Universidad de Sancti Spíritus "JOSÉ MARTÍ PÉREZ"
Facultad De Ciencias Técnicas
Carrera Ingeniería Informática

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERÍA EN INFORMÁTICA.

Título: Implementación de Thing Descriptions para Nodos sensores

Autor: Enrique Ricardo Ruiz García

Tutor: MSc Bernardo León Ávila

Sancti Spíritus
2022

Pensamiento

“Cuando lo inalámbrico esté perfectamente desarrollado, el planeta entero se convertirá en un gran cerebro, que de hecho ya lo es, con todas las cosas siendo partículas de un todo real y rítmico…

Nikola Tesla

Resumen

El *Internet of Things*, (*IoT*), y la *Web of Things*, (*WoT*), son dos piezas fundamentales para entender el segmento de la economía digital. El componente clave de los bloques de construcción de *WoT* es el ***Thing Description***, al actuar como capa de comunicación; facilitándole al sensor, interactuar y ser interactuado con el exterior. La presente investigación tuvo como objetivo general implementar un *Thing Description* para nodos sensores, dando salida a la problemática detectada en la interoperabilidad entre estos dispositivos. Para ello se utilizó como metodología el *Rational Unified Process*, cumpliendo sus principios y aplicando el Lenguaje Unificado de Modelado, facilitando la gestión de construcción de la aplicación. Se toma el *Django* como *framework* y se utiliza el Python como lenguaje de programación, para el desarrollo de la interfaz gráfica se trabajó con *Nuxtjs* como *framework* de desarrollo basado en *Nodejs* y utilizando *Veutify* como *framework* de Material *Design*. Como resultado se ha creado una aplicación que implementa un *Thing Description* para nodos sensores que facilitó la uniformidad en la presentación de datos, reduciendo la complejidad en la integración de los dispositivos.

Palabras claves: *Internet of Things*, *Web of Thing*, *Thing Description* y *Nodos sensores*.

Abstract

The Internet of Things, (IoT), and Web of Things, (WoT), are two fundamental pieces to understand the segment of the digital economy. The key component of WoT's Architecture is the Thing Description (TD). TD acts as a layer of communication, by enabling a device to interact with its environment. This research purpose was to design a Thing Description for motes, to tackle the main problem identified in the interoperability between the devices. The Rational Unified Process was used as methodology, by applying its principles and using the Unified Modeling Language (UML), to help the construction management of the application. For the design, Django was used as framework, and Python as programming language, for the development of the graphical interface we worked with Nuxtjs as a development framework based on Nodejs and using Veutify as a Material Design framework. As a result, an application was created to design a Thing Description for motes that facilitates a unified presentation of data, which reduces the complexity of the devices integration.

Keywords: *Internet of Things, Web of Things, Thing Description, and Motes.*

Índice

Contenido

Índice

Índice de Figuras

Índice

Introducción:

La Internet de las Cosas (*IoT*) es un tema importante en la industria de la tecnología, las políticas y los círculos de ingeniería y se ha convertido en noticia de primera plana, tanto en la prensa especializada como en los medios populares. Esta tecnología se encarna en una amplia gama de productos, sistemas y sensores en red, que aprovechan los avances en la potencia de cálculo, la miniaturización de los componentes electrónicos y las interconexiones de red para ofrecer nuevas capacidades que antes no eran posible. (Rose, Eldridge, y Chapin s. f.)

Aunque el término “Internet de las Cosas” es relativamente nuevo, el concepto de combinar computadoras y redes para monitorear y controlar diferentes dispositivos ha existido durante décadas. Por ejemplo, a fines de la década de 1970 ya había en el mercado sistemas disponibles para monitorear los medidores conectados a la red eléctrica de forma remota a través de las líneas telefónicas. En la década de 1990, los avances en la tecnología inalámbrica permitieron la difusión de soluciones corporativas e industriales “máquina a máquina” (*M2M*) para monitorear y operar diferentes equipos. Sin embargo, muchas de estas primeras soluciones *M2M* se basaban en redes, especialmente construidas para este propósito y en estándares específicos de la industria, no en redes basadas en el Protocolo de Internet (IP) y los estándares de Internet.(Rose et al. s. f.)

A comienzos del siglo XXI, gracias a la popularización de la conectividad inalámbrica (celular o *WiFi*), se produjo la primera explosión en el crecimiento de los objetos conectados. Este crecimiento se ha consolidado especialmente en lo últimos años, según han ido surgiendo nuevos conceptos como el WSN (Wireless Sensor Networks) o las nuevas tecnologías de acceso a radio. (Abbate s. f.)

El año 2008 fue un hito importante en la historia del Internet, este fue el primer año en el que los dispositivos conectados a Internet superaron al número de personas conectadas. (Abbate s. f.)

La historia del *IoT* se sigue escribiendo a diario. Por instantes aparecen nuevos dispositivos, nuevos protocolos, nuevas tecnologías de acceso, etc. que confluyen con los avances en otras tecnologías como la computación en la nube, *big data*, e

inteligencia artificial, enriqueciendo y dando cada día más oportunidades de crecimiento al universo *IoT*. Todo esto ha tenido un impacto enorme en el desarrollo de la humanidad, tanto económico como social. (School 2022)

Si las proyecciones de crecimiento para la *IoT* se convierten en realidad, podríamos ser testigos de un cambio hacia una interacción más pasiva en Internet, una interacción entre usuarios y objetos tales como componentes de automóviles, electrodomésticos y dispositivos de monitoreo personal; estos dispositivos envían y reciben datos en nombre del usuario, con poca intervención humana e incluso sin que nadie tenga conciencia de lo que está ocurriendo. (Rose et al. s. f.)

Para entender el desarrollo del *IoT* hay que contemplar cinco aspectos esenciales: esto implica una apuesta decidida por la miniaturización, por la que los componentes serán cada vez más pequeños (nanotecnología), lo que facilitará la conexión de prácticamente de cualquier cosa, desde cualquier sitio y momento. La capa de comunicaciones debe necesariamente construirse en torno al protocolo 5G y eliminando muchas de las barreras de la infraestructura de telefonía móvil actual. Conlleva la proliferación de servicios, aplicaciones, utilidades, rutinas y procedimientos que ponen en uso y en valor la cantidad de información creada a partir del *IoT*. Va a suponer un avance en la gestión de los datos y de la información de manera estructurada, eficiente y segura.(vida" y Pedagogía 2020)

La capacidad de obtención de datos continuos con sensores inteligentes hará que con esos mismos datos puedan tomar decisiones o ser capaces de acciones como inyectar la cantidad adecuada de insulina a un paciente en un momento determinado o facilitar la venta de acciones en bolsa obteniendo el mayor beneficio. (vida" y Pedagogía 2020)

Las aplicaciones del *IoT* son muy amplias: en el desarrollo de servicios e infraestructuras para una movilidad inteligente, las aplicaciones digitales para el ahorro energético y el control medioambiental o la atención remota a los mayores, donde la presencia física en el territorio es una ventaja competitiva, darán paso al desarrollo de nuevos productos y servicios. Específicamente en el sector de las *Smart Cities* y en el de la electricidad, el concepto del *IoT* es ya una realidad. (vida" y Pedagogía 2020)

La electricidad constituye la plataforma natural de desarrollo rápido de *ITE*, por disponer de unas infraestructuras ya instaladas con grandes posibilidades por el ahorro de costes que esto puede significar, dando soporte a numerosas aplicaciones del *IoT*. De hecho, la investigación en la tecnología sin cables está dando muy buenos resultados. Por ejemplo, en Los Ángeles se están colocando farolas con bombillas LED que iluminan y suministran wifi a los ciudadanos. En otras ciudades; las farolas también suministran energía para la recarga de los coches eléctricos. (vida" y Pedagogía 2020)

Asimismo, muchas aplicaciones de la domótica están recurriendo a utilizar los campos magnéticos para manejar sin cables los diferentes sistemas de control del domicilio. Por medio de la tecnología *Anywire* de transmisión de datos por red eléctrica, se conseguirá minimizar el cableado de los dispositivos integrados y las estanterías serán auto-organizativas, entre otras funcionalidades. (Gillis s. f.)

Por tanto, el *IoT* afectará a la gran mayoría de la población activa de los países desarrollados y emergentes. Ej.: la industria del automóvil ha iniciado la transformación digital de los vehículos, incorporando sensores que ofrecen nuevos servicios digitales relacionados con la seguridad o el confort. Siendo una realidad, la accesibilidad de los automóviles eléctricos y los vehículos sin conductor en los carriles. (Gillis s. f.)

Con marcada frecuencia se desencadenan innovaciones en la forma de fabricar, instalar y gestionar los nuevos productos y servicios, por lo que tendremos que cambiar radicalmente nuestras competencias profesionales. El trabajo se realiza cada vez más con aplicaciones en la nube, en movilidad, utilizando videoconferencia para trabajar en equipo y en cualquier parte utilizando herramientas como: el correo electrónico, web, marketing digital y explotando la infraestructura de Internet.

Durante los primeros 25 años de vida un joven habrá jugado 5.000 horas con juegos interactivos, habrá enviado 250.000 e-mails y mensajes de texto, habrá utilizado el móvil 10.000 horas, habrá pasado más horas en Internet y *Youtube* que viendo la televisión. Su educación habrá transcurrido entre pizarras interactivas, métodos de aprendizaje basados en la gamificación de las aulas, el *e-Learning* y completará su formación con varios *Mooc (Massive Online Open Courses* o Cursos *online* masivos y abiertos) que son la evolución desde las plataformas educativas cerradas a entornos de

aprendizaje abiertos desde una concepción conectivista, donde la creación del conocimiento se basa en el establecimiento de conexiones que ofrecen más posibilidades de aprendizaje. (Gillis s. f.)

El ocio discurre en gran parte en las redes sociales, viendo contenidos multimedia, animaciones, utilizando la Smart TV para hacer programaciones personalizadas, consolas, las *web*-series, video bajo demanda, *YouTube*, videojuegos con realidad virtual, realidad aumentada, o representación inmersiva en 3D.

En conclusión, el *IoT* será estratégico para la sociedad, los negocios y será fundamental en los procesos de toma de decisiones, otorgando más agilidad a las acciones, mejorando procedimientos y logrando empresas más innovadoras y productivas, pero también consiguiendo ahorrar costes y mejorar la seguridad. (Abbate s. f.)

Pero no todo será sencillo. El *IoT* es capaz de desarrollarse en tantos ámbitos y hallar tantas aplicaciones que puede encontrar en su propia diversidad el principal obstáculo para su crecimiento. (Pardo 2018)

Porque, en un entorno en el que operarán innumerables dispositivos de distinta naturaleza y perfil técnico (desde electrodomésticos a *wearables*, pasando por vehículos autónomos o drones, entre otros muchos), fabricados a su vez por miles de marcas diferentes (cada una con sus propios estándares), desarrollar la habilidad para que todos ellos se comuniquen entre sí no sólo será un desafío técnico, sino también una cuestión de consenso.

Es por ello que la interoperabilidad del *IoT* surge como una necesidad de primer orden para el desarrollo del *IoT*, constituyendo su valor central más básico; el primer requisito de la conectividad a Internet es que los sistemas “conectados” deben poder “hablar el mismo idioma” en cuanto a protocolos y codificaciones. (Pardo 2018)

Los profesionales de la tecnología, suelen utilizar los términos interoperabilidad e integración cuando hablan de los sistemas *IoT*. Su integración significa que las diferentes piezas necesarias para que los proyectos de *IoT* funcionen, como dispositivos, datos, plataformas y aplicaciones, se pueden unir y funcionar como un solo sistema.

La interoperabilidad de *IoT* lleva ese significado un paso más allá y define el ideal de la integración, en el que todas las partes de un sistema de *IoT* funcionan juntas sin problemas, sin que se le solicite ni se esfuerce. (Yacchirema s.f.c)

Sin interoperabilidad, los dispositivos y las aplicaciones no pueden comunicarse ni compartir datos, por lo que su costo y complejidad aumentan rápidamente. La falta de interoperabilidad de *IoT* puede limitar la adopción de tecnologías más complejas, incluidos los gemelos digitales o el aprendizaje automático. También puede conducir al bloqueo del proveedor, donde a las organizaciones les resultaría difícil adoptar dispositivos de *IoT* adicionales o transferir datos a través de plataformas o dominios. Las organizaciones que no pueden integrar nuevos dispositivos y servicios se verán limitadas en cuanto a la capacidad de escalar y adoptar un proyecto de IoT. (Rose et al. s. f.)

Cada uno de los dispositivos y plataformas de *IoT* viene con su propia arquitectura, formatos de datos, protocolos y API. Los proveedores y desarrolladores de *IoT* no tienen un conjunto específico de estándares o bloques de construcción que deban usar en sus dispositivos o servicios, aparte de las normas de seguridad básica. La gran cantidad de tecnología disponible ofrece a los desarrolladores de estos dispositivos muchas opciones y combinaciones para elegir, pero crea un mercado fragmentado, donde es posible que los dispositivos no funcionen juntos automáticamente. (Rose et al. s. f.)

Se hace evidente, que uno de los objetivos principales, debe ser garantizar la interoperabilidad, en todas las etapas, desde la recopilación de los datos, hasta su transmisión, almacenamiento y análisis. La falta de estandarización de los servicios en la nube, o la variedad en cuanto a protocolos máquina a máquina (M2M), sistemas operativos o firmware, son ejemplos de problemas en la interoperabilidad. Sus efectos son una menor flexibilidad, al quedar «atados» a proveedores o tecnologías muy concretas, y mayores costes. (vida" y Pedagogía 2021)

Por lo tanto, la interoperabilidad de dispositivos es uno de los principales desafíos a enfrentar en el ámbito de la investigación de IoT. En tal sentido la abstracción de la heterogeneidad *hardware* y *software* subyacentes de los dispositivos y la conversión de

protocolos para el intercambio de información entre los mismos presenta una estrategia clave. (vida" y Pedagogía 2021)

Tomando en consideración lo antes planteado y dando respuesta a la necesidad de la interoperabilidad entre los dispositivos nos planteamos la siguiente Situación Problémica:

Para el desarrollo de diversos proyectos e investigaciones, se utilizan diferentes tipos de nodos sensores, que debido a que poseen recursos computacionales limitados, se comunican utilizando protocolos de aplicación restringidos por redes WSN como CoAP o MQTT. Los fabricantes de los mismos, emplean protocolos de comunicación y arquitectura propios, trayendo como consecuencia que no exista uniformidad en la forma de presentación de los datos y problemas de interoperabilidad, afectando la obtención de un consenso en la presentación de datos a nivel de la aplicación.

Lo expuesto anteriormente nos conduce al siguiente Problema de Investigación:

¿Cómo lograr un consenso en la presentación de datos a nivel de la aplicación para nodos sensores?

Como forma de solución al problema anterior planteamos como:

Objetivo General de la investigación:

Implementar un *Thing Description* para nodos sensores que permita obtener un consenso en la presentación de datos a nivel de la aplicación.

Un análisis del objetivo general nos permite plantearnos las siguientes:

Preguntas de Investigación:

1. ¿Cuáles son los fundamentos teórico-metodológicos que sustentan la implementación de un *TD* para nodos sensores, que permita obtener un consenso en la presentación de datos a nivel de la aplicación?

2. ¿Cómo diseñar un *TD* para nodos sensores que permita obtener un consenso en la presentación de datos a nivel de la aplicación?

3. ¿Cómo implementar un *TD* para nodos sensores que permita obtener un consenso en la presentación de datos a nivel de la aplicación?

Para cumplir el objetivo propuesto y dar respuesta a las preguntas de investigación se proponen las siguientes Tareas de Investigación:

1. Determinar los fundamentos teórico-metodológicos que sustentan el desarrollo de un *TD* para nodos sensores, que permita obtener un consenso en la presentación de datos a nivel de la aplicación.

2. Diseñar un *TD* para nodos sensores, que permita obtener un consenso en la presentación de datos a nivel de la aplicación.

3. Implementar un *TD* para nodos sensores, que permita obtener un consenso en la presentación de datos a nivel de la aplicación.

Justificación y principales contribuciones de la investigación:

Mediante el desarrollo de esta investigación se logrará un consenso en la presentación de datos para nodos sensores. Con este objetivo cumplido se podrán crear redes basadas en los protocolos de la Web de las Cosas, permitiendo la comunicación entre los propios sensores y las personas, facilitando la obtención, procesamiento y análisis de la información. Esta investigación puede ser aplicada en multitud de campo: agricultura de precisión, en la industria. En fin, en cualquier tipo de sensores, evitando, grandes impedimentos económicos.

Estructura de la investigación:

Atendiendo a lo planteado anteriormente, la tesis queda estructurada en: introducción, tres capítulos, conclusiones, recomendaciones y referencias bibliográficas:

Introducción: se caracteriza la situación problemática y se fundamenta el problema científico a resolver. **Primer capítulo**: tendrá como objetivo establecer los fundamentos teórico-metodológicos que sustentarán la investigación. **Segundo capítulo**: se describe el desarrollo del software a través de la metodología RUP. **Tercer capítulo**: se muestran las principales interfaces del prototipo inicial y se detallan los resultados de las pruebas aplicadas al software. **Conclusiones**: se verifica el cumplimiento de los objetivos trazados al inicio de la investigación. **Recomendaciones**: en la cual se plasma las propuestas de la automatización de la aplicación. **Bibliografía**: referencia bibliográfica citada a través de las Normas APA.

CAPÍTULO 1: Fundamentación teórica y metodológica para la Implementación de un *Thing Description* orientado a Nodos sensores.

1.1. Introducción:

En este capítulo se desarrollarán los temas asociados a la fundamentación teórica sobre el objeto de estudio de la investigación y la metodología necesaria para el desarrollo del sistema propuesto. Se abordan las tecnologías actuales y las herramientas existentes que sirven de sustento para su estructuración.

1.2. Sensores.

Un sensor es un dispositivo que está capacitado para detectar acciones o estímulos externos y responder en consecuencia. Estos aparatos pueden transformar las magnitudes físicas o químicas en magnitudes eléctricas. Los sensores se pueden categorizar de múltiples maneras. Un enfoque común es clasificarlos como activos o pasivos o teniendo en cuenta el tipo de factores ambientales que monitorean. (Industrial 2021)

Otra forma de clasificar los sensores es si son analógicos o digitales, según el tipo de salida que producen. Los sensores digitales se han vuelto bastante comunes en todas las industrias, reemplazando a los sensores analógicos en muchas situaciones. Por ejemplo, los sensores digitales ahora se utilizan para medir la humedad, la temperatura, la presión atmosférica, la calidad del aire y muchos otros tipos de fenómenos ambientales. (Marketing 2022)

Ninguna de las categorías de los sensores es excluyentes; por ejemplo, un sensor de nivel que rastrea el nivel de un material también podría considerarse un sensor óptico o de presión. Los sensores se han vuelto tan comunes, que a menudo su uso apenas se nota. (Marketing 2022)

Varios sensores pueden ser combinados dentro de un sistema de sensores que permita el control de estos, mediante una interfaz de control, además de cada uno de ellos por separado. Estos sistemas se caracterizan por tener una salida e interfaz de control que refleja su unidad organizacional, la que puede variar en tipo y naturaleza. Las partes

individuales pueden proveer interfaces, aun cuando formen parte de un sistema de sensores.

1.2.1. Sensores inteligentes.

Un sensor inteligente es un dispositivo que toma datos del entorno físico y utiliza recursos informáticos integrados para realizar funciones predefinidas al detectar una entrada específica y luego procesar los datos antes de transmitirlos. (SensorGO 2021)

Los sensores inteligentes permiten una recopilación más precisa y automatizada de datos ambientales con menos ruido erróneo entre la información registrada con precisión. Estos dispositivos se utilizan para monitorear y controlar los mecanismos en una amplia variedad de entornos que incluyen redes inteligentes, reconocimiento del campo de batalla, exploración y una gran cantidad de aplicaciones científicas. El sensor inteligente también es un elemento crucial e integral en *IoT*.

Su implementación es como componentes de una red inalámbrica de sensores y actuadores (*WSAN*) cuyos nodos pueden contarse por miles, cada uno de los cuales está conectado con uno o más sensores y concentradores de sensores, así como con actuadores individuales.

Los recursos informáticos suelen ser proporcionados por microprocesadores móviles de baja potencia. Como mínimo, un sensor inteligente está hecho de un sensor, un microprocesador y tecnología de comunicación de algún tipo. Los recursos informáticos deben ser una parte integral del diseño físico: un sensor que simplemente envía sus datos para su procesamiento remoto no se considera un sensor inteligente. (SensorGO 2021)

Un sensor inteligente también puede incluir otros componentes además del sensor primario. Ej.: transductores, amplificadores, control de excitación, filtros analógicos y compensación e incorpora elementos definidos por software que proporcionan funciones como conversión de datos, procesamiento digital y comunicación a dispositivos externos. (SensorGO 2021)

1.3. Internet de las cosas

Por lo general, el término *IoT* se refiere a escenarios en los que la conectividad de red y la capacidad de cómputo se extienden a objetos, sensores y artículos de uso diario que habitualmente no se consideran computadoras, permitiendo que estos dispositivos generen, intercambien y consuman datos con una mínima intervención humana. Sin embargo, no existe ninguna definición única y universal. (Rose et al. s. f.)

El *IoT* puede ser visto como una combinación de sensores y actuadores que son capaces de proporcionar y recibir información digitalizada y colocarla en redes bidireccionales capaces de transmitir todos los datos para ser utilizados por una gran cantidad de diferentes servicios y usuarios finales. (vida" y Pedagogía 2020)

Múltiples sensores se pueden unir a un objeto o dispositivo para medir una amplia gama de variables físicas o fenómenos y luego transmitir todos los datos a la nube.

Características de IoT (Joskowicz José 2022)

- Objetos/*things*/dispositivos: Un componente imperativo de *IoT* son los dispositivos denominados objetos o simplemente "cosas"
- Conectividad: es la presencia de una infraestructura de red o conectividad de red, que habilite la comunicación de los dispositivos *IoT*, su integración perfecta y un esquema de direccionamiento único.
- Datos: El verdadero valor de *IoT* radica en los datos asociados a cada dispositivo que se obtienen a partir de la interconexión entre ellos.
- Autonomía: Gracias a esta característica los dispositivos *IoT* deben tener la capacidad de realizar la mayoría de sus tareas sin la intervención humana o de otros dispositivos, y un grado de control sobre sus propias acciones y su propio estado interno según el contexto, las condiciones cambiantes o el entorno detectado
- Servicios: Los servicios que pueden ser complejos o simples deben estar disponibles para interactuar con los dispositivos *IoT*, consultar su estado y cualquier información asociada a ellos, teniendo en cuenta las restricciones, como protección de la privacidad y coherencia semántica entre los dispositivos *IoT* y sus correspondientes objetos virtuales.

- Heterogeneidad: La heterogeneidad de los dispositivos *IoT* plantea problemas de interoperabilidad y es otra característica a menudo enfatizada y de primordial importancia a considerar para el desarrollo y adopción de esta tecnología.

IoT aborda una gran diversidad de dispositivos, generalmente con capacidades de comunicación y computación muy básicas, que desafía la suposición de que cualquier dispositivo presenta una pila de protocolos completa.

Componentes de *IoT (Pardo 2018)*

- Identificación: La capacidad de identificar de forma única los dispositivos es fundamental para el éxito de *IoT*. El identificador del dispositivo hace referencia a su nombre, mientras la dirección se refiere a la dirección única del dispositivo dentro de una red de comunicaciones.
- Detección/actuación: La detección de *IoT* consiste en recopilar datos de dispositivos relacionados dentro de la red. Los datos recopilados se envían a un almacén de datos, base de datos o *al cloud* para su almacenamiento y posterior procesamiento de manera que permitan tomar acciones específicas basadas en los servicios requeridos.
- Comunicación: La comunicación permite que los dispositivos puedan enviar y recibir información. Generalmente los dispositivos *IoT* operan con un bajo consumo de energía y en entornos de enlaces de comunicación ruidosos y con altos niveles de pérdidas de información.
- Servicios: Los servicios de *IoT* pueden clasificarse en cuatro clases: servicios relacionados con la identidad, servicios de agregación de la información, servicios basados en la colaboración y servicios ubicuos. Para cada aplicación un servicio *IoT* puede ser aplicado.
- Semántica: La semántica en *IoT* se refiere a la capacidad de extraer conocimiento de los datos a través del uso de recursos e información de modelado para dar sentido al servicio solicitado. Este componente es compatible con las tecnologías de la Web semántica, como el *Resource Description Framework* (RDF), la ontología *Web Ontology Language* (OWL) y el formato *Efficient* XML *Interchange* (EXI) (W3C, 2014).

Desafíos en el desarrollo de IoT

IoT puede ofrecer enormes beneficios económicos a través de la implementación de diferentes servicios y aplicaciones enfocados a varios aspectos de nuestra vida, pero también enfrenta muchos desafíos técnicos clave. A continuación, se describen brevemente algunos de ellos. ***(Tavizon, Guajardo, y Laines Alamina 2016)***

- Conectividad: La conectividad perfecta consumirá extremadamente poca energía, tendrá una amplia cobertura y será capaz de transmitir grandes cantidades de datos. Esta conectividad perfecta no existe. La heterogeneidad de los dispositivos involucrados lo hace aún más difícil, ya que se pueden esperar muchas interconexiones físicas diferentes. El desafío de *IoT* es abordar esta heterogeneidad en la fase de diseño identificando los posibles casos de uso en cada sector.
- Arquitectura del sistema: La naturaleza de múltiples dominios de aplicación de *IoT* dificulta la creación de una arquitectura y una aplicación única y excepcional. En pocas palabras, las soluciones para un determinado dominio no son aplicables a otros, ya sea debido a requisitos funcionales o debido a diferencias de hardware. El desafío en *IoT* es construir tales arquitecturas, teniendo en cuenta la portabilidad, la integración y la conectividad. Además, las arquitecturas deben ser abiertas y, utilizar tecnologías estándar, a fin de no restringir a los usuarios para que utilicen soluciones propietarias de extremo a extremo.
- Interoperabilidad e integración: *IoT* se caracteriza por una gran heterogeneidad en cuanto a los dispositivos que participan en un sistema, que presentan capacidades muy diferentes en términos de los componentes de comunicación y computación; debido a que están construidos por diferentes fabricantes. Su integración perfecta solo puede ser posible si se construyen sobre estándares abiertos. Puede haber múltiples estándares para las mismas áreas (p.ej. diferentes estándares de redes inalámbricas), pero se debe establecer la interoperabilidad entre ellos (p.ej., a través de *gateways* entre diferentes redes físicas)
- Complejidad computacional y de almacenamiento: Los dispositivos *IoT* generarán cantidades masivas de datos. Su desafío es desarrollar y mantener infraestructuras complejas.

- Seguridad, confianza y privacidad: La penetración de *IoT* en la vida diaria enfatiza la necesidad de soluciones seguras adecuadas. Si bien, ha dado lugar a importantes avances tecnológicos, la confianza y aceptación dependerán del equilibrio adecuado entre promover esta innovación y garantizar la seguridad y privacidad de los usuarios.

De los desafíos enumerados, es obvio que el desarrollo del *IoT* depende del progreso de varias disciplinas, incluidas las redes inalámbricas de sensores, la computación en la nube, computación en la niebla y la seguridad informática. En esta investigación abordamos el desafío de la interoperabilidad entre dispositivos *IoT*.

Interoperabilidad de dispositivos en IoT

La interoperabilidad de dispositivos se puede definir como la capacidad de dos o más dispositivos para conectarse, intercambiar información e interactuar entre sí. La misma es un aspecto clave en el desarrollo de sistemas *IoT* en el que se pretende que interactúen sin fisuras diferentes dispositivos heterogéneos. Si además nos centramos en el caso particular, y en el objetivo de implementar la interacción entre dispositivos y plataformas *IoT*, el reto es aún mayor ya que no es una tarea fácil debido a la heterogeneidad inherente en las capacidades de memoria, procesamiento, comunicación y formato de datos admitido por cada dispositivo. (Pardo 2018) Se comentará con detalle en la siguiente sección.

Heterogeneidad de los dispositivos

IoT está compuesto por una variedad de dispositivos, incluso más que el internet tradicional. En los ecosistemas actuales de *IoT*, los distintos dispositivos y aplicaciones operan en sus propias plataformas sin la compatibilidad adecuada con dispositivos de diferentes proveedores. Por ejemplo, una pulsera inteligente no puede interactuar con una bombilla inteligente sin la aplicación privada relevante proporcionada por el mismo proveedor. Como resultado, se establecen islas de funcionalidad *IoT* que conducen a una intranet de dispositivos en lugar de un internet de dispositivos. Estos problemas se presentan debido a la gran variedad y heterogeneidad de los dispositivos. Los dispositivos *IoT* son heterogéneos al estar basados en hardware muy variado de

plataformas y redes que se traducen en distintas capacidades de memoria, procesamiento, comunicación y formato de datos. (Tavizon et al. 2016)

- Dispositivos Clase 0: Generalmente son etiquetas RFID, sensores de bajo costo y actuadores que poseen restricciones severas, tanto en memoria como en las capacidades de procesamiento. Estas restricciones impiden que se comuniquen directamente a Internet de una manera segura. Sin embargo, pueden comunicarse a Internet con la ayuda de dispositivos con mayor capacidad (p. ej. dispositivos de Clase 2) que actúan como servidores proxy, *gateways* o servidores. Los dispositivos clase 0 son configurados con un conjunto de datos muy pequeño para su funcionamiento.
- Dispositivos Clase 1: Están bastante limitados en el espacio de código y en las capacidades de procesamiento. No pueden comunicarse fácilmente con otros nodos de Internet que utilizan una pila de protocolos completa como HTTP, *Transport Layer Security* (TLS) y protocolos de seguridad relacionados y basados en representaciones de datos como XML (*Extensible Markup Language*). Sin embargo, pueden usar una pila de protocolos diseñada específicamente para nodos restringidos como el *Constrained Application Protocol* (CoAP) y participar en conversaciones significativas sin la ayuda de un nodo intermediario. Por lo general, estos protocolos son livianos, energéticamente eficientes y consumen menor ancho de banda en comparación con los protocolos tradicionales de Internet. Además, estos protocolos pueden proporcionar soporte para la seguridad requerida en una red. Por lo tanto, los dispositivos Clase 1 pueden integrarse en una red IP. No obstante, deben ahorrar memoria, espacio de código y, en muchos casos, también energía.
- Dispositivos Clase 2: Tienen más recursos, pero aún están muy limitados en comparación con los dispositivos tradicionales. Estos dispositivos se caracterizan por tener recursos “suficientes” para ejecutar software basado en sistemas operativos tradicionales (OS), como Linux, o BSD. Estos dispositivos incluyen computadoras de una sola placa (SBC, por sus siglas en inglés, *single-board computer).*

Capacidades de comunicación: Actualmente los dispositivos pueden utilizar varias soluciones de comunicación desde comunicaciones ubicuas tradicionales (p. ej., comunicaciones celulares 3/4/ 5G y las tecnologías *Wi-Fi*) hasta soluciones propietarias no estándar (p.ej., *Sigfox, LoRa*), pasando por tecnologías comerciales (p. ej., *Bluetooth*, NFC, ANT +) así como protocolos y mecanismos de comunicación tradicionales para sensores y actuadores que están abriendo nuevas posibilidades para *IoT* (p.ej., *ZigBee*, *WirelessHart*, IEEE802.11 ah, *ZigBee*, etc.). Adicionalmente, los dispositivos también pueden utilizar diferentes soluciones para el enrutamiento (p.ej. RPL, CORPL, etc.), y encapsulación (p. ej., 6LowPAN, 6Lo, *Thread*, etc.). Desde una perspectiva del *hardware*, el estado tecnológico de la técnica también es muy heterogéneo, debido a que se han desarrollado varias soluciones comerciales (p.ej., *Arduino, TelosB, pcDuino, Libelium waspmote*, etc.) (Gillis s. f.)

Formato de datos: El mundo de los datos provenientes de los dispositivos y su semántica también presentan una gran heterogeneidad debido a la presencia de diferentes formatos (p.ej., *WSDL, JSON, XML, W3C Semantic Sensor Network XG, SWE, ASN*.12, etc.) (Tavizon et al. 2016)

Niveles de Interoperabilidad

El reto de la interoperabilidad entre dispositivos *IoT* se puede abordar a 3 niveles o capas; interoperabilidad técnica, sintáctica y semántica.(Åkerman 2016)

- Interoperabilidad técnica: A menudo se centra en los protocolos de comunicación y en la infraestructura necesaria (componentes de *hardware* y *software*) para permitir la conectividad de dispositivos heterogéneos; sin conectividad no hay comunicación. Aquí los dispositivos solo pueden intercambiar mensajes sin interpretar su contenido interno.
- Interoperabilidad sintáctica: (Yacchirema s. f. c) se asocia con los formatos de los datos de los mensajes de los dispositivos, ya que estos deben tener una sintaxis y codificación bien definida. Aquí los dispositivos pueden interpretar el contenido de los datos dentro de la estructura del mensaje. Para evitar cualquier posible ambigüedad durante la interpretación de los datos, es necesario utilizar formatos estandarizados.

- Interoperabilidad semántica: en este nivel, los dispositivos involucrados son capaces de interpretar el contenido y el significado de los datos intercambiados. Las ontologías y las tecnologías semánticas son medios para facilitar la interoperabilidad semántica. La falta de semántica explícita no ha sido un problema en los sistemas de comunicación tradicionales, como los teléfonos y las computadoras en general, porque la interoperabilidad a nivel semántico ha sido resuelta por usuarios humanos que se comunican entre sí mediante el uso de dispositivos. Sin embargo, en *IoT*, los dispositivos también se convierten en usuarios y, por lo tanto, necesitan comunicarse directamente entre sí e interpretar el significado de la información en tiempo de ejecución.

1.4. *Web* de las Cosas

La *Web* de las cosas (*WoT*) busca contrarrestar la fragmentación de la *IoT* mediante el uso y la ampliación de las tecnologías web estandarizadas existentes. Al proporcionar metadatos estandarizados y otros componentes tecnológicos reutilizables, *W3C WoT* permite una fácil integración entre las plataformas de *IoT* y los dominios de aplicaciones. (Tavizon, Guajardo, y Laines Alamina 2016)

Por lo general, en los proyectos de *IoT* clásicos, los desarrolladores deben enfrentar una situación desafiante. Deben comprender un panorama tecnológico heterogéneo que consta de diversos sistemas y servicios de *IoT* de diferentes proveedores y fabricantes. Esta diversidad incluye variaciones en los protocolos de comunicación, modelos de datos para el intercambio de datos de carga útil y requisitos de seguridad. Las aplicaciones de *IoT* generalmente se desarrollan con un gran esfuerzo aplicado a un caso de uso estrecho y específico. Durante su vida útil, dichas aplicaciones son difíciles de ampliar, mantener o reutilizar. (elsuper.com 2019)

La *WoT* proporciona un conjunto de bloques de construcción de tecnología estandarizados que ayudan a simplificar el desarrollo de aplicaciones de *IoT* siguiendo el conocido y exitoso paradigma *Web*. Este enfoque aumenta la flexibilidad y la interoperabilidad, especialmente para aplicaciones entre dominios, además de permitir la reutilización de estándares y herramientas establecidos. *WoT* desbloquea el potencial comercial que se ve frenado por la fragmentación de *IoT*. (Yacchirema s. f.-c)

WoT introduce una abstracción de interacción simple basada en propiedades, eventos y acciones. Cualquier interfaz de red *IoT* se puede describir en términos de esta abstracción. Al usar esta abstracción, las aplicaciones tienen un ancla común para recuperar los metadatos de un servicio de *IoT*, así como una forma de comprender a qué y cómo se puede acceder a los datos y las funciones de los servicios de *IoT*.

Se definen cuatro capas de la *WoT*: Acceso, búsqueda, compartir y componer. (Tavizon et al. 2016)

- La capa de acceso: Si queremos que una Internet Thing se convierta en una Web Thing necesitamos que sea accesible y que a su vez pueda interactuar con otros dispositivos. El protocolo HTTPS (Hypertext Transfer Protocol Secure) y las APIs, son los actores principales de esta capa. Las APIs permiten la interacción desde y con otros sistemas. Y se comunican utilizando el protocolo HTTPS, principalmente.
- La capa de descubrimiento: Discovery es un término que se utiliza mucho en la web moderna y que describe mejor el propósito que Find, buscar. En este mismo sentido los dispositivos que se comunican en la *WoT* deben hacerlo de forma entendible y esto es lo que permite la capa de descubrimiento. Las Cosas Web serían esencialmente una API que puede ser descubierta, entendida, por otras *WoT*, servicios, sistemas o elementos del mundo real, como los propios seres humanos.
- Capa de comunicación: En esta capa la Cosa Web utilizará los protocolos que permiten enviar y recibir información a través de Internet, además de forma segura si fuera preciso. Para lograrlo utilizará protocolos como TLS (Transport Layer Security) u otros.
- La capa de creación: Finalmente, para que se produzca la comunicación dentro de la *WoT*, entre esta y otras *WoT* o los sistemas, y al mismo tiempo con los humanos, serán necesarias diferentes tipos de aplicaciones e interfaces para interactuar con ellas por los medios necesarios.
- Estamos hablando de actores y dispositivos muy diferentes, que pueden usar diferentes protocolos y formatos de información. Toda esa información

incompatible y modos heterogéneos deben encapsularse en aplicaciones que faciliten la interacción. En esta capa estarían los frameworks o librerías que permiten crear las aplicaciones para interactuar con la Web Thing.

Una cosa es la abstracción de una entidad física o virtual (por ejemplo, un dispositivo o una habitación) y se describe mediante metadatos estandarizados. En W3C *WoT*, los metadatos de descripción deben ser un *Thing Description WoT*. Los consumidores deben poder analizar y procesar el formato de representación *TD*, que se basa en *JSON*. El formato se puede procesar a través de bibliotecas *JSON* clásicas o un procesador *JSON-LD*, ya que el modelo de información subyacente está basado en gráficos y su serialización es compatible con *JSON-LD* 1.1. El uso de un procesador *JSON-LD* para el procesamiento de un *TD* permite adicionalmente el procesamiento semántico incluyendo la transformación a triples *RDF*, la inferencia semántica y la realización de tareas dadas en base a términos ontológicos, lo que haría que los Consumidores se comportaran de manera más autónoma. Un *TD* es específico de la instancia (es decir, describe una Cosa individual, no tipos de Cosas) y es la representación textual (*web*) externa predeterminada de una Cosa. Puede haber otras representaciones de una cosa, como una interfaz de usuario basada en *HTML*, simplemente una imagen de la entidad física, o incluso representaciones no web en sistemas cerrados. (Yacchirema s. f.-c)

1.4.1. Thing Description

El Thing Description de WoT es un formato de representación estandarizado y comprensible por máquina que permite a los Consumidores descubrir e interpretar las capacidades de una Cosa (a través de anotaciones semánticas) y adaptarse a diferentes implementaciones (por ejemplo, diferentes protocolos o estructuras de datos) cuando interactúan con una Cosa, lo que permite la interoperabilidad entre diferentes plataformas de *IoT*, es decir, diferentes ecosistemas y estándares. (Hassine, Korkan, y Steinhorst s. f.)

La especificación *WoT Thing Description* define un modelo de información basado en un vocabulario semántico y una representación serializada basada en *JSON*. Los *TD* proporcionan metadatos enriquecidos para *Things* de una manera que es legible por

humanos y comprensible por máquina. Tanto el modelo de información como el formato de representación de los *TD* están alineados con los datos vinculados, por lo que, además del procesamiento *JSON* sin procesar, las implementaciones pueden optar por utilizar *JSON-LD* y bases de datos gráficas para permitir una semántica poderosa y el procesamiento de los metadatos. (Hassine et al. s. f.)

Una descripción de cosa (*TD*) describe instancias de cosa con metadatos generales, como nombre, ID, descripciones, y también puede proporcionar metadatos de relación a través de enlaces a cosas relacionadas u otros documentos. Los *TD* también contienen metadatos de *Interaction Affordance* basados en el modelo de interacción y metadatos de comunicaciones que definen enlaces de protocolo. (*Thing descriptio*n s. f.)

Idealmente, el *TD* es creado y/o alojado por la propia Cosa y recuperado al ser descubierto. Sin embargo, también se puede alojar externamente cuando una cosa tiene restricciones de recursos (por ejemplo, espacio de memoria limitado, energía limitada) o cuando un dispositivo existente se adapta para convertirse en parte de la *WoT*. Un patrón común para mejorar el descubrimiento (p. ej., para dispositivos restringidos) y para facilitar la administración de dispositivos es registrar los *TD* en un directorio.

Se recomienda que los Consumidores utilicen un mecanismo de almacenamiento en caché de *TD* combinado con un mecanismo de notificación, que les informará cuando sea necesario obtener una nueva versión del *TD*, en caso de que se actualice la Cosa.

Para la interoperabilidad semántica, los *TD* pueden hacer uso de un vocabulario específico del dominio, para el cual se proporcionan puntos de extensión explícitos. (*Thing description* s. f.)

El objetivo principal del *Thing Description* es proporcionar una descripción de interfaz legible por humanos e interpretable por máquina de un dispositivo/cosa *IoT*, no se limita a un protocolo de comunicación específico, sino que proporciona un marco denominado plantilla de vinculación de *WoT*. Dicho enlace de protocolo define el mapeo de un rendimiento de interacción a mensajes concretos de un protocolo IoT específico como *MQTT, HTTP, CoAP, Modbus* u OPC UA.

Los bloques de construcción de *WoT* proporcionan una forma de implementar sistemas que se ajustan a la arquitectura de *WoT*. El componente clave de los bloques de construcción de *WoT* es el *Thing Description*. (Hassine et al. s. f.)

- ✓ Un *Thing Description* describe un dispositivo virtual o físico (cosa).
- ✓ Se puede considerar como el principal punto de entrada para una cosa, como una página index.html para un sitio web.
- ✓ Los *TD* fomentan la interoperabilidad proporcionando metadatos legibles (y comprensibles) tanto humanos como mecánicos sobre una cosa, como el título, la identificación, las descripciones, etc.
- ✓ Un *Thing Description* también describe todas las acciones, eventos y propiedades disponibles de una cosa, así como toda la seguridad disponible y los mecanismos para acceder a ellos.
- ✓ *TD* en muy flexible para garantizar la interoperabilidad.
- ✓ *TD* es un modelo de información de una cosa basado en vocabulario semántico, abierta libre de regalías con un formato de representación basado en JSON para *IoT*.

Un *TD* proporciona una forma unificada de describir las capacidades de un dispositivo o servicio de *IoT* con su modelo de datos y funciones ofrecidos, el uso del protocolo y otros metadatos. El uso de descripciones de cosas ayuda a reducir la complejidad de integrar dispositivos *IoT* y sus capacidades en aplicaciones *IoT*. (Consortium, 2022)

El modelo de interacción de W3C *WoT* define tres tipos de prestaciones de interacción: las propiedades (*clase PropertyAffordance*) se pueden utilizar para detectar y controlar parámetros, como obtener el valor actual o establecer un estado de operación. Las acciones (*clase ActionAffordance*) modelan la invocación de procesos físicos (y, por lo tanto, consumen mucho tiempo), pero también se pueden usar para abstraer llamadas similares a RPC de plataformas existentes. Los eventos (*clase EventAffordance*) se utilizan para el modelo de comunicación *push* en el que las notificaciones, los eventos discretos o los flujos de valores se envían de forma asíncrona al receptor.(Charpenay, Kabisch, y Kosch s. f.)

Para el desarrollo de la aplicación, es clave la correcta selección de las tecnologías. Pues de ella depende en gran medida el éxito del proyecto. En este apartado se describen las herramientas y tecnologías usadas para la ejecución y desarrollo del proyecto.

1.5. Metodologías de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de técnicas y métodos organizativos que se aplican para diseñar soluciones de software informático. El objetivo de las distintas metodologías es el de intentar organizar los equipos de trabajo para que estos desarrollen las funciones de un programa de la mejor manera posible. Cuando se trata de desarrollar productos o soluciones para un cliente o mercado concreto, es necesario tener en cuenta factores como los costes, la planificación, la dificultad, el equipo de trabajo disponible, los lenguajes utilizados, etc. Todos ellos se engloban en una metodología de desarrollo que permite organizar el trabajo de la forma más ordenada posible. (iomkt.domainlogic@gmail.com 2022)

El presente trabajo utiliza como metodología de desarrollo de software RUP la cual permite reducir el nivel de dificultad, organizar las tareas, agilizar el proceso y mejorar el resultado final de las aplicaciones a desarrollar.

1.5.1 Metodología RUP

***El Rational Unified Process o Proceso Unificado de Racional*:** es un proceso de ingeniería de software que suministra un enfoque para asignar tareas y responsabilidades dentro de una organización de desarrollo.

Su objetivo es asegurar la producción de software de alta y de mayor calidad para satisfacer las necesidades de los usuarios que tienen un cumplimiento al final dentro de un límite de tiempo y presupuesto previsible. (Paredes et al. 2019)

Es una metodología de desarrollo iterativo que es enfocada hacia “diagramas de los casos de uso, y manejo de los riesgos y el manejo de la arquitectura” como tal.

El *RUP* mejora la productividad del equipo ya que permite que cada miembro del grupo sin importar su responsabilidad específica pueda acceder a la misma base de datos incluyendo sus conocimientos. Esto hace que todos compartan el mismo lenguaje, la

misma visión y el mismo proceso acerca de cómo desarrollar un software. (Paredes et al. 2019)

El *RUP* está basado en 6 principios clave que son los siguientes:

1. Adaptar el proceso
2. Equilibrar prioridades
3. Demostrar valor iterativamente
4. Elevar el nivel de abstracción
5. Enfocarse en la calidad.

1.5.2 Lenguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. UML tiene aplicaciones más allá del desarrollo de software, p. ej., en el flujo de procesos en la fabricación. (Mujica-Sequera 2022)

Es comparable a los planos usados en otros campos y consiste en diferentes tipos de diagramas. En general, los diagramas UML describen los límites, la estructura y el comportamiento del sistema y los objetos que contiene.

UML no es un lenguaje de programación, pero existen herramientas que se pueden usar para generar código en diversos lenguajes usando los diagramas UML, guarda una relación directa con el análisis y el diseño orientados a objetos y permite planificar y documentar cómo se construyen los programas informáticos, ayudando así al proceso de gestión de la construcción del software. (Mujica-Sequera 2022)

1.6. Framework de Desarrollo

En el desarrollo de *software*, un entorno de trabajo es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de *software*, que puede servir de base para la organización y su desarrollo. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras

herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. (define et al. s. f.)

Representa una arquitectura de *software* que modela las relaciones generales de las entidades del dominio, y provee una estructura y una especial metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio.

Algunos entornos de trabajo conocidos son *Spring Framework o Hibernate*, donde lo esencial para ser denominados entornos de trabajo es estar constituidos por objetos casi estáticos con funcionalidad definida a nivel grupo de objetos y no como parte constitutiva de estos, por ejemplo, en sus métodos, en cuyo caso se habla de una API o librería.

En la presente investigación se escoge *Django* como *Framework*

1.6.1 Django como Framework

Los orígenes de Django en la administración de páginas de noticias son evidentes en su diseño, ya que proporciona una serie de características que facilitan el desarrollo rápido de páginas orientadas a contenidos. (MDN s. f.)

Django proporciona una aplicación incorporada para administrar los contenidos, que puede incluirse como parte de cualquier página hecha con *Django* y que puede administrar varias páginas a partir de una misma instalación; la aplicación administrativa permite la creación, actualización y eliminación de objetos de contenido, llevando un registro de todas las acciones realizadas sobre cada uno, y proporciona una interfaz para administrar los usuarios y los grupos de usuarios (incluyendo una asignación detallada de permisos).

La distribución principal de Django también aglutina aplicaciones que proporcionan un sistema de comentarios, herramientas para sindicar contenido vía *RSS* y/o Atom, "páginas planas" que permiten gestionar páginas de contenido sin necesidad de escribir controladores o vistas para esas páginas, y un sistema de redirección de URLs. (W3Schools s. f.)

Otras características de *Django* son:

- Un mapeador objeto-relacional.

- Aplicaciones "enchufables" que pueden instalarse en cualquier página gestionada con Django.
- Una API de base de datos robusta.
- Un sistema incorporado de "vistas genéricas" que ahorra tener que escribir la lógica de ciertas tareas comunes.
- Un sistema extensible de plantillas basado en etiquetas, con herencia de plantillas.
- Un despachador de URLs basado en expresiones regulares.
- Un sistema "middleware" para desarrollar características adicionales; por ejemplo, la distribución principal de Django incluye componentes middleware que proporcionan cacheo, compresión de la salida, normalización de URLs, protección CSRF y soporte de sesiones.
- Soporte de internacionalización, incluyendo traducciones incorporadas de la interfaz de administración.
- Documentación incorporada accesible a través de la aplicación administrativa (incluyendo documentación generada automáticamente de los modelos y las bibliotecas de plantillas añadidas por las aplicaciones).

1.6.2 Python como lenguaje de Programación

Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código, se utiliza para desarrollar aplicaciones de todo tipo, ejemplos: *Instagram, Netflix, Spotify*, Panda 3D, entre otros.

Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma. (W3Schools.w s. f.)

Administrado por *Python Software Foundation*, posee una licencia de código abierto, denominada *Python Software Foundation License*. *Python* se clasifica constantemente como uno de los lenguajes de programación más populares.

Características y paradigmas (Santander.P s. f.)

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite

varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones. *Python* usa tipado dinámico y conteo de referencias para la gestión de memoria.

Una característica importante de *Python* es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. *Python* puede incluirse en aplicaciones que necesitan una interfaz programable. Aunque la programación en *Python* podría considerarse en algunas situaciones hostiles a la programación funcional tradicional del *Lisp*, existen bastantes analogías entre *Python* y los lenguajes minimalistas de la familia *Lisp* como puede ser Scheme.

Una característica importante de *Python* es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Es, además, fácil de aprender, tiene una sintaxis limpia y cuenta con una gran comunidad en línea que lo apoya. *Python* se convierte en una excelente opción cuando se trata de IoT. Podemos usarlo para el desarrollo de *backend* o el desarrollo de *software* de dispositivos. Además, está disponible para funcionar en dispositivos Linux y podemos utilizar *MicroPython* para microcontroladores.

Python es el lenguaje de codificación que podemos usar para reducir el volumen de datos con los que debemos tratar, accesible en la nube. Reconoce las necesidades independientemente de si creamos el proyecto *IoT* desde cero o interactuamos con actuadores, sensores y accesorios.

Algunos de los muchos beneficios de trabajar con *Python* para dispositivos *IoT* son la gran cantidad de bibliotecas para todo tipo de plataformas y la velocidad que ofrece para desarrollar el código.

Python es un gran aliado para desarrollar prototipos de dispositivos. Incluso si reescribimos algunos de los scripts mientras producimos en C, C++ o Java para mejorar el rendimiento.

1.7. Node.js

Node.js es un entorno de tiempo de ejecución de *JavaScript* de código abierto y multiplataforma, que se utiliza para crear aplicaciones de red del lado del cliente y del servidor. Los desarrolladores lo consideran mayoritariamente como uno de los principales marcos de *JavaScript* para el desarrollo *web*. Proporciona optimizadores, intérpretes y compiladores integrados y se ejecuta en el potente motor V8 de Google. (Node.js s. f.)

NodeJS tiene la reputación de crear aplicaciones escalables y de alto rendimiento con su rica pila tecnológica y su ecosistema robusto. También elimina el dilema entre el desarrollo del lado del cliente y del lado del servidor, ya que *Node* es eficientemente útil para ambos. Por lo tanto, algunos de los marcos de JavaScript más rápidos, como *TezJS*, usan *Node*.js para la configuración del desarrollo y para habilitar la administración de variables de entorno amigables con el tipo tanto para el lado del cliente como para el lado del servidor. (Radix 2021)

Una de las principales razones detrás de la popularidad de *Node*.js es el hecho de que es una herramienta de JavaScript. Esto le permite diseñar tanto el desarrollo *frontend* como el *backend* con el mismo lenguaje de programación. Entonces, resulta ser bastante eficiente en términos de recursos. Tiene funcionalidad multiplataforma y puede usar la misma aplicación de escritorio en *Mac, SO Linux y Windows*.

- *Nodejs* tiene una amplia gama de casos de uso: aplicación web, aplicación móvil e incluso soluciones en la nube e *IoT*.

Node.js tiene un umbral de entrada bajo, lo que significa que hay muchos desarrolladores incompetentes. Las capacidades de subprocesos múltiples de *Node*JS ayudan con un alto rendimiento, pero dado que procesa varias solicitudes simultáneamente, las excepciones de tiempo de ejecución ocurren con frecuencia y los errores pueden ser difíciles de manejar.

