ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

BrickIoT: Plataforma Interactiva para Monitoreo y Visualización de Sensores IoT en Edificios Inteligentes con Ontología Brick

PROYECTO INTEGRADOR

Previo a la obtención del Título de:

Ingeniería en Telemática

Presentado por:

Kevin Adonis Vargas Benavides

Pier Alejandro Colina Arteaga

GUAYAQUIL - ECUADOR

Año: 2025

DEDICATORIA

Kevin Adonis Vargas Benavides

Dedico este proyecto a mi padre, mi guía y apoyo incondicional desde el primer día. Sé que desde el cielo se enorgullece de este logro, fruto de sus enseñanzas sobre el camino de la rectitud y el honor de llevar su apellido.

Pier Alejandro Colina Arteaga

Dedico este trabajo a mis padres, por su apoyo constante e incondicional y por motivarme a alcanzar mis objetivos académicos. Su esfuerzo y apoyo han sido de gran importancia para lograr esta meta.

AGRADECIMIENTOS

Kevin Adonis Vargas Benavides

Agradezco a Dios por este logro. A mi padre, por su apoyo incondicional y por nunca dudar de mí, siendo mi mayor inspiración. También a mi madre, hermano, profesores y amigos, quienes me acompañaron en este camino. Finalmente, a la ESPOL por su invaluable apoyo en mi formación y proyectos.

Pier Alejandro Colina Arteaga

Agradezco sinceramente a mis profesores del Laboratorio IoT y Sistemás Telemáticos por su guía, paciencia, dedicación y la confianza depositada en mi durante el desarrollo de este proyecto, así como a la universidad por brindarme los recursos y el apoyo necesario para llevarlo a cabo. El compromiso con la excelencia ha sido fundamental para mi aprendizaje y crecimiento profesional.

DECLARACIÓN EXPRESA

”Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; Pier Colina y Kevin Vargas y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual”

Firmado electrónicamente por:
PIER ALEJANDRO COLINA ARTEAGA

Pier Alejandro Colina Arteaga
ESTUDIANTE

Firmado electrónicamente por:
KEVIN ADONIS VARGAS BENAVIDES

Kevin Adonis Vargas Benavides
ESTUDIANTE

EVALUADORES

Maria Isabel Mera Collantes

Time Stamping Security Data

Firmado electrónicamente por:
CHRISTOPHER JAVIER VACCARO CEDILLO

Validar únicamente con FirmaEC

María Isabel Mera Collantes

PROFESORA DE LA MATERIA

Christopher Javier Vaccaro Cedillo

PROFESOR TUTOR

ÍNDICE GENERAL

RESUMEN

La presente investigación propone un esquema fundamentado en la ontología Brick con el propósito de establecer una representación estandarizada y unificada de la infraestructura del campus de la Escuela Superior Politécnica del Litoral (ESPOL). El objetivo principal consiste en enriquecer la descripción semántica de la arquitectura del campus, estableciendo relaciones precisas entre los datos generados por los sensores y sus correspondientes ubicaciones físicas. En este estudio, el esquema propuesto se delimitó al Edificio 11C de la Facultad de Ingeniería en Electricidad y Computación, incorporando laboratorios, áreas exteriores, zonas, equipos y sensores, y evaluando su expresividad en comparación con un modelo equivalente basado en Haystack, actualmente implementado, a fin de optimizar la interoperabilidad y la organización de la información. Adicionalmente, a partir del diccionario estandarizado de Brick, se integraron dispositivos y sensores IoT al esquema con el fin de desarrollar un sistema automatizado capaz de asignar de manera semántica los puntos de datos a sus respectivas ubicaciones físicas dentro del edificio.

Para el desarrollo del esquema se emplearon tecnologías como Python con la biblioteca RDFLib para modelar la infraestructura en RDF y Brick, y Django, React y ReactFlow para la integración, gestión, monitorización y visualización interactiva de los datos. El sistema resultante permitió representar de manera semántica la relación entre sensores y ubicaciones, mejorando la interoperabilidad y reduciendo la variabilidad semántica.

Palabras Clave: Brick, Haystack, RDF, IoT, Monitorización, Django, React, React-Flow

ABSTRACT

This research proposes a schema based on the Brick ontology with the aim of establishing a standardized and unified representation of the infrastructure at the Escuela Superior Politécnica del Litoral (ESPOL) campus. The main objective is to enhance the semantic description of the campus architecture by establishing precise relationships between the data generated by sensors and their corresponding physical locations. In this study, the proposed schema was limited to Building 11C of the Faculty of Electrical and Computer Engineering, incorporating laboratories, outdoor areas, zones, equipment, and sensors, and evaluating its expressiveness in comparison with an equivalent model based on Haystack, currently implemented, in order to optimize interoperability and information organization. Additionally, using Brick's standardized dictionary, IoT devices and sensors were integrated into the schema to develop an automated system capable of semantically mapping data points to their respective physical locations within the building.

For the development of the schema, technologies such as Python with the RDFLib library were used to model the infrastructure in RDF and Brick, while Django, React, and ReactFlow were used for data integration, management, monitoring, and interactive visualization. The resulting system enabled a semantic representation of the relationship between sensors and locations, improving interoperability and reducing semantic variability.

Keywords: Brick, Haystack, RDF, IoT, Monitoring, Django, React, React-Flow

ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Litoral
FIEC	Facultad de Ingeniería en Electricidad y Computación
RDF	Resource Description Framework
OWL	Web Ontology Language
SHACL	Shapes Constraint Language
W3C	World Wide Web Consortium
SPARQL	SPARQL Protocol and RDF Query Language
IoT	Internet Of Things
JSON	JavaScript Object Notation
MQTT	Message Queuing Telemetry Transport
GPS	Global Positioning System
DHT	Humidity and Temperature Sensor
HVAC	Heating, Ventilation and Air Conditioning

ÍNDICE DE FIGURAS

ÍNDICE DE CÓDIGOS

CAPÍTULO 1

1. INTRODUCCIÓN

En la actualidad, la gestión eficiente de la infraestructura en edificios inteligentes, como los de la Escuela Superior Politécnica del Litoral (ESPOL), requiere una integración efectiva de diversas tecnologías, principalmente Internet de las Cosas (IoT). Los sensores distribuidos en diferentes áreas del campus generan grandes cantidades de datos, que deben ser gestionados e interpretados de manera coherente para optimizar el uso de los recursos, el monitoreo ambiental y el control energético. Sin embargo, la integración de estos datos en un sistema único enfrenta desafíos debido a la falta de estandarización en las ontologías utilizadas.

Esta investigación propone como solución la ontología Brick, que se basa en un modelo formal RDF/OWL, para representar de manera unificada y estandarizada la infraestructura del campus de ESPOL. El propósito principal es mejorar la descripción semántica de los datos generados por los sensores IoT y resolver las limitaciones observadas en Haystack, que actualmente se utiliza para describir los activos de los edificios. Se busca adaptar Brick a los espacios específicos del edificio 11C de la Facultad de Ingeniería en Electricidad y Computación. Se asume que el uso de Brick reducirá significativamente la ambigüedad en las descripciones de la infraestructura académica y permitirá una verificación precisa de la información.

El desarrollo del proyecto se estructura en diversas fases, iniciando con la adaptación de la ontología Brick a la infraestructura existente, seguida de la integración de los dispositivos IoT disponibles, con el propósito de garantizar que cada sensor esté correctamente asociado a su ubicación física correspondiente. Los resultados obtenidos permitirán la construcción de un modelo semántico estructurado que optimice la gestión en tiempo real de las infraestructuras del campus y abrir paso para la implementación de tecnologías como los gemelos digitales.

1.1. Descripción de la problemática

En la actualidad, el Laboratorio de IoT y Sistemas Telemáticos de la Escuela Superior Politécnica del Litoral (ESPOL) utiliza una aplicación propia basada en la ontología Project Haystack para describir tanto la estructura física como digital de los edificios e instalaciones del campus. Esta herramienta permite modelar espacios, dispositivos y sistemas de control mediante un sistema de etiquetas (tags), facilitando en cierta medida, la organización, visualización y gestión de los datos generados por la infraestructura. El enfoque orientado a etiquetas ha resultado útil para representar puntos de datos de edificios, contribuyendo a una gestión más eficiente de los recursos tecnológicos.

Sin embargo, Haystack presenta limitaciones significativas que afectan su escalabilidad, interoperabilidad y precisión semántica. Una de sus principales debilidades es la ausencia de un sistema formal y normado para la definición de etiquetas. Haystack presenta un alto índice de personalización de etiquetas y esta falta de estandarización conduce a ambigüedades semánticas, inconsistencias en el modelado y errores de interpretación cuando diferentes equipos o desarrolladores aplican esquemas similares sin una base en común. En entornos de gran escala, esto puede derivar en problemas de interoperabilidad entre sistemas y reducir la confiabilidad del modelo de datos.

Adicionalmente, Haystack carece de una estructura ontológica formal basada en tecnologías semánticas como RDF (Resource Description Framework) u OWL (Web Ontology Language), lo que limita su capacidad para integrarse con otras ontologías del dominio o con sistemas externos que utilicen estos estándares. Esta carencia también dificulta la validación del esquema y de los datos que se generan a partir de él, lo que representa una desventaja importante en contextos donde se requiere trazabilidad y análisis de la información generada.

Ante estas limitaciones, la ontología Brick emerge como una alternativa más sólida y estandarizada para la representación semántica de edificios inteligentes y su infraestructura asociada. Brick está fundamentada en los principios del modelado semántico formal y utiliza los estándares RDF y OWL, ampliamente adoptados por la comunidad. A diferencia de Haystack, Brick ofrece un vocabulario estructurado, extensible y verificable, que define de forma explícita las clases, propiedades y relaciones entre entidades como edificios, zonas, laboratorios, sistemas, equipos, sensores, puntos de

medición y otras entidades relevantes en entornos estructurales.

Gracias a esto, Brick permite la interoperabilidad con otras ontologías y bases de conocimiento, así como la ejecución de consultas avanzadas mediante lenguajes como SPARQL. Además, facilita la validación del esquema y la semántica del modelo, la automatización de tareas, y la trazabilidad de los dispositivos en relación con su ubicación física y su función dentro del sistema general.

