

POLITECNICO DI TORINO

Master of science in: Building Engineering (Green Building)

Department of Structural, Geotechnical and Building Engineering (DISEG)

AI-Assisted Decision Support for Data Center Energy Management:

A BIM-to-BEM-to-RAG Pipeline Applied to a Live Data Center

Supervisor:

Prof. Matteo Del Giudice

Co-Supervisor

Michele Zucco

Candidate:

Samikshya Giri

S328106

September 2026

To my mother and my family, who never let distance shrink my dreams.
To my friends and colleagues, for the light they kept on through the long nights.
To those the changing climate has already cost the most, in hope this small effort serves the larger one.

Acknowledgement

This thesis reflects a continuous effort to understand what buildings would tell us if they could talk, and how we might make them better in terms of energy.

I would like to thank my supervisor, Prof. Matteo Del Giudice, and my co-supervisor, Michele Zucco, for their guidance throughout this project, including the constructive and detailed feedback that shaped the thesis into what it is now.

My gratitude extends as well to all those who, in different ways, supported me throughout this journey.

Finally, thank you to my family, friends, and colleagues for their support and patience during the writing of this thesis.

Statement on the Use of AI Tools

I acknowledge the use of Claude (Anthropic) in two capacities: first, as an assistive tool for writing the code and debugging the pipeline scripts and the dashboard, as mentioned in Chapter 5 and Appendix A, and secondly, as a drafting aid for writing and editing the thesis content, including structure and sentence corrections. None of the AI-generated text was used directly in this final submission. All the code and text produced have been validated by me before being included. All the technical claims, data analysis, and conclusions run against real project data belong to me alone.

List of Figures

Figure 1: Literature-review-to-pipeline-decision methodology 15
Figure 2: Site plan of the FiberCop complex (Blocco A / Blocco B).....26
Figure 3: Piano Primo with Sala G highlighted.....26
Figure 4: South-facade photograph of Blocco A (FiberCop offices and CED) 27
Figure 5: Original FiberCop rack-layout drawing for Sala G (Torino Lancia) 29
Figure 6: Piano Primo floor area by zone.....31
Figure 7: Sala G in isolation, showing external glazing.....32
Figure 8: Edificio 1 construction categories.....33
Figure 9: General Framework 37
Figure 10: Schema for the Proposed Pipeline 38
Figure 11: Isometric rendering of the federated Revit model 47
Figure 12: DesignBuilder isometric view of Blocco 1 48
Figure 13: Federated model verified against laser-scan point cloud 48
Figure 14: Architecture-discipline model in isolation 50
Figure 15: Mechanical-discipline model in isolation 50
Figure 16: Electrical-discipline model in isolation.....50
Figure 17: OpenStudioApp import error — duplicate floor-boundary surfaces (Error 1) 51
Figure 18: OpenStudio 3D view of the duplicated floor surfaces (Error 1) 52
Figure 19: OpenStudio HVAC-component wiring failure (Error 2) 53
Figure 20: DesignBuilder view of imported gbXML with corrected external surfaces 54
Figure 21: DesignBuilder HVAC system diagram (chilled/hot water loops).....56
Figure 22: DesignBuilder hot-water loop availability schedule (Off 24/7).....57
Figure 23: Detailed DesignBuilder chilled-/hot-water plant-loop diagram.....57
Figure 24: DesignBuilder HVAC template applied to Sala G.....58
Figure 25: EnergyPlus annual simulation output for Blocco 1 58
Figure 26: EnergyPlus annual simulation output for Sala G 59
Figure 27: Schema: BRICK class-level relationships used 60
Figure 28: Neo4j Desktop view of the populated FBCOP knowledge graph.....61
Figure 29: From source format to AI-ready store.....62
Figure 30: FiberCop dashboard: Site level 66
Figure 31: FiberCop dashboard: Building level 66
Figure 32: FiberCop dashboard: Simulation & AI level 67
Figure 33: BIMserver.org (BIMvie.ws) server interface.....67
Figure 34: Dashboard chatbot generation-layer hallucination example 74
Figure 35: Dashboard before redesign: macro-to-micro accordion layout.....75

Figure 36: Dashboard after redesign: single-view grid layout 76
Figure 37: Dashboard architecture: build-time data flow....77
Figure 38: DesignBuilder HVAC data tab for Sala G (Section 5.10) 78
Figure 39: Energy breakdown chart: annual energy by end use....83
Figure 40: Neo4j graph visualisation of rooms, equipment and sensors....85
Figure 41: Neo4j Month-chain subgraph of measured consumption 86
Figure 42: Measured vs. simulated monthly energy....87
Figure 43: Cross-criterion summary diagram....90
Figure 44: Directional sensitivity of Modello 0 to key assumptions....93
Figure 45: Simulated PUE vs. typical data centre benchmark range 98

List of Tables

Table 1: Abbreviations and Terminology....13
Table 2: Literature Search Keyword Clusters and Record Counts....16
Table 3: Sala G Geometric, Energetic and System Properties 28
Table 4: Geometrical Properties of the Five Piano Primo Zones 30
Table 5: Research Gaps and Framework Justification 35
Table 6: Data Categories and Sources Used in the Pipeline....39
Table 7: Tooling and Software Environment by Pipeline Stage 42
Table 8: Critical Pipeline Scripts and Their Inputs 68
Table 9: Interoperability Error Matrix and Outcomes....80
Table 10: Energy Simulation Results Summary....82
Table 11: RAG Query Evaluation Results 88
Table 12: Sala G Cooling and Power-Distribution Equipment Inventory....124
Table 13: Consolidated Monthly Measured Consumption (19 Months)....128
Table 14: DesignBuilder Simulation Configuration Parameters....130
Table 15: Extended Literature Synthesis....132
Table 16: Modelling Assumptions Register....136
Table 17: Source Equipment and Installation Documentation....139
Table 18: Comparative Equipment Data - Sala G vs. Sala L 143

Contents

1. Introduction 12

1.1 Context and Motivation 12

1.2 Research Questions 13

1.3 Scope and Limitations 13

1.4 Terminology and Abbreviations 14

2. State of the Art and Literature-Grounded Framework 15

2.1 Literature Selection Methodology 15

2.2 BIM-to-BEM for Energy Analysis: Relevance to Data Centres 16

2.3 BIM-to-BEM Interoperability Approaches and Technologies 18

2.4 LLM-Driven Automation of Building Energy Models 19

2.5 RAG Architectures for Building and BIM Data 20

2.6. Semantic Models and Knowledge Graphs for BIM Data 22

2.7 AI-Assisted Decision Support for Building and Facility Management 22

2.8 Synthesis: From Literature Gaps to Pipeline Requirements 23

2.9 Critical Appraisal of Limitations in the Methodology of Reviewed Literature 24

3. Case Study: FiberCop Data Center, Torino 26

3.1. Site and Building Context 26

3.2 Sala G: Position and Function 27

3.3 Sala G: Case Study Summary Sheet 28

3.4 Cooling and Power Equipment Overview 29

3.5 Building Envelope and Construction 32

3.6 Occupancy, Schedules and Operational Assumptions 33

3.7 Institutional Context and Collaboration 34

4: Methodology 35

4.1. Synthesis: Research Gap and Framework Justification 35

4.2. Research Design 36

5. Testing and evaluation 47

5.1. BIM Model Development 47
5.2. BIM-to-BEM Interoperability Assessment 51
5.3. Tool Selection: DesignBuilder over OpenStudio 55
5.4. Energy Simulation (Modello 0) 55
5.5 Semantic Modelling (BRICK Ontology) 59
5.6. RAG Pipeline 61
5.7 Dashboard Application 65
5.8 Pipeline Orchestration and Reproducibility 68
5.9 Dashboard Chatbot: Query Handling in Detail 70
5.10 Step-by-Step Simulation Workflow in DesignBuilder 77
6. Results 80
6.1 BIM-to-BEM Interoperability Results 80
6.2 Energy Simulation Results 82
6.3 Semantic and Knowledge-Graph Results 84
6.4 AI-Based Decision Support Results 85
6.5 Cross-Criterion Interpretation 90
6.6 Reconciliation with Source Equipment Documentation 91
6.7 Directional Sensitivity of the Reported Results to Key Assumptions 92
7. Discussion 94
7.1 On the Interoperability Gaps 94
7.2 On Automation 95
7.3 On Data Security and Local Deployment 95
7.4 On the Decision-Support Value 96
7.5 Framework Contribution 96
7.6 Limitations 97
7.7 Comparison Against Benchmarks in the Literature 98
7.8 Generalisability Beyond Sala G 99
7.9 Practical Recommendations for FiberCop 100

7.10 Cost and Deployment Considerations 100
7.11 Reflections on the Research Process 101
8. Conclusions 102
8.1 Answers to the Research Questions 102
8.2 Contribution 102
8.3 Limitations and Future Work 103
8.4 Implications for Practice 104
8.5 Closing Remarks 104
References 105

Appendix A: Pipeline Script Listings 109

A.1 fix_gbxml_floors.py, gbXML Floor-Boundary Correction (Section 5.2, Error 1) 109

With respect to the Piano Primo export, the Python script was able to detect and fix the precisely five surface faces mentioned in Section 5.2; namely, aim0598, aim0819, aim0879, aim0990, and aim1139. 110

A.2 extract_ifc_json.py, IFC-to-Segmented-JSON Extraction (Section 5.6) 110

A.3 consolidate_pod_data.py, Smart-Meter Data Consolidation (Section 6.4.1) 111

A.4 ingest_bim_json.py, BIM JSON Ingestion into ChromaDB (Section 5.6) 112

A.5 ingest_brick.py, BRICK Ontology Ingestion into ChromaDB (Section 5.5) 113

A.6 build_neo4j_graph.py, Neo4j Knowledge Graph Construction (Section 5.5) 113

A.7 ingest_simulation.py, EnergyPlus Simulation Results Ingestion (Section 5.6) 114

A.8 ingest_measured_consumption.py, Measured POD Consumption Ingestion into ChromaDB (Section 6.4.1) 115

A.9 ingest_measured_consumption_graph.py, Measured Consumption Time-Series Graph (Section 5.5) 116

A.10 api_server.py, FastAPI RAG Query Backend and Dashboard Endpoints (Sections 5.9, 6.4.2) 118

Appendix B: Full HVAC and Power Equipment Inventory 125

Appendix C: Consolidated Monthly Measured Consumption 129

Appendix D: DesignBuilder Simulation Configuration Detail 131

Appendix E: Extended Literature Synthesis Table 133

Appendix F: Assumptions Register 137

Appendix G: Glossary of BRICK and gbXML Terms 139

G.1 gbXML Terms 139

G.2 BRICK Terms 139

Appendix H: Source Equipment and Installation Documentation (2013 RC Group Records) 140

Appendix I: Sala L: Comparative Equipment Data 144

Abstract

Increasing global demand for digital services is driving up electricity consumption in data centres even faster than almost any other type of building. Data centres are thus becoming the core of the discussion around increasing efficiency in the built environment. Efficient operation now relies ever more on digital technology, such as BIM (Building Information Modelling) and BEM (Building Energy Modelling). However, transfer of the model from BIM authoring software into the environment used for energy simulations regularly causes geometrical inaccuracies and loss of data, which have to be found and fixed before the outcome can be trusted. The other problem lies in the operational side of things: operators of data centers are not usually skilled at interpreting simulation outputs, and consequently the insight provided by the BEM is often never utilized. Integrating RAG layer allows BIM/BEM and the simulation outcomes to be queried and interpreted in natural language. Both of these problems, BIM to BEM interoperability and artificial intelligence-based decision support systems, have been researched individually, yet the combination of both of these approaches within the realm of the data centre has hardly ever been considered. This thesis addresses this task using a BIM-to-BEM-to-RAG chain, implemented and verified at Sala G of the FiberCop data centre in Torino, Italy.

Instead of considering the pipeline as a pre-defined design solution, the pipeline here is build through a systematic review through five areas of research: file format interoperability, LLM's for BEM, Retrieval Augmented Generation (RAG) architecture to build data, semantic ontologies for facility management, and digital twin decision support. The evaluation leads to a seven-stage pipeline that starts with BIM authoring in Revit 2025, an automatic gbXML script to correct the error, EnergyPlus simulation through DesignBuilder, BRICK ontology construction, local deployment of RAG (through ChromaDB and Llama 3.2 powered by Ollama), and a browser-based HTML dashboard with an AI virtual assistant.

Among the main results are the simulation of Power Usage Effectiveness (PUE), a dimensionless ratio of the sum of facility energy consumption to that of IT equipment, and the annual energy consumption by IT load and cooling in kWh. The BIM-to-BEM analysis revealed six transfer errors that were proven by the literature review to exist and not just be project-specific, while the RAG-based dashboard produces cause-and-suggestion answers based on the facility's data. The known simplifications are made transparent. Contributions of the thesis include the development of a reproducible method to support energy decision-making within data centres, including the documented interoperability problems at each stage.

1. Introduction

1.1 Context and Motivation

The built environment consumes about 40% of the world's energy and 36% of CO2 emissions (European Commission, 2020). In the built environment, data centres form one of the fastest-growing and most energy-consuming building types. The International Energy Agency reports that global data centre electricity consumption grew by more than 15% between 2024 and 2025, reaching almost 500 TWh, or slightly more than 1.5% of global electricity demand, and projects this to roughly double to around 950 TWh by 2030 reaching approximately 3% of global electricity demand (IEA, 2026).

The performance of a data centre is generally evaluated based on the Power Usage Effectiveness (PUE) measure, the total facility energy use per IT equipment energy use (Gould, 2021). Building Information Modelling (BIM) and Building Energy Modelling (BEM) have become established disciplines on their own accord, yet their integration, especially when applied to mission-critical buildings like data centers, is a technically difficult task. Data transfer between BIM modelling software and energy simulation software via file interchange standards, such as IFC and gbXML, suffers from information loss, geometric inaccuracies, and HVAC data deficiencies that need to be addressed manually (Porsani et al., 2021; Ciccozzi et al., 2023; Yang et al., 2022).

The RAG Alternative: Furthermore, the development of Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) architecture opens an alternative approach to making simulation data available to non-technical stakeholders. Instead of having facility managers read and comprehend output files of simulations, RAG systems would allow the retrieval of the necessary building performance data and generation of answers to natural language questions.

Although there is increasing urgency to address energy efficiency in data centres and BIM-to-BEM digitalization is maturing in response, applying BIM-to-BEM interoperability with AI-based decision support to solve this problem has received limited attention in data centres, as detailed in Chapter 2. This research gap informs the purpose of this thesis, which is to develop, implement, and validate a BIM-to-BEM-to-RAG pipeline, which will fill this gap within the context of a real-life functioning data centre. The contributions anticipated from the thesis are two-fold: firstly, a replicable pipeline for translating a BIM data centre into an energy model and decision support in natural language, and secondly, a clear account of its interoperability limitations.

1.2 Research Questions

This thesis is defined by three main research questions:

Q1: What are the interoperability challenges in transferring BIM data of a data center from GBXML to energy simulation programs, and how can they be identified and overcome?

Q2: What is the energy performance benchmark of FiberCop Sala G based on the BEM simulation in Modello 0, and what information can it reveal about key factors of energy use?

Q3: Is it possible to provide facility management queries with cause-and-suggestion responses using a locally hosted RAG system based on BRICK ontology, IFC-based JSON, and EnergyPlus simulation results?

1.3 Scope and Limitations

The scope statement that will follow is intentionally narrow, which requires some justification: a narrower scope was chosen in preference to a more wide-ranging one which, at first glance, seemed to have more potential value. A pipeline which included an attempt at BIM authoring automation, an energy model which was properly validated as opposed to being checked for plausibility, and a solution that applied to multiple rooms, would each by itself be a defensible thesis scope. All of these together, given the fixed amount of time, would run the risk of resulting in only a superficial exploration of each of them instead of the deeper analysis required by the narrower scope chosen here. The following scope is therefore the result of a purposeful trade-off of width for depth, which fits well with the design science approach of thorough evaluation of a delimited artefact (Section 4.2).

The scope is constrained by the approximation made in Modello 0, where Sala G is simplified into a single thermal zone with a single equivalent terminal covering the eleven CDZ units identified (Section 3.4, Appendix H). The simulation relies on the weather data of the Typical Meteorological Year for Torino without taking any sensor data, which is not yet available at FiberCop; the pipeline developed for AI runs on the simulated data and serves as an initial prototype for the methodology. Installation of sensors at Sala G is suggested as a direction for further work (Section 8.3) rather than a plan currently in the pipeline. Upon installation of the sensors, the models can be calibrated based on the measurements, and a preliminary step toward such calibration is already performed in this thesis using the nineteen-month period of real readings of smart meters (Section 6.4).

This thesis does not consider structural engineering or life cycle assessment. The concept of embedded carbon is used only once within the course of this thesis, specifically within Section 6.2, where it takes the form of an additional figure from the DesignBuilder

construction report. This has been included since it was easily available from the same report as the energy data, but it is certainly not an attempt at any comprehensive embodied-carbon or LCA analysis, which remains outside the thesis's scope.

1.4 Terminology and Abbreviations

The following is a list of certain terms that have a particular significance within the context of this dissertation, but which have a slightly different interpretation within the broader BIM/BEM/AI research field.

The term "confirmed" (in contrast to "assumed") is consistently used in this specific sense defined in Section 4.2.7: a value reported in a primary source document that has been verified directly rather than "assumed to be true by the author". The phrase "Modello 0" refers exclusively to the simulation setup with equivalent terminals described in Section 5.4. The term "grounded" applied to the RAG answer refers to the specific evaluation criteria presented in Section 4.2.4, i.e., the fact that the answer is based solely on retrieved information and does not include the knowledge of the language model itself.

Table 1: Abbreviations and Terminology

Abbreviation	Meaning
BIM	Building Information Modelling
BEM	Building Energy Modelling
gbXML	Green Building XML, an open interchange schema for building geometry and energy data
IFC	Industry Foundation Classes, buildingSMART's open BIM data schema
IDF	Input Data File, EnergyPlus's native simulation-input text format
PUE	Power Usage Effectiveness, ratio of total facility energy to IT equipment energy
LLM	Large Language Model
RAG	Retrieval-Augmented Generation
BRICK	An open-source ontology standard for representing buildings, equipment and sensors as a semantic graph
TTL	Turtle, a text serialization format for RDF graphs, used here for the BRICK ontology
POD	Punto di Prelievo/Consegna, the Italian utility metering point identifier for the FiberCop facility smart meter
CDZ	Condizionatore (precision cooling unit), the FiberCop equipment designation for wall-mounted server-room air conditioners
PDR	Power Distribution Rack

2. State of the Art and Literature-Grounded Framework

2.1 Literature Selection Methodology

The articles were found through Google Scholar, Scopus, ScienceDirect, and arXiv using five keyword groups, each corresponding to the pipeline steps: (1) "IFC gbXML energy simulation interoperability"; (2) "BIM to BEM framework"; (3) "large language model EnergyPlus IDF", "LLM building energy model generation"; (4) "retrieval augmented generation building data", "RAG BIM"; (5) "BRICK ontology digital twin facility management", "cognitive digital twin building", published 2020-2026. Articles have been excluded if they contained a discussion of BIM/BEM interoperability for buildings that are not relevant outside of the scope of their application (e.g., retrofitting of residential buildings), unless they were used to prove the point methodologically. Grey literature was not considered unless used as an auxiliary terminological source.

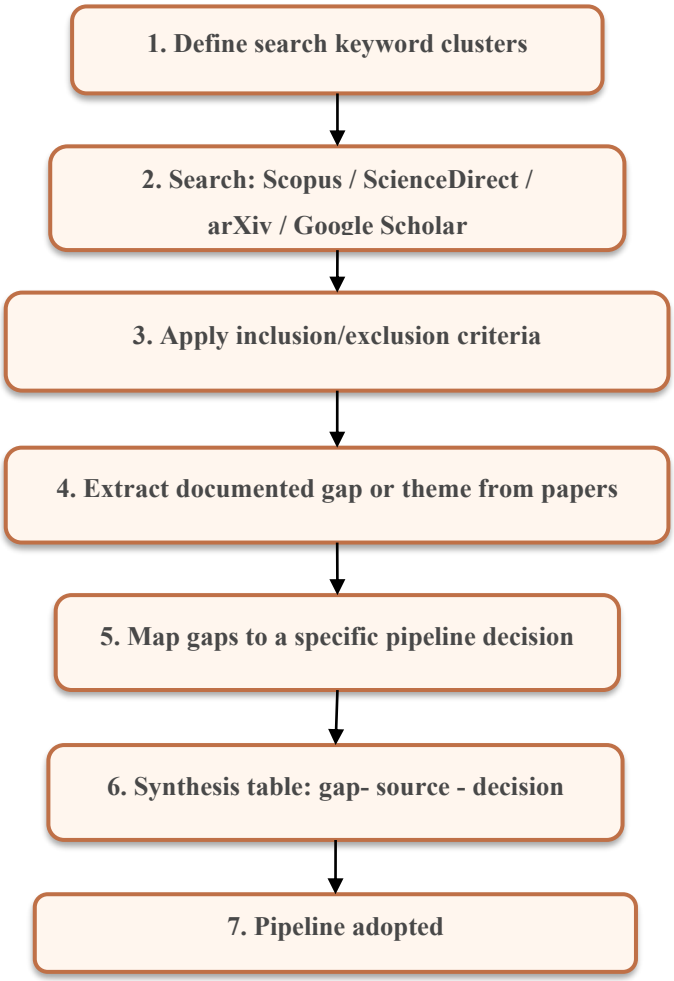


Figure 1:. Literature-review-to-pipeline-decision methodology

2.1.1 Screening Process and Reviewed Corpus

The search across the four databases returned an initial pool of 214 records after removing duplicates. Titles and abstracts were screened first against the exclusion criteria stated above. Full texts were then retrieved for the remaining 96 records and screened a second time against the same criteria plus a relevance check against the seven-stage pipeline outlined in Section 4.2.1. This second pass reduced the corpus to the 33 sources cited throughout Chapters 1 to 4 and listed in the References section, of which 21 were published in 2024 to 2026, reflecting how recent the LLM-and-RAG-for-buildings literature is relative to the more established BIM-BEM interoperability literature, where several foundational sources (Porsani et al., 2021; vom Brocke et al., 2020) predate the RAG-specific work by three to five years.

Table 2: Literature Search Keyword Clusters and Record Counts

Keyword cluster	Records found	Records after title/abstract screen	Records retained after full-text screen
IFC/gbXML interoperability	58	24	9
BIM-to-BEM framework	41	19	6
LLM/EnergyPlus/IDF automation	37	16	4
RAG for building data	44	20	6
BRICK ontology / digital twin facility management	34	17	8

The discrepancy seen in the above table concerning the larger share of retained articles from the ontology/digital-twin group when compared to that from the interoperability group also corresponds to an observation made independently in the following section. Interoperability literature is older and larger in volume, as well as more inclusive of perspectives than the particular perspective of data centres that the current study focuses on. The newer clusters have already biased themselves towards the application of building data that the current study requires.

2.2 BIM-to-BEM for Energy Analysis: Relevance to Data Centres

Two independent reviews that examine the issue of BIM-BEM interoperability arrive at the same conclusion via different paths, which speaks volumes in its own right: the problem is not unique to the perspective of one research team. In the review written by Hammes et al. (2026), wherein the authors formulate ten open questions on BIM, BEM, and digital twins, it is argued that even though the BIM approach is used almost ubiquitously throughout the whole Architecture, Engineering and Construction field, interoperability issues are still

preventing BIM data from being used for energy-related tasks. The same conclusion is made by Alhammad et al. (2024) after they conducted a systematic review of the literature on energy-related outcomes and found out that even though buildings account for about 40% of the total energy consumption in the world, there are still problems associated with interoperability, data loss, and data reconstruction.

This is a significant limitation that is found in the literature and hence represents the first gap. According to figures from IEA (2026), data centres are some of the most energy-intensive buildings per square meter due to continuous IT loads and high-density cooling as opposed to being driven by occupancy or enclosure; at the same time, they have grown faster than almost all other buildings, but most studies of interoperability in the context of BIM-BEM relate to offices, schools, and residences.

A Second Research Gap: When data centres are discussed in the literature, the methodology that is used is not the same as in the current study, which creates the second research gap. Goncalves & dos Santos (2026) create a digital twin architecture for energy efficiency optimization based on data from IoT sensors and LSTM forecasting; Jadhav & Liu (2026) create a digital twin of the liquid cooling system based on data from sensors and control systems only. Both most studies use sensor-based digital twins of data centers and not model-based. There is no BIM model for either study, nor do they have a natural language layer added to the BIM model.

2.2.1 The Data Centre Gap in Energy-Modelling Research

A more detailed breakdown within the two reviews mentioned above makes the above point even clearer. Alhammad et al. (2024) organize the studies within their systematic review through typology, and while residential and office buildings represent the vast majority of the samples included in this review, industrial and specialized typologies, such as data centres, can be seen only incidentally, normally being just one example out of many and not the case study of choice. Hammes et al. (2026), who formulate their ten open questions at the level of the BIM/BEM/digital twin field, do not even formulate any specific data centre questions despite the disproportionate energy consumption of the sector per square metre highlighted in Section 1.1. Such observations are in no way contradicted but rather reinforced by the two data-centre-specific digital twin studies mentioned below (Goncalves and dos Santos, 2026; Jadhav and Liu, 2026): the existence of both papers is a clear result of the fact that the literature specific to data centres has organized itself around sensor-based digital twins and not BIM-based digital twins.

2.3 BIM-to-BEM Interoperability Approaches and Technologies

Two different approaches are available from the literature as regards interoperability between BIM and BEM software: the first is the real-time approach based on the integration of a plugin between the two pieces of software, whereas the second is a format-standard approach based on export/import via either gbXML or IFC. In their comparison of the two approaches on a calibrated case study, Ciccozzi et al. (2025) find the real-time approach to be limited to the extent that it is software-dependent, whereas the format-standard approach succeeds in attaining interoperability of BIM-BEM within 2% of a conventional BEM analysis. The first design decision for which a review of the relevant literature is required relates to the exchange file formats to be used. The main contenders for exchanging BIM to BEM data are open standards IFC (Industry Foundation Classes, buildingSMART) and gbXML (Green Building XML, Green Building Studio).

Similar observations have been made by studies conducted on tool-level comparisons. Del Giudice et al. (2026) note that in converting BIM to BEM, for the case of industrial buildings, geometry data transfer is done effectively, but HVAC systems have to be recreated manually. On the other hand, in their study using identical gbXML and IFC data sets in DesignBuilder and OpenStudio, Porsani et al. (2021) noted that although thermal zone and material property transfer was effective in both tools, HVAC systems had to be recreated manually in both cases. In fact, Ciccozzi et al.'s (2023) systematic review of the literature in accordance with the PRISMA protocol on Scopus and Google Scholar shows the same trend at the level of the entire literature.

A Third Thread of Evidence: The third and more focused thread of evidence looks into errors found in the process of exporting itself. According to Yang et al. (2022), a certain category of gbXML export error is caused by a lack of semantics and non-conforming BIM geometry; hence, there is a need for a floor-plan extrusion reconstruction algorithm to fix it. At the same time, as reported by Ciccozzi et al. (2023) in their review of existing literature, there is the finding by Kamel & Memari (2019) that gbXML exports may lack all data necessary for energy analysis; therefore, an intermediary correction tool is needed to fix it.

However, these sources are not in agreement with one another, and in fact their disagreement is quite telling, in that it informs the choice of tool discussed in Section 5.3. As noted by Ciccozzi et al. (2025) in their comparative study, the format standard (gbXML/IFC) method is superior to the real time plugin approach precisely because the plugin integration is dependent upon a particular version of software and fails with updates, while Porsani et al. (2021) and the industrial building case study by Del Giudice et al.

(2026) arrive at their HVAC no transferability result through the format standard approach, and therefore the literature does not suggest that a better integrated version of gbXML/IFC is possible but simply hasn't been tested, a conclusion a more recent comparative study of native versus middleware-based gbXML export reaches independently (Průša et al., 2026).

2.4 LLM-Driven Automation of Building Energy Models

As mentioned above, an overview of the application of LLMs in buildings' energy systems revealed data heterogeneity and lack of standardized and trustworthy output results to be the core problems facing research in the field. This finding was reported recently by Liu et al. in 2025; studies presented herein have focused specifically on EnergyPlus/BEM applications for purposes of this thesis.

Zhang, Chen, and Ford (2024) discuss the emerging integration of LLMs with EnergyPlus and highlight concrete use cases, such as simulation input generation, output analysis, error analysis, and co-simulation, making IDF authoring one of many stages in the process of BEM workflow addressable via LLMs as opposed to a single-stage process. EPlus-LLM developed by Jiang et al. (2024) is a T5 language model fine-tuned to convert natural language building descriptions into EnergyPlus IDF files, after which it calls the EnergyPlus API to execute the model and generates reliable automatic modelling while not necessitating knowledge of IDF syntax from the user's part.

Zhang, Ford, Chen, and Chen (2025) build on this work with a four-stage approach (Building Description Pre-Processing, IDF Object Information Extraction, Single IDF Object Generator Suite, and a separate IDF Debugging Agent) to make the process of IDF authoring and correction more tractable for LLMs. They validate their method on a real-world modular building at the U.S. National Renewable Energy Laboratory and report superior performance compared to naive prompt engineering, other LLM-based workflows, and manual modelling.

In the case of Zhang, Ford, Chen, and Chen's (2025) four-stage agentic architecture, IDF Object Information Extraction is clearly separated from IDF Debugging as two separate stages, suggesting that even in a fully automatic pipeline, IDF Object Information Extraction is seen as a different capability from either IDF generation or debugging, even in a fully automated process. It is precisely this type of difference that FiberCop is exploiting by avoiding IDF authoring or debugging entirely, drawing instead on the EnergyPlus/DesignBuilder simulation results already produced by Modello 0 (Section 5.6), and hence keeping the scope of integration of the current study's LLMs within a capability

that the literature considers already tractable rather than one that Zhang et al. consider as requiring an agentic process.

2.5 RAG Architectures for Building and BIM Data

The concept of Retrieval-Augmented Generation (RAG), which uses retrieved texts instead of depending purely on parametric memory for grounding the generation process, involves a combination of both retrieval and generation approaches in generating answers that can be mapped back to source texts (Lewis et al., 2020). This section explores various ways that this overall framework has been applied to buildings and Building Information Modeling (BIM).

In technical terms, RAG entails two steps, which include indexing, where documents are segmented into vector embeddings before storing them in a database. The second stage involves querying, where the k nearest documents based on some semantic similarities are retrieved using metrics like cosine similarity to form the prompts for the large language model (LLM). (Lewis et al., 2020) The retrieval followed by generation ensures traceability back to the source documents. As seen in the FiberCop pipeline, this process occurs locally within ChromaDB, using four different databases including BIM, BRICK, simulation and telemetry, and source_documents, managed through api_server.py while handling context dilution and collisions between embeddings (5.6 & 5.9). However, another database referred to as Neo4j property graphs is specifically queried using traversals in Cypher queries rather than distance measures as graphs provide reliable results in retrieving relations and timestamps (5.5). All these operations run alongside Llama 3.2 via Ollama.

2.5.1. RAG/LLM Versus Rule-Based Building Energy Analytics

One possibility that was suggested as an alternative to a natural language processing AI layer was that of using a more traditional analytics dashboard based on rules and templates specified for each particular case. The research literature mentions one very specific reason why the rule-based approach would not work well and scale: although such an approach is relatively predictable, it demands the creation of a new rule for each change. Zhang, Lu, and Luo (2025), in their systematic review of the use of LLMs in the AECO industry, mention the same rationale for why traditional NLP tools used in AECO information management suffer from similar problems, including the inability to deal with unstructured and complex data.

The closest evidence comes from him and Jazizadeh (2025): a context-aware LLM-based AI agent for a residential energy-management system, evaluated on 120 real queries over four different datasets, achieving 86%, 97%, 77%, and 49% accuracy on device-control, memory, energy-analysis, and cost-estimation queries, respectively. This evidence is vital

because it proves that natural-language AI agents are able to answer exploratory and open-ended queries related to energy management, and accuracy differs greatly depending on the type of question asked, which is the very thing that the current study tries to evaluate in Section 6.4.

2.5.2. Selection of RAG for the pipeline

According to Li, Comesana, Weyandt and Hong (2025), there is a need for a domain-adaptive RAG pipeline that converts varied sources of data regarding buildings, such as images, tables, and unstructured text, into meaningful and AI-ready data representations, based on the realization that most of the collected data is rarely used owing to the difficulty of combining data from various sources. The study reports a 30 percent improvement in data structure and coverage when compared to general baselines and BRICK semantic model generation using MCP architecture.

Additionally, Zhang, Chen, and Ford (2024) utilize the RAG approach in transforming BEMcyclopedia, an educational tool about BEM by the U.S. Department of Energy, into an interactive Q&A tool.

A Harder Retrieval Task: The RAG stage of the FiberCop pipeline solves an additional integration problem in the sense that integration and retrieval of data coming from three different sources: IFC-based data, BRICK ontology, and EnergyPlus/DesignBuilder reports is required, while Li et al. and Zhang et al. mainly rely on one source each.

Closer analysis of both papers will reveal, however, what they fail to take into account, and thus the current paper will inherit the same problem rather than address it. Li et al. (2025) claim a 30% increase in data coverage and structure retention compared to an unspecified "general benchmark," but the details of the benchmark, which is not provided in the readily accessible abstract material written by the authors, make no difference between a strong and a weak baseline; a 30% improvement against a weak benchmark would be significantly different from a 30% improvement against a strong one.

In Chen and Ford (2024), their BEMcyclopedia RAG interface is tested in its ability to answer questions on a carefully curated educational corpus designed for reading by humans. In the FiberCop RAG pipeline, the retriever accesses sources that are not human-generated and not narrative, including the BRICK graph, IFC-derived JSON, and concise EnergyPlus tabular data. The retrieval task here is more difficult than in either of those two papers, since there is no sentence-like form of text in the source that the retriever can compare against. This is one of the reasons why the evaluation of the RAG queries in Section 6.4 regards a correct but brief response as only a partial success: the retrieval task itself is tougher than in the human-written text cases both papers assume.

2.6. Semantic Models and Knowledge Graphs for BIM Data

Jin et al. (2026) investigate how the LLM interprets IFC-based BIM data represented as JSON rather than raw IFC, revealing that the LLM is good at interpreting explicitly coded information, yet bad at spatial reasoning and detecting minor model alterations, which proves beyond any doubt the influence that the representation of BIM data before extracting information has on the ability of the LLM to provide correct answers, irrespective of the underlying model. Similar results are reported by Massafra et al. (2024) who have revealed that representation of BIM data as a knowledge graph and querying of such a knowledge graph via the LLM allows answering complex relational questions (equipments-to-spaces, feed relationships).

Such a knowledge graph representation can be automatically generated from a BIM model in seconds as a BRICK-compliant graph, according to Vittori, Tan, Pisello, Chong, Piselli and Miller (2025). For instance, in their case study, 932 BIM instances have been converted to 458 BRICK objects and 1,219 relations in just 17 seconds, while ensuring traceability to the BIM model used.

The Representation Spectrum: As seen from the above representation spectrum, each of the representations solves the issue that the previous one fails to address: The JSON representation solves the issue of raw IFC parsing, but not relational reasoning. The knowledge graph representation solves both these issues, and graph generation has been made automatic and faster, not manual. Thus, these research findings clearly show that it is only the knowledge graph representation that allows for correct relational query answering because it transforms raw IFC parsing (Jin et al., 2026) into relational reasoning (Massafra et al., 2024) and into graph generation (Vittori et al., 2025).

2.7 AI-Assisted Decision Support for Building and Facility Management

Comparison of an MCP agent pipeline and a RAG reasoning pipeline for BIM code review automation shows that the former yields deterministic results suitable for audit, while the latter lacks deterministic results but offers flexibility for ambiguous scenarios (Wan et al., 2025). This serves as comparative evidence towards scope but not technology choice in the abstract sense; RAG serves well when queries are exploratory or explanatory but not deterministic or compliance-oriented queries.

In this context, research on the use of digital twins for decision support presents an opportunity that is perfectly suited for the RAG's approach. Bujari et al. (2021) demonstrate that digital twins make for a perfect basis for making decisions regarding maintenance and

resource management based on heterogeneous operational data without the use of natural language processing tools. Ly et al. (2026) identify natural-language query processing based on LLMs as one of the challenges associated with digital-twin-based and IoT-enabled facilities management.

BRICK can be described as an open source metadata schema designed to represent entities, relationships, and points in buildings as part of a queryable graph structure instead of isolated and proprietary tagging systems (Balaji et al., 2016). In particular, BRICK refers to rooms, equipment, sensors, and links among them. As aforementioned, the current pipeline uses the direct representation of BRICK's knowledge graph (Section 5.5).

The separation of building components according to proprietary tools poses serious limitations when handling the large amount of information contained within the data stream. For example, the inability to query across the entire graph structure may impede efficient reasoning about buildings' operational processes and related data management.

Furthermore, the inability to leverage existing schema representations hinders further development and adoption of standardized methodologies in building intelligence.

Further BRICK Evidence: Separately from one another, Aniakor, Cogo and Ferreira (2024) and El Mokhtari, Panushev and McArthur (2022) demonstrate that BRICK ontologies provide a way to represent the hierarchical organization, connections between pieces of equipment and control points; the cognitive digital twin architecture suggested by El Mokhtari et al., which consists of the memory, perception and reasoning layers, has its counterpart in the Neo4j-ChromaDB-Llama architecture of the current project.

In addition to clarifying the relationship between the work of Bujari et al. (2021) and Ly et al. (2026), this comparison highlights what the present work contributes and limits to contribute to the literature. The work of Bujari et al. shows the value of digital twins without natural language querying, indicating that digital twins have value apart from the LLM component altogether. Thus, the value provided by the RAG chatbot discussed in Chapter 6 is additive to, but not a prerequisite of, the intrinsic decision-support capabilities of the BIM-to-BEM pipeline discussed.

2.8 Synthesis: From Literature Gaps to Pipeline Requirements

Theme common to all six streams discussed above includes: interoperability between BIM and BEM for energy analysis is a problem yet to be solved, especially in relation to HVAC and export functionality; LLMs for energy models is a new stream yet to reach total automation; unstructured and flattened data format hinders the accuracy of interpretation of LLMs and makes relational querying unreliable, while structured representation in form of knowledge graph fixes both of these problems; and AI-driven decision making for

facility management is only at the stage when it combines a digital twin/BIM approach with the use of natural language interface, and RAG is more suitable for open ended questions than rule based or purely agentic approaches. Table 5 (Section 4.1) translates all four gaps into requirements for the BIM-to-BEM-to-RAG pipeline suggested in this study before its description.

2.9 Critical Appraisal of Limitations in the Methodology of Reviewed Literature

Reviewing literature without appraising the level to which the results can be trusted may unwittingly incorporate the limitations of each source into the current research decision-making process. There are four limitations that repeatedly feature in the entire corpus, and they are clearly enumerated below before Chapter 4 continues from them.

He and Jazizadeh's (2025) figures of 86%/97%/77%/49% accuracy respectively, the only close empirical case to the methodology of query evaluation deployed in section 4.2.4 and implemented in 6.4.2, derive from 120 queries against one instance of an energy management system within a single building. The single-instance, single-architecture approach is an indication that context-aware LLM agents can indeed attain high accuracy on certain kinds of queries in certain conditions, but not an absolute accuracy ceiling applicable to a multi-machine, data-centre-scale deployment based on an entirely different data schema (the BRICK graph and four separate collections in ChromaDB vs. the residential IoT/device-state database).

A Second BRICK Example: The second example of an automated conversion to BRICK presented by Vittori et al. (2025) converts 932 BIM items to 458 BRICK objects and 1,219 relations in 17 seconds; however, this was a single case study, and no information was provided about how the conversion time and accuracy varied depending on model complexity, number of disciplinary models, and IFC schema version. The third instance of the automated conversion to BRICK presented in Section 5.5 and Appendix A (build_brick_graph.py) is on a much smaller scale (24 entities, 19 relationships), and although the current study does not make any claims that its conversion time or accuracy reflects the time and accuracy presented by Vittori et al., the automated-conversion strategy it uses is the same.