1.7.1 Nuxt.js:

Nuxt.js es el *framework Vue* más intuitivo disponible en la actualidad. Combina la potencia de *Vue.js* con las características de renderizado del lado del servidor para hacerlo más potente. Puedes construir una aplicación completa de renderizado del lado del cliente de *Vue*.js, una aplicación completa generada estáticamente y una aplicación monolítica. (kinsta s. f.)

Nuxt.js resuelve el problema de la estructuración de tu proyecto Vue.js, ya que viene con una arquitectura de desarrollo *frontend* preparada para la empresa. Las características de *Nuxt*.js ya están estructuradas utilizando los estándares de la industria para crear aplicaciones empresariales.

Nuxt.js permite crear muchos tipos de aplicaciones diferentes. (kinsta s. f.) Páginas Generadas Estáticamente, aplicaciones de una sola página (SPA) y aplicaciones Universales.

Nuxt.js resuelve el problema del *SSR* en *Vue.js*, lo que es útil para la optimización de los motores de búsqueda (SEO). Nuxt.js puede incluso ampliar las aplicaciones universales para dar cabida a una aplicación monolítica completa, en la que el *frontend* y el *backend* comparten una única base de código.

Nuxt.js funciona de la misma manera que un *framework* del lado del servidor cuando un usuario visita un sitio web. Si la renderización del lado del servidor está activada, las peticiones se renderizan en el servidor cada vez que el usuario solicita una página, por lo que se necesita un servidor para poder servir la página en cada petición. Además, si se habilita la renderización del lado del cliente, se renderiza el contenido de la página en el navegador utilizando *JavaScript*.

Acciones y métodos utilizados en *Nuxt.js*

- nuxtServerInit (Acción): Es el primer hook del ciclo de vida que se llama en el lado del servidor si el almacén *Vuex* está activado. Es una acción *Vuex* que se llama sólo en el lado del servidor para pre-llenar el almacén y, finalmente, puede utilizarse para enviar otras acciones en el almacén *Vuex*.

- Validate() (Función): El método validate se llama antes de renderizar los componentes de la página. Es útil para validar los parámetros dinámicos de un componente de página.
- El método asyncData() se utiliza para obtener datos y renderizarlos en el lado del servidor, mientras que el método fetch() se utiliza para llenar el almacén antes de renderizar la página.

Ventajas (kinsta s. f.):

- ✓ *Nuxt.js* hace que crear aplicaciones de renderizado del lado del servidor sea muy fácil, saltando obstáculos difíciles debido a las innumerables opciones de configuración disponibles tanto para el lado del servidor como para el lado del cliente.
- ✓ La función SSR ya está integrada en *Nuxt.js* y es fácil de usar. Permite acceder a las propiedades *isServer* y *isClient* de tus componentes para decidir si estás renderizando algo en el lado del cliente o del servidor.
- ✓ También proporciona el método *asyncData* dedicado a obtener y renderizar datos en el lado del servidor de tu componente de página.
- ✓ En *Nuxt.js*, un sitio web generado estáticamente es como construir una potente aplicación universal sin un servidor que potencie la función SSR.

1.7.2 Vuetify

Vuetify es un marco de interfaz de usuario completo construido sobre *Vue.js*. Su objetivo es proporcionar a los desarrolladores las herramientas que necesitan para crear experiencias de usuario ricas y atractivas. A diferencia de otros marcos, *Vuetify* está diseñado desde cero para que sea fácil de aprender y gratificante de dominar con cientos de componentes cuidadosamente elaborados de la especificación *Material Design*. (*vuejs* s. f.)

Vuetify adopta un primer enfoque móvil para el diseño, lo que significa que su aplicación simplemente funciona de inmediato, ya sea en un teléfono, tableta o computadora de escritorio.

Desde su lanzamiento inicial en 2014, *Vue.js* se ha convertido en uno de los marcos de *JavaScrip*t más populares del mundo. Una de las razones de esta popularidad es el

amplio uso de componentes que permiten a los desarrolladores crear módulos concisos para usar y reutilizar a lo largo de su aplicación. Las bibliotecas de interfaz de usuario son colecciones de estos módulos que implementan una guía de estilo específica y brindan las herramientas necesarias para crear aplicaciones web expansivas.

Vuetify se desarrolla exactamente de acuerdo con la especificación de *Material Design* con cada componente meticulosamente diseñado para ser modular, receptivo y de alto rendimiento. Lo que permite desarrollar aplicaciones con diseños únicos y dinámicos y personalizar los estilos de los componentes utilizando variables SASS.

1.8 Conclusiones parciales:

En este capítulo se abordaron los fundamentos teóricos útiles para emprender el desarrollo de la propuesta. Durante la preparación de este capítulo ha sido expuesta la metodología RUP para guiar el desarrollo del software y las tecnologías a emplear en el sistema.

Dentro de las herramientas de desarrollo e implementación: se toma el *Django* como *framework* y se utiliza el Python como lenguaje de programación, para el desarrollo de la interfaz gráfica se trabajó con *Nuxtjs* como *framework* de desarrollo basado en *Nodejs* y utilizando *Veutify* como *framework* de *Material Design*.

Capítulo II: Diseño de la aplicación propuesta. (Modelo del Negocio y Modelo del Sistema)

2.1 Introducción:

En este capítulo se detallan las reglas y los procesos del negocio, identificando los actores y trabajadores que en él intervienen, el diagrama de casos de uso del negocio, los diagramas de actividades correspondientes y el diagrama de clases del modelo de objetos. Además, se detallan los requerimientos funcionales y no funcionales, los casos de uso, los actores y el diagrama de casos de uso del sistema que dará solución al problema planteado. Esto se logra a través del uso de los artefactos que propone la metodología *RUP* y el lenguaje *UML*.

2.2 Breve Descripción del Negocio:

El negocio se basa en la gestión de nodos sensores mediante la introducción de sus datos por un usuario a través de una interfaz. El usuario, podrá entonces obtener un *Thing Description* de estos nodos sensores.

2.3 Reglas del negocio a considerar

Las reglas del negocio constituyen un conjunto de políticas o patrones a ser cumplidos con el fin de garantizar el adecuado funcionamiento del proceso del negocio. En el ambiente de trabajo asociado al sistema se han determinado las siguientes reglas del negocio:

El usuario, a través de los datos previamente insertados de un nodo sensor, obtiene la información de un *Thing Description*.

2.4 Modelo de Casos de Uso del Negocio

Un modelo de casos de uso del negocio describe los procesos de negocio de una empresa en términos de casos de uso del negocio y actores del negocio que se corresponden con los procesos del negocio y los clientes, respectivamente. El modelo de casos de uso del negocio presenta un sistema desde la perspectiva de su uso y esquematiza cómo proporciona valor a sus usuarios.

2.4.1 Actores del Negocio

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados.

Actor	Descripción
Usuario	Es el encargado de recopilar e introducir la información de los nodos sensores.
Software o dispositivo	Puede consumir la información de un *Thing Description*.

2.4.2 Trabajadores del negocio

Un trabajador del negocio es una abstracción de una persona (o grupo de personas), una máquina o sistema automatizado; que actúa en el negocio realizando una o varias actividades, interactuando con otros trabajadores del negocio y manipulando entidades del negocio.

Trabajador	Descripción
Usuario	Es el encargado de recopilar e introducir la información. Además obtiene el *Thing Description*

2.4.3 Diagrama de casos de uso del Negocio

Un diagrama de casos de uso del negocio representa gráficamente a los procesos del negocio y su interacción con los actores del negocio.

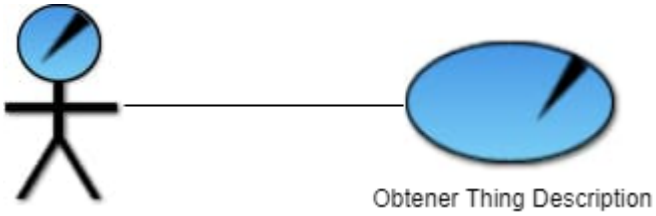

Ilustración 1: Diagrama de caso de uso del negocio

2.5 Diagrama de Actividades

Un diagrama de actividades es el diagrama que muestra el flujo de actividad a actividad, tratando la vista dinámica de un sistema. Es un caso especial de diagrama de estados en el cual todos los estados son estados de acción y en el cual todas o casi todas las transiciones son disparadas por la terminación de las acciones en los estados origen.

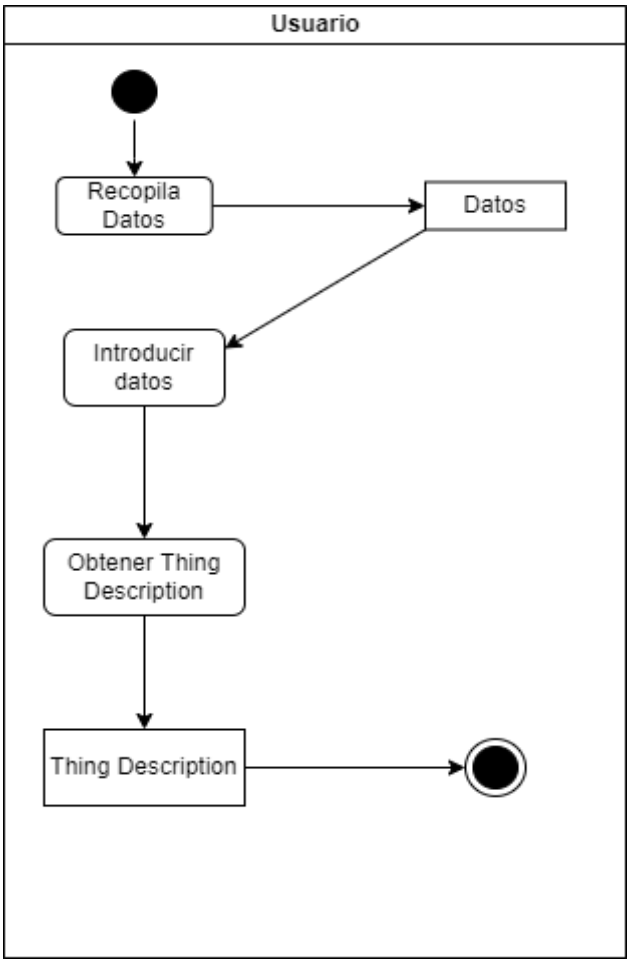

Ilustración 2: Diagrama de actividades

2.6 Modelo de Objetos

Un modelo de objetos del negocio es un modelo interno a un negocio. Describe cómo cada caso de uso del negocio es llevado a cabo por parte de un conjunto de trabajadores que utilizan un conjunto de entidades del negocio y de unidades de trabajo.

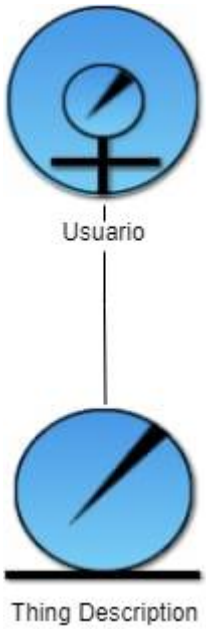

Ilustración 3: Diagrama de objeto

2.7 Requisitos no funcionales

Los requisitos funcionales especifican una acción que debe ser capaz de realizar el sistema, sin considerar restricciones físicas.

- Apariencia o interfaz externa
- Usabilidad
- Portabilidad
- Seguridad
- Confiablidad
- Ayuda y documentación en línea

- Software
- Hardware

2.8 Requisitos funcionales:

- R1 Gestionar nodo sensor
    1. Insertar nodo sensor
    2. Modificar nodo sensor
    3. Eliminar nodo sensor
    4. Mostrar nodo sensor

- R2 Gestionar Propietario
    1. Insertar Propietario
    2. Modificar Propietario
    3. Eliminar Propietario
    4. Mostrar Propietario

- R3 Gestionar Lugar
    1. Insertar Lugar
    2. Modificar Lugar
    3. Eliminar Lugar
    4. Mostrar Lugar

- R4 Gestionar Arquitectura del nodo sensor
    1. Insertar Arquitectura del nodo sensor
    2. Modificar Arquitectura del nodo sensor
    3. Eliminar Arquitectura del nodo sensor
    4. Mostrar Arquitectura del nodo sensor

- R5 Gestionar Marca
    1. Insertar Marca
    2. Modificar Marca

3. Eliminar Marca
4. Mostrar Marca

- R6 Gestionar Interfaz de Red
    1. Insertar Interfaz de Red
    2. Modificar Interfaz de Red
    3. Eliminar Interfaz de Red
    4. Mostrar Interfaz de Red

- R7 Gestionar Interfaz de red cableada
    1. Insertar Interfaz de red cableada
    2. Modificar Interfaz de red cableada
    3. Eliminar Interfaz de red cableada
    4. Mostrar Interfaz de red cableada

- R8 Gestionar Interfaz de red Inalámbrica
    1. Insertar Interfaz de red Inalámbrica
    2. Modificar Interfaz de red Inalámbrica
    3. Eliminar Interfaz de red Inalámbrica
    4. Mostrar Interfaz de red Inalámbrica

- R9 Gestionar Fuente de Alimentación
    1. Insertar Fuente de Alimentación
    2. Modificar Fuente de Alimentación
    3. Eliminar Fuente de Alimentación
    4. Mostrar Fuente de Alimentación

- R10 Gestionar Fuente de Alimentación por cable
    1. Insertar Fuente de Alimentación por cable
    2. Modificar Fuente de Alimentación por cable
    3. Eliminar Fuente de Alimentación por cable

4. Mostrar Fuente de Alimentación por cable

- R11 Gestionar Fuente de Alimentación por batería
    1. Insertar Fuente de Alimentación por batería
    2. Modificar Fuente de Alimentación por batería
    3. Eliminar Fuente de Alimentación por batería
    4. Mostrar Fuente de Alimentación por batería

- R12 Gestionar Placa
    1. Insertar Placa
    2. Modificar Placa
    3. Eliminar Placa
    4. Mostrar Placa

- R13 Gestionar Sensor
    1. Insertar Sensor
    2. Modificar Sensor
    3. Eliminar Sensor
    4. Mostrar Sensor

- R14 Gestionar MCU
    1. Insertar MCU
    2. Modificar MCU
    3. Eliminar MCU
    4. Mostrar MCU

- R15 Gestionar Magnitud Variable
    1. Insertar Magnitud Variable
    2. Modificar Magnitud Variable
    3. Eliminar Magnitud Variable

4. Mostrar Magnitud Variable

- R16 Gestionar Medición
    1. Insertar Magnitud Medición
    2. Modificar Magnitud Medición
    3. Eliminar Magnitud Medición
    4. Mostrar Magnitud Medición

2.9 Modelo de casos de uso del Sistema

El modelo de casos de uso del sistema permite que los desarrolladores de software y los clientes lleguen a un acuerdo sobre los requisitos (...). El modelo de casos de uso del sistema sirve como acuerdo entre clientes y desarrolladores, y proporciona la entrada fundamental para el análisis, el diseño y las pruebas.

2.10 Actores del sistema

Un actor es una idealización de una persona externa, de un proceso, o de una cosa que interactúa con un sistema. Cada actor participa en uno o más casos de uso. Un actor puede ser un ser humano, otro sistema informático, o un cierto proceso ejecutable.

Actores	Justificación
Usuario	Es el encargado de recabar e introducir la información de los nodos sensores.

2.11 Casos de uso del sistema

Un caso de uso es una secuencia de acciones que el sistema lleva a cabo para ofrecer algún resultado de valor para un actor.

El sistema propuesto cuenta con el siguiente caso de uso:

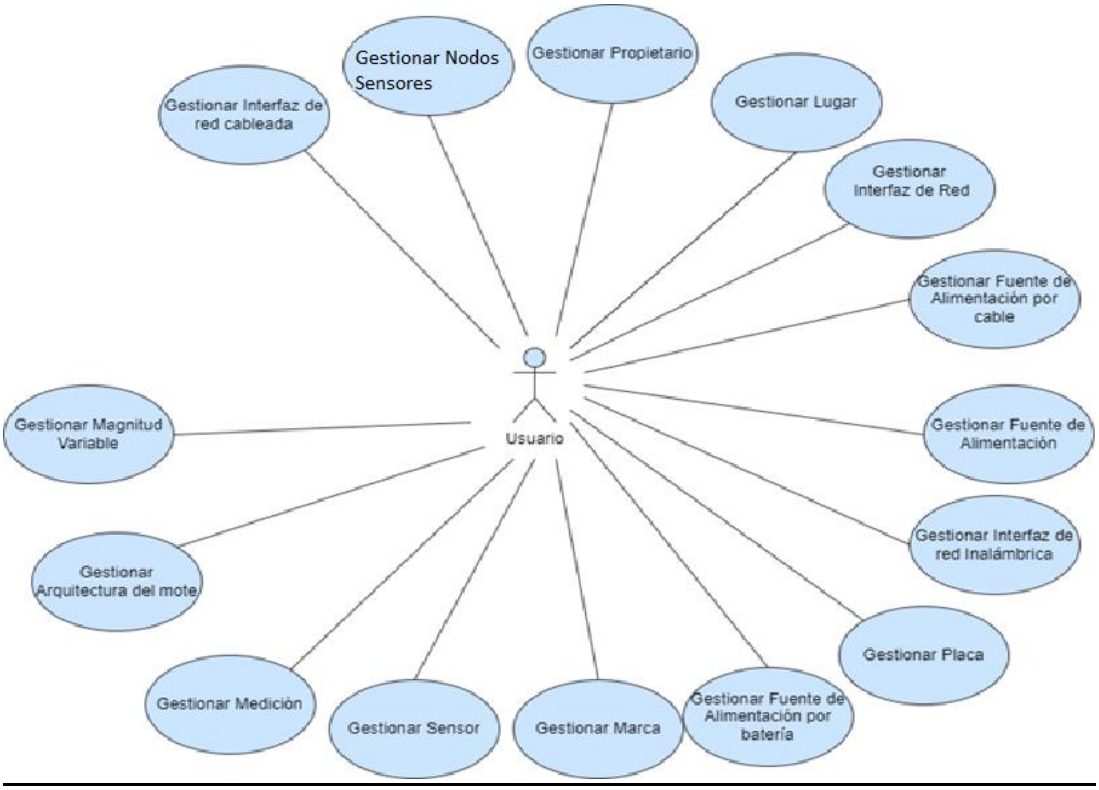

Ilustración 4: Casos de uso del sistema

Expansión textual de los casos de uso del sistema.

CU1	**Gestionar nodo sensor**
Propósito:	Crear, modificar y eliminar nodo sensor.
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita ingresar un nuevo nodo sensor al sistema, o se presenta alguna modificación con los datos de los nodos sensores ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados.
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de

	nodos sensores.
Postcondiciones:	El sistema ha gestionado un nodo sensor.
Sección: "Insertar nodo sensor "	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar una nueva entidad.
3. El usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema inserta los datos de la nueva entidad. Culmina el caso de uso.
Sección: "Modificar nodo sensor "	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos datos en el formulario.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema actualiza los datos en la base de datos. 6. El sistema actualiza el listado de las entidades. Culmina el caso de uso.
Sección: "Eliminar nodo sensor "	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.

3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar nodo sensor"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El usuario accede al apartado de Nodos sensores de su interfaz de trabajo.	2. El sistema muestra un listado con las entidades existentes. Culmina el caso de uso
Prioridad	Alto

Tabla 1: Descripción de caso de uso del sistema <Gestión de Nodos sensores>

2.12 Análisis y Diseño del sistema

En el diseño modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos. Además, impone una estructura del sistema que debemos esforzarnos por conservar lo más fielmente posible cuando demos forma al sistema.

2.12.1 Diagrama de clases del diseño

El diagrama de clases de diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Contiene: clases, asociaciones y atributos; interfaces, con sus operaciones y constantes; métodos; información sobre los tipos de atributos; navegabilidad y dependencias. A diferencia del modelo

conceptual, un diagrama de este tipo contiene las definiciones de las entidades del software en vez de conceptos del mundo real.

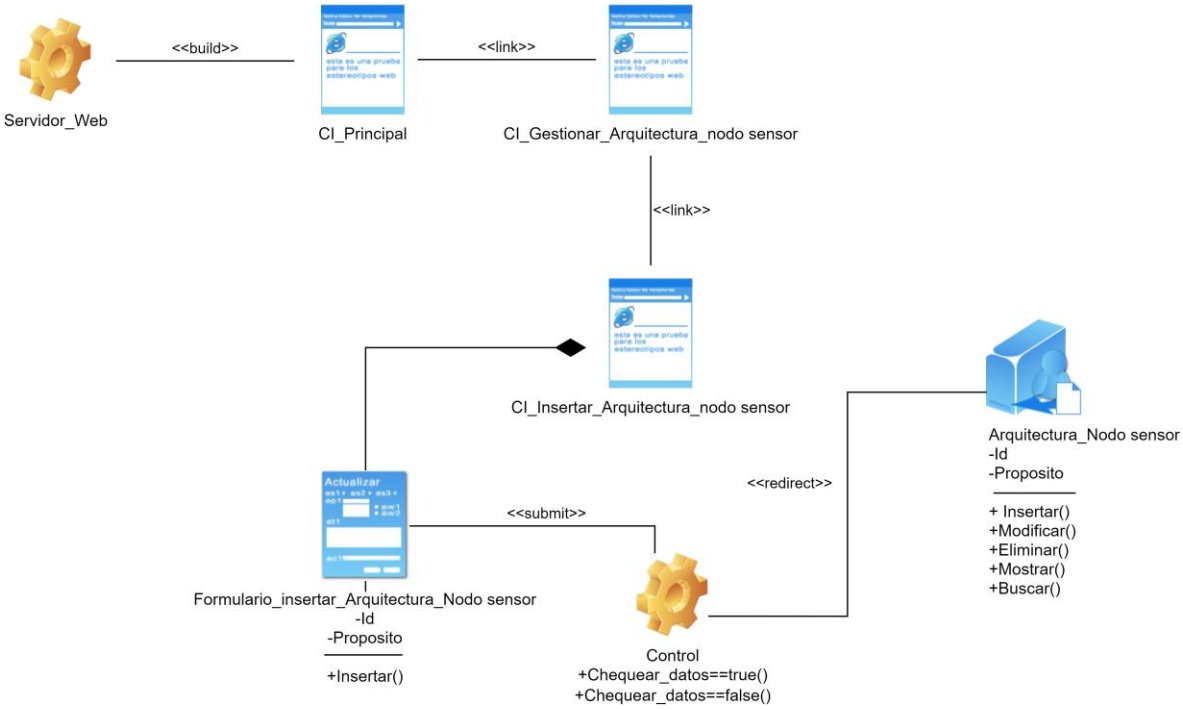

Ilustración 5 Diagrama de Diseño <Insertar Nodo Sensor>

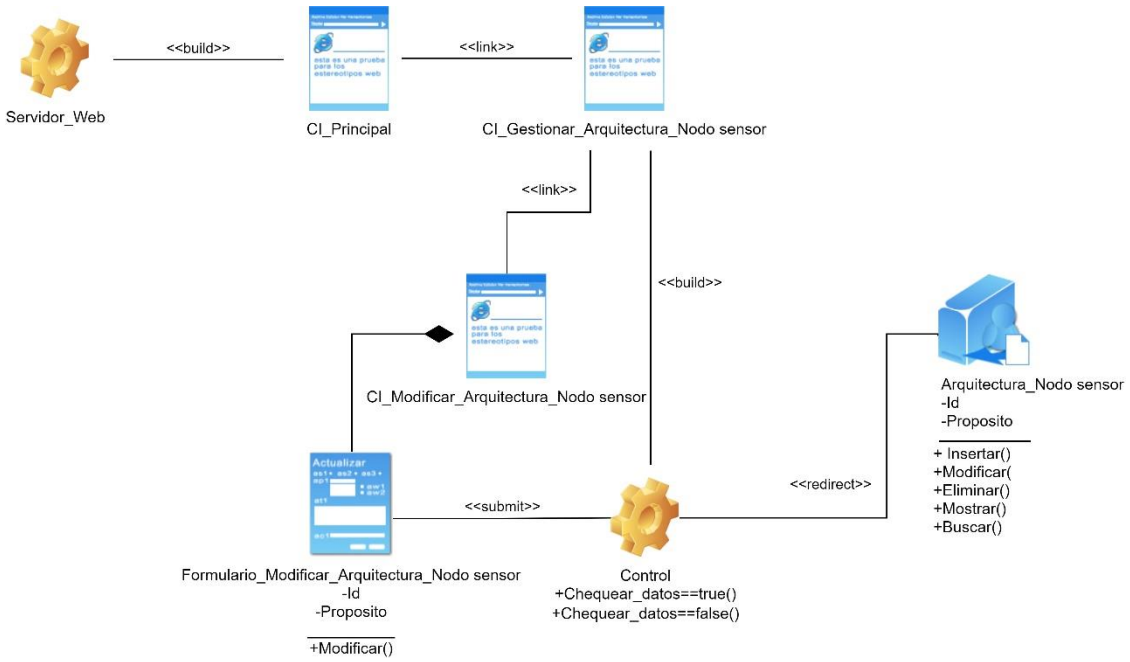

Ilustración 6 Diagrama de Diseño <Modificar Nodo Sensor>

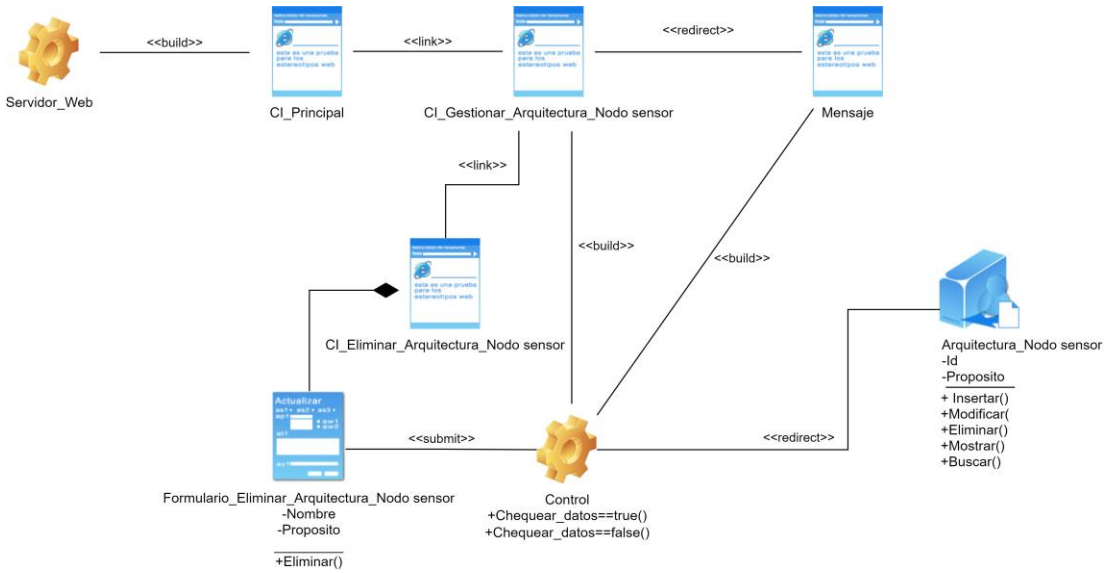

Ilustración 7: Diagrama de Diseño <Eliminar Nodo Sensor>

2.12.2 Diagrama de colaboración

Muestra las interacciones entre objetos, creando enlaces entre ellos y añadiendo a estos enlaces mensajes, donde el nombre del mensaje debe denotar el propósito del objeto que se invoca en la interacción.

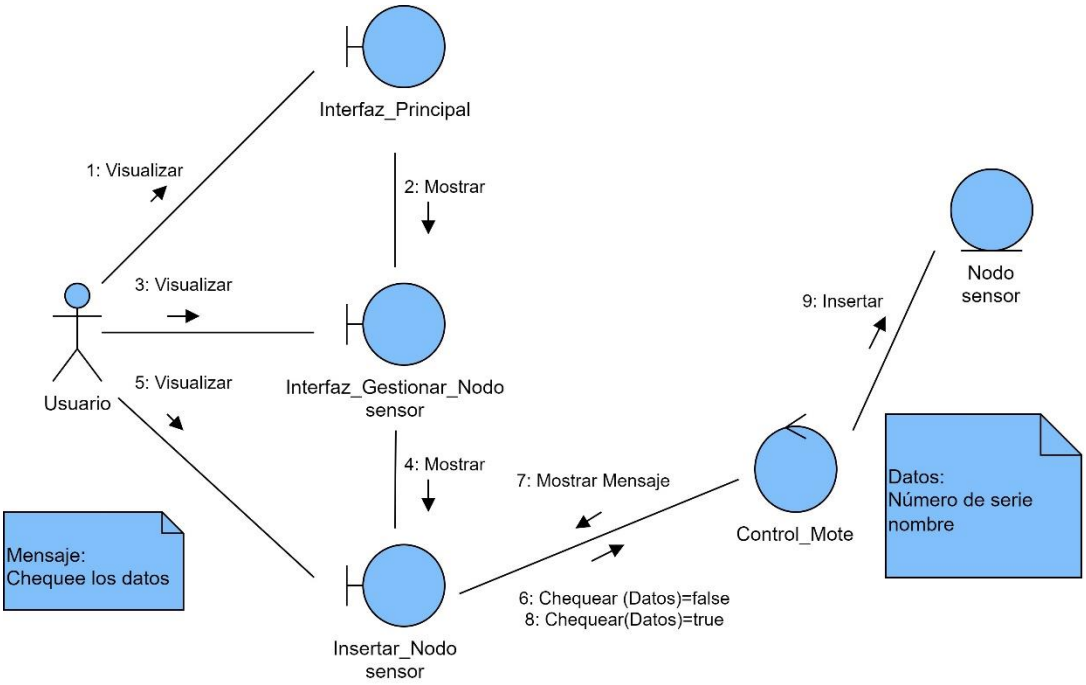

Ilustración 8: Diagrama de Colaboración <Insertar Nodo Sensor>

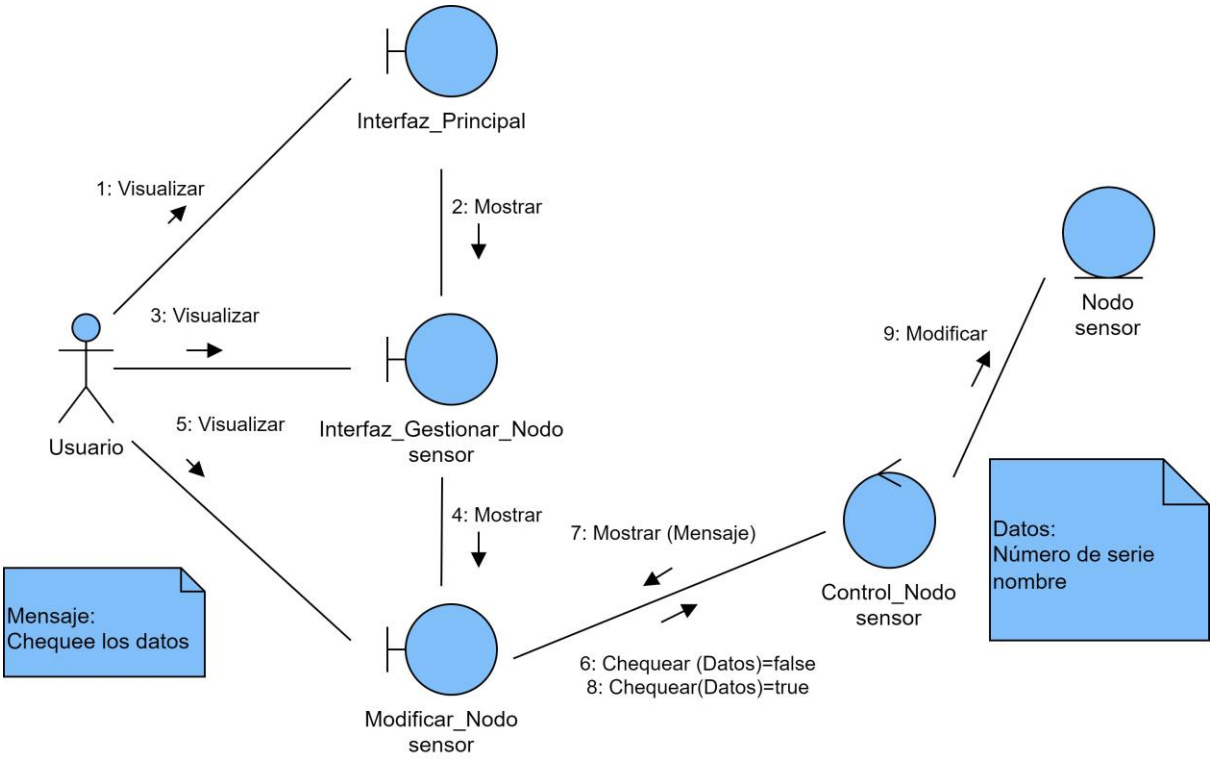

Ilustración 9: Diagrama de Colaboración <Modificar Nodo Sensor>

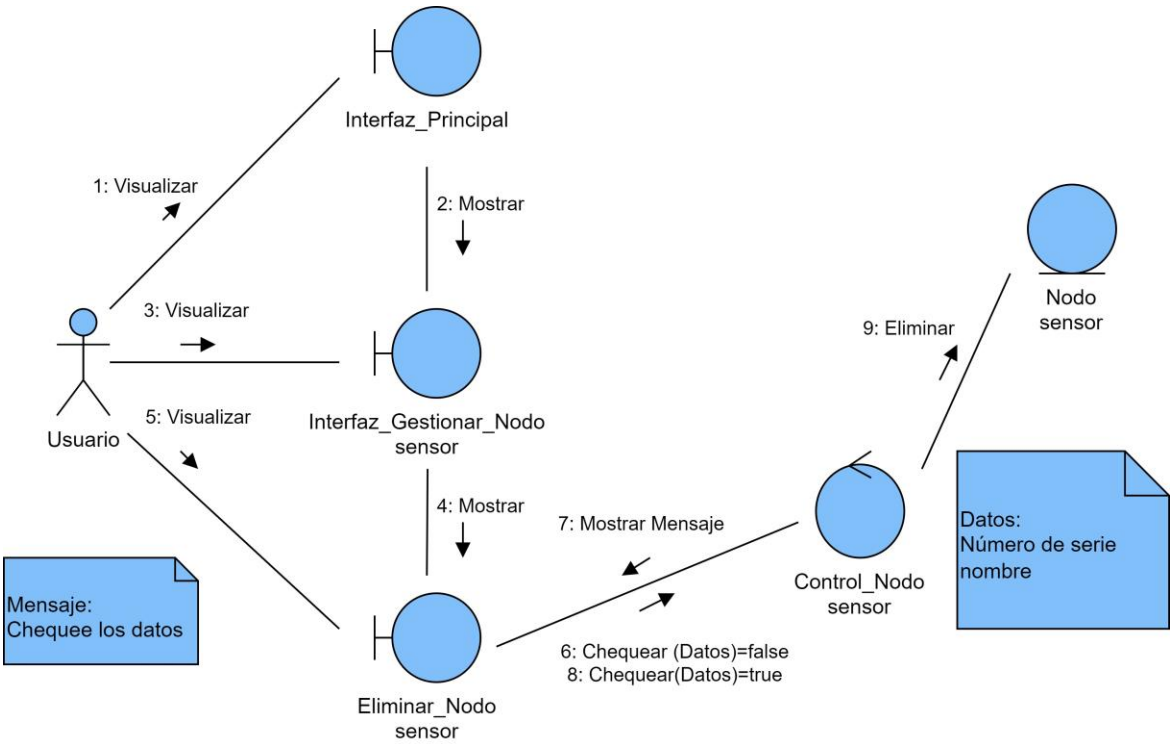

Ilustración 10: Diagrama de Colaboración <Eliminar Nodo Sensor>

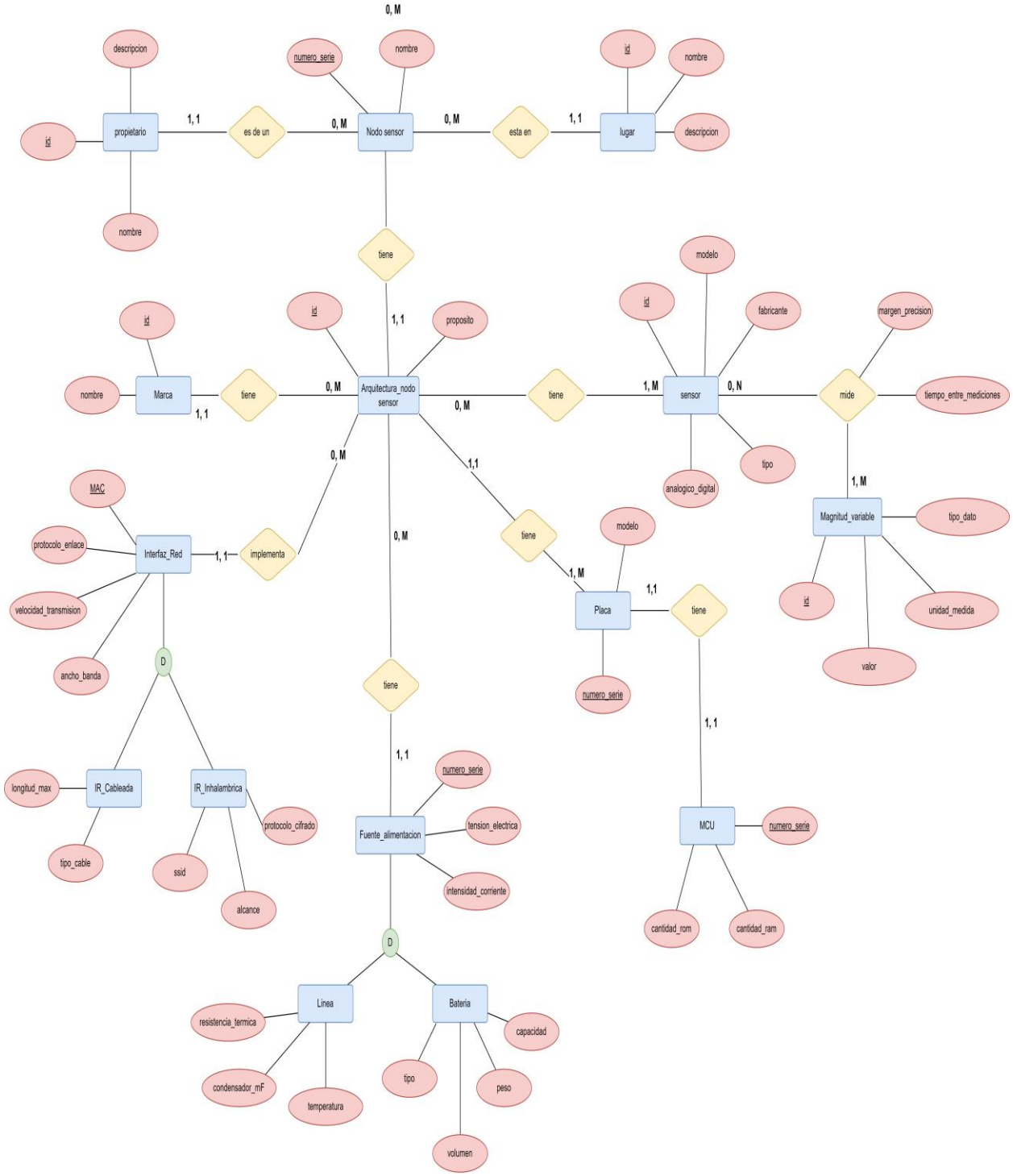

Ilustración 11: Diagrama de Entidad Relación

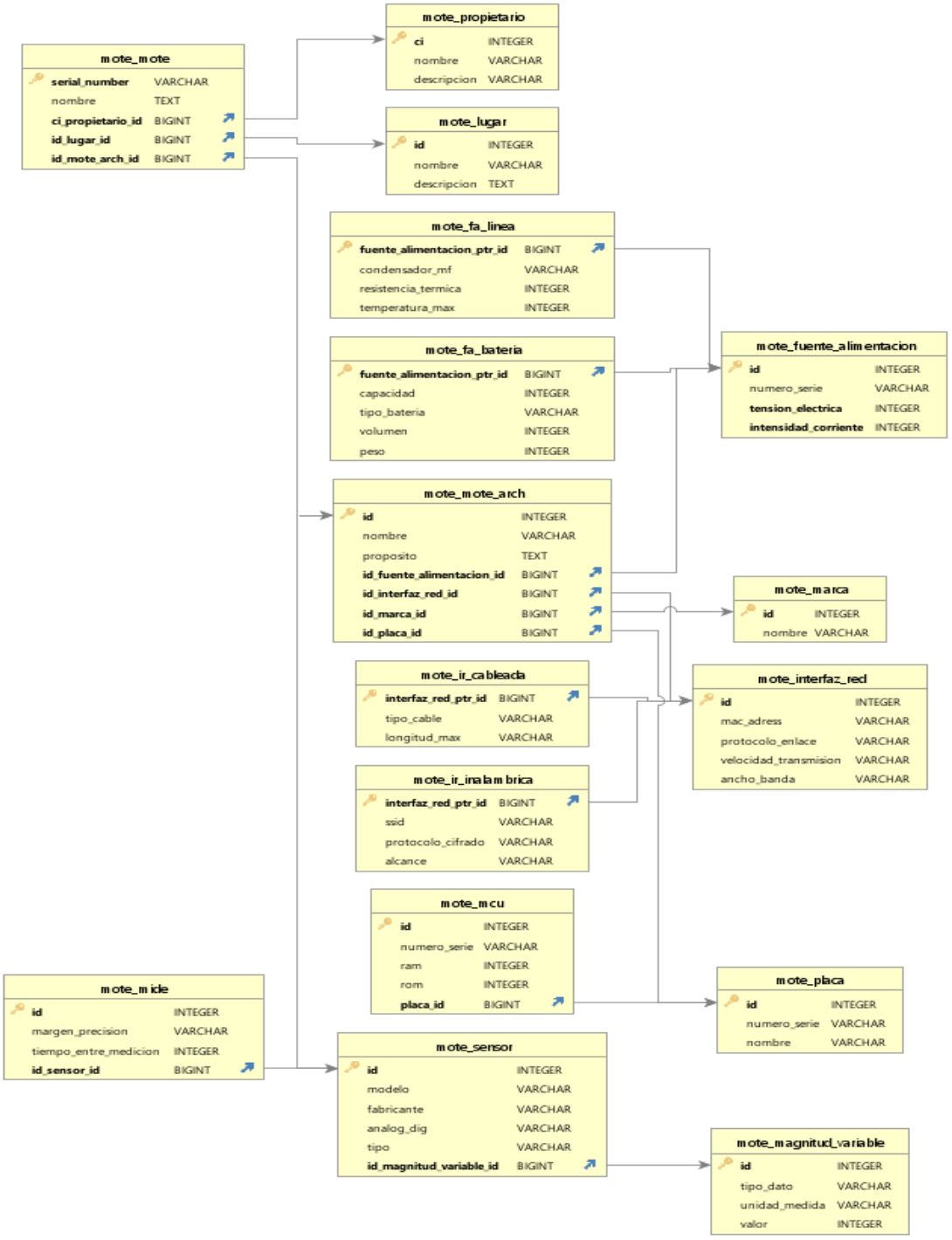

Ilustración 12: Diagrama de base de datos

Conclusiones parciales

En este capítulo se describieron los elementos que componen el proceso estudiado a través del lenguaje UML, utilizando la metodología RUP. Se analizaron los diferentes roles de los usuarios y trabajadores de este proceso, y se determinaron los actores y trabajadores del negocio, las operaciones principales que producen resultados en forma de casos de uso y su descripción literal, se elaboraron los diagramas de actividad de los casos de uso y el modelo de objetos. Fueron definidos los requerimientos funcionales y no funcionales. El sistema propuesto fue descrito a través de la determinación de los actores del sistema, los paquetes que contienen los casos de uso del sistema y su descripción textual. Se muestran también los diagramas de clase de diseño y los diagramas de colaboración.

CAPÍTULO 3: Implementación y prueba de la aplicación

3.1 Introducción

En este capítulo describirá la construcción del sistema a través la elaboración de los diagramas de despliegue y de componente. Se observa también el prototipo de la interfaz de usuario y se realizará además las pruebas de caja negra a los casos de uso más significativos.

3.2 Ayuda y Tratamiento de errores Ayuda:

La ayuda del sistema se encuentra en el menú principal de la aplicación y consiste en: un mapa de navegación, que muestra los pasos a seguir para llegar a cada una de las funcionalidades del *software*; las sugerencias en las interfaces, que le indican al usuario cuáles son los campos obligatorios a llenar o si ha cometido un error y una ayuda general que explica cómo interactuar con el sistema. Tratamiento de errores: El sistema propuesto presenta una interfaz diseñada, implementada y dirigida a evitar excepciones y errores. El mismo tiene la obligación de detectar problemas en el proceso de autentificación por parte del usuario, presenta mecanismos de validación de la información con el propósito de minimizar las posibilidades de introducir información errónea, y aclara al usuario el tipo de información que debe manipular, para esto aprovecha constantemente las opciones de selección de listas para minimizar la entrada de errores por teclado. Todo esto a través de una serie de mensajes de error de fácil comprensión para los usuarios.

