

BrickRECon25

June 2, Savannah

The session starts at
EDT 10.00am / CEST 16.00

And ends at
EDT 12.00pm / CEST 18.00



Today's agenda

10.00am/16.00

Welcome address and practical things

James McHale, Managing Director at Memoori Research

10.05am/16.05

The ROI of Open Standards

10.30am/16.30

Generative AI and REC/BRICK – Demos

11.15am/17.15

From the technical committees:

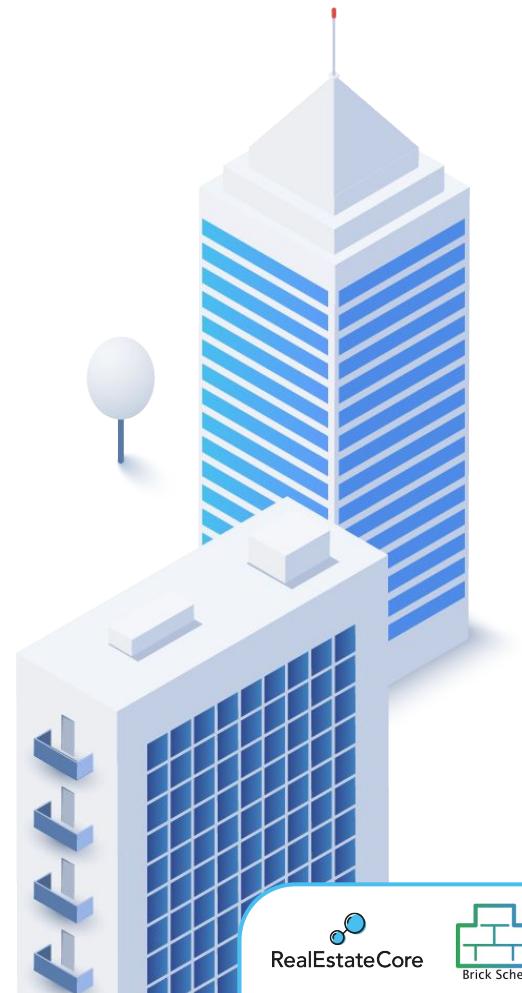
Time to start to build, Stabilizing phase, and the future

11.30am/17.30

Live Q&A with the presenters and friends

12.00pm/18.00

Conference ends



Housekeeping

Ask questions in the chat channel
during the presentation

The presentations will be available
after the conference at:

<https://www.realestatecore.io/brickrecon25/>



Presenters



Gabe Fiero

Colorado School of Mines



James McHale

Memoori



Erik Wallin

ProptechOS



Jason Koh

Mapped



Karl Hammar

Microsoft



The ROI of Open Standards

Based on the white paper: The ROI of Open Standards – How Open Data Standards Improve the Bottom Line of Managing Real Estate

- **Communication vs. Data: Understanding the Gap**
- **Debunking the "Open" Marketing Myth**
- **The Path to Official Standardization**
- **Impact on Building Owners and Operators**

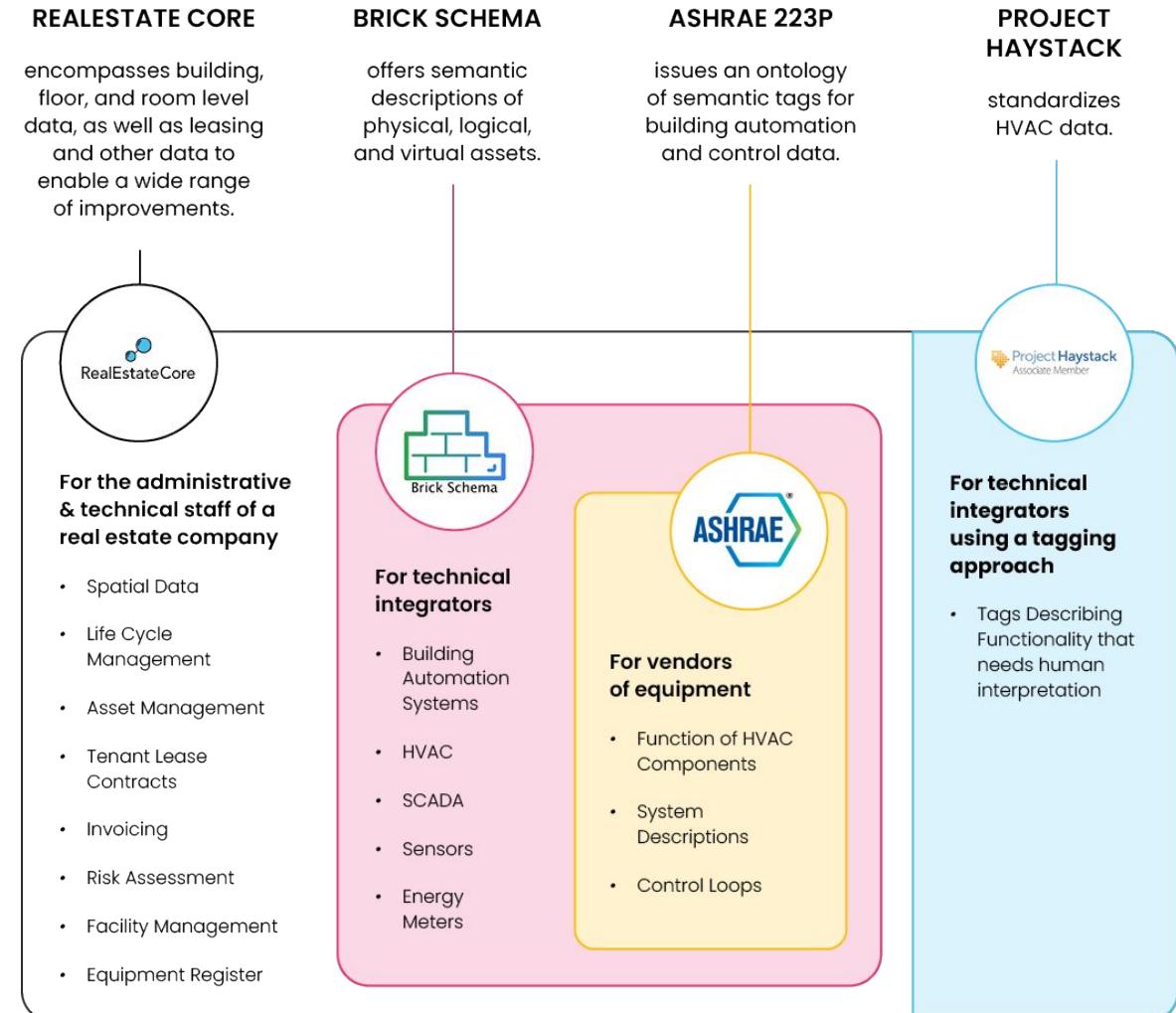
<https://www.realestatecore.io/white-paper-the-roi-of-open-standards/#download-now>

Ecosystem – The Power of Cooperation

Open equals cooperation.

Brick-REC actively works to align standards such as Haystack, ASHRAE 223p, FIWARE, VBIS, etc.

Schneider Electric EcoStruxure, Microsoft Azure Digital Twins, JCI,



ROI & NOI – no single silver bullet

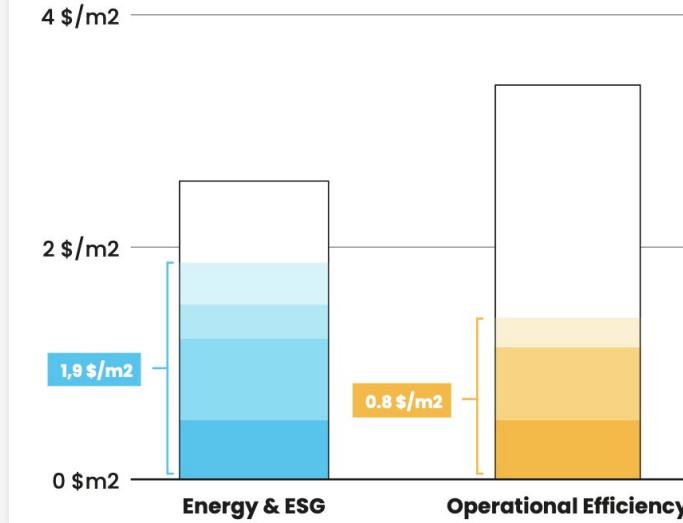
Background

- Each application offers individual measurable benefits, but can be implemented in a single onboarding project.
- Numbers are yearly and are based on 10 buildings and 100,000 m² (1M sq ft) from real-life experiences.

Examples of NOI opportunity

- Analyse & Measure: \$0.30/m² → \$30,000
- Heating/Cooling Optimization: \$0.40/m² → \$40,000
- Ventilation Optimization: \$0.50/m² → \$50,000
- Compressor Optimization: \$0.40/m² → \$40,000
- Demand Response: \$0.30/m² → \$30,000

PropTech Application
yearly NOI contribution

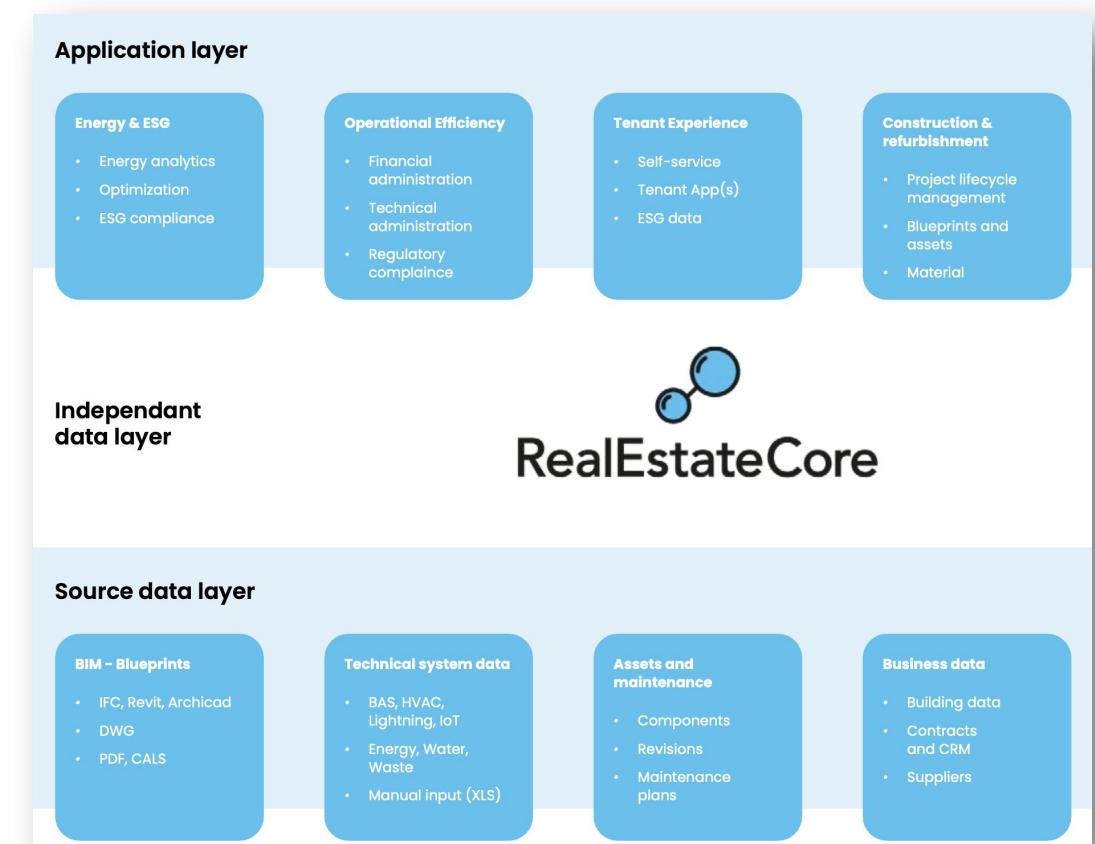


- Total NOI Improvement: \$190,000/year
- Implementation Cost: \$110,000 | ROI (3 yrs): \$570,000
- Payback Time: 10 months

NOI = Net Operating Income

RealEstateCore/Brick enable multiple uses of the same data

- Many different applications uses the same data
- Integrate the data source/building 1 time
- Create an IDL – Independent Data Layer



Industry insights

Case studies

Howard Hughes

Case study #1

How Brick/REC enables automated sustainability reporting for Howard Hughes Holding's portfolio.