1.2. Justificación

La justificación de este proyecto radica en las deficiencias observadas al emplear Haystack, ya que la falta de normas formales en el etiquetado genera variabilidad semántica, ambigüedad en la interpretación de los datos y problemas de integración o validación de esquemas con sistemas externos. Estas deficiencias afectan notablemente la gestión automatizada de los recursos energéticos y el monitoreo de las condiciones ambientales de los laboratorios y sus alrededores, ya que no se pueden implementar consultas integradoras que utilicen información proveniente de múltiples dispositivos. En contraste, Brick proporciona un diccionario estándar y verificable que valida las relaciones entre edificios, laboratorios, zonas, sensores y actuadores, mejorando la semántica y la trazabilidad de la información.

Además, la estandarización proporcionada por Brick facilita la escalabilidad de la solución: al utilizar un esquema RDF/OWL, es posible agregar nuevos nodos (como sensores adicionales o módulos de automatización) sin ambigüedades, lo que permite que el grafo semántico se expanda sin necesidad de remodelar las ontologías existentes. Este enfoque es particularmente relevante para la transformación del campus de la ESPOL en un entorno inteligente y sostenible, en línea con los Objetivos de Desarrollo Sostenible 9 (Industria, Innovación e Infraestructura), 11 (Ciudades y Comunidades Sostenibles) y 13 (Acción por el Clima).

En el contexto del campus de la ESPOL, la adopción de Brick representa una oportunidad para evolucionar hacia un modelo de gestión de infraestructura más coherente, interoperable y alineado con las tendencias actuales en modelado semántico de entornos inteligentes. La capacidad de implementar gemelos digitales, realizar consultas SPARQL uniformes y tomar decisiones basadas en datos integrados refuerza la justificación para la migración de Haystack a Brick, facilitando la representación y el monitoreo en tiempo real de la infraestructura.

1.3. Objetivos

Objetivo General:

Desarrollar un modelo de representación basado en la ontología Brick y tecnologías IoT que permita describir de manera estructurada los aspectos físicos, lógicos y virtuales del campus de ESPOL, para facilitar su monitoreo, gestión y escalabilidad para futuras tecnologías.

Objetivos Específicos:

- Diseñar un modelo semántico utilizando la ontología Brick, aplicado a espacios representativos del campus de ESPOL, con especial énfasis en el Edificio 11C y sus laboratorios.
- Integrar los dispositivos y sensores IoT actualmente desplegados en los laboratorios al modelo propuesto, estableciendo relaciones claras entre los datos generados y su ubicación física.
- Identificar las limitaciones del modelo actual basado en Haystack, considerando criterios como interoperabilidad, escalabilidad, flexibilidad semántica y eficiencia en la gestión de datos.
- Validar el modelo desarrollado en un entorno controlado, específicamente en el Laboratorio de IoT y Sistemas Telemáticos, utilizando sensores reales (temperatura, energía, entre otros) para asegurar su aplicabilidad práctica.
- Evaluar el potencial del modelo Brick como base para un gemelo digital del campus, analizando su utilidad en tareas de monitoreo, simulación y gestión inteligente de la infraestructura universitaria.

Hipótesis

Se plantea como hipótesis que la adopción de la ontología Brick, basada en tecnologías semánticas como RDF y OWL, permitirá reducir de manera significativa la ambigüedad presente en los conceptos empleados para etiquetar y describir espacios físicos, dispositivos y sistemas dentro del campus de la ESPOL. Esta característica superará las limitaciones de la ontología Haystack, actualmente utilizada en el Laboratorio de IoT y Sistemas Telemáticos, la cual presenta dificultades en términos de consistencia semántica y capacidad de razonamiento automático.

La reducción de ambigüedad facilitará la interoperabilidad de los datos generados por sistemas y equipos heterogéneos, al proporcionar un esquema unificado, extensible y verificable. Esto promoverá la integración efectiva de información de sensores y mejorará la calidad y verificabilidad de las descripciones mediante la validación formal de modelos, que permitirá detectar inconsistencias o redundancias.

Se espera que la adopción de Brick posibilitará una escalabilidad natural para incorporar nuevas tecnologías y dispositivos sin perder coherencia, y habilitará procesos automatizados avanzados y el control eficiente de recursos a través de razonamiento automático y consultas complejas.

En conjunto, estos beneficios contribuirán a una gestión más precisa e integrada de la infraestructura académica y tecnológica del campus, que permitirá una descripción coherente, verificable y reutilizable, para mejorar la interoperabilidad de los sistemas y la calidad del análisis de datos en el entorno universitario.

1.4. Marco Teórico

1.4.1. Ontología

En la rama de la informáttica, las ontologías son representaciones formales de conocimiento que permiten estructurar información en un dominio específico mediante clases, propiedades y relaciones entre conceptos. En el contexto de edificios inteligentes y sistemas ciberfísicos, las ontologías permiten una semántica compartida entre dispositivos, aplicaciones y usuarios, facilitando la interoperabilidad, integración de datos heterogéneos y automatización de procesos de gestión.

Dos de las ontologías más relevantes en esta investigación son Brick Schema y Project Haystack. Brick es una ontología basada en OWL (Web Ontology Language) diseñada específicamente para describir la semántica de los sistemas de gestión de edificios. Define equipos, sensores, espacios, relaciones funcionales y espaciales entre estos elementos. Su diseño permite consultas complejas y consistentes a través de grafos RDF, lo que resulta útil para construir aplicaciones de monitoreo, análisis energético y gemelos digitales.

Project Haystack, por otro lado, utiliza un modelo más ligero basado en etiquetas (tags) para describir dispositivos y puntos de datos. Aunque es más simple de implementar, su falta de semántica formal limita su capacidad de razonamiento y consultas avanzadas. Sin embargo, es ampliamente adoptado en la industria.

1.4.2. Modelado Semántico y Tecnologías RDF/OWL

El modelado semántico es un enfoque para representar datos de manera estructurada y comprensible tanto para humanos como para máquinas, permitiendo que la información se relacione con significado dentro de un contexto específico. Este tipo de modelado es fundamental para la interoperabilidad entre sistemas heterogéneos, especialmente en aplicaciones del IoT, donde la diversidad de dispositivos y datos puede dificultar su integración y análisis.

Entre las tecnologías clave del modelado semántico se encuentran RDF (Resource Description Framework) y OWL (Web Ontology Language). RDF proporciona un marco estándar para describir recursos en la web mediante tripletas del tipo

sujeto-predicado-objeto, lo que permite establecer relaciones entre entidades. OWL, por su parte, extiende RDF y permite definir clases, propiedades, relaciones jerárquicas, restricciones y axiomas lógicos, proporcionando mayor expresividad para representar conocimiento.

Estas tecnologías permiten la construcción de ontologías: esquemas formales que describen un dominio específico (por ejemplo, la infraestructura de un edificio o campus), lo cual facilita la inferencia automática, validación de datos y búsqueda semántica. En contextos de IoT, el uso de RDF y OWL ayuda a mapear sensores, dispositivos, ubicaciones físicas y tipos de datos, haciendo posible la automatización de tareas como la identificación de sensores mal conectados o el análisis de patrones de consumo energético.

El valor del modelado semántico radica en su capacidad para estandarizar y enlazar datos provenientes de diversas fuentes, permitiendo su uso en sistemas complejos, visualizaciones dinámicas y procesos automatizados. Además, al estar basado en estándares del W3C, garantiza compatibilidad a largo plazo y la posibilidad de integración con otras plataformas semánticas, como SPARQL para consultas o SHACL para validación de grafos.

En este proyecto, el uso de RDF y OWL permite representar de manera unificada los sensores IoT, sus ubicaciones físicas y los datos que generan, sirviendo como base para la visualización, monitoreo y gestión inteligente del entorno construido.

1.4.3. Project Haystack

Project Haystack es una iniciativa de código abierto diseñada para estandarizar la semántica de los datos generados por dispositivos y sistemas en edificios inteligentes. Su principal objetivo es proporcionar un modelo de etiquetas (tags) y taxonomías que facilite la interpretación y reutilización de datos provenientes de sensores y sistemas de gestión de edificios. Al emplear un enfoque basado en metadatos y etiquetas estructuradas, Haystack permite a las aplicaciones entender el contexto y la funcionalidad de los datos sin necesidad de complejos procesos de integración manual.

En el contexto del campus ESPOL, Haystack ha sido adoptado como solución semántica para describir parte de la infraestructura IoT desplegada, especialmente en laboratorios y zonas académicas. No obstante, se han identificado limitaciones

importantes. Por ejemplo, la ausencia de una ontología formal dificulta la interoperabilidad con otras fuentes de datos externas o sistemas de análisis semántico más avanzados. Asimismo, la falta de una estructura jerárquica rigurosa puede dificultar el escalamiento del modelo conforme se incrementa la complejidad de los dispositivos y las relaciones entre ellos.

En comparación con Brick Schema, Haystack resulta más fácil de implementar inicialmente, pero su modelo puede volverse ambigo en escenarios que requieren una representación precisa y extensible del edificio como un sistema complejo. Esta comparación es esencial para evaluar la transición hacia ontologías más robustas como Brick, especialmente si se busca implementar gemelos digitales que demandan modelos de datos ricos y formalmente estructurados.

1.4.4. BrickSchema

BrickSchema es una ontología abierta diseñada para describir la infraestructura de edificios inteligentes, modelando espacios físicos, equipos, sistemas de control y puntos de datos de sensores en un formato semántico unificado. Desarrollado sobre tecnologías como RDF y OWL, Brick permite representar con precisión las relaciones entre diferentes componentes de un edificio, facilitando así la interoperabilidad, la automatización y el análisis de sistemas complejos.

Uno de los principales beneficios de Brick es su vocabulario estandarizado. Este incluye clases como Room, HVAC, VAV, Temperature Sensor, entre muchas otras, y relaciones como hasPoint, feeds, hasPart, que describen cómo se interconectan los componentes del sistema físico. De esta manera, es posible, por ejemplo, describir que una sala específica contiene un sistema HVAC que a su vez posee un sensor de temperatura cuyo valor puede ser consultado.

En el contexto de IoT, Brick se convierte en una herramienta esencial para mapear sensores físicos con su representación semántica, lo que facilita la integración con motores de inferencia y herramientas de visualización. Por ejemplo, mediante una consulta SPARQL, es posible recuperar todos los sensores de temperatura de un edificio junto con las habitaciones a las que pertenecen, sin necesidad de conocer la estructura física exacta de cada instalación.

Brick también fomenta la reutilización de modelos y la escalabilidad de soluciones.

Una vez construido un modelo semántico de un edificio o campus, este puede ampliarse o modificarse fácilmente para incluir nuevos dispositivos o espacios, manteniendo la coherencia del sistema.