Prototipos de la Interfaz de Usuario

Ilustración 13: Primera página, donde el usuario puede obtener información de la aplicación

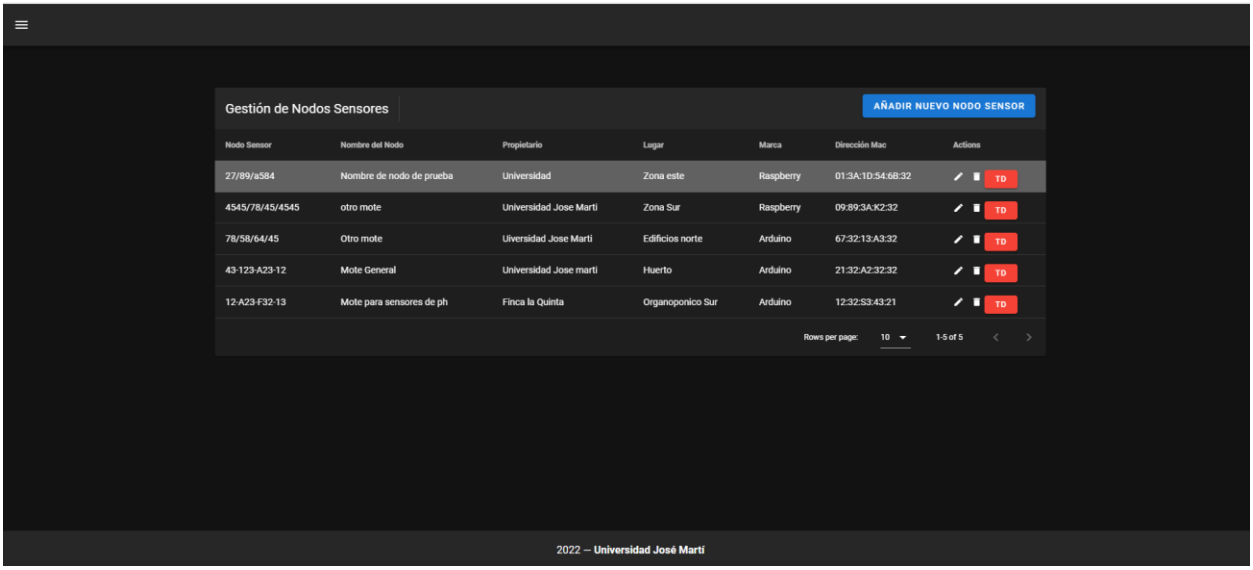

Ilustración 14: Página de Gestión de Nodos Sensores

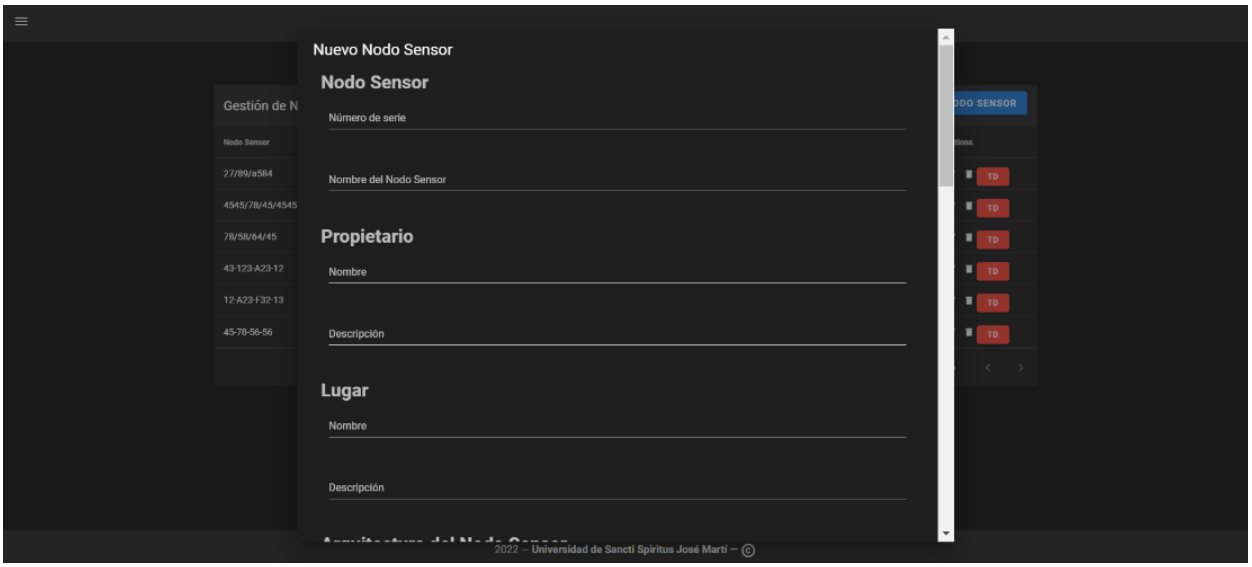

Ilustración 15: Ventana emergente que nos permitirá introducir un nuevo nodo sensor con sus respectivas estructuras

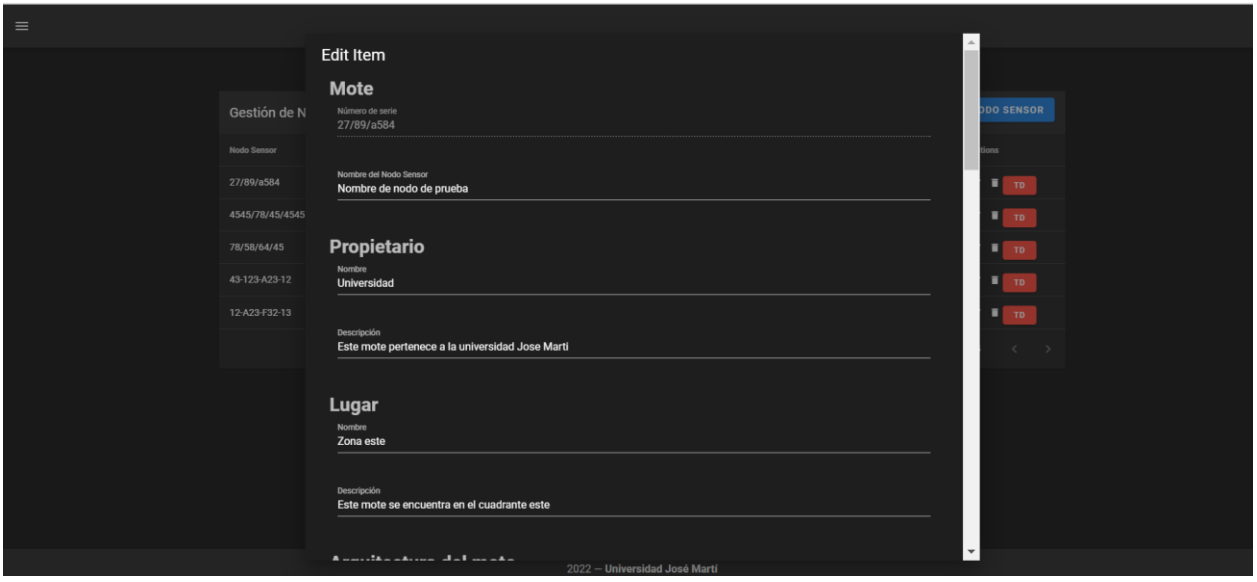

Ilustración 16: Ventana emergente que nos permite editar los datos de las estructuras de los nodos sensores

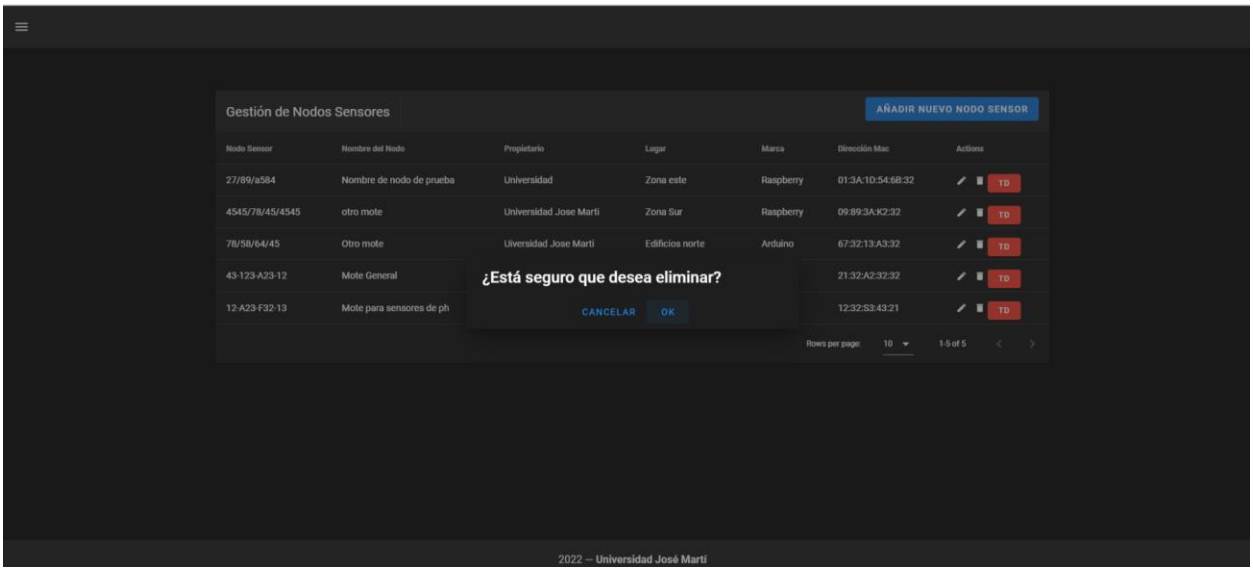

Ilustración 17: Cuadro emergente para la confirmación de eliminar la entidad seleccionada

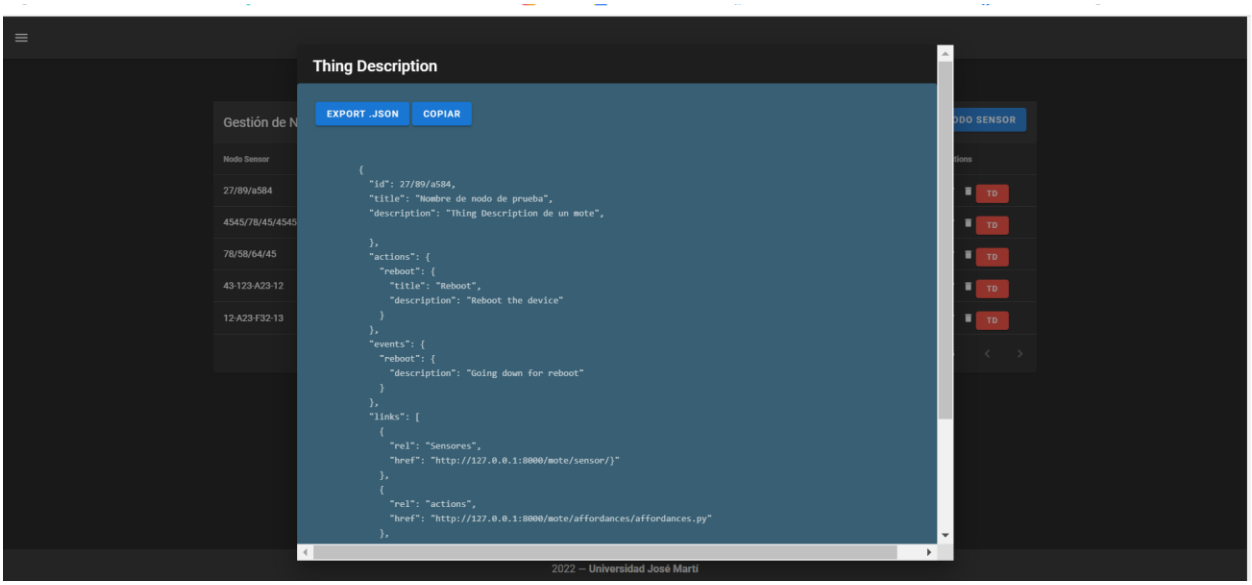

Ilustración 18: *Thing Description* general de un nodo sensor

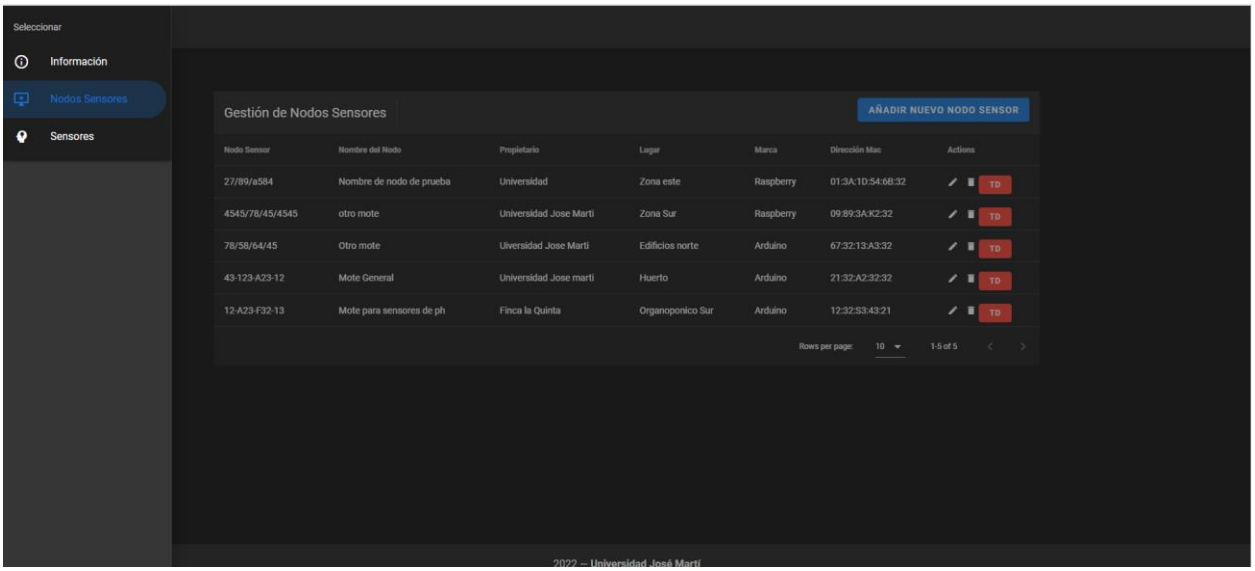

Ilustración 19: *Drawer* implementado para facilitar la navegación

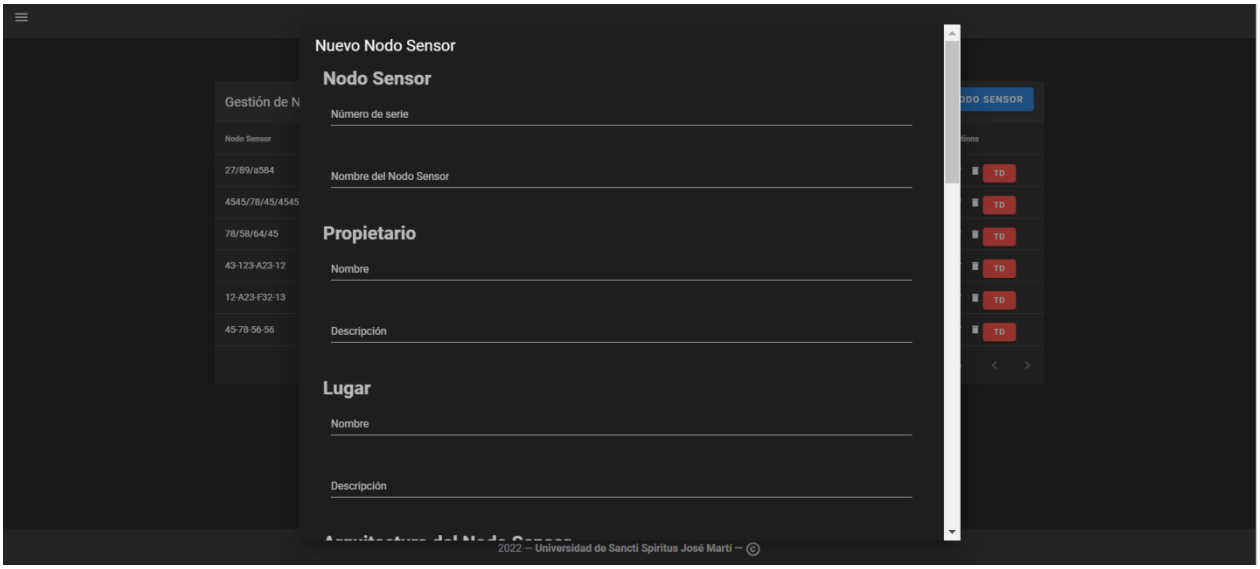

Ilustración 20: Ventana emergente que nos permite insertar un nuevo sensor

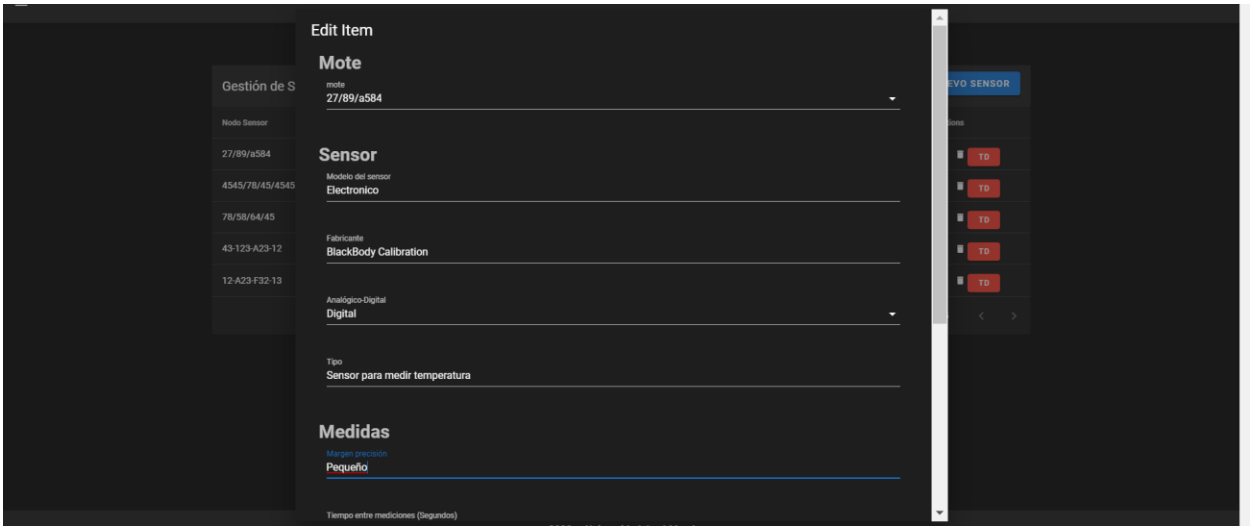

Ilustración 21: Ventana emergente que nos permite editar los datos de un sensor

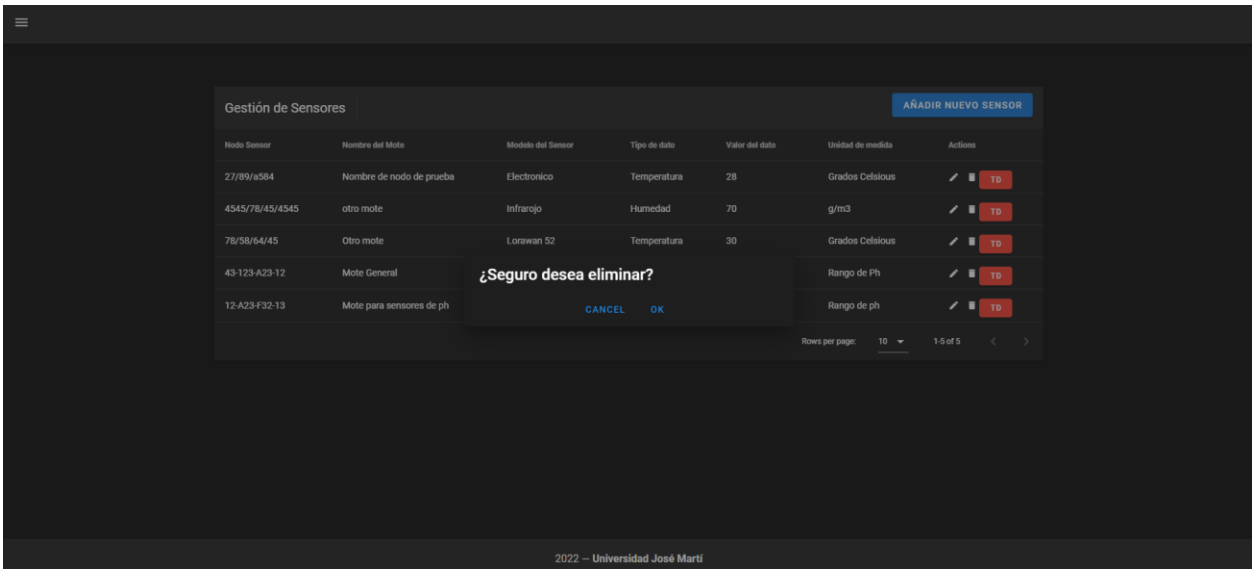

Ilustración 22: Ventana emergente para confirmar la eliminación del sensor seleccionado

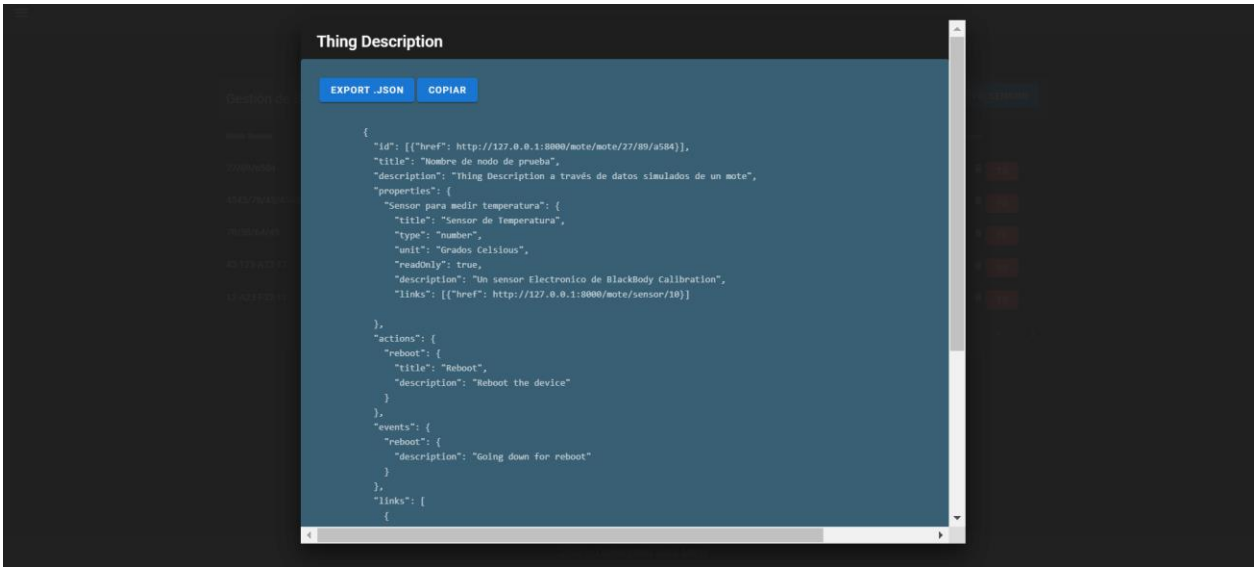

Ilustración 23: *Thing Description* del sensor seleccionado

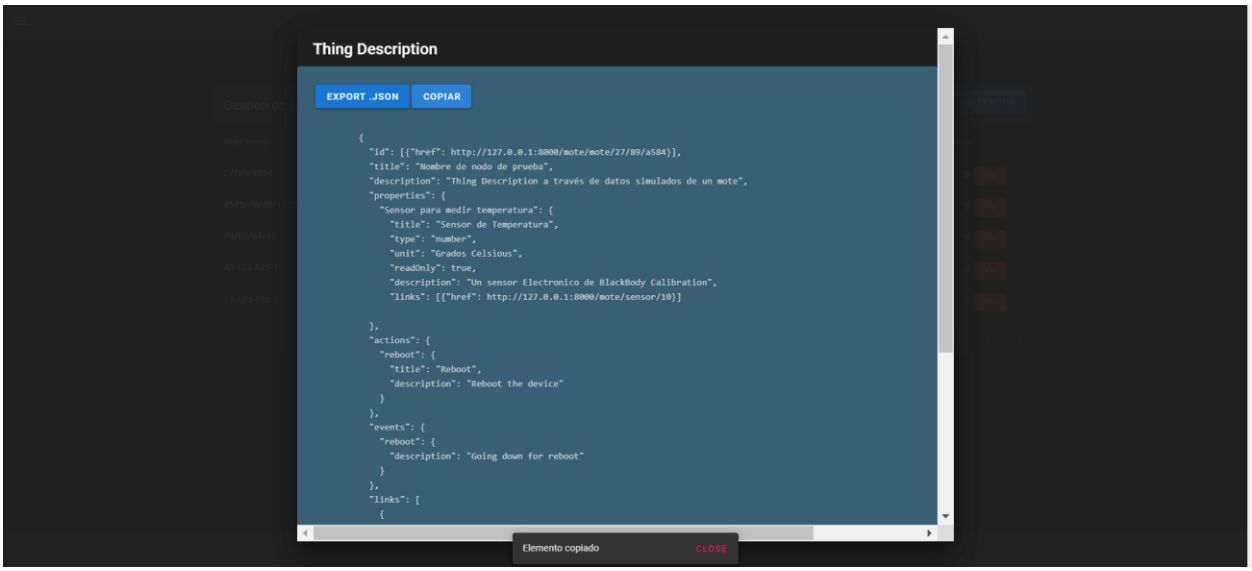

Ilustración 24: Alerta emergente que nos da a conocer que el elemento ha sido copiado correctamente

3.3 Diagrama de Despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los

nodos de cómputo. El modelo de despliegue se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño.

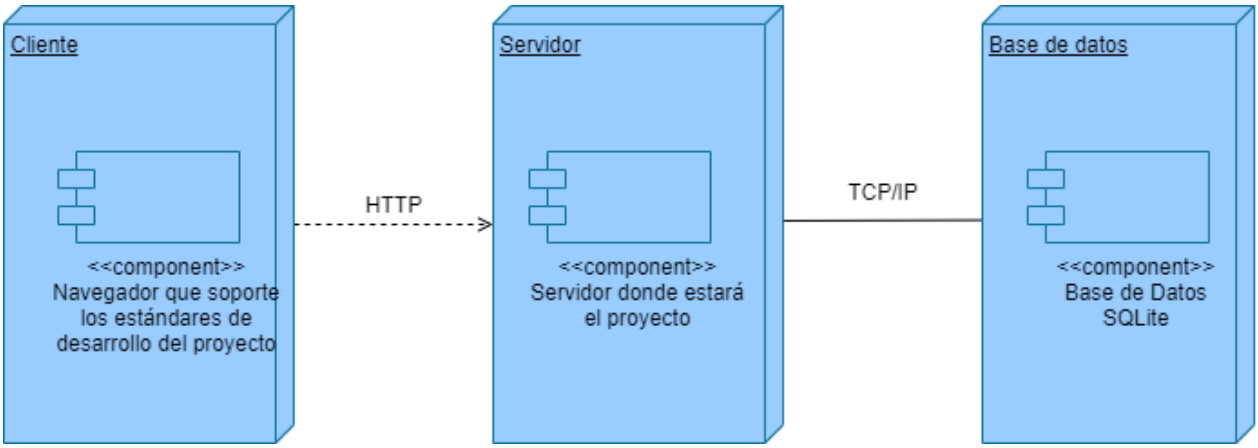

Ilustración 25: Modelo de despliegue

3.4 Diagrama de Componentes

Los diagramas de componentes muestran una vista física del modelo, así como las relaciones entre los componentes del *software*. Estos diagramas son usados por los responsables de implementar el sistema y les indicará en qué orden necesitan ser compilados los componentes.

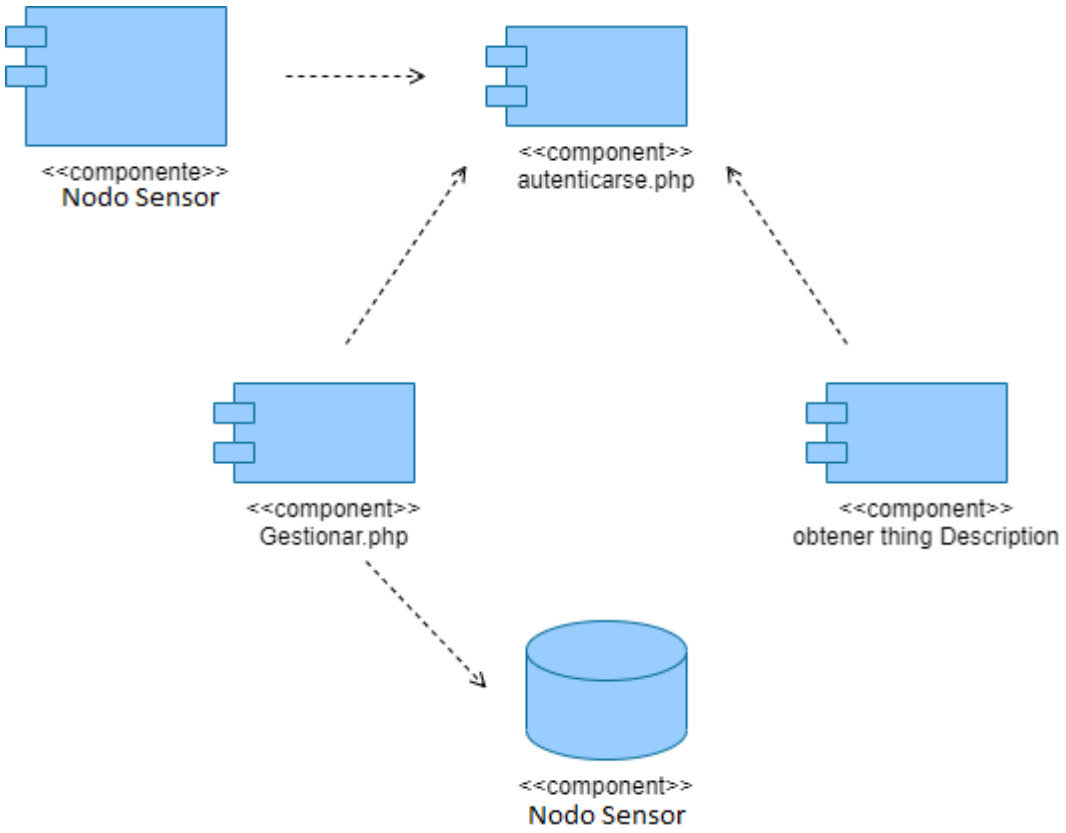

Ilustración 26: Diagrama de Componentes

3.5 Prueba de caja negra

Las pruebas de caja negra permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. Estas pruebas se realizan mediante la técnica de partición de equivalencia que consiste en dividir el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del *software*.

La interfaz “Gestionar Magnitud Variable” toma como entradas a validar las siguientes:

- Tipo de dato
- Unidad de medida
- Valor de la magnitud

En este caso:

- Tipo de dato: es un campo que admite caracteres alfanuméricos
- Unidad de medida: es un campo que admite caracteres alfanuméricos
- Valor de la magnitud: solo admite números

Condición de Entrada	Clases Válidas	Clases Inválidas
Tipo de dato:	Caracteres alfanuméricos	Campo nulo
Unidad de medida:	Caracteres alfanuméricos	Campo nulo
Valor de la magnitud:	Número *float*	Campo nulo Letras Caracteres especiales

Tabla de Equivalencia

Este método intenta dividir el dominio de entrada de un programa en un número finito de clases de equivalencia.

Objetivo

- Definición del menor número de casos de prueba que descubran clases de errores

Código de entrada	Tipo	Clases válidas	Clases no válidas	Regla
Tipo de dato	Valor	a-z(8) A-Z(9) ·”ª!·$%/(())¿¿	Campo nulo	*CharField*
Unidad de medida	Valor	a-z(8) A-Z(9) ·”ª!·$%/(())¿¿	Campo nulo	*CharField*
Valor de la magnitud	Valor	Valor de tipo fecha	Otro campo q no sea fecha	*Float*

Casos de prueba			Clases	Resultado
Tipo de dato	Unidad de medida	Valor de la magnitud		
Temperatura	Grados Celsius	45	1	Correcto
“ ”	“ ”	“ ”	2	Incorrecto
Ph del suelo	Unidades de ph	1	3	Correcto
Humedad	g/m3	Treinta	4	Incorrecto

3.6 Conclusiones parciales

En este capítulo se definieron los principios de diseño seguidos en el sistema y el modelo de implementación. Se mostraron los prototipos de interfaces visuales que se van a desarrollar. Se representó el diagrama de componentes integrado al de despliegue. Finalmente se llevó a cabo las pruebas de caja negra realizadas al caso de uso más ilustrativo.

Conclusiones Generales

Se evidenció las potencialidades de los fundamentos teóricos metodológicos analizados, como herramientas claves para el desarrollo de la aplicación. Su diseño estuvo sustentado por la metodología RUP, estableciendo las pautas para su desarrollo. La implementación de la aplicación, de forma general, permite obtener un consenso en la presentación de datos a nivel de la aplicación

Bibliografía

Åkerman, Magnus. 2016. «Towards interoperable information and communication systems for manufacturing operations».

Yacchirema. s. f.-c. «Yacchirema - Arquitectura de Interoperabilidad de dispositivos físicos para el Internet de las C....pdf».

Carlemany. s. f. «Metodologías de desarrollo de software | Universitat Carlemany». Recuperado 25 de noviembre de 2022 (https://www.universitatcarlemany.com/actualidad/metodologias-de-desarrollo-de-software).

Charpenay, Victor, Sebastian Kabisch, y Harald Kosch. s. f. «Introducing Thing Descriptions and Interactions: An Ontology for the Web of Things». 12.

Define, Framework La palabra inglesa framework, En Términos Generales, Un Conjunto Estandarizado De Conceptos, Prácticas Y. Criterios Para Enfocar Un Tipo De Problemática Particular, y Que Sirve Como Referencia Para Enfrentar Y. Resolver Nuevos Problemas De Índole Similar. s. f. «Framework - EcuRed». Recuperado 25 de noviembre de 2022 (https://www.ecured.cu/Framework).

Gillis, Alexander. s. f. «What Is IoT (Internet of Things) and How Does It Work? - Definition from TechTarget.Com». *IoT Agenda*. Recuperado 25 de noviembre de 2022 (https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT).

Hassine, Hassib Belhaj, Ege Korkan, y Sebastian Steinhorst. s. f. «Virtual-Thing: Thing Description Based Virtualization». 3.

Industrial. 2021. «Sensores: Qué Son, Cómo Funcionan, Características Y Tipos». Recuperado 25 de noviembre de 2022 (https://sdindustrial.com.mx/blog/sensores/).

iomkt.domainlogic@gmail.com. 2022. «Metodologías de desarrollo de software 2022». *Domain Logic*. Recuperado 25 de noviembre de 2022 (https://domainlogic.io/metodologias-de-desarrollo-de-software-2022/).

Joskowicz José. 2022. «Internet de Las Cosas: ¿cómo Funciona? | Isbel». Recuperado 25 de noviembre de 2022 (https://isbel.com/internet-de-las-cosas-como-funciona/).

kinsta. s. f. «What Is Nuxt.js? Learn More About the Intuitive Vue Framework». Recuperado 25 de noviembre de 2022 (https://kinsta.com/knowledgebase/nuxt-js/).

Marketing. 2022. «Los 12 tipos de sensores más usados: características y funciones». *EDS Robotics*. Recuperado 25 de noviembre de 2022 (https://www.edsrobotics.com/blog/tipos-sensores-mas-usados/).

MDN. s. f. «Introducción a Django - Aprende sobre desarrollo web | MDN». Recuperado 25 de noviembre de 2022 (https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction).

Mujica-Sequera, Ruth M. 2022. «✍ ¿QUÉ ES EL LENGUAJE DE MODELADO UNIFICADO (UML)? | DOCENTES 2.0». *Docentes 2.0*. Recuperado 25 de noviembre de 2022 (https://blog.docentes20.com/2022/06/✍-que-es-el-lenguaje-de-modelado-unificado-uml-docentes-2-0/).

Node.js. s. f. «Documentation». *Node.Js*. Recuperado 25 de noviembre de 2022 (https://nodejs.org/en/docs/).

Norberto Mateos. s. f. «El Internet de las Cosas y su impacto transformador en la sociedad». Recuperado 25 de noviembre de 2022 (https://www.universia.net/co/actualidad/orientacion-academica/el-internet-de-las-cosas-y-su-impacto-transformador-en-la-sociedad.html).

Pardo, Dimas. 2018. «Interoperabilidad del IoT; la clave para su desarrollo». *Pandora FMS - The Monitoring Blog*. Recuperado 29 de noviembre de 2022 (https://pandorafms.com/blog/es/interoperabilidad-del-iot/).

Paredes, Daniel Alexander Vera, Luis Cristobal Córdova Martínez, Ricauter Moises López Bermúdez, y Silvia Rosa Pacheco Mendoza. 2019. «Análisis de la metodología RUP en el desarrollo de software académico mediante la herramienta DJANGO». *RECIMUNDO* 3(2):964-79. doi: 10.26820/recimundo/3.(2).abril.2019.964-979.

Radix. 2021. «NodeJS vs Python - Which Language is Best for Backend Web Development?» *Radixweb*. Recuperado 25 de noviembre de 2022 (https://radixweb.com/blog/nodejs-vs-python).

Rose, Karen, Scott Eldridge, y Lyman Chapin. s. f. «LA INTERNET DE LAS COSAS— UNA BREVE RESEÑA». 83.

Santander.P. s. f. «Python: qué es y por qué deberías aprender a utilizarlo». Recuperado 25 de noviembre de 2022 (https://www.becas-santander.com/es/blog/python-que-es.html).

School, Tokio. 2022. «Historia y evolución del Internet de las Cosas (IoT) | Tokio». *Tokio School*. Recuperado 25 de noviembre de 2022 (https://www.tokioschool.com/noticias/internet-de-las-cosas-evolucion/).

SensorGO. 2021. «Sensores Inteligentes Para la Medición Doméstica e Industrial». *SensorGO*. Recuperado 25 de noviembre de 2022 (https://sensorgo.mx/sensores-inteligentes/).

Tavizon, Arturo, Tania Guajardo, y Cristina I. Laines Alamina. 2016. «IOT, el internet de las cosas y la innovación de sus aplicaciones». 2.

Thing description. s. f. «Fig. 5 WoT Thing Description Core Concepts». *ResearchGate*. Recuperado 25 de noviembre de 2022 (https://www.researchgate.net/figure/WoT-Thing-Description-core-concepts_fig4_334084550).

vida", Paloma Recuero de los Santos Especialista en generación de contenidos tecnológicos para los canales digitales de Telefónica Tech AI of Things Licenciada en Ciencias Físicas y Máster en Tecnología Educativa Apasionada por las "tecnologías para la, y Las Que Nos Hacen La Vida Más Fácily Por La Pedagogía. 2020. «Breve historia de Internet de las cosas (IoT)». *Think Big*. Recuperado 25 de noviembre de 2022 (https://empresas.blogthinkbig.com/breve-historia-de-internet-de-las-cosas-iot/).

vida", Paloma Recuero de los Santos Especialista en generación de contenidos tecnológicos para los canales digitales de Telefónica Tech AI of Things Licenciada en

Ciencias Físicas y Máster en Tecnología Educativa Apasionada por las “tecnologías para la, y Las Que Nos Hacen La Vida Más Fácily Por La Pedagogía. 2021. «Iot4All: Los desafíos que debe enfrentar la IoT». *Think Big*. Recuperado 29 de noviembre de 2022 (https://empresas.blogthinkbig.com/iot4all-los-desafios-que-debe-enfrentar-la-iot/).

vuejs. s. f. «VueJS + Vuetify: Crea interfaces de usuario reutilizables | OpenWebinars». Recuperado 25 de noviembre de 2022 (https://openwebinars.net/blog/vuejs-vuetify-crea-interfaces-de-usuario-reutilizables/).

W3Schools. s. f. «Django Tutorial». Recuperado 25 de noviembre de 2022 (https://www.w3schools.com/django/).

W3Schools.w. s. f. «Python Tutorial». Recuperado 25 de noviembre de 2022 (https://www.w3schools.com/python/).

Anexos

CU2	**Gestionar Propietario**
Propósito:	Crear, modificar y eliminar Propietario.
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita ingresar un nuevo Propietario al sistema, o se presenta alguna modificación con los datos de los ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de propietarios .
Postcondiciones:	El sistema ha gestionado un Propietario.
Sección: "Insertar Propietario"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar un nuevo propietario.
3. El Usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema inserta los datos del nuevo propietario. Culmina el caso de uso.
Sección: "Modificar Propietario"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción	2. El sistema devuelve los datos en un formulario

EDITAR de su interfaz de trabajo.	de la entidad seleccionada.
3. El Usuario actualiza los nuevos datos en el formulario.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema actualiza los datos en la base de datos. 6. El sistema actualiza el listado de los propietarios. Culmina el caso de uso.
Sección: "Eliminar Propietario"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar Propietario"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El especialista accede al apartado de Propietarios de su interfaz de trabajo.	2. El sistema muestra las entidades existentes. Culmina el caso de uso
Prioridad	Alto

Tabla 2: Descripción del caso de uso <Gestionar Propietario>

CU3	Gestionar Lugar
Propósito:	Crear, modificar y eliminar Lugar.
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita ingresar un nuevo Lugar al sistema, o se presenta alguna modificación con los datos de los ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de Lugares.
Postcondiciones:	El sistema ha gestionado un Lugar.
Sección: "Insertar Lugar"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar un nuevo lugar.
3. El Usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema inserta los datos del nuevo lugar. Culmina el caso de uso.
Sección: "Modificar Lugar"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos	4. El sistema comprueba que no exista error en

datos en el formulario.	los datos que se entraron. 5. El sistema actualiza los datos en la base de datos. 6. El sistema actualiza el listado de los lugares. Culmina el caso de uso.
Sección: "Eliminar Lugar"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar Lugar"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El especialista accede al apartado de Lugar de su interfaz de trabajo.	2. El sistema muestra los lugares existentes. Culmina el caso de uso

Tabla 3: Descripción del caso de uso <Gestionar Lugar>

CU4	**Gestionar Arquitectura del Nodo Sensor**
Propósito:	Crear, modificar y eliminar Arquitectura del nodo Sensor.
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita

	ingresar una nueva Arquitectura al sistema, o se presenta alguna modificación con los datos de los ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de Arquitecturas.
Postcondiciones:	El sistema ha gestionado una Arquitectura del nodo sensor.
Sección: "Insertar Arquitectura"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar una nueva arquitectura.
3. El Usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema inserta los datos de la nueva arquitectura. Culmina el caso de uso.
Sección: "Modificar Arquitectura"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos datos en el formulario.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema actualiza los datos en la base de

	datos. 6. El sistema actualiza el listado de las arquitecturas. Culmina el caso de uso.
Sección: "Eliminar Arquitectura"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar Arquitectura"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El especialista accede al apartado de Arquitectura de Nodos sensores de su interfaz de trabajo.	2. El sistema muestra las arquitecturas existentes. Culmina el caso de uso

Tabla 4: Descripción del caso de uso <Gestión de Arquitectura de Nodos sensores>

CU5	Gestionar Marca
Propósito:	Crear, modificar y eliminar Marca.
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita ingresar una nueva Marca al sistema, o se presenta alguna modificación con los datos de los ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados.
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de marcas.
Postcondiciones:	El sistema ha gestionado una marca.
Sección: "Insertar Marca"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar una nueva marca.
3. El Usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema inserta los datos de la nueva marca. Culmina el caso de uso.
Sección: "Modificar Marca"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos	4. El sistema comprueba que no exista error en

datos en el formulario.	los datos que se entraron. 5. El sistema actualiza los datos en la base de datos. 6. El sistema actualiza el listado de las marcas. Culmina el caso de uso
Sección: "Eliminar Marca"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar Marca"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El especialista accede al apartado de Marca de su interfaz de trabajo.	2. El sistema muestra las marcas existentes. Culmina el caso de uso

Tabla 5: Descripción de caso de uso <Gestión Marcas>

CU6	**Gestionar Interfaz de red**
Propósito:	Crear, modificar y eliminar Interfaz de red.
Actores:	Usuario

Resumen:	El caso de uso inicia cuando el usuario necesita ingresar una nueva Interfaz de red al sistema, o se presenta alguna modificación con los datos de las ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados.
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de Interfaces de redes.
Postcondiciones:	El sistema ha gestionado una Interfaz de red.
Sección: "Insertar Interfaz de red"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar una nueva interfaz de red.
3. El Usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema inserta los datos de la nueva interfaz de red. Culmina el caso de uso.
Sección: "Modificar Interfaz de red"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos datos en el formulario.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema actualiza los datos en la base de

	datos. 6. El sistema actualiza el listado de las Interfaces de red. Culmina el caso de uso.
Sección: "Eliminar Interfaz de red"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar Interfaz de red"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El especialista accede al apartado de Interfaz de red de su interfaz de trabajo.	2. El sistema muestra los interfaces existentes. Culmina el caso de uso

Tabla 6: Descripción de caso de uso <Gestión Interfaz de red>

CU7	Gestionar Interfaz de red cableada
Propósito:	Crear, modificar y eliminar Interfaz de red cableada.
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita ingresar una nueva Interfaz de red cableada al sistema, o se presenta alguna modificación con los datos de los ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados.
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de Interfaces de redes.
Postcondiciones:	El sistema ha gestionado una interfaz de red cableada.
Sección: "Insertar Interfaz de red cableada"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar una nueva interfaz de red cableada.
3. El Usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema inserta los datos de la nueva interfaz de red cableada. Culmina el caso de uso.
Sección: "Modificar Interfaz de red cableada"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**

1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos datos en el formulario.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema actualiza los datos en la base de datos. 6. El sistema actualiza el listado de las interfaces cableadas. Culmina el caso de uso.
Sección: "Eliminar Interfaz de red cableada"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar Interfaz de red cableada"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El especialista accede al apartado de Interfaz de red cableada de su interfaz de trabajo.	2. El sistema muestra las interfaces cableadas existentes. Culmina el caso de uso

Tabla 7: Descripción del caso de uso <Gestión de interfaz de red cableada>

CU8	Gestionar Interfaz de red inalámbrica
Propósito:	Crear, modificar y eliminar interfaz de red inalámbrica.
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita ingresar una nueva interfaz de red inalámbrica al sistema, o se presenta alguna modificación con los datos de los ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de interfaces de redes inalámbricas.
Postcondiciones:	El sistema ha gestionado una Interfaz de red Inalámbrica.
Sección: "Insertar Interfaz de red inalámbrica"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar una nueva Interfaz de red inalámbrica.
3. El Usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema inserta los datos de la nueva interfaz de red inalámbrica.

	Culmina el caso de uso.
Sección: "Modificar Interfaz de red inalámbrica"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos datos en el formulario.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema actualiza los datos en la base de datos. 6. El sistema actualiza el listado de las interfaces inalámbricas. Culmina el caso de uso.
Sección: "Eliminar Interfaz inalámbrica"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar Interfaz de red Inalámbrica"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El especialista accede al	2. El sistema muestra las Interfaces inalámbricas

apartado de Interfaz de red inalámbrica de su interfaz de trabajo.	existentes. Culmina el caso de uso

Tabla 8: Descripción del caso de uso <Gestión de Interfaces de redes inalámbricas>

CU9	**Gestionar Fuente de alimentación**
Propósito:	Crear, modificar y eliminar Fuente de alimentación.
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita ingresar una nueva Fuente de alimentación al sistema, o se presenta alguna modificación con los datos de los ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de Fuente de alimentación.
Postcondiciones:	El sistema ha gestionado una Fuente de alimentación.
Sección: "Insertar Fuente de alimentación"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar una nueva Fuente de alimentación.
3. El Usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron.

	5. El sistema inserta los datos de la nueva fuente de alimentación. Culmina el caso de uso.
Sección: "Modificar Fuente de alimentación"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos datos en el formulario.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema actualiza los datos en la base de datos. 6. El sistema actualiza el listado de las fuentes de alimentación. Culmina el caso de uso.
Sección: "Eliminar Fuente de alimentación"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar Fuente de alimentación"	
Flujo Normal	

Acción del actor	Respuesta del sistema
1. El especialista accede al apartado de Fuente de alimentación de su interfaz de trabajo.	2. El sistema muestra las Fuentes de alimentación existentes. Culmina el caso de uso

Tabla 9: Descripción del caso de uso <Gestión de Fuente de alimentación>

CU10	Gestionar Fuente de alimentación de línea
Propósito:	Crear, modificar y eliminar Fuente de alimentación de línea.
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita ingresar una nueva Fuente de alimentación de línea al sistema, o se presenta alguna modificación con los datos de los ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de una Fuente de alimentación de línea.
Postcondiciones:	El sistema ha gestionado una Fuente de alimentación de línea.
Sección: "Insertar Fuente de alimentación de línea"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar una nueva Fuente de alimentación de línea.
3. El Usuario inserta los datos de	4. El sistema comprueba que no exista error en

la nueva entidad.	los datos que se entraron. 5. El sistema inserta los datos de la nueva fuente de alimentación. Culmina el caso de uso.
Sección: "Modificar Fuente de alimentación de línea"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos datos en el formulario.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema actualiza los datos en la base de datos. 6. El sistema actualiza el listado de las fuentes de alimentación. Culmina el caso de uso.
Sección: "Eliminar Fuente de alimentación de línea"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar Fuente de alimentación de línea"	

Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El usuario accede al apartado de Fuente de alimentación de su interfaz de trabajo.	2. El sistema muestra las Fuentes de alimentación existentes. Culmina el caso de uso

Tabla 10: Descripción del caso de uso <Gestionar Fuente de alimentación de línea>

CU11	**Gestionar Fuente de alimentación de batería**
Propósito:	Crear, modificar y eliminar Fuente de alimentación de batería.
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita ingresar una nueva Fuente de alimentación de batería al sistema, o se presenta alguna modificación con los datos de los ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados.
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de una Fuente de alimentación de batería.
Postcondiciones:	El sistema ha gestionado una Fuente de alimentación de batería.
Sección: "Insertar Fuente de alimentación de batería"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de	2. El sistema muestra un formulario para insertar una nueva Fuente de alimentación de línea.

trabajo.	
3. El Usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema inserta los datos de la nueva fuente de alimentación. Culmina el caso de uso.
Sección: "Modificar Fuente de alimentación de batería"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos datos en el formulario.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema actualiza los datos en la base de datos. 6. El sistema actualiza el listado de las fuentes de alimentación. Culmina el caso de uso.
Sección: "Eliminar Fuente de alimentación de batería"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada.

	Culmina el caso de uso
Sección: "Mostrar Fuente de alimentación de batería"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El usuario accede al apartado de Fuente de alimentación de su interfaz de trabajo.	2. El sistema muestra las Fuentes de alimentación existentes. Culmina el caso de uso

Tabla 11: Descripción del caso de uso <Gestionar Fuente de alimentación de batería>

CU12	**Gestionar Placa**
Propósito:	Crear, modificar y eliminar Placa
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita ingresar una nueva Placa, o se presenta alguna modificación con los datos de los ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de una Placa.
Postcondiciones:	El sistema ha gestionado una Placa.
Sección: "Insertar Placa"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar una nueva Placa.

3. El Usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema inserta los datos de la Placa. Culmina el caso de uso.
Sección: "Modificar Placa"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos datos en el formulario.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema actualiza los datos en la base de datos. 6. El sistema actualiza el listado de las Placas. Culmina el caso de uso.
Sección: "Eliminar Placa"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar Placa"	
Flujo Normal	

Acción del actor	Respuesta del sistema
1. El Usuario accede al apartado de Placas en la interfaz de trabajo.	2. El sistema muestra las Placas existentes. Culmina el caso de uso

Tabla 12: Descripción del caso de uso <<Gestionar Placa>>

CU13	Gestionar Sensor
Propósito:	Crear, modificar y eliminar Sensor
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita ingresar un nuevo Sensor, o se presenta alguna modificación con los datos de los ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados.
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de una Placa.
Postcondiciones:	El sistema ha gestionado un sensor
Sección: "Insertar Sensor"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar una nueva Sensor.
3. El Usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron.

	5. El sistema inserta los datos de la Sensor. Culmina el caso de uso.
Sección: "Modificar Sensor"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos datos en el formulario.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema actualiza los datos en la base de datos. 6. El sistema actualiza el listado de los Sensores. Culmina el caso de uso.
Sección: "Eliminar Sensores"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar Sensor"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede al apartado	2. El sistema muestra los Sensores existentes.

de Sensores en la interfaz de trabajo.	Culmina el caso de uso

Tabla 13: Descripción del caso de uso <Gestionar Sensor>

CU14	**Gestionar Mcu**
Propósito:	Crear, modificar y eliminar Mcu
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita ingresar un nuevo Mcu, o se presenta alguna modificación con los datos de los ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados.
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de un Mcu.
Postcondiciones:	El sistema ha gestionado una Mcu.
Sección: "Insertar Mcu"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar un nuevo Mcu.
3. El Usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema inserta los datos del Mcu. Culmina el caso de uso.
Sección: "Modificar Mcu"	

Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos datos en el formulario.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema actualiza los datos en la base de datos. 6. El sistema actualiza el listado de los Mcu. Culmina el caso de uso.
Sección: "Eliminar Mcu"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar Mcu"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede al apartado de Mcu de trabajo.	2. El sistema muestra los Mcu existentes. Culmina el caso de uso

Tabla 14: Descripción del caso de uso <Gestionar MCU>

CU15	**Gestionar Magnitud variable**

Propósito:	Crear, modificar y eliminar Magnitud Variable
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita ingresar una nueva Magnitud variable, o se presenta alguna modificación con los datos de los ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados.
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de una Magnitud Variable.
Postcondiciones:	El sistema ha gestionado una Magnitud Variable.
Sección: "Insertar Magnitud Variable"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar una nueva magnitud Variable.
3. El Usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema inserta los datos de la Magnitud Variable. Culmina el caso de uso.
Sección: "Modificar Magnitud Variable"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos	4. El sistema comprueba que no exista error en

datos en el formulario.	los datos que se entraron. 5. El sistema actualiza los datos en la base de datos. 6. El sistema actualiza el listado de las Magnitudes Variables. Culmina el caso de uso.
Sección: "Eliminar Magnitud Variable"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar Magnitud Variable"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede al apartado de Magnitudes variables de trabajo.	2. El sistema muestra las magnitudes Variables existentes. Culmina el caso de uso

Tabla 15: Descripción del caso de uso <<Gestionar Magnitud Variable>>

CU16	**Gestionar Medición**
Propósito:	Crear, modificar y eliminar Medición

Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario necesita ingresar una nueva Medición, o se presenta alguna modificación con los datos de los ya existentes, así como la eliminación de alguno. El caso termina cuando el sistema guarda los cambios realizados.
Referencias:	RF-1
Responsabilidades:	Mantener la integridad de la información.
Precondiciones:	El sistema está disponible para la gestión de una Medición.
Postcondiciones:	El sistema ha gestionado una Medición.
Sección: "Insertar Medición"	
Flujo normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción INSERTAR de su interfaz de trabajo.	2. El sistema muestra un formulario para insertar una nueva Medición.
3. El Usuario inserta los datos de la nueva entidad.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema inserta los datos de la Medición. Culmina el caso de uso.
Sección: "Modificar Medición"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción EDITAR de su interfaz de trabajo.	2. El sistema devuelve los datos en un formulario de la entidad seleccionada.
3. El Usuario actualiza los nuevos datos en el formulario.	4. El sistema comprueba que no exista error en los datos que se entraron. 5. El sistema actualiza los datos en la base de

	datos. 6. El sistema actualiza el listado de las mediciones. Culmina el caso de uso.
Sección: "Eliminar Medición"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede a la opción ELIMINAR de su interfaz de trabajo.	2. El sistema muestra una ventana para confirmar la operación de eliminar.
3. El Usuario confirma.	4. El sistema verifica que la entidad no haya sido utilizada. 5. En caso de no estar utilizada elimina la entidad. 6. El sistema actualiza el listado de las entidades. 7. El sistema confirma entidad eliminada. Culmina el caso de uso
Sección: "Mostrar Medición"	
Flujo Normal	
Acción del actor	**Respuesta del sistema**
1. El Usuario accede al apartado de Medición de su interfaz de trabajo.	2. El sistema muestra las mediciones existentes. Culmina el caso de uso