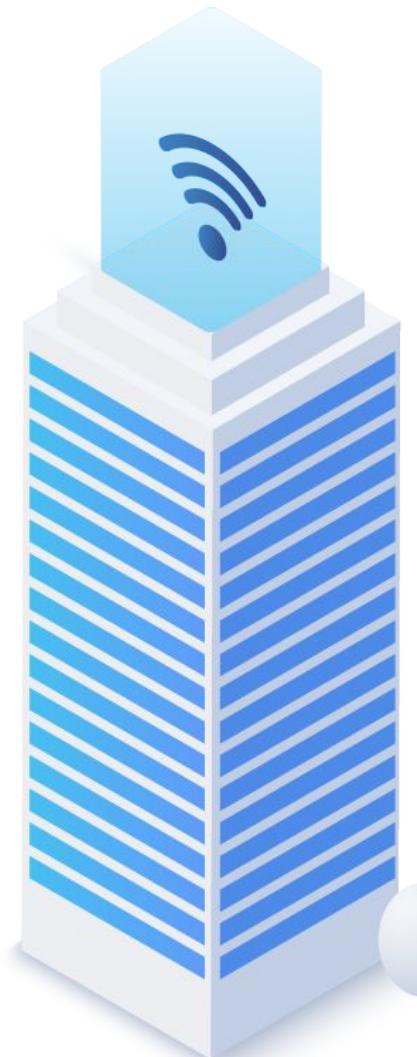
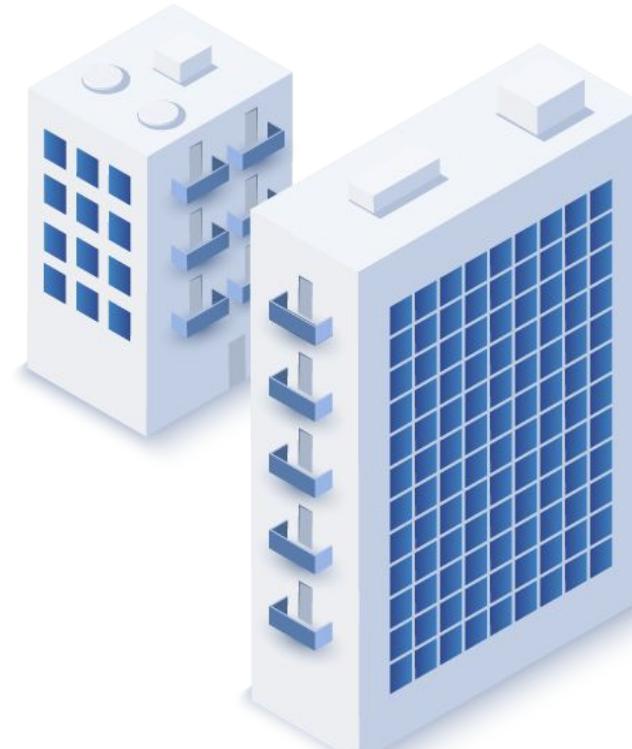
Howard Hughes Holdings

Customer Overview

- Leading U.S. real estate developer with 118,000+ acres, 10M sq ft commercial space, 6,000+ residential units.
- Serves over 387,000 residents across master-planned communities.

Challenge

- Needed centralized, automated system for sustainability and ESG data.
- Manual data collection was time-consuming and complex across diverse asset types and vendors.
- Enable fast scaling for BAS and IoT for a variety of use cases



Solution: RealEstateCore & HHX Innovation

Solution

- Adopted RealEstateCore open data standard through HHX Innovation Hub.
- Implemented a blockchain-validated data pipeline pilot.

Implementation Highlights

- Automated 92% of environmental metrics.
- Enabled fast, accurate ESG reporting (e.g., GRESB).
- IoT “package”



Results & ROI

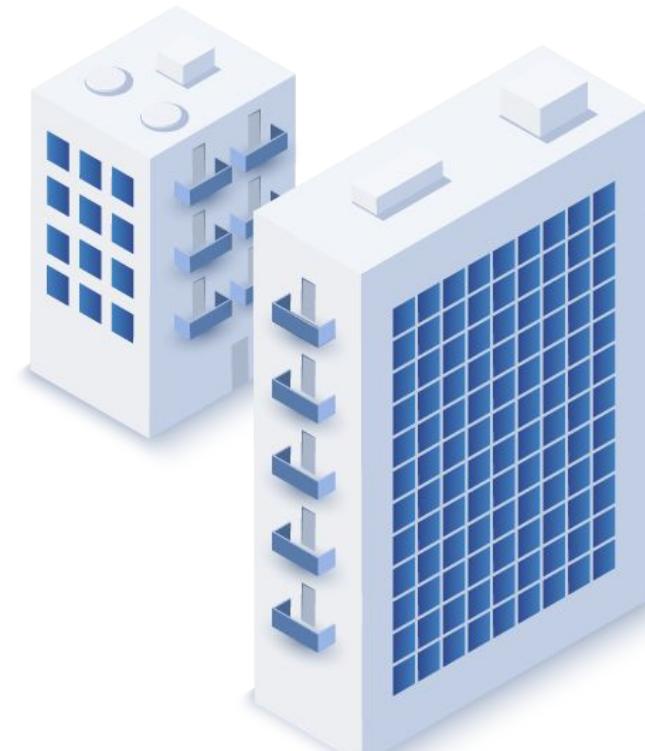
Results

- Manual data collection reduced from 6 months to near real-time.
- Improved reporting accuracy and enabled third-party assurance.

Return on Investment

- \$6,000 investment in leak detection saved \$30,000+ in potential damage.
- Enabled digital twinning, space utilization, lease management.

"The ROI was clear... it all adds up to digital twinning and better operations."
– Marcus Spillane, SVP Innovation



Case study #2

How Akademiska Hus increased usage and enabled digital transformation



AKADEMISKA HUS



Akademiska Hus

Customer Overview

- State-owned real estate leader in Sweden with 60% market share.
- Manages 34M sq. ft. of educational real estate across 15 cities.
- Property value: \$11.5B, Project portfolio: \$1.3B.

Challenge

- Diverse legacy systems and tech complexity across portfolio.
- Need to centralize data for better analysis and space optimization.



Solution: RealEstateCore & Open Architecture

Solution

- Early adopter of RealEstateCore open standard.
- Created an independent data layer for integration.
- Gradual onboarding from single buildings to entire campuses.

Implementation Highlights

- 85% of sites now feed data using RealEstateCore.
- Enabled data analysis across building, campus, and portfolio levels.



Results & Digital Transformation

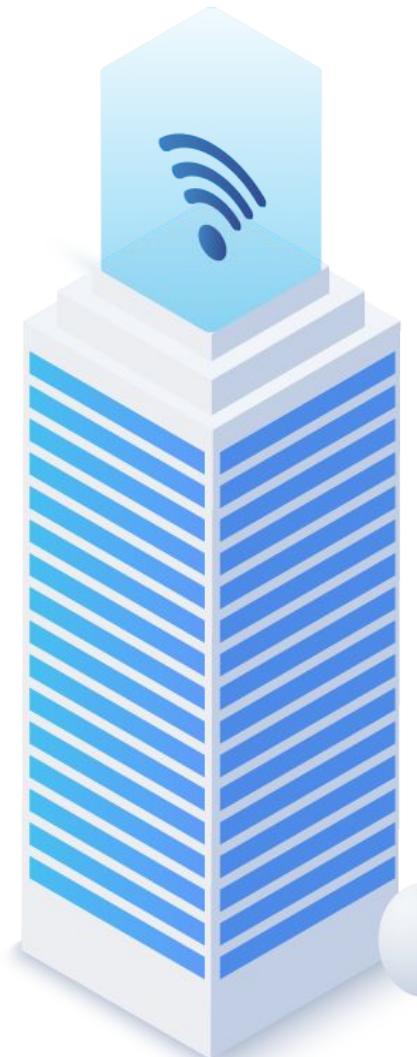
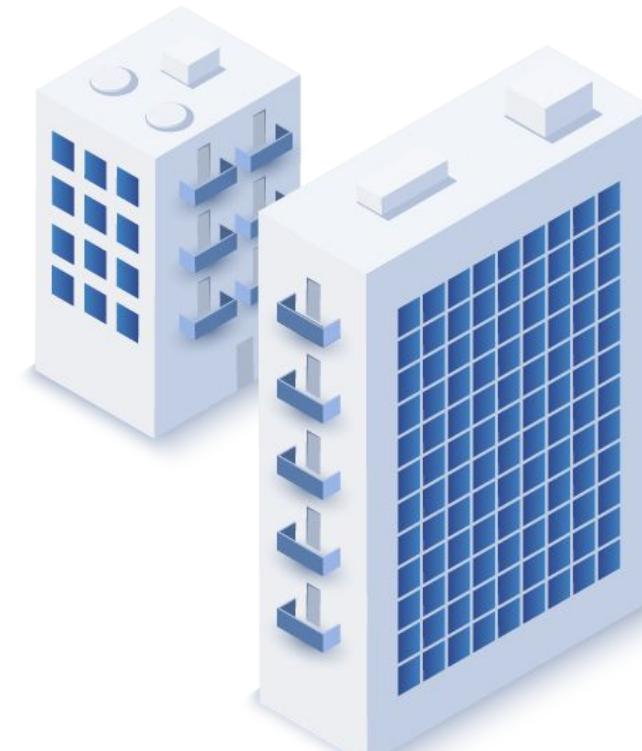
Results

- Increased daytime space utilization from 33% to 37%.
- “Saved” 3,000,000 sq.ft. of space via better insights.
- Reduced costs and improved campus experiences for users.

Transformation

- Shifted from traditional landlord to data-driven advisor.
- RealEstateCore enabled simulation, modeling, and scalable energy & maintenance use cases.

