a 2.5 μm não deve ultrapassar os 25 μg/m3. No entanto, é importante mencionar que não há regulamentação específica para partículas com diâmetro inferior a 1 μm, sendo que, foi assumido que a sua concentração também deve ser limitada a 25 μg/m3. Além disso, a portaria estabelece que a concentração de dióxido de carbono (CO_2) não deve exceder 1250 ppm (partes por milhão) [43].

De acordo com a legislação portuguesa, é estabelecido um valor máximo para os compostos orgânicos voláteis (COV), porém não é especificado um limite para o formaldeído (HCHO) [43]. Num estudo realizado nos Estados Unidos, recomenda-se um limite de exposição ao formaldeído de 100 ppb como um nível adequado para ambientes internos [44].

Outro parâmetro que a legislação portuguesa não aborda é a velocidade do vento no interior dos edifícios, de acordo com a norma Americana ASHRAE 55, que aborda as condições térmicas e de qualidade do ar interior, não é especificado um valor máximo para a velocidade do vento no interior dos edifícios, contudo recomenda-se que não ultrapasse os 0.2m/s [23].

O fluxo luminoso no interior dos edifícios também não é regulamentar, contudo é aconselhado a que, em ambientes de ensino, esteja compreendido entre 250 lux e 500 lux. Os sensores do Digiroom decompõe o fluxo em 2 componentes, horizontal e vertical, para isso, foi admitido um ângulo de incidência solar de 80º com a horizontal e feita a decomposição dos valores, deste modo o fluxo luminoso horizontal deve estar compreendido (aproximadamente) entre 30 lux e 90 lux e o fluxo luminoso vertical entre 250 lux e 500 lux [24].

O sensor de movimento contabiliza as vezes em que é detetado movimento no interior da sala. Este parâmetro não está sujeito a limites estabelecidos por normas ou regulamentos, visto que sua função é apenas fornecer informações sobre a presença ou ausência de movimento.

3.4.1. Inserção dos dados recolhidos pelos sensores no Power BI

O primeiro procedimento necessário foi garantir que a ligação estava estabelecida, visto que o acesso ao sistema de monitorização era realizado pela rede "EP_base", este processo foi bastante simples, a sua conexão foi realizada de forma similar à conexão a uma rede wi-fi, introduzindo a chave de acesso fornecida pelos responsáveis pelo Digiroom, Fig. 16.

Feita a ligação à rede, foi necessário instalar o Maria DB através de um conector admitido pelo Power BI, desta forma a exportação era possível utilizando novamente a chave de acesso. Após a ligação estabelecida foram selecionadas 4 tabelas, que eram as únicas que continham informações relevantes para a leitura dos dados.

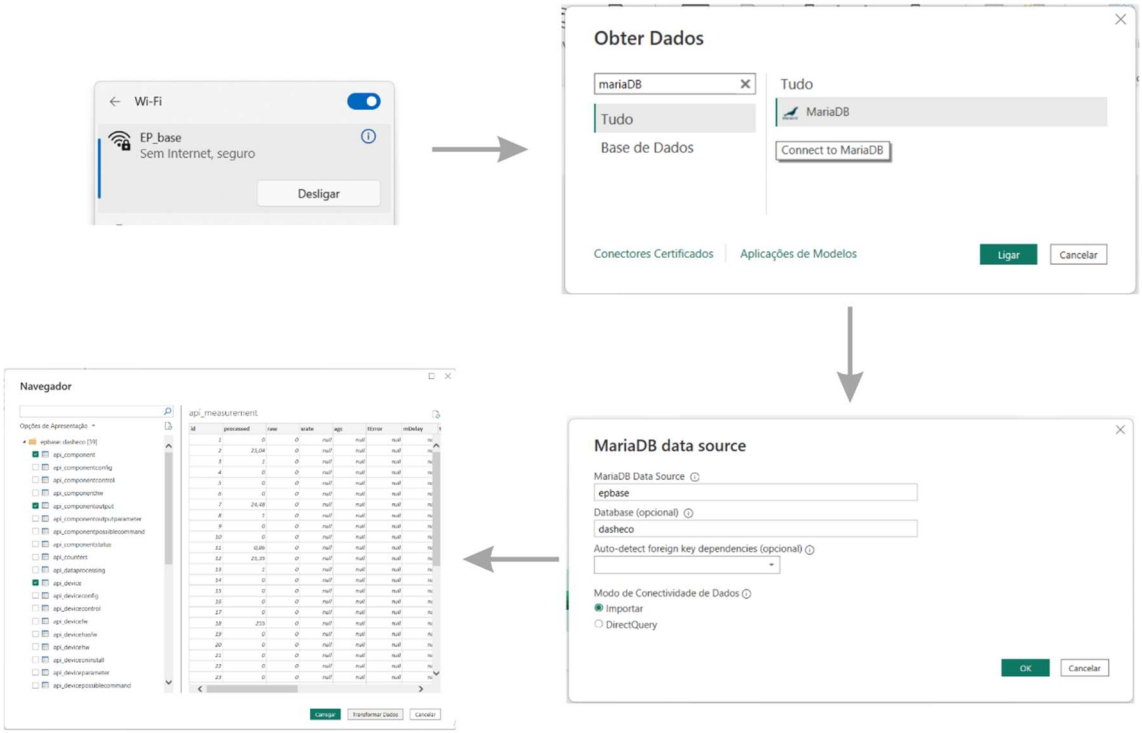


Fig. 16 - Ligação do sistema de monitorização ao Power BI

3.5. Dados no Power BI

3.5.1. Organização dos dados

Os valores medidos pelo sensor para avaliação do IEQ não estavam devidamente organizados por colunas numa única tabela, foi necessário exportar 4 tabelas para conseguir interpretar os resultados. As tabelas exportadas foram as seguintes: "api_device", "api_component", "api_componentoutput" e "api_measurement". Estas tabelas estão interligadas, sendo necessário procurar e filtrar certos valores em cada tabela, a Fig. 17 demonstra de forma simplista, que posteriormente será detalhado, o conjunto de procedimentos necessários para a obtenção dos dados recolhidos pelo sensor.

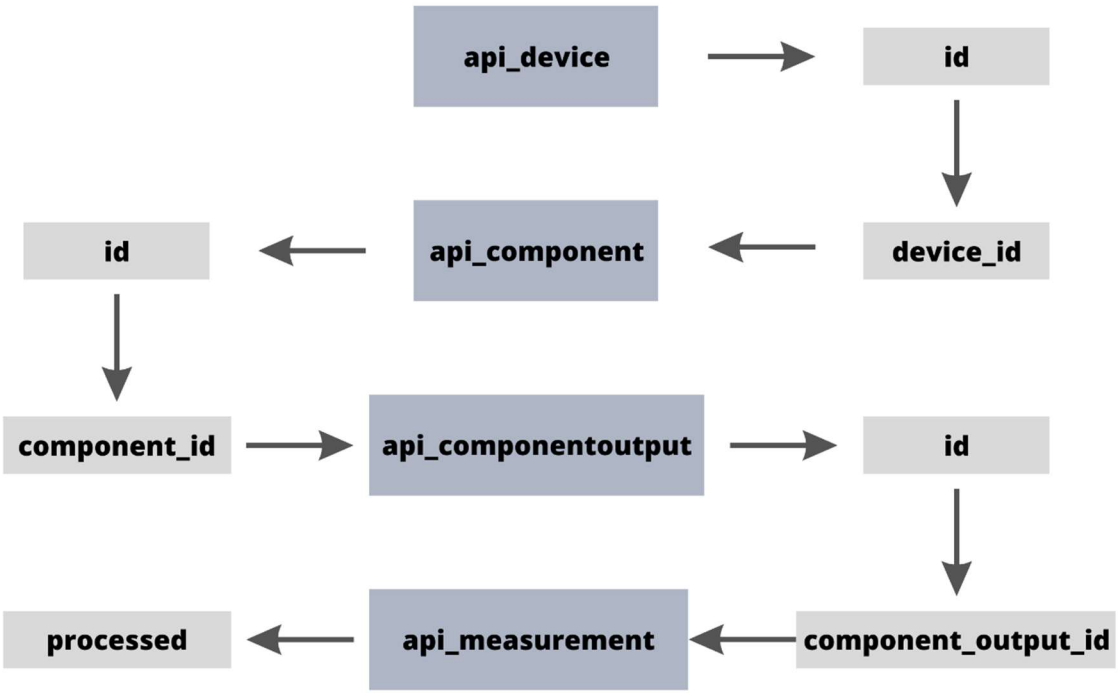


Fig. 17 - Organização das tabelas

A tabela “api_device” dava informação sobre o tipo de dispositivo que queríamos monitorizar, visto que os dados vindos desta tabela traziam informações sobre todos os dispositivos da sala. Na Fig. 18 é possível verificar a existência dos vários dispositivos, o sistema de monitorização que mede os parâmetros da sala está designado como “IEQ” e encontra-se na primeira linha da tabela, daqui retiramos o número que o identifica (“id”=1) na tabela seguinte “api_component”.

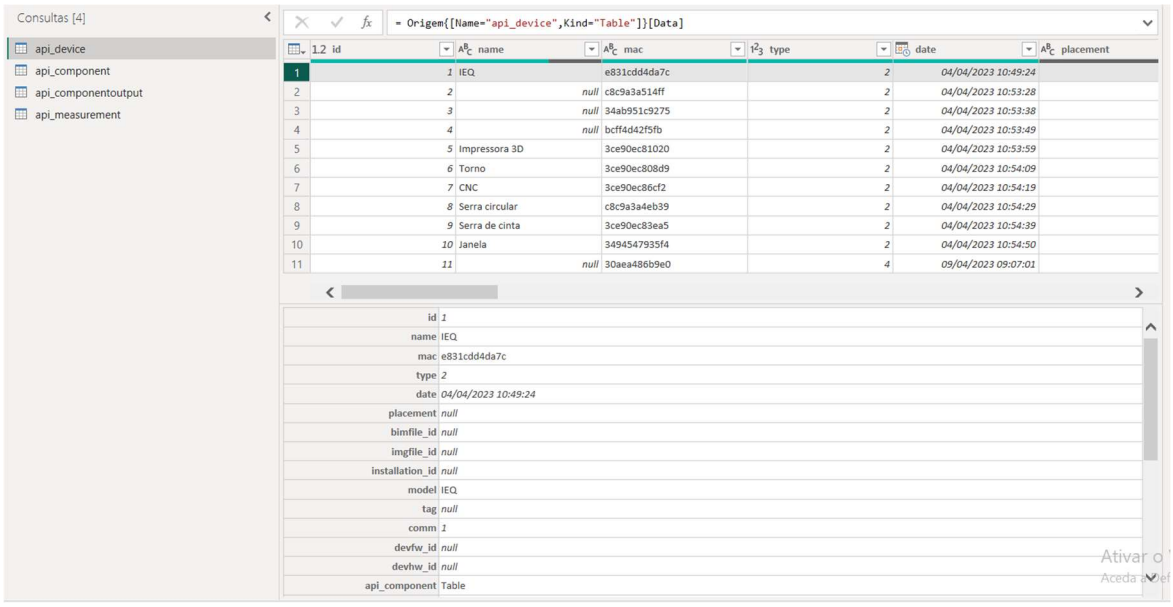


Fig. 18 - Tabela "api_device"

Após detetado o dispositivo pretendido e identificado pelo seu "id", na tabela "api_component" existe uma coluna denominada "device_id" que foi filtrada de modo a apresentar os resultados unicamente com "device_id" =1, para que desta forma fossem encontrados os parâmetros medidos pelo dispositivo "IEQ". Sabendo que o sensor que faz a leitura da concentração de CO_2 é o componente denominado "K30", ficamos a saber que o "id" destinado à medição da concentração de CO_2 no Digiroom é o "id" =8, de forma análoga ao que foi descrito anteriormente este "id" =8 serve de identificador na tabela seguinte "api_componentoutput" Fig. 19.

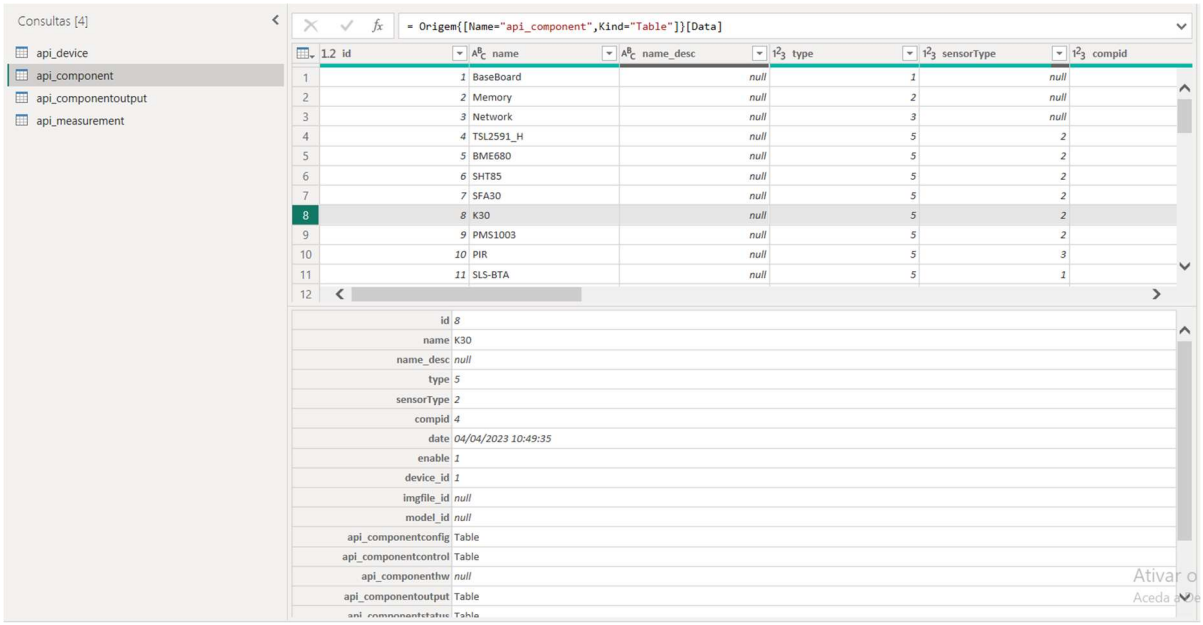


Fig. 19 - Tabela "api_component"

Seguindo o raciocínio, procurando na coluna "component_id" pelo número 8, que, como visto anteriormente era o "id" que identificava a concentração de CO_2, obtem-se um novo "id"= 12 que irá guiar-nos à ultima tabela onde serão apresentados os valores lidos pelo sensor, Fig. 20.

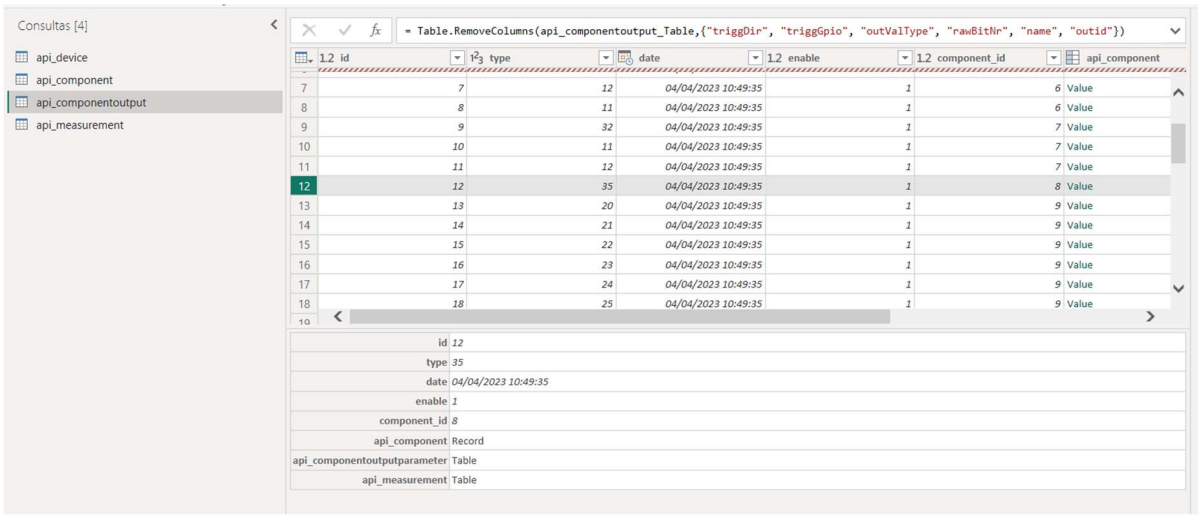
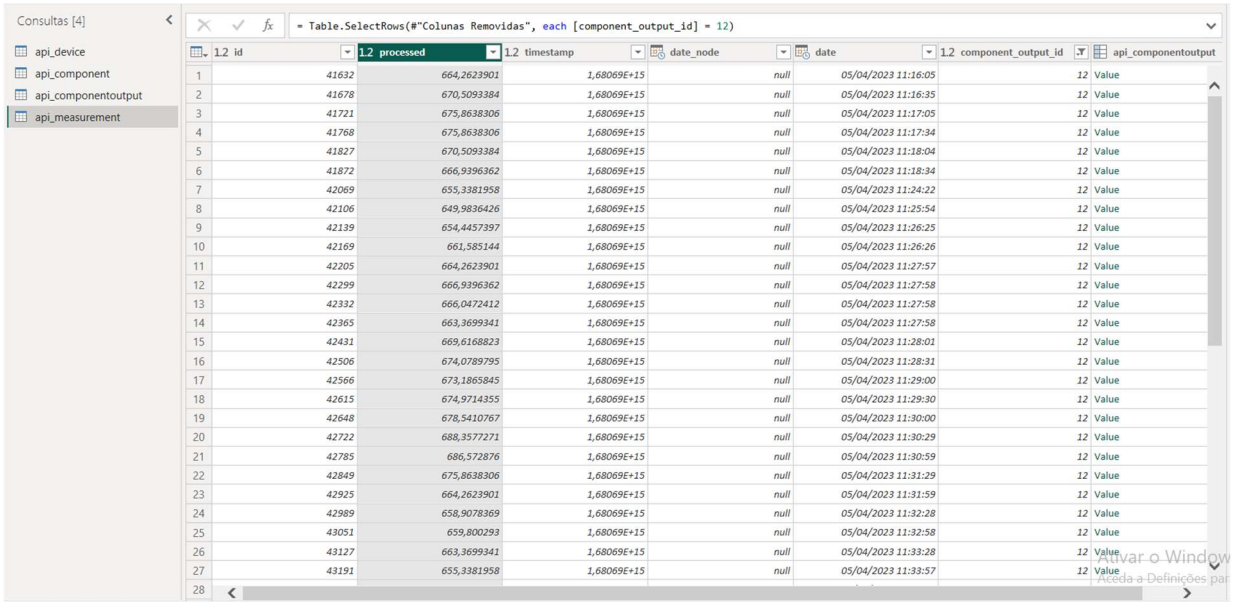


Fig. 20 - Tabela "api_componentoutput"

Finalmente, na tabela “api_measurement” são apresentados os valores na coluna “processed”. Contudo esta coluna exibe todos os valores de todos os parâmetros, e é então necessário filtrar a coluna “component_output_id” de forma que o valor apresentado seja o desejado. Como descrito anteriormente, para obter os valores da concentração de CO_2 é necessário procurar na coluna “component_output_id” pelo valor 12, identificado na tabela anterior, “api_componentoutput” Fig. 21.



	1.2 id	1.2 processed	1.2 timestamp	date_node	date	1.2 component_output_id	api_componentoutput
1	41632	664,2623901	1,68069E+15	null	05/04/2023 11:16:05	12	Value
2	41678	670,5093384	1,68069E+15	null	05/04/2023 11:16:35	12	Value
3	41721	675,8638306	1,68069E+15	null	05/04/2023 11:17:05	12	Value
4	41768	675,8638306	1,68069E+15	null	05/04/2023 11:17:34	12	Value
5	41827	670,5093384	1,68069E+15	null	05/04/2023 11:18:04	12	Value
6	41872	666,9396362	1,68069E+15	null	05/04/2023 11:18:34	12	Value
7	42069	655,3381958	1,68069E+15	null	05/04/2023 11:24:22	12	Value
8	42106	649,9836426	1,68069E+15	null	05/04/2023 11:25:54	12	Value
9	42139	654,4457397	1,68069E+15	null	05/04/2023 11:26:25	12	Value
10	42169	661,585144	1,68069E+15	null	05/04/2023 11:26:26	12	Value
11	42205	664,2623901	1,68069E+15	null	05/04/2023 11:27:57	12	Value
12	42299	666,9396362	1,68069E+15	null	05/04/2023 11:27:58	12	Value
13	42332	666,0472412	1,68069E+15	null	05/04/2023 11:27:58	12	Value
14	42365	663,3699341	1,68069E+15	null	05/04/2023 11:27:58	12	Value
15	42431	669,6168823	1,68069E+15	null	05/04/2023 11:28:01	12	Value
16	42506	674,0789795	1,68069E+15	null	05/04/2023 11:28:31	12	Value
17	42566	673,1865845	1,68069E+15	null	05/04/2023 11:29:00	12	Value
18	42615	674,9714355	1,68069E+15	null	05/04/2023 11:29:30	12	Value
19	42648	678,5410767	1,68069E+15	null	05/04/2023 11:30:00	12	Value
20	42722	688,3577271	1,68069E+15	null	05/04/2023 11:30:29	12	Value
21	42785	686,572876	1,68069E+15	null	05/04/2023 11:30:59	12	Value
22	42849	675,8638306	1,68069E+15	null	05/04/2023 11:31:29	12	Value
23	42925	664,2623901	1,68069E+15	null	05/04/2023 11:31:59	12	Value
24	42989	658,9078369	1,68069E+15	null	05/04/2023 11:32:28	12	Value
25	43051	659,800293	1,68069E+15	null	05/04/2023 11:32:58	12	Value
26	43127	663,3699341	1,68069E+15	null	05/04/2023 11:33:28	12	Value
27	43191	655,3381958	1,68069E+15	null	05/04/2023 11:33:57	12	Value
28							

Ativar o Windows
Aceda a Definições par

Fig. 21 - Tabela "api_measurement"

Desta forma, é possivel aceder a todos os valores obtidos pelo sensor para um determinado parâmetro, neste caso a concentração de CO_2, mas para que estes valores possam ser interpretados de forma gráfica é necessário criar uma coluna exclusiva para os valores de cada parâmetro.

O Power BI possui uma ferramenta bastante útil que permite a criação de colunas condicionais, ou seja, é possível criar colunas em que são impostas determinadas condições de forma a obter unicamente os valores desejados. Como referido anteriormente, na tabela “api_measurement”, todos os valores da coluna “component_output_id” que possuam o valor 12 são referentes à concentração de CO_2, desta forma, foi criada uma coluna condicional, com o nome “Concentração de CO2”, em que foi imposta a condição de apresentar unicamente os valores da coluna “processed” quando os valores da coluna “component_output_id” tomassem o valor 12, foi assim criada uma coluna que apresenta unicamente esses valores, conforme apresentado na Fig. 22.

O mesmo processo foi realizado para os restantes 11 parâmetros medidos pelos sensores, de forma a organizar todos os dados do sistema de monitorização.

