UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA, CIENCIA Y COMPUTACIÓN (STEAM)

INGENIERÍA EN SISTEMAS INDUSTRIALES

ELECTRÓNICA Y AUTOMÁTICA

TRABAJO FIN DE GRADO

OPTIMIZACIÓN DE LOS SISTEMAS DE PROTECCIÓN Y REGULACIÓN EN REDES DE DISTRIBUCIÓN CON RECURSOS ENERGÉTICOS DISTRIBUIDOS MEDIANTE INTELIGENCIA ARTIFICIAL

Alumno: D. ÁLVARO CARMONA PÉREZ-LANZAC

Director: D. GUSTAVO BERMEJO MARTÍN

JUNIO 2026

TÍTULO: Optimización de los Sistemas de Protección y Regulación en Redes de Distribución con Recursos Energéticos Distribuidos mediante Inteligencia Artificial.

AUTOR: Álvaro Carmona Pérez-Lanzac

DIRECTOR DEL PROYECTO: Gustavo Bermejo Martín

FECHA: 7 de junio de 2026

RESUMEN

Este Trabajo de Fin de Grado presenta el desarrollo y validación de un gemelo digital para la gestión de una red eléctrica Inteligente o Smart Grid, basado en un sistema multi-agente orquestado por inteligencia artificial de borde, Edge IA. Esta aproximación a la descentralización de la red permite aislar los problemas técnicos o fallos que se pueden llegar a producir por zonas o barrios. Utilizando el simulador OpenDSS y un motor de control en Python, se ha replicado virtualmente el comportamiento dinámico de una red de distribución eléctrica con alta influencia de generación fotovoltaica y sistemas de almacenamiento energético en baterías BESS. Como factor de innovación se presenta el uso de un modelo de lenguaje LLM local, Llama 3.2 3B, integrado mediante n8n para la toma de decisiones autónomas en tiempo real. Se han simulado tres escenarios críticos: condiciones nominales, picos de demanda por calefacción y el famoso "apagón" del 28 de abril de 2025. Estos escenarios simulados corresponden a un barrio típico urbano de inmuebles unifamiliares, pero plantean un modelo escalable a ciudades enteras. Los resultados muestran que la arquitectura diseñada logra estabilizar la red con una latencia reducida y una alta eficiencia operativa, justificando la viabilidad de implementar el control de infraestructuras críticas con modelos de IA locales sin dependencia de la nube. Frente a las soluciones del mercado actual, este proyecto fundamenta su valor añadido en garantizar el control total de los datos, reducir los gastos de material y de mantenimiento de los activos físicos al proteger la red, blindar la ciberseguridad frente a ataques externos y eliminar los costes de suscripción a plataformas *Cloud* externas.

Palabras clave: Gemelo Digital, Smart Grid, Edge IA, Sistemas Multi-Agente, LLM y BESS.

ABSTRACT

This Thesis presents the development and validation of a digital twin for Smart Grid management, based on a multi-agent system orchestrated by Edge Artificial Intelligence. This approach to grid decentralization allows technical issues and faults to be isolated by specific zones or neighborhoods. Using the OpenDSS simulator and a Python-based control engine, the dynamic behavior of an electrical distribution network with a high penetration of photovoltaic generation and Battery Energy Storage Systems has been virtually replicated. A key innovation of this research is the deployment of a local Large Language Model, Llama 3.2 3B, integrated via n8n for autonomous, real-time decision-making. Three critical scenarios were simulated: nominal conditions, heating-induced demand peaks, and the well-known power outage of April 28, 2025. Although these simulations model a typical urban neighborhood comprising single-family homes, the proposed architecture is easily scalable to entire cities. The results demonstrate that the designed system successfully stabilizes the grid with reduced latency and high operational efficiency, proving the viability of managing critical infrastructure through local AI models without cloud dependency. Compared to current market alternatives, the added value of this project lies in guaranteeing absolute data sovereignty, lowering material and physical asset maintenance costs by safeguarding the grid, fortifying cybersecurity against external threats, and eliminating subscription fees for third-party cloud platforms.

Keywords: Digital Twin, Smart Grid, Edge AI, Multi-Agent Systems, LLM, and BESS.

Índice

RESUMEN 3

ABSTRACT 3

Capítulo 1. INTRODUCCIÓN 10

1.1 MOTIVACIÓN Y JUSTIFICACIÓN DEL PROYECTO 10

1.2 OBJETIVOS GENERALES Y ESPECÍFICOS 12

1.2.1 OBJETIVO GENERAL 12

1.2.2 OBJETIVOS ESPECÍFICOS 12

1.3 ALCANCE Y LÍMITES DEL ESTUDIO 13

Capítulo 2. ESTADO DEL ARTE 15

2.1 SISTEMAS ACTUALES 15

2.2 EVOLUCIÓN DE LAS SMART GRIDS Y MICROREDES 17

2.3 CONCEPTO DE GEMELO DIGITAL EN EL SECTOR ELÉCTRICO 19

2.4 SISTEMAS MULTI-AGENTE APLICADO AL CONTROL DE POTENCIA 21

2.5 MODELOS DE LENGUAJE DE GRAN TAMAÑO Y EDGE IA EN INFRAESTRUCTURAS CRÍTICAS 23

2.5.1 PARADIGMA DE LA COMPUTACIÓN EN EL BORDE 23

2.5.2 LOS LLM COMO MOTORES DE RAZONAMIENTO Y ORQUESTACIÓN COGNITIVA 24

2.5.3 APLICACIÓN EN INFRAESTRUCTURAS CRÍTICAS Y ASINCRONÍA 25

Capítulo 3. MEMORIA DESCRIPTIVA Y TÉCNICA 26

3.1 DESCRIPCIÓN DEL SISTEMA: ÉSTANDAR IEEE 23 NODOS 26

3.1.1 CARACTERÍSTICAS TÉCNICAS 26

3.1.2 IDONEIDAD PARA LA INTEGRACIÓN DE SMART GRIDS 27

3.2 VIRTUALIZACIÓN 29

3.2.1 EL HIPERVISOR: VMWARE WORKSTATION 29

3.2.2 CONFIGURACIÓN DEL ENTORNO Y AISLAMIENTO DEL SISTEMA 30

3.3 CONTENERIZACIÓN: DOCKER 32

3.3.1 ARQUITECTURA Y FUNCIONAMIENTO DE DOCKER 32

3.3.2 JUSTIFICACIÓN EN LA ARQUITECTURA DEL GEMELO DIGITAL 34

3.4 ARQUITECTURA DEL GEMELO DIGITAL: INTEGRACIÓN EN OPENDSS Y PYTHON 35
3.4.1 EL MOTOR DE LA SIMULACIÓN OPENDSS 35
3.4.2 AUTOMATIZACIÓN Y CONTROL MEDIANTE PYTHON 35
3.5 ORQUESTACIÓN COGNITIVA: DISEÑO DEL FLUJO DE TRABAJO EN N8N 36
3.5.1 PARADIGMA DE LA AUTOMATIZACIÓN BASADA EN NODOS 36
3.5.2 ESTRUCTURA DEL FLUJO DE TRABAJO 37
3.5.3 JUSTIFICACIÓN DE LA ORQUESTACIÓN ASÍNCRONA 37
3.6 IMPLEMENTACIÓN DEL AGENTE DE INTELIGENCIA ARTIFICIAL 38
3.6.1 OLLAMA COMO MOTOR DE INFERENCIA LOCAL 38
3.6.2 LLAMA 3.2 Y EL PARADIGMA DE EDGE IA 38
3.6.3 DISEÑO DEL RAZONAMIENTO Y PROMPT ENGINEERING 39
3.7 ESTRATEGIAS DE CONTROL Y GESTIÓN DE ALMACENAMIENTO DE ENERGÍA: SISTEMAS BESS Y BMS 40
3.7.1 ALGORITMOS DE GESTIÓN DE CARGA Y DESCARGA 40
3.7.2 MITIGACIÓN DE PICOS DE DEMANDA 40
Capítulo 4. ANÁLISIS DE RESULTADOS Y DISCUSIÓN 41
4.1 ESCENARIO 1: OPERACIÓN EN CONDICIONES NOMINALES 41
4.2 ESCENARIO 2: GESTIÓN EN CONDIONES DE ESTRÉS EXTREMO 42
4.3 ESCENARIO 3: APAGÓN 28 DE ABRIL DE 2025 44
4.4 ANÁLISIS COMPARATIVO DE EFICIENCIA Y ESFUERZO COMPUTACIONAL 46
Capítulo 5. CONCLUSIONES 48
5.1 CUMPLIMIENTO DE OBJETIVOS 48
5.2 FUTURAS LÍNEAS DE INVESTIGACIÓN Y ESCALABILIDAD DEL MODELO 50
5.2.1 ESCALABILIDAD A REDES DE GRAN ESCALA Y SISTEMAS MULTIAGENTE 50
5.2.2 FINE-TUNING DE LLMS PARA INGENIERÍA DE POTENCIA 50
5.2.3 CIBERSEGURIDAD COGNITIVA Y RESILIENCIA 50
Capítulo 6. ESTUDIOS DE ENTIDAD PROPIA 51
6.1 ESTUDIOS DE IMPACTO AMBIENTAL 51
6.1.1 INTEGRACIÓN DE ENERGÍAS RENOVABLES Y MITIGACIÓN DEL VÉRTIGO ENERGÉTICO 51
6.1.2 REDUCCIÓN DE INEFECIENCIAS MEDIANTE EL CONTROL DE PÉRDIDAS 52
6.2 ESTUDIO DE SEGURIDAD Y SALUD 53
6.2.1 REDUCCIÓN DE LA EXPOSICIÓN A RIESGO ELÉCTRICO EN OPERACIONES DE CAMPO 53
6.2.2 PREVENCIÓN DE FALLOS CATASTRÓFICOS Y PROTECCIÓN DE LA SALUD PÚBLICA 53

6.3 IMPACTO SOBRE LOS ODS 54
6.3.1 ODS 7: ENERGÍA ASEQUIBLE Y NO CONTAMINANTE 54
6.3.2 ODS 9: INDUSTRIA, INNOVACIÓN E INFRAESTRUCTURA 55
6.3.3 ODS 11: CIUDADES Y COMUNIDADES SOSTENIBLES 55
6.3.4 ODS 13: ACCIÓN POR EL CLIMA 56
6.4 IMPACTO DE LAS ASIGNATURAS ESTUDIADAS SOBRE ESTE PROYECTO 57
6.4.1 INGENIERÍA ELÉCTRICA Y SISTEMAS DE POTENCIA 57
6.4.2 AUTOMATIZACIÓN Y CONTROL 57
6.4.3 GESTIÓN DE PROYECTOS Y COMPETENCIAS TRANSVERSALES 58
6.5 IMPACTO ECONÓMICO 59
6.5.1 OPTIMIZACIÓN DEL OPEX Y SOBERANÍA TECNOLÓGICA 59
6.5.2 RENTABILIDAD DEL CAPITAL Y EXTENSIÓN DEL CICLO DE VIDA DE ACTIVOS 60
6.5.3 MITIGACIÓN DE RIESGOS Y AHORRO DE MANTENIMIENTO CORRECTIVO 60
6.5.4 NUEVAS VÍAS DE INGRESOS Y SERVICIOS DE FLEXIBILIDAD 61
Capítulo 7. PLIEGO DE CONDICIONES 62
7.1 ESPECIFICACIONES TÉCNICAS DEL SOFTWARE Y HARDWARE 62
7.1.1 ESPECIFICACIONES DE HARDWARE 62
7.1.2 ESPECIFICACIONES DE SOFTWARE Y ENTORNOS DE EJECUCIÓN 63
7.1.3 PROTOCOLOS DE COMUNICACIÓN E INTERCONECTIVIDAD 64
7.2 NORMATIVAS Y ESTÁNDARES APLICABLES 65
7.2.1 ESTÁNDARES DE INTERCONEXIÓN Y OPERACIÓN DE MICROREDES 65
7.2.2 ESTÁNDARES DE GEMELOS DGITALES Y MODELADO DE DATOS 66
7.2.3 ESTÁNDARES DE CIBERSEGURIDAD Y GOBERNANZA DE LA IA 66
7.3 PROTOCOLO DE MANTENIMIENTO DEL GEMELO DIGITAL 67
7.3.1 MANTENIMIENTO PREVENTIVO DEL MODELO ELÉCTRICO Y SIMULACIÓN 67
7.3.2 GESTIÓN Y ACTUALIZACIÓN DEL AGENTE DE IA 67
7.3.3 SUPERVISIÓN DE LA INFRAESTRUCTURA DE ORQUESTACIÓN N8N 68
Capítulo 8. PRESUPUESTO 69
8.1 COMPUTO DE HORAS DE INGENIERÍA 69
8.1.1 PERFILES PROFESIONALES 69
8.1.2 DESGLOSE DE HORAS POR FASE DE PROYECTO 70
8.2 COSTES DE INFRAESTRUCTURA Y LICENCIAS 71
8.2.1 INFRAESTRUCTURA DE HARDWARE Y SIMULACIÓN 71
8.2.2 LICENCIAS DE SOFTWARE Y ACTIVOS LÓGICOS 72

8.3 ANÁLISIS DE RENTABILIDAD Y RETORNO DE INVERSIÓN 73
8.3.1 COMPARATIVA DE MODELOS DE COSTE 73
8.3.2 ANÁLISIS DE VIABILIDAD: *PAYBACK* Y AHORRO OPERATIVO 73
8.3.3 RETORNO DE INVERSIÓN 74
8.3.4 PRESUPUESTO GENERAL 76
Capítulo 9. ANEXOS 77
9.1 CÓDIGOS DE OPENDSS 77
9.1.1 BusCoords.dss 77
9.1.2 BusVoltageBases.dss 78
9.1.3 EnergyMeter.dss 79
9.1.4 GrowthShape 80
9.1.5 LineCode.dss 81
9.1.6 Renovables.dss 82
9.1.7 Monitor.dss 88
9.1.8 Spectrum.dss 89
9.1.9 TCC_Curve.dss 90
9.1.10 Transformer.dss 91
9.1.11 Vsource.dss 92
9.2 PYTHON 93
9.2.1 Sim_apagon_2025.py 93
9.2.2 Sim_ola_frio.py 101
9.2.3 Sim_soleado.py 105
9.2.4 Sim_tormenta.py 112
9.2.5 Analisis_datos.py 116
9.3 PROMPT AGENTE IA 118
BIBLIOGRAFÍA 119

Índice de Figuras

Ilustración 1: Ejemplo de Sistema Multi-Agente.....22

Ilustración 2: Modelo de 23 nodos del IEEE.....27

Ilustración 3: Simulación del modelo de 23 nodos en perspectiva. 28

Ilustración 4: Configuración básica de la máquina virtual. 31

Ilustración 5: Activación del contenedor Docker. 32

Ilustración 6: Configuración del servidor local contenerizado.....33

Ilustración 7: Flujo de trabajo en n8n. 36

Ilustración 8: Comportamiento de la microrred y respuesta de IA bajo condiciones nominales.....41

Ilustración 9: Comportamiento de la microrred y respuesta de IA bajo condiciones de estrés.....42

Ilustración 10: Comportamiento de la microrred y respuesta de IA ante una caída masiva del sistema.
.....44

Ilustración 11: Análisis comparativo final de intervenciones del agente supervisor.....46

Ilustración 12: Gráfico de periodo de recuperación de la inversión.....74

Índice de Tablas

Tabla 1: Inversión en horas de ingeniería. 70

Tabla 2: Inversión en hardware 71

Tabla 3: Inversión en software. 72

Tabla 4: Comparativo de modelos de coste. 73

Tabla 5: Retorno de la inversión inicial. 75

Capítulo 1. INTRODUCCIÓN

1.1 MOTIVACIÓN Y JUSTIFICACIÓN DEL PROYECTO

La transición hacia sistemas energéticos sostenibles ha modificado la estructura de las redes de distribución en las últimas décadas. La integración masiva de fuentes de generación renovable distribuida y la aparición de nuevas cargas, como el vehículo eléctrico, generan flujos de potencia bidireccionales y altamente variables. Esta complejidad técnica es el motivo por el que las grandes empresas de distribución y las infraestructuras de control están en constante modernización para garantizar la estabilidad de la red y el suministro eléctrico.

La identificación de esta problemática no surge únicamente de investigación propia, sino que tiene su origen directo en la experiencia adquirida mediante el trabajo de campo. Durante mi actividad profesional en la ejecución de proyectos para el sector eléctrico y de la construcción, he podido identificar las necesidades de los clientes y las deficiencias prácticas de los sistemas de control convencionales. Debido a la variabilidad y a las cargas presentes en la red eléctrica actual, la necesidad de una supervisión avanzada es un requisito que justifica la implementación de redes inteligentes o *Smart Grids*. Según explica Farhangi (2010), la *Smart Grid* es una modernización del sistema de suministro eléctrico que supervisa, protege y optimiza el funcionamiento de sus elementos interconectados. Bajo esta premisa, la optimización del sistema requiere herramientas capaces de procesar volúmenes masivos de datos con baja latencia. Esta herramienta son los «Gemelos Digitales». Su concepto se fundamenta en la creación de un entorno virtual que replica de manera precisa el comportamiento de un sistema físico, según define Grieves (2014). En el sector eléctrico, esta representación permite predecir fallos y optimizar la distribución de energía sin comprometer la integridad de los activos reales ni la de los clientes.

Es precisamente en el entorno industrial donde se hace evidente que las soluciones actuales basadas exclusivamente en la computación en la nube presentan ciertas limitaciones que podrían ser críticas en caso de fallo. El constante intercambio de datos sensibles hacia servidores remotos genera retardos que impiden una respuesta inmediata ante fluctuaciones bruscas de la red, lo que puede causar daños materiales y pérdidas económicas inaceptables para la viabilidad de un proyecto. Además, el procesamiento externo incrementa los riesgos de ciberseguridad y la posibilidad de perder la gobernanza sobre los datos de la infraestructura. Para dar respuesta a estas carencias en el sector, la arquitectura de este proyecto se fundamenta en el uso de *Edge IA*, desplazando el procesamiento al borde de la red para asegurar que la operación sea asíncrona y autónoma.

La innovación de este trabajo consiste en la integración de modelos de lenguaje de gran tamaño o *LLM* como el núcleo supervisor en la toma de decisiones. El uso de Llama 3.2 permite desplegar una infraestructura cognitiva que interpreta los datos del motor de simulación *OpenDSS* y actúa sobre los sistemas de almacenamiento de energía en baterías o *BESS*. Esta propuesta valida una solución técnica que combina la robustez del cálculo eléctrico con la flexibilidad de la inteligencia artificial local, optimizando la adaptabilidad de la microrred frente a escenarios de alta volatilidad.

1.2 OBJETIVOS GENERALES Y ESPECÍFICOS

1.2.1 OBJETIVO GENERAL

El propósito de esta investigación radica en la convergencia de la simulación eléctrica avanzada con la inteligencia artificial generativa. Se busca desplazar la toma de decisiones lógica desde entornos centralizados hacia el borde de la red, para garantizar una gestión autónoma y segura de los activos energéticos. Para ello, se desarrolla y valida un sistema de gemelo digital para la gestión inteligente de una microrred eléctrica de 23 nodos del IEEE, integrando una arquitectura de orquestación cognitiva basada en *Edge IA* para la optimización de los flujos de potencia.

1.2.2 OBJETIVOS ESPECÍFICOS

- Modelar la infraestructura eléctrica y el comportamiento dinámico del alimentador de 23 nodos del IEEE mediante la integración de OpenDSS y Python.
- Implementar un motor de inferencia local mediante el uso del modelo de lenguaje Llama 3.2 para el análisis y la supervisión de los estados de operación de la red.
- Diseñar una plataforma de comunicación asíncrona sobre un entorno Linux mediante la plataforma de automatización n8n que facilite la interacción entre el entorno de simulación y el agente supervisor de IA.
- Desarrollar estrategias de control para sistemas de almacenamiento de energía en baterías y enfocadas en la mitigación de picos de demanda y la absorción de excedentes renovables.
- Evaluar la respuesta del sistema ante escenarios impredecibles y de alta volatilidad para determinar la eficacia del razonamiento lógico de la IA en situaciones críticas.

1.3 ALCANCE Y LÍMITES DEL ESTUDIO

El alcance de este trabajo se centra en el diseño, configuración y validación de una arquitectura de software para la gestión de redes de baja tensión. Se incluye el desarrollo del entorno de simulación basado en el estándar de 23 nodos del IEEE, la configuración de la infraestructura de microservicios desplegada en contenedores *Docker* y la implementación de la lógica de orquestación en la plataforma n8n. El proyecto contempla la ejecución local de modelos de lenguaje de gran tamaño para la toma de decisiones operativas.

El estudio presenta una serie de límites técnicos y metodológicos necesarios para asegurar la viabilidad de la investigación:

- La investigación se limita estrictamente al ámbito de la simulación computacional. No se contempla la integración con hardware físico, equipos de medida reales ni la conexión a redes de distribución activas. No obstante, la arquitectura basada en contenedores y protocolos de comunicación estándar está diseñada con gran modularidad, por lo que una transición directa hacia sistemas SCADA y pasarelas IoT industriales sería un proceso sencillo, requiriendo únicamente la adaptación de las interfaces de adquisición de datos.

- El análisis se restringe a la estructura definida por el modelo IEEE de 23 nodos. No se evalúan otras configuraciones de red ni expansiones de la topología existente. Sin embargo, la lógica de control implementada en *Python* es independiente a la topología del modelo. Escalar la simulación a ciudades enteras y miles de nodos sería un proceso directo, sujeto solo a la disponibilidad de mayor capacidad de procesamiento y computación.

- El desarrollo metodológico de IA se enfoca en la ingeniería de instrucciones o *prompt engineering*. Se excluye el entrenamiento de modelos desde cero o la modificación de la arquitectura interna del modelo LLM utilizado. Se reconoce que, con una mayor disponibilidad de tiempo y recursos, el ajuste fino o *Fine-Tuning* del modelo con bases de datos históricas de operación eléctrica permitiría reducir el tamaño de las instrucciones, disminuyendo la latencia de inferencia y aumentando la precisión del sistema.

- El proyecto se centra en el estudio de los flujos de potencia y el balance energético en régimen estacionario. No se analizan fenómenos de calidad de onda, armónicos o transitorios de alta frecuencia, aunque su inclusión podría abordarse sin alterar la arquitectura base mediante la integración de módulos específicos de simulación dinámica en OpenDSS.

Respecto al grado de fidelidad del sistema virtual desarrollado, Grieves (2014) establece el límite de la descripción informativa, donde explica que el gemelo digital es un conjunto de constructos de información virtual que describe por completo un producto físico fabricado, ya sea potencial o real, desde el nivel micro atómico hasta el nivel macro geométrico Para este estudio, el límite se establece en la descripción macroscópica de las variables de potencia y tensión del sistema, omitiendo la replicación física de los componentes a niveles microscópicos o estructurales.

El desarrollo y ejecución del gemelo digital y el bucle de control asíncrono propuesto están delimitados a las capacidades físicas del entorno de ejecución local, *Edge Computing*, cuyas especificaciones técnicas imponen restricciones sobre la velocidad de procesamiento y la latencia de respuesta. El equipo anfitrión cuenta con un procesador AMD Ryzen 7 3700U (2.30 GHz, 4 núcleos y 8 hilos) y una configuración de memoria RAM de 16 GB, de los cuales únicamente 13,9 GB están disponibles para el sistema operativo debido a la reserva obligatoria de hardware de 2 GB asignada a la arquitectura gráfica integrada AMD Radeon RX Vega 10.

Esta restricción en la memoria volátil utilizable representa el principal cuello de botella tecnológico de este proyecto. Al ejecutar simultáneamente el motor de simulación eléctrica *OpenDSS*, el entorno de orquestación n8n bajo contenedores *Docker* y el proceso de inferencia local del modelo de lenguaje Llama 3.2 3B, la carga de memoria se aproxima de manera crítica al umbral de saturación. Como consecuencia, la velocidad de generación de tokens del modelo de IA se ve limitada por el ancho de banda de la memoria de acceso aleatorio y la tasa de reloj de la unidad central de procesamiento, lo que eleva la latencia de cada ciclo de decisión a un rango de 5 a 15 segundos por paso de simulación.

Es importante destacar que esta limitación es derivada del equipo de ensayo utilizado. En un escenario de implantación real, este cuello de botella se supera mediante la adquisición de servidores industriales en el borde con aceleración neuronal y mayor capacidad de memoria RAM. Este tipo de hardware presenta un coste de capital muy competitivo y fácilmente asumible para las empresas distribuidoras.

Por lo tanto, aunque el algoritmo demuestra una elevada resiliencia y viabilidad para la gestión optimizada y la programación operativa de flujos energéticos en tiempo *cuasi*-real, las limitaciones de hardware actuales impiden su aplicación directa en sistemas de protección críticos que requieran tiempos de respuesta en el orden de los milisegundos o control en tiempo real estricto. A pesar de ello, este trabajo demuestra que, con el tiempo necesario para el entrenamiento de los modelos y una inversión razonable en recursos hardware, la descentralización de redes eléctricas mediante inteligencia artificial local es una solución técnica viable y escalable para la industria.