En este proyecto, Brick se utiliza como base para representar el esquema de infraestructura del campus universitario, integrando sensores IoT en su estructura semántica. Esto permite no solo el monitoreo en tiempo real, sino también consultas avanzadas y automatización de tareas de mantenimiento y supervisión energética.

1.5. Estado del Arte

Los edificios inteligentes contemporáneos se configuran como ecosistemas digitales de alta complejidad, donde la interoperabilidad armónica de tecnologías diversas resulta imperativa para maximizar el rendimiento, la eficiencia energética y la experiencia del usuario. No obstante, la praxis revela una fragmentación persistente, con sistemas operando en aislamiento y manipulando datos propietarios, lo que impone barreras insalvables a la integración. En este contexto, las ontologías se erigen como la herramienta paradigmática para erradicar esta disgregación, fungiendo como un lenguaje semántico universal que impone la cohesión entre sistemas heterogéneos. Sin embargo, la proliferación de estándares ontológicos ha forjado un paisaje semántico fracturado, que, si bien genera oportunidades para la innovación, impone desafíos críticos en términos de estandarización y adopción industrial.

1.5.1. Project Haystack

Project Haystack se presenta como una propuesta práctica para la gestión de datos en edificios inteligentes, con un fuerte enfoque en sistemas de calefacción, ventilación y aire acondicionado. Su diseño está orientado a la simplicidad y a una implementación rápida, dejando de lado abstracciones complejas, lo que lo ha convertido en un estándar ampliamente aceptado en la industria. Basado en un modelo de etiquetas (tags), Haystack permite definir entidades como sensores, actuadores o puntos de control mediante metadatos flexibles, lo que facilita implementaciones adaptables a distintos entornos.

Sin embargo, esta flexibilidad trae consigo limitaciones importantes. La estructura de etiquetado libre de Haystack, aunque práctica, puede generar ambigüedades semánticas debido a la falta de definiciones formales. Por ejemplo, el término *temp sensor* puede interpretarse de manera diferente según el implementador, lo que afecta la interoperabilidad entre sistemas. Esta situación es comparable a un dialecto de un lenguaje hablado, funciona bien en contextos reducidos, pero tiende a generar confusión cuando se amplía la escala.

La arquitectura basada en entidades y etiquetas se asemeja a la organización de una biblioteca: cada elemento tiene su lugar, pero su uso depende de conocer el sistema de clasificación interno. Esta facilidad ha hecho que Haystack sea la opción favorita

para integradores, quienes priorizan la practicidad frente a soluciones más complejas. No obstante, su popularidad ha creado cierta resistencia al cambio, ya que las inversiones en herramientas y sistemas propios dificultan la transición hacia modelos semánticos más completos. La falta de una semántica formal, aunque suficiente para aplicaciones específicas, limita su capacidad para soportar razonamientos automáticos avanzados o integraciones a gran escala, algo parecido a una ciudad construida sin planos detallados.

1.5.2. BrickSchema

Por otro lado, BrickSchema propone un enfoque innovador que prioriza la claridad semántica y una estructura bien definida. Basado en estándares del W3C como RDF y OWL, Brick ofrece una forma organizada y formal de representar metadatos en edificios, lo que permite realizar inferencias automáticas y análisis avanzados. A diferencia del crecimiento más flexible de Haystack, Brick se construye sobre principios sólidos, similar a comparar un sistema GPS con un mapa básico; mucho más preciso, aunque menos intuitivo para quienes no son expertos.

El uso de RDF y OWL le permite a Brick modelar relaciones complejas entre equipos, espacios y sistemas con un nivel de detalle que supera ampliamente a Haystack. Por ejemplo, Brick puede inferir automáticamente la relación entre un sensor de temperatura y el sistema HVAC correspondiente, facilitando aplicaciones como el mantenimiento predictivo o la optimización energética. Sin embargo, esta potencia tiene un costo, su implementación requiere conocimientos avanzados en tecnologías semánticas, lo que crea una barrera significativa para su adopción masiva en la industria.

Brick, aunque ofrece una solución técnicamente superior, su aplicación práctica sigue siendo limitada. Mientras Haystack puede ser implementado por integradores con conocimientos básicos de bases de datos, Brick exige experiencia en ontologías y razonamiento semántico, habilidades poco comunes en la automatización de edificios. Además, resolver errores en un sistema basado en Brick es mucho más complicado que en Haystack, ya que implica revisar procesos de inferencia semántica y no solo etiquetas mal asignadas. En resumen Brick tiene un potencial enorme, pero para aprovecharlo se necesita un cambio profundo en cultura y capacitación técnica.

En conclusión, Haystack y Brick representan dos caminos diferentes para resolver los retos de interoperabilidad en edificios inteligentes. Haystack ha logrado un gran alcance

gracias a su simplicidad, mientras Brick ofrece una visión más completa y poderosa, aunque con barreras importantes para su implementación. Esta convivencia refleja la tensión constante entre soluciones prácticas y la innovación tecnológica.

1.5.3. Tecnologías Semánticas en IoT

Las tecnologías semánticas han emergido como un elemento fundamental para dotar a los sistemas IoT de interoperabilidad, representación del conocimiento y capacidades avanzadas de automatización en entornos inteligentes. Entre los pilares de estas tecnologías se encuentran RDF y OWL, los cuales proporcionan mecanismos robustos para estructurar, enlazar y razonar sobre datos distribuidos.

RDF permite modelar la información en forma de tripletas (sujeto-predicado-objeto), lo que facilita la integración y consulta de datos heterogéneos provenientes de múltiples fuentes. Por su parte, OWL ofrece un marco formal para la construcción de ontologías con alta expresividad, estableciendo axiomas y restricciones que habilitan inferencias lógicas sobre los datos, incrementando así el nivel de inteligencia de las aplicaciones IoT.

Un componente clave en este ecosistema es SPARQL, el lenguaje estándar para la consulta de datos RDF. Su implementación en infraestructuras inteligentes permite ejecutar consultas complejas en tiempo real sobre repositorios distribuidos, lo que habilita procesos como la gestión automatizada de dispositivos, el control dinámico de recursos y la toma de decisiones basada en reglas y ontologías.

Otro concepto relevante es el de gemelos digitales, que constituyen representaciones virtuales de edificios, sistemas o infraestructuras completas. Estos gemelos digitales, al incorporar ontologías semánticas, permiten modelar no solo los componentes físicos, sino también sus relaciones, comportamientos y contextos operativos. Gracias a ello, se posibilitan simulaciones precisas, análisis predictivos y estrategias de mantenimiento proactivo, optimizando la eficiencia y reduciendo costos.

Asimismo, la interoperabilidad constituye un desafío central en IoT debido a la diversidad de protocolos, fabricantes y plataformas. Las ontologías actúan como un puente semántico que permite la integración efectiva entre sistemas heterogéneos, asegurando la reutilización de datos y promoviendo la comunicación fluida entre dispositivos. Esto favorece la construcción de ecosistemas colaborativos y escalables, condición indispensable para la evolución hacia infraestructuras inteligentes

verdaderamente autónomas.

En conclusión, la adopción de tecnologías semánticas en IoT, aunque aún en proceso de consolidación, se perfila como un pilar indispensable para el desarrollo de entornos inteligentes que no solo recolecten datos, sino que los comprendan y utilicen para generar conocimiento, anticipar comportamientos y mejorar la toma de decisiones.

1.5.4. Trabajos Relacionados

Diversos estudios y proyectos han explorado la aplicación de soluciones ontológicas y tecnologías semánticas en el contexto de la automatización y monitoreo de edificios inteligentes y campus universitarios. Entre ellos, destaca el uso de la ontología Brick en entornos académicos, donde se ha demostrado su capacidad para proporcionar una gestión centralizada de la información, monitoreo en tiempo real e integración con sistemas complementarios como HVAC, iluminación y control de acceso. Estos experimentos confirman que Brick facilita la modelación detallada de entidades y relaciones en un edificio, mejorando la interoperabilidad y reduciendo la dependencia de soluciones propietarias.

Brick ofrece una expresividad semántica superior y soporta capacidades de inferencia que permiten automatizar procesos complejos, como la detección de inconsistencias o la optimización de recursos. Sin embargo, esta sofisticación implica una curva de aprendizaje pronunciada y la necesidad de contar con personal especializado en ontologías y tecnologías semánticas. Por el contrario, Haystack, basado en un sistema flexible de etiquetas, resulta más sencillo de implementar, lo que ha favorecido su adopción masiva en la industria, aunque con limitaciones significativas en interoperabilidad y razonamiento automático.

Adicionalmente, se han desarrollado arquitecturas orientadas a la integración IoT-Ontologías que proponen marcos híbridos combinando enfoques pragmáticos y formales. Estos modelos buscan equilibrar la facilidad de implementación con la expresividad semántica necesaria para soportar aplicaciones avanzadas como gemelos digitales, análisis predictivo y control autónomo de infraestructuras. Algunos trabajos incluso plantean la extensión de vocabularios y la adopción de estándares abiertos como RDF y OWL para garantizar escalabilidad y compatibilidad entre plataformas.

En síntesis, la literatura revisada coincide en señalar el enorme potencial

transformador de las ontologías para la evolución de sistemas de automatización y gestión de edificios inteligentes. No obstante, también advierte sobre desafíos críticos como la capacitación de personal, la integración con sistemas heredados y la necesidad de herramientas que simplifiquen la implementación sin sacrificar interoperabilidad ni capacidades de inferencia.

CAPÍTULO 2

2. METODOLOGÍA Y DISEÑO DEL SISTEMA

2.1. Materiales

En el desarrollo de este proyecto se emplearon componentes de hardware y software cuidadosamente seleccionados para garantizar una integración óptima entre la ontología Brick y las tecnologías IoT en el contexto del campus de la ESPOL. Estos recursos facilitan la captura y el procesamiento de datos en tiempo real y aseguran la escalabilidad, la interoperabilidad y la robustez del sistema propuesto. Esta infraestructura sirve como base para demostrar las ventajas que ofrece el modelo semántico Brick frente a enfoques tradicionales, al proporcionar un marco estandarizado que mejora la gestión de dispositivos, la trazabilidad de la información y la capacidad de realizar inferencias para optimizar la operación de los edificios inteligentes.