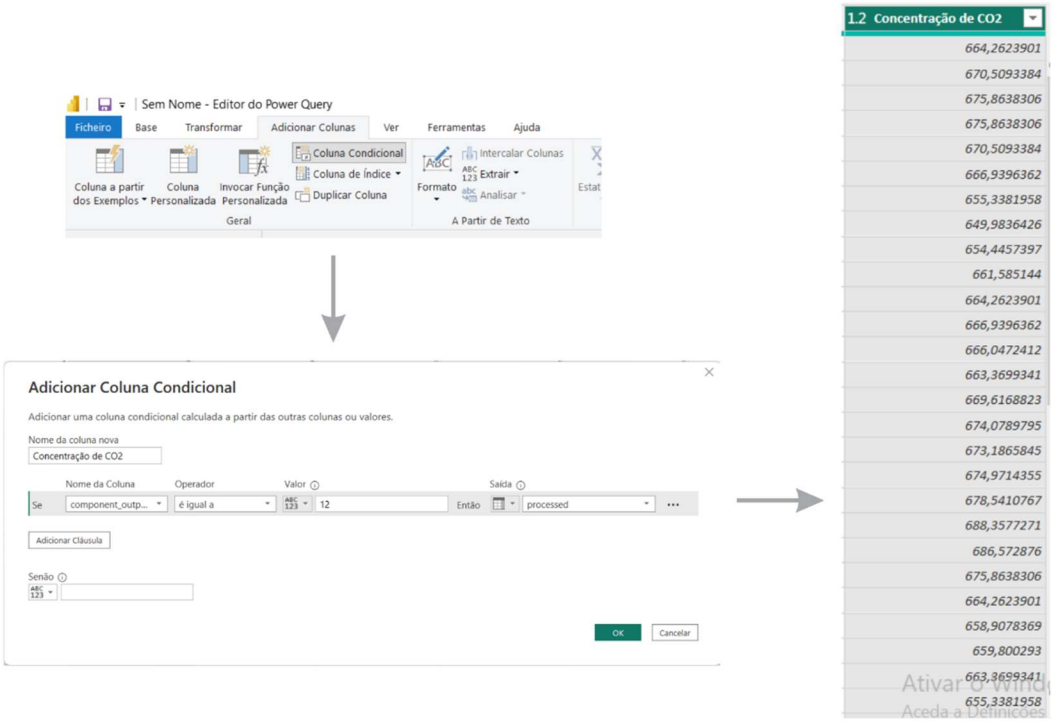


Fig. 22 - Power BI - Coluna condicional

3.5.2. CRIAÇÃO DE INDICADORES DE DADOS EM TEMPO REAL

Feita a organização dos dados recebidos, foi necessário trabalhá-los de forma a surgirem para o utilizador de forma mais apelativa. Foram criados os indicadores de dados em tempo real, que são expressos num bloco que contém a informação relativa ao parâmetro medido e a respetiva unidade, a Fig. 23 apresenta o indicador em tempo real para o parâmetro "Concentração de CO_2".



Fig. 23 - Indicador em tempo real – Concentração de CO_2

A criação deste tipo de blocos exige a utilização de uma ferramenta de visualização presente no Power BI, denominada "cartão". Esta forma de visualização permite que seja apresentado um valor único num formato destacado. De forma a simplificar o processo, serão apresentadas as etapas de criação unicamente para o parâmetro "Concentração de CO_2", uma vez que para os restantes parâmetros o processo é semelhante, Fig. 24.

Após selecionar o elemento de visualização "cartão" foi necessário atribuir o valor a ser apresentado. Sabendo que o valor em tempo real desejado é o último valor lido pelo sensor para o parâmetro "Concentração de CO_2", foi selecionada a coluna "CO_2" como se pode ver na Fig. 24 à direita. Por defeito o valor que surge é o somatório de toda a coluna, para chegar ao valor desejado, foi criado um filtro utilizando a coluna que faz o registo da data e hora em que os dados são recolhidos para que o valor apresentado fosse o último valor enviado pelo sensor na coluna "CO_2", isto é, o mais recente.

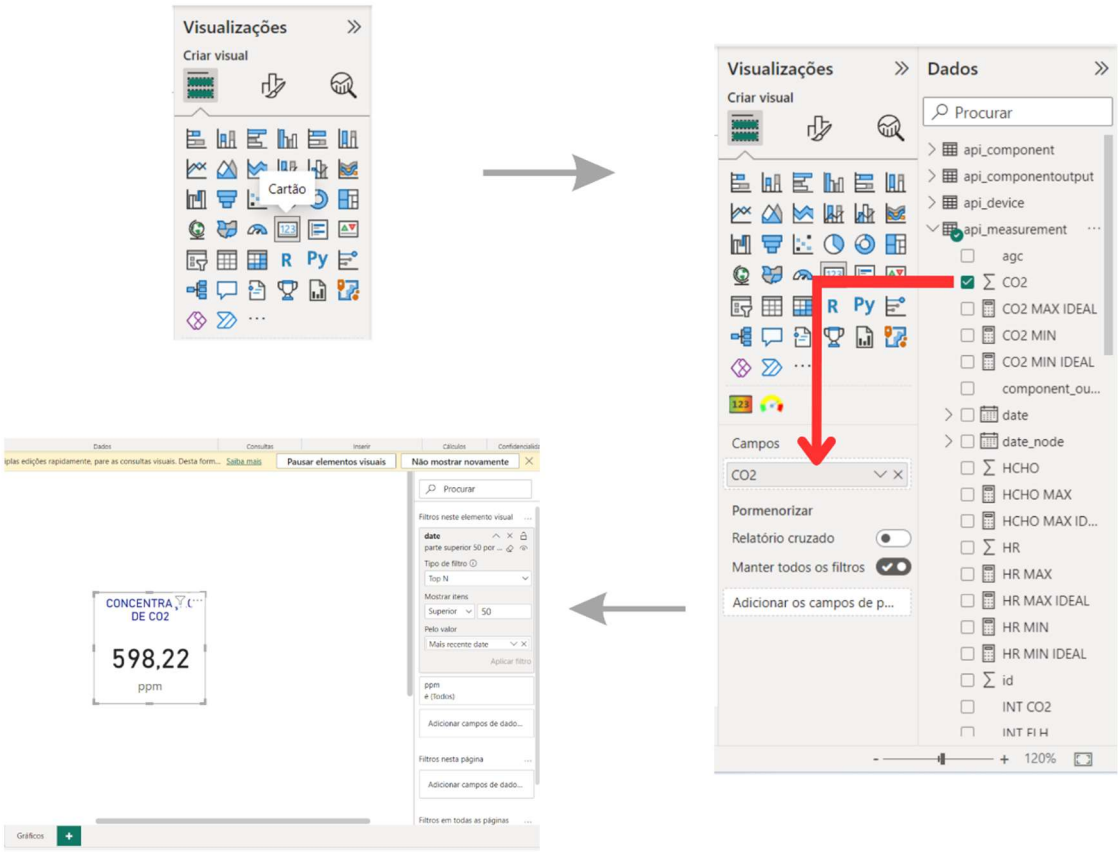


Fig. 24 - Criação de indicador em tempo real

3.5.3. Dados Históricos

A criação de elementos visuais com registos históricos foi importante para que o utilizador pudesse aceder aos dados históricos do Digiroom, foram criados indicadores e gráficos para facilitar essa análise, de forma a permitir a filtragem por data foi criada uma "Segmentação de Dados". Esta ferramenta disponível no Power BI é bastante simples de utilizar e garante um ajuste dos indicadores e dos gráficos perante o período selecionado Fig. 25.



Fig. 25 – Filtragem temporal

Este filtro foi criado com a utilização da ferramenta "Segmentação de dados" presente no Power BI, foi introduzida a coluna referente às datas e criada uma hierarquia com ano, mês e dia. Desta forma o utilizador pode optar por escolher o dia, mês ou ano que pretende analisar e assim todos os gráficos que serão apresentados no capítulo 4, apresentarão os dados relativos ao intervalo escolhido, Fig. 26.

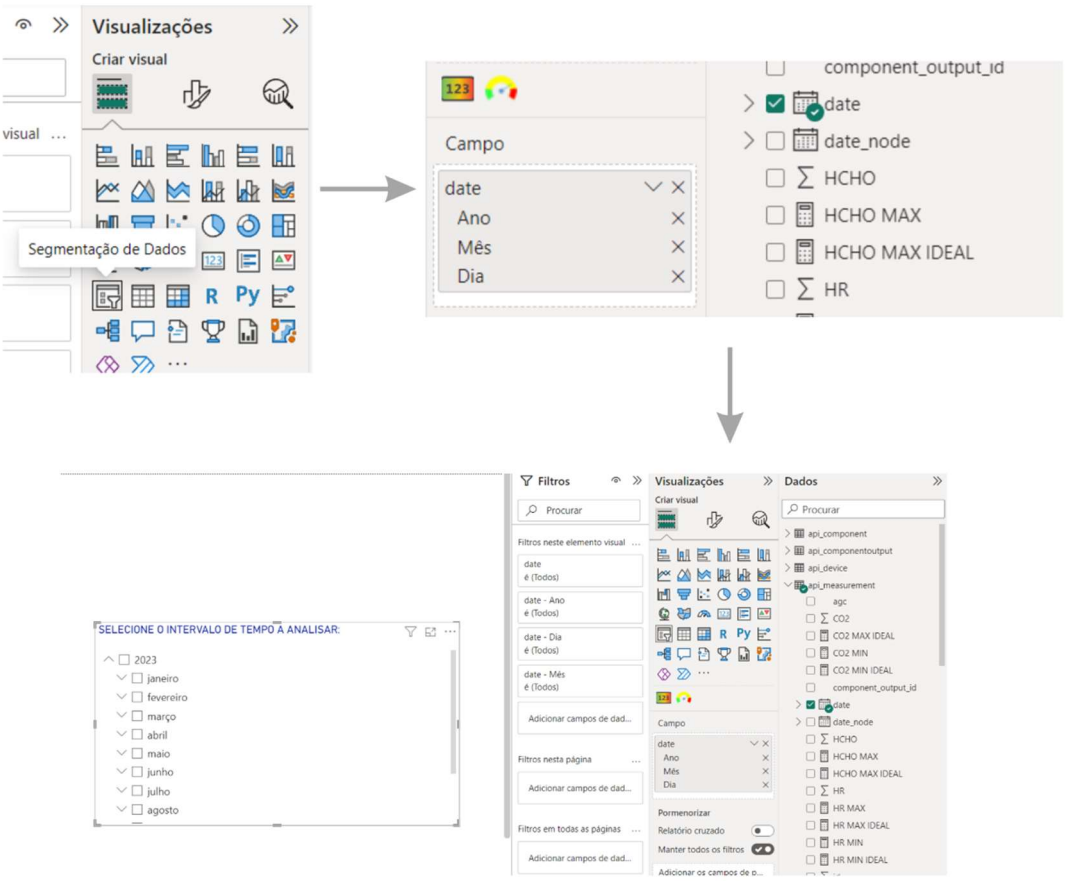


Fig. 26 - Processo de criação do filtro

3.5.3.1. Criação de indicadores de valor médio

A utilização de indicadores visuais para a análise qualitativa dos parâmetros monitorizados é bastante interessante para o utilizador. Este modelo de transmissão de informação é bastante intuitivo e de fácil interpretação. Na Fig. 27 está representado o indicador referente à concentração de CO_2 expressa em parte por milhão (ppm). A cor vermelha é indicativa dos valores inadequados, para o caso, se a concentração de CO_2 for inferior a 1250 ppm o utilizador, através da apresentação das cores, sabe que os níveis no interior da sala são inapropriados. De outro modo, a cor verde apresenta o intervalo de valores apropriados para cada propriedade, neste caso, o valor que é apresentado (402,63 ppm) é um valor adequado para a sala em estudo e por isso o ponteiro surge na cor verde.

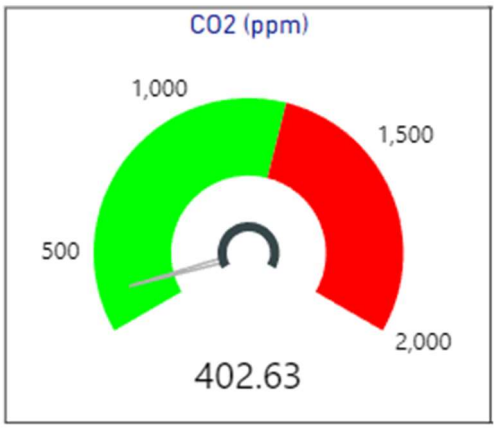


Fig. 27 – Indicador – Concentração de CO_2

O desenvolvimento destes indicadores foi possível através da obtenção do elemento visual "Tachometer", que por defeito não está inserido no painel de visualização do Power BI.

Foram criadas as medidas necessárias para que o indicador apresentasse os valores pretendidos, a concentração mínima de CO_2 medida atualmente em qualquer parte do mundo está acima dos 200 ppm, deste modo foi criada a medida "CO2 MIN" cujo valor atribuído foi 200. Da mesma forma, foi criada as medida sobre a qual estão inseridos os valores da concentração de CO_2 ideais. Foi considerado como valor máximo aceitável 1250 ppm, atribuindo o valor à medida "CO2 MAX IDEAL".

De seguida foram adicionados aos campos desejados as respetivas variáveis, a variável "Value" é a responsável pelo valor a ser medido, então, foi adicionada a coluna que apresenta os valores da concentração de CO_2, com a instrução de que o valor a ser apresentado seria a média consoante o intervalo de tempo selecionado.

Os restantes campos serviram unicamente para fazer o balizamento dos valores a serem apresentados pelo indicador, aos quais foram atribuídas as medidas referidas anteriormente. Desta forma, o indicador estaria finalizado e funcional, contudo foram feitos alguns ajustes estéticos para que surja de forma mais apelativa, alterando o tamanho do título, as cores a serem apresentadas e contornos, Fig. 28.

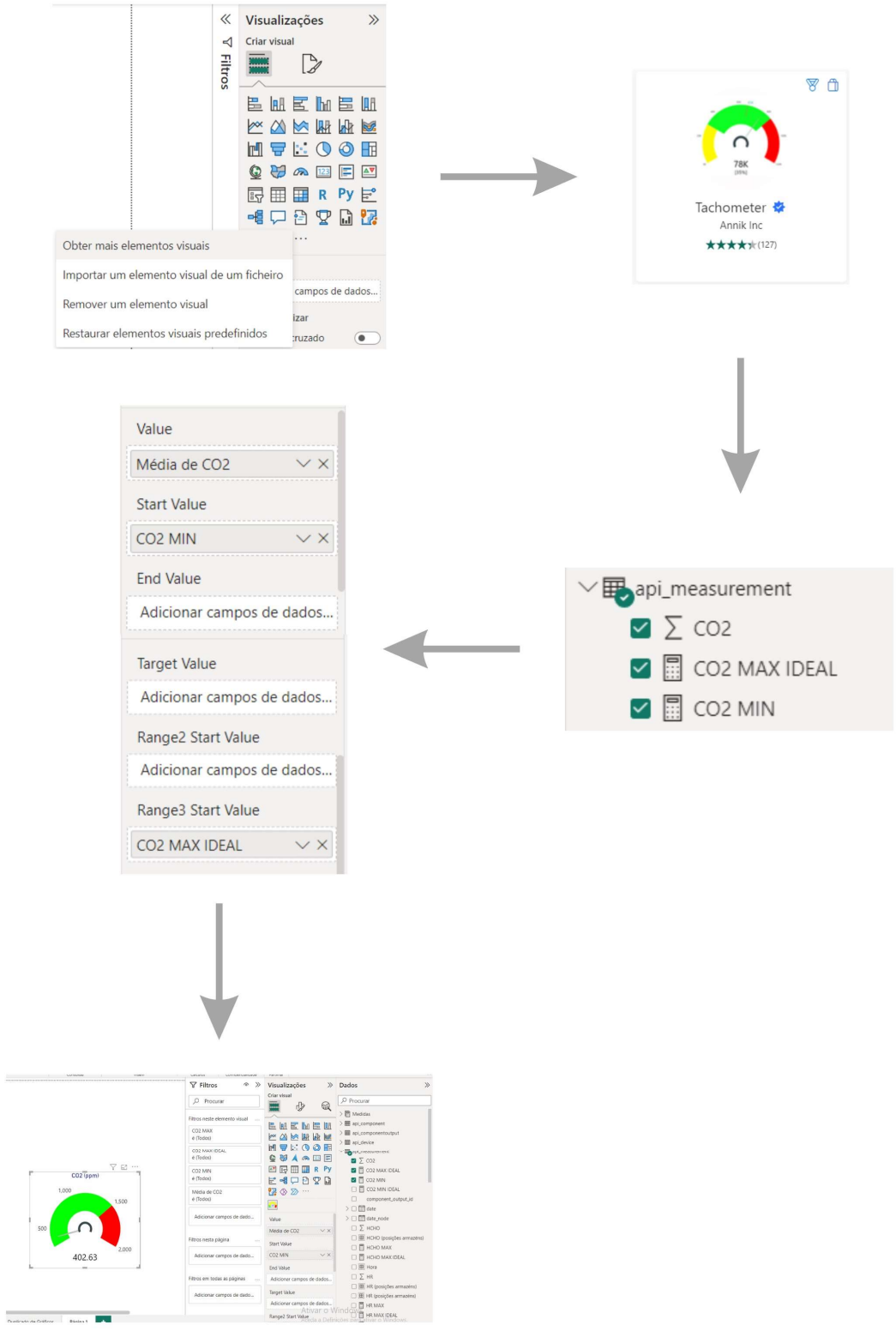


Fig. 28 - Processo de criação dos indicadores

3.5.3.2. Criação de gráficos de linhas

Os gráficos de linhas são uma ferramenta valiosa para ajudar a entender como um determinado parâmetro varia ao longo do tempo e para tomar decisões informadas com base nesses registos históricos. Na Fig. 29 é visível o gráfico de linhas correspondente à concentração de CO_2, para o período considerado. Este tipo de gráfico permite ao utilizador visualizar o valor do parâmetro pretendido no instante desejado passando com o cursor sobre o gráfico, conforme se pode comprovar na Fig. 29, no dia 14/04/2023 às 01:26:04h a concentração de CO_2 no Digiroom era 471,50 ppm.

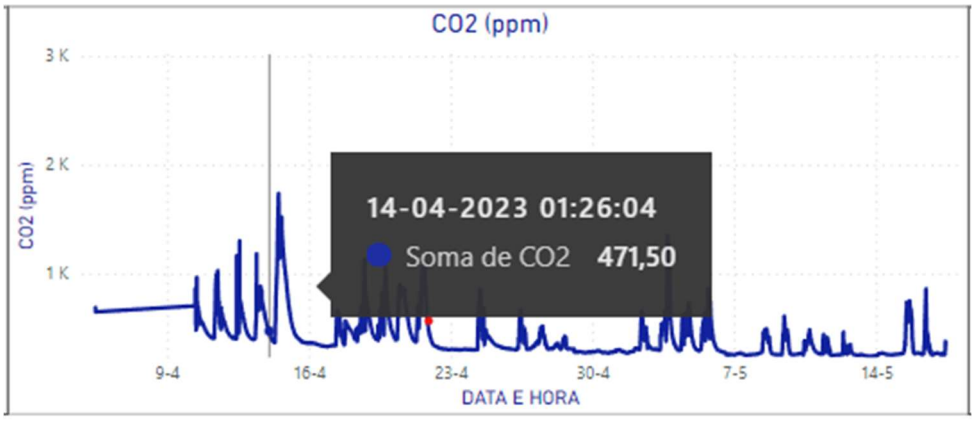


Fig. 29 - Gráfico de linhas - Concentração de CO_2

A criação de gráficos de linhas é uma ferramenta que já está incorporada nos modelos de visualização do Power BI, este elemento gráfico é bastante acessível, necessita apenas da atribuição de um conjunto de valores para os eixos x e y.

Com o objetivo de transmitir um gráfico temporal foi atribuído ao eixo x a coluna relativa ao tempo "date", desta forma o eixo x apresenta a data e hora em que o parâmetro foi medido.

Para a avaliação de todos os parâmetros medidos foram criados diversos gráficos de linhas, todos com o mesmo procedimento, analisando o exemplo da concentração de CO_2, foi atribuído ao eixo y os valores contidos na coluna "CO2". Desta forma, o gráfico de linhas estava criado, de modo a manter um padrão nas cores dos títulos e eixos, foi feito o ajuste visual de acordo com os elementos gráficos já referidos,

Fig. 30.

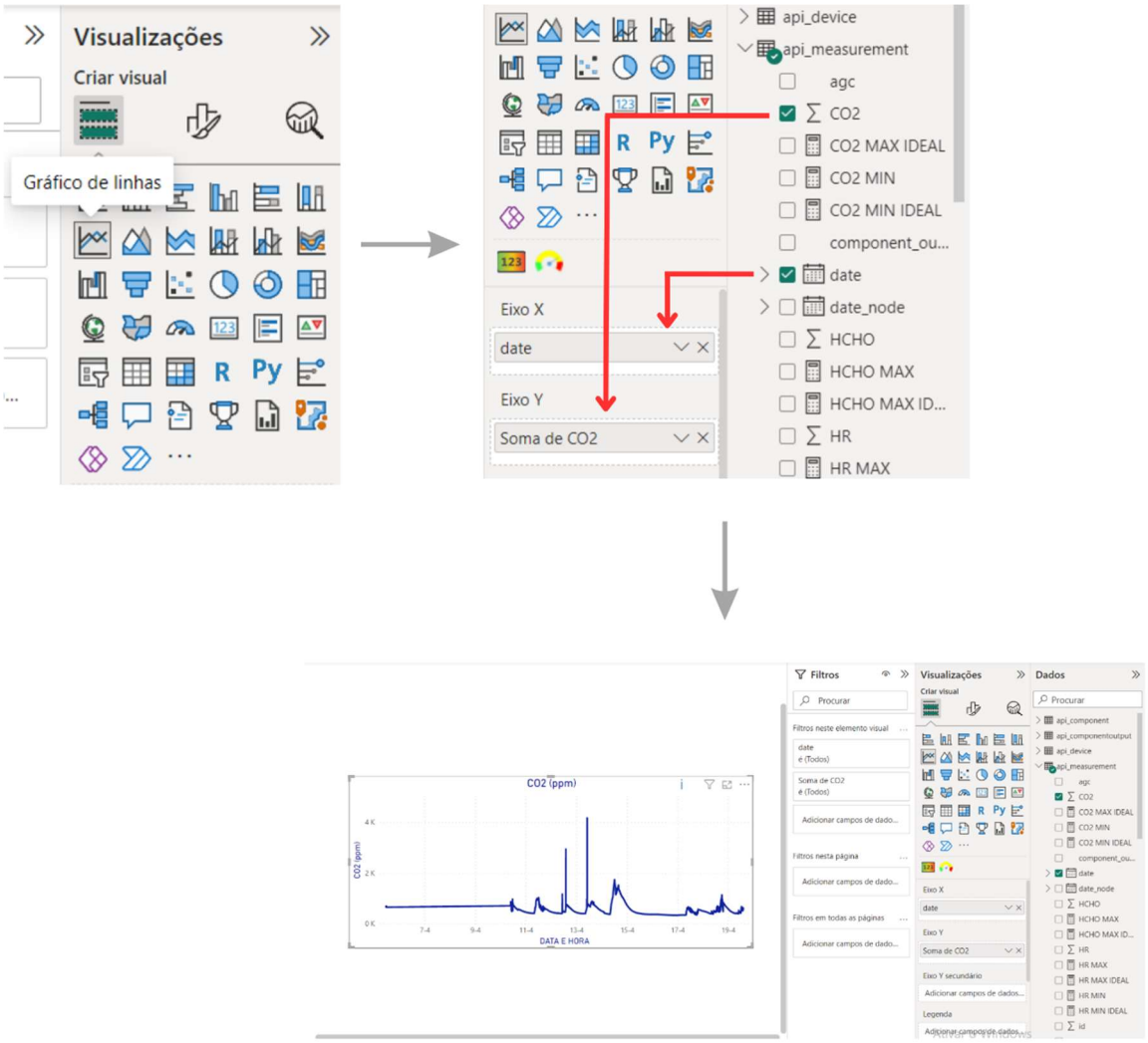


Fig. 30 - Criação de Gráficos de linhas

3.5.3.3. Criação de gráficos de barras

Os gráficos de barras são bastante úteis para a análise da distribuição dos dados relativamente a cada parâmetro. Todos os resultados são expressos em percentagem para uma interpretação mais simples. O eixo x dos gráficos apresentam os intervalos definidos para cada um dos parâmetros, já o eixo y apresenta a percentagem relativa a cada intervalo. Os gráficos apresentam as cores verde e vermelha, conforme o intervalo de valores seja desejado ou indesejado, respetivamente, estas cores estão em sintonia com as cores apresentadas nos indicadores, tornando a visualização e interpretação acessíveis, Fig. 31.