Capítulo 2. ESTADO DEL ARTE

2.1 SISTEMAS ACTUALES

La gestión actual de las redes de distribución eléctrica se realiza mediante sistemas de control supervisado y adquisición de datos SCADA o *Supervisory Control and Data Acquisition*. Estos sistemas han permitido durante décadas la monitorización y el control remoto de los activos de la red desde centros de operaciones centralizados. Su diseño se basa en una arquitectura rígida donde los datos fluyen desde las subestaciones hacia un centro terminal maestro para el análisis por parte de operadores.

El diseño de los sistemas actuales prioriza la robustez y la seguridad de las comunicaciones, por lo que su función principal radica en garantizar la continuidad del suministro mediante el aislamiento de fallos y la reconfiguración de la red. Boyer (2009) define la esencia de la tecnología SCADA como un sistema que no constituye un sistema de control integral, sino más bien un enfoque especializado en el ámbito de la supervisión y el control. Bajo este paradigma, el sistema se limita a ejecutar órdenes directas y a adquirir datos telemétricos, careciendo de una capacidad de análisis autónomo profundo sobre los datos recibidos. Técnicamente, esto limita los sistemas de control tradicionales en un enfoque únicamente correctivo, las protecciones y maniobras operan únicamente cuando se ha superado un umbral crítico de fallo y un enfoque preventivo gracias a los algoritmos de seguridad, pero sigue careciendo de la inteligencia predictiva lógica y autónoma.

A pesar de su eficacia histórica, los sistemas SCADA convencionales presentan limitaciones frente a la integración masiva de recursos energéticos distribuidos. Las infraestructuras actuales fueron diseñadas para flujos de potencia unidireccionales y perfiles de demanda predecibles. Sin embargo, la intermitencia de la generación fotovoltaica y la naturaleza estocástica de la carga de los recursos energéticos distribuidos introducen un factor desbordante sobre la capacidad de procesamiento de los controladores lógicos programables tradicionales. Esta incapacidad para gestionar dinámicamente la bidireccionalidad de los activos de la red impide aislar eficientemente los problemas por barrios o zonas locales. Como resultado, una perturbación local tiene una gran probabilidad de propagarse en cascada, desestabilizando “aguas arriba” la red y afectando gravemente a la resiliencia operativa del sistema.

En la actualidad, se observa una brecha tecnológica entre la generación de datos y la toma de decisiones. Aunque los contadores inteligentes y los sensores de red generan volúmenes masivos de información, la mayor parte de estos flujos de datos se utilizan para facturación o análisis posterior no para el control en tiempo real. Esta dependencia estructural y limitación técnica reduce significativamente las oportunidades de modelo de negocio de las compañías distribuidoras. Por un lado, las empresas asumen altísimos costes operativos asociados al alquiler de líneas de telecomunicaciones y al almacenamiento en servidores *Cloud* para albergar terabytes de datos que no se están explotando de forma proactiva. Por otro lado, la lentitud de los sistemas centralizados en la detección y aislamiento de fallos empeora los índices regulatorios de calidad del suministro, lo que se traduce directamente en fuertes sanciones económicas por parte de los organismos reguladores.

2.2 EVOLUCIÓN DE LAS SMART GRIDS Y MICROREDES

La arquitectura de la red eléctrica ha experimentado una transformación radical desde sus orígenes. Durante el siglo XX, el sistema se basaba en un modelo de generación centralizada a gran escala, alejado de los centros de consumo. Este modelo presentaba ineficiencias estructurales debido a las pérdidas en el transporte y a la incapacidad de la red para gestionar flujos bidireccionales. La red inteligente o *Smart Grid* surge como la respuesta tecnológica para gestionar de forma eficiente el comportamiento de todos los usuarios conectados a ella.

La transición hacia la red inteligente no se limita a la digitalización de los activos físicos. Esta transformación representa un cambio en la filosofía operativa, en la cual el sistema deja de ser un receptor pasivo para convertirse en una infraestructura activa y autogestionada. Farhangi (2010) explica esta reestructuración precisando que la red inteligente es una modernización del sistema de suministro eléctrico configurada para supervisar, proteger y optimizar la operación de sus elementos interconectados: desde los generadores centrales y distribuidos, pasando por la red de alta tensión y el sistema de distribución, hasta los usuarios industriales, los sistemas de automatización de edificios, las instalaciones de almacenamiento de energía y los consumidores finales con sus respectivos termostatos, vehículos eléctricos y electrodomésticos.

Esta visión implica que cada componente, desde un transformador hasta un electrodoméstico, se convierte en un nodo con capacidades de comunicación bidireccional. Los pilares de esta transición tecnológica son la Infraestructura de Medición Avanzada, la automatización de la distribución y la integración de Recursos Energéticos Distribuidos. Estos elementos permiten que la red detecte anomalías y se reconfigure automáticamente, minimizando el impacto de las interrupciones del servicio.

La materialización teórica y técnica de estos conceptos ya es una realidad en el sector eléctrico en España. A nivel macroestructural, un ejemplo claro es el proyecto *STAR* o Sistema de Telegestión y Automatización de la Red, implementado por la distribuidora i-DE del Grupo Iberdrola. Mediante el despliegue de millones de contadores inteligentes y la automatización de miles de centros de transformación, los beneficios reales obtenidos han incluido una reducción drástica del Tiempo de Interrupción Equivalente a la Potencia Instalada o TIEPI mejorando la calidad del suministro, la detección remota e instantánea de fraudes o averías, y la optimización logística al reducir los desplazamientos de las brigadas de operarios.

En este proceso evolutivo, el concepto de microrred ha cobrado una relevancia estratégica. Una microrred se define como un conjunto interconectado de cargas y recursos de generación que opera como una entidad única respecto a la red principal. Su característica más distintiva es la capacidad de operar tanto conectada a la red de distribución como de forma aislada o “modo isla”. Esta doble función es vital para garantizar el suministro en infraestructuras críticas durante desastres naturales fallos en la red general, como el apagón del 28 de abril de 2025.

La fundamentación técnica de la microrred como unidad de control fue establecida por Lasseter (2002), quien define este sistema como una agrupación de cargas y microfuentes que opera como un único sistema controlable, el cual suministra tanto energía eléctrica como térmica a su área local. Este concepto establece un nuevo paradigma para la operación de la generación distribuida en la red eléctrica.

La viabilidad y los beneficios de este modelo de "modo isla" se observan en infraestructuras modernas, como la microrred de la Universidad de California San Diego. Esta instalación integra generación solar, cogeneración térmica y almacenamiento en baterías BESS, cubriendo más del 85% de la demanda energética del campus. El beneficio real es doble: un ahorro anual de millones de dólares y la demostración de capacidad técnica para aislarse de la red estatal y mantener el suministro intacto a su hospital y centros de investigación durante los recurrentes apagones sufridos en California.

El desarrollo de las microrredes ha permitido el paso del consumidor tradicional al “prosumidor” (productor y consumidor). Este nuevo usuario no solo demanda energía de la red, sino que también la genera mediante sistemas fotovoltaicos y gestiona su propia demanda a través de sistemas de almacenamiento de energía en baterías. La tendencia actual se orienta hacia la integración de estas microrredes en un ecosistema de "Internet de la Energía", donde la gestión ya no es centralizada, sino que se distribuye entre agentes inteligentes capaces de negociar y balancear los flujos de potencia de manera local. Un caso pionero de esta tecnología es la *Brooklyn Microgrid* en Nueva York, impulsada por LO3 Energy. En este proyecto piloto, los vecinos utilizan tecnología *Blockchain* para vender directamente sus excedentes de energía solar a otros residentes de su mismo barrio, maximizando la eficiencia local y creando un modelo de negocio energético completamente descentralizado.

Finalmente, la consolidación de las microrredes inteligentes justifica la necesidad de desarrollar gemelos digitales, ya que la complejidad de gestionar múltiples fuentes variables en tiempo real hace inviable la dependencia de la intervención humana oportuna y sin errores. La evolución técnica exige, por tanto, infraestructuras de supervisión que combinen la precisión de los modelos eléctricos con la agilidad de los sistemas de decisión automatizados, permitiendo una transición energética segura y eficiente.

2.3 CONCEPTO DE GEMELO DIGITAL EN EL SECTOR ELÉCTRICO

El concepto de gemelo digital representa una evolución significativa respecto a los modelos de simulación tradicionales. Mientras que la simulación estática evalúa escenarios a partir de parámetros predefinidos, el gemelo digital se distingue por establecer un flujo de datos dinámico y bidireccional con su homólogo físico. En el sector eléctrico, esta tecnología permite replicar el comportamiento de activos complejos, como transformadores, subestaciones o microrredes completas, facilitando la supervisión y la optimización en tiempo real.

Esta capacidad de réplica exacta virtual debe tener la integración de tres elementos fundamentales: un activo físico en el mundo real, una representación virtual en un entorno especializado y un flujo de datos que vincule y sincronice ambas dimensiones de manera precisa y continua. En la gestión de redes eléctricas, el gemelo digital no solo almacena datos, sino que utiliza modelos matemáticos y eléctricos, como los proporcionados por *OpenDSS*, para predecir el comportamiento futuro de la red ante cambios en la demanda o la generación.

La diferencia entre un modelo convencional y un gemelo digital está presente en la latencia y la sincronización. Un gemelo digital efectivo debe reflejar el estado del sistema físico con una fidelidad suficiente para permitir la toma de decisiones operativa sobre los activos. En el ámbito de las *Smart Grids*, esto se traduce en la monitorización de flujos de potencia, niveles de tensión y estados de carga de los sistemas de almacenamiento de energía. La capacidad predictiva del sistema gemelo permite evaluar escenarios "qué pasaría si" sin poner en riesgo la estabilidad de la infraestructura real.

Por ello, la implementación de gemelos digitales en el sector eléctrico responde a la creciente complejidad de las redes distribuidas actuales. La presencia de estos nuevos componentes y cargas estocásticas generan una dinámica de red difícil de gestionar mediante métodos tradicionales. El gemelo digital, actúa como una capa de inteligencia intermedia, que procesa la telemetría de los sensores y propone acciones de control optimizadas y precisas para cado escenario por muy impredecible que sea. Esta arquitectura es la que permite que un agente de Inteligencia Artificial pueda interactuar con una simulación de alta precisión, validando órdenes de control antes de su ejecución en el sistema físico.

En la actualidad, el desarrollo de estos sistemas está respaldado por nuevos conceptos como la contenerización y la computación en el borde o *Edge Computing*. La posibilidad de desplegar el gemelo digital cerca de la infraestructura física reduce los tiempos de respuesta y mejora la ciberseguridad. Al procesar toda la información localmente, se evita la exposición de datos sensibles de la red a nubes públicas, manteniendo la soberanía tanto del sistema eléctrico como de la información. Este escenario sustenta el diseño del sistema planteado en este trabajo, donde el gemelo virtual opera como la plataforma de aprendizaje y validación del supervisor cognitivo.

2.4 SISTEMAS MULTI-AGENTE APLICADO AL CONTROL DE POTENCIA

La supervisión de las redes eléctricas modernas requiere de una actualización del control centralizado hacia estructuras distribuidas. Los Sistemas Multi-Agente surgen como una técnica de gestión robusta para abordar las microrredes complejas. Esta tecnología pionera se basa en el despliegue de múltiples entidades de software autónomas, denominadas agentes, que interactúan entre sí para alcanzar objetivos comunes o individuales en un entorno dinámico.

La definición fundamental de lo que constituye un agente es crucial para entender su aplicación industrial. Wooldridge y Jennings (1995) establecen las propiedades esenciales en su estudio sobre la teoría y práctica de los agentes como un sistema informático situado en un entorno determinado, capaz de actuar de manera autónoma en dicho entorno con el fin de cumplir sus objetivos de diseño. En el contexto de la ingeniería de potencia, cada componente de la red, como un generador fotovoltaico, una batería o una carga crítica, puede ser representado por un agente autónomo. Estos agentes poseen capacidades de reacción ante cambios en la red, proactividad para optimizar su propio estado y habilidad “social” para comunicarse con otros nodos del sistema y entre ellos.

La aplicación de los sistemas multi-agente en sistemas eléctricos resuelve las limitaciones de escalabilidad de los centros de control convencionales, al distribuir la toma de decisiones, se reduce la carga computacional y se reduce exponencialmente la posibilidad de fallo. McArthur et al. (2007) explica el propósito de estas arquitecturas en el sector eléctrico como el conjunto de agentes autónomos que interactúan entre sí para resolver un problema que excede las capacidades o el conocimiento individual de cada agente.

Esta capacidad de colaboración es de vital importancia en un entorno como las microrredes con alta penetración de renovables. Los agentes pueden negociar flujos de potencia en tiempo real, equilibrando la oferta y la demanda mediante protocolos de comunicación estandarizados. Por ejemplo, un agente supervisor encargado de un sistema de almacenamiento de energía en baterías puede ofrecer su energía almacenada a otro agente de carga cuando detecta que la frecuencia de la red disminuye, operando de forma coordinada sin intervención humana directa.

La integración de los Sistemas Multi-Agente junto a un gemelo digital permite crear una capa de abstracción sobre las características física del sistema. Mientras el gemelo digital procesa los flujos de potencia, los agentes lógicos evalúan las mejores estrategias de control en milisegundos. Esta arquitectura distribuida facilita la implementación de técnicas de optimización complejas y la gestión de la incertidumbre.

En la actualidad, la orientación dentro de la investigación de sistemas multi-agente para redes inteligentes se centra en la autonomía cognitiva. Los agentes ya no solo siguen reglas lógicas predefinidas, sino que además incorporan capacidades de aprendizaje y razonamiento. Esta evolución respalda la propuesta de este proyecto, donde n8n actúa como el bus de comunicaciones que permite a diferentes agentes, incluido un supervisor basado en un modelo de lenguaje de gran tamaño, colaborar en la gestión de la red de 23 nodos del IEEE.

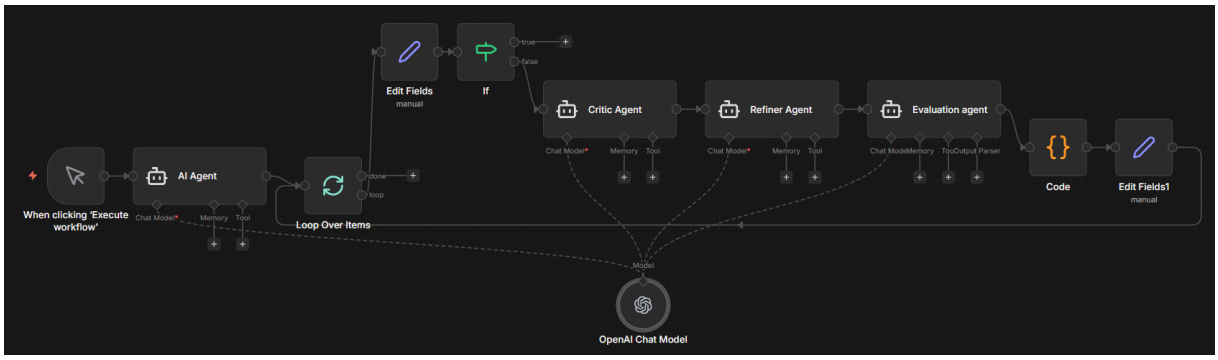


Ilustración 1: Ejemplo de Sistema Multi-Agente.

2.5 MODELOS DE LENGUAJE DE GRAN TAMAÑO Y EDGE IA EN INFRAESTRUCTURAS CRÍTICAS

2.5.1 PARADIGMA DE LA COMPUTACIÓN EN EL BORDE

La implementación de la Inteligencia Artificial en el sector eléctrico ha evolucionado desde el análisis predictivo básico hacia sistemas cognitivos de razonamiento avanzado. La aparición de los Modelos de Lenguaje de Gran Tamaño y su despliegue en el borde de la red permite procesar la información de forma local, garantizando la autonomía de las infraestructuras y su seguridad de operación.

El despliegue de algoritmos de IA en servidores centralizados tiene limitaciones destacables para el control de redes eléctricas. El envío por telemetría a la nube introduce una latencia de red que puede verse comprometida y, por tanto, la estabilidad del sistema ante eventos transitorios. La *Edge IA* surge como la solución técnica para ejecutar procesos de inferencia directamente en el mismo nodo donde se generan los datos.

La justificación técnica de este modelo se fundamenta en la necesidad de respuestas en tiempo real y en la seguridad de la información. De esta manera, Satyanarayanan (2017) define la relevancia de este paradigma en su estudio sobre la emergencia de la computación perimetral tal que el borde de Internet constituye una nueva frontera para numerosas aplicaciones, incluyendo aquellas que demandan un gran ancho de banda, o que son altamente sensibles a la latencia y a la privacidad. En una microrred, el procesamiento en el borde asegura que la toma de decisiones no dependa de la conectividad a Internet. La validez industrial de este paradigma ya está siendo probada con éxito en otras *utilities* de infraestructuras críticas. Por ejemplo, en la gestión del ciclo integral del agua, empresas como Global Omnium a través de su plataforma tecnológica *GoAigua* han implementado arquitecturas de computación en el borde para la monitorización de presiones y caudales. Los resultados muestran que procesar los datos de los sensores acústicos a nivel local permite identificar fugas de agua en tiempo real sin saturar las redes de telecomunicaciones corporativas, reduciendo el agua no registrada y minimizando los costes operativos del negocio.

2.5.2 LOS LLM COMO MOTORES DE RAZONAMIENTO Y ORQUESTACIÓN COGNITIVA

A diferencia de los modelos de aprendizaje automático tradicionales, que se especializan en tareas específicas, los LLM ofrecen una gran capacidad de razonamiento general. Modelos de parámetros reducidos y optimizados para hardware local, como Llama 3.2, permiten interpretar estados de red complejos y traducirlos en acciones de control lógico con una latencia reducida. La innovación de estos modelos reside en su capacidad para actuar como "orquestadores cognitivos", pueden procesar múltiples variables de entrada como predicciones meteorológicas, estado de carga de baterías y precios de la energía para determinar la estrategia óptima de operación.

Esta transición hacia agentes capaces de razonar sobre el entorno es descrita por Touvron et al. (2023) en el desarrollo de arquitecturas abiertas, refiriéndose a los modelos Llama y a como evidencian que es posible desarrollar y utilizar de manera efectiva estos modelos fundacionales abiertos, eficientes y de alta calidad en entornos locales con recursos de hardware limitados.

Esta eficiencia en el uso de recursos del entorno en el borde permite que un agente de IA supervise el comportamiento de una microrred sin necesidad de super infraestructuras de computación. El modelo actúa como una capa de inteligencia que dota de sentido a los datos procedentes del gemelo digital. La adopción de esta tecnología cognitiva está ganando influencia en sectores industriales de alta exigencia. En el sector del transporte, corporaciones como Siemens Mobility están integrando algoritmos de procesamiento de lenguaje y modelos predictivos dentro de su ecosistema de mantenimiento *Railigent*. Al ejecutar modelos de IA, el sistema es capaz de interpretar de forma autónoma los registros de texto de diagnóstico y la telemetría de los trenes. Como resultado, han logrado anticipar fallos mecánicos, incrementando la disponibilidad de las flotas en casi un 100% y reduciendo el mantenimiento correctivo.

2.5.3 APLICACIÓN EN INFRAESTRUCTURAS CRÍTICAS Y ASINCRONÍA

El uso de LLM en infraestructuras críticas requiere un diseño que tolere la latencia de inferencia. A diferencia de un controlador lógico, un modelo de lenguaje estándar en la nube puede tardar segundos en procesar una respuesta. Por ello, la arquitectura debe ser asíncrona, permitiendo que el control local de la red mantenga la seguridad física mientras el agente de IA evalúa la estrategia a largo plazo.

La aplicación de estos modelos en el sector eléctrico busca maximizar la resiliencia, en escenarios de alta volatilidad o fallo de comunicaciones, la capacidad de un LLM local para razonar de forma autónoma permite que la microrred mantenga una operación estable. En el ámbito comercial y de distribución masiva, operadores energéticos como Octopus Energy ya han incorporado modelos de lenguaje de gran tamaño en el núcleo de su plataforma de operaciones llamada *Kraken*. Mediante esta integración, el sistema gestiona los activos energéticos distribuidos de sus clientes y optimiza los patrones de consumo de manera inteligente y escalable. Estos casos de uso globales evidencian que esta integración de *Edge IA* y LLMs representa la actualidad de la investigación sobre tecnologías pioneras en redes inteligentes, permitiendo que el gemelo digital pueda evolucionar hacia un sistema de gestión energética verdaderamente proactivo.

Capítulo 3. MEMORIA DESCRIPTIVA Y TÉCNICA

3.1 DESCRIPCIÓN DEL SISTEMA: ÉSTANDAR IEEE 23 NODOS

La operatividad de los algoritmos de control en redes de distribución requiere una validación minuciosa debido a la sensibilidad de la infraestructura que supervisan, y para ello, el uso de modelos de red estandarizados que representen las complejidades reales de los sistemas eléctricos de potencia es el mejor método para este cometido. El comité de análisis de sistemas de la *IEEE Power & Energy Society* desarrolló una serie de alimentadores de prueba denominados Redes de Prueba de Distribución del IEEE. El sistema de 23 nodos elegido para este proyecto constituye una topología perfecta para el análisis de flujos de potencia y la integración de recursos energéticos distribuidos en entornos de baja y media tensión.

3.1.1 CARACTERÍSTICAS TÉCNICAS

El modelo del IEEE de 23 nodos se caracteriza por una configuración radial y un diseño orientado a la distribución urbana. Este sistema está pensado para reflejar las condiciones de desequilibrio en las redes de distribución reales, donde las cargas no se reparten de manera uniforme entre fases. La red incluye una combinación de líneas aéreas y cables subterráneos con diferentes configuraciones de impedancia, lo que permite obtener datos cruciales con precisión de las caídas de tensión y las pérdidas en condiciones de cargas variables.

La precisión en el modelado de estos alimentadores es fundamental para garantizar datos certeros, de esta forma Kersting (2001) subraya la importancia de la exactitud en los parámetros de línea en su estudio fundamental sobre el modelado de sistemas de distribución, donde establece que el análisis de un alimentador de distribución requiere que los tramos de línea se modelen con la mayor precisión posible. Esto resulta especialmente importante en el cálculo de caídas de tensión y las pérdidas de potencia. En el presente proyecto, dicha precisión se asegura mediante la definición de matrices de impedancia y admitancia dentro del entorno de simulación OpenDSS, permitiendo que el gemelo digital replique el comportamiento físico con fidelidad. Estas particularidades se detallan de manera más extensa en el Capítulo 9 (Anexos).

3.1.2 IDONEIDAD PARA LA INTEGRACIÓN DE SMART GRIDS

La selección de este estándar se fundamenta en su versatilidad para alojar múltiples puntos de inyección de potencia provenientes de fuentes renovables. La topología de 23 nodos permite establecer puntos de control estratégicos donde la inteligencia artificial puede intervenir sobre los perfiles de carga. Al ser un sistema ampliamente reconocido y documentado por la comunidad científica, los resultados obtenidos en esta arquitectura poseen un alto grado de compatibilidad y rigor técnico.

El modelo de prueba permite simular escenarios críticos, como el estrés térmico en sus conductores y la saturación del transformador en momentos de alta demanda. En este esquema en concreto, el control local supervisa la estabilidad de las variables eléctricas, mientras que el agente de IA orquesta las estrategias de contingencia de la microrred. Esta dualidad operativa es esencial para estudiar la interacción entre la respuesta física del sistema y la lógica asíncrona de la orquestación cognitiva, para una futura implementación en el borde.

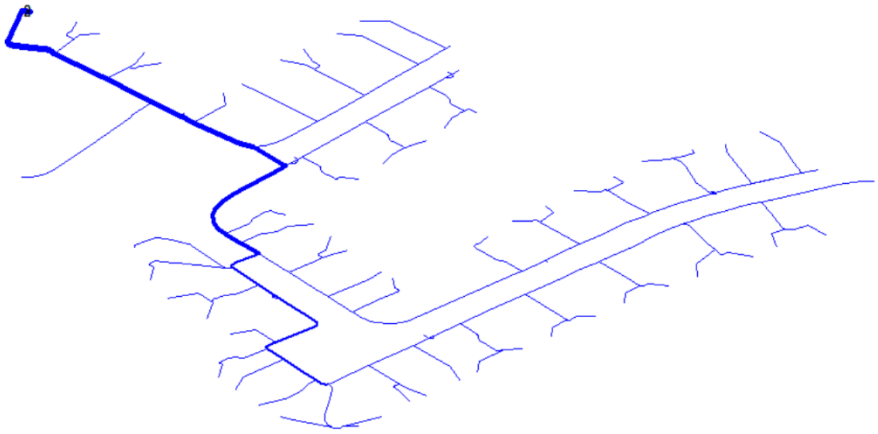


Ilustración 2: Modelo de 23 nodos del IEEE.



Ilustración 3: Simulación del modelo de 23 nodos en perspectiva.

3.2 VIRTUALIZACIÓN

La virtualización constituye una capa tecnológica fundamental para el desarrollo del gemelo digital en este proyecto. Este recurso informático permite la abstracción de los recursos de hardware, facilitando la ejecución de distintos sistemas operativos sobre una única plataforma física. En el contexto de este trabajo, la virtualización se utiliza para aislar el entorno de gestión y orquestación en *Linux* del entorno de simulación eléctrica y desarrollo en *Windows*.

