

a.a. 2024/2025



UNIVERSITÀ
DI CAMERINO

Graphitect: BIM Data Integration and Visualization with Neo4j

Diego Marinangeli

Professor:

Massimo Callisto De Donato

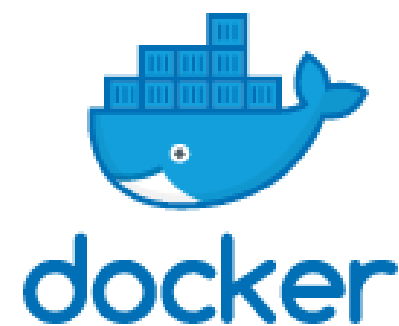


Project description / Objectives

- 1. Develop a Python-based tool to import and export Industry Foundation Classes (IFC) files into a Neo4j graph database, enabling advanced analysis and visualization of Building Information Models (BIM).**
- 2. Graph analysis using the Neo4j GDS library (Graph Data Science).**
- 3. Creation and connection of a custom IoT sensor into the BIM graph.**
- 4. Export functionality to allow reuse and distribution of graph data.**

 <https://github.com/Diego-m4i/TBDMProject>

Methodologies and Technologies



neo4j/graph-data-
science



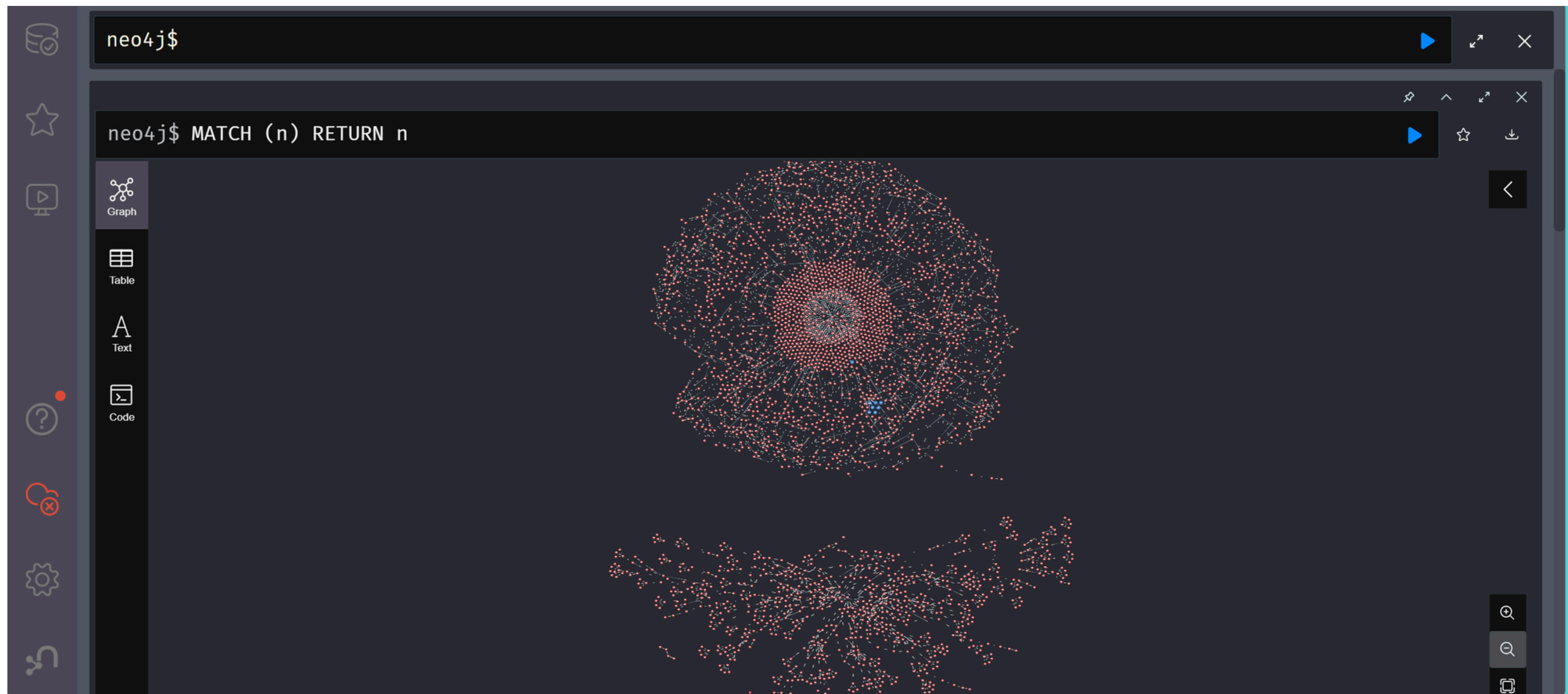
Technical Implementation - Graph Deployment and Access