*“Now we’re more like an advisory... RealEstateCore was a key piece of the puzzle.”
– Ulf Däversjö, Chief Business Technology & Development Officer*



Generative AI and REC/BRICK

Demos

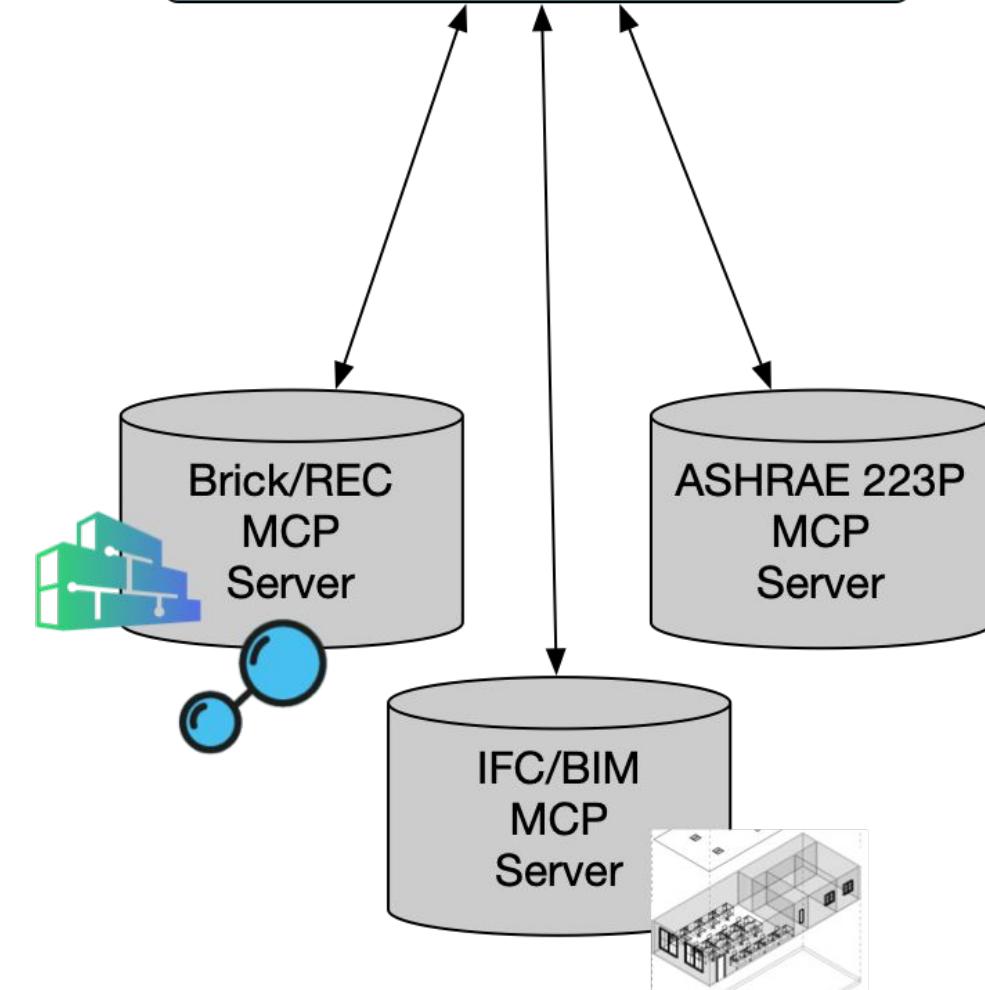
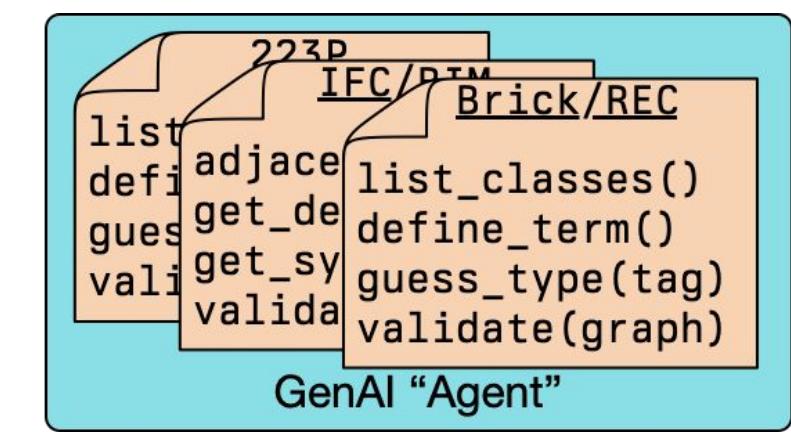
Demo#1 – Brick and MCP

- Gabe Fiero
- How LLMs and Ontologies complement each other
- Description/intro of Model Context Protocol and the rdf-mcp demo
- Code/analytics generation?
- Q&A Dataset call for action (with link)
- Demo from-mep-and-screens-shots-2-working-brick-modell
- Use the REC repo as a git submodule to build Brick, and I use:
<https://github.com/BrickSchema/Brick/blob/master/bricksrc/recpatches.ttl>
- <https://github.com/gtfierro/rdf-mcp>



Ontologies and MCP

- Model Context Protocol is a generic way for genAI agents to use “tools”
- Easy/cheap way to make your agent an “expert” in a new domain
- MCP tells the agent what “tools” are available and what they do



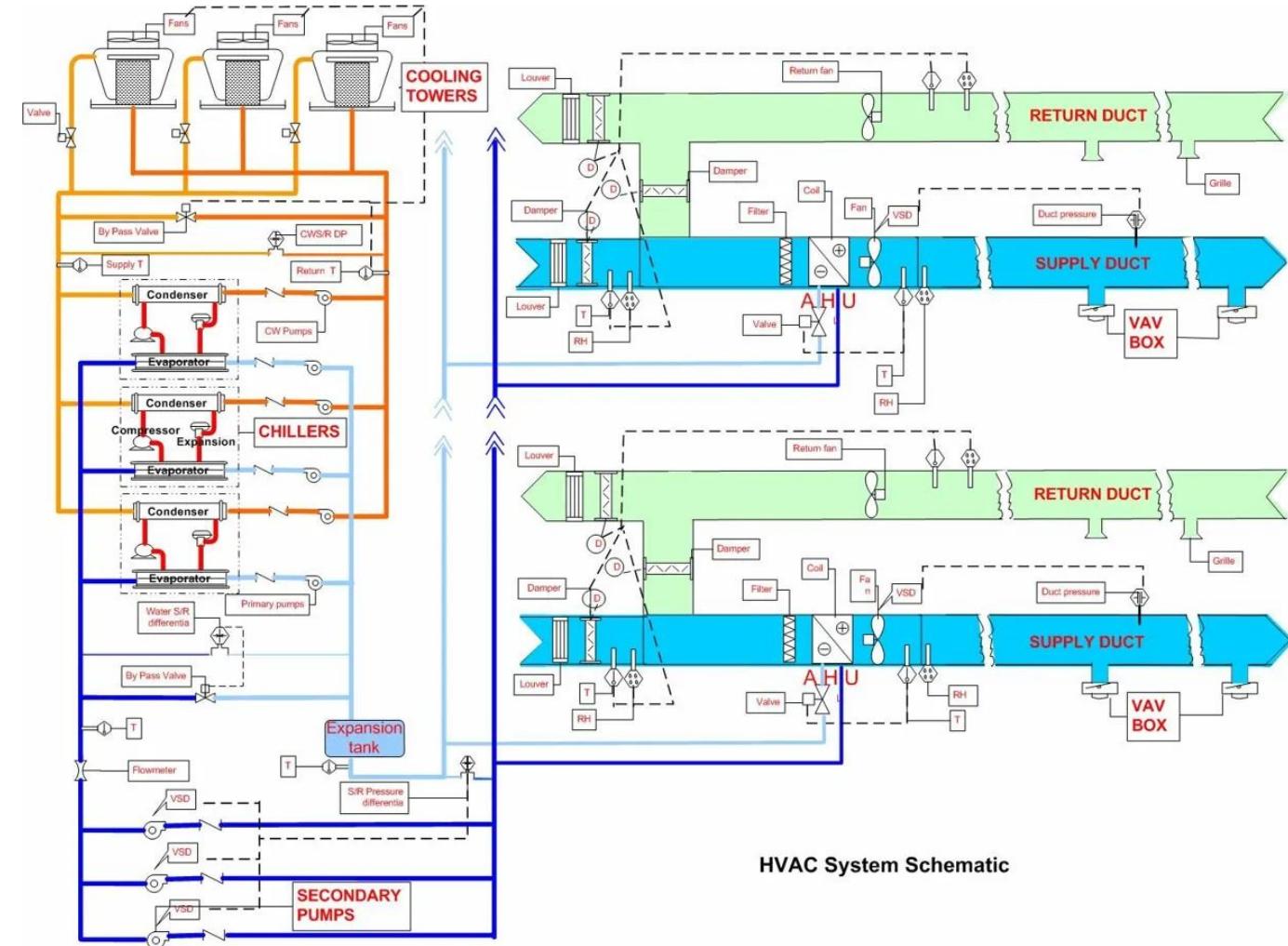
MCP Server for Brick, REC, 223P

Tool	Purpose
get_terms()	Lists all classes, concepts, types
get_properties()	Lists all property, relationship, edge types
guess_type(tag)	Guesses the type of the entity based on abbreviations, semantic embeddings
get_available_properties(type)	Gets all possible edge types for a given type
get_definition(type)	Gets the RDF definition of the type
validate(graph)	Checks the (sub) graph for validity against the ontology

- 6 simple functions, spanning ~200 lines of Python code
- Re-use for several tasks: model creation, Q&A, etc
- Open-source: <https://github.com/gtfierro/rdf-mcp>

Demo

- Input: mechanical diagram I downloaded off the internet
- Output: validated and functional Brick model
- Take advantage of native image processing capabilities of “multi-modal” GenAI models



HVAC System Schematic

Demo

- The model recognizes the equipment types, retrieves definitions
- Calls tools as needed

```
G get_definition_brick

brick:Cooling_Tower a owl:Class,
    sh:NodeShape ;
    rdfs:label "Cooling Tower"@en ;
    rdfs:seeAlso <https://en.wikipedia.org/wiki/Cooling\_tower> ;
    rdfs:subClassOf brick:HVAC_Equipment ;
    skos:definition "A cooling tower is a heat rejection device that rejects waste heat to the atmosphere through the cooling of a water stream to a lower temperature. Cooling towers may either use the evaporation of water to remove process heat and cool the working fluid to near the wet-bulb air temperature or, in the case of closed circuit dry cooling towers, rely solely on air to
```

I'll help you build a Brick model of this HVAC system. Let me start by analyzing the diagram and then verifying the necessary Brick terms and their relationships.