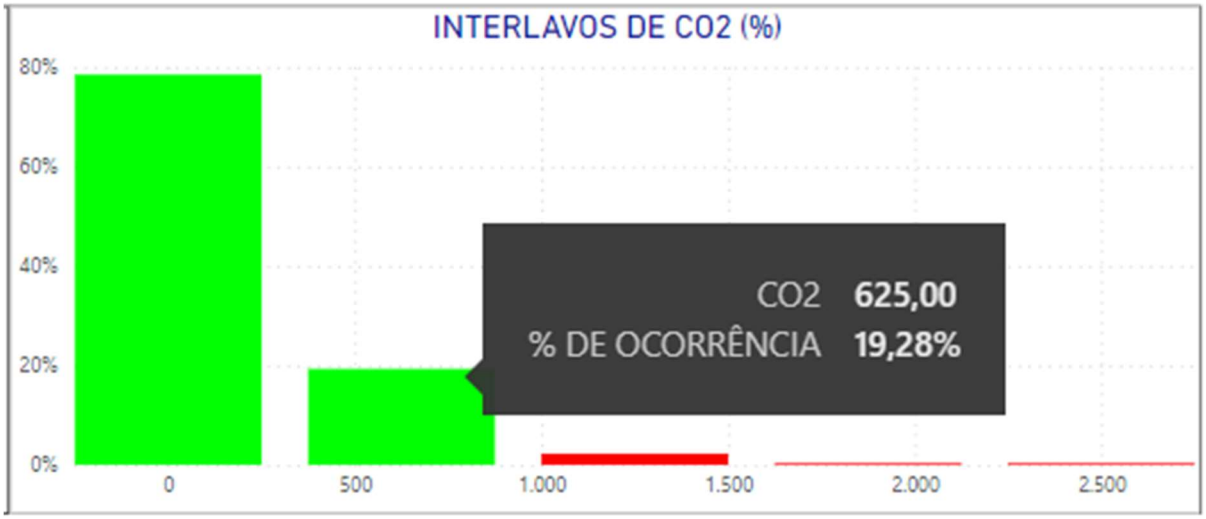


Fig. 31 - Gráfico de barras

Assim como os gráficos de linhas, os gráficos de barras também são uma ferramenta existente na interface do Power BI, denominado "gráfico de colunas empilhadas". Inicialmente foram adicionados os valores do eixo y, com o objetivo de estudar a percentagem de ocorrência de um determinado parâmetro, foi adicionada a coluna "CO2", este valor foi editado de forma a surgir como "Percentagem do total geral". Posteriormente foram atribuídos os valores ao eixo y, foi utilizada a mesma coluna, "CO2", contudo foi necessário criar grupos, esta ferramenta permite separar a amostra em intervalos, neste caso foram selecionados intervalos de 625 ppm.

Os intervalos de valores foram ajustados para cada parâmetro, de modo a obter gráficos visualmente interessantes e com amostras de valores significativas. Como é visível na Fig. 31, 19,28% da medições de concentração de CO_2, inserem-se no intervalo compreendido entre 625 ppm e 1250 ppm.

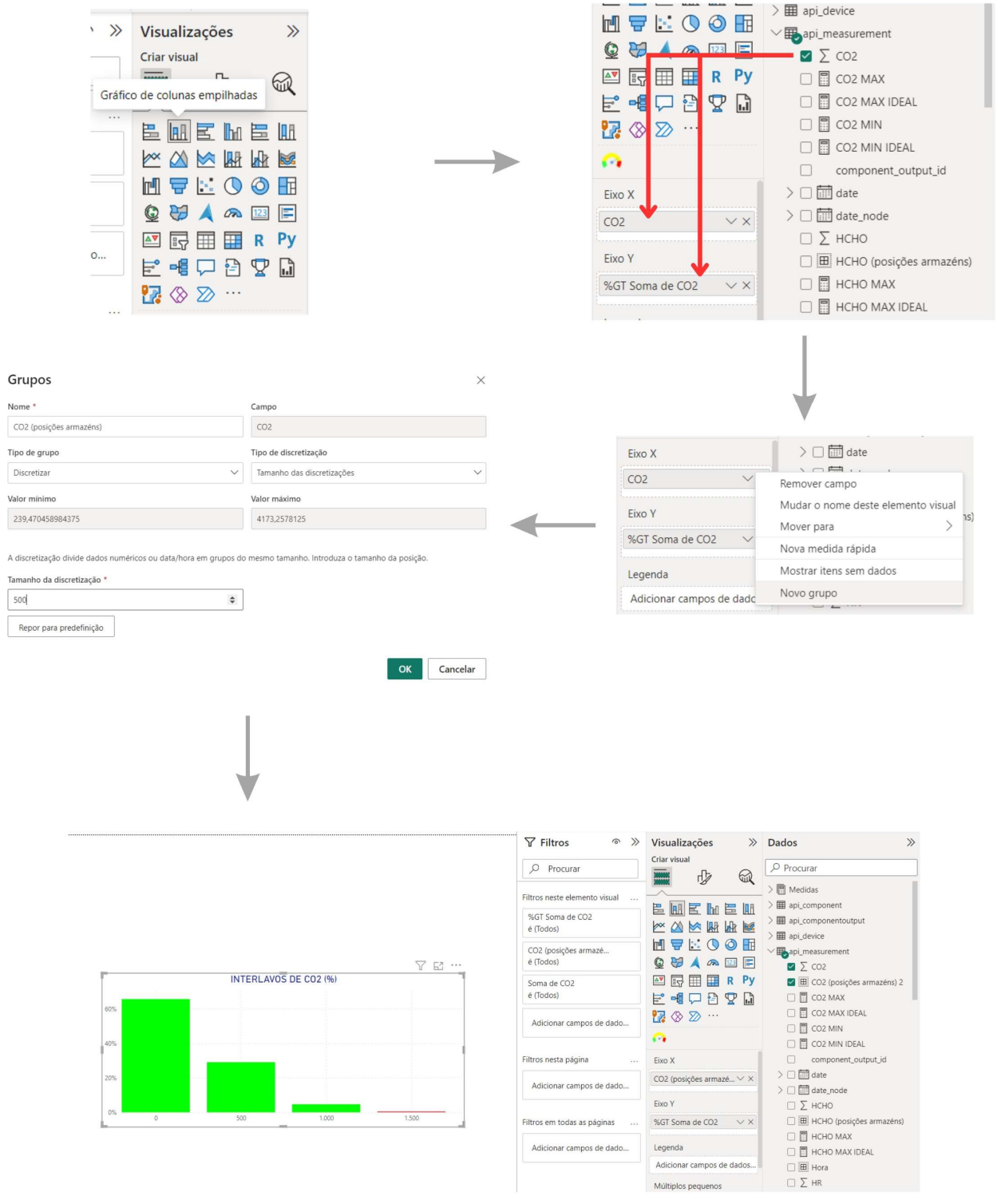


Fig. 32 - Criação de gráficos de barras

3.5.4. DADOS ESTATÍSTICOS

Foram também adicionados dados estatísticos ao Dashboard, este tratamento de dados é bastante útil pois permite analisar padrões e tendências das diferentes amostras existentes. Na Fig. 33estão representados os dados estatísticos relativos à concentração de CO_2.

CO2 (ppm)					
239,47	278,74	332,28	459,01	1.735,17	402,63
MÍNIMO	1º QUARTIL	MEDIANA	3º QUARTIL	MÁXIMO	MÉDIA

Fig. 33 - Dados estatísticos – Concentração de CO_2

Estes elementos foram criados utilizando o "cartão de linhas múltiplas", este elemento visual permite adicionar diversas medidas a serem apresentadas. Como forma de tratamento estatístico foram consideradas as seguintes medidas estatísticas: valor mínimo, 1º quartil, mediana, 3º quartil, valor máximo e média, Tab. 3.

Tab. 3 - Fórmulas de Cálculo Estatístico

Medida estatística	Fórmula de cálculo no Power BI
Valor mínimo	`MIN CO2 = MIN(api_measurement[CO2])`
1º Quartil	`1Q CO2 = PERCENTILE.INC(api_measurement[CO2], 0.25)`
Mediana	`MEDIANA CO2 = MEDIAN(api_measurement[CO2])`
3º Quartil	`3Q CO2 = PERCENTILE.INC(api_measurement[CO2],0.75)`
Valor máximo	`MAX CO2 = MAX(api_measurement[CO2])`
Média	`MEDIA CO2 = AVERAGE(api_measurement[CO2])`

3.6. CRIAÇÃO DO DASHBOARD

Como já foi referido anteriormente, um Dashboard é uma interface gráfica que apresenta informações relevantes num formato visual fácil de entender. O Power BI Service possui uma ferramenta que permite a criação de Dashboards e a partir dai adicionar as informações contidas nos relatórios, criados no Power BI Desktop.

Para a criação do Dashboard foi necessário aceder ao Power BI Service, através da aba "A minha área de trabalho", selecionando a opção "Novo", surgem várias ferramentas, foi selecionado "Dashboard" e atribuído o nome "Dashboard", Fig. 34.

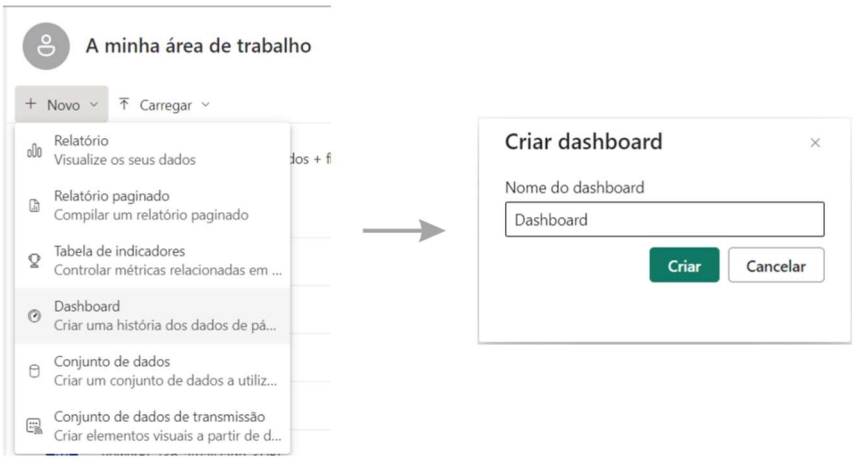


Fig. 34 - Criação do Dashboard

Para tornar o Dashboard visualmente mais apelativo foi adicionado um tema de fundo com uma imagem da FEUP, fornecida pela Universidade, deste modo o contacto do utilizador com o Dashboard torna a sua experiência mais aprazível.

Utilizando o Power BI Service, foi aberto o Dashboard criado, na aba "Editar", selecionando a opção "Tema do Dashboard" foi possível alterar a predefinição e adicionar a imagem de fundo pretendida, Fig. 35.

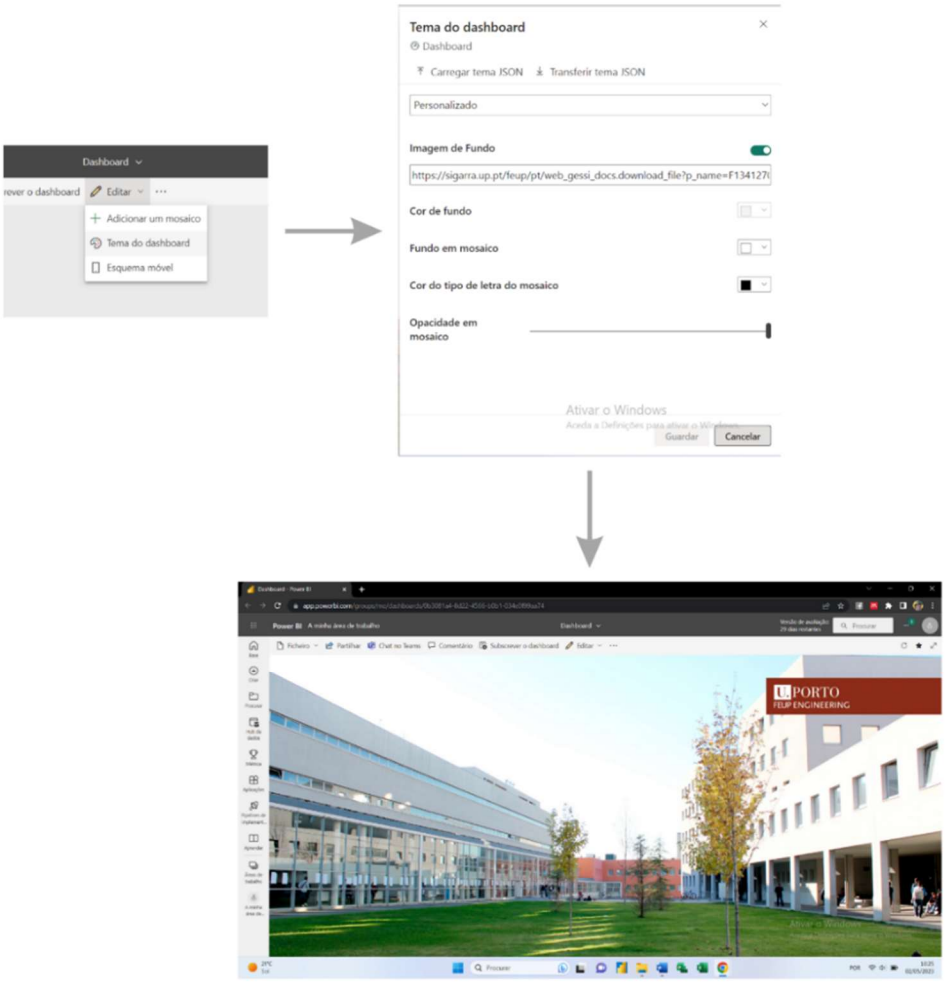


Fig. 35 - Introdução do tema do Dashboard

Todos os elementos abordados anteriormente, são eles, o modelo 3D, os dados sobre o Digiroom e salas adjacentes, os indicadores em tempo real e os dados históricos foram adicionados ao Dashboard, de forma a garantir que o utilizador aceda a todos os dados de forma simultânea e dinâmica, no capítulo seguinte será apresentada a disposição de todos estes elementos gráficos no Dashboard.

3.7. Atuador de Janela

O Digiroom possui um dispositivo de acionamento de janela, assim, é possível abrir e fechar a janela da sala através de comandos acionados à distância. De forma a possibilitar a conexão com a janela foi utilizado o software Mosquitto, já referido anteriormente. Na Fig. 36 é possível identificar o atuador de janela, localizado na parte inferior da mesma.

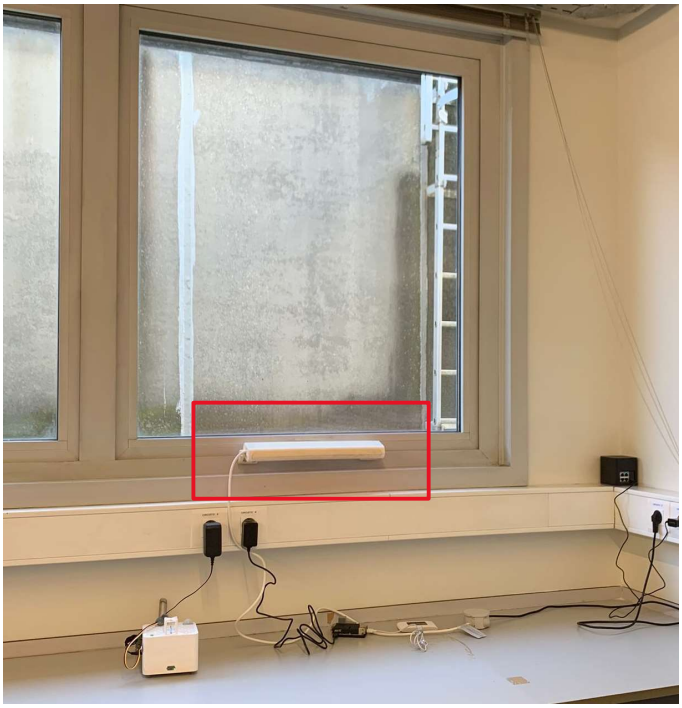


Fig. 36 - Atuador de janela

3.7.1. CONTROLO DA JANELA ATRAVÉS DA LINHA DE COMANDOS

Inicialmente, de forma a conseguir acionar o atuador, foi desenvolvido um código diretamente pela linha de comandos onde era feita a ligação ao Mosquitto e consequentemente à rede da sala, EP_base, Fig. 37, assim, com a ligação estabelecida restava enviar o código correto que acionasse a janela conforme o comando pretendido.

Deste modo, conciliando os comandos foi possível abrir, fechar e parar a janela, Tab. 4, Fig. 38. Contudo, esta forma de conexão ao atuador não era a mais desenvolvida, visto que era necessário reescrever o código completo sempre que fosse necessário realizar uma das ações.

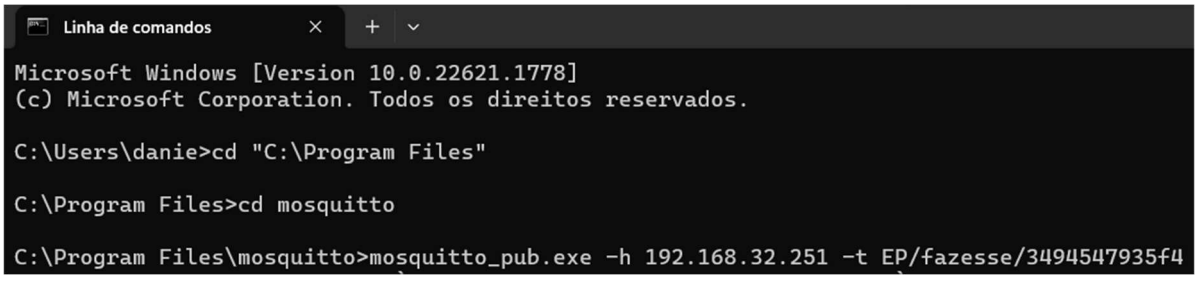


Fig. 37 - Comando para conectar à janela

Tab. 4 - Comandos para atuador de janela - Linha de comandos

Ação a executar	Comando utilizado
Abrir a janela	`-m {\"command\":\"roller/0?go=open\"}`
Fechar a janela	`-m {\"command\":\"roller/0?go=close\"}`
Parar a janela	`-m {\"command\":\"roller/0?go=stop\"}`

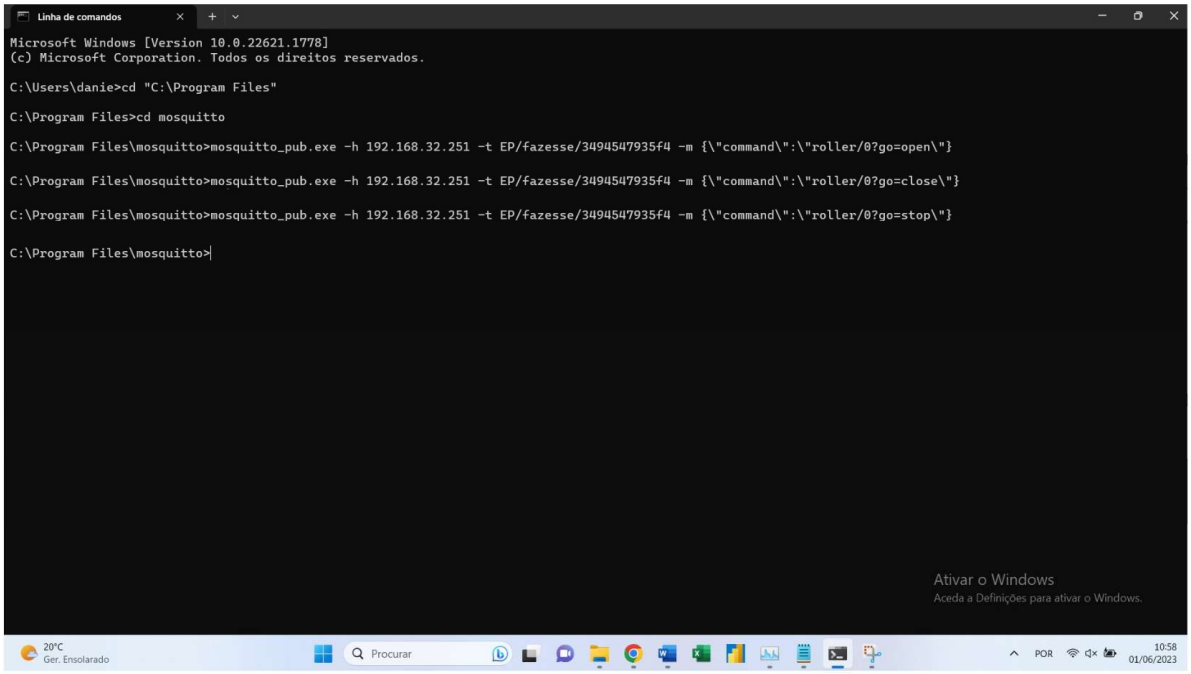


Fig. 38 - Código atuador de janela completo - Linha de comandos

3.7.2. CONTROLO DA JANELA ATRAVÉS DO PYTHON

Posteriormente, de forma a simplificar o processo de manuseamento da janela, pensando na operabilidade de qualquer utilizador, foram desenvolvidos vários ficheiros em Python, todos eles com a mesma base de código, 3 desses com comandos elementares, abrir, fechar e parar a janela. Estes ficheiros são em tudo semelhantes, com exceção do comando enviado ao atuador de janela (Anexos 1, 2 e 3). Ao abrir cada um dos ficheiros, a ação para a qual o ficheiro está destinado é executada imediatamente e o programa fecha de forma automática, deste modo, com um duplo click o utilizador pode acionar na janela o movimento pretendido. Além disso, a base do código está apta para receber novos comandos que permitam acionar os restantes equipamentos do Digiroom, no entanto, não foram utilizados neste estudo.

Com o objetivo de automatizar o atuador de janela foi criado outro código em Python, utilizando a mesma base anteriormente referida, que combina 2 parâmetros, a pressão parcial de vapor de água e a concentração de CO_2. Este código funciona de forma automática, quando aberto, fica imediatamente conectado ao atuador da janela e emite mensagens com comandos para abrir ou fechar a janela consoante

os valores recebidos pelos sensores. Além disso, com o objetivo de guardar informação sobre os valores medidos e sobre os parâmetros calculados, o código foi programado para enviar essas mensagens para um ficheiro de texto (.txt). De forma a monitorizar a pressão parcial do vapor de água e a concentração de CO_2 os dados foram também enviados para uma folha em Excel, deste modo, os valores poderiam ser trabalhados no Power BI, na forma de gráficos, conforme apresentado no capítulo seguinte.