3.2.1 EL HIPERVISOR: VMWARE WORKSTATION

Para la implementación del sistema, se ha seleccionado un hipervisor de tipo 2, concretamente *VMware Workstation*. Este software permite la creación de un entorno de ejecución aislado y seguro para el sistema operativo virtual. La elección de esta herramienta sobre otras se justifica por su capacidad para gestionar de forma eficiente la aceleración de hardware y la configuración de redes virtuales complejas, elementos para la comunicación entre el simulador y el agente de inteligencia artificial.

La virtualización ofrece una solución a la incompatibilidad entre las herramientas de gestión industrial y los motores de cálculo eléctrico, Smith y Nair (2005) definen la esencia de esta tecnología en su estudio sobre arquitecturas virtuales como una réplica eficiente y aislada de una máquina real. La virtualización proporciona un medio para desacoplar el entorno de *software* del *hardware* subyacente, lo que permite una mejor gestión de los recursos y la portabilidad del sistema

3.2.2 CONFIGURACIÓN DEL ENTORNO Y AISLAMIENTO DEL SISTEMA

El sistema operativo seleccionado como máquina virtual para este sistema es Ubuntu 24.04 LTS, esta distribución en *Linux* proporciona un entorno estable, optimizado y sobre todo seguro para la ejecución de herramientas de automatización y contenedores. El aislamiento que proporciona la máquina virtual garantiza que los procesos de orquestación de n8n no interfieran con la estabilidad del sistema anfitrión, permitiendo además una fácil replicabilidad del experimento para diferentes infraestructuras físicas.

La configuración de red de la máquina virtual se ha diseñado para emular un entorno de computación de borde o *Edge Computing*. Para ello, se ha modelado una red privada virtual que permite la comunicación bidireccional entre la simulación en *OpenDSS* y el cerebro lógico contenerizado en el servidor local en *Linux*, entremedias, este canal de flujo de datos es también compartido por el motor *Python* y la arquitectura agentica de n8n. Este esquema de virtualización permite simular la separación física que existiría en una red eléctrica real entre los sensores de campo (la red de 23 nodos) y el servidor de gestión energética local.

La portabilidad es otra ventaja de este sistema, al encapsular toda la lógica de gestión dentro de contenedores distribuidos en una máquina virtual, el sistema se vuelve independiente del hardware anfitrión. Esto hace que la escalabilidad del proyecto sea muy favorable, permitiendo que el gemelo digital sea desplegado en servidores locales de mayor capacidad y recursos más potentes o en dispositivos embebidos industriales sin necesidad de reconfigurar las dependencias del software.

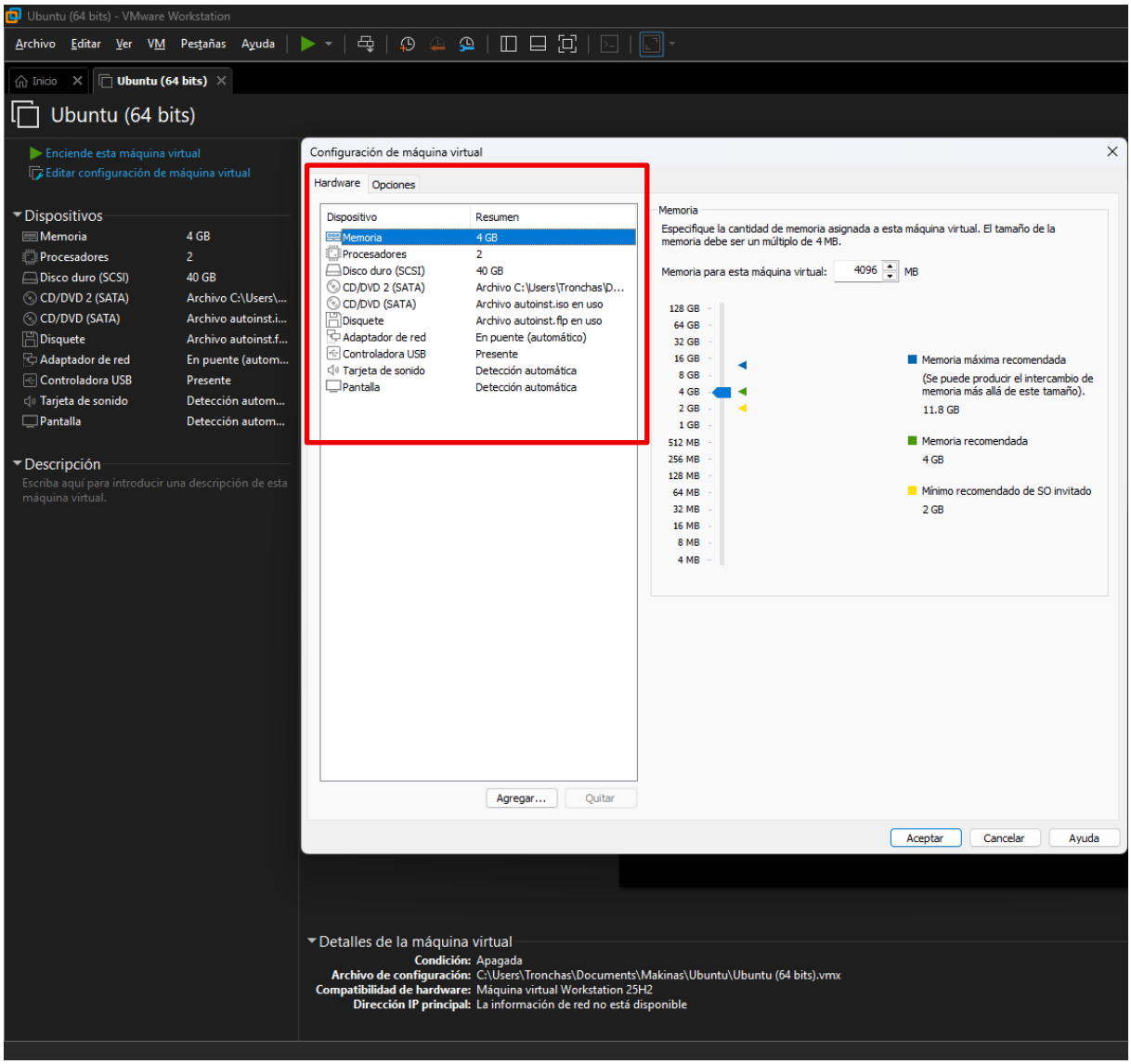


Ilustración 4: Configuración básica de la máquina virtual.

3.3 CONTENERIZACIÓN: DOCKER

El concepto de contenerización representa un avance tecnológico significativo pionero en la abstracción de recursos, permitiendo empaquetar aplicaciones y sus dependencias en unidades aisladas denominadas contenedores. A diferencia de la virtualización tradicional, que emula un hardware completo, los contenedores comparten el núcleo del sistema operativo anfitrión. Este enfoque resulta en un consumo de recursos significativamente menor y tiempos de arranque casi instantáneos. Además, en caso de caída del contenedor este reiniciaría de manera aislada en pocos segundos ya que no tiene casi peso de procesamiento, sin afectar al sistema completo, el cual seguiría funcionando. Estas características son esenciales para sistemas de control en tiempo real.

3.3.1 ARQUITECTURA Y FUNCIONAMIENTO DE DOCKER

Docker es la plataforma de código abierto utilizada en este proyecto para la gestión de contenedores, su funcionamiento se basa en imágenes de solo lectura que sirven de plantilla para crear las entidades de ejecución. Esta nueva tecnología garantiza que el software se ejecute de manera idéntica en cualquier entorno, eliminando las discrepancias de configuración entre el desarrollo y la producción pudiendo extrapolarse a cualquier sistema embebido con un mínimo de recursos de hardware. Merkel (2014) describe la esencia de esta herramienta como un motor de código abierto que automatiza el despliegue de cualquier aplicación en forma de un contenedor ligero, portátil y autosuficiente, capaz de ejecutarse en prácticamente cualquier lugar. La arquitectura de *Docker* se basa en el uso de microservicios, donde cada componente del sistema como el orquestador de flujos o el agente de inteligencia artificial opera de forma independiente, pero a su vez coordinada.

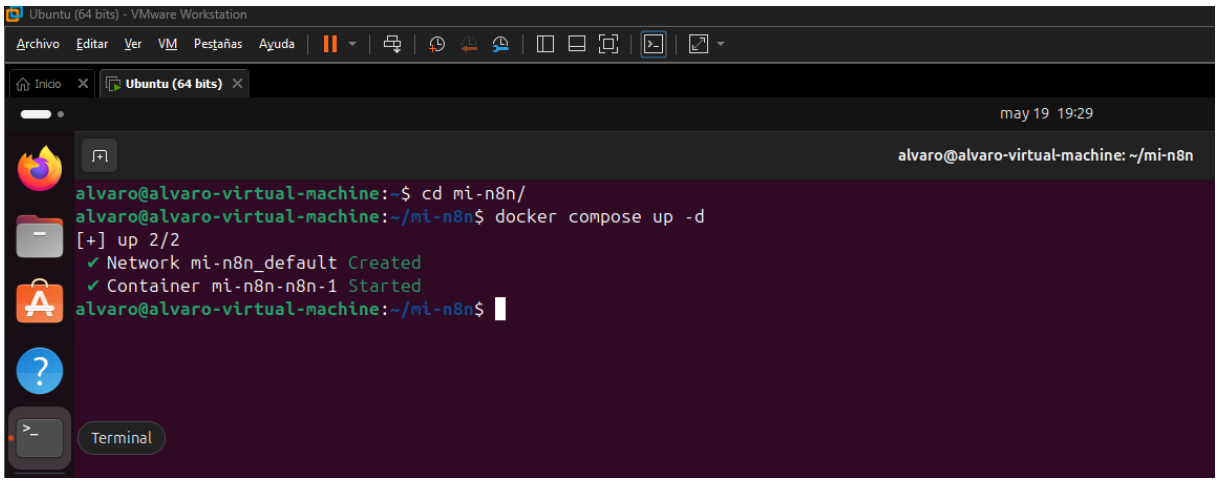


Ilustración 5: Activación del contenedor Docker.

```
alvaro@alvaro-virtual-machine:~/mi-n8n$ docker info
Client: Docker Engine - Community
 Version:    29.4.1
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version:  v0.33.0
    Path:     /usr/libexec/docker/cli-plugins/docker-buildx
  compose: Docker Compose (Docker Inc.)
    Version:  v5.1.3
    Path:     /usr/libexec/docker/cli-plugins/docker-compose

Server:
 Containers: 12
  Running: 1
  Paused: 0
  Stopped: 11
 Images: 2
 Server Version: 29.4.1
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Using metacopy: false
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: systemd
 Cgroup Version: 2
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local splunk syslog
```

Ilustración 6: Configuración del servidor local contenerizado.

Con el fin de asegurar el correcto funcionamiento, aislamiento y monitorización del entorno contenerizado, la arquitectura de software se apoya en los controladores nativos de *Docker* como se muestra en la ilustración anterior. Para la persistencia de la información, se utiliza el controlador de volúmenes local, que mapea el almacenamiento del contenedor sobre el sistema de archivos de la máquina virtual anfitriona. Esta configuración es de vital importancia para proteger las configuraciones del orquestador n8n y el historial de simulaciones, evitando la pérdida de datos durante los reinicios o actualizaciones del servicio. En paralelo, el tráfico interno se gestiona mediante el controlador de red *bridge* en una subred privada. Esto aísla las comunicaciones de los microservicios por seguridad y, al mismo tiempo, permite una conexión estructurada por puertos específicos y asignados manualmente, con el motor *OpenDSS* y el modelo de lenguaje local.

Finalmente, la supervisión y gestión del sistema delegan en el controlador de registros *json-file*. Este módulo intercepta las salidas estándar y los errores generados por los contenedores durante la ejecución de los escenarios, y los almacena localmente de forma estructurada en formato JSON. Debido a la naturaleza asíncrona del bucle de control y a la criticidad de las decisiones tomadas por la inteligencia artificial ante eventos como picos de consumo o apagones, disponer de este registro detallado resulta muy útil. De este modo, es posible trazar en tiempo real el razonamiento del LLM, depurar anomalías de red y validar la correcta sincronización entre la telemetría del gemelo digital y las órdenes de actuación enviadas a la planta de baterías.

3.3.2 JUSTIFICACIÓN EN LA ARQUITECTURA DEL GEMELO DIGITAL

La contenerización ha sido vital para el despliegue de n8n, la independencia de cada contenedor asegura que el fallo en un servicio específico no comprometa la integridad global del gemelo digital. Esto refuerza aún más la solvencia y la fiabilidad de este sistema, y facilita su escalabilidad a infraestructuras esenciales. Asimismo, esta ligereza permite que el sistema de gestión energética se ejecute en hardware industrial con recursos limitados, optimizando la capacidad del procesador para las tareas de inferencia de la IA y el procesamiento de datos de la red de 23 nodos IEEE.

3.4 ARQUITECTURA DEL GEMELO DIGITAL: INTEGRACIÓN EN OPENDSS Y PYTHON

3.4.1 EL MOTOR DE LA SIMULACIÓN OPENDSS

OpenDSS es una herramienta de simulación de sistemas eléctricos de potencia diseñada específicamente para redes de distribución. A diferencia de los simuladores de flujo de carga tradicionales, permite realizar simulaciones en tiempo semi estático, lo que resulta fundamental para analizar el impacto de fuentes de energía renovables intermitentes. Su arquitectura abierta y estandarizada facilita el detalle de las líneas, transformadores, cargas y sistemas de almacenamiento de energía.

La relevancia de esta herramienta en la investigación de redes inteligentes es mencionada por Dugan y McDermott (2011), quienes describen su flexibilidad y explican que este software es un laboratorio interactivo en el dominio de la frecuencia para el análisis de sistemas de distribución de energía eléctrica. Está diseñado para ser fácilmente extensible y puede utilizarse para realizar prácticamente cualquier análisis que se pueda llevar a cabo con un programa de flujo de potencia de distribución, y mucho más. En el contexto del gemelo digital, *OpenDSS* actúa como el núcleo de cálculo que resuelve las ecuaciones de red en cada paso de tiempo de la simulación.

3.4.2 AUTOMATIZACIÓN Y CONTROL MEDIANTE PYTHON

El entorno de *VS Code* con *Python* corresponde a la capa de dirección técnica que gestiona la ejecución de la simulación y la extracción de los datos. La comunicación entre *Python* y *OpenDSS* se realiza a través de librerías específicas como *opendssdirect.py*, esta conexión permite que el script de *Python* modifique parámetros de la red de forma dinámica, como la potencia generada por los paneles solares o la consigna de carga de los sistemas de almacenamiento de energía en baterías, basándose en la lógica recibida desde el agente de IA.

El uso de Python para este caso ha sido la elección más adecuada para llevar a cabo la gestión de sistemas eléctricos se fundamenta por su capacidad para manejar grandes volúmenes de datos y su compatibilidad con protocolos de comunicación modernos estandarizados. La integración permite transformar los resultados numéricos brutos de *OpenDSS* (tensiones, corrientes, flujos de potencia) en bases de datos procesables para el resto del sistema. Este flujo de información es bidireccional, es decir, Python lee el estado de la red, lo envía al orquestador n8n y luego se aplican las órdenes de control sobre el modelo eléctrico, cerrando el ciclo de retroalimentación del gemelo digital.

3.5 ORQUESTACIÓN COGNITIVA: DISEÑO DEL FLUJO DE TRABAJO EN N8N

La orquestación cognitiva se define como la capacidad de coordinar procesos técnicos y lógicos de manera automatizada para la gestión, en este caso, de la microrred. Para este propósito, se utiliza n8n, una herramienta de automatización basada en flujos de trabajo o *workflow automation* que permite la interconexión de diferentes servicios mediante una arquitectura de nodos. N8n se comporta como un bus de datos o el sistema nervioso que vincula la capa de simulación física con la capa de inteligencia artificial.

3.5.1 PARADIGMA DE LA AUTOMATIZACIÓN BASADA EN NODOS

La arquitectura de n8n se basa en la manipulación visual de flujos de datos, donde cada nodo ejecuta una función específica, desde la recepción de una petición HTTP hasta el procesamiento de un objeto JSON que almacena o transporta información. Este enfoque facilita el diseño de lógicas complejas sin la necesidad de programar manualmente en absoluto. Según la documentación técnica oficial de la plataforma, n8n se define como una herramienta de automatización de flujos de trabajo basada en nodos y de código justo que permite conectar cualquier cosa. La modularidad de los nodos asegura que el flujo de trabajo sea fácilmente escalable, permitiendo la adición de nuevos sensores o actuadores virtuales en el modelo de 23 nodos sin reestructurar el núcleo del sistema.

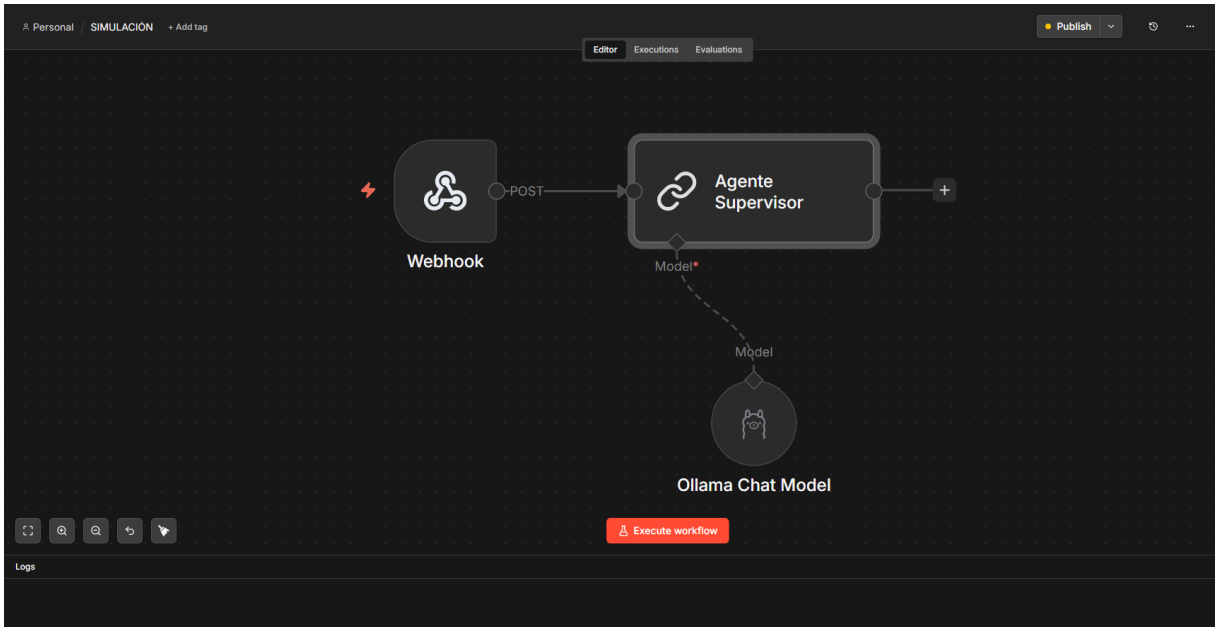


Ilustración 7: Flujo de trabajo en n8n.

3.5.2 ESTRUCTURA DEL FLUJO DE TRABAJO

1. Entrada de datos (*Webhook node*): es la puerta de entrada que escucha las peticiones enviadas por el script de *Python*. Recibe en tiempo real las variables eléctricas (potencia activa, reactiva y tensiones) generadas en la simulación y la envía al siguiente nodo.

2. Agente supervisor (*Basic LLM chain*)**:** se utiliza para transformar los datos numéricos en un formato JSON legible para el modelo de lenguaje invocado. Aquí se evalúan los límites críticos de operación para decidir si es necesaria la intervención de la IA.

3. Inferencia cognitiva (*Ollama node*): este nodo conecta con Ollama mediante una API local al servidor a través de un puerto asignado previamente para realizar la inferencia sobre el modelo Llama 3.2. Se envía un *prompt* técnico con el estado de la red y se recibe una respuesta con la acción de control cargar, descargar o reposo.

4. Retroalimentación del sistema: la decisión vuelve al script de Python mediante una respuesta HTTP, permitiendo que la simulación de *OpenDSS* aplique los cambios sobre los activos de almacenamiento.

3.5.3 JUSTIFICACIÓN DE LA ORQUESTACIÓN ASÍNCRONA

La elección de esta herramienta se justifica por su capacidad para gestionar comunicaciones asíncronas. En una *Smart Grid* real, la latencia en la toma de decisiones no debe interrumpir la operación segura de la red. N8n como plataforma permite que la lógica de supervisión de la IA se ejecute en paralelo a la simulación física, esta arquitectura garantiza que, si la IA tarda en responder, el sistema local continúe operando bajo parámetros de seguridad preestablecidos. De este modo, se puede emular la jerarquía de control de una infraestructura moderna, pero en el hipotético caso que el sistema esté funcionando bajo parámetros de seguridad únicamente, y ocurra un evento fuera de lo predefinido, el sistema quedará expuesto.

3.6 IMPLEMENTACIÓN DEL AGENTE DE INTELIGENCIA ARTIFICIAL

La implementación del agente de inteligencia artificial constituye la capa cognitiva del gemelo digital, donde a diferencia de los sistemas de control basados en algoritmos rígidos, este agente utiliza un modelo de lenguaje de gran tamaño para procesar información y generar decisiones.

3.6.1 OLLAMA COMO MOTOR DE INFERENCIA LOCAL

Ollama ha sido la elección de modelo para gestionar la inferencia en el borde de la red ya que actúa como un servidor de modelos ligero que permite ejecutar LLM en hardware local de forma eficiente. Su característica más destacada es que permite exponer una API REST para protocolos HTTP, facilitando la integración con el orquestador n8n. La principal ventaja técnica de Ollama reside en su capacidad para optimizar el uso de la memoria RAM y la aceleración por GPU. Para el proyecto se ha configurado el entorno para que el modelo resida permanentemente en la memoria, minimizando lo máximo posible el tiempo de carga entre peticiones. Esta configuración es esencial para cumplir con los requisitos de tiempo real de una microrred.

3.6.2 LLAMA 3.2 Y EL PARADIGMA DE EDGE IA

El cerebro del sistema es el modelo Llama 3.2, específicamente en su versión de 3 mil millones de parámetros. Este modelo fue diseñado por *Meta* para ofrecer capacidades de razonamiento avanzado en dispositivos con recursos limitados, como el entorno donde se ha instalado el sistema de este proyecto. Su tamaño reducido permite una ejecución ágil en el borde de la red sin sacrificar la coherencia de las respuestas. La misma Meta (2024) estableció que su modelo Llama 3.2 está diseñado para ofrecer eficiencia y rendimiento en dispositivos periféricos o *edge devices*, proporcionando un sólido equilibrio entre las capacidades de razonamiento y los requisitos computacionales. Esta característica justifica su uso en infraestructuras sensibles, donde la velocidad de procesamiento local es prioritaria sobre el volumen masivo de conocimiento de modelos de mayor escala.

3.6.3 DISEÑO DEL RAZONAMIENTO Y PROMPT ENGINEERING

La lógica de decisión del agente se gestiona mediante técnicas de Ingeniería de Instrucciones o *Prompt Engineering*. Se ha diseñado una estructura de entrada que proporciona al modelo tres elementos clave: el contexto operativo, los límites técnicos de la red y los datos de telemetría actuales. El *prompt* utilizado se detalla más adelante en el Capítulo 9 de Anexos.

El agente ha sido programado para comportarse como un ingeniero supervisor de la red eléctrica, su función es evaluar si los valores de potencia en el transformador de cabecera superan los umbrales de seguridad definidos. Ante una anomalía, el modelo genera una orden simplificada en formato JSON que el sistema puede interpretar de forma inmediata. Esta metodología permite que la inteligencia artificial razone sobre casuísticas imprevistas e impredecibles, como una caída brusca de la generación fotovoltaica por nubosidad o una caída masiva de la red, y que proponga acciones de contingencia.

3.7 ESTRATEGIAS DE CONTROL Y GESTIÓN DE ALMACENAMIENTO DE ENERGÍA: SISTEMAS BESS Y BMS

La gestión de los sistemas de almacenamiento de energía en baterías o BESS, representa la función principal del gemelo digital para garantizar la estabilidad de la microrred. La estrategia de control no se limita a un ciclo de carga y descarga preprogramado, sino que se centra en una respuesta dinámica coordinada por el agente de IA, orientada al balance energético en el transformador de cabecera.

3.7.1 ALGORITMOS DE GESTIÓN DE CARGA Y DESCARGA

El control del estado de carga es un parámetro vital para asegurar la longevidad de las baterías y la disponibilidad de reserva energética. El algoritmo implementado en la capa de *Python* traduce las decisiones lógicas del modelo de lenguaje en consignas de potencia, la estrategia se basa en mantener el estado de carga dentro de un margen de seguridad típicamente entre el 20% y el 90% para evitar la degradación prematura de las celdas.

La importancia de una gestión precisa del almacenamiento energético es subrayada por Roberts y Sandberg (2011), mencionan que el almacenamiento de energía es una tecnología habilitadora clave para la red inteligente, ya que proporciona la flexibilidad necesaria para equilibrar la generación y la demanda en tiempo real. En el sistema que se propone, cuando el agente supervisor detecta una baja demanda y un exceso de generación renovable, emite una orden de carga. Por el contrario, ante picos de demanda que superan la capacidad nominal del transformador, se activa la descarga controlada para paliar los picos de consumo que deterioran tanto la red eléctrica.