From the diagram, I can see:

- Cooling towers with fans and pumps
- Chillers with condensers, evaporators, and compressors
- Air Handling Units (AHUs) with supply and return ducts
- VAV boxes
- Various sensors, valves, and control components
- Secondary pumps
- Expansion tank

Let me start by verifying key Brick terms that I'll need:



Fetches definition
of cooling tower

Demo

- The model recognizes the equipment types, retrieves definitions
- Calls tools as needed
- Retrieves both Brick and REC definitions

```
G get_definition_brick

brick:Cooling_Tower a owl:Class,
    sh:NodeShape ;
    rdfs:label "Cooling Tower"@en ;
    rdfs:seeAlso <https://en.wikipedia.org/wiki/Cooling\_tower> ;
    rdfs:subClassOf brick:HVAC_Equipment ;
    skos:definition "A cooling tower is a heat rejection device that rejects waste heat to the atmosphere through the cooling of a water stream to a lower temperature. Cooling towers may either use the evaporation of water to remove process heat and cool the working fluid to near the wet-bulb air temperature or, in the case of closed circuit dry cooling towers, rely solely on air to
```

I'll help you build a Brick model of this HVAC system. Let me start by analyzing the diagram and then verifying the necessary Brick terms and their relationships.

From the diagram, I can see:

- Cooling towers with fans and pumps
- Chillers with condensers, evaporators, and compressors
- Air Handling Units (AHUs) with supply and return ducts
- VAV boxes
- Various sensors, valves, and control components
- Secondary pumps
- Expansion tank

Let me start by verifying key Brick terms that I'll need:



Fetches definition
of cooling tower

Demo

- Generates Brick/REC model
- Runs validation to check that the model “makes sense”
- Iterates as necessary

Now let me build the complete model. I'll start by updating the artifact with a comprehensive model:

Complete HVAC System Brick Model
Code · Version 2



Now let me try to validate this model:



```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix unit: <http://qudt.org/vocab/unit/> .
@prefix ex: <http://example.org/building/> .
```

```
ex: a owl:Ontology ;
    rdfs:label "HVAC System Model" .
```

```
# ===== SYSTEMS =====
```

```
ex:chilled_water_system a brick:Chilled_Water_System ;
    brick:hasPart ex:chiller_1,
    ex:chiller_2,
    ex:chiller_3,
    ex:primary_chw_pump_1,
    ex:primary_chw_pump_2,
    ex:primary_chw_pump_3,
    ex:secondary_pump_1,
    ex:secondary_pump_2,
    ex:secondary_pump_3,
    ex:expansion_tank .
```

```
ex:condenser_water_system a brick:Condenser_Water_System ;
```

```
brick:hasPart ex:cooling_tower_1,
    ex:cooling_tower_2,
    ex:cooling_tower_3,
    ex:condenser_water_pump_1 .
```



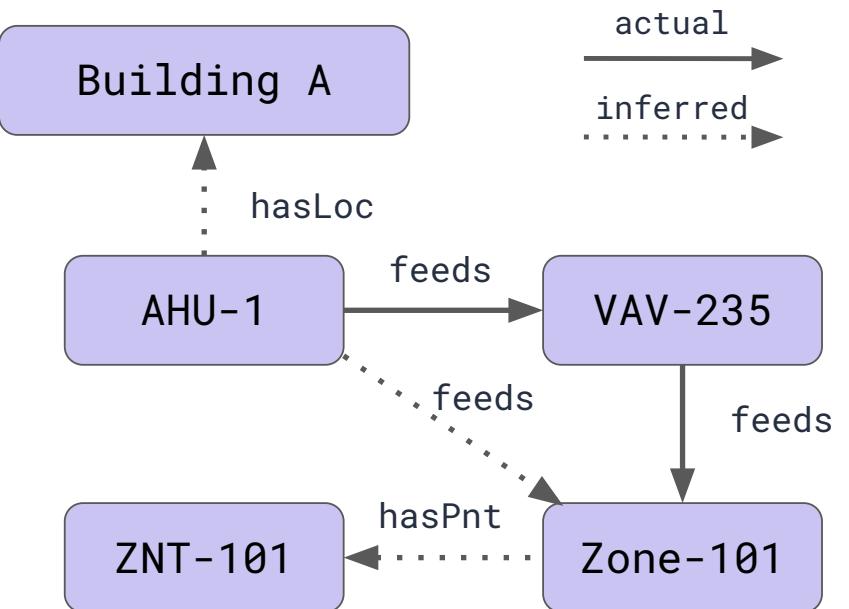
Demo#2 – Mapped

- Jason Koh
- Demo data exploration with LLMs and the ontology

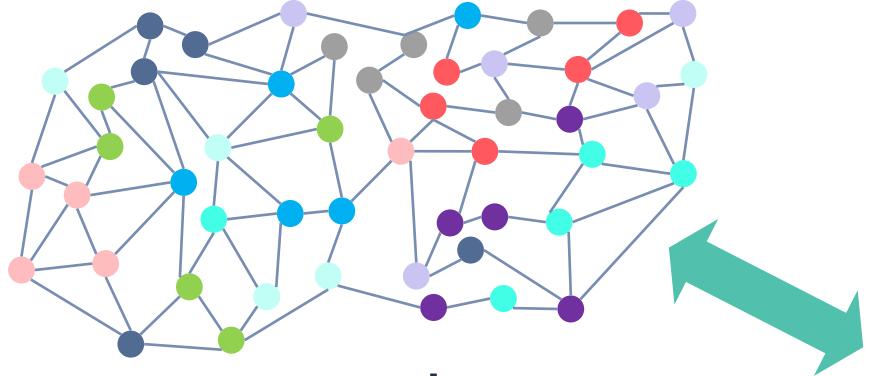
Mapped GraphQL

- GraphQL is a **developer-friendly** querying model for structured data.
- GraphQL can federate **different endpoints**
(E.g., timeseries data, Gremlin, SQL, gRPC)
- **Mapped's opinionated view** on Brick / REC, proven by our users

```
{buildings {  
    things { name type  
        feeds { ... on Zone  
            { name type  
                points {name type series (latest: true) {  
                    timestamp value  
                }  
            }  
        }  
    }  
}
```



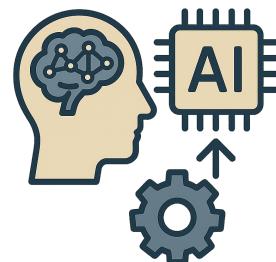
Chat Agent: Anaxi



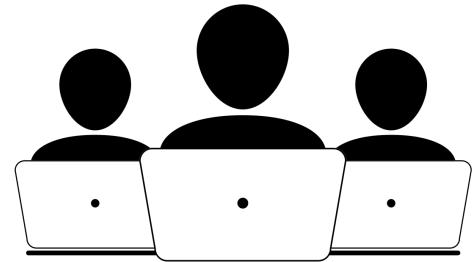
Graph



Time Series



The room with the lowest temperature?



Users

Room 101 is the coldest with temperature 18 C.

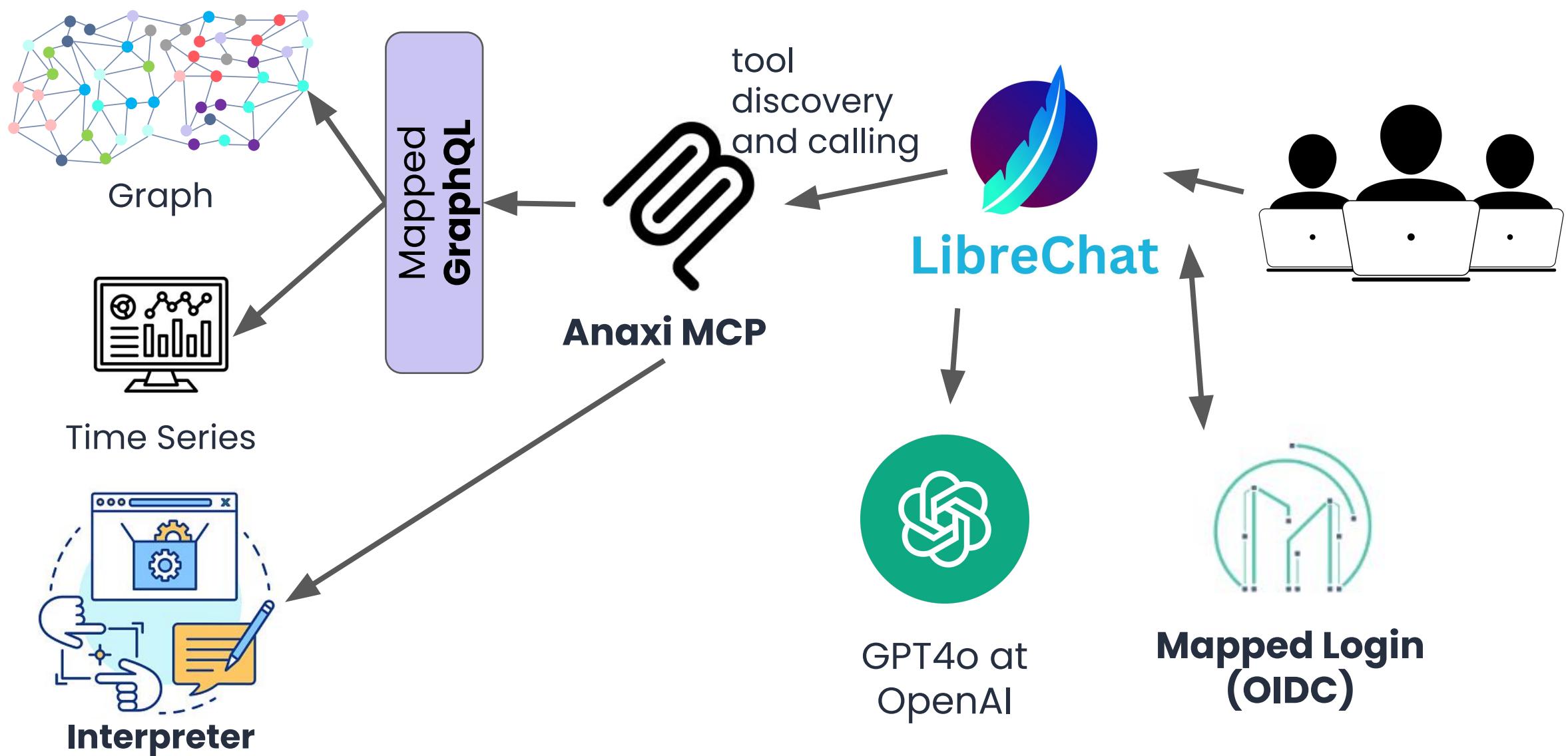
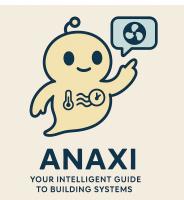


chat.mapped.com
(after registration at app.mapped.com)

Anaxi Chat Enablers

1. GraphQL generation out of
~6196 LoC GraphQL schema & ~1,000 Brick classes.
 - a. Slice and dice and search
 - b. Example lookup
 - c. Ontology summarization
2. Analytical code generation
 - a. Instructions for preferred output
 - b. Example Lookup
 - c. Safe execution at mcp-runner with deno
3. Integration
 - a. MCP to expose those tools
 - b. LibreChat as a chat interface with our profile

Anaxi Architecture



Live Chat 1/3

JA Jason

list orgs



GPT-4o

✓ Ran list_orgs ▾

Here are the organizations you can access:

1. Mapped Demo by Jason (ID: ORGBkUaTBj7ZxVffC7J9ytcR3)
2. TenancyWorks (ID: ORGUmvDdxFU7UBvJ8Msx2A6J)

If you need more information about these organizations, let me know!



