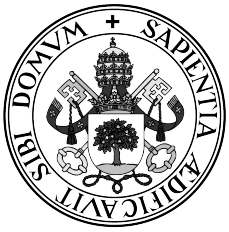


Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
(Mención en Ingeniería de Software)

Desarrollo de una ontología y creación de un servicio para la gestión eficiente de la energía en edificios

Autor:
D. Daniel Jiménez García

Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
(Mención en Ingeniería de Software)

Desarrollo de una ontología y creación de un servicio para la gestión eficiente de la energía en edificios

Autor:
D. Daniel Jiménez García

Tutores:
D. Jesús Cámara Moreno
Dña. Susana Gutiérrez Caballero

Agradecimientos

Deseo expresar mi más sincero agradecimiento a todos los profesores que me han acompañado a lo largo de mi trayectoria académica. Cada uno, desde sus fortalezas y exigencias particulares, ha contribuido a forjar mi determinación y a permitirme llegar hasta donde estoy hoy.

Quiero destacar especialmente a mis dos tutores, Jesús y Susana, quienes me han acompañado en esta etapa final brindándome siempre su ayuda y apoyo. Sus comentarios y correcciones han sido claves para realizar el mejor trabajo posible. Además, quiero agradecer a la Fundación CARTIF por brindarme esta oportunidad, por prestarme los medios y los recursos necesarios para realizar este Trabajo Fin de Grado, y a todas las personas que lo componen por ser una gran fuente de inspiración.

Ademñas, deseo reconocer a Yania Crespo y a Belarmino Pulido sus valiosas orientaciones en algunas partes de este trabajo.

Por último, no me voy a olvidar de todos aquellos amigos y familiares que me han acompañado y a los que siempre guardaré un especial cariño.

Resumen

En este trabajo, se ha desarrollado una ontología en base a la integración de conceptos de SAREF y su extensión SAREF for Building. La ontología generada a partir del mapping de los elementos definidos en un fichero IFC sirve como modelo de datos para la autoconfiguración de un servicio de climatización. Se ha tratado que este servicio de autonfiguración fuera agnóstico y las consultas sean delegadas en otra capa que si sea dependiente de dicho modelo. Aunque la conversión de los sensores y actuadores funcionan para cualquier tipo de estos definidos en la semántica de IFC, el servicio, a través del ingestor de datos IFC, únicamente mapea los sensores de temperatura, humedad y CO2.

Las pruebas realizadas han involucrado la obtención de datos del edificio de CARTIF3 a través de ficheros CSV debido a que no se ha podido ejecutar una prueba real en el edificio. Por este motivo las actuaciones necesarias que habría que realizar se ven por consola. Para terminar de comprobar que la autoconfiguración es útil en más edificios y que no se trata de una solución ad hoc en el edificio de CARTIF3, se ha creado una demo a partir de otro fichero IFC descargado de internet. En esta demo, como no se disponían de datos previos, se ha implementado una función para generar predicciones de temperatura que posteriormente se insertan en una base de datos alojada en una máquina virtual de Oracle Cloud Infrastructure y simular una arquitectura parecida a la que se tendría en un entorno real.

Abstract

In this work, an ontology has been developed based on integrating concepts from SAREF and its extension, SAREF for Building. The ontology generated from the mapping of elements defined in an IFC file subsequently serves as a data model for the auto-configuration of a climate control service. Efforts have been made to ensure this auto-configuration service remains agnostic, delegating queries to another layer that depends explicitly on the data model. Although the sensor and actuator conversion works for any type defined in the IFC semantics, the service, through the IFC data ingestor, only maps temperature, humidity, and CO2 sensors.

The tests conducted involved obtaining building data from CARTIF3 through CSV files because it was not possible to execute a real test in the building itself. Therefore, the required actions are displayed in the console. To further confirm that auto-configuration is effective in multiple buildings and not a solution tailored specifically for the CARTIF3 building, a demo has been created using another IFC file downloaded from the internet. In this demo, due to the lack of previous data, a function has been implemented to generate temperature predictions, which are subsequently inserted into a database hosted on an Oracle Cloud Infrastructure virtual machine, simulating an architecture similar to one that would exist in a real environment.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Durante las últimas décadas, la eficiencia energética se ha convertido en un objetivo estratégico para todos los gobiernos, de ahí que hayan propuesto diversas metas para frenar una crisis energética ocasionada por factores como el aumento de la demanda, la inestabilidad en los mercados internacionales, o la necesidad urgente de reducir las emisiones de carbono.

Varias de estas metas tienen su origen en organizaciones gubernamentales como la ONU o la propia Unión Europea (UE). La primera propone 17 ODS (Objetivos de Desarrollo Sostenible) [1] con los que se pretende fomentar una agenda viable y lograr un futuro mejor para todos, siendo los objetivos 7 y 13 los que están relacionados con garantizar el acceso a una energía limpia y asequible, así como alertar de los peligros del cambio climático. La UE, por su parte, a través del EGD (European Green Deal) y la EED (Energy Efficiency Directive), pretende conseguir una reducción del consumo energético en los edificios, los cuales representan aproximadamente el 40 % del consumo total de la energía en Europa [2].

Para lograr estos objetivos existen actualmente múltiples sistemas de monitorización que hacen uso de dispositivos IoT y plataformas de gestión energética que operan con distintos formatos de datos y modelos conceptuales. Esta diversidad en la representación de los datos dificulta la interoperabilidad entre sistemas, de ahí que el uso de ontologías sea considerada una opción adecuada para dar solución a este tipo de problema, pues permiten la creación de una web semántica donde se represente de forma única la información utilizada por diferentes sistemas.

1.1. Contexto

Este trabajo surge a raíz de las tareas realizadas en la Fundación CARTIF durante la fase de prácticas curriculares. Inicialmente, se analizó el estado en que se encontraba la ontología dCO que estaban utilizando en el Proyecto Europeo DEDALUS. La principal aportación de esta ontología consiste en unificar los conceptos de energía proporcionados por la ontología de SAREF [3] con el concepto de Thing que específica la W3C [4, 5]. La multitud de conceptos abordados, así como los *pitfalls* encontrados con OOPS! [6] han sido los principales motivos que han dado lugar al presente trabajo, donde se lleva a cabo la definición de una ontología especificando todos los requisitos necesarios, así como la creación de un servicio de auto-configuración que hace uso de la ontología.

1.2. Objetivos

Una de las metas de este TFG es poner en práctica los conocimientos y habilidades adquiridos durante la carrera, los cuales han permitido que pueda afrontar las diferentes tareas necesarias para llevar a cabo con éxito el presente trabajo. Para ello, se han propuesto los siguientes objetivos:

- Generar una ontología que permita representar de manera estructurada la información relacionada con el consumo de energía en edificios, facilitando su integración y análisis.
- Desarrollar un servicio de control climático climático que se autoconfigura en base a la información contenida en la ontología.

Estos objetivos conlleva aprender:

- Los procesos involucrados en la administración energética de edificios, así como las tecnologías, herramientas y metodologías utilizadas actualmente en los sistemas de control.
- Estándares semánticos (IFC, OWL, RDF/Turtle) para la representación, gestión y análisis del comportamiento energético y climático en edificios.

Con estos objetivos se pretende, a su vez, desarrollar la capacidad de adaptación a un nuevo campo de conocimiento y ser capaz de aportar una solución innovadora que permita, en este caso, resolver un problema de gestión energética combinando el uso de diferentes tecnologías.

1.3. Estructura del documento

El presente documento se ha estructurado de la siguiente manera:

- **Introducción**: describe la temática que se aborda y justifica su interés con el fin de motivar la realización de este trabajo. Asimismo, se establecen los principales objetivos a conseguir.
- **Estado del Arte**: se lleva a cabo una revisión de los trabajos previos relacionados con el problema a resolver, se identifican las limitaciones de las soluciones existentes y se justifica la necesidad de afrontar una nueva propuesta.
- **Metodología**: se detalla el ciclo de vida utilizado para el desarrollo del proyecto, incluyendo una cronología, el trabajo a desarrollar, sus entregables, los riesgos y costes asociados y el seguimiento del proyecto con los cambios realizados en cada incremento.
- **Herramientas Software**: se muestran todas las tecnologías utilizadas en la elaboración de este proyecto.
- **Especificación de requisitos**: se recopilan los requisitos del servicio y se detallan los modelos de casos de uso.
- **Análisis del modelo ontológico**: se describen los conceptos, propiedades y definiciones relevantes con las que se construirá el modelo de datos con el que el servicio va a funcionar.
- **Diseño e Implementación**: se describe con detalle las decisiones técnicas tomadas, la implementación de la base de datos PostgreSQL, la arquitectura del servicio, los diagramas detallados de los módulos y los diagramas de secuencia.
- **Pruebas**: se describe los demos utilizados para la ejecución del servicio con los resultados obtenidos en la ejecucuión de estas.
- **Conclusiones**: se analizan las principales conclusiones obtenidos, justificando si han permitido cumplir los objetivos inicialmente establecidos. Además, se lleva a cabo una reflexión sobre problemas encontrados y se proponen posibles líneas de trabajo futuro.
- **Bibliografía**: incluye todas las referencias bibliográficas utilizadas en este trabajo, siguiendo el formato estándar propuesto por la organización IEEE.

- **Anexos**: proporcionan documentación adicional, como guías de instalación de herramientas software utilizadas, un manual de desarrollador que explica cómo continuar con el trabajo desarrollado y el material complementario con imágenes relevantes.

1.4. Glosario de Términos

A continuación se indica el significado de los principales términos que se irán mencionando a lo largo del presente trabajo y están relacionados con la temática del mismo.

AI	Artificial Intelligence
BAS	Building Automation Systems
BEMS	Building Energy Modelling Systems
BIM	Building Information Modelling
BMS	Building Management Systems
CQ	Competency Questions
dCO	domOS Common Ontology
DF	Demand Flexibility
DL	Description Logic
DR	Demand Response
EED	Energy Efficiency Directive
EGD	European Green Deal
EMCS	Energy Management Control Systems
EMS	Energy Management Systems
EPMS	Energy Power Management Systems
FOL	First Order Logic
HI	Heat Index
HTML	HyperText Markup Language
HVAC	Heating, Ventilation and Air Conditioning
IEEE	Institute of Electrical and Electronics Engineers
IFC	Industry Foundation Classes
IoT	Internet of Things
IRI	Internationalized Resource Identifier
JSON-LD	JSON for Linked Data

JVM	Java Virtual Machine
KB	Knowledge Base
KBS	Knowledge-Based Systems
LD	Linked Data
LTS	Long Term Support
MQTT	Message Queuing Telemetry Transport
OAV	Objeto-Atributo-Valor
OCI	Oracle Cloud Infrastructure
OEG	Ontology Engineering Group
OWL	Ontology Web Language
PBS	Product-Breakdown Structure
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RH	Relative Humidity
T	Temperature
TTL	Terse-RDF Triple Language
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VSC	Visual Studio Code
W3C	World Wide Web Consortium
WBS	Work-Breakdown Structure
WSL	Windows Subsystem for Linux
WoT	Web of Things
XML	eXtensible Markup Language
XSD	XML Schema Definition

Capítulo 2

Estado del Arte

El cambio climático ha obligado a la sociedad a buscar la manera de utilizar la energía de forma más eficiente y estudiar alternativas energéticas con las que reducir la huella de carbono. Esto ha derivado en la utilización simultánea de diferentes sistemas encargados de gestionar la energía, monitorizar su uso y actuar en consecuencia. Estos sistemas trabajan con vocabularios, modelos, interpretaciones y fuentes de datos únicos, por tanto, es aquí donde las ontologías ayudan a crear modelos compartidos con el fin de que los diferentes sistemas y usuarios puedan integrar fuentes de datos procedentes de diferentes orígenes y trabajar de forma interoperable.

En esta sección se va a realizar un recorrido por las áreas de conocimiento implicadas en el desarrollo de este TFG, indicando las tecnologías, metodologías y técnicas fundamentales utilizadas en el modelado de edificios y la optimización del uso de la energía.

2.1. Representación del Conocimiento

Este campo de estudio es uno de los pilares fundamentales de la cognición humana y de la IA. Los avances producidos han contribuido al desarrollo de sistemas inteligentes, pues proporciona los medios formales y estructurados necesarios para modelar información compleja de forma que pueda ser interpretada tanto por humanos como por máquinas. Además, facilita el razonamiento automático al permitir que los sistemas deduzcan nueva información a partir de hechos existentes, mejorando así la toma de decisiones y el análisis de datos.

En el contexto de las ontologías, estas ofrecen las herramientas necesarias para capturar la semántica de un dominio específico y definir de forma explícita los conceptos, relaciones y restricciones que existen entre estos. La importancia del uso de ontologías radica, por tanto, en que ofrecen la posibilidad de compartir y reutilizar conocimiento de forma interoperable, lo cual es esencial en entornos donde la integración de datos heterogéneos es un desafío constante.

En el siglo XX aparecen, de la mano de Charles Sanders Peirce, las primeras formulaciones modernas para la representación del conocimiento en forma de grafos existenciales (sistemas Alpha: lógica proposicional, sistemas Beta: FOL -First Order Logic- y, sistemas Gamma: lógicas de orden superior, lógica modal, lógica de multitudes, abstracciones y colecciones) [7, 8]. Posteriormente, Allan M. Collins y M. Ross Quillian [9] formularon la estructura de la memoria de un computador como una zona para el almacenamiento de información semántica, concretamente, palabras que se apuntan unas a otras y que, en conjunto, representan el significado de cada una de ellas (Figura 2.1). Años más tarde, Marvin Minsky introdujo su sistema de *frames* y *slots* [10], lo que hoy en día se conoce como diagramas de clases o diagramas Entidad-Relación. Otros autores, como Russel, Norvig, Poole y Mackworth [11, 12] sentaron las bases de los agentes racionales (o inteligentes) a partir de la lógica como medio para la representación interna del conocimiento con el que estos agentes poseen información del entorno que les rodea y de cómo deben interactuar con el mismo.

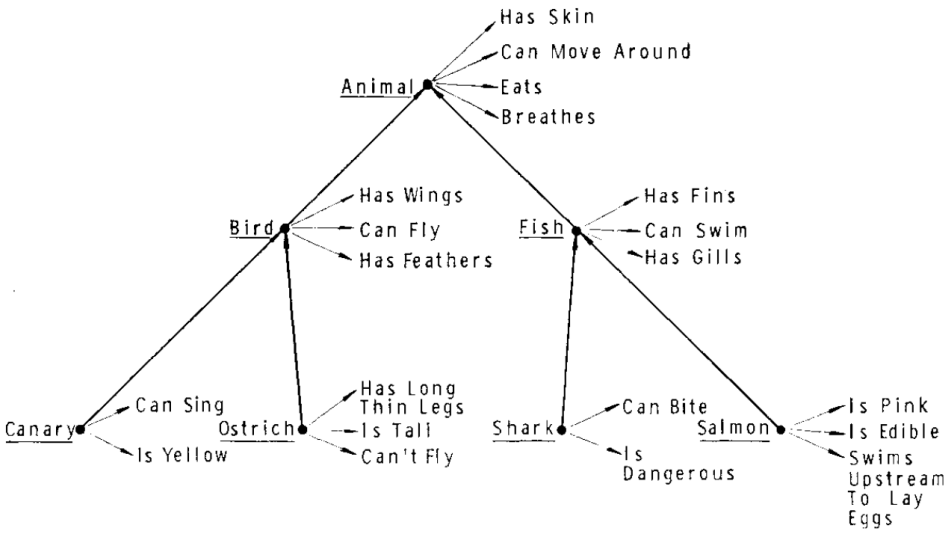


Figura 2.1: Ejemplo de la Red Semántica
[9]

2.1.1. Ontologías y Web Semántica

Si bien los avances en redes semánticas, sistemas de frames y lógicas formales permitieron estructurar la información para su interpretación automática, las ontologías surgen como una evolución natural al integrar rigurosidad conceptual, estandarización y capacidad de interoperabilidad.

Según Thomas R. Grubber, "una ontología es una especificación explícita de una conceptualización" [13]. Otra posible definición es la proporcionada por Studer: "una ontología es un entendimiento común y compartido sobre un dominio particular y que puede ser comunicado a través de personas y máquinas" [14]. Se puede deducir, por tanto, que una ontología expone un vocabulario con el que se declaran explícitamente conceptos de una manera sistemática y formaliza las relaciones y restricciones entre ellos para que no haya posibles interpretaciones entre los desarrolladores, usuarios y/o máquinas que la compartan. Esta representación compartida ayuda a la interoperabilidad de la información generando un esquema único que todos pueden entender, lo que facilita la integración de los datos y el razonamiento automatizado.

En función del nivel de detalle y de dependencia que exista con la tarea que se quiera resolver, las ontologías se pueden clasificar de la siguiente forma [15]:

- **Ontologías de alto nivel**: definen conceptos comunes e independientes de dominios específicos y proporcionan un marco común para permitir que distintos sistemas compartan una base conceptual sobre la que construir sus modelos específicos.
- **Ontologías específicas del dominio:** modelan el conocimiento de un sector concreto, capturando los conceptos, relaciones y restricciones propias del sector, es decir, estandariza los conceptos para que los desarrolladores puedan usar un vocabulario común y compartido, facilitando la organización y consulta de la información dentro de ese dominio.
- **Ontologías orientados a la tarea:** especializan las de alto nivel con conceptos de una determinada disciplina, pero se centran en resolver una tarea o actividad genérica.
- **Ontologías de aplicación:** dependen de las ontologías específicas de dominio y de las orientadas a la tarea. Son especializaciones pensadas para resolver un problema determinado donde no hay un conocimiento compartido o consenso entre desarrolladores, pues suele corresponderse con la visión particular de un grupo ante un problema concreto y, además, no es frecuente que se reutilice tras el desarrollo de la propia ontología.

Debido al concepto de "conocimiento compartido" propio de las ontologías, es lógico que la creación de una ontología esté fundamentada en otras ya establecidas. Más aún en el contexto de la eficiencia energética, donde se depende de la coordinación de múltiples fuentes de datos (sensores, protocolos, normativas) y actores (sistemas de climatización, redes eléctricas, usuarios). Al ofrecer un modelo unificado y compartido, no solo se resuelven desafíos de interoperabilidad, sino que se habilita una base para optimizar decisiones, simular escenarios y garantizar la coherencia en un dominio marcado por la diversidad tecnológica.

Estrechamente relacionado con el concepto de ontología, surge la Web Semántica como una evolución natural en la representación del conocimiento. Desde sus orígenes en 1990, la Web ha evolucionado hasta permitir la vinculación y el entendimiento de información a través de datos enlazados [16], lo que ha permitido que la infraestructura de la red se convierta en un entorno inteligente capaz de integrar información de diversas fuentes y compartirla entre distintos destinatarios, ya sean humanos o máquinas. En este contexto, las ontologías juegan un papel fundamental, ya que proporcionan las herramientas necesarias para llevar a cabo la tarea de interoperabilidad entre diversos sistemas. La Web Semántica, por su parte, se consolida como un elemento clave para la integración de información compleja en entornos digitales e infraestructuras IoT. Para ello, hace uso de modelos semánticos como RDF, RDFS y OWL [17], que permiten la codificación y exposición de ontologías en la Web, así como ser serializados, además, en formatos como XML, JSON-LD, Manchester o TTL.

Estos lenguajes, a su vez, se fundamentan en la lógica descriptiva (DL), que se utiliza como base formal para especificar con una sintaxis precisa los vocabularios de las ontologías. Las lógicas descriptivas son variantes de la lógica de primer orden, que ofrece un compromiso entre expresividad y escalabilidad. Esto da lugar a diferentes subconjuntos de lenguajes en DL, como DL Full, DL y DL Lite, donde cada uno define un lenguaje descriptivo que especifica lo que se permite hacer o las restricciones de los operadores, condicionando de esta forma el rendimiento del razonamiento.

OWL dispone de varios sub-lenguajes en función de la expresividad que se quiera conseguir, así como tres perfiles (OWL2 EL, OWL2 QL y OWL2 RL) dependiendo del razonamiento que se quiera utilizar [18]. Cabe destacar el proceso de razonamiento e inferencia que se hace en OWL2, ligado a la arquitectura de su base de conocimiento (Figura 2.2). La arquitectura separa la información en dos componentes denominados TBox y ABox. El primero define los conceptos y las relaciones (la terminología del dominio), y el segundo contiene las aserciones sobre instancias concretas.

Este enfoque permite, por un lado, comprobar la jerarquía de clases mediante la equivalencia, la satisfacibilidad de los conceptos (si son válidos) y la pertenencia de instancias a clases y, por otro, ofrecer un proceso de inferencia de nuevo conocimiento a través de la definición de reglas.

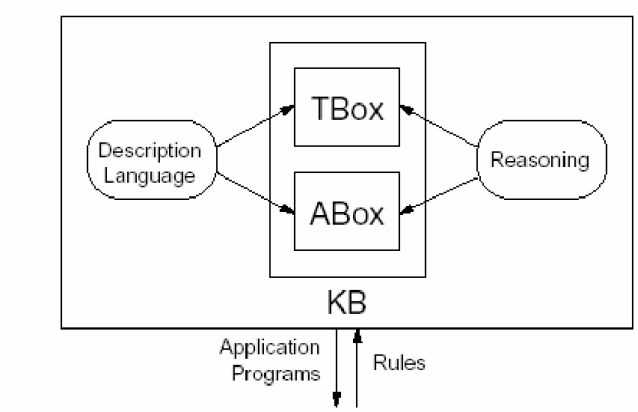


Figura 2.2: Architectura TBox-ABox
[19]

2.2. Gestión Energética en Edificios

Los edificios representan uno de los mayores puntos de consumo de energía en Europa, siendo los sistemas de calefacción los responsables del mayor porcentaje de esta demanda (Figura 2.3). Como consecuencia, la gestión energética en edificios se ha convertido en un aspecto crítico en el contexto global de la sostenibilidad y la eficiencia, donde el principal objetivo es reducir el consumo energético y maximizar el rendimiento de los sistemas instalados. Por este motivo se ha abordado esta temática en este TFG. No obstante, antes de describir en qué consiste la solución propuesta, es imprescindible examinar la evolución que ha experimentado este sector y la situación actual en que se encuentra.

Uno de los primeros pasos consiste en el diseño del edificio. En cualquier proyecto de construcción se lleva a cabo un proceso denominado *document-centric* (centrado en los documentos). Estos documentos son generados a medida que el proyecto avanza y deben ser considerados continuamente debido a posibles dependencias entre ellos. La metodología empleada en la gestión de esta documentación determina en gran medida la eficacia de los procesos y, por tanto, la calidad del resultado final.

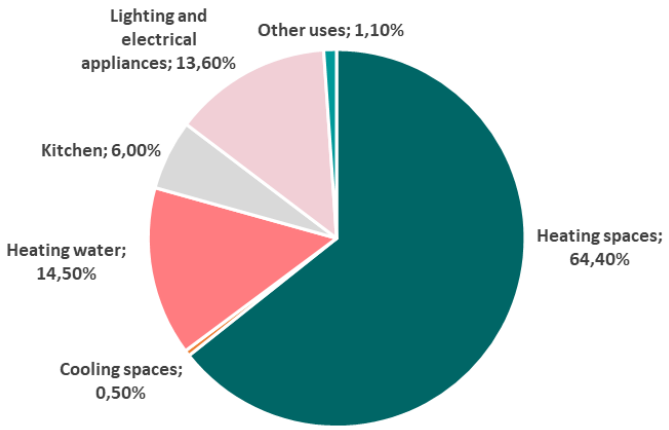


Figura 2.3: Consumo de energía en hogares europeos.
Fuente: Norvento Enerxía.

Aquí es donde BIM emerge como un paradigma transformador, pues ofrece un marco metodológico - respaldado por normativas internacionales [20] y estándares nacionales [21] - que sistematiza la creación, gestión e intercambio de información estructurada sobre un activo construido, desde su concepción hasta su demolición o reutilización. La importancia de BIM radica en hacer que los diferentes actores implicados en el proyecto (arquitectos, ingenieros, gestores, ...) converjan en una misma plataforma que permita eliminar silos de información y errores al centralizar la información, así como hacerla accesible y actualizable por todas las partes interesadas. Esta información puede describir cada elemento por medio de sus atributos (costes, materiales, restricciones, relaciones con otros atributos, ...), permitiendo simular escenarios, optimizar recursos o detectar problemas.

Así como BIM representa la metodología empleada para la elaboración de proyectos de construcción e instalaciones (Figura 2.4), IFC es un estándar libre que funciona como un lenguaje común e independiente del software específico, lo que facilita la colaboración en proyectos bajo la metodología BIM. El estándar IFC está publicado bajo la norma ISO 16739 [22] y su esquema de datos se define mediante el lenguaje de modelado EXPRESS (ISO 10303-11). Este lenguaje permite describir la información de un activo construido [23] gracias a estructurar entidades, relaciones y reglas de integridad en un formato legible. Además, para facilitar el intercambio de una manera confiable, se suele acompañar de un XSD que valida la sintaxis y semántica de los archivos IFC en formato XML (Figura 2.5).

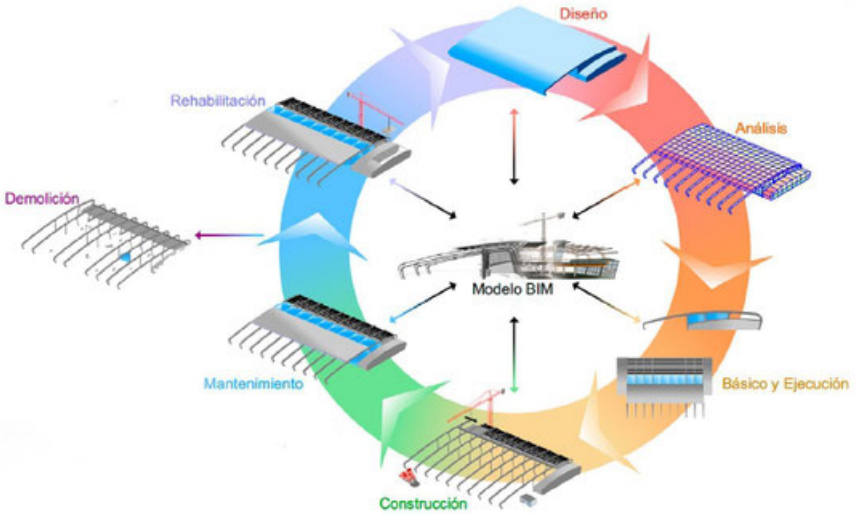


Figura 2.4: Ciclo de vida del Modelo BIM.
Fuente: buildingSmart.

```
#497= IFCFACE((#496));
#499= IFCCLOSEDSHELL((#451,#463,#470,#477,#487,#492,#497));
#501= IFCFACETEDBREP(#499);
#502= IFCSTYLEDITEM(#501,(#180),$);
#505= IFCCARTESIANPOINT((50.,-3.,-0.15));
#507= IFCCARTESIANPOINT((24.,-3.,-0.15));
#509= IFCCARTESIANPOINT((50.,-11.,-0.15));
```

Figura 2.5: Ejemplo de sintaxis en IFC.
Fuente: buildingSmart.

2.2.1. Sistemas de Gestión

La incorporación de fuentes de energía renovable en los edificios, como los paneles solares o los sistemas de geotermia, contribuye a alcanzar los objetivos de sostenibilidad establecidos por la UE. No obstante, su implementación exige una integración cuidadosa con la infraestructura ya existente. Por ello, sistemas como los BAS, BEMS, BMS, EMCS, EMS o EPMS buscan automatizar el control del equipamiento mecánico y eléctrico, como puertas automáticas, redes de fontanería y conductos o sistemas HVAC, supervisando el consumo energético mediante la recopilación de datos en tiempo real para, de esta forma, generar recomendaciones orientadas a optimizar la eficiencia energética del edificio [24, 25, 26, 27, 28]. Esta integración de diferentes sistemas de gestión forma parte de la estrategia promovida por la UE, cuya finalidad es disponer de diferentes infraestructuras y fuentes de energía que permitan la optimización de los recursos gracias a la digitalización y la computación [29].