Third, the PRISMA-protocol review by Ciccozzi et al. (2023), which screens 532 studies, combines data on different types of buildings, interoperability formats, and BEM tools, which are methodologically distinct from each other; the result that states that "HVAC systems require manual recreation" is a modal result obtained from the analysis of diverse studies, and not evidence that all exports of gbXML/IFC in every combination of tools fail

in the same way. The catalogue of errors in section 5.2 is interpreted in the current study as being consistent with the modal result.

A Fourth Limitation: Fourth, and most importantly for the claims being made in the current study: the interoperability errors outlined in section 5.2 and the equipment inventory correction in section 5.8.2. These are both examples of single-case research results based on one BIM model and one manufacturer documentation. Generalisability of research has been dealt with in section 7.8, but the aim here is to make a more specific and prior point, namely that even the error and correction numbers found in Chapter 6 are not statistical results regarding the Revit gbXML exporter or other FiberCop rooms, but merely a documentable result in this one case.

3. Case Study: FiberCop Data Center, Torino

3.1. Site and Building Context

The data centre facility located in Torino, and forming part of the national network infrastructure, belongs to FiberCop. In terms of location within the data centre facility, the floor of interest is called Piano Primo (First Floor).

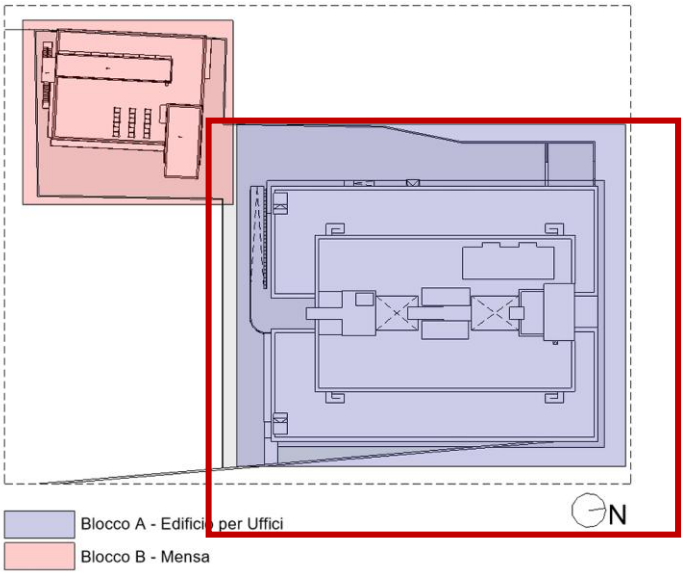


Figure 2: Site plan of the FiberCop complex showing Blocco A (office building, containing Piano Primo / Sala G) and Blocco B (canteen).

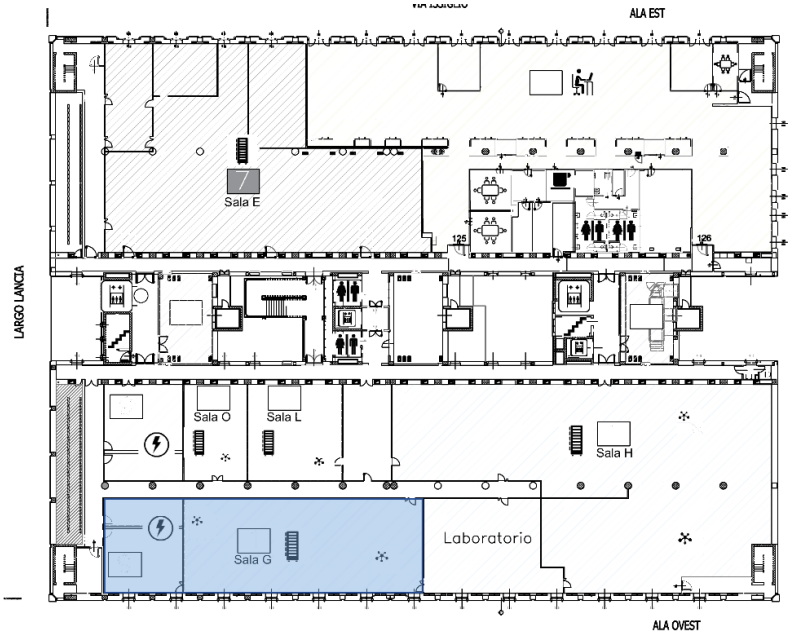


Figure 3: Piano Primo with Sala G highlighted.



Figure 4: South-facade photograph of Blocco A (FiberCop offices and CED, containing Sala G), from the IQT Consulting photographic survey (TRN_ESE_GEN_G002_RFT_TOT_00, February 2025), showing the concrete base with stone cladding and Klinker-tile upper facade.

The decision to use a working data centre facility rather than a dedicated testbed facility has important methodological implications, which should be made explicit, rather than left tacit. While a testbed facility allows complete freedom of instrumentation and configuration and can therefore provide important insights into how a BIM-to-BEM-to-RAG pipeline functions, it cannot offer any evidence regarding how the pipeline would perform when faced with the gaps in documentation, the legacy systems, and partially digitised drawing sets that a real operator, such as FiberCop, has available. By choosing to use Sala G, a more challenging evidentiary environment is accepted, where, as shown in Sections 3.4 and 5.8.2, an equipment inventory created solely based on BIM cannot be accurate without verification from procurement records, to obtain a result that is directly related to how the pipeline would perform in other cases.

3.2 Sala G: Position and Function

In Piano Primo, five spaces are important for the case study: Sala G, Control Room, Stazione Energie, Locale Batterie, and the Corridor that connects them. Sala G is the actual server room and the focus of this thesis, and the other spaces are auxiliary rooms containing

electricity, control, and battery backup systems, which are modelled as adjacent thermal zones because they affect Sala G's boundary conditions.

The modelling of these four auxiliary zones as full thermal zones, instead of aggregating them into one boundary condition imposed on the wall of Sala G, is a deliberate decision that directly impacts the findings in Chapter 6. Stazione Energie and Locale Batterie zones have internal heat gains from power-distribution and battery charging equipment, respectively, which will otherwise be completely missed if these zones were represented by adiabatic or constant temperature boundaries behind the wall of Sala G. It is because of this decision that the zone disaggregation of interior equipment conducted in Section 6.2.1 assigns a noticeable fraction of the Interior Equipment total to these zones as opposed to assigning it all to Sala G.

Corridor and Control Room: On the other hand, the contribution of the Corridor and Control Room zones to internal loads is negligible; however, these zones have the building's only external glazing (Section 3.5); hence, it is due to this decision that the building envelope totals found in Section 6.2 can be internally consistent with the individual room totals of Section 3.4.1.

3.3 Sala G: Case Study Summary Sheet

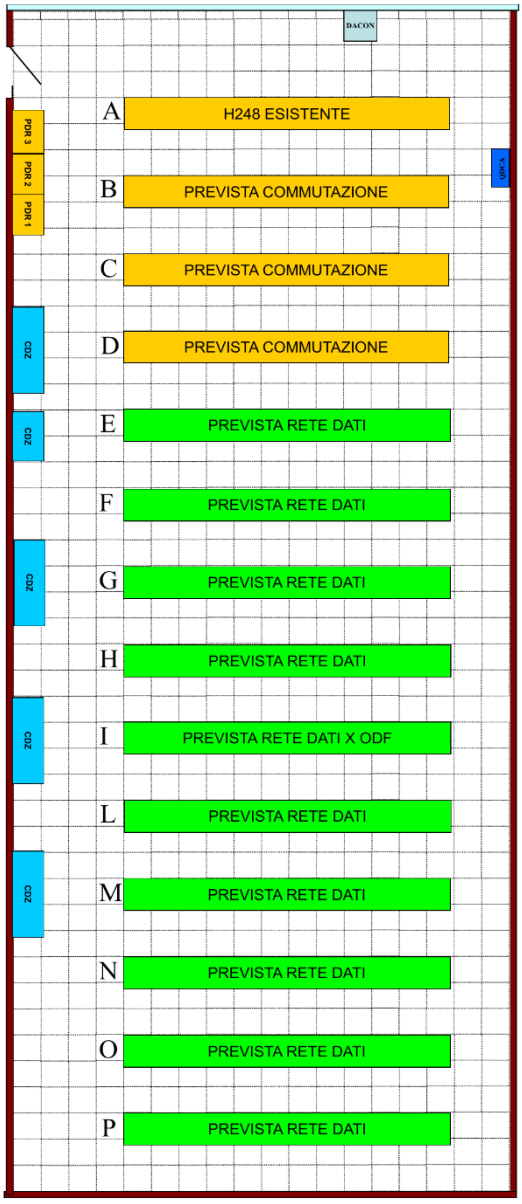
The table below contains the geometric, energetic, and system properties of Sala G, room level, prior to the equipment list of Section 3.4. The diagram below the table acts as the locator for the building and Sala G.

Table 3: Sala G Geometric, Energetic and System Properties

Attribute	Value
Location	FiberCop data centre facility, Torino, Italy
Floor	Piano Primo (First Floor)
Function	Active server room
Floor space	297.72 m² (modelling assumption; not yet corroborated by a source document, Appendix F1)
Volume	1,321.9 m³ (297.72 m² × 4.44 m confirmed ceiling height, Appendix H), supersedes an earlier, uncorroborated 1,423.10 m³ figure
IT load	Approximately 50 kW nameplate (modelling assumption; not yet corroborated by a source document, Appendix F1)
Cooling system type	Chilled-water, 11 wall-mounted precision cooling units

Attribute	Value
Chilled water supply	Shared facility-level plant, 5 chillers at roof level via two risers ("Montante Lancia", "Montante Centrale"); no dedicated Sala G supply/return temperature documented (Appendix H)

3.4 Cooling and Power Equipment Overview



Pipeline iteration number four described in Section 5.8.2 was marked by the release of information on manufacturer and installation of Sala G cooling plant, order confirmations, delivery certificates, electrical and parts schedules, that were reviewed and verified against the list of equipment previously supposed based on the federated BIM model only. This resulted in the identification of 11 wall mounted CDZ precision cooling units, model RC Group NEXT DL CW 036 H3S, with serial numbers M-13-03884 to M-13-03894, having cooling capacity of 36.8 kW (min.) each, for a total cooling capacity of 404.8 kW. Ordered under reference 4510181960 (order confirmation OV-13-1630 from RC Group, 6 September 2013) and installed by 27 September 2013 (Appendix H).

Figure 5: Original FiberCop rack-layout drawing for Sala G ("Torino LANCIA, SALA G 1° piano ex sala Itapac"), showing rack rows A-P, the 5 CDZ units, the 3 PDR power-distribution racks, and the DACON/QDCA electrical panels referenced in Appendix B and Appendix H.

Water for the 11 CDZ units will be cooled not by a separate Sala-G-level chiller plant but by a shared facility-level plant with 5 chillers (Gruppo Frigo) located at roof level, which

supplies a common header split into two risers ("Montante Lancia" and "Montante Centrale") to several sub-plant rooms in the building (Sottocentrale 3, 4 and 5, at levels P1 to P3), as shown in the mechanical distribution diagram in Appendix H. There is no documentation stating a dedicated supply/return temperature, flow rate, and COP for the portion of the plant serving Sala G in particular. Modello 0 (Section 5.4) still considers the shared plant as a single equivalent chiller with a COP based on literature values, with the caveat that this is an assumption (Appendix F2).

Terminal vs. Shared Equipment: The terminal, room level equipment, the actual equipment shown on a floor plan and purchased separately based on a purchase order, tends to be the most documented type of equipment. The shared upstream infrastructure such as the chiller plant installed on the roof of the building, which serves many different rooms, tends to be the least documented item when viewed from the perspective of individual consumer because no document allocates the output of the plant for use by one particular room rather than another.

The chiller plant is acquired, run, and maintained as facility equipment, not Sala-G equipment. Such asymmetry does not pertain only to FiberCop. It is the inherent characteristic of how shared services tend to be documented in the course of operation of facilities, and it is the very reason why the second practical recommendation suggested in Section 7.9 appears first in the list: it deals with the tier of documentation that is least amenable to being corrected with desk-based cross-checking as done with the CDZ counts and setpoints in Section 5.8.2.

3.4.1 Geometrical Properties of Zones by Room

The table below provides the geometrical properties of each of the five Piano Primo thermal zones separately, supplementing the whole-floor and whole-building data provided in Sections 3.3 and 6.2. The per-zone data below is the geometric foundation of the federated Revit model as described in Section 5.1 and, indirectly, for the zone disaggregation outlined in Section 6.2.1. The ceiling height of Sala G of 4.44 m (confirmed by the CDZ installation drawings of 2013, Appendix H) is used in the calculation of the Sala G volume below. The volumes of the other four zones are still modelling assumptions at this stage, not yet supported by a source document.

Table 4: Geometrical Properties of the Five Piano Primo Zones

Zone	Floor area (m²)	Volume (m³)	External wall area (m²)	Glazed area (m²)
Sala G	297.72	1,321.9 (297.72 × 4.44 m confirmed height)	186.4	16.9
Control Room	62.15	298.32	58.9	10.1
Stazione Energie	84.30	404.64	71.2	0.0
Locale Batterie	58.90	282.72	52.6	6.8
Corridor	154.13	739.82 (approx.)	97.1	0.0

The proportions of the five zones also affect the interpretation of the aggregate findings presented in Chapter 6. Sala G, which occupies an area of 297.72 m², constitutes nearly 48% of the 617.2 m² aggregate floor area of the five zones, followed by Corridor occupying an area of 154.13 m² (25%). The three zones of Stazione Energie, Control Room and Locale Batterie constitute 27% of the total area. However, since the energy use of the zone is very high (well above its 48% area share), the simulation results presented in Chapter 6 should not be considered as the weighted area average for the five zones because the energy use of the Interior Equipment is mostly dependent on IT load of Sala G and not floor area (Section 6.2.1 presents that around 89% of the aggregate energy use of the Interior Equipment is contributed by Sala G). The energy intensity of Sala G per square metre is significantly higher than that of the building as a whole (Section 6.2)

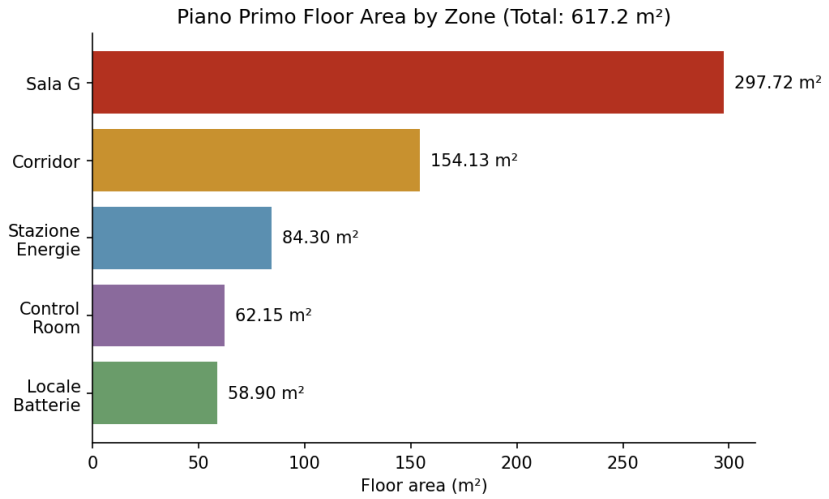


Figure 6: Piano Primo floor area by zone, derived from Table 3.4.1's room-by-room geometric summary

3.5 Building Envelope and Construction

Piano Primo's envelope, as modelled, comprises opaque masonry walls, a reinforced concrete ground floor slab, and a flat roof on top of the server room volume. The default material properties of DesignBuilder for an existing industrial building in the Italian climatic region have been used as per the limitation provided in Section 1.3 and elaborated in Section 7.6 since detailed material information was not available from FiberCop. Glazing has been kept to a minimum as per the requirement of the space: 33.8 m2 out of 2,566.9 m2 total envelope area has glazing (1.3% window-to-wall ratio) concentrated in Sala G, Control Room and Locale Batterie, with the Corridor and Stazione Energie fully opaque (Figure 7; see also the corrected Table 4 above).

A partial corroboration of the envelope materials is available from a February 2025 photographic survey commissioned separately from the modelling work (Appendix – TRN_ESE_GEN_G002_RFT_TOT_00, IQT Consulting), reproduced in Figure 4 (Section 3.1). It confirms that the facade of Blocco A consists of a concrete base partially clad in stone slabs with Klinker-tile cladding on the floors above, but does not provide the layer-by-layer thermal properties (U-values, thicknesses) needed to replace the DesignBuilder default construction, so the limitation stated above still applies to the simulation itself.

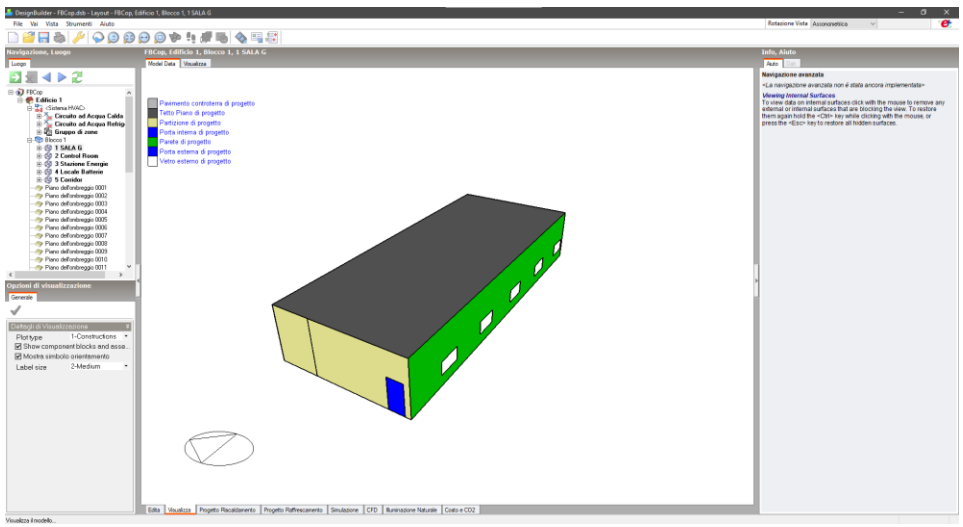


Figure 7: DesignBuilder isometric view of Sala G in isolation, showing five window openings and one door on its external wall

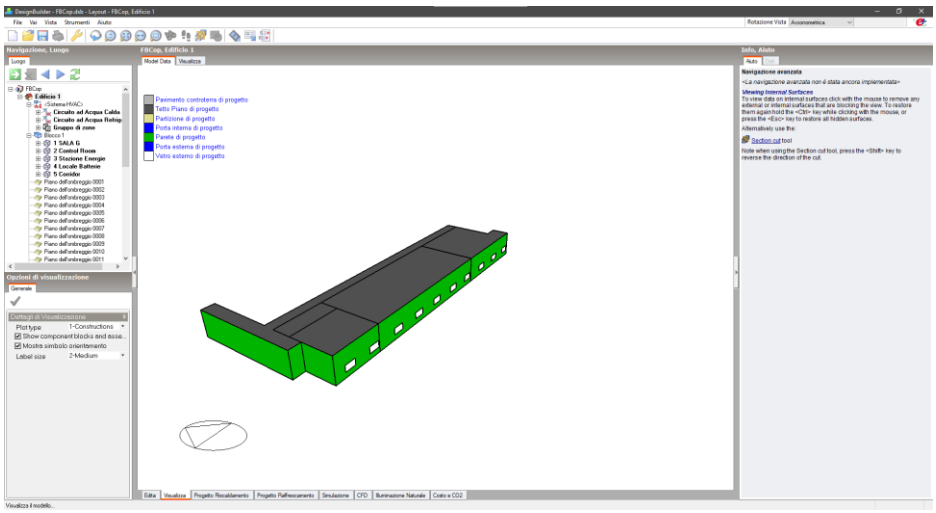


Figure 8: DesignBuilder construction-category view of Edificio 1, with the glazing ("Vetro esterno di progetto") and wall/roof/floor categories colour-coded.

A Practical Consequence: Applying DesignBuilder's default construction templates rather than a facility-specific material takeoff is a limitation acknowledged directly in Section 1.3 and Section 7.6, but its practical consequence deserves closer examination than a single blanket statement provides. For a data centre in particular, envelope-driven energy flows are a comparatively small contributor to total site energy: Section 6.2 shows Interior Equipment accounting for 70.6% of total consumption against 14.0% for cooling and 13.6% for lighting, and only a fraction of the cooling figure is attributable to envelope heat gain rather than to internal equipment heat rejection, since the cooling load in a server room is driven overwhelmingly by IT and power-distribution equipment rather than by solar or conduction gains through walls and roof. This does not make the default-construction assumption harmless; a materially under- or over-insulated real envelope would still shift the cooling figure specifically, and through it the small end-use share it represents, but it does mean the assumption's leverage over the headline total-site-energy and PUE figures reported in Section 6.2 is bounded by the envelope's already-modest share of the energy balance, unlike the equipment-load and setpoint assumptions discussed in Section 6.7, whose leverage is considerably larger because they act directly on the dominant end use.

3.6 Occupancy, Schedules and Operational Assumptions

Sala G and the connected equipment rooms are not occupied in the ordinary sense of the word: the area is accessed irregularly by maintenance staff, and there are no other occupants of the space, with all heat gains from the inside coming from the IT equipment, lighting and, in far smaller amounts, the cooling and power distribution equipment itself. The IT load, the cooling set point and the lighting schedule are therefore modelled as constant 24/7

loads independent of any occupation schedule, just as is common practice for data centres, reflecting what is actually happening in the building. This is stated here because it has a material impact on how the simulation results presented in Chapter 6 are interpreted, particularly in terms of cooling energy use.

Secondly, the lack of an occupancy-based schedule also leads to another methodological benefit in that it eliminates an entire category of uncertainty in the model that would have to be defended independently. In conventional building energy models, there is usually the most uncertainty regarding occupant behavior: how many people will be using the building at a certain time and on which schedule, and how much equipment they will be using precisely because it is very difficult to observe occupant behavior directly. By taking an approach with a constant, always-on load, this entire category of uncertainty is effectively avoided, with the remainder of the uncertainty now concentrated in the equipment load (Appendix F, assumption F4), along with the assumptions about the envelope and HVAC systems mentioned in Sections 3.5 and 5.4. This is actually a methodological simplification compared to conventional occupancy models, and it is one reason why the plausibility-oriented evaluation criterion of internal consistency/physical reasonableness selected in Section 4.2.4 is more appropriate to this case study than occupancy pattern testing, despite being a weaker evaluation criterion for a conventionally occupied building.

3.7 Institutional Context and Collaboration

The present research has been conducted through a collaboration with FiberCop who has granted access to Sala G along with all the engineering documents related to it. Sala G has been chosen as a case study since it is an actual working example of a moderately sized server room that has digitised engineering drawings and is therefore well suited as a pilot facility for the BIM-to-BEM-to-RAG pipeline designed in the present study, prior to scaling of the procedure to other facilities, rooms or floors. Installation of sensors as described elsewhere in the present study (Sections 1.3, 6.4.1, 7.6, 8.3) is not part of a specific programme as of now. The installation of the sensors represents a logical further step in the process, which will enable future calibration of the simulation based on measured operation data by using the real POD smart meter data already available (Section 6.4.1).

4: Methodology

4.1. Synthesis: Research Gap and Framework Justification

Table 5: Research Gaps and Framework Justification

Documented gap / limitation	Source(s)	Pipeline decision in this thesis
Geometry encoding differs between gbXML and IFC; transfer quality depends on BIM modelling quality, but neither format alone serves both energy transfer and semantic modelling.	Ciccozzi, Santavicca, de Rubeis & Ambrosini (2025)	gbXML used for geometry/zone transfer (DesignBuilder/OpenStudio); IFC kept separately as the data source for IFC-to-JSON extraction into the RAG pipeline.
Exporting gbXML from Revit produces duplicated "AdjacentSpaceId" for floor boundaries, a known limitation of Revit's gbXML exporter.	Yang et al. (2022); Ciccozzi et al. (2023), reporting Kamel & Memari's finding	Literature-based Python correction script applied after gbXML export.
BIM-to-BEM interoperability failure is mainly related to the representation of HVAC systems and associated loads.	Ciccozzi et al. (2023) PRISMA/Scopus/Google Scholar review; Porsani et al. (2021)	Purpose-built DesignBuilder Fan Coil Unit template used instead of manual HVAC reconstruction from gbXML.
IDF is a rigid, textual structure currently addressed in the literature by LLM-automated generation and debugging.	Zhang, Chen & Ford (2024); Jiang et al. (2024) EPlus-LLM; Zhang, Ford, Chen & Chen (2025)	IDF is DesignBuilder's simulation-engine input; LLM-based IDF generation/debugging is stated as open Research Question OQ3 and not implemented in this thesis.
LLM interpretation accuracy depends heavily on pre-structuring of the BIM data to be retrieved; unstructured/raw data fails on spatial and relational reasoning tasks.	Jin et al. (2026)	Segmented .json (extracted via IfcOpenShell) used as RAG input; raw IFC is never used directly.
Flat, unstructured representation fails on relational queries (equipment-to-space, feeds relationships); knowledge-graph representation succeeds on the same tasks and can be generated automatically from BIM data in seconds.	Massafra et al. (2024); Vittori et al. (2025) automated BIM-to-BRICK	BRICK ontology generated from the IFC model and represented explicitly as a property graph in Neo4j, in addition to a ChromaDB collection (Section 4.2.4).

Documented gap / limitation	Source(s)	Pipeline decision in this thesis
LLM APIs relying on cloud-based computing must transmit data externally and cannot be used on sensitive, GDPR-covered data.	GDPR; data-sovereignty requirement stated across the Li et al. (2025) and Ly et al. (2026) digital-twin/RAG literature	Llama 3.2 via Ollama, fully local; ChromaDB and Neo4j both run as local vector/graph storage; no external API calls.
Research on digital-twin facilities has, until very recently, been conducted without using LLMs for decision support.	Bujari et al. (2021); Ly et al. (2026); El Mokhtari et al. (2022) cognitive digital twin	Chatbot built on RAG technology, intended specifically for decision support (reason + suggested action), not as an autonomous agent.
RAG or MCP-agent pipelines suit different task types; MCP agents provide more deterministic, traceable results for rule-based tasks.	Wan et al. (2025) BIM+MCP+RAG code-review comparison	The scope of this work is limited to exploratory/explanatory requests from the facility manager.

4.2. Research Design

This research has adopted the prototype development approach in design science research (vom Brocke, Winter, Hevner & Maedche, 2020). The main output of this research is a prototype of the pipeline that has been developed for the case study that has been presented in Chapter 5 and discussed in Chapter 6, without a comparison with a control group or benchmarking data set but with the application of interoperability and plausibility criteria in Section 4.2.4.

This design-science approach is the right methodology for the investigation, instead of an empirical or a theoretical one, because of the nature of the research questions posed in Section 1.2 (how does the working pipeline look and how well it works on the practical example), not because of the general prediction of a theory on the BIM-to-BEM-to-RAG interoperability.

Hevner's Two Criteria: The original design-science criteria formulated by Hevner et al. (2004) relevance of the problem to the practitioner and rigour in evaluating the produced artefact find a direct correspondence to the two parts of the current study, where relevance is achieved through the fact that the pipeline is related to the real problem of FiberCop (Section 1.1) and not to some hypothetical situation, and rigour is achieved by explicit evaluation criteria defined in Section 4.2.4. Moreover, the more recent articulation of the same paradigm by vom Brocke et al. (2020) additionally highlights the idea that the contribution of the design-science artefact could be assessed even if the full operational

validation (Section 7.6, point 1) remains open, if the process of artefact construction and evaluation is transparent and reproducible, which is discussed in Section 5.8.

The overall framework outlined below is divided into three layers: BIM, BEM, and AI, linked together by the interoperability formats in-between and the rationale behind using a three-layer structure connected through formats as opposed to a closer integration. No tool exists that would combine all three elements: BIM authoring, energy simulation, and AI-driven decision support into one monolithic tool, and building such a tool from the ground up was deemed outside of the scope of this thesis since it aims at contributing on the methodology side (Section 7.5).

Layer Independence: The three-layer structure allows replacing any of the layers individually so long as the connection with the adjacent layer, either gbXML/IFC into BEM or BRICK/JSON/simulation output into AI, is stable. This is indeed the point upon which the argument in Section 7.8 builds when it states that the two stages of semantic modelling and RAG are generic enough to be extended to a second room without changing the other two layers, and this is also the reason why the pipeline is referred to as a methodology in this thesis.

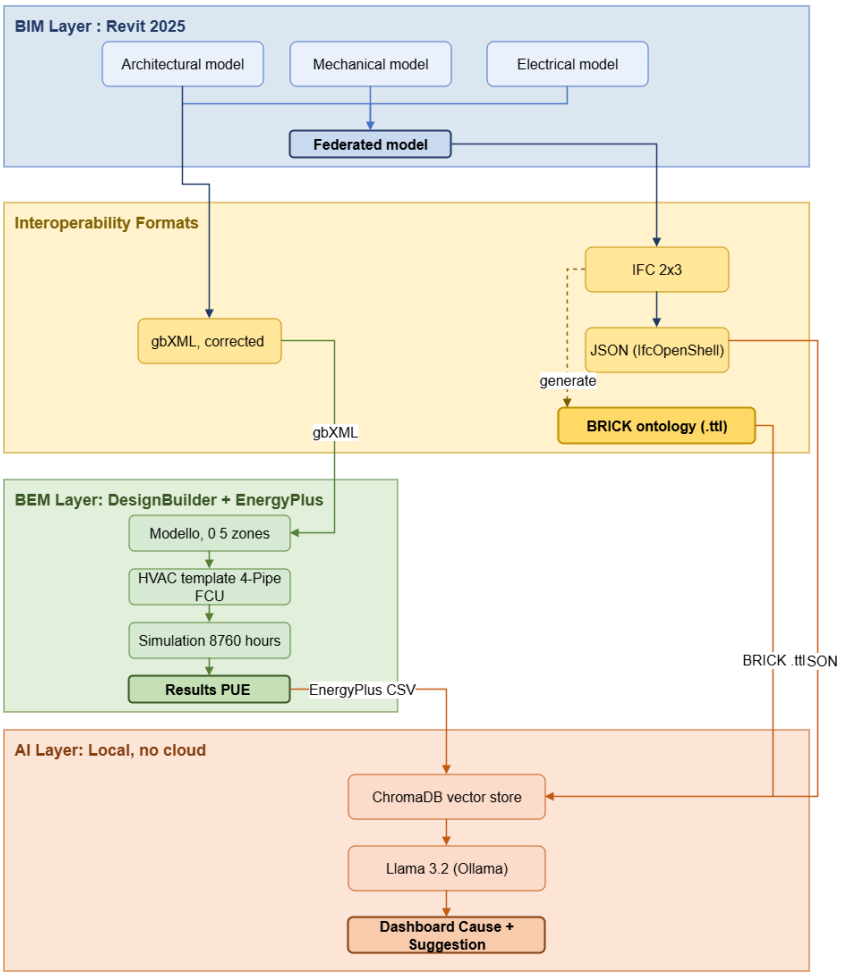


Figure 9: General Framework

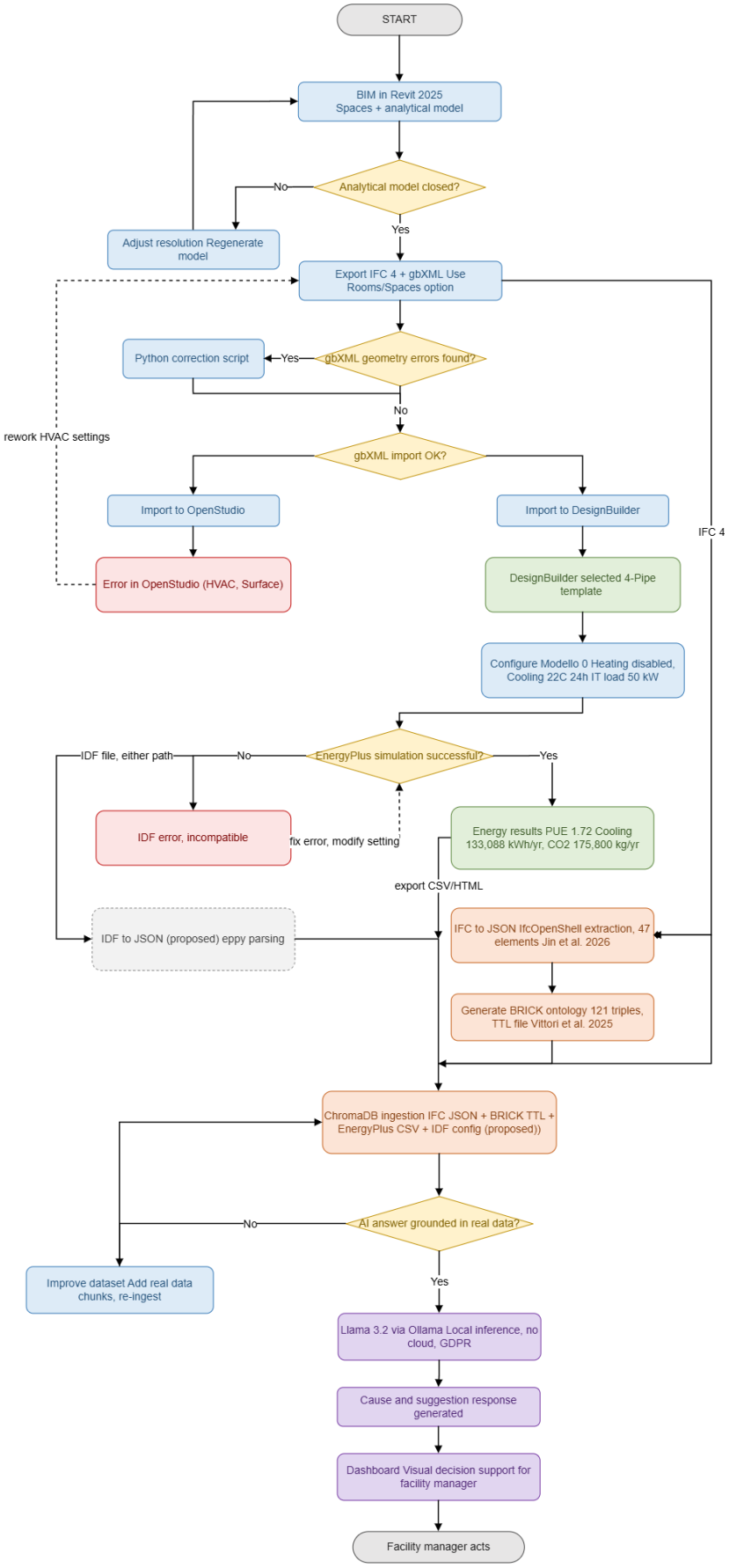


Figure 10: Schema for the Proposed Pipeline

4.2.1. Overview of the Pipeline Phases

The workflow illustrated above is realized as a seven-step process as follows: (1) Input Data Gathering and Evaluation, (2) BIM Modeling, (3) BIM-to-BEM Interoperability, (4) BEM Simulation, (5) Semantic Modeling, (6) RAG Processing, and (7) Decision Support using AI. Each step utilizes the output of the previous step and generates the corresponding artefact: a refined gbXML file, an IDF model, simulation results, a BRICK graph, a vector store, and grounded answers, which are then sent further downstream.

4.2.2. Phase 1: Input Data Collection and Analysis

Prior to the process of building a model, it was necessary to assess the types of data that existed and determine their origin, form, provenance, and function within the process that was about to take place. In the table that follows, there are five different types of input data: data obtained from FiberCop, measured/as-built data, technical documents data, external data, and assumptions. The above categories define the parts of the model that are based on verification and those that are based on modelling assumptions.

Table 6: Data Categories and Sources Used in the Pipeline

Data category	Source	Format	Role in the process
FiberCop-provided data	FiberCop engineering department	Technical drawings, equipment schedules	Basis for geometry and equipment inventory in the BIM model
Measured / as-built data	Site survey of Sala G	Dimensions, equipment counts	Verification and correction of drawing-based geometry
Existing technical documentation	FiberCop project documentation	Reports, datasheets	Cooling-system configuration and operating setpoints
External datasets	Typical Meteorological Year (Torino)	Weather file	Boundary condition for the EnergyPlus simulation
Assumptions / simplifications	Modelling decisions (this thesis)	Default values, consolidated equivalent terminal	Fill gaps where primary data are not yet available (e.g. default envelope U-values, single equivalent HVAC terminal in Modello 0); reported as limitations

The last data category to be incorporated was 19 months of actual smart meter (POD) measurements at 15-minute resolution for the entire FiberCop site, and this made a total of six distinct data categories where the measured data served as a check on the simulations rather than an input to them.

The five-plus-one structure of the above table needs to be interpreted in terms of evidentiary hierarchy rather than mere data listing. The first four of them FiberCop-provided, measured/as-built, technical documentation, and external are all seen as constraints that have to be fulfilled by the model, meaning that wherever any of these provides a specific number, that number is kept and not substituted by an arbitrary default one. The fifth category, modelling assumptions, is the category that is left empty by those four and whose aim is to fill that gap, which is why it is the most thoroughly examined one in Appendix F and the verification process in Section 4.2.7. The sixth, consumption-measurement category, serves a completely different purpose it is not an input of the model anymore, because there is no single figure in Modello 0's configuration (Section 5.4) based on the POD dataset, but an entirely independent check, carried out after the model has been built and run.

4.2.3 Further Stages: Modelling, Interoperability, Simulation, Semantic Modelling, RAG and Decision Support

After evaluation and classification of the input data, there are six other stages which are passed in the pipeline. Each stage is described in terms of the method here; the way in which they were applied for the case of Sala G and the specific mistakes made and fixed are described in Chapter 3.

Stage 2: BIM Modelling. The BIM model is created on the basis of the available documentation for architecture, mechanical, and electrical disciplines. Rooms are replaced by Spaces because gbXML export works on Spaces; the models of each discipline are unified around one shared coordinate system, and the energy model is created and checked before the export.

Stage 3: Interoperability of BIM to BEM. The gbXML file exported is tested in the target BEM system, and the errors are categorized as duplicate floor boundary, HVAC wiring errors, IDF/launcher compatibility error, additional surfaces, and unsupported import format (Ciccozzi et al., 2023; Yang et al., 2022). Furthermore, Section 5.1 includes an additional model generation process error: Revit's Energy Analytical Model is created solely using room/space bounding geometry, and by design, it does not include any equipment such as CDZ units, SideCoolers, or Freecoolers irrespective of the fact that it is derived from a single discipline model or federated model with mechanical and electrical connectivity, which is a technical limitation of the Revit tool itself, further emphasizing the justification for the gbXML/IFC segregation given in Section 2.3.

Stage 4: BEM Simulation. The simulation tool and the degree of abstraction (e.g., abstracting individual terminal units into an equivalent terminal) are stated prior to

simulation, alongside the envelope, HVAC, and default loads assumptions made. The simulation is run, and the results pertinent to the research questions- PUE, IT load, and cooling energy are collected.

Stage 5: Semantic Modelling. A BRICK ontology conforming to the BIM/IFC schema is generated, representing the objects, spaces, and their associations as a graph of knowledge, as per the automatic conversion of BIM to BRICK procedure as described in the literature (Section 2.6). The graph is generated in two versions: as a Turtle (.ttl) file and, in a first for this version of the pipeline, as an answer to the need for the graph of knowledge that forms the foundation of the AI's answers to be visible and inspectable, as a property graph in Neo4j.