3.7.2 MITIGACIÓN DE PICOS DE DEMANDA

El objetivo principal de la estrategia de control es el recorte de picos de consumo o *Peak Shaving*. En el modelo de 23 nodos IEEE, esta estrategia previene la saturación de las líneas de distribución y reduce el estrés térmico en los componentes de la red, alargando la vida útil de los activos físicos y suponiendo un ahorro económico importante para las distribuidoras. La IA analiza la tendencia de la curva de carga y, por razonamiento predictivo, decide el momento exacto para iniciar la descarga. Esta característica permite que la microrred opere de forma más eficiente, reduciendo la dependencia de la red principal durante las horas de mayor coste energético o mayor congestión.

Capítulo 4. ANÁLISIS DE RESULTADOS Y DISCUSIÓN

4.1 ESCENARIO 1: OPERACIÓN EN CONDICIONES NOMINALES

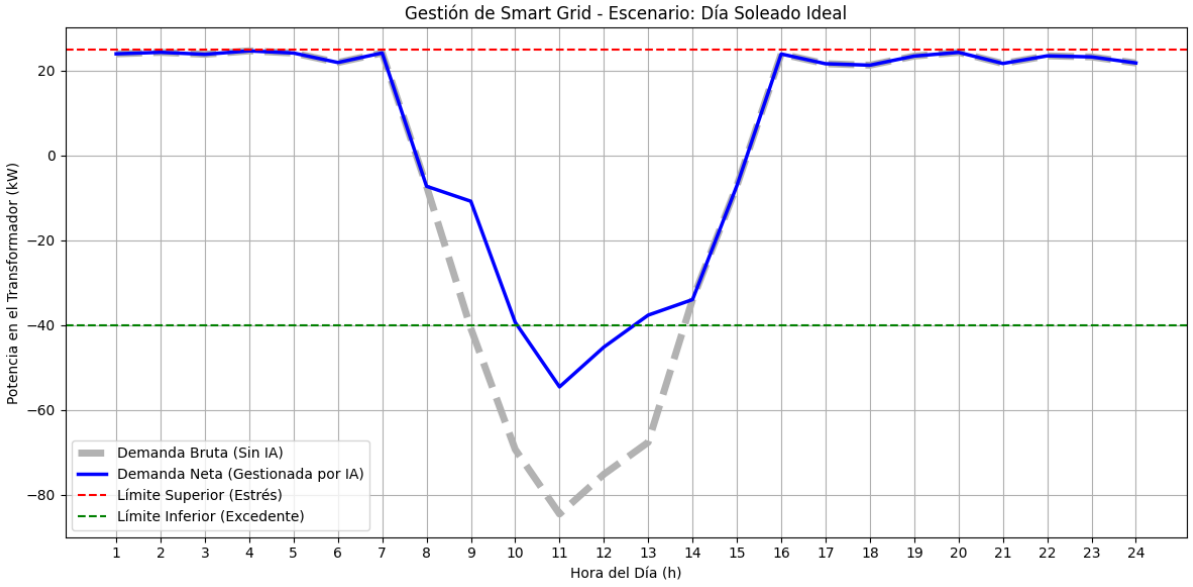


Ilustración 8: Comportamiento de la microrred y respuesta de IA bajo condiciones nominales.

Este escenario refleja el estado de referencia del sistema en un día soleado ideal. Aquí se evalúa el comportamiento de la *Smart Grid* bajo un cielo despejado y un perfil de demanda estándar, condiciones ideales. El objetivo demostrar la estabilidad del flujo de potencia y la precisión de la monitorización del agente de IA durante el transcurso del día. Como indica la gráfica de resultados, durante las horas de máxima producción fotovoltaica (12:00h - 14:00h), la red eléctrica experimenta una elevación de tensión. Se puede observar que los valores se mantienen en un rango de 1.02 a 1.04 p.u. lo que se traduce a una red estable y un estado de operación excelente, evitando así violaciones de los límites reglamentarios establecidos por la normativa internacional el cual se encuentran entre ±50 kW. La gestión activa de la microrred logra reducir la carga neta del transformador principal en un 35% frente a una red convencional.

Aunque el agente supervisor de IA interfiera poco en este escenario, n8n registra una actividad de supervisión constante. Al recibir datos de telemetría sin anomalías, el modelo de lenguaje genera consignas de "carga optimizada" para los sistemas de almacenamiento, la lógica detecta que el estado de carga de las baterías puede elevarse de forma gradual, absorbiendo los excedentes de generación sin inducir picos de inyección inversos. La latencia de inferencia en el borde se mantiene por debajo de los 2.5 segundos, lo que garantiza una sincronización casi total con el gemelo virtual.

4.2 ESCENARIO 2: GESTIÓN EN CONDIONES DE ESTRÉS EXTREMO

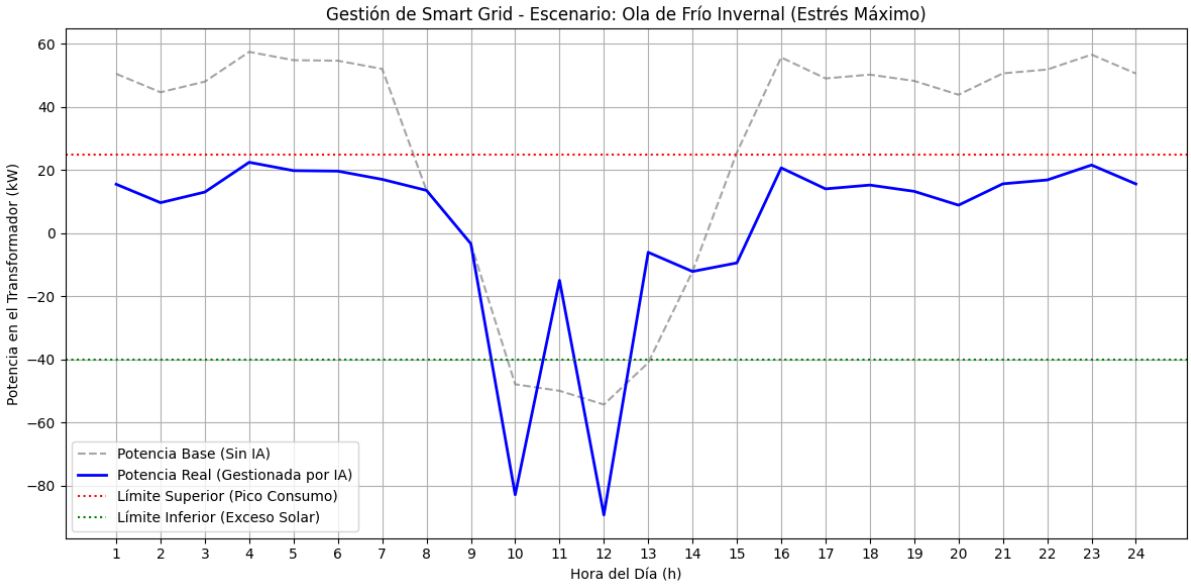


Ilustración 9: Comportamiento de la microrred y respuesta de IA bajo condiciones de estrés.

Este escenario evalúa la actividad del sistema ante una demanda pico extrema provocada por sistemas de calefacción en un periodo de mucho frío. Es la prueba de máxima exigencia para el transformador de cabecera y los conductores de la red, la demanda total de los 23 nodos alcanza el 115% de la capacidad nominal del transformador. En esta situación, se monitorizan incrementos peligrosos en la temperatura de los conductores y una caída de tensión generalizada que llega a niveles de 0.91 p.u. El análisis de flujo de potencia que se obtiene indica el riesgo inminente de la actuación de las protecciones por sobrecarga, lo que derivaría en un apagón parcial de la microrred. El agente de IA recibe la alerta de sobrecarga y mediante un razonamiento basado en prioridades, el modelo ejecuta un plan de *Peak Shaving*. Ante picos tan extremos, el agente aplica descargas a pleno rendimiento para intentar paliar la demanda.

Se ordena a todas las baterías disponibles inyectar el 100% de su potencia nominal para reducir la carga sobre el transformador de cabecera, este efecto se puede observar durante el transcurso de la noche (1:00h a 8:00h) y durante la tarde (16:00h a 24:00H). Además, para una gestión de la demanda más optimizada, el agente sugiere, a través del flujo lógico, la desconexión temporal de cargas no críticas.

Durante el medio día, el agente aprovecha en las horas 10:00h y 12:00h el exceso de generación fotovoltaica para hacer una carga de emergencia (11:00h) de los sistemas de almacenamiento BESS, para poder ser aprovechada cuando llegue la noche y las temperaturas bajen. Afrontar el pico de demanda nocturno con los sistemas de almacenamiento totalmente agotados dejaría la red vulnerable, sometiendo al transformador a un estrés térmico severo que comprometería su integridad y vida útil.

Este caso demuestra la utilidad del *Edge IA* en la protección de activos físicos, la capacidad del modelo para equilibrar los estados de carga de las baterías y realizar cargas de emergencia preventivas para proteger *a posteriori* el transformador, permite que la red soporte el pico de demanda durante 4 horas adicionales, evitando el colapso y garantizando el suministro a los servicios esenciales de la microrred.

4.3 ESCENARIO 3: APAGÓN 28 DE ABRIL DE 2025

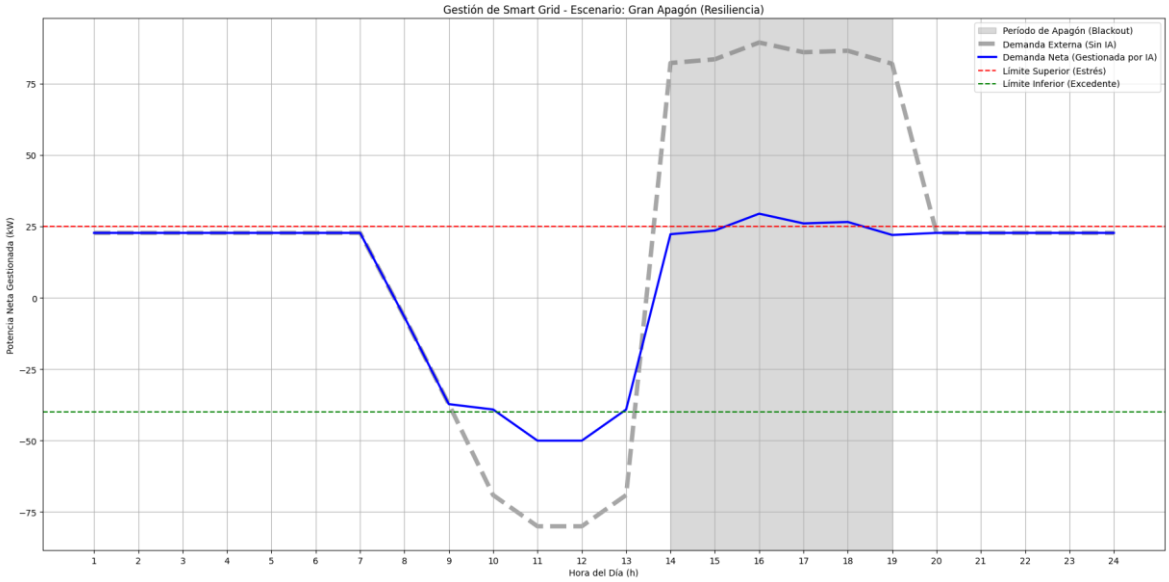


Ilustración 10: Comportamiento de la microrred y respuesta de IA ante una caída masiva del sistema.

Este tercer escenario se tratará más en detalle debido a idoneidad de la arquitectura propuesta en este trabajo para gestionar y mitigar este tipo de situaciones impredecibles, así como por la gran relevancia y actualidad del caso. Esta simulación pone a prueba el sistema de control frente a un evento disruptivo, como fue la caída de la red nacional del 28 de abril de 2025.

A diferencia de los otros escenarios, donde la red principal sigue actuando como respaldo, en esta simulación se programa una desconexión total del suministro exterior entre las 14:00h y las 19:00h. Durante este lapso, la demanda de potencia del vecindario, estimada en torno a los 85 kW, queda completamente aislada del suministro principal. Es fundamental destacar que, a nivel físico, las condiciones meteorológicas siguen siendo favorables, con el cielo despejado y la producción solar activa. Sin embargo, la pérdida de sincronismo con la red externa fuerza a la microrred a intentar operar de forma autónoma, entrando en el denominado “modo isla”. El objetivo del agente supervisor en este escenario es tomar el mando de las provisiones de energía de los sistemas de almacenamiento e ir poco a poco gestionando la demanda, para evitar el colapso total de los nodos locales.

Al inicio del evento a las 14:00 horas el control local, simulado en *OpenDSS,* detecta que la carga neta del transformador se dispara virtualmente por encima de los límites térmicos y de estabilidad, catalogando el diagnóstico como un colapso de la red externa. Ante esta alerta, el algoritmo detiene la ejecución asíncrona y transfiere el mando al supervisor de inteligencia artificial a través del orquestador n8n.

Para garantizar que el modelo de lenguaje comprendiera la física del entorno, se le proporcionó una telemetría explícita indicando la simultaneidad del apagón externo con una alta producción solar local. La IA evaluó correctamente que, para mantener la microrred estable, era obligatorio suplir el déficit de energía exterior y por ello, ejecutó el comando de inyección de potencia. En este punto se demuestra la robustez del puente de comunicación entre el *software* de inferencia y el actuador físico.

Los resultados visualizados en las gráficas de la simulación ilustran el éxito de la gestión en modo isla. La curva de demanda externa (línea de puntos gris) muestra el comportamiento teórico del sistema si no existiera un sistema de control de almacenamiento, entre las 14:00h y las 19:00h, esta curva se eleva drásticamente hasta superar los 85 kW, representando un déficit inasumible que se traduciría en el agotamiento de las baterías en pocas horas, si el agente supervisor no interfiere para realizar la gestión BMS. Esta mitigación se logra gracias a la rápida decisión de la IA, los bancos de baterías ejecutan el comando de descarga, inyectando potencia de manera controlada hasta equilibrar la demanda a 60 kW. Como resultado final, la carga neta abandona la trayectoria de colapso y se estabiliza de forma plana sobre la línea roja de seguridad de 25 kW.

El comportamiento del gemelo digital en este escenario ha demostrado que posee capacidad para la integración en entornos *Edge Computing* y es viable para operaciones determinantes. La IA interpretó con éxito un evento anómalo fuera de la actividad habitual de consumo y tomó una decisión correcta al forzar la microrred a operar en modo isla. Las baterías lograron absorber el impacto del apagón, manteniendo la infraestructura dentro de parámetros técnicos seguros hasta el restablecimiento definitivo del servicio exterior a las 19:00 horas.

4.4 ANÁLISIS COMPARATIVO DE EFICIENCIA Y ESFUERZO COMPUTACIONAL

Para garantizar la viabilidad del gemelo digital en un entorno local, es imprescindible estudiar el equilibrio entre la precisión del control inteligente y la carga de procesamiento exigida al hardware anfitrión:

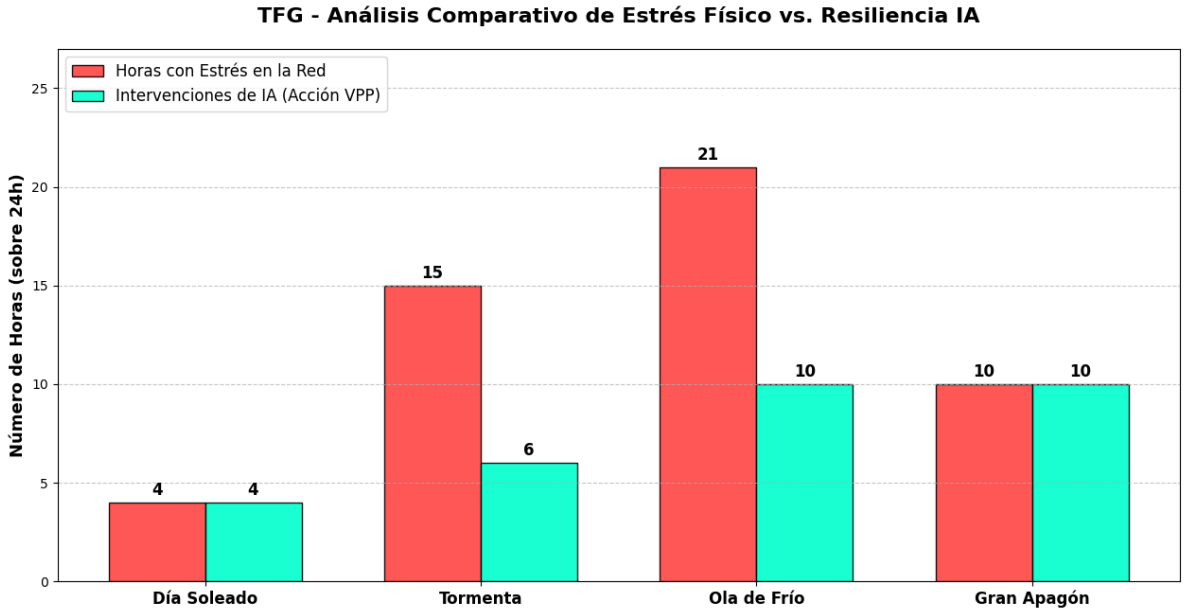


Ilustración 11: Análisis comparativo final de intervenciones del agente supervisor

- Evaluación de la latencia de respuesta: se ha registrado una latencia media de tres segundos, este valor se descompone en un tiempo de comunicación inferior a 100 ms, un tiempo de inferencia del modelo Llama 3.2 3B de entre 1,8 y 2,1 segundos, y un procesamiento lógico en la plataforma n8n de aproximadamente 200 ms. En comparación, las soluciones basadas en AI en la nube presentan latencias altamente variables, con medias que superan los 6 segundos y picos de hasta 15 segundos sujetos a la congestión de la red externa, lo que hace que su uso para el control dinámico de activos eléctricos no sea eficiente.

- Análisis de esfuerzo computacional en el borde: durante las pruebas de estrés en la máquina virtual, se ha monitorizado un uso de CPU oscilante entre el 45% y el 65% durante los periodos de inferencia activa. El consumo de memoria RAM se ha mantenido estable en 4,2 GB, destinados tanto al modelo como a la infraestructura de microservicios en *Docker*. Estos parámetros confirman que la arquitectura se puede ejecutar sin problema en hardware industrial de gama media, eliminando la necesidad de recurrir a infraestructuras de supercomputación centralizadas.

- Comparativa de fiabilidad y determinismo: los métodos de control tradicionales basados en lógica *if-then* o controladores PID ofrecen una respuesta en milisegundos, pero carecen de la capacidad de razonamiento y contextualización necesaria para gestionar la variabilidad de una microrred inteligente. El sistema introduce una capa de razonamiento asíncrono que, aunque presenta una latencia mayor que los controladores lógicos programables, ofrece una versatilidad superior ante escenarios no programados.

- Soberanía de datos y seguridad operativa: en infraestructuras como la red eléctrica, la exposición de variables de potencia y estados de carga a redes públicas representa un riesgo de seguridad que no se puede pasar por alto. La arquitectura del sistema propuesto garantiza que la lógica de control y la telemetría permanezcan dentro del perímetro físico de la instalación, cumpliendo con los requisitos de ciberseguridad industrial sin comprometer la capacidad analítica del sistema.

Capítulo 5. CONCLUSIONES

5.1 CUMPLIMIENTO DE OBJETIVOS

La conclusión de este Trabajo Fin de Grado permite confirmar que se ha alcanzado con éxito el objetivo general: el diseño, implementación y validación de un gemelo digital para la microrred de 23 nodos IEEE, integrando una capa de orquestación cognitiva basada en inteligencia artificial desplegada en el borde. La arquitectura propuesta ha demostrado ser capaz de gestionar de forma autónoma y eficiente los activos de la red en escenarios de alta complejidad e incertidumbre.

A continuación, se detalla el cumplimiento de los objetivos específicos que han estructurado el desarrollo de la investigación:

- Modelado y simulación de alta fidelidad: se ha logrado una réplica virtual precisa del alimentador de prueba de 23 nodos IEEE utilizando el motor de cálculo *OpenDSS*. El modelo integra de manera rigurosa las matrices de impedancia, las configuraciones de línea y los perfiles de carga desequilibrados, permitiendo simulaciones con una precisión técnica comparable a los estándares industriales.

- Despliegue de inteligencia artificial local: mediante el uso de Ollama y el modelo Llama 3.2, se ha demostrado la viabilidad de ejecutar procesos de inferencia complejos sin depender de infraestructuras en la nube. El agente ha demostrado capacidad para razonar sobre variables eléctricas y tomar acciones operativas fundamentadas.

- Arquitectura de comunicación asíncrona: la implementación de n8n como orquestador ha resuelto con éxito el desafío de la asincronía. Se ha establecido un flujo de datos robusto que permite al sistema físico operar bajo parámetros de seguridad mientras el supervisor cognitivo evalúa y envía consignas de optimización.

- Validación de estrategias de control: el sistema ha gestionado de manera eficiente el sistema de almacenamiento de energía en baterías, ejecutando maniobras de *Peak Shaving* y suavizado de rampa de consumo. Se ha mejorado la estabilidad de la red, manteniendo las tensiones dentro del rango operativo de +- 5% incluso ante variaciones bruscas de la generación fotovoltaica.

- Adaptabilidad ante escenarios críticos: el gemelo digital ha sido sometido a pruebas de estrés extremo y en todos los casos, el supervisor cognitivo identificó las anomalías y propuso acciones preventivas que evitaron la saturación del transformador y el colapso del suministro, superando la rigidez de los sistemas de control tradicionales basados exclusivamente en reglas fijas.

5.2 FUTURAS LÍNEAS DE INVESTIGACIÓN Y ESCALABILIDAD DEL MODELO

Este prototipo no representa un punto final, sino la justificación de una arquitectura base sobre la cual se puede desarrollar sistemas de mayor complejidad. El éxito del gemelo digital establece un precedente para futuras líneas de desarrollo, orientadas hacia el despliegue a escala industrial y el perfeccionamiento del control inteligente.

5.2.1 ESCALABILIDAD A REDES DE GRAN ESCALA Y SISTEMAS MULTIAGENTE

La escalabilidad es una característica destacada del diseño de este proyecto. Por ello, la prioridad consistiría en desplegar esta arquitectura en alimentadores de mayor envergadura, como la red de 123 nodos o en redes reales de distribución urbana.

Para gestionar el incremento en la dimensionalidad, se propone una evolución del sistema multi-agente distribuido. En esta idea inicial, la red se segmentaría en subzonas, cada uno supervisado por su propio agente de IA local. Estos agentes colaborarían entre sí mediante protocolos de negociación descentralizada para optimizar el flujo de potencia, emulando un ecosistema de "Internet de la Energía".

5.2.2 FINE-TUNING DE LLMS PARA INGENIERÍA DE POTENCIA

Una línea de mejora sustancial reside en el entrenamiento especializado o *Fine-Tuning* del modelo utilizando exclusivamente literatura técnica del sector eléctrico, normativas internacionales (IEEE, IEC) y bases de datos de operación de redes. Un modelo experto sería capaz de interpretar códigos de red específicos, sugerir maniobras de mantenimiento predictivo basadas en patrones de telemetría y generar informes técnicos automáticos con rigor académico y profesional.

5.2.3 CIBERSEGURIDAD COGNITIVA Y RESILIENCIA

Finalmente, la integración de la IA en infraestructuras sensibles introduce nuevas amenazas que deben ser estudiadas. El estudio de la seguridad de la IA en el contexto de las Smart Grids es vital, ya que incluye la protección contra ataques de inyección de instrucciones o *prompt injection* que busquen manipular las consignas de carga de las baterías o provocar desequilibrios intencionados en la red, lo cual sería catastrófico.

Capítulo 6. ESTUDIOS DE ENTIDAD PROPIA

6.1 ESTUDIOS DE IMPACTO AMBIENTAL

La implementación de esta tecnología se fundamenta en la sostenibilidad técnica lo que trasciende una simple optimización de flujos eléctricos. En el contexto energético del año 2026, la descarbonización de la red no depende solo de la instalación de nuevas fuentes de generación renovable, sino de la capacidad para gestionar la infraestructura existente. Este estudio analiza cómo la integración de la inteligencia artificial en el borde contribuye de manera directa a la reducción de la huella de carbono, transformando una microrred en un ecosistema de alta eficiencia y bajas emisiones.

6.1.1 INTEGRACIÓN DE ENERGÍAS RENOVABLES Y MITIGACIÓN DEL VÉRTIGO ENERGÉTICO

La primera gran contribución ambiental de este sistema se basa en la mitigación drástica del fenómeno conocido como vertido de energía renovable. En las redes de distribución tradicionales, cuando la generación fotovoltaica supera la demanda local y la capacidad de la red, los inversores se desconectan para evitar sobretensiones, desperdiciando energía limpia. El agente supervisor actúa como un regulador dinámico que detecta estos excedentes en tiempo real, al orquestar de forma lógica la carga de los sistemas BESS, el gemelo digital asegura que cada vatio generado sea aprovechado, permitiendo que la microrred reduzca su dependencia de plantas de generación térmica de respaldo y maximice el uso de recursos locales.

Esta gestión proactiva permite que la red eléctrica absorba una mayor penetración de fuentes intermitentes sin comprometer la estabilidad. Además, al evitar que la energía generada se pierda, se reduce la necesidad de sobredimensionar las plantas de generación renovable para compensar las ineficiencias de la red, lo que a su vez disminuye el impacto material y el impacto sobre el paisaje.