3.7.2.1. Pressão de saturação e pressão parcial do vapor de água

De modo a demonstrar as potencialidades do Digital Building Twins, foi utilizada a pressão parcial do vapor de água, visto que este indicador está diretamente relacionado com a temperatura e a humidade relativa. Mostrando assim a possibilidade de combinar vários parâmetros medidos num único índice ou medida.

A pressão parcial de vapor é uma medida que depende da humidade relativa e da pressão de saturação que por sua vez varia em função da temperatura. Seguidamente são apresentadas as expressões utilizadas para o cálculo da pressão de saturação e pressão parcial do vapor de água (pv) [45] a partir da pressão de saturação (psat). A pressão parcial do vapor de água (Pa) resulta do produto entre o valor da humidade relativa (%) e a pressão de saturação (Pa).

$$psat = 611 * e^{\left(\frac{17.08*T}{234.18+T}\right)} \quad , \qquad para\ T \geq 0°C \tag{3.1}$$

$$pv = HR * psat \tag{3.2}$$

Assim, tendo em conta os intervalos de valores definidos como ideais para a temperatura e a humidade relativa foi calculado o intervalo de valores ideais para a pressão do vapor de água no Digiroom, utilizando as fórmulas apresentadas (3.1 e 3.2).

De modo a obter um valor crítico para possível demonstração do funcionamento do código tendo em conta este parâmetro, foi calculada a pressão de saturação para os valores de temperatura máxima admissível (25º) e humidade relativa máxima admissível (70%). Assim, foi obtido o valor máximo de 2221,47 Pa, de forma a simplificar sem comprometer a qualidade do ambiente interior, este valor foi arredondado por defeito para 2000 Pa.

No que diz respeito ao código realizado no Python para automatizar a abertura e fecho da janela, foi considerado o valor da pressão parcial de vapor de água ideal máxima de 2000 Pa, e a concentração de CO_2 ideal abaixo de 900 ppm. Estes valores não são críticos, foram utilizados de forma a testar o bom funcionamento do sistema e detetar possíveis falhas do mesmo.

Quanto à concentração de CO_2, o comando de abertura da janela é bastante útil, uma vez que a principal fonte de emissão de CO_2 é a respiração humana e a entrada de ar exterior faria baixar este valor [22]. Já para o controlo da pressão parcial do vapor de água, os comandos de abertura e fecho da janela não garantem a estabilização deste parâmetro, uma vez que não existem dispositivos instalados que façam a medição das condições ambientais exteriores. Por esse motivo, a pressão parcial do vapor de água ideal mínima não foi aplicada no código, o valor máximo deste parâmetro foi utilizado como valor limite para demonstrar a possibilidade de combinar mais que um parâmetro para ativar o atuador de janela.

3.7.2.2. Descrição do código criado em Python

No código em Python (Anexo 4), foi feita a ligação ao Mosquitto, que apresentava os valores medidos relativos à temperatura, humidade relativa e concentração de CO_2, aplicando as fórmulas referidas (3.1 e 3.2) eram obtidos os valores de pressão de saturação e pressão parcial do vapor de água. Deste modo foi possível combinar 2 parâmetros (pressão parcial do vapor de água e concentração de CO_2) de forma a criar condições que levassem à ativação da janela. Assim, sempre que os 2 parâmetros se encontravam abaixo do valor definido, a janela fechava, quando pelo menos um dos parâmetros estava acima do desejado a janela abria, de forma a estabilizar os valores medidos. Na Fig. 39, é apresentado um fluxograma representativo das condicionantes que fazem abrir ou fechar a janela consoante os valores medidos.

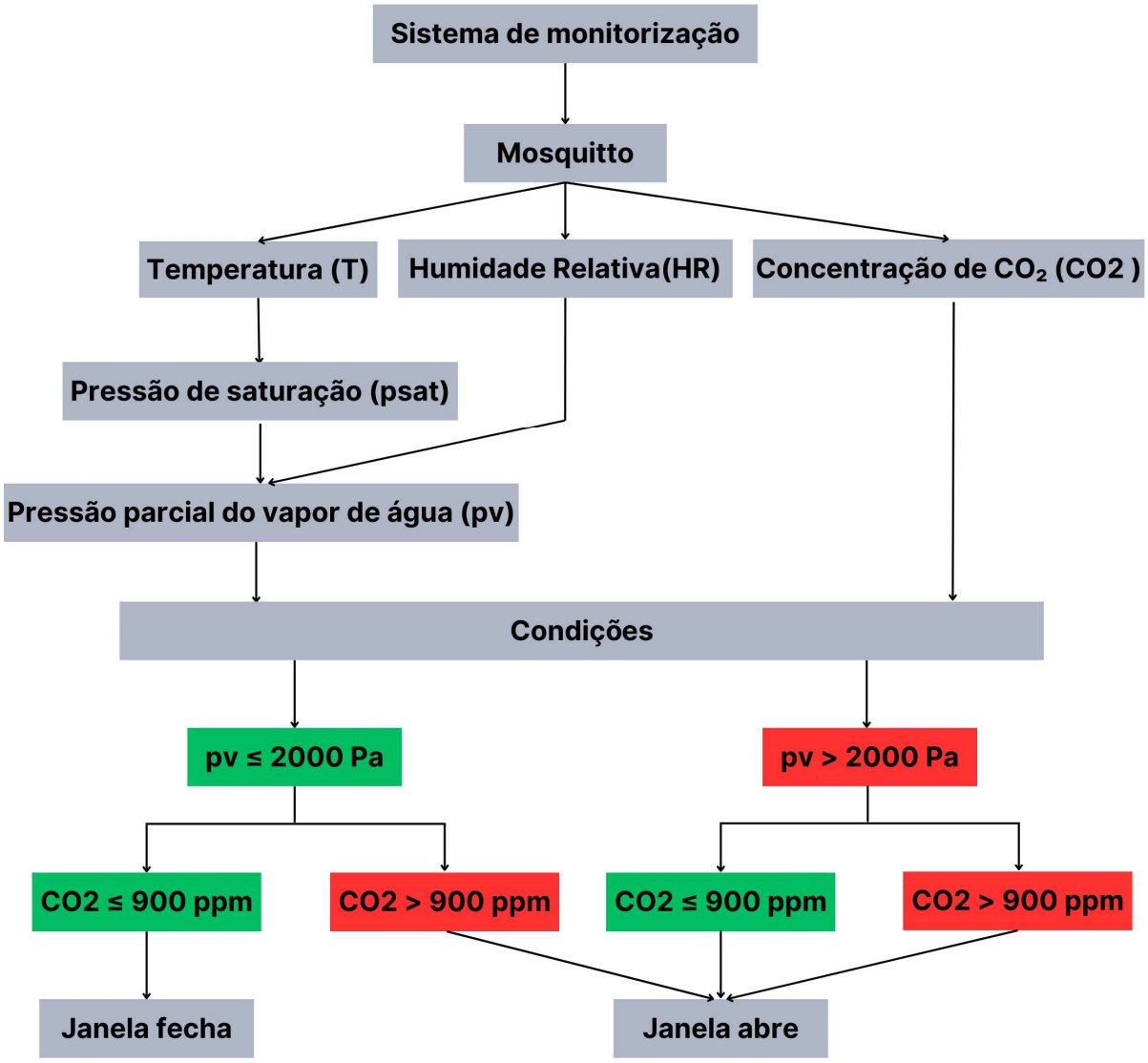


Fig. 39 – Fluxograma com condições para ativar a janela

O código em Python desenvolvido é bastante extenso, totalizando mais de 400 linhas de código, contudo este código tem 4 objetivos principais:

- Calcular a pressão parcial do vapor de água;

- Enviar os comandos para abrir ou fechar a janela;
- Guardar o registo das mensagens num ficheiro de texto;
- Guardar os dados numa folha Excel.

Para que a conexão entre o sistema de monitorização e a janela existisse, foi necessário estabelecer essa ligação através do Mosquitto, que é compatível com o Python.

Para o cálculo da pressão parcial do vapor de água foi necessário utilizar o "import math", este comando importa o módulo "math", que fornece funções matemáticas comuns, como operações de arredondamento, cálculos trigonométricos, funções exponenciais, entre outras. Seguidamente, após a definição das variáveis necessárias foi possível introduzir as fórmulas, primeiro a fórmula da pressão de saturação e depois a fórmula da pressão parcial do vapor de água. Foi definido que todos os valores teriam 2 casas decimais e que iriam surgir mensagens a apresentar os 3 valores, pressão de saturação, pressão parcial do vapor de água e concentração de CO_2. Na Fig. 40 é apresentado um excerto do código referente ao cálculo da pressão parcial do vapor de água.

```
while True:
    try:
        #### Exemplo: cálculo de psat e pv ####
        if temperature and rh and co2:
            temperature = float(temperature)
            rh = float(rh)
            co2 = float(co2)
            psat = 611 * math.exp((17.08 * temperature) / (234.18 + temperature))
            pv = (rh/100) * psat
            psat_rounded = round(psat, 2)
            pv_rounded = round(pv, 2)
            co2_rounded = round(co2, 2)
            print("A pressão de saturação é: {} Pa".format(psat_rounded))
            print("A pressão parcial do vapor de água é: {} Pa".format(pv_rounded))
            print("A concentração de CO2 é: {} ppm".format(co2_rounded))
```

Fig. 40 - Python - Cálculo da pressão parcial do vapor de água

De forma a conseguir operar a janela, foram também criadas algumas linhas de código, foi necessário definir as variáveis dos comandos desejados, Fig. 41 e compatibilizá-los com os códigos lidos pelo Mosquitto, para conectar com o atuador de janela, Fig. 42. Com a criação de condições foi possível automatizar os movimentos da janela tendo em conta os valores da pressão parcial do vapor de água e a concentração de CO_2, Fig. 43.

```
######################
## Window commands ##
######################
cmd_window_open = '{"command":"roller/0?go=open"}'      # Open window
cmd_window_close = '{"command":"roller/0?go=close"}'    # Close window
cmd_window_stop = '{"command":"roller/0?go=stop"}'      # Stop window movement
window_cmd_response = {}    # stored device response
```

Fig. 41 - Python - Definição dos comandos da janela

```python
def send_cmd_to_dev(mac_addr, cmd):
    global window_cmd_response
    global heater_cmd_response

    mqtt_client.publish("EP/fazesse/" + re.sub('\W+', '',
                        mac_addr), payload=cmd, retain=False)
    if mac_addr == window:
        window_cmd_response["command"]=cmd
        window_cmd_response["response"]=None
    elif mac_addr == heater:
        heater_cmd_response["command"]=cmd
        heater_cmd_response["response"]=None
    else:
        print("Device "+mac_addr+" is does not accept commands")
```

Fig. 42 - Python - Envio dos comandos ao atuador de janela

```python
#### Exemplo: controlo da janela através da pv e do co2 ####
if pv:
    if pv <= 2000 and co2 <= 900:
        send_cmd_to_dev(window, cmd_window_close)
        print("Janela fechada porque Pressão parcial de vapor ≤ 2000 Pa e concentração de CO2 ≤ 900 ppm")
        save_message_to_file("Janela fechada porque Pressão parcial de vapor ≤ 2000 Pa e concentração de CO2 ≤ 900 ppm")
    elif pv > 2000 or co2 > 900:
        send_cmd_to_dev(window, cmd_window_open)
        if pv > 2000 and co2 <= 900:
            print("Janela aberta porque Pressão parcial de vapor > 2000 Pa")
            save_message_to_file("Janela aberta porque Pressão parcial de vapor > 2000 Pa")
        if pv <= 2000 and co2 > 900:
            print("Janela aberta porque Concentração de CO2 > 900 ppm")
            save_message_to_file("Janela aberta porque Concentração de CO2 > 900 ppm")
        if pv > 2000 and co2 > 900:
            print("Janela aberta porque Pressão parcial de vapor > 2000 Pa e porque Concentração de CO2 > 900 ppm")
            save_message_to_file("Janela aberta porque Pressão parcial de vapor > 2000 Pa e Concentração de CO2 > 900 ppm")
```

Fig. 43 - Python - Condições para abrir e fechar janela

Conforme descrito acima, um dos objetivos era guardar o registo das mensagens num ficheiro de texto, denominado “mensagens”. Como o objetivo era guardar essa informação juntamente com a data e hora dos dados recolhidos, foi necessário adicionar essa informação, através do “import datetime”. Assim, foi possível criar o ficheiro de texto (.txt) e adicionar a data e hora, Fig. 44, bem como a informação dos dados em estudo, Fig. 45. Foram também enviadas informações sobre quando a janela abria ou fechava e a que se devia esse movimento, visível na Fig. 43.

```python
def save_message_to_file(message):
    data_hora = datetime.datetime.now()
    data_hora_formatada = data_hora.strftime("%d/%m/%Y %H:%M:%S")

    with open('mensagens.txt', 'a') as arquivo:
        arquivo.write("Data/Hora: {}\n".format(data_hora_formatada))
        arquivo.write(message + "\n\n")
```

Fig. 44 - Python - Registo da data e hora e criação de ficheiro de texto

```python
# Guardar mensagem no arquivo
message = "A pressão de saturação é: {} Pa\nA pressão parcial do vapor de água é: {} Pa\
\nA concentração de CO2 é: {} ppm".format(psat_rounded, pv_rounded,co2_rounded)
save_message_to_file(message)
```

Fig. 45 - Python - Envio de dados para ficheiro de texto

O último objetivo do código criado em Python era guardar os dados numa folha Excel. Inicialmente foi necessário importar a função que permite trabalhar com arquivos do Microsoft Excel, "import openpyxl". Foram definidas as variáveis que seriam guardadas no documento (pressão de saturação, pressão parcial do vapor de água e concentração de CO_2), estes valores surgiriam por colunas, foi definido que na primeira coluna surgiria a data e hora e nas 3 seguintes, os valores obtidos. A primeira linha da folha continha um título sugestivo da informação contida em cada coluna. Sempre que o código era executado, a informação era adicionada à folha Excel sem apagar os dados anteriores, fazendo assim um registo histórico para que posteriormente fosse possível trabalhar os dados, Fig. 46.

```python
# Função para guardar os valores no Excel
def save_to_excel(psat, pv, co2):
    # Carregar ou criar um arquivo do Excel
    try:
        wb = openpyxl.load_workbook('dados.xlsx')
    except FileNotFoundError:
        wb = openpyxl.Workbook()

    # Selecionar a folha ativa (a primeira folha por padrão)
    sheet = wb.active

    # Obter a data e hora atual
    data_hora_atual = datetime.datetime.now()

    # Verificar se a primeira linha está vazia
    if sheet.cell(row=1, column=1).value is None:
        sheet.cell(row=1, column=1).value = "DATA e HORA"
        sheet.cell(row=1, column=2).value = "PSAT"
        sheet.cell(row=1, column=3).value = "PV"
        sheet.cell(row=1, column=4).value = "CO2"

    # Adicionar valores às colunas
    row_count = sheet.max_row + 1
    sheet.cell(row=row_count, column=1).value = data_hora_atual
    sheet.cell(row=row_count, column=2).value = psat
    sheet.cell(row=row_count, column=3).value = pv
    sheet.cell(row=row_count, column=4).value = co2

    # Salvar o arquivo do Excel
    wb.save('dados.xlsx')
```

Fig. 46 - Python - Guardar dados numa folha Excel

Assim, o código estava pronto a executar corretamente, no anexo 4 é possível visualizar o código completo. Como já referido anteriormente, este código possui diversas linhas úteis para desenvolvimentos futuros, estão designados os dispositivos que, assim como a janela, poderão ser controlados à distância e os respetivos comandos de ativação. O código conta com bastantes comentários, apresentados na cor verde, que são partes do código que o Python ignora, o símbolo "#" introduzido é usado para iniciar um comentário, desta forma, qualquer utilizador compreenderá o propósito das linhas que surgem após os comentários, de forma a facilitar qualquer alteração que se preveja necessária.

4. RESULTADOS

4.1. ATUADOR DE JANELA

Conforme abordado no capítulo 1, um dos objetivos desta dissertação era conseguir implementar um atuador no Digiroom, no caso, o atuador de janela, de modo a ser operado através do Digital Building Twin. Na Fig. 47, surge uma fotografia tirada à janela do Digiroom instantes após esta ser fechada de forma automática, os parâmetros avaliados, pressão parcial do vapor de água e concentração de CO_2 encontravam-se abaixo do limite definido, e por isso, a janela fechou.



Fig. 47 - Janela fechada

Os comandos desenvolvidos estavam programados para abrir a janela quando pelo menos um dos parâmetros medidos, pressão parcial do vapor de água e/ou concentração de CO_2, ultrapassasse o limite definido, de forma a baixar esses parâmetros para valores aceitáveis, a Fig. 48 ilustra esse acontecimento.



Fig. 48 - Janela aberta

De modo a possibilitar uma futura consulta dos dados e dos acontecimentos foi criado um ficheiro de texto, conforme mencionado no capítulo anterior. Neste ficheiro eram introduzidos: data e hora da recolha dos dados, valor da pressão de saturação, valor da pressão parcial do vapor de água, concentração de CO_2 e uma mensagem que informava o leitor se a janela estava aberta ou fechada e o motivo que a levou a abrir ou fechar Fig. 49.

Existem 4 mensagens a explicar o motivo pelo acionamento da janela, são elas:

- "Janela fechada porque Pressão parcial de vapor < 2000 Pa e concentração de CO_2 < 900 ppm"
- "Janela aberta porque Pressão parcial de vapor > 2000 Pa"
- "Janela aberta porque Concentração de CO_2 > 900 ppm"
- "Janela aberta porque Pressão parcial de vapor > 2000 Pa e porque Concentração de CO_2 > 900 ppm"

Desta forma o utilizador ficava com um registo completo sobre os movimentos da janela e sabia qual a razão do acionamento da mesma.

De forma a testar o bom funcionamento do recetor de mensagens, um estudante colocou-se junto do sistema de monitorização, direcionando a sua respiração, pontualmente, para o mesmo. As mensagens eram enviadas com um intervalos de 1 minuto. No dia 12/06/2023 às 11:01:52h os valores medidos eram admissíveis, a janela estava fechada, no minuto seguinte, devido à aproximação do estudante do sistema de monitorização a concentração de CO_2 ultrapassou o limite definido, fazendo com que a janela abrisse. Com o afastamento do estudante do sistema de monitorização, este valor voltou a reduzir, tendo sido verificado, novamente, um valor admissível, que fez a janela fechar. Deste modo foi possível comprovar que não existiam erros no código e que a receção das mensagens estava a funcionar na perfeição.

```
Data/Hora: 12/06/2023 11:01:52
A pressão de saturação é: 3263.95 Pa
A pressão parcial do vapor de água é: 1859.56 Pa
A concentração de CO2 é: 518.8 ppm

Data/Hora: 12/06/2023 11:01:52
Janela fechada porque Pressão parcial de vapor < 2000 Pa e concentração de CO2 < 900 ppm

Data/Hora: 12/06/2023 11:02:52
A pressão de saturação é: 3267.05 Pa
A pressão parcial do vapor de água é: 1862.17 Pa
A concentração de CO2 é: 925.74 ppm

Data/Hora: 12/06/2023 11:02:52
Janela aberta porque Concentração de CO2 > 900 ppm

Data/Hora: 12/06/2023 11:03:52
A pressão de saturação é: 3269.64 Pa
A pressão parcial do vapor de água é: 1863.25 Pa
A concentração de CO2 é: 576.81 ppm

Data/Hora: 12/06/2023 11:03:52
Janela fechada porque Pressão parcial de vapor < 2000 Pa e concentração de CO2 < 900 ppm
```

Fig. 49 - Registo de mensagens

4.1.1. COMPARAÇÃO DE DADOS: IMPACTO DO USO E NÃO USO DE ATUADOR

Para avaliar a influência do funcionamento dos algoritmos de controlo automático do atuador de janela na concentração de CO_2 no Digiroom, foi feita a comparação entre a utilização e não utilização desta automação, procurando compreender de que forma o atuador pode ter impacto na qualidade do ar interior.

De modo a garantir condições semelhantes, os dados foram recolhidos em dois dias consecutivos, no mesmo horário. No primeiro dia, todas as janelas foram mantidas fechadas, com a automação da abertura da janela ativa, enquanto no dia seguinte, também com todas as janelas fechadas, a automação foi desativado. Essa abordagem permitiu comparar diretamente o funcionamento dos algoritmos de controlo automático do atuador de janela na concentração de CO_2 no Digiroom. Foi ainda tida em conta a quantidade de ocupantes que permaneciam na sala, uma vez que a presença de pessoas no espaço é o principal fonte de emissão de CO_2.

Durante a recolha de dados com o atuador ativo, o primeiro valor registado foi de 363,52 ppm. Esse valor foi considerado relativamente baixo devido à ausência de ocupantes na sala nas 14 horas anteriores e às janelas terem permanecido abertas nesse período. Com a presença de 1 ocupante na sala, a concentração de CO_2 aumentou gradualmente. Às 10:24 h, a ocupação da sala aumentou para 2 pessoas, o que resultou no aumento esperado na concentração de CO_2. Às 10:32 h, o atuador foi acionado automaticamente, abrindo a janela devido à pressão parcial do vapor de água ter ultrapassado o limite definido. A partir desse momento, houve uma leve queda nos valores, seguida da sua estabilização até a entrada de outra pessoa na sala às 11:26 h, quando os valores começaram a subir novamente. Com a entrada de mais 2 pessoas no Digiroom, totalizando 5 ocupantes às 12:27 h, a concentração de CO_2 sofreu um aumento acentuado. Mesmo após a saída de 2 ocupantes às 12:48 h, foi registado o valor máximo de 937,48 ppm. Após atingir esse valor, a concentração de CO_2 diminuiu abruptamente devido à rápida redução na ocupação do Digiroom, que passou de 5 para 1 pessoa num curto período, Fig. 50.

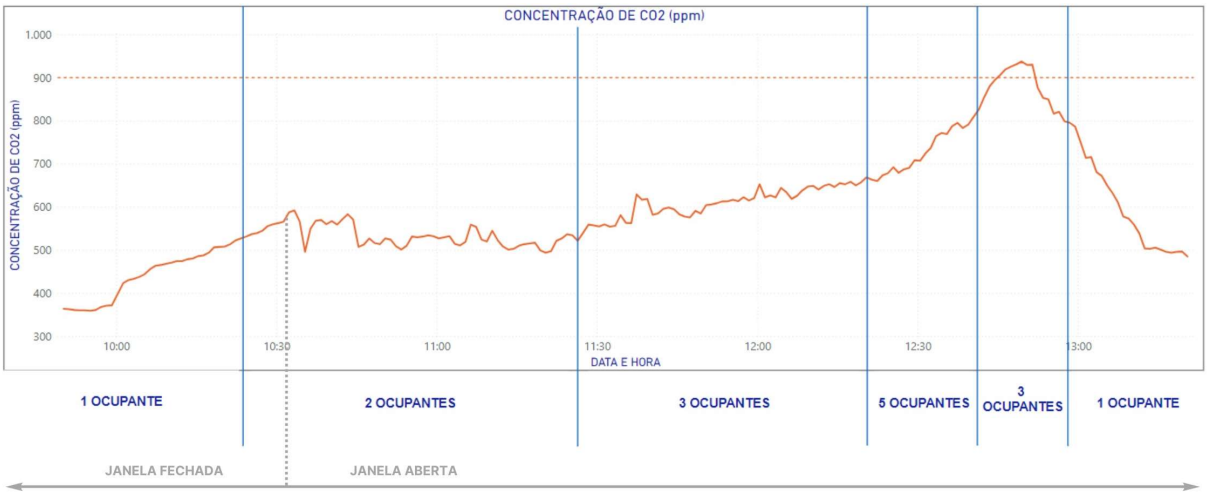


Fig. 50 - Concentração de CO_2 com o atuador de janela ativo

No dia seguinte, foi feita a recolha de dados com a automação inativa, o primeiro valor registado foi de 393,22 ppm. Um valor bastante próximo do registado no dia anterior, à mesma hora, visto as condições terem sido repetidas. Com a presença de 1 ocupante na sala, a concentração de CO_2 aumentou

gradualmente. Às 10:31 h, a ocupação da sala aumentou para 2 pessoas e a concentração de CO_2 continuou a aumentar de forma constante, mesmo com a chegada de outro ocupante, pelas 11:29 h. Com a entrada de mais 2 pessoas no Digiroom, totalizando 5 ocupantes às 12:22 h, a concentração de CO_2 sofreu um aumento significativo que não se tinha verificado até então. Mesmo após a saída de 2 ocupantes às 12:45 h, os valores continuaram a subir. Pelas 12:59 h apenas permanecia 1 ocupante na sala, contudo a concentração de CO_2 continuou a aumentar, embora de forma menos expressiva, chegando a atingir o valor máximo de 1285,68 ppm, que já era superior ao máximo admitido pela Portaria n.º 138-G/2021 [43].