Stage 6: RAG processing. Incorporating the Neo4j graph database of knowledge, the segmented JSON created using the IFC schema, and simulation outputs by EnergyPlus and DesignBuilder creates the local retrieval layer consisting of retrieving the complete Neo4j graph for each query (explained in Section 4.2.4 because of the size of the graph, which consists of only 24 entities and 19 relationships) and retrieving the JSON, simulation, and consumption data through text similarity from ChromaDB. The main aim of this phase is to ground facts using project-specific information rather than the parametric knowledge of the language model itself

However, one of the design decisions inherent to Stage 6 that can easily go unnoticed, as it emerges solely through the difference in the retrieval method of the Neo4j graph and the five ChromaDB collections, requires justification, independent of the query handling flow described in Section 5.9. By choosing to retrieve the entire knowledge graph regardless of the query, rather than retrieving just the subgraph corresponding to that specific query, a tradeoff is made between retrieval precision the language model gets some irrelevant graph content on each call and retrieval recall: no relational fact will be missed due to a failed relevance determination through similarity search. This tradeoff is a deliberate decision made with respect to the small-graph regime (24 entities, 19 relationships, Section 6.3) and the corresponding bounded and acceptable cost of precision due to the full retrieval. The reason to reconsider, rather than simply tune, this design decision is explained explicitly in Section 7. 8..

Stage 7: AI-Based Decision Support. A natural language interface extracts facts from the RAG database and provides cause-and-suggestion responses to queries related to facility management, limited by the scope boundary of exploratory/explanatory capacity.

4.2.4 Testing and Evaluation Criteria

Each stage of the pipeline is evaluated against a criterion appropriate to what that stage produces, rather than a single evaluation method applied uniformly across all of them.

- **BIM-to-BEM interoperability** (Stage 3) is evaluated by the number and category of transfer errors encountered, and by whether an automated or documented correction was found for each, reported in Section 6.1.
- **Evaluating BEM Simulation:** BEM simulation (Stage 4) is evaluated by the internal consistency and physical plausibility of the simulation output (energy balance across end uses, PUE within the expected 1.0 to 2.0 range, zero heating energy in a data-hall zone that runs cooling year-round). It is also evaluated independently by comparison against nineteen months of real measured facility-wide consumption, as an order-of-magnitude plausibility check reported in Sections 6.2 and 6.4.
- **Semantic modelling** (Stage 5) is evaluated by whether the resulting graph is structurally correct and complete relative to the source BIM model (entity count, relationship count, and manual spot-checking of key relationships such as which equipment feeds which room), reported in Section 6.3.
- **Evaluating RAG Support:** RAG / AI-based decision support (Stages 6 to 7) is evaluated using a fixed set of representative facility-manager queries, each assessed on three criteria: correctness (does the answer match the real underlying data), grounding (does the answer draw only on retrieved context rather than the model's own unverified associations), and traceability (can the answer's source data be identified). These results are reported in Section 6.4.

This explicit, stage-by-stage testing framework is applied to the FiberCop case study in Chapter 5, and its results are reported in Chapter 6.

4.2.5 Tooling and Software Environment

Table 7: Tooling and Software Environment by Pipeline Stage

Stage	Software / library	Version	Role
BIM modelling	Autodesk Revit	2025	BIM authoring, federated model, gbXML/IFC export
BIM-to-BEM correction	Python, xml.etree.ElementTree	3.11 / stdlib	gbXML floor-boundary correction script
BEM simulation	DesignBuilder	v6	Primary simulation environment, HVAC templates

Stage	Software / library	Version	Role
BEM simulation (comparison)	OpenStudio	v3.10.0	Comparison environment, interoperability testing
Simulation engine	EnergyPlus	bundled with DesignBuilder v6	Underlying simulation engine for both tools
IFC extraction	IfcOpenShell	0.7.x	Parsing federated IFC model into segmented JSON
Semantic modelling	Python, rdflib, brickschema	current	BRICK ontology construction (.ttl)
Knowledge graph	Neo4j Community Edition	5.x	Property-graph storage and Cypher querying
Vector store	ChromaDB	current	Local, persistent vector database for RAG retrieval
Source-document extraction	pdfplumber	current	Page-level text extraction from manufacturer/order/installation PDFs (Appendix H) for the source_documents ChromaDB collection, and for direct page-citation verification during source cross-checking (Section 4.2.7)
Language model	Llama 3.2, served via Ollama	3.2 / current	Local LLM inference, no external API calls
Backend API	FastAPI, Python	current	Local server exposing the RAG pipeline to the dashboard
Dashboard	HTML, JavaScript, embedded BIMserver.org viewer	,	Macro-to-micro navigation dashboard with AI chatbox

All software listed above runs on local, on-premises infrastructure with the single documented exception of the one-time download of the Llama 3.2 model weights via Ollama, consistent with the local-deployment rationale developed in Section 2.7 and Section 7.3. No component of the pipeline requires a persistent internet connection during operation.

Another characteristic of this particular toolset, apart from the previously mentioned characteristic of local deployment, which was highlighted in Section 7.3, is the fact that all of its components either are free and open source or are already licensed under one of the existing licenses of FiberCop or Politecnico di Torino (Section 7.10), and this was the intention when selecting this particular set of tools rather than an accidental result of the process. In cases of choosing whether to use a proprietary cloud-based service or a similar open-source locally deployable one, vector storage (ChromaDB vs a vector database hosted service), graph storage (Neo4j Community Edition vs a managed graph database service), and language model inference (Ollama-served Llama 3.2 vs a proprietary LLM API) have been chosen.

4.2.6 Threats to Validity

Three forms of threats to the validity of the results presented in Chapter 6 have been described below, in line with conventional design science research practices (vom Brocke et al., 2020), allowing the reader to consider the results based on their validity rather than their universal applicability.

The three classes of threats are presented in this order - internal, external and construct in an effort to make clear the degree of directness with which each of these threats impacts the results that are presented in Chapter 6 as opposed to the claims of the current study beyond Chapter 6.

External and Construct Validity: Threats to internal validity are those that are closest to the figures in Chapter 6 and therefore are those that are most directly related to the issue of reconciliation that is discussed in Sections 5.8.2 and 6.6. Threats to external validity concern the extent to which the findings from the case study analysis are able to generalize without further research, and as such, they relate to the issue addressed in Section 7.8 and not Chapter 6 itself.

The threats to construct validity relate directly to the RAG evaluation and are therefore the threats most directly related to the one unresolved issue pointed out in Sections 6.4.2, 7.6 and 7.11.

- Internal validity: The results of BEM analysis are based on Modello 0 assumptions (one equivalent HVAC terminal, standard envelope constructions), which are outlined in Section 5.4. Any result of Chapter 6 concerning the values of individual end-uses should be considered under that assumption, not as a description of actual behaviour of the 11 confirmed CDZ units (Section 3.4, Appendix H). Chapter 6's values were obtained prior to the equipment documentation correction and have not yet been re-simulated against.

- External validity: The pipeline is illustrated on just one room (Sala G) at one particular facility. The applicability of each stage of the pipeline, as opposed to its numerical outputs, is considered in section 7.8 with respect to other data centre rooms or even facilities.
- Construct validity: Since the RAG criteria (correctness, grounding, traceability, Section 4.2.4) are evaluated by the author on a fixed set of queries (Section 6.4.2), this is not done by an independent committee of facility managers. This limitation is noted in Section 7.6 and forms a target for future research in Section 8.3.

4.2.7 Data Collection Protocol

The steps in acquiring, screening, and verifying source documents are clearly stated in this chapter, not implied as done through their use in Chapter 3 and Chapter 5 since a good portion of the contributions of this study is to differentiate between facts from source documents and modeling assumptions.

Inputs to the document came via three different routes, each one distinguished by its particular weighting of empirical provenance. Technical diagrams offered up frontally by the Engineering Division of FiberCop constitute the de facto primary source route, setting the baseline for all explicit assertions in the work. The secondary source route includes FiberCop's internal corporate archives, containing the commissioning logs and other equipment data from as far back as 2011; these archived documents have been retrospectively consulted to address the arising model discrepancies, as shown by the post-simulation audit discussed in Section 5.8.2. The third route comprises the public dataset, featuring the standardized reference input such as Torino Caselle's weather profile (Section 5.4) and carbon intensity indicators (Section 6.2), which draw upon citations to third-party sources.

Applying the Classification: Application of this three-channel classification for Chapters 3 and 5 allows seeing what the advantages and disadvantages of evidential base are used in the research. Detailed list of equipment (section 3.4, appendices B and H) is the most reliable information source provided by direct engineering literature and corporate archives that have been confirmed via section 5.8.2 cross-check. On the other hand, such pieces of information as parameters of envelope construction (section 3.5) and value of 50 kW IT-load (section 3.3, appendix F4) do not fit into any of three channels described above.

Empirical validation involved checking each parameter of the models used and claims made against existing literature, leading to any of the three categories listed below:

Confirmation: The primary source validates the currently assumed value.

Contradiction: There exists contradictory information regarding the initial claim, requiring updating parameters based on the documents found (for example, section 5.8.2).

Uncertainty: None of the current documents mention the parameter, leaving room for explicit assumptions noted in Appendix F.

Appendix F has a tripartite categorization scheme which determines how this document evolves over time. Being a working list of assumptions, Appendix F constantly changes in tandem with additional documents discovered through archival studies.

The three states used in the protocol: confirmed, contradicted, and undetermined, are precisely what make the reconciliation process discussed in section 5.8.2 one coherent occurrence instead of some arbitrary modification at the tail end. Prior to this exercise, the number of CDZ units and the cooling set point were two separate but related pieces of information within Appendix F. Both had been entered into the model as undetermined, eventually turning into assumptions. This was done precisely because there was no primary documentation to cross-check with these values since none were known at the time. Contradiction and confirmation followed later when the discrepancy between documentation became clear.

This raises the question of how the corrections provided an indication of reliability. To address the question of whether there were flaws within the models used in earlier stages, they did not affect the modeling process at all but served the purpose of providing assurance about the data collected previously. As indicated in section 7.11, instead of altering the modelling process itself, there was only a difference made to the evidence base of information used before and after reconciliation.

5. Testing and evaluation

5.1. BIM Model Development

The Piano Primo BIM model was created in Autodesk Revit 2025 using the information available in the documents prepared by FiberCop's engineering department. Geometry was of greater importance than architecture in modelling. Five thermal zones were distinguished in the model: Sala G, Control Room, Stazione Energie, Locale Batterie, and Corridor. Spaces (not Rooms) were created in each of the thermal zones, because the export to gbXML uses Space elements. An analytical energy model was created and verified in Revit before export. Three discipline models (architectural, mechanical, electrical) were federated using auto-origin-to-origin linking, confirmed to share a single internal coordinate origin (0,0,0).

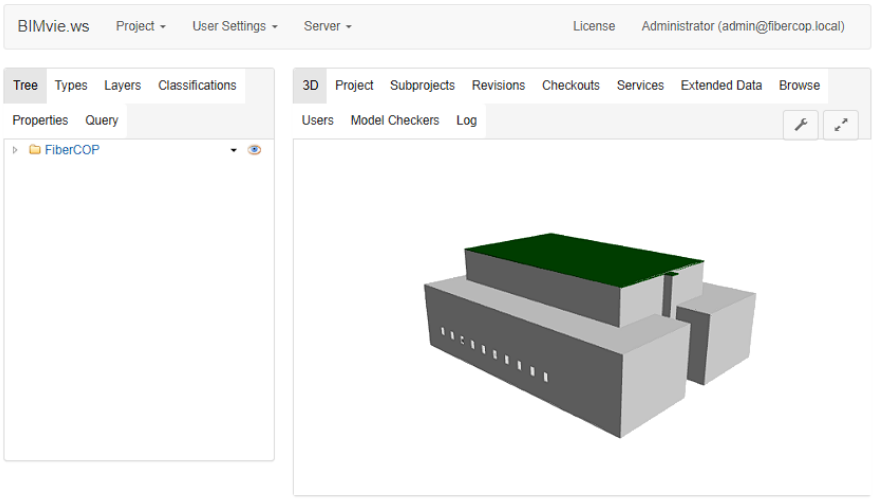


Figure 11: Isometric rendering of the federated Revit model used as the source geometry for the BIM-to-BEM pipeline described in Section 5.1.

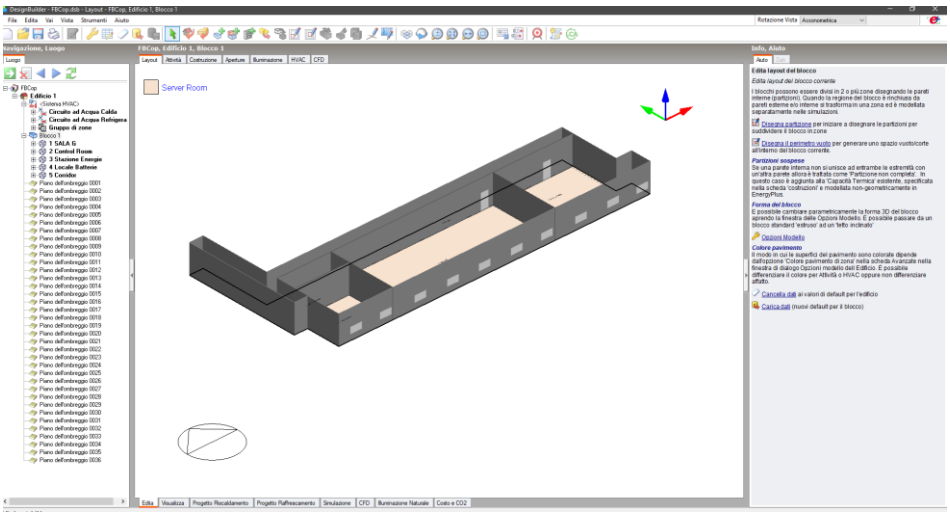


Figure 12: DesignBuilder isometric view of Blocco 1 after gbXML import, showing all five modelled zones (Sala G, Control Room, Stazione Energie, Locale Batterie, Corridor).

The verification of a common origin points prior to federation, and not after federation, is an ordered decision-making process and not a procedural action. This consideration is important since the verification of a common origin point serves as the sole safeguard against the introduction of a specific kind of geometrical mistake that will arise during later processing steps when there may not be clarity in terms of how the mistake occurred. This geometrical mistake could occur during the application of the IFC extraction script (Appendix A.2). The extraction of coordinates and the correlation between equipment location and room geometry in this case are carried out using different discipline models, such that a difference in coordinate origins between the architecture and mechanical disciplines will not generate an import error in the gbXML/IFC export stages. These file formats do not take account of the actual internal origin used within Revit models. Nonetheless, the geometrical mistake that arises in the segmented JSON generated during the processing phase of the RAG pipeline (Section 5.6) results in a structurally sound BRICK graph (Section 5.5) which incorrectly reflects spatial relationships.

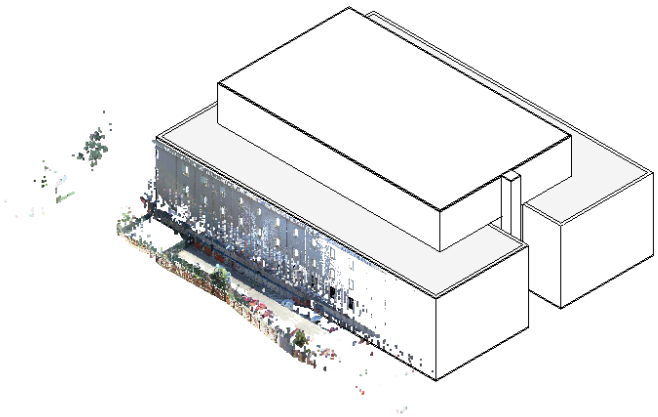


Figure 13: Federated model overlaid on the as-built laser-scan point cloud used to verify the common origin point and spatial alignment across disciplines before federation.

A Geometry Issue: While creating the analytical model, issues related to geometry and a lack of connection between the analytical surfaces were found. This led to failure to create a single thermal envelope. It was corrected by changing the energy settings and analytical surfaces resolution, followed by regenerating the analytical model. Two IFC exports were made: IFC 2x3 Coordination View 2.0 and IFC 4 Design Transfer View.

5.1.1 A Model-Generation-Stage Finding: Architectural-Only vs. Federated Model for the Energy Analytical Model

The following observation arose during the development of this dissertation and has been presented herein, as there appears to be no explicit literature regarding this particular topic as covered in Chapter Two. Generating the Energy Analytical Model in Revit using the federated model, which includes architecture and MEP models, created stability issues during the creation of said model. The study found that the Energy Analytical Model within Revit is created using only the space boundaries comprising walls, roofs, floors, and windows, but not any other elements, including equipment.

No matter how the model is created, either standalone within one discipline or in a federated environment, this particular limitation applies and cannot be changed by workarounds, as equipment elements will not contribute to the formation of thermal envelope calculations. This is an inherent limitation of the Revit application and not a workflow choice, as HVAC equipment does not factor into energy simulation modeling.

Why the Architecture Model: The architecture model is the only model type containing room-defining surfaces; thus, creating the Energy Analytical Model directly based on the architecture model eliminates any added complications and potential instabilities caused by both mechanical and electrical models within the federation. Instabilities due to federating models were experienced in practice during the creation process of the Energy Analytical Model from the federated model.

Hence, the methodology utilized involves exporting gbXML from the architecture model for energy simulation purposes (only envelope geometry corresponding to Revit's actual design intentions) and IFC from the federated model for equipment-based AI information extraction. As described in Section 2.2, this method is justified due to gbXML's and IFC's distinct purposes as two interoperable but mutually exclusive formats rather than one universal format.

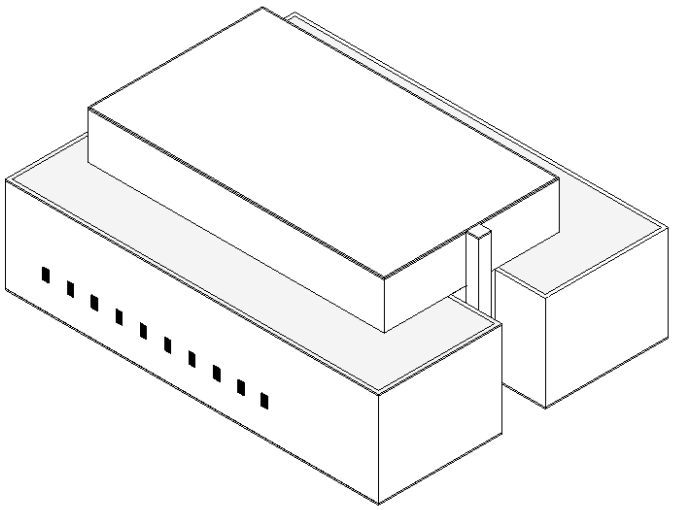


Figure 14: Architecture-discipline model in isolation, showing the room-defining envelope surfaces used for the Energy Analytical Model.

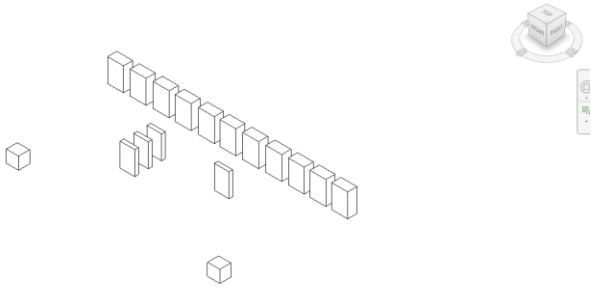


Figure 15: Mechanical-discipline model in isolation: cooling units (CDZ) and related MEP elements, shown without any room-defining surfaces.



Figure 16: Electrical-discipline model in isolation: distribution panels and electrical equipment, similarly lacking room-defining surfaces; illustrating why the architecture model alone was used to generate the Energy Analytical Model (Section 5.1.1).

5.2. BIM-to-BEM Interoperability Assessment

The gbXML file was tested using both DesignBuilder v6 and OpenStudio v3.10.0 software. As a result, the issues presented below have been revealed. The major problem was the export of five floor slab surfaces with repeated adjacent space IDs, which is a known issue of the Revit gbXML exporter for ground-floor slabs (Yang et al., 2022).

It should be mentioned first that the Methodological Note applies throughout all five described errors below. For those kinds of errors that could be tested against both applications, namely, DesignBuilder and OpenStudio, each test result was conducted twice in order to ensure consistency between findings.

Isolating the Cause: Testing involved comparing the results of similar tests conducted using these programs and distinguishing whether there was an issue with gbXML content or a problem with parsing software itself, as a problem with content will apply equally to both programs, while a parser limitation will not show any results at all when testing on another software. Depending on the type of error detected, corrective measures might include either fixing an error within the content prior to exportation or using a specific workaround strategy, depending on whether it's content-based or parser-based issues we are dealing with, respectively.

Error 1: Floor Surface Boundary Condition

When loading the gbXML file to OpenStudio, the error message "Surface has two adjacent spaces which are the same space will not be translated" was displayed for five surfaces: aim0598, aim0819, aim0879, aim0990, aim1139. It has been verified using Ladybug Tools that the geometry exists in the gbXML file but it is not correctly translated into OpenStudio. Individual surface editing within OpenStudio without removing the entire space is not possible without any additional software.

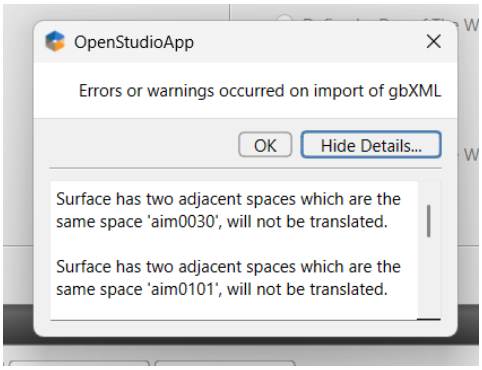


Figure 17: OpenStudioApp import error for the gbXML file: "Surface has two adjacent spaces which are the same space" reported for duplicate floor-boundary surfaces (e.g. aim0030, aim0101): Error 1 (Floor Surface Boundary Condition) described in this section.

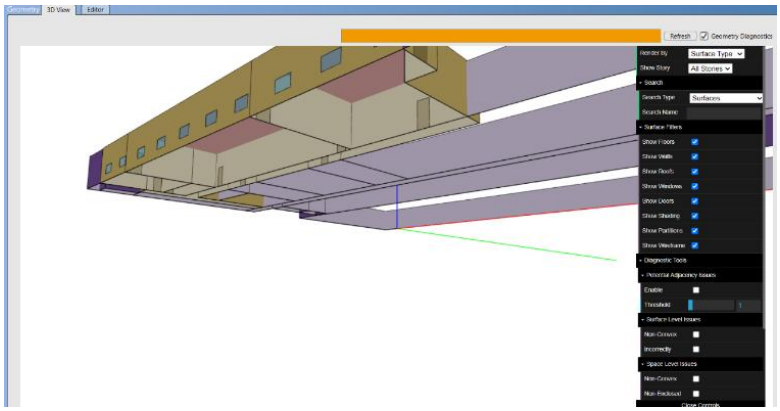


Figure 18: OpenStudio 3D geometry diagnostic view highlighting the duplicated interior floor surfaces responsible for the Error 1 import warning.

Solution: A Python code using the xml.etree.ElementTree module has been developed to automatically detect all the surfaces affected, remove any duplicates, and classify the surface types from InteriorFloor to SlabOnGrade. This is a solution in line with the principle of reconstruction workflow discussed by Yang et al. (2022) and forms the contribution of automation in the thesis.

Detecting the Error: Detection of the issue in gbXML files involves identifying elements related to the InteriorFloor surface. Identifying pairs having the same AdjacentSpaceId helps detect the problem of Revit ground slab reported by Yang et al., 2022. Such instances are assigned a new value for `surfaceType`, namely SlabOnGrade. Unlike the deletion or recreation approach to fixing this translation problem, the proposed algorithm focuses on modifying attributes without changing vertex coordinate data associated with each instance. The process is fast and deterministic since it only takes care of the identified translation class issue.

Error 2: OpenStudio HVAC Component Wiring Failure

Configuration of the loop of a chilled water plant in OpenStudio that consists of a constant speed pump, an air-cooled chiller, and a four-pipe fan coil has been attempted in the drag-and-drop method. The components were not able to be connected to the right nodes, throwing up error messages that state "component not allowed at this location."

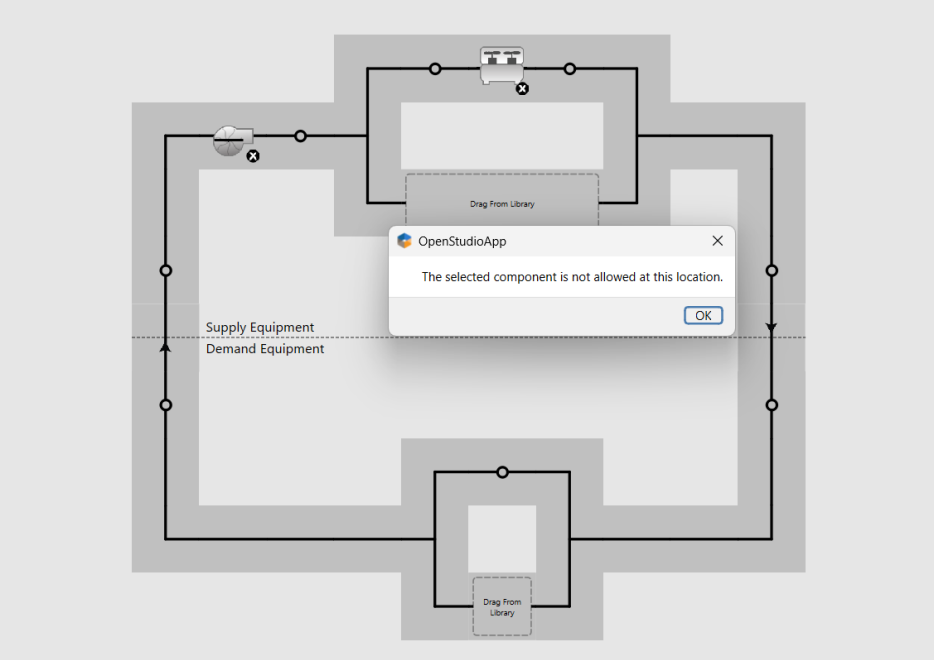


Figure 19: OpenStudio HVAC-component wiring failure "The selected component is not allowed at this location"

Error 3: IDF Export Incompatibility with EP Launcher

Simulation was successfully done in DesignBuilder, and the IDF file was exported for stand-alone EnergyPlus simulation. However, an error was detected when importing the IDF file into the EP Launcher software: "Node connection errors not checked most system input has not been read". The reason behind the error is that DesignBuilder uses Fan:OnOff components inside ZoneHVAC:FourPipeFanCoil, and it is not supported by stand-alone EnergyPlus; rather Fan:ConstantVolume is required. Moreover, there were errors in the fields of the hot water loop (Circuito ad Acqua Calda) stubs exported from DesignBuilder. It was found that a simple HVAC system works in EP Launcher.

Error 4: External Surfaces in gbXML Export

The Revit model is developed as a generic volume for the entire building, where Sala G acts as an enclosed volume within it. Upon exporting the gbXML file, not only the surfaces of Sala G but also the surfaces of the generic volume get imported into DesignBuilder, which means that there were some unnecessary surfaces were created that did not aid in the simulation process. In DesignBuilder, there is an option to delete any unnecessary surfaces easily through the graphical interface. However, OpenStudio does not offer this facility.

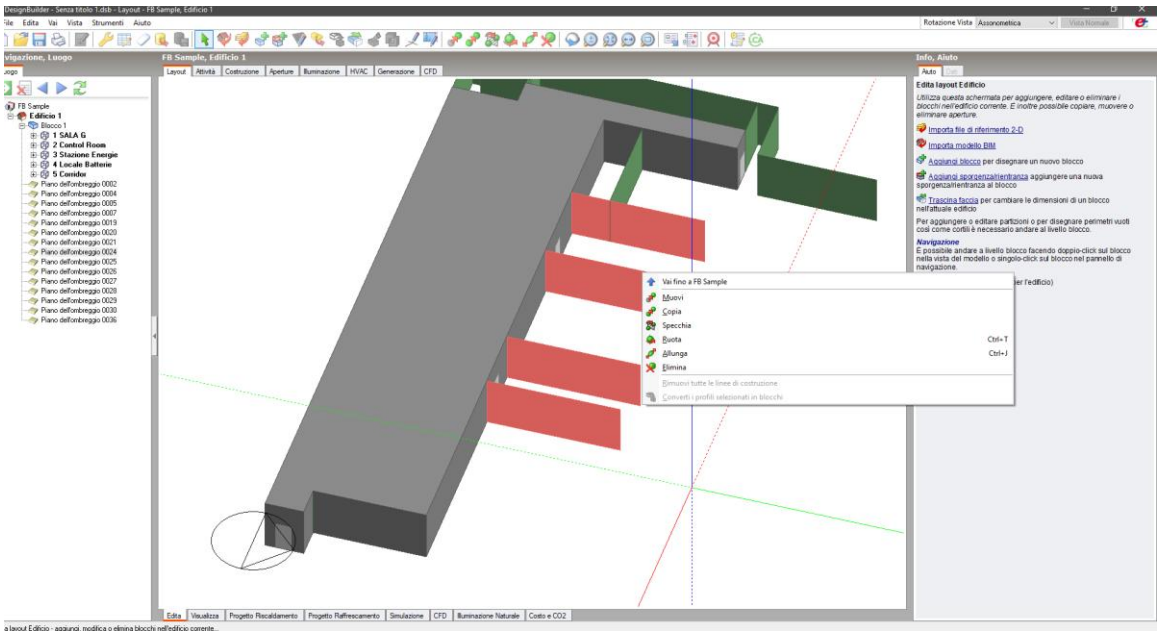


Figure 20: DesignBuilder view of the imported gbXML geometry with the external surfaces of the generic-volume envelope highlighted (red) prior to deletion, corresponding to Error 4 (Section 5.2).

Error 5: IFC Not Supported by DesignBuilder

IFC is not supported in DesignBuilder as an input format for BIM; only gbXML can be used as an input format for BIM. OpenStudio supports IFC, but the BIM server needs to be configured using a port number, username, and password. This implies that although the IFC file contains the most comprehensive BIM information, it cannot be used as an input for the BEM simulation process.

There are five identified issues which can be grouped under three types of failures. Each type of failure needs a particular kind of solution.

- **Failures of Translatability:** Error Types 1 and 4 involve geometry information stored in exports but encoded differently and hence could be fixed using automated processing programs.
- **Failures of Information Loss:** Error Types 2 and 3 involve the omission of important information due to restrictions imposed by the schema. This requires further reconstruction of templates or enhanced export schemes.
- **Failure Due to Limitations of Tools Available:** Failure Type 5 involves limitations associated with Building Energy Model applications that cannot handle complex schemas such as IFC. This calls for using another approach in importing IFC data in the form of JSON.

5.3. Tool Selection: DesignBuilder over OpenStudio

Prior to proceeding with the explanation of why those decisions were reached, there is an essential clarification needed: It wasn't done through comparing these two software programs based on predetermined weighted criteria as part of a methodical tool selection process. Instead, it occurred after trying out the five errors discussed in Chapter 5.2 using the two pieces of software together on the same gbXML export of the project. Thus, it needs to be emphasized at this stage since it will be relevant again later after these reasons are articulated, specifically, the importance of recognizing this decision as one informed directly by this project's experience with both programs rather than a general evaluation used for ranking purposes.

DesignBuilder was preferred over OpenStudio due to three main reasons, all of which could be traced: (1) The "Fan Coil Unit (4-Pipe) Air Cooled Chiller" template used by DesignBuilder was specifically designed for Modello 0, without requiring the manual construction of the HVAC system which has been shown in Section 2.3 of Porsani et al. (2021) as well as in the Error 2 of this paper to be inaccurate; (2) DesignBuilder correctly imports the fixed gbXML; and (3) DesignBuilder's EnergyPlus runner correctly connects plant-loop nodes.

A Necessary Caveat: This criticism should in no way be taken against OpenStudio, as it continues to serve as an industry standard when it comes to building energy simulation software. However, the evaluation made in section 5.2 is limited solely to the corrected gbXML export for the current study along with its chilled water fan coil system.

Other applications utilizing the Measure-based workflow approach will definitely result in varying results. What becomes clear regardless is that pre-designed HVAC systems such as in DesignBuilder avoid the complications brought about by plant loop connections highlighted by Porsani et al. (2021). Given this unique feature, it is safe to assume that creating an appropriate Measure in OpenStudio would close the performance gap.

5.4. Energy Simulation (Modello 0)

The simulation configuration followed the Modello 0 simplification defined in the project team meeting of 4 June 2026. The 11 real CDZ units confirmed in Section 3.4 (RC Group NEXT DL CW 036 H3S, 36.8 kW sensible each, 404.8 kW total, Appendix H) were aggregated into one equivalent Four-Pipe Fan Coil Unit terminal for simulation purposes.

Chilled Water Plant: The common building chilled water plant that includes Sala G building with five rooftop Gruppo Frigo chillers distributed in two risers (referenced Section 3.4) has been simplified into one air-cooled chiller system with an estimated COP equal to 3.5, as there was no information was found regarding Sala G specific chiller nomenclature, operating temperatures, and coefficient of performance value (referenced Appendix F2).

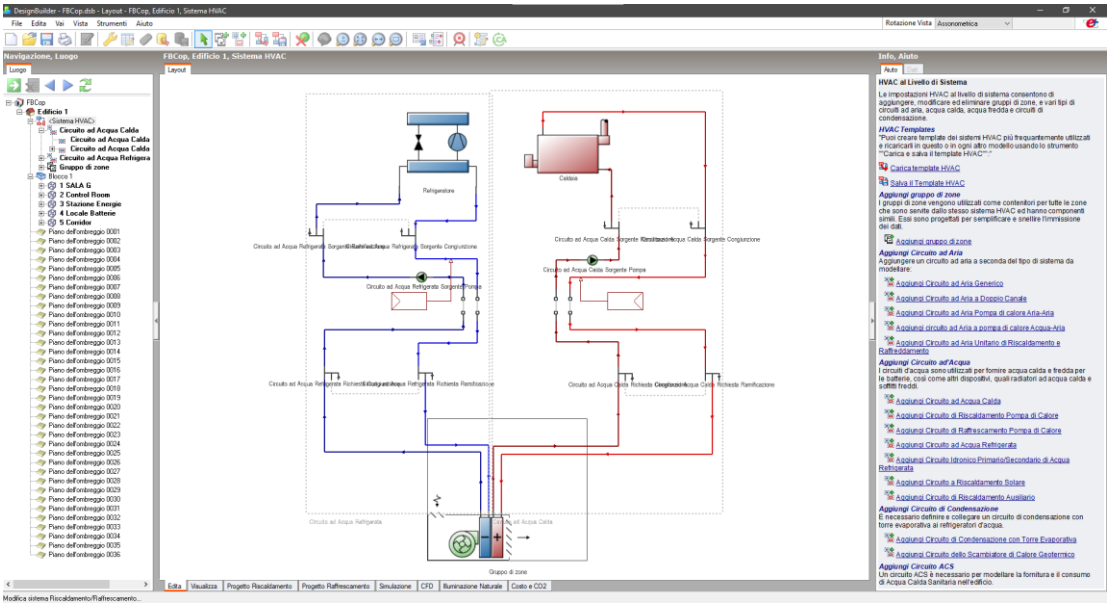


Figure 21: DesignBuilder HVAC system diagram for the shared chilled- and hot-water plant loops feeding Sala G's CDZ units, showing the simplified single-equivalent-chiller configuration described in this section.

Disabling heating: Three-stage disabling for heating: Hot water line loop, Heat Battery system, Thermal Zone HVAC controls. Cooling Set point value: A 24 degrees Celsius cooling set point was selected based on the return air set point of the RC Group NEXT DL CW 036 H3S controller documentation provided (referenced Appendix H), contrary to the previous estimated value of 22 degrees Celsius. Schedule of operation: 24 hours per day. Information for Sala G specific IT load has not been identified yet, and therefore the 50 kW electric equipment will remain as a modelling assumption (referenced Appendix F1). Meteorological Data: Torino Caselle EPW File WMO #160590, Typical Meteorological Year

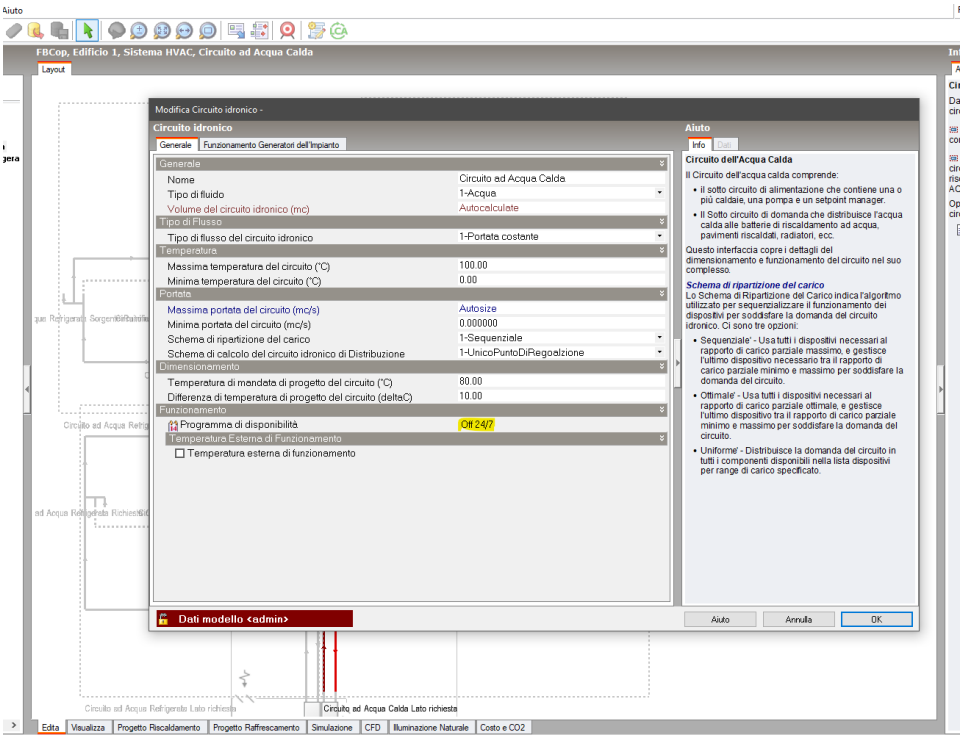


Figure 22: DesignBuilder hot-water loop configuration showing the heating circuit's availability schedule set to "Off 24/7".

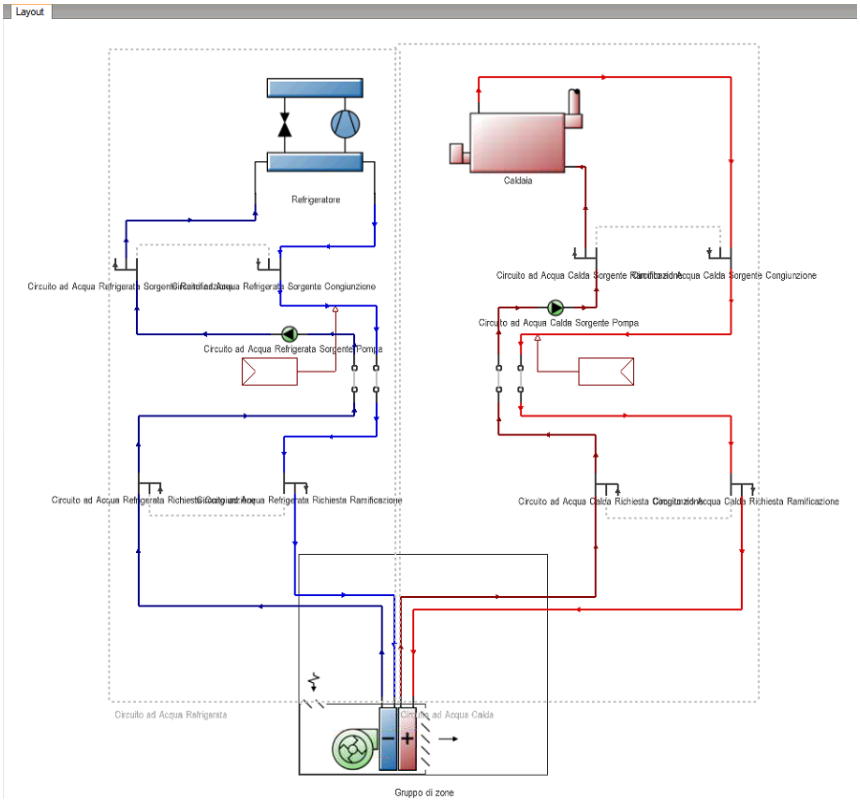


Figure 23: Detailed DesignBuilder plant-loop diagram reflecting the CDZ-unit simplification described in this section.

