PROYECTO FIN DE GRADO

TÍTULO: **Blockchain aplicada a denominación de origen para energías renovables**

AUTOR: Álvaro Mancebo García de la Rosa

TITULACIÓN: Grado en Sistemas de Telecomunicaciones

TUTOR: Jesús Rodríguez Molina

DEPARTAMENTO: Ingeniería Telemática y Electrónica

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE: Waldo Saúl Pérez Aguiar

TUTOR: Jesús Rodríguez Molina

SECRETARIO: Pedro Castillejo Parrilla

Fecha de lectura:

Calificación:

El Secretario/La Secretaria,

Agradecimientos

A mis padres, por el esfuerzo incondicional en ofrecerme la oportunidad de enfrentar una carrera universitaria.

A mi hermana, a todos mis amigos y compañeros cercanos que han estado desde el primer día sumando y apoyando en este camino y hasta el final.

A mi tutor por presentar la idea atractiva del proyecto que ha hecho que ponga punto final a una etapa.

Resumen

Este proyecto estudia los cálculos de etiquetado y certificación energética bajo los desafíos del impacto climático y la necesidad de transición a fuentes de energía renovables.

Se establece la falta de trazabilidad de la procedencia de la energía como uno de los principales problemas para el impulso de energías sostenibles, tanto para las compañías y productores, como para los consumidores. Los consumidores que, sin conocer la fuente de la energía, no pueden evaluar el impacto ambiental ni apoyar activamente fuentes de energía renovables y sostenibles y, por otro lado, las compañías que se ven afectadas por los esfuerzos de diferenciar su producto en el mercado.

En este proyecto se construye, aplicando tecnología blockchain, una plataforma que ofrece al usuario una propuesta ágil a la hora de emitir denominaciones de origen para la energía generada. Ofreciendo a la solución todas las ventajas que proporciona el uso de blockchain como son la seguridad criptográfica y la propuesta de descentralización de organizaciones.

Abstract

This project studies energy labelling and certification calculations under the challenges of climate impact and the need for transition to renewable sources.

It establishes the lack of traceability of the energy source as one of the main problems for the promotion of sustainable energies, both for companies and producers, as well as for consumers. Consumers who, without knowing the source of the energy, cannot assess the environmental impact and cannot actively support renewable and sustainable energy sources and, on the other hand, companies that are affected by efforts to differentiate their product in the market.

This project builds, using blockchain technology, a platform that offers the user an agile approach to issuing designations of origin for the energy generated. The solution offers all the advantages provided using blockchain, such as cryptographic security and the proposal for decentralization of organizations.

Índice de contenidos

Listado de figuras

Listado de Tablas

Tabla de acrónimos

Acrónimo	Significado
ONU	Organización de Naciones Unidas
AIE	Agencia Internacional de la Energía
REE	Red Eléctrica de España
CNMC	Comisión Nacional de los Mercados y la Competencia
BOE	Boletín Oficial del Estado
PoW	Proof of Work
PoS	Proof of Stake
API	Application Programming Interface
REST	Representational State Transfer
EVM	Ethereum Virtual Machine o Máquina Virtual de Ethereum
URL	Uniform Resource Locator
IDE	Integrated Development Environment
dApps	Decentralized Applications
GUI	Graphical User Interface
CRUD	Create, Read, Update y Delete
POJO	Plain Old Java Object
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
RPC	Remote Procedure Call
GEI	Gases de Efecto Invernadero
JSON	JavaScript Object Notation
MVC	Modelo Vista Controlador
UML	Unified Modeling Language
ODS	Objetivos de Desarrollo Sonstenible

1. Introducción

Uno de los mayores desafíos a los que nos enfrentamos en la actualidad como sociedad es la generación de energía de manera sostenible y respetuosa con el medio ambiente. El uso de fuentes de energía tradicionales, como los combustibles fósiles, ha tenido un impacto significativo en el cambio climático, la calidad del aire y la disponibilidad de recursos naturales.

El consumo de electricidad se ha convertido en una necesidad básica y, sin embargo, a menudo desconocemos cualquier tipo de información relativa a la procedencia de la electricidad que consumimos y cómo se ha generado.

El presente proyecto tiene como objetivo promover el uso de energías renovables mediante el desarrollo de un sistema de etiquetado de la electricidad que asegure y avale la procedencia y calidad de la energía producida. Con ello se pretende brindar transparencia, trazabilidad y confiabilidad a los consumidores y participantes del mercado energético.

Para abordar este desafío, se establecerán mecanismos de certificación y verificación utilizando contratos inteligentes en blockchain, una tecnología que nos ofrece la oportunidad de crear un mercado energético más eficiente y descentralizado.

1.1. Objetivos

Los objetivos principales de este proyecto son:

- Analizar el impacto ambiental y los beneficios de las energías renovables frente a las no renovables.
- Conocer el sistema de etiquetado de la energía en España, así como los organismos y normativas de legislación vigente.
- Estudiar la tecnología blockchain con sus posibles beneficios y aplicaciones.
- Desarrollar un sistema que facilite al usuario la certificación y consulta de la energía renovable que se produce.
- Investigar propuestas de escalabilidad y futuros proyectos.

1.2. Estructura del proyecto

Tras la introducción al proyecto se presenta un marco tecnológico donde se analizan los antecedentes del proyecto, el impacto ambiental de energías renovables frente a las no renovables, los organismos y normativas de etiquetado eléctrico en España, y se describe la tecnología blockchain y sus aplicaciones en las denominaciones de origen.

Tras encajar el marco teórico donde se desarrolla el proyecto se presentan las especificaciones y restricciones de diseño del problema a resolver. Es en este apartado donde listamos las características funcionales y no funcionales del sistema, así como las limitaciones a la hora de implementarlo.

Con las especificaciones y restricciones de diseño establecidas, pasamos a describir el desarrollo de la solución planteada. Se presentan las herramientas y tecnologías de diseño utilizadas y se describe la arquitectura del sistema a bajo nivel. Se resaltan los fragmentos de código más relevantes en el proceso de implementación del contrato inteligente y la aplicación que lo opera dentro de la red blockchain.

Con la solución construida y funcionando pasamos a la fase de resultados donde plasmamos todas las evidencias por las que damos por concluidos nuestros objetivos.

Tras las evidencias de funcionamiento, estimamos el presupuesto del proyecto, facilitamos el código de la aplicación y un manual de usuario.

Para terminar, presentamos las conclusiones del trabajo, el impacto generado y un listado de referencias bibliográficas de las que se han hecho uso.

2. Marco tecnológico

2.1. Impacto de la producción energética en el medioambiente

La producción de energía es una actividad fundamental para el funcionamiento de la sociedad moderna, pero también tiene un impacto significativo en el medio ambiente. La dependencia de los combustibles fósiles en la generación de energía conlleva la emisión masiva de gases de efecto invernadero que contribuyen al calentamiento global y cambio climático. Estas emisiones están directamente relacionadas con los diferentes efectos del cambio climático tales como el aumento de la temperatura global, el derretimiento de los casquetes polares y el aumento del nivel del mar, el agotamiento de recursos, las pérdidas de biodiversidad, la contaminación del aire y agua que tiene efectos negativos en la salud humana, y la escasez de alimentos [1].