La evolución experimentada en la construcción de edificios inteligentes como consecuencia de la digitalización e impulsada, a su vez, por los dispositivos IoT y su interconexión a través de la Web, trae consigo la existencia de diferentes dispositivos (como actuadores o sensores) que acceden y modifican fuentes de información diferentes, lo que dificulta la integración de protocolos (ZigBee, BACnet y MQTT) para la monitorización y automatización de los sistemas. Este problema de incompatibilidad puede ser resuelto gracias al uso de ontologías, ya que permiten representar los datos independientemente y conseguir que toda la información sea interpretada de la misma forma, tanto para los dispositivos como para los sistemas y los protocolos.

2.3. Convergencia entre Ontologías y la Gestión Energética

Para terminar con este capítulo, se va poner en relación los dos apratados vistos. Por un lado, se ha mostrado cómo las ontologías proporcionan un marco formal para describir conocimiento, y por otro qué implicaciones aparecen en la operación eficiente de la energía en los edificios. Aplicando las ontologíais se evita la ambigüedad propia de los BAS/BMS, en los que cada proveedor introduce su propia nomenclatura. Al exponer posteriormente los datos mediante servicios de consultas como SPARQL, los sistemas de gestión energética pueden consultar, fusionar y razonar sobre la información sin acoplarse a protocolos propietarios. El resultado es un ecosistema en el que los elementos instalados en el edificio se alineen automáticamente con los conceptos de la ontología definidos. Entonces, para llevar a capo esto lo que se debe de hacer es investigar el vocabulario que

el sistema de gestión energética va a utilizar, y buscar y reutilziar una ontología ya desarrollada. Aunque hay veces que no se utiliza una única ontología y recupilar los conceptos de varias se hace obligatior, es entonces cuando se debe aplicar procesos de alineamiento o crear nuevos conceptos heredados de los definidios en dichas ontologías. La sección 6.3 del Capítulo 6 se abordará las ontologías escogidas y en Capítulo 7 se expondrá el mapeo entre IFC y los conceptos de la ontología utilizada.

Capítulo 3

Metodología

En los últimos 30 años se han desarrollado metodologías específicas para la creación de modelos ontológicos. Estas metodologías se encuentran a medio camino entre los métodos presentes para la creación de bases de conocimiento, como CommonKADS o IDEAL, y los métodos empleados durante el proceso de desarrollo del software, como el Proceso Unificado de Desarrollo.

Entre las metodologías utilizadas para la creación de ontologías destacan Methontology, NeOn Methodology, Stuart, On-To-Knowlodge, DILIGENT, o los procesos definidos por Mike Uschold, Michael Gruninger y Martin King para la obtención de conocimiento [30, 31].

En este trabajo se ha optado por seguir el enfoque propuesto por OEG [32, 33] en Methontology. Esta metodología sigue una distribución de actividades parecida a la propuesta por Uschold, Gruninger y King, pero no define el modelo de proceso seguir. En su lugar, establece el uso de modelos incrementales y evolutivos, ya que permiten incluir definiciones no planeadas y dar soporte para la creación del modelo ontológico en base a las necesidades del proyecto. Aunque esta metodología está enfocada a la creación de ontologías de dominio, se ha decidido hacer uso de ella para declarar los conceptos que se quieren capturar en la ontología de aplicación del presente trabajo.

En esta metodología aparecen una serie de actividades y tareas entre las que existen dependencias intrínsecas, si bien es cierto que muchas de ellas no requieren ser realizadas en orden. En la Figura 3.1 se puede apreciar cómo se relaciona la fase de planificación con las etapas y las actividades necesarias para la construcción del modelo ontológico.

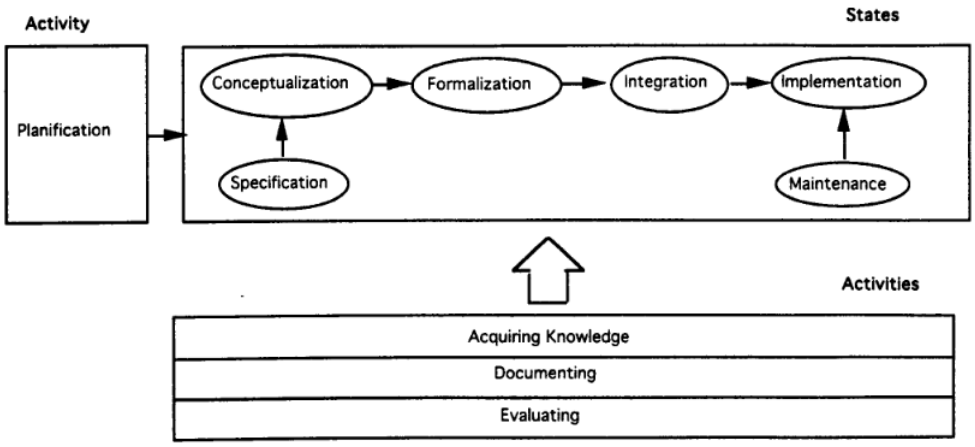


Figura 3.1: Actividades y Etapas de Methontology [32]

En primer lugar, es necesario planificar las actividades con el fin de asignar su duración y los recursos necesarios. Estas actividades consisten en:

1. **Adquisición de Conocimiento**: consulta de libros, artículos, expertos en la materia y todo el material que sirva para completar información sobre el entorno en el que se enmarca el problema a resolver.

2. **Documentación** de todo el proceso realizado: especificación de requisitos, representaciones intermedias, formalización, implementación y evaluación.

3. **Evaluación** del modelo ontológico obtenido mediante la validación del léxico, la sintaxis utilizada y su contenido.

El siguiente paso consiste en llevar a cabo un conjunto de etapas cuya ejecución estará condicionada por la forma en que se han planificado de las actividades. Estas etapas son:

- **Especificación**: producir un documento en lenguaje natural con un estilo informal, semiformal o formal, pero que incluya el propósito, el alcance y el nivel de formalismo utilizado.

- **Conceptualización**: crear un glosario de términos del dominio con el que poder construir después un Diccionario de Datos, un árbol de clasificación de conceptos, una tabla de atributos, una tabla de clases, una tabla de instancias, una tabla de constantes, una tabla de fórmulas, un árbol de clasificación de atributos.

- **Formalización**: transformar el modelo conceptual en una descripción detallada haciendo uso de la lógica descriptiva o de lenguajes basados en frames.
- **Integración**: inspeccionar otras ontologías de las que se puedan reutilizar conceptos para evitar su creación desde cero y aumentar así la velocidad en su desarrollo.
- **Implementación**: elegir un lenguaje con el que codificar los conceptos definidos. Esta etapa suele requerir de un analizador sintáctico y semántico, un traductor y buscador para investigar definiciones de conceptos en otros lenguajes, un editor para añadir o eliminar definiciones y un evaluador del conocimiento.
- **Mantenimiento**: una vez creada y publicada la ontología, se comparte y se corrigen los errores encontrados en la definición de sus conceptos.

La metodología aconseja seguir un proceso iterativo (o evolutivo) con el fin de controlar la incertidumbre del proyecto. Se ha considerado un nivel moderado de incertidumbre debido a la poca experiencia en el desarrollo de este tipo de proyectos. Estos motivos son los que han llevado a optar por un modelo evolutivo, ya que permite ir elaborando prototipos por fases (controlando la incertidumbre), conseguir un desarrollo rápido y tener una retroalimentación de los tutores con la que poder gestionar los cambios. A continuación se enumeran los pasos que Methontology sugiere seguir para el desarrollo de modelos ontológicos:

1. Planificación: establecer duración de actividades y adecuar el volumen de trabajo.
2. Obtención de conocimiento: referencias para obtener información.
3. Desarrollo de la ontología:
    - Especificación de los requisitos de la ontología.
    - Conceptualización basada en la construcción de un glosario de términos.
        - Elaborar diccionario de datos.
        - Elaborar árbol grafo de clasificación de conceptos.
        - Elaborar tabla de atributos de las instancias.
        - Elaborar tabla de atributos de clase.
        - Elaborar tabla de clases.
        - Elaborar tabla de instancias.

    - Elaborar tabla de constantes.
    - Elaborar tabla de fórmulas.
    - Elaborar árbol de clasificación de atributos.
  - Integración de otras ontologías.
  - Implementación.
  - Evaluación
  - Mantenimiento.

4. Documentación del proceso.

La Sección 6 describe con detalle cómo se han llevado a cabo estas actividades propuestas, aunque la elaboración del árbol de clasificación de atributos ha sido descartada por razones de coherencia y adecuación al contexto del trabajo. Cabe destacar que las tareas indicadas en la fase de conceptualización no requieren de un orden explícito, pero este ha sido el que se ha seguido durante la etapa de desarrollo. Por otro lado, mencionar que la tarea que consiste en elaborar el árbol de clasificación de atributos no ha sido realizada, ya que carece de sentido detallar la relación entre propiedades de las entidades cuando estas relaciones no entran en juego en la ejecución del servicio. Asimismo, destacar en este trabajo la fusión de la tabla de atributos de clase y atributos en una única tabla de propiedades para no generar un gran número de tablas, algunas de ellas con muchos huecos vacíos, e intentar facilitar así su lectura.

3.1. Planificación y Alcance

En este apartado se muestra la planificación del proyecto y se describe su alcance. Es importante no confundirlo con el alcance de la ontología. El primero hace referencia a cuánto trabajo se va a realizar en el proyecto. El segundo, en cambio, expone el número de conceptos tratados por la ontología. El alcance del proyecto, además, está condicionado por el tiempo disponible para completarlo. En este caso, el proyecto se inició el 27 de enero de 2025 y ha finalizado el 13 de junio de 2025, lo que supone una duración de 20 semanas. De este total, se dedicaron dos semanas para realizar la planificación y otras dos, la semana de vacaciones de Semana Santa y otra adicional, para corregir problemas surgidos durante el desarrollo. Como consecuencia, se obtiene un período de 16 semanas desde que se empezó a desarrollar en CARTIF la parte técnica del TFG, lo que da lugar a 8 iteraciones teniendo en cuenta una duración de 2 semanas por iteración.

Para organizar las actividades, tareas y artefactos a realizar, se ha creado una estructura de desglose de trabajo basada en un enfoque híbrido entre WBS y PBS. De esta forma, se pretende detallar todo el trabajo necesario, identificando de manera anticipada las actividades y asegurando su realización. La Figura 3.2 muestra la estructura creada, donde las actividades corresponden a los nodos en blanco con texto en negrita, y los artefactos a los nodos en color cyan con texto en cursiva.

3.1.1. Cronograma

La Tabla 3.1 muestra el cronograma elaborado al comienzo del proyecto, donde se puede observar la duración de cada incremento, las actividades a realizar y la fecha estimada de finalización.

Actividad	**Duración**	**Comienzo**	**Fin**
Planificación	10 días	lun 27/01/25	vie 07/02/25
Definición de objetivos	3 días	lun 27/01/25	mié 29/01/25
Definición de alcance	2 días	jue 30/01/25	vie 31/01/25
Creación del cronograma	4 días	lun 03/02/25	jue 06/02/25
Investigación previa	10 días	lun 27/01/25	vie 07/02/25
Desarrollo del sistema	90 días	lun 10/02/25	vie 13/06/25
Desarrollo del sistema 1	10 días	lun 10/02/25	vie 21/02/25
Investigación	2 días	lun 10/02/25	mar 11/02/25
Desarrollo de la ontología	3 días	mar 11/02/25	jue 13/02/25
Desarrollo del servicio	5 días	vie 14/02/25	jue 20/02/25
Validación	2 días	jue 20/02/25	vie 21/02/25
Release del incremento	0 días	vie 21/02/25	vie 21/02/25
Desarrollo del sistema 2	10 días	lun 24/02/25	vie 07/03/25
Investigación	2 días	lun 24/02/25	mar 25/02/25
Desarrollo de la ontología	3 días	mar 25/02/25	jue 27/02/25
Desarrollo del servicio	5 días	vie 28/02/25	jue 06/03/25
Validación	2 días	jue 06/03/25	vie 07/03/25
Release del incremento	0 días	vie 07/03/25	vie 07/03/25
Desarrollo del sistema 3	10 días	lun 10/03/25	vie 21/03/25
Investigación	2 días	lun 10/03/25	mar 11/03/25
Desarrollo de la ontología	3 días	mar 11/03/25	jue 13/03/25

Desarrollo del servicio	5 días	vie 14/03/25	jue 20/03/25
Validación	2 días	jue 20/03/25	vie 21/03/25
Release del incremento	0 días	vie 21/03/25	vie 21/03/25
Desarrollo del sistema 4	10 días	lun 24/03/25	vie 04/04/25
Investigación	2 días	lun 24/03/25	mar 25/03/25
Desarrollo de la ontología	3 días	mar 25/03/25	jue 27/03/25
Desarrollo del servicio	5 días	vie 28/03/25	jue 03/04/25
Validación	2 días	jue 03/04/25	vie 04/04/25
Release del incremento	0 días	vie 04/04/25	vie 04/04/25
Desarrollo del sistema 5	20 días	lun 07/04/25	vie 02/05/25
Investigación	2 días	lun 07/04/25	mar 08/04/25
Desarrollo de la ontología	3 días	mar 08/04/25	jue 10/04/25
Desarrollo del servicio	14 días	vie 11/04/25	mié 30/04/25
Validación	3 días	mié 30/04/25	vie 02/05/25
Release del incremento	0 días	vie 02/05/25	vie 02/05/25
Desarrollo del sistema 6	10 días	lun 05/05/25	vie 16/05/25
Investigación	2 días	lun 05/05/25	mar 06/05/25
Desarrollo de la ontología	3 días	mar 06/05/25	jue 08/05/25
Desarrollo del servicio	5 días	vie 09/05/25	jue 15/05/25
Validación	2 días	jue 15/05/25	vie 16/05/25
Release del incremento	0 días	vie 16/05/25	vie 16/05/25
Desarrollo del sistema 7	10 días	lun 19/05/25	vie 30/05/25
Investigación	2 días	lun 19/05/25	mar 20/05/25
Desarrollo de la ontología	3 días	mar 20/05/25	jue 22/05/25
Desarrollo del servicio	5 días	vie 23/05/25	jue 29/05/25
Validación	2 días	jue 29/05/25	vie 30/05/25
Release del incremento	0 días	vie 30/05/25	vie 30/05/25
Desarrollo del sistema 8	10 días	lun 02/06/25	vie 13/06/25
Investigación	2 días	lun 02/06/25	mar 03/06/25
Desarrollo de la ontología	3 días	mar 03/06/25	jue 05/06/25
Desarrollo del servicio	5 días	vie 06/06/25	jue 12/06/25
Validación	2 días	jue 12/06/25	vie 13/06/25
Release del incremento	0 días	vie 13/06/25	vie 13/06/25

Documentación	100 días	lun 27/01/25	vie 13/06/25
Elaborar memoria	100 días	lun 27/01/25	vie 13/06/25
Elaborar defensa	10 días	sáb 14/06/25	jue 26/06/25
Revisar memoria	2 días	sáb 14/06/25	lun 16/06/25
Revisar sistema	2 días	lun 16/06/25	mar 17/06/25
Crear presentación	4 días	mar 17/06/25	vie 20/06/25
Ensayar presentación	4 días	vie 20/06/25	mié 25/06/25
Realizar defensa	0 días	jue 26/06/25	jue 26/06/25

Tabla 3.1: Cronograma inicial del proyecto.

Como se puede comprobar, al tratarse de un desarrollo evolutivo, la planificación inicial no es acorde al trabajo realizado. Esto se debe a que, conforme avanzaba el proyecto, ha sido necesario ir perfilando el objetivo final que pretendía alcanzar la empresa, repercutiendo en un reajuste del tiempo disponible para realizar las actividades. Por esta razón, las actividades han ido cambiando a lo largo de los meses. Un claro ejemplo es el cambio de concepto de crear una API para el acceso a la ontología y al servicio, a crear un servicio elaborado con scripts de Python.

3.1.2. Gestión de riesgos

Durante la planificación inicial del proyecto es fundamental analizar los posibles riesgos que pueden afectar a su desarrollo. Un riesgo se entiende como un evento o una condición incierta que, de materializarse, podría influir negativamente en el desarrollo del proyecto. Por ello, anticiparse a estos escenarios es clave para garantizar su éxito.

Este apartado se centra en identificar los riesgos que se pueden producir, definir estrategias que permitan minimizar su impacto o reducir la probabilidad de que sucedan, y establecer planes de contingencia que permitan actuar de manera eficaz en caso de que lleguen a materializarse. Los riesgos identificados, de menor a mayor orden de importancia, aparecen reflejados en la Tabla 3.2. Las Tablas 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, por su parte, muestran una descripción detallada de cada uno de ellos.

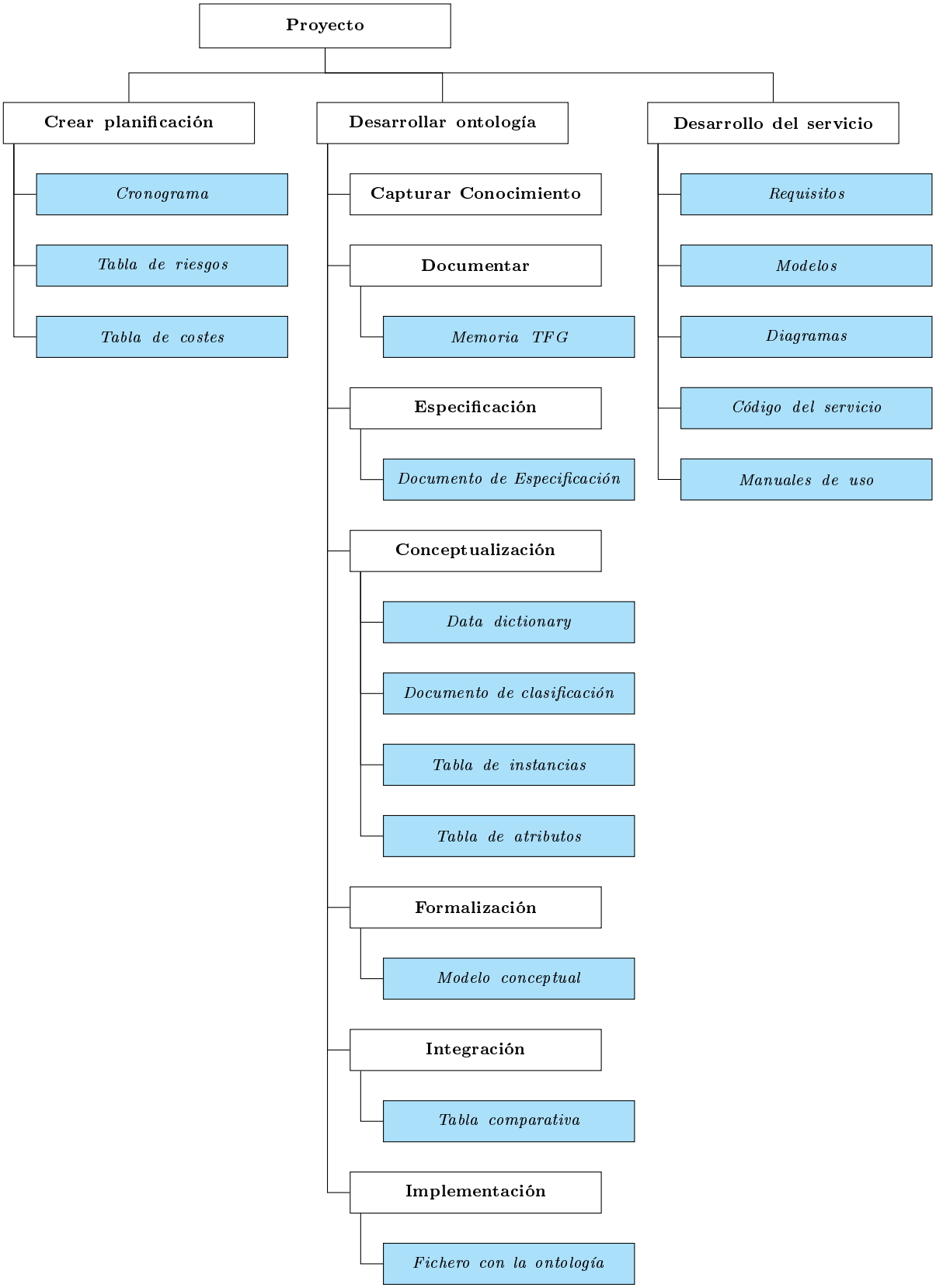


Figura 3.2: Enfoque híbrido entre un WBS y un PBS del trabajo a realizar.

Riesgo	**Impacto**
RK006	Problemas con los equipos informáticos.
RK005	Estimación incorrecta del tiempo.
RK004	Cambios de última hora.
RK002	Desarrollar las funcionalidades software incorrectas.
RK003	Gold plating.
RK001	No tener claro el alcance y los objetivos.

Tabla 3.2: Riesgos identificados ordenados por probabilidad de impacto.

ID	RK001
Título	No tener claro el alcance y los objetivos.
Tipo	Definición de requisitos.
Descripción	Falta de una definición concisa de objetivos y alcance desde el inicio que puede generar malentendidos con los tutores y provocar desviaciones en el cronograma.
Estrategia	Prevención
Plan de mitigación	Definición temprana de alcance y objetivos con los tutores.
Plan de contingencia	Implementar reuniones semanales para comprobar que se trabaja persiguiendo los objetivos y que la carga de trabajo es adecuada al alcance.

Tabla 3.3: Riesgo RK001.

ID	RK002
Título	Desarrollar las funcionalidades software incorrectas.
Tipo	Desarrollo técnico.
Descripción	Implementación de características que no se adecuan con las necesidades reales del proyecto.
Estrategia	Evitación
Plan de mitigación	Definir en cada incremento el trabajo a realizar.
Plan de contingencia	Usar prototipos iterativos con los que validar el trabajo realizado en cada hito.

Tabla 3.4: Riesgo RK002.

ID	RK003
Título	Gold plating.
Tipo	Calidad
Descripción	Tendencia a añadir funcionalidades extras no solicitadas.
Estrategia	Evitación
Plan de mitigación	Establecer criterios de aceptación claros y realizar revisiones periódicas.
Plan de contingencia	Eliminar características extras.

Tabla 3.5: Riesgo RK003.

ID	RK004
Título	Cambios de última hora.
Tipo	Gestión de cambios.
Descripción	Solicitudes de modificaciones importantes cuando el proyecto está avanzado, afectando la planificación establecida.
Estrategia	Mitigación
Plan de mitigación	Realizar reuniones con las que acordar qué es lo que se quiere en cada incremento.
Plan de contingencia	Usar tiempo reservado para problemas imprevistos y efectuar los cambios.

Tabla 3.6: Riesgo RK004.

ID	RK005
Título	Estimación incorrecta del tiempo.
Tipo	Planificación
Descripción	Errores en la estimación de duración de los incrementos debido a subestimar la complejidad de las tareas.
Estrategia	Aceptación
Plan de mitigación	Ajustarse al calendario planificado.
Plan de contingencia	Priorizar funcionalidades críticas y replanificar.

Tabla 3.7: Riesgo RK005.

ID	RK006
Título	Problemas con los equipos informáticos.
Tipo	Técnico
Descripción	Pérdida de datos por fallos en los ordenadores.
Estrategia	Evitación
Plan de mitigación	Hacer copias de seguridad en la nube con GitLab.
Plan de contingencia	Restaurar una copia de seguridad previa.

Tabla 3.8: Riesgo RK006.

3.1.3. Gestión de costes

En este apartado se lleva a cabo un estudio del coste que ha supuesto para la empresa el trabajo realizado en este TFG. Para ello, se han establecido tres categorías: hardware, software y personal.

En cuanto al hardware, se ha utilizado en la empresa un portátil HP ProBook 440 14 G11 y un monitor HP Compaq LA2306x. En la Tabla A.1 del Apéndice A se pueden consultar las especificaciones técnicas de los equipos. Otro dispositivo hardware que se ha utilizado pero no ha generado coste ha sido una máquina virtual de Oracle Cloud Infrastructure en la que se ha alojado la base de datos para una de las demos.

En lo que respecta al software, todas las herramientas que se han utilizado son gratuitas, a excepción de Visual Studio Code y ChatGPT Plus. Por la licencia de Visual Studio Code ya se pagaba una licencia de 641 €/año para toda la división. Esto representa un total de 267 € en los cinco meses que ha durado el proyecto. Por su parte, la licencia de uso de 1 mes de ChatGPT Plus únicamente ha ocasionado un gasto de 20 €. También se ha hecho uso de la versión gratuita para estudiantes de Java 11.16 de Oracle. Su uso comercial conllevaría un gasto de 15 €/mes, es decir, 75 € en total. Por último, se ha utilizado la herramienta Astah Professional. Este software, aunque dispone de una licencia para estudiantes, puede ocasionar un gasto de 11 €/mes en su versión comercial, es decir, 55 € para el desarrollo de este trabajo.

Finalmente, hay que considerar los gastos relativos al personal de la empresa, es decir, el sueldo que percibe el trabajador. En este caso, el sueldo ha sido de 800 €/mes brutos. Teniendo en cuenta que la fecha de inicio del contrato fue el 17 de febrero de 2025 y la fecha de finalización ha sido el

13 de junio de 2025, el gasto salarial asciende a 4000 €. La Tabla 3.9 muestra el coste total que ha supuesto para la empresa la realización de este proyecto.

Categoría	**Coste (€)**
Hardware (Portátil + Monitor)	1025.00
Software (Licencias)	417.00
Personal (Salario)	4000.00
Total (€)	5442.00

Tabla 3.9: Coste del proyecto para la empresa.

En cambio, si se desea realizar este proyecto de manera independiente, habría que sufragar por cuenta propia el coste del hardware. En este caso, un portátil MSI Prestige 14H B12UCX que ha tenido un coste de 700 € y un monitor Samsung de 200 €. Además, se debería tener en cuenta una serie de costes indirectos presentes en cualquier entorno de trabajo (laboral o doméstico) y que no se han contemplado en los estudios realizados.

3.2. Seguimiento del proyecto

El seguimiento del proyecto se ha gestionado mediante reuniones periódicas tanto con el tutor académico como con la tutora en la empresa. Semanalmente se ha mantenido una reunión informal con el tutor académico para revisar el progreso del TFG, mientras que con la tutora en la empresa se ha realizado una evaluación más detallada del trabajo al finalizar cada incremento. Además, se han mantenido comunicaciones puntuales para resolver dudas y recibir orientación sobre decisiones técnicas o metodológicas.

Aunque el proyecto se ha completado dentro del plazo previsto -incluso con cierta antelación debido a que se terminó el 23 de mayo-, cabe señalar algunos aspectos relevantes que han surgido durante su ejecución. Uno de ellos ha sido la superposición de incrementos, como el ocurrido en la semana 3, donde se juntó la finalización de la planificación con el comienzo del desarrollo de la ontología. Esta circunstancia, junto con el descarte de ideas de desarrollo -como el acceso al servicio a través de una API- derivó en un retraso hasta el incremento 3 por no tener clara la idea de cómo sería el servicio.

Del mismo modo, es importante señalar que algunos *releases* se realizaron los jueves en lugar de los viernes. Esta modificación menor respondió a la necesidad de anticipar la retroalimentación para conocer con mayor detalle si el trabajo realizado era acorde a lo que se requería.

Otro aspecto imprevisto fue la interrupción de la jornada de trabajo el 28 de abril de 2025 debido a un apagón producido en España, que dejó sin suministro eléctrico a muchas zonas desde las 12:00 del mediodía. Aunque este suceso no afectó significativamente al cumplimiento global de los plazos, se considera un incidente relevante no contemplado en el análisis de riesgos del proyecto.

Finalmente, la Tabla 3.10 muestra el cronograma efectivo del proyecto, donde se puede observar la correspondencia entre los incrementos y las diferentes actividades ejecutadas.

Actividad	**Incremento**	**Comienzo**	**Fin**
Planificación	1-3	lun 27/01/25	vie 07/02/25
Estudiar metodologías	1-3	lun 10/02/25	vie 28/02/25
Estudiar ontologías existentes	1-3	lun 10/02/25	vie 21/03/25
Desarrollo de la ontología	3-7	lun 10/03/25	vie 04/04/25
Desarrollo del servicio	3-7	lun 10/03/25	lun 26/05/25
Elaboración de la memoria	1-8	lun 27/01/25	vie 13/06/25

Tabla 3.10: Cronograma efectivo del proyecto realizado.