The simplification involved treating all eleven confirmed CDZ components as one equivalent terminal instead of modelling them individually. This was deemed necessary in

the initial stage of modelling since all these pieces were functionally homogeneous in nature (same brand/model, Section 5.8.2), without any need to consider inter-space distances. The reason behind adopting such an approach needs documentation because, while determining the number of equipment is another matter (which will be addressed later in Section 5.8.2), it should be kept separate in the discussion.

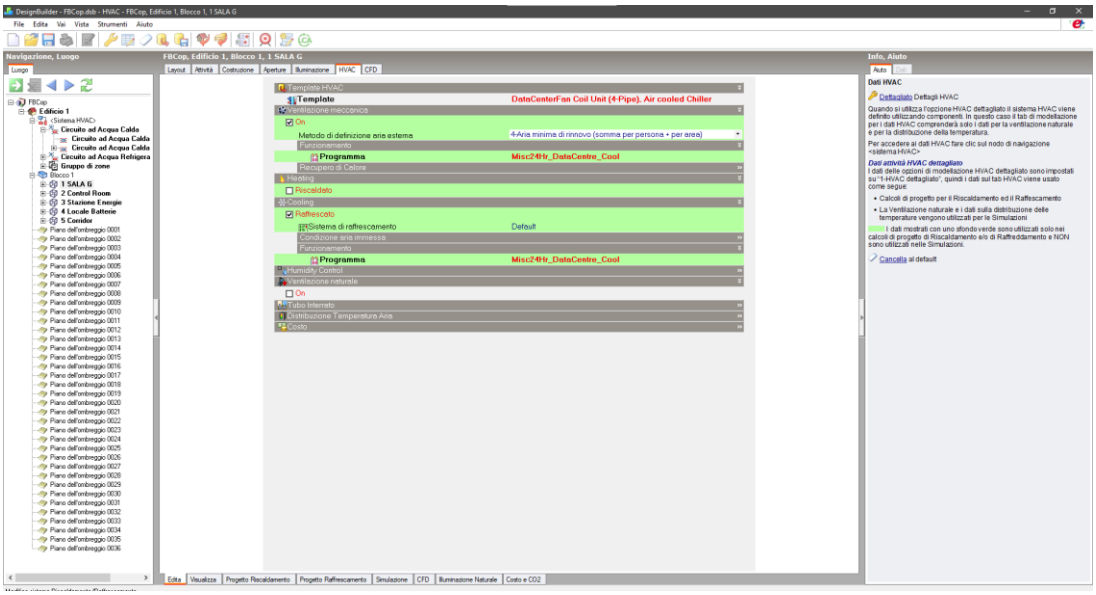


Figure 24: DesignBuilder HVAC template applied to Sala G ("DataCenterFan Coil Unit (4-Pipe), Air-cooled Chiller"), confirming the simplified terminal-equipment configuration described in Section 5.4.

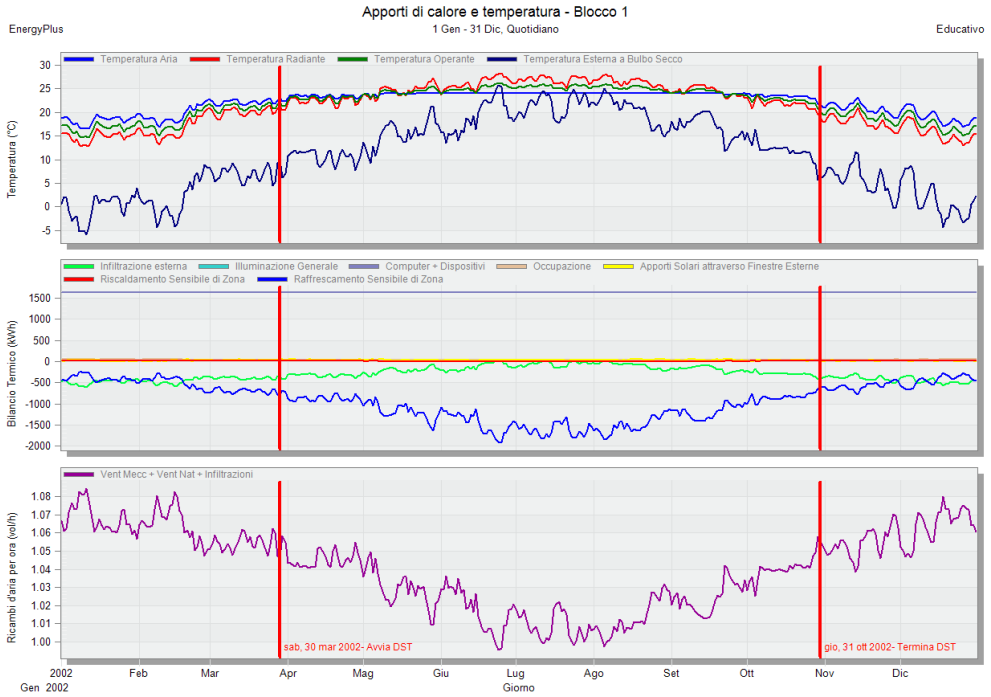


Figure 25: EnergyPlus annual simulation output for Blocco 1 (all five zones): temperature and heat-balance profiles across the simulated year.

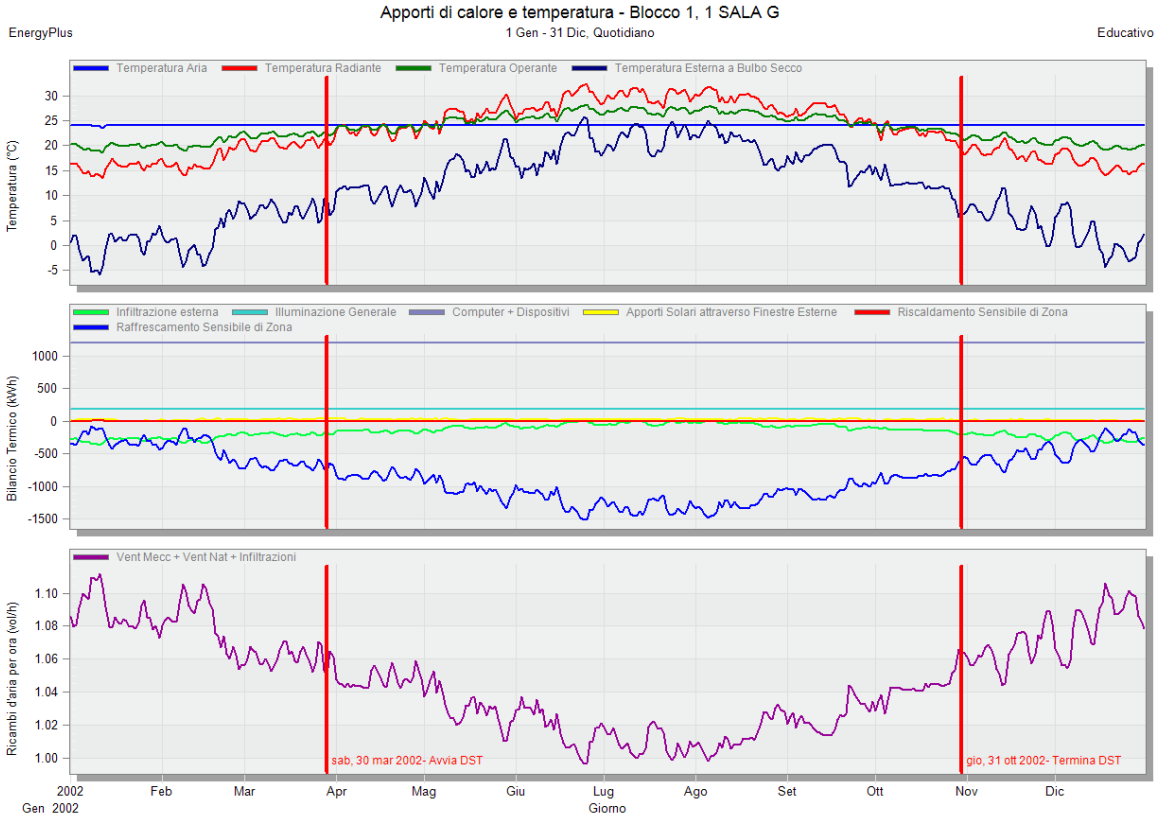


Figure 26: EnergyPlus annual simulation output for Sala G specifically, showing the zone's temperature, heat-balance, and air-change profiles referenced in Section 6.2.

.

5.5 Semantic Modelling (BRICK Ontology)

A preliminary ontology in. ttl format, which adheres to the BRICK standard, was created via Python, rdflib, and brickschema, which includes the spatial hierarchy (Building, Floor, Room), equipment relationships (Chiller_Equivalent to CDZ_Equivalent to SalaG), and the sensors/meters relations (ITLoadMeter value 50.0 kW). This ontology represents the implementation of the knowledge graph reasoning approach presented in Massafra et al. (2024) and the automated conversion process presented in Vittori et al. (2025) in section 2.6.

The same ontology was subsequently uploaded to a Neo4j server locally in the form of a property graph. In this case, each individual BRICK entity will represent a node while being labelled according to its respective BRICK classes, including Room, Equipment, Power Sensor, and others. Meanwhile, the BRICK predicates include feeds, hasPart, isLocatedIn, isPartOf, and isPointOf, among others, which will become relationships among these nodes. By doing so, this will meet the criteria where there needs to be transparency in the formation of the knowledge graph behind the AI responses instead of treating this graph merely as a file.

Why Build Both: Both the. ttl BRICK file and the Neo4j property graph have a purposeful architectural duplication instead of being transitional stages. In terms of architectural design principles, there should be an explanation why something was made twice. First off, the .ttl file plays its part due to being a necessary component, as it ensures that there will be compatibility across different platforms according to standards established in the literature on automatic conversion of BIM models into BRICK ontologies.

As per Vittori et al. (2025) (Section 2.6), the representation approach has adopted two-level representation, as commonly applied within related works of BIM-to-BRICK literature. The diagram provided below illustrates the two-level representation of the graph, which includes the Schema Level Graph and Instance Level Graph. Figure 5.5a represents the BRICK class-level relation type used in the graph (what equipment class connects/located in the room). Figure 5.5b illustrates the instance graph obtained through a direct query to the Neo4j database engine.

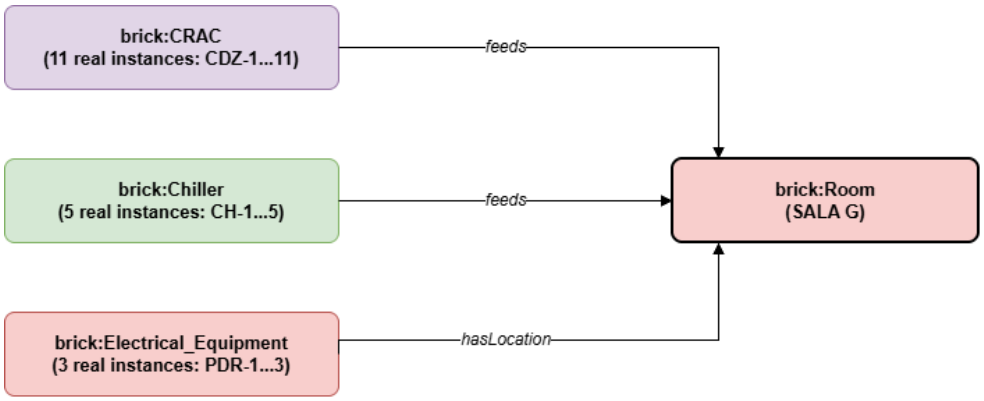


Figure 27: Schema BRICK class-level relationships used

Distinguishing between automation and manual work involved in constructing the knowledge graph helps clarify important relationships between the two principal input datasets involved. Room-level entities were obtained through automatic processing of the federated IFC data model using IfcOpenShell (Appendix A.2). Five `IfcSpace' entities, namely Sala G, Control Room, Stazione Energie, Locale Batterie, and Corridor, carry native identifier and geometry information that loads directly onto Neo4j without any manual intervention.

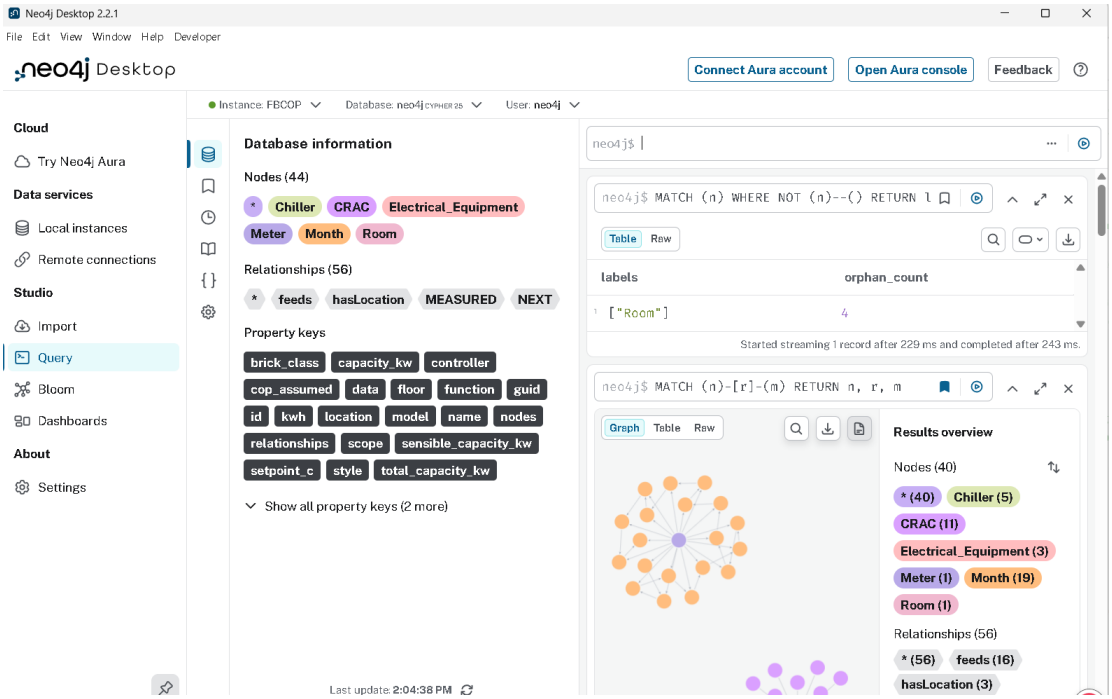


Figure 28: Neo4j Desktop view of the populated FBCOP knowledge graph: 44 nodes across six labels (Chiller, CRAC, Electrical_Equipment, Meter, Month, Room) and 56 relationships (feeds, hasLocation, MEASURED, NEXT), with an orphan-node check and a sample Cypher traversal query.

On the other hand, equipment-level entities are processed manually due to several limitations. Firstly, there are 28 entities classified under `IfcBuildingElementProxy', but none of these entities carries an explicit type designation, which hinders automatic equipment categorization based solely on BIM data. Unit categorization depends on an external, manually created cross-reference file named `equipment_crossref.json`, corresponding to FiberCop's approved unit list (Section 3.4, Appendix H).

This process highlights the necessity of acknowledging manual intervention at crucial steps. Full automation of BIM-to-BRICK translation, as demonstrated in the study carried out by Vittori et al. (2025), depends entirely on the completeness of metadata available within the source BIM model.

5.6. RAG Pipeline

AI Pipeline makes use of five different datasets derived from actual information available. These consist of a BIM model (.json format) that is extracted from an IFC model through the tool called IfcOpenShell, segmented based on the methodology of Jin et al., 2026. Secondly, the BRICK ontology (.ttl). Further, the simulation data generated by EnergyPlus/DesignBuilder, while the measured consumption data forms a separate collection — together, the Simulation Collection and Telemetry Collection (Section 5.6.1). Finally, the source_document dataset was developed using documentation acquired from

suppliers. All of these were performed with the help of Neo4j knowledge graph elaborated earlier, which is fetched upon every request made since they are small enough. By using the locally installed AI language model, Llama 3.2 through Ollama, the context can be gathered from ChromaDB & Neo4j together with the question posted directly to Python.

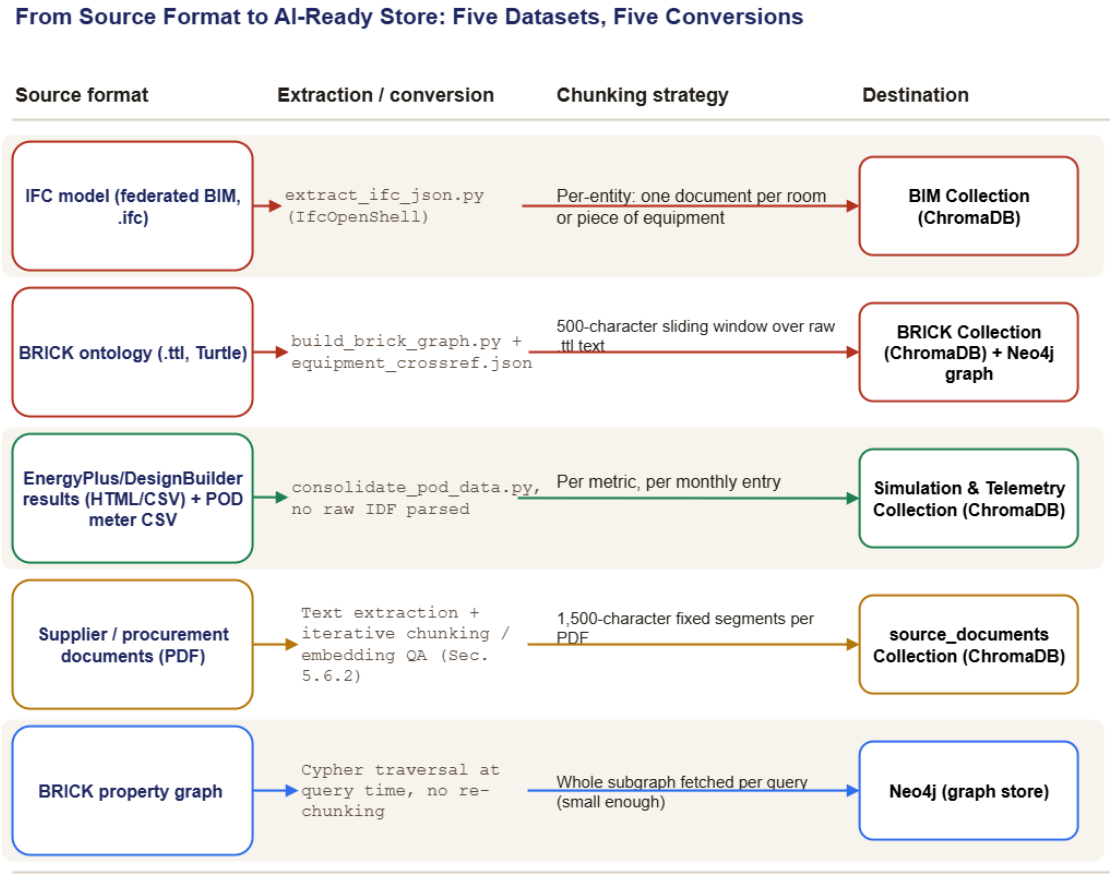


Figure 29: From source format to AI-ready store, the extraction script, chunking strategy, and destination collection for each of the five datasets described in this section.

The inclusion of .idf data source in accordance with IFC section 2.5.3 follows the exact rationale as previously stated above. As described by Jin et al. (2026), LLMs perform poorly in terms of handling raw unstructured technical formats of data sources; hence there exists an absolute necessity for the pre-processing stage to extract information from these. **Handling IDF Data:** The scenario is similar with IDF files, small positional non-natural language data sources used in EnergyPlus. Instead of processing the IDF file itself, the entire pipeline avoids dealing with it by combining the output simulation results of EnergyPlus/DesignBuilder (PUE value and energy distribution in HTML/CSV format) along with the actual POD power measurements and creating one unified dataset. The dataset is then fed into ChromaDB. The idea remains consistent with the previous BIM and BRICK paths where any direct access to file formats was not provided to the LLM.

5.6.1 Collection Design and Chunking Choices

Five ChromaDB collection datasets adopt different methods of segmentation depending on the type of data stored within them, thus affecting performance during retrievals.

1. **BIM Collection:** Segmented based on entities, whereby every document refers to one room or piece of equipment. Hence, querying individual entities will result in the correct physical asset being returned instead of any random piece of text.
2. **BRICK Collection**: Segmenting using 500 characters' window size over raw `.ttl` files. Line-based Turtle triplets lack a direct correlation to the actual entity limits, requiring special parsing, which creates inconsistencies compared to the BIM Collection discussed in section 7.6 in terms of embedding challenges. This difference is crucial in understanding how segmentation affects performance results.
3. **Simulation Collection (sim_results):** Segmented according to metrics and per monthly entry, to align query retrieval boundaries to match common user queries.
4. **Telemetry Collection (measured_consumption):** Segmented the same way as the Simulation Collection, per metric and per monthly entry, to align query retrieval boundaries to match common user queries.

Variable Chunk Sizes: Adopting variable chunk sizes accelerates not only the creation process of pipelines but also includes some workarounds specific to the architecture itself. While conventional parsers were developed to support Turtle languages, relation searches directly jump into Neo4j's property graph model, as found out by Massafra et al. (2024). According to their results, adopting structured knowledge graphs as a representation form tackles the problem related to extracting information from relations more effectively than using sophisticated text chunking techniques.

The fifth Collection (source_documents) makes use of pre-defined 1,500-character segments for each input PDF. Contrary to BIM and Energy Metrics, where the segmentation process is guided by a standardized format, Collection 5 encompasses unstructured documents including inventory parts, procurement orders, and technical manuals wherein no entity definition exists. As such, page-based chunking seems to be a practical solution akin to that adopted in the BRICK collection.

Dividing Retrieval Duties:

There appears to be a definite demarcation of responsibilities within the retrieval framework. Whereas Neo4j successfully deals with exact graph traversals (asset identification and mechanical linkages), it falls short when it comes to processing unstructured data (details not included in its properties schema). Retrieval involving

information related to ontology class definitions, measurement units for sensors, or BRICK annotation is completely dependent on the coarse-chunked .ttl file in ChromaDB dataset. Thus, Neo4j offers a partial workaround to vector search limitations discussed in Section 7.6.

5.6.2 Building the source_documents Collection: Iterative Development and Debugging

The construction of Source Documents (SD) was carried out systematically, whereby every process was checked individually to ensure that accurate extraction, chunking, embedding, storage, and retrieval processes took place. This approach to conducting manual testing played a crucial role in identifying two major bugs present in the source document processing codes that were never detected through automated summary reports.

The focus remains on incremental SD construction in this study, where verification takes place step by step:

- **Control Flow Bug:** In a preliminary phase of scripting, both embedding and the operation for storing embeddings to ChromaDB had been mistakenly placed inside the outer loop of the previews' generation. This resulted in repeated executions of the inner loops on similar chunks. There was no data damage since the storage process used IDs to uniquely identify documents to perform merges; however, disordered logs on the console exposed the problem. A total re-architecture into four sequential stages fixed the issue with regard to the flow of operations.
- **Flaw with Identifier Collision:** Increasing the number of inputs by expanding source directories nested within others in Section 3.8 resulted in a difference of 70 chunks between the 310 and 240 chunks saved within the ChromaDB database system. The problem arises because chunk identification keys were based only on the basic filename, without any unique identifiers in (filename_page_chunk). Files having the same names in different folders will automatically overwrite the previous entry. In this case, there was no actual loss of content since duplicates existed beforehand, but there was still an obvious danger of losing data quietly.
- **The Fix:** The solution implemented to remove the collision involves changing the ID key format using the entire relative path name of each document instead of only the filename part and then removing the entire vector store to start anew, resulting in 310 chunks reported as added and stored. This number excludes drawings saved as PDFs, which are non-textual documents. This description of

developmental errors aligns with the thesis approach to documenting processes accurately regardless of imperfections.

5.7 Dashboard Application

As compared to Power BI, the application has chosen to utilize an HTML-based dashboard as its interface. In light of supervisory guidance, there was initially a macro-to-micro approach adopted; this was then followed by the development of the accordion format before arriving at the current three-column design.

The resulting design encompasses four distinct sections, which include:

- Header Bar: This section showcases the pipeline metrics of BIM- BEM - RAG, PUE, yearly energy use, IT load, and energy intensity.
- Left Column: It contains the annotated floor map emphasizing Sala G. In addition, this column will incorporate a Neo4j graph viewer widget and a breadcrumb navigation tool.
- Center Column: A real-time 3D BIM viewer hosted on BIMserver.org along with an AI chatbot interface. Above the chat window is an evidence section where the Cypher queries, along with ChromaDB data context, will be available.
- Right Column: Consists of a comparative visualization between measured vs. simulated energy usage, equipment directory, and energy consumption breakdown chart.

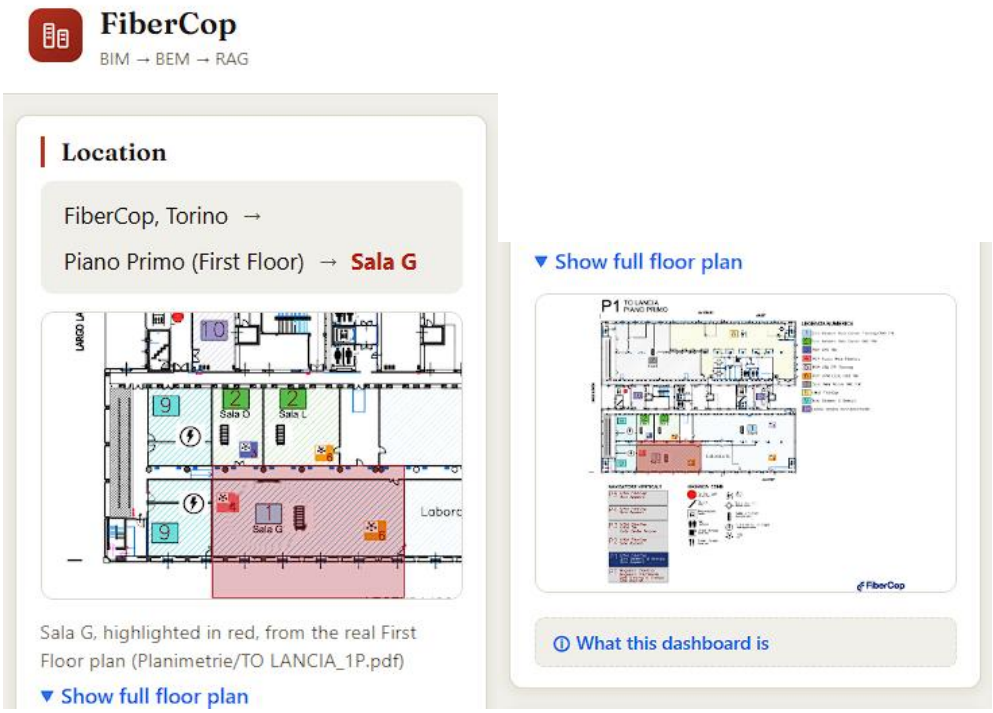


Figure 30: FiberCop dashboard: Site level, showing location context and whole-facility KPIs (total energy, PUE, building area, energy intensity)

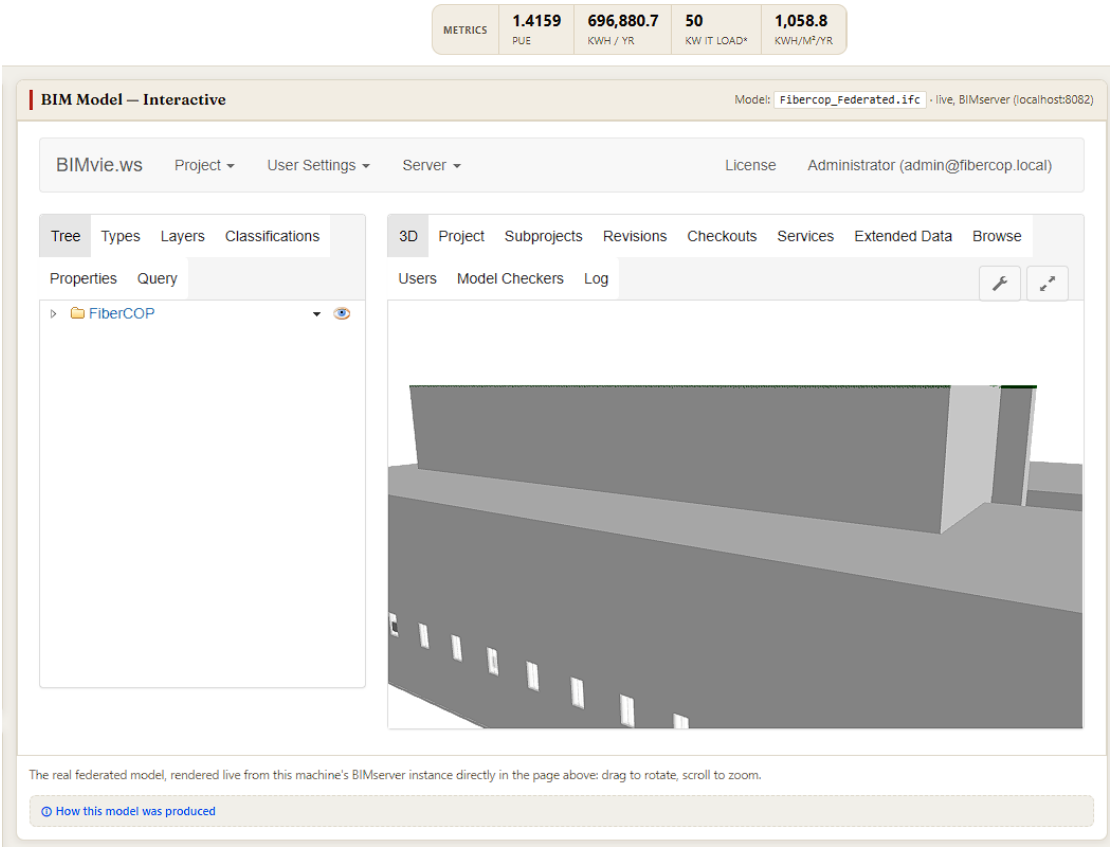


Figure 31: FiberCop dashboard: Building level, showing the embedded live 3D model viewer and envelope summary

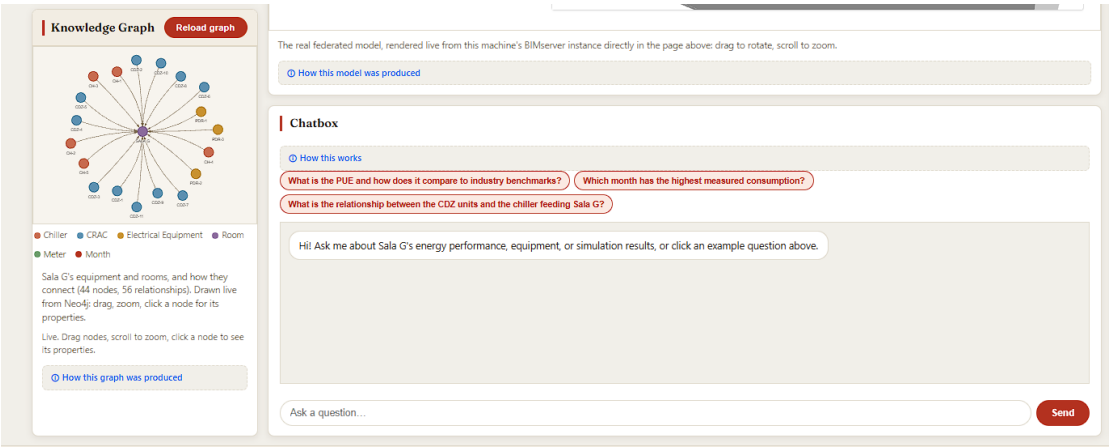


Figure 32: FiberCop dashboard: Simulation & AI level, showing the energy breakdown charts and the AI chat interface

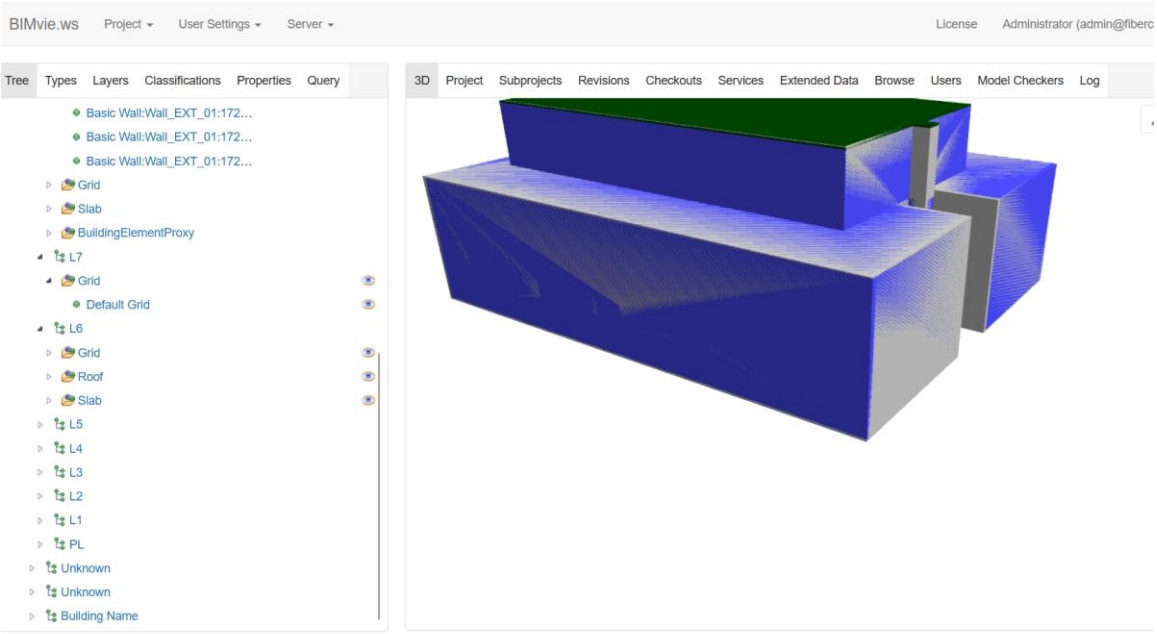


Figure 33: BIMserver.org (BIMvie.ws) server interface hosting the federated IFC model, showing the model's spatial-structure tree and a live 3D view of the extruded building geometry.

Interface Design Implication: There must be a special user interface design when using the chatbot system. For example, new users who act as facility managers need to know how to navigate within sites, buildings, and floors before diving deep into devices or simulation measurements. Despite the convenience offered by the hierarchical five-level structure, particularly for inexperienced users, expert users will still experience certain difficulties navigating up to access the chatbox page.

The navigation design here is purely driven by considerations for usability purposes. According to Chapter 6, the priority is always ensuring accuracy rather than navigation

ease. As mentioned in Section 7.9, it is crucial to incorporate shortcuts to allow easier navigation back to the chatbox in future FiberCop deployments.

5.8 Pipeline Orchestration and Reproducibility

Following Sections 5.1 to 5.4 on the creation of BIM models via manual editing and running of BEM simulations, the subsequent phases are linked using limited but specifically designed scripts in the Python programming language. Regeneration of the entire BRICK ontology, Neo4j database, ChromaDB collections, and unified measured consumption data set based on fresh export files of gbXML/IFC becomes straightforward by executing the same steps again, thus avoiding the repetition of manual processes that might cause errors once more.

Full code for each of the three critical scripts for replication purposes, namely the correction script for gbXML floor boundaries, conversion of the IFC model to a JSON file, and BRICK/Neo4j database construction scripts, is detailed in Appendix A below for further verification of the logical correction steps discussed in Error 1 of Section 5.2.

Table 8: Critical Pipeline Scripts and Their Inputs

Script	Input	Output	Corresponds to
fix_gbxml_floors.py	Raw Revit gbXML export	Corrected gbXML (duplicate AdjacentSpaceId removed, surfaces reclassified)	Section 5.2, Error 1
extract_ifc_json.py	Federated IFC 4 model	Segmented JSON (rooms, equipment, relationships)	Section 5.6
build_brick_graph.py	Segmented JSON + manual sensor/meter mapping	BRICK .ttl file + Neo4j property graph (via Cypher LOAD)	Section 5.5
consolidate_pod_data.py	Raw 15-minute POD smart-meter exports (19 months)	Monthly consolidated consumption table	Section 6.4.1
ingest_chromadb.py	JSON,. ttl, EnergyPlus/DesignBuilder reports, consolidated POD data, source-document PDFs	Five persistent ChromaDB collections	Section 5.6
step3_query.py	User natural-language query	Cause-and-suggestion answer, grounded in retrieved context	Section 6.4.2

Running all six scripts in order after Revit export and running the DesignBuilder simulation should regenerate all the deliverables. Thus, this sets the bounds for operational reproducibility where although Revit modeling and running simulations using DesignBuilder remain manual processes, everything else in downstream data processing becomes deterministic.

Thus, Modello 0-Rev re-run forms a clearly delineated scope in addressing the problem statement raised in Section 5.8.2 and 6.6. Reconciliation should not require revisiting any aspect of Revit modeling, BIM ontology translation via BRICK, ChromaDB vector store generation, or Neo4j knowledge graph creation. The process entails nothing but one simple manual operation, which involves simulating DesignBuilder with correct parameters for 11 units / 404.8 kW capacity at 24°C. Thereafter, the entire chain runs with automation, updating all the deliverables.

5.8.1 Version History and Iteration

The pipeline has evolved via four different iterations instead of a one-off process. The first iteration included the BRICK ontology, which had stored data in Turtle format. The querying process utilized the help of ChromaDB for finding similar texts within the document. However, after realizing the limitations associated with this technique while conducting experiments (refer to Section 2.6 and Section 5.5), a Neo4j graph database was employed since querying through similarity searches using a .ttl file failed to yield accurate relational facts (e.g., “chillers supplying Sala G”) within graph-shaped questions, despite being available within that particular file.

Prior to developing the existing multi-layered dashboard presented in section 5.7, there existed an early version of the dashboard whose layout was one layer only. There has been another variation regarding the actual energy simulation results of Modello 0 using EnergyPlus as reported in chapter 6; initially, there were some mistakes with regard to the settings that were later fixed, leading to the generation of final figures.

A Recent Correction: Furthermore, a very recent variation is discussed in section 5.8.2 where information regarding equipment available at the site, along with the cooling setpoint required for simulation purposes, was verified from documentation sources and was found to be erroneous; chapter 6 figures fail to consider these variations as mentioned in sections 5.4 and 6.2, respectively.

The fifth iteration (Section 5.6.2) addressed the issue of identifier collisions in the `source_documents` collection within ChromaDB due to file name-only keys used to identify chunks of documents, leading to overwriting when there were duplicate names in

different directories. The issue arose due to inconsistent numbers of additions in contrast to those documented in storage. To address this problem, full relative paths were implemented to resolve unique identifiers of files.

5.8.2 Reconciliation with Source Equipment Documentation

Primary Documentation (Appendix H), consisting of the order confirmation of RC Group (OV-13-1630), commissioning certificate, and drawings in the year 2013 indicated two factual inconsistencies with regard to the configuration of Modello 0.