Para entender cómo podemos reducir eficazmente las emisiones y qué emisiones se pueden eliminar o no con las tecnologías actuales, es importante comprender de dónde provienen. En la figura 1 se muestra un desglose de las emisiones globales por sector, donde el sector energético es responsable de casi las tres cuartas partes de las emisiones.

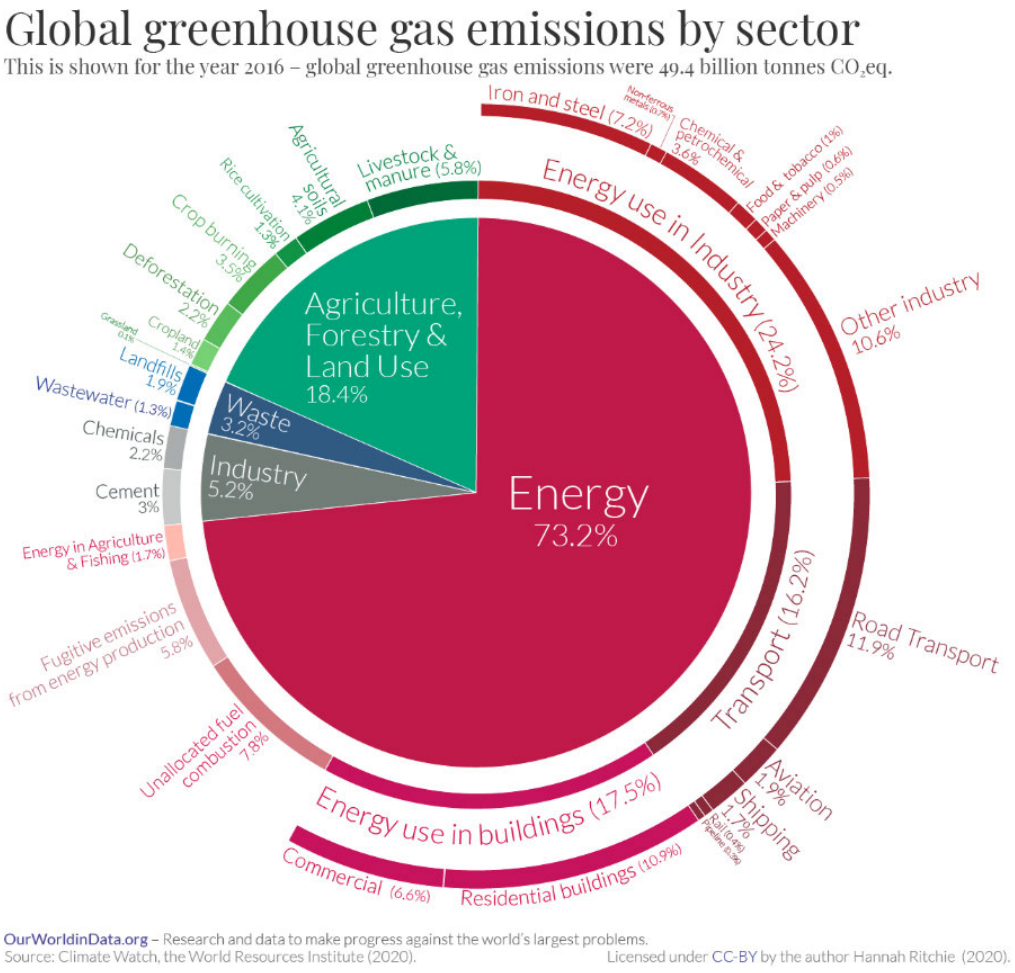

Fig. 1. Emisiones de gases de efecto invernadero por sector [2].

Naciones Unidas destaca la importancia de abordar los desafíos ambientales causados por la generación y el consumo de energía [3]. Se comunica la necesidad de realizar una transición a fuentes de energía renovable para mitigar los impactos negativos de la producción de energía en el medioambiente cuando antes.

Así, la ONU insta al uso de fuentes de energía sostenible como la energía solar, eólica, hidroeléctrica y geotérmica. Estas fuentes de energía tienen un menor impacto ambiental y contribuyen a la reducción de las emisiones de gases de efecto invernadero.

Diversas son las normativas que llaman a la necesidad de reducir las emisiones de CO2 describiendo las energías renovables no sólo como una alternativa limpia, accesible, sostenible y fiable para reemplazar los combustibles fósiles, sino también más económica [4].

Según un análisis de la Agencia Internacional de la Energía [5], las emisiones mundiales de dióxido de carbono relacionadas con la energía siguieron aumentando en 2021, alcanzando un nivel récord en millones de toneladas. Este incremento se debe principalmente al aumento del uso del carbón, que impulsó las emisiones en más de 2,000 millones de toneladas, el mayor aumento anual en términos absolutos.

La Figura 2 muestra la evolución en el aumento de emisiones de CO2 generales en el mundo durante los últimos 60 años.

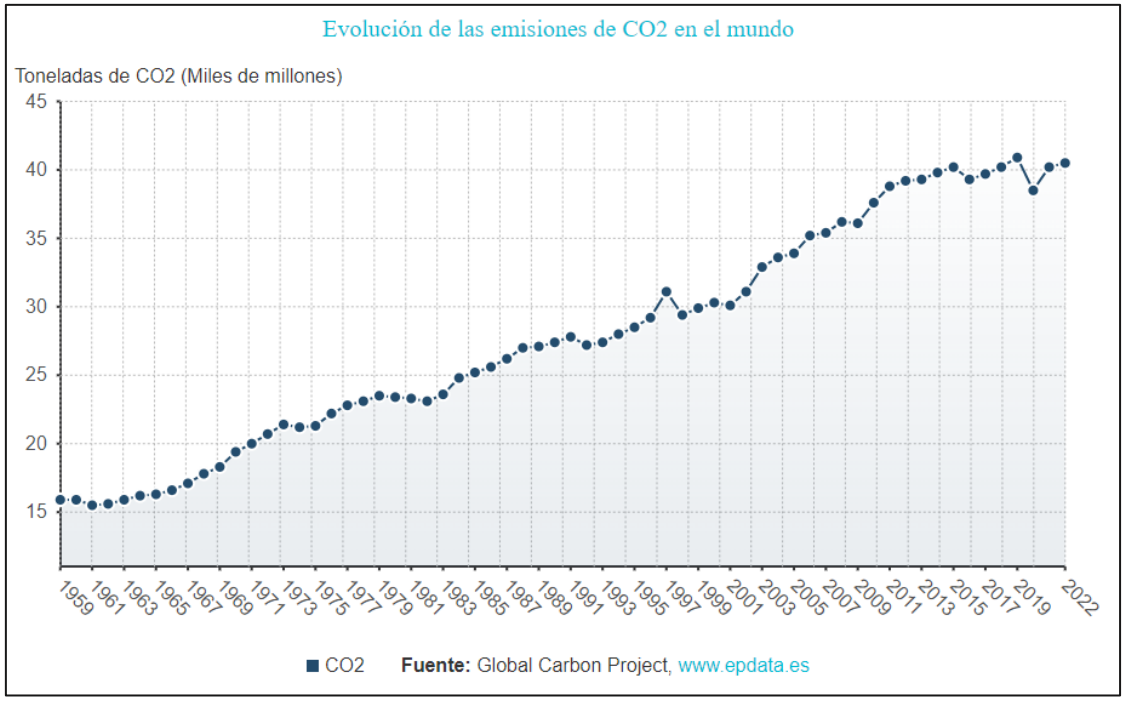

Fig. 2. Emisiones de CO2 mundiales [6].