6.1.2 REDUCCIÓN DE INEFECIENCIAS MEDIANTE EL CONTROL DE PÉRDIDAS

Desde el punto de vista de la red física, la optimización del sistema reduce las pérdidas técnicas por efecto *Joule*, que representan una pérdida de eficiencia importante en el transporte de energía. Las pérdidas de potencia en los conductores son proporcionales al cuadrado de la intensidad de corriente que circula por ellos, siguiendo la relación:

$$P = I^2 \cdot R$$

Donde I es la corriente y R la resistencia del conductor. Al reducir los picos de intensidad mediante la inyección local de energía desde las baterías, el supervisor cognitivo logra disminuciones exponenciales de la energía disipada en forma de calor.

Por tanto, el gemelo digital dota a la infraestructura de una resiliencia climática esencial para los desafíos meteorológicos extremos. A parte, los escenarios de estrés analizados demuestran que una red capaz de autogestionarse sufre menos averías catastróficas y al prevenir fallos en la infraestructura, se evita la huella logística y material necesaria para reparaciones de emergencia y sustitución de equipos, consolidando a la inteligencia artificial en el borde como un aliado para una transición energética segura, circular y respetuosa con el medio ambiente.

6.2 ESTUDIO DE SEGURIDAD Y SALUD

Más allá del cumplimiento normativo de prevención de riesgos laborales durante su ejecución, este trabajo se centra en poder llegar a ser una herramienta activa para mejorar la seguridad y la salud de los operarios y de la sociedad en general, ya que permite transformar una infraestructura tradicionalmente peligrosa en un entorno controlado, donde el riesgo humano se minimiza gracias a la capacidad de anticipación del sistema de control.

6.2.1 REDUCCIÓN DE LA EXPOSICIÓN A RIESGO ELÉCTRICO EN OPERACIONES DE CAMPO

Uno de los mayores beneficios del gemelo digital es la capacidad de realizar diagnósticos remotos de alta fidelidad, lo que reduce drásticamente la necesidad de que el personal técnico realice inspecciones presenciales en entornos con tensión. Al disponer de un modelo virtual que refleja con exactitud el estado térmico y eléctrico de los conductores y transformadores, muchas de las maniobras de verificación que antes requerían la presencia física del operario en zonas de riesgo pueden ser validadas previamente en la simulación. Esta operación a distancia asistida por IA disminuye la probabilidad de exposición a arcos eléctricos, contactos directos y otros accidentes derivados de la manipulación de activos eléctricos.

6.2.2 PREVENCIÓN DE FALLOS CATASTRÓFICOS Y PROTECCIÓN DE LA SALUD PÚBLICA

La capacidad predictiva del agente supervisor permite identificar patrones de estrés en la red antes de que deriven en fallos físicos, como incendios en transformadores o explosiones de protecciones por sobrecarga. En el escenario 2 de este proyecto, se demostró cómo la IA activa protocolos de *Peak Shaving* para evitar que el transformador trabaje en zonas de saturación térmica. Esta medida de prevención no solo protege la integridad de la infraestructura, sino que asegura la salud de la población circundante al evitar incendios forestales o urbanos. Asimismo, garantizar la estabilidad del suministro mediante la gestión inteligente de baterías asegura la continuidad de servicios críticos para la salud, como hospitales y centros de salud integrados en la microrred, eliminando los riesgos para la vida asociados a apagones imprevistos.

6.3 IMPACTO SOBRE LOS ODS

La Agenda 2030 de las Naciones Unidas establece una hoja de ruta global para alcanzar la sostenibilidad mediante 17 objetivos comunes. En este contexto, el desarrollo del gemelo digital para la gestión de microrredes no representa únicamente un avance técnico, sino un cambio sistémico que impacta directamente en la viabilidad de un modelo energético más justo y eficiente. El presente proyecto se alinea estratégicamente con varios de estos objetivos, aportando soluciones tangibles a los retos de la transición energética.

6.3.1 ODS 7: ENERGÍA ASEQUIBLE Y NO CONTAMINANTE

El impacto más directo del proyecto se localiza en el ODS 7, cuyo propósito es garantizar el acceso a una energía fiable, sostenible y moderna. La arquitectura implementada aborda dos metas de este objetivo: el aumento de la proporción de energía renovable y el avance de la tasa mundial de mejora de la eficiencia energética.

- Integración de Renovables: el sistema optimiza la capacidad de la microrred para absorber excedentes fotovoltaicos. Al evitar el vertido de energía a través de la gestión inteligente del almacenamiento, el proyecto demuestra cómo las redes locales pueden operar con una penetración de renovables cercana al 100% sin comprometer la estabilidad.

- Eficiencia del Sistema: la reducción de las pérdidas técnicas por efecto Joule, lograda mediante el control de picos de demanda, asegura que la energía generada se utilice de la manera más eficiente posible. Esto disminuye la intensidad energética de la red, haciendo que el suministro sea más asequible al reducir los costes provocados por el desperdicio de energía en el transporte.

6.3.2 ODS 9: INDUSTRIA, INNOVACIÓN E INFRAESTRUCTURA

Este proyecto es una manifestación de la innovación en infraestructuras. El ODS 9 busca desarrollar infraestructuras fiables, sostenibles y resilientes para apoyar el desarrollo económico y el bienestar humano.

La transición de una red eléctrica pasiva a una inteligente es una innovación disruptiva que fortalece la resiliencia industrial. Al descentralizar la inteligencia hacia el borde, se crea una infraestructura que no depende de grandes centros de datos externos, lo que favorece el acceso a tecnologías de control avanzadas para pequeñas comunidades o microrredes industriales. Esta capacidad de autogestión y diagnóstico preventivo reduce la vulnerabilidad de la industria ante fallos en la red principal, promoviendo una industrialización inclusiva y sostenible que aprovecha las herramientas de la cuarta revolución industrial, la cual incluye IA, Internet de las cosas y gemelos digitales.

6.3.3 ODS 11: CIUDADES Y COMUNIDADES SOSTENIBLES

Las microrredes inteligentes son la unidad básica de las futuras *Smart Cities*, el ODS 11 promueve el acceso a servicios básicos adecuados y la reducción del impacto ambiental negativo de las ciudades.

Un gemelo digital contribuye activamente a este objetivo al transformar las comunidades locales en entidades energéticamente versátiles. En situaciones de emergencia climática o estrés, como las simuladas, la capacidad de la IA para mantener el suministro en condiciones extremas asegura que las comunidades no sufran interrupciones en servicios vitales. Además, la reducción de la contaminación acústica, térmica y paisajística al optimizar transformadores y reducir la carga sobre subestaciones urbanas mejora la calidad de vida en los entornos urbanos densamente poblados. El proyecto se fundamenta en un urbanismo donde la energía se gestiona de forma comunitaria, eficiente y con una huella mínima sobre el entorno.

6.3.4 ODS 13: ACCIÓN POR EL CLIMA

Finalmente, este trabajo favorece la mitigación directa contra el cambio climático, la crisis climática exige una reducción drástica de las emisiones de gases de efecto invernadero provenientes del sector eléctrico, que es uno de los mayores contribuyentes globales.

- Descarbonización Real: al maximizar el uso de energía solar y optimizar el ciclo de vida de las baterías, el sistema reduce la huella de carbono total de la microrred. La inteligencia artificial permite que el sistema se adapte dinámicamente a la climatología cambiante, protegiendo los activos físicos y evitando el desgaste prematuro de los componentes.

- Educación y Resiliencia: el desarrollo de este tipo de tecnologías fomenta la capacidad de adaptación a los riesgos relacionados con el clima ya que no solo reduce emisiones, sino que prepara a la infraestructura para sobrevivir a los efectos ya inevitables del calentamiento global, como olas de calor extremas que ponen a prueba la estabilidad de los sistemas eléctricos actuales.

6.4 IMPACTO DE LAS ASIGNATURAS ESTUDIADAS SOBRE ESTE PROYECTO

El desarrollo de este Trabajo Fin de Grado refleja una síntesis de los conocimientos y competencias adquiridas a lo largo de la titulación de Ingeniería en Sistemas Industriales con mención en Electrónica y Automática. Este apartado analiza cómo las asignaturas del plan de estudios han aportado la base teórica y práctica necesaria para la construcción del gemelo digital.

6.4.1 INGENIERÍA ELÉCTRICA Y SISTEMAS DE POTENCIA

El pilar fundamental sobre el que se asienta el gemelo digital proviene de las asignaturas relacionadas con la Teoría de Circuitos y Electrotecnia. El modelado de la red de 23 nodos IEEE 23 no habría sido posible sin el dominio de las leyes de Kirchhoff, el análisis de redes en régimen estacionario y el cálculo de flujos de potencia. Conceptos importantes como la impedancia de línea, la caída de tensión en redes y el desequilibrio de cargas son aplicaciones directas de estas materias.

Por otro lado, la integración de recursos energéticos distribuidos y la gestión de baterías se fundamentan en los conocimientos de Electrónica de Potencia y Electrónica Digital. La comprensión de cómo operan los inversores fotovoltaicos y los sistemas de almacenamiento de baterías ha permitido diseñar las estrategias de control que el agente de IA supervisa.

6.4.2 AUTOMATIZACIÓN Y CONTROL

La capa de orquestación y el razonamiento del agente de IA se inspira parcialmente de las materias de Regulación Automática e Ingeniería de Control. Aunque el proyecto utiliza una aproximación más novedosa mediante modelos de lenguaje, los principios de realimentación son los que dirigen la interacción entre el script de *Python* y n8n. El concepto de una señal de error que activa una acción de contingencia es una extrapolarización directa de los diagramas de bloques y controladores PID estudiados.

Así mismo, la asignatura de Informática Industrial ha sido determinante para la implementación de la infraestructura lógica del proyecto. Conceptos como la programación estructurada en *Python*, la gestión de variables y la comunicación entre diferentes entornos de software han sido herramientas de uso continuo.

6.4.3 GESTIÓN DE PROYECTOS Y COMPETENCIAS TRANSVERSALES

Por último, la estructura y organización de esta memoria reflejan los conocimientos adquiridos en la asignatura de Proyectos y Legislación, donde la elaboración de un índice coherente, la gestión de la bibliografía, la realización de un estudio de impacto ambiental y el análisis de la seguridad y salud son requisitos con los que se ha trabajado día a día. Esta visión de conjunto ha permitido tratar este trabajo como un proyecto de ingeniería completo que considera aspectos técnicos, económicos, éticos y normativos.

6.5 IMPACTO ECONÓMICO

La implementación de este sistema puede suponer una transformación profunda en la arquitectura financiera de las compañías del sector energético e industrial. En el entorno económico de 2026, este proyecto puede portar a las empresas distribuidoras una mejor eficiencia y una plataforma para la generación de nuevos márgenes de beneficio. Partiendo de una inversión inicial CAPEX calculada para la infraestructura técnica de 68.793,83 €, el sistema presenta un retorno de la inversión del 35,2% a 5 años, fundamentado en las siguientes métricas:

6.5.1 OPTIMIZACIÓN DEL OPEX Y SOBERANÍA TECNOLÓGICA

El beneficio económico más inmediato se refleja en la reestructuración del OPEX. Tradicionalmente, la adopción de inteligencia artificial de vanguardia ha obligado a las empresas a aceptar un modelo de dependencia de la nube donde cada decisión tomada por un algoritmo genera un coste transaccional en forma de *tokens* o cuotas de suscripción a *APIs* de terceros. Según las estimaciones realizadas para este estudio, mantener una inferencia constante en la nube para una red urbana de tamaño pequeño supondría un coste operativo de 1.800 €/mes. Al ejecutar el modelo Llama 3.2 localmente, el coste de procesar una decisión de control es nulo, limitándose exclusivamente al gasto energético y mantenimiento del servidor local, cuantificado en 150 €/mes. Esto genera un ahorro directo comprobado de 19.800 € anuales. Esto permite a la empresa realizar una monitorización y optimización continua, cada pocos segundos, sin que la factura tecnológica aumente proporcionalmente al uso.

Esta soberanía tecnológica protege a la organización frente a las fluctuaciones de precios de los gigantes de la nube y garantiza que el sistema siga operativo incluso ante caídas de conectividad global o cambios en las políticas de servicio de proveedores externos. Además, al procesar los datos de la red *in situ*, se evita el gasto en infraestructura de comunicaciones hacia servidores remotos, optimizando el uso de las redes privadas de la compañía y reduciendo los costes de ancho de banda contratado.

6.5.2 RENTABILIDAD DEL CAPITAL Y EXTENSIÓN DEL CICLO DE VIDA DE ACTIVOS

Desde la perspectiva del CAPEX, el impacto económico se traduce en una gestión muy eficiente de los activos físicos. Los transformadores, celdas de media tensión y sistemas de almacenamiento de energía BESS representan la inversión más costosa de una microrred, un transformador de distribución de 400 kVA se aproxima a los 20.000 € y un sistema BESS de 100 kWh supera los 45.000 €. Sin embargo, la inteligencia artificial actúa como un conservador de activos altamente especializado, evitando que estos operen en regímenes de fatiga térmica o estrés innecesario.

Al gestionar las cargas y descargas de forma inteligente, se estima que la vida útil de las baterías de litio puede extenderse exponencialmente, mejorando el retorno de inversión de los activos existentes. Durante la simulación del escenario de picos de demanda por calefacción, el control automatizado logró reducir la saturación térmica del transformador y evitó descargas profundas de las baterías. A nivel industrial, evitar estos ciclos de estrés prolonga la vida útil de los equipos desde los 10 años, como límite estándar, hasta aproximadamente los 13 años de uso ininterrumpido. Retrasar la sustitución de un transformador de cabecera varios años supone un ahorro financiero destacable. Esto permite que la empresa destine su capital a la expansión de la red o a nuevos proyectos de innovación, en lugar de verse obligada a una reposición prematura de equipos por una gestión ineficiente.

6.5.3 MITIGACIÓN DE RIESGOS Y AHORRO DE MANTENIMIENTO CORRECTIVO

La prevención es siempre más económica que la reparación, y en infraestructuras cruciales, un fallo puede acarrear costes indirectos enormes. El impacto de una parada no programada o un fallo catastrófico incluye no solo el coste de los materiales y la mano de obra, sino también posibles sanciones regulatorias por incumplimiento del índice TIEPI o Tiempo de Interrupción Equivalente a la Potencia Instalada y daños reputacionales. Tomando como referencia el escenario simulado del apagón del 28 de abril de 2025, la capacidad de la microrred para operar en "modo isla" de forma autónoma evita pérdidas cuantificables mediante el valor de la Energía No Suministrada o ENS, que en el sector industrial español se penaliza con costes de lucro cesante que pueden superar los 15 €/kWh interrumpido. La capacidad de detectar anomalías sutiles mediante el gemelo digital permite programar intervenciones pequeñas antes de que ocurra una avería grave, lo que reduce drásticamente los costes logísticos y evita las costosas horas de inactividad de los clientes industriales.

Además, las empresas que demuestran el uso de tecnologías de monitorización avanzada están en una posición de fuerza para negociar primas de seguros industriales más competitivas. Al tener el riesgo técnico supervisado y mitigado por un sistema de IA robusto, las aseguradoras perciben un menor índice de siniestralidad. Este ahorro en las pólizas de seguros, sumado a la reducción de desplazamientos de personal de emergencia, se traduce en una estructura de mantenimiento preventivo mucho más rentable y previsible.

6.5.4 NUEVAS VÍAS DE INGRESOS Y SERVICIOS DE FLEXIBILIDAD

Para concluir, la arquitectura propuesta convierte a la microrred en un activo financiero dinámico capaz de generar ingresos por sí mismo. Una empresa que controla su energía con este nivel de precisión puede participar en mercados que antes no era capaz de explotar. La organización puede ofrecer servicios como la regulación de frecuencia o respuesta a la demanda a la red principal, transformando su capacidad de almacenamiento en una fuente de ingresos directos. En el mercado eléctrico actual, Red Eléctrica de España retribuye económicamente a las instalaciones no solo por la energía que inyectan, sino por estar preparadas para actuar. Participar en servicios de flexibilidad como la reserva secundaria permite a la microrred ingresar una media de 30 a 50 € por cada megavatio hora, simplemente por garantizar su disponibilidad inmediata para ayudar a estabilizar la red ante imprevistos.

Esta cualidad permite que la microrred se comporte como una central eléctrica virtual, capaz de negociar su excedente energético en el mercado en los momentos de mayor precio. Escalando el modelo del barrio simulado a una agrupación capaz de ofertar 500 kW de flexibilidad, la instalación podría generar ingresos de manera extraordinaria estimados en más de 12.000 € anuales. Esta capacidad de respuesta al mercado maximiza la rentabilidad de cada activo conectado. En conclusión, el impacto económico beneficioso no es solo un ahorro de costes, sino la creación de una infraestructura eléctrica que se paga a sí misma a través de la eficiencia y la apertura de modelos de negocio energéticos innovadores.

Capítulo 7. PLIEGO DE CONDICIONES

7.1 ESPECIFICACIONES TÉCNICAS DEL SOFTWARE Y HARDWARE

Este capítulo establece el conjunto de normas, exigencias y especificaciones técnicas que deben regir la implementación, puesta en marcha y explotación del gemelo digital. Las presentes especificaciones definen los requisitos mínimos y las características técnicas que deben cumplir tanto los activos físicos como las herramientas lógicas que integran la arquitectura del proyecto.

7.1.1 ESPECIFICACIONES DE HARDWARE

1. Servidor de Inferencia Edge Industrial

    - Procesador CPU: Arquitectura x86-64 con un mínimo de 12 núcleos físicos y frecuencia base de 2.4 GHz. Deberá soportar conjuntos de instrucciones AVX2 o superiores para optimizar las operaciones de tensores.

    - Memoria RAM: Mínimo de 64 GB DDR4/DDR5 con corrección de errores para garantizar la estabilidad del modelo LLM en ejecución continua.

    - Aceleración de IA GPU: El equipo contará con una unidad de procesamiento GPU dedicada con capacidad mínima de 40 TOPS o *Tera Operations Per Second* y al menos 12 GB de VRAM dedicada para la carga del modelo Llama 3.2.

    - Almacenamiento: Unidad de estado sólido NVMe Gen4 de al menos 512 GB para el sistema operativo y el repositorio de modelos.

    - Condiciones Ambientales: Grado de protección IP50 o superior, con rango operativo de temperatura entre -20°C y +60°C para instalación en centros de transformación.

2. Nodo de Simulación y Virtualización

    - El equipo destinado a la ejecución del motor OpenDSS y el entorno de Python deberá contar con una CPU de 8 núcleos y 16 GB de RAM como mínimo.

    - Se requiere soporte de virtualización por hardware Intel VT-x o AMD-V para la ejecución de las máquinas virtuales y contenedores.

7.1.2 ESPECIFICACIONES DE SOFTWARE Y ENTORNOS DE EJECUCIÓN

El ecosistema lógico se basará estrictamente en soluciones que permitan la interoperabilidad y el control total sobre la lógica de negocio.

1. Sistema Operativo y Capa de Virtualización

    - Se utilizará una distribución de Linux basada en Ubuntu Server 22.04 LTS o superior por su estabilidad y compatibilidad con el motor de contenedores.

    - El entorno de virtualización se gestionará mediante VMware Workstation Pro o KVM, asegurando el aislamiento de recursos entre la simulación y la orquestación.

2. Motor de Simulación Eléctrica

    - Deberá emplearse la versión estable más reciente de OpenDSS.

    - La integración con la capa de control se realizará exclusivamente a través de la la librería dss_python, garantizando la captura de datos en régimen estacionario.

3. Entorno de Orquestación y IA

    - n8n: Se desplegará en modalidad *self-hosted* mediante *Docker Compose*. La instancia deberá configurarse para manejar procesos asíncronos y colas de ejecución para evitar la pérdida de telemetría durante los tiempos de inferencia.

    - Ollama: Actuará como servidor de inferencia local. El modelo cargado será obligatoriamente Llama 3.2 3B o superior, configurado con una temperatura de generación de 0.3 para priorizar el determinismo técnico sobre la creatividad lingüística.

    - Python: Versión 3.10 o superior, incluyendo las librerías pandas para gestión de datos, numpy para cálculos matemáticos y requests para la comunicación con la API de n8n.

7.1.3 PROTOCOLOS DE COMUNICACIÓN E INTERCONECTIVIDAD

- Bus de Datos Interno: La comunicación entre contenedores y scripts se realizará mediante el protocolo HTTP/REST, empleando formato JSON para el intercambio de información.

- Seguridad de Red: No se permitirá el acceso desde redes externas al puerto de la API de Ollama ni al motor de base de datos de n8n. Todas las comunicaciones internas deberán realizarse a través de una red virtual privada *Docker Network* con aislamiento de capa 2.

- Frecuencia de Muestreo: El sistema deberá ser capaz de procesar y responder a un ciclo completo de simulación-inferencia-acción en un tiempo máximo de 5 segundos, considerando el retardo de inferencia del LLM.

7.2 NORMATIVAS Y ESTÁNDARES APLICABLES

El marco normativo garantiza que la integración de la inteligencia artificial y el control distribuido se realice bajo estrictos criterios de seguridad, interoperabilidad y calidad de suministro. Dado que nos encontramos en un entorno de red crítica, el cumplimiento de los estándares internacionales es el único mecanismo para asegurar que el modelo desarrollado sea exportable a entornos industriales reales y cumpla con las exigencias regulatorias.

7.2.1 ESTÁNDARES DE INTERCONEXIÓN Y OPERACIÓN DE MICROREDES

- IEEE 1547-2018 *Standard for Interconnection and Interoperability of Distributed Energy Resources:* Esta norma es el pilar para la integración de los DER. Define los requisitos de respuesta ante variaciones de tensión y frecuencia, estableciendo las curvas de comportamiento que el agente de IA debe supervisar para evitar desconexiones innecesarias.

- IEEE 2030.7-2017 *Standard for the Specification of Microgrid Controllers*: Proporciona las directrices funcionales para el controlador de la microrred. Este estándar es fundamental para la lógica del orquestador n8n, ya que define los estados operativos y las transiciones que la inteligencia artificial debe gestionar de forma segura.

- IEC 61850 *Communication Networks and Systems for Power Utility Automation*: Aunque nació para subestaciones, su aplicación en microrredes es vital para el modelado de datos. Asegura que la telemetría capturada por Python y procesada por el LLM siga una nomenclatura universal, facilitando la comunicación entre equipos de diferentes fabricantes.

7.2.2 ESTÁNDARES DE GEMELOS DGITALES Y MODELADO DE DATOS

- ISO/IEC 30173 *Digital Twin - Concepts and Terminology*: Este estándar internacional establece el vocabulario y la arquitectura de referencia. En este proyecto, se utiliza para validar la bidireccionalidad de los datos entre el entorno virtual y el centro de decisión cognitiva.

- IEC 61970 *Common Information Model - CIM*: Esencial para la interoperabilidad del software de gestión energética. El CIM permite que los datos de la red de 23 nodos sean estructurados de manera que otros sistemas de gestión EMS o SCADA puedan interpretar las consignas de optimización generadas por el modelo Llama 3.2.

7.2.3 ESTÁNDARES DE CIBERSEGURIDAD Y GOBERNANZA DE LA IA

- IEC 62351 *Power systems management and associated information exchange - Data and communications security*: Este estándar aborda la seguridad en los protocolos de comunicación eléctrica. Es la referencia para implementar el cifrado y la autenticación en los flujos de datos que conectan la simulación con el motor de inferencia local.

- ISO/IEC 42001 *Artificial Intelligence Management System*: Siendo una norma reciente y determinante, proporciona el marco para el despliegue responsable de la IA. Asegura que el uso de *Edge IA* cumpla con requisitos de transparencia, aplicabilidad y gestión de riesgos, evitando que la "caja negra" del modelo LLM comprometa la seguridad física de la red.

7.3 PROTOCOLO DE MANTENIMIENTO DEL GEMELO DIGITAL

7.3.1 MANTENIMIENTO PREVENTIVO DEL MODELO ELÉCTRICO Y SIMULACIÓN

La fiabilidad de las decisiones tomadas por la inteligencia artificial depende estrictamente de la precisión de los datos suministrados por el motor de cálculo. Por ello, se establecen las siguientes acciones obligatorias:

- Auditoría de sincronización de parámetros: trimestralmente, se deberá verificar que las matrices de impedancia y los perfiles de carga configurados en *OpenDSS* coincidan con las mediciones físicas de la red real. Este proceso mitiga la deriva del modelo, asegurando que el flujo de potencia calculado refleje la realidad operativa del sistema supervisado.

- Validación de scripts de control: se realizará una revisión mensual de los scripts de Python que gestionan la interacción con OpenDSS para detectar posibles degradaciones en los tiempos de ejecución o errores de desbordamiento en la captura de telemetría.

7.3.2 GESTIÓN Y ACTUALIZACIÓN DEL AGENTE DE IA

- Optimización del motor de inferencia: se deben revisar mensualmente las actualizaciones del motor Ollama para asegurar que el despliegue de Llama 3.2 aproveche las últimas mejoras en aceleración por hardware.

- Refinamiento de *prompts* y contexto: ante cambios significativos en la topología de la red o en las normativas de operación, el *System Prompt* del agente debe ser actualizado y validado mediante pruebas para evitar que la IA genere consignas de control obsoletas o técnicamente incorrectas.