Capítulo 4

Herramientas Software

En este trabajo se han utilizado diferentes herramientas software, algunas destinadas al análisis y modelado de la información y otras para el desarrollo, documentación, gestión y comunicación del proyecto. La selección de estas herramientas se ha realizado en función de su utilidad y adecuación para el correcto desarrollo del proyecto.

4.1. Herramientas de modelado

Astah Professional

Astah es una herramienta de modelado UML que permite crear diagramas de clases, de casos de uso, de estados, de actividades y de secuencia, entre otros. Se ha utilizado para realizar el modelado del servicio y documentar el desarrollo realizado. Además, gracias a la licencia proporcionada por la Universidad de Valladolid, se ha podido hacer uso de la versión profesional.

Protégé

Protégé es una plataforma gratuita open-source desarrollada por la Universidad de Stanford para la creación y edición de ontologías y bases de conocimiento. Es ampliamente utilizada en el campo de la IA, especialmente para el desarrollo de la web semántica y de los sistemas basados en conocimiento. Soporta varios formatos de ontologías, incluyendo OWL y RDF. En este trabajo se ha utilizado para modificar y visualizar las ontologías generadas con Python, así como aquellas estandarizadas en la industria, como SAREF o BRICK.

4.2. Herramientas de desarrollo

Python, Pyenv y TOML

Python es un lenguaje de programación interpretado, de alto nivel y de propósito general. En este trabajo se ha empleado la versión 3.12.3 para desarrollar el servicio. Con Python se han utilizado TOML y Pyenv (versión 2.5.3) para crear los archivos de configuración y los entornos con los que gestionar las dependencias del proyecto, respectivamente. Además, se han utilizado varias librerías, siendo owlready2 (versión 0.47), ifcopenshell (versión 0.8.1) y psycopg2-binary (versión 2.9.10) las más importantes. El fichero requirements.txt en Anexo Material Complementario **??** muestra las librerías utilizadas y sus dependencias.

Java

Java es un lenguaje de programación orientado a objetos y de propósito general. Se caracteriza por su portabilidad, robustez y seguridad. En este trabajo se ha utilizado para crear un primer prototipo, que posteriormente fue descartado. Además, es necesario para usar Protégé. Se ha hecho uso de la versión 11.16 de Oracle gracias a la licencia proporcionada por la Universidad.

Visual Studio Code

Visual Studio Code (VSC) es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Es gratuito, open-source y ofrece un amplio soporte para el desarrollo de aplicaciones por medio de la instalación de extensiones. Este editor ha sido el escogido para trabajar con Python.

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacionales de código abierto y basado en el estándar SQL. En este proyecto se ha utilizado para almacenar los datos de los sensores y acceder posteriormente a ellos.

Oracle Cloud Infrastructure

Oracle Cloud Infrastructure (OCI) es una plataforma cloud desarrollada por Oracle que proporciona servicios de infraestructura y aplicaciones de manera homogénea en un entorno de alta

disponibilidad. En este trabajo se ha utilizado como infraestructura para alojar una máquina virtual que actúe como servidor de la base de datos PostgreSQL.

Draw.io y Chowlk

Chowlk es una aplicación gratuita que permite crear organigramas, diagramas de flujo, UML, ER y otros diagramas conceptuales. Permite colaboración en tiempo real, integración con servicios de almacenamiento en la nube, como Google Drive y OneDrive, y exportación de los diagramas en múltiples formatos.

Draw.io, en cambio, es una librería y un servicio web. Como librería, se puede importar desde Draw.io para generar diagramas de la ontología, los cuales se introducirán posteriormente en el servicio web de Chowl Linked Data para exportarlos en una serialización de OWL2 en TTL.

Estas herramientas han sido utilizadas en las iteraciones iniciales del proyecto para analizar su funcionamiento, pero conforme avanzaba el desarrollo del mismo se reemplazaron por Python con el fin de utilizar de forma dinámica la ontología en la que se basa el servicio implementado.

4.3. Sistemas Operativos

En este trabajo se han utilizado dos versiones de Ubuntu. Por un lado, la 22.04 LTS de forma nativa en el ordenador personal y, por otro, la 24.04.1 LTS en Windows Subsystem for Linux (WSL). sobre Windows 11 Pro.

4.4. Herramientas de navegación y documentación

Buscador Web

Se ha utilizado Firefox, navegador web gratuito y de código abierto que destaca por aspectos como la privacidad, capacidad de personalización y uso de estándares web abiertos. En este trabajo se ha utilizado principalmente para la búsqueda de información en la web y la visualización de ontologías en texto plano.

Editor de textos

Aplicación que permite crear y editar archivos de texto plano. A diferencia de los procesadores de texto, no incluye formato de texto avanzado. En este trabajo ha sido de gran utilidad para editar archivos de configuración, modificar ficheros escritos en markdown y explorar las ontologías en texto plano. Como editores utilizados han sido el Bloc de notas de Windows y gedit de Linux.

Texmaker

Texmaker es un editor gratuito y multiplataforma que integra las herramientas necesarias para desarrollar documentos en LaTeX. Es ampliamente utilizado en los ámbitos académico y científico, ya que permite crear documentos de forma estructurada y con aspecto profesional. Se ha utilizado para redactar el presente documento, ya que ofrece, entre otros aspectos, resaltado de sintaxis, corrector ortográfico y visor de PDF integrado, lo que facilita la escritura y generación del documento final.

4.5. Herramientas de gestión y control

Git y GitLab

Git es un sistema de control de versiones distribuido que permite rastrear cambios en el código fuente durante el desarrollo del proyecto. GitLab, por su parte, es una plataforma web basada en Git que ofrece alojamiento de repositorios y otras herramientas de colaboración. En este trabajo se ha utilizado Git 2.43.0 junto con GitLab como servicio en la nube, gracias a una licencia educativa. El uso de estas herramientas ha permitido guardar el trabajo de forma progresiva en un repositorio remoto, facilitando el seguimiento de los cambios por parte de los tutores.

ProjectLibre

ProjectLibre es una alternativa de código abierto a Microsoft Project. Facilita la gestión de proyectos mediante la creación de diagramas de Gantt, estructuras de desglose de trabajo, diagramas de red, gestión de recursos y de costes. Además, es multiplataforma y compatible con archivos creados en Microsoft Project, lo que facilita la transición desde este software comercial. En este trabajo se ha utilizado principalmente para planificar el tiempo de las iteraciones en la fase inicial del proyecto.

4.6. Herramientas de comunicación

Outlook

Outlook es un cliente de correo electrónico desarrollado por Microsoft para la gestión de correo, calendario, contactos y tareas. Durante el desarrollo del proyecto se ha gestionado toda la comunicación mediante esta aplicación en su versión web.

Rocket.Chat

Rocket.Chat es una herramienta de código abierto para mensajería instantánea en el entorno de trabajo. Su uso ha permitido mantener una interacción más ágil con la tutora en la empresa.

4.7. Herramientas de IA Generativa

ChatGPT, DeepSeek y Claude

Se ha hecho uso de versiones gratuitas de herramientas de inteligencia artificial generativa, como DeepSeek o Claude, para acelerar la redacción de texto, resolver dudas complejas, corregir errores en el código y desarrollar parte del mismo. Además el último mes de trabajo se pagó la versión de ChatGPT Plus para probar su funcionamiento y los servicios que ofrece dicha versión.

Capítulo 5

Especificación de requisitos

5.1. Requisitos

En esta sección se detallan los requisitos relativos al comportamiento del servicio, así como las condiciones sobre las que debe ejecutarse.

5.1.1. Requisitos funcionales

En el contexto del software y los sistemas, los requisitos funcionales describen las acciones, comportamientos, funcionalidades o tareas que un sistema debe realizar para cumplir su propósito. Estos requisitos deben garantizar que el software cumpla con las expectativas de las partes interesadas y satisfaga las necesidades de negocio previstas. La Tabla 5.2 muestra los requisitos funcionales que se han definido para el servicio desarrollado.

5.1.2. Requisitos no funcionales

Los requisitos no funcionales se refieren a los atributos de calidad de un sistema que definen su rendimiento, no sus funciones. A diferencia de los requisitos funcionales, que especifican las acciones y tareas que debe realizar un sistema, los requisitos no funcionales se centran en las características generales y el comportamiento del sistema en diversas condiciones. Abordan aspectos como el rendimiento, la usabilidad, la fiabilidad y la escalabilidad, garantizando que el sistema cumpla con los estándares de calidad y proporcione una experiencia de usuario satisfactoria. La Tabla 5.4 muestra los requisitos no funcionales definidos para el servicio.

ID	Nombre	Descripción
RF01	Generar ontología	El sistema deberá ser capaz de procesar el fichero ifc y convertir los datos del edificio a datos de la ontología de partida.
RF02	Acceso a Base de Datos	El sistema deberá acceder a una base de datos relacional de donde extraer los valores de monitorización de sensores.
RF03	Lectura Datos de ontología	El sistema deberá leer la ontología generada en formato RDF.
RF04	Toma decisiones	El sistema deberá decidir a qué actuador mandar la señal de activación/desactivación según el estado térmico observado.
RF05	Actualizar umbral	El sistema deberá permitir modificar el umbral que se quiere usar.
RF06	Ejecución repetida	El servicio deberá ejecutarse cada cierto intervalo de tiempo.

Tabla 5.2: Requisitos funcionales del sistema.

ID	Nombre	Descripción
RNF01	Usabilidad	El usuario únicamente ejecutará el servicio por consola.
RNF02	Actuación	El sistema usará un estándar abierto para abrir o cerrar las válvulas de los actuadores.

Tabla 5.4: Requisitos no funcionales del sistema.

5.1.3. Requisitos de información

Los requisitos de información son especificaciones de las necesidades y expectativas que debe cumplir un sistema de información para satisfacer las necesidades de los usuarios y el negocio. Estos requisitos definen qué información debe recolectar, procesar, almacenar y comunicar el sistema. La Tabla 5.6 muestra los requisitos de información definidos para el servicio.

ID	Nombre	Descripción
RI01	Edificio	El sistema debe almacenar la información del edificio.
RI02	Espacios	El sistema debe almacenar la información de cada espacio del edificio.
RI03	Plantas	El sistema debe almacenar la información de cada planta del edificio.
RI04	Sensores	El sistema debe almacenar la información de todos los sensores del edificio.
RI05	Actuadores	El sistema debe almacenar la información de todos los actuadores del edificio.
RI06	Umbral	El sistema debe almacenar los valores del umbral a utilizar.
RI07	Temperatura mínima	El sistema debe almacenar el valor de la temperatura mínima.
RI08	Temperatura máxima	El sistema debe almacenar el valor de la temperatura máxima.

Tabla 5.6: Requisitos de información del sistema.

5.1.4. Restricciones

Las restricciones en el desarrollo software son imposiciones o límites que se generan al desarrollar el producto. Estas restricciones pueden venir dadas por las decisiones tomadas en la arquitectura, por la elección de la tecnología o por tener que cumplir con alguna ley de seguridad o normas de estilo. La Tabla 5.8 muestra las restricciones sobre las que debe funcionar el servicio.

ID	Nombre	Descripción
RT01	Datos monitorización externos	El sistema debe de almacenar los valores numéricos de los datos proporcionados por los sensores en una base de datos externa a la ontología.
RT02	Acceso PostgreSQL	El sistema debe de utilizar a una base de datos PostgreSQL donde recoger los valores proporcionados por los sensores.

Tabla 5.8: Restricciones del sistema.

5.2. Casos de Uso

5.2.1. Diagrama de Casos de Uso

La Figura 5.1 muestra el diagrama de casos de uso con las principales acciones que puede realizar un usuario para utilizar el sistema.

5.2.2. Descripción de los Casos de Uso

Las tablas 5.9, 5.11 y 5.13 describen con detalle en qué consiste cada uno de los casos de uso.

Id	CU01
Nombre	Iniciar Servicio
Descripción	Permite al usuario iniciar el servicio de control de climatización, procesando un fichero IFC de entrada, recolectando datos de sensores y actuadores, y aplicando el umbral de temperatura para abrir o cerrar válvulas.
Actor	Usuario desarrollador
Precondición	Se debe haber poblado la base de datos PostgreSQL con los identificadores de los sensores y actuadores del edificio.

Secuencia Principal	1. El actor ejecuta el servicio desde la terminal con el parámetro `--output_rdf`. 2. El sistema comprueba que existe el fichero especificado en el argumento de `--output_rdf`. 3. El sistema recoge del fichero `pyproject.toml` la configuración con la base de datos. 4. El sistema recoge del fichero `pyproject.toml` la configuración con la base de datos. 5. El sistema recoge del fichero `pyproject.toml` la configuración del intervalo entre ejecuciones. 6. El sistema recoge del fichero `pyproject.toml` la configuración del umbral y las temperaturas mínima y máxima a utilizar. 7. El sistema se conecta con la base de datos PostgreSQL y almacena los valores indicados. 8. El sistema recupera todos los espacios del edificio del RDF. 9. El sistema obtiene del RDF, por cada espacio, los sensores y sus identificadores en la base de datos. 10. Con esos IDs, el sistema recupera de la base de datos los datos de temperatura. 11. El sistema aplica la función de activación con el umbral para decidir si abre o cierra cada válvula.

Postcondición	1. Las válvulas quedan ajustadas según el umbral. 2. El caso de uso se ejecuta de manera indefinida cada intervalo de tiempo definido en `pyproject.toml` hasta que el usuario lo detenga.
Alternativas	1. El actor puede especificar un nuevo valor para el umbral con el parámetro *—update_ threshold*. En este caso se ejecuta el CU02. 2. Si el fichero no existe, el sistema solicita al usuario que proporcione un fichero IFC. 3. El actor introduce el nombre del fichero IFC para iniciar la transformación. 4. El sistema comprueba que el fichero existe. Si no existe, informa del error y el caso de uso queda sin efecto. 5. El sistema confirma que el fichero existe y se ejecuta el CU03.

Tabla 5.9: Especificación del Caso de Uso: Iniciar Servicio.

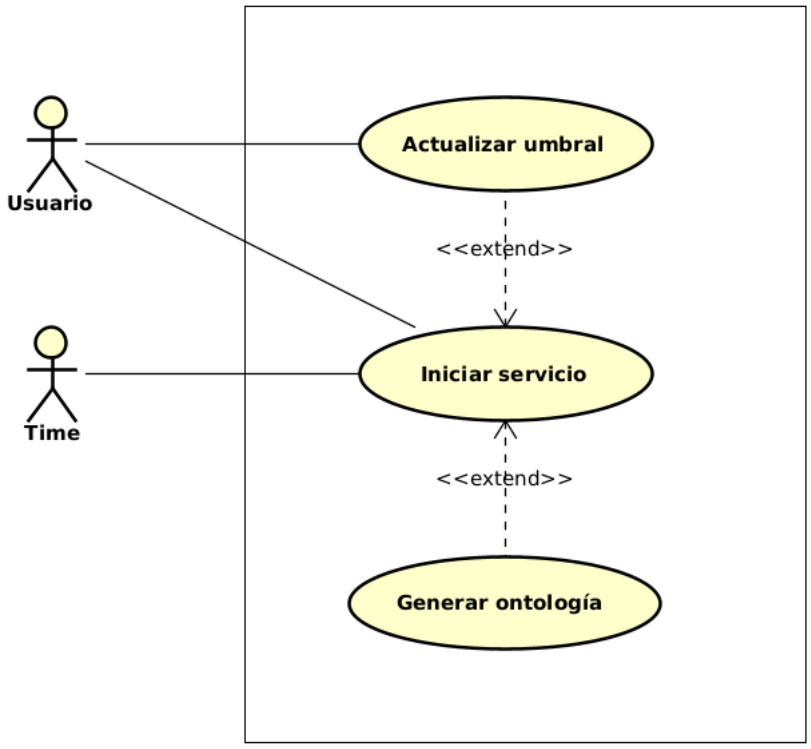


Figura 5.1: Casos de Uso del Servicio.

Id	CU02
Nombre	Cambiar Umbral Temperatura
Descripción	Permite actualizar el valor del umbral de temperatura utilizado por el servicio.
Actor	Usuario desarrollador
Precondición	No requiere precondiciones.
Secuencia Principal	1. El actor introduce `--update_threshold` con un valor. 2. El sistema comprueba que el valor proporcionado es un float positivo. 3. El sistema actualiza el umbral en la base de datos.
Postcondición	El nuevo valor del umbral se almacena en la base de datos PostgreSQL.
Alternativas	2a. Si el valor es negativo, el sistema muestra un mensaje de error y el caso de uso queda sin efecto. 2b. Si el valor es una cadena, el sistema muestra un mensaje de error y el caso de uso queda sin efecto.

Tabla 5.11: Especificación del Caso de Uso: Cambiar Umbral Temperatura.

Id	CU03
Nombre	Generar Ontología
Descripción	Genera un fichero RDF de la ontología a partir de un IFC de entrada, escogiendo las entidades relevantes del modelo de dominio.
Actor	Usuario desarrollador
Precondición	Debe existir el fichero IFC especificado como entrada.
Secuencia Principal	1. El sistema lee el fichero IFC de entrada. 2. El sistema lo abre usando la librería IfcOpenShell. 3. El sistema convierte a RDF las entidades del edificio (Sensores, Actuadores, Edificio, Plantas (Storeys) y Espacios) con las propiedades más relevantes.
Postcondición	Se crea un fichero RDF con los datos extraídos del IFC.
Alternativas	Si hay un error al abrir el fichero IFC, el sistema informa del fallo y el caso de uso queda sin efecto dejando el servicio fuera de ejecución.

Tabla 5.13: Especificación del Caso de Uso: Generar Ontología.

Capítulo 6

Análisis del modelo ontológico

Una vez proporcionada la especificación de requisitos y los modelos de caso de uso (Capítulo 5), se va a detallar en qué consiste la fase de análisis mediante la elaboración de la ontología como modelo conceptual. Esta ontología se va a desarrollar siguiendo la metodología Methontology y contendrá los elementos fundamentales para el funcionamiento del servicio. Los nombres utilizados en los ejemplos que muestran las relaciones entre atributos son ficticios y no contienen valores reales, con el fin de no revelar información confidencial.

6.1. Especificación

En primer lugar hay que crear el documento de especificación de requisitos de la ontología. Este documento se muestra en la Tabla 6.1.

6.2. Conceptualización

Esta tarea tiene como objetivo organizar el conocimiento del dominio. Para ello, es necesario crear un modelo conceptual que defina un vocabulario con el significado de los conceptos, evitando así posibles interpretaciones por parte de los desarrolladores del proyecto.

Para llevar a cabo esta tarea, se toma como punto de partida la elaboración de un Glosario de Términos exhaustivo mediante la especificación de un Diccionario de Datos, un Árbol de Clasificación de Conceptos, una Tabla de Atributos de Clases y una Tabla de Atributos de Instancias. En este caso, estas tablas se han creado como Tablas de Propiedades, Tabla de Instancias y Tabla

Dominio	Modela la información de los sistemas de calefacción presentes en el edificio de CARTIF3.
Fecha	10 de abril de 2025.
Conceptualización	Daniel Jiménez García.
Implementación	Daniel Jiménez García.
Objetivo	Representar la información relacionada con los sistemas de gestión del calor en los edificios y crear un servicio sencillo que use el modelo ontológico para activar o cerrar las válvulas de los sistemas de calefacción.
Alcance	Limitado a los conceptos definidos por IFC y condicionado por las CQ desarrolladas.
Fuentes	Catálogo Almena, documentación de IFC y de los proyectos DEDALUS, DigiBUILD, SAREF, BRICK, y Sosa/SSN.

Tabla 6.1: Documento de especificación de requisitos de la ontología.

de Fórmulas. A continuación se muestra cómo se ha llevado a cabo la especificación de cada uno de estos elementos.

6.2.1. Diccionario de Datos

Esta sección muestra la definición de las entidades que conforman la ontología. La parte relativa a las propiedades incluye solo las más relevantes de las que se cargan al generar la ontología desde el fichero IFC del edificio de CARTIF3. La definición de cada concepto se puede consultar en las Tablas 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8 y 6.9.

6.2.2. Árbol de Clasificación de Conceptos

Esta tarea consiste en la construcción taxonómica de conceptos. Para ello, se organizan los conceptos definidos e identificados en el Diccionario de Datos para una mejor comprensión del dominio. La Figura 6.1 muestra el modelo creado con la herramienta Draw.io

Nombre	Building
Sinónimos	Construction, Edifice.
Descripción	Construcción con niveles (plantas), espacios delimitados (muchas veces por paredes)
Instancias	CARTIF 3 - Office Floor 2.
Propiedades	BuildingName, Category, OrganizacitonDescription
Ejemplo de propiedad	OrganizationDescription = Energy Division

Tabla 6.2: Definición de Edificio.

Nombre	BuildingSpace
Sinónimos	Espacio general.
Descripción	Zonas del edificio bien delimitadas.
Instancias	GroundFloor, SecondFloor.
Propiedades	BuildingStorey, Category, FamilyandType, Name, Structural
Ejemplo de propiedad	constainsDevice = device-actuator

Tabla 6.3: Definición de Zona.

Nombre	Space
Sinónimos	Area, Site, Zone.
Descripción	Zona delimitada dentro de cada planta.
Instancias	LABORATORIO_PROTOTIPOS
Propiedades	Category, Id, Name, Number
Ejemplo de propiedad	Name="LABORATORIO_PROTOTIPOS", Number="6"

Tabla 6.4: Definición de Espacio.

Nombre	Floor
Sinónimos	Storey
Descripción	Superficie horizontal que divide el edifico en alturas y donde se encuentras los espacios.
Instancias	GroundFloor, SecondFloor.
Propiedades	BuildingStorey, Category, FamilyandType, Name, Structural.
Ejemplo de propiedad	BuildingStorey=true

Tabla 6.5: Definición de Planta.

Nombre	Device
Sinónimos	Distribution Control Element, Asset.
Descripción	Dispositivo presente en el edificio y que cumple una acción específica.
Instancias	TemperatureSensor
Propiedades	Category, Description, Family, FamilyandType, FamilyName, Manufacturer, Mark, Model, OmniClassTitle.
Ejemplo de propiedad	OmniClassTitle="Monitoring and Control of Internal Climate"

Tabla 6.6: Definición de Device.

Nombre	Actuator
Sinónimos	Accionador
Descripción	Dispositivo que controla algún mecanismo físico del edificio.
Instancias	Accionador de tres puertas.
Propiedades	AssemblyCode, AssemblyDescription, ArticleDescription, Series, TypeName.
Ejemplo de propiedad	AssemblyCode="D30"

Tabla 6.7: Definición de Actuador.

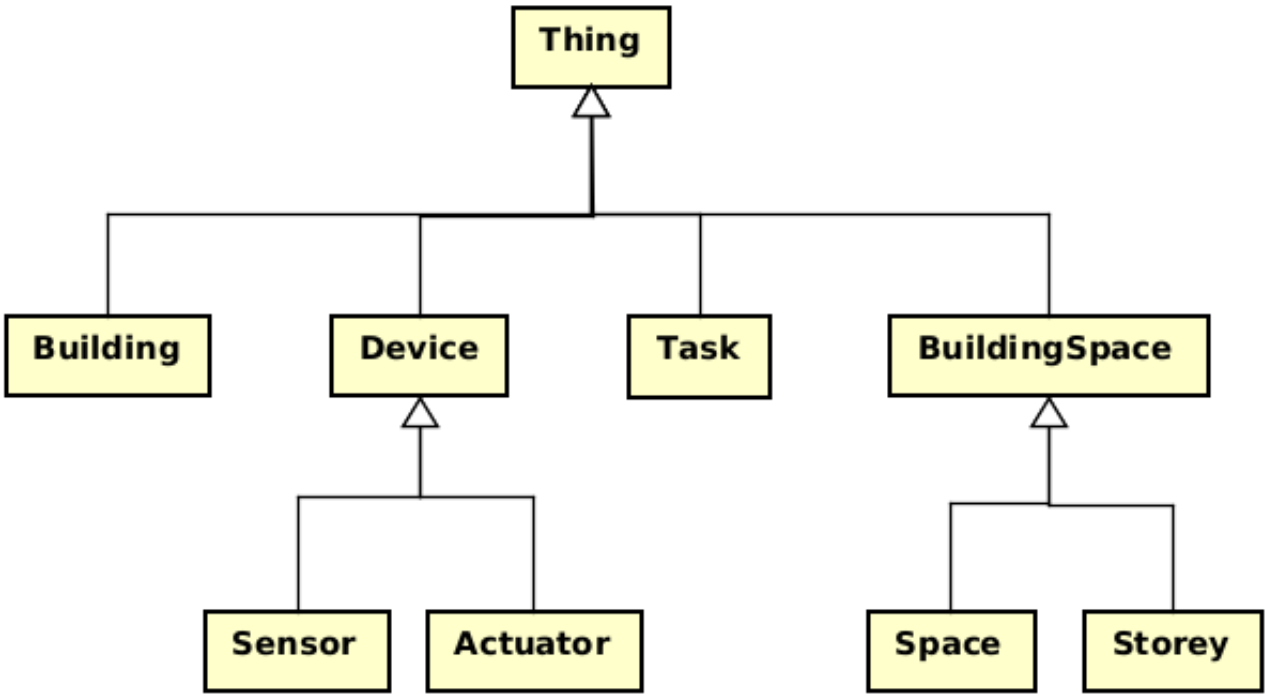


Figura 6.1: Árbol de Clasificación de Conceptos.

Nombre	Sensor
Sinónimos	Detector, Sonda.
Descripción	Dispositivo que detecta o mide una cualidad física.
Instancias	SENSOR:SchneiderTEMPERATURE
Propiedades	Reference, ProductName.
Ejemplo de propiedad	Reference="Schneider STR100 TEMPERATURE"

Tabla 6.8: Definición de Sensor.

Nombre	Task
Sinónimos	Tarea, Job.
Descripción	Trabajo específico a realizar en un tiempo determinado.
Instancias	OpenValve
Propiedades	Reference, ProductName.
Ejemplo de propiedad	Reference="Schneider STR100 TEMPERATURE"

Tabla 6.9: Definición de Tarea.

6.2.3. Tablas de Propiedades

La metodología escogida describe los atributos de instancias como los valores concretos que tienen los conceptos, mientras que los atributos de clases conforman las características del concepto relacionado. Por ejemplo, en un dominio sobre Personas, un atributo de clase sería que todos poseen cuatro extremidades, y un atributo de instancia el peso de cada individuo.

La integración de atributos de clases y de instancias en una sola tabla de propiedades está fundamentada, principalmente, en la propia naturaleza del dominio en cuestión. En este caso, la diferencia entre ambos tipos de atributos es poco relevante y la creación independiente de cada tabla generaría muchos valores de tipo "unknown" o nulos. Además, con la construcción de una plantilla única para todas las propiedades, se simplifica la construcción y consulta de la información. Esta plantilla consta de los siguientes campos: nombre de la propiedad, descripción, tipo de valor al que hace referencia, entidad relacionada con el concepto y cardinalidad. Tal como se ha indicado al definir las entidades que conforman la ontología, sólo se muestran las propiedades más relevantes de las importadas desde el fichero IFC. Por último, destacar que la propiedad Mark importada se transforma en la propiedad `hasDBreference`. En la Tablas 6.10, 6.11, 6.12, 6.13, 6.15,

6.16, 6.17, 6.18, 6.19, 6.20, 6.21, 6.22 y 6.23 se pueden ver algunas de las propiedades presentes en la ontología.

Property Name	Name
Description	Identificación que se le da a las plantas y espacios contenidos en el edificio.
Value Type	string
Related Entity Name	BuildingStorey y Space.
Cardinality	1
Exmaple	DESPACHO 1.

Tabla 6.10: Propiedad Name.

Property Name	Reference
Description	Nombre del sensor y propiedad que mide.
Value Type	string
Related Entity Name	Sensor
Cardinality	1
Example	Schneider STR100 TEMPERATURE

Tabla 6.11: Propiedad Reference.