Aunque los combustibles fósiles aún dominan la producción de energía, las fuentes renovables están en crecimiento, representando alrededor del 29 % de la electricidad

actualmente [7]. La Figura 3 muestra la evolucion de la potencia electrica instalada y las principales fuentes del sistema electrico en España.

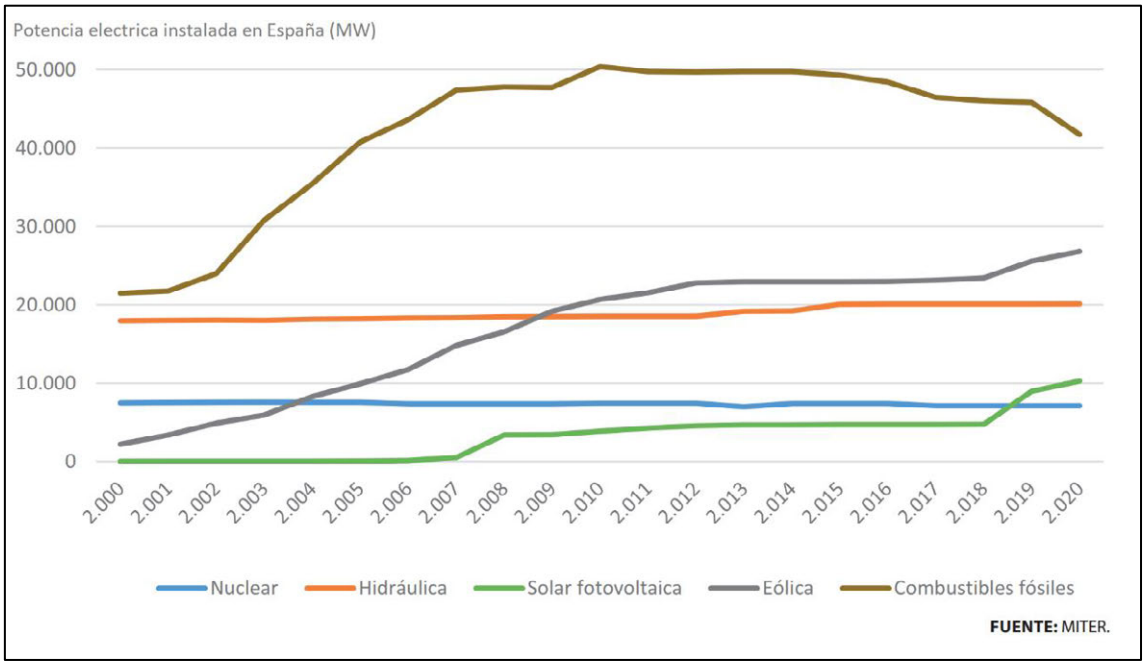

Fig. 3. Principales fuentes del sistema eléctrico en España [7].

Uno de los principales retos es la disponibilidad intermitente de algunas fuentes de energía renovable, como la energía solar y la energía eólica. A diferencia de las fuentes no renovables, como el petróleo y el gas natural (cuya disponibilidad es de manera efectiva relativamente constante mientras existan), las energías renovables dependen de factores naturales, como la radiación solar y la fuerza del viento. Esto significa que la generación de energía renovable puede variar según las condiciones climáticas y no puede proporcionar un suministro constante y totalmente predecible.

Para superar este reto, se requiere el desarrollo de tecnologías de almacenamiento de energía más eficientes y económicas. Los sistemas de almacenamiento permiten capturar la energía generada en momentos de mayor producción y utilizarla cuando la demanda es alta o cuando las fuentes renovables no están disponibles. Las baterías de almacenamiento, los sistemas de bombeo hidroeléctrico y otras tecnologías emergentes están desempeñando un papel crucial en el aumento de la capacidad de almacenamiento de energía renovable.

Otro desafío importante es la integración de las energías renovables en las redes eléctricas existentes. **Las redes eléctricas fueron diseñadas originalmente para distribuir energía generada por fuentes no renovables de manera centralizada**. La generación distribuida a partir de fuentes renovables plantea nuevos desafíos técnicos, como la gestión de la variabilidad de la generación comentada y la implementación de sistemas de gestión inteligente de la energía.

La modernización y adaptación de las infraestructuras de transmisión y distribución son esenciales para facilitar la transición hacia un sistema energético basado en fuentes renovables. Esto implica inversiones en la mejora de las redes eléctricas, la implementación de tecnologías de comunicación avanzadas y la adopción de enfoques innovadores, como las microredes y las redes inteligentes.

Además de los desafíos técnicos, también existen desafíos económicos y políticos. Aunque las energías renovables han experimentado reducciones significativas en los costos de generación, todavía existen barreras financieras y regulaciones desfavorables que dificultan su adopción masiva [8]. La implementación de políticas de apoyo, como incentivos fiscales y tarifas de alimentación, así como la eliminación de los subsidios a los combustibles fósiles, pueden ayudar a nivelar estas circunstancias y promover una mayor penetración de las energías renovables [9].

2.2. Sistema Eléctrico Español

El Sistema Eléctrico Español es un conjunto de infraestructuras, normativas y procesos que permiten la generación, transmisión, distribución y comercialización de la electricidad en España [10]. Su principal objetivo es asegurar el suministro eléctrico a todos los consumidores del país de manera segura, eficiente y económica.

Las principales características y funcionalidades del Sistema Eléctrico Español son:

- Generación de electricidad: El sistema cuenta con una diversidad de fuentes de generación eléctrica, incluyendo centrales térmicas, hidroeléctricas, nucleares, eólicas, solares, entre otras. Estas centrales producen la energía eléctrica que se distribuye a nivel nacional.

- Transmisión de electricidad: Una extensa red de líneas de alta tensión y subestaciones permite la transmisión de la electricidad generada desde las centrales hasta los centros de consumo en todo el país. Esta red de transmisión es operada y gestionada por el operador del sistema eléctrico, que en España es Red Eléctrica de España.

- Distribución de electricidad: Una vez que la electricidad es transmitida a nivel nacional, se distribuye a través de redes de distribución a nivel regional y local. Estas redes están compuestas por líneas de media y baja tensión que llegan a los hogares, empresas e industrias.

- Comercialización de electricidad: En el Sistema Eléctrico Español existen empresas comercializadoras que se encargan de la venta de la electricidad a los

consumidores. Estas empresas ofrecen distintas tarifas y contratos de suministro eléctrico, adaptados a las necesidades de los clientes.

- Mercado eléctrico: El sistema cuenta con un mercado eléctrico regulado en el que se negocia el precio de la electricidad en función de la oferta y la demanda. Este mercado permite la compra y venta de energía eléctrica entre generadores, comercializadoras y consumidores, facilitando la eficiencia y competencia en el sector.

2.3. Denominaciones de origen y etiquetado de la electricidad

En la Circular 2/2021 de la Comisión Nacional de los Mercados y la Competencia publicado en el BOE el 19 de febrero de 2021 se establece la metodología y las condiciones para el etiquetado de la electricidad en España [11]. El etiquetado de la electricidad es un sistema de información que proporciona detalles sobre el origen de la electricidad consumida y su impacto ambiental.