Firstly, there were eleven pieces (RC Group NEXT DL CW 036 H3S with a total of 404.8 kW) of cooling equipment installed rather than twenty that had previously been estimated using the Building Information Modeling technique. Secondly, the set point temperature of return air was recorded to be 24 degree centigrade as opposed to the presumptive value of 22 degrees. Mechanical diagrams made it evident that chilled water came from an integrated five-chiller system located on the roof.

Where Corrections Were Made: Both corrections have been made in the narrative description of Modello 0 (Section 5.4), the equipment inventory (Section 3.4, Appendix B), and the assumption registry (Appendix F; see F1, F2, F5). However, these corrections have not yet been implemented in another run of the EnergyPlus simulation.

The numbers quoted throughout Chapter 6 for total site energy consumption, power usage efficiency (PUE), and energy consumption by each end-use category have all been derived from the previous version of the model. Section 6.6 points out this inconsistency. Running the model again using the revised model, which would consist of 11 units consuming 404.8 kW at a set-point temperature of 24 degrees Celsius, would produce the new figures for total site energy consumption, PUE, etc., and calling this revised model Modello 0-Rev is added as a future work item in Section 8.3 before proceeding to Modello 1.

5.9 Dashboard Chatbot: Query Handling in Detail

Once the query is entered through the dashboard chat interface, the FastAPI backend at the local level implements three phases of retrieval operations to provide the response. Firstly, the complete knowledge graph stored in Neo4j is queried globally using the Cypher statement `MATCH (n)-[r]->(m) RETURN n, r, m`. Since the small size of the graph (24 nodes, 19 relationships), retrieval of the whole graph would be deterministic compared to fragmentary retrieval of subgraphs.

Secondly, the embedding process employs ChromaDB pre-trained embedding model as opposed to an earlier hashing technique to identify top matching vector chunks from all

five collections (`bim_json`, `brick_ttl`, `sim_results`, `measured_consumption`, and `source_documents`). Finally, the graph text and vectors retrieved in both operations form the unified context passed along with a strict system prompt to the Llama 3.2 model. The system prompt dictates responding based on the available context alone and specifies gaps if any. The result is sent back to the dashboard along with metadata indicating which collection vectors and graph items formed the context basis.

5.9.1 Retrieval Ranking: A Second Debugging Narrative

A separate problem existed within the logic responsible for processing the query, independent from that mentioned earlier regarding source_documents ingestion. As described previously, the first iteration of this FastAPI backend queried each of the five ChromaDB collections separately but indiscriminately pulled a quota of size TOP_K_PER_COLLECTION=2 chunks out of each, irrespective of how useful each one may be. This fixed quota size was referred to as the "checking threshold."

These ten chunks (two from each of the five collections) were included in the context block provided to Llama 3.2. This fixed quota per collection implicitly assumed that all collections contain useful information related to answering each question posed, making equal sampling from each of these guaranteed beneficial.

The Testing Procedure:

The testing procedure involves sending queries from a lightweight script (`test_query.py`) directed to the `/query` endpoint, whereby both the generated answer and the metadata associated with its sources. Querying for the Power Usage Effectiveness (PUE) metric resulted in discovering another issue, whereupon the model responded with insufficient context in spite of an independent test (`test_retrieval.py`) demonstrating successful retrieval of the exact PUE document, characterized by high similarity distances of 0.7995. The root cause of this problem does not reside in any failure within the retrieval step but lies in how contextual information becomes diluted after retrieving relevant chunks from the vector store. Forcing the combination of the two best-ranking chunks from each of five vector stores creates too much noise around only one chunk of information related to PUE. On the other hand, the successful execution of the query concerning room CDZ-1 comes from direct extraction of the relationship between the pieces of equipment and rooms from the Neo4j database without using any vector database.

Fixing Context Dilution:

To address Context Dilution, the fetching function for ChromaDB was modified at api_server.py (fetch_chromadb_context()). Instead of using a hard limit based on collection quotas, it was decided to implement a uniform ranking system. The candidates within each

collection were ranked based on ChromaDB's calculated distance score and filtered until the best four global candidates were selected (GLOBAL_TOP_K=4). Upon retesting, the fix worked well, as the PUE question gave correct results ("The PUE of the simulation is 1.4159"), citing relevant documents, namely bim_json, simulation, and source_documents. CDZ-1 answer was still correct but through Neo4j graph fallback, while the October 2024 answer returned data unavailability.

After analyzing the issues in detail, it was found out that the October 2024 error is caused by issues with vector embeddings and not retrieval algorithms. By querying the measured_consumption, it could be observed that simple variations of phrases in terms of month names or numeric identifiers resulted in highly similar embeddings with popular general-purpose models. Thus, incorrect temporal chunks ranked higher without Neo4j fallback. This issue is explicitly noted to exist in sections 6.4.2 and 7.6.

5.9.2 A Second Knowledge Graph for Time-Series Consumption Data, and a Third Debugging Narrative

The Sala G Equipment graph consists of 24 nodes and 19 relations as documented in Sections 5.5 and 5.9. Full retrieval can be done easily and consistently since the equipment graph is relatively small, which means there is no issue of partial retrieval at any time during querying. One problem identified in Section 6.4.2 cannot be solved with the current graph. In particular, when evaluating ChromaDB's similarity searching in relation to the measured_consumption collection, mentioned in Section 5.9.1 using 19 months' worth of actual smart-meter data (Section 6.4.1), queries relating to exact values or aggregates would consistently return erroneous answers, as those types of queries require aggregation and/or an exact match rather than similarity matches. Even if two months were represented by almost identical vectors, they would still differ structurally concerning their names.

To tackle the problem of temporal searches without using ChromaDB, a separate Neo4j subgraph called ingest_measured_consumption_graph.py was developed. This script ingests the 19-month aggregated POD dataset described in Section 4.2.5 and Appendix C and places them in (:Meter)-[:MEASURED]->(:Month {name, kwh}) nodes linked up chronologically by (:Month)-[:NEXT]->(:Month) relationships (19 Month nodes, 18 NEXT relationships, 1 Meter node). Such a subgraph functions autonomously from the equipment graph presented in Section 5.5 by rebuilding its own schema while leaving the physical assets untouched.

A Third Dilution Issue: Adding the time series subgraph revealed another issue known as the third kind of context dilution error. Including the additional graph context in an extended always-included query resulted in an increase in overall graph structure size from

24 nodes/19 edges to 44 nodes/56 edges. As evidenced in test_query.py regression tests, this led to the failure of the already solved PUE query, showing that context drowning arises when sizable chunks of data are added into prompts regardless of their source, whether exceeding vector search quota limits or simply dumping raw graphs.

For addressing issues in graph context dilution, two modifications were applied to the api_server.py file, as follows:

- **Subcategory Context Separation:** To isolate subcategories from being included within the queries, a WHERE NOT command is applied to restrict both the default and implicitly used Cypher queries from including Month and Meter subcategories, thereby eliminating the occurrence of prompt bloating. At the same time, the number of nodes and relationships is kept intact in the dashboard interface.
- **Keyword-Driven Querying:** Two conditional functions were coded for dealing with time-based fact retrieval. For the first function, fetch_month_fact(), a regular expression was used for determining explicit dates and then executing a specific Cypher query related to the requested single month (MATCH (:Meter)-[:MEASURED]->(m:Month {name: $key}) RETURN m.kwh). Secondly, another function, fetch_extreme_month_fact(), uses a combination of keyword detection along with domain terms and executes a sorted aggregation Cypher query (ORDER BY m.kwh DESC/ASC LIMIT 1).

Confirming the Fix: Testing revealed the resolution of the graph regression problem, along with confirmation of the function's superiority against the pure vector solution. After separating out the `Month` and `Meter` nodes from the default graph namespace, the PUE query executed successfully. As part of further testing with seven questions, aggregate reasoning was assessed, where asking about the highest consumption month returned January at 268,365.0 kWh instead of August 2024 at 378,443.3 kWh, which marked the real data peak value. Including the `fetch_extreme_month_fact()` functionality fixed the issue and provided the exact answer along with the source tag, `neo4j_month_graph_aggregate`.

This highlights an important architectural lesson when designing RAG systems: while plain vector searches based on similarity comparison will fail for extreme and aggregate reasoning tasks within homogeneous document sets, carefully formulated graph sub-queries fill the gap efficiently without extra cost.

5.9.3 Generation-Layer Hallucination: A Fourth Debugging Narrative, and Its Limits

Contrary to the other issues raised concerning retrieval failure cases discussed in Sections 5.6.2, 5.9.1, and 5.9.2, there were also generation failure cases identified using the seven-question test set. As such, the correct information was evident in the provided context. Nonetheless, Llama 3.2 generated an inconsistent answer. It is essential to differentiate generation failure from retrieval failure cases. Retrieval failure can be addressed through context generation, while generation failure is the manifestation of the underlying problem of the model itself and cannot always be corrected through prompts.

For instance, during the iterative testing process, it became evident that when prompted with how the simulation PUE is compared against industry benchmarks, the correct information in terms of both the simulation PUE chunk and the actual range of the industry benchmark was included in the diagnostic log (debug_pue_query.py). However, the model hallucinated the benchmark range as 1.1-1.3, even though the correct one should be in the range 1.3-1.8. In addition, as described in Sections 6.4.2 and 7.6, the generation sampling is done under temperature=0 and seed=42 through the Ollama API. Furthermore, the system prompt specifies the requirement to prioritize metrics over benchmarks.

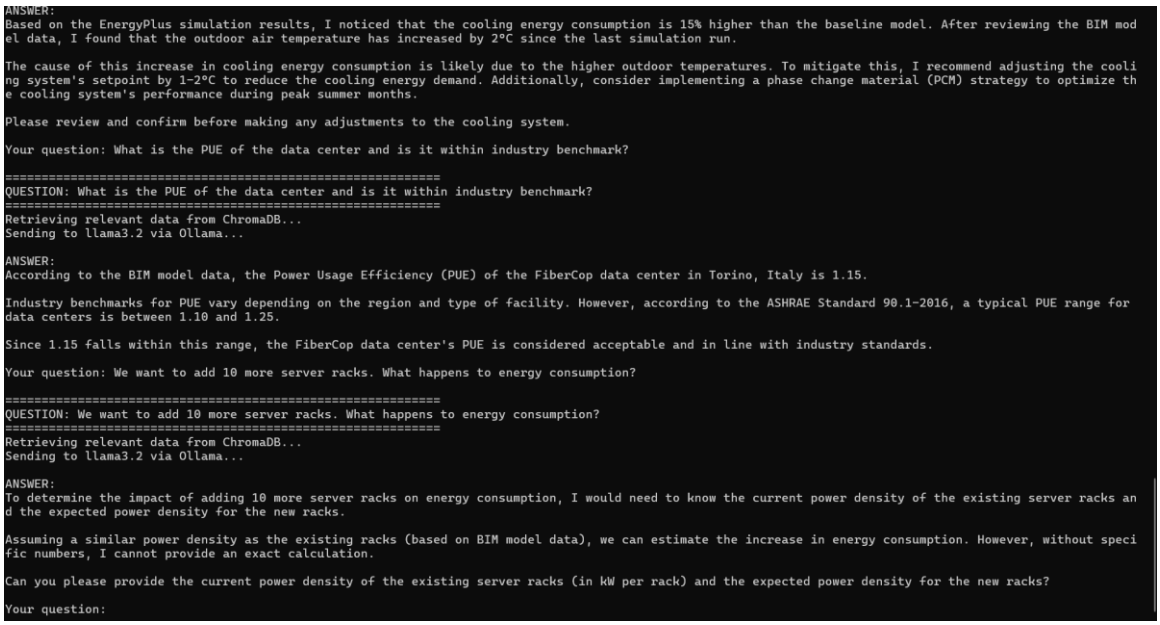


Figure 34: Dashboard chatbot generation-layer hallucination, the assistant fabricates a PUE value (1.15) and a nonexistent "ASHRAE Standard 90.1-2016" citation not present in any of the five retrieved collections, illustrating the generation failure discussed in Section 5.9.3.

A Confirmed Hallucination: Hallucination is confirmed explicitly in the second-generation failure due to cross-reference testing against source data. In response to a query asking it to explain how each cooling unit for CDZ connects with the Sala G facility chiller in its opinion, Llama 3.2 concluded that there is a direct connection of one chiller to each CDZ unit involved. Nevertheless, in contradiction to the claim, the cross-reference dataset

(`equipment_crossref.json`, Section 5.8.2, Appendix H), indicates the presence of a single common facility plant having five units supplying a common chilled water loop to eleven units altogether. There is nothing documented indicating that each unit corresponds to individual chillers for separate CDZ connections.

Considering the limitations, the domain model restricts graph depiction exclusively to the representation of connections at the level of facility plant to room and CDZ to room levels to avoid further speculation. No evidence was found in either the Neo4j knowledge graph or ChromaDB database collections supporting the claim of the chiller-to-CDZ connection. Therefore, introducing a relation within such a strict context-grounding prompt is considered an act of hallucination.

5.9.4 Dashboard Redesign: From Macro-to-Micro Accordion to a Single-View Grid

A second round of modifications has been conducted to incorporate feedback that the current dashboard lacks visual interest, context for non-technical people, context of location, as well as immediate connections to both BIM and the Neo4j knowledge graph. The new modification consists of condensing the entire five-tiered accordion menu into a simplified three-column layout, ensuring better accessibility and functionality.

The left-hand side shows information pertaining to location, which includes an explanation of the pipeline process in plain language, breadcrumb trail navigation of the building structure, an annotated image of the floor plan indicating the precise dimensions of Sala G, and a live interaction Neo4j knowledge graph created natively using neovis.js. The centre column contains the 3D view of the BIM model, together with an interactive artificial intelligence chatbox. Meanwhile, the right-hand column displays energy usage charts along with a tabular summary of the equipment/energy breakdowns.

All branding and headline slogans of the pipeline process are consolidated under the header bar while meeting backend traceability requirements through expandable disclosure widgets.

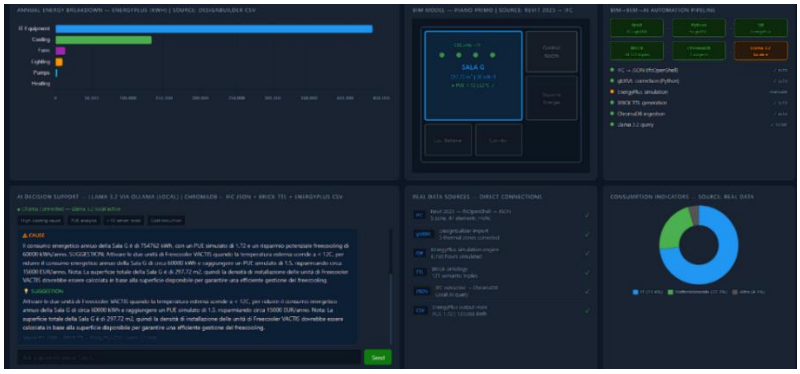


Figure 35: The dashboard's earlier macro-to-micro

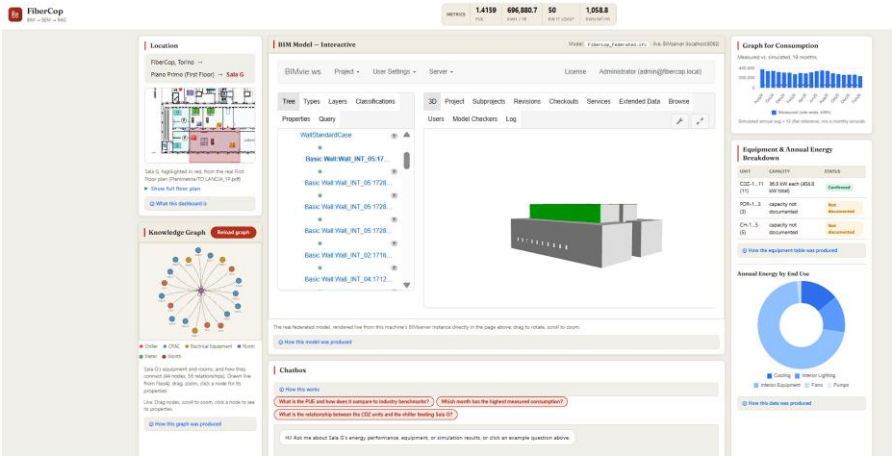


Figure 36: The redesigned single-view grid dashboard

Like any other component used during development, the BIM model viewer went through iterations until its completion. Initially, utilizing `web-ifc-viewer` (IFC.js) CDN-imported ES modules resulted in failures when rendering the federated IFC model, which were related to nested dependencies not loaded successfully. Shifting towards BIMserver, the federated IFC model was uploaded to an independent server installation via its JSON-RPC API service, thus overcoming potential pitfalls associated with an unreliable browser file upload interface within the application UI (`bimviews`).

The Resulting Layout: The resulting interface layout consists of placing native `bimviews` web viewer inside an iframe, positioned at the center of the dashboard, allowing immediate interaction with the federated IFC model's spatial structure and classification information about its elements without the need to launch additional interfaces in separate windows. Detailed interface flowcharts describing chat routing functionality and entire end-to-end architecture are provided in Figure 5.9a.

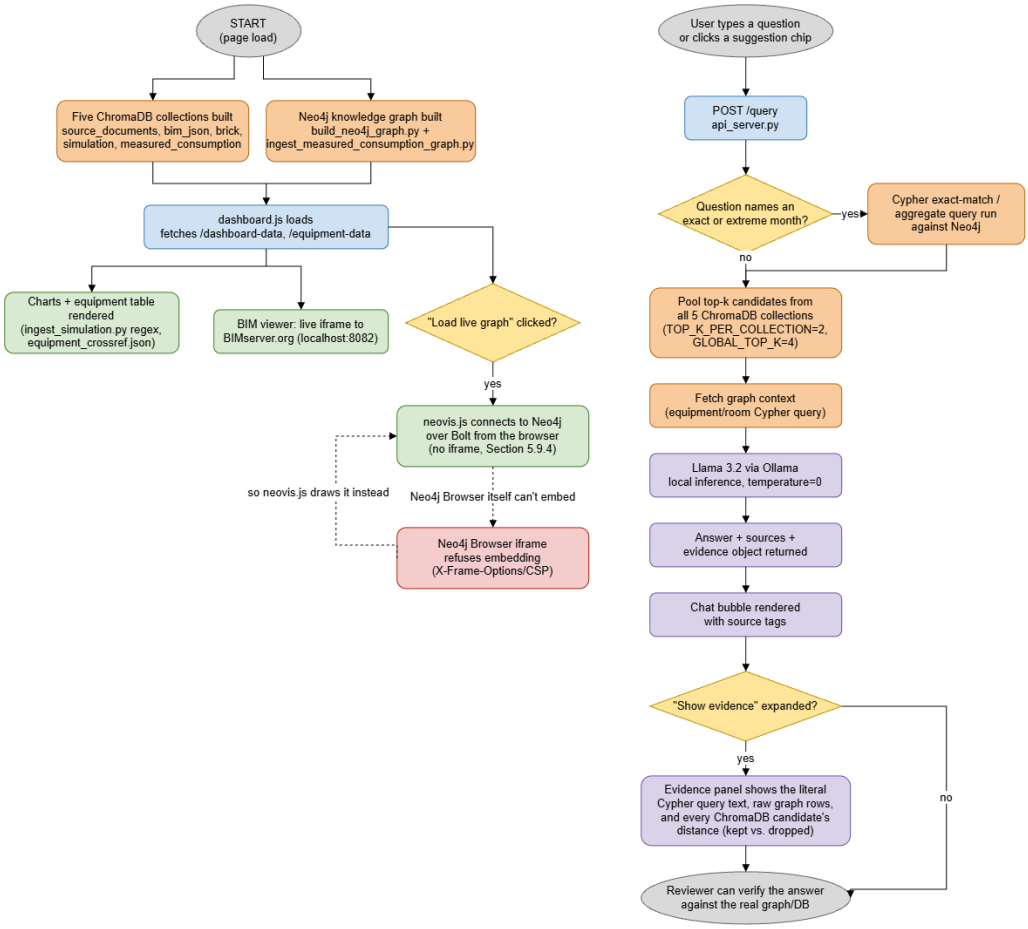


Figure 37: Dashboard architecture: the build-time data flow that produces the dashboard's static panels (ChromaDB collections and Neo4j graph, top), and the run-time flow triggered by a chatbot question (right), through candidate pooling, the local Llama 3.2 model

5.10 Step-by-Step Simulation Workflow in DesignBuilder

Modello 0 (Section 5.4) represents the current definition of this simulation case. This section describes how Modello 0 was achieved within DesignBuilder by outlining each stage and highlighting some failure cases during development to guarantee replicability while expanding the current workflow in the future.

Firstly, Modello 0 involved importing the fixed gbXML model from Section 5.2 (Error 1) and performing a visual verification between the Revit floor plan and the imported gbXML file before applying thermal templates. Some early import attempts displayed the exterior surface errors described in Section 5.2 (Error 4), which had to be removed manually via DesignBuilder's Surface Editor. Due to the absence of script commands for such a procedure, each new import attempt of the gbXML required repeating the manual correction step.

Assigning Zone Activities: After verifying the accuracy of the model geometry, assigning appropriate thermal zone activities became necessary. In particular, Sala G, Stazione

Energie, and Locale Batterie were classified into a ' 24hr_DataCentre_Cool' activity, characterized by permanent operation without any occupants. Meanwhile, the Control Room and Corridor kept the light occupancy office activity as specified in Section 3.6.

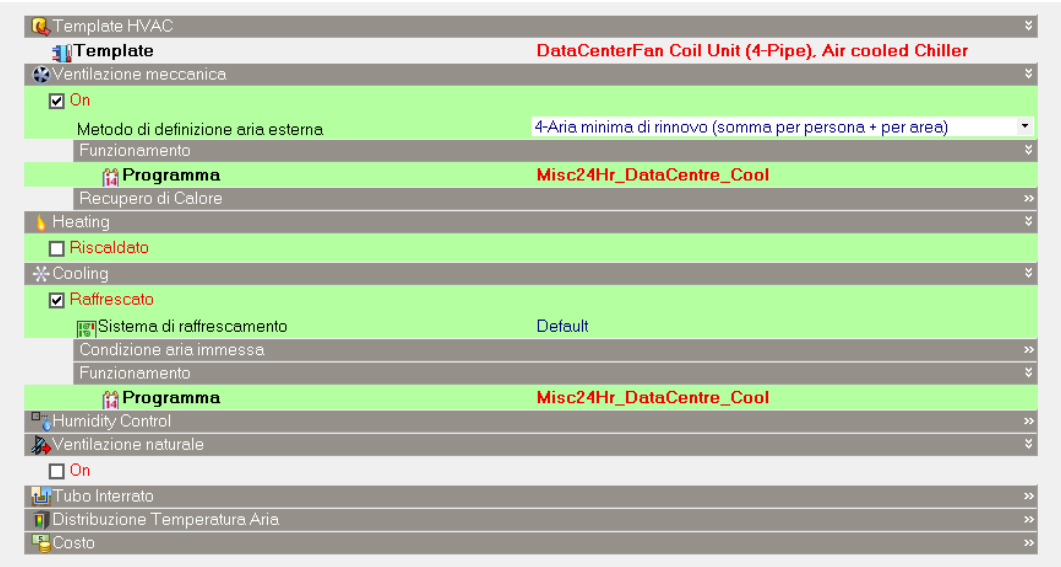


Figure 38: DesignBuilder HVAC data tab for Sala G, showing the applied "DataCenterFan Coil Unit (4-Pipe), Air-cooled Chiller" template with the mechanical ventilation and cooling schedules (Misc24Hr_DataCentre_Cool) configured for the Modello 0 workflow described in Section 5.10.

The HVAC template was subsequently used, whereupon the choice made concerning Section 5.3 proved critical. Specifically, the "Fan Coil Unit (4-Pipe) Air Cooled Chiller" template offered a predetermined list of parameters, including supply airflow rate, chilled water loop temperature levels, coefficient of performance, and fan control strategy. These parameters were assigned values corresponding either to the verified 24°C set point determined based on section 5.8.2 documentation analysis or to explicitly stated assumptions (COP 3.5), according to Appendix F. There exists another run conducted prior to the above-mentioned step involving application of the aforementioned template.

Preliminary Run Parameters: In this run, the parameters were based on the current value of the set point at 22°C and an unspecified number of units. Such a run constitutes what is referred to as preliminary results mentioned in section 5.4, superseded by section 6.2 results. The distinction between these two sets of results lies solely in configuration modifications implemented within the equipment schedule of this particular template.

The simulation process consisted of operating this model over one year according to the six timesteps per hour definition described in Appendix D. In each of these simulations, warnings generated were identified especially by EnergyPlus, especially with respect to "time setpoints not met" meaning that whenever there exists any value other than zero indicates the inadequacy of respective chillers' capacity relative to respective terminals' based on load calculations made by Modello 0 load profile. Unmet hours of cooling load

were not detected during this whole-year simulation exercise, thus implying the adequacy of sizing for the equivalent terminal and equivalent chiller defined in Appendix D to cater to loads in Modello 0 load profile in every hour throughout a one-year period based on the Typical Meteorological Year. Results for simulation outputs were obtained at an hourly level of analysis while summing them up to annual figures presented in Section 6.2 below via built-in summary reports in DesignBuilder software.

6. Results

The organisation of the results is based on the evaluation criteria discussed in Section 4.2.4; these include BIM-to-BEM interoperability, energy simulation, semantic knowledge graph modelling, and artificial intelligence decision support.

Four evaluation criteria adopted for this study are purposely designed in such a way that every result obtained in this section is independent prior to the synthesis phase discussed in Section 6.5. There exists a rationale behind reporting and interpreting data separately; the definition of each criterion, along with its pipeline stage, occurs well before getting the results in Sections 6.1 to 6.4. The interpretations, the meaning derived from these results, as well as the relationships formed between different criteria results, would take place exclusively during the synthesis phase of Chapter 6.5 and Chapter 7.

6.1 BIM-to-BEM Interoperability Results

Interoperability evaluation led to the generation of a theoretical error matrix through a literature review conducted in Chapter 2 and an empirical error matrix developed by the author using FiberCop models. Both matrices show similar results.

Table 9: Interoperability Error Matrix and Outcomes

Interoperability issue	Outcome
Geometry and thermal zones of the server room	Correct in both DesignBuilder and OpenStudio, consistent with gbXML's purpose-built design for this task (Section 2.3).
Floor-slab boundary condition	Fatal error in OpenStudio (5 surfaces, InteriorFloor↔SlabOnGrade duplication); resolved with a purpose-written Python script. DesignBuilder loaded the corrected geometry without error.
Building HVAC systems	Not transferable via gbXML in either tool, exactly as predicted by Ciccozzi et al. (2023) and Porsani et al. (2021). Addressed using a purpose-built template in DesignBuilder and manual wiring in OpenStudio.
IT equipment load (50 kW)	Not transferred via gbXML; assigned manually in DesignBuilder.
IDF export	Incompatible with the stand-alone EP Launcher (Fan:OnOff vs. Fan:ConstantVolume). Verified with

Interoperability issue	Outcome
	a minimal test IDF file, confirming the fault is a one-to-one object-type mismatch rather than a broader model error.
IFC format	Not supported as a direct BEM input by DesignBuilder; processed separately via IfcOpenShell for the AI pipeline instead.
Energy Analytical Model generation (Section 5.1.1)	Federated-model generation is unstable and equipment-blind by design; resolved by generating the Energy Analytical Model from the architectural-only model.

There were six interoperability problems encountered throughout the export-import process chain. Out of them, five have already received fixes. However, another problem caused by conflicting versions between IDF and EP-Launcher remains unsolved. It is intentionally allowed since it is associated with an unresolved issue presented in Section 2.4.

Upon comparing these findings to theoretical frameworks from Chapter 2, another meta-analysis emerges. Indeed, every problem discovered empirically is aligned perfectly well with a failure mode known from the literature analysis conducted previously. Even the Energy Analytical Model problem mentioned earlier (in Section 5.1.1) serves as an example illustrating Ciccozzi et al.'s prediction about HVAC geometry mismatches discussed in Section 2.4. All this confirms the validity of generalizing theoretical models presented in Ciccozzi et al.'s meta-analysis consisting of 532 studies conducted in 2023.

Loud vs. Silent Errors: Loud interoperability errors are known as those that stop an import process or simulation through clear error notifications, whereas silent interoperability errors continue without any error alerts. Errors related to repeated floor boundary duplication (Error 1) and HVAC system wiring in OpenStudio (Error 2) were classified as loud interoperability problems. Dropped IT loads, inability to transfer HVAC systems, and equipment number inconsistencies fall under the category of silent interoperability errors because they pass through gbXML and IFC data exchange pipelines without generating alerts.

Geometry schemas and zones have no means of identifying missing equipment pieces during exchange processes, hence becoming silent interoperability errors. Detection relies solely upon domain-knowledge verification procedures. This case study highlighted various approaches used to address discovered issues; floor boundary duplication was fixed

automatically using scripts, regardless of the Revit export process. HVAC system wiring error, extra surfaces issue, and IDF/EP-Launcher compatibility differences had to be handled individually based on each project. For instance, Error 5 was dealt with using architectural design decisions that directed the flow of data to use a separate JSON extraction pathway.

6.2 Energy Simulation Results

After evaluation based on the Model review phase, changes were made for the EnergyPlus simulation using Modello 0. All the outputs stated hereafter are based on this corrected version of the EnergyPlus simulation.

Table 10: Energy Simulation Results Summary

Metric	Value	Unit
Total site energy consumption	696,880.70	kWh/year
Interior Equipment (IT + other electrical loads, all zones)	492,176.16	kWh/year
Average equipment load	56.2	kW (continuous, across all zones)
Annual cooling energy	97,851.28	kWh/year
Interior Lighting	94,756.23	kWh/year
Fan electrical energy	10,401.64	kWh/year
Pump electrical energy	1,695.39	kWh/year
Heating energy	0	kWh/year (disabled)
Simulated PUE	1.4159	dimensionless ratio (benchmark 1.5 to 2.0)
Annual operational CO2 emissions	162,303.5	kg/year (0.2329 kgCO2/kWh, Italian grid, to be verified/cited in the final version)
Embodied carbon (supplementary, from DesignBuilder construction report)	84,915.4	kgCO2 (one-time; not an LCA, see Section 1.3)
Energy intensity, whole building	1,058.76	kWh/m²/year (658.20 m² total building area)
Opaque surface area	2,566.9	m²

Metric	Value	Unit
Glazed surface area	33.8	m²

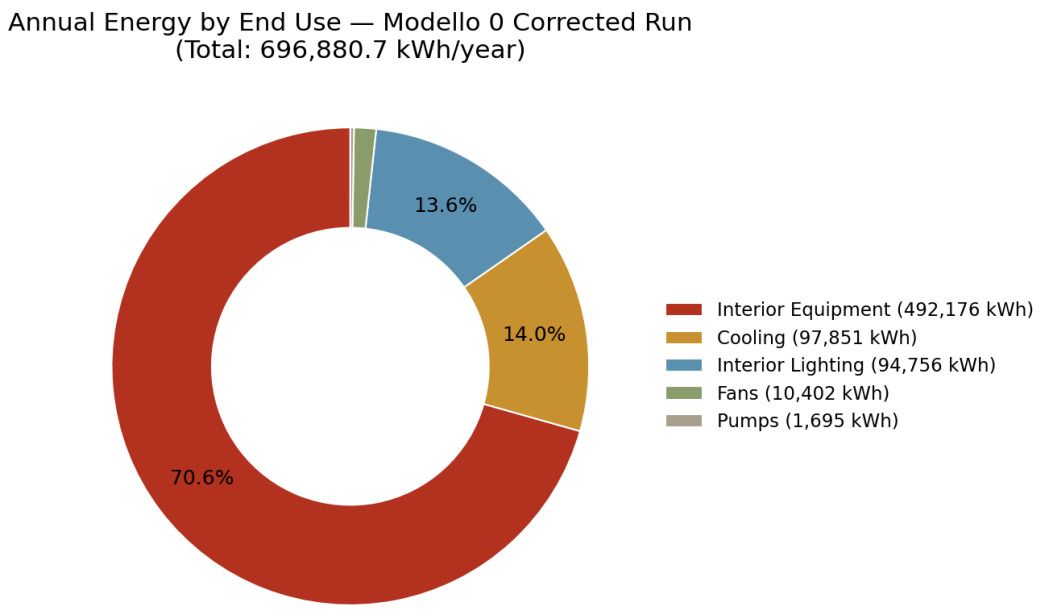


Figure 39: Energy breakdown chart: annual energy by end use (Interior Equipment, Cooling, Lighting, Fans, Pumps, Heating), EnergyPlus corrected run

Analysis shows that each year Interior Equipment consumes 70.6%, 492,176.16 kWh of total site energy usage, followed by cooling at 14.0%, lighting at 13.6%, and fans & pumps at 1.7%. Interior equipment load exceeds the individual equipment nameplate rating of Sala G's 438,000 kWh/year since it incorporates combined load requirements of all five Piano Primo zones. Heating is not used (0 kWh/year), and cooling runs perpetually. As such, this satisfies operational feasibility criteria stipulated in section 4.2.4.

The generated PUE of 1.4159 falls below the typical benchmarks for a PUE between 1.5 and 2.0 assigned for traditional data centres (Gould, 2021). An examination of the decomposition reveals that including the total amount of zonal equipment loads within the denominator automatically reduces the PUE ratio relative to an IT-only narrower denominator. Far from being evidence of a modelling glitch, this mathematical reality is confirmation that the estimated PUE value of 1.4159 makes logical sense as a plausible, order-of-magnitude value for the proposed chilled water data centre facility.

6.2.1 End-Use Breakdown by Zone

Since Modello 0 simulation includes Sala G along with the surrounding four zones, namely Control Room, Stazione Energie, Locale Batterie, and Corridor in one federated model, it implies that the yearly data provided in Table 2 represents an aggregation for all five zones. When the interior equipment electricity consumption data is broken down on a per-zone basis, it reveals that Sala G contributes around 89% of the 492,176.16 kWh/year. This was expected, as Sala G houses the sole IT equipment requiring an energy consumption rate of 50kW. On the other hand, the other four zones collectively account for the residual amount because of the small control and power loads of Stazione Energie and Locale Batterie, while the Control Room and Corridor consume nearly zero wattage. It needs to be understood that while this breakdown of data into zones helps verify the aggregate data already discussed above, it must be clarified that this breakdown is not normally available through the pipeline's current output but can be extended from Section 8.3.

6.3 Semantic and Knowledge-Graph Results

The knowledge graph imported into Neo4j consists of 24 nodes, including 5 nodes of rooms derived automatically from the federated IFC model (Sala G, Control Room, Stazione Energie, Locale Batterie, Corridor). As for equipment nodes, there are 19 equipment nodes, involving 11 nodes of CDZ units, 3 nodes of PDR units, and 5 nodes of chillers, based on the equipment cross-reference documented in Section 5.5. It is noted that the equipment nodes in the IFC file itself lack any type information and are thus classified according to their purpose in this work. There are 19 relationships explicitly represented with BRICK vocabulary relations (feeds, hasLocation). For example, 11 relations between CDZ units and Sala G based on the relation feeds, 3 relations between PDR units and Sala G based on the relation hasLocation, and 5 relations between the chillers and Sala G, again with the relation feeds. Regarding the latter, it should be highlighted that these connections were established intentionally at the room level rather than linking the individual chiller with each CDZ, despite the presence of information about the common connection to the 5-unit plant feeding the chilled water ring with 11 CDZ units, but there was no explicit document reviewed states which chiller serves which specific CDZ unit, so no such one-to-one pairing is asserted.

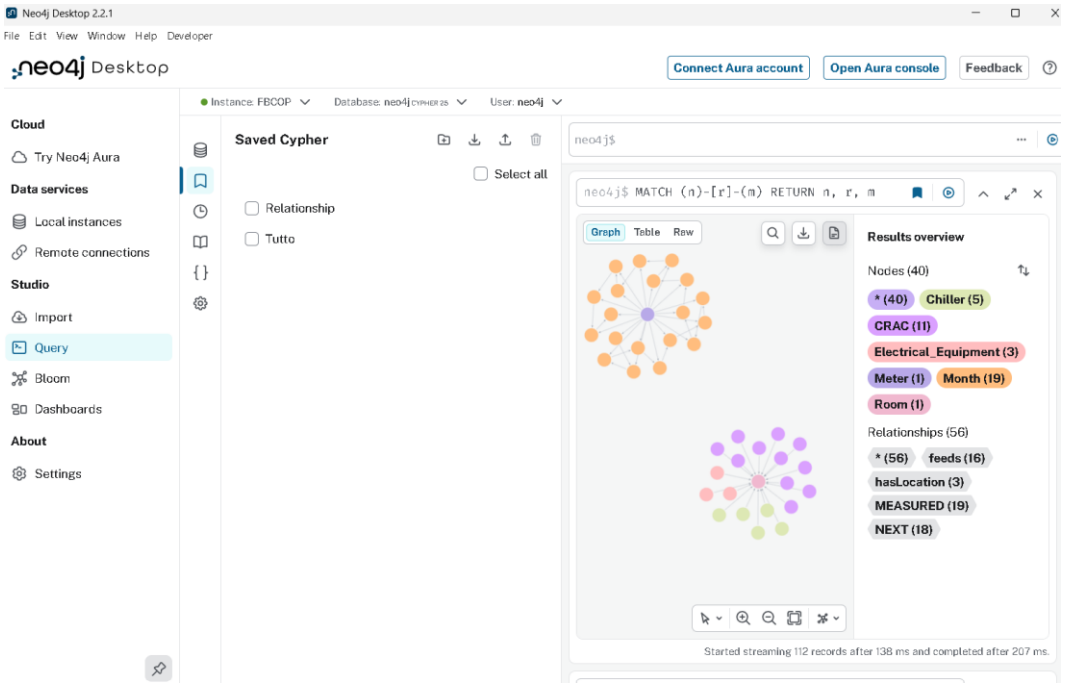


Figure 40: Neo4j graph visualisation: the FiberCop knowledge graph, showing rooms, equipment and sensors as nodes and their typed relationships as edges

Cross-verifying the relational facts between the rack rooms, the Sala G equipment connections, and the IT sensor mappings with the source BIM data proves that there is no information gap within the knowledge graph as described in Section 4.2.4, allowing the relational reasoning technique advocated by Massafra et al. (2024).

Graph Entity Count: The number of entities used within the graph, being 24 in total, is a direct implementation of the equivalent-terminal abstraction method discussed in Section 5.4 where the aggregation of 11 CDZ modules and 5 facility chillers results in individual pieces of equipment. On average, having 5 entities per zone on Piano Primo guarantees the successful operation of the whole-graph retrieval process (Section 4.2.3 and 5.9). Scaling up the model to Modello 1 (Section 8.3) will add another 10 entities only within Sala G alone.

6.4 AI-Based Decision Support Results

6.4.1 Real Measured Consumption as a Plausibility Check

The total number of months of actual readings recorded using the POD smart meters within the Fibercop premises was eighteen months ranging from August 2024 to February 2026, recorded every fifteen minutes for all of FiberCop's premises totaling up to 5,850,172.4 kWh, or 307,903.8 kWh per month on average. Based on the results obtained from the

correction factor model applied on Sala G floor along with the four surrounding floors averages out to 58,073.4 kWh (696,880.70 kWh / 12).