6.2.4. Tabla de Instancias

La Tabla 6.25 muestra las instancias que aparecen en la ontología perteneciente al edificio de CARTIF3. Por motivos de confidencialidad, tanto el nombre de las variables de la empresa como de los sensores y actuadores son ficticios. Asimismo, se han omitido las columnas relativas a los atributos de las instancias y sus valores concretos, con el fin de no llenar la hoja con términos que se pueden ver directamente en el fichero RDF de la ontología, como el que se proporciona a modo de ejemplo en el Anexo C.1

6.2.5. Tabla de fórmulas

En este trabajo se han empleado fundamentalmente tres fórmulas. La primera de ellas calcula el Heat Index (HI) del espacio por medio de la Ecuación 6.1. Este indicador proporciona un índice

Property Name	Category
Description	Nombre que categoriza al individuo.
Value Type	string
Related Entity Name	Individuos de Spaces, Doors, Floors, Sensors y Actuators.
Cardinality	1
Example	Communication Devices.

Tabla 6.12: Propiedad Category

Property Name	FamilyandType
Description	Especifica el nombre de la familia del dispositivo y su tipo, además de los detalles del fabricante, el modelo y las características.
Value Type	string
Related Entity Name	Device
Cardinality	1
Example	SensorSchneiderSTR100Temperature

Tabla 6.13: Propiedad FamilyandType.

Property Name	OmniClassTitle
Description	Se almacena el nombre descriptivo del ítem de clasificación según el estándar norteamericano OmniClass.
Value Type	string
Related Entity Name	Dispositivo
Cardinality	1
Example	sensor OmniClassTitle "Monitoring of Internal Climate"

Tabla 6.14: Propiedad OmniClassTitle

Property Name	Model
Description	Información detallada que especifica el tipo de dispositivo.
Value Type	string
Related Entity Name	Device
Cardinality	1
Example	STR100

Tabla 6.15: Propiedad Model.

Property Name	BuildingName
Description	Nombre del edificio.
Value Type	string
Related Entity Name	Building
Cardinality	1
Example	CARTIF III - Office Floor 2

Tabla 6.16: Propiedad BuildingName.

Porperty Name	OrganizationName
Description	Nombre de la organización a la que pertenece el edificio.
Value Type	string
Related Entity Name	Building
Cardinality	1
Example	FUNDACION CARTIF

Tabla 6.17: Propiedad OrganizationName.

Porperty Name	OrganizationDescription
Description	Descripción de la organización a la que pertenece el edificio.
Value Type	string
Related Entity Name	Building
Cardinality	1
Example	Energy Division

Tabla 6.18: Propiedad OrganizationDescription

Porperty Name	ArticleDescription
Description	Descripción de la función que realiza el actuador.
Value Type	string
Related Entity Name	Actuator
Cardinality	1
Example	Three-way motorised zone valve

Tabla 6.19: Propiedad ArticleDescription.

Porperty Name	hasDBreference
Description	Identidicador del device en la Base de Datos relacional.
Value Type	string
Related Entity Name	Device
Cardinality	1
Example	REF9_SENSOR:SchneiderSTR100TEMPERATURE:03

Tabla 6.20: Propiedad hasDBreference.

Porperty Name	accomplishesTask
Description	Relación entre los individuos del device y las tareas que realiza.
Value Type	Task
Related Entity Name	Device
Cardinality	1
Example	Measure_Temperature

Tabla 6.21: Propiedad accomplishesTask.

Porperty Name	containsBuildingSpace
Description	Relación entre los edificios y los espacios que contienen.
Value Type	BuildingSpace
Related Entity Name	Building
Cardinality	1
Example	cartif3_building containsBuildingSpace space10

Tabla 6.22: Propiedad containsBuildingSpace.

Porperty Name	containsDevice
Description	Relación entre los espacios y los dispositivos que contienen.
Value Type	Device
Related Entity Name	BuildingSpace
Cardinality	1
Example	space1 containsDevice temperature_sensor

Tabla 6.23: Propiedad containsDevice.

de la temperatura aparente que experimenta el cuerpo humano en función de la temperatura (Temperature, T) y la humedad relativa (Relative Humidity, RH) obtenida a través de los sensores. La precondición que se debe cumplir es que la temperatura venga dada en grados Fahrenheit. Si no hay humedad relativa, el servicio desarrollado asume que el Heat Index del espacio es la temperatura medida.

$$\begin{aligned} \mathrm{HI} = {} & -42{,}379 + 2{,}04901523\,T + 10{,}14333127\,\mathrm{RH} \\ & - 0{,}22475541\,T\,\mathrm{RH} - 6{,}83783 \times 10^{-3}\,T^2 - 5{,}481717 \times 10^{-2}\,\mathrm{RH}^2 \\ & + 1{,}22874 \times 10^{-3}\,T^2\,\mathrm{RH} + 8{,}5282 \times 10^{-4}\,T\,\mathrm{RH}^2 - 1{,}99 \times 10^{-6}\,T^2\,\mathrm{RH}^2 \end{aligned} \tag{6.1}$$

La segunda fórmula (Ecuación 6.2) determina el porcentaje de apertura/cierre de los actuadores de tres vías.

$$\mathrm{ValveOpening}(HI) = \begin{cases} 0\,\%, & HI \geq T_{\text{máx}} + \Delta \\ 25\,\%, & T_{\text{máx}} < HI < T_{\text{máx}} + \Delta \\ 50\,\%, & T_{\text{mín}} < HI \leq T_{\text{máx}} \\ 75\,\%, & T_{\text{mín}} - \Delta < HI \leq T_{\text{mín}} \\ 100\,\%, & HI \leq T_{\text{mín}} - \Delta \end{cases} \tag{6.2}$$

La tercera fórmula (Ecuación 6.3), extraída de [34], permite generar un valor para la temperatura dada en un momento concreto del día aplicando una pequeña variabilidad al resultado. Para

Nombre	**Descripción**
10	Espacio SALA_OFICINAS_1 de 2ª planta de CARTIF3.
9	Espacio SALA_OFICINAS_2 de 2ª planta de CARTIF3.
8	Espacio DESPACHO_1 de 2ª planta de CARTIF3.
7	Espacio DESPACHO_2 de 2ª planta de CARTIF3.
6	Espacio LABORATORIO_PROTOTIPOS de 2ª planta de CARTIF3.
5	Espacio LABORATORIO_EQUIPOS de 2ª planta de CARTIF3
4	Espacio LABORATORIO_SONDAS de 2ª planta de CARTIF3.
SchneiderSensor4_Temp	Sensor de temperatura del espacio 4.
SchenideSensor5_Temp	Sensor de temperatura del espacio 5.
SchneiderSensor7_Temp	Sensor de temperatura del espacio 7.
SchneiderSensor8_Temp	Sensor de temperatura del espacio 8.
SchneiderSensor9_Temp	Sensor de temperatura del espacio 9.
SchneiderSensor9_Hum	Sensor de húmedad relativa del espacio 9.
SchneiderSensor9_CO_2	Sensor de concentración de CO_2 del espacio 9.
SchneiderSensor10_Temp	Sensor de temperatura del espacio 10.
ActuatorThreeWay4_Open	Actuador de apertura de válvula del espacio 4.
ActuatorThreeWay4_Close	Actuador de cierre de válvula del espacio 4.
ActuatorThreeWay6_Open	Actuador de apertura de válvula del espacio 6.
ActuatorThreeWay6_Close	Actuador de cierre de válvula del espacio 6.
ActuatorThreeWay9_Open	Actuador de apertura de válvula del espacio 9.
ActuatorThreeWay9_Close	Actuador de cierre de vávlula del espacio 9.
ActuatorThreeWay10_Open	Actuador de apertura de válvula del espacio 10.
ActuatorThreeWay10_Close	Actuador de cierre de válvula del espacio 10.
Measure_CO_2	Tarea de medir la concentración de CO_2.
Measure_Humidity	Tarea de medir la humedad relativa.
Measure_Temperature	Tarea de medir la temperatura ambiente.
Open_Valve	Tarea de los actuadores de abrir las válvulas.
Close_Valve	Tarea de los actuadores de cerrar las válvulas.

Tabla 6.25: Tabla de instancias.

ello, se calcula inicialmente el minuto actual del día:

$$m \;=\; \text{today.hour} \times 60 \;+\; \text{today.minute}$$

A continuación, se obtienen los valores de t_1 y t_2 en función del minuto en que amanece (s_{rise}) y anochece (s_{set}):

$$\text{Si } m < s_{\text{rise}}: \quad t_1 = t_{\text{máx}}^{\text{prev}}, \quad t_2 = t_{\text{mín}}$$

$$\text{Si } s_{\text{rise}} \le m < s_{\text{set}}: \quad t_1 = t_{\text{mín}}, \quad t_2 = t_{\text{máx}}$$

$$\text{Si } m \ge s_{\text{set}}: \quad t_1 = t_{\text{máx}}, \quad t_2 = t_{\text{mín}}^{\text{next}}$$

Tras ello, se define la temperatura base (T_{base}) en función de la hora mediante tres tramos:

$$T_{\text{base}}(m) \;=\; \begin{cases} t_2 \;+\; (t_1 - t_2)\,\cos\!\left(\pi\,\dfrac{m + 1440 - s_{\text{set}}}{1440 - (s_{\text{set}} - s_{\text{rise}})}\right), & m \;<\; s_{\text{rise}}, \\[2ex] t_1 \;+\; (t_2 - t_1)\,\sin\!\left(\pi\,\dfrac{m - s_{\text{rise}}}{s_{\text{set}} - s_{\text{rise}}}\right), & s_{\text{rise}} \;\le\; m \;<\; s_{\text{set}}, \\[2ex] t_1 \;+\; (t_2 - t_1)\,\cos\!\left(\pi\,\dfrac{m - s_{\text{set}}}{1440 - s_{\text{set}}}\right), & m \;\ge\; s_{\text{set}} \end{cases} \tag{6.3}$$

Finalmente, se añade ruido gaussiano con desviación estándar proporcional a la amplitud térmica del día:

$$\sigma \;=\; \text{máx}\big(0{,}09,\; 0{,}075\,(t_{\text{máx}} - t_{\text{mín}})\big), \quad \eta \;\sim\; \mathcal{N}(0,\, \sigma)$$

De este modo, la temperatura final es:

$$T = T_{\text{base}}(m) \;+\; \eta$$

donde la función retorna T como un número en coma flotante.

6.3. Integración de otras ontologías

En este apartado se analizan diferentes ontologías de dominio en la fase de evaluación y selección de componentes ontológicos. Siguiendo el enfoque propuesto en Methontology, se ha creado una matriz comparativa que analiza siete ontologías relevantes en el modelado energético en edificios mediante nueve criterios propuestos de forma personal. Los cuatro primeros están relacionados con el estado de la ontología y su especificación, y los otros cinco con los conceptos que define. La Tabla 6.27 muestra la comparativa realizada en función de los valores definidos en la escala aplicada para la evaluación de cada uno de estos criterios, cuyo significado se indica a continuación:

- **Verbosidad:** suma del número de conceptos y propiedades presentes en la ontología. Escala: {POCA, MODERADA, MUCHA}, siendo POCA menos de 50; MODERADA entre 50 y 250; MUCHA más de 250.
- **Autodocumentación:** uso de las propiedades de auto-documentación que ofrecen RDFS y OWL. Escala de valores: {SI, NO}
- **Soporte:** mantenimiento actual por la organización que creó la ontología. La principal forma de averiguarlo es consultando las últimas publicaciones en la página web de la organización, o los últimos commits en el repositorio donde se encuentra alojada la ontología. Escala de valores: {SI, NO}
- **Consistencia:** el razonador funciona sin errores de satisfacibilidad. Escala: {SI, NO}
- **Localización:** presencia de conceptos para ubicar las zonas de edificios. Escala: {SI, NO}
- **Energía:** presencia de conceptos para la energía y los procesos implicados. Escala: {SI, NO}
- **Dispositivos:** presencia del vocabulario adecuado para definir dispositivos físicos como calderas, sistemas empotrados, etc. Escala de valores: {SI, NO}
- **Sensorización:** presencia de definiciones de medidas de sensores y de observaciones que se hacen de los mismos. Escala de valores: {SI, NO}
- **Distritos:** presencia de conceptos adecuados para representar la relación energética entre edificios. Escala de valores: {SI, NO}

El objetivo de la comparativa mostrada en la Tabla 6.27 es facilitar la identificación de la ontología más adecuada para el proyecto, seleccionando aquella que permita un buen grado de

	BoT	DigiBUILD	SAREF	BRICK	IfcOWL	dCO	SOSA/SSN
Verbosidad	POCA	MODERADA	MODERADA	MUCHA	MUCHA	MUCHA	POCA
Autodocumentación	SI	NO	SI	NO	NO	NO	SI
Soporte	NO	SI	SI	NO	SI	NO	SI
Consistencia	SI	SI	SI	SI	SI	SI	SI
Localización	SI	SI	SI	SI	SI	SI	NO
Energía	NO	SI	SI	SI	NO	NO	NO
Dispositivos	NO	SI	SI	SI	SI	SI	SI
Sensorización	NO	SI	SI	SI	SI	SI	SI
Distritos	NO	SI	NO	NO	NO	NO	NO

Tabla 6.27: Estudio comparativo de ontologías de dominio.

reutilización, esté mayormente auto-documentada, disponga de soporte, sea consistente y contenga la mayor parte de los conceptos requeridos. Se puede observar, por tanto, que las ontologías más adecuadas son ifcOWL, SAREF y SOSA/SSN, pero debido a la falta de auto-documentación en IfcOWL, la ausencia de conceptos de energía en SOSA/SSN, el uso estándar de SAREF a nivel Europeo y los módulos construidos en esta (SAREF for Buiding o SAREF for Energy Flexibility), hacen que SAREF CORE y su extensión SAREF for Building (S4BLDG) se conviertan en la ontología base para este proyecto.

6.4. Implementación

Este apartado se describe con más detalle en el Capítulo 7. De momento, mencionar que se ha empleado el lenguaje de programación Python con la librería owlready2 para desarrollar los conceptos de la ontología.

6.5. Evaluación

6.5.1. Competency Questions

La fase de evaluación en Methontology tiene como principal objetivo verificar que la ontología cumple con las necesidades del dominio. Estas necesidades se expresan en forma de Competency Questions (CQ). Para cada una, se define un escenario de prueba, se formula la consulta SPARQL

correspondiente y se compara la respuesta obtenida con el resultado esperado. Si la respuesta coincide exactamente con el conjunto de respuestas previamente definidas, el caso de prueba se considera exitoso.

Las cuatro consultas realizadas en la prueba de evaluación se han creado en un script de Python (*ontology_ evalutaion.py*). Para no sobrecargar las preguntas escribiendo todos los prefijos, se han utilizado los siguientes en todas las CQ:

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:  <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
PREFIX onto: <http://www.semanticweb.org/danieljimgar/ontologies/2025/ontoenergy>
```

CQ 1: ¿Qué dispositivos hay en un espacio concreto?

Con esta pregunta se espera obtener un individuo de tipo `Space` con un número de identificación igual a `"4"`, al que están vinculados tres dispositivos mediante la propiedad `onto:containsDevice`.

SPARQL Query:

```
SELECT ?device WHERE {
    ?space a onto:Space ;
    onto:Number "4" ;
    onto:containsDevice ?device.
}
```

Resultado Esperado:
SchneiderSensor4_Temp, ActuatorThreeWay4_Open y ActuatorThreeWay4_Close

CQ 2: ¿Qué mide un sensor específico y cuál es su modelo?

Con esta pregunta se busca un sensor concreto, identificado por su URI, con el fin de recuperar la propiedad `onto:FamilyandType` que describe su función de medida (temperatura, humedad,...)

SPARQL Query:

```
SELECT ?property WHERE {
    ?sensor rdf:type saref:Sensor ;
    onto:FamilyandType ?property .
    VALUES ?sensor {<uri_del_sensor>} .
}
```

Resultado Esperado:
Un sensor específico para la <URI>proporcionada.

CQ 3: ¿Cuántos espacios hay en un edificio?

Con esta pregunta se pretende contar el número de instancias `:Space` contenidas en el edificio de prueba `:Building_B`.

SPARQL Query:,

```
SELECT (COUNT(?space) AS ?numSpaces) WHERE {
    ?building onto:containsBuildingSpace ?space .
}
```

Resultado Esperado:
Se espera obtener los siete espacios relacionados con el edificio de la planta 2 de CARTIF3.

CQ 4: ¿Cuál es la referencia al nombre de cada dispositivo en la base de datos y qué función realiza?

Esta pregunta pretende recuperar para cada dispositivo (sensor o actuador) su ID en la base de datos y la tarea que realiza. Para ello, se usan las propiedades `onto:hasDBreference` y `onto:accomplishesTask`.

SPARQL Query:

```
SELECT ?device ?task ?reference WHERE {
    {
        ?device rdf:type saref:Sensor ;
        onto:hasDBreference ?reference ;
        onto:accomplishesTask ?task .
    }
```