El etiquetado de la electricidad busca brindar a los consumidores información clara y transparente sobre la procedencia de la electricidad que utilizan, permitiéndoles tomar decisiones informadas en función de criterios ambientales. De esta manera, los consumidores pueden conocer si la electricidad que consumen proviene de fuentes renovables o no renovables, como la energía solar, eólica, hidroeléctrica, nuclear o de combustibles fósiles.

La circular establece los requisitos y la metodología que deben seguir los comercializadores de electricidad para el etiquetado. Esto incluye la obligación de proporcionar información sobre la composición de la energía eléctrica vendida, especificando el porcentaje de cada fuente de energía utilizada, así como las emisiones de CO2 y los residuos radiactivos asociados a su generación.

Además, la circular también establece las condiciones para el uso del **logotipo de garantía de origen renovable**, que certifica que la electricidad proviene de fuentes renovables. Este logotipo permite a los consumidores identificar de manera fácil y rápida si la electricidad que están adquiriendo tiene un impacto ambiental reducido.

Con la implementación de esta circular, se pretende fomentar la transparencia en el mercado eléctrico y promover el consumo responsable de energía. Al proporcionar información detallada sobre el origen y el impacto ambiental de la electricidad, se busca incentivar la demanda de energía renovable y contribuir a la reducción de las emisiones de gases de efecto invernadero.

2.4. Tecnología Blockchain

La tecnología blockchain es un enfoque revolucionario para la gestión de datos y transacciones en línea. A diferencia de los sistemas tradicionales, que dependen de una autoridad centralizada para validar y registrar las transacciones, la tecnología **blockchain opera en una red distribuida y descentralizada** de nodos interconectados.

En su forma más básica, una blockchain es una estructura de datos que consiste en una cadena de bloques enlazados. Cada bloque contiene un conjunto de transacciones y una referencia al bloque anterior, formando una secuencia cronológica inmutable. Esta estructura de bloques encadenados asegura que las transacciones sean transparentes, seguras y resistentes a la manipulación. El proceso mediante el cual se verifica y valida las transacciones en una red blockchain se denomina minería. En la Figura 4 se describe un ejemplo de este funcionamiento [12].

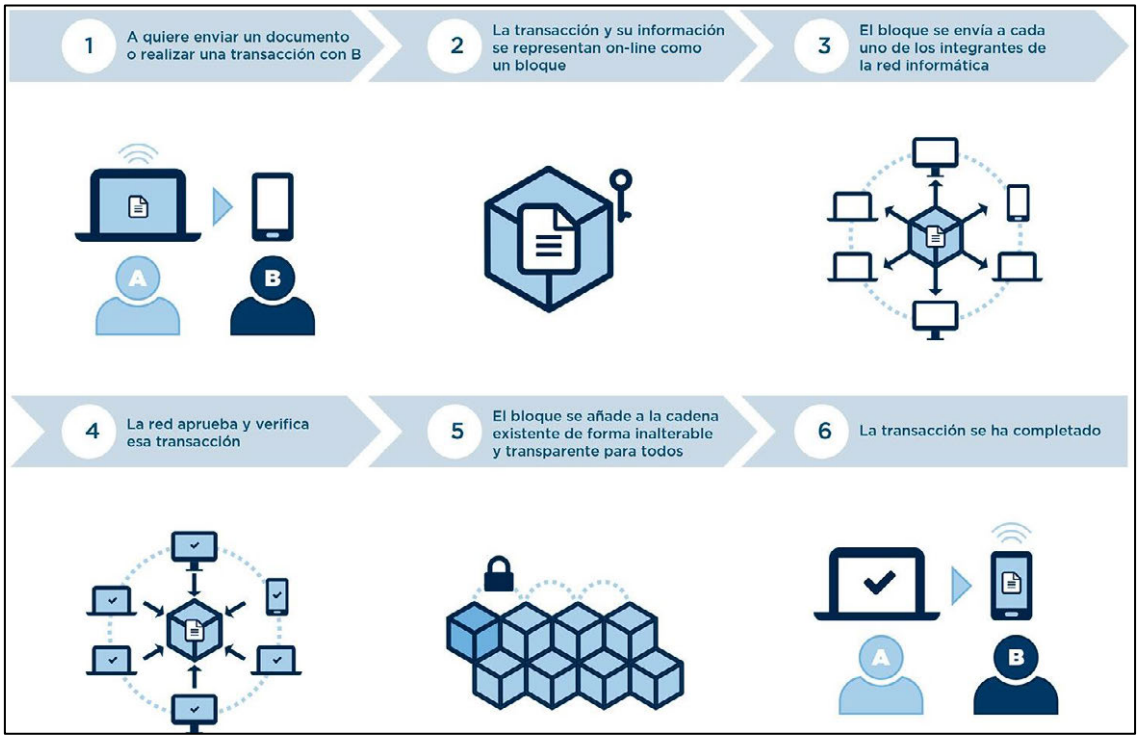

Fig. 4. Esquema del funcionamiento de blockchain [12].

Tres de los conceptos fundamentales en los que se basa la tecnología blockchain son:

- Descentralización: En lugar de confiar en una entidad centralizada, como un banco o un gobierno, la blockchain distribuye la autoridad y la responsabilidad a todos los nodos participantes en la red. Cada nodo tiene una copia del registro completo de transacciones y participa en el proceso de consenso para validar y confirmar nuevas transacciones.

- Criptografía: La seguridad en la blockchain se logra mediante técnicas criptográficas avanzadas. Cada transacción se cifra y se agrega a un bloque

utilizando algoritmos criptográficos, lo que garantiza la integridad y confidencialidad de los datos. Además, las firmas digitales se utilizan para verificar la autenticidad de las transacciones y la identidad de los participantes.

- Consenso: La tecnología blockchain utiliza algoritmos de consenso para llegar a un acuerdo sobre el estado y el orden de las transacciones en la red. Esto evita la necesidad de un intermediario central y garantiza que todos los nodos estén de acuerdo en la validez de las transacciones. Ejemplos de algoritmos de consenso populares son [13]:

    - El Proof of Work (PoW) o Prueba de Trabajo. En este sistema, los mineros compiten entre sí para resolver problemas criptográficos complejos utilizando una gran cantidad de poder de cómputo. El primer minero en encontrar la solución correcta tiene el derecho de agregar un bloque de transacciones a la cadena y es recompensado por ello.
    - El Proof of Stake (PoS) o Prueba de Participación. En este sistema, en lugar de competir por resolver problemas matemáticos, los validadores son seleccionados aleatoriamente para validar y confirmar las transacciones en función de la cantidad de fondos que poseen y están dispuestos a arriesgar como garantía.

2.4.1. Contratos inteligentes

Los contratos inteligentes o *smart contracts*, son programas informáticos que se ejecutan automáticamente dentro de una red blockchain cuando se cumplen ciertas condiciones predefinidas. Estos contratos se basan en la lógica de programación para establecer y hacer cumplir acuerdos entre diferentes partes sin necesidad de intermediarios. Una vez que se programa el contrato inteligente y se despliega en la cadena de bloques, se vuelve inmutable y su ejecución es automática y autónoma [14].

La principal ventaja de los contratos inteligentes es su capacidad para eliminar intermediarios y automatizar la ejecución de acuerdos. Al eliminar la necesidad de confiar en terceros, como instituciones financieras, los *smart contracts* ofrecen una forma más segura, transparente y eficiente de realizar transacciones.