Tabla 16: Descripción del caso de uso <Gestionar Medición>

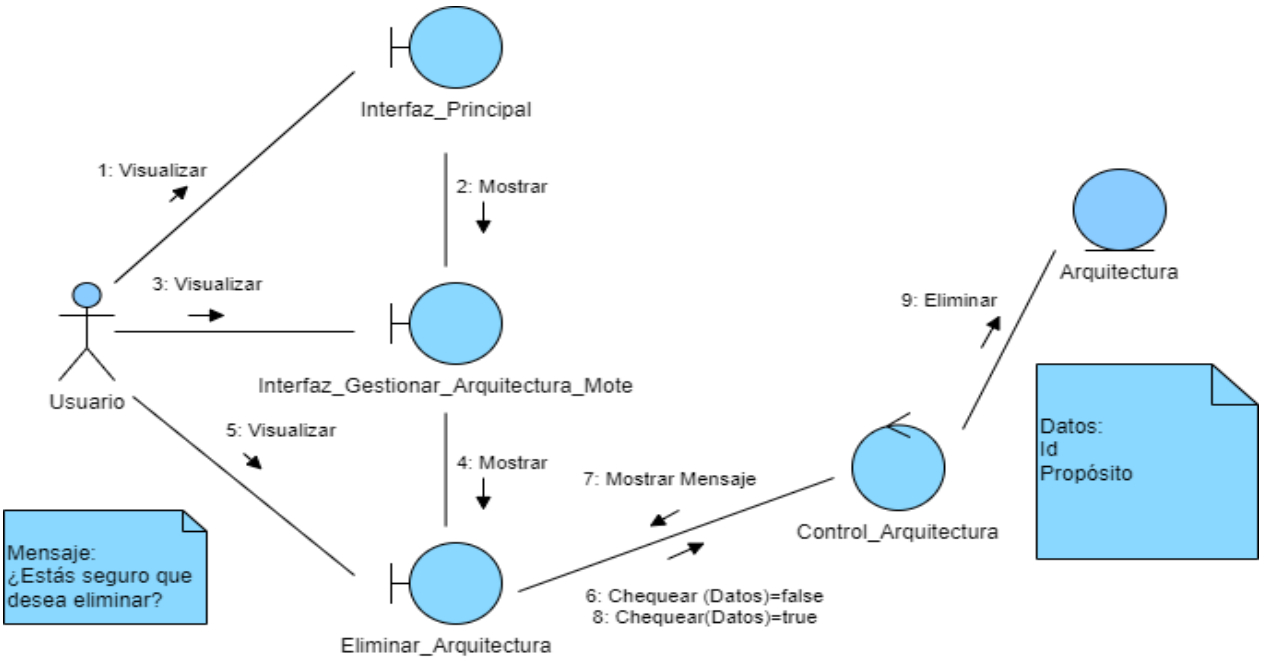

Ilustración 27: Diagrama de colaboración <Eliminar Arquitectura del nodo sensor>

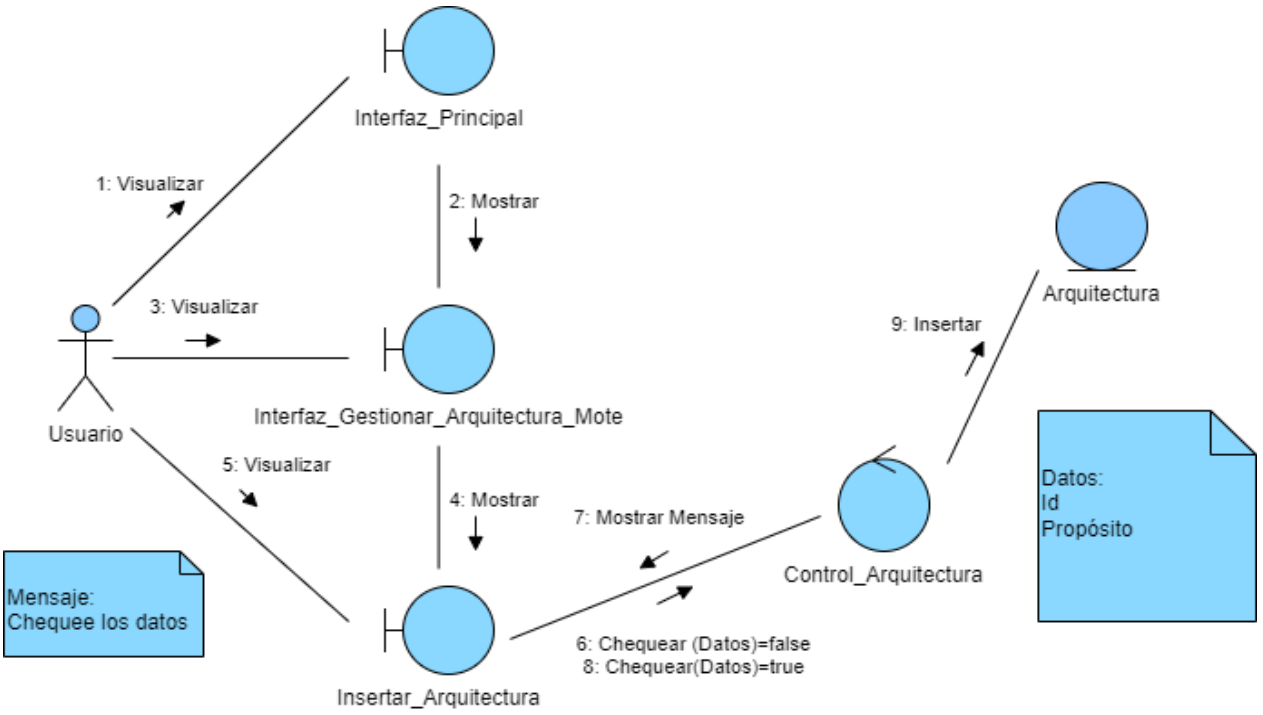

Ilustración 28: Diagrama de colaboración <Insertar Arquitectura del nodo sensor>

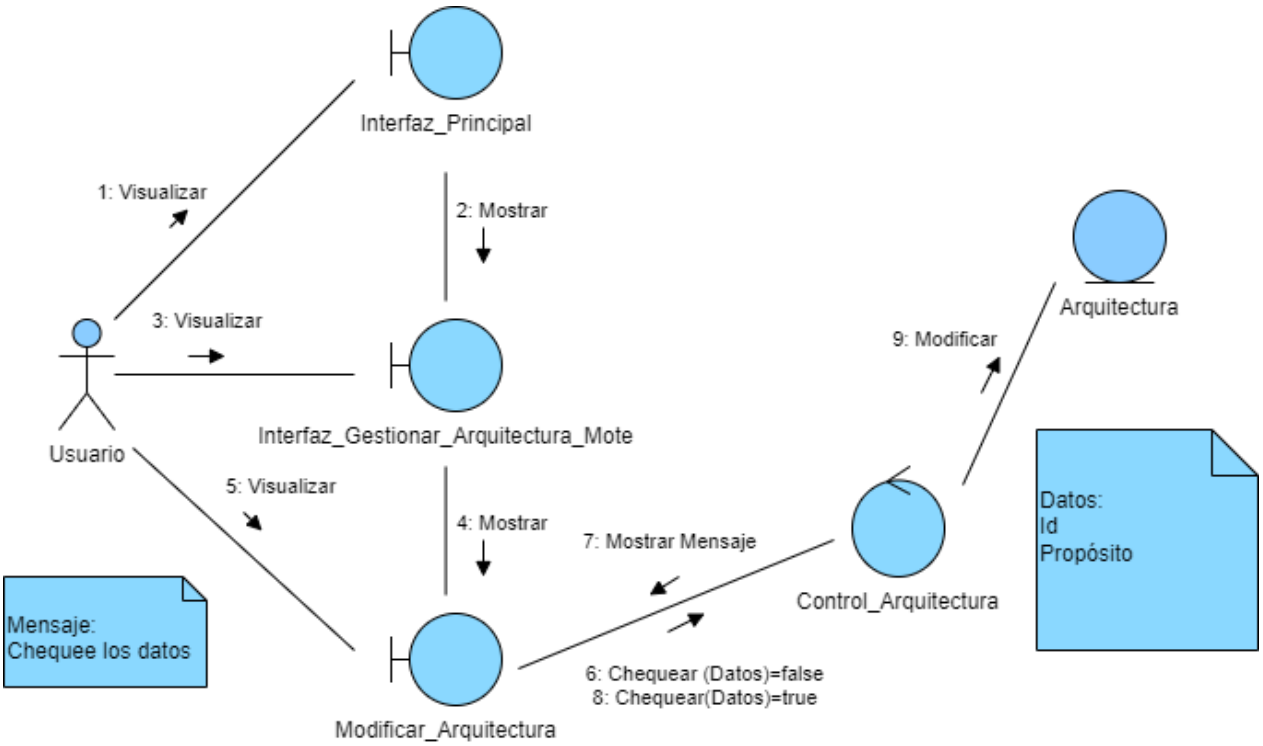

Ilustración 29: Diagrama de colaboración <Modificar Arquitectura del nodo sensor>

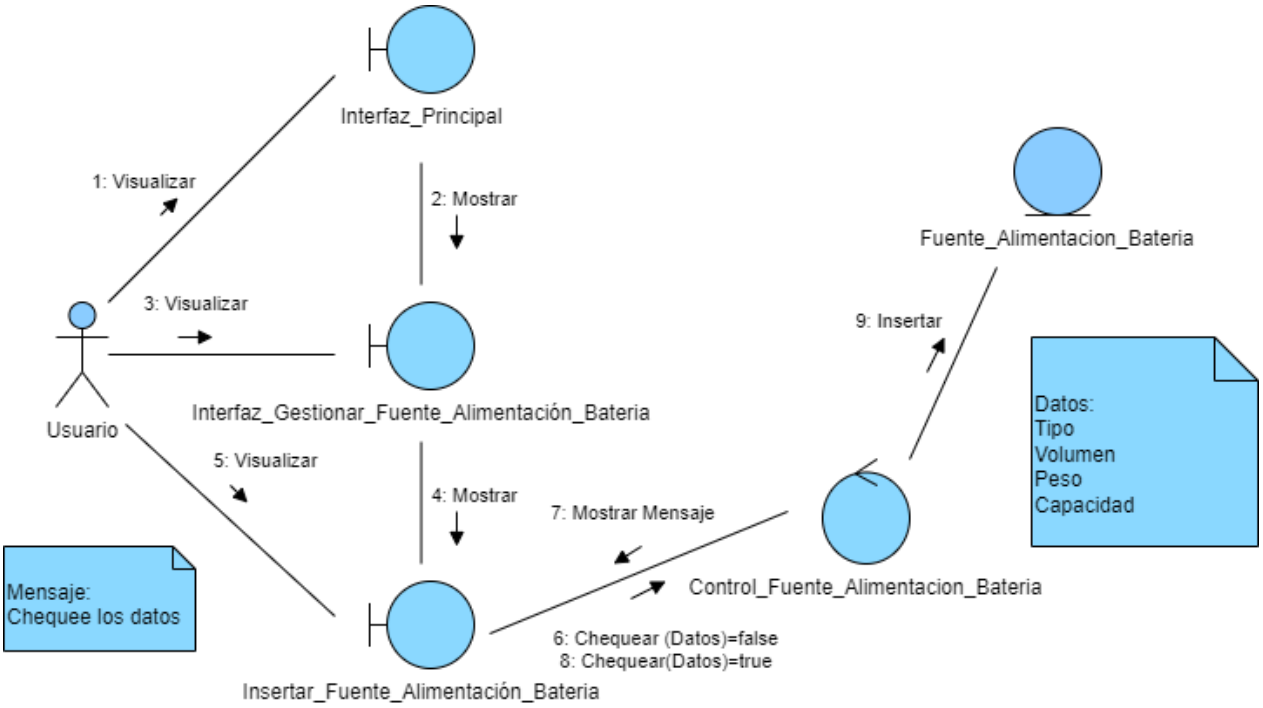

Ilustración 30: Diagrama de colaboración <Insertar Fuente de alimentación Batería>

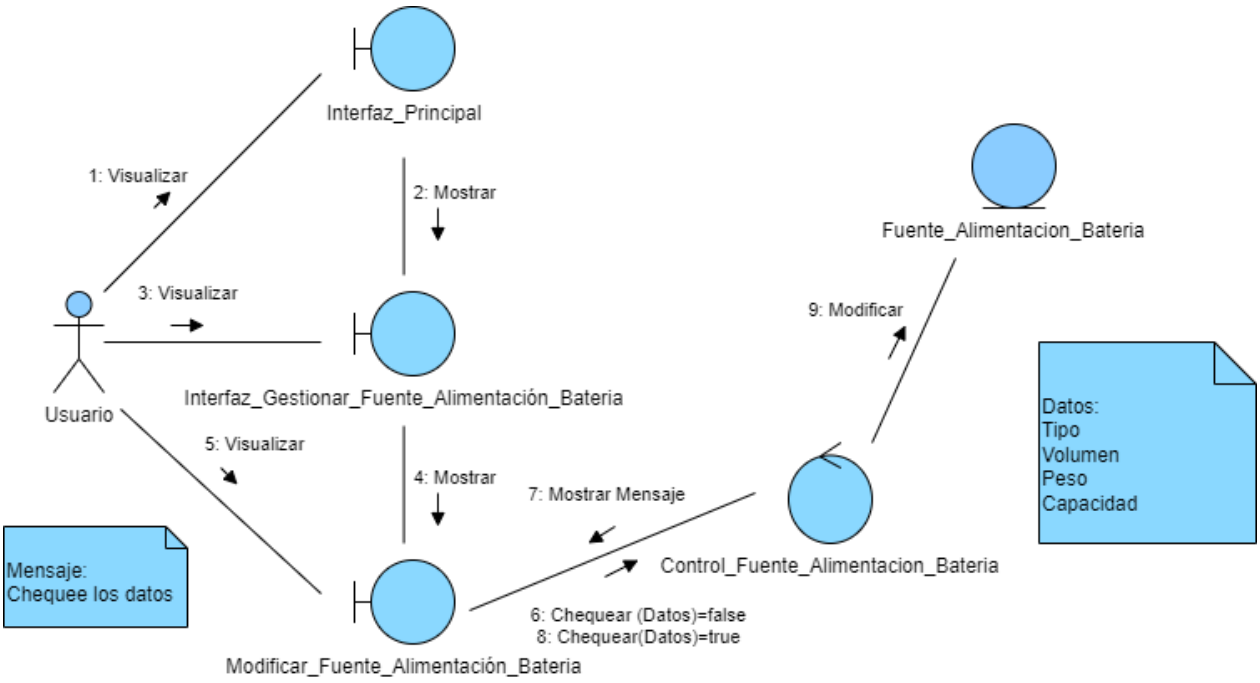

Ilustración 31: Diagrama de colaboración <Modificar Fuente de alimentación Batería>

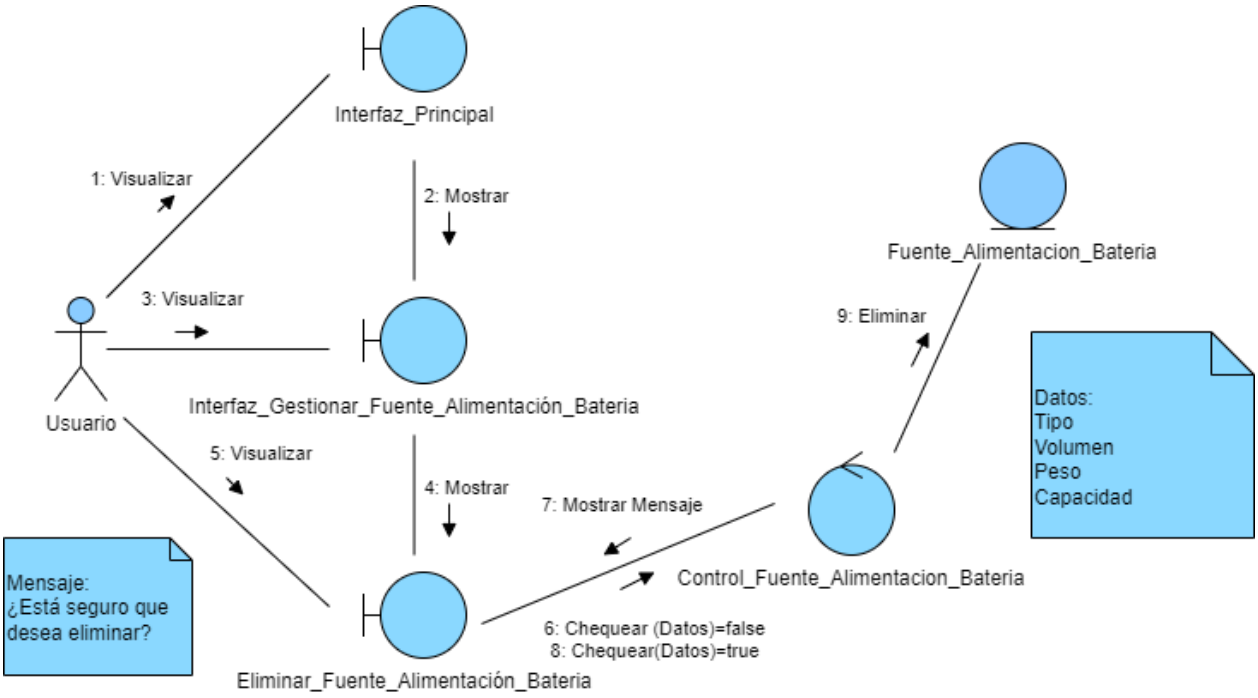

Ilustración 32: Diagrama de colaboración <Eliminar Fuente de alimentación Batería>

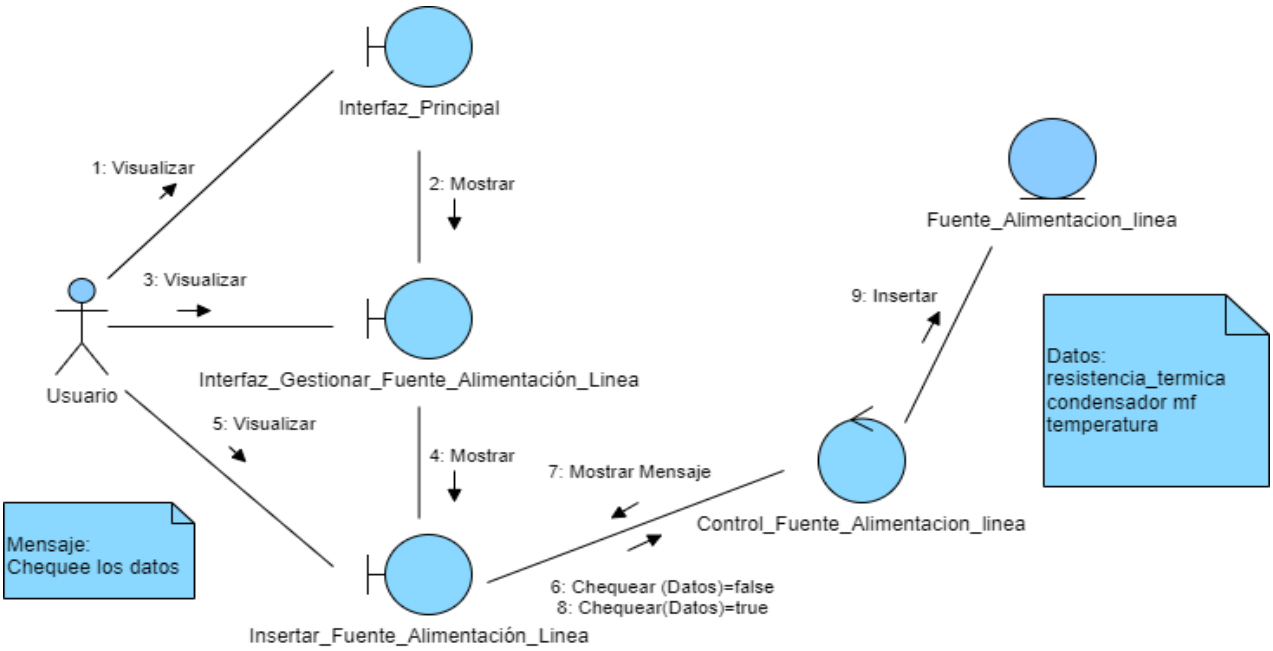

Ilustración 33: Diagrama de colaboración <Insertar Fuente de alimentación Línea>

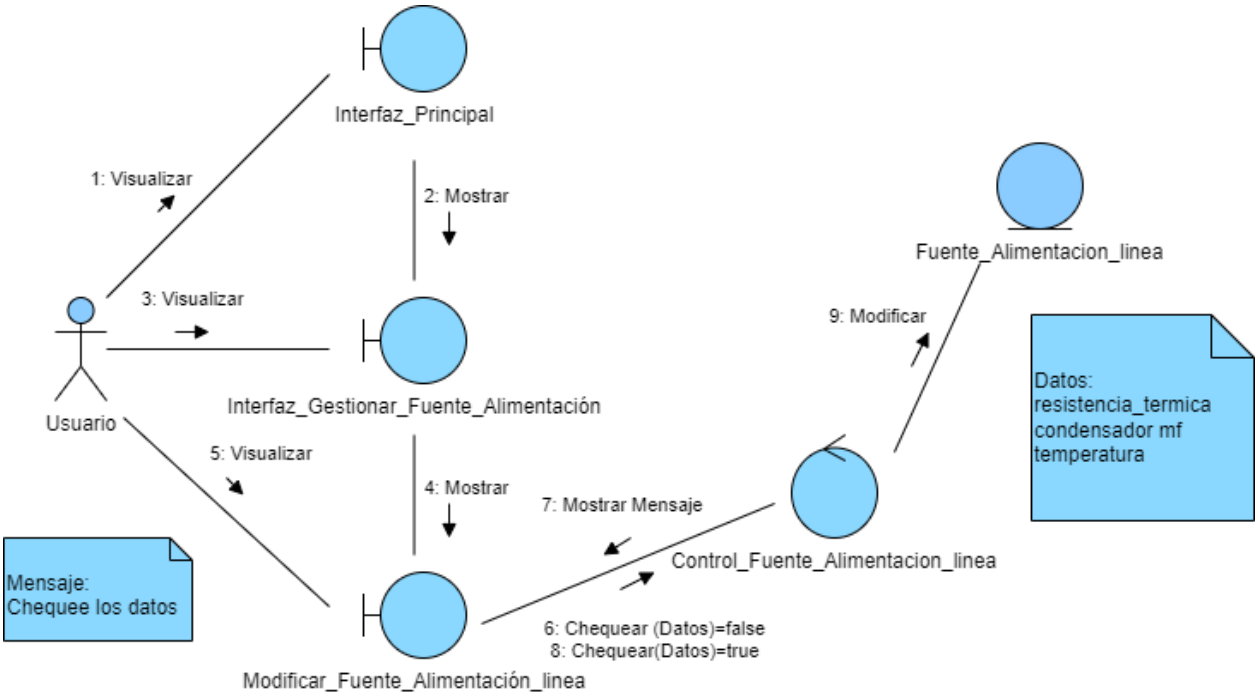

Ilustración 34: Diagrama de colaboración <Modificar Fuente de alimentación Línea>