1. Deploy a Neo4j instance using Docker Compose;
2. Start the Neo4j Database and input Neo4j database
 - username → neo4j
 - password → diegodiego
3. Access the database using Neo4j Database with the url:
<http://localhost:7474/browser/>
4. Project the graph for analysis

```
1 CALL gvds.graph.project(  
2   'IfcOpenHouseGraph', // Nome del grafo  
3   '*', // Tutti i nodi  
4   '*' // Tutte le relazioni  
5   );
```

Achieved Results

A complete pipeline for importing and visualizing IFC data in Neo4j.



Achieved Results

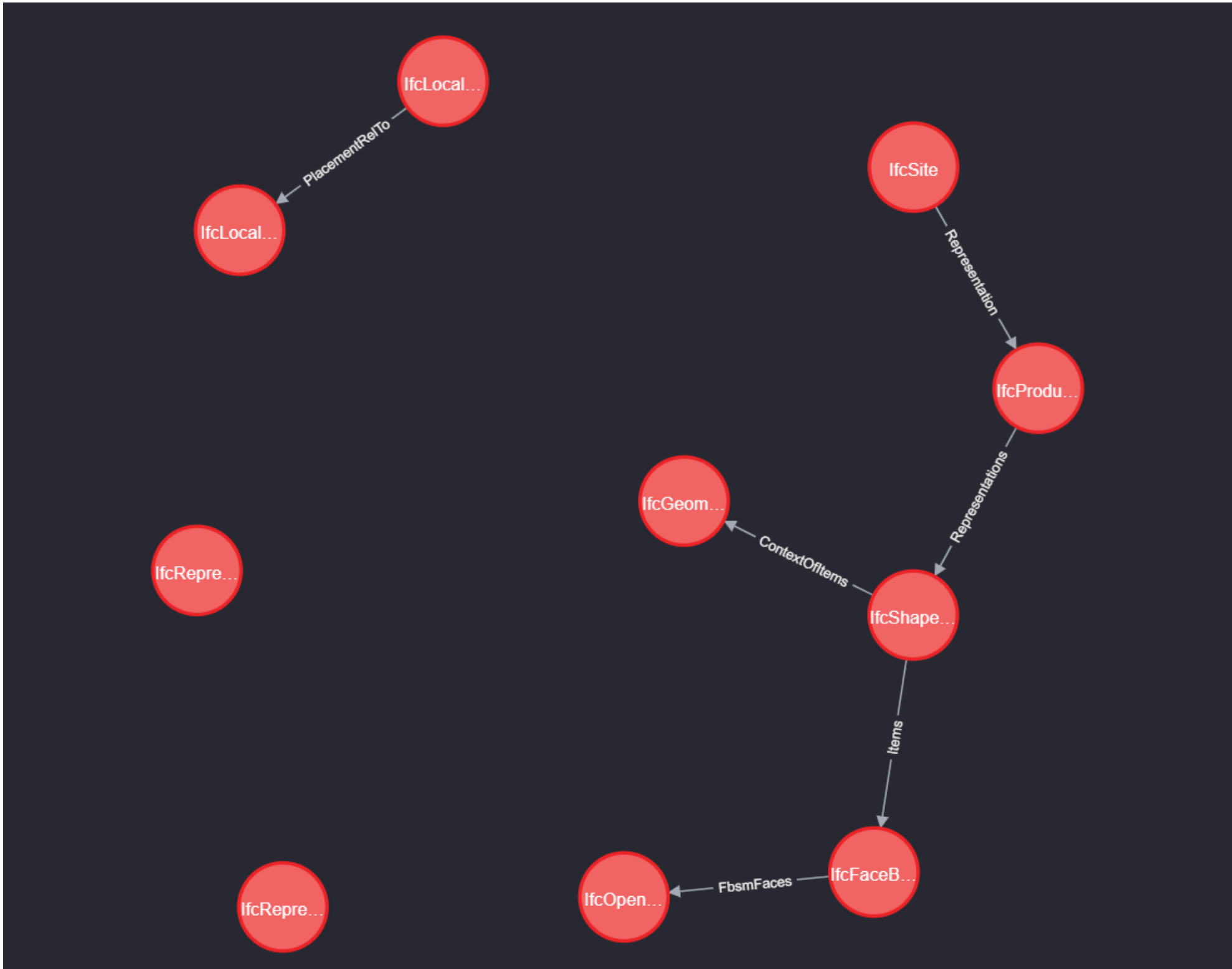
Graph centrality analysis

- Definition
- gds library
- class CentralityAnalyzer.py

```
20 def get_centrality_data(self, limit=10):
21     """
22     Calcola la centralità di intermediazione (Betweenness Centrality) e restituisce i nodi
23     con i punteggi di centralità più elevati.
24
25     :param limit: Numero massimo di nodi da restituire (default 10)
26     :return: Pandas DataFrame con i nodi e i loro punteggi di centralità
27     """
28     query = f"""
29     CALL gds.betweenness.stream('{self.graph_name}')
30     YIELD nodeId, score
31     MATCH (n) WHERE id(n) = nodeId
32     RETURN n.elementId AS id, n.ClassName AS class, score AS centrality
33     ORDER BY score DESC