Algunas de las aplicaciones de los contratos inteligentes son: la gestión de la cadena de suministro, los seguros, la propiedad intelectual, los sistemas de votación, entre otros. Su utilización permite reducir costos, eliminar la posibilidad de fraude y agilizar los procesos comerciales.

Sin embargo, es importante tener en cuenta que los contratos inteligentes son tan seguros como el código que los implementa. Errores o vulnerabilidades en el código pueden llevar a resultados no deseados o explotación por parte de terceros. Por esta razón, es

fundamental contar con un desarrollo y auditoría rigurosa de los contratos para garantizar su seguridad y funcionalidad [15].

2.4.2. Denominaciones de origen en blockchain

Las denominaciones de origen en el contexto de blockchain se refieren a la certificación y trazabilidad de productos agrícolas, alimentarios, artesanales u otros que poseen una calidad, reputación o características específicas debido a su origen geográfico.

En la cadena de bloques, se utiliza esta tecnología para garantizar la autenticidad y veracidad de la información relacionada con las denominaciones de origen. Cada vez más, se están desarrollando soluciones basadas en blockchain para proteger y promover estas denominaciones, ya que proporcionan una forma segura y transparente de registrar y rastrear la procedencia de los productos.

Utilizando blockchain, es posible realizar registros de información que se vuelven inmutables, lo que significa que no se puede modificar ni falsificar. Esto ayuda a proteger la integridad de las certificaciones y brinda confianza a los consumidores al garantizar la autenticidad de los productos. Se permite rastrear el historial de un producto desde su origen hasta el punto de venta y los consumidores pueden verificar fácilmente la autenticidad de un producto y conocer su origen exacto, lo que promueve y fomenta la comercialización justa y el reconocimiento de los productos con estas características distintivas [16].

2.4.3. Crítica en el impacto energético de blockchain

Antes de la implementación del algoritmo de consenso Proof of Stake, el consumo de energía en las redes blockchain basadas en Proof of Work tenía varios impactos ambientales negativos [17]:

- Consumo de energía y huella de carbono: Los algoritmos PoW requieren que los participantes resolvieran problemas criptográficos complejos para validar transacciones y asegurar la red. Esto implicaba el uso intensivo de recursos computacionales y, por lo tanto, un alto consumo de energía.

- Desperdicio de recursos: La minería de criptomonedas en PoW requiere un hardware especializado y potente. A medida que aumenta la competencia, los mineros invierten en equipos más eficientes pero quedan obsoletos rápidamente, lo que conduce al desperdicio de recursos tecnológicos y energéticos.

- Centralización de la minería: El alto costo y el consumo de energía asociados con la minería llevan a una centralización geográfica de la actividad minera. Las regiones con electricidad barata tienen una ventaja competitiva, lo que puede

conducir a la concentración del poder y la toma de decisiones en manos de unos pocos actores.

Estas críticas han llevado a un mayor enfoque en el desarrollo de algoritmos de consenso más eficientes desde el punto de vista energético, como PoS, que requiere una fracción significativamente menor de energía para validar transacciones y asegurar la red. Además, se están explorando soluciones para utilizar energías renovables en la minería de criptomonedas y reducir aún más la huella de carbono [18].

2.5. Trabajos previos relacionados con este proyecto

El grupo Iberdrola ha implementado un proyecto piloto basado en blockchain para garantizar en tiempo real que la energía suministrada y consumida es 100 % renovable [19]. Mediante el uso la cadena de bloques, Iberdrola logra conectar las plantas de generación con los puntos de consumo con el fin de rastrear su producción y origen, aumentando la transparencia y fomentando el uso de energía renovable.

Este proceso acelera y automatiza los procesos de certificación de energía renovable y garantiza un mayor grado de trazabilidad. Esto es especialmente relevante en contratos de compraventa de energía a largo plazo basados en activos renovables, donde se requiere demostrar el origen de la electricidad suministrada.

Iberdrola considera la tecnología blockchain una herramienta clave para acelerar la descarbonización de la economía al proporcionar trazabilidad, seguridad y rapidez a las transacciones de energía verde. La compañía lanza la propuesta que plantea conectar parques eólicos y centrales hidroeléctricas con sedes corporativas, asegurando a sus clientes que la electricidad que reciben es limpia y permitiéndoles rastrear su procedencia.

Para este proyecto, Iberdrola ha utilizado la plataforma de blockchain escalable y de código abierto de Energy Web Foundation, una organización sin fines de lucro que se dedica a acelerar la adopción de tecnologías de blockchain en el sector energético [20].

El blockchain es uno de los pilares de la estrategia de transformación digital de Iberdrola y se utiliza no solo para certificar el origen de la energía renovable, sino también para garantizar transacciones seguras y agilizar los contratos entre las empresas del grupo.

La compañía también participa en proyectos que exploran el uso de blockchain en operaciones de compraventa de energía y gas natural sin intermediarios y en certificar información sobre eventos de la red para mejorar la calidad del servicio y la atención al cliente en su negocio de distribución.

3. Especificaciones de diseño

3.1. Requisitos de diseño

Requisitos funcionales:

- Los usuarios deben poder emitir certificados energéticos dentro de la blockchain y obtener una caracterización o logotipo correspondiente a la información enviada.

- Solo los usuarios titulares del certificado emitido podrán transferir una única vez dicho certificado a otro usuario.

- Los usuarios deben poder realizar una consulta de la información de un certificado concreto dentro de la blockchain.

- El sistema debe presentar una vista web donde se listen todos los certificados actualizados y emitidos por la plataforma.

Requisitos técnicos no funcionales:

- Se utiliza Java como lenguaje de programación en el desarrollo del sistema.

- El sistema debe integrar un API REST que con el uso implícito del protocolo HTTP permita realizar operaciones tipo GET, POST, PUT, o DELETE.

3.2. Restricciones de diseño

El sistema se desarrolla en un entorno local:

- No se realiza una publicación a internet del sistema.

- No se realiza interacciones con una blockchain real debido a la necesidad de contar con una cuenta y billetera con fondos reales y donde nuestro contrato inteligente estaría expuesto a posibles vulnerabilidades en el código, ataque de hackers o errores humanos.

4. Descripción de la solución propuesta

4.1. Componentes y tecnologías de diseño seleccionadas

Para cumplir las especificaciones de diseño y llegar al objetivo del proyecto, pasamos a describir los componentes y tecnologías que se han utilizado en el desarrollo del sistema.

4.1.1. Ethereum

Ethereum es una plataforma blockchain descentralizada que permite la creación y ejecución de contratos inteligentes. Ethereum proporciona una infraestructura para desarrollar aplicaciones descentralizadas y ejecutar contratos inteligentes, la máquina virtual de Ethereum [21].

Ethereum es la plataforma blockchain elegida en nuestro proyecto debido a que ha abrazado el algoritmo de consenso Proof of Stake como parte de su actualización a Ethereum 2.0. Recordamos que este algoritmo traer mejoras significativas en términos de escalabilidad y seguridad, pero sobre todo de eficiencia energética.

4.1.2. Solidity

Solidity es el lenguaje de programación utilizado para escribir contratos inteligentes en la plataforma Ethereum. Se basa en la sintaxis de JavaScript y está diseñado para ser utilizado en el entorno de ejecución de la EVM. Permite a los desarrolladores definir la lógica y el comportamiento de los contratos inteligentes, que son programas autónomos que se ejecutan en la blockchain de Ethereum [22].