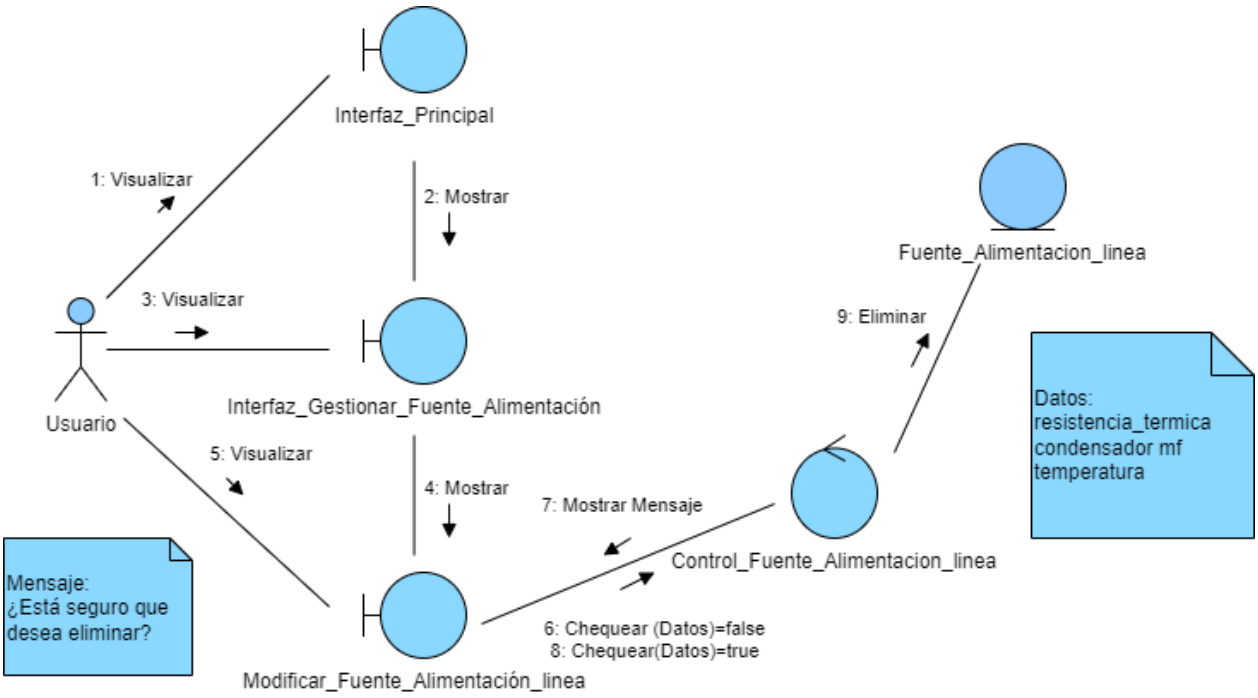

Ilustración 35: Diagrama de colaboración <Eliminar Fuente de alimentación Línea>

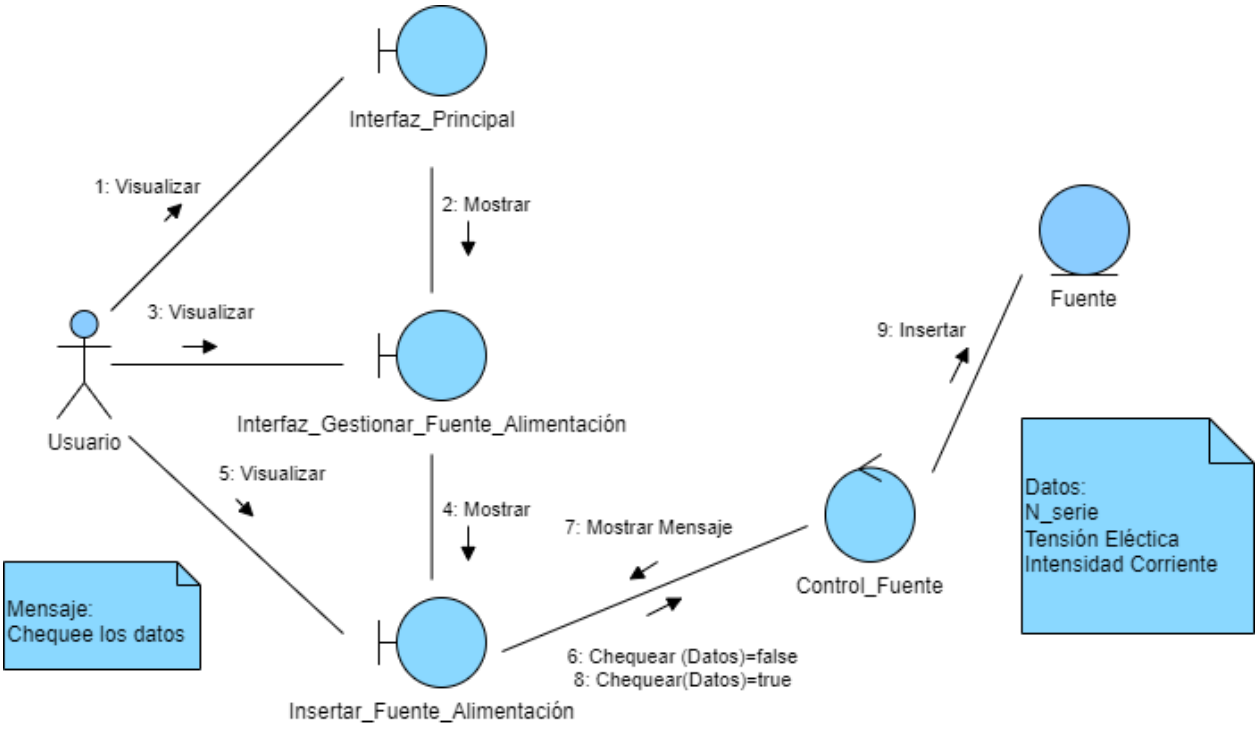

Ilustración 36: Diagrama de colaboración <Insertar Fuente de alimentación>

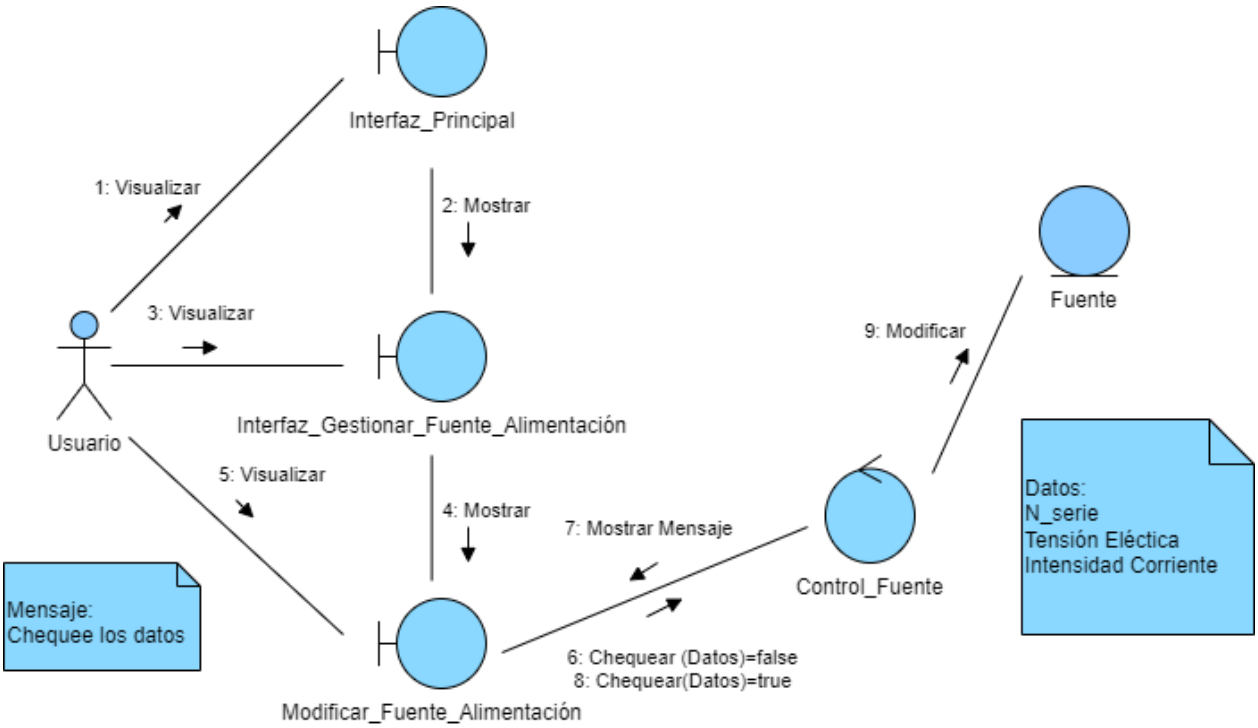

Ilustración 37: Diagrama de colaboración <Modificar Fuente de alimentación>

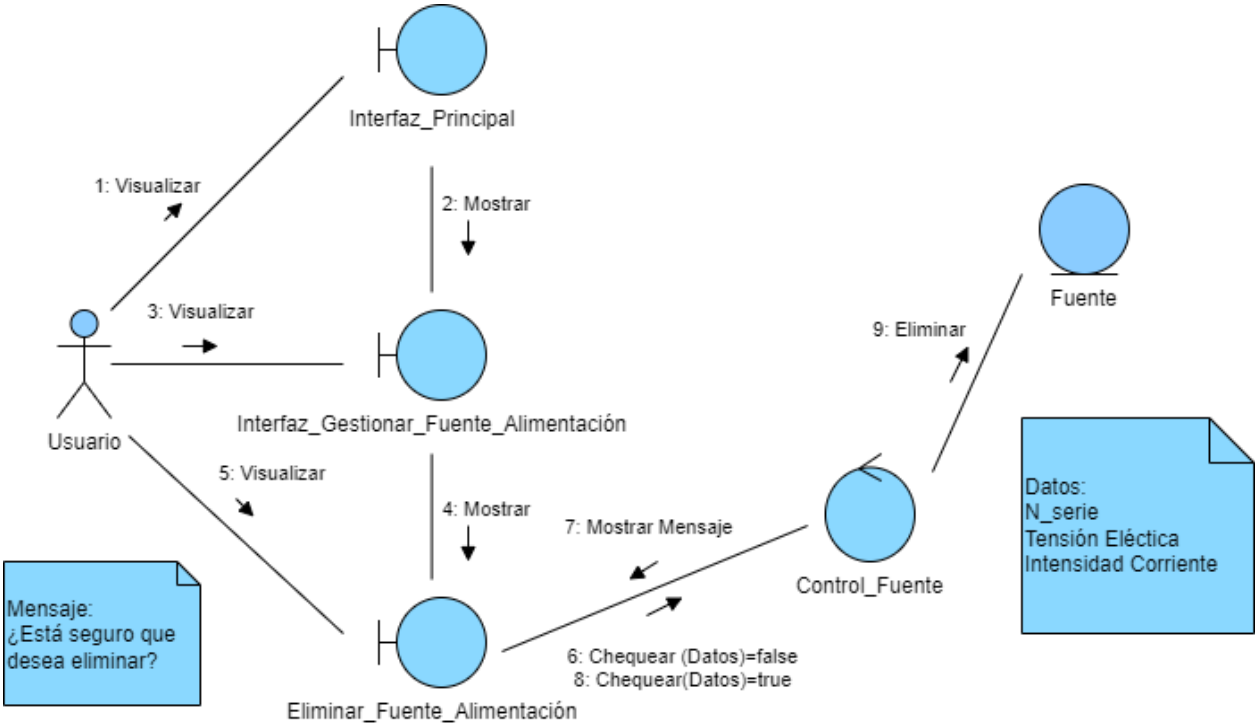

Ilustración 38: Diagrama de colaboración <Eliminar Fuente de alimentación>

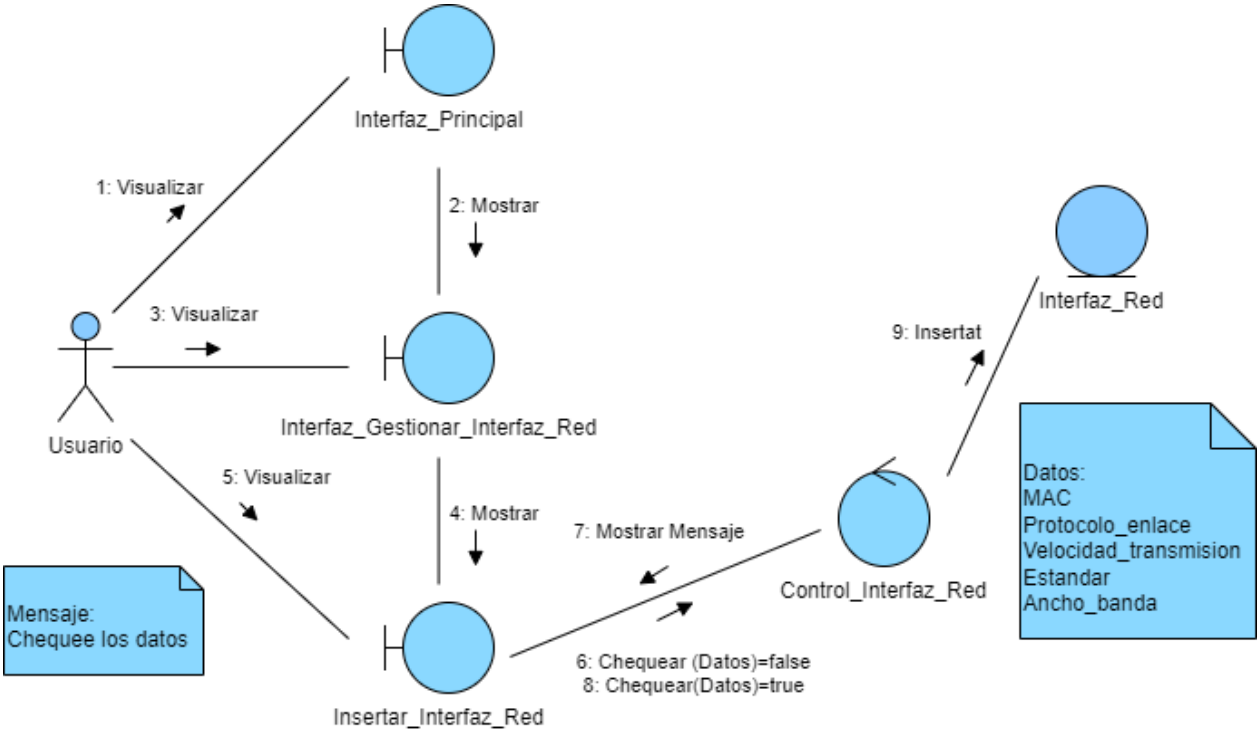

Ilustración 39: Diagrama de colaboración <Insertar Interfaz de red>

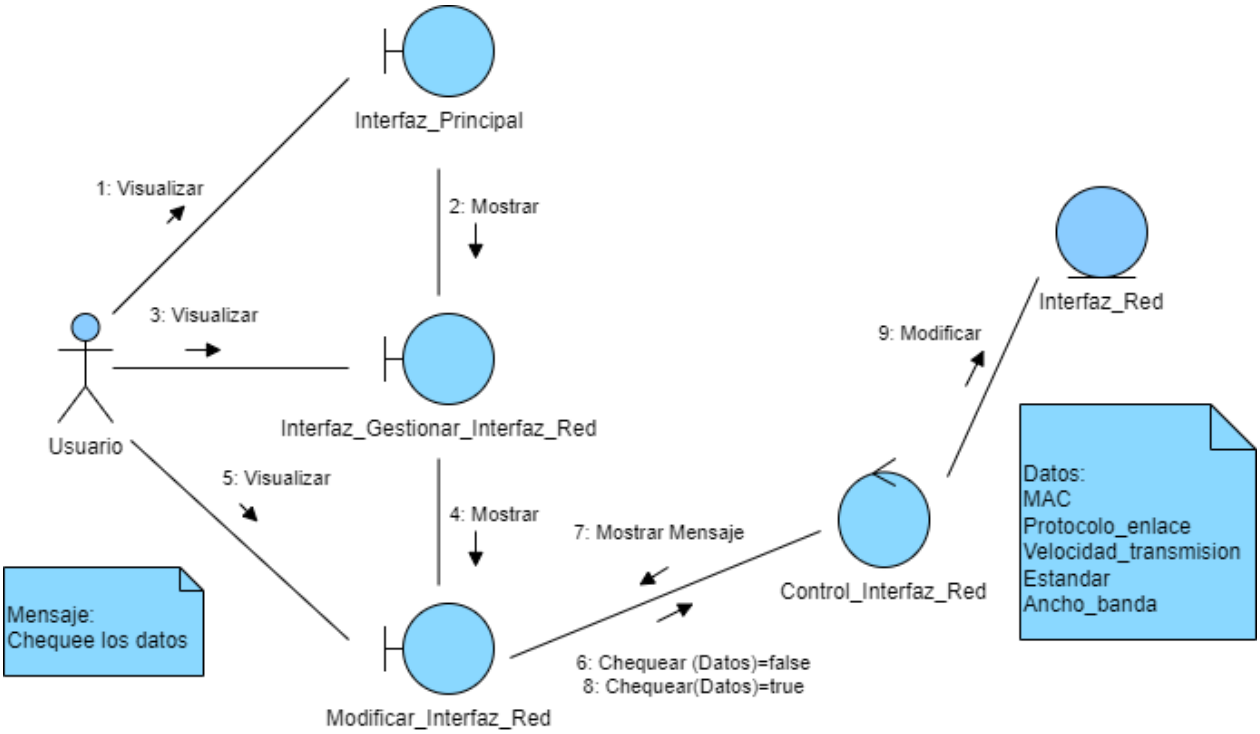

Ilustración 40: Diagrama de colaboración <Modificar Interfaz de red>

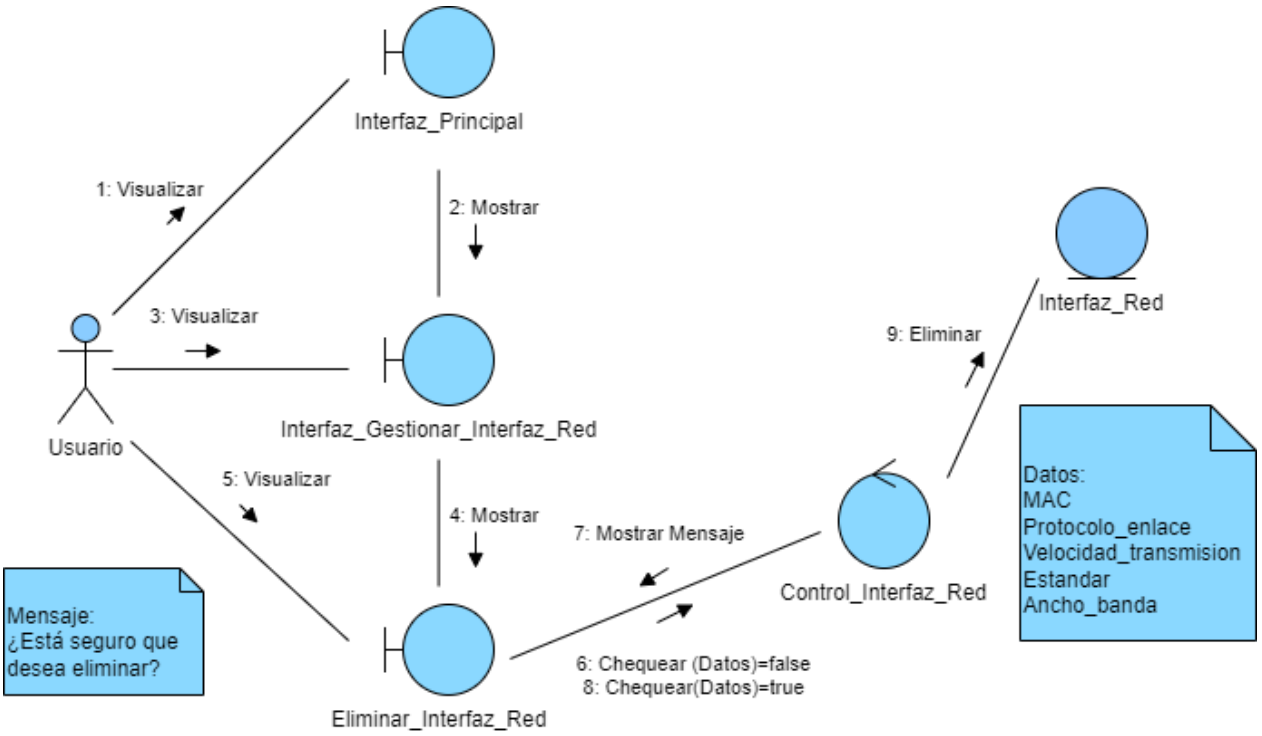

Ilustración 41: Diagrama de colaboración <Eliminar Interfaz de red>

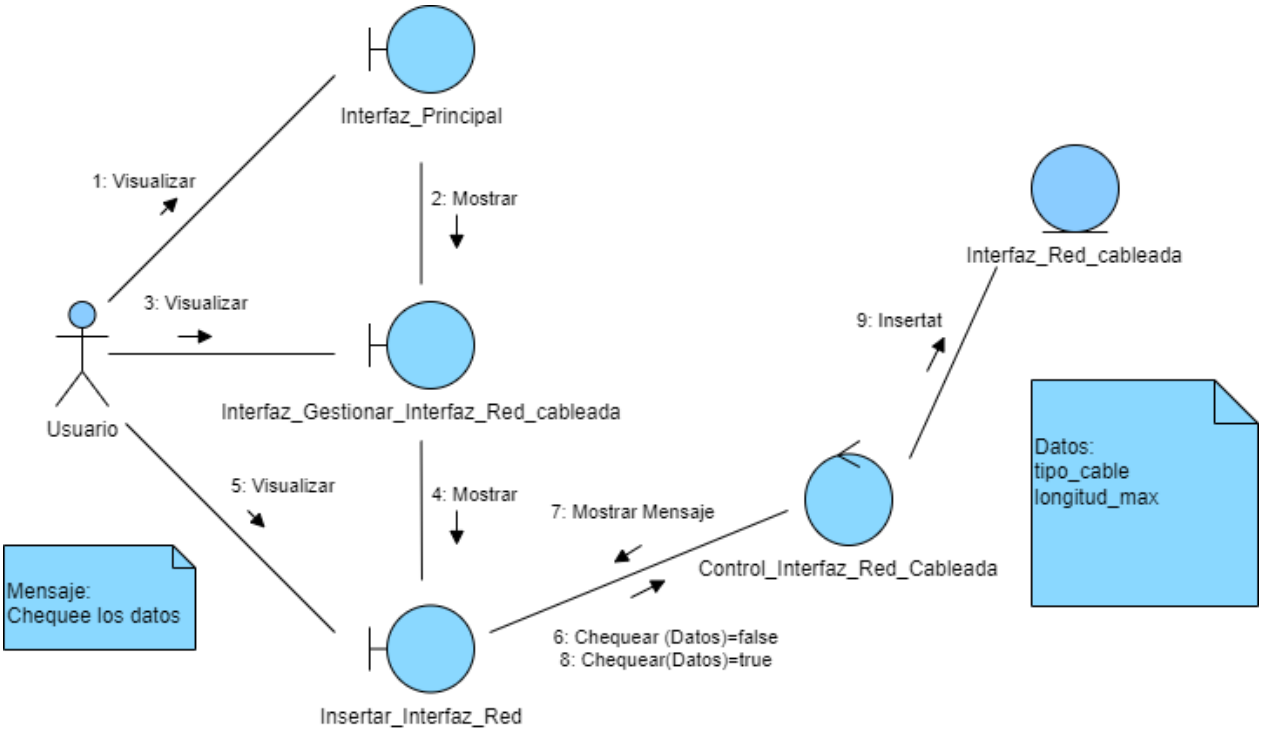

Ilustración 42: Diagrama de colaboración <Insertar Interfaz de red cableada>

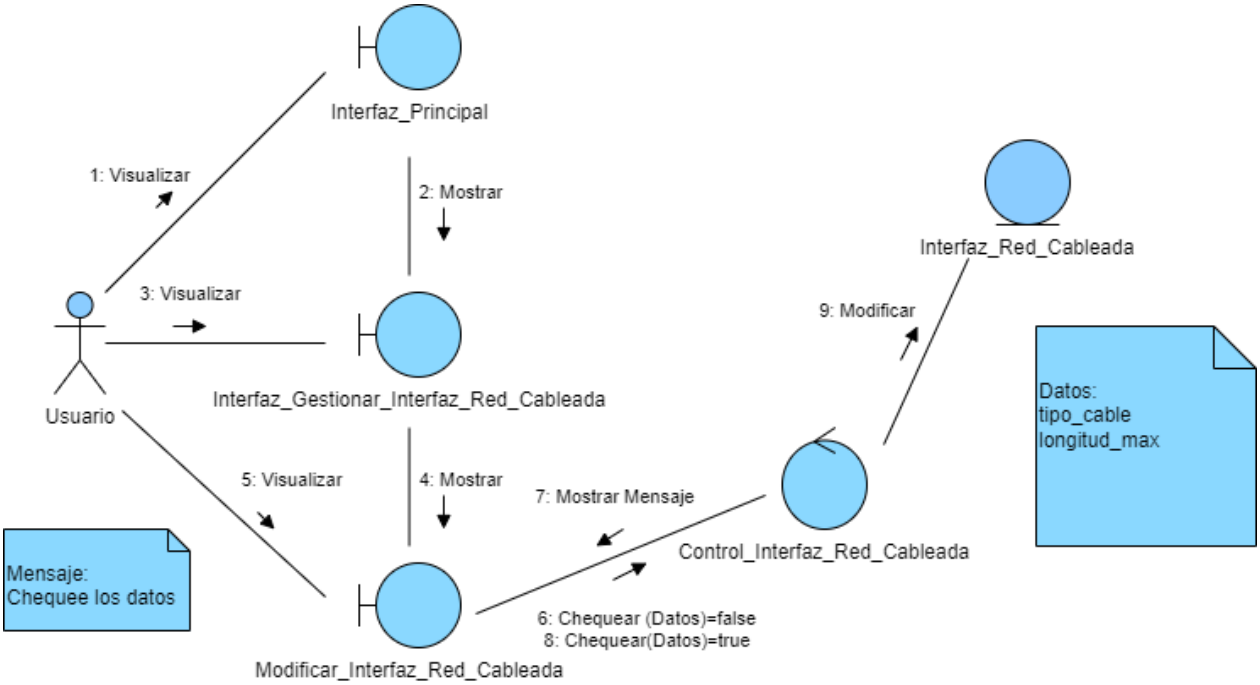

Ilustración 43: Diagrama de colaboración <Modificar Interfaz de red cableada>

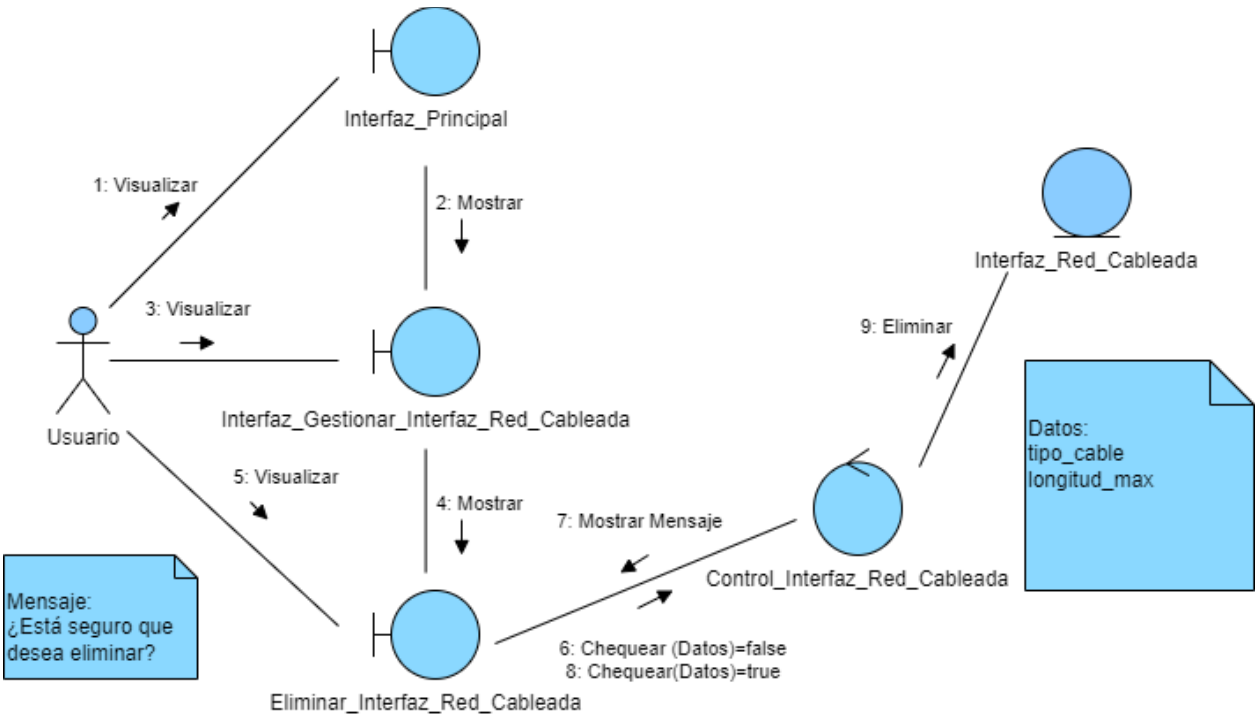

Ilustración 44: Diagrama de colaboración <Eliminar Interfaz de red cableada>

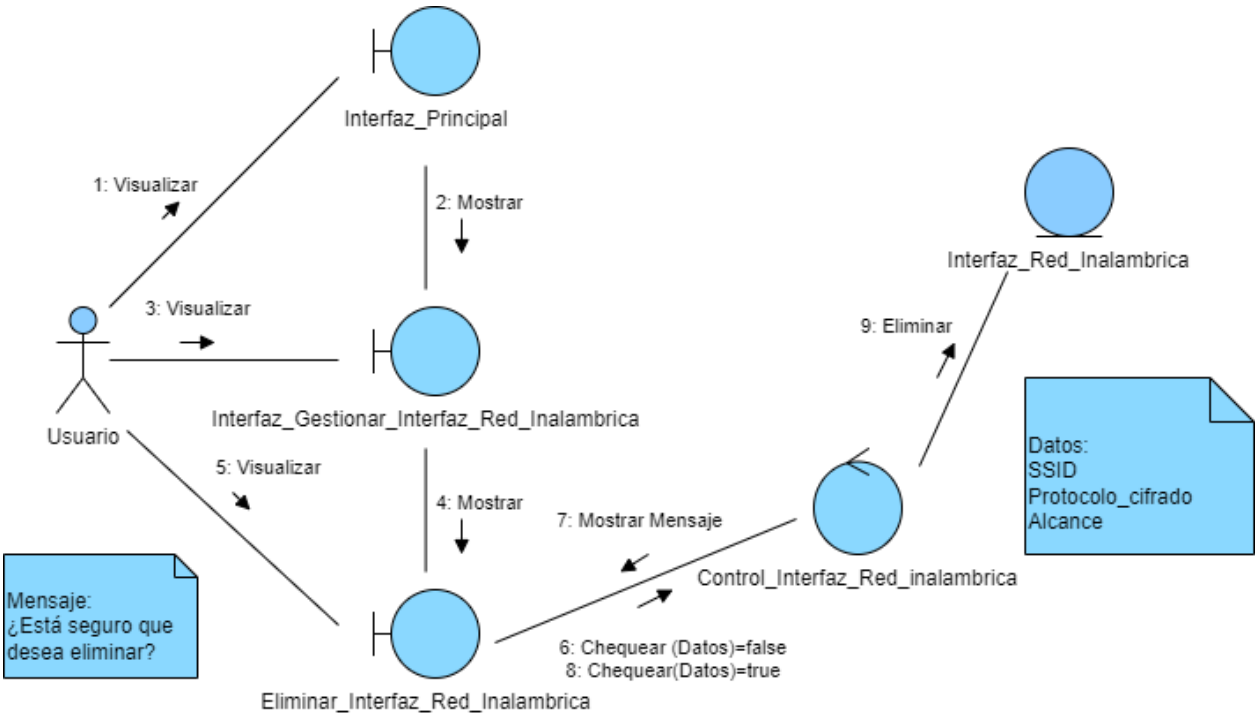

Ilustración 45: Diagrama de colaboración <Eliminar Interfaz de red inalámbrica>

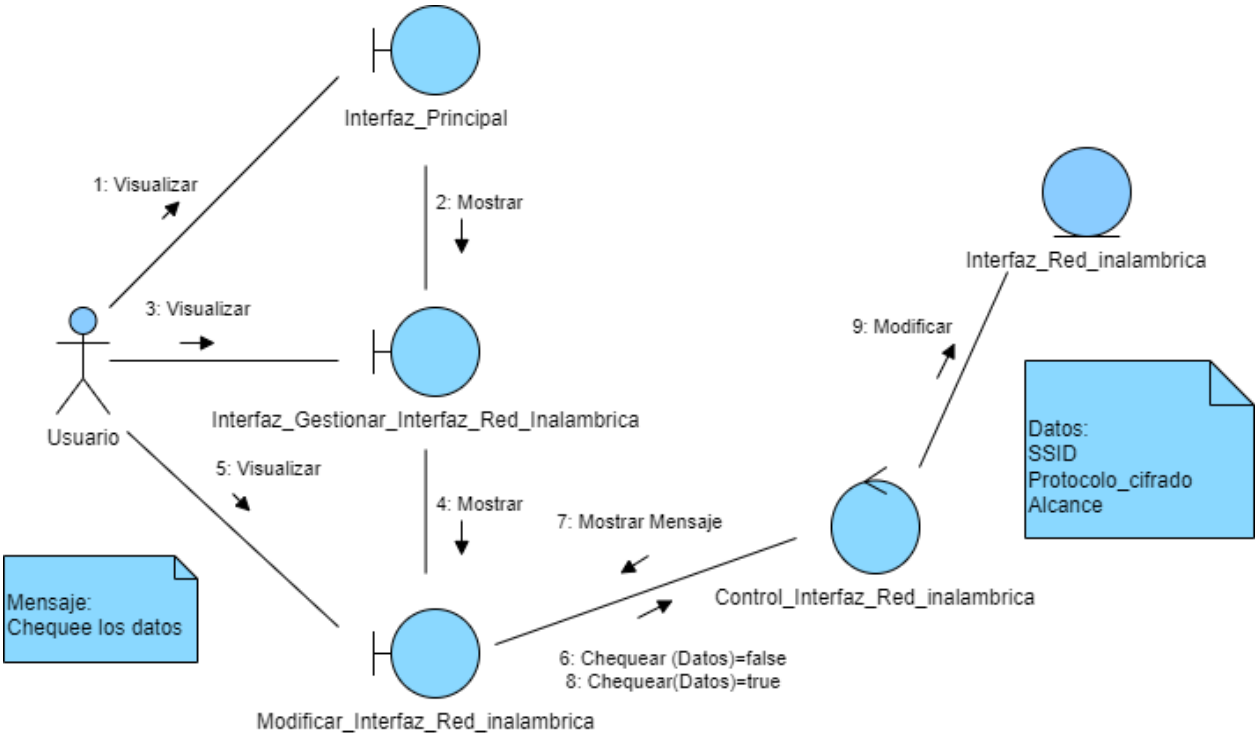

Ilustración 46: Diagrama de colaboración <Modificar Interfaz de red inalámbrica>

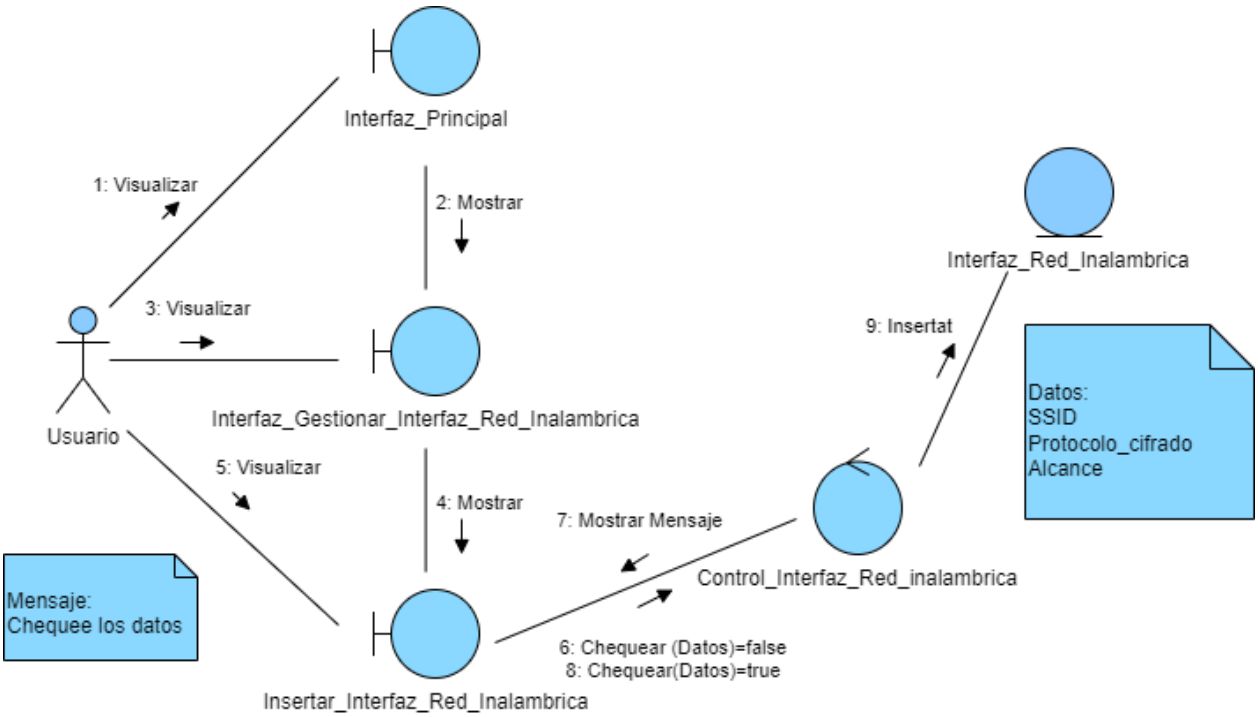

Ilustración 47: Diagrama de colaboración <Insertar Interfaz de red inalámbrica>

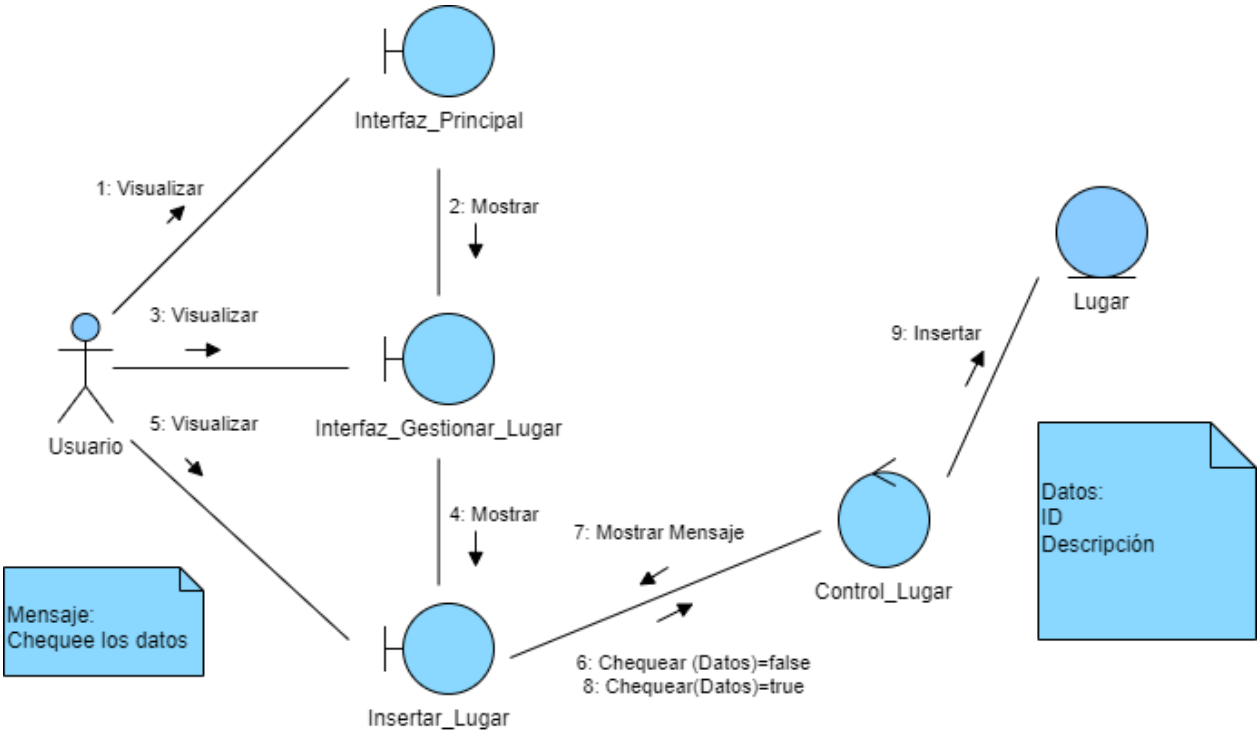

Ilustración 48: Diagrama de colaboración <Insertar Lugar>

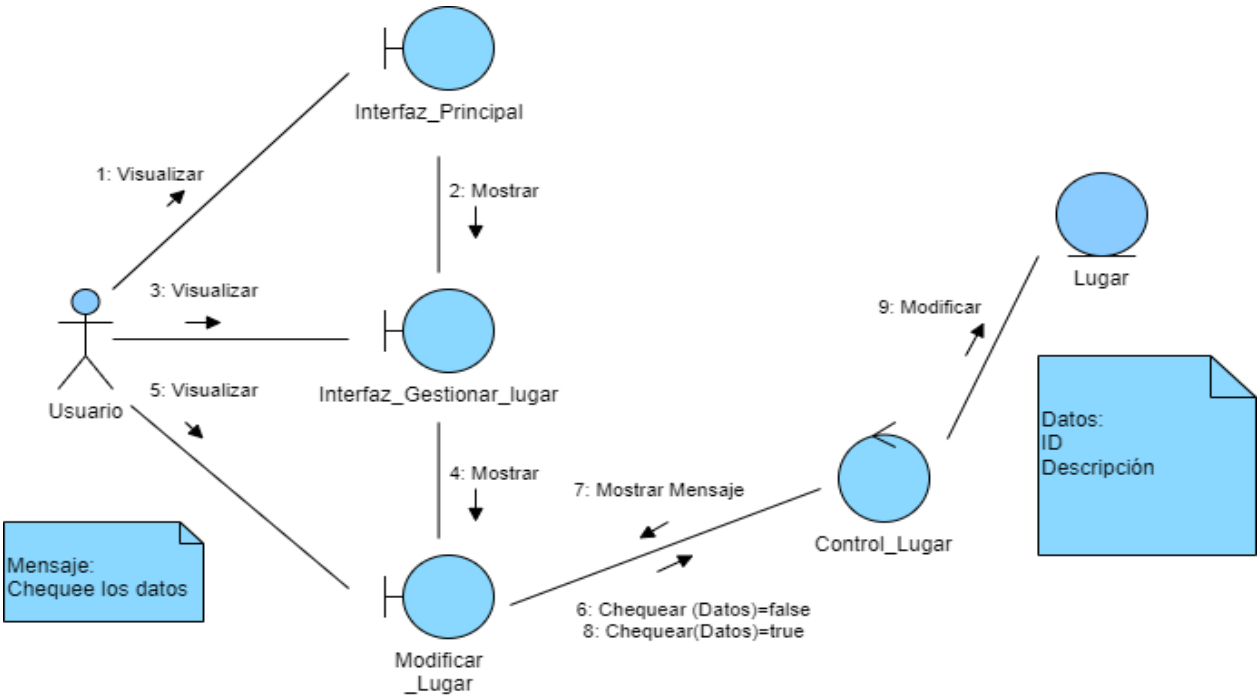

Ilustración 49: Diagrama de colaboración <Modificar Lugar>

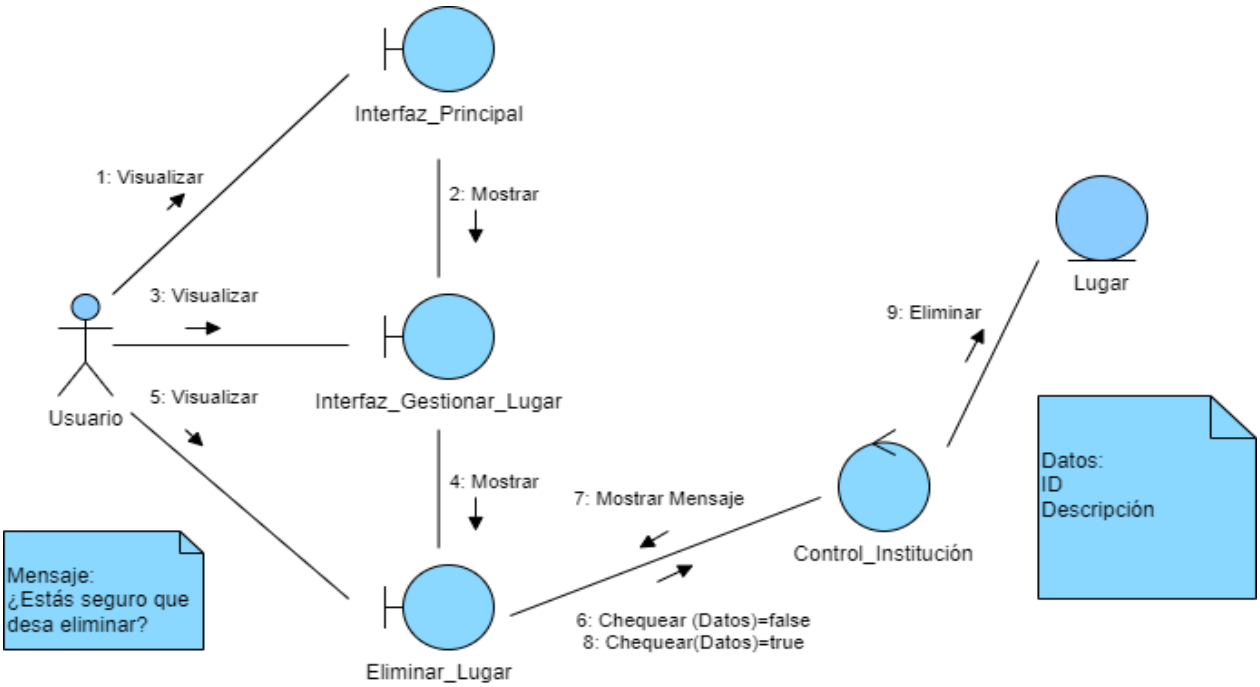

Ilustración 50: Diagrama de colaboración <Eliminar Lugar>

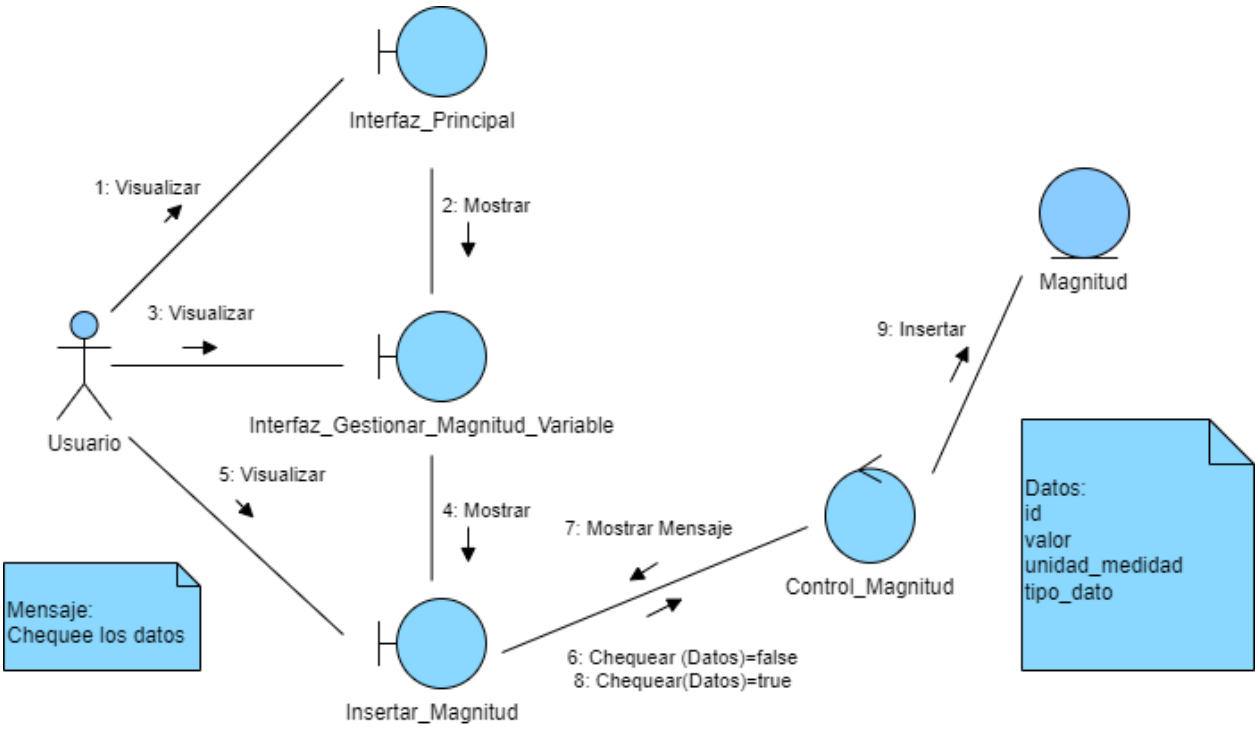

Ilustración 51: Diagrama de colaboración <Insertar Magnitud Variable>

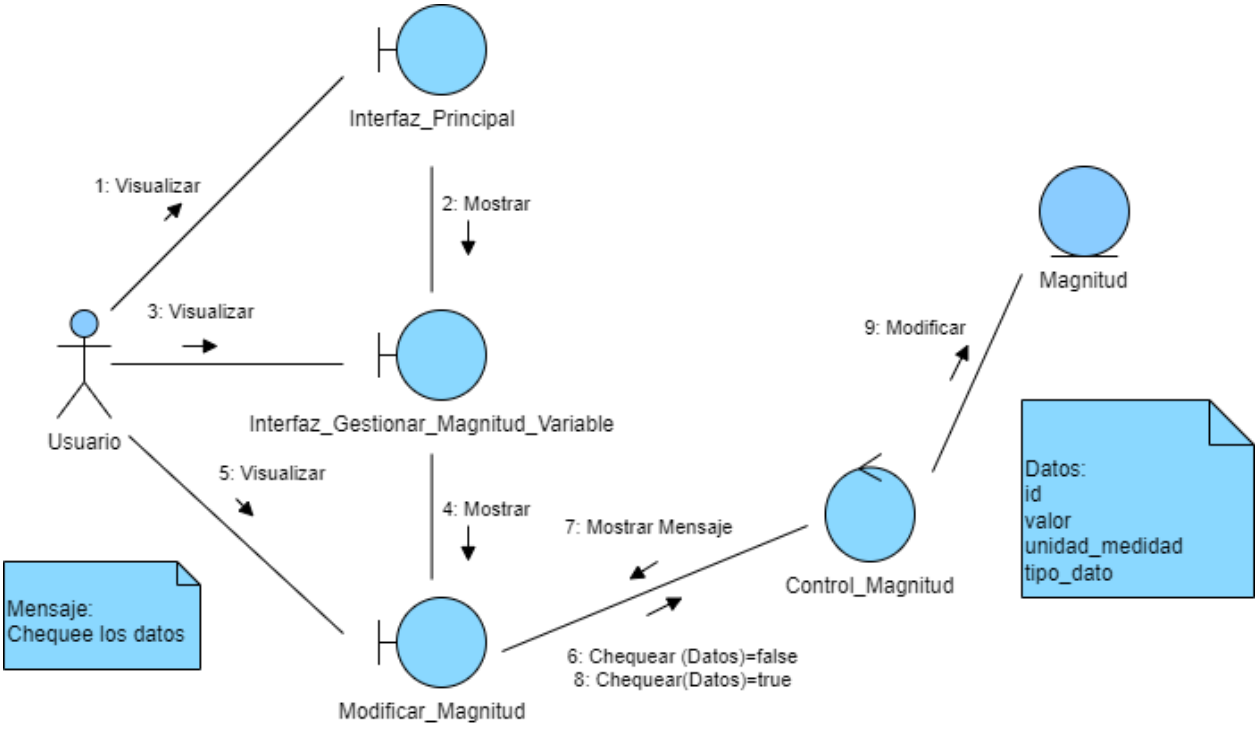

Ilustración 52: Diagrama de colaboración <Modificar Magnitud Variable>

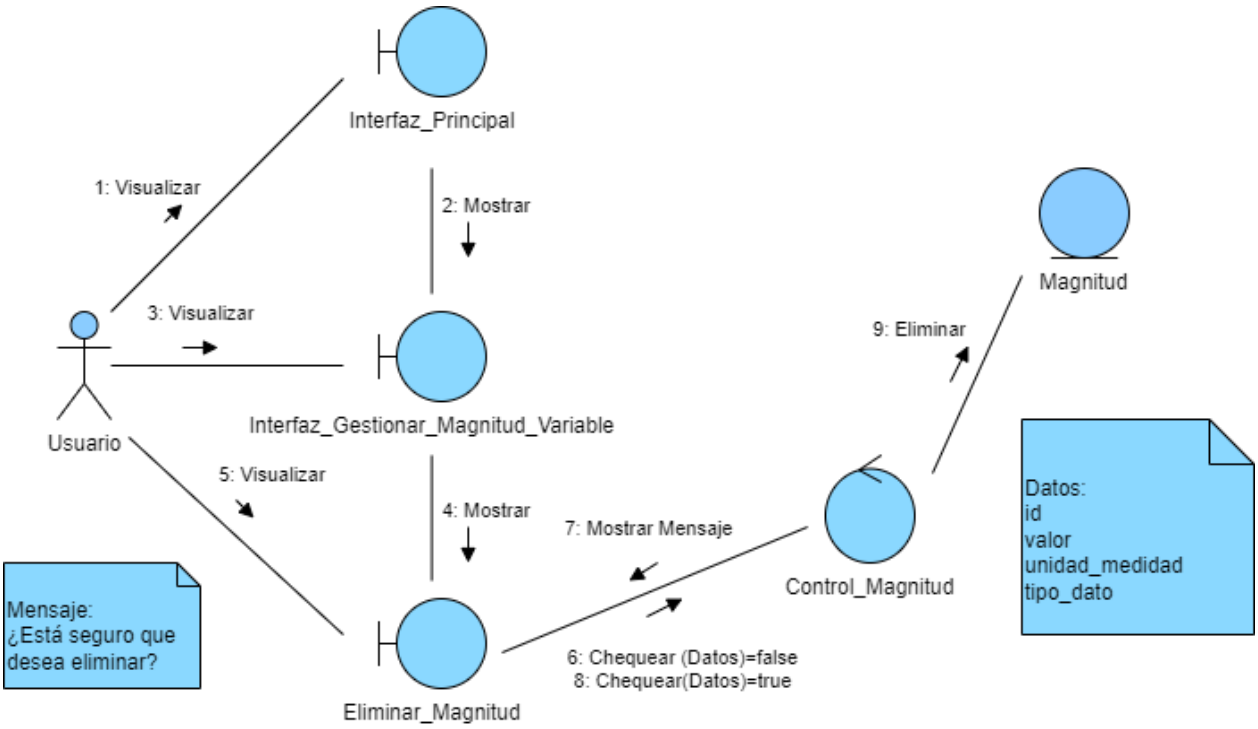

Ilustración 53: Diagrama de colaboración <Eliminar Magnitud Variable>

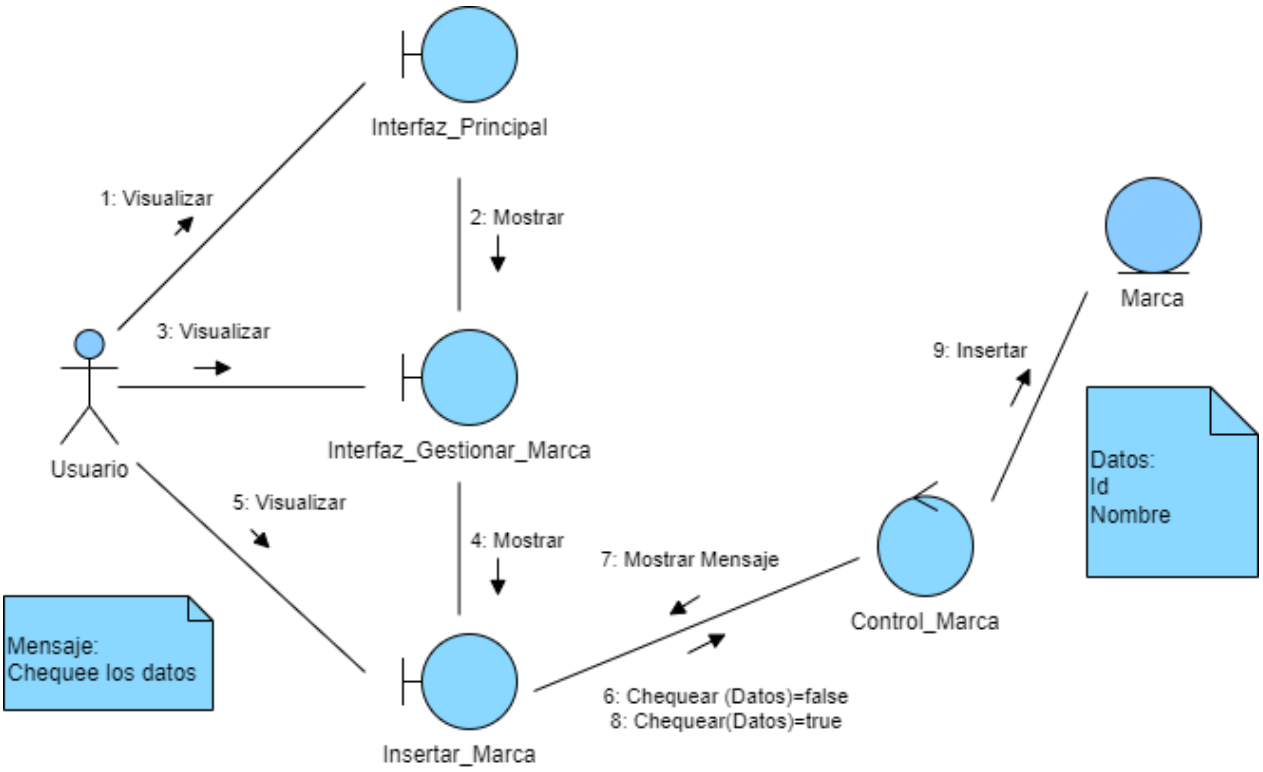

Ilustración 54: Diagrama de colaboración <Insertar Marca>