34     LIMIT {limit};
35     """
36
37     with self.driver.session() as session:
38         result = session.run(query)
39         # Restituisci i risultati come un DataFrame
40         return pd.DataFrame([dict(record) for record in result])
41
```

```
Received notification from DBMS server: {severity: WARNING} {code: Ne
    id          class  centrality
0  None      IfcOpenShell    14119.0
1  None  IfcFaceBasedSurfaceModel    12108.0
2  None      IfcShapeRepresentation    8094.0
3  None  IfcProductDefinitionShape    6072.0
4  None      IfcSite          4058.0
5  None  IfcLocalPlacement        1600.0
6  None  IfcLocalPlacement        1057.0
7  None  IfcGeometricRepresentationContext    855.0
8  None      IfcRepresentationMap    825.0
9  None      IfcRepresentationMap    825.0
Received notification from DBMS server: {severity: WARNING} {code: Ne
Data esportati in ../output/centrality_results.csv
```


Achieved Results



class	centrality
"IfcOpenShell"	14118.99999
"IfcFaceBasedSurfaceModel"	12107.99999
"IfcShapeRepresentation"	8093.99999
"IfcProductDefinitionShape"	6071.99999
"IfcSite"	4057.99999
"IfcLocalPlacement"	1600.0
"IfcLocalPlacement"	1057.0
"IfcGeometricRepresentationContext"	855.0
"IfcRepresentationMap"	825.0