Algunas características clave de Solidity [23] incluyen el tipado estático, donde se deben declarar explícitamente los tipos de datos de las variables utilizadas en el contrato. También permite la herencia, lo que facilita la reutilización de código y la organización de la funcionalidad en módulos separados.

Existen modificadores en Solidity para alterar el comportamiento de las funciones o aplicar restricciones a su ejecución, lo que permite verificar condiciones previas antes de ejecutar una función. Se utilizan **eventos** para emitir y registrar información relevante sobre las interacciones y cambios en el contrato inteligente, lo que **permite a los usuarios y aplicaciones externas obtener información sobre lo que sucede en la blockchain**.

Elegimos Solidity como lenguaje de programación para nuestro contrato inteligente dado que se ha convertido en uno de los lenguajes de programación más populares para el desarrollo de contratos inteligentes con una amplia comunidad, y cumple con algunas características como los eventos, que nos facilitan el diseño del proyecto.

4.1.3. Remix

Remix es una herramienta en línea utilizada para desarrollar, compilar y depurar contratos inteligentes en la plataforma Ethereum. Es un IDE basado en web que proporciona una interfaz fácil de usar para interactuar con la cadena de bloques Ethereum y escribir contratos inteligentes en el lenguaje de programación Solidity [24].

Remix ofrece una variedad de características que facilitan el desarrollo de contratos inteligentes, como resaltado de sintaxis, autocompletado, verificación estática del código, compilación y despliegue de contratos en una red de prueba o en la red principal de Ethereum. También brinda herramientas de depuración que permiten rastrear y solucionar errores en los contratos inteligentes.

Debido a la interfaz de uso intuitivo y ágil que proporciona, utilizamos Remix en el proceso de desarrollo y depuración del contrato.

4.1.4. Ganache

Ganache es una herramienta de desarrollo de blockchain que se utiliza principalmente en el entorno de Ethereum. Proporciona un entorno local de blockchain para desarrolladores, permitiéndoles crear y probar aplicaciones descentralizadas en un entorno controlado y seguro [25].

Ganache simula una red blockchain privada en la que los desarrolladores pueden crear cuentas, realizar transacciones, desplegar contratos inteligentes y realizar pruebas sin necesidad de conectarse a la red principal de Ethereum. Permite a los desarrolladores emular el comportamiento de una red blockchain completa en su propia máquina, lo que agiliza el proceso de desarrollo y depuración de aplicaciones descentralizadas.

Además, Ganache ofrece una interfaz gráfica de usuario que muestra información detallada sobre el estado de la blockchain simulada, como saldos de cuentas, transacciones realizadas y eventos generados por contratos inteligentes. Esto facilita la comprensión y el análisis de los resultados de las pruebas y por ello utilizamos esta herramienta en el proyecto.

4.1.5. Solc-windows

Solc-windows es una versión del compilador de Solidity para Windows. Se utiliza para convertir el código fuente escrito en Solidity en bytecode ejecutable en la blockchain de Ethereum.

Esta versión proporciona una interfaz de línea de comandos que permite a los desarrolladores compilar sus contratos inteligentes escritos en Solidity directamente en un entorno Windows [26].

4.1.6. Java

Java es un lenguaje de programación de alto nivel, orientado a objetos y multiplataforma. Fue desarrollado por Sun Microsystems, adquirida posteriormente por Oracle Corporation, y lanzado inicialmente en 1995. Desde entonces, Java se ha convertido en uno de los lenguajes más populares y ampliamente utilizados en el mundo de la programación [27].

Una de las características más destacadas de Java es que el código Java puede ser compilado en un formato intermediario llamado bytecode, que puede ser ejecutado en cualquier máquina virtual Java y ofrecer independencia del sistema operativo subyacente. Esto ha permitido que las aplicaciones Java sean altamente portables y funcionen en diferentes plataformas, desde computadoras de escritorio hasta dispositivos móviles y sistemas integrados.

Java se utiliza en una amplia variedad de aplicaciones, desde aplicaciones de escritorio y servidores de aplicaciones empresariales hasta aplicaciones web, dispositivos móviles, sistemas embebidos y más. Es ampliamente utilizado en el desarrollo de software empresarial, juegos, aplicaciones móviles Android, sistemas de control y monitoreo, y muchas otras áreas.

Proporciona características como la herencia, el polimorfismo, el encapsulamiento y la abstracción, que permiten la reutilización de código, la modularidad y el desarrollo de aplicaciones más mantenibles y escalables.

El uso de Java como lenguaje de programación es un requisito inicial de diseño en el proyecto.

4.1.7. HTML

HTML es un lenguaje de representación utilizado para crear y estructurar contenido en la web. Permite definir la estructura y el formato de los documentos web mediante el uso de etiquetas o elementos. Estas etiquetas se utilizan para marcar y organizar el contenido, como encabezados, párrafos, enlaces, imágenes y listas. HTML se complementa con CSS para estilizar la apariencia de la página y con JavaScript para agregar interactividad y funcionalidad dinámica [28].

Elegimos HTML para desarrollar el diseño de una vista web en nuestra plataforma.

4.1.8. Gradle

Gradle es una herramienta de construcción de código abierto utilizada en el desarrollo de software. Proporciona un sistema flexible y potente para compilar, construir y gestionar proyectos de software de manera eficiente.

En términos simples, Gradle es una herramienta que ayuda a los desarrolladores a automatizar tareas repetitivas relacionadas con la construcción y el despliegue de su software. Permite definir y gestionar las dependencias del proyecto, compilar el código fuente, ejecutar pruebas, empaquetar y distribuir la aplicación, entre otras tareas.

Una de las características principales de Gradle es su flexibilidad y capacidad de personalización. Utiliza un lenguaje de dominio específico basado en Groovy o Kotlin, que permite a los desarrolladores describir la estructura del proyecto y las tareas de construcción de forma declarativa y legible [29].

Gradle cuenta con un sistema de gestión de dependencias integrado que facilita la incorporación y actualización de bibliotecas y módulos externos en el proyecto. Permite descargar automáticamente las dependencias necesarias y gestionar las versiones de forma eficiente.

La elección de usar Gradle frente a Maven, otra opción ampliamente escogida, se ha hecho siguiendo la mejor gestión de dependencias y capacidad de personalización.

4.1.9. API REST

Un API REST es un conjunto de reglas y convenciones que permite a los sistemas informáticos comunicarse entre sí a través de la web de manera uniforme y estandarizada. REST es un estilo arquitectónico que se basa en el protocolo HTTP y utiliza sus métodos y códigos de estado para realizar operaciones sobre recursos.

Una breve descripción de los métodos de uso más comunes es:

- **GET**: Solicita la obtención de un recurso específico o una colección de recursos.
- **POST**: Crea un nuevo recurso utilizando los datos proporcionados en la solicitud.
- **PUT**: Actualiza un recurso existente con los datos proporcionados en la solicitud.
- **DELETE:** Elimina un recurso específico.

En un API REST, los recursos son elementos de información o funcionalidades que pueden ser accedidos y manipulados mediante las solicitudes HTTP. Cada recurso es identificado mediante una URL única y puede representar cualquier cosa, como datos,

objetos, servicios o acciones. El API REST define cómo se deben estructurar las URLs, cómo se deben enviar las solicitudes y cómo se deben recibir las respuestas.