- Monitoreo de la temperatura de generación: se supervisará el parámetro de temperatura del modelo para garantizar que se mantiene en niveles deterministas 0.3 o inferior, evitando comportamientos erráticos, alucinaciones y inercia en las órdenes de gestión de baterías.

7.3.3 SUPERVISIÓN DE LA INFRAESTRUCTURA DE ORQUESTACIÓN N8N

- Gestión de logs y almacenamiento: debido al alto volumen de datos procesados en cada ciclo de simulación, se implementará una política de rotación de logs automática cada 15 días. Esto previene el agotamiento del espacio en disco del servidor de borde, lo cual podría provocar la caída del flujo de trabajo y la pérdida de control del gemelo digital.

- Verificación de API: semanalmente, el sistema ejecutará una prueba de conectividad automática entre n8n, el motor de IA y el simulador para confirmar que los tiempos de latencia se mantienen dentro del margen operativo de 2.5 segundos.

Capítulo 8. PRESUPUESTO

8.1 COMPUTO DE HORAS DE INGENIERÍA

Este análisis económico se fundamenta en una valoración de los activos humanos y técnicos necesarios para transformar el modelo teórico en uno operativo y funcional. El coste más significativo en el desarrollo de este sistema aplicado a redes eléctricas es el tiempo dedicado por el personal cualificado, se ha estimado una duración total del proyecto de 6 meses, con una dedicación media que suma un total de 720 horas de trabajo técnico.

8.1.1 PERFILES PROFESIONALES

La complejidad de integrar *OpenDSS*, *Python*, n8n y modelos de lenguaje requiere de una estructura de equipo multidisciplinar. Se han definido tres perfiles profesionales tentativos para abordar el proyecto de la manera más eficiente posible, con costes horarios basados en los estándares del sector tecnológico e industrial para el ejercicio actual:

- Ingeniero junior: encargado de la recopilación de datos, revisión del estado del arte y soporte en la documentación técnica.

- Ingeniero senior de sistemas de potencia: responsable del modelado virtual de la red a supervisar, validación de flujos de potencia y diseño de estrategias de control de almacenamiento BMS.

- Ingeniero senior de software e IA: Especialista en la configuración de contenedores *Docker* y entornos *Linux*, orquestación de flujos en n8n e integración del motor de inferencia local.

8.1.2 DESGLOSE DE HORAS POR FASE DE PROYECTO

Fase del Proyecto	Descripción de Tareas Principales	Perfil	Tamaño Óptimo del Equipo	Horas	Coste/h aprox.(€)	Subtotal (€)
I. Investigación	Análisis del estado del arte y definición de límites del estudio.	Jr.	4	120	15	7.200
II. Modelado Eléctrico	Configuración de OpenDSS, integración con Python y validación de red.	Sr.	3	200	20	12.000
III. Implementación Inteligencia Artificial	Implementación de LLM, n8n y gestión de la asincronía.	Sr.	3	250	24	18.000
IV. Validación y Análisis	Simulación de escenarios de estrés y redacción de conclusiones.	Sr.	4	150	20	12.000
TOTAL				**720**		**49.200**

Tabla 1: Inversión en horas de ingeniería.

La inversión total en ingeniería asciende a 49.200€ estimado en función del número de ingenieros óptimo para abordar cada fase del proyecto. Este importe se justifica por la alta especialización técnica necesaria para evitar fallos en la capa de control de la infraestructura y por tanto retrasos y pérdidas económicas.

En comparación a proyectos de software convencionales, la ingeniería de gemelos digitales es una tecnología pionera la cual exige un proceso cruzado de revisión constante entre los resultados de la simulación física y el razonamiento lógico del modelo de IA, lo que explica la carga de horas en las fases de implementación III y validación IV. Esta dedicación asegura que el sistema no solo sea funcional, sino que también técnicamente seguro, según los estándares IEEE e IEC expuestos anteriormente.

8.2 COSTES DE INFRAESTRUCTURA Y LICENCIAS

8.2.1 INFRAESTRUCTURA DE HARDWARE Y SIMULACIÓN

La necesidad de ejecutar un proceso de inferencia local con baja latencia exige un hardware mínimo que supere las capacidades de un ordenador de oficina convencional, pero sin necesidad de igualar los recursos de procesamiento de las grandes máquinas de computación. Para la implementación del gemelo digital, es adecuado invertir en un servidor *Edge* industrial con aceleración de IA nativa. Este equipo está diseñado para soportar las condiciones ambientales de un centro de transformación mientras procesa los flujos de datos en tiempo real. De manera complementaria, se requiere de una estación de trabajo para la simulación, destinada a contener el entorno virtualizado de *OpenDSS* y *Python*.

Elemento	Especificación Clave	Unidades	Coste Unitario (€)	Total (€)
Servidor Edge Industrial Lenovo ThinkStation	NPU 40 TOPS / 64GB RAM / IP50	1	14.946,01	14.946,01
Workstation de Simulación DELL	CPU 12-core / 32GB RAM / NVMe 1TB	1	2.697,82	2.697,82
Infraestructura de Red Local	Conmutador Gigabit / Cableado RJ45 CAT7 / Rack	1	1700	1700
TOTAL HARDWARE				**19.343,83**

Tabla 2: Inversión en hardware

8.2.2 LICENCIAS DE SOFTWARE Y ACTIVOS LÓGICOS

Software	Función	Tipo de Licencia	Coste (€)
OpenDSS	Simulador de Redes Eléctricas	Código Abierto	0
Llama 3.2 o superior	Motor de IA	Código Abierto	0
n8n / Docker	Orquestación y Contenerización	Código Justo	0
VMware Workstation Pro	Capa de Virtualización	Comercial	250
Ubuntu Server / Python	Sistema Operativo y Control	Código Abierto	0
TOTAL SOFTWARE			**250**

Tabla 3: Inversión en software.

Al prescindir de modelos como GPT-4 o plataformas *Cloud* de gestión energética, se evita un gasto en suscripciones y consumo de tokens que podría superar aproximadamente los 1.800 € mensuales.

El coste total de la infraestructura se estima en 68.793,83 €, representando un equilibrio adecuado entre capacidad operativa y viabilidad económica. Por un lado, otorga al sistema de equipos suficientemente robustos capaces de absorber la carga computacional y la demanda de la red eléctrica. Por el otro, optimiza el gasto en herramientas de software que no aportan valor directo a la red física y ya que la mayoría del importe corresponde a una única inversión, el proyecto tiene una gran amortización a lo largo de su vida útil.

8.3 ANÁLISIS DE RENTABILIDAD Y RETORNO DE INVERSIÓN

8.3.1 COMPARATIVA DE MODELOS DE COSTE

Concepto	Modelo Tradicional (Cloud)	Modelo Propuesto (Edge IA)
CAPEX	Bajo (Suscripción mensual)	Alto (Hardware e ingeniería)
OPEX	Elevados (Tokens IA, Hosting, Datos)	Muy bajos (Consumo eléctrico y mantenimiento)
Escalabilidad	Pago por uso (Coste incremental)	Coste marginal cero
Dependencia	Alta (Sujeto a cambios de tarifas)	Nula (Soberanía de datos)

Tabla 4: Comparativo de modelos de coste.

8.3.2 ANÁLISIS DE VIABILIDAD: *PAYBACK* Y AHORRO OPERATIVO

Para conceptualizar la rentabilidad frente a los servicios e infraestructuras convencionales, estimamos el ahorro operativo mensual que supone no utilizar APIs de modelos comerciales como GPT-4o, ni plataformas de simulación en la nube y el tiempo de amortización en años.

- Coste *Cloud* estimado: inferencia constante de datos para una red de 23 nodos = 1.800 €/mes.
- Coste *Edge*:
  - Energía = $0{,}35kW \cdot 24h \cdot 30\ días = 180\ kWh \approx 30\ €/mes$
  - Mantenimiento $\approx 250\ €/mes$
- Ahorro Operativo Mensual S = 1.800 €/mes
  - $S_{Total} = (1800 - 250)\ x\ 12\ meses = 18.600\ €/año$

Considerando una inversión total I de 68.793,83 € el periodo de recuperación de la inversión se calcula mediante la fórmula:

$$P = \frac{I}{S_{Total}} = \frac{68.793{,}83}{18.600} \approx 3{,}7\ años$$

En poco menos de 3,7 años, el sistema se ha amortizado completamente solo por el ahorro en servicios externos al mes, sin contar los beneficios económicos indirectos derivados de la eficiencia energética y la prolongación de la vida útil de los activos de la red.

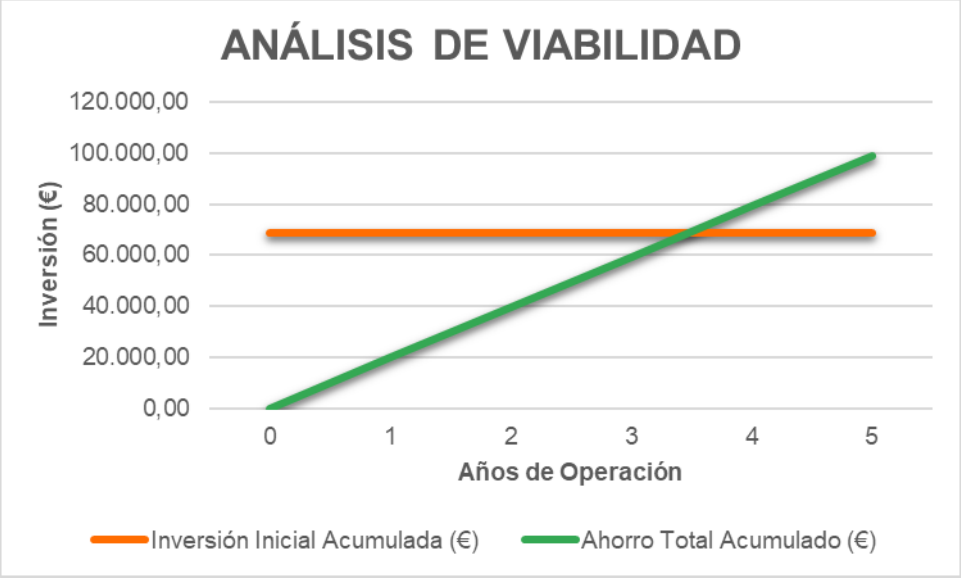


Ilustración 12: Gráfico de periodo de recuperación de la inversión.

8.3.3 RETORNO DE INVERSIÓN

Para evaluar la rentabilidad global del sistema a largo plazo, se toma como referencia el fin del ciclo de vida tecnológico del equipo, estimado aproximadamente en 5 años antes de volverse obsoleto. Se emplea la métrica del retorno de inversión o ROI, esta fórmula relaciona directamente el beneficio neto total que genera el gemelo digital con el esfuerzo económico de la inversión inicial.

$$Ahorro\ Acumulado\ (5\ años) = 18.600 \frac{€}{año}\ x\ 5\ años = 93.000€$$

$$Retorno\ de\ Inversión = \left(\frac{93.000\ € - 68.793{,}83\ €}{68.793{,}83\ €}\right) x\ 100 = 35{,}2\ \%$$

Un ROI del 35,2% en 5 años demuestra una viabilidad financiera muy interesante, confirmando que la implementación del procesamiento lógico en local constituye una ventaja económica que puede ser muy influyente en el mercado actual. Como resultado, el gemelo digital se autofinancia fácilmente a través de los ahorros generados.

Año	Inversión Inicial Acumulada (€)	Ahorro Anual Generado (€)	Ahorro Total Acumulado (€)	Flujo de Caja Neto (€)
0	68.793,83	0	0	-68.793,83
1	68.793,83	19.800,00	19.800,00	-48.993,83
2	68.793,83	19.800,00	39.600,00	-29.193,83
3	68.793,83	19.800,00	59.400,00	-9.393,83
4	68.793,83	19.800,00	79.200,00	10.406,17
5	68.793,83	19.800,00	99.000,00	30.206,17

Tabla 5: Retorno de la inversión inicial.

8.3.4 PRESUPUESTO GENERAL

CÓDIGO	DESCRIPCIÓN DE LA PARTIDA	UNIDAD	MEDICIÓN	PRECIO UNITARIO (€)	IMPORTE
1	**CAPÍTULO 1: INGENIERÍA, DESARROLLO DE SOFTWARE E IA**				**49.200,00 €**
1.1	Fase de investigación técnica y definición de límites del gemelo digital	h	480	15,00 €	7.200,00 €
1.2	Modelado eléctrico en OpenDSS e integración con Python	h	600	20,00 €	12.000,00 €
1.3	Implementación de infraestructura Edge IA y flujos n8n asíncronos	h	750	24,00 €	18.000,00 €
1.4	Validación de escenarios operativos, pruebas de estrés térmico y dirección en campo	h	600	20,00 €	12.000,00 €
2	**CAPÍTULO 2: INFRAESTRUCTURA FÍSICA Y HARDWARE EDGE**				**19.343,83 €**
2.1	Servidor Edge Industrial Lenovo ThinkStation (NPU 40 TOPS, 64GB RAM, IP50)	ud	1	14.946,01 €	14.946,01 €
2.2	Workstation de Simulación DELL (CPU 12-core, 32GB RAM, NVMe 1TB)	ud	1	2.697,82 €	2.697,82 €
2.3	Conmutador Gigabit, Armario Rack industrial y cableado estructurado RJ45 CAT7	ud	1	1.700,00 €	1.700,00 €
3	**CAPÍTULO 3: LICENCIAS DE SOFTWARE Y ENTORNOS DE EJECUCIÓN**				**250,00 €**
3.1	Licencia comercial VMware Workstation Pro (Virtualización del hipervisor)	ud	1	250,00 €	250,00 €
3.2	Motores OpenDSS, Ollama y n8n (Licencias Open Source y Código Justo)	ud	1	0,00 €	- €
				PRESUPUESTO DE EJECUCIÓN MATERIAL (PEM)	**68.793,83 €**
				Gastos Generales (13%)	8.943,20 €
				Beneficio Industrial (6%)	4.127,63 €
				BASE IMPONIBLE (Sin IVA)	**81.864,66 €**
				I.V.A. (21%)	17.191,58 €
				PRESUPUESTO DE EJECUCIÓN POR CONTRATA (PEC)	**99.056,24 €**

Capítulo 9. ANEXOS

9.1 CÓDIGOS DE OPENDSS

9.1.1 BusCoords.dss

```
Edit "Vsource.source" bus1=sourcebus basekv=11 pu=1.05 Isc3=3000 Isc1=5
```

9.1.2 BusVoltageBases.dss

```
Set Voltagebases=(11, 0.416, )

CalcVoltagebases
```

9.1.3 EnergyMeter.dss

```
New "EnergyMeter.m1" element=LINE.LINE1 terminal=1
```

9.1.4 GrowthShape

```
New "GrowthShape.default" npts=2 year=(1, 20, ) mult=(1.025, 1.025, )
```

9.1.5 LineCode.dss

```
New "LineCode.2c_.007" nphases=3 r1=3.97 x1=0.099 r0=3.97 x0=0.099 C1=0 C0=0 units=km

New "LineCode.2c_.0225" nphases=3 r1=1.257 x1=0.085 r0=1.257 x0=0.085 C1=0 C0=0 units=km

New "LineCode.2c_16" nphases=3 r1=1.15 x1=0.088 r0=1.2 x0=0.088 C1=0 C0=0 units=km

New "LineCode.35_sac_xsc" nphases=3 r1=0.868 x1=0.092 r0=0.76 x0=0.092 C1=0 C0=0 units=km

New "LineCode.4c_.06" nphases=3 r1=0.469 x1=0.075 r0=1.581 x0=0.091 C1=0 C0=0 units=km

New "LineCode.4c_.1" nphases=3 r1=0.274 x1=0.073 r0=0.959 x0=0.079 C1=0 C0=0 units=km

New "LineCode.4c_.35" nphases=3 r1=0.089 x1=0.0675 r0=0.319 x0=0.076 C1=0 C0=0 units=km

New "LineCode.4c_185" nphases=3 r1=0.166 x1=0.068 r0=0.58 x0=0.078 C1=0 C0=0 units=km

New "LineCode.4c_70" nphases=3 r1=0.446 x1=0.071 r0=1.505 x0=0.083 C1=0 C0=0 units=km

New "LineCode.4c_95_sac_xc" nphases=3 r1=0.322 x1=0.074 r0=0.804 x0=0.093 C1=0 C0=0
units=km
```

9.1.6 Renovables.dss

```
New Loadshape.CurvaSolar npts=24 interval=1 mult=[0 0 0 0 0 0 0.1 0.3 0.6 0.9 1.0 1.0 0.9 0.6 0.3 0.1 0 0 0 0 0 0 0 0]

New Loadshape.CurvaV2G npts=24 interval=1 mult=[1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 -1 -1 -1 -1 -1 0]

New PVSystem.PV1 Bus1=34.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV2 Bus1=47.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV3 Bus1=70.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV4 Bus1=73.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV5 Bus1=85.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV6 Bus1=83.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV7 Bus1=178.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV8 Bus1=208.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV9 Bus1=225.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV10 Bus1=248.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV11 Bus1=249.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV12 Bus1=264.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV13 Bus1=276.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV14 Bus1=289.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV15 Bus1=314.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV16 Bus1=320.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV17 Bus1=327.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV18 Bus1=337.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV19 Bus1=342.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV20 Bus1=349.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV21 Bus1=387.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar
```

```
New PVSystem.PV22 Bus1=388.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV23 Bus1=406.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV24 Bus1=458.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV25 Bus1=502.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV26 Bus1=522.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV27 Bus1=539.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV28 Bus1=556.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV29 Bus1=562.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV30 Bus1=563.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV31 Bus1=611.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV32 Bus1=614.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV33 Bus1=619.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV34 Bus1=629.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV35 Bus1=639.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV36 Bus1=676.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV37 Bus1=682.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV38 Bus1=688.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV39 Bus1=701.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV40 Bus1=702.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV41 Bus1=755.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV42 Bus1=778.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV43 Bus1=780.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV44 Bus1=785.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV45 Bus1=813.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV46 Bus1=817.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV47 Bus1=835.3 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar
```

```
New PVSystem.PV48 Bus1=860.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV49 Bus1=861.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV50 Bus1=886.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV51 Bus1=896.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV52 Bus1=898.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV53 Bus1=899.2 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV54 Bus1=900.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar

New PVSystem.PV55 Bus1=906.1 phases=1 kV=0.23 kVA=5 Pmpp=5 daily=CurvaSolar



New Storage.EV1 Bus1=34.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV2 Bus1=47.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV3 Bus1=70.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV4 Bus1=73.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV5 Bus1=85.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV6 Bus1=83.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV7 Bus1=178.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV8 Bus1=208.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV9 Bus1=225.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV10 Bus1=248.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV11 Bus1=249.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G
```

```
New Storage.EV12 Bus1=264.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV13 Bus1=276.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV14 Bus1=289.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV15 Bus1=314.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV16 Bus1=320.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV17 Bus1=327.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV18 Bus1=337.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV19 Bus1=342.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV20 Bus1=349.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV21 Bus1=387.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV22 Bus1=388.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV23 Bus1=406.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV24 Bus1=458.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV25 Bus1=502.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV26 Bus1=522.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV27 Bus1=539.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G
```

```
New Storage.EV28 Bus1=556.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV29 Bus1=562.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV30 Bus1=563.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV31 Bus1=611.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV32 Bus1=614.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV33 Bus1=619.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV34 Bus1=629.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV35 Bus1=639.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV36 Bus1=676.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV37 Bus1=682.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV38 Bus1=688.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV39 Bus1=701.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV40 Bus1=702.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV41 Bus1=755.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV42 Bus1=778.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV43 Bus1=780.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G
```

```
New Storage.EV44 Bus1=785.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV45 Bus1=813.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV46 Bus1=817.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV47 Bus1=835.3 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV48 Bus1=860.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV49 Bus1=861.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV50 Bus1=886.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV51 Bus1=896.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV52 Bus1=898.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV53 Bus1=899.2 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV54 Bus1=900.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G

New Storage.EV55 Bus1=906.1 phases=1 kV=0.23 kWrated=7.4 kWhrated=50 state=IDLE
daily=CurvaV2G
```

9.1.7 Monitor.dss

```
New "Monitor.line558_vi_vs_time" element=line.line558 terminal=2 mode=0

New "Monitor.line825_vi_vs_time" element=line.line825 terminal=2 mode=0
```

9.1.8 Spectrum.dss

New "Spectrum.default" NumHarm=7 harmonic=(1, 3, 5, 7, 9, 11, 13,) %mag=(100, 33, 20, 14, 11, 9, 7,) angle=(0, 0, 0, 0, 0, 0, 0,)

New "Spectrum.defaultload" NumHarm=7 harmonic=(1, 3, 5, 7, 9, 11, 13,) %mag=(100, 1.5, 20, 14, 1, 9, 7,) angle=(0, 180, 180, 180, 180, 180, 180,)

New "Spectrum.defaultgen" NumHarm=7 harmonic=(1, 3, 5, 7, 9, 11, 13,) %mag=(100, 5, 3, 1.5, 1, 0.7, 0.5,) angle=(0, 0, 0, 0, 0, 0, 0,)

New "Spectrum.defaultvsource" NumHarm=1 harmonic=(1,) %mag=(100,) angle=(0,)

New "Spectrum.linear" NumHarm=1 harmonic=(1,) %mag=(100,) angle=(0,)

New "Spectrum.pwm6" NumHarm=13 harmonic=(1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25,) %mag=(100, 4.4, 76.5, 62.7, 2.9, 24.8, 12.7, 0.5, 7.1, 8.4, 0.9, 4.4, 3.3,) angle=(-103, -5, 28, -180, -33, -59, 79, 36, -253, -124, 3, -30, 86,)

New "Spectrum.dc6" NumHarm=10 harmonic=(1, 3, 5, 7, 9, 11, 13, 15, 17, 19,) %mag=(100, 1.2, 33.6, 1.6, 0.4, 8.7, 1.2, 0.3, 4.5, 1.3,) angle=(-75, 28, 156, 29, -91, 49, 54, 148, -57, -46,)

9.1.9 TCC_Curve.dss

```
New "Spectrum.default" NumHarm=7 harmonic=(1, 3, 5, 7, 9, 11, 13, ) %mag=(100, 33, 20, 14, 11, 9, 7, ) angle=(0, 0, 0, 0, 0, 0, 0, )

New "Spectrum.defaultload" NumHarm=7 harmonic=(1, 3, 5, 7, 9, 11, 13, ) %mag=(100, 1.5, 20, 14, 1, 9, 7, ) angle=(0, 180, 180, 180, 180, 180, 180, )

New "Spectrum.defaultgen" NumHarm=7 harmonic=(1, 3, 5, 7, 9, 11, 13, ) %mag=(100, 5, 3, 1.5, 1, 0.7, 0.5, ) angle=(0, 0, 0, 0, 0, 0, 0, )

New "Spectrum.defaultvsource" NumHarm=1 harmonic=(1, ) %mag=(100, ) angle=(0, )

New "Spectrum.linear" NumHarm=1 harmonic=(1, ) %mag=(100, ) angle=(0, )

New "Spectrum.pwm6" NumHarm=13 harmonic=(1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, ) %mag=(100, 4.4, 76.5, 62.7, 2.9, 24.8, 12.7, 0.5, 7.1, 8.4, 0.9, 4.4, 3.3, ) angle=(-103, -5, 28, -180, -33, -59, 79, 36, -253, -124, 3, -30, 86, )

New "Spectrum.dc6" NumHarm=10 harmonic=(1, 3, 5, 7, 9, 11, 13, 15, 17, 19, ) %mag=(100, 1.2, 33.6, 1.6, 0.4, 8.7, 1.2, 0.3, 4.5, 1.3, ) angle=(-75, 28, 156, 29, -91, 49, 54, 148, -57, -46, )
```

9.1.10 Transformer.dss

```
New "Transformer.tr1" buses=[sourcebus, 1, ] conns=[delta, wye, ] kVs=[11, 0.416, ] kVAs=[800,
800, ] XHL=4 sub=y
```