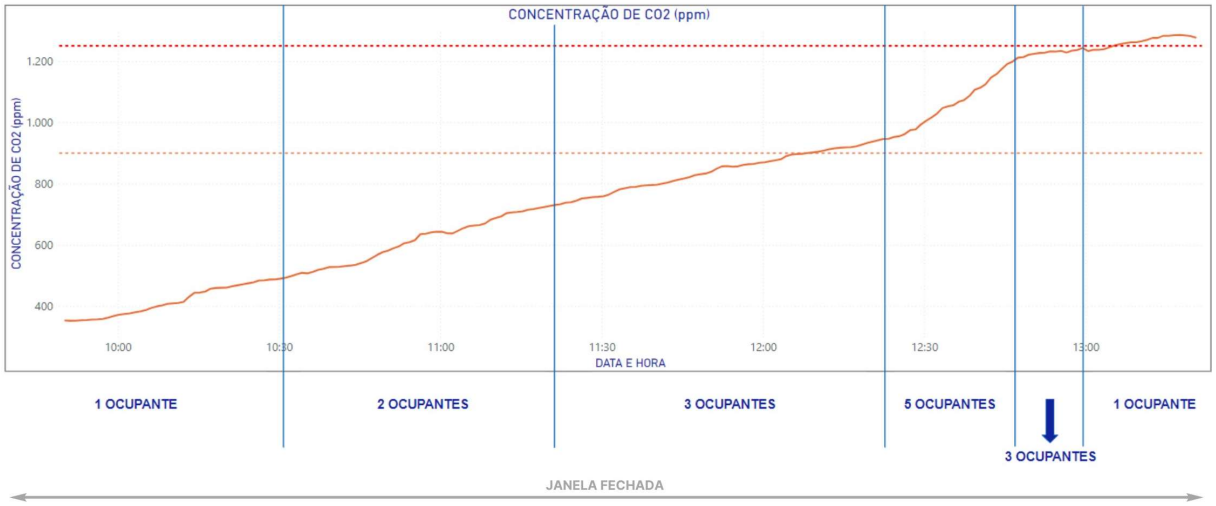


Fig. 51 - Concentração de CO_2 com o atuador de janela inativo

Analisando os 2 gráficos apresentados, é evidente o impacto que a abertura da janela causou nos valores obtidos. Comparando os resultados até às 10:32 h, momento em o atuador foi acionado e a janela abriu, existe semelhança no desenvolvimento dos gráficos, ambos crescem de forma relativamente constante.

A partir desse instante surgem as primeiras diferenças, no caso em que o atuador estava ativo, a abertura da janela levou a uma ligeira queda e posterior estabilização dos parâmetros até entrar outra pessoa na sala, já no caso em que a janela se manteve fechada a concentração de CO_2 continuou a aumentar de forma constante.

Na recolha de dados com o atuador ativo, a concentração de CO_2 voltou a aumentar com a chegada do terceiro ocupante, registando o valor de 668,57 ppm, mesmo com a janela aberta. Já no dia em que a automação da abertura da janela estava inativa, o parâmetro continuou a ganhar expressão, chegando a atingir 946,14 ppm.

Com a presença de 5 ocupantes, em ambos os casos, os valores aumentaram significativamente, e mesmo após a redução da ocupação para 3 pessoas, estes valores continuaram a subir.

A principal disparidade de valores surge quando a sala fica novamente ocupada unicamente por 1 pessoa. No cenário em que os algoritmos de funcionamento controlavam o estado da janela, os valores desceram drasticamente. Já no cenário em que a janela se encontrava fechada (automação desligada) a concentração de CO_2 continuou a subir, embora de forma menos expressiva, chegando a ultrapassar o limite regulamentar de 1250 ppm.

Com estes resultados é possível verificar que a automatização do atuador de janela contribuiu de forma positiva para regular a concentração de CO_2 e, consequentemente na qualidade do ar interior. Na Fig.

52 são visíveis os box plots relativas às leituras de concentração de CO_2 com o atuador ativo e com o atuador inativo, cujos dados estatísticos são apresentados na Tab. 5.

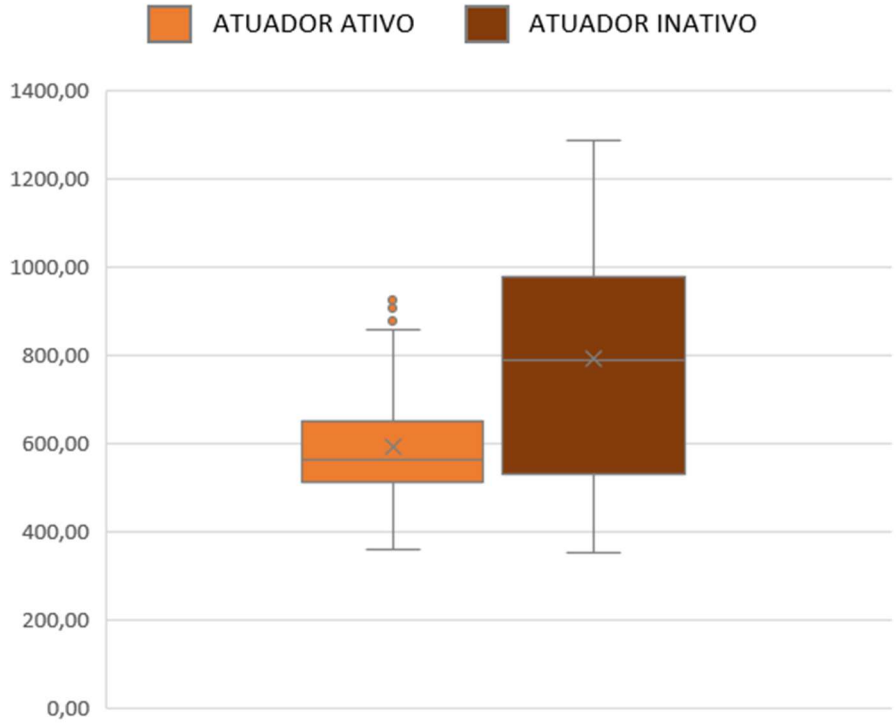


Fig. 52 - Box plots do registo de concentração de CO_2

Tab. 5 - Dados estatísticos da concentração de CO_2

Medida estatística	Atuador ativo	Atuador inativo
Mínimo	423,31	372,41
1º Quartil	518,80	581,79
Mediana	570,12	798,58
3º Quartil	652,58	1011,36
Máximo	937,48	1285,68
Média	608,15	825,65

4.2. DASHBOARD

Outro dos principais objetivos desta dissertação era a criação de um Dashboard que representasse o Digiroom. Assim, através dos processos apresentados no capítulo 3 foi possível chegar ao resultado que se segue. O Dashboard conta com 4 elementos visuais.

O primeiro, Fig. 53, apresenta o modelo 3D do Digiroom de forma dinâmica, sendo possível selecionar cada elemento individualmente, conta ainda com informações adicionais da sala, área e volume.

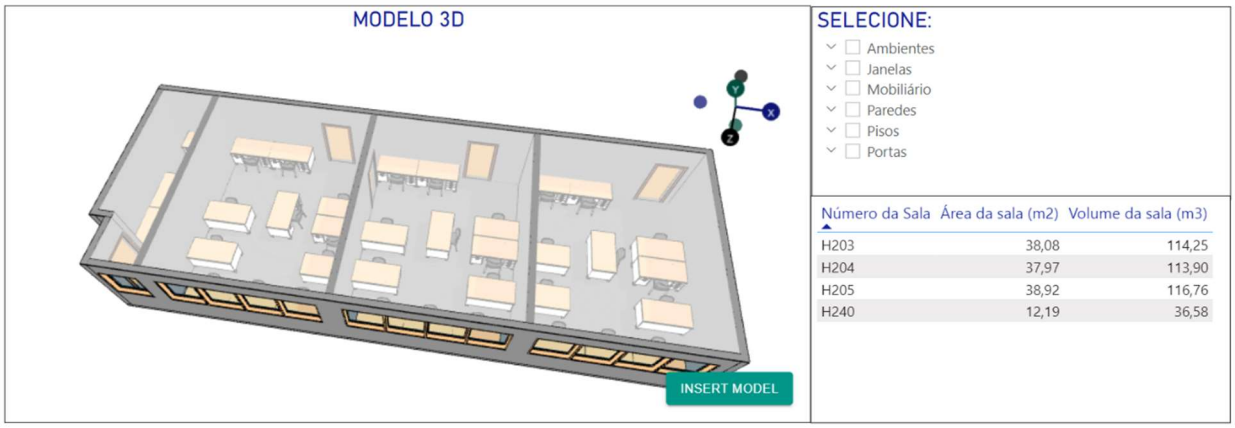


Fig. 53 – Dashboard - Modelo 3D e informações

O segundo elemento visual diz respeito aos dados em tempo real, onde é possível visualizar os últimos valores de todos os parâmetros medidos pelo sistema de monitorização, contém ainda a unidade de medida utilizada, Fig. 54. O segundo elemento visual diz respeito ao dados em tempo real, onde é possível visualizar os últimos valores de todos os parâmetros medidos pelo sistema de monitorização, contém ainda a unidade de medida utilizada, Fig. 54.



Fig. 54 – Dashboard - Dados em tempo real

De seguida surge o elemento visual que contém o registo histórico de todos os parâmetros, cada parâmetro é analisado com um indicador de valor médio, um gráfico de linhas e um gráfico de barras. No topo deste elemento visual surge o filtro temporal que permite o utilizador selecionar o intervalo a analisar, Fig. 55, Fig. 56.

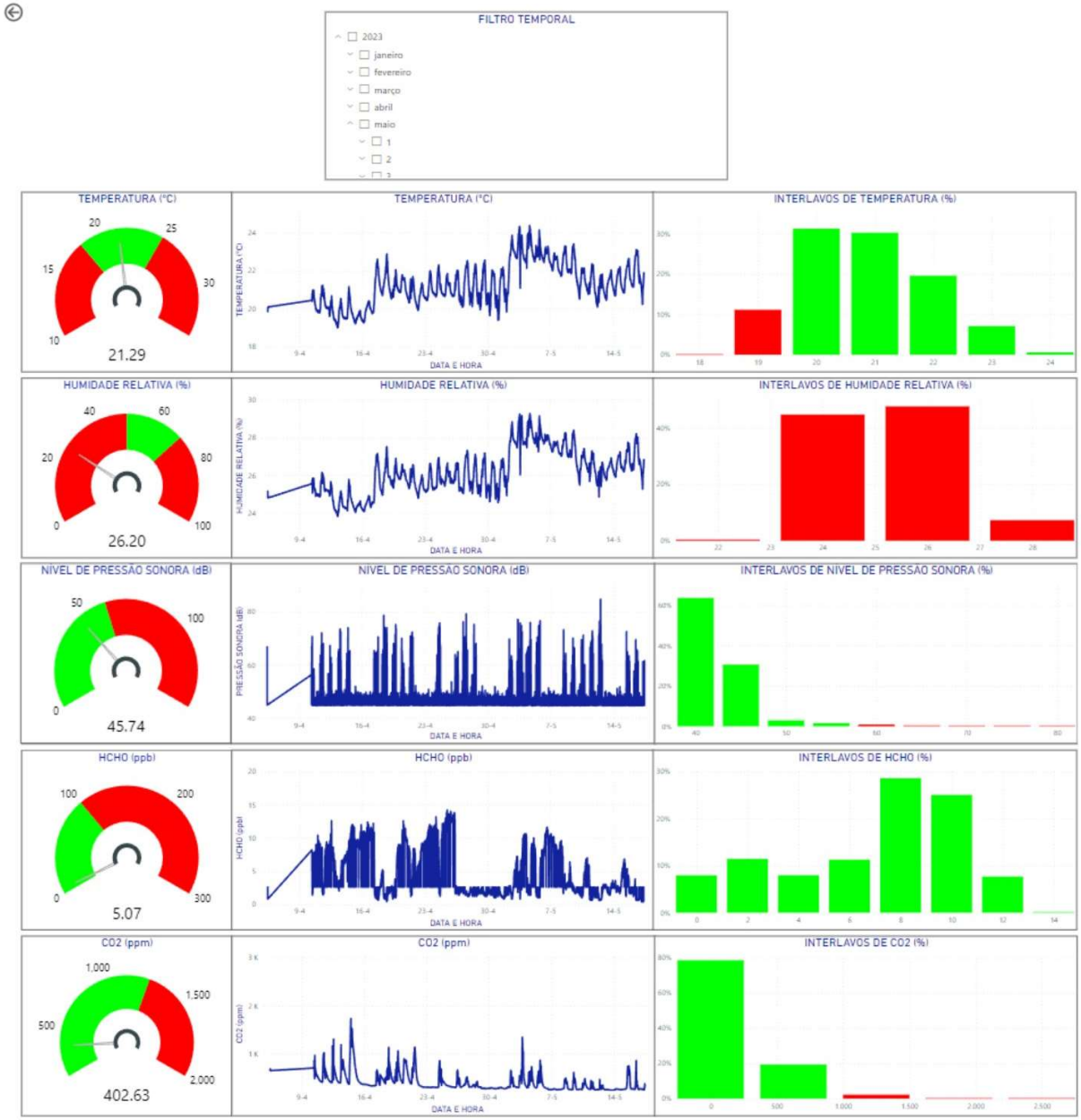


Fig. 55 – Dashboard - Dados históricos

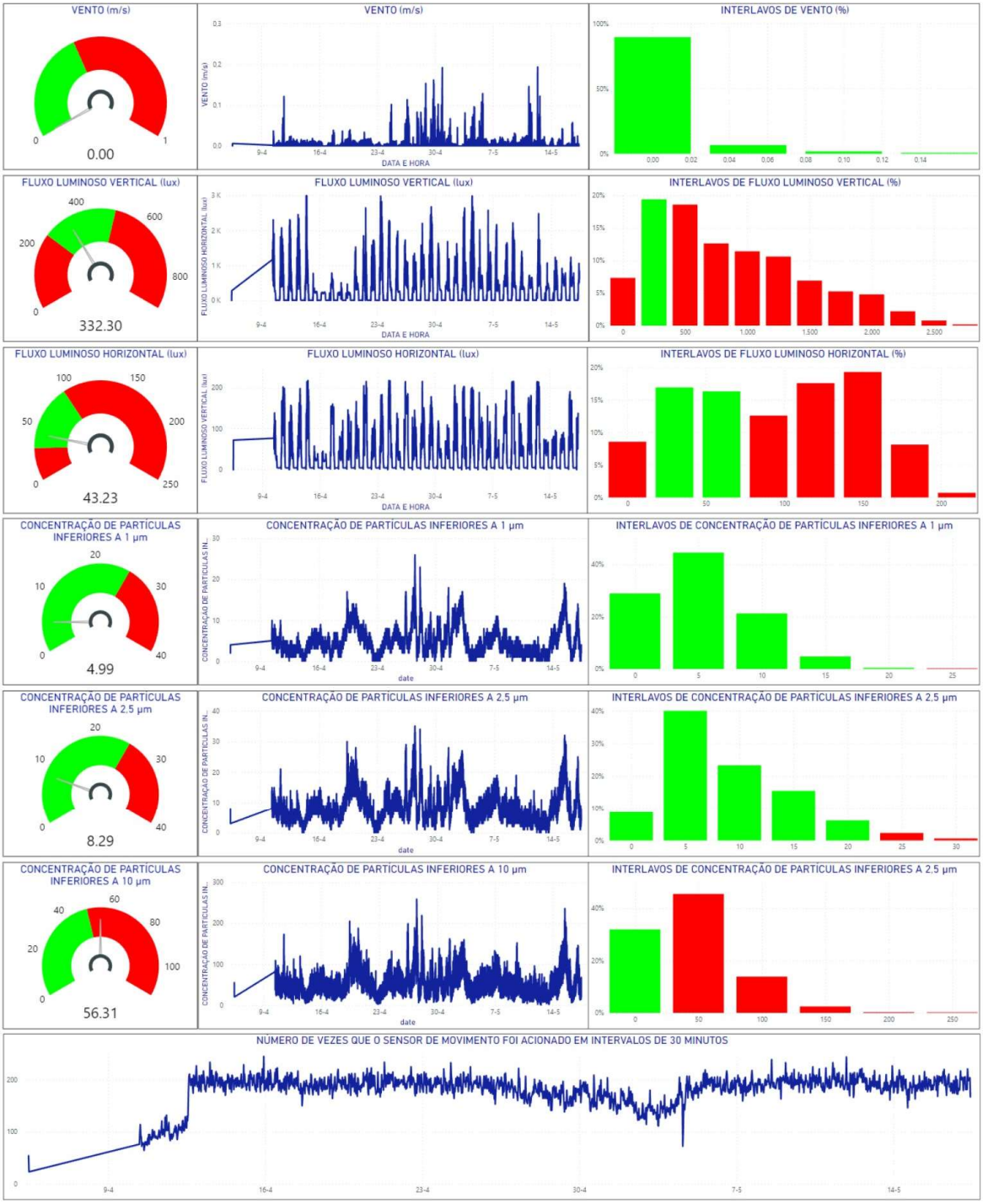


Fig. 56 – Dashboard - Dados históricos (continuação)

O quarto, e último elemento visual diz respeito aos dados estatísticos dos parâmetros medidos pelo sistema de monitorização, com exceção do parâmetro medido pelo sensor de movimento. Todos eles contam com 6 medidas estatisticas, já referidas no capitulo 3, Fig. 57.

TEMPERATURA (°C)					
18,98	20,54	21,18	22,03	24,37	21,29
MÍNIMO	1º QUARTIL	MEDIANA	3º QUARTIL	MÁXIMO	MÉDIA
HUMIDADE RELATIVA (%)					
23,83	25,33	26,13	27,01	29,27	26,20
MÍNIMO	1º QUARTIL	MEDIANA	3º QUARTIL	MÁXIMO	MÉDIA
NÍVEL DE PRESSÃO SONORA (dB)					
44,52	44,88	44,90	45,20	84,69	45,74
MÍNIMO	1º QUARTIL	MEDIANA	3º QUARTIL	MÁXIMO	MÉDIA
HCHO (ppb)					
0,40	1,80	3,40	8,60	14,20	5,07
MÍNIMO	1º QUARTIL	MEDIANA	3º QUARTIL	MÁXIMO	MÉDIA
CO2 (ppm)					
239,47	278,74	332,28	459,01	1.735,17	402,63
MÍNIMO	1º QUARTIL	MEDIANA	3º QUARTIL	MÁXIMO	MÉDIA
VENTO (m/s)					
0,00	0,00	0,00	0,00	0,19	0,00
MÍNIMO	1º QUARTIL	MEDIANA	3º QUARTIL	MÁXIMO	MÉDIA
FLUXO LUMINOSO VERTICAL (lux)					
0,00	3,15	155,72	490,79	2.995,84	332,30
MÍNIMO	1º QUARTIL	MEDIANA	3º QUARTIL	MÁXIMO	MÉDIA
FLUXO LUMINOSO HORIZONTAL (lux)					
0,00	3,66	19,56	66,59	217,71	43,23
MÍNIMO	1º QUARTIL	MEDIANA	3º QUARTIL	MÁXIMO	MÉDIA
CONCENTRAÇÃO DE PARTÍCULAS INFERIORES A 1 µm					
0,00	3,00	4,00	6,00	26,00	4,99
MÍNIMO	1º QUARTIL	MEDIANA	3º QUARTIL	MÁXIMO	MÉDIA
CONCENTRAÇÃO DE PARTÍCULAS INFERIORES A 2,5 µm					
0,00	5,00	7,00	10,00	35,00	8,29
MÍNIMO	1º QUARTIL	MEDIANA	3º QUARTIL	MÁXIMO	MÉDIA
CONCENTRAÇÃO DE PARTÍCULAS INFERIORES A 10 µm					
0,00	33,00	47,00	66,00	259,00	56,31
MIN PM10	1º QUARTIL	MEDIANA	3º QUARTIL	MÁXIMO	MÉDIA

Fig. 57 – Dashboard - Dados estatísticos

5. CONCLUSÕES

5.1. CONCLUSÕES FINAIS

Concluído este trabalho, pode-se afirmar que os objetivos iniciais propostos foram alcançados. Após um estudo aprofundado sobre os Digital Building Twins, mediante uma análise detalhada de suas características, funcionalidades e aplicações, foi possível adquirir conhecimento teórico sobre o assunto. Além disso, por meio da exploração de exemplos práticos e casos de estudo relevantes, foi possível obter uma compreensão mais concreta sobre as possibilidades e benefícios desta tecnologia inovadora.

Foi possível aplicar de forma efetiva os conceitos e princípios dos Digital Building Twins no caso de estudo. Através da aplicação dos conhecimentos adquiridos, foi possível desenvolver com sucesso o modelo Digital Building Twin do Digiroom, enfrentando desafios e ultrapassando barreiras de domínio tecnológico.

Para a criação do Dashboard o Power BI foi uma ferramenta muito útil, demonstrando a sua versatilidade e abrangência. A capacidade de realizar várias etapas do processo simplificou a criação e apresentação do Dashboard final. Através das suas funcionalidades foi possível realizar a limpeza, organização e transformação dos dados de forma eficiente e precisa para posterior tratamento. A capacidade de introduzir o modelo 3D no Dashboard acrescentou valor à apresentação dos dados, oferecendo uma perspetiva visual mais interativa. Assim, juntamente com os indicadores, gráficos e registos estatísticos foi possível obter um Dashboard bastante dinâmico, permitindo desde a representação do espaço físico ao tratamento dos dados recolhidos pelo sistema de monitorização.

Através do Mosquitto e do Python é possível realizar várias ações, combinando diferentes parâmetros e desenvolvendo códigos que façam ativar atuadores que estejam conectados, tal como foi utilizado para o controlo do atuador de janela através de 2 parâmetros escolhidos, a pressão parcial do vapor de água e a concentração de CO_2. O Mosquitto e o Python complementam-se, demonstrando uma funcionalidade sólida quando utilizados em conjunto. Esta combinação permite o desenvolvimento de soluções avançadas e personalizadas, fornecendo os recursos necessários para automatizar e controlar sistemas em tempo real.