2.1.1. Componentes de Hardware

Los componentes de hardware del sistema se basa en una red de sensores y dispositivos IoT distribuidos en el Edificio 11C de la Facultad de Ingeniería en Electricidad y Computación, que abarca el Laboratorio IoT y Sistemas Telemáticos, el Laboratorio de Sistemas en la Nube y el Laboratorio de Redes de Datos. Esto permite adquisición continua de datos en tiempo real sobre variables ambientales, energéticas y de operación.

En cuanto a la medición de parámetros ambientales, se utilizan sensores diseñados para registrar condiciones que afectan la calidad y estabilidad del entorno interno y externo de los laboratorios. Entre estos destacan los sensores de temperatura, implementados principalmente mediante módulos DHT. Integran la función de medición de humedad relativa, asegurando el control de la climatización para la protección de

equipos y experimentos.

Para el análisis energético, cuenta con dispositivos especializados, como los medidores trifásicos Shelly EM3, que permiten registrar los valores de voltaje, corriente y potencia en cada una de las tres fases (A, B y C). Estos equipos ofrecen información en tiempo real, lo que posibilita la identificación de patrones de consumo, detección de desequilibrios en la carga y la generación de alertas ante variaciones.

Figura 2.1: Equipo de Estación Meteorológica frente al Laboratorio IoT y Sistemas Telemáticos.

Estos dispositivos, como la estación meteorológica mostrada en la Figura 2.1 soportan conectividad mediante Wi-Fi, LoRa o Ethernet, garantizando un canal de comunicación estable y seguro. La transmisión de datos se realiza empleando protocolos ligeros como MQTT, que optimizan el consumo de ancho de banda y energía. En conjunto, esta infraestructura de hardware permite la captura confiable de datos, y constituye la base para la integración con la capa semántica y el sistema de almacenamiento.

2.1.2. Componentes de Software

El software implementado en este proyecto está diseñado para garantizar la interoperabilidad entre la ontología Brick, la infraestructura IoT y las aplicaciones orientadas a la gestión inteligente de edificios dentro del campus de ESPOL. Para ello, se seleccionaron herramientas y tecnologías que permiten un flujo completo de adquisición, almacenamiento, procesamiento y visualización de datos en tiempo real, manteniendo los estándares de escalabilidad y compatibilidad semántica.

En la capa lógica del servidor, el proyecto se fundamenta en Python 3.x como lenguaje principal, dada su adopción de librerias para la manipulación de grafos RDF, entornos de análisis de datos y desarrollo backend. Sobre esta base se integra Django, un framework robusto y modular que facilita la creación de APIs RESTful para la comunicación entre el sistema IoT y los servicios semánticos. Django permite estructurar de forma clara la lógica del sistema, garantizando seguridad y capacidad de integración con librerías externas.

Para la gestión semántica de datos, se utilizan herramientas orientadas a RDF (Resource Description Framework). La librería RDFLib es la base de toda la lógica de backend, donde se emplea para crear, manipular y serializar grafos RDF, elemento esencial para modelar la información siguiendo los estándares de la ontología Brick. A su vez, SPARQLWrapper proporciona una interfaz eficiente para la ejecución de consultas SPARQL, lo que permite recuperar y relacionar datos semánticos de forma flexible y optimizada.

En cuanto al almacenamiento y consulta de grafos RDF, se implementa Apache Jena/Fuseki, una solución ampliamente utilizada en aplicaciones para la persistencia de datos estructurados RDF. Este triplestore no solo soporta consultas SPARQL, sino que también garantiza integridad semántica y escalabilidad en la gestión de grandes volúmenes de datos.

Para los datos dinámicos provenientes de sensores IoT, se incorpora MongoDB como base de datos no relacional, debido a su capacidad para manejar documentos JSON y procesar lecturas en tiempo real con baja latencia. Esta separación entre el repositorio semántico y el repositorio operativo de los datos permite optimizar las consultas históricas de datos, y se complementan manteniendo la semántica dentro del esquema para las relaciones entre los equipos y su identificación dentro del repositorio de datos.

En la capa de presentación, se emplea React.js, una librería moderna de JavaScript

orientada a la creación de interfaces reactivas y modulares. Sobre esta base se integra ReactFlow, una herramienta especializada en la visualización interactiva de grafos, que permite representar de la manera más adecuada la estructura de la ontología Brick y las relaciones entre dispositivos, sensores y espacios físicos del campus. Esta combinación mejora la experiencia del usuario al proporcionar interacciones dinámicas y una visualización más intuitiva del sistema.

Esta arquitectura software soporta la implementación de Brick en un escenario real y sus ventajas frente a enfoques tradicionales, al combinar principios de la web semántica con tecnologías modernas de IoT y desarrollo web. Esta integración asegura un sistema flexible, escalable y alineado con los estándares de edificos inteligentes.

2.2. Metodología

La metodología adoptada en este proyecto se fundamenta en un enfoque orientado a la adaptación y aplicación de la ontología Brick en el contexto específico del campus ESPOL, con el objetivo de garantizar una representación semántica precisa y consistente de los dispositivos y equipos. Este enfoque se orienta a la integración de datos provenientes de dispositivos IoT propios de los espacios del campus, asegurando su correspondencia con las clases y relaciones definidas en el esquema ontológico. Para alcanzar este propósito, se han definido fases claramente estructuradas que incluyen el diseño conceptual del modelo, la implementación técnica mediante herramientas compatibles con Brick y la evaluación del desempeño y la interoperabilidad lograda en escenarios reales del campus.

Además, la metodología contempla la identificación de los distintos subsistemas presentes en el entorno, como laboratorios, aulas, oficinas y equipos de medición ambiental, estableciendo un mapa detallado de dispositivos y sus características. Una vez definida esta estructura conceptual, se procede a la normalización de los datos y a la verificación de la compatibilidad con los estándares definidos por Brick, lo cual permite reducir ambigüedades y asegurar la consistencia en la representación.

La implementación técnica incluye el empleo de plataformas de gestión de datos que permiten tanto la carga como la consulta de información semántica, aprovechando la flexibilidad de las propiedades de las clases del diccionario de Brick para enlazar datos. Finalmente, la fase de evaluación implica la validación de la interoperabilidad del sistema frente a escenarios de uso reales, verificando la eficiencia del modelo en tareas como la monitorización de condiciones ambientales, la administración de recursos y la optimización de procesos internos.

De esta manera, la metodología busca comprobar la conformidad del esquema implementado y evidenciar los beneficios prácticos de la aplicación de Brick en la gestión inteligente de la ESPOL, destacando la mejora en la calidad del análisis de datos y la posibilidad de escalar la solución hacia otros entornos con requerimientos similares.

2.3. Diseño de la Arquitectura del Sistema

La arquitectura propuesta se encuentra organizada en un modelo de múltiples capa que prioriza la modularidad, y asegura la escalabilidad y robustez en todo el sistema. En la capa de adquisición, los sensores IoT desplegados en el Edificio 11C realizan la captura de datos crudos relacionados con variables ambientales, energéticas y de operación. La capa de middleware, desempeña un papel fundamental al encargarse de la normalización y transformación de los paquetes de datos, asegurando su consistencia y preparación para el almacenamiento.

La capa de almacenamiento se compone de dos repositorios que son complementarios: una base de datos documental en MongoDB, que permite el almacenamiento eficiente de lecturas de sensores y equipos en formato JSON para consultas rápidas y flexibles; y por otro, un grafo RDF gestionado mediante Apache Jena Fuseki, que mantiene la estructura semántica de las entidades conforme al modelo Brick Schema, garantizando la interoperabilidad y la vinculación semántica entre dispositivos, ubicaciones y atributos.

La capa de procesamiento semántico aplica mecanismos de inferencia basados en RDF/OWL, lo que habilita la generación de conocimiento derivado a partir de las relaciones en el grafo, mejorando la capacidad de análisis y trazabilidad. Finalmente, la capa de aplicación y visualización proporciona interfaces interactivas de flujo para usuarios y administradores, permitiendo el monitoreo en tiempo real, la consulta de históricos y la validación de inferencias, todo ello a través de nodos dinámicos y consultas SPARQL.

Esta arquitectura de multiples capas optimiza la organización funcional del sistema y ofrece ventajas sobre otras soluciones en términos de escalabilidad, mantenimiento y eficiencia, asegurando la capacidad de integrar nuevos dispositivos, servicios y reglas semánticas sin comprometer la estabilidad del sistema.

2.3.1. Actualización de Firmware

Uno de los aspectos esenciales durante la implementación del esquema semántico fue la capacidad de mantener la información actualizada frente a cambios en la infraestructura física o en el estado de los dispositivos. Para esto, se realizó la prueba de actualización

de firmware en dispositivos IoT desplegados en los laboratorios del edificio 11C.

El objetivo principal de esta actualización fue habilitar la comunicación mediante el protocolo MQTT hacia un broker centralizado alojado en los servidores del laboratorio, garantizando un canal de transmisión confiable para la recolección de datos. Otro aspecto clave que se buscaba con esta actualización fue que, tras la actualización del firmware, el dispositivo mantuviera su identificación y asociación correcta dentro del modelo semántico, evitando la pérdida de consistencia semántica en la representación de las entidades. Asimismo, se buscó garantizar que las instancias de dispositivos en el grafo RDF continuaran vinculadas a sus respectivas ubicaciones y sensores, asegurando así su trazabilidad.

airClimate.ino

```
/* DEVICE SPECIFIC VARIABLES
 *
 * These variable are meant to be changed for every device
 */
#define DHTPIN 4

#define DHTTYPE DHT22 // (DHT11/DHT22) commonly used.

#define WAKEUP_TIME_SEC 180 // Change time if needed

#define WIFI_SSID "ENTER-SSID"

#define WIFI_PASSWD "ENTER-PASSWD"

#define HAYIOT_SENSOR_ID "HAYIOT-UNIQUE-ID"

#define MQTT_TOPIC "@LAB/sensores/@CLIENT-ID" //change the variables @LAB and @CLIENT-ID accordingly

#define MQTT_CLIENT_ID "@CLIENT-ID" // change accordingly

/*END*/

DHT dht(DHTPIN, DHTTYPE);
```

Figura 2.2: Variables de configuración en el Firmware para dispositivos IoT basados en ESP32/ESP8266.