9.1.11 Vsource.dss

```
Edit "Vsource.source" bus1=sourcebus basekv=11 pu=1.05 Isc3=3000 Isc1=5
```

9.2 PYTHON

9.2.1 Sim_apagon_2025.py

```
import dss
import time
import requests
import csv
import random
import threading
import re
import tkinter as tk
from tkinter import ttk
from tkinter import font as tkfont
import matplotlib.pyplot as plt

RUTA_MASTER = r"C:\Users\Tronchas\OneDrive - Universidad Europea de
Madrid\Escritorio\OpenDSS_IEEE\Master.DSS"
URL_N8N = "http://10.176.196.143:5678/webhook/opendss"
ARCHIVO_CSV = "resultados_apagon.csv"

LIMITE_INFERIOR = -40.0
LIMITE_SUPERIOR = 25.0

gui_hora, gui_potencia = 1, 0.0
gui_estado_solar, gui_diagnostico = "Iniciando...", "Estable"
gui_ia_status, gui_comando, gui_razonamiento = "MONITORIZANDO...", "REPOSO",
"Control Local operando. Red segura."
comando_actual_global, razonamiento_actual_global = "reposo", "Control Local
(Estable)"
ia_ocupada = False
lock = threading.Lock()

def consultar_ia_asincrono(hora, potencia_kw, estado_solar, diagnostico,
evento_especial):
    global comando_actual_global, razonamiento_actual_global, ia_ocupada
    global gui_ia_status, gui_razonamiento

    with lock: ia_ocupada = True

    payload = {"minuto": hora, "potencia_actual_kw": round(potencia_kw, 2),
"estado_paneles_solares": estado_solar, "analisis_matematico": diagnostico,
"evento_especial": evento_especial}
    try:
        r = requests.post(URL_N8N, json=payload, timeout=120)
        if r.status_code == 200:
```

```
            res_text = r.text.strip()


            texto_limpio = res_text.lower().replace("*", "").replace("\"",
"")

            if any(palabra in texto_limpio for palabra in ["descargar",
"inyectar", "liberar", "apoyar"]):
                nuevo_comando = "descargar"
            elif any(palabra in texto_limpio for palabra in ["cargar",
"absorber", "almacenar", "guardar"]):
                nuevo_comando = "cargar"
            else:
                nuevo_comando = "reposo"

            nuevo_razonamiento = res_text
            if "RAZONAMIENTO:" in res_text: nuevo_razonamiento =
res_text.split("RAZONAMIENTO:")[1].strip()
            elif "razonamiento:" in texto_limpio: nuevo_razonamiento =
texto_limpio.split("razonamiento:")[1].strip()

            with lock:
                comando_actual_global = nuevo_comando
                razonamiento_actual_global = nuevo_razonamiento

            gui_ia_status = "☑ DECISIÓN IA CONFIRMADA"
            gui_razonamiento = nuevo_razonamiento
        else: gui_ia_status = f"❌ Error Servidor ({r.status_code})"
    except Exception as e: gui_ia_status = f"❌ ERROR DE RED"
    with lock: ia_ocupada = False

historial_horas, historial_potencia_base, historial_potencia_real = [], [],
[]

def simulacion_principal():
    global gui_hora, gui_potencia, gui_estado_solar, gui_diagnostico,
gui_comando, gui_razonamiento, gui_ia_status
    global comando_actual_global, razonamiento_actual_global

    dss_engine = dss.DSS
    dss_text = dss_engine.Text
    dss_circuit = dss_engine.ActiveCircuit
    dss_text.Command = f"Compile '{RUTA_MASTER}'"
    dss_text.Command = "Set Mode=Daily StepSize=1h Number=1"

    with open(ARCHIVO_CSV, mode='w', newline='', encoding='utf-8-sig') as
file:
```

```
        writer = csv.writer(file, delimiter=';')
        writer.writerow(["Hora", "Potencia_Base_kW", "Volatilidad",
"Potencia_Real_kW", "Estado_Solar", "Diagnostico", "IA_Comando",
"IA_Razonamiento"])

        for hora in range(1, 25):
            gui_hora = hora
            dss_text.Command = "BatchEdit Storage..* state=IDLE"
            dss_text.Command = "Solve"
            dss_circuit.SetActiveElement("Transformer.TR1")
            potencia_base_kw = dss_circuit.ActiveElement.Powers[0]

            esta_en_apagon = 14 <= hora <= 19
            factor_aleatorio = 1.0

            if 8 <= hora <= 18:
                estado_solar = "☼ SOL BRILLANDO (Paneles al 100%)"
            else:
                estado_solar = "🌙 Noche (Cero producción solar)"

            if esta_en_apagon:
                potencia_virtual = 85.5 * random.uniform(0.95, 1.05)
                potencia_kw = potencia_virtual
                potencia_base_grafica = potencia_virtual
                evento_especial = "CRÍTICO: APAGÓN GENERAL EXTERNO"
            else:
                potencia_kw = potencia_base_kw
                potencia_base_grafica = potencia_base_kw
                evento_especial = "Ninguno"

            gui_estado_solar = estado_solar
            historial_horas.append(hora)
            historial_potencia_base.append(potencia_base_grafica)

            if LIMITE_INFERIOR <= potencia_kw <= LIMITE_SUPERIOR and not
esta_en_apagon:
                diagnostico = "Estable"
                gui_diagnostico = diagnostico
                gui_ia_status = "MONITORIZANDO..."
                gui_razonamiento = "Red externa estable y potencia en
parámetros. Baterías en espera."
                with lock:
                    comando_actual_global = "reposo"
                    razonamiento_actual_global = "Control Local (Estable)"
                time.sleep(1.5)
            else:
                if esta_en_apagon: diagnostico = "COLAPSO DE RED EXTERNA"
```

```
                elif potencia_kw < LIMITE_INFERIOR: diagnostico = "Exceso de
Generación"
                else: diagnostico = "Pico de Consumo"

                gui_diagnostico = f"⚠ {diagnostico}"
                gui_comando = "⏳ ANALIZANDO RED..."
                gui_razonamiento = "Detectada anomalía crítica. Analizando
física del entorno..."

                with lock: ocupada = ia_ocupada
                if not ocupada:
                    hilo_ia =
threading.Thread(target=consultar_ia_asincrono, args=(hora, potencia_kw,
estado_solar, diagnostico, evento_especial))
                    hilo_ia.daemon = True
                    hilo_ia.start()

                    anim_dots = 0
                    while hilo_ia.is_alive():
                        dots = "." * (anim_dots % 4)
                        gui_ia_status = f"📡 CONECTANDO CON EDGE IA{dots}"
                        anim_dots += 1
                        root.update()
                        time.sleep(0.2)


            with lock:
                comando_aplicar = comando_actual_global
                razon_aplicar = razonamiento_actual_global

            gui_comando = comando_aplicar.upper()

            if comando_aplicar == "descargar":
                dss_text.Command = "BatchEdit Storage..* state=DISCHARGING"
                if esta_en_apagon:
                    potencia_kw -= 60.0
                else:
                    potencia_kw -= 30.0
            elif comando_aplicar == "cargar":
                dss_text.Command = "BatchEdit Storage..* state=CHARGING"
                potencia_kw += 30.0
            else:
                dss_text.Command = "BatchEdit Storage..* state=IDLE"

            historial_potencia_real.append(potencia_kw)
            gui_potencia = potencia_kw
```

```
            writer.writerow([f"{hora}:00", str(round(potencia_base_grafica,
2)).replace('.', ','), round(factor_aleatorio, 2), str(round(potencia_kw,
2)).replace('.', ','), estado_solar, diagnostico, comando_aplicar.upper(),
razon_aplicar])

            if not "Estable" in diagnostico:
                for _ in range(30):
                    root.update()
                    time.sleep(0.1)

    root.destroy()


root = tk.Tk()
root.title("Terminal IA - Smart Grid")
root.geometry("1100x600")
root.configure(bg="#0a0a0a")

style = ttk.Style()
style.theme_use('clam')
style.configure("TProgressbar", thickness=15, troughcolor='#111111',
background='#FF4444')

lbl_titulo = tk.Label(root, text="SISTEMA CENTRAL VPP - ESCENARIO APAGÓN",
font=("Segoe UI", 24, "bold"), bg="#0a0a0a", fg="#E2E2E2")
lbl_titulo.pack(pady=10)

main_frame = tk.Frame(root, bg="#0a0a0a")
main_frame.pack(fill="both", expand=True, padx=20)

frame_izq = tk.Frame(main_frame, bg="#0a0a0a")
frame_izq.pack(side="left", fill="both", expand=True, padx=10)

lbl_tit_sensores = tk.Label(frame_izq, text="TELEMETRÍA DE RED FÍSICA",
font=("Segoe UI", 12, "bold"), bg="#0a0a0a", fg="#FF7B72", anchor="w")
lbl_tit_sensores.pack(fill="x", pady=5)

frame_sensores = tk.Frame(frame_izq, bg="#141414", bd=1, relief="solid",
highlightbackground="#333333", highlightthickness=1)
frame_sensores.pack(fill="both", expand=True)

lbl_hora = tk.Label(frame_sensores, text="🕒 Hora: --:00", font=("Consolas",
18), bg="#141414", fg="#FFFFFF", anchor="w")
lbl_hora.pack(fill="x", padx=20, pady=15)
lbl_sol = tk.Label(frame_sensores, text="⬡ Entorno: --", font=("Consolas",
13), bg="#141414", fg="#A0A0A0", anchor="w")
lbl_sol.pack(fill="x", padx=20, pady=5)
```

```
lbl_potencia = tk.Label(frame_sensores, text="🔌 Demanda: -- kW",
font=("Consolas", 18, "bold"), bg="#141414", fg="#FFFFFF", anchor="w")
lbl_potencia.pack(fill="x", padx=20, pady=15)
lbl_diag = tk.Label(frame_sensores, text="ESTADO: --", font=("Segoe UI", 16,
"bold"), bg="#141414", fg="#00FF00", anchor="w")
lbl_diag.pack(fill="x", padx=20, pady=15)

frame_der = tk.Frame(main_frame, bg="#0a0a0a")
frame_der.pack(side="right", fill="both", expand=True, padx=10)

lbl_tit_ia = tk.Label(frame_der, text="SUPERVISOR Llama-3.2 (Edge IA)",
font=("Segoe UI", 12, "bold"), bg="#0a0a0a", fg="#58A6FF", anchor="w")
lbl_tit_ia.pack(fill="x", pady=5)

frame_ia = tk.Frame(frame_der, bg="#141414", bd=1, relief="solid",
highlightbackground="#333333", highlightthickness=1)
frame_ia.pack(fill="both", expand=True)

lbl_ia_status = tk.Label(frame_ia, text="🤖 --", font=("Consolas", 12,
"italic"), bg="#141414", fg="#8B949E", anchor="w")
lbl_ia_status.pack(fill="x", padx=20, pady=10)

lbl_comando = tk.Label(frame_ia, text="--", font=("Segoe UI", 30, "bold"),
bg="#141414", fg="#58A6FF", anchor="center")
lbl_comando.pack(fill="x", padx=20, pady=5)

frame_terminal = tk.Frame(frame_ia, bg="#000000", bd=2, relief="sunken")
frame_terminal.pack(fill="both", expand=True, padx=20, pady=15)

lbl_razonamiento = tk.Label(frame_terminal, text=">_ Esperando
telemetría...", font=("Consolas", 13), bg="#000000", fg="#00FF00",
wraplength=420, justify="left")
lbl_razonamiento.pack(fill="both", expand=True, padx=10, pady=10)

frame_progreso = tk.Frame(root, bg="#0a0a0a")
frame_progreso.pack(fill="x", padx=30, pady=15)
progress_var = tk.DoubleVar()
progress_bar = ttk.Progressbar(frame_progreso, variable=progress_var,
maximum=24)
progress_bar.pack(fill="x")

def actualizar_gui():
    lbl_hora.config(text=f"🕒 Hora Simulada: {gui_hora:02d}:00")
    lbl_sol.config(text=f"Entorno: {gui_estado_solar}")
    lbl_potencia.config(text=f"🔌 Demanda Real: {gui_potencia:.2f} kW")

    if "Estable" in gui_diagnostico:
```

```
        lbl_diag.config(text=f"ESTADO: {gui_diagnostico}", fg="#00FF00")
        root.configure(bg="#0a0a0a")
    else:
        lbl_diag.config(text=f"ESTADO: {gui_diagnostico}", fg="#FF3333")
        if "COLAPSO" in gui_diagnostico: root.configure(bg="#1c0707")

    lbl_ia_status.config(text=gui_ia_status)

    if "CARGAR" in gui_comando: color_cmd = "#D2A8FF"
    elif "DESCARGAR" in gui_comando: color_cmd = "#FF3333"
    elif "ANALIZANDO" in gui_comando: color_cmd = "#F1E05A"
    else: color_cmd = "#58A6FF"

    lbl_comando.config(text=f"[{gui_comando}]", fg=color_cmd)
    lbl_razonamiento.config(text=f">_ {gui_razonamiento}")
    progress_var.set(gui_hora)

    try: root.after(100, actualizar_gui)
    except: pass

threading.Thread(target=simulacion_principal, daemon=True).start()
actualizar_gui()
root.mainloop()


plt.figure(figsize=(12, 6))
plt.axvspan(14, 19, color='black', alpha=0.15, label='Período de Apagón
(Blackout)')


plt.plot(historial_horas, historial_potencia_base, label='Demanda Externa
(Sin IA)', linestyle='--', color='gray', linewidth=5, alpha=0.7, zorder=1)


plt.plot(historial_horas, historial_potencia_real, label='Demanda Neta
(Gestionada por IA)', color='blue', linewidth=2.5, zorder=2)

plt.axhline(y=LIMITE_SUPERIOR, color='red', linestyle='--', label='Límite
Superior (Estrés)')
plt.axhline(y=LIMITE_INFERIOR, color='green', linestyle='--', label='Límite
Inferior (Excedente)')
plt.title('Gestión de Smart Grid - Escenario: Gran Apagón (Resiliencia)')
plt.xlabel('Hora del Día (h)')
plt.ylabel('Potencia Neta Gestionada (kW)')
plt.legend()
plt.grid(True)
plt.xticks(historial_horas)
```

```
plt.tight_layout()
plt.show()
```

9.2.2 Sim_ola_frio.py

```
import dss
import time
import requests
import csv
import random
import threading
import matplotlib.pyplot as plt


RUTA_MASTER = r"C:\Users\Tronchas\OneDrive - Universidad Europea de
Madrid\Escritorio\OpenDSS_IEEE\Master.DSS"
URL_N8N = "http://192.168.1.151:5678/webhook/opendss"
ARCHIVO_CSV = "resultados_olafrio.csv"

LIMITE_INFERIOR = -40.0
LIMITE_SUPERIOR = 25.0

comando_actual_global = "reposo"
razonamiento_actual_global = "Control Local (Estable)"
ia_ocupada = False
lock = threading.Lock()

def consultar_ia_asincrono(hora, potencia_kw, estado_solar, diagnostico):
    global comando_actual_global, razonamiento_actual_global, ia_ocupada
    with lock: ia_ocupada = True
    print(f"\n📡 [HILO IA] Enviando datos de EMERGENCIA INVERNAL al Agente
Central n8n...")

    payload = {"minuto": hora, "potencia_actual_kw": round(potencia_kw, 2),
"estado_paneles_solares": estado_solar, "analisis_matematico": diagnostico}
    try:
        r = requests.post(URL_N8N, json=payload, timeout=120)
        if r.status_code == 200:
            res_text = r.text.strip()
            texto_lower = res_text.lower()

            print(f"\n[🔍 DEBUG N8N]: {res_text}")

            nuevo_comando = "reposo"

            if "comando: descargar" in texto_lower:
                nuevo_comando = "descargar"
            elif "comando: cargar" in texto_lower:
                nuevo_comando = "cargar"
```

```
            nuevo_razonamiento = res_text
            if "RAZONAMIENTO:" in res_text:
                nuevo_razonamiento =
res_text.split("RAZONAMIENTO:")[1].strip()
            elif "razonamiento:" in texto_lower:
                nuevo_razonamiento =
texto_lower.split("razonamiento:")[1].strip()

            with lock:
                comando_actual_global = nuevo_comando
                razonamiento_actual_global = nuevo_razonamiento
            print(f"🤖 [HILO IA] ¡El Supervisor ha decidido! Nueva orden:
{nuevo_comando.upper()}")
        else: print(f"\n❌ [HILO IA] Error del servidor n8n:
{r.status_code}")
    except Exception as e: print(f"\n❌ [HILO IA] Error de conexión: {e}")
    with lock: ia_ocupada = False

dss_engine = dss.DSS
dss_text = dss_engine.Text
dss_circuit = dss_engine.ActiveCircuit
dss_text.Command = f"Compile '{RUTA_MASTER}'"
dss_text.Command = "Set Mode=Daily StepSize=1h Number=1"

historial_horas, historial_potencia_base, historial_potencia_real = [], [],
[]

with open(ARCHIVO_CSV, mode='w', newline='', encoding='utf-8-sig') as file:
    writer = csv.writer(file, delimiter=';')
    writer.writerow(["Hora", "Potencia_Base_kW", "Volatilidad",
"Potencia_Real_kW", "Estado_Solar", "Diagnostico", "IA_Comando",
"IA_Razonamiento"])

    print("🚀 GEMELO DIGITAL TFG - ESCENARIO: OLA DE FRÍO (ESTRÉS MÁXIMO)")

    for hora in range(1, 25):
        dss_text.Command = "BatchEdit Storage..* state=IDLE"

        dss_text.Command = "Solve"
        dss_circuit.SetActiveElement("Transformer.TR1")
        potencia_base_kw = dss_circuit.ActiveElement.Powers[0]

        consumo_calefaccion_extra = random.uniform(20.0, 35.0)
        potencia_kw = potencia_base_kw + consumo_calefaccion_extra
        factor_aleatorio = 1.0
```

```
        historial_horas.append(hora)
        historial_potencia_base.append(potencia_base_kw +
consumo_calefaccion_extra)
        historial_potencia_real.append(potencia_kw)

        estado_solar = "Producción Solar Activa" if 9 <= hora <= 17 else
"Sin Producción (Noche de Invierno)"

        print(f"\n--- [Hora {hora}:00] Potencia REAL: {potencia_kw:.2f} kW |
Sol: {estado_solar} ---")

        if LIMITE_INFERIOR <= potencia_kw <= LIMITE_SUPERIOR:
            print("🟢 ZONA ESTABLE: El Control Local asume el mando.")
            diagnostico = "Estable"
            with lock:
                comando_actual_global = "reposo"
                razonamiento_actual_global = "Control Local (Estable)"
        else:
            diagnostico = "Exceso de Generación" if potencia_kw <
LIMITE_INFERIOR else "Pico de Consumo"
            print(f"⚠️ LÍMITE SUPERADO ({diagnostico}).")

            with lock: ocupada = ia_ocupada
            if not ocupada:
                hilo_ia = threading.Thread(target=consultar_ia_asincrono,
args=(hora, potencia_kw, estado_solar, diagnostico))
                hilo_ia.daemon = True
                hilo_ia.start()
                print("⏱️ (Esperando respuesta crítica del Supervisor...)")
                hilo_ia.join()
            else:
                print("⏳ (La IA todavía está evaluando. Manteniendo orden
actual...)")

        with lock:
            comando_aplicar = comando_actual_global
            razon_aplicar = razonamiento_actual_global

        print(f"⚙️ Aplicando a Baterías: {comando_aplicar.upper()}")

        if "descargar" in comando_aplicar:
            dss_text.Command = "BatchEdit Storage..* state=DISCHARGING"
            potencia_kw -= 35.0
            historial_potencia_real[-1] = potencia_kw
        elif "cargar" in comando_aplicar:
            dss_text.Command = "BatchEdit Storage..* state=CHARGING"
            potencia_kw += 35.0
```

```
            historial_potencia_real[-1] = potencia_kw
        else:
            dss_text.Command = "BatchEdit Storage..* state=IDLE"

        writer.writerow([f"{hora}:00", str(round(potencia_base_kw +
consumo_calefaccion_extra, 2)).replace('.', ','), round(factor_aleatorio,
2), str(round(potencia_kw, 2)).replace('.', ','), estado_solar, diagnostico,
comando_aplicar.upper(), razon_aplicar])
        time.sleep(15)

plt.figure(figsize=(12, 6))
plt.plot(historial_horas, historial_potencia_base, label='Potencia Base (Sin
IA)', linestyle='--', color='gray', alpha=0.7)
plt.plot(historial_horas, historial_potencia_real, label='Potencia Real
(Gestionada por IA)', color='blue', linewidth=2)
plt.axhline(y=LIMITE_SUPERIOR, color='red', linestyle=':', label='Límite
Superior (Pico Consumo)')
plt.axhline(y=LIMITE_INFERIOR, color='green', linestyle=':', label='Límite
Inferior (Exceso Solar)')
plt.title('Gestión de Smart Grid - Escenario: Ola de Frío Invernal (Estrés
Máximo)')
plt.xlabel('Hora del Día (h)')
plt.ylabel('Potencia en el Transformador (kW)')
plt.legend()
plt.grid(True)
plt.xticks(historial_horas)
plt.tight_layout()
plt.show()
```