```
    UNION
    {
        ?device rdf:type saref:Actuator ;
        onto:hasDBreference ?reference ;
        onto:accomplishesTask ?task .
    }
}
```

Resultado Esperado:
Se muestra en la Tabla 6.28.

Dispositivo	**Tarea**	**Referencia en la BD**
SchneiderSensor4_Temp	Measure_temperature	VARIABLE_SENSOR4TEMP_BD
SchneiderSensor5_Temp	Measure_temperature	VARIABLE_SENSOR5TEMP_BD
SchneiderSensor9_Temp	Measure_temperature	VARIABLE_SENSOR9TEMP_BD
SchneiderSensor9_CO2	Measure_temperature	VARIABLE_SENSOR9CO2_BD
SchneiderSensor9_Hum	Measure_temperature	VARIABLE_SENSOR9HUM_BD
SchneiderSensor10_Temp	Measure_temperature	VARIABLE_SENSOR10TEMP_BD
SchneiderSensor7_Temp	Measure_temperature	VARIABLE_SENSOR7TEMP_BD
SchneiderSensor8_Temp	Measure_temperature	VARIABLE_SENSOR8TEMP_BD
ActuatorThreeWay4_Close	Measure_temperature	VARIABLE_ACTUATOR4_CLOSE_BD
ActuatorThreeWay10_Close	Close_Valve	VARIABLE_ACTUATOR10_CLOSE_BD
ActuatorThreeWay6_Open	Open_Valve	VARIABLE_ACTUATOR6_OPEN_BD
ActuatorThreeWay9_Open	Open_Valve	VARIABLE_ACTUATOR9_OPEN_BD
ActuatorThreeWay6_Close	Close_Valve	VARIABLE_ACTUATOR6_CLOSE_BD
ActuatorThreeWay9_Close	Close_Valve	VARIABLE_ACTUATOR9_CLOSE_BD
ActuatorThreeWay4_Open	Open_Valve	VARIABLE_ACTUATOR4_OPEN_BD
ActuatorThreeWay10_Open	Open_Valve	VARIABLE_ACTUATOR10_OPEN_BD

Tabla 6.28: Respuesta esperada en la CQ 4.

6.5.2. Evaluación con OOPS!

Tras la implementación de la ontología, descrita con detalle en el Capítulo 7, se han ejecutado con el escáner OOPS! las ontologías generadas para los dos demos. Los resultados obtenidos para

la ontología de CARTIF3 y la demo de Haus se muestran en las Figuras 6.2 y 6.3, respectivamente.

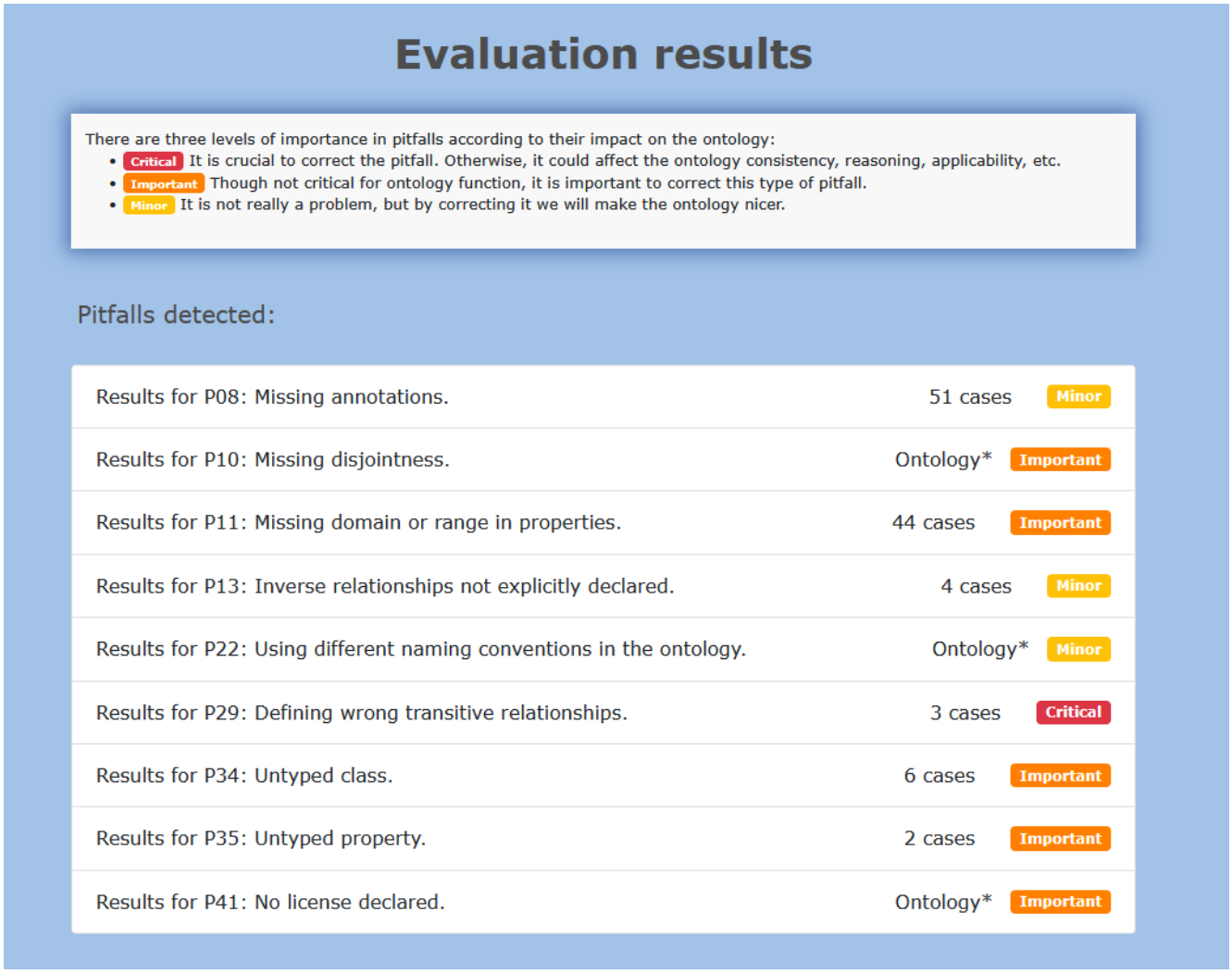


Figura 6.2: Pitfalls encontrados en la ontología de CARTIF3.

De los resultados obtenidos en ambas evaluaciones se pueden sacar las mismas conclusiones. Lo más preocupante es el error crítico por haber definido una propiedad transitiva, pero tras revisarlo indica que no es necesaria. Esto se debe a que no cuestiona el dominio del problema, por tanto, se considera que no es muy relevante. Los otros aspectos más graves son no haber declarado previamente las clases a las que se instancian los individuos, ya que las clases de las que se instancian se nombran solo cuando se crea el `NamedIndividual`, como se puede observar en el Apéndice C.1. Asimismo, comentar la falta de declaración de entidades y propiedades disjuntas,

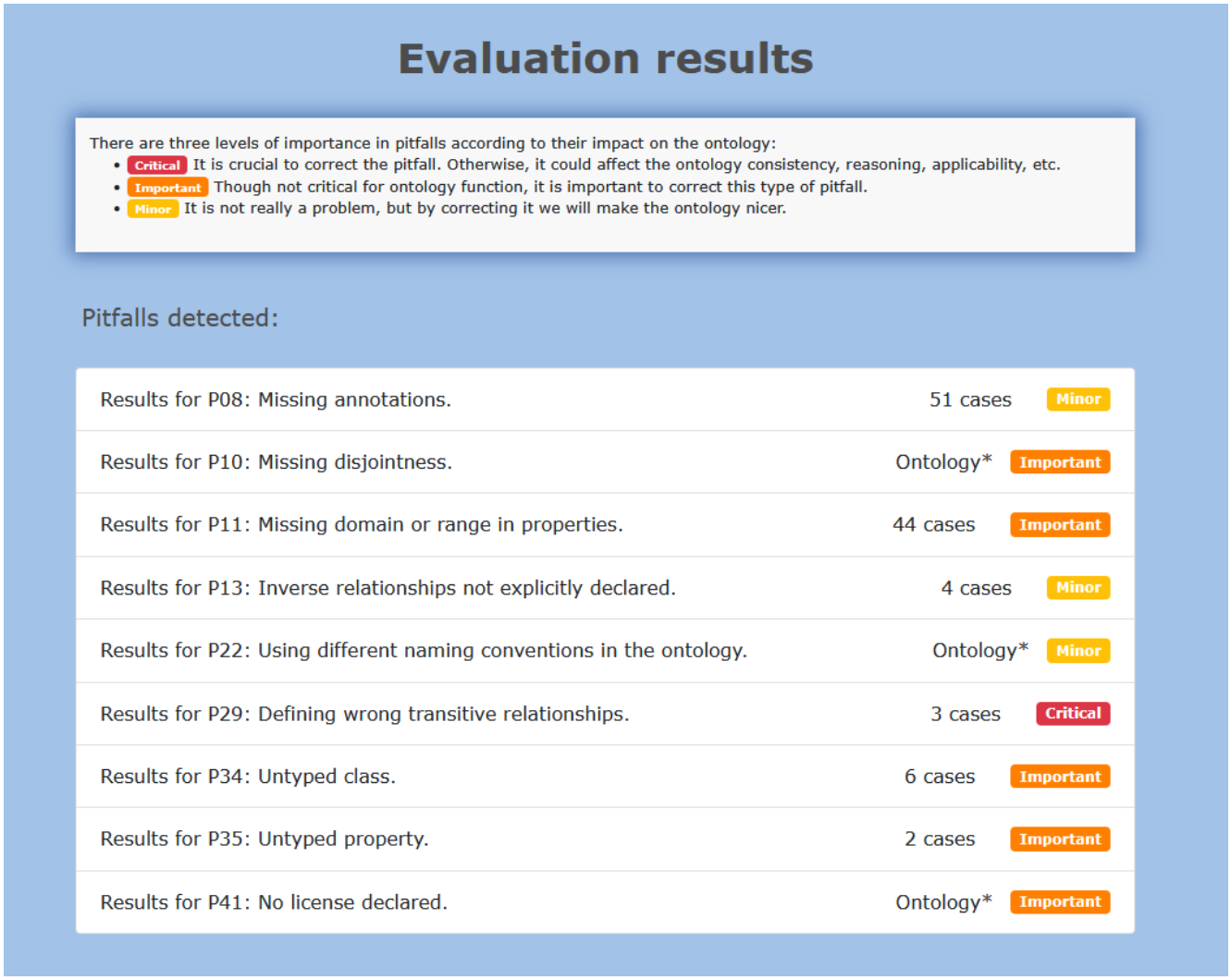


Figura 6.3: Pitfalls encontrados en la ontología de Haus.

o la falta de especificación en los dominios y en los rangos de las propiedades, pero esto último es debido a la forma en que se importan las entidades con owlready 7.19.

6.5.3. Ejecución del razonador

Para comprobar que realmente se puede trabajar con las ontologías generadas, se ha ejecutado el razonador Pellet desde Protégé. El principal objetivo es comprobar la satisfacibilidad de ambas ontologías. Hay que destacar que la ejecución se ha realizado sobre el RDF del demo de CARTIF3, pero también se ha ejecutado con el RDF del demo de Haus, siendo idéntico el resultado obtenido en ambos casos. La Figura 6.4 permite observar que no se ha producido ningún mensaje de error.

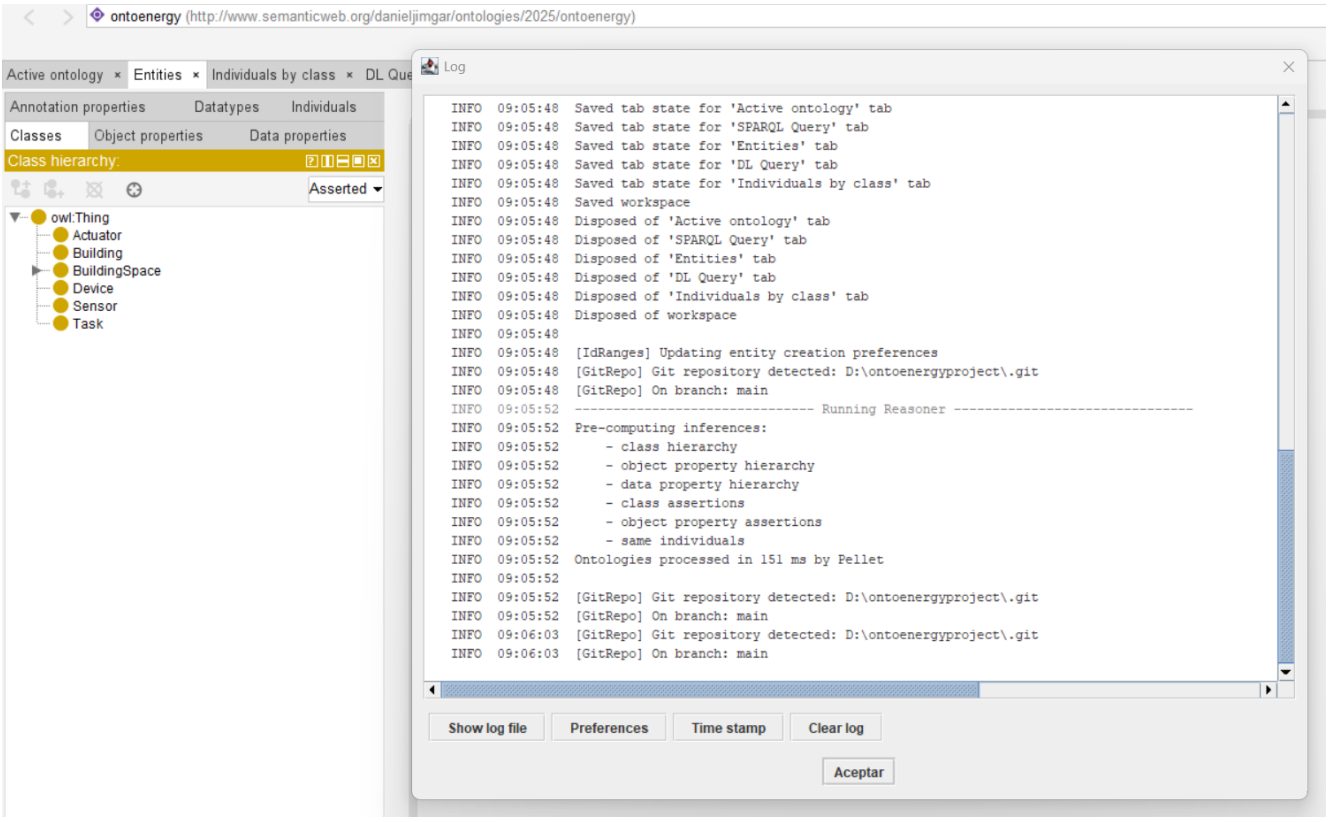


Figura 6.4: Ejecución del razonador Pellet.

6.6. Mantenimiento

Debido a limitaciones tanto en el alcance del proyecto como en el tiempo disponible, no se ha incluido la fase de mantenimiento.

Capítulo 7

Diseño e Implementación

En este capítulo se detallan las decisiones tomadas y la arquitectura propuesta con el fin de comprender cómo está construido el servicio desarrollado durante las 20 semanas de proyecto.

7.1. Decisiones de diseño

Desde el inicio del proyecto se estableció Python como lenguaje de programación y OWLready2 como herramienta para gestionar la ontología creada. Otra decisión que se adoptó fue la conversión del fichero IFC que se toma como punto de partida a formato RDF, siempre que no se encontrase ya creado. Además, por cuestiones de rendimiento, se ha utilizado una base de datos PostgreSQL para simular el comportamiento de una base de datos de series temporales para, de esta forma, no almacenar valores numéricos en la ontología.

7.2. Diseño lógico de la base de datos

En la Figura 7.1 se puede observar el Modelo Relacional de la Base de Datos PostgreSQL correspondiente al acceso a los valores relevantes para la ejecución del servicio.

7.2.1. Arquitectura lógica del sistema

Una vez tomadas las decisiones de diseño se ha optado por establecer una separación modular entre responsabilidades, sin llegar a usar ningún patrón específico. Esta separación modular se estructura en un paquete principal, `src`, que contiene el ejecutable `main.py` que hace uso del paquete

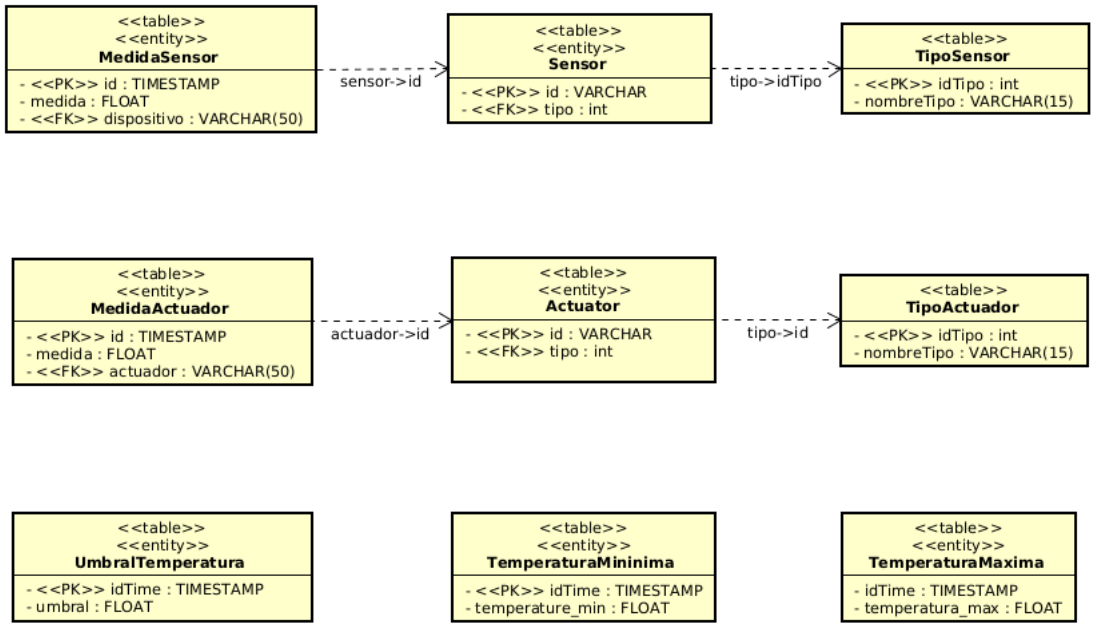


Figura 7.1: Modelo Relacional de la Base de Datos.

service, y este delega en los paquetes databaseprocessor e ingestor las consultas a fuentes externas de datos. El primero, datebaseprocessor, se ocupa de gestionar las peticiones a la base de datos PostgreSQL, y el segundo, ingestor, se ocupa de obtener los conceptos necesarios del fichero RDF de la ontología por medio de la clase `sarefingestor`. La Figura 7.2 muestra con un diagrama de paquetes la arquitectura general del servicio aplicando el estilo Uses Style.

7.3. Diseño detallado de los módulos

En esta sección se presentan los diseños detallados de los módulos principales que conforman el sistema. En la Figura 7.3 se muestra el diseño del módulo thermal_comfort_service, donde se describen las funciones que contiene con la clase definida. Del mismo modo, se muestra también el diseño de los paquetes ingestor (Figura 7.4) y databaseprocessor (Figura 7.5) con las funciones y clases que se han creado en ellos. El paquete ingestor detalla los métodos encargados de recoger las entidades de la ontología. El paquete databaseprocessor, por su parte, especifica la estructura encargada de la interacción con la base de datos PostgreSQL. La Figura 7.5 muestra, además, las clases y los métodos utilizados para la persistencia y recuperación de los datos.

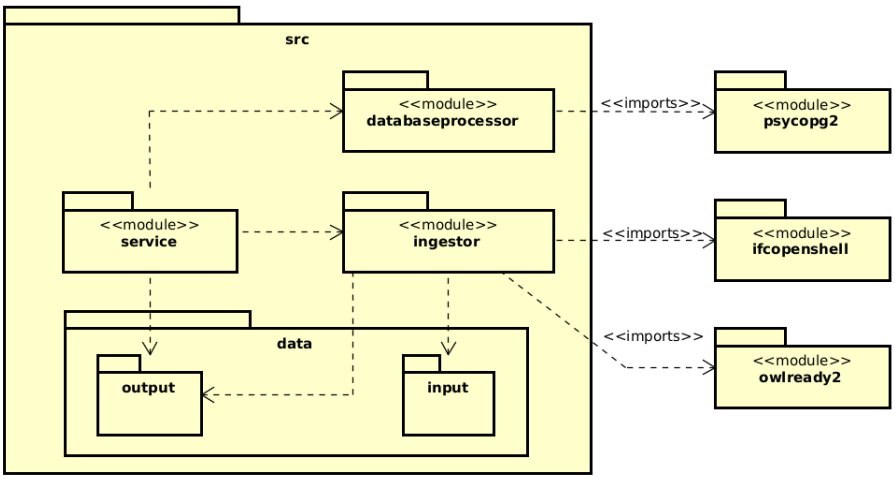


Figura 7.2: Arquitectura lógica del sistema.

7.4. Diagramas de despliegue

Este apartado muestra cómo ha sido desplegado el servicio para las dos pruebas realizadas. En la Figura 7.6 se observa el despliegue realizado en los equipos haciendo uso de PostgreSQL de forma local, mientras que en la Figura 7.7 se observa la disposición del servicio y de la base de datos haciendo uso de la máquina virtual alojada en Oracle.

7.5. Flujo de datos del servicio

La Figura 7.8 presenta un diagrama de actividad con el objetivo de ilustrar el proceso que sigue el servicio durante su ejecución. A partir de la lectura del fichero IFC, el sistema transforma la información a datos de la ontología en formato RDF y, de forma periódica, procesa el fichero RDF para obtener los espacios del edificio, sus sensores y actuadores, calcular el *heat index* de cada espacio y, finalmente, ejecutar la función de activación correspondiente. Este diagrama permite visualizar de manera clara y ordenada cada uno de los pasos por los que pasan los datos dentro del servicio, facilitando su comprensión y posterior mantenimiento.

Figura 7.3: Diseño detallado del paquete service.

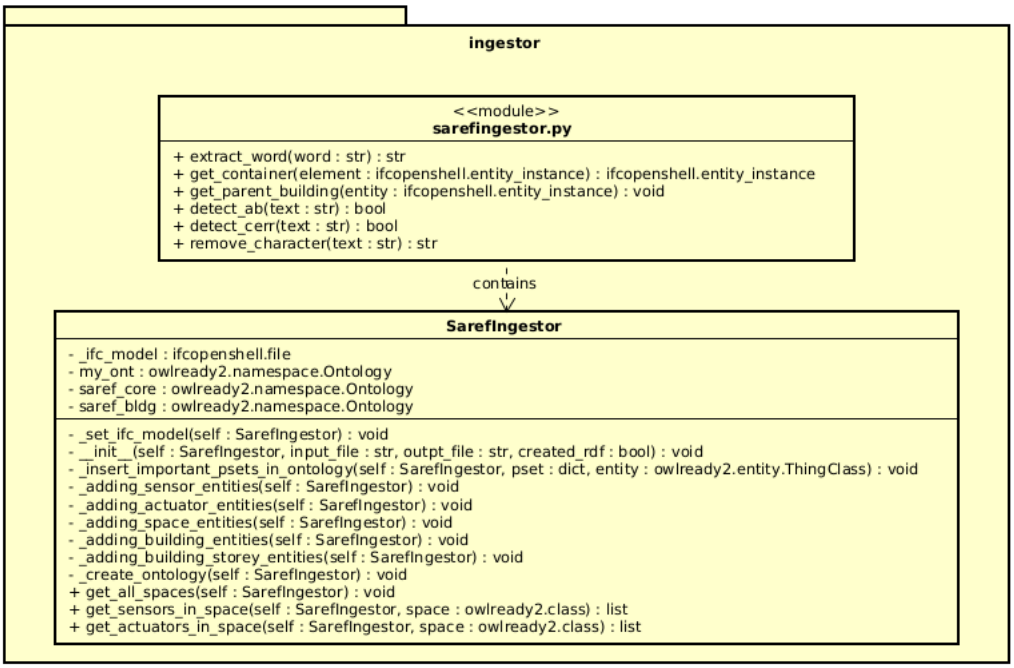


Figura 7.4: Diseño detallado del paquete ingestor.

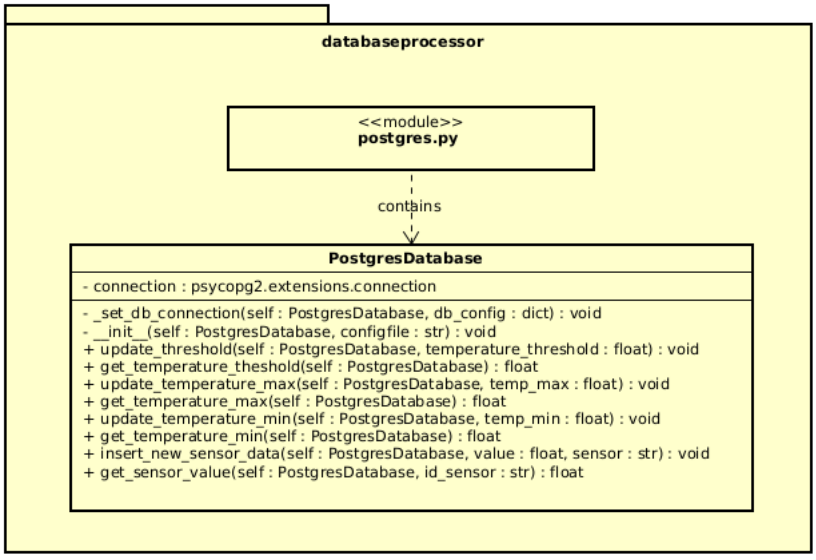


Figura 7.5: Diseño detallado del paquete databaseprocessor.

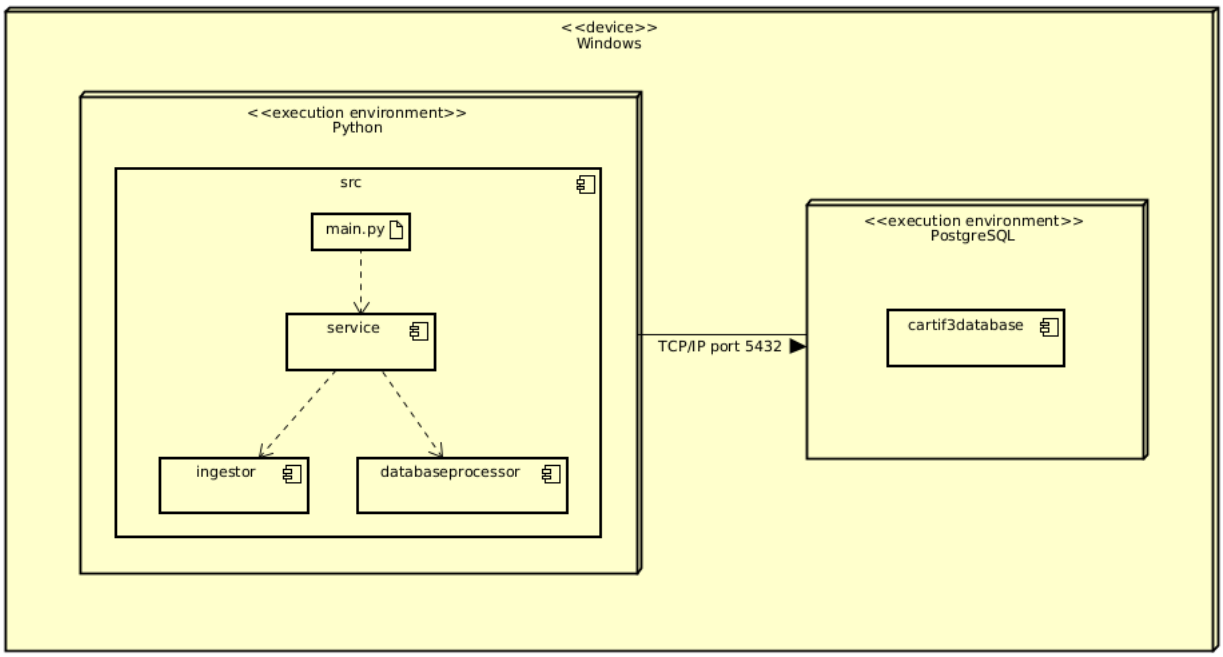


Figura 7.6: Diagrama de despliegue en fase de desarrollo.

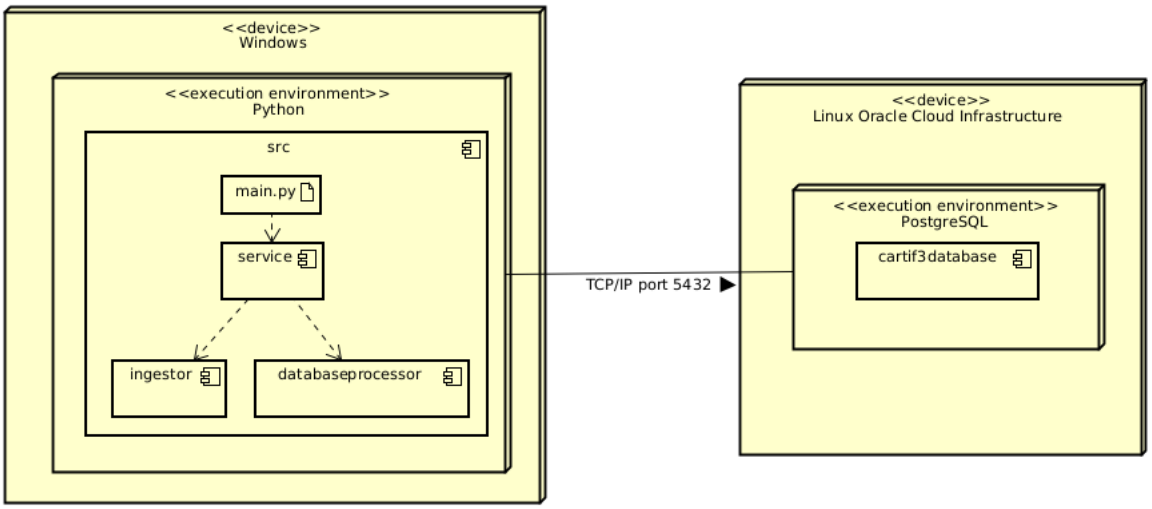


Figura 7.7: Diagrama de despliegue en 2 niveles con OCI.

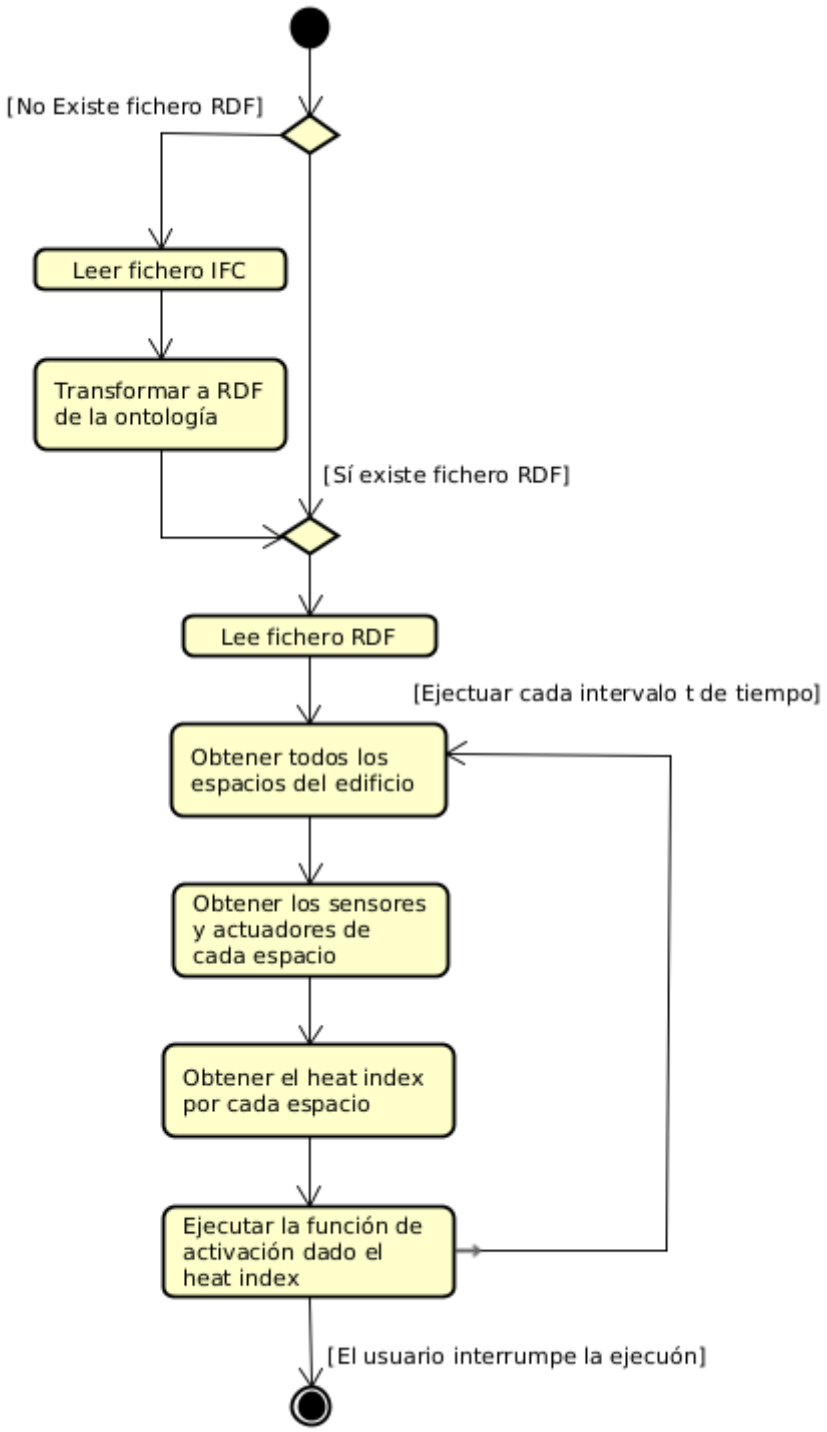


Figura 7.8: Diagrama de actividad sobre el flujo de datos del servicio.

7.6. Realización de los Casos de Uso

Tras haber indicado a grandes rasgos cómo es el proceso de ejecución del servicio, se va a ilustrar su funcionamiento con diagramas de secuencia con los mensajes y la elección de responsabilidades.

7.6.1. Realización del CU01: Iniciar Servicio

En este caso de uso se inicializan todos los elementos necesarios para que permanezcan de manera continua. Para ello, se debe tener en cuenta que el CU03 "Generar Ontología" se ejecuta al inicializar la clase ThermalComfortService (Figura 7.10) sólo si el fichero RDF de la ontología especificada no existe. El CU02 "Actualizar Umbral" (Figura 7.17), en cambio, se puede ejecutar en cualquier momento una vez ha sido inicializado el servicio.

7.6.2. Realización del CU02: Cambiar Umbral

Tras la puesta en marcha del servicio o al ejecutarlo en línea de comandos con el parámetro `--update_threshold`, se puede cambiar el umbral definido en el fichero pyproject.toml. En la Figura 7.17 se muestra la secuencia de mensajes necesarios para actualizar el umbral.

7.6.3. Realización del CU03: Generar Ontologia

Por último, en las Figuras 7.18 y 7.19 se muestra la secuencia de mensajes para crear el RDF de la ontolología. Cabe recordar que este fichero no debe existir previamente.

7.7. Modelo de dominio de las clases de la ontología

Tal como se ha indicado en la Sección 6.3, la ontología de partida es SAREF, y su extensión S4BLDG. Para poblar la ontología, se ha hecho un mapping entre los elementos descritos en formato IFC con la librería ifcopenshell y la ontología.

La Figura 7.20 muestra las entidades y relaciones de la ontología. Estas entidades son mapeadas desde el fichero IFC al RDF, que posteriormente owlready2 convierte en clases al ejecutar las consultas SPARQL desde sarefingestor.py. Las Figuras 7.21 y 7.22, por su parte, muestran el modelo de dominio de los conceptos propuestos por la ontología basada en SAREF y su extensión, S4BLDG.

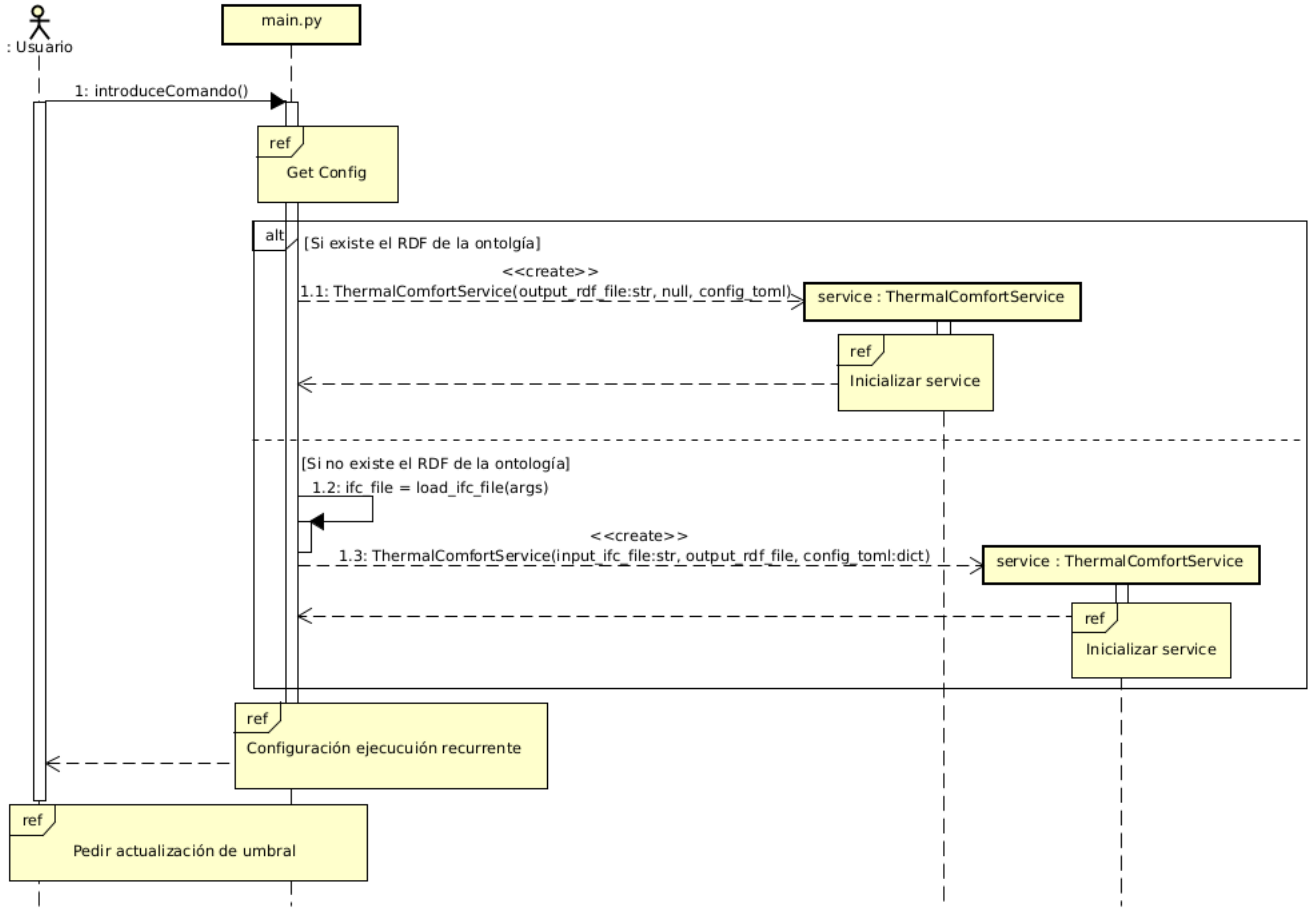


Figura 7.9: Estructura de ejecución del servicio.

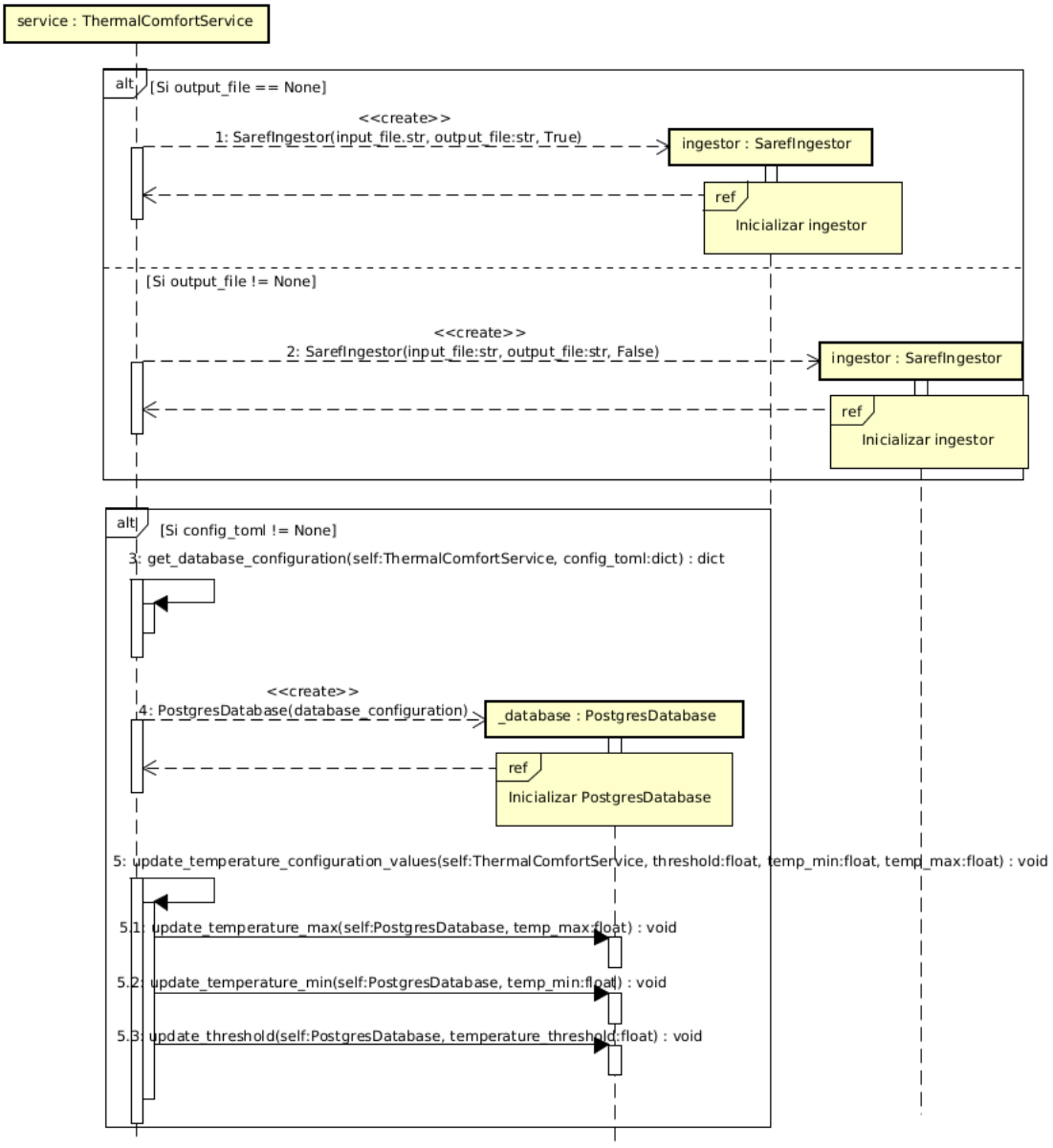


Figura 7.10: Secuencia de inicialización de la clase ThermalComfortService.

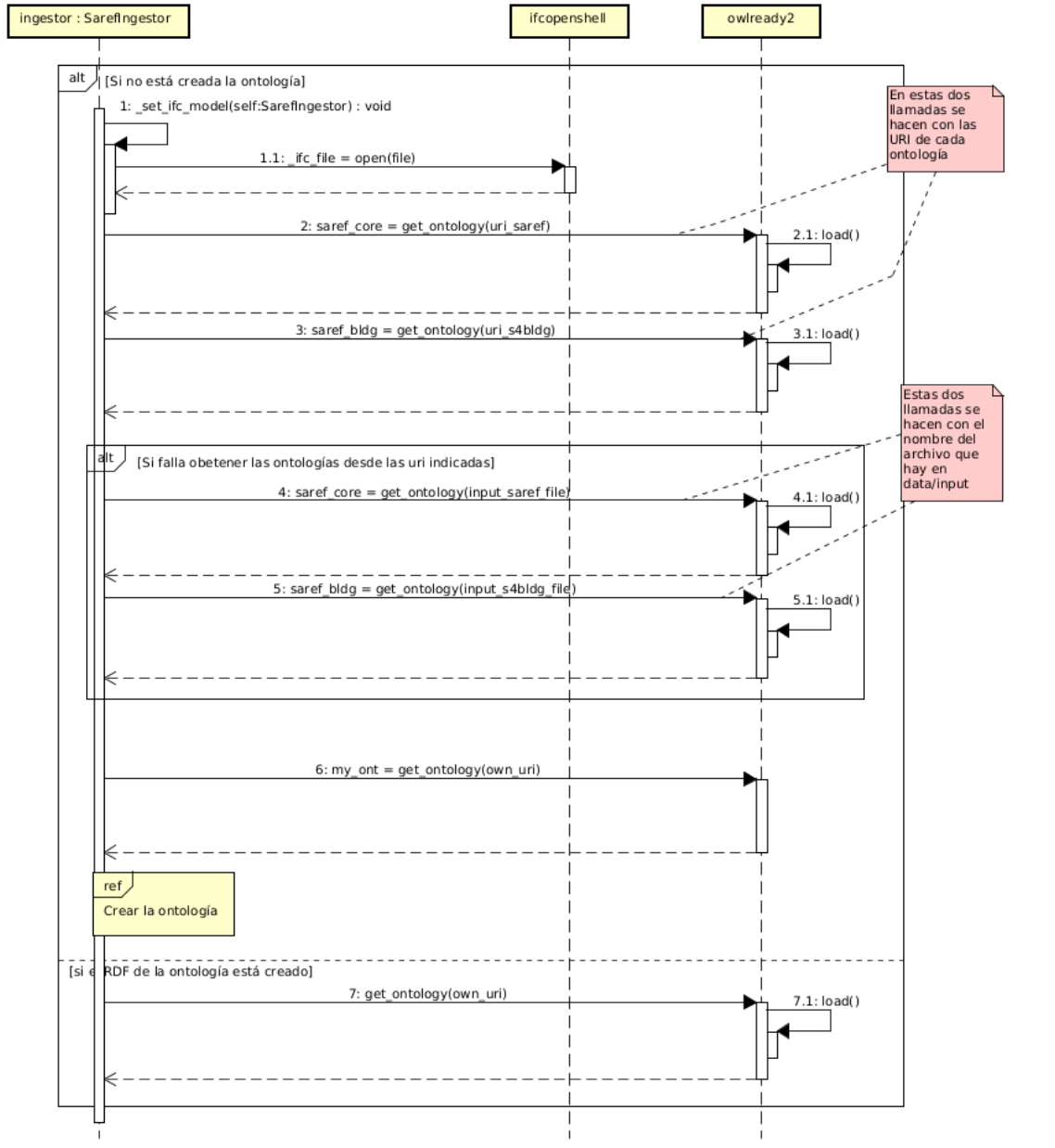


Figura 7.11: Secuencia de inicialización de la clase SarefIngestor.

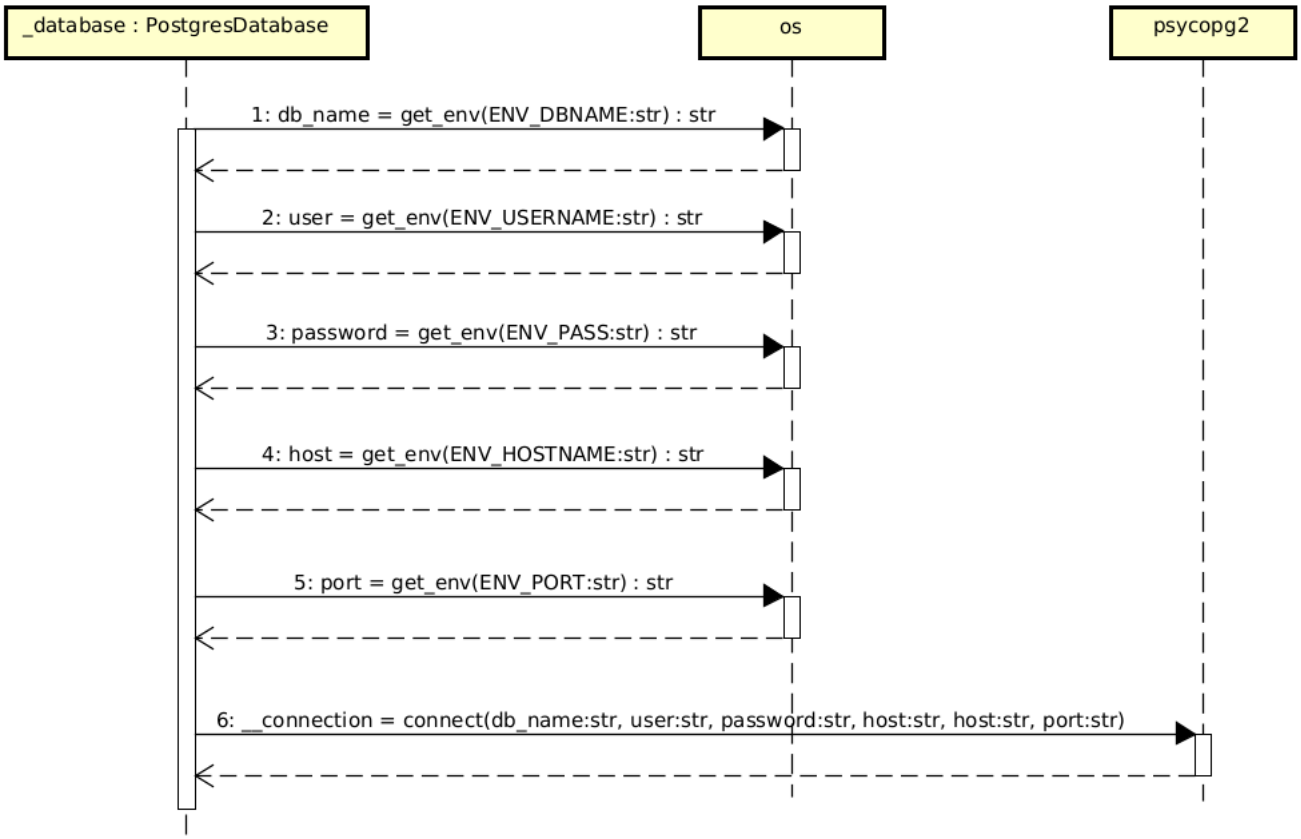


Figura 7.12: Secuencia de la inicialización de la clase PostgresDatabase.

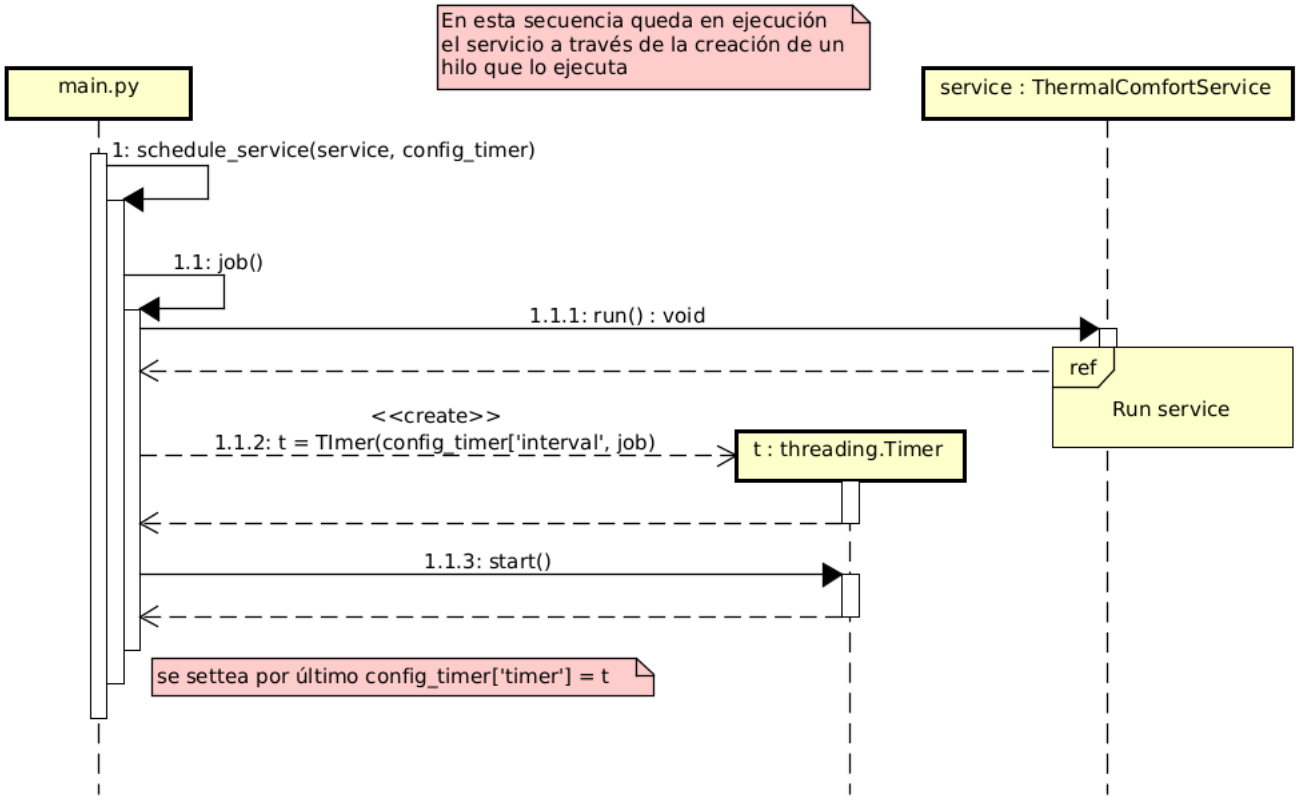


Figura 7.13: Secuencia de configuración de la ejecución recurrente del servicio.

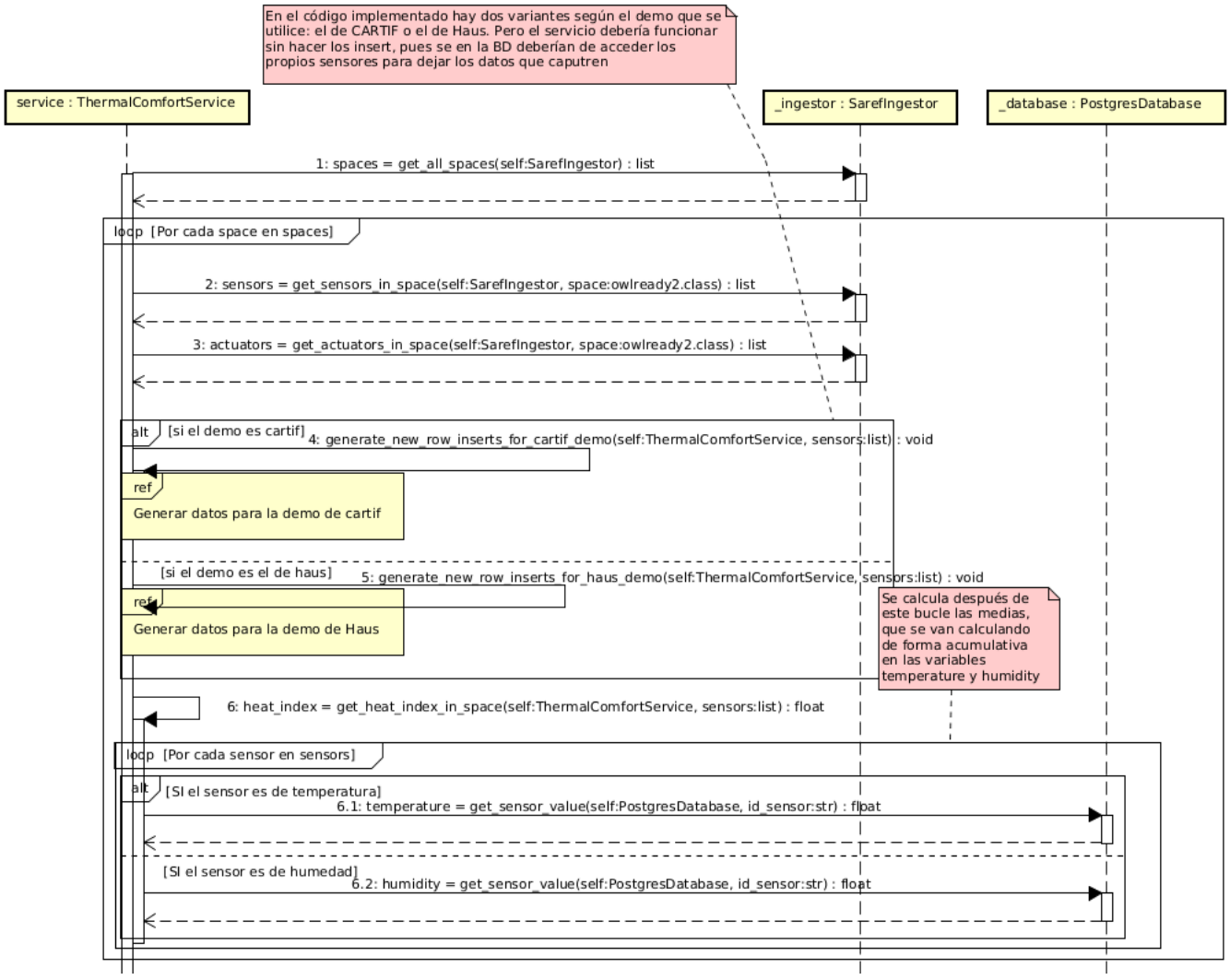


Figura 7.14: Parte 1 de la secuencia del método `run` del ThermalComfortService.

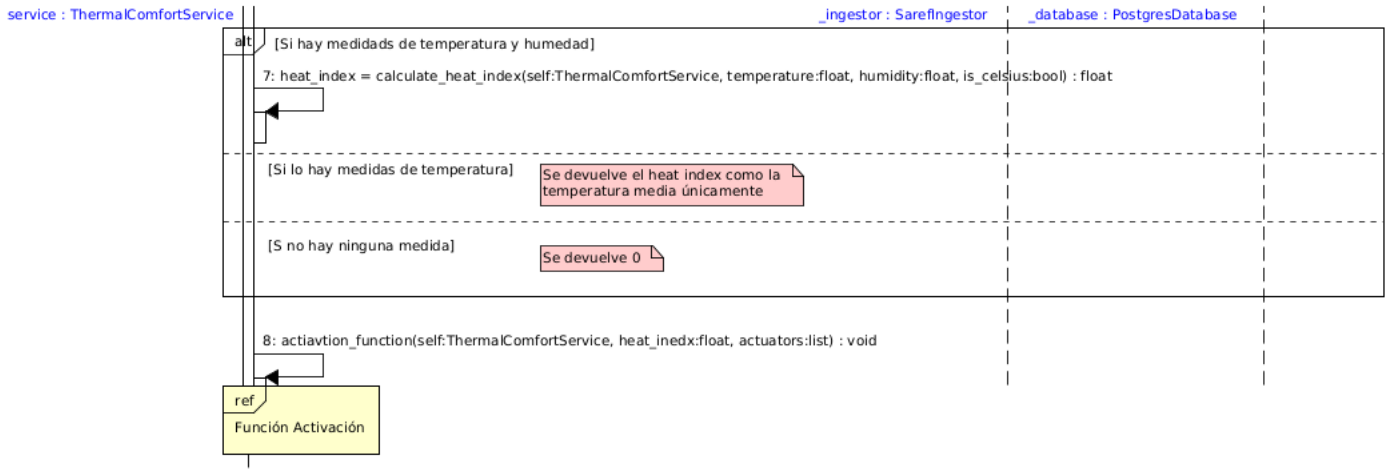


Figura 7.15: Parte 2 de la secuencia del método `run` del ThermalComfortService.

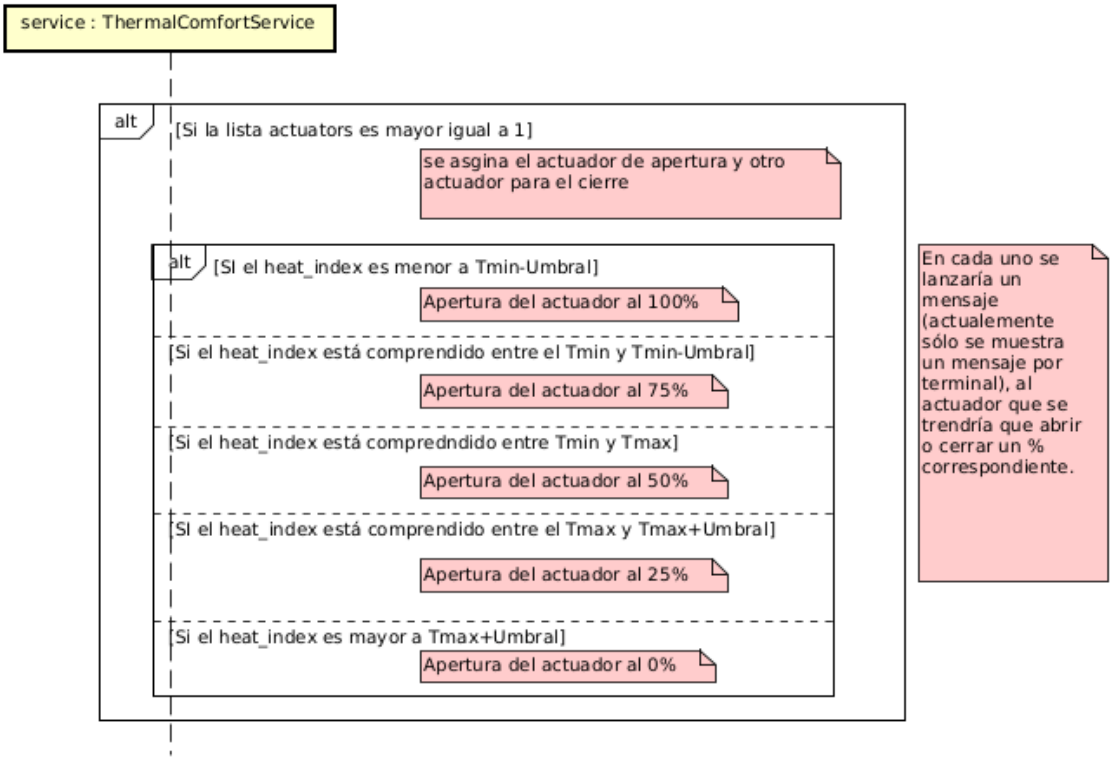


Figura 7.16: Función de activación de los actuadores.

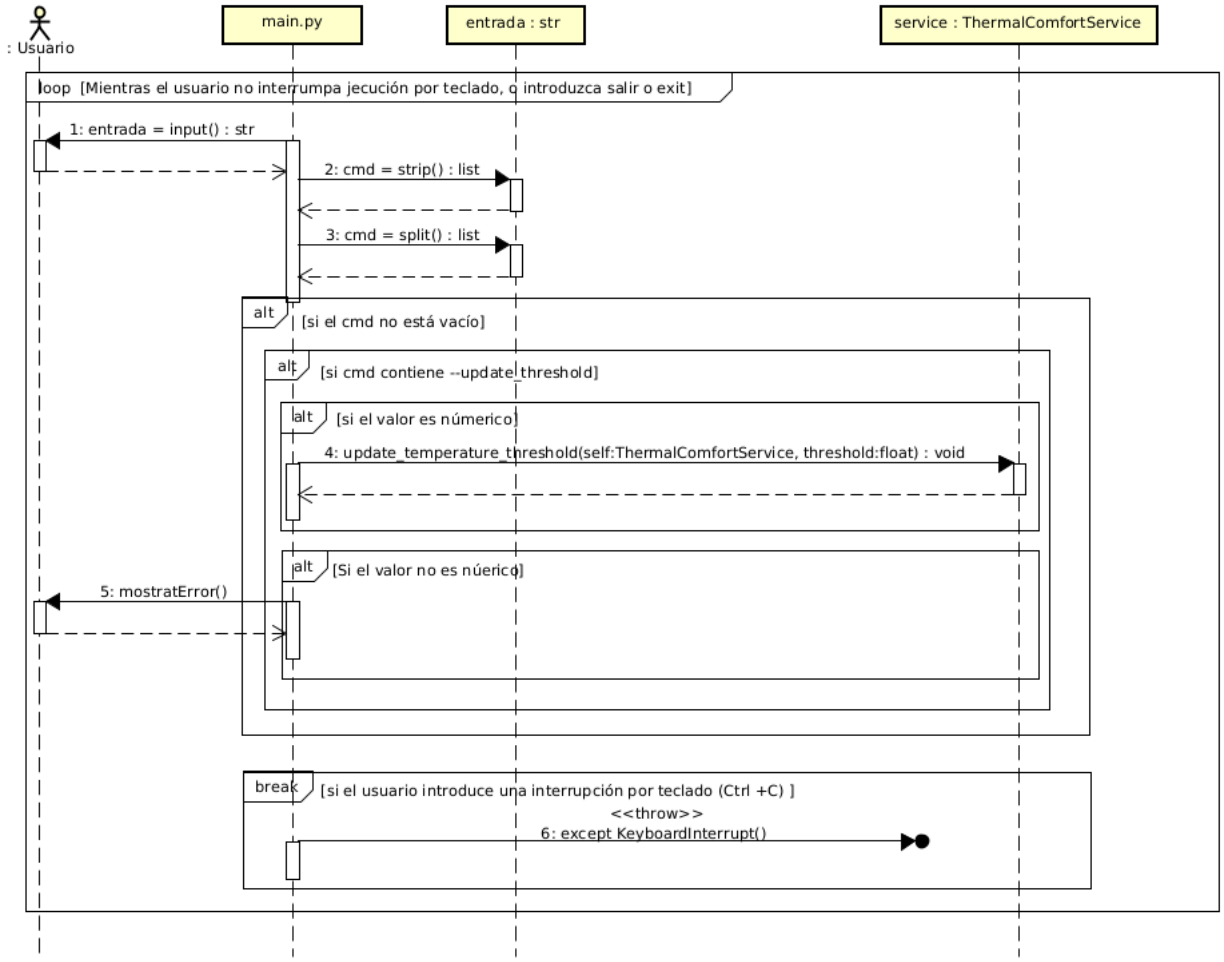


Figura 7.17: Secuencia de mensajes para actualizar el umbral.

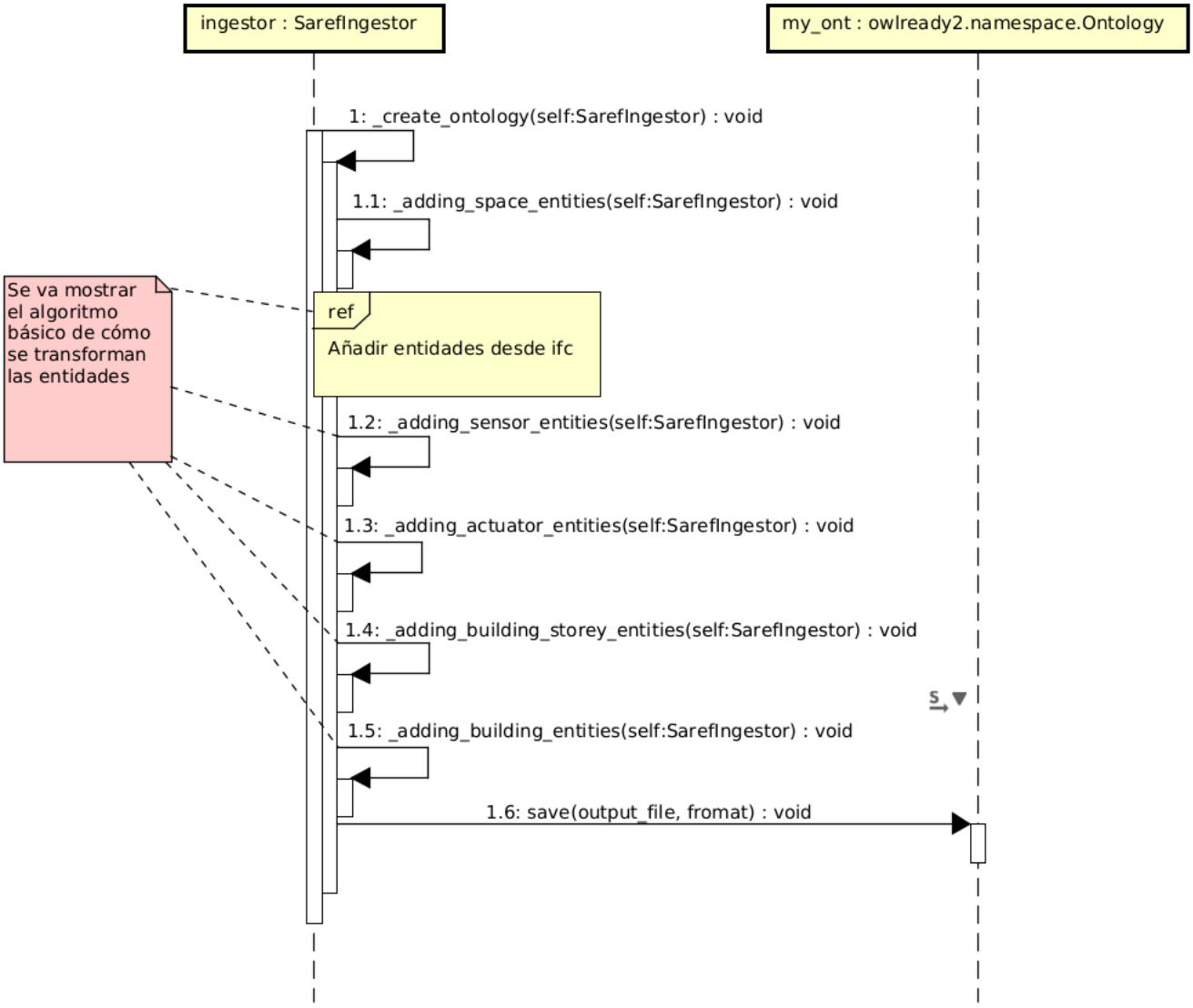


Figura 7.18: Secuencia de mensajes para crear el RDF.

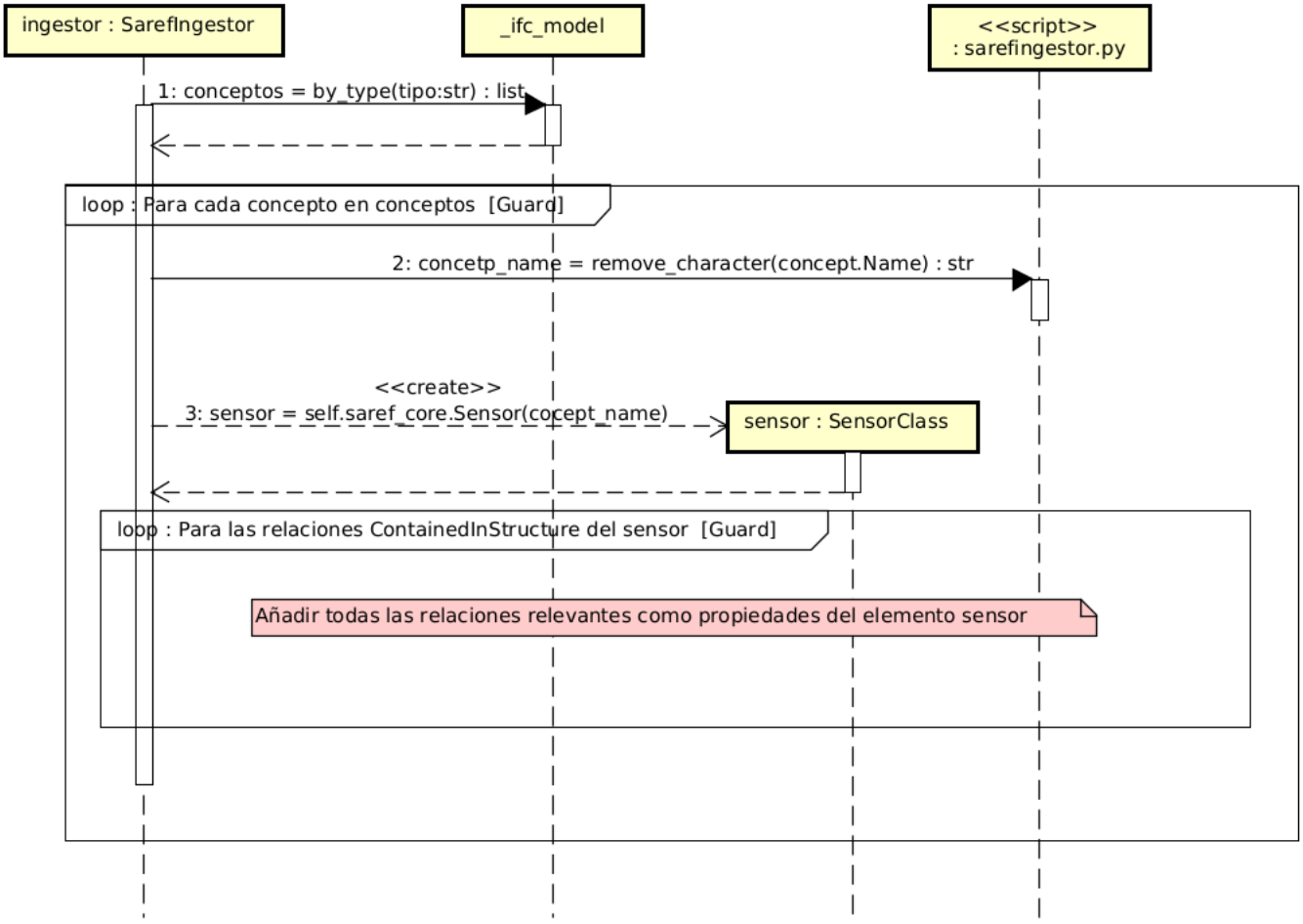


Figura 7.19: Secuencia de mensajes para importar las entidades y sus propiedades.

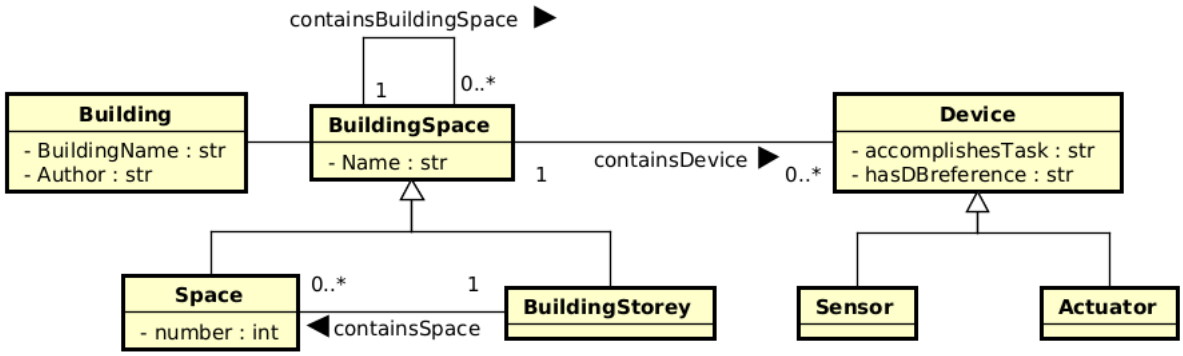


Figura 7.20: Modelo de dominio de entidades de la ontología implicadas en ejecución del servicio.

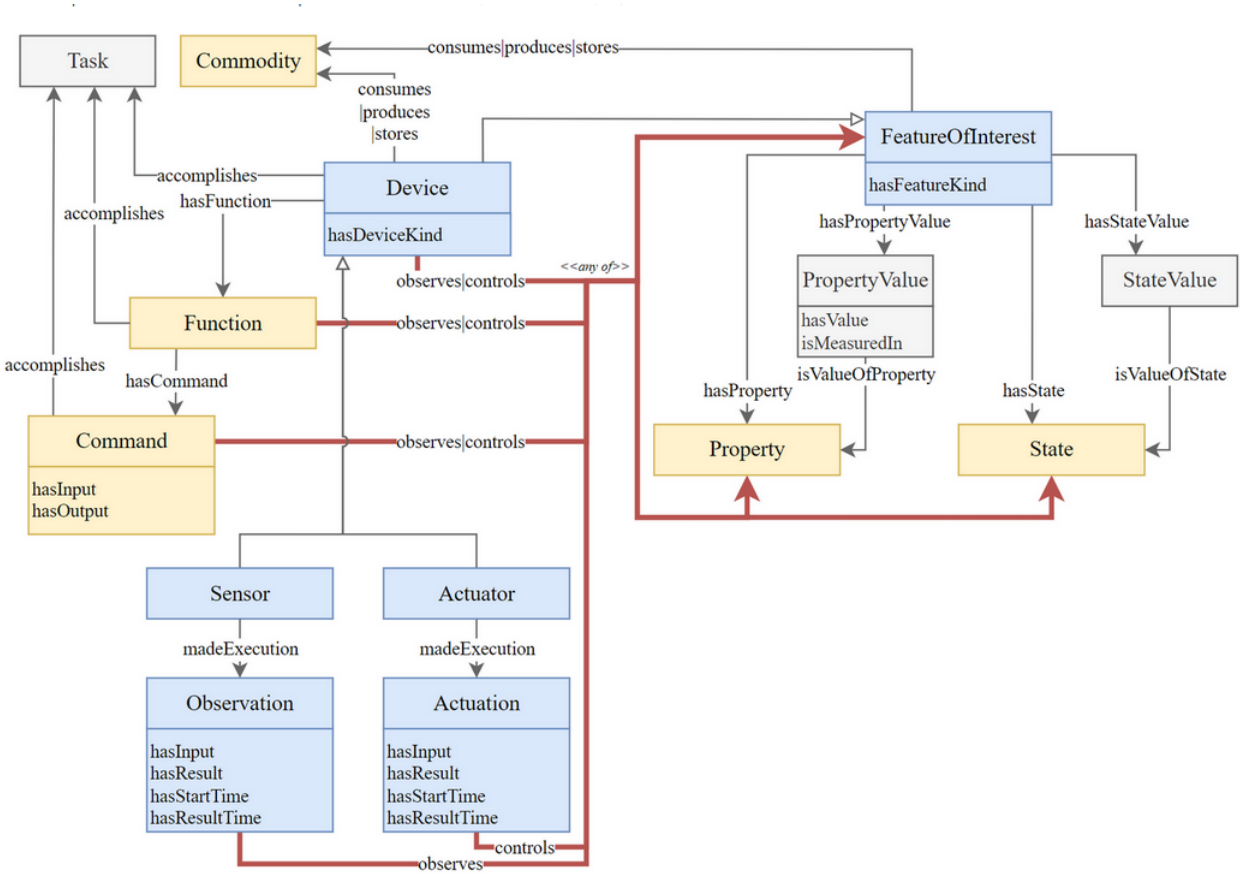


Figura 7.21: Modelo de dominio de SAREF.
Fuente: [ETSI]

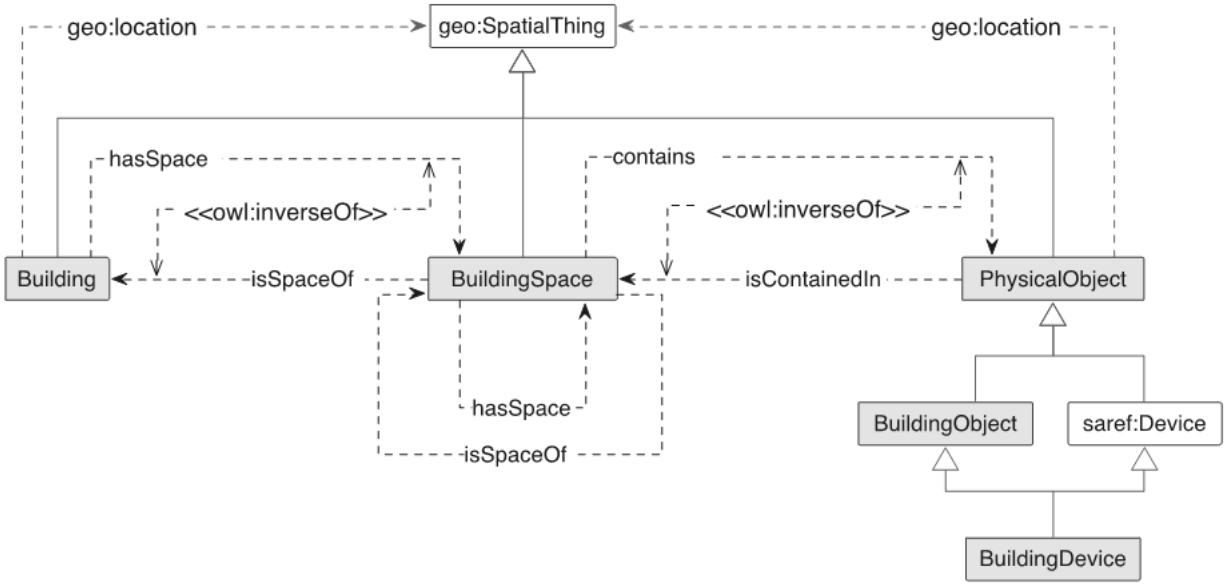


Figura 7.22: Modelo de dominio de S4BLDG.
Fuente: [ETSI]

Capítulo 8

Pruebas

En cualquier proyecto software, la fase de pruebas es de obligado cumplimiento. En este caso, solo se han documentado las pruebas relativas al sistema en su completitud. Se ha optado por realizar pruebas de caja negra, con el fin de comprobar si las entradas proporcionadas permiten obtener los resultados esperados.

Se han llevado a cabo dos pruebas. La primera con un fichero IFC del edificio de CARTIF3, y la segunda con una demo creada a partir del fichero IFC proporcionado por ifcwiki, donde se han añadido los sensores y actuadores necesarios para ejecutar el servicio. El principal objetivo es comprobar si se activarían o no los actuadores en función de la temperatura en cada espacio del edificio.

En los siguientes apartados se describen las pruebas de caja negra realizadas sobre las dos demos basadas en ficheros IFC:

- **Demo 1**: Fichero IFC original de un edificio propio.
- **Demo 2**: Fichero IFC descargado de Internet y poblado con entidades de control.

8.1. Demo 1: Edificio CARTIF3

8.1.1. Características del edificio

La demo 1 del edificio CARTIF3 cuenta con tres plantas, pero las pruebas solo se han centrado en la segunda planta (Figura 8.1), que dispone de siete espacios equipados con distintos sensores y actuadores distribuidos como se indica a continuación:

- **Espacio 4**: Cuenta con un sensor de temperatura, un actuador de cierre y otro de apertura.
- **Espacio 5**: Equipado únicamente con un sensor de temperatura.
- **Espacio 6**: Dispone de un actuador de cierre y otro de apertura.
- **Espacio 7**: Incluye solo un sensor de temperatura.
- **Espacio 8**: Cuenta únicamente con un sensor de temperatura.
- **Espacio 9**: Equipado con un sensor de temperatura, un sensor de humedad, un sensor de concentración de CO_2, además de un actuador para apertura y otro para cierre de válvulas.
- **Espacio 10**: No presenta una separación física con el espacio 9, pero dispone de un sensor de temperatura y actuadores para apertura y cierre de válvulas.

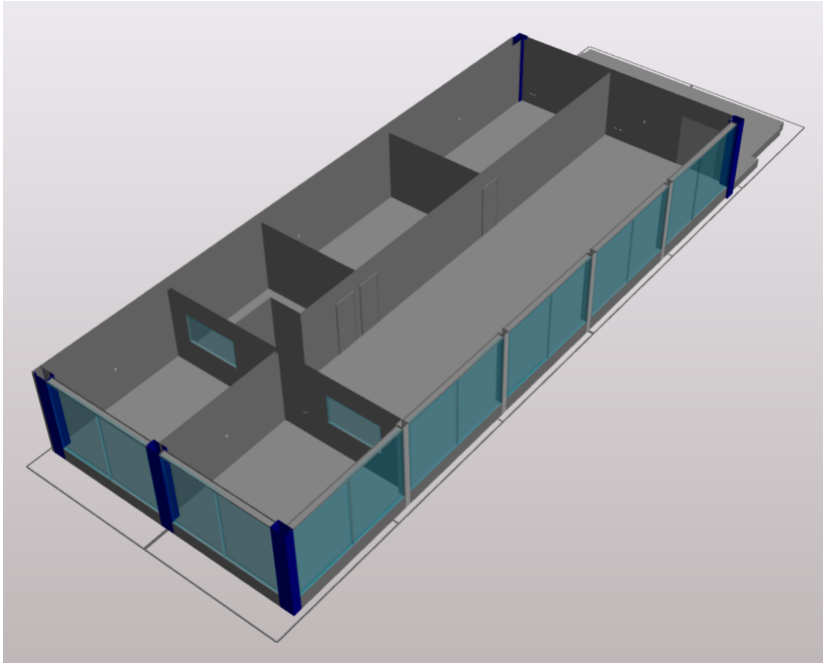


Figura 8.1: Representación 3D de la 2ª Planta de CARTIF3.

8.1.2. Descripción

Para realizar las pruebas se ha partido de datos almacenados en ficheros CSV, que disponen de valores correspondientes a los primeros 15 días de mayo. Estos datos son recogidos por los sensores del edificio cada 5 minutos. En las pruebas solo se van a usar los datos monitorizados por estos sensores, ya que las temperaturas se recogerán del fichero de configuración pyproject.toml. Así pues, lo que hace esta prueba es recoger datos de los CSV en un Dataframe y en cada ejecución del servicio, sin importar la duración del intervalo, coge los valores de la siguiente fila del Dataframe.

8.1.3. Resultados

Durante la ejecución del servicio de confort térmico se han observado tres situaciones destacadas.

En primer lugar, se identifican espacios donde el servicio funciona correctamente debido a que cuentan con sensores de temperatura y actuadores asociados. En estos casos, la temperatura media oscila entre 20 y 22 ^{o}C, valores que se encuentran dentro del rango de confort especificado por el sistema, por tanto, la decisión tomada por el servicio es abrir parcialmente (al 50 %) las válvulas.

En segundo lugar, aparecen espacios donde, pese a existir sensores y registrarse temperaturas razonables, no hay actuadores instalados. El sistema detecta correctamente esta ausencia e indica claramente en los resultados que no se puede realizar ninguna acción porque no existe ningún actuador en esas áreas específicas.

Finalmente, el problema más relevante que se ha identificado es que en algunas zonas existen actuadores pero no se dispone de sensores para medir temperatura ni humedad. Como consecuencia, el índice térmico calculado toma un valor de 0 ^{o}C, interpretado por la lógica del sistema como una situación de temperatura extremadamente baja. Esta situación provoca que el servicio ordene una apertura completa (al 100 %) del actuador correspondiente, pudiendo generar situaciones no deseadas como un aumento innecesario del consumo energético o disconfort térmico en los usuarios.

En conclusión, el servicio opera correctamente cuando dispone de los sensores y actuadores en los espacios proporcionados. Sin embargo, en las zonas con actuadores pero sin sensores, se comprueba que el sistema no funciona correctamente.

8.2. Demo 2: Edificio Haus

8.2.1. Características del edificio

La segunda demo, correspondiente al edificio Haus (Figura 8.2), dispone de una planta baja y un desván. Sin embargo, en las pruebas solo se ha considerado la planta baja, estructurada en seis espacios de la siguiente manera:

- **Espacio 2**: Equipado con un sensor de temperatura y dos actuadores para apertura y cierre de válvulas.
- **Espacio 3**: Dispone únicamente de un sensor de temperatura.
- **Espacio 4**: Presenta la misma configuración que el Espacio 2.

Los espacios 1, 5 y 6 no cuentan con límites físicos entre ellos, siendo el Espacio 1 un pasillo que comunica con los otros espacios:

- **Espacio 1**: Dispone de un sensor de temperatura.
- **Espacio 5**: Equipado con un sensor de temperatura, un sensor de humedad y un sensor de concentración de CO_2.
- **Espacio 6**: Cuenta con un actuador de cierre y otro de apertura.