Una característica fundamental de un API REST es que es stateless, lo que significa que no guarda información sobre el estado de las comunicaciones anteriores. Cada solicitud HTTP es independiente y contiene toda la información necesaria para ser procesada y responder de forma adecuada. Los datos en un API REST se transmiten generalmente en formato JSON o XML [30].

Un API REST permite a los desarrolladores construir aplicaciones modulares, flexibles y fáciles de mantener, ya que separa claramente el frontend o interfaz de usuario del backend o lógica de la aplicación y almacenamiento de datos.

La integración de un API REST es un requisito técnico en el diseño del proyecto.

4.1.10. Spring Boot

Spring Boot es un framework de desarrollo de aplicaciones de Java que facilita la creación de aplicaciones sólidas y escalables. Se basa en el framework Spring y proporciona una forma simplificada de configurar y desarrollar aplicaciones Java con una configuración mínima y un enfoque de convención sobre configuración.

Spring Boot ofrece varias características y beneficios, como [31]:

- Configuración automática: Spring Boot utiliza la configuración automática basada en convenciones para configurar automáticamente gran parte del entorno de la aplicación. Esto permite a los desarrolladores centrarse en el desarrollo de la lógica de negocio en lugar de configuraciones tediosas.

- Incorporación de servidores web: Spring Boot incluye un servidor web incorporado, como Tomcat o Jetty, lo que facilita la ejecución de la aplicación sin necesidad de configuraciones adicionales.

- Gestión de dependencias: Spring Boot proporciona una forma sencilla de gestionar las dependencias de la aplicación a través de Maven o Gradle. También ofrece un repositorio centralizado de dependencias y versiones, lo que simplifica la gestión de las bibliotecas utilizadas en el proyecto.

- Actuación como microservicios: Spring Boot es una opción popular para la creación de microservicios debido a su enfoque ligero y modular. Permite crear y desplegar fácilmente servicios independientes y escalables.

- Integración con Spring ecosystem: Spring Boot se integra perfectamente con otros proyectos y módulos de Spring, como Spring Data, Spring Security y Spring Cloud, lo que brinda a los desarrolladores una amplia gama de herramientas para construir aplicaciones empresariales completas.

En el contexto del framework Spring, los conceptos de Controller, Service, Repository y Model se utilizan para seguir el **patrón de diseño MVC** [32] y facilitar la separación de responsabilidades en una aplicación web. A continuación, se explica el uso de cada uno de estos componentes en Spring [33]:

- El Controller o controlador es el componente encargado de recibir las solicitudes HTTP, procesarlas y enviar las respuestas adecuadas. En Spring, los controladores se anotan con la anotación @Controller o @RestController y definen los endpoints de la API REST. Dentro de los controladores, se definen los métodos que se ejecutan cuando se accede a cada endpoint, y se pueden acceder a los datos enviados en la solicitud y devolver los datos necesarios en la respuesta. El controlador también puede llamar a los servicios correspondientes para procesar la lógica de negocio.

- El Service o servicio es el componente encargado de implementar la lógica de negocio de la aplicación. Los servicios encapsulan operaciones y funcionalidades relacionadas, y se utilizan para realizar operaciones más complejas que no corresponden directamente al controlador. Los servicios se anotan con @Service y se pueden inyectar en los controladores u otros servicios mediante la anotación @Autowired. Los servicios pueden interactuar con los repositorios para acceder a los datos almacenados.

- El Repository o repositorio es el componente encargado de interactuar con la capa de persistencia de datos, como una base de datos o una API externa. Los repositorios se anotan con @Repository y proporcionan métodos para realizar operaciones CRUD en los datos. Los repositorios se pueden inyectar en los servicios para acceder y manipular los datos de manera eficiente.

- El Model o modelo representa la estructura de datos utilizada en la aplicación. En Spring, los modelos suelen ser clases POJO que contienen atributos y métodos getter/setter para acceder a los datos. Los modelos se utilizan para transferir y manipular datos entre el controlador, el servicio y el repositorio.

Por todas las características y beneficios, junto con la presentación del MVC, utilizamos la herramienta Spring Boot en nuestro proyecto.

4.1.11. Web3j

Web3j es una biblioteca de Java que facilita la interacción con la red Ethereum. Permite a los desarrolladores conectarse a nodos de Ethereum, enviar transacciones, leer datos de contratos inteligentes y gestionar eventos en la red [34]. Una de las características destacadas de Web3j es su **capacidad para generar automáticamente clases de Java a partir de contratos inteligentes escritos en Solidity**. Esto simplifica en gran medida la interacción con los contratos, ya que proporciona una interfaz de programación fuertemente tipada y autocompletada.

Web3j cuenta con una comunidad activa de desarrollo y un equipo de mantenimiento dedicado. La biblioteca se actualiza regularmente para agregar nuevas características, solucionar problemas y mantenerse al día con los cambios en la red Ethereum. También cuenta con una documentación completa y soporte disponible para ayudar a los desarrolladores a utilizar y aprovechar al máximo Web3j en sus proyectos.

En la Figura 5 se muestra una arquitectura que contiene una aplicación Java, web3j y una red de Ethereum, donde JSON-RPC actúa como un protocolo ligero de comunicación que permite a las aplicaciones enviar solicitudes y recibir respuestas en formato JSON con la red Ethereum.

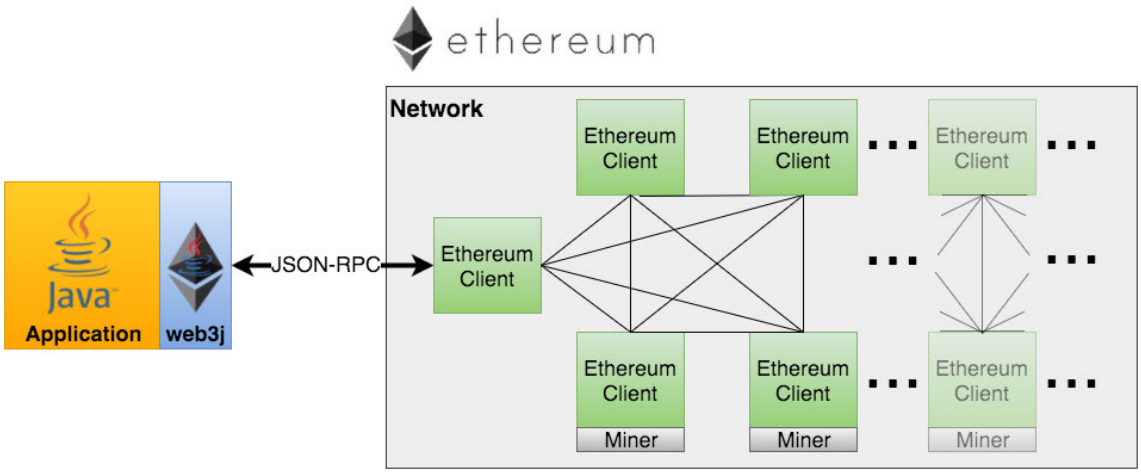

Fig. 5. Comunicación entre Java Ethereum a través de web3j [34].