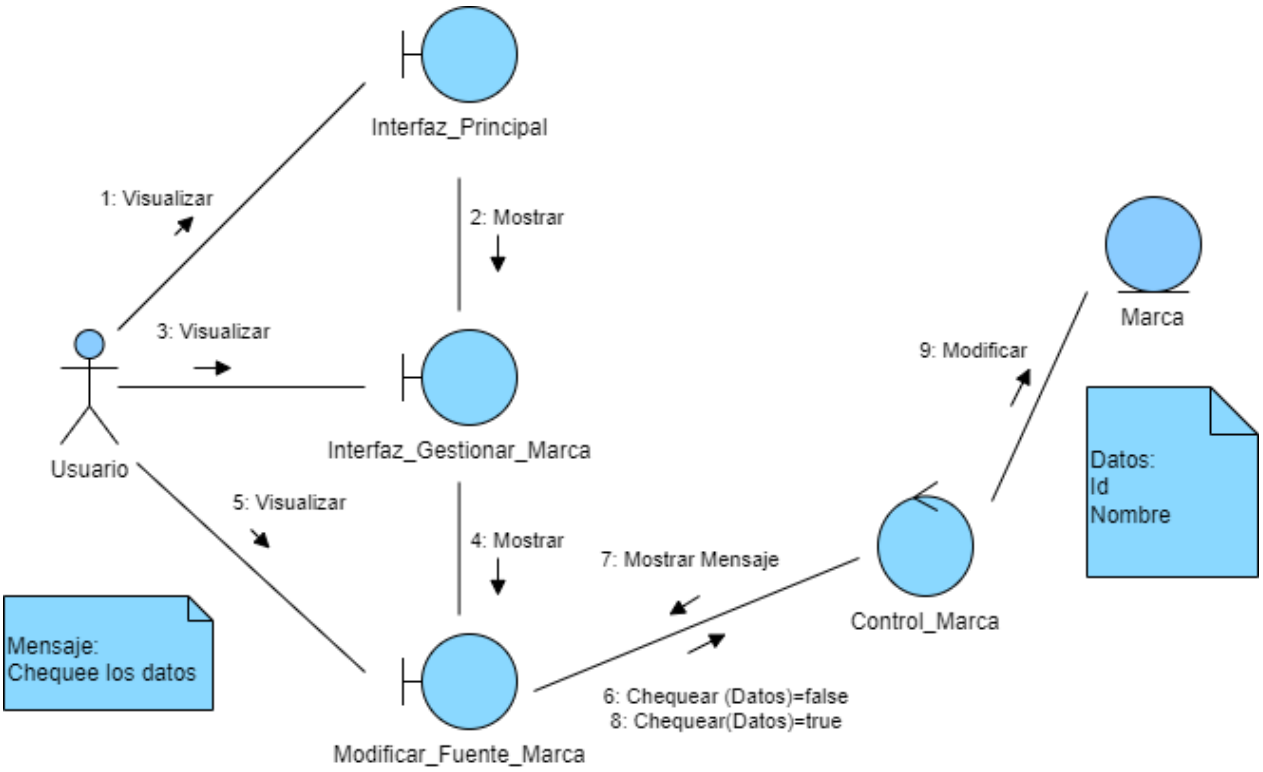

Ilustración 55: Diagrama de colaboración <Modificar Marca>

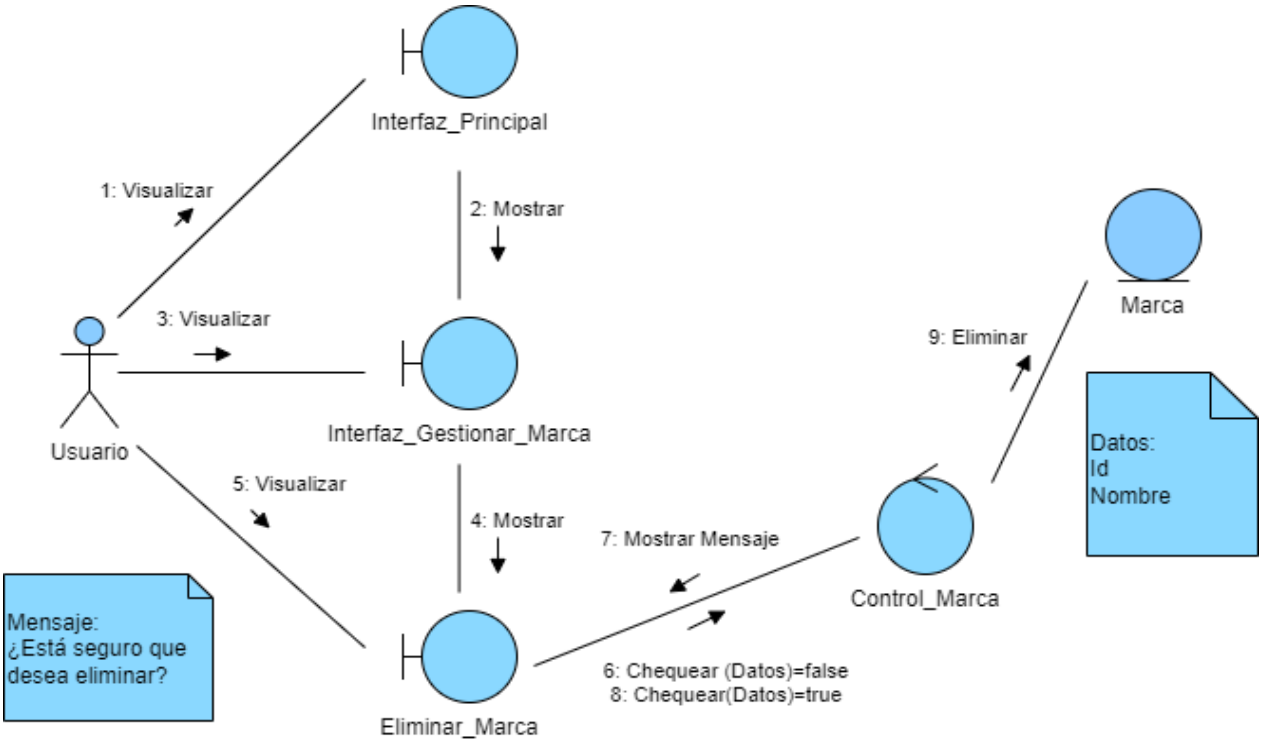

Ilustración 56: Diagrama de colaboración <Eliminar Marca>

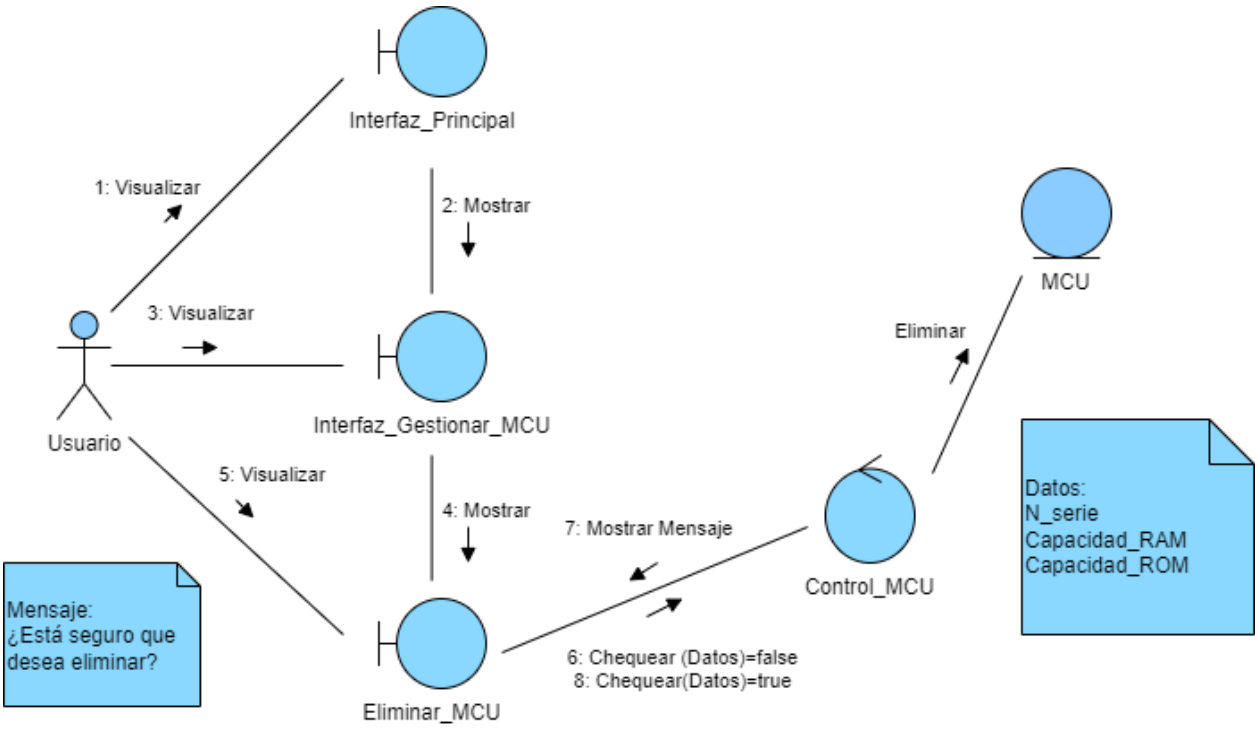

Ilustración 57: Diagrama de colaboración <Eliminar Unidad de control principal>

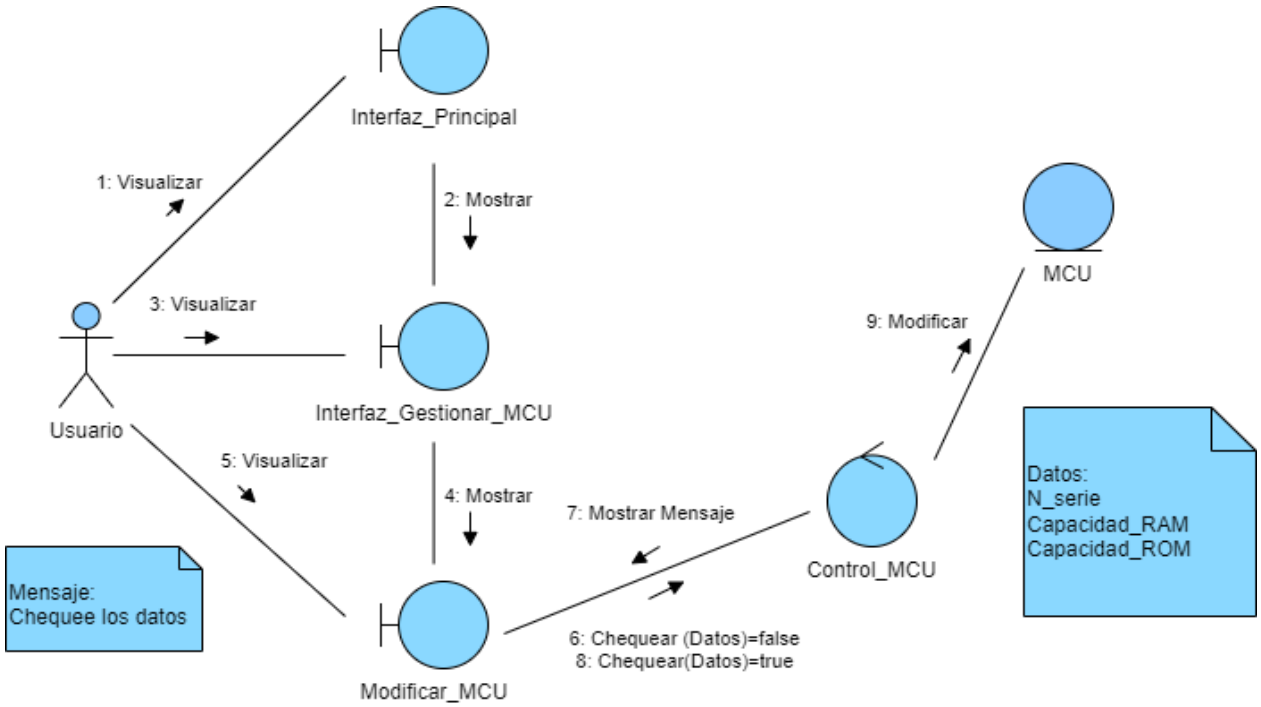

Ilustración 58: Diagrama de colaboración <Modificar Unidad de control principal>

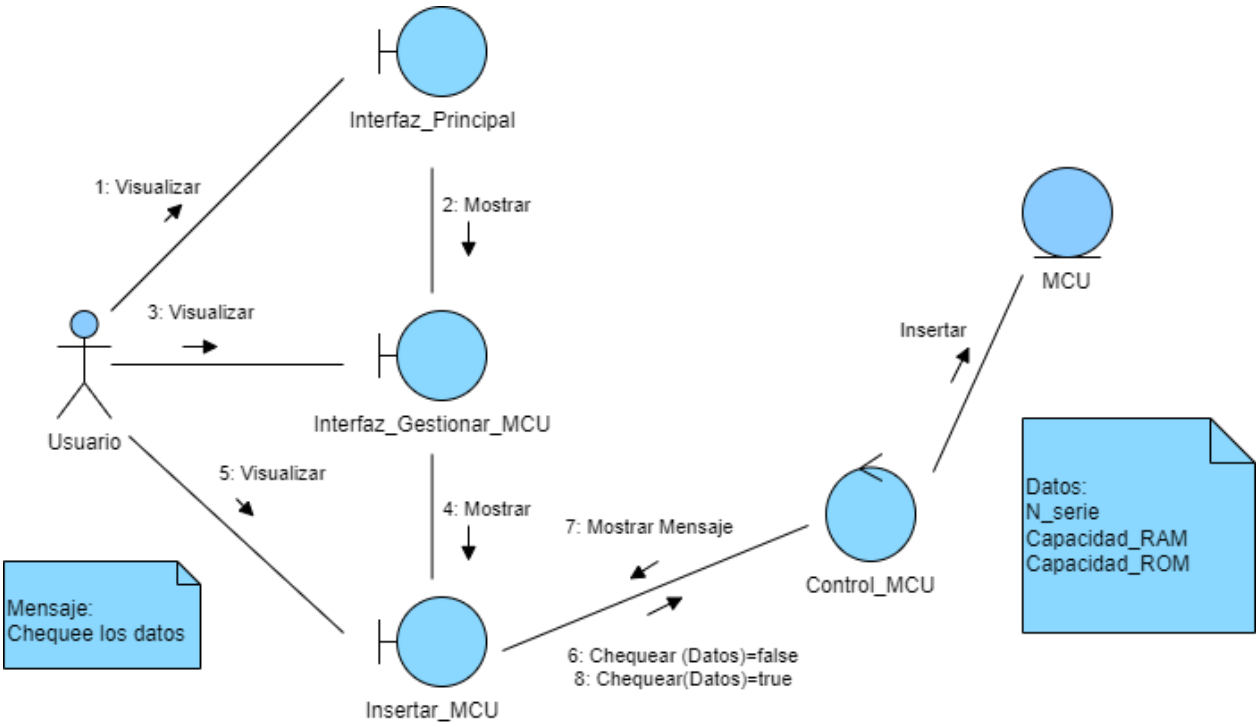

Ilustración 59: Diagrama de colaboración <Insertar Unidad de control principal>

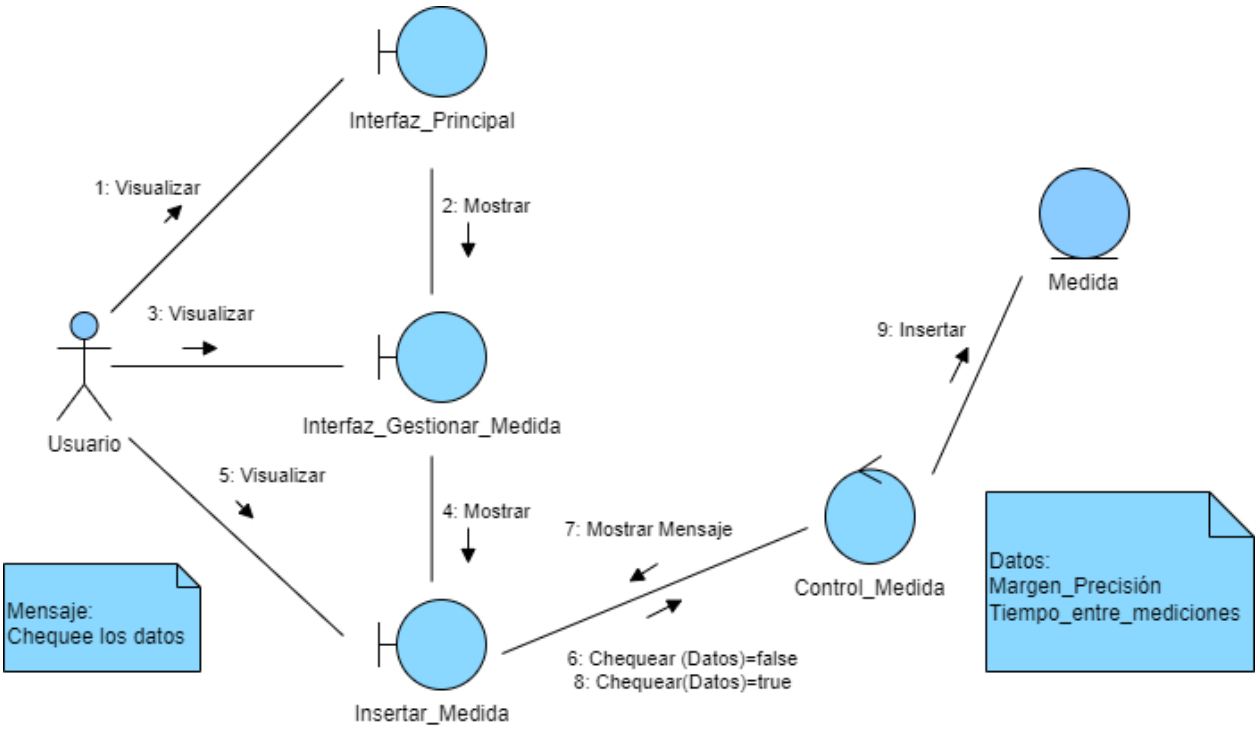

Ilustración 60: Diagrama de colaboración <Insertar Medida>

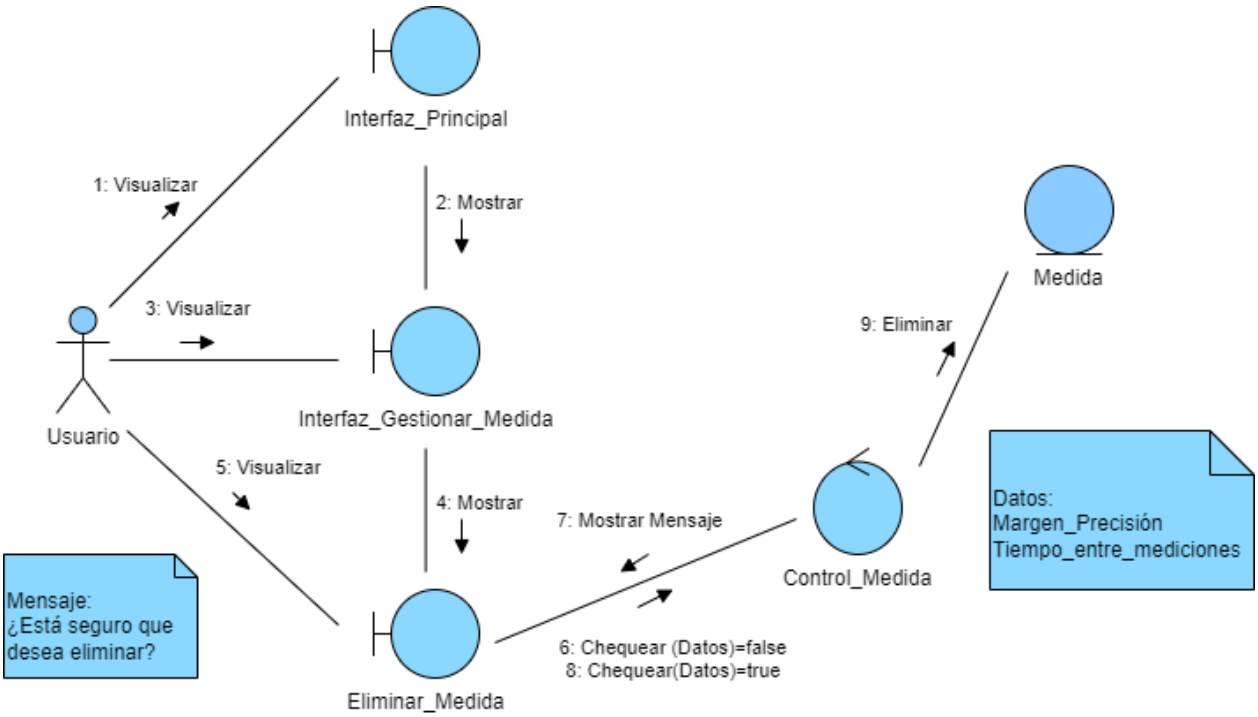

Ilustración 61: Diagrama de colaboración <Eliminar Medida>

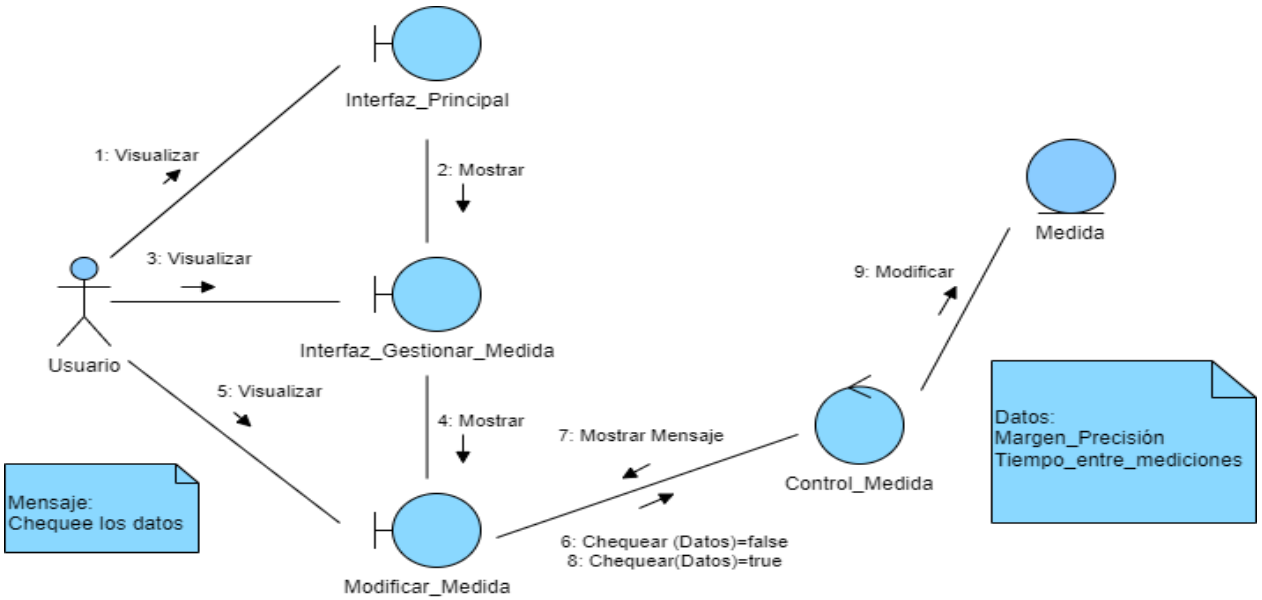

Ilustración 62: Diagrama de colaboración <Modificar Medida>

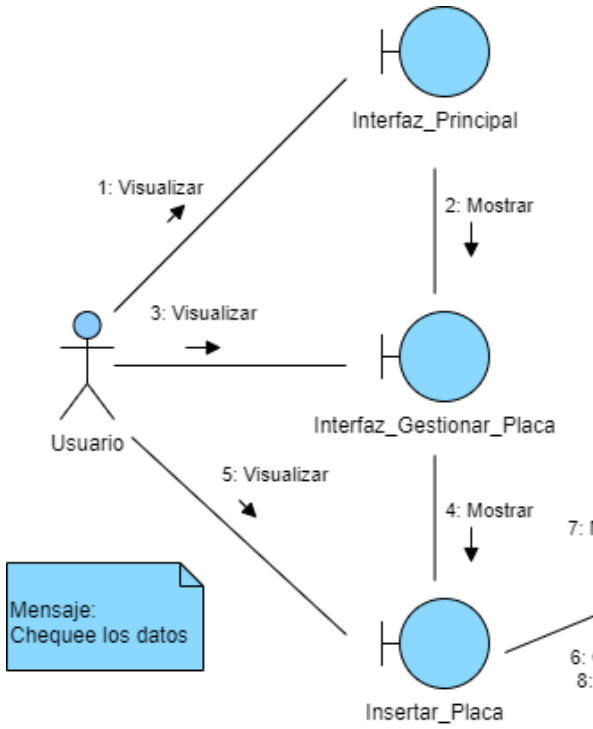

Ilustración 63: Diagrama de colaboración <Insertar Placa>

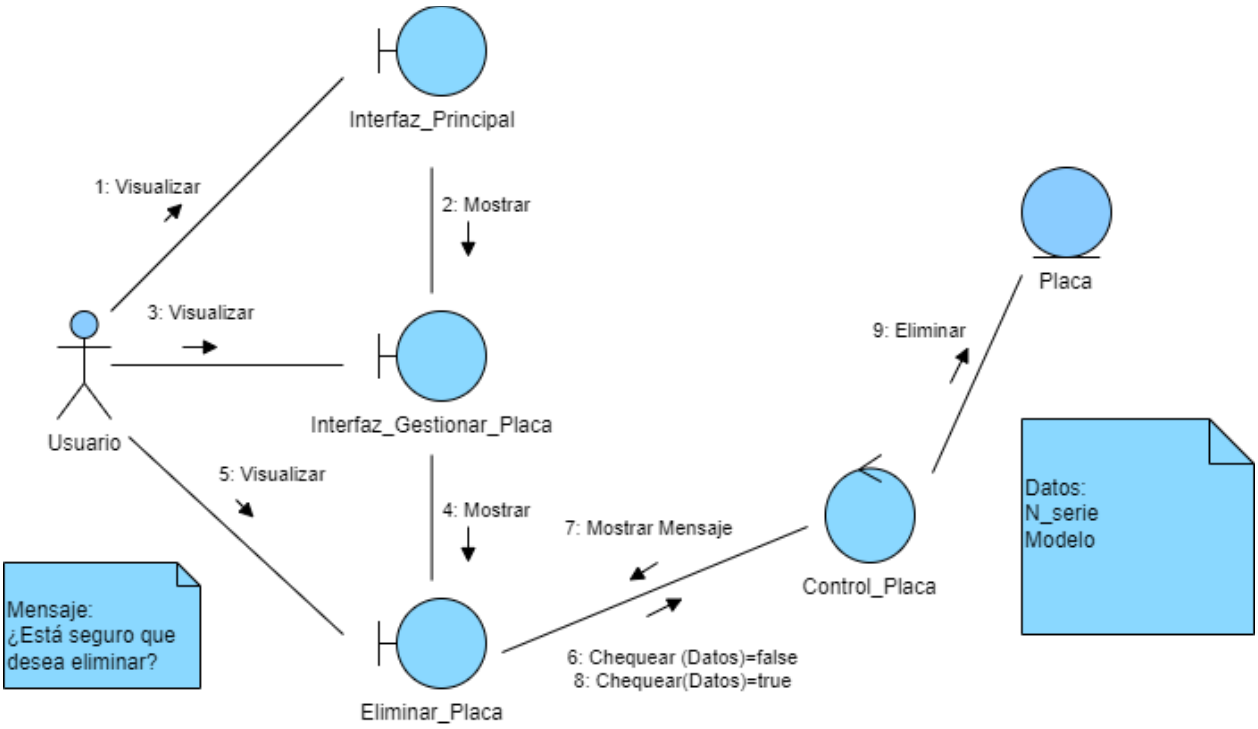

Ilustración 62: Diagrama de colaboración <Eliminar Placa>

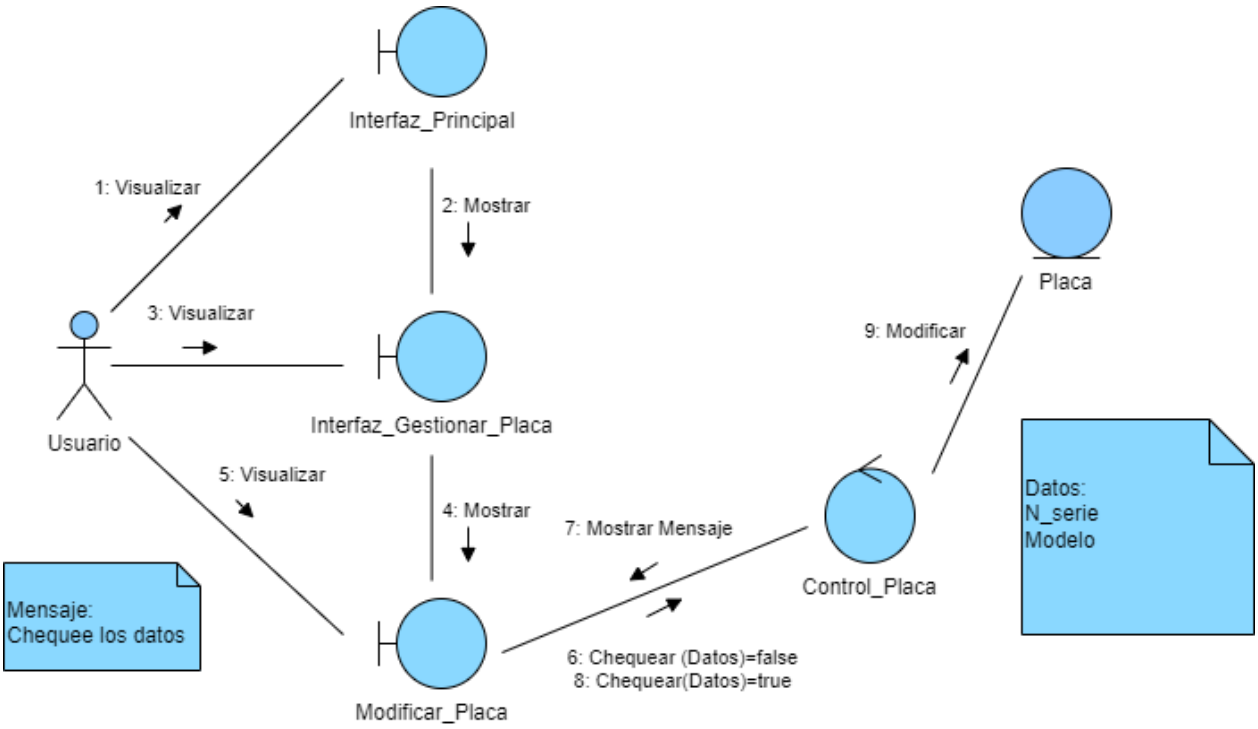

Ilustración 63: Diagrama de colaboración <Modificar Placa>

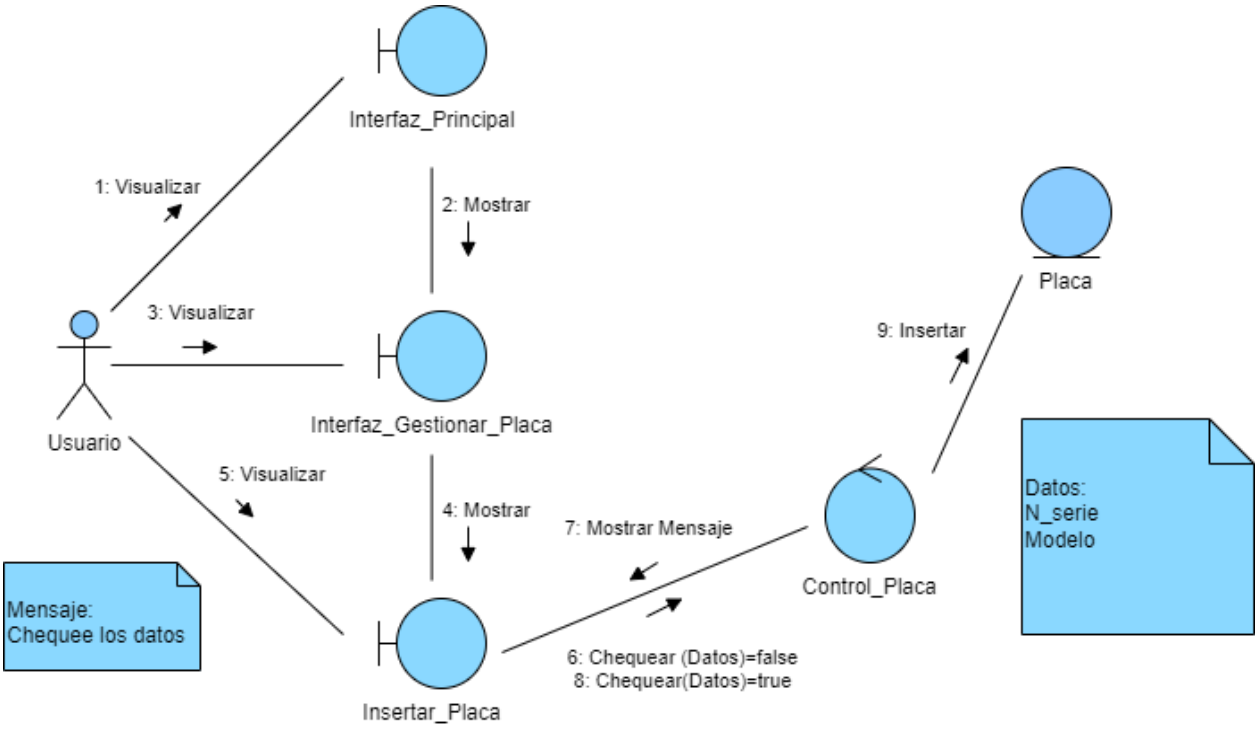

Ilustración 64: Diagrama de colaboración <Insertar Placa>

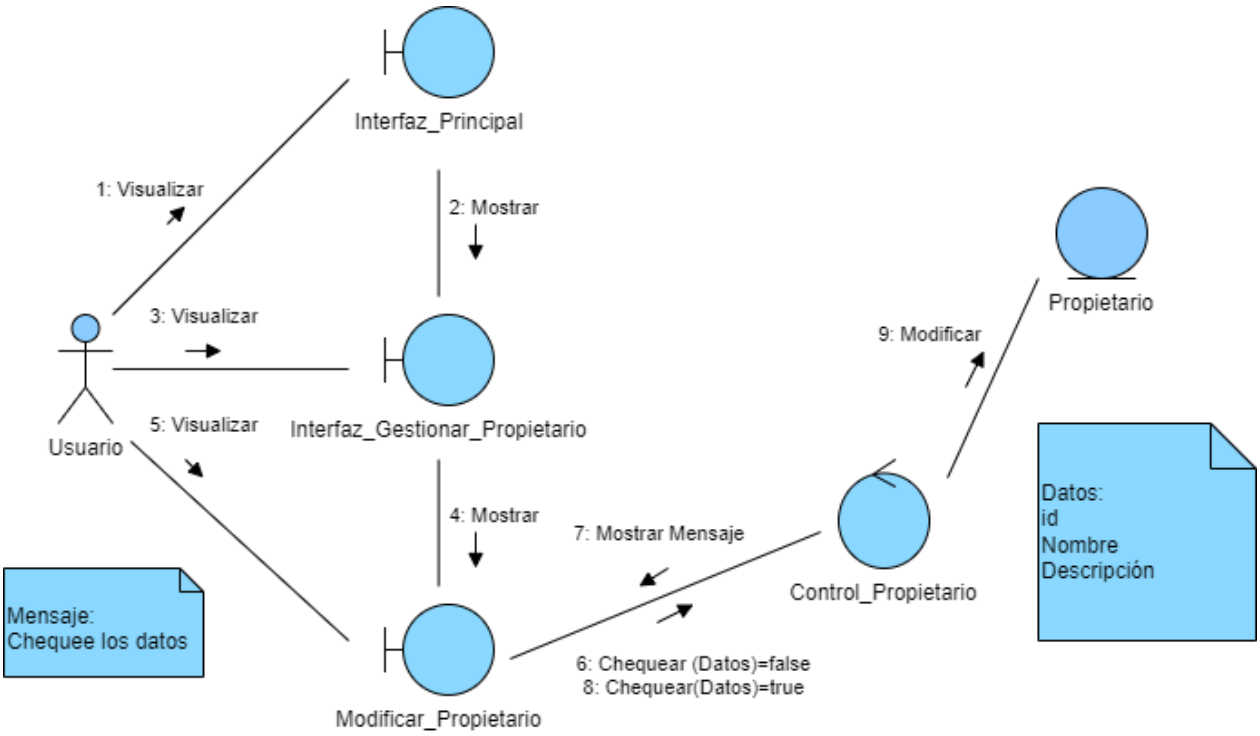

Ilustración 65: Diagrama de colaboración <Modificar Propietario>

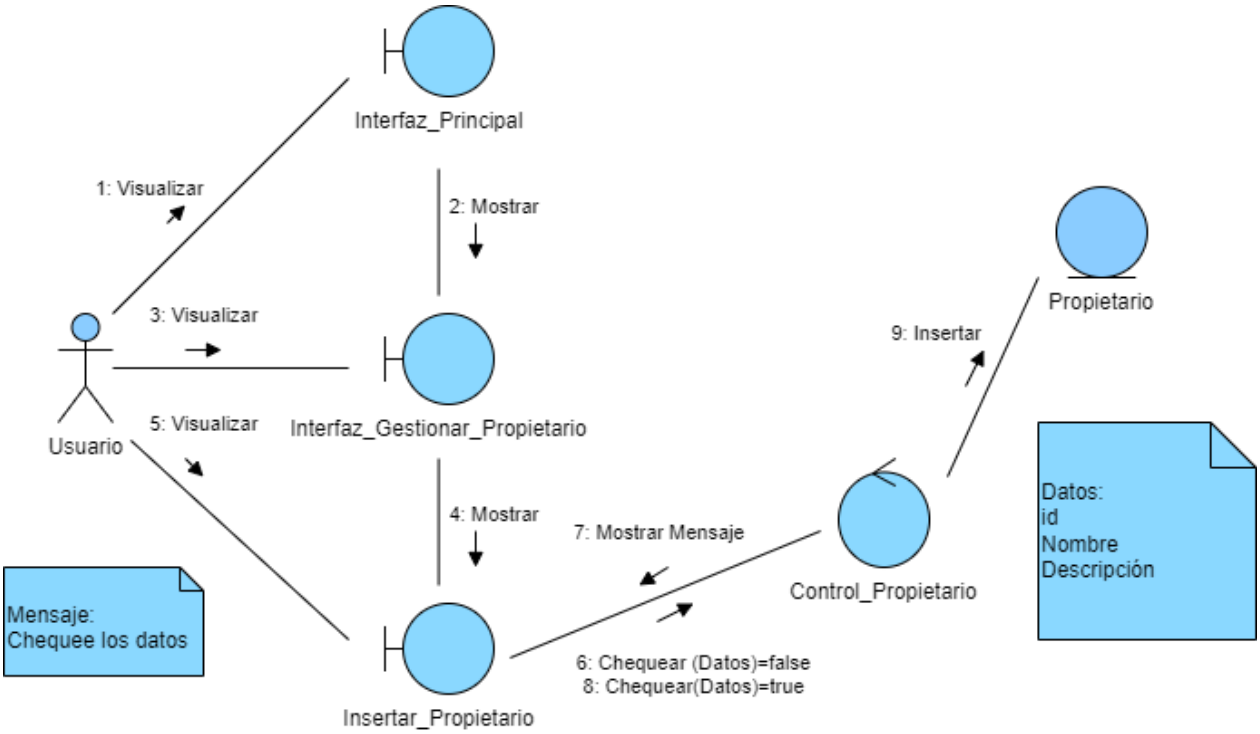

Ilustración 66: Diagrama de colaboración <Insertar Propietario>

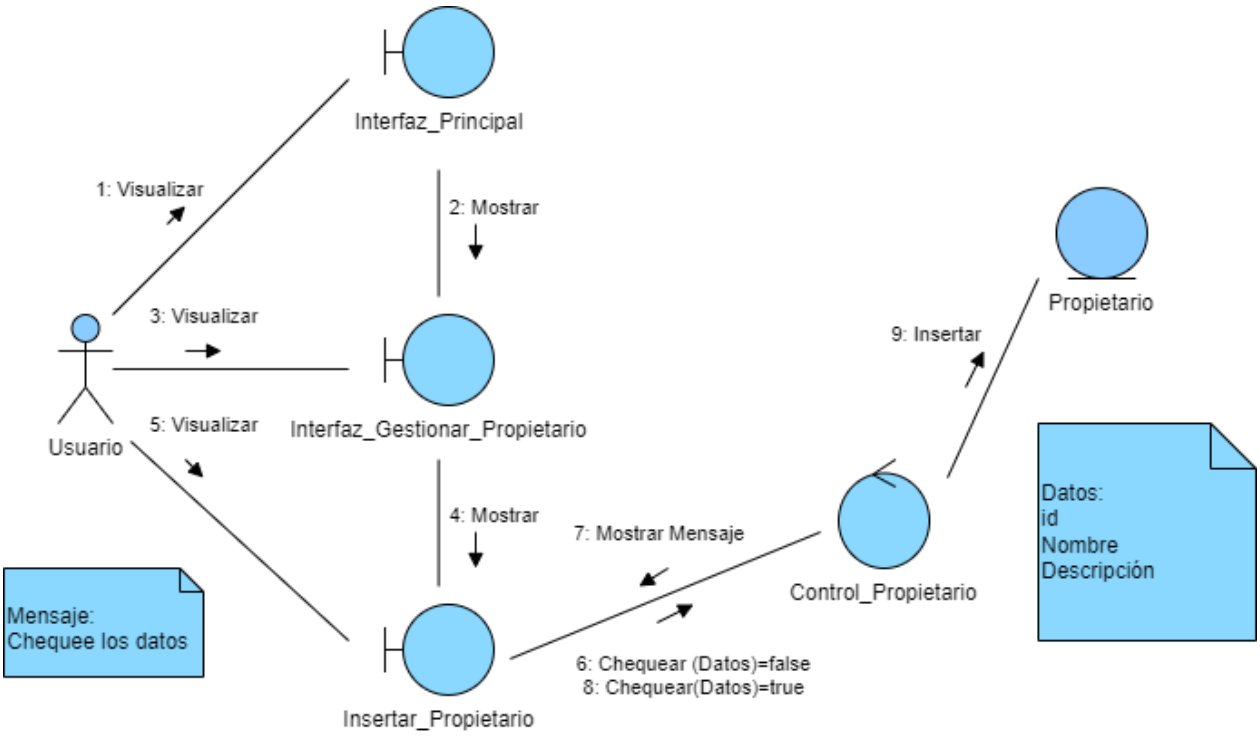

Ilustración 67: Diagrama de colaboración <Eliminar Propietario>

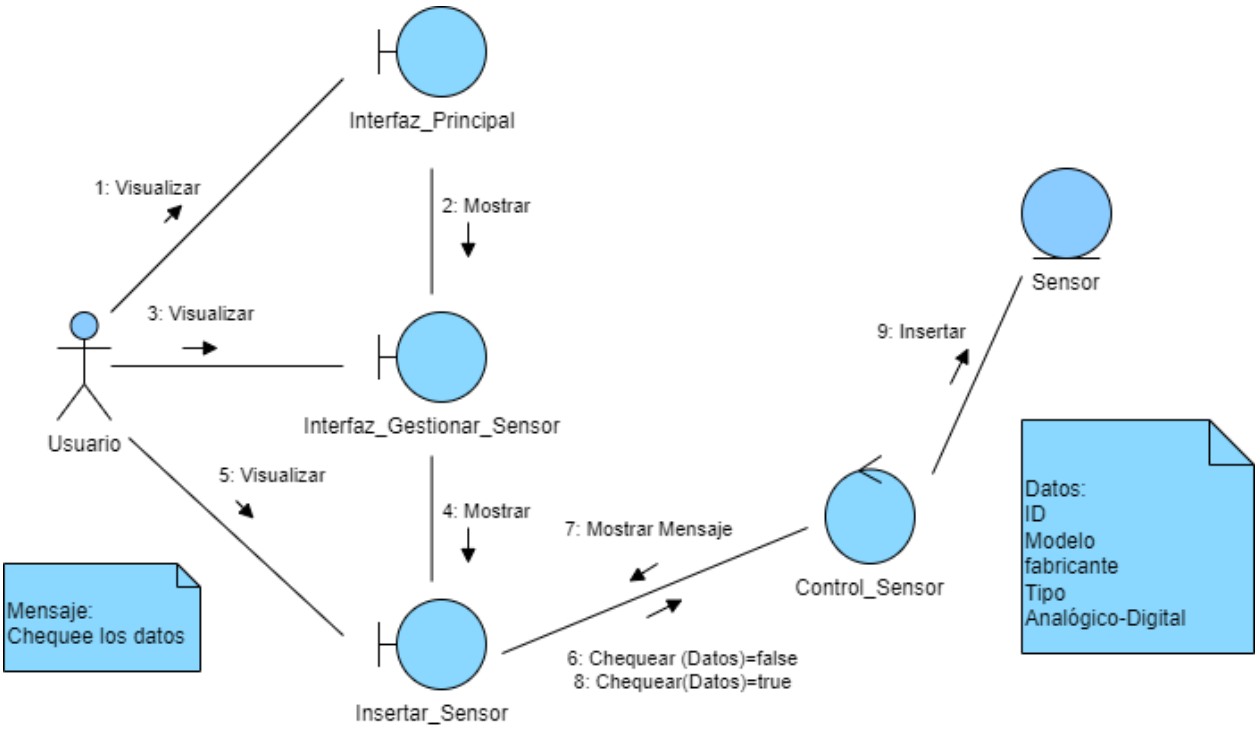

Ilustración 68: Diagrama de colaboración <Insertar Sensor >

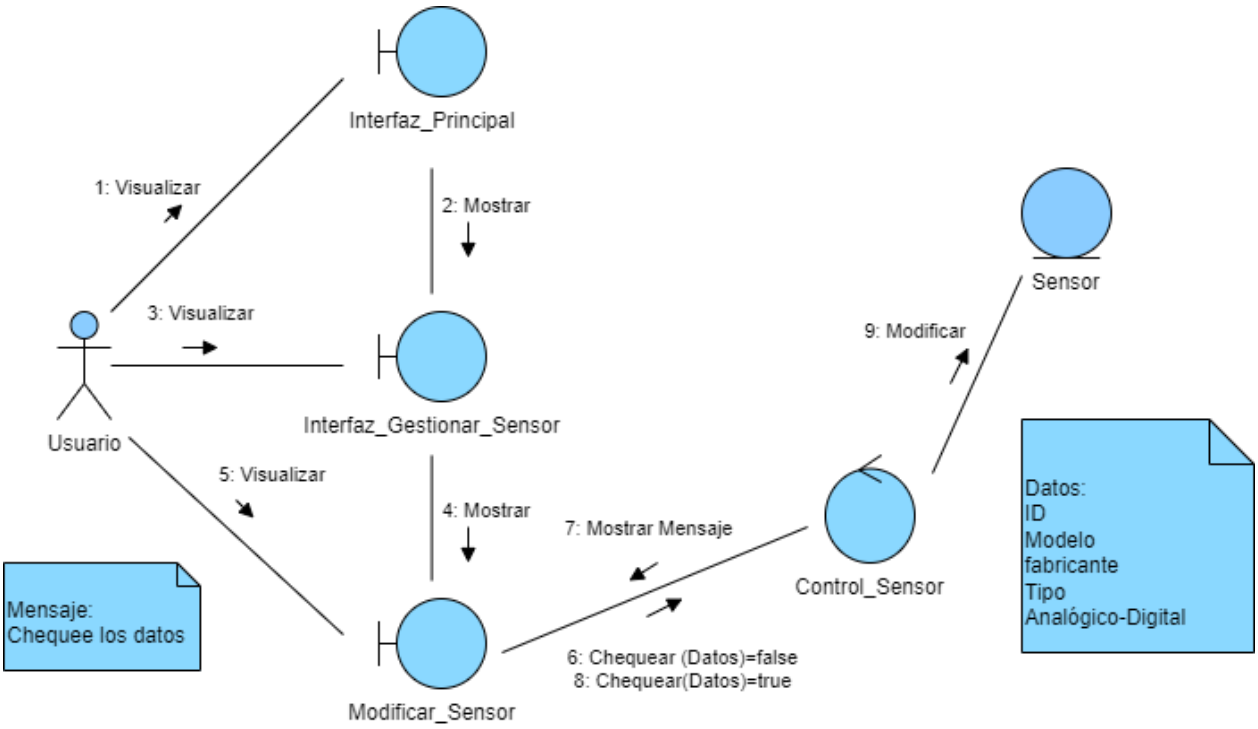

Ilustración 69: Diagrama de colaboración <Modificar Sensor >

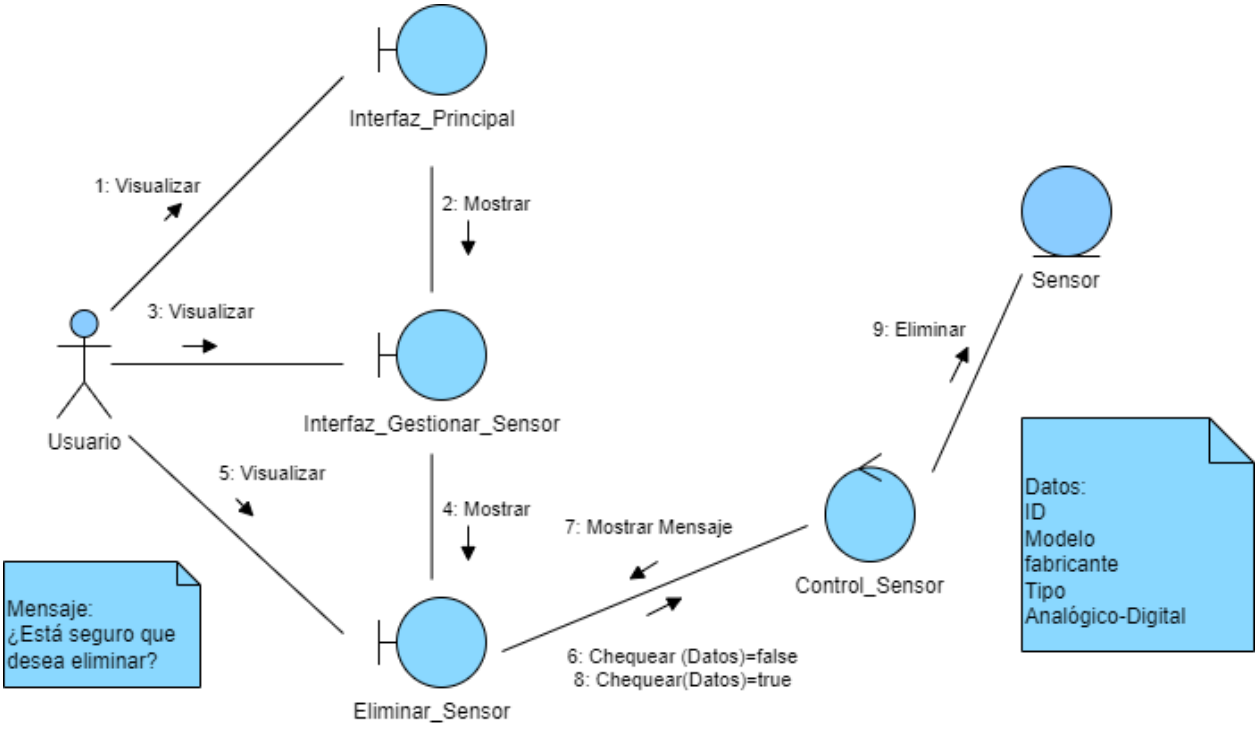

Ilustración 70: Diagrama de colaboración <Eliminar Sensor>

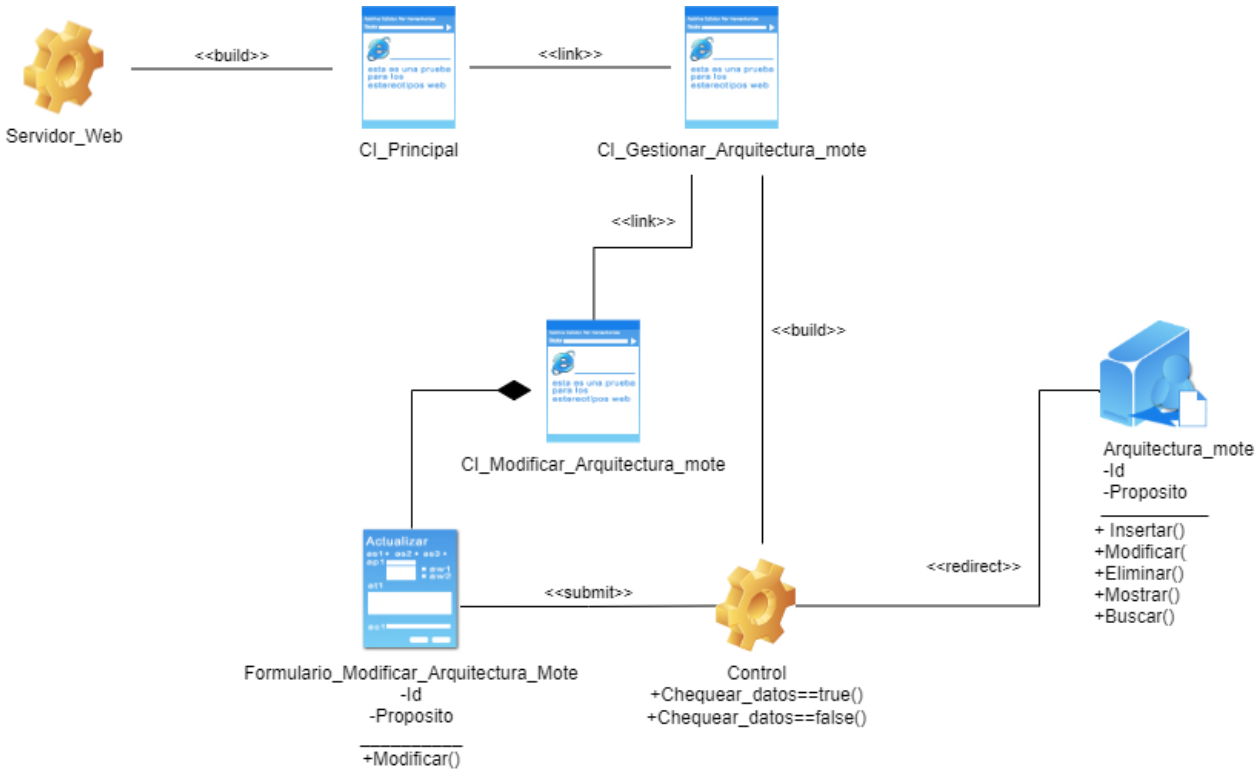

Ilustración 71: Diagrama de diseño <Modificar Arquitectura del Nodo sensor>

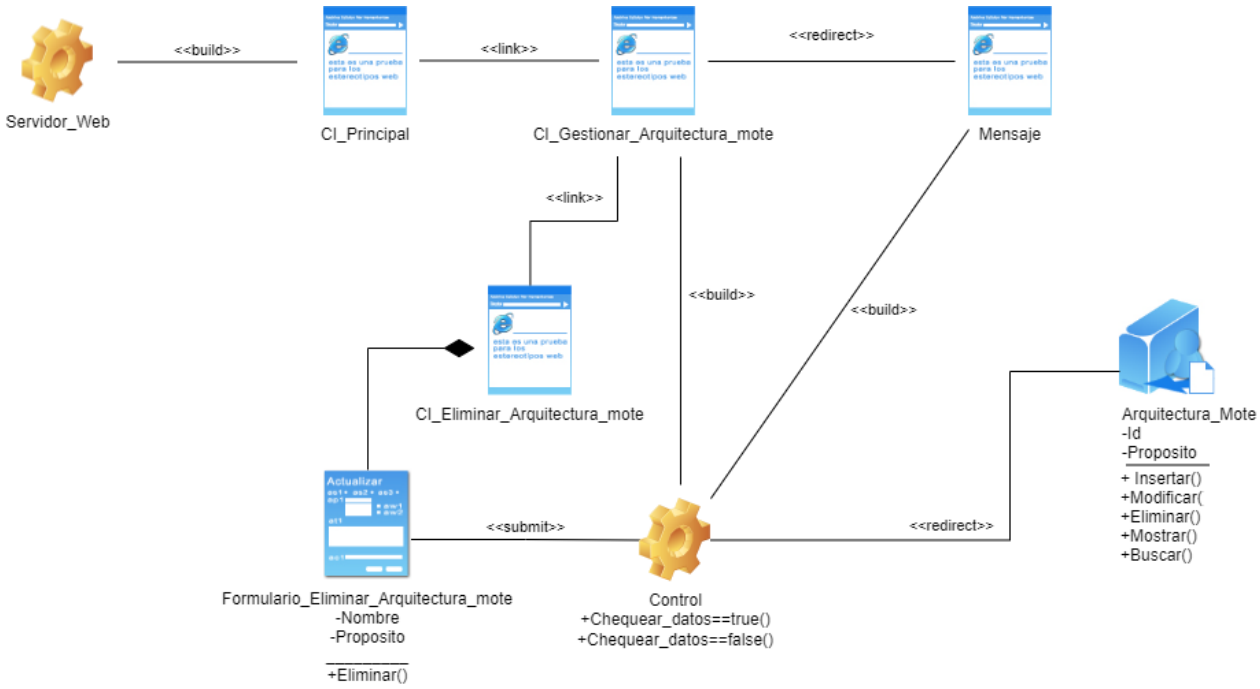

Ilustración 72: Diagrama de diseño <Eliminar Arquitectura del nodo sensor>

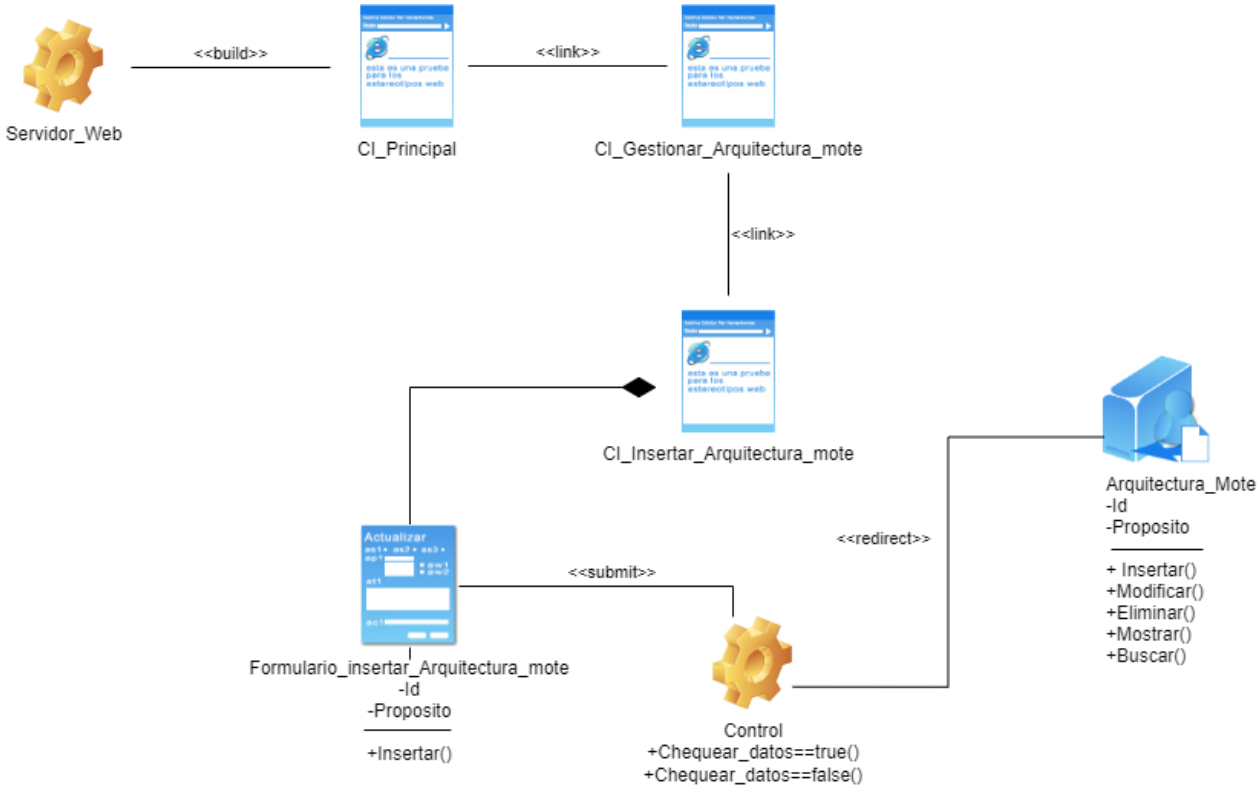

Ilustración 73: Diagrama de diseño <Insertar Arquitectura del nodo sensor>