Através da implementação de um atuador de janela, foi possível atingir plenamente outro dos objetivos estabelecidos. Este dispositivo demonstrou um desempenho notável. A comparação dos resultados entre a utilização e a não utilização do atuador de janela evidencia a relevância do emprego deste dispositivo para diminuir a concentração de CO_2. A abertura controlada da janela mostrou-se essencial para melhorar a qualidade do ambiente interior, proporcionando uma ventilação adequada que levou à redução da concentração de CO_2. Estes dados espelham a eficácia do sistema automatizado na manutenção de um ambiente saudável e confortável.

5.2. DESENVOLVIMENTOS FUTUROS

Em relação aos possíveis desenvolvimentos futuros do tema abordado, podem ser consideradas algumas sugestões.

Uma delas é a combinação de diferentes parâmetros além dos estudados, adotando uma abordagem voltada para a eficiência energética. Essa análise proporcionaria uma compreensão diferente à estudada, permitindo o desenvolvimento de estratégias mais eficientes da gestão dos edifícios.

Outra sugestão seria a melhoria do Dashboard. É importante investir no aperfeiçoamento da interface do utilizador, tornando-a ainda mais dinâmica e com recursos avançados de visualização e análise de dados. Isso possibilitaria aos utilizadores acompanhar e interpretar as informações recolhidas de forma mais eficiente.

Além disso, a gestão da base de dados também pode ser otimizada para garantir uploads mais rápidos e eficientes. Isso pode envolver a adoção de softwares alternativos para reduzir o tempo de transferência e armazenamento dos dados recolhidos. A implementação de um sistema seguro de gestão de dados é crucial para preservar a integridade das informações e permitir uma análise precisa e contínua.

Por fim, é recomendável considerar a automação de outros dispositivos na sala, bem como a instalação de atuadores em todas as janelas. Essa automação permitiria um controle mais preciso e eficiente dos parâmetros ambientais e seria uma alavanca ao estudo da eficiência energética, de forma a permitir o ajuste automático da ventilação, iluminação e outros sistemas.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] D. G. J. Opoku, S. Perera, R. Osei-Kyei, M. Rashidi, T. Famakinwa, and K. Bamdad, "Drivers for Digital Twin Adoption in the Construction Industry: A Systematic Literature Review," *Buildings*, vol. 12, no. 2. MDPI, Feb. 01, 2022. doi: 10.3390/buildings12020113.

[2] "Digital Twin: Manufacturing Excellence through Virtual Factory Replication DFAM-Design for Additive Manufacturing and Additive Manufacturing evaluation View project Digital Twin View project Michael Grieves Digital Twin Institute Digital Twin: Manufacturing Excellence through Virtual Factory Replication." [Online]. Available: https://www.researchgate.net/publication/275211047

[3] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, Springer International Publishing, 2016, pp. 85–113. doi: 10.1007/978-3-319-38756-7_4.

[4] M. Nour, P. F. Pereira, J. Poç As Martins, and N. M. M. Ramos, "Article Digital Twins for Construction Assets using BIM Standard 2 Specifications 3," *Buildings 2022*, vol. 12, 2022, doi: 10.3390/xxxxx.

[5] M. Bolpagni, R. Gavina, and D. Ribeiro, Eds., *Industry 4.0 for the Built Environment*, vol. 20. in Structural Integrity, vol. 20. Cham: Springer International Publishing, 2022. doi: 10.1007/978-3-030-82430-3.

[6] S. H. Khajavi, N. H. Motlagh, A. Jaribion, L. C. Werner, and J. Holmstrom, "Digital Twin: Vision, benefits, boundaries, and creation for buildings," *IEEE Access*, vol. 7, pp. 147406–147419, 2019, doi: 10.1109/ACCESS.2019.2946515.

[7] Z. Liu, A. Zhang, and W. Wang, "A framework for an indoor safety management system based on digital twin," *Sensors (Switzerland)*, vol. 20, no. 20, pp. 1–20, Oct. 2020, doi: 10.3390/s20205771.

[8] M. Nour El-Din, P. F. Pereira, J. Poças Martins, and N. M. M. Ramos, "Digital Twins for Construction Assets Using BIM Standard Specifications," *Buildings*, vol. 12, no. 12, Dec. 2022, doi: 10.3390/buildings12122155.

[9] "SpHERE BIM DIGITAL TWIN PLATFORM DIGITAL TWIN DEFINITIONS FOR BUILDINGS How to move digital twin environments to the AECOO sector." [Online]. Available: www.sphere-project.eu

[10] S. H. Khajavi, N. H. Motlagh, A. Jaribion, L. C. Werner, and J. Holmstrom, "Digital Twin: Vision, benefits, boundaries, and creation for buildings," *IEEE Access*, vol. 7, pp. 147406–147419, 2019, doi: 10.1109/ACCESS.2019.2946515.

[11] Z. Ni, Y. Liu, M. Karlsson, and S. Gong, "Enabling Preventive Conservation of Historic Buildings Through Cloud-Based Digital Twins: A Case Study in the City Theatre, Norrköping," *IEEE Access*, vol. 10, pp. 90924–90939, 2022, doi: 10.1109/ACCESS.2022.3202181.

[12] "Microsoft Azure." https://azure.microsoft.com/pt-pt/free/virtual-network/search/?ef_id=_k_CjwKCAjwyeujBhA5EiwA5WD7_fZ-

ucKGqD_49ATPjxoDBEtV0bjRhJbxyeKtG-iqbskagdNlSi9nRxoC3ZgQAvD_BwE_k_&OCID=AIDcmmj8drqamm_SEM__k_CjwKCAjwyeujBhA5EiwA5WD7_fZ-ucKGqD_49ATPjxoDBEtV0bjRhJbxyeKtG-iqbskagdNlSi9nRxoC3ZgQAvD_BwE_k_&gclid=CjwKCAjwyeujBhA5EiwA5WD7_fZ-ucKGqD_49ATPjxoDBEtV0bjRhJbxyeKtG-iqbskagdNlSi9nRxoC3ZgQAvD_BwE (accessed Jun. 03, 2023).

[13] T. Evangelou, M. Gkeli, and C. Potsiou, "Building Digital Twins for Smart Cities: A Case Study in Greece," in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Copernicus Publications, Oct. 2022, pp. 61–68. doi: 10.5194/isprs-annals-X-4-W2-2022-61-2022.

[14] E. Seghezzi *et al.*, "Towards an occupancy-oriented digital twin for facility management: Test campaign and sensors assessment," *Applied Sciences (Switzerland)*, vol. 11, no. 7, Apr. 2021, doi: 10.3390/app11073108.

[15] I. Merini, A. Molina-García, M. Socorro García-Cascales, M. Mahdaoui, and M. Ahachad, "Analysis and comparison of energy efficiency code requirements for buildings: A Morocco-Spain case study," *Energies (Basel)*, vol. 13, no. 22, Nov. 2020, doi: 10.3390/en13225979.

[16] "Commission welcomes agreement on energy performance of buildings," 2017.

[17] H. H. Hosamo, P. R. Svennevig, K. Svidt, D. Han, and H. K. Nielsen, "A Digital Twin predictive maintenance framework of air handling units based on automatic fault detection and diagnostics," *Energy Build*, vol. 261, Apr. 2022, doi: 10.1016/j.enbuild.2022.111988.

[18] H. Seo and W. S. Yun, "Digital Twin-Based Assessment Framework for Energy Savings in University Classroom Lighting," *Buildings*, vol. 12, no. 5, May 2022, doi: 10.3390/buildings12050544.

[19] W. Wei, P. Wargocki, J. Zirngibl, J. Bendžalová, and C. Mandin, "Review of parameters used to assess the quality of the indoor environment in Green Building certification schemes for offices and hotels," *Energy and Buildings*, vol. 209. Elsevier Ltd, Feb. 15, 2020. doi: 10.1016/j.enbuild.2019.109683.

[20] M. Frontczak, S. Schiavon, J. Goins, E. Arens, H. Zhang, and P. Wargocki, "Quantitative relationships between occupant satisfaction and satisfaction aspects of indoor environmental quality and building design," *Indoor Air*, vol. 22, no. 2, pp. 119–131, Apr. 2012.

[21] I. A. Sakellaris *et al.*, "Perceived indoor environment and occupants' comfort in European 'Modern' office buildings: The OFFICAIR Study," *Int J Environ Res Public Health*, vol. 13, no. 5, Apr. 2016, doi: 10.3390/ijerph13050444.

[22] M. A. Passarelli, "QAI e Eficiência Energética. O dilema da concentração de CO 2 em edifícios."

[23] "ASHRAE55-version2017".

[24] A. Augusto Xavier Solange Goulart Renata De Vecchi Professor Roberto Lamberts, "UNIVERSIDADE FEDERAL DE SANTA CATARINA CENTRO

TECNOLÓGICO-DEPARTAMENTO DE ENGENHARIA CIVIL CONFORTO E STRESS TÉRMICO.” [Online]. Available: www.labeee.ufsc.br

[25] D. Kazado, M. Kavgic, and R. Eskicioglu, “Integrating Building Information Modeling (BIM) and sensor technology for Facility Management,” 2019. [Online]. Available: http://www.itcon.org/2019/23

[26] M. M. Abdelrahman, A. Chong, and C. Miller, “Personal thermal comfort models using digital twins: Preference prediction with BIM-extracted spatial–temporal proximity data from Build2Vec,” *Build Environ*, vol. 207, Jan. 2022, doi: 10.1016/j.buildenv.2021.108532.

[27] D. Filipe, C. Ribeiro, D. António, and P. Oliveira De Carvalho, “REABILITAÇÃO ACÚSTICA EM CANTINAS ESCOLARES CASO DE ESTUDO: CANTINA DA FEUP,” 2017.

[28] Y. Al horr, M. Arif, M. Katafygiotou, A. Mazroei, A. Kaushik, and E. Elsarrag, “Impact of indoor environmental quality on occupant well-being and comfort: A review of the literature,” *International Journal of Sustainable Built Environment*, vol. 5, no. 1. Elsevier B.V., pp. 1–11, 2016. doi: 10.1016/j.ijsbe.2016.03.006.

[29] A. Zaballos, A. Briones, A. Massa, P. Centelles, and V. Caballero, “A smart campus’ digital twin for sustainable comfort monitoring,” *Sustainability (Switzerland)*, vol. 12, no. 21, pp. 1–33, Nov. 2020, doi: 10.3390/su12219196.

[30] T. Baptista, M. Almeida-Silva, D. Silva, C. Diogo, and N. Canha, “Indoor Air Quality Assessment in Grocery Stores,” *Applied Sciences (Switzerland)*, vol. 12, no. 24, Dec. 2022, doi: 10.3390/app122412940.

[31] P. F. Pereira, N. M. M. Ramos, and A. Ferreira, “Room-scale analysis of spatial and human factors affecting indoor environmental quality in Porto residential flats,” *Build Environ*, vol. 186, Dec. 2020, doi: 10.1016/j.buildenv.2020.107376.

[32] L. C. Tagliabue, F. R. Cecconi, S. Maltese, S. Rinaldi, A. L. C. Ciribini, and A. Flammini, “Leveraging digital twin for sustainability assessment of an educational building,” *Sustainability (Switzerland)*, vol. 13, no. 2, pp. 1–16, Jan. 2021, doi: 10.3390/su13020480.

[33] “Revit.” https://www.autodesk.pt/products/revit/overview?term=1-YEAR&tab=subscription (accessed Jun. 07, 2023).

[34] “3DBI for Revit.” https://kg-dev.be/project/3dbi-for-revit/ (accessed Jun. 07, 2023).

[35] “Dynamo”, Accessed: Jun. 07, 2023. [Online]. Available: https://help.autodesk.com/view/RVT/2024/PTB/?guid=RevitDynamo_Dynamo_for_Revit_html

[36] “Excel”, Accessed: Jun. 07, 2023. [Online]. Available: https://www.microsoft.com/pt-pt/microsoft-365/excel

[37] “Maria DB”, Accessed: Jun. 07, 2023. [Online]. Available: https://mariadb.org/

[38] “Python”, Accessed: Jun. 07, 2023. [Online]. Available: https://www.python.org/

[39] “Microsoft Power BI Tutorial for Beginners from Scratch.” https://www.edureka.co/blog/power-bi-tutorial/ (accessed Apr. 20, 2023).

[40] "Mosquitto", Accessed: Jun. 08, 2023. [Online]. Available: https://mosquitto.org/

[41] "Decreto-Lei nº 243/86 de 20 de Agosto. Regulamento Geral de Higiene e Segurança do Trabalho nos Estabelecimentos Comerciais, de Escritório e Serviços".

[42] "Decreto-Lei n.º 129/2002 de 11 de Maio. Regulamento dos requisitos acústicos dos edifícios".

[43] "Portaria n.º 138-G/2021. Diário da República n.º 126/2021 Série I. Saúde e ambiente e ação climática".

[44] R. Golden, "Identifying an indoor air exposure limit for formaldehyde considering both irritation and cancer hazards," *Critical Reviews in Toxicology*, vol. 41, no. 8. pp. 672–721, Sep. 2011. doi: 10.3109/10408444.2011.573467.

[45] V. Peixoto Freitas and Silva Pinto Paulo, "LFC - NIT 002: Permeabilidade ao vapor de materiais de construção - Condensações internas." FEUP, Porto, Portugal, 1998.

ANEXOS

ANEXO 1 – PYTHON - CÓDIGO PARA ABRIR JANELA

```
import paho.mqtt.client as mqtt
import time
import re
import json
import traceback

mqtt_broker = 'epbase.local'    # MQTT broker address (hostname or IP)
# mqtt_broker = '192.168.32.251'  # MQTT broker address (hostname or IP)
mqtt_port = 1883                # MQTT broker port
mqtt_ttl = 60                   # MQTT time to live (s)

#############
## devices ##
#############
ieq = 'e831cdd4da7c'
heater = 'e831cdd4d780'
shelly_3dprinter = '3ce90ec81020'
shelly_torno = '3ce90ec808d9'
shelly_cnc = '3ce90ec86cf2'
shelly_serra_circular = 'c8c9a3a4eb39'
shelly_serra_fita = '3ce90ec83ea5'
window = '3494547935f4'
#############

########################
## IEQ output values ##
########################
lux_h = bmetmp = atmpress = bemrh = gasres = temperature = rh = hoco = sfarh = sfatmp\
    = co2 = pm1 = pm25 = pm10 = pm1_atm = pm25_atm = cuatm = pnr0p3 = pnr0p5\
    = pnr1p0 = pnr2p5 = pnr5p0 = pnr10 = mv = spdb = lux_v = wind = None
#########################
## Shelly output values ##
#########################
power_3dprint = power_torno = power_cnc = power_serra_circular = power_serra_fita =
power_window\
              = temperature_3dprint = temperature_torno = temperature_cnc =
              temperature_serra_circular\
              = temperature_serra_fita = temperature_window = state_3dprint =
              state_torno = state_cnc\
              = state_serra_circular = state_serra_fita = state_window = None
#########################
## Heater output values ##
#########################
heater_state = heater_in_temp = heater_return_temp = heater_flux = None
#########################

####################
## Window commands ##
####################
cmd_window_open = '{"command":"roller/0?go=open"}'      # Open window
cmd_window_close = '{"command":"roller/0?go=close"}'    # Close window
cmd_window_stop = '{"command":"roller/0?go=stop"}'      # Stop window movement
window_cmd_response = {}    # stored device response
####################
## Heater commands ##
####################
cmd_heater_open = '{"command":"set_state","args":[1],"senId":3}'    # Start opening
the heater valve
cmd_heater_close = '{"command":"set_state","args":[0],"senId":3}'   # Start closing
the heater valve
heater_cmd_response = {}    # stored device response
####################

def send_cmd_to_dev(mac_addr, cmd):
    global window_cmd_response
    global heater_cmd_response

    mqtt_client.publish("EP/fazesse/" + re.sub('\W+', '',
                        mac_addr), payload=cmd, retain=False)
    if mac_addr == window:
        window_cmd_response["command"]=cmd
        window_cmd_response["response"]=None
```

```
68      elif mac_addr == heater:
69          heater_cmd_response["command"]=cmd
70          heater_cmd_response["response"]=None
71      else:
72          print("Device "+mac_addr+" is does not accept commands")
73
74  def mqtt_init():
75      global mqtt_client
76      mqtt_client = mqtt.Client()
77      try:
78          mqtt_client.connect(mqtt_broker, mqtt_port, mqtt_ttl)
79          mqtt_client.on_connect = on_connect
80          mqtt_client.on_disconnect = on_disconnect
81          mqtt_client.loop_start()
82
83          mqtt_client.message_callback_add(
84              'EP/measurements/+', on_measurement)
85          mqtt_client.message_callback_add(
86              'EP/Shelly/measurements/+', on_measurement)
87          mqtt_client.message_callback_add(
88              'EP/ack/+', on_ack)
89      except Exception as e:
90          print("Error mqtt_client.connect\nerror: "+str(e))
91          # traceback.print_exc()
92          time.sleep(5)
93          mqtt_init()
94          return
95      print('MQTT Connected')
96
97  def check_timestamp_us(ts):
98      test = (ts > minimum_timestamp_us and ts <
99              (time.time()*1000000+60*1000000))
100     if not test:
101         printd('Invalid timestamp!')
102     return test
103
104 def on_connect(client, userdata, flags, rc):
105     print("Connected with result code " + str(rc))
106     if rc == 0:
107         print("MQTT client connected OK")
108         client.subscribe("EP/#", qos=2)  # get all MQTT messages
109
110 def on_disconnect(client, userdata, flags):
111     print("MQTT client disconnected!")
112
113 def on_ack(client, userdata, msg):
114     try:
115         devId = msg.topic.split("/")[-1]
116         ack_msg_str=msg.payload.decode()
117         ack_msg = json.loads(ack_msg_str)
118         if "mqttid" in ack_msg:
119             print("MQTT message ID: ", ack_msg["mqttid"])
120     except Exception as e:
121         print('Error parsing ack message!\nerror: '+str(e))
122         print('MQTT message: ', msg.payload)
123         return
124     print("Device "+devId+" ACK:")
125     print("\t"+ack_msg_str)
126     #### TAKE ACTION ON DEVICE RESPONSE ####
127
128 def on_measurement(client, userdata, msg):
129     ######################################################
130     ##                  IEQ variables                   ##
131     ######################################################
132     global lux_h        # Lux horizontal (lux)
133     global bmetmp       # Internal temperature (□)
134     global atmpress     # Atmosferic pressure (Pa)
135     global bemrh        # Internal relative humidity (%)
136     global gasres       # Gas resistance (Ω)
137     global temperature  # Ambient temperature (□)
138     global rh           # Ambient relative humidity (%)
139     global hoco         # Formaldehyde (HOCO)
```

```
    global sfarh        # Internal relative humidity (%)
    global sfatmp       # Internal temperature (□)
    global co2          # Carbon dioxide (CO2)
    global pm1          # PM 1.0 µm con unit
    global pm25         # PM 2.5 µm con unit
    global pm10         # PM 10 µm con unit
    global pm1_atm      # PM 1.0 µm con unit atmosphere
    global pm25_atm     # PM 2.5 µm con unit atmosphere
    global cuatm        # Con unit atmosphere
    global pnr0p3       # Particle number 0.3 µm
    global pnr0p5       # Particle number 0.5 µm
    global pnr1p0       # Particle number 1.0 µm
    global pnr2p5       # Particle number 2.5 µm
    global pnr5p0       # Particle number 5.0 µm
    global pnr10        # Particle number 10 µm
    global mv           # Movement (true/false)
    global spdb         # Sound pressure (Db)
    global lux_v        # Lux vertical (lux)
    global wind         # Wind speed (m/s)
    ############################################################
    ##                    Heater variables                    ##
    ############################################################
    global heater_state         # Heater state (on/off)
    global heater_in_temp       # Water input temperature (□)
    global heater_return_temp   # Water return temperature (□)
    global heater_flux          # Water flux (L/m)
    ############################################################
    ##                   Window variables                     ##
    ############################################################
    global state_window         # Window state (open/closed)
    global power_window         # Window power (Watt)
    global temperature_window   # Window controler temperature (□)
    ############################################################################
    ##                      Shelly plug variables                             ##
    ############################################################################
    global state_3dprint                # 3D printer state (on/off)
    global power_3dprint                # 3D printer power (Watt)
    global temperature_3dprint          # 3D printer selly plug temperature (□)
    global state_torno                  # Torno state (on/off)
    global power_torno                  # Torno power (Watt)
    global temperature_torno            # Torno selly plug temperature (□)
    global state_cnc                    # CNC state (on/off)
    global power_cnc                    # CNC power (Watt)
    global temperature_cnc              # CNC selly plug temperature (□)
    global state_serra_circular         # Serra circular state (on/off)
    global power_serra_circular         # Serra circular power (Watt)
    global temperature_serra_circular   # Serra circular selly plug temperature (□)
    global state_serra_fita             # Serra de fita state (on/off)
    global power_serra_fita             # Serra de fita power (Watt)
    global temperature_serra_fita       # Serra de fita selly plug temperature (□)

    devId = msg.topic.split("/")[-1]
    # print('Received measurement from device ', devId)
    try:
        measurement = json.loads(msg.payload.decode())
    except Exception as e:
        print('Error parsing values!\nerror: '+str(e))
        print('MQTT message: ', msg.payload)
        return
    if not "sensors" in measurement:
        print('Invalid measurement. Missing sensors')
        return
    for sen in measurement["sensors"]:
        if not "senId" in sen:
            print('Invalid measurement. Missing sensor ID')
            return
        if not "outs" in sen or len(sen["outs"]) == 0:
            print('Invalid measurement. Missing sensor outs')
            return
        try:
            ####    IEQ data process    ####
            if devId == ieq:
```

```
212                 if sen["senId"] == 0:
213                     if not "processed" in sen["outs"][0]:
214                         print('Error: missing processed value on measurement!')
215                         return
216                     lux_h=sen["outs"][0]["processed"]
217                 elif sen["senId"] == 1:
218                     if not "processed" in sen["outs"][0]:
219                         print('Error: missing processed value on measurement!')
220                         return
221                     bmetmp=sen["outs"][0]["processed"]
222                     atmpress=sen["outs"][1]["processed"]
223                     bemrh=sen["outs"][2]["processed"]
224                     gasres=sen["outs"][3]["processed"]
225                 elif sen["senId"] == 2:
226                     if not "processed" in sen["outs"][0]:
227                         print('Error: missing processed value on measurement!')
228                         return
229                     temperature=sen["outs"][0]["processed"]
230                     rh=sen["outs"][1]["processed"]
231                 elif sen["senId"] == 3:
232                     if not "processed" in sen["outs"][0]:
233                         print('Error: missing processed value on measurement!')
234                         return
235                     hoco=sen["outs"][0]["processed"]
236                     sfarh=sen["outs"][1]["processed"]
237                     sfatmp=sen["outs"][2]["processed"]
238                 elif sen["senId"] == 4:
239                     if not "processed" in sen["outs"][0]:
240                         print('Error: missing processed value on measurement!')
241                         return
242                     co2=sen["outs"][0]["processed"]
243                 elif sen["senId"] == 5:
244                     if not "processed" in sen["outs"][0]:
245                         print('Error: missing processed value on measurement!')
246                         return
247                     pm1=sen["outs"][0]["processed"]
248                     pm25=sen["outs"][1]["processed"]
249                     pm10=sen["outs"][2]["processed"]
250                     pm1_atm=sen["outs"][3]["processed"]
251                     pm25_atm=sen["outs"][4]["processed"]
252                     cuatm=sen["outs"][5]["processed"]
253                     pnr0p3=sen["outs"][6]["processed"]
254                     pnr0p5=sen["outs"][7]["processed"]
255                     pnr1p0=sen["outs"][8]["processed"]
256                     pnr2p5=sen["outs"][9]["processed"]
257                     pnr5p0=sen["outs"][10]["processed"]
258                     pnr10=sen["outs"][11]["processed"]
259                 elif sen["senId"] == 6:
260                     if not "processed" in sen["outs"][0]:
261                         print('Error: missing processed value on measurement!')
262                         return
263                     mv=sen["outs"][0]["processed"]
264                 elif sen["senId"] == 7:
265                     if not "processed" in sen["outs"][0]:
266                         print('Error: missing processed value on measurement!')
267                         return
268                     spdb=sen["outs"][0]["processed"]
269                 elif sen["senId"] == 8:
270                     if not "processed" in sen["outs"][0]:
271                         print('Error: missing processed value on measurement!')
272                         return
273                     lux_v=sen["outs"][0]["processed"]
274                 elif sen["senId"] == 9:
275                     if not "processed" in sen["outs"][0]:
276                         print('Error: missing processed value on measurement!')
277                         return
278                     wind=sen["outs"][0]["processed"]
279             ####    Window data process    ####
280             elif devId == window:
281                 if sen["senId"] == 0:
282                     power_window = sen["outs"][-1]["processed"]
283                 elif sen["senId"] == 2:
```

```
                    temperature_window = sen["outs"][-1]["processed"]
            ####    3D printer shelly data process    ####
            elif devId == shelly_3dprinter:
                if sen["senId"] == -1:
                    power_3dprint = sen["outs"][0]["processed"]
                elif sen["senId"] == 0:
                    temperature_3dprint = sen["outs"][0]["processed"]
                elif sen["senId"] == 1:
                    state_3dprint = sen["outs"][0]["processed"]
            ####    CNC shelly data process    ####
            elif devId == shelly_cnc:
                if sen["senId"] == -1:
                    power_cnc = sen["outs"][-1]["processed"]
                elif sen["senId"] == 0:
                    temperature_cnc = sen["outs"][-1]["processed"]
                elif sen["senId"] == 1:
                    state_cnc = sen["outs"][-1]["processed"]
            ####    Serra de fita shelly data process    ####
            elif devId == shelly_serra_fita:
                if sen["senId"] == -1:
                    power_serra_fita = sen["outs"][-1]["processed"]
                elif sen["senId"] == 0:
                    temperature_serra_fita = sen["outs"][-1]["processed"]
                elif sen["senId"] == 1:
                    state_serra_fita = sen["outs"][-1]["processed"]
            ####    Serra circular shelly data process    ####
            elif devId == shelly_serra_circular:
                if sen["senId"] == -1:
                    power_serra_circular = sen["outs"][-1]["processed"]
                elif sen["senId"] == 0:
                    temperature_serra_circular = sen["outs"][-1]["processed"]
                elif sen["senId"] == 1:
                    state_serra_circular = sen["outs"][-1]["processed"]
            ####    Torno shelly data process    ####
            elif devId == shelly_torno:
                if sen["senId"] == -1:
                    power_torno = sen["outs"][-1]["processed"]
                elif sen["senId"] == 0:
                    temperature_torno = sen["outs"][-1]["processed"]
                elif sen["senId"] == 1:
                    state_torno = sen["outs"][-1]["processed"]
        except Exception as e:
            print('Error processing measurement\nerror: ', e)
            traceback.print_exc()
            return


if __name__ == '__main__':
    print("Script is starting...")
    mqtt_init()

    while True:
        try:
            send_cmd_to_dev(window, cmd_window_open)
        except Exception as e:
            print('Error processing main code!\nerror: ', e)
            traceback.print_exc()
            exit()
        time.sleep(1)  # Sensor data check interval (s)
        exit()