En la figura 2.2 se observa un fragmento del firmware para los dispositivo IoT, específicamente en los sensores de calidad de aire y sensores de temperatura, donde se definen variables específicas para la configuración de cada dispositivo. Se definen identificadores cruciales para la integración semántica y la comunicación mediante MQTT, como MQTT_TOPIC, que establece la ruta del tema en el broker para publicar los datos, y el MQTT_CLIENT_ID, que identifica al cliente único en el sistema de mensajería. Estas definiciones permiten personalizar la configuración del dispositivo asegurando su conexión y trazabilidad en la arquitectura semántica y de mensajería. Durante el proceso se monitorearon los siguientes indicadores:

- **Continuidad en la transmisión de datos**: Comprobar que el flujo de datos desde

el dispositivo hacia la base de datos.

- **Integridad del modelo**: Asegurar que la actualización no generara duplicidad ni inconsistencias en las relaciones definidas en la ontología Brick.

2.3.2. Recolección, Transformación y Almacenamiento de Datos

Para garantizar la integridad y estandarización de la información proveniente de los dispositivos IoT, se automatizó el proceso de adquisición y tratamiento de datos mediante el uso de Node-RED, que permite transformar los paquetes recibidos desde los dispositivos en un formato estandarizado y verificable antes de su almacenamiento. Este flujo asegura la consistencia de los datos y facilita su vinculación con las entidades definidas en el esquema Brick.

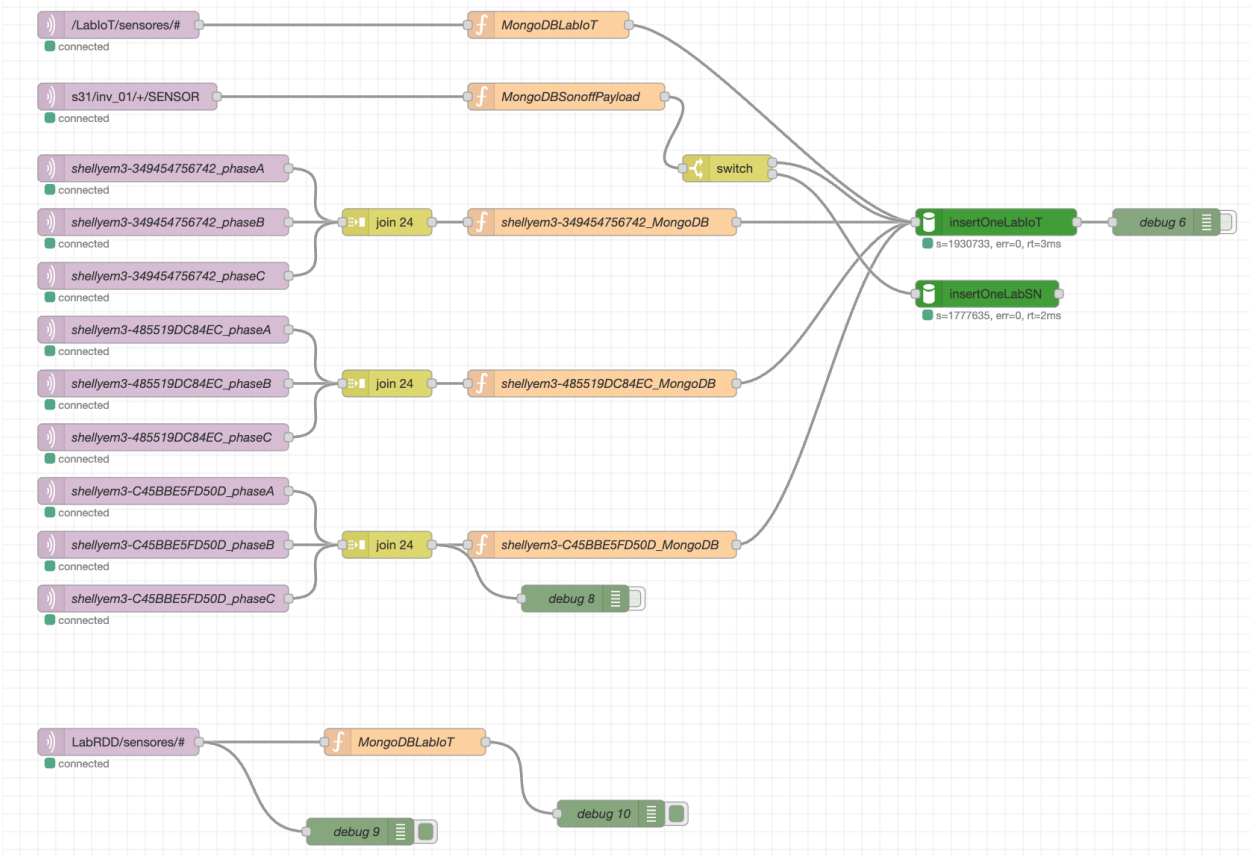

Figura 2.3: Flujo de transformación y almacenamiento de datos en Node-RED.

El flujo de trabajo implementado se muestra en la Figura 2.3 y se compone de las siguientes etapas:

1. **Suscripción a tópicos MQTT:** Los nodos iniciales del flujo están dedicados a escuchar los tópicos publicados por los dispositivos. Estos tópicos siguen una

jerarquía definida en su firmware que permite organizar los datos por laboratorio, dispositivo y tipo de medición. Por ejemplo, se reciben datos de sensores DHT (temperatura y humedad) y de analizadores de energía trifásicos (Shelly EM3 o Accuenergy), los cuales publican valores asociados a las fases A, B y C.

2. **Agrupación de datos:** En el caso de algunos dispositivos IoT, como los medidores trifásicos, se utilizan nodos join para combinar en un solo objeto JSON las mediciones de las tres fases, dado que publican sus mediciones en diferentes tópicos. Esto garantiza que los datos relacionados se procesen de manera conjunta, reduciendo inconsistencias en la base de datos y facilitando el análisis de los datos.

3. **Transformación y normalización:** Una vez agrupados, los mensajes pasan por nodos de tipo `function`, donde se ejecutan scripts que estandarizan los nombres de las variables, añaden metadatos relevantes, como el identificador único del sensor, timestamp, ubicación semántica, y validan que los valores cumplan con las unidades y rangos esperados. Este paso es crítico para garantizar la interoperabilidad con el modelo basado en Brick, dado que estos campos incluyen referencias a las URIs correspondientes en el grafo RDF, garantizando la asociación directa entre con dispositivo físico.

4. **Almacenamiento en base de datos:** Finalmente, los datos transformados se insertan en MongoDB mediante nodos de NodeRed denominados `insertOne`, específicos para el driver de MongoDB. Cada inserción se confirma en el flujo, mostrando el estado (éxito, error, tiempo de respuesta) en nodos de debug, lo cual facilita el monitoreo en tiempo real y la detección temprana de fallas. Los datos procesados fueron almacenados en una base de datos MongoDB desplegada en los servidores del laboratorio. La elección de MongoDB responde a su flexibilidad para trabajar con estructuras de datos no relacionales y a la posibilidad de emplear pipelines de agregación para consultas avanzadas, lo cual resulta especialmente útil para la consulta de datos IoT.

2.3.3. Modelado Semántico con Brick

El modelado ontológico es la base para garantizar la interoperabilidad, escalabilidad y automatización en entornos inteligentes de este proyecto. Brick, como una ontología de código abierto, ofrece un vocabulario estandarizado para describir edificios, laboratorios, zonas, sistemas, equipos, sensores, etc. permitiendo una representación uniforme y semánticamente rica de la infraestructura del campus. El modelado ontológico no solo consistió en instanciar entidades y relaciones, sino también en extender la ontologíía Brick para cubrir requerimientos específicos del proyecto, como la integración con bases de datos externas de MongoDB y la validación semántica mediante SHACL (Shapes Constraint Language). Esta subsección describe el proceso de adaptación, diseño del grafo RDF e integración de dispositivos IoT en el contexto del campus ESPOL.

2.3.3.1. Adaptación de Brick al Campus ESPOL

La adaptación de Brick comenzó con la identificación de las entidades principales del dominio académico: edificios, pisos, laboratorios, zonas, espacios exteriores, equipos de medición y sensores. En este caso de estudio, se seleccionó el Edificio 11C de la Facultad de Ingeniería en Electricidad y Computación como escenario inicial, dado que concentra múltiples espacios con gran densidad de sensores.

Utilizando el extenso vocabulario semántico de Brick, se empleó la jerarquía estándar: `brick:Building`, `brick:Floor`, `brick:Room`, para modelar la estructura física, extendiendo el vocabulario mediante la creación de clases personalizadas bajo el namespace `espol:`. Por ejemplo, se definió la clase `espol:LaboratorioRedes` como instancia de `brick:Laboratory`, permitiendo representar laboratorios con equipamiento especializado.

Además, el vocabulario de Brick ofrece propiedades únicas para cada una de sus clases, para representar sus relaciones y sus ubicaciones físicas. Se establecieron relaciones semánticas como:

`brick:isPartOf` para asociar laboratorios y zonas exteriores con el edificio.

`brick:hasPoint` para vincular sensores con espacios físicos.

`brick:isLocationOf` para representar la ubicación jerárquica.

Esta adaptación resolvió limitaciones comunes en sistemas como Project Haystack,

donde términos similares pueden usarse sin un significado formal. Por ejemplo, la distinción clara entre `brick:Temperature_Sensor` y `brick:Humidity_Sensor` en Brick evita inconsistencias semánticas que ocurren en sistemas basados solo en etiquetas.

2.3.3.2. Diseño del Grafo RDF

El grafo RDF se contruyó siguiendo el modelo Brick, complementado con un namespace personalizado `espol:` para instancias y propiedades específicas del caso de uso. Uno de los desafíos clave fue mantener la interoperabilidad semántica mientras se vincula con metadatos para la descripción de equipos en MongoDB, donde se registran lecturas de sensores.