De esta forma se pretende verificar si el sistema inserta y gestiona correctamente la información relativa a los sensores y actuadores definidos en cada demo.

8.2.2. Descripción de la prueba

En esta demo se generan temperaturas aleatorias con el método de Chow y Levermore [34]. Este método recibe como entrada unos valores de temperatura y genera un modelo que devuelve una posible temperatura para una hora concreta. A este modelo se le aplica ruido aleatorio a través de una distribución normal. En este caso, se ha decidido que la toma de datos con los que generar el modelo sean los del día 1 de abril de 2025 a las 12 del mediodía, y que el fichero de configuración pyproject.toml contenga el umbral a 3 grados Celsius, la temperatura mínima a 18 grados Celsius y la temperatura máxima a 24 grados Celsius.

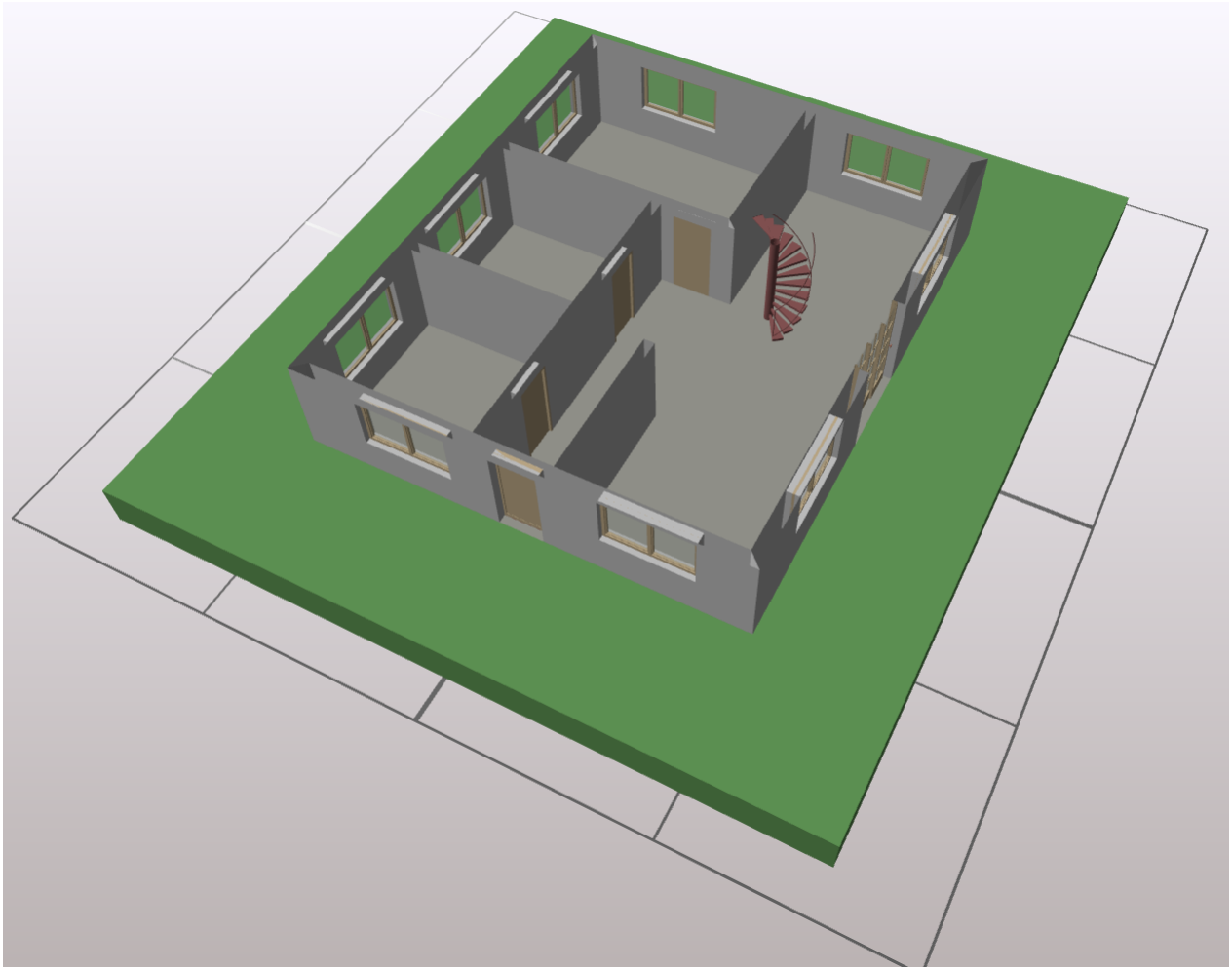


Figura 8.2: Representación 3D del edificio de Haus.
Fuente: Fichero IFC tomado de ifcwiki

8.2.3. Resultados

Durante la ejecución, al generar los datos en función de la fecha proporcionada, se dieron dos situaciones. En la primera, las temperaturas se mantuvieron dentro del rango configurado, dando lugar a que el sistema tomase la decisión de ajustar las válvulas a un 50 % de apertura. En cambio, el Espacio 2 presentó temperaturas ligeramente inferiores al rango de confort, por lo que el sistema incrementó la apertura de las válvulas hasta el 75 % para aumentar la temperatura. En la segunda, como en el caso anterior, dado que hay espacios que no tienen sensores de humedad y el cálculo del heat index está basado únicamente en los valores de temperatura, se reduce la precisión de confort térmico en condiciones que podrían implicar niveles altos de humedad relativa. Esto puede desembocar en que ante valores muy pequeños de la temperatura, por ejemplo en estaciones frías como invierno, el heat index llegue a ser negativo, lo que no tiene mucho sentido. No obstante, el servicio seguiría funcionando y ejecutaría la opción de abrir las válvulas al 100 %.

Capítulo 9

Conclusiones

Este TFG me ha permitido introducirme en el campo del modelado de información en edificios, así como descubrir la gran variedad de tecnologías existentes. Además, he podido comprobar cómo el uso de modelos semánticos ayuda a crear un modelo de datos único con el fin de disponer de un mismo vocabulario para todas las partes interesadas. Asimismo, he observado que aún falta mucho por explorar en cuanto a la aplicación de modelos semánticos se refiere, ya que suelen crearse con un objetivo específico con el fin de satisfacer unas necesidades concretas. A pesar de que todos comparten la misma tecnología subyacente no siguen una metodología de forma rigurosa, lo que termina generando modelos mal formados. A todo esto se suma la falta de documentación y de aclaraciones sobre cómo utilizar estos modelos, ya que sólo muestran cómo se definen los conceptos, ralentizando el desarrollo de nuevos modelos basados en trabajos previos.

9.1. Objetivos cumplidos

Una vez concluido todo el trabajo planificado, se puede afirmar que se han cumplido los objetivos principales propuestos en el Capítulo 1.2. Por un lado, se ha construido una ontología mediante el uso de SAREF, S4BLDG e IFC, pero con un número reducido de conceptos y sin llegar a publicar los datos para que se puedan consultar de forma externa. Por otro lado, se ha desarrollado un servicio separando responsabilidades en paquetes para entender mejor qué hace cada uno de ellos, aunque se podría haber modularizado más el código para separar aún más esas responsabilidades. No obstante, el trabajo realizado ha sido gratificante y con alto nivel de satisfacción.

9.2. Trabajo futuro

Existen determinados aspectos que se pueden tener en cuenta con el fin de refinar y extender la funcionalidad implementada. Uno de ellos consiste en mejorar la función de activación con intervalos continuos del Heat Index y no sobre una función definida a trozos. Además, se puede considerar el nivel de apertura de los actuadores para determinar con más detalle el tipo de mensaje a enviar.

Otro aspecto a considerar son las pruebas. Dado que no se ha podido probar su funcionamiento en un entorno real para verificar si realmente ayuda a reducir el gasto energético, sería adecuado realizar pruebas en el edificio de CARTIF3 para analizar cómo se comporta el servicio en un entorno con múltiples sistemas críticos y que es utilizado, a su vez, por otros miembros de la empresa para experimentar. Esto conllevaría realizar un estudio de la arquitectura en la que los sensores almacenan la información y de cómo acceden a esta, así como el tipo de comunicación que hay que utilizar para enviar mensajes a los actuadores para que lleven a cabo las funciones necesarias. Un último aspecto que se podría abordar en esta fase de pruebas es la refactorización de código, ya que las tres capas construidas no dejan de ser 'God Components' y, por tanto, tienen más responsabilidades de las que debieran.

El siguiente aspecto estaría enfocado en cómo enriquecer la ontología mediante su integración con otras ontologías, como TUBES, para la interconexión entre edificios. Entre ellas se encuentra la extensión de SAREF for Energy (S4ENER), que permite contemplar la flexibilidad de la demanda.

Por último, otro aspecto relevante que destaca por su contribución científica está relacionado con la investigación y desarrollo de protocolos avanzados para la integración de ontologías. Si bien ya existen en la actualidad diversos algoritmos que permiten el matching y alineamiento entre ontologías, se podrían evaluar protocolos que faciliten el matching inicial y la gestión continua de actualizaciones en ontologías dinámicas o en constante evolución. Este enfoque permitiría mantener integración de ontologías de forma robusta y eficiente a lo largo del tiempo, lo cual es crítico en aplicaciones que demanden interoperabilidad semántica constante.

Bibliografía

[1] Organización de Naciones Unidas. *Objetivos de Desarollo Sostenible*. Consultado el 4 de febrero de 2025. 2022. URL: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/.

[2] Parlamento Europeo. *Ahorro de energía: medidas de la UE para reducir el consumo energético*. Consultado el 4 de febrero de 2025. 2023. URL: https://www.europarl.europa.eu/topics/es/article/20221128STO58002/ahorro-de-energia-medidas-de-la-ue-para-reducir-el-consumo-energetico.

[3] Raúl García-Castro y col. «The ETSI SAREF ontology for smart applications: a long path of development and evolution». En: *Energy Smart Appliances: Applications, Methodologies, and Challenges* (2023), págs. 183-215.

[4] Amir Laadhar, Christian Thomsen y Torben Bach Pedersen. «domOS Common Ontology: Web of Things Discovery in Smart Buildings». En: *The Semantic Web: ESWC 2022 Satellite Events*. Ed. por Paul Groth y col. Cham: Springer International Publishing, 2022, págs. 95-100. ISBN: 978-3-031-11609-4.

[5] Sebastian Kaebisch, Michael McCool y Ege Korkan. *Web of Things (WoT) Thing Description 1.1*. Inf. téc. Consultado el 4 de febrero de 2025. W3C, 2023. URL: https://www.w3.org/TR/wot-thing-description11/.

[6] María Poveda-Villalón, Asunción Gómez-Pérez y Mari Carmen Suárez-Figueroa. «OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation». En: *International Journal on Semantic Web and Information Systems (IJSWIS)* 10.2 (2014), págs. 7-34.

[7] Minghui Ma y Ahti-Veikko Pietarinen. «Gamma Graph Calculi for Modal Logics». En: *Synthese* 195.8 (2018), págs. 3621-3650. URL: http://www.jstor.org/stable/26750494.

[8] D.D. Roberts. *The Existential Graphs of Charles S. Peirce*. Approaches to semiotics. Mouton, 2009. ISBN: 9783110226218. URL: https://books.google.es/books?id=7gIOVybdtHMC.

[9] Allan M. Collins y M. Ross Quillian. «Retrieval time from semantic memory». En: *Journal of Verbal Learning and Verbal Behavior* 8.2 (1969), págs. 240-247. ISSN: 0022-5371. DOI: https://doi.org/10.1016/S0022-5371(69)80069-1. URL: https://www.sciencedirect.com/science/article/pii/S0022537169800691.

[10] Marvin Minsky. «A Framework for Representing Knowledge». En: (1974). Consultado el 3 de marzo de 2025. URL: https://courses.media.mit.edu/2004spring/mas966/Minsky%201974%20Framework%20for%20knowledge.pdf.

[11] Stuart J. Russell y Peter Norving. *Artificial Intelligence : A modern approach*. eng. 3rd. ed. Prentice Hall series in artificial intelligence. Upper Saddle River, New Jersey: Prentice-Hall, 2010. ISBN: 978-0-13-207148-2.

[12] David L. Poole y Alan K Mackworth. *Artificial intelligence : foundations of computational agents*. eng. New York: Cambridge University Press, 2018. ISBN: 978-1-107-19539-4.

[13] Thomas R. Gruber. «Toward principles for the design of ontologies used for knowledge sharing?» En: *International Journal of Human-Computer Studies* 43.5 (1995), págs. 907-928. ISSN: 1071-5819. DOI: https://doi.org/10.1006/ijhc.1995.1081. URL: https://www.sciencedirect.com/science/article/pii/S1071581985710816.

[14] Rudi Studer, V.Richard Benjamins y Dieter Fensel. «Knowledge engineering: Principles and methods». En: *Data & Knowledge Engineering* 25.1 (1998), págs. 161-197. ISSN: 0169-023X. DOI: https://doi.org/10.1016/S0169-023X(97)00056-6. URL: https://www.sciencedirect.com/science/article/pii/S0169023X97000566.

[15] Nicola Guarino. «Semantic matching: Formal ontological distinctions for information organization, extraction, and integration». En: *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology*. Ed. por Maria Teresa Pazienza. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, págs. 139-170. ISBN: 978-3-540-69548-6.

[16] Riccardo Tommasini. *Streaming Linked Data : From Vision to Practice*. eng. 1st ed. 2023. Cham: Springer International Publishing, 2023. ISBN: 9783031153709.

[17] Dean Allemang y James A Hendler. *Semantic web for the working ontologist : modeling in RDF, RDFS and OWL*. eng. Amsterdam ; Morgan Kaufmann Publishers/Elsevier, 2008. ISBN: 978-0-12-373556-0.

[18] W3C. *OWL 2 Web Ontology Language: Profiles*. Consultado el 24 de marzo de 2025. 2009. URL: https://www.w3.org/TR/2009/REC-owl2-profiles-20091027/.

[19] JiaoYue Wang. «Large ABox Store (LAS) : Database Support for TBox Queries». En: 2005. URL: https://api.semanticscholar.org/CorpusID:18429247.

[20] International Organization for Standardization. *ISO 19650-1:2018: Organization and digitization of information about buildings and civil engineering works, including building information modeling (BIM) — Information management using building information modeling — Part 1: Concepts and principles*. Geneva, Switzerland: ISO, 2018.

[21] National Institute of Building Sciences. *National BIM Standard - United States*. Washington, D.C., USA: National Institute of Building Sciences (NIBS), 2022.

[22] buildingSMART International. *Industry Foundation Classes (IFC)*. https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/. Consultado el 25 de ferbero 2025. 2023.

[23] buildingSMART International. *Industry Foundation Classes (IFC)*. Consultado el 27 de febrero de 2025. 2023. URL: https://www.buildingsmart.es/bim/openbim/ifc/.

[24] Mohammed A. Saeed y col. «Energy management system in smart buildings based coalition game theory with fog platform and smart meter infrastructure». En: *Scientific Reports* 13.1 (2023), pág. 2023. ISSN: 2045-2322. DOI: 10.1038/s41598-023-29209-4. URL: https://doi.org/10.1038/s41598-023-29209-4.

[25] California Energy Commission. *Glossary: Energy Management Control System (EMCS)*. Consultado el 20 de marzo de 2025. California Energy Commission, 2019. URL: https://energycodeace.com/site/custom/public/reference-ace-2019/index.html#!Documents/gloss_energymanagementcontrolsystememcs.htm.

[26] Nlyte Software. *BMS versus BAS: Key Differences and FAQs*. https://www.nlyte.com/faqs/bms-versus-bas/. Consultado el 3 de marzo 2025. 2023. URL: https://www.nlyte.com/faqs/bms-versus-bas/.

[27] TekWorx. *Energy Management Systems*. Consultado el 3 de marzo 2025. 2023. URL: https://www.tekworx.us/blog/energy-management-systems/.

[28] HVAC Web Connection. *BAS or EMS*. Consultado el 3 de marzo 2025. 2016. URL: https://www.hvacwebconnection.com/news16/bas_or_ems.htm.

[29] European Commission. *Energy System Integration*. https://energy.ec.europa.eu/topics/eus-energy-system/energy-system-integration_en. Consultado el 28 de febrero 2025. 2023. URL: https://energy.ec.europa.eu/topics/eus-energy-system/energy-system-integration_en.

[30] Michael Uschold y Martin King. «Towards a Methodology for Building Ontologies». En: 1995. URL: https://api.semanticscholar.org/CorpusID:13963021.

[31] Mike Uschold y Michael Gruninger. «Ontologies: principles, methods and applications». En: *The Knowledge Engineering Review* 11.2 (1996), págs. 93-136. DOI: 10.1017/S0269888900007797.

[32] M Fernández-López, A Gómez-Pérez y Natalia Juristo Juzgado. «METHONTOLOGY: From Ontological Art Towards Ontological Engineering». eng. En: Facultad de Informática (UPM), 1997.

[33] A Gómez-Pérez, M Fernández y A. de Vicente. «Towards a Method to Conceptualize Domain Ontologies». eng. En: Facultad de Informática (UPM), 1996.

[34] D.H.C. Chow y Geoff J. Levermore. «New algorithm for generating hourly temperature values using daily maximum, minimum and average values from climate models». En: *Building Services Engineering Research & Technology* 28.3 (2007), págs. 237-248. DOI: 10.1177/0143624407078642. eprint: https://doi.org/10.1177/0143624407078642. URL: https://doi.org/10.1177/0143624407078642.

Apéndice A

Especificaciones técnicas

Primero que nada, hay que apuntar que todo el software ha sido ejecutado en 2 máquinas diferentes y cuyas especificaciones vienen en la tabla A.1.

Se ha lanzado un ps aux con el servicio en ejecución y con ello comprobar el tamaño que ocupa el proceso en ejecución. El resultado puede verse en A.1. En ellos se deja ver cómo el porcentaje de uso de CPU no es muy elevado y con lo que se puede afirmar que el proceso no genera mucha carga, y que el uso de memoria es de 203 940 KB, aproximadamente 200 MB. Por esta razón, las especificaciones técnicas de los equipos utilizados no se consideran los requisitos mínimos para la ejecución del servicio y se considera que con equipos más modestos el servicio puede ser ejecutado perfectamente.

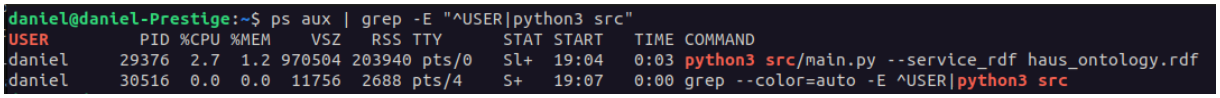


Figura A.1: Imagen de la ejecución de ps aux.

Equipo	**MSI Prestige 14H B12UCX**	**HP ProBook 440 14 G11 (WSL2)**
Sistema Operativo	Ubuntu 22.04 LTS	Ubuntu 24.04 LTS (sobre WSL2 en Windows 11 Pro)
Modelo del Equipo	MSI Prestige 14H B12UCX	HP ProBook 440 14 G11
Procesador	Intel Core i5-1245H (12 núcleos / 16 hilos, hasta 4.40 GHz)	Intel Core i5-125U (10 núcleos / 12 hilos, 1.30 GHz base, hasta 4.40 GHz)
Memoria RAM	16 GB LPDDR5 a 4800 MHz	16 GB LPDDR5 a 5600 MHz
Almacenamiento	SSD NVMe de 512 GB	SSD NVMe de 512 GB
GPU	NVIDIA GeForce RTX 2050, 4 GB GDDR6	No aplica (WSL2 sin GPU dedicada)
Entorno Linux	Nativo	Virtualizado (WSL2 sobre Hyper-V)

Tabla A.1: Especificaciones de las Máquinas de Desarrollo Utilizadas

Apéndice B

Manuales

En este anexo se indican los pasos que debería seguir un empleado de la empresa CARTIF3 para extender la funcionalidad del servicio desarrollado en este trabajo.

B.1. Manual de Instalación

B.1.1. PostgreSQL y Configuración del OCI

En primer lugar, hay que configurar la base de datos PostgreSQL. Esto conlleva instalar el sistema gestor de forma local, configurar el puerto por defecto, crear una base de datos, asignarle un nombre y crear un usuario con su contraseña de acceso.

En la máquina virtual de OCI donde reside la base de datos, la configuración es la misma que en local. La única diferencia radica en que hay que configurar el servidor para sea accesible desde una red externa. Para ello, es necesario realizar los siguientes ajustes:

- Crear una Compute Instance con una imagen de Ubuntu.
- Crear una Virtual Network Interface Card (VNIC) con una subred pública.
- Modificar la tablas de seguridad para añadir el acceso vía TCP al puerto especificado.
- Acceder vía SSH a la instancia creada con la *ssh-key* privada.
- Modificar iptables para que permita aceptar conexiones TCP en el puerto de la base de datos.

- Instalar PostgreSQL en la máquina que actuará como servidor: 'sudo apt-get install postgresql'

- Acceder a la base de datos PostgreSQL (psql -i -u), crear un usuario, una contraseña, una base de datos y asignar privilegios al usuario creado.

B.1.2. Instalación del Software

En primer lugar, se recomienda tener instalado Ubuntu en su versión LTS más reciente o, al menos, Windows 10 en su versión 2004.

0) Instalar Git y Descargar el Repositorio