JA Jason

What buildings do I have in TenancyWorks?



GPT-4o

✓ Ran list_buildings ▾

You have access to the following building in the TenancyWorks organization:

- Intellicare Infirmary (ID: BLDG5o26DguWKu5T9nRvSYn5Em)

If you need more details about this building or wish to perform any actions, feel free to ask!



JA Jason

show me all the AHUs in the building



GPT-4o

✓ Ran analyze_building ▾

Here are the Air Handling Units (AHUs) in the Intellicare Infirmary building:

1. AHU-2

- ID: THG5WW9ARhXBfdR7F3AB6CXaY

2. AHU-1

- ID: THGQsM3h5rJ6iL9CDg6KZN3nQ

If you need further details or analysis on these AHUs, just let me know!



JA Jason

what's the difference between discharge air temperature and its setpoint for AHU-1?



GPT-4o

✓ Ran analyze_building ▾

The difference between the discharge air temperature and its setpoint for AHU-1 is 0.00°C, indicating that the actual temperature is perfectly matching the setpoint. If there's anything more you'd like to explore, feel free to ask!



Live Chat 2/3

JA

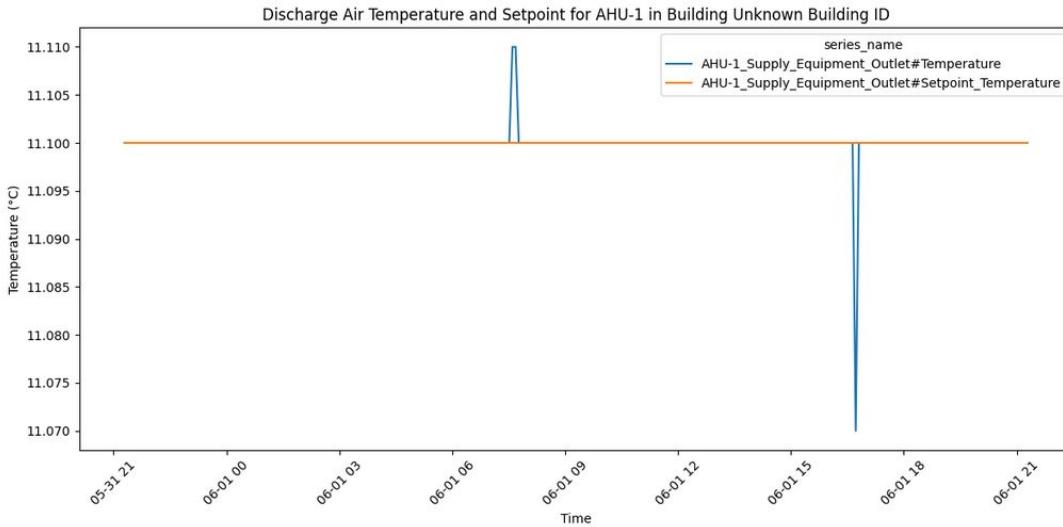
Jason

Can you plot them in the last 24 hours?



GPT-4o

✓ Ran plot_graph ▾



Here's a plot showing the discharge air temperature and its setpoint for AHU-1 over the last 24 hours. If you have any questions or need further analysis, feel free to ask!



Live Chat 3/3



GPT-4o

✓ Ran analyze_building ▾

Here is the number of spaces on each floor of the Intellicare Infirmary building:

- **Floor ID FLR8R1K5f9HtjRtVgohCAEGas:** 110 spaces
- **Floor ID FLRUDY8ChSobVHrTvpfKk4fru:** 74 spaces
- **Floor ID FLR297rRVtCm34Qzp2idcynFC:** 44 spaces

If you need more detailed information about any specific floor or space, let me know!



Jason

What's the average room temperature spaces in the floor FLR8R1K5f9HtjRtVgohCAEGas?



GPT-4o

✓ Ran analyze_building ▾

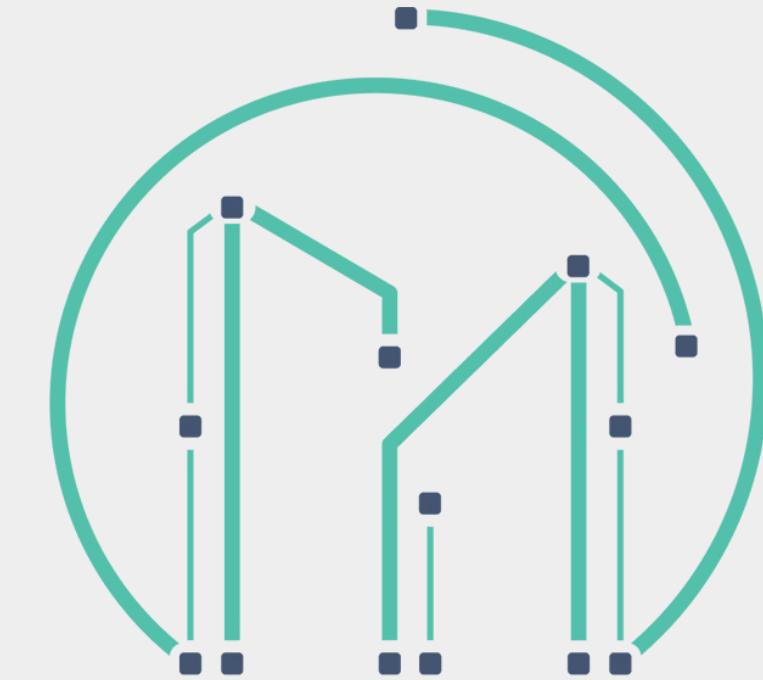
The average room temperature of spaces on the floor with ID FLR8R1K5f9HtjRtVgohCAEGas is 21.26°C. Please note that there is missing data for 55 out of the 110 spaces on this floor. If you need further information or analysis, feel free to ask!



Live Demo at
Booth #409

Dr. Jason Koh
Chief Data Scientist
jason@mapped.com

chat.mapped.com
(after registration at app.mapped.com)



MAPPED

Demo#3 – MS Copilot Studio

- **Karl Hammar**
 - **MS Copilot Studio**
 - **Property owner: KLP Properties**
 - **Building Eufemia: 200,000 sq ft, offices**
 - **Systems: Swegon, ProptechOS, Siemens Desigo CC, MS Dynamics**
 - **Use case#1 – Room finder for tenants**
 - **Use case#2 – Agent for occupancy analytics**



1
kdp

Copilot Studio

Low-code / no-code solution for business users.

Use cases:

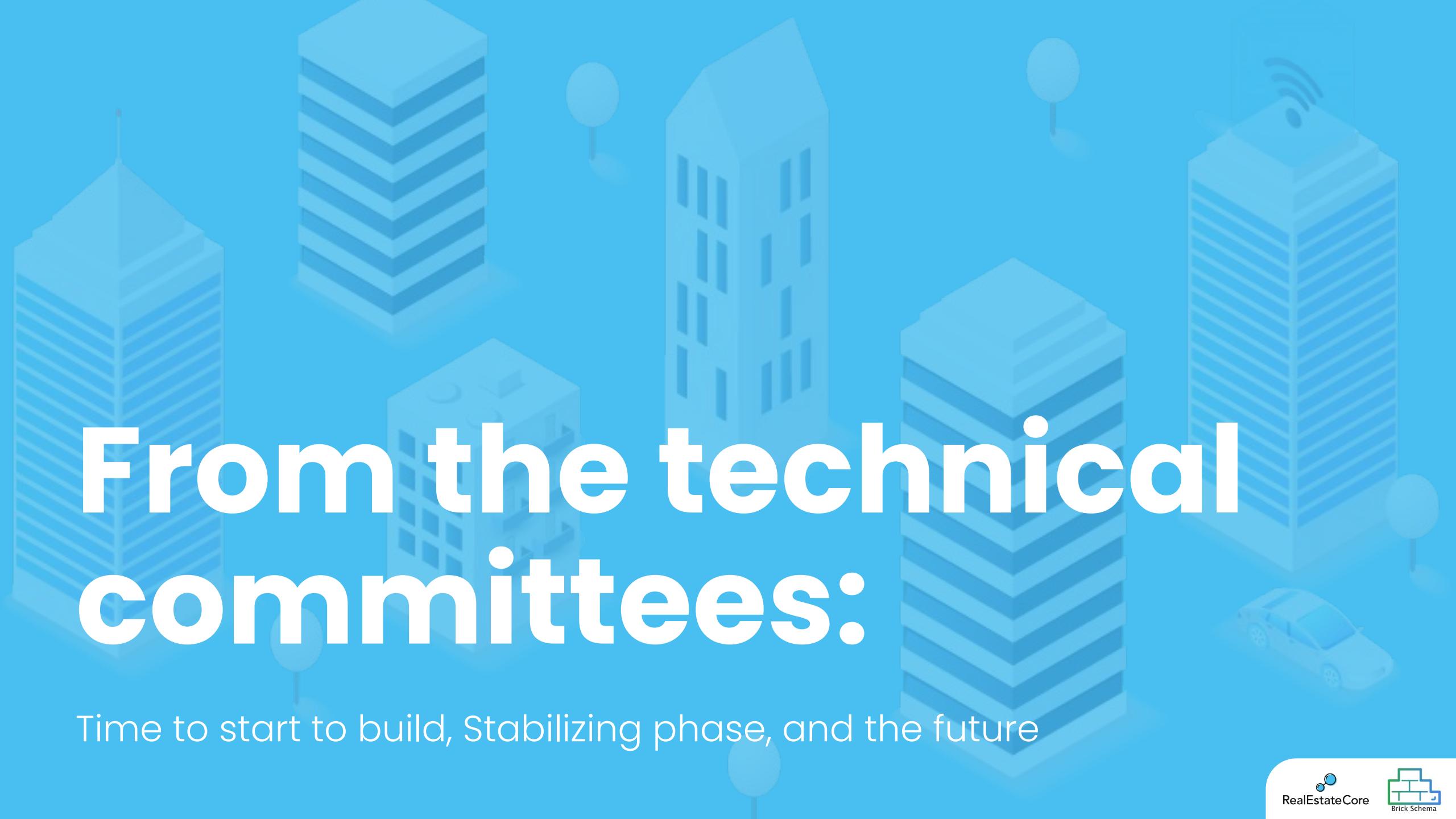
- Add knowledge sources and tools to base LLM models.
- Orchestrate multi-agent problem solving.
- Run autonomously.

Connectors for broad variety of systems and data sources

Deploy to Teams, M365 Copilot, Web, Slack, Telegram, ...

Takeaways

- Shared standard APIs enable connector reuse and next-to-instant time-to-value.
- APIs need to be documented for LLM understanding.
- Make sure you have the data.
- *While enabling the problem to be solved:*
 - Endpoints should offer as little variability as possible.
 - Endpoints should return as little data as possible.
- Build task-specific agents and orchestrate them.