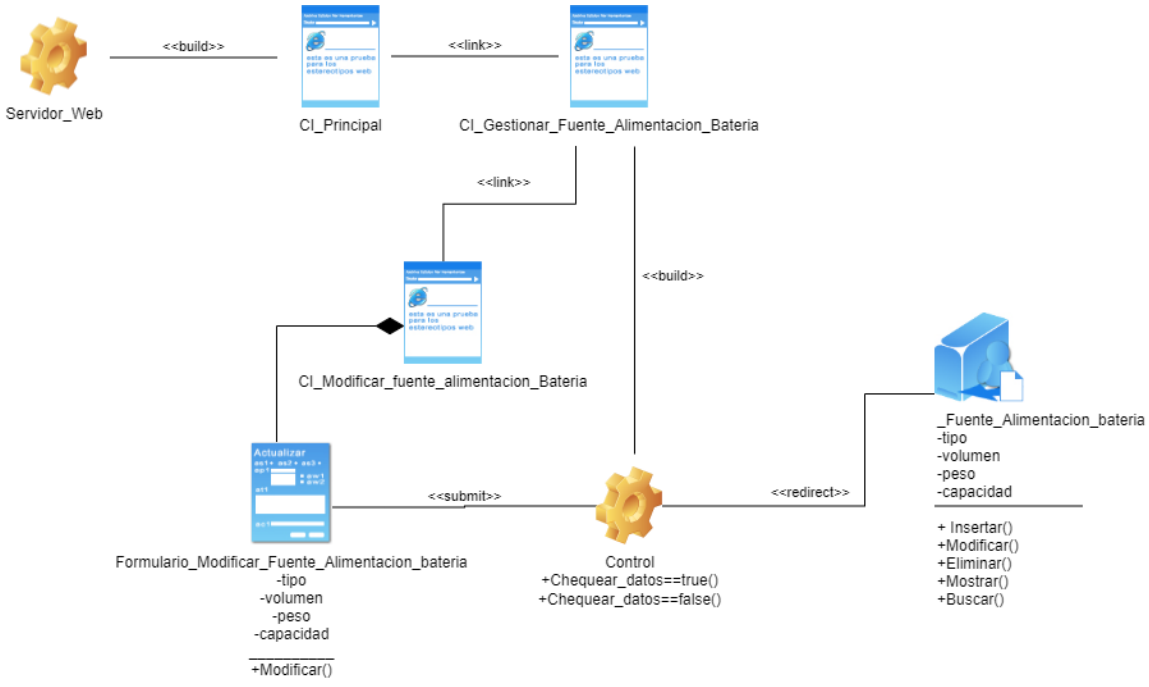

Ilustración 74: Diagrama de diseño <Modificar Fuente de alimentación Batería>

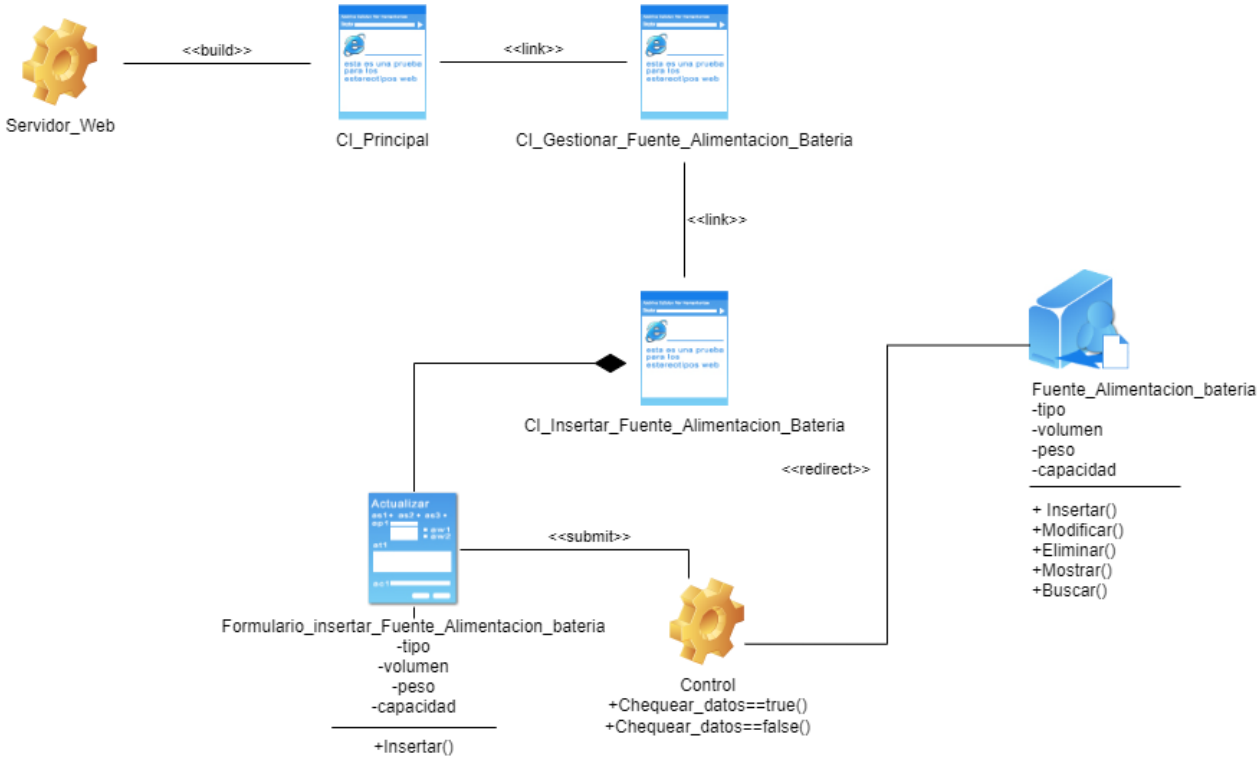

Ilustración 75: Diagrama de diseño <Insertar Fuente de alimentación Batería>

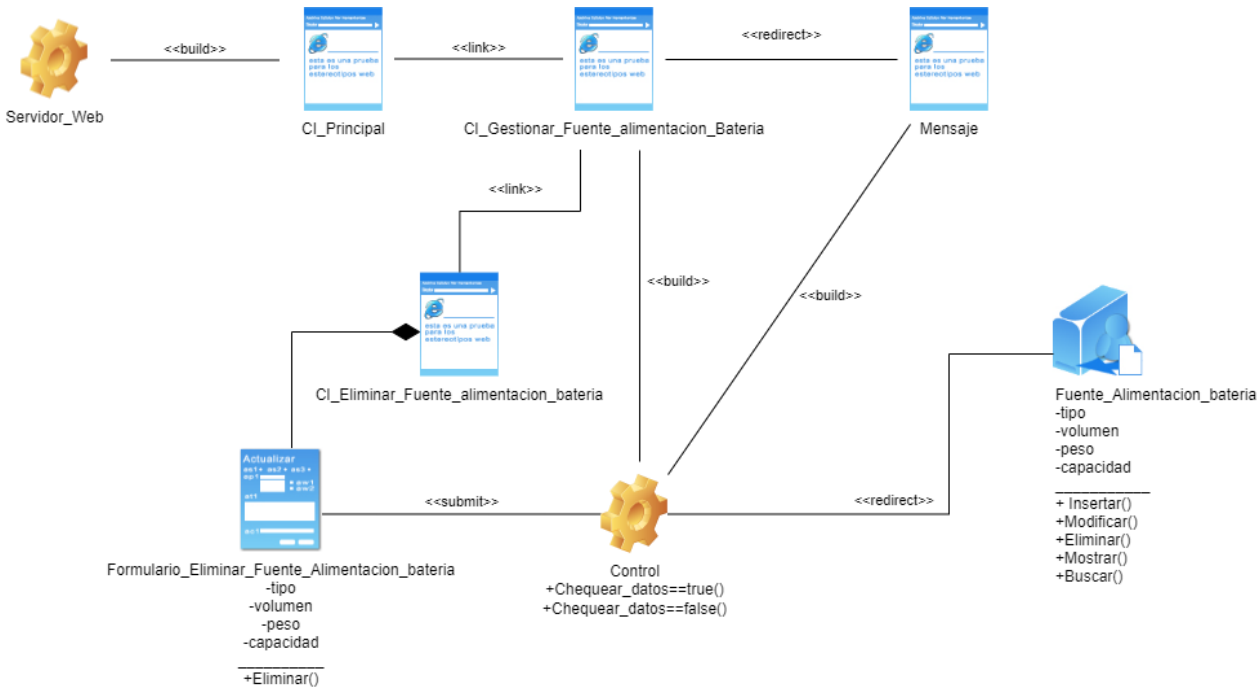

Ilustración 76: Diagrama de diseño <Eliminar Fuente de alimentación Batería>

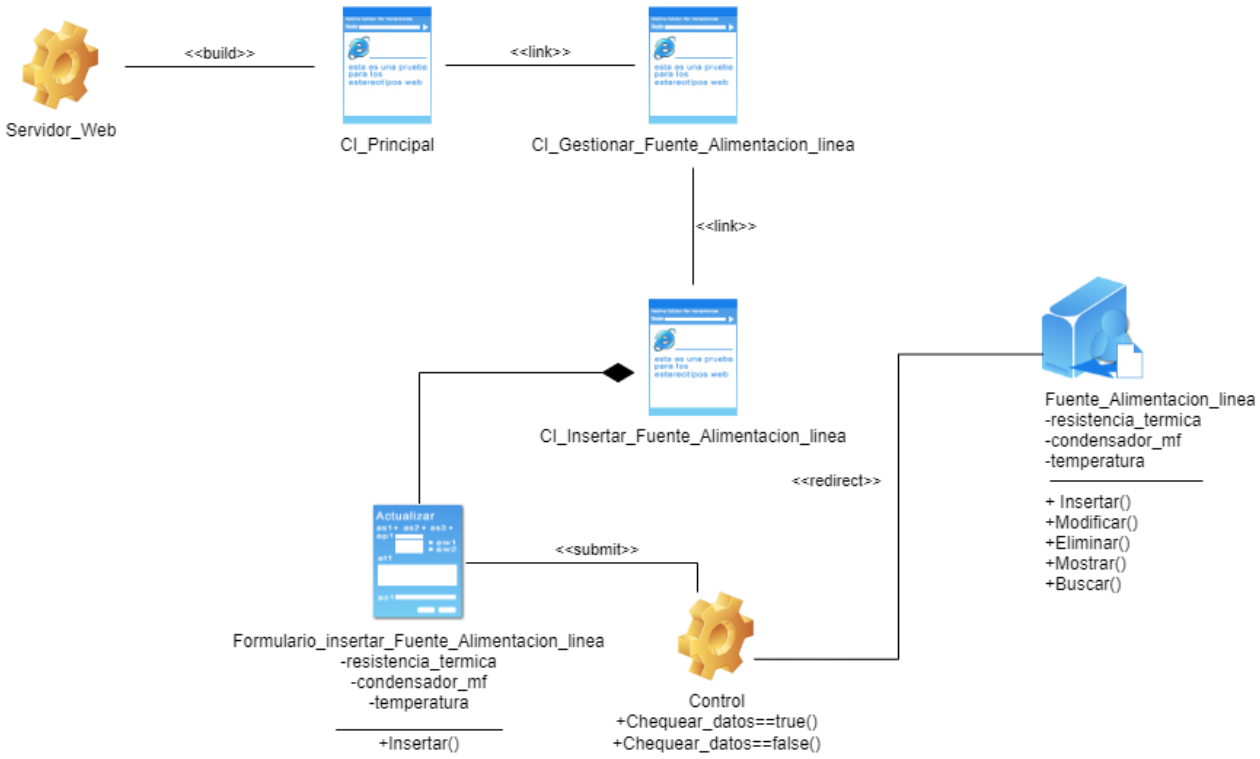

Ilustración 77: Diagrama de diseño <Insertar Fuente de alimentación Línea>

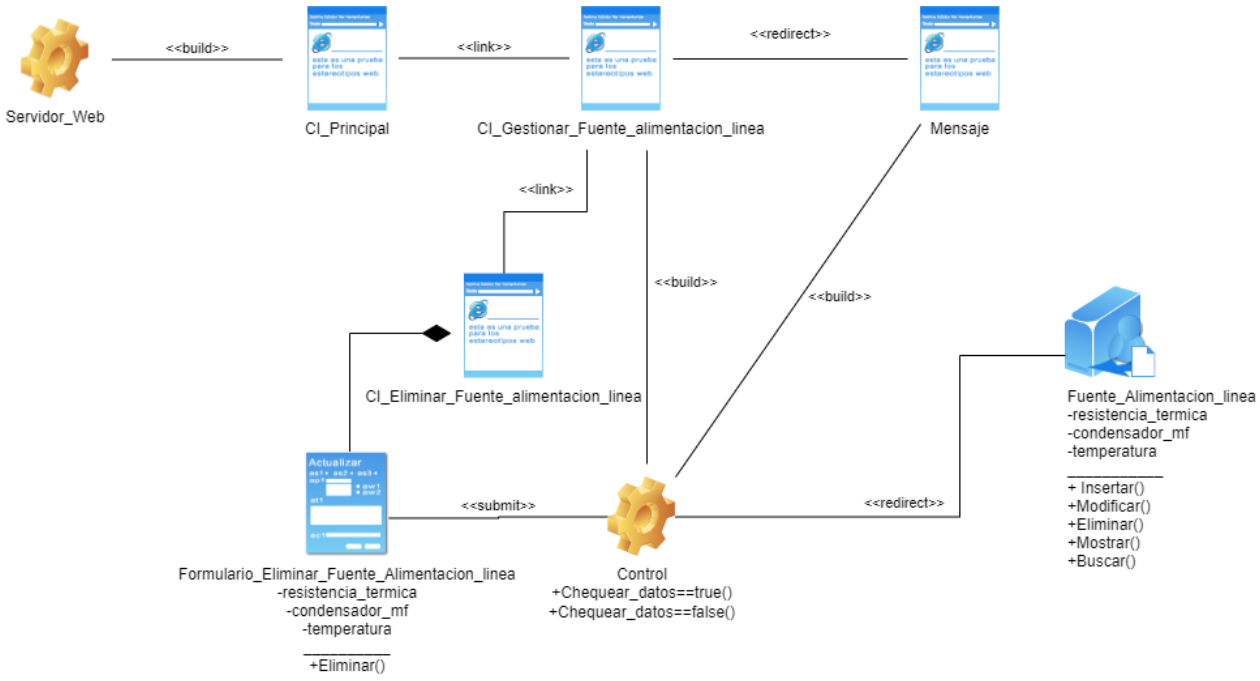

Ilustración 78: Diagrama de diseño <Eliminar Fuente de alimentación Línea>

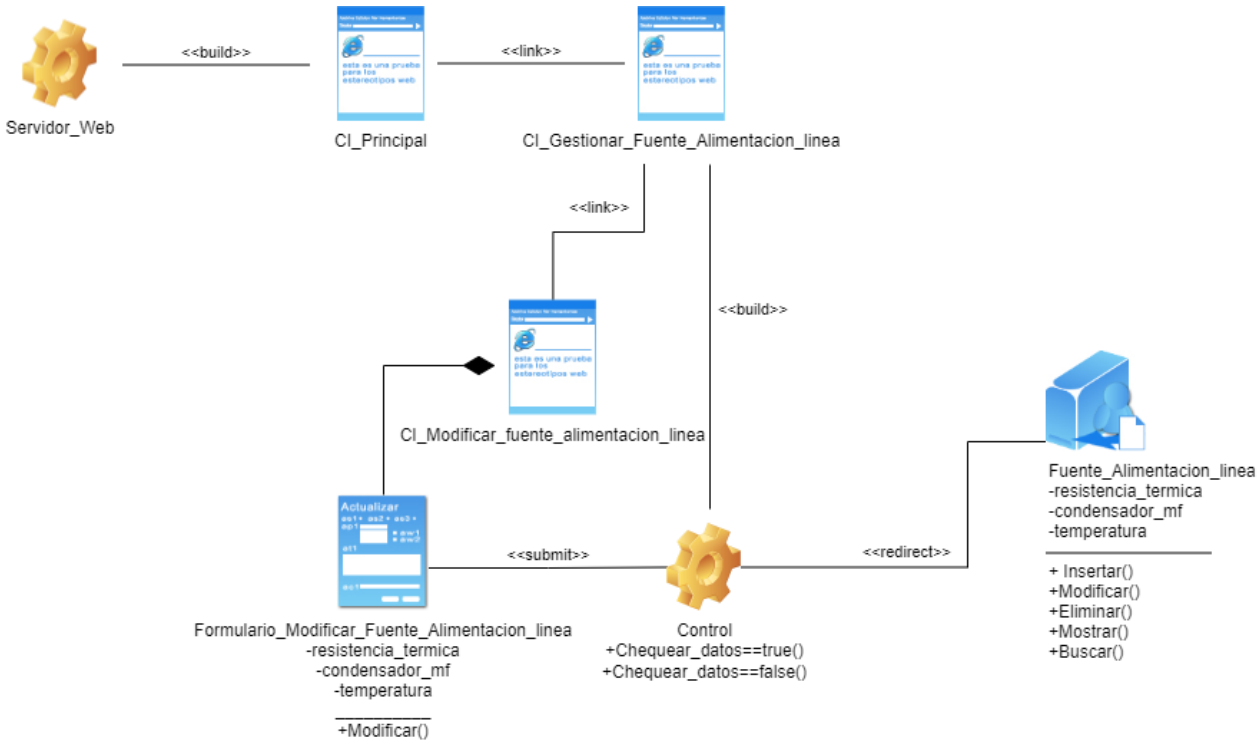

Ilustración 79: Diagrama de diseño <Modificar Fuente de alimentación Línea>

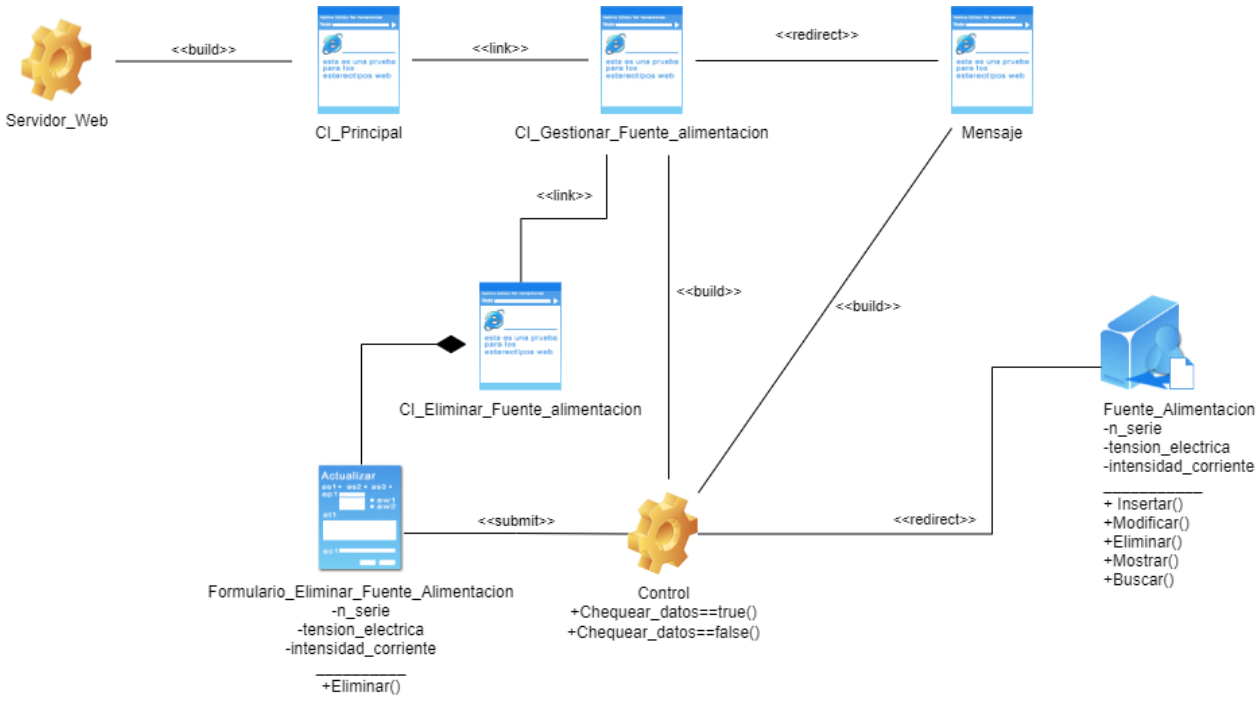

Ilustración 80: Diagrama de diseño <Eliminar Fuente de alimentación>

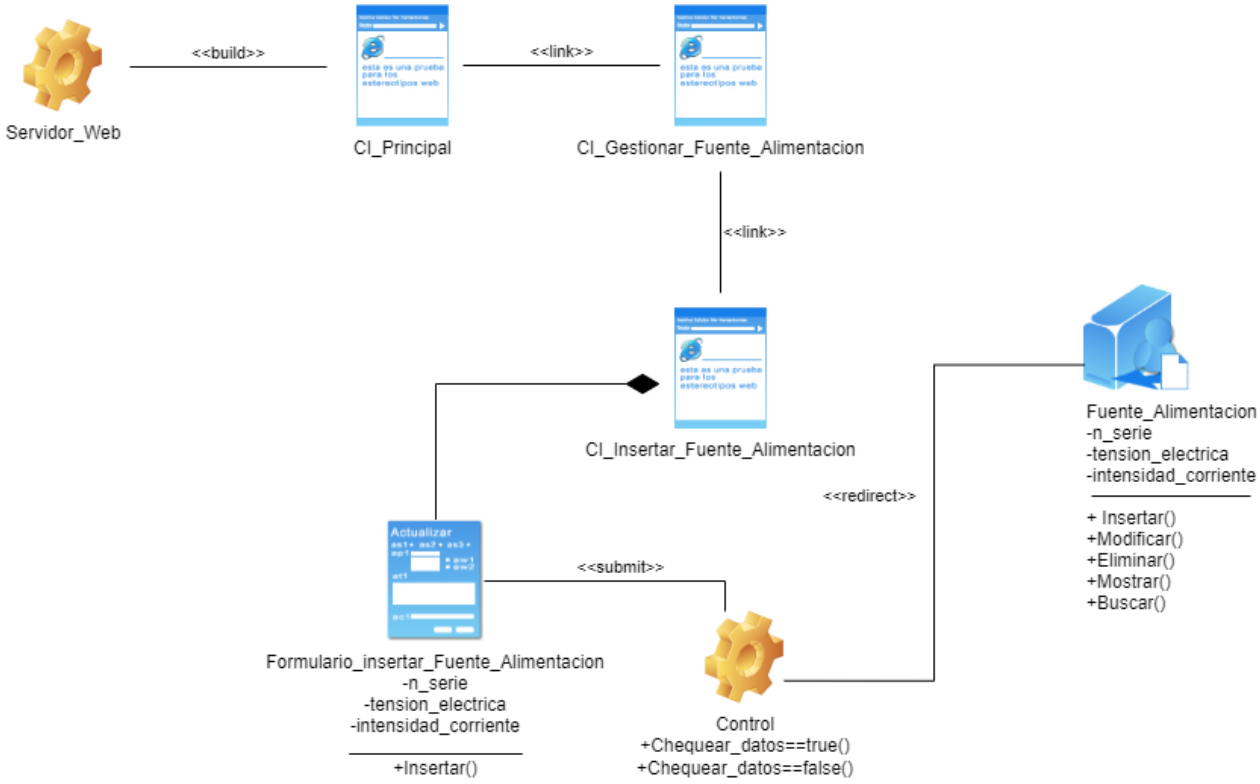

Ilustración 81: Diagrama de diseño <Insertar Fuente de alimentación>

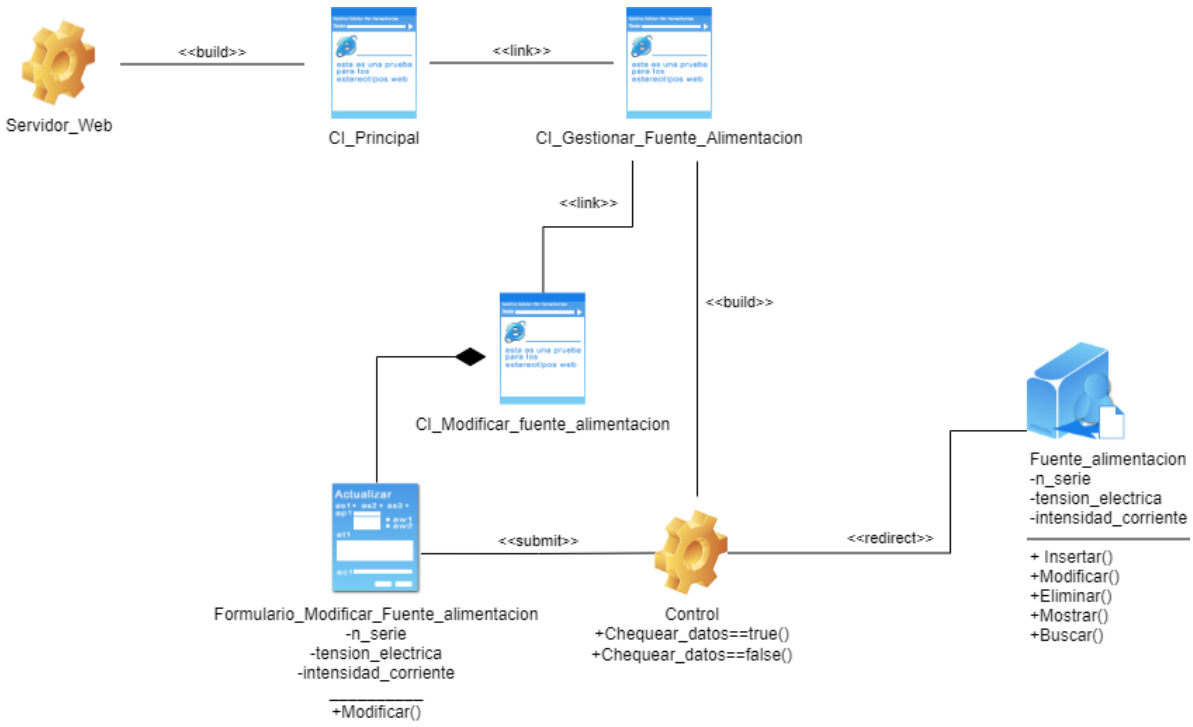

Ilustración 82: Diagrama de diseño <Modificar Fuente de alimentación>

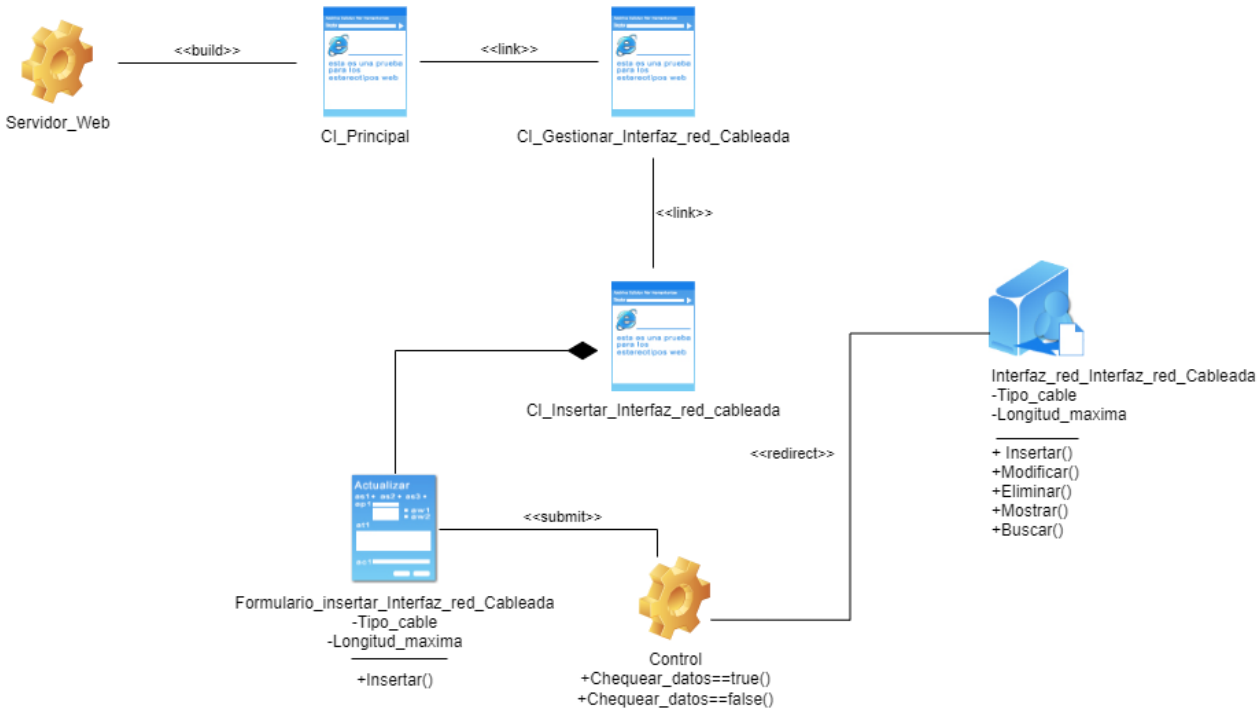

Ilustración 83: Diagrama de diseño <Insertar Interfaz de red cableada>

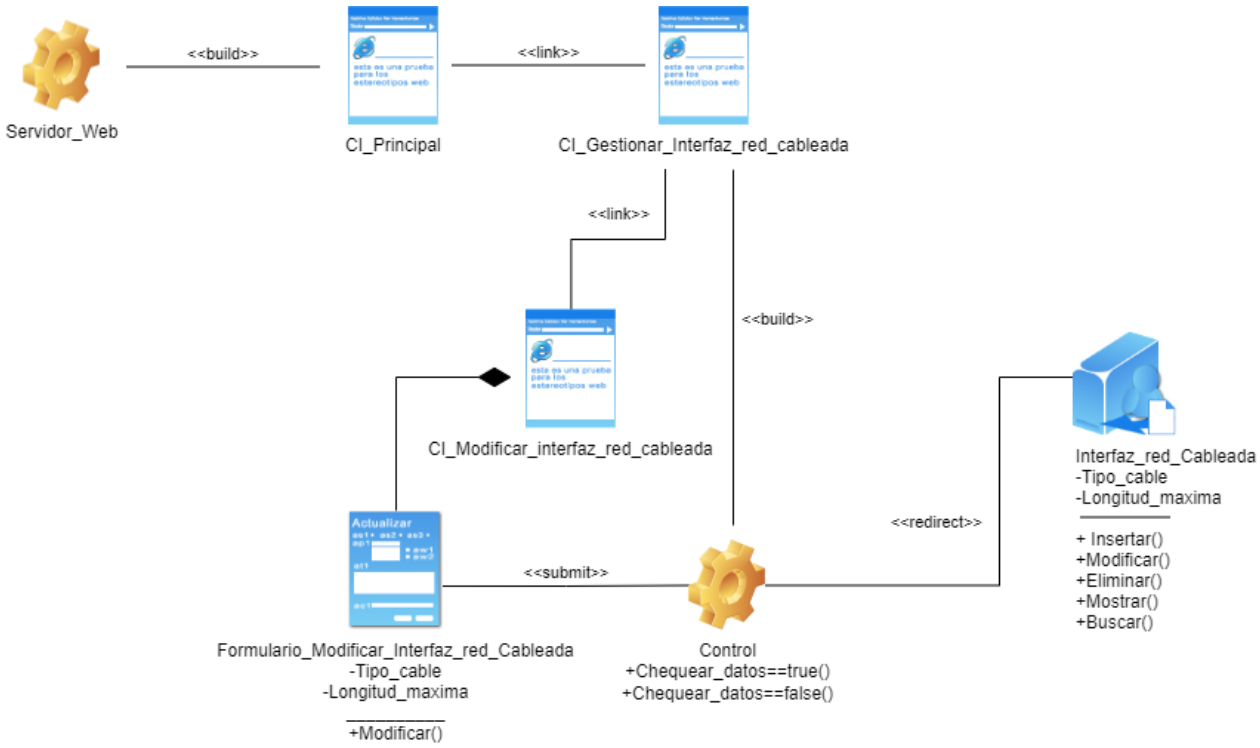

Ilustración 84: Diagrama de diseño <Modificar Interfaz de red cableada>

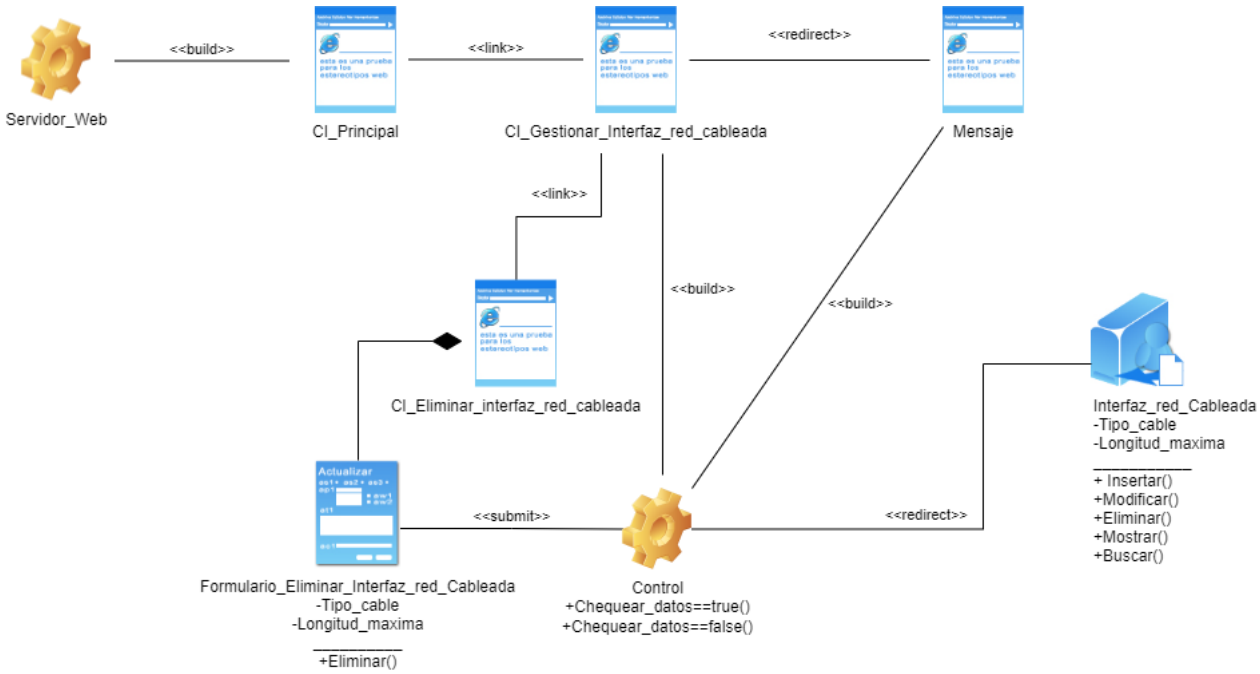

Ilustración 85: Diagrama de diseño <Eliminar Interfaz de red cableada>

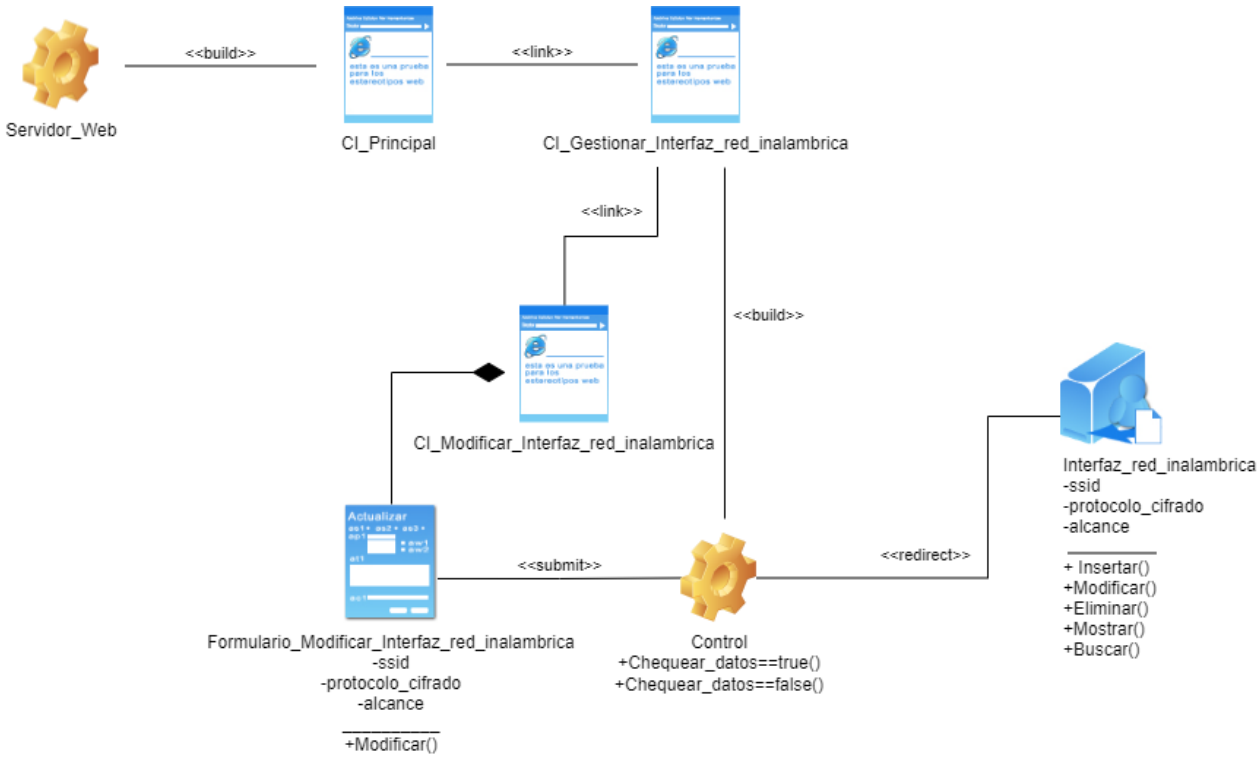

Ilustración 86: Diagrama de diseño <Modificar Interfaz de red inalámbrica>

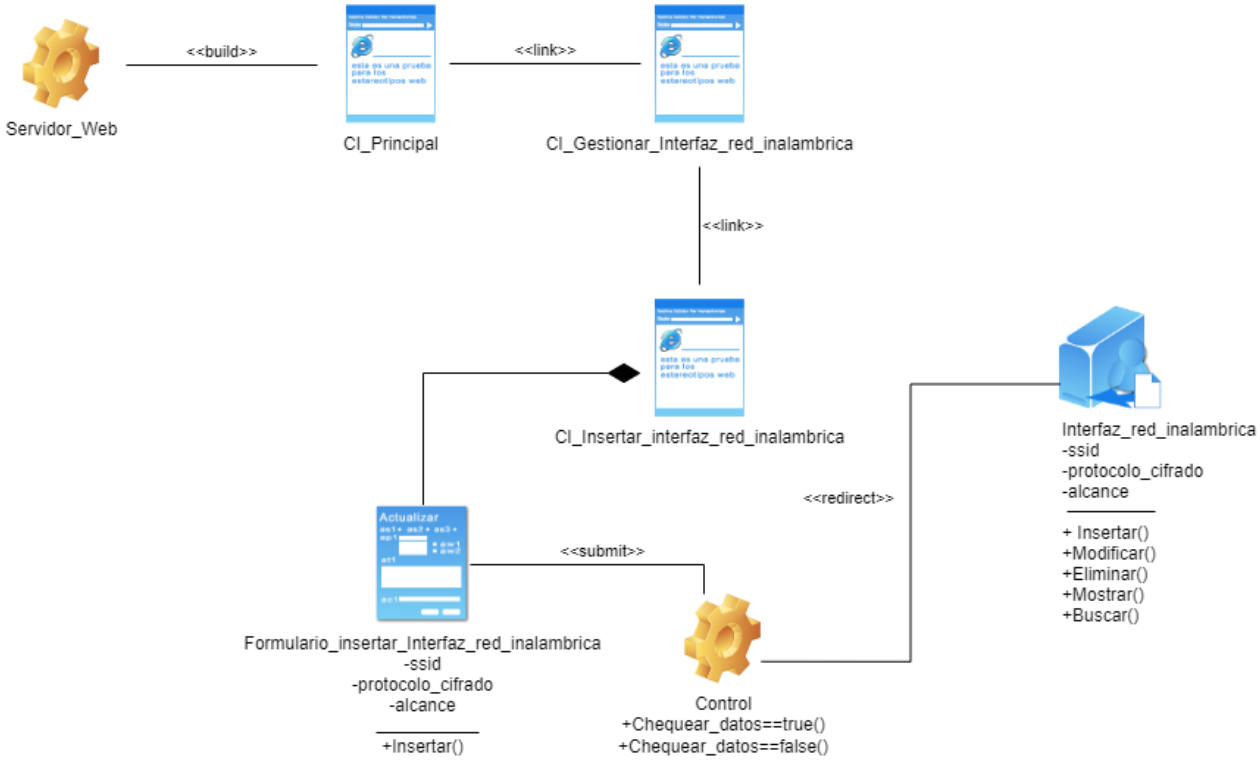

Ilustración 87: Diagrama de diseño <Insertar Interfaz de red inalámbrica>

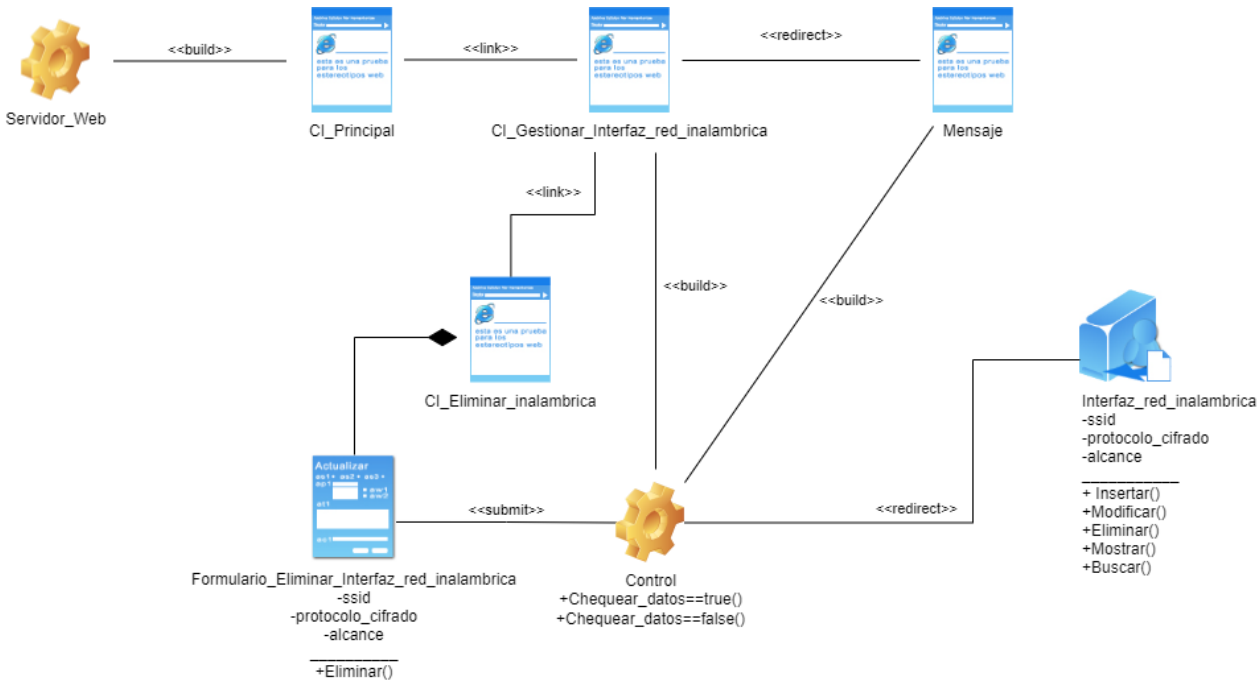

Ilustración 88: Diagrama de diseño <Eliminar Interfaz de red inalámbrica>

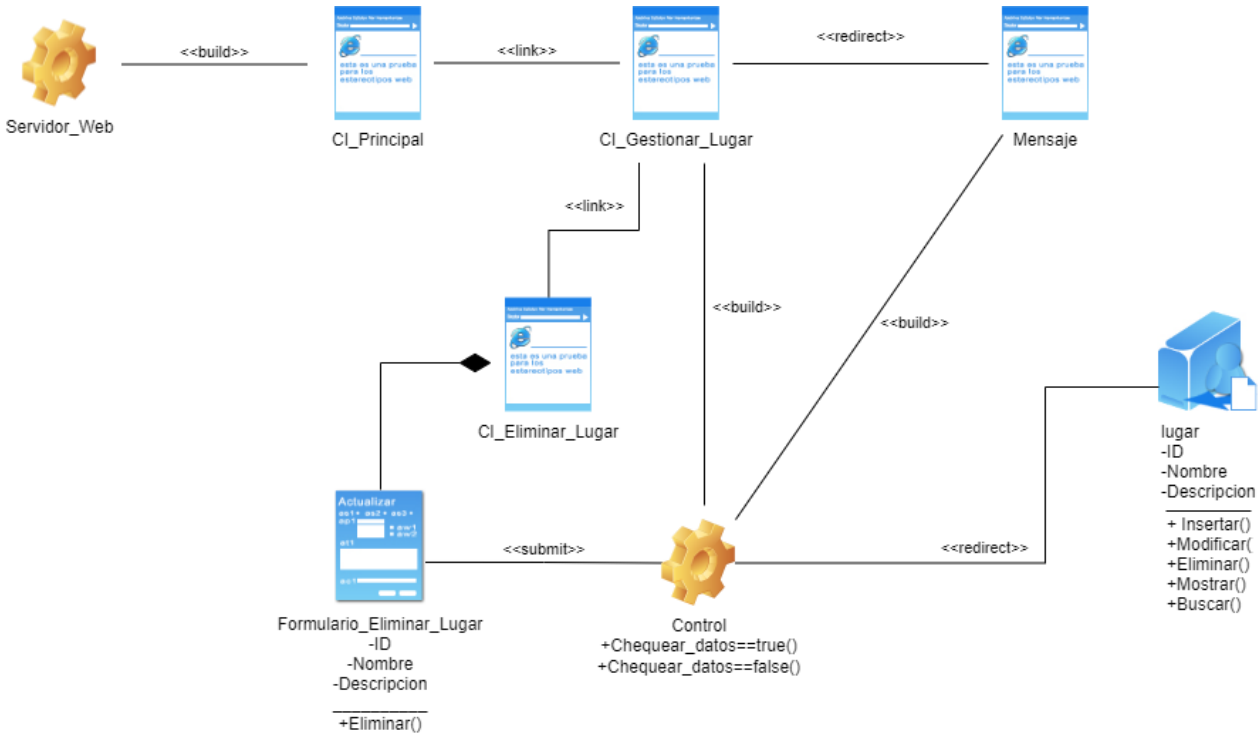

Ilustración 89: Diagrama de diseño <Eliminar Lugar>

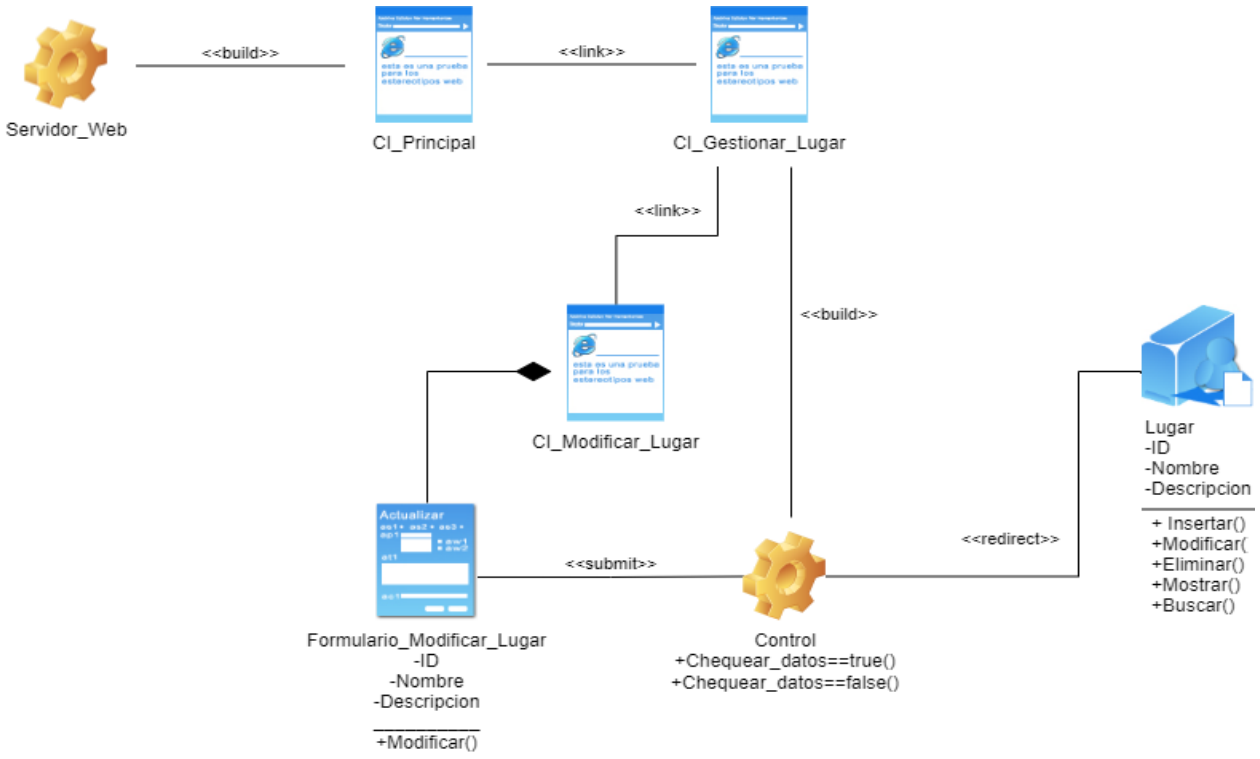

Ilustración 90: Diagrama de diseño <Modificar Lugar>

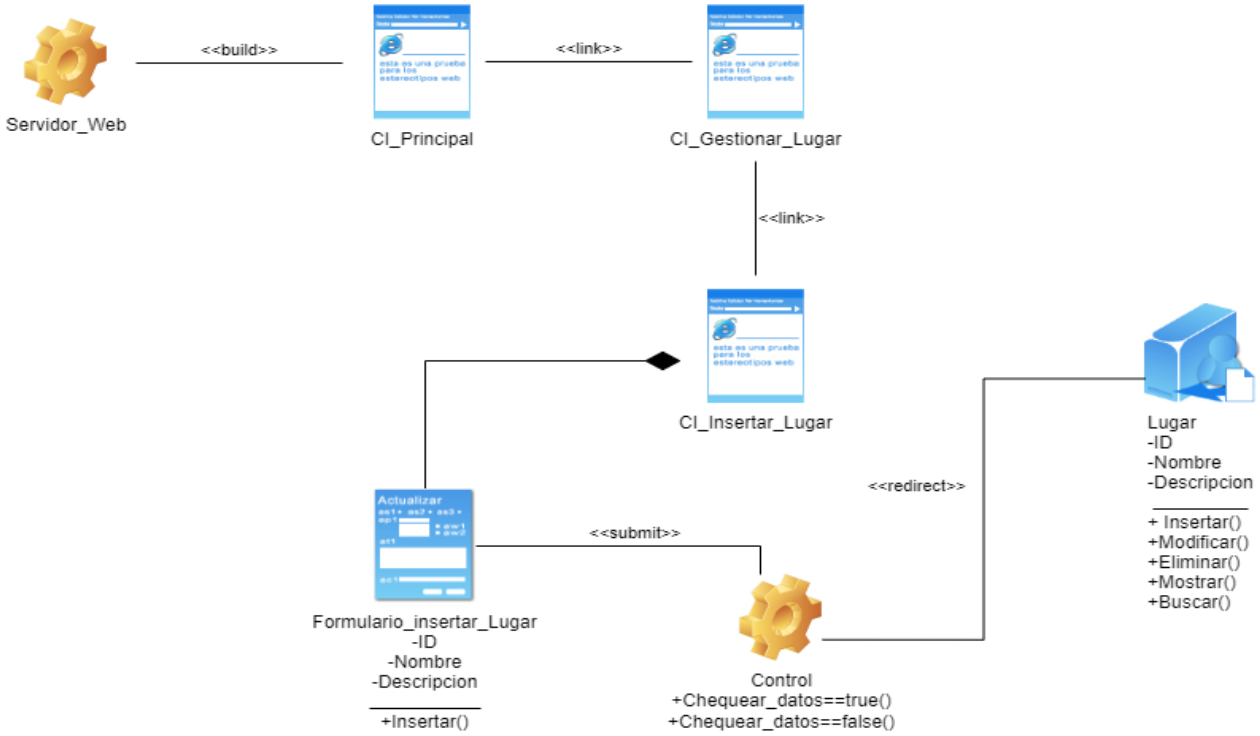

Ilustración 91: Diagrama de diseño <Insertar Lugar>

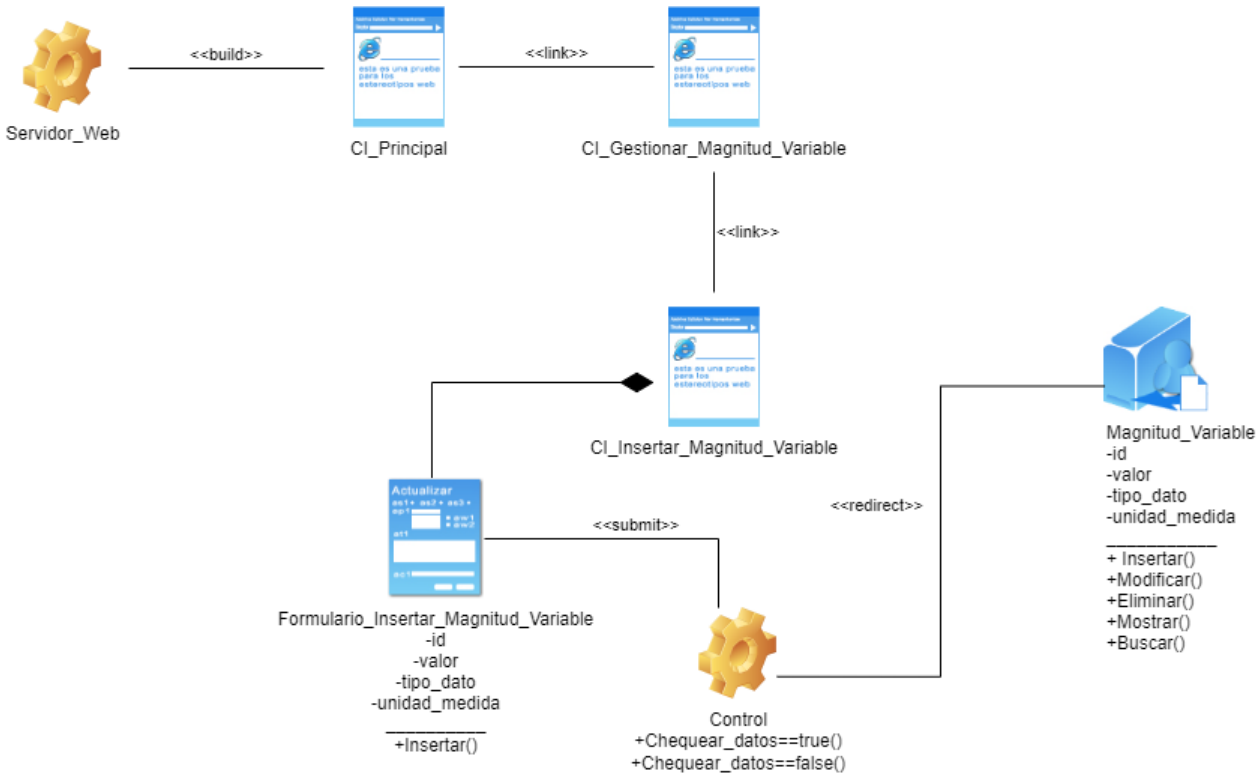

Ilustración 92: Diagrama de diseño <Insertar Magnitud Variable>

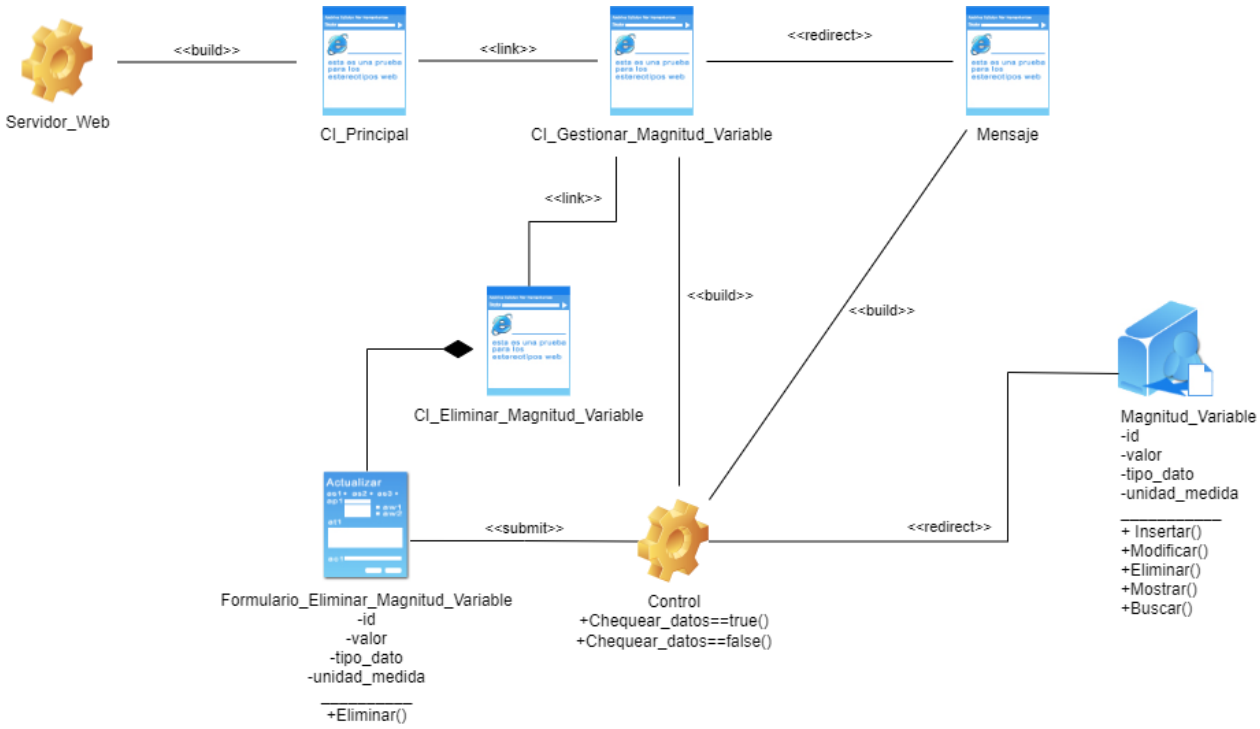

Ilustración 93: Diagrama de diseño <Eliminar Magnitud Variable>