```
git clone https://venus.cartif.es/energias/tfg_danieljimenez
```

1) Instalar PostgreSQL

Ubuntu

`https://www.postgresql.org/download/linux/ubuntu/`

Windows

`https://www.postgresql.org/download/windows/`

Nota 1: Además de configurar el nombre del host, el nombre de usuario, el nombre de la base de datos y el puerto, hay que crear las variables de entorno tal y como vienen especificadas en el fichero pyproject.toml

Nota 2: Si se va usar una base de datos externa que no esté hosteada en el PC de desarrollo, hay que tener en cuenta también el nombre del host, el nombre del usuario, su contraseña y el puerto que se utiliza.

2) Instalar Python

Requiere tener instalado Python `>= 3.10` en el Sistema Operativo (Linux o Windows)

3) (Opcional) Instalar `pyenv` y Preparar Entorno Virtual

En Linux:

```
curl https://pyenv.run | bash
exec $SHELL
```

En Windows 10/11: Para utilizar los entornos de Python hay que instalar el Subsistema de Windows para Linux (WSL) y, desde este, crear los entornos de Python como se ha indicado previamente. Para instalar WSL se recomienda seguir los pasos indicados en: https://learn.microsoft.com/es-es/windows/wsl/install

4) Verificar Instalación de `pip`