Achieved Results

Dynamic addition of a custom IoT sensor.

```
def add_iot_to_ifc_node(graph, ifc_file_path, target_node_id, iot_device):  
    """  
    Aggiunge un dispositivo IoT a un nodo IfcNode nel grafo Neo4j.  
  
    :param graph: Connessione al database Neo4j  
    :param ifc_file_path: Percorso del file IFC (non utilizzato nel codice corrente)  
    :param target_node_id: ID del nodo IfcNode a cui collegare il dispositivo IoT  
    :param iot_device: Oggetto IoTDevice da aggiungere al nodo IfcNode  
    """  
    print(f"Lettura file IFC: {ifc_file_path}")  
    f = ifcopenshell.open(ifc_file_path)  
  
    print(f"Verifica nodo IfcNode con ID: {target_node_id} nel grafo")  
    node_exists = graph.evaluate("MATCH (n:IfcNode {nid: $nid}) RETURN n", nid=target_node_id)  
  
    if not node_exists:  
        print(f"ERRORE: Il nodo IfcNode con ID {target_node_id} non esiste in Neo4j.")  
        return # Esce se il nodo non esiste  
  
    # Creazione del nodo IoTDevice  
    iot_node = iot_device.create_node(graph)  
  
    print(f"Collegamento del dispositivo IoT {iot_device.device_id} al nodo IfcNode {target_node_id}")  
  
    # Crea la relazione tra IfcNode e IoTDevice  
    query = """  
    MATCH (n:IfcNode {nid: $node_id}), (iot:IoTDevice {device_id: $device_id})  
    CREATE (n)-[:HAS_IOT_DEVICE]->(iot)  
    """  
    graph.run(query, node_id=target_node_id, device_id=iot_device.device_id)  
  
    print(f"✅ Dispositivo IoT {iot_device.device_id} collegato con successo a IfcNode {target_node_id}!")
```

```
C:\Users\marin\UNIVERSITÀ\TBDM\IFC-Neo4j-converter\venv\Scripts  
Lettura file IFC: ../ifc_files/IfcOpenHouse_original.ifc  
Verifica nodo IfcNode con ID: 2319 nel grafo  
Creazione del nodo IoTDevice con ID: MK-03  
Collegamento del dispositivo IoT MK-03 al nodo IfcNode 2319  
✅ Dispositivo IoT MK-03 collegato con successo a IfcNode 2319
```


```
Process finished with exit code 0
```


Achieved Results



Achieved Results

An export feature for saving or sharing the graph.

2856	# 2850=IFCARTESIANPOINT(840.0,5.0);	
2857	# 2851=IFCARTESIANPOINT(-840.0,5.0);	
2858	# 2852=IFCDIRECTION(0.0,0.0,1.0);	
2859	# 2853=IFCDIRECTION(1.0,0.0,0.0);	
2860	# 2854=IFCDIRECTION(0.0,0.0,1.0);	
2861	# 2855=IFCARTESIANPOINT(0.0,0.0);	
2862	# 2856=IFCELEMENT(\$,\$,'IfcAxis2Placement3D');	
2863	# 2857=IFCPOLYLINE();	
2864	# 2858=IFCELEMENT(\$,\$,'IfcArbitraryClosedProfileDef');	
2865	# 2859=IFCEXTRUDEDAREASOLID();	
2866	# 2860=IFCDIRECTION(1.0,0.0,0.0);	
2867	# 2861=IFCDIRECTION(0.0,0.0,1.0);	
2868	# 2862=IFCARTESIANPOINT(930.0,45.0);	
2869	# 2863=IFCELEMENT(\$,\$,'IfcAxis2Placement3D');	
2870	# 2864=IFCELEMENT(\$,\$,'IfcLocalPlacement');	
2871	# 2865=IFCELEMENT(\$,\$,'IfcPlate');	
2872	# 2866=IFCELEMENT(\$,\$,'IfcColourRgb');	
2873	# 2867=IFCELEMENT(\$,\$,'IfcSurfaceStyleRendering');	
2874	# 2868=IFCELEMENT(\$,\$,'IfcSurfaceStyle');	
2875	# 2869=IFCELEMENT(\$,\$,'IfcPresentationStyleAssignment');	
2876	# 2870=IFCELEMENT(\$,\$,'IfcStyledItem');	
2877	# 2871=IFCELEMENT(\$,\$,'IfcRelAggregates');	
2878	# REL_2319_MK-03=IFCRELASSOCIATES(\$,# 2319,# MK-03);	
2879	ENDSEC;	
2880	END-ISO-10303-21;	
2881		