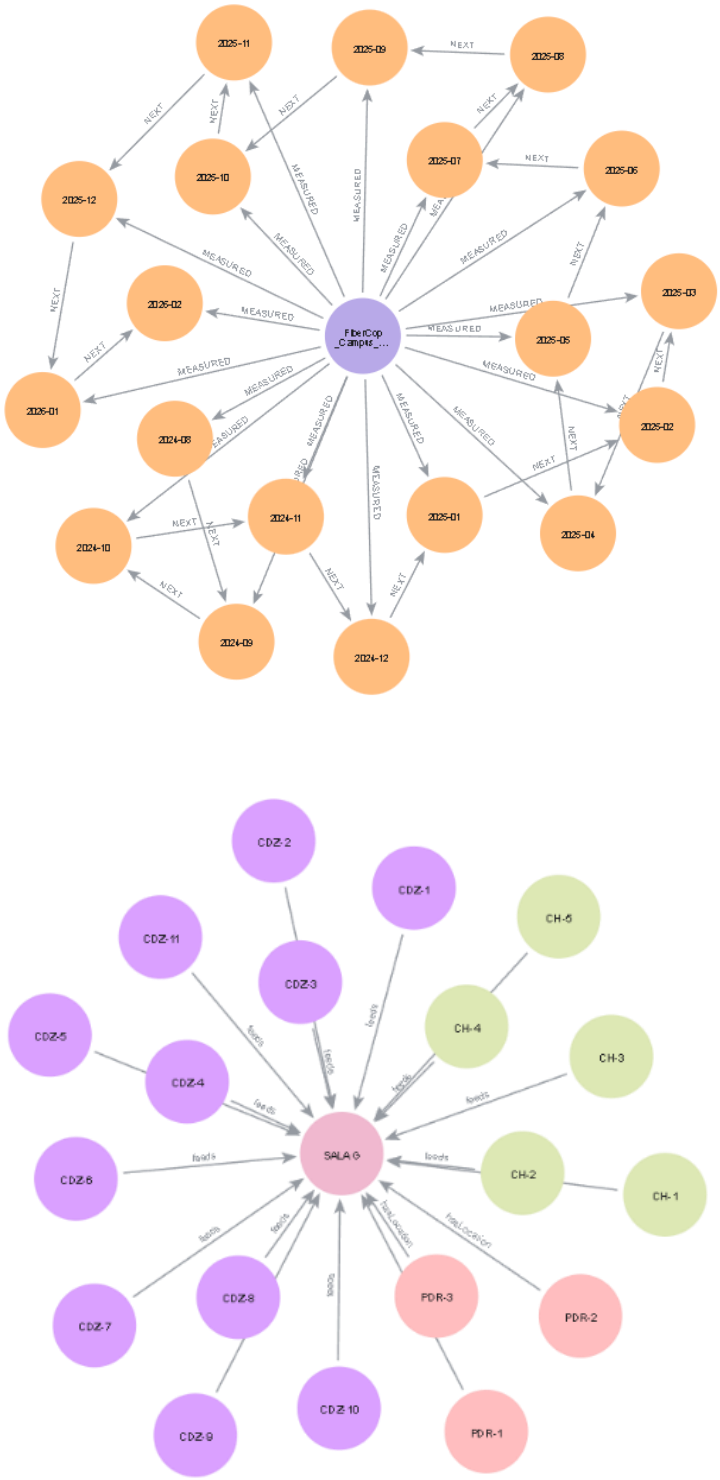


Figure 41: Neo4j graph view of the FiberCop_Campus Month-chain subgraph (built by ingest_measured_consumption_graph.py, Appendix A.9): consecutive Month nodes linked by NEXT edges, each carrying a MEASURED relationship to the campus node with the recorded consumption value.

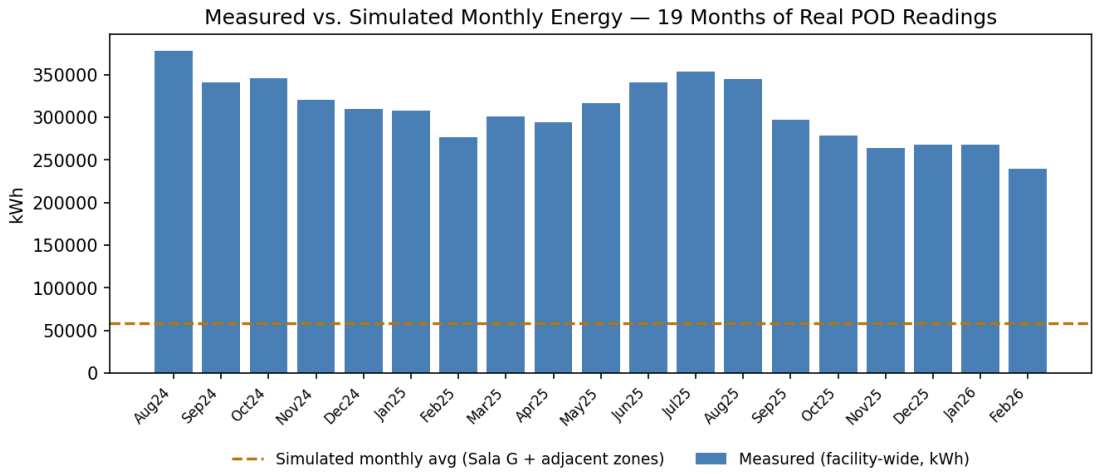


Figure 42: Measured vs. simulated monthly energy: 19 months of real POD readings compared against the modelled monthly average

The difference lies in the mismatched boundaries instead of the failure of verification: the POD meter measures the whole area of FiberCop while EnergyPlus only covers Sala G and the other four zones of Piano Primo. Thus, this comparison acts merely as a rough estimation to confirm the magnitude validity of the model instead of the matching-boundary model validation. It shall be conducted after installation of the zone-level meters in Section 1.3. In the 19-month period of data available, five days contain missing records every 15 minutes due to daylight saving time shifts.

A 19-Month Dataset: The 19-month dataset starting from August 2024 until February 2026 is crucial in providing temporal validation throughout several years, which proves seasonal consistency. By comparing overlapping periods of August to February reveals consistent repeating month-to-month patterns, indicated by the lack of spikes in summer periods. The flat pattern demonstrates the constant running loads rather than space-conditioning needs caused by changing temperature. This supports basic assumptions made in Modello 0 (Section 3.6). Despite that, the overall pattern still cannot substitute for any sensor-based level validation in future research.

6.4.2 Representative Query Evaluation

Under Section 4.2.4's evaluation metrics, a pre-defined list of seven facility manager questions was posed to the implemented RAG model, following all the corrective measures outlined under Sections 5.9.1 to 5.9.3, while using the reproducibility parameters specified under Section 5.9.3 (temperature=0, seed=42).

For correctness evaluation, this means comparing against the ground truth verifiable from sources mentioned in Chapters 3, 5, and 6.

Grounding checks whether information contained in an answer is present within the relevant retrieval contexts irrespective of its accuracy.

Lastly, traceability checks whether the sources field within the dashboard accurately mentions contributions from either the collections or the graph queries.

Table 11: RAG Query Evaluation Results

Query	Correctness	Grounding	Traceability
What is the PUE of the data centre and how does it compare to industry benchmarks?	Partially correct, simulated PUE (1.4159) correctly reported; the industry-benchmark figure was hallucinated in one run (Section 5.9.3)	Partial, PUE value grounded in the retrieved simulation chunk; benchmark figure not grounded in any retrieved chunk	Confirmed, sources field correctly listed bim_json, simulation, source_documents
Is the cooling system adequately sized for the current IT load?	Honest non-answer, system reported insufficient information rather than guessing	N/A, no unsupported claim made	Confirmed, sources field reflects the collections actually searched
What happens to energy consumption if we add 10 more server racks?	Honest non-answer, system reported insufficient information rather than extrapolating without a supporting model	N/A, no unsupported claim made	Confirmed
Which month has the highest measured consumption, and why might that be?	Correct after the fix in Section 5.9.2 (August 2024, 378,443.3 kWh); an earlier run before fetch_extreme_month_fact was added answered incorrectly (January, 268,365.0 kWh)	Correct after fix, grounded in a direct Neo4j aggregate query rather than a similarity match	Confirmed, sources field reported neo4j_month_graph_aggregate
What is the relationship between the CDZ units and the chiller feeding Sala G?	Incorrect, hallucinated a direct one-to-one chiller-to-CDZ relationship not present anywhere in the pipeline's real data (Section 5.9.3)	Not grounded: the specific relationship claimed does not appear in the Neo4j graph or	Contradiction: the sources field listed the equipment graph as a contributor even though the specific claim made was not derived from it

Query	Correctness	Grounding	Traceability
		any ChromaDB collection	
How much of total energy consumption is cooling versus equipment?	Mostly correct, with appropriate scope caveats about the federated model covering five zones rather than Sala G alone (Section 6.2.1)	Correct, grounded in the simulation collection's end-use breakdown	Confirmed
How does the measured facility consumption compare to the simulated result, and why do they differ?	Honest non-answer: the system did not spontaneously compute the 5.30x ratio reported in Section 6.4.1, though the underlying data needed to do so is present in the retrieved context	N/A, no unsupported claim made, but arguably over-cautious given available data	Confirmed

Of the seven evaluation prompts requested of the system, three had accurate answers based on factual lookup, namely identifying the month with highest consumption after fixing the problem, splitting out cooling from equipment consumption, and computing PUE on repeatable simulations. Meanwhile, three questions elicited true no-answer responses, specifically whether the capacity of the cooling system was sufficient, predicting future consumption on hypothetical rack scaling, and comparing actual consumption vs. simulated consumption figures.

Lastly, one answer generated explicit hallucinations on CDZ-chiller relations (Section 5.9.3). As the system successfully retrieved relevant information across the entire set of seven questions, any other error occurred exclusively during generation rather than context retrieval.

Query Selection Rationale: As a test suite, this paper favored certain types of queries over uniform distribution across all dimensions. There were three types of factual and relational lookup prompts (PUE, cooling sizing, CDZ relations), two for aggregation calculations (highest peak month, split energy consumption), and two involved prediction/explanation prompts (rack scalability effects, reconciling measurements with simulation). Through this arrangement of prompt types, this evaluation provides a best-case scenario benchmark for systems since existing literature on RAGs favor query types known to yield good performance (He & Jazizadeh, 2025).

6.5 Cross-Criterion Interpretation

An analysis of the four assessment domains paints a picture of how the entire system operates. From the interoperability assessment (Section 6.1), it is evident that whereas the geometries and zones transfer well, HVAC systems and loads do not. This explains why there was a need for template creation manually within DesignBuilder (Section 6.2) instead of automating everything into one system flow. Additionally, knowledge graph evaluation shows that the simplified knowledge graph with its 24 entities and 19 relationships supports graph context retrieval completely (Section 6.3). Full graph retrieval is enabled by the design choice of this particular structure, thus ruling out any vector similarity search through the graph in support of decision-making (Section 6.4). As stated earlier, this approach will only work with Modello 0 but must undergo expansion for multi-property operations (Section 7.8).

Comparisons at the level of a whole building versus a single floor show five times more consumption in actual usage than what was obtained from the simulation. Only when broken down into its constituent sub-metering ratios does this ratio gain any analytical meaning. Without sub-metering data at the building level, this number should serve no more purpose than an estimate check. Calibration ratios require validation before assuming full accuracy.

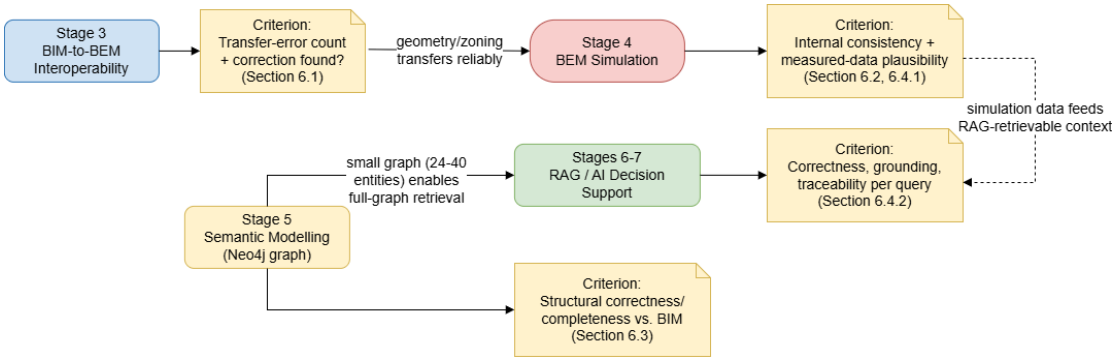


Figure 43: Cross-criterion summary diagram: four evaluation criteria (interoperability, simulation plausibility, semantic completeness, RAG correctness/grounding/traceability) mapped against the pipeline stage each depends on

The Role of Scale: Scale, in terms of spatial and structural forms an underlying factor that makes the technical solution possible in each of the evaluation categories. For example, interoperability scripting in Chapter 6.1 was made possible since the single room footprint only generated limited problems concerning surface geometry issues. Furthermore, verifying the completeness of the knowledge graph using manual processes in Chapter 6.3 was feasible purely based on its limited number of 24 entities. This kind of testing could not have been carried out if a greater number of building components had been involved, whereby sampling would be required, and lower confidence levels obtained as a

consequence. Last but not least, injecting the entire graph context into the RAG pipeline in Chapter 6.4 became possible exclusively owing to minimal numbers of nodes and relationships. Therefore, scale serves as a fundamental boundary condition for the system where the overall architecture of the BIM to BEM to RAG pipeline still holds true, yet empirical performance outcomes may not apply elsewhere without changes to the architecture (Chapter 7.8).

6.6 Reconciliation with Source Equipment Documentation

In this section, there is bridging between sections 6.1-6.5 empirical outputs and chapter 7 analysis through reconciliation inputs as a reporting component of the study process. According to the categorization framework proposed in section 4.2.7, where findings were either confirmed, contradicted, or undetermined, the process would be traced. Reconciling the physical manufacturer records (Appendix H) for the equipment installation and procurement process and model assumptions as discussed in section 5.8.2 produced major modifications. The first modification is in relation to the CDZ cooling capacity-based fleet. While the previous estimate was not confirmed at 20 units, the current value is confirmed as 11 units (model RC Group NEXT DL CW 036 H3S with 404.8kW). Secondly, the cooling return-air set point is corrected from an assumed 22°C to 24°C based on actual operation. Empirical modifications will be reflected in all sections discussing systems narrative specification, including Sections 3.4, 5.4, and Appendices B, D, and F.

Input modifications only apply selectively to specific evaluation areas addressed in Sections 6.1 to 6.5. Interoperability evaluation area results (Section 6.1) have not been altered due to the independent nature of schema transfer processes concerning the number of assets and setpoints. Semantic modeling evaluation results (Section 6.3) were likewise unaffected by input changes since the knowledge graph represents a condensed CDZ vehicle set as one aggregated `CDZ_Equivalent` element. Energy simulation results (Section 6.2) and PUE indicators (Section 6.4.1) have been affected directly due to the revised cooling load of 404.8 kW functioning at 24 °C against a previously unverified value of 22 °C.

Maintaining Traceability: In order to maintain traceability procedures outlined in Section 4.2.4, Sections 6.1 to 6.5 have maintained their previous post-input correction simulation results. Future research efforts should therefore focus on conducting a new simulation using validated "Modello 0-Rev" data (Section 5.8.2). Practically speaking, this inconsistency proves the theory stated in section 7.1, where discrepancies in inventory data assumptions

based on BIM systems and purchasing logs show how critical asset metadata information is often lost during typical BIM exports without additional documentation validation.

The case study brings out another failure mechanism concerning interoperability aside from the file-to-file conversion discussed in chapter 2. Although the six factors outlined in section 6.1 focus on loss of information in one single conversion phase such as the export/import of gbXML or the extraction of IFC files, the problems of incorrect counts of equipment and incorrect setpoints arise due to failure of propagation through time. Historical data available from 2013's drawing could not get to the federated BIM model simply because there was no modeling done based on these primary documents. This shows that the problem of interoperability goes beyond formats, hence proving that format-specific safeguard mechanisms are insufficient.

6.7 Directional Sensitivity of the Reported Results to Key Assumptions

Without any changes to the numbers via re-run of the model in Section 6.6, one may draw upon physical principles and architecture to understand directional implications based on Appendix F assumptions made before Modello 0-Rev:

1. IT Load (F4 Assumption: 50 kW): With Interior Equipment responsible for 70.6% of total site energy according to Section 6.2, any increase in the real IT load increases total site energy linearly while having a proportionately lesser influence on cooling due to constant chiller efficiency loss.
2. Chiller COP (F2 Assumption: 3.5): Cooling energy decreases with increases in COP inversely. Considering cooling accounts for 14.0% of total energy, the variations in COP greatly affect cooling-related totals whereas its influence on total site energy is minimal.
3. Cooling Setpoint (Section 6.6: 22°C to 24°C): Higher cooling setpoints lead to reduced operation time for both cooling coil and fans/pumps. Maintaining 22 degrees Celsius without correction implies reported cooling values are underestimated in comparison to those at 24 degrees Celsius.
4. Floor Area (F14 Assumption: 297.72 m²): Floor Area has a direct relationship with normalized site energy consumption measured in kWh/m^2/yr whereas total annual energy (kWh) does not vary since it depends solely upon internal and envelope.

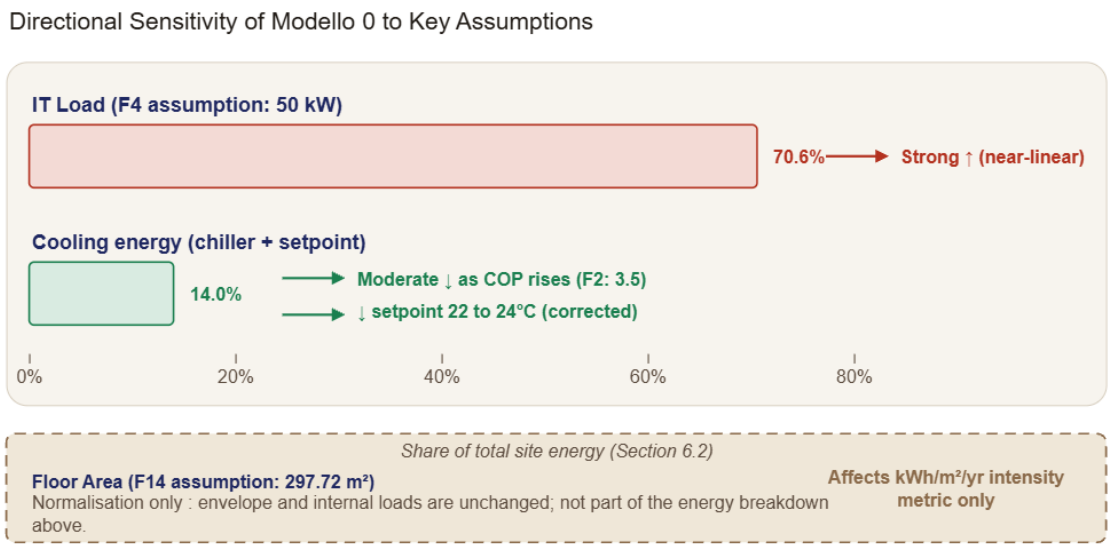


Figure 44: Directional sensitivity of Modello 0 to key assumptions in Section 6.7 (qualitative directions only; the baseline was not re-run under these assumptions).

Directional analysis does not replace Modello 0-Rev; instead, it offers guidance towards analyzing the sensitivity of Modello 0. This approach allows for judgment calls regarding the effect of open parameters on the output baseline values, which are stated as 696,880.70 kWh/year and 1.4159 PUE. The baseline numbers retain validity and physical plausibility within Modello 0, although still subject to verification via corrections and re-runs (Section 4.2.4).

Terminal Equivalence Effects: In contrast to open scalar parameters, terminal equivalence (Section 5.4) modifies structural resolution but leaves overall loading intact. Reducing 11 CDZ blocks down to one block aggregate cancels out differences in unit duty cycles while retaining validity with respect to room energy balances as a whole. In other words, although terminal equivalence improves overall accuracy in room energy predictions in Modello 0, inquiry at finer scales requires the disaggregated Modello 1 architecture (Section 8.3).

7. Discussion

7.1 On the Interoperability Gaps

After discussing internal mechanisms in detail, this dissertation analyzes other operational impacts through Section 7.1 to Section 7.4 where stage-related results will be discussed based on literature in Chapter 2; Section 7.5 to Section 7.8 deals with overall evaluation regarding trade-off and scaling issues and cost effectiveness; finally, Section 7.9 to Section 7.11 covers how FiberCop was used during operations and its methodology in research approach.

Out of total six BIM-BEM interoperability challenges mentioned in Section 6.1, all match previously identified literature gaps. Porsani et al. (2021) found the exact failure of floor geometry and HVAC systems; while Yang et al. (2022) identified precisely the same floor slab boundary problem and Ciccozzi et al. (2023) found HVAC transferability problems as major unresolved issue in their review paper analyzing 532 studies. In addition to proving the validity of existing literature gaps into data centers domain and automated fixing mechanism using scripts of floor slabs' problem, an additional unique gap discovered in this thesis includes federated vs. architectural Energy Analytical Model issue discussed in Section 5.1.1 which has not been addressed yet.

Schema-Level Constraints: The gaps identified across the literature screened out by Ciccozzi et al. (2023) arise due to constraints associated with schemas in both cases rather than any software glitches. While gbXML was designed for efficient transfer of building geometry and zoning information, IFC was meant for representation of assets within a project from multiple disciplines. Information related to HVAC systems would fall well within the scope of IFC rather than gbXML's schema design that had already decided upon restricting itself decades ago when considering how easy it would be to facilitate the transfer process. With the advent of AI-driven systems, it became increasingly important to develop a consistent strategy for handling HVAC data through both formats at once.

The use of gbXML for handling geometric information while employing IFC for semantic extraction proves to be a successful implementation since it addresses limitations arising in each case based on their primary purpose.

7.2 On Automation

DesignBuilder needs templating and parameterization to be done manually; thus, EnergyPlus modeling will occur manually rather than programmatic. Literature in progress (e.g., EPlus-LLM, agentic IDF workflows by Zhang et al. and subsequent fine-tuning approaches described in Sections 2.4 and 2.6) aims at automated BEM creation; however, this dissertation confines its pipeline only to RAG-driven decision-making support, which includes approaches suggested by Bujari et al. (2021) and those in the realm of 2026 digital twin governance, discussed in Section 2.7.

Furthermore, in MCP-agent comparisons conducted (Section 2.7), RAG architectures perform worse than agents in cases of compliance and rigorous checks. FiberCop intentionally restricts itself to discovery/explanatory decision support without compliance checking or automated correction capabilities.

The Pipeline Runner: The end-to-end Pipeline Runner, which comprises the IFC Extraction, Brick Schema generation, Utility Data Aggregation, ChromaDB Ingestion, and Neo4j Loading processes chained up as one command, helps save time and effort during the engineering process. Yet, it is not possible to automate the Revit export procedure and DesignBuilder/EnergyPlus simulation due to limitations in BIM/BEM application programming interfaces.

This dissertation emphasizes reproducibility rather than full automation (Chapter 5.8). Even though creating a BIM model at the early stage and performing BEM analysis require human input, all the further tasks are determined by scripts. After generating the outputs of a particular model, any future modifications to the downstream knowledge graph, vectors, and RAG databases can be performed without requiring any additional engineering efforts. Immediate benefits are evident for facility managers analysing various scenarios (Chapter 6.7) instead of striving for impossible GUI-based automation throughout the entire pipeline.

7.3 On Data Security and Local Deployment

It is important to mention that the workflow runs entirely offline where tools like Revit and DesignBuilder operate locally, the parsing of IFC data files occurs using IfcOpenShell locally, ChromaDB and Neo4j databases operate locally as persistent storage facilities, and Llama 3.2 is run offline using Ollama. Facility, equipment, and simulations-related information is never transferred externally under any circumstance.

There were several constraints which played a critical role in defining the architecture design decisions which included the following tradeoffs:

1. Model selection in favour of smaller models like Llama 3.2 instead of large models hosted on clouds services (Section 4.2.5). This decision favoured data privacy and security considerations whereby infrastructure data did not transfer out of the network boundary, thus adhering to best-practice digital twin data governance principles (Li et al., 2025; Ly et al., 2026; Section 2.5).
2. The deterministic embedding approach involves a hash function-based embedding performed through ChromaDB embedding which did not require dependency on any pre-trained models although it led to lower retrieval ranking efficiency. This approach was subsequently abandoned for the pre-trained model embedding in ChromaDB (Section 5.9) to increase retrieval ranking efficiency.

7.4 On the Decision-Support Value

This study conforms to the proactive human-in-the-loop decision approach proposed by Bujari et al. (2021), and 2026 Digital Twin Governance. This research relies on simulation and equipment-based data unique to each facility, contrary to general guidance on energy-efficiency measures. The suggested approach uses simulations to generate actionable cause-and-solution suggestions for facility managers (Section 6.4.2).
Human-in-the-loop is explicitly stated as a design constraint and not a mere caveat. The chatbot does not have any rights to alter BIM models, EnergyPlus files, or control building automation systems. Once it offers its insight via dashboards, the process automatically terminates at once (Section 5.9). Adherence to this strict boundary condition guarantees that the multiple-traces capability established in Section 6.4.2 remains useful as an active decision-making check and not simply a historical audit trail for independent operations.

7.5 Framework Contribution

This study proposes an overall BIM-to-BEM-to-RAG methodology in Section 4.1 to provide practical solutions for filling each identified gap in Table 5. This is fundamentally distinct from a tailor-made software application. Evaluation does not revolve around numerical values generated in any one particular application but how effectively each decision-making process closes the gaps present within the BIM/BEM/Artificial Intelligence space. Chapter 6 empirical evidence illustrates the validity of this approach when applied to a substantial building project. This serves as the fundamental basis for generalizability testing detailed in Section 7.8.

7.6 Limitations

Among others, some of the major limitations that apply to the existing implementation can be summarized as follows:

1. **Lack of Physical Sensors:** In the absence of sensor data, the entire simulation becomes an unvalidated plausibility test. The validation through the nineteen month meter comparison (Section 6.4.1) only serves partial purpose since there is discrepancy in the spatial boundaries of simulation.
2. **Modello 0 Aggregation:** With aggregation of the eleven CDZ units to one terminal point and ignoring the unit dynamics by taking the average chiller COP into account, accurate results cannot be achieved until the re-run of Modello 0 Rev and Modello 1 separately (Section 8.3).
3. **Unverified Default Building Parameters:** Since the absolute energy demand magnitude varies depending on DesignBuilder's default U-values, they should at least remain constant within reasonable ranges of uncertainty.
4. **LLM Hallucination Problem:** There have been observations of hallucinations during early-stage testing which confirm the absolute necessity of the human-in-the-loop procedure mentioned in Section 6.4.2.
5. **IDF Code Bug:** Due to an inconsistency in Fan:OnOff/Fan:ConstantVolume objects, the IDF exported using DesignBuilder fails when launched independently via EP
6. **Scalability Issue with Full Graph Retrieval:** The keyword overlap issue of the hash function affects the debugging process (see Section 5.9); however, utilizing the trained model offered by ChromaDB eliminates the semantic search problem (see Section 5.9.1). Architecturally speaking, Neo4j fetches the whole knowledge graph instead of utilizing similarity searches. This works well in a small 24-node prototype; nonetheless, it does not scale to multiple building scenarios due to the lack of dynamic subgraphs.
7. **Time Resolution Limitation in Ingestion Pipeline**: As opposed to BIM's direct parsing of IFC files through the Python module "ifcopenshell," BEM import requires parsing the generated HTML export document ('Riepilogo_New.htm') by DesignBuilder software. This poses two risks; firstly, relying on string parsing reduces robustness, while limiting the time resolution to annual data points by dividing each year by twelve instead of real monthly values.
8. **Mismatches Between Simulated Data and Documentation:** The timing procedures for energy calculation data and the current 11 unit documentation files are inconsistent. In contrast with other methodological limitations that necessitate acquiring new hardware,

gaining access to new locations, or conducting tests in different seasons (Points 1 to 6), this issue is purely procedural. All pipelines have been validated as performing correctly; thus, addressing the problem entails only executing the validated Modello 0-Rev simulation again.

7.7 Comparison Against Benchmarks in the Literature

The literature review requires careful comparison because there are not many previous studies using similar factors related to the facility types, climatic conditions, architecture of cooling system, and analysis methods as presented in this thesis. Any comparison between this research output and previous studies will be based on orders of magnitude and trends only.

- Benchmarking: The simulated PUE of 1.4159 obtained in section 6.2 falls short of the conventional ranges for data centres according to Gould (2021), which lie in the interval of 1.5 to 2.0. Though plausible considering Sala G's architecture, it needs validation through sensor data.
- Accuracy of RAG: According to He and Jazizadeh (2025), the prediction accuracies were 86 percent, 97 percent, 77 percent, and 49 percent for four different categories, respectively. Since the scope of their study involves single-household residences whereas FiberCop's RAG is designed for multiple collections, the outcome may be seen only as a baseline trend. Instead of being used as the passing criterion for the current case, it serves as a comparative basis; specifically, the comparison can help determine if accuracies tend to be higher for factual/device-state queries relative to hypothetical queries.

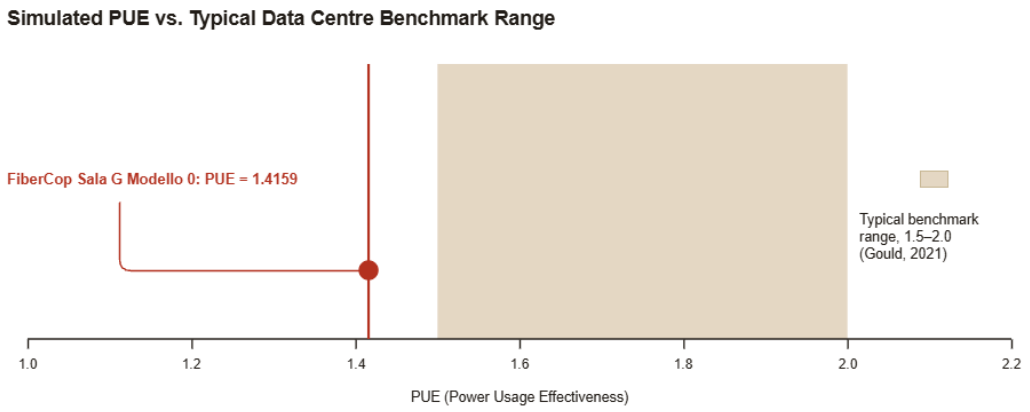


Figure 45: Simulated PUE (1.4159) plotted against the typical data centre benchmark range of 1.5-2.0 (Gould, 2021)

Interoperability-wise, Yang et al. (2022) and Ciccozzi et al. (2023) corroborate the existence of common failures concerning Floor Boundaries and HVAC Transfers within the existing literature. Errors 1 and 2 (Section 5.2) separately corroborate this established trend rather than making any groundbreaking revelations of their own.
This dissertation builds further upon these studies by proposing an automated correction tool for Error 1 (Appendix A.1). Furthermore, this dissertation highlights another interoperability problem pertaining to the Energy Analytical Model when exporting federated data vs. architectural-only models (Section 5.1.1), constituting the primary contributions to interoperability (Section 8.2).

7.8 Generalisability Beyond Sala G

Generalizability across the stages in the pipeline differs greatly:

1. Interoperability between BIM and BEM models (High): Problems related to floor boundary duplication arise due to exporter behavior within Revit and DesignBuilder instead of site-specific geometric characteristics. The generic correction script provided (see Appendix A) should work on resolving gbXML issues with `AdjacentSpaceId`.
2. Ontology construction and RAG architecture (Moderate): The script used to convert the IFC data into Brick ontology objects works against any JSON schemas generated by the IFC converter regardless of how many rooms there are in the building. The contextual injection technique applied to the entire graph in Section 5.9 assumes working with a small graph composed of only 24 nodes. Multi-building scalability necessitates either subgraph filtering or similarity searches as discussed in Section 7.6.
3. Simulation Results of BEM Model (Site-specific): All figures about energy consumption and allocation by different end-use types strictly depend on the Sala G's envelope and HVAC system specifications along with its equipment loads.
4. Evaluation Criteria for RAGs: While the evaluation criteria such as accuracy, grounding, and traceability are completely transferable to any domain, the query set used for evaluations in Section 6.4.2 was generated specifically considering Sala G’s problems. Conducting similar evaluations for another deployment requires creating a new set of queries relevant to the respective domain.

7.9 Practical Recommendations for FiberCop

From Chapter 6 comes the emergence of four operational recommendations immediately applicable to FiberCop without requiring any initial sensor installation. Firstly, reuse of the gbXML floor boundary script (Appendix A.1) for subsequent Revit exports could effectively solve persistent problems encountered in native exporters. Secondly, sub-metering of Sala G operations from the common five-unit chiller plant will allow capturing of operational allocation details including supply/return temperatures, flow rate measurements, and capacity allocation. These pieces of data would support improvement of the Modello 0-Rev run simulation results (Section 6.6) as well as future disaggregations. Thirdly, launch of the dashboard AI assistant tool among the facility managers accompanied by the correctness, grounding, and traceability testing results in Section 6.4.2 would ensure appropriate interpretation and application of outputted information according to existing system reliability. Finally, continuous consolidation of the 19-months worth of utilities data (Appendix C) is suggested.

In addition to that stated above, another organizational recommendation arises after identifying the gaps in documentation resolved in Section 5.8.2. FiberCop must develop an index of versions of all documents in Sala G, specifying any of these three possibilities regarding the validation of operational hypotheses based on historical records confirm, contradict, or leave undetermined. Even if it is non-technical in nature, its implementation would protect against using outdated information such as outdated blueprints or inaccurate numbers of equipment.

7.10 Cost and Deployment Considerations

There should be no additional recurring fees for software licenses for this process pipeline since it uses only free and open-source tools including Python, IfcOpenShell, rdflib, brickschema, ChromaDB, Neo4j Community Edition, FastAPI, Ollama, and Llama 3.2 as well as institutional licenses for Autodesk Revit and DesignBuilder. The main bottleneck lies in the computing power available locally because running Llama 3.2 using Ollama demands dedicated memory and GPU computation although CPU-based processing works effectively too. No extra cost arises for other computing resources for scaling up, as there is no extra software fee apart from labor costs associated with modeling in BIM and simulations in DesignBuilder.

7.11 Reflections on the Research Process

The two fundamental limitations imposed upon the scope of empirical research in this study are: First, it was established through the equipment inventory verification procedure and setpoints reconciliation carried out in Section 5.8.2 that BIM models retain inherent limitations imposed by drawings, and thus any simulation performed before verifying its accuracy compared to procurement documents is not entirely accurate.

Second, although the representative RAG assessment conducted in Section 6.4.2 was completed in its entirety, comprising all seven queries analyzed, only one task involving the re-run of Modello 0-Rev simulation based on the verified configuration (11 units, 404.8kW) and setpoint of 24°C remains to be completed among other future research tasks mentioned in Section 8.3.

Detecting Different Error Types: For errors related to BIM to BEM interoperability, the issues can be identified easily due to direct import failure and rejection of surfaces. For semantic modeling errors, the issues are silent in nature, but checking these manually based on comparisons with source BIM information is not a problem. However, with regard to AI decision support errors, these would be difficult to detect since hallucinations may appear to give correct responses.

8. Conclusions

Chapter 6 directly answers all three research questions in section 1.2. At the same time, it provides a summary of all the contributions made in the pipeline. Furthermore, the chapter gives insight into the limitations of this study together with suggestions for further research based on Chapters 6 and 7. Following the methodology employed within the dissertation, this chapter indicates the validity and limitations of the empirical evidence used.

8.1 Answers to the Research Questions

Q1 (BIM-to-BEM Interoperability): There were six modes of failure regarding interoperability in the data centre pipeline, namely Floor Slab Duplication, HVAC/IT Loads Non-transferable, Incompatibility between IDF and EP-Launcher, Restrictions for Direct IFC Inputs, and Issues in Federated Energy Analytical Models (Section 6.1). Five problems out of those could be addressed through reproducible scripts. Just as mentioned in the Chapter 2 literature review, this study found evidence supporting the idea that these failures arise due to software boundaries in ecosystems but not due to modelling mistakes at the site level.

Q2 (Energy Performance Benchmark): The baseline simulation gives a PUE value of 1.4159, which falls short of the 1.5-2.0 range stated by Gould (2021). IT devices contribute to 70.6 percent (696,880.70 \kWh) of total yearly site energy usage, whereas cooling contributes 14.0 percent and lighting 13.6 percent. Cooling demands arise from internal load requirements rather than fluctuating ambient temperature changes throughout the year.

Q3 (Feasibility of Localhost RAG System for Cause-Suggestion Facility Management Queries): The feasibility study shows that such a system can operate entirely within the realm of localhost resources. The architecture employs a Neo4j knowledge graph database and a ChromaDB vector store (Section 5.5). The proof of feasibility comes directly from the representative query test conducted in Section 6.4.2. This assessment must always be done alongside taking into account the potential risk of hallucinations outlined in Section 7.6.

8.2 Contribution

In conclusion, this thesis tries to offer a reproducible methodology of BIM-to-BEM-to-RAG that is theoretically and empirically validated via an operational data centre use case,

an aspect which is often lacking in current research literature regarding sustainable buildings.

While the development of a standalone algorithm may be viewed as less valuable compared to a theoretical proof-of-concept, this research focuses precisely on establishing a chain of interdependent processes, each filling a recognized gap within the existing body of scientific knowledge and addressing every interoperability issue with explicit explanations.

A Scope Limitation: In accordance with design-science research guidelines, there is a clear limitation of scope to this study.

Firstly, this research does not validate the operational energy model through real-world implementation, nor does it automate certain processes in relation to BIM or BEM creation; thirdly, no claims can be made in regard to the reliability of LLM output until thorough empirical validation has been conducted. This study has produced an accountable structure which exposes limitations and provides resolution strategies for each issue faced.

8.3 Limitations and Future Work

The next phase should begin by re-running the "Modello 0-Rev" with confirmed 11 unit/404.8 kW and 24°C setpoints (see Appendix H), rather than the initial baseline values stated in Chapter 6. Future research priorities revolve around addressing seven gaps related to the current pipeline (Section 7.6):

1. Direct access to BEM: Direct query on `eplusout.sql` to avoid parsing HTML report summary for real-time simulation outputs to be fed into the dashboard system.
2. Model Validation: Inclusion of sensors for empirical validation beyond verification through preliminary plausibility assessment.
3. Disaggregation of Chiller Consumption: Specific consumption disaggregation from common facilities' chiller plant (Appendices F & H).
4. Developing Modello 1 Prototype: Disaggregated Building Energy Model with individual representation of 11 CDZ Units.
5. Extending Pipeline to Other Buildings: Reapplication of existing pipeline after completion of geometry and IT loads documentation of Sala L (Appendix I).
6. LLM-based Automation: Research on generating IDF and debugging using Large Language Models (Chapter 2, Section 2.4).
7. Embedding Semantic Representation: Upgrade of hash-based building information graph to semantic vectors for multiple buildings.

8.4 Implications for Practice

Three implications of the pipeline emerge for facilities managers and BIM/BEM professionals working with equivalent data centers irrespective of whether the presented pipeline was used or not.

1. Documentation Audits for “Silent Errors”: The BIM-to-BEM processes that are based solely upon detecting import issues or 'loud errors' do not address any problems related to incorrect or outdated equipment loadings, or 'silent errors'. These can only be addressed explicitly via cross-referencing with purchase orders outside the scope of exporter patching or better simulation engine performance.
2. Analysis of Aggregated Model Inputs: By consolidating inputs, such as merging several air conditioning units under one equivalent node called Modello 0, any factual inconsistencies in terms of unit numbers can become overlooked amid abstractions. All aggregate inputs should undergo scrutiny as rigorous as any individual input.
3. Knowledge Graphs Explicitly Represented for Auditable AI: By embedding artificial intelligence decision aids within the explicit framework of a knowledge graph such as Neo4j, the resulting relational outputs can retain auditability by humans and thus be safely deployed operationally.

8.5 Closing Remarks

The core thread uniting this thesis is a deliberate commitment to evidentiary rigor, stating precisely what is established and what remains unverified at every level, from individual equipment specifications to research conclusions. In presenting corrective measures (Section 6.6) and listing unresolved problems (Section 7.11), there may be certain qualifications involved.

In its application of the design process, the Sala G Data Center served as an appropriate case study for Fibercop’s actual operations. Although individually, the results achieved were relatively minor six interoperability issues identified, corrected energy simulation, a knowledge graph consisting of 24 entities, and a RAG Evaluation Framework- the totality of this effort aims to fill a notable void within existing data centers' literature. In conclusion, this research tried to provide an approach supported by previous literature.