From the technical committees:

Time to start to build, Stabilizing phase, and the future



- **Deeper integration with existing ontologies: QUDT, REC, ASHRAE 223P**
- **Additional validation/testing, born out of bug fixes from people deploying Brick**
- **Validation testing lets us do “point” releases more often, with the changes requested by the community**
- **Focuses for the next year:**
 - How to specify Brick/REC in contracts, and how to verify that the brick model meets your requirements. Supported with tooling
 - BuildingMOTIF overview
 - Aligning with the REC Procurement documentation
<https://github.com/RealEstateCore/procurement>
- **How to deploy/embed brick? Need help from the community on this one. What are the problems?**
 - Updating embedded devices
 - Compile Brick to tables with SHACL shapes



- **Upcoming: Brick 2.0 and REC 5.0**

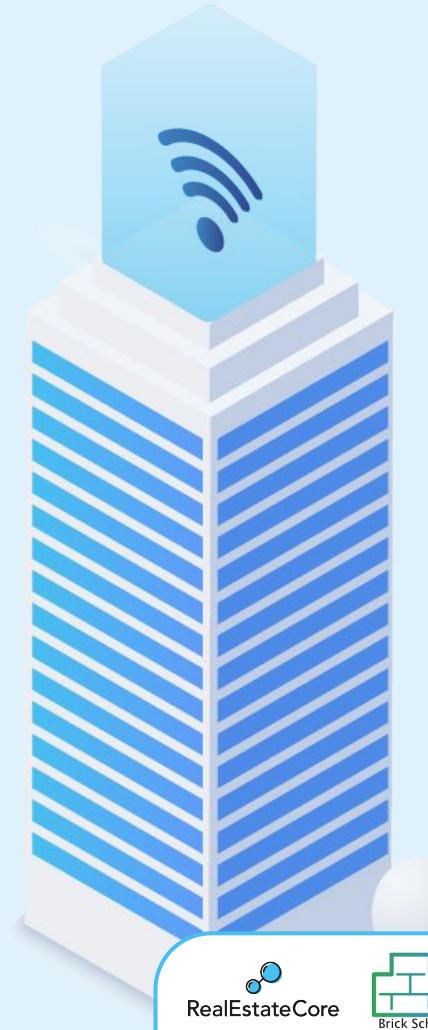
- Possibly next year (Brick 2.0 and REC 5): most things stay the same, but we will fix our old designs that we no longer need
- REC goes SHACL first to align better with Brick and ASHRAE 223p (DTDL will still be supported)
 - Development friendliness, Easy Extensibility, Backward Compatibility

- **Community**

- Follow our work on Github
- <https://github.com/RealEstateCore>
- <https://github.com/BrickSchema/Brick>
- REC Gitter will be retired and replaced by Brick Google groups

- **Lighting sub-committee started**

- Spearheaded by Fagerhult Group





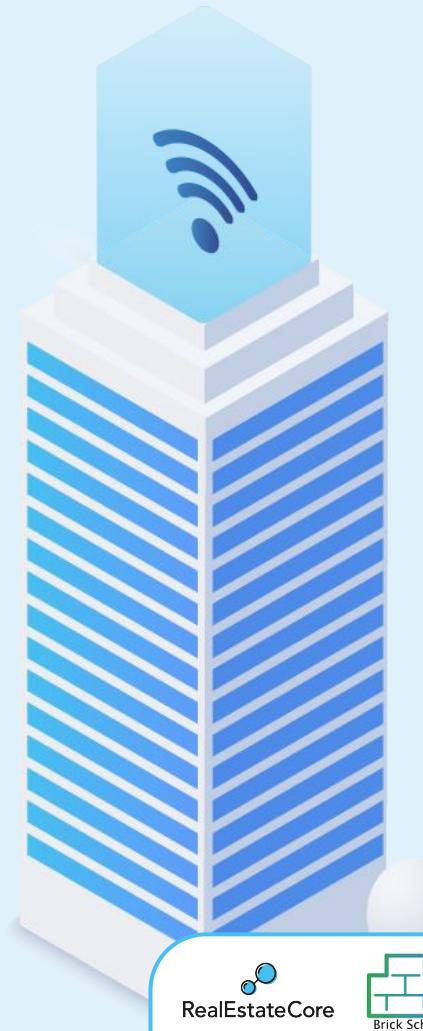
RealEstateCore Updates

Core Requirements:

- Openness: Use open standards (e.g., W3C), open-source artifacts, and open APIs.
- Development Friendliness: Simplify development by standardizing on OWL/SHACL.
- Easy Extensibility: Allow users to extend and contribute to the ontology.
- Backward Compatibility: Maintain tooling to convert OWL/SHACL to DTDL.

Next Steps:

- OWL/SHACL rewriting underway.
- Some semantic adjustments (e.g., rec:Spaces) documented via Ontology Decision Records (ODRs).
- User feedback encouraged via GitHub.

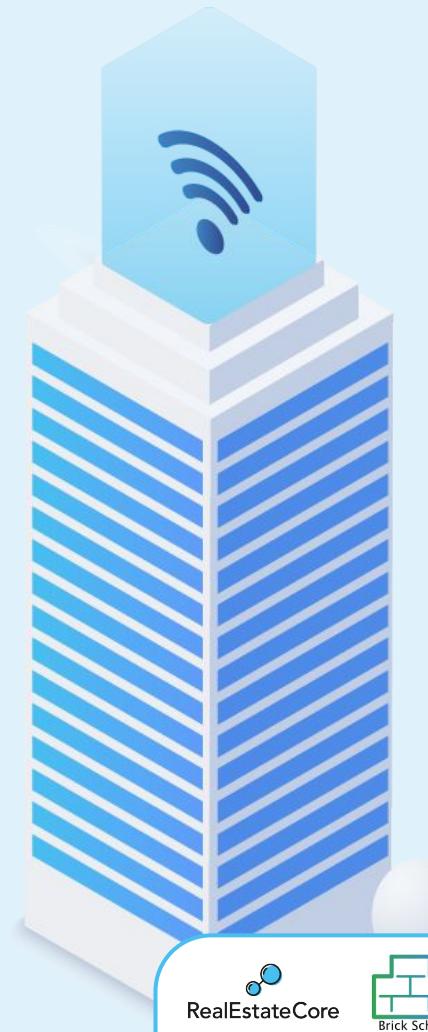




Brick Updates

- **More rapid “point” releases:**
 - 1.4.2, 1.4.3, 1.4.4 all released in 6month window
 - Lets us respond to community requests more quickly
- **Integrated build process**
 - Brick and REC now distributed together on brickschema.org
 - Available in the same file!
- **Increased validation for future releases**
 - Catch bugs before they show up in products

Version	Release Date	Downloads			Changelog
latest	Nightly	Turtle			Changelog
1.4.4	2025-05-01	JSON-LD	RDF/XML	Turtle	Changelog
1.4.3	2025-03-20	JSON-LD	RDF/XML	Turtle	Changelog
1.4.2	2024-09-20	JSON-LD	RDF/XML	Turtle	Changelog
1.4.1	2024-08-22	JSON-LD	RDF/XML	Turtle	Changelog
1.4.0	2024-04-15	JSON-LD	RDF/XML	Turtle	Changelog





Brick Updates

- **Deeper integration with existing ontologies**
 - RealEstateCore support remains “first class”
 - ASHRAE 223P development is finished (waiting on votes)
 - Support for new QUDT v3 ontology
- As of Brick 1.4.4, the vast majority of quantity kinds in Brick have been replaced with QUDT equivalents
- IFC-LD (ifc-id.org) is a new effort to support RDF encoding of IFC models
 - early in development, but a possible future collaborator

quantitykind:AtmosphericPressure ttl html Atmospheric Pressure	quantitykind:AtomScatteringFactor ttl html Atom Scattering Factor
quantitykind:AtomicEnergy ttl html	quantitykind:AtomicMass Atomic Mass ttl html
quantitykind:AuditoryThresholds ttl html Auditory Thresholds	quantitykind:AuxillaryMagneticField ttl html Auxillary Magnetic Field
quantitykind:AverageSpecificImpulse ttl html Average Specific Impulse	quantitykind:AverageVacuumThrust ttl html Average Vacuum Thrust
quantitykind:BandwidthLengthProduct ttl html	quantitykind:Basicity Basicity ttl html
quantitykind:BetaDisintegrationEnergy ttl html Beta Disintegration Energy	quantitykind:BevelGearPitchAngle ttl html Bevel Gear Pitch Angle
quantitykind:BiodegradationHalfLife ttl html Biodegradation Half Life	quantitykind:BiogeochemicalRate ttl html Biogeochemical Rate
quantitykind:BloodGlucoseLevel_Mass ttl Blood Glucose Level by Mass html	quantitykind:BodyMassIndex Body Mass Index ttl html
quantitykind:Breadth breadth ttl html	quantitykind:BucklingFactor Buckling Factor ttl html
quantitykind:BurnRate Burn Rate ttl html	quantitykind:BurnTime Burn Time ttl html
quantitykind:ByteRate ttl html	quantitykind:ByteTransmissionRate ttl html
quantitykind:CENTER-OF-GRAVITY_Z ttl html Center of Gravity in the Z axis	quantitykind:CENTER-OF-MASS ttl html Center of Mass (CoM)
quantitykind:CONTROL-MASS Control Mass. ttl html	quantitykind:CanonicalPartitionFunction ttl Canonical Partition Function html
quantitykind:CartesianArea Cartesian Area ttl html	quantitykind:CartesianCoordinates ttl html Cartesian coordinates
quantitykind:CatalyticActivityConcentration ttl Catalytic Activity Concentration html	quantitykind:CelsiusTemperature ttl html Celsius temperature