Para esto, se definieron propiedades personalizadas como `espol:db_id` y `espol:point_type`, las cuales permiten enlazar cada instancia RDF con el identificador único del equipo en MongoDB y su tipo funcional, respectivamente. Estas propiedades se declararon en el grafo RDF de la siguiente manera:

Código 2.1: Definición de prpiedades personalizadas para la vinculación de sensores

```
# Declarar la propiedad personalizada db_id para vinculación con MongoDB
g.add((ESPOL["db_id"], A, RDF.Property))
g.add((ESPOL["db_id"], RDFS.label, Literal("db_id")))
g.add((ESPOL["db_id"], RDFS.comment, Literal("MongoDB sensor ID
for linking sensor metadata.")))

# Declarar la propiedad personalizada point_type para el tipo de punto de
# medición
g.add((ESPOL["point_type"], A, RDF.Property))
g.add((ESPOL["point_type"], RDFS.label, Literal("point_type")))
g.add((ESPOL["point_type"], RDFS.comment, Literal("MongoDB sensor point type
for linking sensor metadata.")))
```

Estas propiedades permiten la trazabilidad entre el grafo semántico y la base de datos MongoDB, y también fueron validadas mediante SHACL para garantizar la consistencia del modelo y las propiedades. Para ello, se definieron PropertyShapes que establecen restricciones sobre el tipo de dato y hacia a que tipo de entidades apuntan:

Código 2.2: Definición de Propiedades SHACL para la validación de propiedades en el esquema

```
# Validación SHACL para db_id
g.add((ESPOL["db_idShape"], A, SH.PropertyShape))
g.add((ESPOL["db_idShape"], SH.maxCount, Literal(1, datatype=XSD.integer)))
g.add((ESPOL["db_idShape"], SH.path, ESPOL["db_id"]))
g.add((ESPOL["db_idShape"], SH.datatype, XSD.string))

# Añadir la regla al esquema brick:Point
g.add((BRICK.Point, SH.property, ESPOL["db_idShape"]))
```

Este mecanismo asegura que cada punto del grafo, instancia de `brick:Point`, tenga exactamente un `espol:db_id` válido y único, evitando inconsistencias y errores de integridad.

2.3.3.3. Integración de Sensores IoT

La integración de sensores IoT se mejoró mediante la incorporación de las propiedades `espol:db_id` y `espol:point_type`, que enlazan cada sensor físico con su representación RDF y sus datos operativos en MongoDB. Esto permite que una consulta SPARQL identifique la ubicación semántica del sensor y recupere su identificador para extraer datos históricos en tiempo real.

Ejemplo de instancia RDF para un sensor de temperatura:

Código 2.3: Entidades de sensores y equipos definidas dentro del esquema RDF semántico ESPOL

```
espol:airQuality1 a brick:Equipment ;
    brick:hasLocation espol:ZonaEntrada ;
    brick:hasPoint espol:airQuality1_co2,
        espol:airQuality1_humidity,
        espol:airQuality1_temp ;
    espol:db_id "11C-LabIoT:airQ1" .

espol:airQuality1_temp a brick:Temperature_Sensor ;
    brick:hasUnit unit:DEG_C ;
    brick:isPointOf espol:airQuality1 ;
    espol:point_type "temp" .
```

Gracias a esta estructura, es posible ejecutar consultas que combinan SPARQL, para la capa semántica, con filtros sobre MongoDB, habilitando escenarios como:

- Consultar todos los sensores de un laboratorio y obtener sus últimas lecturas desde MongoDB.
- Verificar si los sensores cumplen con reglas semánticas antes de integrarse al sistema.
- Inferir automáticamente el tipo de sensor y su ubicación en base a propiedades del esquema RDF.
- Mediante SHACL, asegurar que cualquier sensor que no tenga las propiedades `espol:db_id` o `espol:point_type` sea identificado como inconsistente.

2.3.4. Desarrollo del Sistema de Gestión

2.3.4.1. Backend con Django

Django es un framework web de alto nivel basado en Python que permite el desarrollo rápido y seguro de aplicaciones web. En este proyecto, Django se utiliza como backend, estructurado bajo una arquitectura API RESTful, para gestionar la base de datos semántica y manejar las consultas provenientes del frontend. Esta elección permite la separación entre la capa de presentación y la lógica del sistema, asegurando modularidad, escalabilidad y flexibilidad para futuras integraciones.

Dado que este proyecto trabaja con grafos RDF, se integra además con bibliotecas como rdflib y SPARQLWrapper para gestionar tripletas RDF y consultas SPARQL. En la arquitectura del sistema, Django actúa como puente entre el modelo semántico, la base de datos y la interfaz gráfica, procesando las consultas, transformando los resultados y exponiéndolos mediante una API RESTful. Esta estrategia permite un balance entre consultas semánticas complejas y rendimiento en escenarios de alta concurrencia. Además, se asegura la integridad de los datos y la posibilidad de extender el sistema con nuevos endpoints sin comprometer la estabilidad de los vistas existentes.

Para mejorar la eficiencia, se implementan mecanismos de cacheo en las respuestas más solicitadas y se gestionan conexiones concurrentes mediante el uso del servidor WSGI que ofrece Django, que permite escalar el servicio de acuerdo con la demanda.

En conjunto, Django cumple el rol de motor de consultas y el de garante de rendimiento y mantenibilidad, siendo fundamental dentro de la arquitectura propuesta para la gestión de datos semánticos. Por ejemplo, se implementó la consulta `getAllSensors` haciendo uso de una consulta SPARQL de la siguiente manera:

Código 2.4: Definición de un view en el framework de Django para la obtención de todos los sensores y sus propiedades dentro del esquema RDF de ESPOL

```
# Get all sensors from ESPOL's RDF schema.
def getAllSensors(request):
    # Only GET Method is allowed
    if request.method == 'GET':
        try:
            store = SPARQLStore(fuseki_endpoint, returnFormat="json", auth=fuseki_auth)
            g = Graph(store=store)

            query = """
            PREFIX brick: <https://brickschema.org/schema/Brick#>
            PREFIX espol: <https://www.espol.edu.ec/ESPOL#>
            SELECT ?sensor ?equipment ?unit ?db_id
            WHERE {
                ?sensor brick:isPointOf ?equipment .
                OPTIONAL { ?sensor brick:hasUnit ?unit. }
                FILTER(STRSTARTS(STR(?sensor), STR(espol:)))
                ?equipment a brick:Equipment ;
                            espol:db_id ?db_id .
            }
            """
```

La figura 2.4 nos muestra como una vista fue diseñada para obtener información completa de todos los sensores, incluyendo identificadores únicos, unidades de medida y metadatos asociados.

Para el almacenamiento de la información, se integraron Apache Jena Fuseki y MongoDB, formando una arquitectura que combina las capacidades semánticas de Fuseki con la flexibilidad y escalabilidad de MongoDB. Mientras Fuseki se encarga de gestionar el estructurado en RDF, MongoDB almacena datos no estructurados y lecturas en tiempo real, lo que garantiza un sistema robusto tanto para consultas complejas como para el manejo de grandes volúmenes de datos.

Esta arquitectura nos proporciona interoperabilidad con sistemas IoT y edificios inteligentes, y supera las limitaciones de escalabilidad y extensibilidad presentes en enfoques más rígidos, al ofrecer un modelo flexible y escalable que facilita la integración con sistemas de análisis y control avanzado.

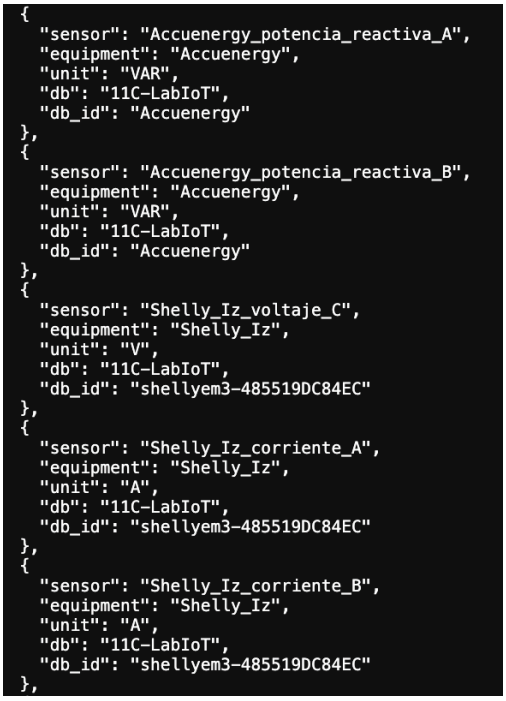

Figura 2.4: Resultados de la consulta `getAllSensors`, mostrando información completa de todos los sensores del esquema.

2.3.4.2. React y ReactFlow

React es una biblioteca de JavaScript para la construcción de interfaces de usuario interactivas y dinámicas. Su enfoque basado en componentes permite modularidad, facilitando el desarrollo, mantenimiento y reutilización de código en la aplicación web, y en el caso de este proyecto, React se utiliza para crear una interfaz visual que represente el modelo semántico de la infraestructura del campus. Además, gracias a su virtual DOM, React optimiza el renderizado y asegura una respuesta rápida a las interacciones del usuario, incluso cuando se trabaja con volúmenes grandes de datos.

ReactFlow es una extensión específica para la creación de diagramas de flujo interactivos dentro de aplicaciones React. Esta herramienta permite representar nodos y relaciones mediante elementos visuales que se pueden arrastrar, conectar y editar. En este proyecto, ReactFlow se emplea para visualizar el grafo semántico generado a partir del modelo Brick, permitiendo a los usuarios explorar gráficamente las relaciones entre sensores, dispositivos y espacios físicos de forma más intuitiva y alineada con la naturaleza de los grafos RDF. Esta representación visual no solo facilita la comprensión del sistema, sino que también actúa como un puente entre expertos técnicos y usuarios finales con menos experiencia en modelado semántico.

La combinación de React y ReactFlow proporciona una experiencia de usuario fluida. Los usuarios pueden ver en tiempo real cómo cambian los estados de sensores o cómo se relacionan entre sí los componentes de la infraestructura. Además, se pueden implementar funcionalidades interactivas como filtros, búsquedas o vistas detalladas al hacer clic en un nodo específico del grafo, como se puede observar en la Figura 2.5. Estas funcionalidades aumentan el nivel de control y personalización de la interfaz, permitiendo que cada usuario adapte la visualización a sus necesidades específicas.

Figura 2.5: Entidades y Edificios representados como nodos en la aplicación utilizando ReactFlow.

La aplicación React consume una API servida por Django, que devuelve los datos estructurados del modelo semántico en formato JSON. Estos datos son luego transformados en nodos y aristas para ser renderizados por ReactFlow, ofreciendo así una visión clara y actualizada del sistema físico y su representación lógica. A futuro, esta integración podría ampliarse con características como la edición directa del grafo desde la interfaz o la generación automática de reportes a partir de la interacción con los nodos.