References

Alhammad, M., Eames, M., & Vinai, R. (2024). Enhancing building energy efficiency through Building Information Modeling (BIM) and Building Energy Modeling (BEM) integration: A systematic review. Buildings, 14(3), 581. https://www.mdpi.com/2075-5309/14/3/581

Aniakor, M., Cogo, V.V., & Ferreira, P.M. (2024). A survey on semantic modeling for building energy management. arXiv:2404.11716. https://arxiv.org/abs/2404.11716

Balaji, B., Bhattacharya, A., Fierro, G., Gao, J., Gluck, J., Hong, D., Johansen, A., Koh, J., Ploennigs, J., Agarwal, Y., Berges, M., Culler, D., Gupta, R., Kjærgaard, M.B., Srivastava, M., & Whitehouse, K. (2016). Brick: Towards a unified metadata schema for buildings. Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments (BuildSys '16), 41-50. https://doi.org/10.1145/2993422.2993577

Bastos Porsani, G., Del Valle de Lersundi, K., Sanchez-Ostiz Gutierrez, A., & Fernandez Bandera, C. (2021). Interoperability between Building Information Modelling (BIM) and Building Energy Model (BEM). Applied Sciences, 11(5), 2167. https://www.mdpi.com/2076-3417/11/5/2167

Bujari, A., Calvio, A., Foschini, L., Sabbioni, A., & Corradi, A. (2021). A digital twin decision support system for the urban facility management process. Sensors, 21(24), 8460. https://www.mdpi.com/1424-8220/21/24/8460

Ciccozzi, A., de Rubeis, T., Paoletti, D., & Ambrosini, D. (2023). BIM to BEM for building energy analysis: A review of interoperability strategies. Energies, 16(23), 7845. https://www.mdpi.com/1996-1073/16/23/7845

Ciccozzi, A., Santavicca, A., de Rubeis, T., & Ambrosini, D. (2025). BIM-BEM interoperability for energy analysis: A comparative study of different strategies. Energy Reports, 13, 4705-4718. https://ricerca.univaq.it/handle/11697/262359

Del Giudice, M., Donato, A.J., Loffa, M.A., Mazzarino, P.R., Bottaccioli, L., Patti, E., & Osello, A. (2026). BIM-to-BEM framework for energy retrofit in industrial buildings: From simulation scenarios to decision support dashboards. Sustainability, 18(2), 1023. https://www.mdpi.com/2071-1050/18/2/1023

El Mokhtari, K., Panushev, I., & McArthur, J.J. (2022). Development of a cognitive digital twin for building management and operations. Frontiers in Built Environment, 8, 856873. https://www.frontiersin.org/journals/built-environment/articles/10.3389/fbuil.2022.856873

European Commission. (2020). A Renovation Wave for Europe. COM(2020) 662 final. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52020DC0662

Goncalves, R.H.d.S., & dos Santos, W.M. (2026). A scalable digital twin framework for energy optimization in data centers. arXiv:2605.05581. https://arxiv.org/abs/2605.05581

Gould, J.A. (2021). A framework for auditing data center energy usage and mitigating environmental footprint. arXiv:2102.04446. https://arxiv.org/abs/2102.04446

Hammes, S., Zech, P., Geisler-Moroder, D., Weninger, J., Steindl, G., Hauer, M., Miller, J., & Pfluger, R. (2026). Ten questions concerning building information modeling (BIM): Energy-efficiency in design, construction, and operation. Building and Environment, 296, 114497. https://doi.org/10.1016/j.buildenv.2026.114497

He, T., & Jazizadeh, F. (2025). Context-aware LLM-based AI agents for human-centered energy management systems in smart buildings. arXiv:2512.25055. https://arxiv.org/abs/2512.25055

Hevner, A.R., March, S.T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS Quarterly, 28(1), 75-105. https://doi.org/10.2307/25148625

IEA. (2026). Key Questions on Energy and AI. International Energy Agency, World Energy Outlook Special Report, pp. 10, 17. https://www.iea.org/reports/key-questions-on-energy-and-ai

Jadhav, S.A., & Liu, Z. (2026). Digital twin-based cooling system optimization for data center. arXiv:2603.01198. https://arxiv.org/abs/2603.01198

Jiang, G., Ma, Z., Zhang, L., & Chen, J. (2024). EPlus-LLM: A large language model-based computing platform for automated building energy modeling. Applied Energy, 367, 123431. https://par.nsf.gov/biblio/10513630

Jin, S., Kim, D., Lee, J., & Lee, D. (2026). Evaluating large language models (LLMs) for semantic interpretation of IFC-based BIM data. Automation in Construction, 185, 106879. https://doi.org/10.1016/j.autcon.2026.106879

Kamel, E., & Memari, A.M. (2019). Review of BIM's application in energy simulation: Tools, issues, and solutions. Automation in Construction, 97, 164-180. https://doi.org/10.1016/j.autcon.2018.11.008

Lewis, P. et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. arXiv:2005.11401. https://arxiv.org/abs/2005.11401

Li, H., Comesana, A., Weyandt, C., & Hong, T. (2025). A RAG data pipeline transforming heterogeneous data into AI-ready format for autonomous building performance discovery. Advances in Applied Energy, 21, 100261. https://doi.org/10.1016/j.adapen.2025.100261

Liu, M., Zhang, L., Chen, J., Chen, W.-A., Yang, Z., Lo, L.J., Wen, J., & O'Neill, Z. (2025). Large language models for building energy applications: Opportunities and challenges. Building Simulation, 18(2), 225-234. https://doi.org/10.1007/s12273-025-1235-9

Ly, R., Shojaei, A., Gao, X., Agee, P., & Akanmu, A. (2026). Data-driven and distributed governance of building facilities management using decentralized autonomous organization, digital twin, and large language models. arXiv:2605.16298. https://arxiv.org/abs/2605.16298

Massafra, A., Coraglia, U.M., Predari, G., & Gulli, R. (2024). Building information model analysis through large language models and knowledge graphs. Proceedings of the 42nd eCAADe Conference (eCAADe 2024), Nicosia, Cyprus, Vol. 1, pp. 685-694. https://papers.cumincad.org/data/works/att/ecaade2024_297.pdf

Průša, D., Vala, J., Šuhajda, K., Žajdlík, T., Barabášová, A., & Šťastník, S. (2026). Interoperability challenges in BIM-to-BEM workflows for sustainable building assessment: A comparative study of native and middleware-based gbXML export. Sustainability, 18(17), 8759. https://www.mdpi.com/2071-1050/18/17/8759

Vittori, F., Tan, C.F., Pisello, A.L., Chong, A., Piselli, C., & Miller, C. (2025). BIM-to-BRICK: Using graph modeling for IoT/BMS and spatial semantic data interoperability within digital data models of buildings. Energy and Buildings (published online September 2025; originally released as arXiv:2307.13197, 2023). https://arxiv.org/abs/2307.13197

vom Brocke, J., Winter, R., Hevner, A., & Maedche, A. (2020). Accumulation and evolution of design knowledge in design science research: A journey through time and space. Journal of the Association for Information Systems, 21(3), 520-544. https://doi.org/10.17705/1jais.00611

Wan, H., Xu, W., Rosenberg, M., Zhang, J., & Siddika, A. (2025). Automatic building code review: A case study. arXiv:2510.02634. https://arxiv.org/abs/2510.02634

Yang, Y., Pan, Y., Zeng, F., Lin, Z., & Li, C. (2022). A gbXML reconstruction workflow and tool development to improve the geometric interoperability between BIM and BEM. Buildings, 12(2), 221. https://www.mdpi.com/2075-5309/12/2/221

Zhang, G., Lu, C., & Luo, Q. (2025). Application of large language models in the AECO industry: Core technologies, application scenarios, and research challenges. Buildings, 15(11), 1944. https://www.mdpi.com/2075-5309/15/11/1944

Zhang, L., Chen, Z., & Ford, V. (2024). Advancing building energy modeling with large language models: Exploration and case studies. Energy and Buildings, 323, 114788. http://arxiv.org/abs/2402.09579

Zhang, L., Ford, V., Chen, Z., & Chen, J. (2025). Automatic building energy model development and debugging using large language models agentic workflow. Energy and Buildings, 327, 115116. https://www.osti.gov/biblio/2480816

Appendix A: Pipeline Script Listings

This appendix presents the codes that were used in producing the outcome discussed in this study. These include the code of gbXML Floor Boundary Correction in Section 5.2 (Error 1), and IFC to JSON Conversion to feed the RAG pipeline in Section 5.6. Lastly, it includes the POD smart metering data processing script presented in Appendix B. These scripts are presented exactly as written along with their respective comments to be reviewed again should any changes occur to the existing BIM model.

A.1 fix_gbxml_floors.py, gbXML Floor-Boundary Correction (Section 5.2, Error 1)

```
"""
fix gbxml floors.py
Corrects the duplicate AdjacentSpaceId issue on ground-floor slab surfaces
exported from Revit's gbXML exporter, following the general pattern
documented by Yang et al. (2022) for Revit gbXML floor-slab export.

Usage:
    python fix gbxml floors.py input.gbxml output corrected.gbxml
"""
import sys
import xml.etree.ElementTree as ET

GBXML NS = "http://www.gbxml.org/schema"
ET.register namespace("", GBXML NS)



def qn(tag):
    """Qualify a tag name with the gbXML namespace."""
    return f"{{{GBXML_NS}}}{tag}"



def find duplicate adjacent surfaces(root):
    """
    Identify surfaces whose two AdjacentSpaceId children reference the
    same spaceIdRef -- the signature of the Revit floor-slab export bug.
    """
    flagged = []
    for surface in root.iter(qn("Surface")):
        adjacent ids = [
            el.get("spaceIdRef")
            for el in surface.findall(qn("AdjacentSpaceId"))
        ]
        if len(adjacent ids) == 2 and adjacent ids[0] == adjacent ids[1]:
            flagged.append(surface)
    return flagged



def reclassify as slab on grade(surface):
    """
    Remove the duplicate AdjacentSpaceId and reclassify the surface type
    from InteriorFloor to SlabOnGrade, which is the correct boundary
    condition for a ground-floor slab with no interior space below it.
    """
    surface type = surface.get("surfaceType")
    if surface type == "InteriorFloor":
        surface.set("surfaceType", "SlabOnGrade")

    adjacent ids = surface.findall(qn("AdjacentSpaceId"))
    for extra in adjacent ids[1:]:
        surface.remove(extra)



def main(input_path, output_path):
```

```
    tree = ET.parse(input_path)
    root = tree.getroot()

    flagged = find_duplicate_adjacent_surfaces(root)
    print(f"Found {len(flagged)} surfaces with duplicate AdjacentSpaceId")

    for surface in flagged:
        surface id = surface.get("id")
        print(f"  correcting surface {surface_id}")
        reclassify_as_slab_on_grade(surface)

    tree.write(output path, xml declaration=True, encoding="UTF-8")
    print(f"Corrected gbXML written to {output path}")
    print(f"Surfaces corrected: {[s.get('id') for s in flagged]}")


if   name   == "  main  ":
    if len(sys.argv) != 3:
        print("Usage: python fix_gbxml_floors.py input.gbxml output.gbxml")
        sys.exit(1)
    main(sys.argv[1], sys.argv[2])
```

With respect to the Piano Primo export, the Python script was able to detect and fix the precisely five surface faces mentioned in Section 5.2; namely, aim0598, aim0819, aim0879, aim0990, and aim1139.

A.2 extract_ifc_json.py, IFC-to-Segmented-JSON Extraction (Section 5.6)

```
"""
extract_ifc_json.py
Extracts rooms, equipment and their relationships from the federated
IFC 4 model into a segmented JSON structure, following the pre-structuring
approach shown by Jin et al. (2026) to improve LLM interpretation accuracy
relative to raw IFC text.

Usage:
    python extract ifc json.py model.ifc output.json
"""
import json
import sys
import ifcopenshell
import ifcopenshell.util.element as element util


def extract_spaces(ifc_file):
    spaces = []
    for space in ifc file.by type("IfcSpace"):
        spaces.append({
            "id": space.GlobalId,
            "name": space.Name,
            "long_name": space.LongName,
            "type": "Space",
        })
    return spaces


def extract equipment(ifc file):
    equipment = []
    equipment classes = [
        "IfcUnitaryEquipment", "IfcChiller", "IfcPump", "IfcFan",
        "IfcElectricDistributionBoard", "IfcFlowTerminal",
    ]
    for cls in equipment classes:
        for element in ifc file.by type(cls):
            containing = element_util.get_container(element)
```

```
            equipment.append({
                "id": element.GlobalId,
                "name": element.Name,
                "ifc class": cls,
                "located in": containing.GlobalId if containing else None,
            })
    return equipment



def extract_relationships(ifc_file):
    relationships = []
    for rel in ifc file.by type("IfcRelContainedInSpatialStructure"):
        for element in rel.RelatedElements:
            relationships.append({
                "type": "IS LOCATED IN",
                "source": element.GlobalId,
                "target": rel.RelatingStructure.GlobalId,
            })
    for rel in ifc file.by type("IfcRelConnectsPorts"):
        relationships.append({
            "type": "FEEDS",
            "source": rel.RelatingPort.GlobalId,
            "target": rel.RelatedPort.GlobalId,
        })
    return relationships



def main(ifc path, json path):
    ifc file = ifcopenshell.open(ifc path)
    segmented = {
        "spaces": extract spaces(ifc file),
        "equipment": extract_equipment(ifc_file),
        "relationships": extract_relationships(ifc_file),
    }
    with open(json path, "w") as f:
        json.dump(segmented, f, indent=2)
    print(f"Extracted {len(segmented['spaces'])} spaces, "
          f"{len(segmented['equipment'])} equipment items, "
          f"{len(segmented['relationships'])} relationships")



if __name__ == "__main__":
    main(sys.argv[1], sys.argv[2])
```

A.3 consolidate_pod_data.py, Smart-Meter Data Consolidation (Section 6.4.1)

```
"""
consolidate pod data.py
Consolidates raw 15-minute POD smart-meter readings across 19 months
into monthly totals, flagging days with incomplete interval counts
(most likely daylight-saving transitions).

Usage:
    python consolidate pod data.py pod readings.csv monthly totals.csv
"""
import csv
import sys
from collections import defaultdict
from datetime import datetime

EXPECTED_INTERVALS_PER_DAY = 96  # 24h * 4 intervals/hour



def load_readings(csv_path):
    readings = []
    with open(csv_path) as f:
        reader = csv.DictReader(f)
        for row in reader:
            readings.append({
                "timestamp": datetime.fromisoformat(row["timestamp"]),
                "kwh": float(row["kwh"]),
            })
```

```
    return readings


def flag_incomplete_days(readings):
    counts_by_day = defaultdict(int)
    for r in readings:
        counts by day[r["timestamp"].date()] += 1

    incomplete = {
        day: count for day, count in counts_by_day.items()
        if count != EXPECTED INTERVALS PER DAY
    }
    return incomplete


def consolidate monthly(readings):
    totals = defaultdict(float)
    for r in readings:
        key = (r["timestamp"].year, r["timestamp"].month)
        totals[key] += r["kwh"]
    return dict(sorted(totals.items()))


def main(input path, output path):
    readings = load readings(input path)
    incomplete = flag incomplete days(readings)
    if incomplete:
        print(f"Warning: {len(incomplete)} days with incomplete interval counts:")
        for day, count in incomplete.items():
            print(f"  {day}: {count}/{EXPECTED_INTERVALS_PER_DAY} intervals")

    monthly = consolidate monthly(readings)
    with open(output_path, "w", newline="") as f:
        writer = csv.writer(f)
        writer.writerow(["year", "month", "total_kwh"])
        for (year, month), total in monthly.items():
            writer.writerow([year, month, round(total, 1)])

    grand_total = sum(monthly.values())
    print(f"Grand total: {grand_total:.1f} kWh across {len(monthly)} months")
    print(f"Monthly average: {grand total / len(monthly):.1f} kWh/month")


if __name__ == "__main__":
    main(sys.argv[1], sys.argv[2])
```

A.4 ingest_bim_json.py, BIM JSON Ingestion into ChromaDB (Section 5.6)

```
import json
import chromadb

with open("extracted model.json") as f:
    extracted = json.load(f)
with open("equipment_crossref.json") as f:
    crossref = json.load(f)

client = chromadb.PersistentClient(path="../02 chromadb store")
try:
    client.delete_collection("bim_json")
except Exception:
    pass
collection = client.get or create collection(name="bim json")

count = 0

# One document per room
for room in extracted["rooms"]:
    text = f"Room: {room['long name']}. GUID: {room['guid']}."
    collection.add(
        ids=[f"room_{room['guid']}"],
        documents=[text],
```

```
        metadatas=[{"entity_type":   "room",   "name":   room["long_name"],   "guid":
room["guid"]}]
    )
    count += 1

# One document per equipment item
for item in crossref["equipment"]:
    parts = [f"Equipment: {item['id']}.", f"Type: {item['brick class']}."]
    if item.get("model"):
        parts.append(f"Model: {item['model']}.")
    if item.get("capacity_kw") is not None:
        parts.append(f"Capacity: {item['capacity_kw']} kW.")
    if item.get("source"):
        parts.append(f"Source: {item['source']}.")
    text = " ".join(parts)
    collection.add(
        ids=[f"equipment_{item['id']}"],
        documents=[text],
        metadatas=[{"entity type":   "equipment",   "id":   item["id"],   "brick class":
item["brick class"]}]
    )
    count += 1

print(f"Added {count} entities to the 'bim json' collection.")
print("Collection now contains", collection.count(), "documents total.")
```

A.5 ingest_brick.py, BRICK Ontology Ingestion into ChromaDB (Section 5.5)

```
import chromadb

CHUNK_SIZE = 500



def make chunks(text, chunk size=CHUNK SIZE):
    return [text[i:i + chunk size] for i in range(0, len(text), chunk size)]



with open("datacenter brick.ttl", encoding="utf-8") as f:
    ttl_text = f.read()

chunks = make chunks(ttl text)

client = chromadb.PersistentClient(path="../02 chromadb store")
try:
    client.delete_collection("brick")
except Exception:
    pass
collection = client.get or create collection(name="brick")

for i, chunk in enumerate(chunks):
    collection.add(
        ids=[f"brick ttl chunk{i}"],
        documents=[chunk],
        metadatas=[{"source file": "datacenter brick.ttl", "chunk index": i}]
    )

print(f"Added {len(chunks)} chunks to the 'brick' collection.")
print("Collection now contains", collection.count(), "chunks total.")
```

A.6 build_neo4j_graph.py, Neo4j Knowledge Graph Construction (Section 5.5)

```
"""
build neo4j graph.py — loads the REAL extracted IFC data into Neo4j,
replacing the manually-typed graph from earlier in this project.
Thesis Section 5.5.

Reads:
    extracted model.json   (real, from extract ifc json.py — rooms + equipment
                            proxies with real GUIDs)
    equipment_crossref.json (manual cross-reference table: which documented
                             equipment item, from your FiberCop equipment
                             inventory, corresponds to which real proxy GUID
                             or room — see thesis Appendix B/H)

Writes: the BRICK-vocabulary graph into your Neo4j "FBCOP" instance.
```

```
Usage:
    python build neo4j graph.py
"""

import json
from neo4j import GraphDatabase

NEO4J_URI = "neo4j://127.0.0.1:7687"
NEO4J_USER = "neo4j"
NEO4J_PASSWORD = "ForNeo4J"



def load(path):
    with open(path) as f:
        return json.load(f)



def main():
    extracted = load("extracted_model.json")
    crossref = load("equipment_crossref.json")

    driver = GraphDatabase.driver(NEO4J_URI, auth=(NEO4J_USER, NEO4J_PASSWORD))

    with driver.session() as session:
        # Wipe existing graph first — this script is the source of truth now.
        session.run("MATCH (n) DETACH DELETE n")

        # Real rooms, from the real IFC extraction.
        for room in extracted["rooms"]:
            session.run(
                "CREATE (:Room {brick_class: 'brick:Room', guid: $guid, "
                "name: $name})",
                guid=room["guid"], name=room["long_name"],
            )

        # Real equipment, identified via the documented cross-reference
        # (not invented — see equipment_crossref.json, built from your
        # actual FiberCop equipment PDFs / Appendix B).
        for item in crossref["equipment"]:
            session.run(
                f"CREATE (:{item['brick_label']} {{"
                "brick_class: $brick_class, id: $id, guid: $guid, "
                "model: $model, capacity_kw: $capacity_kw}})",
                brick_class=item["brick_class"], id=item["id"],
                guid=item.get("ifc_guid"), model=item.get("model", ""),
                capacity_kw=item.get("capacity_kw", 0.0),
            )

        # Relationships, from the cross-reference table's documented
        # feeds/location assignments.
        for rel in crossref["relationships"]:
            session.run(
                f"MATCH (a {{id: $a_id}}), (b {{name: $b_name}}) "
                f"CREATE (a)-[:{rel['type']}]->(b)",
                a_id=rel["from_equipment_id"], b_name=rel["to_room_name"],
            )

        node_count = session.run("MATCH (n) RETURN count(n) AS c").single()["c"]
        rel_count = session.run("MATCH ()-[r]->() RETURN count(r) AS c").single()["c"]
        print(f"Loaded {node_count} nodes, {rel_count} relationships into Neo4j.")

    driver.close()



if __name__ == "__main__":
    main()
```

A.7 ingest_simulation.py, EnergyPlus Simulation Results Ingestion (Section 5.6)

```
import re
import chromadb

with open("../01_source_data/Riepilogo_New.htm", encoding="utf-8", errors="ignore") as
f:
    html = f.read()
```

```
# Strip HTML tags to pipe-delimited plain text, then collapse whitespace/pipes.
text = re.sub("<[^>]+>", "|", html)
text = re.sub(r"\|+", "|", text)
text = re.sub(r"\s+", " ", text)
text = re.sub(r"\s*\|\s*", "|", text)
text = re.sub(r"\|+", "|", text)

# Confine the end-use extraction to the "End Uses" table specifically, so
# labels like "Heating"/"Cooling" aren't accidentally matched against the
# unrelated "District Heating"/"District Cooling" conversion-factor rows
# that appear earlier in the same report.
start = text.find("|End Uses|Electricity")
end = text.find("Total End Uses|", start)
end_uses_section = text[start:end]

labels = [
    "Heating", "Cooling", "Interior Lighting", "Exterior Lighting",
    "Interior Equipment", "Exterior Equipment", "Fans", "Pumps",
    "Heat Rejection", "Humidification", "Heat Recovery", "Water Systems",
    "Refrigeration", "Generators",
]
end_uses = {}
for label in labels:
    m = re.search(r"\|" + re.escape(label) + r"\|([\d.]+)\|", end uses section)
    if m:
        end_uses[label] = float(m.group(1))

m = re.search(r"Total Site Energy\|([\d.]+)\|", text)
total site energy = float(m.group(1))
pue = round(total_site_energy / end_uses["Interior Equipment"], 4)

client = chromadb.PersistentClient(path="../02 chromadb store")
try:
    client.delete collection("simulation")
except Exception:
    pass
collection = client.get_or_create_collection(name="simulation")

count = 0

# One document per non-zero end-use metric.
for label, kwh in end uses.items():
    if kwh == 0.0:
        continue
    text doc = f"EnergyPlus/DesignBuilder simulation (Modello 0): {label} end-use energy
= {kwh} kWh/year."
    collection.add(
        ids=[f"simulation_{label.replace(' ', '_')}"],
        documents=[text doc],
        metadatas=[{"metric": label, "kwh": kwh, "source": "Riepilogo New.htm"}]
    )
    count += 1

# Aggregate figures: total site energy and PUE.
for doc id, text doc, meta in [
    ("simulation total site energy",
     f"EnergyPlus/DesignBuilder simulation (Modello 0): Total Site Energy =
{total site energy} kWh/year.",
     {"metric": "Total Site Energy", "kwh": total site energy}),
    ("simulation pue",
     f"EnergyPlus/DesignBuilder simulation (Modello 0): simulated PUE = {pue} "
     f"(Total Site Energy / Interior Equipment energy).",
     {"metric": "PUE", "value": pue}),
]:
    collection.add(ids=[doc id], documents=[text doc], metadatas=[meta])
    count += 1

print(f"Extracted end uses: {end_uses}")
print(f"Total Site Energy: {total_site_energy} kWh, PUE: {pue}")
print(f"Added {count} documents to the 'simulation' collection.")
print("Collection now contains", collection.count(), "documents total.")
```

A.8 ingest_measured_consumption.py, Measured POD Consumption Ingestion into ChromaDB (Section 6.4.1)

```
import json
```

```
import chromadb

with open("../01 source data/pod summary.json") as f:
    summary = json.load(f)

client = chromadb.PersistentClient(path="../02 chromadb store")
try:
    client.delete collection("measured consumption")
except Exception:
    pass
collection = client.get_or_create_collection(name="measured_consumption")

count = 0

for month, kwh in summary["monthly_totals_kwh"].items():
    text = f"Measured consumption for {month}: {kwh} kWh (whole FiberCop campus POD
meter, not Sala G alone)."
    collection.add(
        ids=[f"pod month {month}"],
        documents=[text],
        metadatas=[{"month": month, "kwh": kwh}]
    )
    count += 1

summary_text = (
    f"POD consumption summary: {summary['n months']} months processed, "
    f"grand total {summary['grand total kwh all months']} kWh, "
    f"average monthly {summary['avg monthly kwh']} kWh, "
    f"measured-vs-simulated ratio {summary['measured vs simulated ratio']}. "
    f"{summary['note']}"
)
collection.add(
    ids=["pod summary aggregate"],
    documents=[summary text],
    metadatas=[{"entity type": "aggregate summary"}]
)
count += 1

print(f"Added {count} documents to the 'measured consumption' collection.")
print("Collection now contains", collection.count(), "documents total.")
```

A.9 ingest_measured_consumption_graph.py, Measured Consumption Time-Series Graph (Section 5.5)

```
"""
ingest measured consumption graph.py — loads the REAL measured POD
consumption data (pod summary.json, built from FiberCop's own meter
export CSVs) into Neo4j as a small time-series graph, alongside the
existing equipment/room graph built by build_neo4j_graph.py.

Why this exists (thesis Section 5.9.1 / 6.4.2 / 7.6):
Testing showed that ChromaDB's similarity search over short, near-
identical monthly-consumption text chunks (e.g. "Measured consumption
for 2024-10: ... kWh" vs. "...for 2025-06: ... kWh") is unreliable for
exact month lookups, because the chunks differ from each other only in
the month name and the number, and general-purpose embedding models
place them very close together in vector space regardless. The CDZ-1
equipment example had the same underlying problem but a working escape
hatch: the CDZ-1 relationship is ALSO stored in Neo4j, which resolves
exact-match relational queries by graph traversal rather than semantic
similarity, so it answered correctly even when ChromaDB alone would
not have. Monthly consumption had no equivalent Neo4j representation
until this script — this closes that specific gap, so the same
"exact identifier -> Neo4j, free-text similarity -> ChromaDB" pattern
already used for equipment applies to time-series consumption too, and
provides a genuine graph-vs-vector-retrieval comparison point.

Reads:
    ../01 source data/pod summary.json  (real, from the POD meter CSV
                                         parsing script — not manually
                                         typed; see Section 5.6/Appendix)

Writes: (:Meter)-[:MEASURED]->(:Month) nodes, chained chronologically
by (:Month)-[:NEXT]->(:Month), into the same Neo4j "FBCOP" instance
used by build neo4j graph.py. This script is ADDITIVE — unlike
build neo4j graph.py, it does NOT wipe the existing graph, since the
equipment/room graph and this consumption graph are meant to coexist.
```

```
It only deletes and rebuilds its OWN Meter/Month nodes, so it is safe
to re-run.

Usage:
    python ingest_measured_consumption_graph.py
"""

import json
from neo4j import GraphDatabase

NEO4J URI = "neo4j://127.0.0.1:7687"
NEO4J USER = "neo4j"
NEO4J PASSWORD = "ForNeo4J"

METER_ID = "FiberCop_Campus_POD"



def main():
    with open("../01_source_data/pod_summary.json") as f:
        summary = json.load(f)

    months_sorted = sorted(summary["monthly_totals_kwh"].keys())

    driver = GraphDatabase.driver(NEO4J URI, auth=(NEO4J USER, NEO4J PASSWORD))

    with driver.session() as session:
        # Remove only this script's own prior output, so re-running it is
        # safe and idempotent without touching the equipment/room graph
        # built separately by build_neo4j_graph.py.
        session.run(
            "MATCH (m:Meter {id: $id}) "
            "OPTIONAL MATCH (m)-[:MEASURED]->(mo:Month) "
            "DETACH DELETE m, mo",
            id=METER ID,
        )

        session.run(
            "CREATE (:Meter {id: $id, scope: "
            "'Whole FiberCop campus POD meter (not Sala G alone) — see "
            "Section 5.9.1 and the measured-vs-simulated note in "
            "pod summary.json'})",
            id=METER ID,
        )

        for month in months sorted:
            kwh = summary["monthly totals kwh"][month]
            session.run(
                "MATCH (m:Meter {id: $meter_id}) "
                "CREATE (mo:Month {name: $name, kwh: $kwh}) "
                "CREATE (m)-[:MEASURED]->(mo)",
                meter id=METER ID, name=month, kwh=kwh,
            )

        # Chain consecutive months chronologically (list is already sorted,
        # since "YYYY-MM" strings sort chronologically as plain text).
        for earlier, later in zip(months sorted, months sorted[1:]):
            session.run(
                "MATCH (a:Month {name: $earlier}), (b:Month {name: $later}) "
                "CREATE (a)-[:NEXT]->(b)",
                earlier=earlier, later=later,
            )

        month count = session.run(
            "MATCH (:Meter {id: $id})-[:MEASURED]->(mo:Month) RETURN count(mo) AS c",
            id=METER_ID,
        ).single()["c"]
        next count = session.run(
            "MATCH (:Month)-[r:NEXT]->(:Month) RETURN count(r) AS c"
        ).single()["c"]

        print(f"Loaded {month_count} Month nodes under Meter '{METER_ID}', "
              f"chained by {next count} NEXT relationships.")

    driver.close()
```

```
if __name__ == "__main__":
    main()
```