```

ANEXO 2 – PYTHON - CÓDIGO PARA FECHAR JANELA

```
import paho.mqtt.client as mqtt
import time
import re
import json
import traceback

mqtt_broker = 'epbase.local'    # MQTT broker address (hostname or IP)
# mqtt_broker = '192.168.32.251'  # MQTT broker address (hostname or IP)
mqtt_port = 1883                # MQTT broker port
mqtt_ttl = 60                   # MQTT time to live (s)

#############
## devices ##
#############
ieq = 'e831cdd4da7c'
heater = 'e831cdd4d780'
shelly_3dprinter = '3ce90ec81020'
shelly_torno = '3ce90ec808d9'
shelly_cnc = '3ce90ec86cf2'
shelly_serra_circular = 'c8c9a3a4eb39'
shelly_serra_fita = '3ce90ec83ea5'
window = '3494547935f4'
#############

########################
## IEQ output values ##
########################
lux_h = bmetmp = atmpress = bemrh = gasres = temperature = rh = hoco = sfarh = sfatmp\
    = co2 = pm1 = pm25 = pm10 = pm1_atm = pm25_atm = cuatm = pnr0p3 = pnr0p5\
    = pnr1p0 = pnr2p5 = pnr5p0 = pnr10 = mv = spdb = lux_v = wind = None
#########################
## Shelly output values ##
#########################
power_3dprint = power_torno = power_cnc = power_serra_circular = power_serra_fita =
power_window\
              = temperature_3dprint = temperature_torno = temperature_cnc =
              temperature_serra_circular\
              = temperature_serra_fita = temperature_window = state_3dprint =
              state_torno = state_cnc\
              = state_serra_circular = state_serra_fita = state_window = None
#########################
## Heater output values ##
#########################
heater_state = heater_in_temp = heater_return_temp = heater_flux = None
#########################

####################
## Window commands ##
####################
cmd_window_open = '{"command":"roller/0?go=open"}'      # Open window
cmd_window_close = '{"command":"roller/0?go=close"}'    # Close window
cmd_window_stop = '{"command":"roller/0?go=stop"}'      # Stop window movement
window_cmd_response = {}    # stored device response
####################
## Heater commands ##
####################
cmd_heater_open = '{"command":"set_state","args":[1],"senId":3}'    # Start opening
the heater valve
cmd_heater_close = '{"command":"set_state","args":[0],"senId":3}'   # Start closing
the heater valve
heater_cmd_response = {}    # stored device response
####################

def send_cmd_to_dev(mac_addr, cmd):
    global window_cmd_response
    global heater_cmd_response

    mqtt_client.publish("EP/fazesse/" + re.sub('\W+', '',
                        mac_addr), payload=cmd, retain=False)
    if mac_addr == window:
        window_cmd_response["command"]=cmd
        window_cmd_response["response"]=None
```

```
    elif mac_addr == heater:
        heater_cmd_response["command"]=cmd
        heater_cmd_response["response"]=None
    else:
        print("Device "+mac_addr+" is does not accept commands")

def mqtt_init():
    global mqtt_client
    mqtt_client = mqtt.Client()
    try:
        mqtt_client.connect(mqtt_broker, mqtt_port, mqtt_ttl)
        mqtt_client.on_connect = on_connect
        mqtt_client.on_disconnect = on_disconnect
        mqtt_client.loop_start()

        mqtt_client.message_callback_add(
            'EP/measurements/+', on_measurement)
        mqtt_client.message_callback_add(
            'EP/Shelly/measurements/+', on_measurement)
        mqtt_client.message_callback_add(
            'EP/ack/+', on_ack)
    except Exception as e:
        print("Error mqtt_client.connect\nerror: "+str(e))
        # traceback.print_exc()
        time.sleep(5)
        mqtt_init()
        return
    print('MQTT Connected')

def check_timestamp_us(ts):
    test = (ts > minimum_timestamp_us and ts <
            (time.time()*1000000+60*1000000))
    if not test:
        printd('Invalid timestamp!')
    return test

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    if rc == 0:
        print("MQTT client connected OK")
        client.subscribe("EP/#", qos=2)  # get all MQTT messages

def on_disconnect(client, userdata, flags):
    print("MQTT client disconnected!")

def on_ack(client, userdata, msg):
    try:
        devId = msg.topic.split("/")[-1]
        ack_msg_str=msg.payload.decode()
        ack_msg = json.loads(ack_msg_str)
        if "mqttid" in ack_msg:
            print("MQTT message ID: ", ack_msg["mqttid"])
    except Exception as e:
        print('Error parsing ack message!\nerror: '+str(e))
        print('MQTT message: ', msg.payload)
        return
    print("Device "+devId+" ACK:")
    print("\t"+ack_msg_str)
    #### TAKE ACTION ON DEVICE RESPONSE ####

def on_measurement(client, userdata, msg):
    ##################################################
    ##                IEQ variables                 ##
    ##################################################
    global lux_h        # Lux horizontal (lux)
    global bmetmp       # Internal temperature (□)
    global atmpress     # Atmosferic pressure (Pa)
    global bemrh        # Internal relative humidity (%)
    global gasres       # Gas resistance (Ω)
    global temperature  # Ambient temperature (□)
    global rh           # Ambient relative humidity (%)
    global hoco         # Formaldehyde (HOCO)
```

```
        global sfarh        # Internal relative humidity (%)
        global sfatmp       # Internal temperature (□)
        global co2          # Carbon dioxide (CO2)
        global pm1          # PM 1.0 µm con unit
        global pm25         # PM 2.5 µm con unit
        global pm10         # PM 10 µm con unit
        global pm1_atm      # PM 1.0 µm con unit atmosphere
        global pm25_atm     # PM 2.5 µm con unit atmosphere
        global cuatm        # Con unit atmosphere
        global pnr0p3       # Particle number 0.3 µm
        global pnr0p5       # Particle number 0.5 µm
        global pnr1p0       # Particle number 1.0 µm
        global pnr2p5       # Particle number 2.5 µm
        global pnr5p0       # Particle number 5.0 µm
        global pnr10        # Particle number 10 µm
        global mv           # Movement (true/false)
        global spdb         # Sound pressure (Db)
        global lux_v        # Lux vertical (lux)
        global wind         # Wind speed (m/s)
        ############################################################
        ##                    Heater variables                    ##
        ############################################################
        global heater_state         # Heater state (on/off)
        global heater_in_temp       # Water input temperature (□)
        global heater_return_temp   # Water return temperature (□)
        global heater_flux          # Water flux (L/m)
        ################################################################
        ##                       Window variables                     ##
        ################################################################
        global state_window         # Window state (open/closed)
        global power_window         # Window power (Watt)
        global temperature_window   # Window controler temperature (□)
        ########################################################################
        ##                      Shelly plug variables                         ##
        ########################################################################
        global state_3dprint                # 3D printer state (on/off)
        global power_3dprint                # 3D printer power (Watt)
        global temperature_3dprint          # 3D printer selly plug temperature (□)
        global state_torno                  # Torno state (on/off)
        global power_torno                  # Torno power (Watt)
        global temperature_torno            # Torno selly plug temperature (□)
        global state_cnc                    # CNC state (on/off)
        global power_cnc                    # CNC power (Watt)
        global temperature_cnc              # CNC selly plug temperature (□)
        global state_serra_circular         # Serra circular state (on/off)
        global power_serra_circular         # Serra circular power (Watt)
        global temperature_serra_circular   # Serra circular selly plug temperature (□)
        global state_serra_fita             # Serra de fita state (on/off)
        global power_serra_fita             # Serra de fita power (Watt)
        global temperature_serra_fita       # Serra de fita selly plug temperature (□)

        devId = msg.topic.split("/")[-1]
        # print('Received measurement from device ', devId)
        try:
            measurement = json.loads(msg.payload.decode())
        except Exception as e:
            print('Error parsing values!\nerror: '+str(e))
            print('MQTT message: ', msg.payload)
            return
        if not "sensors" in measurement:
            print('Invalid measurement. Missing sensors')
            return
        for sen in measurement["sensors"]:
            if not "senId" in sen:
                print('Invalid measurement. Missing sensor ID')
                return
            if not "outs" in sen or len(sen["outs"]) == 0:
                print('Invalid measurement. Missing sensor outs')
                return
            try:
                ####    IEQ data process    ####
                if devId == ieq:
```

```
                if sen["senId"] == 0:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    lux_h=sen["outs"][0]["processed"]
                elif sen["senId"] == 1:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    bmetmp=sen["outs"][0]["processed"]
                    atmpress=sen["outs"][1]["processed"]
                    bemrh=sen["outs"][2]["processed"]
                    gasres=sen["outs"][3]["processed"]
                elif sen["senId"] == 2:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    temperature=sen["outs"][0]["processed"]
                    rh=sen["outs"][1]["processed"]
                elif sen["senId"] == 3:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    hoco=sen["outs"][0]["processed"]
                    sfarh=sen["outs"][1]["processed"]
                    sfatmp=sen["outs"][2]["processed"]
                elif sen["senId"] == 4:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    co2=sen["outs"][0]["processed"]
                elif sen["senId"] == 5:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    pm1=sen["outs"][0]["processed"]
                    pm25=sen["outs"][1]["processed"]
                    pm10=sen["outs"][2]["processed"]
                    pm1_atm=sen["outs"][3]["processed"]
                    pm25_atm=sen["outs"][4]["processed"]
                    cuatm=sen["outs"][5]["processed"]
                    pnr0p3=sen["outs"][6]["processed"]
                    pnr0p5=sen["outs"][7]["processed"]
                    pnr1p0=sen["outs"][8]["processed"]
                    pnr2p5=sen["outs"][9]["processed"]
                    pnr5p0=sen["outs"][10]["processed"]
                    pnr10=sen["outs"][11]["processed"]
                elif sen["senId"] == 6:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    mv=sen["outs"][0]["processed"]
                elif sen["senId"] == 7:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    spdb=sen["outs"][0]["processed"]
                elif sen["senId"] == 8:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    lux_v=sen["outs"][0]["processed"]
                elif sen["senId"] == 9:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    wind=sen["outs"][0]["processed"]
            ####    Window data process    ####
            elif devId == window:
                if sen["senId"] == 0:
                    power_window = sen["outs"][-1]["processed"]
                elif sen["senId"] == 2:
```

```
                    temperature_window = sen["outs"][-1]["processed"]
                ####    3D printer shelly data process    ####
                elif devId == shelly_3dprinter:
                    if sen["senId"] == -1:
                        power_3dprint = sen["outs"][0]["processed"]
                    elif sen["senId"] == 0:
                        temperature_3dprint = sen["outs"][0]["processed"]
                    elif sen["senId"] == 1:
                        state_3dprint = sen["outs"][0]["processed"]
                ####    CNC shelly data process    ####
                elif devId == shelly_cnc:
                    if sen["senId"] == -1:
                        power_cnc = sen["outs"][-1]["processed"]
                    elif sen["senId"] == 0:
                        temperature_cnc = sen["outs"][-1]["processed"]
                    elif sen["senId"] == 1:
                        state_cnc = sen["outs"][-1]["processed"]
                ####    Serra de fita shelly data process    ####
                elif devId == shelly_serra_fita:
                    if sen["senId"] == -1:
                        power_serra_fita = sen["outs"][-1]["processed"]
                    elif sen["senId"] == 0:
                        temperature_serra_fita = sen["outs"][-1]["processed"]
                    elif sen["senId"] == 1:
                        state_serra_fita = sen["outs"][-1]["processed"]
                ####    Serra circular shelly data process    ####
                elif devId == shelly_serra_circular:
                    if sen["senId"] == -1:
                        power_serra_circular = sen["outs"][-1]["processed"]
                    elif sen["senId"] == 0:
                        temperature_serra_circular = sen["outs"][-1]["processed"]
                    elif sen["senId"] == 1:
                        state_serra_circular = sen["outs"][-1]["processed"]
                ####    Torno shelly data process    ####
                elif devId == shelly_torno:
                    if sen["senId"] == -1:
                        power_torno = sen["outs"][-1]["processed"]
                    elif sen["senId"] == 0:
                        temperature_torno = sen["outs"][-1]["processed"]
                    elif sen["senId"] == 1:
                        state_torno = sen["outs"][-1]["processed"]
            except Exception as e:
                print('Error processing measurement\nerror: ', e)
                traceback.print_exc()
                return


if __name__ == '__main__':
    print("Script is starting...")
    mqtt_init()

    while True:
        try:
            send_cmd_to_dev(window, cmd_window_close)
        except Exception as e:
            print('Error processing main code!\nerror: ', e)
            traceback.print_exc()
            exit()
        time.sleep(1)  # Sensor data check interval (s)
        exit()

```

ANEXO 3 – PYTHON - CÓDIGO PARA PARAR JANELA

```
import paho.mqtt.client as mqtt
import time
import re
import json
import traceback

mqtt_broker = 'epbase.local'    # MQTT broker address (hostname or IP)
# mqtt_broker = '192.168.32.251'  # MQTT broker address (hostname or IP)
mqtt_port = 1883                # MQTT broker port
mqtt_ttl = 60                   # MQTT time to live (s)

#############
## devices ##
#############
ieq = 'e831cdd4da7c'
heater = 'e831cdd4d780'
shelly_3dprinter = '3ce90ec81020'
shelly_torno = '3ce90ec808d9'
shelly_cnc = '3ce90ec86cf2'
shelly_serra_circular = 'c8c9a3a4eb39'
shelly_serra_fita = '3ce90ec83ea5'
window = '3494547935f4'
#############

########################
## IEQ output values ##
########################
lux_h = bmetmp = atmpress = bemrh = gasres = temperature = rh = hoco = sfarh = sfatmp\
    = co2 = pm1 = pm25 = pm10 = pm1_atm = pm25_atm = cuatm = pnr0p3 = pnr0p5\
    = pnr1p0 = pnr2p5 = pnr5p0 = pnr10 = mv = spdb = lux_v = wind = None
###########################
## Shelly output values ##
###########################
power_3dprint = power_torno = power_cnc = power_serra_circular = power_serra_fita =
power_window\
              = temperature_3dprint = temperature_torno = temperature_cnc =
              temperature_serra_circular\
              = temperature_serra_fita = temperature_window = state_3dprint =
              state_torno = state_cnc\
              = state_serra_circular = state_serra_fita = state_window = None
###########################
## Heater output values ##
###########################
heater_state = heater_in_temp = heater_return_temp = heater_flux = None
###########################

####################
## Window commands ##
####################
cmd_window_open = '{"command":"roller/0?go=open"}'      # Open window
cmd_window_close = '{"command":"roller/0?go=close"}'    # Close window
cmd_window_stop = '{"command":"roller/0?go=stop"}'      # Stop window movement
window_cmd_response = {}    # stored device response
####################
## Heater commands ##
####################
cmd_heater_open = '{"command":"set_state","args":[1],"senId":3}'    # Start opening
the heater valve
cmd_heater_close = '{"command":"set_state","args":[0],"senId":3}'   # Start closing
the heater valve
heater_cmd_response = {}    # stored device response
####################

def send_cmd_to_dev(mac_addr, cmd):
    global window_cmd_response
    global heater_cmd_response

    mqtt_client.publish("EP/fazesse/" + re.sub('\W+', '',
                        mac_addr), payload=cmd, retain=False)
    if mac_addr == window:
        window_cmd_response["command"]=cmd
        window_cmd_response["response"]=None
```

```
68      elif mac_addr == heater:
69          heater_cmd_response["command"]=cmd
70          heater_cmd_response["response"]=None
71      else:
72          print("Device "+mac_addr+" is does not accept commands")
73
74  def mqtt_init():
75      global mqtt_client
76      mqtt_client = mqtt.Client()
77      try:
78          mqtt_client.connect(mqtt_broker, mqtt_port, mqtt_ttl)
79          mqtt_client.on_connect = on_connect
80          mqtt_client.on_disconnect = on_disconnect
81          mqtt_client.loop_start()
82
83          mqtt_client.message_callback_add(
84              'EP/measurements/+', on_measurement)
85          mqtt_client.message_callback_add(
86              'EP/Shelly/measurements/+', on_measurement)
87          mqtt_client.message_callback_add(
88              'EP/ack/+', on_ack)
89      except Exception as e:
90          print("Error mqtt_client.connect\nerror: "+str(e))
91          # traceback.print_exc()
92          time.sleep(5)
93          mqtt_init()
94          return
95      print('MQTT Connected')
96
97  def check_timestamp_us(ts):
98      test = (ts > minimum_timestamp_us and ts <
99              (time.time()*1000000+60*1000000))
100     if not test:
101         printd('Invalid timestamp!')
102     return test
103
104 def on_connect(client, userdata, flags, rc):
105     print("Connected with result code " + str(rc))
106     if rc == 0:
107         print("MQTT client connected OK")
108         client.subscribe("EP/#", qos=2)  # get all MQTT messages
109
110 def on_disconnect(client, userdata, flags):
111     print("MQTT client disconnected!")
112
113 def on_ack(client, userdata, msg):
114     try:
115         devId = msg.topic.split("/")[-1]
116         ack_msg_str=msg.payload.decode()
117         ack_msg = json.loads(ack_msg_str)
118         if "mqttid" in ack_msg:
119             print("MQTT message ID: ", ack_msg["mqttid"])
120     except Exception as e:
121         print('Error parsing ack message!\nerror: '+str(e))
122         print('MQTT message: ', msg.payload)
123         return
124     print("Device "+devId+" ACK:")
125     print("\t"+ack_msg_str)
126     #### TAKE ACTION ON DEVICE RESPONSE ####
127
128 def on_measurement(client, userdata, msg):
129     ######################################################
130     ##                  IEQ variables                   ##
131     ######################################################
132     global lux_h        # Lux horizontal (lux)
133     global bmetmp       # Internal temperature (□)
134     global atmpress     # Atmosferic pressure (Pa)
135     global bemrh        # Internal relative humidity (%)
136     global gasres       # Gas resistance (Ω)
137     global temperature  # Ambient temperature (□)
138     global rh           # Ambient relative humidity (%)
139     global hoco         # Formaldehyde (HOCO)
```

```
    global sfarh        # Internal relative humidity (%)
    global sfatmp       # Internal temperature (□)
    global co2          # Carbon dioxide (CO2)
    global pm1          # PM 1.0 µm con unit
    global pm25         # PM 2.5 µm con unit
    global pm10         # PM 10 µm con unit
    global pm1_atm      # PM 1.0 µm con unit atmosphere
    global pm25_atm     # PM 2.5 µm con unit atmosphere
    global cuatm        # Con unit atmosphere
    global pnr0p3       # Particle number 0.3 µm
    global pnr0p5       # Particle number 0.5 µm
    global pnr1p0       # Particle number 1.0 µm
    global pnr2p5       # Particle number 2.5 µm
    global pnr5p0       # Particle number 5.0 µm
    global pnr10        # Particle number 10 µm
    global mv           # Movement (true/false)
    global spdb         # Sound pressure (Db)
    global lux_v        # Lux vertical (lux)
    global wind         # Wind speed (m/s)
    ############################################################
    ##                    Heater variables                    ##
    ############################################################
    global heater_state         # Heater state (on/off)
    global heater_in_temp       # Water input temperature (□)
    global heater_return_temp   # Water return temperature (□)
    global heater_flux          # Water flux (L/m)
    ##################################################################
    ##                     Window variables                         ##
    ##################################################################
    global state_window         # Window state (open/closed)
    global power_window         # Window power (Watt)
    global temperature_window   # Window controler temperature (□)
    ##########################################################################
    ##                      Shelly plug variables                           ##
    ##########################################################################
    global state_3dprint                # 3D printer state (on/off)
    global power_3dprint                # 3D printer power (Watt)
    global temperature_3dprint          # 3D printer selly plug temperature (□)
    global state_torno                  # Torno state (on/off)
    global power_torno                  # Torno power (Watt)
    global temperature_torno            # Torno selly plug temperature (□)
    global state_cnc                    # CNC state (on/off)
    global power_cnc                    # CNC power (Watt)
    global temperature_cnc              # CNC selly plug temperature (□)
    global state_serra_circular         # Serra circular state (on/off)
    global power_serra_circular         # Serra circular power (Watt)
    global temperature_serra_circular   # Serra circular selly plug temperature (□)
    global state_serra_fita             # Serra de fita state (on/off)
    global power_serra_fita             # Serra de fita power (Watt)
    global temperature_serra_fita       # Serra de fita selly plug temperature (□)

    devId = msg.topic.split("/")[-1]
    # print('Received measurement from device ', devId)
    try:
        measurement = json.loads(msg.payload.decode())
    except Exception as e:
        print('Error parsing values!\nerror: '+str(e))
        print('MQTT message: ', msg.payload)
        return
    if not "sensors" in measurement:
        print('Invalid measurement. Missing sensors')
        return
    for sen in measurement["sensors"]:
        if not "senId" in sen:
            print('Invalid measurement. Missing sensor ID')
            return
        if not "outs" in sen or len(sen["outs"]) == 0:
            print('Invalid measurement. Missing sensor outs')
            return
        try:
            ####    IEQ data process    ####
            if devId == ieq:
```

```
                if sen["senId"] == 0:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    lux_h=sen["outs"][0]["processed"]
                elif sen["senId"] == 1:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    bmetmp=sen["outs"][0]["processed"]
                    atmpress=sen["outs"][1]["processed"]
                    bemrh=sen["outs"][2]["processed"]
                    gasres=sen["outs"][3]["processed"]
                elif sen["senId"] == 2:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    temperature=sen["outs"][0]["processed"]
                    rh=sen["outs"][1]["processed"]
                elif sen["senId"] == 3:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    hoco=sen["outs"][0]["processed"]
                    sfarh=sen["outs"][1]["processed"]
                    sfatmp=sen["outs"][2]["processed"]
                elif sen["senId"] == 4:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    co2=sen["outs"][0]["processed"]
                elif sen["senId"] == 5:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    pm1=sen["outs"][0]["processed"]
                    pm25=sen["outs"][1]["processed"]
                    pm10=sen["outs"][2]["processed"]
                    pm1_atm=sen["outs"][3]["processed"]
                    pm25_atm=sen["outs"][4]["processed"]
                    cuatm=sen["outs"][5]["processed"]
                    pnr0p3=sen["outs"][6]["processed"]
                    pnr0p5=sen["outs"][7]["processed"]
                    pnr1p0=sen["outs"][8]["processed"]
                    pnr2p5=sen["outs"][9]["processed"]
                    pnr5p0=sen["outs"][10]["processed"]
                    pnr10=sen["outs"][11]["processed"]
                elif sen["senId"] == 6:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    mv=sen["outs"][0]["processed"]
                elif sen["senId"] == 7:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    spdb=sen["outs"][0]["processed"]
                elif sen["senId"] == 8:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    lux_v=sen["outs"][0]["processed"]
                elif sen["senId"] == 9:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    wind=sen["outs"][0]["processed"]
            ####    Window data process    ####
            elif devId == window:
                if sen["senId"] == 0:
                    power_window = sen["outs"][-1]["processed"]
                elif sen["senId"] == 2:
```

```
                    temperature_window = sen["outs"][-1]["processed"]
            ####    3D printer shelly data process    ####
            elif devId == shelly_3dprinter:
                if sen["senId"] == -1:
                    power_3dprint = sen["outs"][0]["processed"]
                elif sen["senId"] == 0:
                    temperature_3dprint = sen["outs"][0]["processed"]
                elif sen["senId"] == 1:
                    state_3dprint = sen["outs"][0]["processed"]
            ####    CNC shelly data process    ####
            elif devId == shelly_cnc:
                if sen["senId"] == -1:
                    power_cnc = sen["outs"][-1]["processed"]
                elif sen["senId"] == 0:
                    temperature_cnc = sen["outs"][-1]["processed"]
                elif sen["senId"] == 1:
                    state_cnc = sen["outs"][-1]["processed"]
            ####    Serra de fita shelly data process    ####
            elif devId == shelly_serra_fita:
                if sen["senId"] == -1:
                    power_serra_fita = sen["outs"][-1]["processed"]
                elif sen["senId"] == 0:
                    temperature_serra_fita = sen["outs"][-1]["processed"]
                elif sen["senId"] == 1:
                    state_serra_fita = sen["outs"][-1]["processed"]
            ####    Serra circular shelly data process    ####
            elif devId == shelly_serra_circular:
                if sen["senId"] == -1:
                    power_serra_circular = sen["outs"][-1]["processed"]
                elif sen["senId"] == 0:
                    temperature_serra_circular = sen["outs"][-1]["processed"]
                elif sen["senId"] == 1:
                    state_serra_circular = sen["outs"][-1]["processed"]
            ####    Torno shelly data process    ####
            elif devId == shelly_torno:
                if sen["senId"] == -1:
                    power_torno = sen["outs"][-1]["processed"]
                elif sen["senId"] == 0:
                    temperature_torno = sen["outs"][-1]["processed"]
                elif sen["senId"] == 1:
                    state_torno = sen["outs"][-1]["processed"]
        except Exception as e:
            print('Error processing measurement\nerror: ', e)
            traceback.print_exc()
            return


if __name__ == '__main__':
    print("Script is starting...")
    mqtt_init()
    try:
        send_cmd_to_dev(window, cmd_window_stop)
    except Exception as e:
        print('Error processing main code!\nerror: ', e)
        traceback.print_exc()
        exit()
    time.sleep(1)  # Sensor data check interval (s)
    exit()