2.4. Esquema de Diseño Propuesto

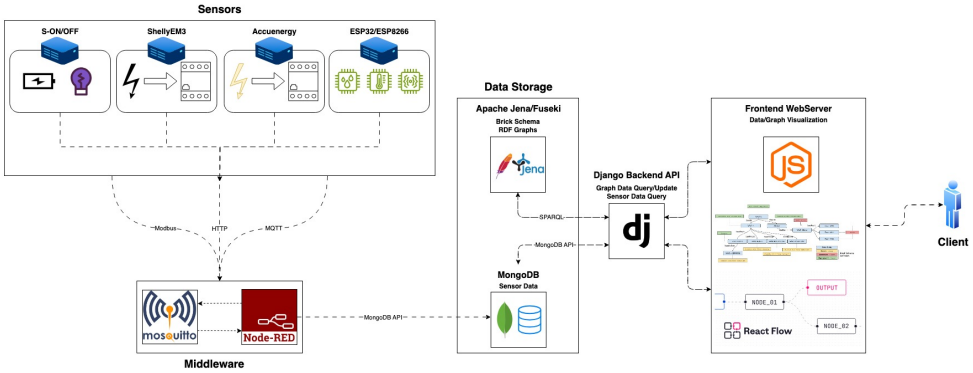

Figura 2.6: Diagrama de la arquitectura del sistema, mostrando el flujo desde sensores IoT (DHT, Shellys, Accuenergy, Dispositivos basado en ESP32/ESP8266) a middleware Node-RED, almacenamiento en Fuseki y MongoDB, backend Django y frontend ReactFlow.

La arquitectura propuesta integra componentes clave con flujos de datos, utilizando tecnologías como Django y React Flow. Como se ilustra en la Figura 2.6, el sistema comienza con sensores IoT, tales como DHT para temperatura y humedad, Shellys y Accuenergy para consumo eléctrico, etc. que capturan datos ambientales en el Edificio 11C. Estos datos se canalizan a través de un middleware basado en Node-RED, que actúa como orquestador para el procesamiento inicial y la transformación, asegurando una integración fluida con estándares semánticos.

Posteriormente, los datos se almacenan en una capa dual: Apache Fuseki para el grafo RDF/OWL basado en Brick, que habilita inferencias semánticas avanzadas, y MongoDB para lecturas en tiempo real de alta demanda. El backend en Django expone APIs RESTful que consultan estos repositorios, mediante SPARQL para metadata semántica y queries directas a Mongo para valores actuales, mientras el frontend, implementado en React con ReactFlow, visualiza el grafo interactivo, permitiendo al usuario explorar relaciones y monitorear datos en vivo. Esta arquitectura no solo demuestra la interoperabilidad superior de Brick, al reducir ambigüedades semánticas mediante relaciones formales como brick:isPointOf, sino que también evidencia, a través de flujos optimizados, mejoras en eficiencia y escalabilidad, con tiempos de respuesta inferiores en pruebas controladas.

CAPÍTULO 3

3. PRUEBAS Y RESULTADOS

En este capítulo se describen las pruebas realizadas para validar la correcta implementación del esquema semántico basado en la ontología Brick en la infraestructura del campus, específicamente en el Laboratorio de IoT y Sistemas Telemáticos, Laboratorio de Redes Avanzadas y Laboratorio de Sistemas en la Nube. El objetivo principal fue comprobar que el esquema definido representara con precisión la infraestructura física, los dispositivos desplegados y sus relaciones, así como también verificar la accesibilidad de la información de las entidades y los datos generados por sus sensores mediante consultas SPARQL.

Las pruebas se centraron en aspectos clave como la integración y mantenimiento de equipos IoT, la correcta transmisión de datos, la exactitud del modelado semántico, la asociación entre dispositivos y ubicaciones, y el desempeño de las consultas sobre el esquema semántico creado. Los resultados obtenidos constituyen evidencia del funcionamiento y la utilidad del modelo implementado en un entorno real.

3.1. PRUEBAS

3.1.1. Pruebas de la transmisión de datos

Para reforzar la integración semántica, en los paquetes de datos enviados por los sensores se incorporaron campos de metadata con propiedades personalizadas que permiten identificar cada dispositivo y sus dentro del modelo Brick. Estos campos incluyen referencias a las URIs correspondientes en el grafo RDF, garantizando la trazabilidad y la asociación directa entre el dispositivo físico, lógico y su representación semántica.

```
107     char buffer[512];
108     int cx = snprintf(buffer, sizeof(buffer),
109                       "{\"id\":\"%s\", \"data\":["
110                       "{\"val\":%.2f,\"type\":\"temp\"},"
111                       "{\"val\":%.2f,\"type\":\"humidity\"}"
112                       "], \"sensedAt\":\"\"}",  // TODO: add a timestamp
113                       sensor_id, isnan(temperature) ? 0.0 : temperature, isnan(humidity) ? 0.0 : humidity);
114
115     if (cx < 0 || cx >= (int)sizeof(buffer)) {
116       Serial.println("Error formatting JSON.");
117       return;
118     }
```

Figura 3.1: Construcción del paquete de datos en equipos ESP32/ESP8266 usando las variables necesarias para su identificación en el esquema.

La prueba consistió en verificar que, tras la adquisición y normalización de los datos mediante Node-RED, las lecturas provenientes de los sensores incluyeran las propiedades db_id y point_type, definidas en el namespace espol:, las cuales permiten establecer la correspondencia con la base de datos MongoDB y el tipo de punto dentro del esquema Brick. Además, se comprobó la integridad de los paquetes transmitidos mediante el protocolo MQTT, garantizando que los mensajes llegaran al broker sin pérdida ni corrupción.

```
1   let sensor_id = msg.topic.split("/")[3];
2   let sensedAt = new Date();
3   let mongo_payload = {
4       metadata: {sensorId : sensor_id},
5       timestamp: sensedAt
6   };
7   if (sensor_id === "Accuenergy") {
8       let datalist = msg.payload.device.readings;
9       const paramTypeMap = {
10          V1: "voltageA", V2: "voltageB", V3: "voltageC",
11          I1: "currentA", I2: "currentB", I3: "currentC",
12          P1: "powerA", P2: "powerB", P3: "powerC",
13          Q1: "reactivepowerA", Q2: "reactivepowerB", Q3: "reactivepowerC",
14          S1: "apparentpowerA", S2: "apparentpowerB", S3: "apparentpowerC",
15          PF1: "pfA", PF2: "pfB", PF3: "pfC"
16      };
```

Figura 3.2: Ejemplo de una función en nodos de NodeRed para verificar y manipular los datos que provienen de los sensores.

El flujo de transformación se ejecutó para asegurar que cada mensaje bruto capturado desde el dispositivo fuera enriquecido con la metadata correspondiente, lo que permite mantener la interoperabilidad entre la capa física de sensores, la capa de almacenamiento y el modelo RDF.

3.1.2. Pruebas del modelado semántico

Se evaluó la consistencia del grafo RDF construido y que está almacenado en la base de datos Apache Jena/Fuseki, validando que las entidades definidas en la ontología Brick reflejaran correctamente la jerarquía espacial de edificios, pisos, laboratorios y los dispositivos asociados, como sensores y equipos. Para ello, se realizaron consultas SPARQL orientadas a comprobar la integridad de las propiedades personalizadas `espol:db_id` y `espol:point_type` para asegurar la interoperabilidad con la base de datos de lecturas MongoDB.

Código 3.1: Contrucción de Query SPARQL para obtener todos los sensores de un equipo y su propiedad espol:db_id

```
PREFIX brick: <https://brickschema.org/schema/Brick#>
PREFIX espol: <https://www.espol.edu.ec/ESPOL#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?equipment ?unit ?db_id ?type
WHERE {{
    espol:{sensor_id} brick:isPointOf ?equipment .
    OPTIONAL {{ espol:{sensor_id} brick:hasUnit ?unit. }}
    espol:{sensor_id} espol:point_type ?type .
    ?equipment a brick:Equipment ;
                espol:db_id ?db_id .
}}
```

Recuperar todos los equipos presentes en un laboratorio específico, validando la asociación correcta con su ubicación `brick:isLocatedIn` y listar las unidades de medida asociadas a cada punto con `brick:hasUnit` y verificar que coincidieran con la configuración física del sensor. Estas pruebas confirmaron que el modelo semántico no solo representaba fielmente la infraestructura física, sino que también permitía la inferencia de relaciones y el filtrado avanzado de datos mediante consultas SPARQL.

CAPÍTULO 4

4. CONCLUSIONES, RECOMENDACIONES Y LÍNEAS FUTURAS

El presente proyecto BrickIoT ha demostrado su viabilidad y ventajas de implementar un esquema semántico basado en la ontología Brick para la gestión de datos en entornos de edificios inteligentes. La arquitectura modular que hemos diseñado ofrece beneficios en términos de escalabilidad, consistencia semántica, trazabilidad e integración en tiempo real, fundamentales para sistemas de monitoreo avanzados como este.

4.1. Conclusiones

El análisis y la implementación realizada a largo de este proyecto permiten nos han permitido concluir que:

- **Escalabilidad:** La incorporación de nuevos dispositivos o tipos de sensores puede realizarse mediante la adición dinámica de nodos o entidades mediante la interfa gráfica que hemos desarrollado sobre el esquema de clases de Brick, sin necesidad de modificar la estructura entera del esquema ni tener que desarrollarlo desde cero. Esto garantiza que la infraestructura pueda crecer de manera ordenada y eficiente.
- **Consistencia semántica:** Cada dato insertado en la base de datos mantiene su asociación con la entidad correspondiente en el grafo RDF de la ESPOL, asegurando coherencia con el modelo Brick y facilitando la interoperabilidad, incluso entre distintos sistemas dentro del campus.
- **Trazabilidad:** Gracias al uso de identificadores únicos y metadatos normalizados y agregados a las entidades como propiedades personalizadas, es posible rastrear el

origen y el contexto de cada medición, lo que simplifica las auditorías, el análisis histórico por sensor o por área y procesos de verificación de datos mendiante SHACL.

- **Integración en tiempo real:** La combinación de MQTT y Node-RED garantiza que la latencia entre la generación de los datos y su almacenamiento sea mínima, permitiendo aplicaciones de monitoreo y control en tiempo real, así como la reacción inmediata ante eventos críticos y alertas inteligentes que se puedan implementar.
- **Flexibilidad y adaptabilidad:** El sistema permite integrar una amplia variedad de tipos de sensores y actuadores que se usan comunmente en proyecto de la carrera de Telemática, así como adaptarse a cambios en la topología de la red o en los requerimientos de análisis, sin comprometer la integridad del modelo semántico.