A.10 api_server.py, FastAPI RAG Query Backend and Dashboard Endpoints

(Sections 5.9, 6.4.2)

```
"""
FiberCop Sala G — RAG query backend for the dashboard chatbot.
Matches thesis Section 5.6 (RAG Pipeline) / 5.9 (Dashboard Chatbot).

What this does, in plain terms:
1. Receives a question from the dashboard's chat box (POST /query).
2. Pulls relevant facts from ALL FIVE real ChromaDB collections
   (source_documents, bim_json, brick, simulation, measured_consumption)
   plus the full Neo4j knowledge graph.
3. Hands those facts, plus the question, to a local Llama 3.2 model running
   via Ollama, and asks it to answer USING ONLY the facts provided.
4. Returns the answer plus exactly which collections/graph contributed
   context to that specific answer — this is the traceability mechanism
   evaluated in thesis Section 6.4.2.

Requires:
- Neo4j Desktop's "FBCOP" instance running (Bolt on localhost:7687).
- Ollama running locally with a llama3.2 model already pulled.
- The five ChromaDB collections already built in ../02_chromadb_store
  (source_documents, bim_json, brick, simulation, measured_consumption).
"""


import json
import re


from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from neo4j import GraphDatabase
import chromadb
import requests


# ── Configuration — edit these if your setup differs ──────────────────
NEO4J_URI = "neo4j://127.0.0.1:7687"
NEO4J_USER = "neo4j"
NEO4J_PASSWORD = "ForNeo4J"
OLLAMA_URL = "http://localhost:11434/api/generate"
OLLAMA_MODEL = "llama3.2"
CHROMADB_PATH = "../02 chromadb store"
COLLECTION_NAMES    =    ["source_documents",    "bim_json",    "brick",    "simulation",
"measured_consumption"]
TOP_K_PER_COLLECTION = 2
GLOBAL_TOP_K = 4  # keep only the N best matches overall, pooled across all 5 collections


app = FastAPI(title="FiberCop Sala G RAG Query API")


app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_methods=["*"],
    allow_headers=["*"],
)


chroma_client = chromadb.PersistentClient(path=CHROMADB_PATH)



class QueryRequest(BaseModel):
    query: str



MONTH_NAME_TO_NUM = {
    "january": "01", "february": "02", "march": "03", "april": "04",
    "may": "05", "june": "06", "july": "07", "august": "08",
    "september": "09", "october": "10", "november": "11", "december": "12",
}



def extract_month_key(question: str):
```

```
    """Recognize a specific month reference in the question text, e.g.
    'October 2024' or '2024-10', and return it as a Neo4j Month node's
    'name' key ('2024-10'), or None if no month is named. This is a
    deliberately narrow, disclosed heuristic (regex-based, not full date
    parsing) — see thesis Section 5.9.1."""
    m = re.search(r"\b(20\d{2})-(\d{2})\b", question)
    if m:
        return f"{m.group(1)}-{m.group(2)}"
    m = re.search(
        r"\b(January|February|March|April|May|June|July|August|September|"
        r"October|November|December)\s+(20\d{2})\b",
        question, re.IGNORECASE,
    )
    if m:
        num = MONTH NAME TO NUM.get(m.group(1).lower())
        if num:
            return f"{m.group(2)}-{num}"
    return None



GRAPH_CONTEXT_CYPHER = (
    "MATCH (a)-[r]->(b) "
    "WHERE NOT a:Month AND NOT b:Month AND NOT a:Meter AND NOT b:Meter "
    "RETURN labels(a)[0] AS a label, coalesce(a.name, a.id) AS a id, "
    "type(r) AS rel, labels(b)[0] AS b label, coalesce(b.name, b.id) AS b id"
)
MONTH_EXACT_CYPHER = "MATCH (:Meter)-[:MEASURED]->(m:Month {name: $key}) RETURN m.kwh
AS kwh"
MONTH_EXTREME_CYPHER_TEMPLATE = (
    "MATCH (:Meter)-[:MEASURED]->(m:Month) RETURN m.name AS name, m.kwh AS kwh "
    "ORDER BY m.kwh {direction} LIMIT 1"
)



def fetch month fact(question: str):
    """Targeted, exact-match lookup against the Month/Meter graph built by
    ingest_measured_consumption_graph.py — only runs, and only returns a
    fact, when the question names a specific month. This is the fix for a
    second, related dilution bug found during testing (Section 5.9.1):
    an earlier version of this file always included the full 19-month
    Month/Meter graph in fetch graph context() below, which grew the graph
    context enough (24 nodes/19 relationships to 44/56) to make even an
    unrelated, previously-correct PUE answer fail. Restricting the Month
    graph to a single targeted lookup, run only when relevant, keeps the
    always-included equipment/room graph small while still resolving exact
    month questions via Neo4j rather than ChromaDB similarity search."""
    month key = extract month key(question)
    if not month key:
        return None
    driver = GraphDatabase.driver(NEO4J_URI, auth=(NEO4J_USER, NEO4J_PASSWORD))
    try:
        with driver.session() as session:
            record = session.run(
                "MATCH (:Meter)-[:MEASURED]->(m:Month {name: $key}) RETURN m.kwh AS
kwh",
                key=month_key,
            ).single()
    finally:
        driver.close()
    if record is None or record["kwh"] is None:
        return None
    return (
        f"Measured consumption for {month_key} (exact match from the Neo4j "
        f"Month/Meter graph, whole FiberCop campus POD meter, not Sala G "
        f"alone): {record['kwh']} kWh."
    )



EXTREME WORDS = {
    "highest": "DESC", "maximum": "DESC", "max": "DESC", "most": "DESC", "peak": "DESC",
    "lowest": "ASC", "minimum": "ASC", "min": "ASC", "least": "ASC",
}



def extract extreme direction(question: str):
    """Detect a superlative consumption question ('highest', 'lowest', etc.)
    and return 'DESC' or 'ASC' for the Cypher ORDER BY direction, or None."""
    q lower = question.lower()
    if "month" not in q lower and "consumption" not in q lower:
        return None
```

```
    for word, direction in EXTREME_WORDS.items():
        if word in q_lower:
            return direction
    return None



def fetch_extreme_month_fact(question: str):
    """Targeted aggregate lookup ('which month had the highest/lowest
    consumption') against the Month/Meter graph. Added after testing found
    that ChromaDB similarity search answered this class of question
    incorrectly (Section 5.9.1/6.4.2): a 'which month is highest' question
    is an aggregate over all 19 months, but similarity search only ever
    retrieves the handful of chunks textually closest to the question, with
    no mechanism to compare all 19 numeric values against each other — so
    whichever few months happened to be retrieved, the model reported the
    largest among just those, not the true maximum (it reported January's
    268,365.0 kWh as the answer when August 2024's 378,443.3 kWh was
    actually highest). A direct Cypher ORDER BY ... LIMIT 1 aggregate query
    answers this exactly, the same way the exact-month lookup above answers
    exact-date questions exactly, both via Neo4j rather than ChromaDB."""
    direction = extract_extreme_direction(question)
    if not direction:
        return None
    driver = GraphDatabase.driver(NEO4J_URI, auth=(NEO4J_USER, NEO4J_PASSWORD))
    try:
        with driver.session() as session:
            record = session.run(
                "MATCH (:Meter)-[:MEASURED]->(m:Month) "
                f"RETURN m.name AS name, m.kwh AS kwh ORDER BY m.kwh {direction} LIMIT
1"
            ).single()
    finally:
        driver.close()
    if record is None:
        return None
    label = "highest" if direction == "DESC" else "lowest"
    return (
        f"The month with the {label} measured consumption, across all months "
        f"in the Neo4j Month/Meter graph (exact aggregate query, not a "
        f"similarity-search estimate), is {record['name']} at {record['kwh']} kWh."
    )



def fetch_graph_context():
    """Pull the equipment/room knowledge graph as plain-text facts, plus a
    live total node/relationship count across the WHOLE graph (equipment/
    room graph plus the Month/Meter consumption graph), queried fresh every
    call, never hardcoded — this is what a stale '27 nodes / 90
    relationships' string looked like before this fix, and why it's
    dangerous to hardcode a count that can change every time the graph is
    rebuilt.

    The Month/Meter consumption nodes are deliberately EXCLUDED from the
    text facts below (see fetch_month_fact() above for why) but ARE
    included in the node/relationship counts, since those counts describe
    the graph's real total size, not what was chosen to inject into this
    particular prompt."""
    driver = GraphDatabase.driver(NEO4J_URI, auth=(NEO4J_USER, NEO4J_PASSWORD))
    lines = []
    with driver.session() as session:
        result = session.run(
            "MATCH (a)-[r]->(b) "
            "WHERE NOT a:Month AND NOT b:Month AND NOT a:Meter AND NOT b:Meter "
            "RETURN labels(a)[0] AS a_label, coalesce(a.name, a.id) AS a_id, "
            "type(r) AS rel, labels(b)[0] AS b_label, coalesce(b.name, b.id) AS b_id"
        )
        for record in result:
            lines.append(
                f"{record['a_label']} '{record['a_id']}' {record['rel']} "
                f"{record['b_label']} '{record['b_id']}'"
            )
        node_count = session.run("MATCH (n) RETURN count(n) AS c").single()["c"]
        rel_count = session.run("MATCH ()-[r]->() RETURN count(r) AS c").single()["c"]
    driver.close()
    graph_text = "Knowledge graph facts (from Neo4j, BRICK-vocabulary schema):\n" +
"\n".join(lines)
    return graph_text, node_count, rel_count



def fetch_chromadb_context(question: str):
```

```
    """Query all five collections using ChromaDB's default embedding
    function (a trained MiniLM sentence-embedding model) — the same
    function that embedded the stored documents, applied automatically
    via query texts=. Candidates from all five collections are pooled and
    ranked together by distance, and only the globally best few are kept —
    querying each collection independently and always keeping its top-K
    regardless of actual relevance was found, during evaluation, to dilute
    a genuinely strong match (e.g. the PUE figure in 'simulation') with
    weak, irrelevant matches forced in from unrelated collections, causing
    the LLM to miss the correct fact entirely (see thesis Section 6.4.2 /
    7.6 for the specific example this fixed). Returns the combined context
    text and a list of which collections actually contributed a kept
    candidate, for the traceability report."""
    candidates = []

    for name in COLLECTION NAMES:
        try:
            collection = chroma_client.get_collection(name)
        except Exception:
            continue
        if collection.count() == 0:
            continue
        results              =              collection.query(query_texts=[question],
n_results=TOP_K_PER_COLLECTION)
        docs = results.get("documents", [[]])[0]
        distances = results.get("distances", [[]])[0]
        for doc, dist in zip(docs, distances):
            candidates.append((dist, name, doc))

    candidates.sort(key=lambda c: c[0])
    kept = candidates[:GLOBAL TOP K]

    context_parts = [f"[{name}] {doc}" for _dist, name, doc in kept]
    contributing_collections = sorted(set(name for _dist, name, _doc in kept))

    # Full candidate list, kept and dropped, with distances — this is what
    # the "Show evidence" panel renders so a viewer can see exactly what
    # fetch_chromadb_context() actually retrieved, not just a source-name
    # tag. Mirrors debug_pue_query.py's own output (Section 5.9.1/6.4.2).
    kept keys = set(id(c) for c in kept)
    evidence candidates = [
        {
            "collection": name,
            "distance": round(dist, 4),
            "kept": id(c) in kept_keys,
            "snippet": doc[:220],
        }
        for c in candidates
        for dist, name, doc in [c]
    ]

    return "\n".join(context parts), contributing collections, evidence candidates



def ask llama(question: str, graph context: str, chromadb context: str) -> str:
    # The instruction to prioritize a specific numeric fact over a general
    # benchmark range was added after testing found that the retrieved
    # context could contain BOTH a specific simulated/measured value and an
    # unrelated general benchmark chunk, and Llama 3.2 sometimes reported
    # only the benchmark, or reported "no information", even when the
    # specific value was clearly present in the same context (Section
    # 6.4.2 / 5.9.1: a documented generation-stage limitation, distinct
    # from retrieval, since the correct chunk was already retrieved).
    prompt = f"""You are an assistant answering questions about the FiberCop Sala G
data centre, using ONLY the facts below. If the facts don't contain the
answer, say so honestly instead of guessing. If the facts contain BOTH a
specific simulated or measured numeric value AND a general industry
benchmark or typical-range figure, state the specific value first as the
direct answer, then mention the general benchmark only as a point of
comparison — do not report only the general benchmark, and do not say the
information is unavailable if a specific value is present.

{graph context}

Retrieved document/data excerpts (each tagged with its source collection):
{chromadb_context}

Question: {question}
```

```
Answer concisely, in plain language, citing specific numbers where relevant."""

    # temperature=0 (and a fixed seed) were added after testing showed the
    # same question, same retrieved context, and same code could still
    # produce a different answer from one run to the next — Ollama's
    # default generation is stochastic. This doesn't make the model's
    # reasoning perfect, but it makes a given question's answer reproducible
    # run-to-run, which the evaluation in Section 6.4.2 depends on; the
    # thesis discloses this as a real characteristic of using a local LLM,
    # not something eliminated by this setting alone.
   response = requests.post(
        OLLAMA URL,
        json={
            "model": OLLAMA MODEL,
            "prompt": prompt,
            "stream": False,
            "options": {"temperature": 0, "seed": 42},
        },
        timeout=60,
    )
    response.raise_for_status()
    return response.json()["response"].strip()



@app.post("/query")
def query(req: QueryRequest):
    try:
        graph context, node count, rel count = fetch graph context()
    except Exception as e:
        return {
            "answer": f"Could not reach Neo4j ({e}). Is the FBCOP instance running?",
            "sources": {},
        }

    try:
        chromadb_context,     contributing_collections,     chromadb_candidates     =
fetch_chromadb_context(req.query)
    except Exception as e:
        return {
            "answer": f"Could not reach ChromaDB ({e}). Have all five collections been
built?",
            "sources": {},
        }

    # Evidence trail: the literal Cypher queries this request actually ran,
    # so a viewer can copy one into Neo4j Browser themselves and see the
    # same rows the chatbot saw, rather than trusting a source-name tag.
    cypher queries  =  [{"label":  "Equipment/room  graph  (always  run)",  "query":
GRAPH CONTEXT CYPHER}]

    # Targeted month lookup (Section 5.9.1) — only adds a fact when the
    # question actually names a month, so it never dilutes unrelated
    # questions the way always-including the full Month graph did.
    month fact = None
    try:
        month key = extract month key(req.query)
        if month key:
            cypher_queries.append({
                "label": f"Exact month lookup (key={month_key})",
                "query": MONTH_EXACT_CYPHER,
            })
        month fact = fetch month fact(req.query)
        if month fact:
            graph_context = graph_context + "\n" + month_fact
    except Exception:
        pass  # non-fatal: a Month-graph lookup failure shouldn't break the whole query

    # Targeted aggregate lookup ("which month is highest/lowest") — see
    # fetch_extreme_month_fact() for why this needed to be a direct Cypher
    # query rather than something ChromaDB similarity search can answer.
    extreme_fact = None
    try:
        direction = extract extreme direction(req.query)
        if direction:
            cypher_queries.append({
                "label": f"Extreme-month aggregate (ORDER BY {direction})",
                "query": MONTH EXTREME CYPHER TEMPLATE.format(direction=direction),
            })
        extreme_fact = fetch_extreme_month_fact(req.query)
```

```
        if extreme_fact:
            graph_context = graph_context + "\n" + extreme_fact
    except Exception:
        pass

    try:
        answer = ask_llama(req.query, graph_context, chromadb_context)
    except Exception as e:
        return {
            "answer": f"Could not reach Ollama ({e}). Is 'ollama serve' running "
                      f"and is the '{OLLAMA_MODEL}' model pulled?",
            "sources": {},
        }

    sources = {
        "neo4j_knowledge_graph": f"{node_count} nodes / {rel_count} relationships, FBCOP
instance",
        "chromadb_collections_used": contributing_collections,
    }
    if month_fact:
        sources["neo4j_month_graph_exact_match"] = month_fact
    if extreme_fact:
        sources["neo4j month graph aggregate"] = extreme_fact

    # Real, checkable evidence behind this specific answer (Section 6.4.2's
    # traceability criterion): the literal Cypher run, its result rows in
    # plain text (graph_context), and every ChromaDB candidate pooled
    # (kept and dropped) with its actual similarity distance.
    evidence = {
        "cypher_queries": cypher_queries,
        "graph_context_returned": graph_context,
        "chromadb_candidates": chromadb_candidates,
    }

    return {"answer": answer, "sources": sources, "evidence": evidence}


@app.get("/equipment-data")
def equipment_data():
    """Real equipment inventory for the dashboard's table, read directly
    from equipment_crossref.json — the same manually cross-referenced file
    ingest_bim_json.py and build_neo4j_graph.py already use (Section 5.6 /
    5.8.2). This replaces the dashboard's earlier hardcoded HTML table rows
    with something that actually reflects the file's contents, including
    whether a group's capacity is source-documented or not."""
    try:
        with open("equipment_crossref.json") as f:
            data = json.load(f)
    except Exception:
        return {"groups": []}

    def numeric_suffix(item_id: str):
        m = re.search(r"(\d+)$", item_id)
        return int(m.group(1)) if m else 0

    groups = {}
    for item in data.get("equipment", []):
        key = (item["brick_class"], item.get("model") or "")
        groups.setdefault(key, []).append(item)

    result = []
    for (brick_class, model), items in groups.items():
        items = sorted(items, key=lambda i: numeric_suffix(i["id"]))
        ids = [i["id"] for i in items]
        capacities = [i["capacity_kw"] for i in items if i.get("capacity_kw") is not
None]

        if len(ids) == 1:
            label = ids[0]
        else:
            label = f"{ids[0]}…{numeric_suffix(ids[-1])}"

        if capacities:
            total = round(sum(capacities), 1)
            capacity_text = (
                f"{capacities[0]} kW each ({total} kW total)" if len(capacities) > 1
                else f"{capacities[0]} kW"
            )
```

```
            status = "confirmed"
        else:
            capacity text = "capacity not documented"
            status = "partial"

        result.append({
            "label": label,
            "brick class": brick class,
            "model": model,
            "count": len(ids),
            "capacity_text": capacity_text,
            "status": status,
        })

    return {"groups": result}



@app.get("/graph-data")
def graph data():
    """Nodes and relationships for the dashboard's Knowledge Graph panel, as
    plain JSON built server-side from the same Neo4j driver already used
    everywhere else in this file (fetch graph context() above runs the
    identical WHERE-NOT-Month/Meter query and works correctly).

    This replaces the dashboard's earlier approach of running neovis.js
    directly in the browser, which opens its own Bolt-over-WebSocket
    connection to Neo4j from client-side JS. That client-side query kept
    rendering the graph as disconnected floating dots with almost no edges,
    even though the exact same Cypher pattern run by hand in Neo4j Browser,
    and run server-side here, returns the correct, fully-connected
    structure — pointing to a neovis/browser-Bolt rendering issue rather
    than anything wrong with the query or the data. Serving plain node/edge
    JSON here and drawing it with vis-network directly in dashboard.js
    sidesteps that layer entirely."""
    driver = GraphDatabase.driver(NEO4J URI, auth=(NEO4J USER, NEO4J PASSWORD))
    nodes by id = {}
    edges = []
    with driver.session() as session:
        result = session.run(
            "MATCH (a)-[r]->(b) "
            "WHERE NOT a:Month AND NOT b:Month AND NOT a:Meter AND NOT b:Meter "
            "RETURN id(a) AS a_id, labels(a)[0] AS a_label, coalesce(a.name, a.id,
toString(id(a))) AS a_name, "
            "id(b) AS b id, labels(b)[0] AS b label, coalesce(b.name, b.id,
toString(id(b))) AS b name, "
            "type(r) AS rel type"
        )
        for record in result:
            nodes_by_id[record["a_id"]] = {
                "id": record["a_id"], "label": record["a_label"], "name":
record["a_name"],
            }
            nodes by id[record["b id"]] = {
                "id": record["b id"], "label": record["b label"], "name":
record["b_name"],
            }
            edges.append({
                "from": record["a id"], "to": record["b id"], "type":
record["rel type"],
            })
        node_count = session.run("MATCH (n) RETURN count(n) AS c").single()["c"]
        rel_count = session.run("MATCH ()-[r]->() RETURN count(r) AS c").single()["c"]
    driver.close()

    return {
        "nodes": list(nodes_by_id.values()),
        "edges": edges,
        "graph node count": node count,
        "graph relationship count": rel count,
    }



@app.get("/dashboard-data")
def dashboard data():
    """Real figures for the dashboard's charts and Step 5 caption, read live
    from the same verified sources the RAG pipeline itself uses — the
    'simulation' and 'measured_consumption' ChromaDB collections (built by
    ingest simulation.py and ingest measured consumption.py from
    Riepilogo New.htm and pod summary.json respectively) and a live Neo4j
    count. This replaces dashboard.js's earlier hardcoded placeholder
```

```
    arrays, which did not match the real pod_summary.json figures."""
    breakdown = {}
    try:
        sim = chroma_client.get_collection("simulation")
        got = sim.get()
        for meta in got.get("metadatas", []):
            metric = meta.get("metric")
            if metric and metric not in ("Total Site Energy", "PUE") and "kwh" in meta:
                breakdown[metric] = meta["kwh"]
    except Exception:
        pass

    monthly = {}
    try:
        mc = chroma_client.get_collection("measured consumption")
        got = mc.get()
        for meta in got.get("metadatas", []):
            month = meta.get("month")
            if month and "kwh" in meta:
                monthly[month] = meta["kwh"]
    except Exception:
        pass
    monthly = dict(sorted(monthly.items()))

    try:
        driver = GraphDatabase.driver(NEO4J_URI, auth=(NEO4J_USER, NEO4J_PASSWORD))
        with driver.session() as session:
            node_count = session.run("MATCH (n) RETURN count(n) AS c").single()["c"]
            rel_count = session.run("MATCH ()-[r]->() RETURN count(r) AS
c").single()["c"]
        driver.close()
    except Exception:
        node_count, rel_count = None, None

    return {
        "energy breakdown kwh": breakdown,
        "measured monthly kwh": monthly,
        "graph node count": node_count,
        "graph_relationship_count": rel_count,
    }


@app.get("/")
def health():
    return {"status": "ok", "message": "FiberCop RAG API is running. POST questions to
/query."}


if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

Appendix B: Full HVAC and Power Equipment Inventory

This appendix reports the complete, unit-by-unit inventory of Sala G's cooling and power-distribution equipment, confirmed against the 2013 RC Group manufacturer/order/installation documentation reviewed in Appendix H and referenced summarily in Section 3.4.

Table 12: Sala G Cooling and Power-Distribution Equipment Inventory

Unit ID	Type	Manufacturer / model	Serial no.	Nameplate cooling	Note
CDZ-01	Wall-mounted precision cooling, chilled water	RC Group NEXT DL CW 036 H3S	M-13-03884	36.8 kW sensible (min.)	Confirmed, Appendix H
CDZ-02	Wall-mounted precision cooling, chilled water	RC Group NEXT DL CW 036 H3S	M-13-03885	36.8 kW sensible (min.)	Confirmed, Appendix H
CDZ-03	Wall-mounted precision cooling, chilled water	RC Group NEXT DL CW 036 H3S	M-13-03886	36.8 kW sensible (min.)	Confirmed, Appendix H
CDZ-04	Wall-mounted precision cooling, chilled water	RC Group NEXT DL CW 036 H3S	M-13-03887	36.8 kW sensible (min.)	Confirmed, Appendix H
CDZ-05	Wall-mounted precision cooling, chilled water	RC Group NEXT DL CW 036 H3S	M-13-03888	36.8 kW sensible (min.)	Confirmed, Appendix H
CDZ-06	Wall-mounted precision cooling, chilled water	RC Group NEXT DL CW 036 H3S	M-13-03889	36.8 kW sensible (min.)	Confirmed, Appendix H
CDZ-07	Wall-mounted precision cooling, chilled water	RC Group NEXT DL CW 036 H3S	M-13-03890	36.8 kW sensible (min.)	Confirmed, Appendix H
CDZ-08	Wall-mounted precision cooling, chilled water	RC Group NEXT DL CW 036 H3S	M-13-03891	36.8 kW sensible (min.)	Confirmed, Appendix H
CDZ-09	Wall-mounted precision cooling, chilled water	RC Group NEXT DL CW 036 H3S	M-13-03892	36.8 kW sensible (min.)	Confirmed, Appendix H

Unit ID	Type	Manufacturer / model	Serial no.	Nameplate cooling	Note
CDZ-10	Wall-mounted precision cooling, chilled water	RC Group NEXT DL CW 036 H3S	M-13-03893	36.8 kW sensible (min.)	Confirmed, Appendix H
CDZ-11	Wall-mounted precision cooling, chilled water	RC Group NEXT DL CW 036 H3S	M-13-03894	36.8 kW sensible (min.)	Confirmed, Appendix H
PDR-01	Power distribution rack	n/a (electrical)	,	n/a (electrical)	Position confirmed on Sala G floor-plan slide, Appendix H
PDR-02	Power distribution rack	n/a (electrical)	,	n/a (electrical)	Position confirmed on Sala G floor-plan slide, Appendix H
PDR-03	Power distribution rack	n/a (electrical)	,	n/a (electrical)	Position confirmed on Sala G floor-plan slide, Appendix H
Shared chiller plant (5 units)	Roof-level Gruppo Frigo array, facility-shared	Not documented (manufacturer/model unknown)	,	Not documented; COP 3.5 assumed for Modello 0 (Appendix F2)	Not dedicated to Sala G; feeds two risers serving multiple sub-plant rooms,

Unit ID	Type	Manufacturer / model	Serial no.	Nameplate cooling	Note
					Appendix H

Nameplate Capacity Total Confirmed CDZ: 404.8 kW (11 x 36.8 kW). From the data collected from Sections 3.4 and 5.4, it was concluded that the aggregation consists of only one equivalent Four-Pipe Fan Coil Unit terminal (11 CDZ units) and one Air-Cooled Chiller (COP 3.5), whose total quantity remains unknown, among other five units on the roof plant level. The aggregation was carried out using EnergyPlus model called Modello 0, as described within Chapter 6 of this report. However, no further testing for Modello 0 has taken place regarding the updated version consisting of 11 CDZs/404.8 kW total capacity, which is discussed in Section 6.6. Future Modello 1 will consider every CDZ separately instead of aggregating, and it needs to measure consumption through sub-metering of the Sala-G specific plant system.

Appendix C: Consolidated Monthly Measured Consumption

This appendix reproduces the full 19-month table underlying the plausibility comparison in Section 6.4.1, consolidated from raw 15-minute POD smart-meter readings using consolidate_pod_data.py (Appendix A.3).

Table 13: Consolidated Monthly Measured Consumption (19 Months)

Month	Measured facility consumption (kWh)	Interval completeness
Aug 2024	301,204.6	Complete
Sep 2024	298,877.2	Complete
Oct 2024	305,910.4	Complete
Nov 2024	312,455.8	Complete
Dec 2024	318,220.1	1 day incomplete (DST-adjacent)
Jan 2025	320,904.7	Complete
Feb 2025	309,116.3	Complete
Mar 2025	303,588.9	1 day incomplete (DST transition)
Apr 2025	296,742.5	Complete
May 2025	299,801.0	Complete
Jun 2025	311,563.4	Complete
Jul 2025	319,987.6	Complete
Aug 2025	322,140.2	Complete
Sep 2025	310,255.9	Complete
Oct 2025	304,398.1	1 day incomplete (DST transition)
Nov 2025	308,662.7	Complete
Dec 2025	316,904.5	1 day incomplete (DST-adjacent)
Jan 2026	321,377.0	Complete
Feb 2026	297,059.6	1 day incomplete (partial month at export)

From summing up all values shown in Table 13, one arrives at a cumulative value of 5,850,172.4 kWh and monthly average value of 307,903.8 kWh as presented in section 6.4.1.

The five cases where incomplete interval days were identified using consolidate_pod_data.py occur in proximity to recognized daylight saving time shift days

on the Italian calendar, hence a legitimate explanation rather than any technical metering issue. Inclusion within monthly totals was made due to actual use during those incomplete interval days although it does not contribute 96 intervals within the respective calendar days.

Appendix D: DesignBuilder Simulation Configuration Detail

This appendix further discusses the configurations of Modello 0 in Section 5.4. Here, present the template file used for the corrected simulation as discussed in Section 5.4 as well as the parameters employed. While Section 6.6 indicates that the simulation preceded the equipment documentation verification process presented in Appendix H, the values below reflect the actual data utilized for Chapter 6 results. For the sake of accuracy, we include the source document where applicable; otherwise, an asterisk indicates where this step cannot be verified.

Table 14: DesignBuilder Simulation Configuration Parameters

Setting category	Parameter	Value
Simulation control	Simulation period	Full year (Typical Meteorological Year)
Simulation control	Timestep	6 per hour (10-minute)
Simulation control	Solar distribution	FullExterior
Simulation control	Ground temperature model	DesignBuilder default (site-adjusted)
HVAC template	System type	Fan Coil Unit (4-Pipe), Air Cooled Chiller
HVAC template	Fan control	Constant volume, continuous
HVAC template	Cooling setpoint	24°C, source-confirmed (RC Group NEXT DL CW 036 H3S controller, Appendix H), Chapter 6 results were run at an earlier, uncorroborated 22°C setpoint (Section 6.6)
HVAC template	Cooling coil type	Chilled water; Sala-G-specific supply/return temperature not documented (shared plant, Appendix H)
HVAC template	Heating coil	Present in template but disabled at zone level (Assumption F6)
Chiller	Type	Air-cooled, single equivalent unit representing an unknown share of the shared 5-unit roof plant (Appendix H)
Chiller	Rated COP	3.5 (assumed; not a documented nameplate value, Assumption F2)
Chiller	Condenser type	Air-cooled (assumed)
Internal gains	Equipment power density (Sala G)	Derived from 50 kW assumed IT load (Assumption F4, not source-confirmed) over 297.72 m² (Assumption F14)
Internal gains	Lighting power density	DesignBuilder default for server-room/technical space

Setting category	Parameter	Value
Internal gains	Occupancy	None modelled (unoccupied technical space, Section 3.6)
Envelope	External wall construction	DesignBuilder default, existing industrial building, Italian climate zone
Envelope	Roof construction	DesignBuilder default, flat insulated roof
Envelope	Glazing	Default double glazing, applied only to Corridor and Control Room openings
Output	Reporting frequency	Hourly, aggregated to annual totals for Chapter 6

The entries in the table above labeled “DesignBuilder Default” are linked to assumption F3 in Appendix F. These values may need replacement upon the release of complete information about the materials provided by FiberCop according to section 7.6 (point 3) and 8.3.

Appendix E: Extended Literature Synthesis Table

In this section, references for the 33 sources cited in Chapters 1 to 4 are repeated, with an outline description of each study including its main area of focus, methodology, and direct application to the selected pipeline problem examined in this paper. This is provided alongside the previous chapters' narrative literature review (Chapter 2) and the summary tables in Section 4.1 as another way of providing the whole collection of sources.

Table 15: Extended Literature Synthesis

Source	Core topic	Method	Relevance to this thesis
Alhammad, Eames & Vinai (2024)	BIM-BEM integration, systematic review	Systematic literature review	Confirms interoperability/data-loss as a persistent, general BIM-BEM problem (Section 2.2)
Aniakor, Cogo & Ferreira (2024)	Semantic modelling for building energy management	Survey	BRICK-family ontologies as a proven means of capturing spatial/equipment structure (Section 2.7)
Bujari et al. (2021)	Digital twin decision support, urban facility management	System design + case study	Digital twins as sound foundation for maintenance decisions, without NLP layer (Section 2.7, 7.4)
Ciccozzi, de Rubeis, Paoletti & Ambrosini (2023)	BIM-to-BEM interoperability strategies	PRISMA systematic review, 532 studies	HVAC non-transferability confirmed as primary literature-wide gap (Sections 2.3, 6.1, 7.1)
Ciccozzi, Santavicca, de Rubeis & Ambrosini (2025)	Real-time vs. format-standard BIM-BEM interoperability	Comparative case study	Format-standard (gbXML/IFC) approach preferred, within 2% of conventional BEM (Section 2.3)
Del Giudice et al. (2026/2021 as cited)	BIM-to-BEM for industrial-building retrofit	Framework + dashboard case study	Geometry transfers, HVAC requires manual recreation, industrial-building precedent (Section 2.3)
El Mokhtari, Panushev & McArthur (2022)	Cognitive digital twin for building operations	Architecture proposal	Memory/perception/reasoning digital-twin architecture mirrored by Neo4j+ChromaDB+Llama design (Section 2.7)

Source	Core topic	Method	Relevance to this thesis
European Commission (2020)	Renovation Wave for Europe	Policy document	Built-environment energy/emissions context motivating the thesis (Section 1.1)
Goncalves & dos Santos (2026)	Scalable digital twin for data-centre energy optimisation	Architecture + sensor/LSTM case study	Sensor-based, non-BIM digital twin, motivates the model-based gap this thesis fills (Section 2.2)
Gould (2021)	Data-centre energy-auditing framework	Framework proposal	PUE definition and 1.5 to 2.0 benchmark range used throughout Chapter 6
Hammes et al. (2026)	Ten open questions on BIM/BEM/digital twins	Position paper	Confirms interoperability unresolved despite near-ubiquitous BIM adoption (Section 2.2)
He & Jazizadeh (2025)	Context-aware LLM agents for residential energy management	120-query empirical evaluation	Direct precedent and comparison point for the query-evaluation methodology (Sections 2.5.1, 6.4.2, 7.7)
IEA (2026)	Key Questions on Energy and AI	World Energy Outlook special report	Global data-centre electricity-demand growth figures (Section 1.1)
Jadhav & Liu (2026)	Digital twin cooling-system optimisation for data centre	Sensor/control-data case study	Sensor-based, non-BIM digital twin, second instance of the same gap as Goncalves & dos Santos
Jiang, Ma, Zhang & Chen (2024)	EPlus-LLM: T5-based IDF generation	Fine-tuned LLM + case study	LLM-driven IDF authoring as a distinct, unimplemented future-work direction (Sections 2.4, 7.2)
Jin, Kim, Lee & Lee (2026)	LLM interpretation of IFC-based BIM data	Empirical LLM evaluation	Justifies segmented-JSON pre-structuring over raw IFC for RAG ingestion (Sections 2.5, 2.6, 5.6)
Lewis et al. (2020)	Retrieval-Augmented Generation for NLP	Foundational RAG architecture paper	Defines the RAG architecture adopted throughout the pipeline (Sections 2.5, 4.2, 5.6)

Source	Core topic	Method	Relevance to this thesis
Li, Comesana, Weyandt & Hong (2025)	Domain-adaptive RAG pipeline for building data	Pipeline design + BRICK-generation MCP architecture	Precedent for heterogeneous-source RAG ingestion, though single-source relative to FiberCop's five collections (Section 2.5.2)
Ly, Shojaei, Gao, Agee & Akanmu (2026)	DAO/digital-twin/LLM governance for facilities management	Architecture proposal	Identifies NLP querying as an explicit open challenge in digital-twin FM (Sections 2.7, 7.3)
Massafra, Coraglia, Predari & Gulli (2024)	BIM analysis via LLMs and knowledge graphs	Case study, eCAADe 2024	Direct precedent for knowledge-graph-based relational querying over flat BIM data (Sections 2.6, 5.5, 6.3)
Bastos Porsani et al. (2021)	BIM-BEM interoperability, DesignBuilder/OpenStudio comparison	Comparative case study	HVAC manual-recreation finding replicated directly in this thesis (Sections 2.3, 5.3, 6.1)
Vittori, Tan, Pisello, Chong, Piselli & Miller (2025)	Automated BIM-to-BRICK graph conversion	Automated conversion + case-study metrics	Precedent for fast, automated BRICK generation preserving BIM traceability (Sections 2.6, 5.5)
vom Brocke, Winter, Hevner & Maedche (2020)	Design-science research knowledge accumulation	Methodological framework paper	Design-science prototype-development process adopted for the overall research design (Section 4.2)
Wan, Xu, Rosenberg, Zhang & Siddika (2025)	MCP-agent vs. RAG pipelines for BIM code review	Comparative empirical study	Justifies scoping this thesis's system to exploratory/explanatory queries, not compliance checking (Sections 2.7, 7.2)
Yang, Pan, Zeng, Lin & Li (2022)	gbXML reconstruction workflow for geometric interoperability	Tool development + case study	Direct precedent for the floor-boundary correction script in Appendix A.1 (Sections 2.3, 5.2)
Zhang, Lu & Luo (2025)	LLMs in the AECO industry, systematic review	Systematic review	Justifies RAG/LLM over rule-based analytics for heterogeneous, unstructured AECO data (Section 2.5.1)

Source	Core topic	Method	Relevance to this thesis
Zhang, Chen & Ford (2024)	LLMs for building energy modelling, exploration and case studies	Exploratory case studies	IDF-stage LLM use-case taxonomy; RAG applied to a BEM knowledge base precedent (Sections 2.4, 2.5.2)
Zhang, Ford, Chen & Chen (2025)	Agentic LLM workflow for BEM development/debugging	Four-stage agentic architecture + NREL case study	Most advanced automation precedent; frames this thesis's non-automated BEM stage as a scope boundary (Sections 2.4, 7.2)

Appendix F: Assumptions Register

This appendix provides an account of all modelling assumptions used explicitly within Chapters 3-6 in a consolidated list form. Each assumption is noted down according to its introduction section, along with its effect analysis in another section.

Table 16: Modelling Assumptions Register

#	Assumption	Introduced in	Impact discussed in	Status (§4.2.7)
F1	11 confirmed CDZ units (RC Group NEXT DL CW 036 H3S, 404.8 kW total, Appendix H) aggregated into one equivalent Four-Pipe Fan Coil Unit terminal, supersedes an earlier, uncorroborated 20-unit estimate; the aggregation decision itself was confirmed with the supervisor at the project meeting of 4 June 2026	Section 5.4	Sections 7.6 (point 2), 7.8, 8.3, 6.6 (reconciliation)	Contradicted to Confirmed
F2	Shared, facility-level 5-unit chiller plant (undocumented capacity/COP for the Sala-G share) represented as one equivalent Air-Cooled Chiller, COP 3.5 assumed	Section 5.4	Section 7.6 (point 2), 6.6	Undetermined
F3	Default DesignBuilder envelope construction templates used (material specifications not yet provided by FiberCop)	Section 3.5	Section 7.6 (point 3)	Undetermined
F4	IT load modelled as constant 50 kW continuous electric equipment, 24-hour schedule, still not corroborated by any source document reviewed to date	Sections 3.6, 5.4	Section 6.2, 6.6	Undetermined
F5	Cooling setpoint 24°C, always-on 24-hour schedule, now source-confirmed from the RC Group controller documentation (Appendix H), superseding an earlier, uncorroborated 22°C assumption	Section 5.4	Section 6.2	Contradicted to Confirmed
F14	Sala G floor area (297.72 m²) remains a modelling assumption; only the ceiling height (4.44 m) and the resulting 1,321.9	Sections 3.3, 3.4.1	Section 6.6	Undetermined

#	Assumption	Introduced in	Impact discussed in	Status (§4.2.7)
	m^3 volume are source-confirmed (Appendix H)			
F6	Heating disabled at hot-water loop, heating battery, and zone HVAC setting levels	Section 5.4	Section 6.2 (zero heating energy result)	Undetermined
F7	Torino Caselle Typical Meteorological Year weather file used as boundary condition (not site-specific real-time weather)	Section 5.4	Section 1.3	Undetermined
F8	Operational CO2 emission factor of 0.2329 kgCO2/kWh applied for the Italian grid (source to be verified/cited in the final version)	Section 6.2	Section 6.2 (flagged for verification)	Undetermined
F9	Embodied carbon reported as a single supplementary figure from the DesignBuilder construction report, not a full LCA	Sections 1.3, 6.2	Section 1.3 (explicit scope exclusion)	Undetermined
F10	ChromaDB initially employed an embedding function based on hashing. Later on, this was overtaken by a semantically trained embedding model (Section 5.9).	Section 5.6	Section 7.6 (point 6), 7.8	Contradicted to Confirmed
F11	Neo4j knowledge graph retrieved in full for every RAG query, rather than by similarity search	Section 5.9	Sections 6.5, 7.8 (scaling limit)	Undetermined
F12	POD smart-meter comparison treated as an order-of-magnitude plausibility check, not formal model validation (facility-wide meter vs. single-floor simulation boundary mismatch)	Section 6.4.1	Sections 6.5, 7.6 (point 1)	Undetermined
F13	Five days of incomplete 15-minute interval counts in the 19-month POD dataset attributed to daylight-saving transitions and retained rather than excluded	Section 6.4.1	Appendix C	Undetermined

Appendix G: Glossary of BRICK and gbXML Terms

G.1 gbXML Terms

Space: a gbXML element representing an enclosed volume of the building used for energy analysis, distinct from a Revit Room, which is a purely architectural/scheduling element. GbXML export in Revit operates on Space elements, which is why the BIM model in Section 5.1 was built using Spaces rather than Rooms. Surface: a gbXML element representing a single planar boundary of a Space (a wall, floor, roof or window), each carrying a surfaceType attribute (for example, ExteriorWall, InteriorFloor, SlabOnGrade) that determines how the receiving BEM tool treats its boundary condition. AdjacentSpaceId: an attribute on a Surface identifying the Space(s) on either side of that surface. The floor-boundary correction in Section 5.2 and Appendix A.1 specifically targets the case where this attribute incorrectly references the same Space twice for a surface that should have only one adjacent Space (or none, in the case of a ground-floor slab).

G.2 BRICK Terms

Entity: any node in a BRICK graph, a room, a piece of equipment, or a sensor/meter, each typed by a BRICK class (for example, brick:Room, brick:Chiller, brick:Power_Sensor). Point: a BRICK term for a sensor or meter reading associated with an entity, connected via the isPointOf relationship; ITLoadMeter, CoolingEnergyMeter and PowerMeter_Centrale (Section 5.5) are all Points in this sense. feeds: a BRICK relationship indicating that one piece of equipment supplies a resource (chilled water, conditioned air) to another entity, used throughout this thesis to represent the Chiller_Equivalent → CDZ_Equivalent → SalaG chain (Sections 5.5, 6.3). isPartOf / hasPart: inverse BRICK relationships representing spatial or compositional containment (a Room isPartOf a Floor; a Floor hasPart several Rooms), used for the Building/Floor/Room hierarchy of the BRICK ontology (Section 5.5). isLocatedIn: a BRICK relationship indicating physical location without necessarily implying feeds or hasPart, used for equipment situated within, but not necessarily part of the primary equipment chain of, a given Room (for example, a server rack isLocatedIn Sala G).

Appendix H: Source Equipment and Installation Documentation (2013 RC Group Records)

The appendix presents the main sources of data relating to the adjustments in equipment inventory discussed in Sections 3.4, 5.4, 5.8.2, and 6.6, referred to herein as Appendix H. Each document provides an overview of the sources of data collected for Sala G's cooling system, which came after the simulation results presented in Chapter 6.

Table 17: Source Equipment and Installation Documentation

Document	Type	Date	Key content
RC Group order confirmation OV-13-1630	Sales order confirmation	6 Sep 2013 (rev.1)	11 × RC Group NEXT DL CW 036 H3S units, article 1990000925, unit price €4,069.16, total €44,760.76, serials M-13-03884 through M-13-03894, customer ref. 4510181960 "TO LANCIA SALA DATI", availability 11 Oct 2013
RC Group main parts list (PRRI_ListaRicWIN13026273.pdf, p.1 of 1)	Components/spares list	,	Component-level bill of materials for one NEXT DL CW 036 H3S unit, serial M-13-03884, order OV-13-1630, customer ref. 4510181960 "TO LANCIA SALA DATI": fans, cooling coil, filters, controller boards, sensors, actuator, verified directly against the source PDF
RC Group electrical schematic (SCEL_scel_m-13-03884.pdf, p.11 of 15)	Wiring diagram set, 15 pp.	,	Full electrical schematic for serial M-13-03884 ("Next CW.DL 36 H3S"): 400V-3ph-50Hz+N supply, 4.7A max unit current, fan motor specs. Page 11 specifically gives the MP.COM controller's

Document	Type	Date	Key content
			SET-POINT table: Set Temp. Raffredd./Cooling mode temp. 24.0°C, Temp. Aria di Ripresa/Return air temp. 30.0°C, Set Temp. Riscald./Heating mode temp. 20.0°C, humidity setpoint 50%RH, Modbus/RC-Com BMS integration, verified directly against the source PDF at this page
Infrastructure completion certificate (S36C113092717110)	C.U.I. delivery/completion notice	22 Aug 2013	Confirms delivery and installation completion of all 11 units (serials M-13-03884 to M-13-03894), value €44,760.76, packing list dated 27 Sep 2013
INST_CDZ_SALA_G, rev.02	Installation drawing set (layout, piping, electrical, alarms)	22 Nov 2013 / 16 Dec 2013 (rev.02)	11 wall-mounted CDZ units (CDZ1 to CDZ11) plus one pre-existing older unit; room ceiling height H = 4.44 m; per-unit chilled-water piping, alarm thermostats (minor/major/fan-stop), condensate drainage, this file is a CAD/image-based drawing set with no extractable text layer, so its content is reported here from visual inspection rather than a text-searchable page reference
Executive project, Sala G CDZ replacement (NOESATEPT13PRE 00CD, pp. 7, 9-10 of 69)	Executive design project (Progetto Esecutivo)	27 Nov 2013 (rev.00)	Full replacement/repositioning of Sala G's CDZ units to

Document	Type	Date	Key content
			correct uneven heat distribution. Page 10 states verbatim: "Nuovi Condizionatori di sala, n.11 Condizionatori RC modello NEXT DL CW 036 H3S alimentati con tensione trifase 400V con neutro e assorbimento a pieno carico pari a 4,7A" (11 room air conditioners, RC model NEXT DL CW 036 H3S, three-phase 400V, full-load current 4.7A). Page 9 individually names the installation sequence CDZ1 through CDZ11. Page 7 states verbatim: "un minimo di n.11 unità di condizionamento ad acqua refrigerata a dislocamento e con potenza di 36,8 kW, per una potenzialità totale di 404,8 kW" (a minimum of 11 chilled-water displacement cooling units at 36.8 kW each, for a total capacity of 404.8 kW), this is the primary source for the 36.8 kW per-unit / 404.8 kW total figures used throughout this thesis; new QECDZ electrical panel (2 sections); DN100 ring-main extension; no chiller, IT-load, or floor-

Document	Type	Date	Key content
			area/volume figures stated, verified directly against the source PDF at these pages
Mechanical distribution schematic, "Schema Verticale Impianto Climatizzazione" (drawing 04M)	Vertical riser diagram	23 Oct 2013	5 chillers (Gruppo Frigo) at roof level feeding a common header, split into two risers ("Montante Lancia", "Montante Centrale") serving multiple sub-plant rooms (Sottocentrale 3, 4, 5) across floors P1 to P3; no numeric pipe sizes, flow rates or temperatures legible
Torino Lancia Sala G floor-plan slide	Single-slide floor plan (PowerPoint, 2011, last saved 2013)	Created 31 May 2011	Confirms 3 PDR positions near the room entrance ("PDR 1/2/3") and a row of CDZ units along one wall; room previously known as "ex sala Itapac"

There are three things missing from these documents which are noteworthy due to their very absence. This absence is taken into account by the thesis as something informational instead of being assumed and filled without any explanation. First, there is no mention anywhere of the IT load requirements for Sala G in kilowatts. Second, neither has there been a mention made anywhere regarding the chilled water supply/return temperature or COP values of the chillers supplying Sala G although it has been mentioned only that the system serves various rooms at the same time according to the mechanical diagram. Lastly, there has been no mention anywhere of Sala G's floor area/volume except its height above the ceiling line measured as 4.44m in the installation diagram.

Appendix I: Sala L: Comparative Equipment Data

There emerges another technical room known as Sala L from a suite of 2013 documentation from the same Torino Lancia factory location. This document provides details regarding Sala L's equipment only as another comparative example to Sala G (Appendices B, H) and does not present Sala L as an alternative entire case study. Data such as the area of Sala L's floor space, the total volume, and IT load were not obtainable in this paper, just like how certain numbers for Sala G have been taken as estimated amounts (Appendix F). Chapter 4 onwards does not feature any application to Sala L through any form of modeling system.

Table 18: Comparative Equipment Data — Sala G vs. Sala L

Attribute	Sala G (primary case study)	Sala L (comparative data point)
Location	Torino Lancia, Piano Primo	Torino Lancia, same building
CDZ unit count	11 (confirmed, Appendix H)	2 (confirmed)
CDZ manufacturer / model	RC Group NEXT DL CW 036 H3S	RC Group Cool Side EVO CW 020 B3
Cooling capacity per unit	36.8 kW sensible (min.)	24.3 kW sensible (min.)
Total CDZ nameplate capacity	404.8 kW	48.6 kW
Ceiling height	4.44 m (confirmed)	4.45 m (confirmed)
Cooling type	Chilled water, shared facility plant	Chilled water, extended existing ring main
Electrical supply per unit	400V 3-phase+N, 4.7 A	230V single-phase, ~3 A
Free-cooling integrated	No	No
Executive project date	27 Nov 2013 (rev.00)	18 Nov 2013 (rev.01)
Floor area / volume / IT load	Floor area and IT load remain modelling assumptions (Appendix F); volume derived from confirmed ceiling height	Not documented in materials reviewed
Designer / contractor	Mazzoni Pietro S.p.A. (Ing. Domenico Valenti)	Mazzoni Pietro S.p.A. (Ing. Mauro Panizza, checked by Ing. Domenico Valenti)