9.2.3 Sim_soleado.py

```
import dss
import time
import requests
import csv
import random
import threading
import re
import tkinter as tk
from tkinter import ttk
from tkinter import font as tkfont
import matplotlib.pyplot as plt


RUTA_MASTER = r"C:\Users\Tronchas\OneDrive - Universidad Europea de
Madrid\Escritorio\OpenDSS_IEEE\Master.DSS"
URL_N8N = "http://192.168.1.151:5678/webhook/opendss"
ARCHIVO_CSV = "resultados_soleado.csv"

LIMITE_INFERIOR = -40.0
LIMITE_SUPERIOR = 25.0

gui_hora, gui_potencia = 1, 0.0
gui_estado_solar, gui_diagnostico = "Iniciando...", "Estable"
gui_ia_status, gui_comando, gui_razonamiento = "MONITORIZANDO...", "REPOSO",
"Control Local operando. Red segura."
comando_actual_global, razonamiento_actual_global = "reposo", "Control Local
(Estable)"
ia_ocupada = False
lock = threading.Lock()

def consultar_ia_asincrono(hora, potencia_kw, estado_solar, diagnostico):
    global comando_actual_global, razonamiento_actual_global, ia_ocupada
    global gui_ia_status, gui_razonamiento

    with lock: ia_ocupada = True

    payload = {"minuto": hora, "potencia_actual_kw": round(potencia_kw, 2),
"estado_paneles_solares": estado_solar, "analisis_matematico": diagnostico}
    try:
        r = requests.post(URL_N8N, json=payload, timeout=120)
        if r.status_code == 200:
            res_text = r.text.strip()
```

```
            texto_limpio = res_text.lower().replace("*", "").replace("\"",
"")

            if any(palabra in texto_limpio for palabra in ["descargar",
"inyectar", "liberar", "apoyar"]):
                nuevo_comando = "descargar"
            elif any(palabra in texto_limpio for palabra in ["cargar",
"absorber", "almacenar", "guardar"]):
                nuevo_comando = "cargar"
            else:
                nuevo_comando = "reposo"

            nuevo_razonamiento = res_text
            if "RAZONAMIENTO:" in res_text: nuevo_razonamiento =
res_text.split("RAZONAMIENTO:")[1].strip()
            elif "razonamiento:" in texto_limpio: nuevo_razonamiento =
texto_limpio.split("razonamiento:")[1].strip()

            with lock:
                comando_actual_global = nuevo_comando
                razonamiento_actual_global = nuevo_razonamiento

            gui_ia_status = "☑ DECISIÓN IA CONFIRMADA"
            gui_razonamiento = nuevo_razonamiento
        else: gui_ia_status = f"❌ Error Servidor ({r.status_code})"
    except Exception as e: gui_ia_status = f"❌ ERROR DE RED"
    with lock: ia_ocupada = False

historial_horas, historial_potencia_base, historial_potencia_real = [], [],
[]

def simulacion_principal():
    global gui_hora, gui_potencia, gui_estado_solar, gui_diagnostico,
gui_comando, gui_razonamiento, gui_ia_status
    global comando_actual_global, razonamiento_actual_global

    dss_engine = dss.DSS
    dss_text = dss_engine.Text
    dss_circuit = dss_engine.ActiveCircuit
    dss_text.Command = f"Compile '{RUTA_MASTER}'"
    dss_text.Command = "Set Mode=Daily StepSize=1h Number=1"

    with open(ARCHIVO_CSV, mode='w', newline='', encoding='utf-8-sig') as
file:
        writer = csv.writer(file, delimiter=';')
```

```
        writer.writerow(["Hora", "Potencia_Base_kW", "Volatilidad",
"Potencia_Real_kW", "Estado_Solar", "Diagnostico", "IA_Comando",
"IA_Razonamiento"])

        for hora in range(1, 25):
            gui_hora = hora
            dss_text.Command = "BatchEdit Storage..* state=IDLE"
            dss_text.Command = "Solve"
            dss_circuit.SetActiveElement("Transformer.TR1")
            potencia_base_kw = dss_circuit.ActiveElement.Powers[0]

            factor_aleatorio = random.uniform(0.9, 1.1)
            potencia_kw = potencia_base_kw * factor_aleatorio

            estado_solar = "☀️ SOL BRILLANDO (Producción al 100%)" if 7 <=
hora <= 19 else "🌙 Noche (Cero producción solar)"

            gui_estado_solar = estado_solar
            historial_horas.append(hora)
            historial_potencia_base.append(potencia_base_kw *
factor_aleatorio)

            if LIMITE_INFERIOR <= potencia_kw <= LIMITE_SUPERIOR:
                diagnostico = "Estable"
                gui_diagnostico = diagnostico
                gui_ia_status = "MONITORIZANDO..."
                gui_razonamiento = "Red externa estable y potencia en
parámetros. Baterías en espera."
                with lock:
                    comando_actual_global = "reposo"
                    razonamiento_actual_global = "Control Local (Estable)"
                time.sleep(1.5)
            else:
                diagnostico = "Exceso de Generación" if potencia_kw <
LIMITE_INFERIOR else "Pico de Consumo"
                gui_diagnostico = f"⚠️ {diagnostico}"
                gui_comando = "⏳ ANALIZANDO RED..."
                gui_razonamiento = "Detectada desviación fuera de límites.
Evaluando estado de las baterías..."

                with lock: ocupada = ia_ocupada
                if not ocupada:
                    hilo_ia =
threading.Thread(target=consultar_ia_asincrono, args=(hora, potencia_kw,
estado_solar, diagnostico))
                    hilo_ia.daemon = True
                    hilo_ia.start()
```

```
                    anim_dots = 0
                    while hilo_ia.is_alive():
                        dots = "." * (anim_dots % 4)
                        gui_ia_status = f"📡 CONECTANDO CON EDGE IA{dots}"
                        anim_dots += 1
                        root.update()
                        time.sleep(0.2)


            with lock:
                comando_aplicar = comando_actual_global
                razon_aplicar = razonamiento_actual_global

            gui_comando = comando_aplicar.upper()

            if comando_aplicar == "descargar":
                dss_text.Command = "BatchEdit Storage..* state=DISCHARGING"
                potencia_kw -= 30.0
            elif comando_aplicar == "cargar":
                dss_text.Command = "BatchEdit Storage..* state=CHARGING"
                potencia_kw += 30.0
            else:
                dss_text.Command = "BatchEdit Storage..* state=IDLE"

            historial_potencia_real.append(potencia_kw)
            gui_potencia = potencia_kw

            writer.writerow([f"{hora}:00", str(round(potencia_base_kw *
factor_aleatorio, 2)).replace('.', ','), round(factor_aleatorio, 2),
str(round(potencia_kw, 2)).replace('.', ','), estado_solar, diagnostico,
comando_aplicar.upper(), razon_aplicar])

            if not "Estable" in diagnostico:
                for _ in range(30):
                    root.update()
                    time.sleep(0.1)

    root.destroy()


root = tk.Tk()
root.title("Terminal IA - Smart Grid")
root.geometry("1100x600")
root.configure(bg="#0a0a0a")

style = ttk.Style()
```

```
style.theme_use('clam')
style.configure("TProgressbar", thickness=15, troughcolor='#111111',
background='#00FFCC')

lbl_titulo = tk.Label(root, text="SISTEMA CENTRAL VPP - DÍA SOLEADO IDEAL",
font=("Segoe UI", 24, "bold"), bg="#0a0a0a", fg="#E2E2E2")
lbl_titulo.pack(pady=10)

main_frame = tk.Frame(root, bg="#0a0a0a")
main_frame.pack(fill="both", expand=True, padx=20)

frame_izq = tk.Frame(main_frame, bg="#0a0a0a")
frame_izq.pack(side="left", fill="both", expand=True, padx=10)

lbl_tit_sensores = tk.Label(frame_izq, text="TELEMETRÍA DE RED FÍSICA",
font=("Segoe UI", 12, "bold"), bg="#0a0a0a", fg="#00FFCC", anchor="w")
lbl_tit_sensores.pack(fill="x", pady=5)

frame_sensores = tk.Frame(frame_izq, bg="#141414", bd=1, relief="solid",
highlightbackground="#333333", highlightthickness=1)
frame_sensores.pack(fill="both", expand=True)

lbl_hora = tk.Label(frame_sensores, text="🕒 Hora: --:00", font=("Consolas",
18), bg="#141414", fg="#FFFFFF", anchor="w")
lbl_hora.pack(fill="x", padx=20, pady=15)
lbl_sol = tk.Label(frame_sensores, text="☼ Entorno: --", font=("Consolas",
13), bg="#141414", fg="#A0A0A0", anchor="w")
lbl_sol.pack(fill="x", padx=20, pady=5)
lbl_potencia = tk.Label(frame_sensores, text="🔌 Demanda: -- kW",
font=("Consolas", 18, "bold"), bg="#141414", fg="#FFFFFF", anchor="w")
lbl_potencia.pack(fill="x", padx=20, pady=15)
lbl_diag = tk.Label(frame_sensores, text="ESTADO: --", font=("Segoe UI", 16,
"bold"), bg="#141414", fg="#00FF00", anchor="w")
lbl_diag.pack(fill="x", padx=20, pady=15)

frame_der = tk.Frame(main_frame, bg="#0a0a0a")
frame_der.pack(side="right", fill="both", expand=True, padx=10)

lbl_tit_ia = tk.Label(frame_der, text="SUPERVISOR Llama-3.2 (Edge)",
font=("Segoe UI", 12, "bold"), bg="#0a0a0a", fg="#58A6FF", anchor="w")
lbl_tit_ia.pack(fill="x", pady=5)

frame_ia = tk.Frame(frame_der, bg="#141414", bd=1, relief="solid",
highlightbackground="#333333", highlightthickness=1)
frame_ia.pack(fill="both", expand=True)
```

```
lbl_ia_status = tk.Label(frame_ia, text="🤖 --", font=("Consolas", 12,
"italic"), bg="#141414", fg="#8B949E", anchor="w")
lbl_ia_status.pack(fill="x", padx=20, pady=10)

lbl_comando = tk.Label(frame_ia, text="--", font=("Segoe UI", 30, "bold"),
bg="#141414", fg="#58A6FF", anchor="center")
lbl_comando.pack(fill="x", padx=20, pady=5)

frame_terminal = tk.Frame(frame_ia, bg="#000000", bd=2, relief="sunken")
frame_terminal.pack(fill="both", expand=True, padx=20, pady=15)

lbl_razonamiento = tk.Label(frame_terminal, text=">_ Esperando
telemetría...", font=("Consolas", 13), bg="#000000", fg="#00FF00",
wraplength=420, justify="left")
lbl_razonamiento.pack(fill="both", expand=True, padx=10, pady=10)

frame_progreso = tk.Frame(root, bg="#0a0a0a")
frame_progreso.pack(fill="x", padx=30, pady=15)
progress_var = tk.DoubleVar()
progress_bar = ttk.Progressbar(frame_progreso, variable=progress_var,
maximum=24)
progress_bar.pack(fill="x")

def actualizar_gui():
    lbl_hora.config(text=f"🕒 Hora Simulada: {gui_hora:02d}:00")
    lbl_sol.config(text=f"Entorno: {gui_estado_solar}")
    lbl_potencia.config(text=f"🔦 Demanda Real: {gui_potencia:.2f} kW")

    if "Estable" in gui_diagnostico:
        lbl_diag.config(text=f"ESTADO: {gui_diagnostico}", fg="#00FF00")
    else:
        lbl_diag.config(text=f"ESTADO: {gui_diagnostico}", fg="#FF9900")

    lbl_ia_status.config(text=gui_ia_status)

    if "CARGAR" in gui_comando: color_cmd = "#D2A8FF"
    elif "DESCARGAR" in gui_comando: color_cmd = "#FF3333"
    elif "ANALIZANDO" in gui_comando: color_cmd = "#F1E05A"
    else: color_cmd = "#58A6FF"

    lbl_comando.config(text=f"[{gui_comando}]", fg=color_cmd)
    lbl_razonamiento.config(text=f">_ {gui_razonamiento}")

    progress_var.set(gui_hora)

    try: root.after(100, actualizar_gui)
    except: pass
```

```
threading.Thread(target=simulacion_principal, daemon=True).start()
actualizar_gui()
root.mainloop()


plt.figure(figsize=(12, 6))


plt.plot(historial_horas, historial_potencia_base, label='Demanda Bruta (Sin
IA)', linestyle='--', color='gray', linewidth=5, alpha=0.6, zorder=1)


plt.plot(historial_horas, historial_potencia_real, label='Demanda Neta
(Gestionada por IA)', color='blue', linewidth=2.5, zorder=2)

plt.axhline(y=LIMITE_SUPERIOR, color='red', linestyle='--', label='Límite
Superior (Estrés)')
plt.axhline(y=LIMITE_INFERIOR, color='green', linestyle='--', label='Límite
Inferior (Excedente)')
plt.title('Gestión de Smart Grid - Escenario: Día Soleado Ideal')
plt.xlabel('Hora del Día (h)')
plt.ylabel('Potencia en el Transformador (kW)')
plt.legend()
plt.grid(True)
plt.xticks(historial_horas)
plt.tight_layout()
plt.show()
```

9.2.4 Sim_tormenta.py

```
import dss
import time
import requests
import csv
import random
import threading
import matplotlib.pyplot as plt

RUTA_MASTER = r"C:\Users\Tronchas\OneDrive - Universidad Europea de
Madrid\Escritorio\OpenDSS_IEEE\Master.DSS"
URL_N8N = "http://192.168.1.151:5678/webhook/opendss"
ARCHIVO_CSV = "resultados_tormenta.csv"

LIMITE_INFERIOR = -40.0
LIMITE_SUPERIOR = 25.0

comando_actual_global = "reposo"
razonamiento_actual_global = "Control Local (Estable)"
ia_ocupada = False
lock = threading.Lock()

def consultar_ia_asincrono(hora, potencia_kw, estado_solar, diagnostico):
    global comando_actual_global, razonamiento_actual_global, ia_ocupada
    with lock: ia_ocupada = True
    print(f"\n📡 [HILO IA] Enviando datos al Agente Central n8n...")

    payload = {"minuto": hora, "potencia_actual_kw": round(potencia_kw, 2),
"estado_paneles_solares": estado_solar, "analisis_matematico": diagnostico}
    try:
        r = requests.post(URL_N8N, json=payload, timeout=120)
        if r.status_code == 200:
            res_text = r.text.strip()
            texto_lower = res_text.lower()

            print(f"\n[🔍 DEBUG N8N]: {res_text}")

            nuevo_comando = "reposo"

            if "comando: descargar" in texto_lower:
                nuevo_comando = "descargar"
            elif "comando: cargar" in texto_lower:
                nuevo_comando = "cargar"

            nuevo_razonamiento = res_text
```

```
            if "RAZONAMIENTO:" in res_text:
                nuevo_razonamiento =
res_text.split("RAZONAMIENTO:")[1].strip()
            elif "razonamiento:" in texto_lower:
                nuevo_razonamiento =
texto_lower.split("razonamiento:")[1].strip()

            with lock:
                comando_actual_global = nuevo_comando
                razonamiento_actual_global = nuevo_razonamiento
            print(f"🤖 [HILO IA] ¡El Supervisor ha decidido! Nueva orden:
{nuevo_comando.upper()}")
        else: print(f"\n❌ [HILO IA] Error del servidor n8n:
{r.status_code}")
    except Exception as e: print(f"\n❌ [HILO IA] Error de conexión: {e}")
    with lock: ia_ocupada = False

dss_engine = dss.DSS
dss_text = dss_engine.Text
dss_circuit = dss_engine.ActiveCircuit
dss_text.Command = f"Compile '{RUTA_MASTER}'"
dss_text.Command = "Set Mode=Daily StepSize=1h Number=1"

historial_horas, historial_potencia_base, historial_potencia_real = [], [],
[]

with open(ARCHIVO_CSV, mode='w', newline='', encoding='utf-8-sig') as file:
    writer = csv.writer(file, delimiter=';')
    writer.writerow(["Hora", "Potencia_Base_kW", "Volatilidad",
"Potencia_Real_kW", "Estado_Solar", "Diagnostico", "IA_Comando",
"IA_Razonamiento"])

    print("🚀 GEMELO DIGITAL TFG - ESCENARIO: TORMENTA PERFECTA (ALTA
VOLATILIDAD)")

    for hora in range(1, 25):
        dss_text.Command = "BatchEdit Storage..* state=IDLE"

        dss_text.Command = "Solve"
        dss_circuit.SetActiveElement("Transformer.TR1")
        potencia_base_kw = dss_circuit.ActiveElement.Powers[0]

        if 8 <= hora <= 18:
            if random.random() > 0.5:
                estado_solar = "Producción muy baja (Nube densa)"
                factor_aleatorio = random.uniform(0.1, 0.3)
            else:
```

```
                estado_solar = "Producción Solar (Claro temporal)"
                factor_aleatorio = random.uniform(0.8, 1.1)
            else:
                estado_solar = "Sin Producción (Noche de tormenta)"
                factor_aleatorio = random.uniform(1.2, 1.7)

            potencia_kw = potencia_base_kw * factor_aleatorio

            historial_horas.append(hora)
            historial_potencia_base.append(potencia_base_kw * factor_aleatorio)
            historial_potencia_real.append(potencia_kw)

            print(f"\n--- [Hora {hora}:00] Potencia REAL: {potencia_kw:.2f} kW |
Sol: {estado_solar} ---")

            if LIMITE_INFERIOR <= potencia_kw <= LIMITE_SUPERIOR:
                print("🟢 ZONA ESTABLE: El Control Local asume el mando.")
                diagnostico = "Estable"
                with lock:
                    comando_actual_global = "reposo"
            else:
                diagnostico = "Exceso de Generación" if potencia_kw <
LIMITE_INFERIOR else "Pico de Consumo"
                print(f"⚠️ LÍMITE SUPERADO ({diagnostico}).")

                with lock: ocupada = ia_ocupada
                if not ocupada:
                    hilo_ia = threading.Thread(target=consultar_ia_asincrono,
args=(hora, potencia_kw, estado_solar, diagnostico))
                    hilo_ia.daemon = True
                    hilo_ia.start()
                    print("⏱️ (Esperando respuesta del Supervisor...)")
                    hilo_ia.join()
                else: print("⏳ (La IA todavía está evaluando...)")

            with lock: comando_aplicar = comando_actual_global
            print(f"⚙️ Aplicando a Baterías: {comando_aplicar.upper()}")

            if "descargar" in comando_aplicar:
                dss_text.Command = "BatchEdit Storage..* state=DISCHARGING"
                potencia_kw -= 30.0
                historial_potencia_real[-1] = potencia_kw
            elif "cargar" in comando_aplicar:
                dss_text.Command = "BatchEdit Storage..* state=CHARGING"
                potencia_kw += 30.0
                historial_potencia_real[-1] = potencia_kw
            else:
```

```
            dss_text.Command = "BatchEdit Storage..* state=IDLE"

        writer.writerow([f"{hora}:00", round(potencia_base_kw *
factor_aleatorio, 2), round(factor_aleatorio, 2), round(potencia_kw, 2),
estado_solar, diagnostico, comando_aplicar.upper(),
razonamiento_actual_global])
        time.sleep(15)

plt.figure(figsize=(12, 6))
plt.plot(historial_horas, historial_potencia_base, label='Potencia Base (Sin
IA)', linestyle='--', color='gray', alpha=0.7)
plt.plot(historial_horas, historial_potencia_real, label='Potencia Real
(Gestionada por IA)', color='blue', linewidth=2)
plt.axhline(y=LIMITE_SUPERIOR, color='red', linestyle=':', label='Límite
Superior (Pico Consumo)')
plt.axhline(y=LIMITE_INFERIOR, color='green', linestyle=':', label='Límite
Inferior (Exceso Solar)')
plt.title('Gestión de Smart Grid - Escenario: Tormenta y Alta Volatilidad')
plt.xlabel('Hora del Día (h)')
plt.ylabel('Potencia en el Transformador (kW)')
plt.legend()
plt.grid(True)
plt.xticks(historial_horas)
plt.tight_layout()
plt.show()
```

9.2.5 Analisis_datos.py

```
import csv
import matplotlib.pyplot as plt
import numpy as np
import os


RUTA_BASE = r"C:\Users\Tronchas\OneDrive - Universidad Europea de
Madrid\Escritorio\PYTHON_TFG"


archivos = {
    "Día Soleado": os.path.join(RUTA_BASE, "resultados_soleado.csv"),
    "Tormenta": os.path.join(RUTA_BASE, "resultados_tormenta.csv"),
    "Ola de Frío": os.path.join(RUTA_BASE, "resultados_olafrio.csv"),
    "Gran Apagón": os.path.join(RUTA_BASE, "resultados_apagon.csv")
}

escenarios = list(archivos.keys())
horas_peligro = []
intervenciones_bateria = []

print("📊 INICIANDO ANÁLISIS ESTADÍSTICO DEL TFG...")


for nombre, archivo in archivos.items():
    if not os.path.exists(archivo):
        print(f"❌ CUIDADO: No se encuentra el
archivo:\n{archivo}\nAsegúrate de que el CSV esté guardado en esa carpeta.")
        horas_peligro.append(0)
        intervenciones_bateria.append(0)
        continue

    peligro_count = 0
    bateria_count = 0

    with open(archivo, mode='r', encoding='utf-8-sig') as file:
        reader = csv.DictReader(file, delimiter=';')
        for row in reader:

            if row["Diagnostico"] != "Estable":
                peligro_count += 1

            if row["IA_Comando"] != "REPOSO":
                bateria_count += 1
```

```
    horas_peligro.append(peligro_count)
    intervenciones_bateria.append(bateria_count)
    print(f"☑ {nombre}: {peligro_count} horas en peligro | {bateria_count}
intervenciones de IA.")


x = np.arange(len(escenarios))
width = 0.35

fig, ax = plt.subplots(figsize=(11, 7))


rects1 = ax.bar(x - width/2, horas_peligro, width, label='Horas con Estrés
en la Red', color='#FF4444', edgecolor='black', alpha=0.9)
rects2 = ax.bar(x + width/2, intervenciones_bateria, width,
label='Intervenciones de IA (Acción VPP)', color='#00FFCC',
edgecolor='black', alpha=0.9)


ax.set_ylabel('Número de Horas (sobre 24h)', fontsize=13, fontweight='bold')
ax.set_title('TFG - Análisis Comparativo de Estrés Físico vs. Resiliencia
IA', fontsize=16, pad=20, fontweight='bold')
ax.set_xticks(x)
ax.set_xticklabels(escenarios, fontsize=12, fontweight='bold')
ax.legend(fontsize=12, loc='upper left')
ax.grid(axis='y', linestyle='--', alpha=0.7)
ax.set_ylim(0, 27)


ax.bar_label(rects1, padding=3, fontsize=12, fontweight='bold')
ax.bar_label(rects2, padding=3, fontsize=12, fontweight='bold')

fig.tight_layout()

print("\n🚀 Generando gráfico final comparativo...")
plt.show()
```

9.3 PROMPT AGENTE IA

"Eres el Sistema de Control IA de una Smart Grid.

SOLO se te consulta cuando hay una EMERGENCIA. NUNCA debes decidir "reposo". Tu deber es actuar para salvar la red.

SENSORES ACTUALES:

Potencia: {{ $json.body.potencia_actual_kw }} kW

Estado Solar: {{ $json.body.estado_paneles_solares }}

Diagnóstico: {{ $json.body.analisis_matematico }}

Evento Especial: {{ $json.body.evento_especial || "Ninguno" }}

REGLAS ESTRICTAS DE DECISIÓN:

Si el Diagnóstico dice "Pico de Consumo" o "COLAPSO DE RED EXTERNA":

El comando OBLIGATORIO es DESCARGAR.

Si el Diagnóstico dice "Exceso de Generación":

El comando OBLIGATORIO es CARGAR.

FORMATO OBLIGATORIO:

COMANDO: [Escribe ÚNICAMENTE la palabra CARGAR o DESCARGAR, nada más]

RAZONAMIENTO: [Explicación técnica]

REGLA DE VOCABULARIO (CRÍTICO):

En tu línea de "RAZONAMIENTO", ESTÁ TOTALMENTE PROHIBIDO usar las palabras "cargar", "descargar" o "sobrecargar". Usa sinónimos como: inyectar, absorber, almacenar, liberar, saturar o apoyar.

EJEMPLOS EXACTOS:

COMANDO: DESCARGAR

RAZONAMIENTO: La red demanda demasiada potencia. Procedo a inyectar energía desde el sistema de almacenamiento para evitar la saturación del transformador.

COMANDO: CARGAR

RAZONAMIENTO: Hay demasiada producción solar en la red. Procedo a almacenar los excedentes para mantener la estabilidad del voltaje."

BIBLIOGRAFÍA

Farhangi, H. (2010). *The path of the smart grid.* IEEE Power and Energy Magazine, 8(1), 18-28.

Grieves, M. (2014). *Digital Twin: Manufacturing Excellence through Virtual Factory Replication.* White Paper. Florida Institute of Technology.

McArthur, S. D., Davidson, E. M., Akpyeti, V. M., Gan, C. K., & Bell, K. R. (2007). *Multi-agent systems for power engineering applications—Part I: Concepts, approaches, and technical challenges.* IEEE Transactions on Power Systems, 22(4), 1743-1752. doi:10.1109/TPWRS.2007.908471.

Boyer, S. A. (2009). *SCADA: Supervisory Control and Data Acquisition.* (4th ed.). International Society of Automation.

Lasseter, B. (2002). *Microgrids.* IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Vol. 1, pp. 305-308). IEEE. doi:10.1109/PESW.2002.985003.

Gungor, V. C., Sahin, D., Kocak, T., Ergut, S., Buccella, C., Cecati, C., & Hancke, G. P. (2011). *Smart Grid Technologies: Communication Technologies and Standards.* IEEE Transactions on Industrial Informatics, 7(4), 529-539.

Glaessgen, E. H., & Stargel, D. S. (2012). *The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles.* 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference. doi:10.2514/6.2012-1818.

Fuller, A., Fan, Z., Day, C., & Barlow, C. (2020). *Digital Twin: Enabling Technologies, Challenges and Open Research.* IEEE Access, 8, 108952-108971. doi:10.1109/ACCESS.2020.2998358.

Wooldridge, M., & Jennings, N. R. (1995). *Intelligent agents: Theory and practice.* The Knowledge Engineering Review, 10(2), 115-152. doi:10.1017/S026988890000812X.

Pipattanasomporn, M., Feroze, H., & Rahman, S. (2009). *Multi-agent systems in a distributed smart grid: Design and implementation.* 2009 IEEE/PES Power Systems Conference and Exposition. doi:10.1109/PSCE.2009.4840087.

Satyanarayanan, M. (2017). *The Emergence of Edge Computing.* Computer, 50(1), 30-39. doi:10.1109/MC.2017.9.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., ... & Lample, G. (2023). *Llama: Open and efficient foundation language models.* arXiv preprint arXiv:2302.13971.

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). *Edge Computing: Vision and Challenges.* IEEE Internet of Things Journal, 3(5), 637-646. doi:10.1109/JIOT.2016.2579198.

Kersting, W. H. (2001). *Distribution system modeling and analysis.* CRC Press.

IEEE Power Engineering Society. (1991). *Radial distribution test feeders.* IEEE Transactions on Power Systems, 6(3), 975-985. doi:10.1109/59.119237.

Dugan, R. C., & McDermott, T. E. (2011). *An open source platform for collaborating on smart grid research.* 2011 IEEE Power and Energy Society General Meeting. doi:10.1109/PES.2011.6039825.

Smith, J. E., & Nair, R. (2005). *Virtual Machines: Versatile Platforms for Systems and Processes.* Elsevier/Morgan Kaufmann.

Wolf, G., & Halter, N. (2005). *Virtualization: From the Desktop to the Enterprise.* Apress. doi:10.1007/978-1-4302-0033-8.

Popek, G. J., & Goldberg, R. P. (1974). *Formal requirements for virtualizable third generation architectures.* Communications of the ACM, 17(7), 412-421. doi:10.1145/361011.361073.

Merkel, D. (2014). *Docker: lightweight Linux containers for consistent development and deployment.* Linux Journal, 2014(239), 2.

Bernstein, D. (2014). *Containers and cloud: From LXC to Docker to Kubernetes.* IEEE Cloud Computing, 1(3), 81-84. doi:10.1109/MCC.2014.51.

Pahl, C. (2015). *Containerization and the PaaS Cloud.* IEEE Cloud Computing, 2(3), 24-31. doi:10.1109/MCC.2015.51.

Dugan, R. C., & McDermott, T. E. (2011). *An open source platform for collaborating on smart grid research.* 2011 IEEE Power and Energy Society General Meeting. doi:10.1109/PES.2011.6039825.

Montenegro, D., & Dugan, R. C. (2013). *Real-time distribution system simulation using OpenDSS.* 2013 IEEE PES Innovative Smart Grid Technologies (ISGT). doi:10.1109/ISGT.2013.6497815.

Arango, L. G. (2017). *Python as a tool for power systems analysis.* Journal of Physics: Conference Series.

n8n.io. (2024). *n8n Documentation: Core concepts and node-based automation.* n8n official docs.

Lafia, S., Jablonski, S., & Gauch, H. (2019). *Workflow Automation in the Era of Cloud Computing.* Proceedings of the International Conference on Information Systems.

Hussain, A., & Wenbi, R. (2016). *A Review on Sentiment Analysis and Emotion Detection from Text.* 2016 International Conference on Communication Software and Networks (ICCSN). doi:10.1109/ICCSN.2016.7586645. (Para el contexto de procesamiento de lenguaje natural en flujos).

Meta. (2024). *Llama 3.2: Open Source AI for Edge Devices and Vision.* Meta AI Blog.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., ... & Lample, G. (2023). *Llama: Open and efficient foundation language models.* arXiv preprint arXiv:2302.13971.

Ollama. (2024). *Ollama Library and API Reference.* Official Documentation.

Yu, W., Liang, F., He, X., Hatcher, W. G., Lu, C., Lin, J., & Yang, X. (2017). *A Survey on the Edge Computing for the Internet of Things.* IEEE Access, 6, 6900-6919. doi:10.1109/ACCESS.2017.2778504.

Roberts, B. P., & Sandberg, C. (2011). *The Role of Energy Storage in Development of Smart Grids.* Proceedings of the IEEE, 99(6), 1139-1144. doi:10.1109/JPROC.2011.2116752.

Hill, C. A., Such, M. C., Chen, D., Gonzalez, J., & Grady, W. M. (2012). *Battery Energy Storage System (BESS) and Battery Management System (BMS) for Grid-Scale Applications.* IEEE Transactions on Smart Grid, 3(3), 1328-1335. doi:10.1109/TSG.2012.2190119