QUDT now contains many HVAC/lighting related quantity kind definitions

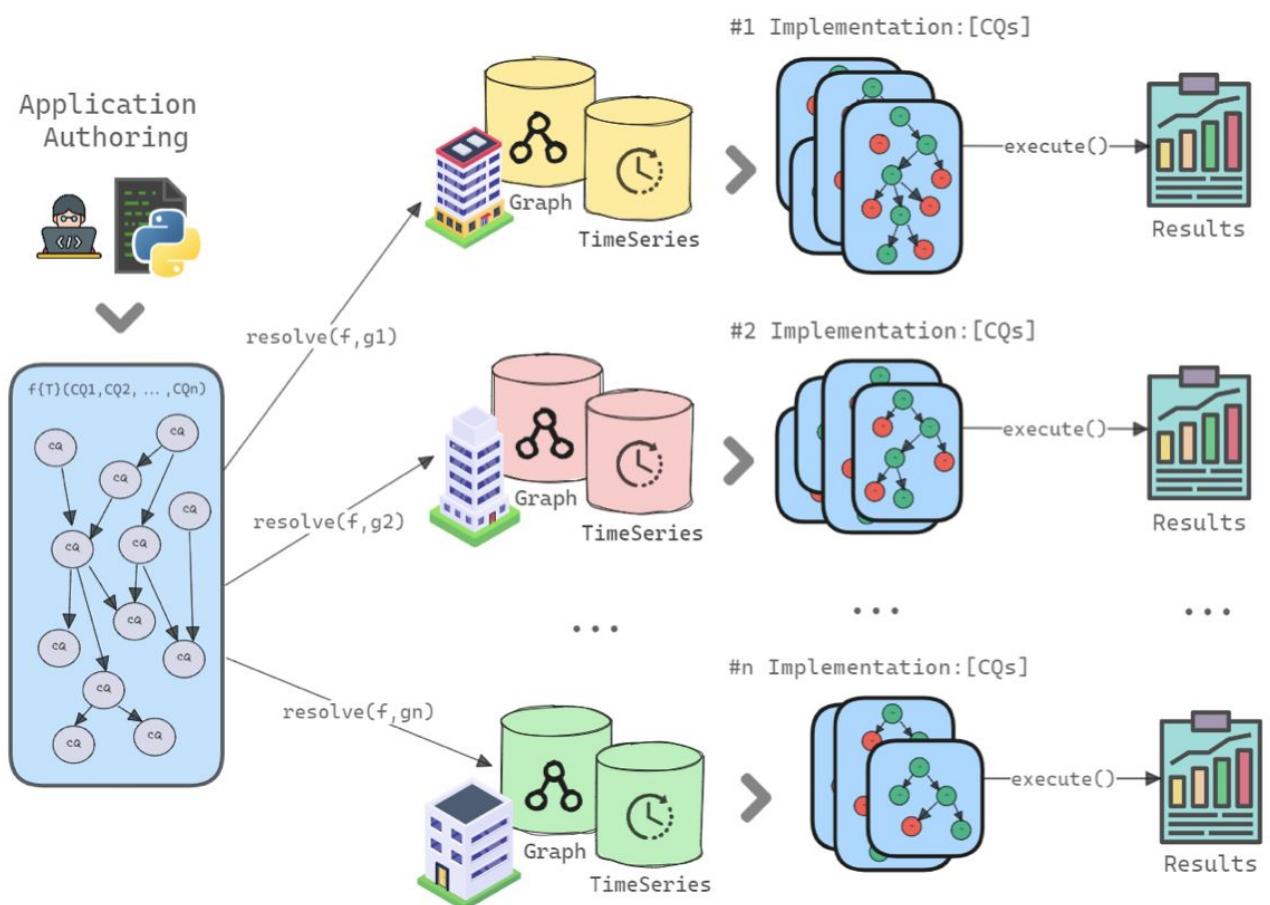
Brick Updates

- (outlines)
- More rapid “point” releases: 1.4.2, 1.4.3, 1.4.4
 - Respond to community requests more quickly
- Increased validation for future releases
 - Catch bugs before they show up in future products
- Integrated Brick/REC build process
- ASHRAE 223P is released! Time to integrate
- In-progress stuff
 - setpoints (Jason)
- **Gabe needs to finish this slide**

Research: New Programming Models

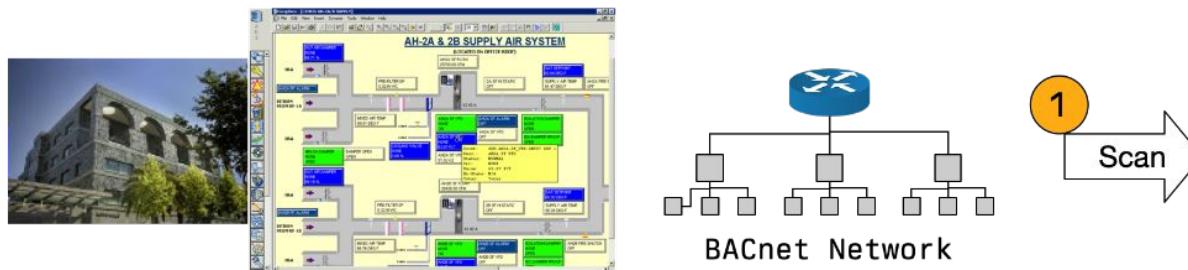
- Write Python applications against concepts defined by metadata ontology
- SeeQ “compiles” the Python code against the metadata model for each building
 - Generates building-specific impl.
- Demonstrated on FDD rules
- Step towards fully **portable** applications

```
1 from SeeQ import *
2 from pandas import DataFrame
3 from G36.CQs import Dmp_Pos, Fsa, Fsp_clg, Fan_s
4 from APAR.CQs import Tsa, Tma, DelTsf, Hc_pos, Epsilon_t
5
6 def APAR_R1(sup: Tsa, mix: Tma, drop: DelTsf, heat_coil: Hc_pos, e: Epsilon_t):
7     is_heating: DataFrame = heating_coil.df > 0
8     supply_air_low: DataFrame = sup.df < (mix.df + drop.df - error.df)
9     violating_records = is_heating & supply_air_low
10    # returns fault if more than 10 violating samples
11    if len(violating_records) > 10:
12        return "fault detected"
13
14 def G36_Dmp_Leaking(pos: Dmp_Pos, sup_flow: Fsa, cool_sp: Fsp_clg, fan: Fan_s):
15     if ((pos.df == 0) and (sup_flow.df > max([0.1*cool_sp.df, 50])) \
16     and (fan.df == "ON")).for_time(600):
17         return "Level 4 alarm"
```



Research: Automated Model Generation

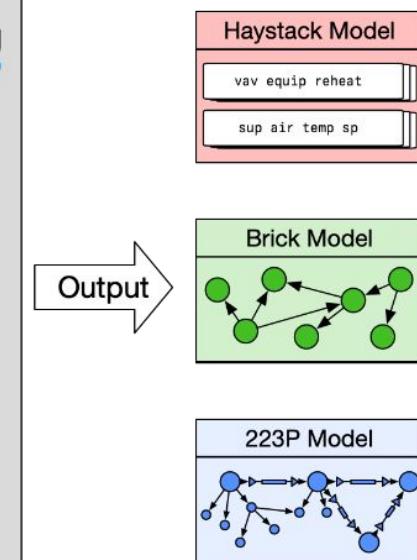
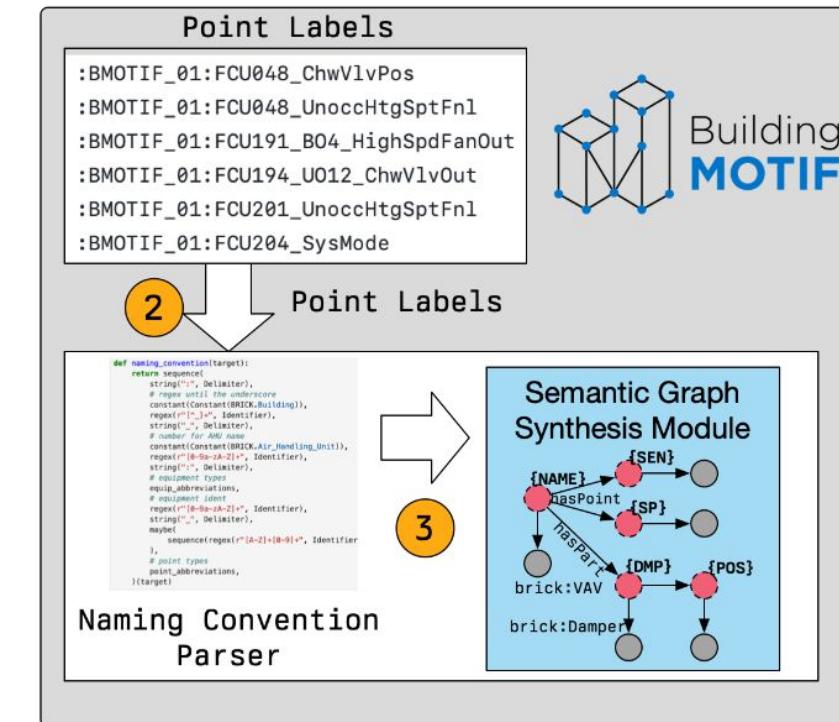
Site A



```
# load in ontologies and equipment templates
bm = BuildingMOTIF('sqlite://')
s223_ontology = Library.load(ontology_graph="http://data.ashrae.org/standard223")
equipment_types = Library.load(directory="my_libraries/equipment")

# define the BMS point label naming convention
def naming_convention(point_label):
    # ...
    # scan the network, apply the naming convention to each label,
    # and synthesize the metadata model
    network = BACnetNetwork("10.0.0.1/24")
    parsed = NamingConventionIngress(network, naming_convention)
    model = SemanticGraphSynthesizer(parsed, [equipment_types]).graph
```

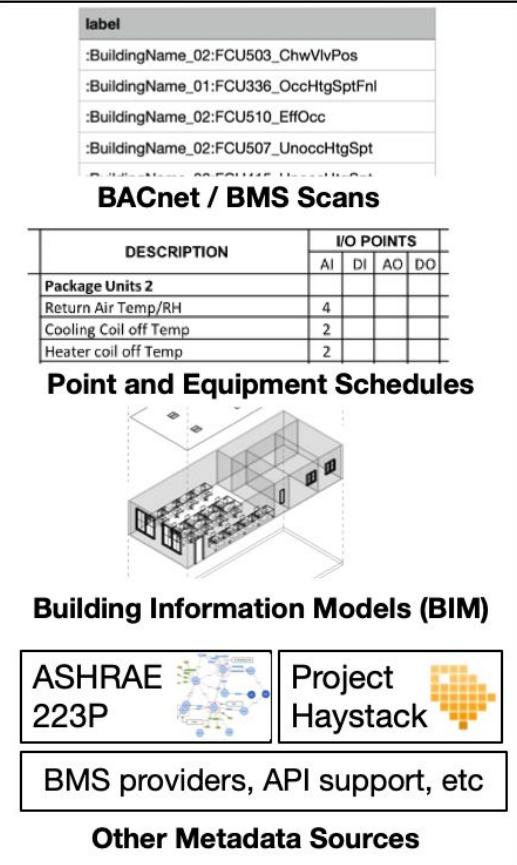
1
2
3



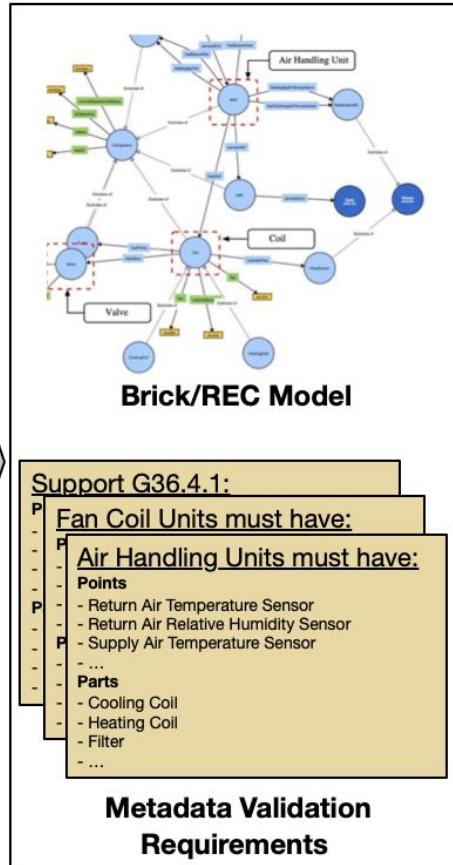
- Ontology-informed generation of metadata models from point labels
- Generates the whole model — not just point tagging!
- Built on DOE-funded BuildingMOTIF platform SDK