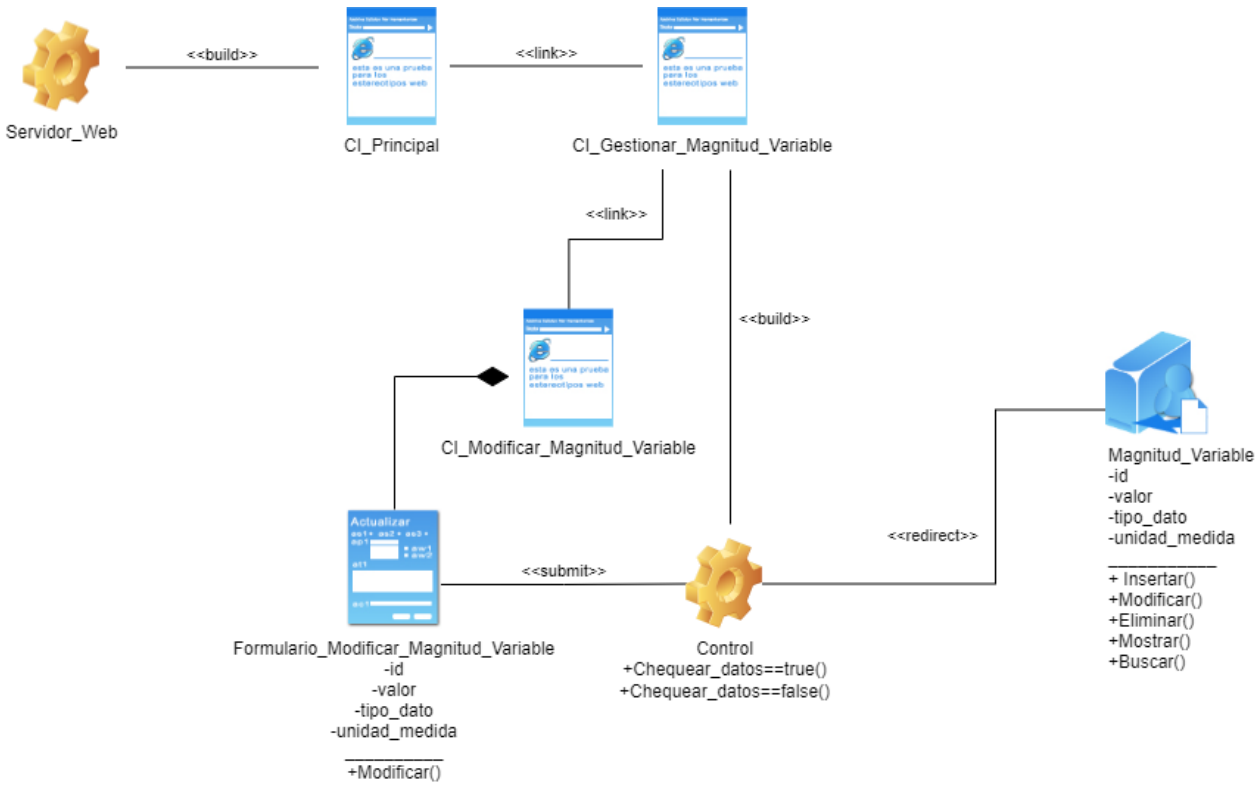

Ilustración 94: Diagrama de diseño <Modificar Magnitud Variable>

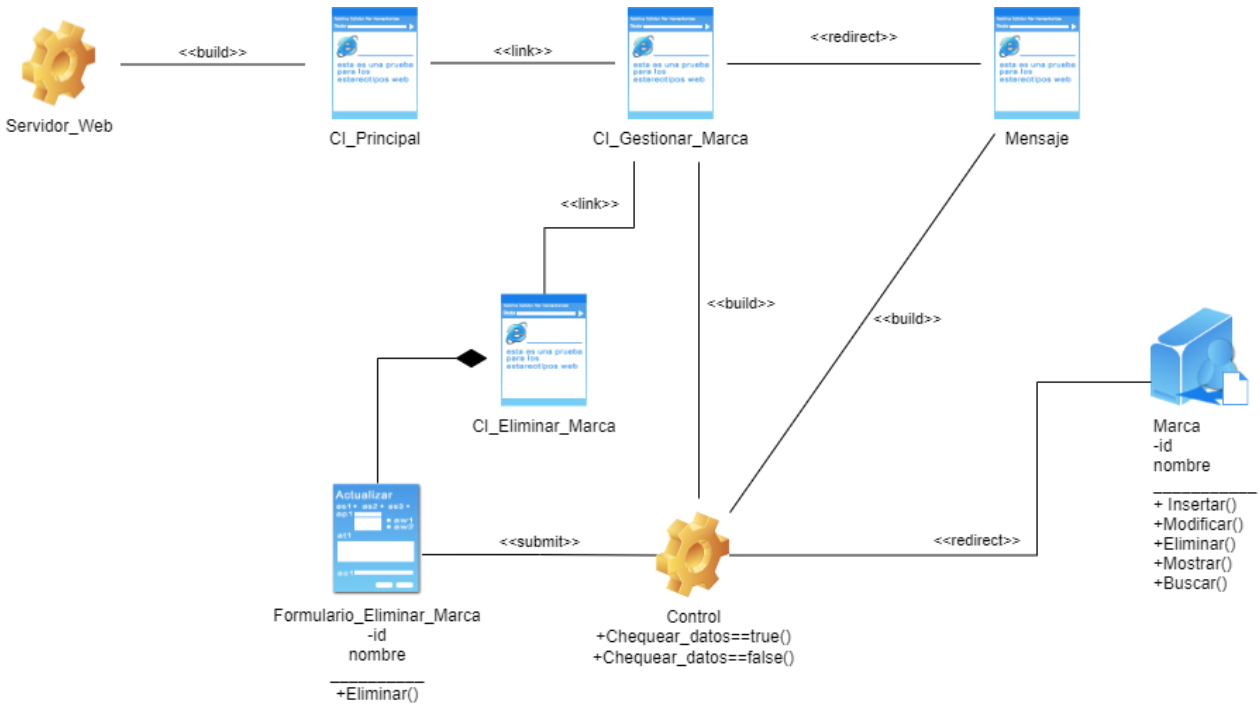

Ilustración 95: Diagrama de diseño <Eliminar Marca>

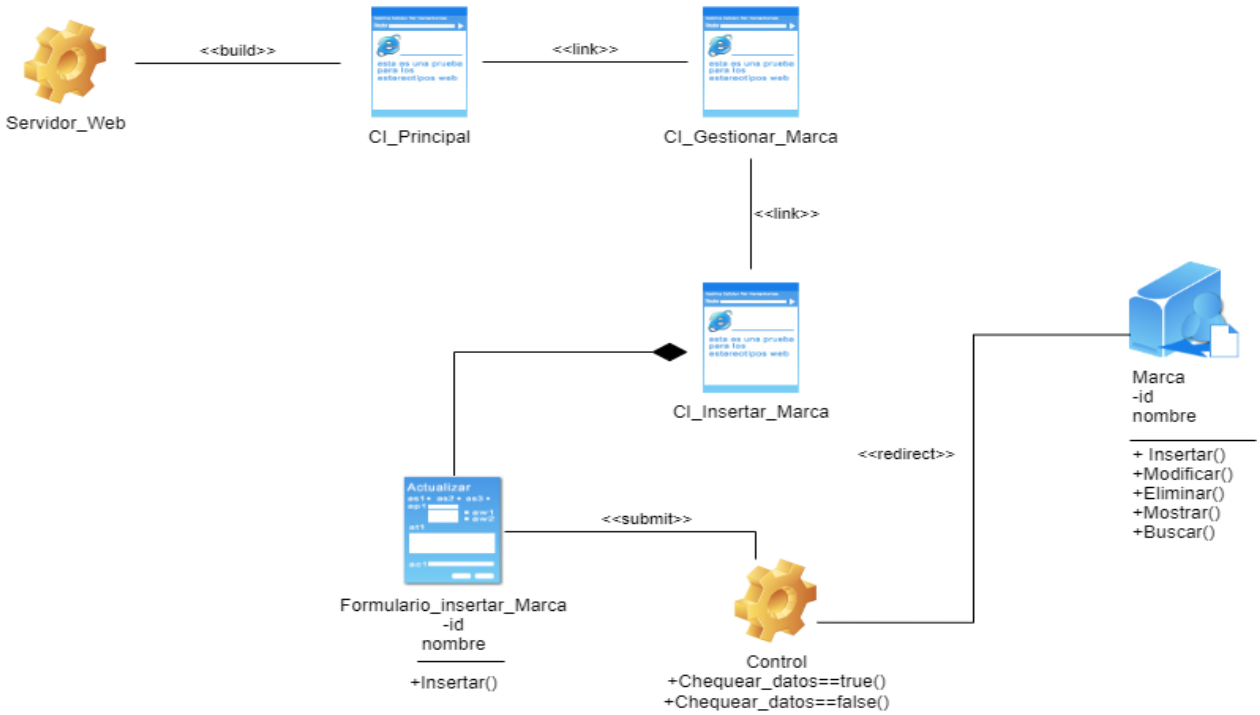

Ilustración 96: Diagrama de diseño <Insertar Marca>

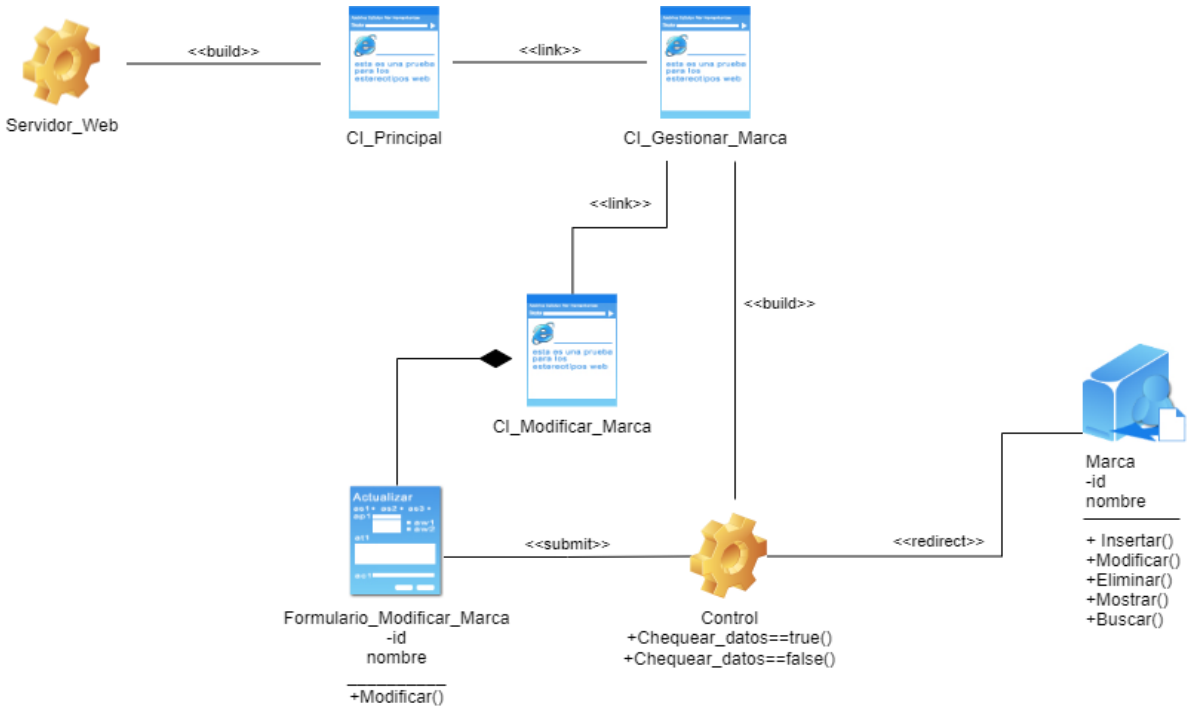

Ilustración 97: Diagrama de diseño <Modificar Marca>

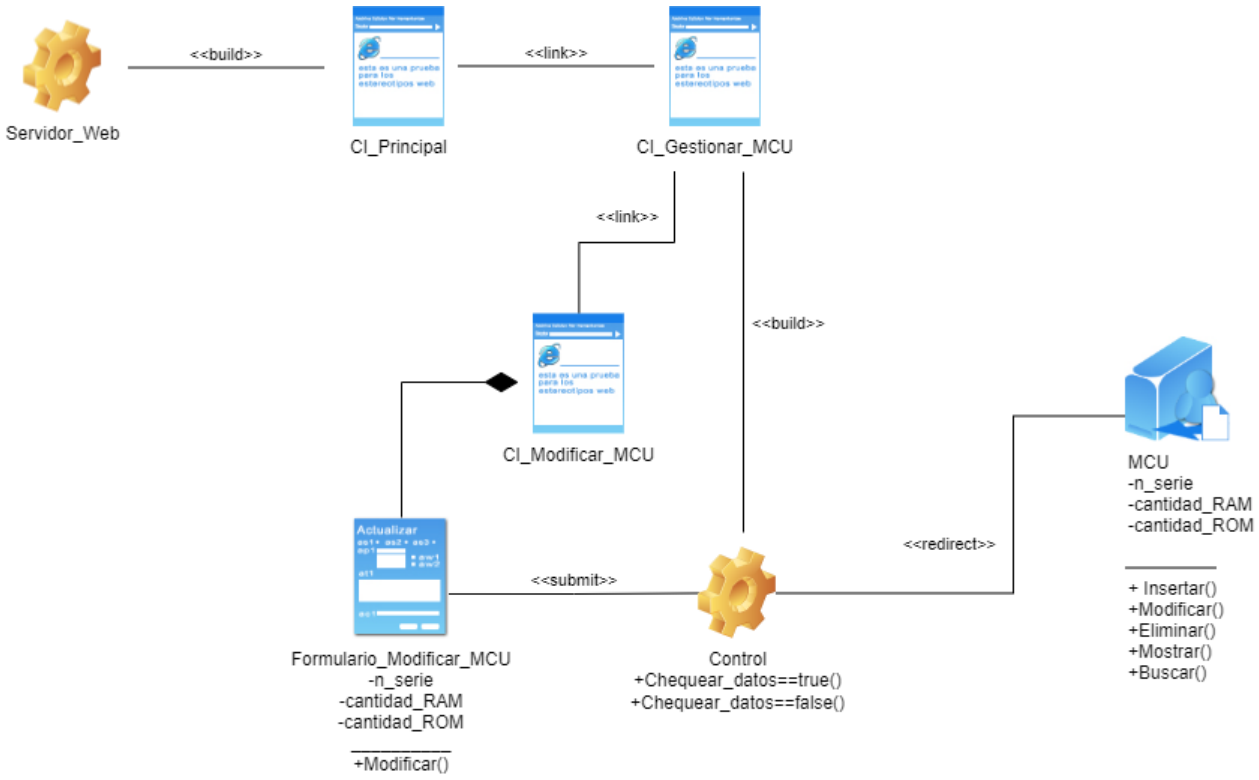

Ilustración 98: Diagrama de diseño <Modificar Unidad de control principal>

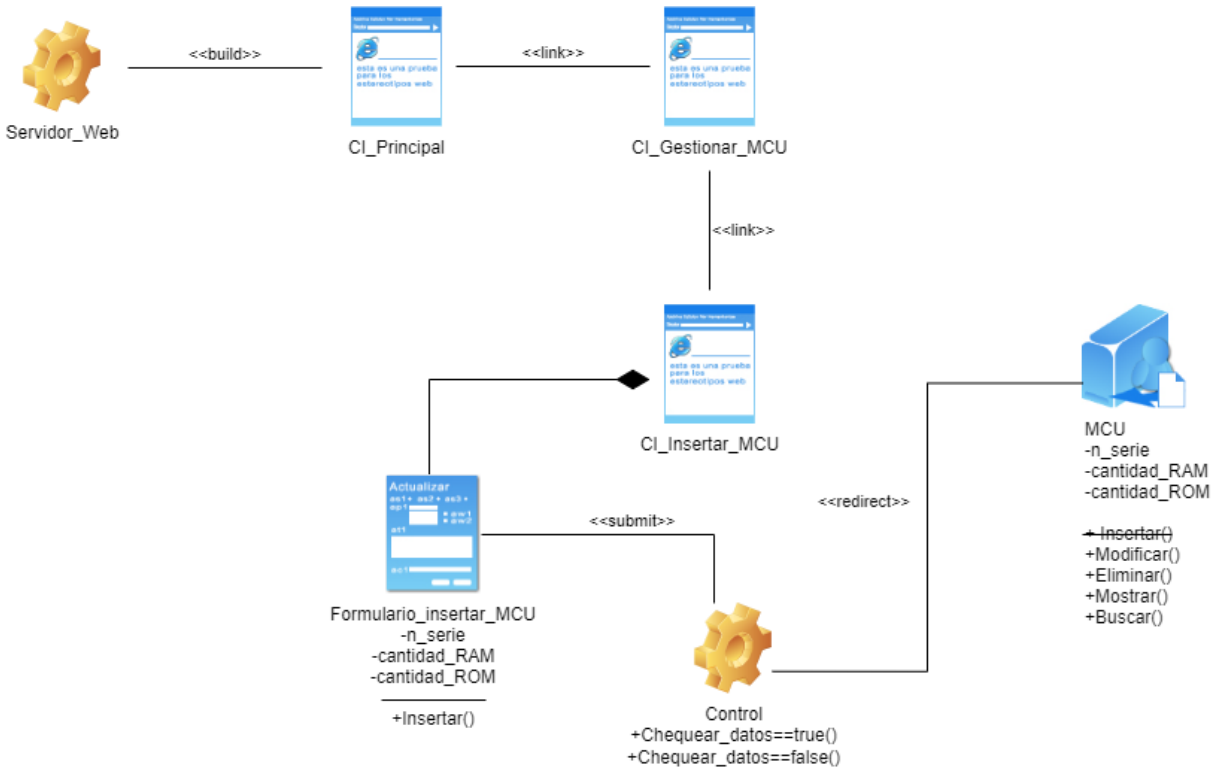

Ilustración 99: Diagrama de diseño <Insertar Unidad de control principal>

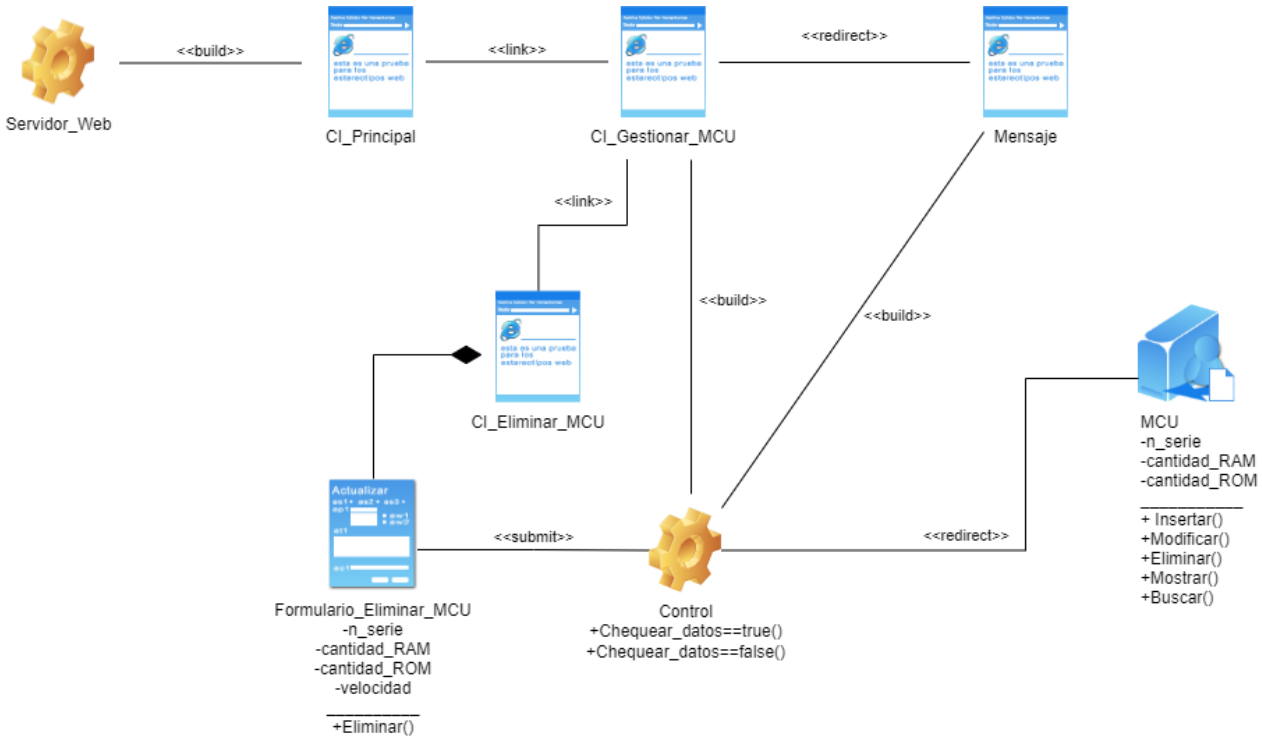

Ilustración 100: Diagrama de diseño <Eliminar Unidad de control principal>

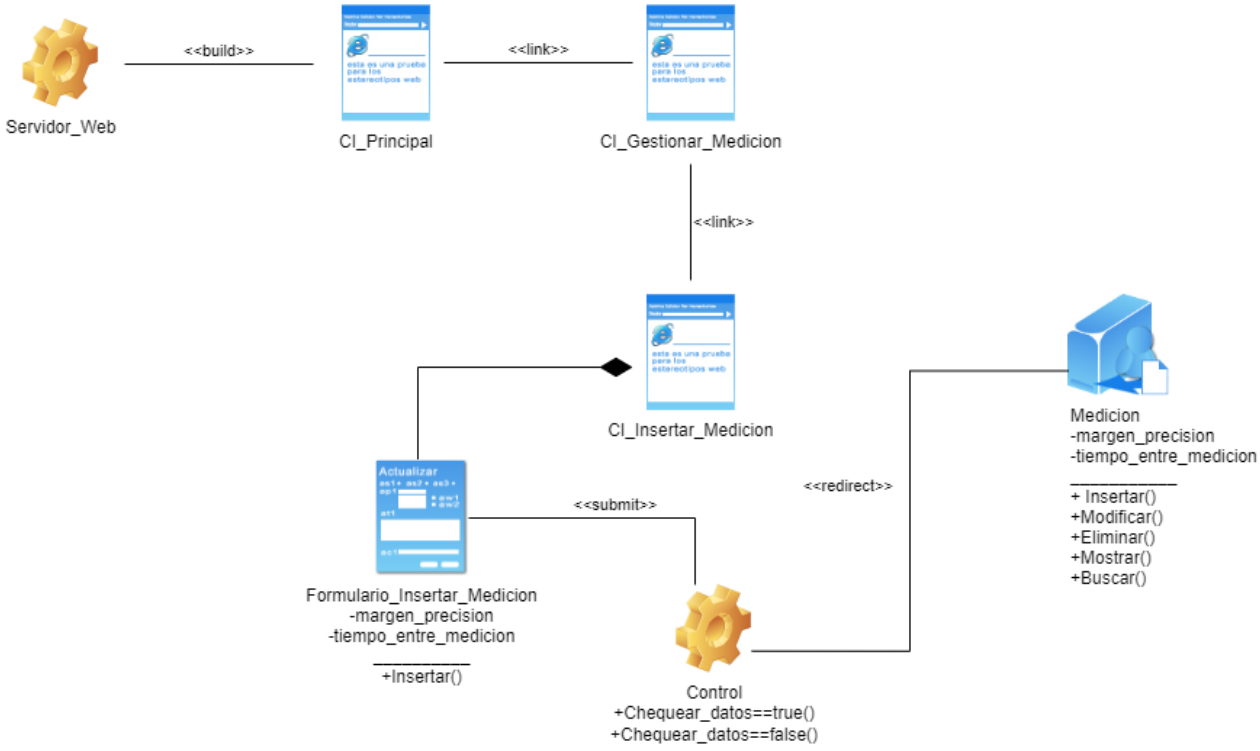

Ilustración 101: Diagrama de diseño <Insertar Medida>

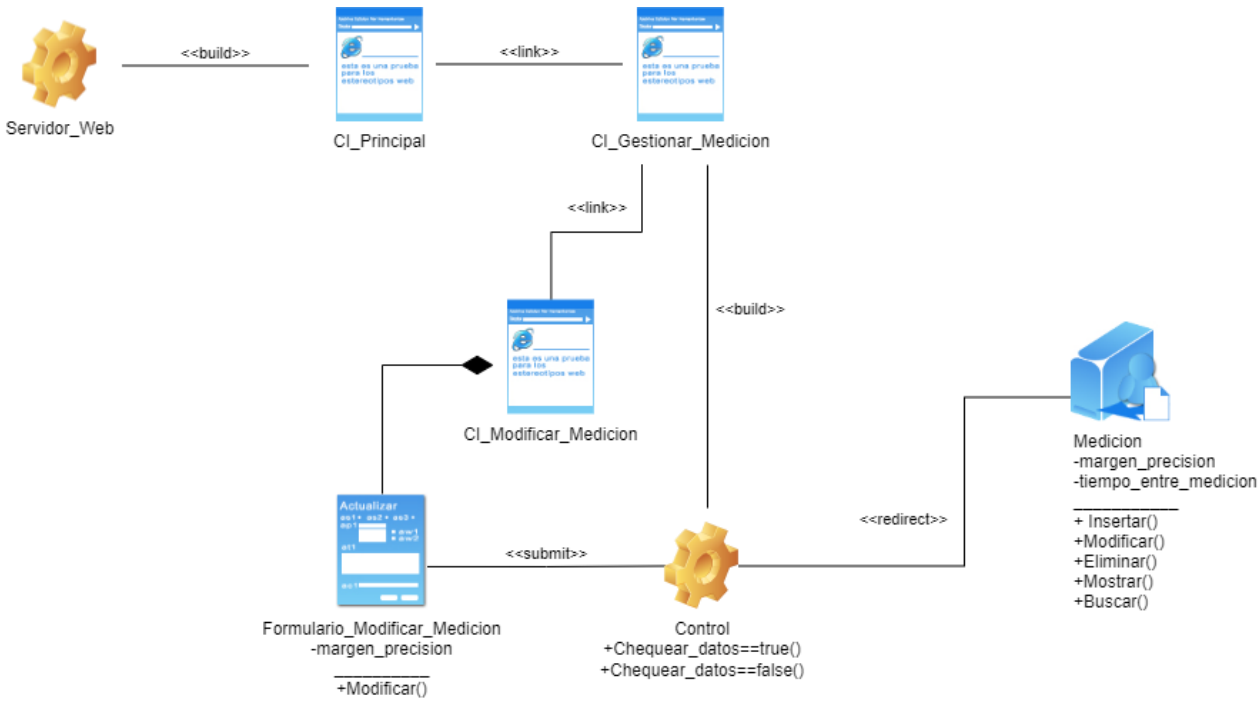

Ilustración 102: Diagrama de diseño <Modificar Medida>

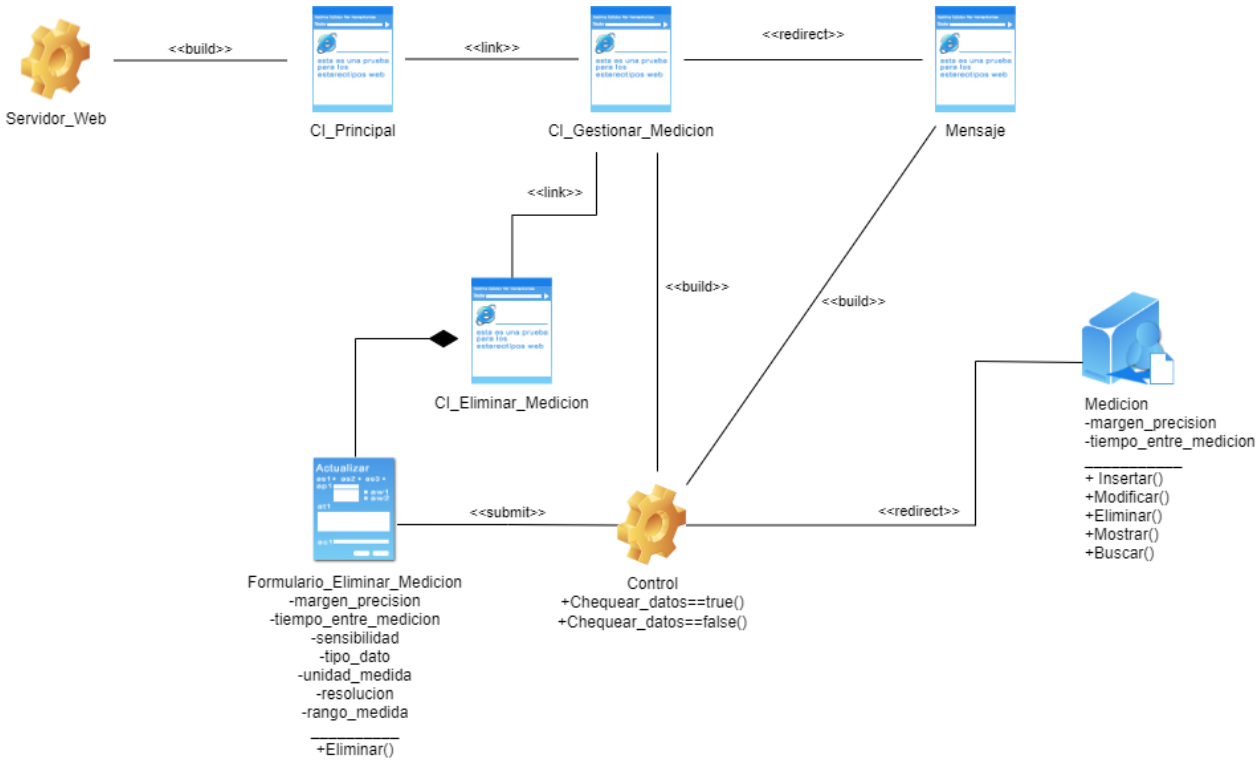

Ilustración 103: Diagrama de diseño <Eliminar Medida>

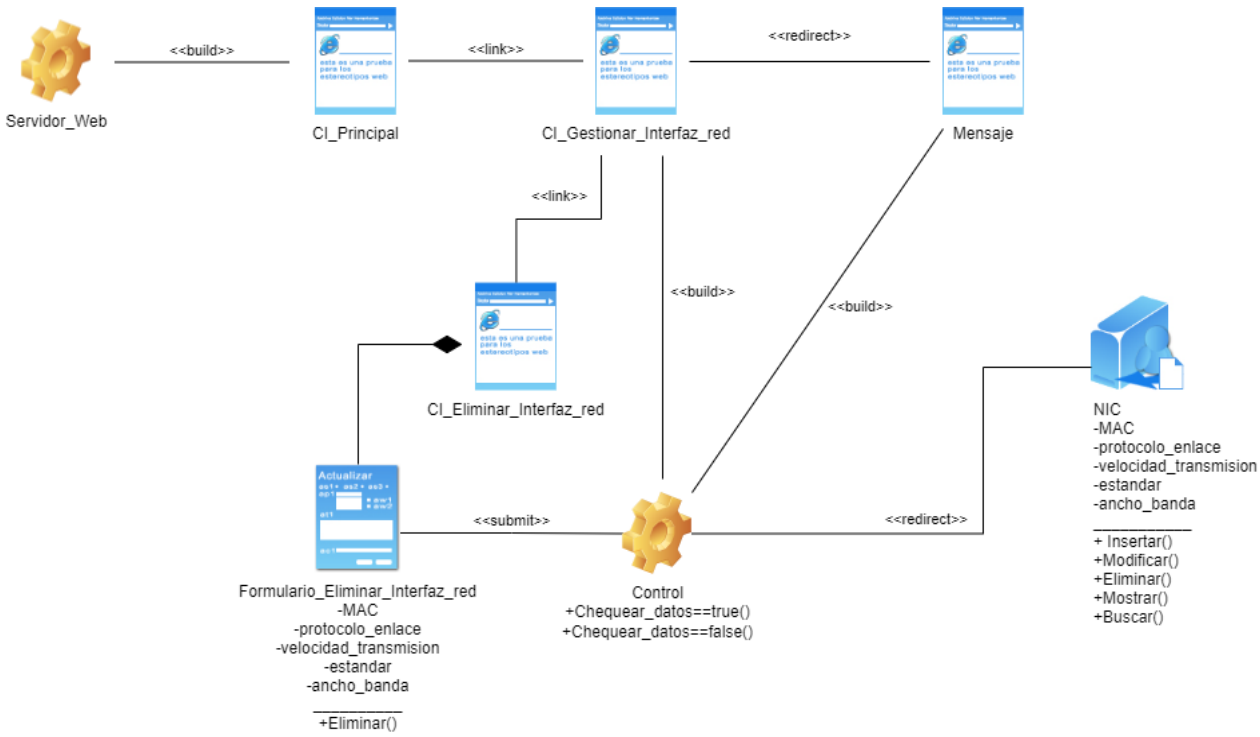

Ilustración 104: Diagrama de diseño <Eliminar Interfaz de Red>

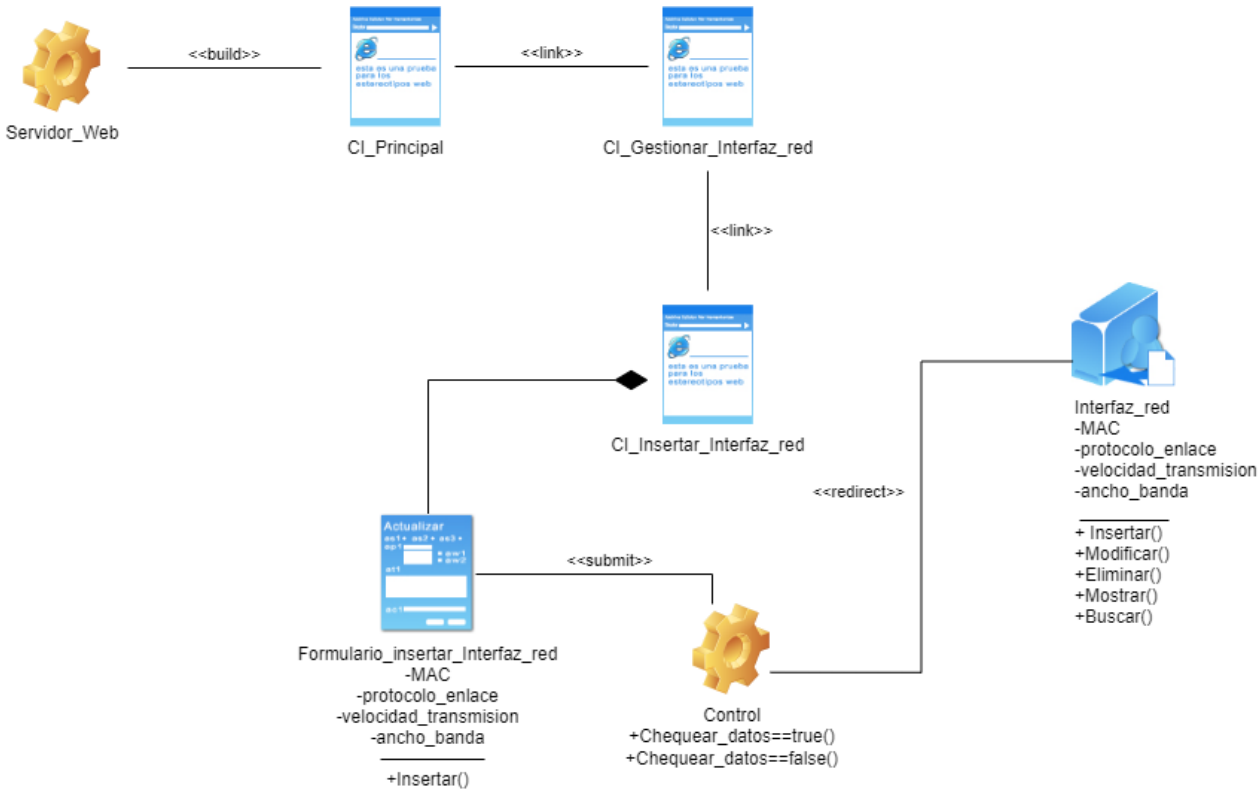

Ilustración 105: Diagrama de diseño <Insertar Interfaz de Red>

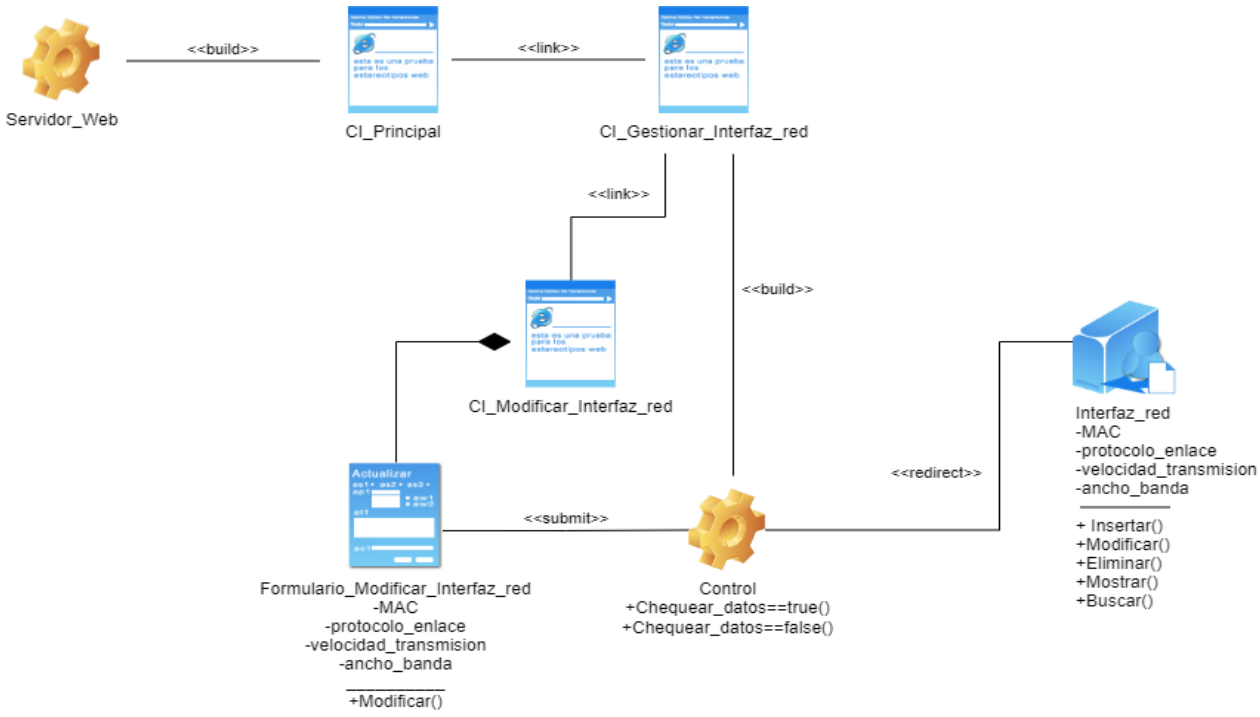

Ilustración 106: Diagrama de diseño <Modificar Interfaz de Red>

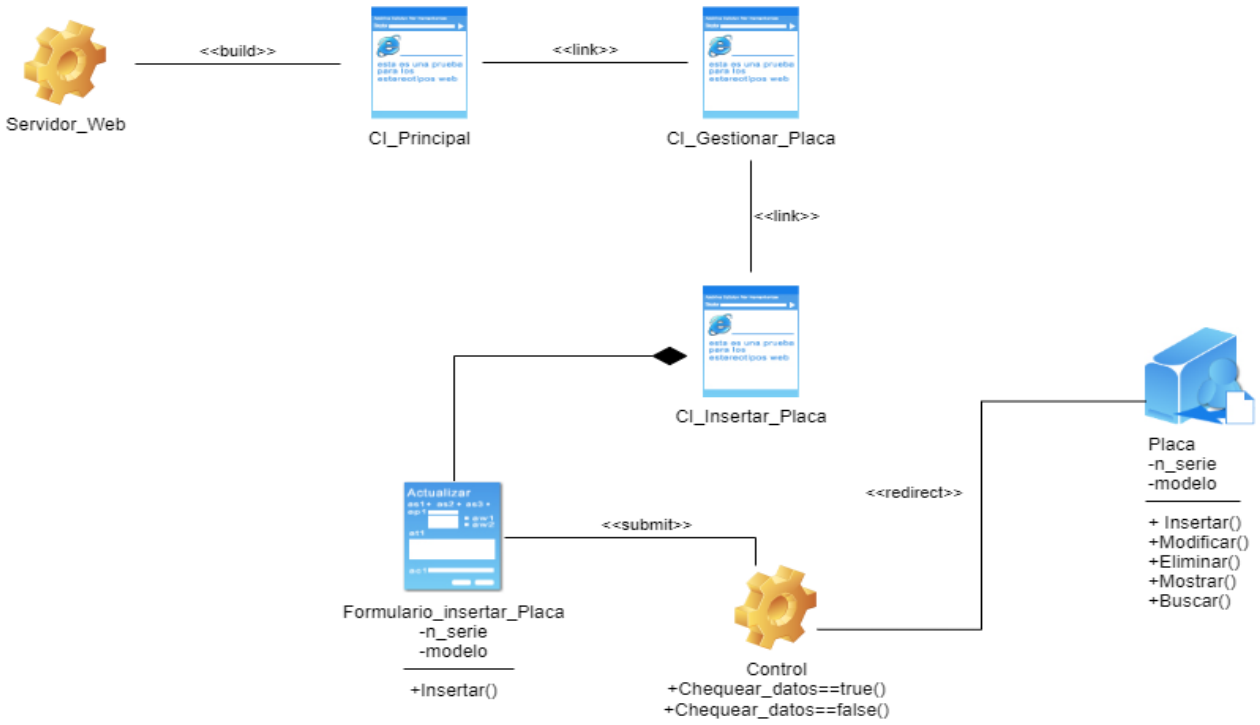

Ilustración 107: Diagrama de diseño <Insertar Placa>

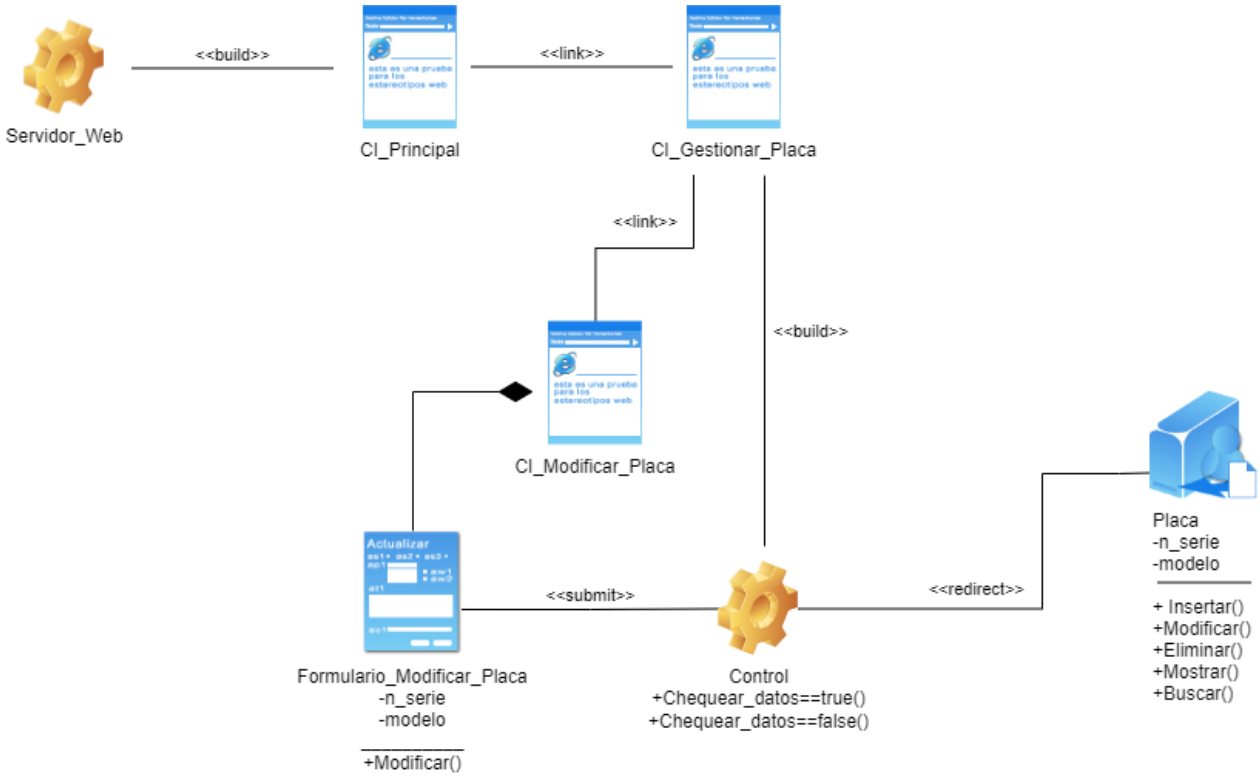

Ilustración 108: Diagrama de diseño <Modificar Placa>

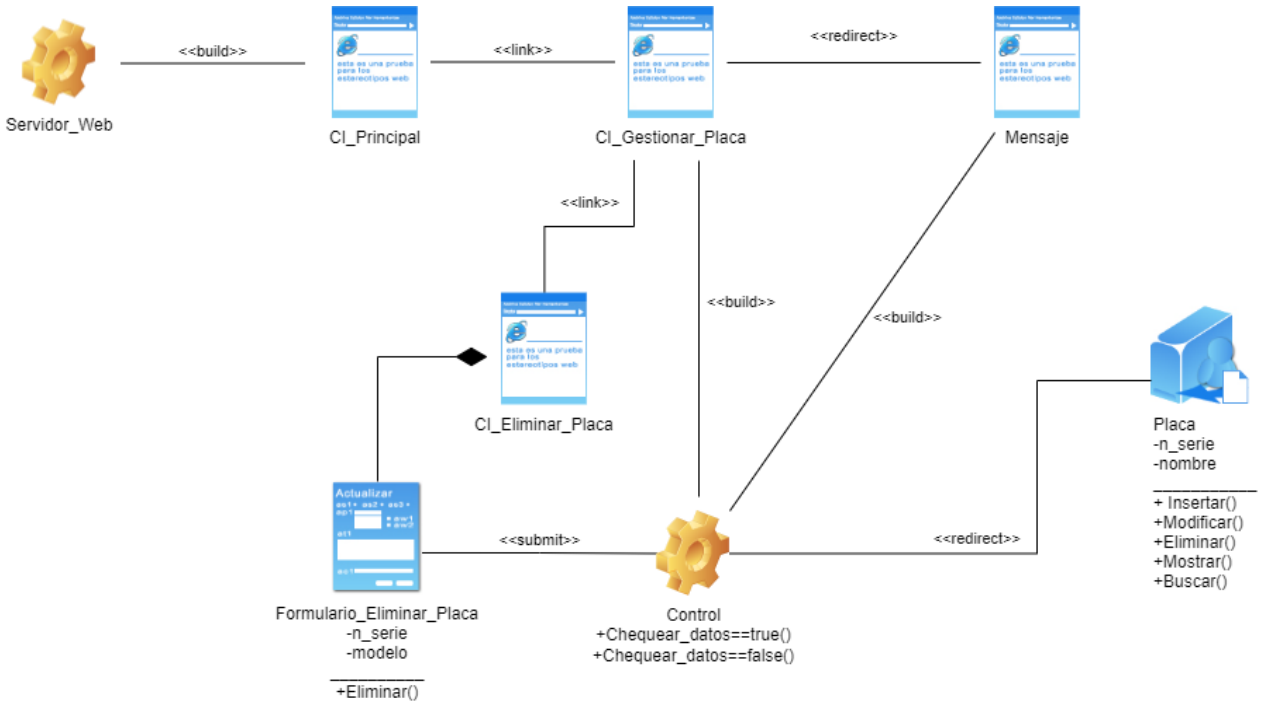

Ilustración 109: Diagrama de diseño <Eliminar Placa>

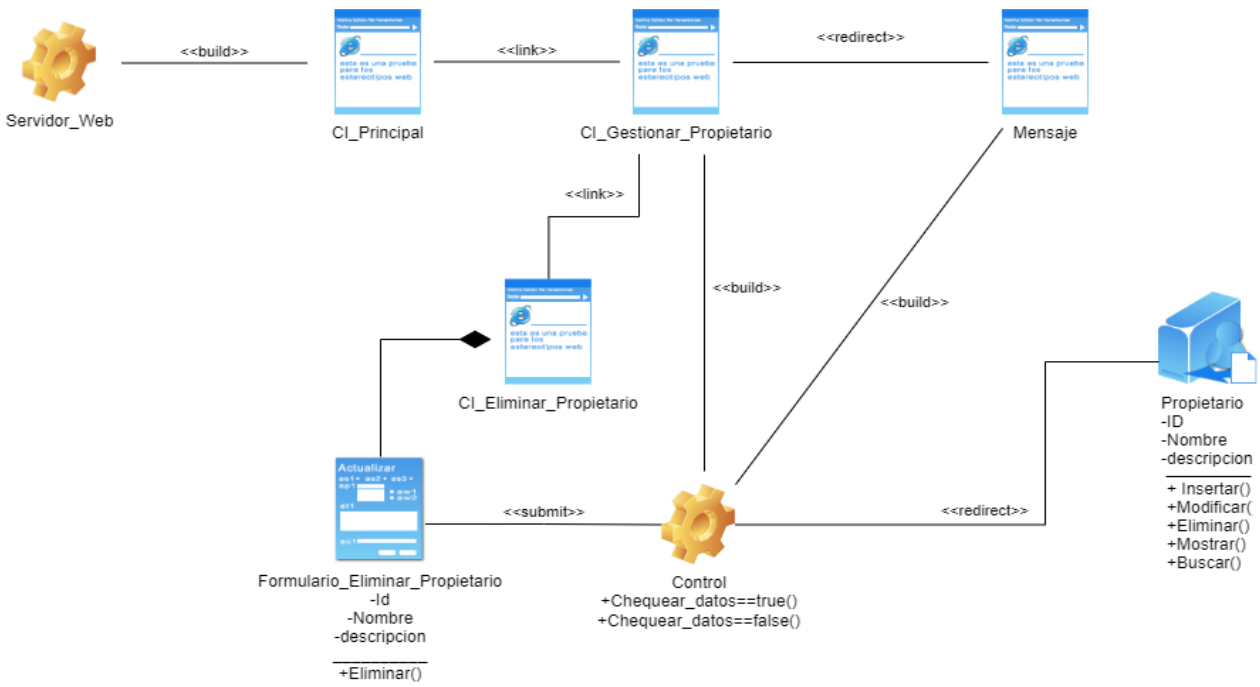

Ilustración 110: Diagrama de diseño <Eliminar Propietario>

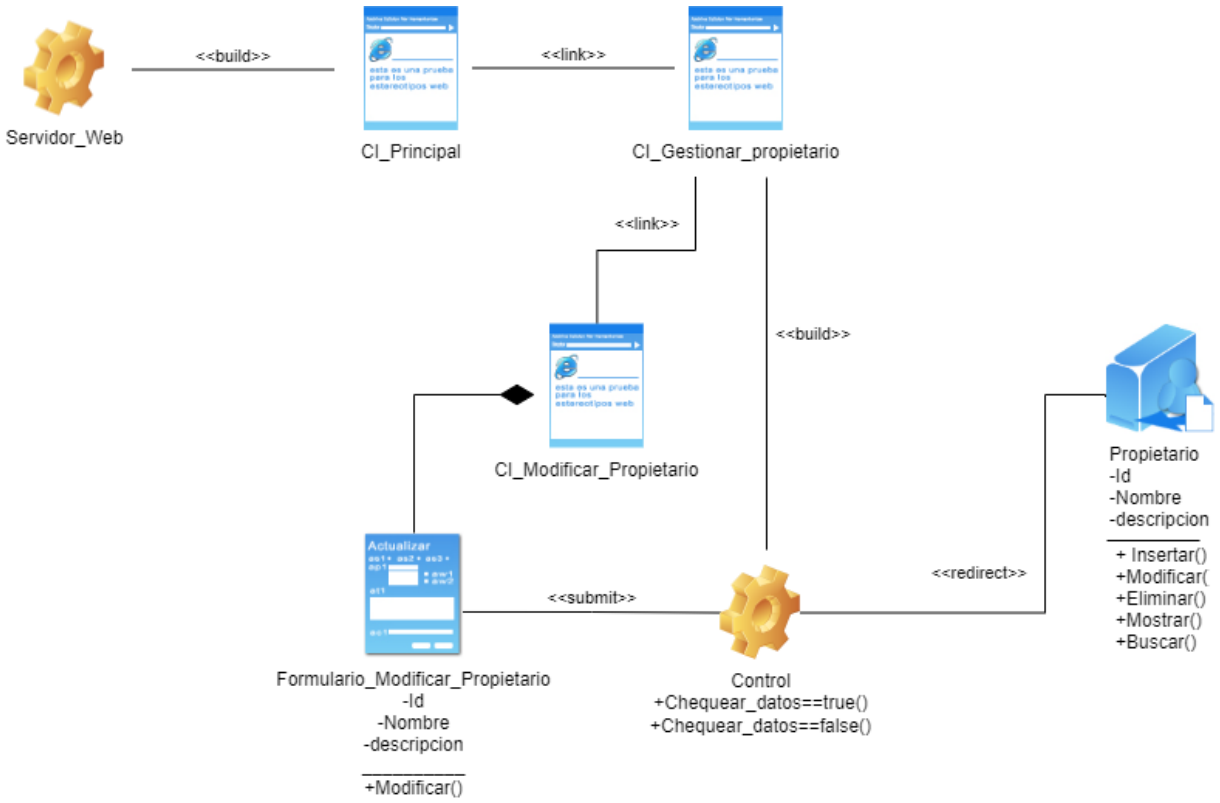

Ilustración 111: Diagrama de diseño <Modificar Propietario>

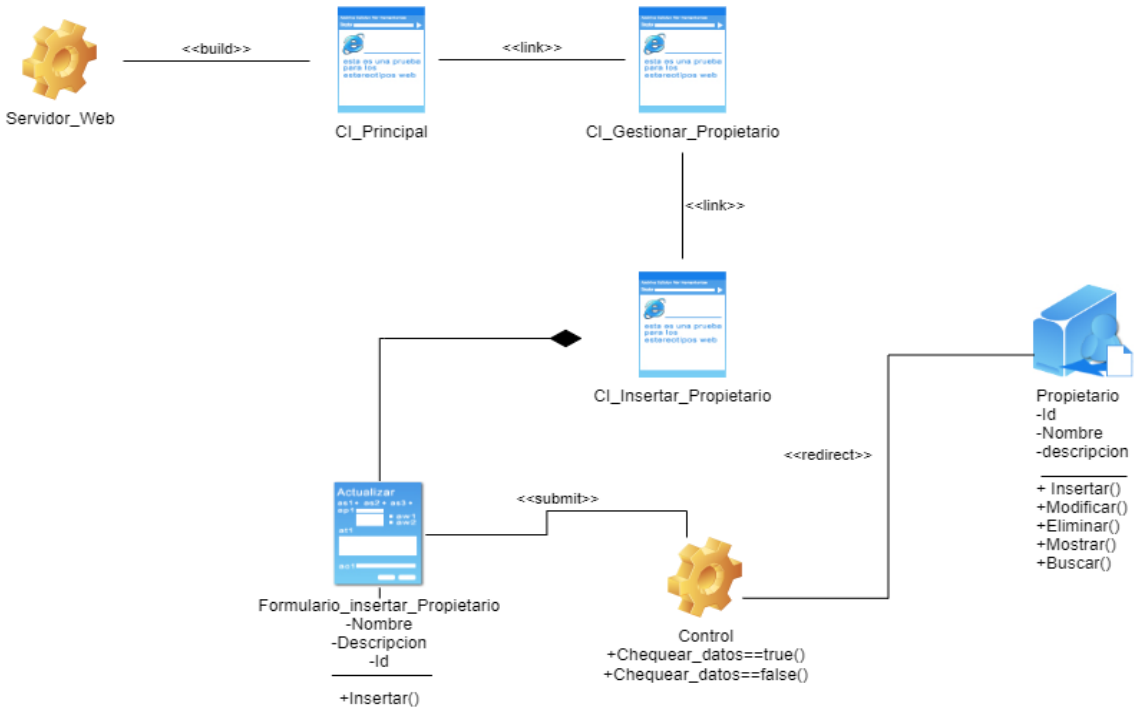

Ilustración 112: Diagrama de diseño <Insertar Propietario>

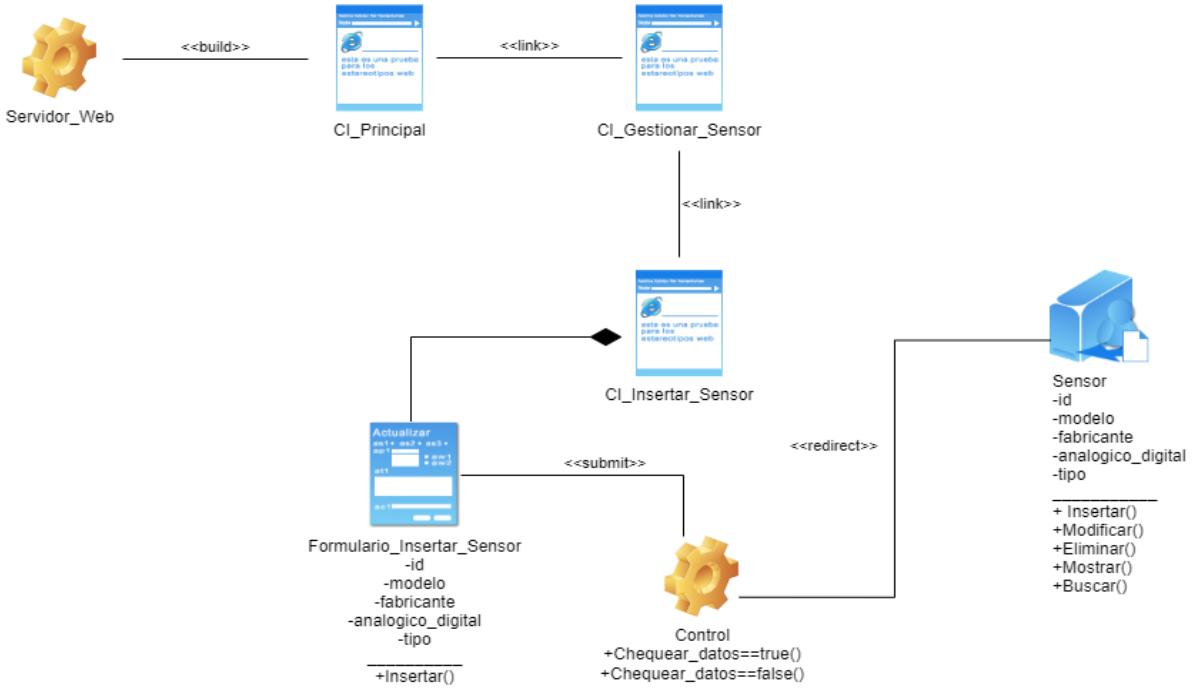

Ilustración 113: Diagrama de diseño <Insertar Sensor>

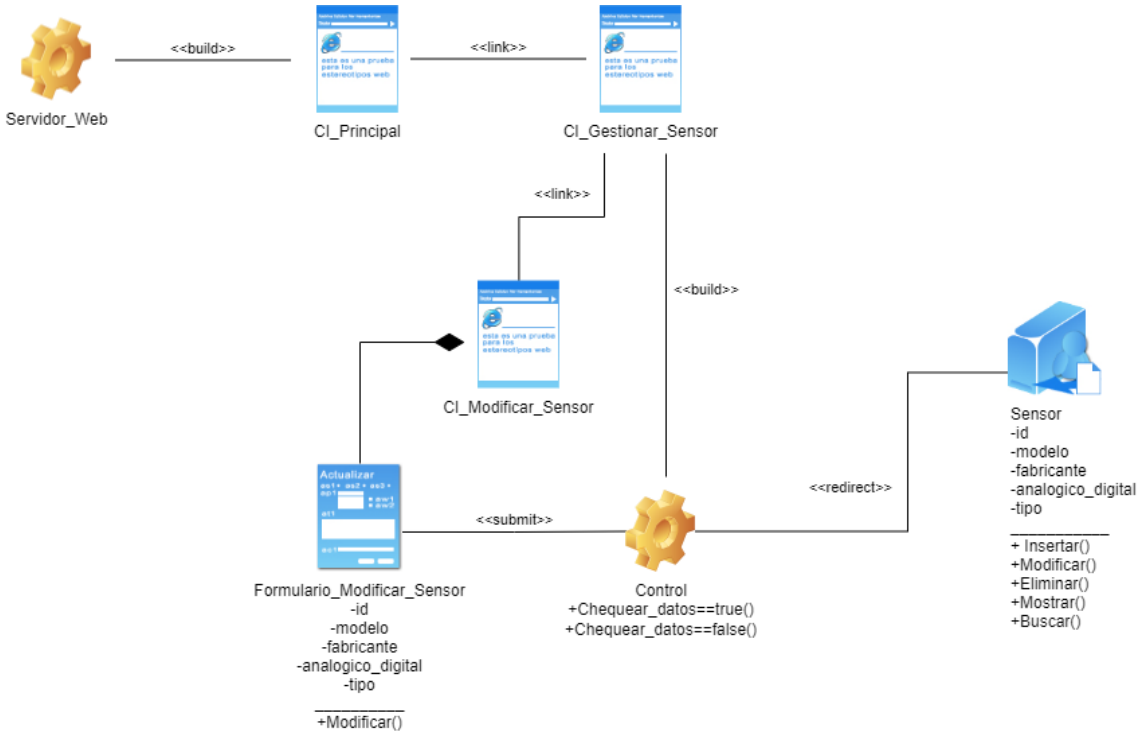

Ilustración 114: Diagrama de diseño <Modificar Sensor>

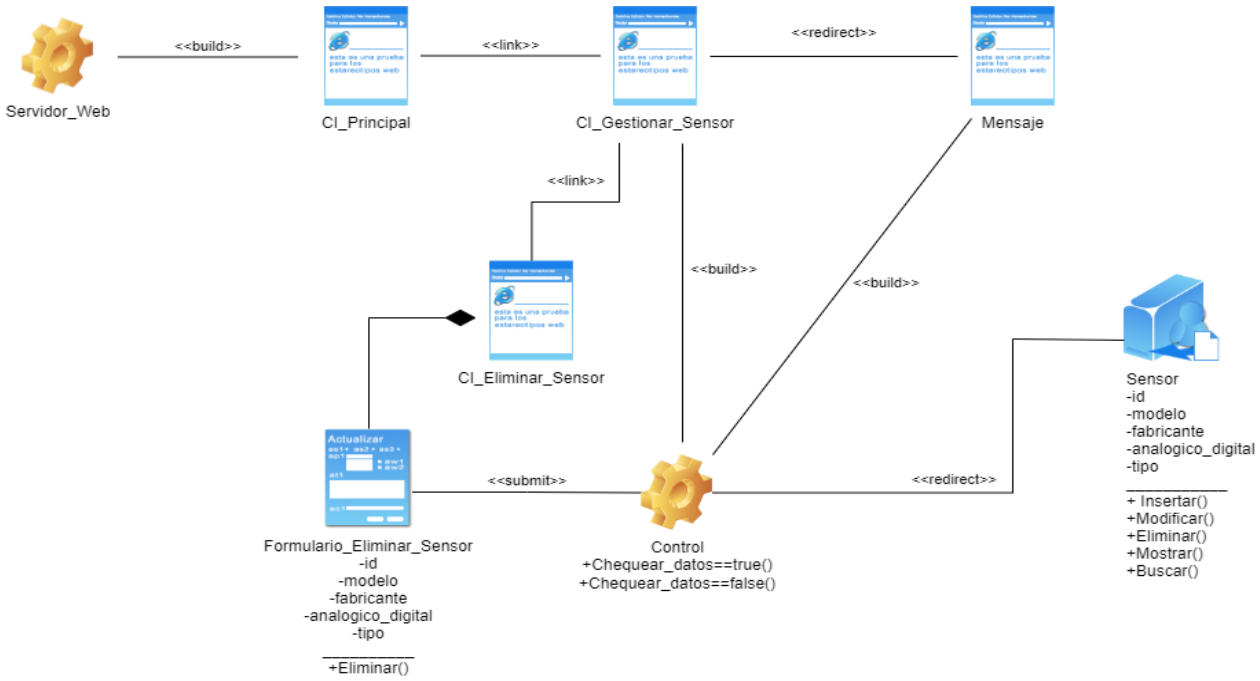

Ilustración 115: Diagrama de diseño <Eliminar Sensor>