```
python3 -m pip --version
```

Si no está instalado, ejecutar:

```
python -m ensurepip --upgrade
```

O descargar el archivo get-pip.py de Bootstrap y ejecutar:

```
python get-pip.py
```

5) (Opcional) Crear y Activar el Entorno Virtual con `pyenv`

5.1) Crear entorno:

```
pyenv virtualenv VERSION_PYTHON NOMBRE_ENTORNO
```

5.2) Activar entorno:

```
pyenv activate NOMBRE_ENTORNO
```

6) Instalar Dependencias desde requirements.txt

Ejecutar desde la carpeta `ontoenergy/` el siguiente comando:

```
pip install -r requirements.txt
```

7) Lanzar el Servicio

Ejecutar desde la carpeta `ontoenergy/` el siguiente comando:

```
python3 src/main.py --service_rdf NOMBRE_ONTOLOGIA_RDF
```

7.1) Parámetros Adicionales

```
--service_input NOMBRE_IFC_EN_DIRECTORIO_INPUT
--update_threshold VALOR_NUMERICO
```

Nota 1: El parámetro `--service_input` no es obligatorio, pero si no se introduce y no está creado el fichero `NOMBRE_ONTOLOGIA_RDF`, pedirá que se introduzca el nombre del fichero IFC con el que se quiere crear el RDF de la ontología.

Nota 2: Los umbrales de temperatura por defecto están definidos en el fichero pyproject.toml. Para modificarlos, basta con editar este archivo.

8) Cierre del Servicio

El servicio continuará en ejecución hasta que se presione `Ctrl+C` o se escriba `exit`.

9) Desactivar el Entorno Virtual

```
pyenv deactivate
```

B.2. Manual de Desarrollo

Para añadir una nueva funcionalidad al sistema hay que seguir los pasos que se describen a continuación. En este caso, se muestran con un ejemplo en el que se añade un servicio de control de CO_2.

Comprobar fichero IFC y RDF

En primer lugar, hay que garantizar que las entidades relacionadas van a estar presentes en el RDF y, por lo tanto, se tiene una referencia de las mismas en la base de datos relacional. En este ejemplo se dispone de una entidad "Sensor de CO_2". Si no fuese así, habría que com-

probar que existe ese concepto en IFC y, después, crear el método que mapee de IFC a RDF. Si no hay una entidad en la sintaxis de IFC para mapear, lo que se puede hacer es crear directamente en el fichero RDF el concepto en cuestión. Esta creación se puede hacer directamente editando el fichero en un bloc de notas y añadiendo la entidad y las propiedades correspondientes (Figura B.1). Otra forma de hacerlo sería de forma dinámica con la librería owlready2 (Figura B.2)

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xml:base="/my_ontology_example.owl"
         xmlns="/my_ontology_example.owl#">

<owl:Ontology rdf:about="/my_ontology_example.owl"/>

<owl:DatatypeProperty rdf:about="#device_db_reference">
  <rdfs:domain rdf:resource="https://saref.etsi.org/core/Device"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:Class rdf:about="#CO2Sensor">
  <rdfs:subClassOf rdf:resource="https://saref.etsi.org/core/Device"/>
</owl:Class>


</rdf:RDF>
```

Figura B.1: Crear entidad en fichero RDF.

```
from owlready2 import *

onto = get_ontology("url").load()
saref_core = get_ontology("http://www.saref.etsi.org/core/").load()
with onto:
    co2_sensor = types.new_class("CO2Sensor", (saref_core.Device,))

    device_db_reference_property = types.new_class("device_db_reference", (DataProperty,))
    device_db_reference_property.domain = [saref_core.Device]
    device_db_reference_property.range = [str]
```

Figura B.2: Crear entidad de manera dinámica.

Crear la Clase para el Nuevo Servicio

Se crea una nueva clase, por ejemplo: `ConcentrationC02Service`, en un nuevo fichero llamado concentration_co2_service.py. A continuación, se crea el `init` de la clase y la funcionalidad del servicio en un método `run` que se inicializará y ejecutará desde el fichero main.py. Esta clase debe acceder al RDF a partir del ingestor, que se puede crear desde cero o reutilizar el ya existente. Sea como fuere, hay que crear los métodos con las consultas SPARQL correspondientes que devuelvan los datos necesarios para ser ejecutados en el método `run` de la clase creada.

Ejecutar el Servicio

En este punto se pueden hacer varias aproximaciones. La primera consiste en crear un nuevo hilo para la ejecución del servicio a partir de una función, tal y como está implementado. La segunda consiste en añadir un parámetro a la función `schedule_service` con la instancia del nuevo servicio creado para que esa función ejecute repetidamente la nueva función `run`.

Apéndice C

Material Complementario

Fichero RDF

La Figura C.1 muestra la implementación de la ontología en el fichero RDF generado.

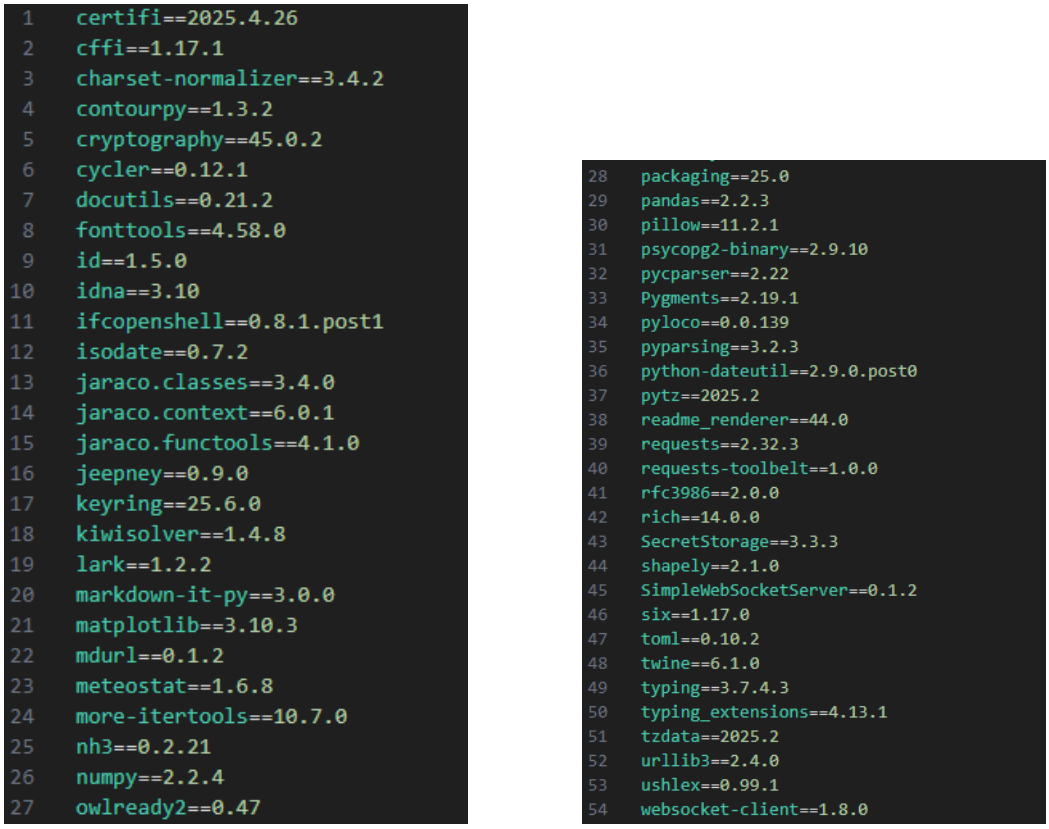
```
<owl:NamedIndividual rdf:about="#9">
  <rdf:type rdf:resource="#Space"/>
  <containsDevice rdf:resource="#SENSORTEMPERATURE:678"/>
  <containsDevice rdf:resource="#SENSORCO2:672"/>
  <containsDevice rdf:resource="#SENSORHUMIDITY:614"/>
  <containsDevice rdf:resource="#Three-way_valve-Caleffi-1060"/>
  <containsDevice rdf:resource="#Three-way_valve-Caleffi-1074"/>
  <Category rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Spaces</Category>
  <PhaseId rdf:datatype="http://www.w3.org/2001/XMLSchema#string">0</PhaseId>
  <id rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">525</id>
  <Name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">SALA_OFICINAS2</Name>
  <Number rdf:datatype="http://www.w3.org/2001/XMLSchema#string">9</Number>
  <RoomName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">9</RoomName>
  <RoomNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string">9</RoomNumber>
  <id rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">531</id>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="#SENSORTEMPERATURE:678">
  <rdf:type rdf:resource="https://saref.etsi.org/core/Sensor"/>
  <accomplishesTask rdf:resource="#Measure_temperature"/>
  <Category rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Communication Devices</Category>
  <FamilyName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">LOGS-01_SENSOR</FamilyName>
  <Family rdf:datatype="http://www.w3.org/2001/XMLSchema#string">LOGS-01_SENSOR</Family>
  <FamilyandType rdf:datatype="http://www.w3.org/2001/XMLSchema#string">SENSOR: Schneider STR100 TEMPERATURE</FamilyandType>
  <Type rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Schneider STR100 TEMPERATURE</Type>
  <TypeId rdf:datatype="http://www.w3.org/2001/XMLSchema#string">641635</TypeId>
  <Description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Room Controller</Description>
  <Manufacturer rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Schneider</Manufacturer>
  <Model rdf:datatype="http://www.w3.org/2001/XMLSchema#string">STR100</Model>
  <OmniClassNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string">23.75.65.11</OmniClassNumber>
  <OmniClassTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Monitoring and Control of Internal Climate</OmniClassTitle>
  <ProductName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">ROOM TEMPERATURE SENSOR STR100</ProductName>
  <TypeName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Schneider STR100 TEMPERATURE</TypeName>
  <URL rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    https://www.schneider-electric.es/es/product/STR100/str100---sensor-de-temperatura-interior/
  </URL>
  <hasDBreference rdf:datatype="http://www.w3.org/2001/XMLSchema#string">VARIABLE\SENSOR\TEMPERATURE_678</hasDBreference>
  <Reference rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Schneider STR100 TEMPERATURE</Reference>
  <id rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">8474</id>
</owl:NamedIndividual>
```

Figura C.1: Propiedades presentes en los conceptos de la ontología.

Fichero requirements.txt

La Figura C.2 muestra el fichero requirements.txt, que contiene las dependencias que existen con otras librerías utilizadas.



```
certifi==2025.4.26
cffi==1.17.1
charset-normalizer==3.4.2
contourpy==1.3.2
cryptography==45.0.2
cycler==0.12.1
docutils==0.21.2
fonttools==4.58.0
id==1.5.0
idna==3.10
ifcopenshell==0.8.1.post1
isodate==0.7.2
jaraco.classes==3.4.0
jaraco.context==6.0.1
jaraco.functools==4.1.0
jeepney==0.9.0
keyring==25.6.0
kiwisolver==1.4.8
lark==1.2.2
markdown-it-py==3.0.0
matplotlib==3.10.3
mdurl==0.1.2
meteostat==1.6.8
more-itertools==10.7.0
nh3==0.2.21
numpy==2.2.4
owlready2==0.47
packaging==25.0
pandas==2.2.3
pillow==11.2.1
psycopg2-binary==2.9.10
pycparser==2.22
Pygments==2.19.1
pyloco==0.0.139
pyparsing==3.2.3
python-dateutil==2.9.0.post0
pytz==2025.2
readme_renderer==44.0
requests==2.32.3
requests-toolbelt==1.0.0
rfc3986==2.0.0
rich==14.0.0
SecretStorage==3.3.3
shapely==2.1.0
SimpleWebSocketServer==0.1.2
six==1.17.0
toml==0.10.2
twine==6.1.0
typing==3.7.4.3
typing_extensions==4.13.1
tzdata==2025.2
urllib3==2.4.0
ushlex==0.99.1
websocket-client==1.8.0
```

Figura C.2: Dependencias existentes con otras librerías.

Repositorio en GitLab

En el siguiente enlace se encuentra disponible el repositorio de GitLab con el código del proyecto.

`https://gitlab.inf.uva.es/dajimen/ontoenergyproject`