Semantic Model Validation

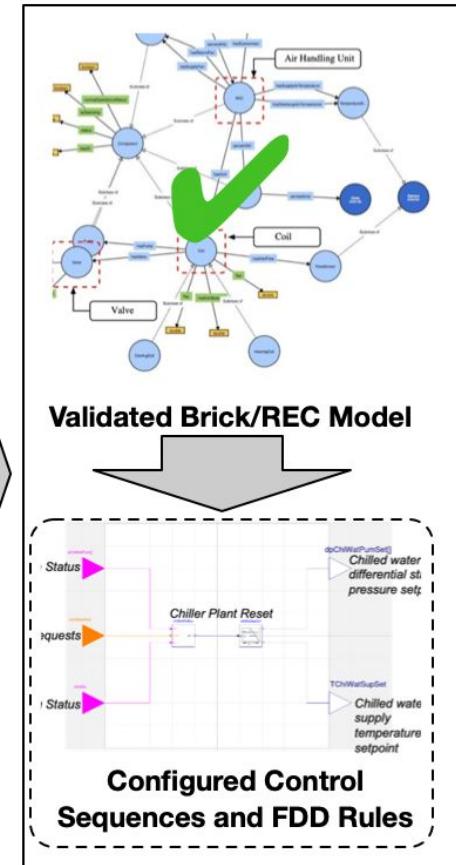
Metadata Sources



Model Validation



Application Configuration

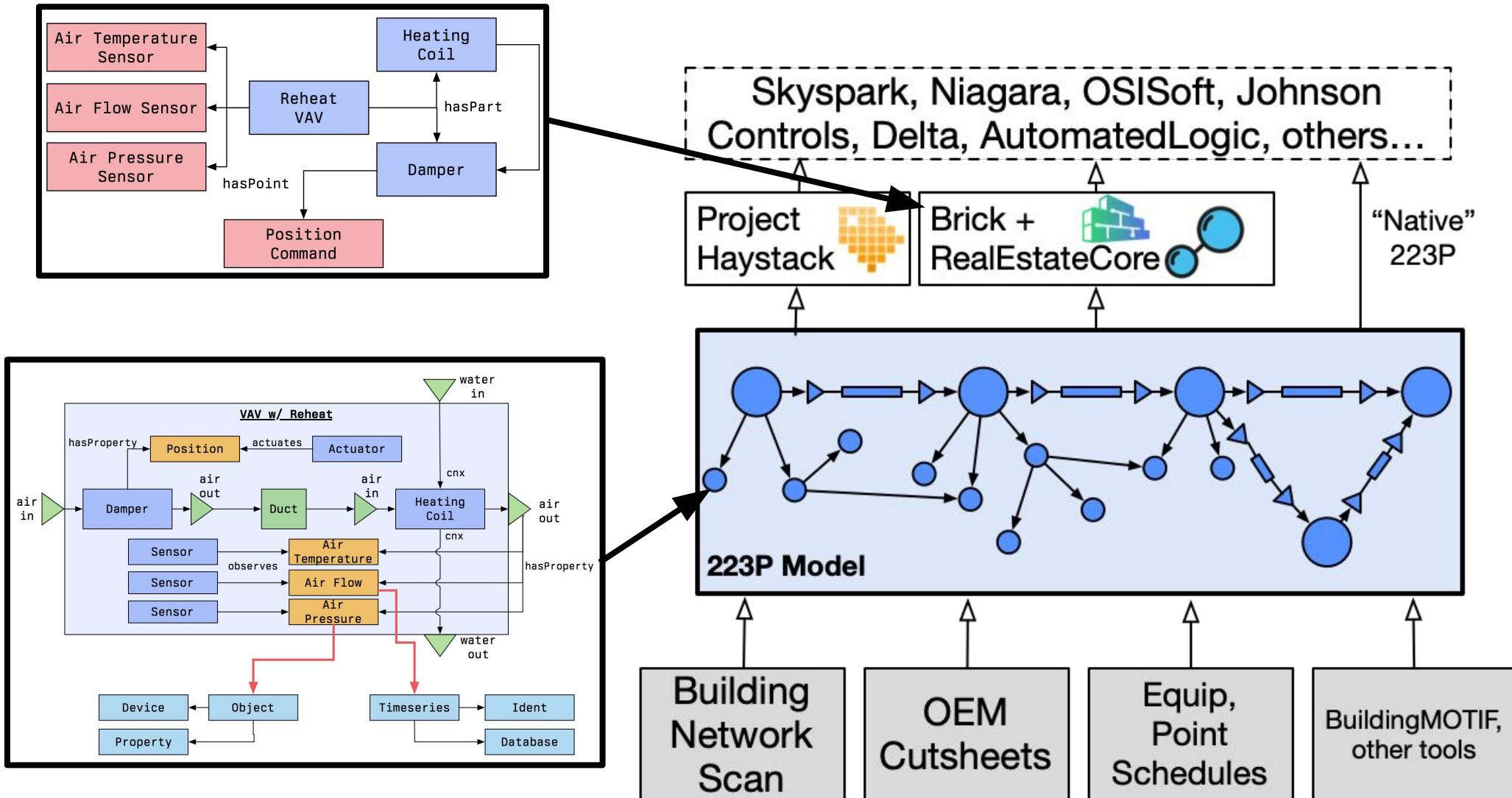


- Validation tools let you specify valid metadata as part of a contract/RFP
- Validate the delivered metadata automatically
- Use it to configure applications: controls, fault detection, dashboards, etc
- Built on DOE-funded **BuildingMOTIF** platform

Time for Brick 2.0?

- Brick was first created in 2015
- Major rewrite for Brick 1.2 in 2017
- Move from OWL to SHACL in Brick 1.3
- RealEstateCore Integration in Brick 1.4 in 2020
- ASHRAE 223P released in 2025
- Opportunity to revisit old design decisions
- Can we be “223P-first” in the design?
- Focus on the value-add of Brick/RealEstateCore above the detailed standard 223P model

Integrating Brick/REC with ASHRAE 223P



Application Platforms

Vendor / Consortium Standards

Metadata Sources

Q&A

- Christoffer Bupp (RealEstateCore Consortium, Acuity Brands)
- Dr. Gabriel Fierro (Brick Consortium, Colorado School of Mines, National Renewable Energy Laboratory)
- Dr. Karl Hammar (RealEstateCore Consortium, Microsoft)
- Dr. Jason Koh (Brick Consortium, Mapped)
- Jim McHale (Memoori Research)
- Dr. Erik Paulson (Brick Consortium)
- Dr. Erik Wallin (RealEstateCore Consortium, ProptechOS)
- Joel Bender (ASHRAE 223P, Cornell University)

Wrapping up

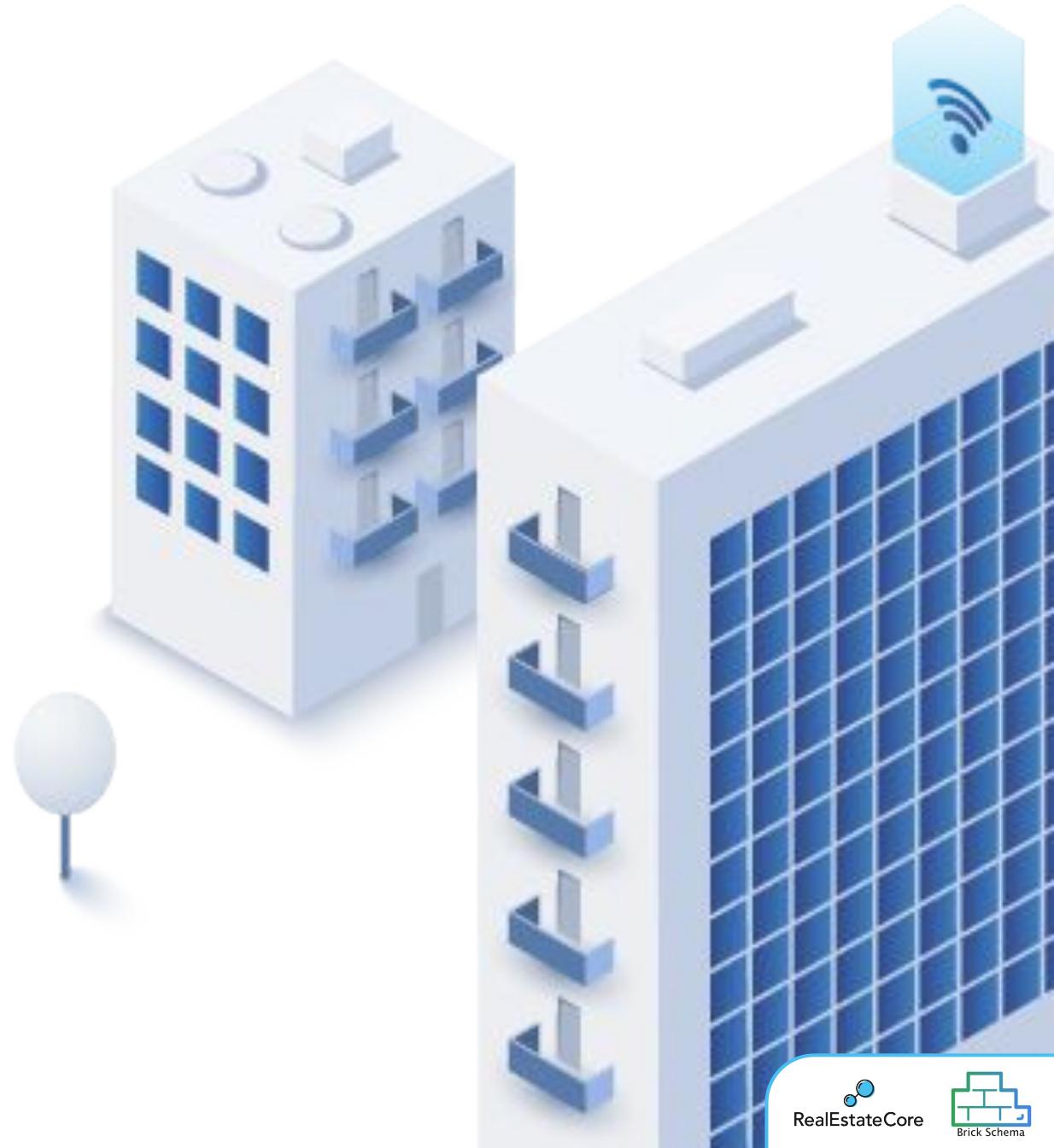
Resources

- <https://brickschema.org/resources>
- <https://github.com/RealEstateCore/rec>

Participate in the community

- <https://groups.google.com/g/brickschema>

The recording will be available after the webinar at
<https://realestatecore.io> and <https://brickschema.org>



Thanks for joining us!