```

ANEXO 4 – PYTHON - CÓDIGO AUTOMATIZADO

```
import paho.mqtt.client as mqtt
import time
import re
import json
import traceback
import math
import datetime
import openpyxl

mqtt_broker = 'epbase.local'    # MQTT broker address (hostname or IP)
# mqtt_broker = '192.168.32.251'  # MQTT broker address (hostname or IP)
mqtt_port = 1883                # MQTT broker port
mqtt_ttl = 60                   # MQTT time to live (s)

#############
## devices ##
#############
ieq = 'e831cdd4da7c'
heater = 'e831cdd4d780'
shelly_3dprinter = '3ce90ec81020'
shelly_torno = '3ce90ec808d9'
shelly_cnc = '3ce90ec86cf2'
shelly_serra_circular = 'c8c9a3a4eb39'
shelly_serra_fita = '3ce90ec83ea5'
window = '3494547935f4'
#############

########################
## IEQ output values ##
########################
lux_h = bmetmp = atmpress = bemrh = gasres = temperature = rh = hoco = sfarh = sfatmp\
    = co2 = pm1 = pm25 = pm10 = pm1_atm = pm25_atm = cuatm = pnr0p3 = pnr0p5\
    = pnr1p0 = pnr2p5 = pnr5p0 = pnr10 = mv = spdb = lux_v = wind = None
###########################
## Shelly output values ##
###########################
power_3dprint = power_torno = power_cnc = power_serra_circular = power_serra_fita =
power_window\
              = temperature_3dprint = temperature_torno = temperature_cnc =
              temperature_serra_circular\
              = temperature_serra_fita = temperature_window = state_3dprint =
              state_torno = state_cnc\
              = state_serra_circular = state_serra_fita = state_window = None
###########################
## Heater output values ##
###########################
heater_state = heater_in_temp = heater_return_temp = heater_flux = None
###########################

####################
## Window commands ##
####################
cmd_window_open = '{"command":"roller/0?go=open"}'      # Open window
cmd_window_close = '{"command":"roller/0?go=close"}'    # Close window
cmd_window_stop = '{"command":"roller/0?go=stop"}'      # Stop window movement
window_cmd_response = {}    # stored device response
####################
## Heater commands ##
####################
cmd_heater_open = '{"command":"set_state","args":[1],"senId":3}'    # Start opening
the heater valve
cmd_heater_close = '{"command":"set_state","args":[0],"senId":3}'   # Start closing
the heater valve
heater_cmd_response = {}    # stored device response
####################

def save_message_to_file(message):
    data_hora = datetime.datetime.now()
    data_hora_formatada = data_hora.strftime("%d/%m/%Y %H:%M:%S")

    with open('mensagens.txt', 'a') as arquivo:
        arquivo.write("Data/Hora: {}\n".format(data_hora_formatada))
```

```
68          arquivo.write(message + "\n\n")
69
70  def send_cmd_to_dev(mac_addr, cmd):
71      global window_cmd_response
72      global heater_cmd_response
73
74      mqtt_client.publish("EP/fazesse/" + re.sub('\W+', '',
75                          mac_addr), payload=cmd, retain=False)
76      if mac_addr == window:
77          window_cmd_response["command"]=cmd
78          window_cmd_response["response"]=None
79      elif mac_addr == heater:
80          heater_cmd_response["command"]=cmd
81          heater_cmd_response["response"]=None
82      else:
83          print("Device "+mac_addr+" is does not accept commands")
84
85  def mqtt_init():
86      global mqtt_client
87      mqtt_client = mqtt.Client()
88      try:
89          mqtt_client.connect(mqtt_broker, mqtt_port, mqtt_ttl)
90          mqtt_client.on_connect = on_connect
91          mqtt_client.on_disconnect = on_disconnect
92          mqtt_client.loop_start()
93
94          mqtt_client.message_callback_add(
95              'EP/measurements/+', on_measurement)
96          mqtt_client.message_callback_add(
97              'EP/Shelly/measurements/+', on_measurement)
98          mqtt_client.message_callback_add(
99              'EP/ack/+', on_ack)
100     except Exception as e:
101         print("Error mqtt_client.connect\nerror: "+str(e))
102         # traceback.print_exc()
103         time.sleep(5)
104         mqtt_init()
105         return
106     print('MQTT Connected')
107
108 def check_timestamp_us(ts):
109     test = (ts > minimum_timestamp_us and ts <
110             (time.time()*1000000+60*1000000))
111     if not test:
112         printd('Invalid timestamp!')
113     return test
114
115 def on_connect(client, userdata, flags, rc):
116     print("Connected with result code " + str(rc))
117     if rc == 0:
118         print("MQTT client connected OK")
119         client.subscribe("EP/#", qos=2)  # get all MQTT messages
120
121 def on_disconnect(client, userdata, flags):
122     print("MQTT client disconnected!")
123
124 def on_ack(client, userdata, msg):
125     try:
126         devId = msg.topic.split("/")[-1]
127         ack_msg_str=msg.payload.decode()
128         ack_msg = json.loads(ack_msg_str)
129         if "mqttid" in ack_msg:
130             print("MQTT message ID: ", ack_msg["mqttid"])
131     except Exception as e:
132         print('Error parsing ack message!\nerror: '+str(e))
133         print('MQTT message: ', msg.payload)
134         return
135     print("Device "+devId+" ACK:")
136     print("\t"+ack_msg_str)
137     #### TAKE ACTION ON DEVICE RESPONSE ####
138
139 def on_measurement(client, userdata, msg):
```

```
    ######################################################
    ##                  IEQ variables                   ##
    ######################################################
    global lux_h        # Lux horizontal (lux)
    global bmetmp       # Internal temperature (□)
    global atmpress     # Atmosferic pressure (Pa)
    global bemrh        # Internal relative humidity (%)
    global gasres       # Gas resistance (Ω)
    global temperature  # Ambient temperature (□)
    global rh           # Ambient relative humidity (%)
    global hoco         # Formaldehyde (HOCO)
    global sfarh        # Internal relative humidity (%)
    global sfatmp       # Internal temperature (□)
    global co2          # Carbon dioxide (CO2)
    global pm1          # PM 1.0 µm con unit
    global pm25         # PM 2.5 µm con unit
    global pm10         # PM 10 µm con unit
    global pm1_atm      # PM 1.0 µm con unit atmosphere
    global pm25_atm     # PM 2.5 µm con unit atmosphere
    global cuatm        # Con unit atmosphere
    global pnr0p3       # Particle number 0.3 µm
    global pnr0p5       # Particle number 0.5 µm
    global pnr1p0       # Particle number 1.0 µm
    global pnr2p5       # Particle number 2.5 µm
    global pnr5p0       # Particle number 5.0 µm
    global pnr10        # Particle number 10 µm
    global mv           # Movement (true/false)
    global spdb         # Sound pressure (Db)
    global lux_v        # Lux vertical (lux)
    global wind         # Wind speed (m/s)
    ############################################################
    ##                    Heater variables                    ##
    ############################################################
    global heater_state         # Heater state (on/off)
    global heater_in_temp       # Water input temperature (□)
    global heater_return_temp   # Water return temperature (□)
    global heater_flux          # Water flux (L/m)
    ############################################################
    ##                   Window variables                     ##
    ############################################################
    global state_window         # Window state (open/closed)
    global power_window         # Window power (Watt)
    global temperature_window   # Window controler temperature (□)
    ###########################################################################
    ##                     Shelly plug variables                             ##
    ###########################################################################
    global state_3dprint                # 3D printer state (on/off)
    global power_3dprint                # 3D printer power (Watt)
    global temperature_3dprint          # 3D printer selly plug temperature (□)
    global state_torno                  # Torno state (on/off)
    global power_torno                  # Torno power (Watt)
    global temperature_torno            # Torno selly plug temperature (□)
    global state_cnc                    # CNC state (on/off)
    global power_cnc                    # CNC power (Watt)
    global temperature_cnc              # CNC selly plug temperature (□)
    global state_serra_circular         # Serra circular state (on/off)
    global power_serra_circular         # Serra circular power (Watt)
    global temperature_serra_circular   # Serra circular selly plug temperature (□)
    global state_serra_fita             # Serra de fita state (on/off)
    global power_serra_fita             # Serra de fita power (Watt)
    global temperature_serra_fita       # Serra de fita selly plug temperature (□)

    devId = msg.topic.split("/")[-1]
    # print('Received measurement from device ', devId)
    try:
        measurement = json.loads(msg.payload.decode())
    except Exception as e:
        print('Error parsing values!\nerror: '+str(e))
        print('MQTT message: ', msg.payload)
        return
    if not "sensors" in measurement:
        print('Invalid measurement. Missing sensors')
```

```
            return
        for sen in measurement["sensors"]:
            if not "senId" in sen:
                print('Invalid measurement. Missing sensor ID')
                return
            if not "outs" in sen or len(sen["outs"]) == 0:
                print('Invalid measurement. Missing sensor outs')
                return
            try:
                ####    IEQ data process    ####
                if devId == ieq:
                    if sen["senId"] == 0:
                        if not "processed" in sen["outs"][0]:
                            print('Error: missing processed value on measurement!')
                            return
                        lux_h=sen["outs"][0]["processed"]
                    elif sen["senId"] == 1:
                        if not "processed" in sen["outs"][0]:
                            print('Error: missing processed value on measurement!')
                            return
                        bmetmp=sen["outs"][0]["processed"]
                        atmpress=sen["outs"][1]["processed"]
                        bemrh=sen["outs"][2]["processed"]
                        gasres=sen["outs"][3]["processed"]
                    elif sen["senId"] == 2:
                        if not "processed" in sen["outs"][0]:
                            print('Error: missing processed value on measurement!')
                            return
                        temperature=sen["outs"][0]["processed"]
                        rh=sen["outs"][1]["processed"]
                    elif sen["senId"] == 3:
                        if not "processed" in sen["outs"][0]:
                            print('Error: missing processed value on measurement!')
                            return
                        hoco=sen["outs"][0]["processed"]
                        sfarh=sen["outs"][1]["processed"]
                        sfatmp=sen["outs"][2]["processed"]
                    elif sen["senId"] == 4:
                        if not "processed" in sen["outs"][0]:
                            print('Error: missing processed value on measurement!')
                            return
                        co2=sen["outs"][0]["processed"]
                    elif sen["senId"] == 5:
                        if not "processed" in sen["outs"][0]:
                            print('Error: missing processed value on measurement!')
                            return
                        pm1=sen["outs"][0]["processed"]
                        pm25=sen["outs"][1]["processed"]
                        pm10=sen["outs"][2]["processed"]
                        pm1_atm=sen["outs"][3]["processed"]
                        pm25_atm=sen["outs"][4]["processed"]
                        cuatm=sen["outs"][5]["processed"]
                        pnr0p3=sen["outs"][6]["processed"]
                        pnr0p5=sen["outs"][7]["processed"]
                        pnr1p0=sen["outs"][8]["processed"]
                        pnr2p5=sen["outs"][9]["processed"]
                        pnr5p0=sen["outs"][10]["processed"]
                        pnr10=sen["outs"][11]["processed"]
                    elif sen["senId"] == 6:
                        if not "processed" in sen["outs"][0]:
                            print('Error: missing processed value on measurement!')
                            return
                        mv=sen["outs"][0]["processed"]
                    elif sen["senId"] == 7:
                        if not "processed" in sen["outs"][0]:
                            print('Error: missing processed value on measurement!')
                            return
                        spdb=sen["outs"][0]["processed"]
                    elif sen["senId"] == 8:
                        if not "processed" in sen["outs"][0]:
                            print('Error: missing processed value on measurement!')
                            return
```

```
                    lux_v=sen["outs"][0]["processed"]
                elif sen["senId"] == 9:
                    if not "processed" in sen["outs"][0]:
                        print('Error: missing processed value on measurement!')
                        return
                    wind=sen["outs"][0]["processed"]
            ####    Window data process    ####
            elif devId == window:
                if sen["senId"] == 0:
                    power_window = sen["outs"][-1]["processed"]
                elif sen["senId"] == 2:
                    temperature_window = sen["outs"][-1]["processed"]
            ####    3D printer shelly data process    ####
            elif devId == shelly_3dprinter:
                if sen["senId"] == -1:
                    power_3dprint = sen["outs"][0]["processed"]
                elif sen["senId"] == 0:
                    temperature_3dprint = sen["outs"][0]["processed"]
                elif sen["senId"] == 1:
                    state_3dprint = sen["outs"][0]["processed"]
            ####    CNC shelly data process    ####
            elif devId == shelly_cnc:
                if sen["senId"] == -1:
                    power_cnc = sen["outs"][-1]["processed"]
                elif sen["senId"] == 0:
                    temperature_cnc = sen["outs"][-1]["processed"]
                elif sen["senId"] == 1:
                    state_cnc = sen["outs"][-1]["processed"]
            ####    Serra de fita shelly data process    ####
            elif devId == shelly_serra_fita:
                if sen["senId"] == -1:
                    power_serra_fita = sen["outs"][-1]["processed"]
                elif sen["senId"] == 0:
                    temperature_serra_fita = sen["outs"][-1]["processed"]
                elif sen["senId"] == 1:
                    state_serra_fita = sen["outs"][-1]["processed"]
            ####    Serra circular shelly data process    ####
            elif devId == shelly_serra_circular:
                if sen["senId"] == -1:
                    power_serra_circular = sen["outs"][-1]["processed"]
                elif sen["senId"] == 0:
                    temperature_serra_circular = sen["outs"][-1]["processed"]
                elif sen["senId"] == 1:
                    state_serra_circular = sen["outs"][-1]["processed"]
            ####    Torno shelly data process    ####
            elif devId == shelly_torno:
                if sen["senId"] == -1:
                    power_torno = sen["outs"][-1]["processed"]
                elif sen["senId"] == 0:
                    temperature_torno = sen["outs"][-1]["processed"]
                elif sen["senId"] == 1:
                    state_torno = sen["outs"][-1]["processed"]
        except Exception as e:
            print('Error processing measurement\nerror: ', e)
            traceback.print_exc()
            return

# Função para salvar os valores no Excel
def save_to_excel(psat, pv, co2):
    # Carregar ou criar um arquivo do Excel
    try:
        wb = openpyxl.load_workbook('dados.xlsx')
    except FileNotFoundError:
        wb = openpyxl.Workbook()

    # Selecionar a folha ativa (a primeira folha por padrão)
    sheet = wb.active

    # Obter a data e hora atual
    data_hora_atual = datetime.datetime.now()

    # Verificar se a primeira linha está vazia
```

```
356         if sheet.cell(row=1, column=1).value is None:
357             sheet.cell(row=1, column=1).value = "DATA e HORA"
358             sheet.cell(row=1, column=2).value = "PSAT"
359             sheet.cell(row=1, column=3).value = "PV"
360             sheet.cell(row=1, column=4).value = "CO2"
361 
362         # Adicionar valores às colunas
363         row_count = sheet.max_row + 1
364         sheet.cell(row=row_count, column=1).value = data_hora_atual
365         sheet.cell(row=row_count, column=2).value = psat
366         sheet.cell(row=row_count, column=3).value = pv
367         sheet.cell(row=row_count, column=4).value = co2
368 
369         # Guardar o arquivo Excel
370         wb.save('dados.xlsx')
371 
372     if __name__ == '__main__':
373         print("Script is starting...")
374         mqtt_init()
375 
376         while True:
377             try:
378                 #### Exemplo: cálculo de psat e pv ####
379                 if temperature and rh and co2:
380                     temperature = float(temperature)
381                     rh = float(rh)
382                     co2 = float(co2)
383                     psat = 611 * math.exp((17.08 * temperature) / (234.18 + temperature))
384                     pv = (rh/100) * psat
385                     psat_rounded = round(psat, 2)
386                     pv_rounded = round(pv, 2)
387                     co2_rounded = round(co2, 2)
388                     print("A pressão de saturação é: {} Pa".format(psat_rounded))
389                     print("A pressão parcial do vapor de água é: {} Pa".format(pv_rounded
                        ))
390                     print("A concentração de CO2 é: {} ppm".format(co2_rounded))
391 
392 
393                     # Guardar mensagem no arquivo
394                     message = "A pressão de saturação é: {} Pa\nA pressão parcial do
                        vapor de água é: {} Pa\nA concentração de CO2 é: {} ppm".format(
                        psat_rounded, pv_rounded,co2_rounded)
395                     save_message_to_file(message)
396 
397                     # Guardar valores no Excel
398                     save_to_excel(psat_rounded, pv_rounded, co2_rounded)
399 
400                     #### Exemplo: controlo da janela através da pv e do co2 ####
401                     if pv:
402                         if pv <= 2000 and co2 <= 900:
403                             send_cmd_to_dev(window, cmd_window_close)
404                             print("Janela fechada porque Pressão parcial de vapor ≤ 2000
                                Pa e concentração de CO2 ≤ 900 ppm")
405                             save_message_to_file("Janela fechada porque Pressão parcial
                                de vapor ≤ 2000 Pa e concentração de CO2 ≤ 900 ppm")
406                         elif pv > 2000 or co2 > 900:
407                             send_cmd_to_dev(window, cmd_window_open)
408                             if pv > 2000 and co2 <= 900:
409                                 print("Janela aberta porque Pressão parcial de vapor >
                                    2000 Pa")
410                                 save_message_to_file("Janela aberta porque Pressão
                                    parcial de vapor > 2000 Pa")
411                             if pv <= 2000 and co2 > 900:
412                                 print("Janela aberta porque Concentração de CO2 > 900 ppm"
                                    )
413                                 save_message_to_file("Janela aberta porque Concentração
                                    de CO2 > 900 ppm")
414                             if pv > 2000 and co2 > 900:
415                                 print("Janela aberta porque Pressão parcial de vapor >
                                    2000 Pa e porque Concentração de CO2 > 900 ppm")
416                                 save_message_to_file("Janela aberta porque Pressão
                                    parcial de vapor > 2000 Pa e Concentração de CO2 > 900
```

```
                            ppm")
417
418
419             #### Exemplo: controlo do aquecedor de acordo com a temperatura do
                ambiente ####
420             if temperature:
421                 if temperature < 18:
422                     send_cmd_to_dev(heater, cmd_heater_open)
423                 elif temperature > 25:
424                     send_cmd_to_dev(heater, cmd_heater_close)
425         except Exception as e:
426             print('Error processing main code!\nerror:', e)
427             traceback.print_exc()
428             exit()
429
430         time.sleep(60)  # Intervalo de verificação dos dados do sensor (em segundos)
431         print('Still running...')
432
```