4.2. Recomendaciones

- Capacitación del personal técnico o de los estudiantes en tecnologías semánticas y ontologías, asegurando un manejo adecuado del esquema RDF de la ESPOL, OWL y la integración con el modelo Brick.
- Establecer un procedimientos de normalización de datos y etiquetado mucho más uniforme que el actual, para mantener consistencia en ambientes con múltiples laboratorios.
- Implementar estrategias de respaldo y recuperación de datos en tiempo real, para así garantizar la continuidad del sistema ante fallos o interrupciones.
- Documentar las configuraciones, flujos de datos y procesos de integración para facilitar el mantenimiento y la escalabilidad futura del sistema.

4.3. Líneas Futuras

Para continuar avanzando en la investigación y optimización de sistemas basados en Brick y tecnologías semánticas en IoT, hemos planteado los siguientes puntos:

- Extensión del modelo para incluir nuevos dominios y tipos de dispositivos, tales como energías renovables, sistemas de seguridad o automatización de espacios públicos del campus.
- Implementación de propiedades personalizadas dentro del esquema RDF semántico de ESPOL para la interoperabilidad adicional de comunicación con sistemas internos y plataformas del campus de manera más fluida y segura.
- Evaluación de la eficiencia energética y la optimización de los recursos mediante simulaciones 3D y gemelos digitales utilizando los mismos datos semánticos del grafo, aprovechando la trazabilidad y consistencia de los datos y sus propiedades para guardar valores de importancia en aplicaciones de visualización 3D.
- Integración de algoritmos de inteligencia artificial y aprendizaje automático sobre los datos semánticamente estructurados, para mejorar predicciones, detección de anomalias y mantenimiento de los edificios, aulas y laboratorios del campus.

BIBLIOGRAFÍA

[1] R. Agarwal, D. G. Fernandez, T. Elsaleh et al., «Unified IoT ontology to enable interoperability and federation of testbeds», en *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, Reston, VA, USA: IEEE, dic. de 2016. doi: 10.1109/wf-iot.2016.7845470. dirección: http://ieeexplore.ieee.org/document/7845470/ (visitado 20-07-2025).

[2] S. H. y Andy Seaborne, *SPARQL 1.1 Query Language*, W3C Recommendation, mar. de 2013. dirección: https://www.w3.org/TR/sparql11-query/.

[3] S. H. y Andy Seaborne, *SPARQL 1.1 Query Results JSON Format*, W3C Recommendation, mar. de 2013. dirección: https://www.w3.org/TR/sparql11-results-json/.

[4] B. Balaji, A. Bhattacharya, G. Fierro et al., «Brick: Towards a Unified Metadata Schema For Buildings», en *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, Palo Alto CA USA: ACM, nov. de 2016, págs. 41-50. doi: 10.1145/2993422.2993577. dirección: https://dl.acm.org/doi/10.1145/2993422.2993577 (visitado 20-07-2025).

[5] B. Balaji, A. Bhattacharya, G. Fierro et al., «Brick : Metadata schema for portable smart building applications», en, *Applied Energy*, vol. 226, págs. 1273-1292, sep. de 2018, Publisher: Elsevier BV, issn: 0306-2619. doi: 10.1016/j.apenergy.2018.02.091. dirección: https://linkinghub.elsevier.com/retrieve/pii/S0306261918302162 (visitado 20-07-2025).

[6] A. Bhattacharya, J. Ploennigs y D. Culler, «Short Paper: Analyzing Metadata Schemas for Buildings: The Good, the Bad, and the Ugly», en *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, Seoul South Korea: ACM, nov. de 2015, págs. 33-34. doi: 10.1145/2821650.2821669. dirección: https://dl.acm.org/doi/10.1145/2821650.2821669 (visitado 20-07-2025).

[7] A. P. y Birte Glimm, *SPARQL 1.1 Federated Query*, W3C Recommendation, mar. de 2013. dirección: https://www.w3.org/TR/sparql11-federated-query/.

[8] B. G. y Chimezie Ogbuji, *SPARQL 1.1 Entailment Regimes*, W3C Recommendation, mar. de 2013. dirección: https://www.w3.org/TR/sparql11-entailment/.

[9] B. Consortium, *Brick Ontology Documentation*, 2025. dirección: https://docs.brickschema.org/.

[10] B. Consortium, *Brick Schema GitHub Repository*, 2025. dirección: https://github.com/BrickSchema/Brick.

[11] J. B. y Dave Beckett, *SPARQL 1.1 Query Results XML Format (Second Edition)*, W3C Recommendation, mar. de 2013. dirección: https://www.w3.org/TR/sparql11-results-xml/.

[12] G. Fierro y D. E. Culler, «Design and Analysis of a Query Processor for Brick», en, *ACM Transactions on Sensor Networks*, vol. 14, n.º 3-4, págs. 1-25, nov. de 2018, Publisher: Association for Computing Machinery (ACM), issn: 1550-4859, 1550-4867. doi: 10.1145/3199666. dirección: https://dl.acm.org/doi/10.1145/3199666 (visitado 20-07-2025).

[13] P. Haystack, *Project Haystack Documentation*, 2025. dirección: https://project-haystack.org/doc/.

[14] G. T. W. y. K. G. C. Lee Feigenbaum y E. Torres, *SPARQL 1.1 Protocol*, W3C Recommendation, mar. de 2013. dirección: https://www.w3.org/TR/sparql11-protocol/.

[15] J. Li, N. Li, R. Yan, K. Farruh, A. Li y K. Li, «Research on Brick Schema Representation for Building Operation with Variable Refrigerant Flow Systems», *Journal of Building Engineering*, vol. 56, pág. 104 792, sep. de 2022, arXiv:2108.07037 [cs], issn: 2352-7102. doi: 10.1016/j.jobe.2022.104792. dirección: http://arxiv.org/abs/2108.07037 (visitado 20-07-2025).

[16] N. Luo, G. Fierro, Y. Liu, B. Dong y T. Hong, «Extending the Brick schema to represent metadata of occupants», en, *Automation in Construction*, vol. 139, pág. 104 307, jul. de 2022, issn: 09265805. doi: 10.1016/j.autcon.2022.104307. dirección: https://linkinghub.elsevier.com/retrieve/pii/S0926580522001807 (visitado 28-05-2025).

[17] C. Ogbuji, *SPARQL 1.1 Uniform HTTP Protocol for Managing RDF Graphs*, W3C Recommendation, mar. de 2013. dirección: https://www.w3.org/TR/sparql11-http-rdf-update/.

[18] E. D. Okonta, F. Rahimian, V. Vukovic y S. Rodriguez, «Semantic interoperability on IoT: Aligning IFC and Smart Application Reference (SAREF) sensor data models», en, *Automation in Construction*, vol. 177, pág. 106 328, sep. de 2025, Publisher: Elsevier BV, issn: 0926-5805. doi: 10.1016/j.autcon.2025.106328. dirección: https://linkinghub.elsevier.com/retrieve/pii/S0926580525003681 (visitado 20-07-2025).

[19] A. P. y. A. P. Paul Gearon, *SPARQL 1.1 Update*, W3C Recommendation, mar. de 2013. dirección: https://www.w3.org/TR/sparql11-update/.

[20] M. Pritoni, D. Paine, G. Fierro et al., «Metadata Schemas and Ontologies for Building Energy Applications: A Critical Review and Use Case Analysis», en, *Energies*, vol. 14, n.º 7, pág. 2024, abr. de 2021, Publisher: MDPI AG, issn: 1996-1073. doi: 10.3390/en14072024. dirección: https://www.mdpi.com/1996-1073/14/7/2024 (visitado 20-07-2025).

[21] M. Pritoni, D. Paine, G. Fierro et al., «Metadata Schemas and Ontologies for Building Energy Applications: A Critical Review and Use Case Analysis», en, *Energies*, vol. 14, n.º 7, pág. 2024, abr. de 2021, Publisher: MDPI AG, issn: 1996-1073. doi: 10.3390/en14072024. dirección: https://www.mdpi.com/1996-1073/14/7/2024 (visitado 20-07-2025).

[22] Z. Qiang, S. Hands, K. Taylor et al., «A systematic comparison and evaluation of building ontologies for deploying data-driven analytics in smart buildings», en, *Energy and Buildings*, vol. 292, pág. 113 054, ago. de 2023, Publisher: Elsevier BV, issn: 0378-7788. doi: 10.1016/j.enbuild.2023.113054. dirección: https://linkinghub.elsevier.com/retrieve/pii/S0378778823002840 (visitado 20-07-2025).

[23] C. Quinn y J. J. McArthur, «Comparison of Brick and Project Haystack to Support Smart Building Applications», *Advanced Engineering Informatics*, vol. 47, pág. 101 233, ene. de 2021, arXiv:2205.05521 [cs], issn: 1474-0346. doi: 10.1016/j.aei.2020.101233. dirección: http://arxiv.org/abs/2205.05521 (visitado 20-07-2025).

[24] C. Quinn y J. McArthur, «A case study comparing the completeness and expressiveness of two industry recognized ontologies», en, *Advanced Engineering Informatics*, vol. 47, pág. 101 233, ene. de 2021, Publisher: Elsevier BV, issn: 1474-0346. doi: 10.1016/j.aei.2020.101233. dirección: https://linkinghub.elsevier.com/retrieve/pii/S1474034620302020 (visitado 20-07-2025).

[25] A. Seaborne, *SPARQL 1.1 Query Results CSV and TSV Formats*, W3C Recommendation, mar. de 2013. dirección: https://www.w3.org/TR/sparql11-results-csv-tsv/.

[26] «Semantic Web Technologies for Internet of Things Semantic Interoperability», en, en *Lecture Notes in Networks and Systems*, ISSN: 2367-3370, 2367-3389, Cham: Springer International Publishing, 2022, págs. 133-143, isbn: 978-3-030-91737-1 978-3-030-91738-8. doi: 10.1007/978-3-030-91738-8_13. dirección: https://link.springer.com/10.1007/978-3-030-91738-8_13 (visitado 20-07-2025).

[27] F. Vittori, C. F. Tan, A. L. Pisello, A. Chong y C. Miller, *BIM-to-BRICK: Using graph modeling for IoT/BMS and spatial semantic data interoperability within digital data models of buildings*, Version Number: 1, 2023. doi: 10.48550/ARXIV.2307.13197. dirección: https://arxiv.org/abs/2307.13197 (visitado 20-07-2025).

[28] G. T. Williams, *SPARQL 1.1 Service Description*, W3C Recommendation, mar. de 2013. dirección: https://www.w3.org/TR/sparql11-service-description/.