Future Improvements

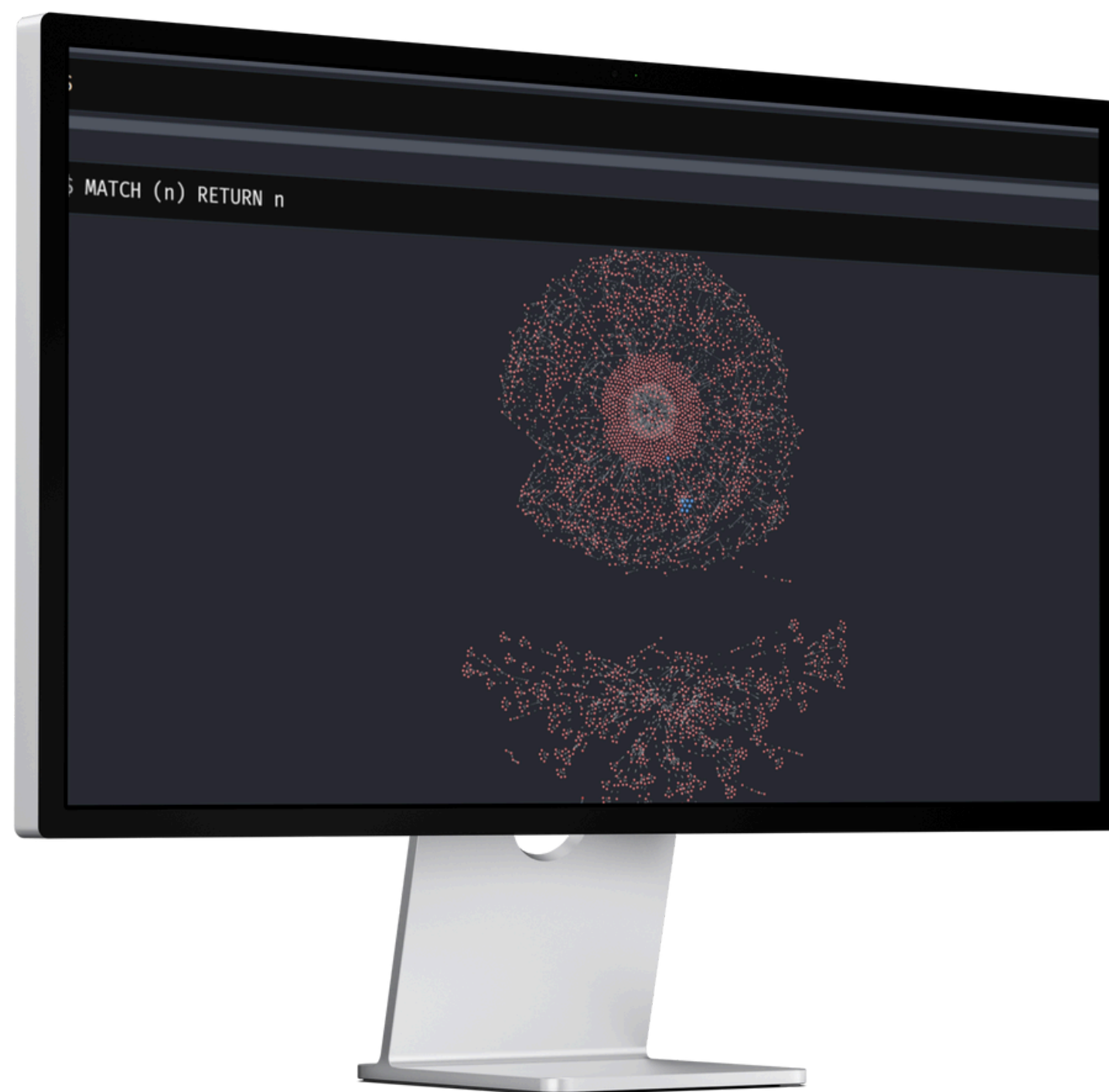
- graphical user interface (GUI)
- **real-time synchronization** between IFC files and the graph database.
- Integration with other **BIM software** (such as Revit or Navisworks).
- **machine learning capabilities** for predictive insights.
- Improving **scalability**



Demo application

<https://github.com/Diego-m4i/TBDMProject>

<http://localhost:7474/browser/>





UNIVERSITÀ
di CAMERINO

Thanks for your attention

Diego Marinangeli