Aunque existen otras librerías para interactuar con redes Ethereum en Java, decidimos usar Web3j por su integración nativa y la capacidad de generar automáticamente clases de Java a partir de contratos en Solidity.

4.1.12. IntelliJ IDEA

IntelliJ IDEA es un entorno de desarrollo integrado creado por JetBrains. Es una herramienta utilizada por desarrolladores de software para escribir, depurar y desplegar

aplicaciones en diversos lenguajes de programación, como Java, Kotlin, Groovy, Scala y otros.

Algunas de las características destacadas incluyen [35]:

- Editor de código inteligente: Proporciona resaltado de sintaxis, finalización de código, navegación rápida entre clases y métodos, refactorización de código y detección de errores en tiempo real.

- Depuración avanzada: Permite depurar el código paso a paso, establecer puntos de interrupción, inspeccionar variables y analizar el flujo de ejecución para encontrar y corregir errores.

- Administración de proyectos: Facilita la gestión de proyectos, la estructura de directorios, la configuración de dependencias y la integración con sistemas de control de versiones como Git.

- Soporte de frameworks y tecnologías: IntelliJ IDEA ofrece soporte nativo para una amplia variedad de frameworks y tecnologías, como Spring, Hibernate, Android, JavaFX, Maven, Gradle, entre otros.

- Integración con herramientas de desarrollo: IntelliJ IDEA se integra con otras herramientas populares, como el sistema de control de versiones Git, el sistema de construcción Maven y Gradle, servidores de aplicaciones y entornos de desarrollo como Docker y Kubernetes.

- Productividad y automatización: Ofrece una variedad de atajos y funciones de automatización, como generación de código, refactorización automática, completado de código inteligente y plantillas personalizables, que agilizan el desarrollo y ahorran tiempo.

Por todas estas características, que nos facilitan el trabajo, tomamos Intellij IDEA como nuestro entorno de desarrollo integrado.

4.1.13. Curl

Curl es una herramienta de línea de comandos utilizada para transferir datos a través de varios protocolos, como HTTP, HTTPS, FTP, entre otros. [36] Es una herramienta muy versátil y ampliamente utilizada en el desarrollo web y en la administración de sistemas.

Con curl se pueden realizar solicitudes a servidores web y obtener respuestas en forma de datos, como HTML, JSON, XML, etc. Puedes especificar diferentes opciones y

parámetros en la línea de comandos para personalizar las solicitudes, como encabezados HTTP, autenticación, métodos de solicitud, y otros.

4.1.14. Postman

Postman es una herramienta de desarrollo de API que permite a los desarrolladores probar, documentar y realizar solicitudes a servicios web. Es una aplicación de escritorio y una plataforma en línea que proporciona una interfaz gráfica de usuario intuitiva para enviar solicitudes HTTP a diferentes endpoints de API y recibir respuestas [37].

Postman nos facilita el realizar peticiones curl desde una interfaz de usuario accesible y por ello aparecerá en la muestra de resultados.

4.2. Diseño y arquitectura del sistema

Aplicando los componentes de diseño seleccionados presentamos la solución al sistema especificado que satisface las siguientes funcionalidades:

- El despliegue de un contrato inteligente en Solidity encargado de la emisión, transferencia y consulta de certificados energéticos dentro de la blockchain de Ethereum en un nodo local de Ganache.

- La implementación en Java de una API REST con Spring que se encarga, utilizando Web3j, de la interacción del usuario con el nodo de blockchain y los métodos de nuestro contrato inteligente desplegado.

- La implementación de una vista web HTML encargada de listar todos los certificados almacenados en la plataforma.

En el diagrama de la Figura 6, se representa la interacción e integración de los diferentes componentes y funcionalidades descritos anteriormente:

- El Navegador es el cliente desde donde el usuario accede a la aplicación web a través de un navegador web.

- Las peticiones HTTP se lanzan sobre el API REST, que proporciona un conjunto de puntos de acceso a través de los cuales servicios externos pueden comunicarse con la aplicación y realizar operaciones.

- La Aplicación Web es el núcleo de la aplicación que coordina las solicitudes del cliente y contiene la lógica de interacción mediante Web3j con el contrato inteligente desplegado en el nodo de la red blockchain.

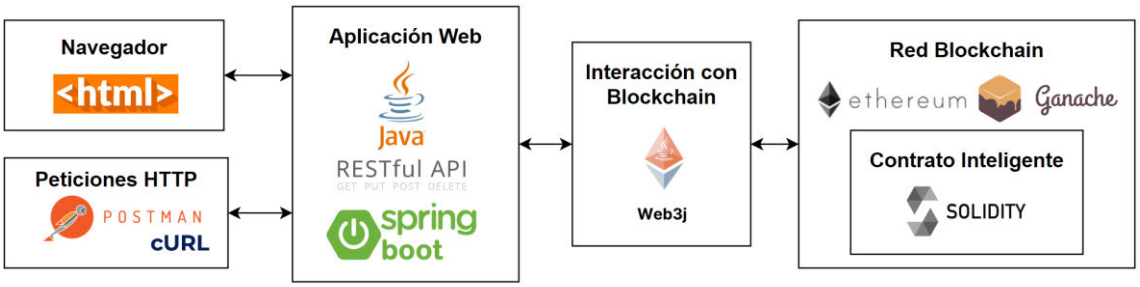

Fig. 6. Arquitectura del sistema por componentes.

4.3. Implementación del contrato inteligente

El contrato creado representa un registro de certificados dentro de la blockchain donde, cada variable de tipo *struct* declarada en la Figura 7, representa la información relativa a cada certificado de energía.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract CertificadorEnergetico{
    struct Certificado {
        address titular;
        uint256 idTecnologiaEmpleada;
        uint256 factorEmisionCO2;
        string etiqueta;
        bool transferido;
        ...
    mapping(bytes32 => Certificado) public certificados;
```

Fig. 7. Declaración del contrato y certificados en Solidity.

Un certificado nuevo se genera haciendo uso de la funcionalidad emitir certificado con la información o parámetros de entrada correspondientes, tal cual se muestra en la Figura 8. Este método genera una clave única por certificado, recibe un etiquetado energético correspondiente a sus datos, y almacena el certificado en una lista de certificados con la clave como identificador de posición.

```
function emitirCertificado(...) public {

    bytes32 certificadoId = keccak256(abi.encodePacked(msg.sender,
block.timestamp, block.difficulty));

    string memory etiqueta = calcularEtiqueta(factorEmisionCO2,
idTecnologiaEmpleada);

    certificados[certificadoId] = Certificado(msg.sender,
idTecnologiaEmpleada, factorEmisionCO2, etiqueta, false, ...);

    emit CertificadoEmitido(certificadoId, msg.sender, etiqueta);
}
```

Fig. 8. Funcionalidad emitir certificado en Solidity.

La función encargada de etiquetar la energía, declarada en la Figura 9, devuelve una letra de la A a la G que referencia al factor de relación entre la de cantidad de emisiones emitidas por la entidad comercializadora y el promedio nacional.

```
function calcularEtiqueta(...) internal pure returns (string memory) {

    uint256 factorRelacion = (factorEmisionCO2 * 100 /
obtenerfactorEmisionCO2Promedio(idTecnologiaEmpleada));

    if (factorRelacion < 35) {
        return "A";
    } else if (factorRelacion >= 35 && factorRelacion < 65) {
        return "B";
    }
    ...
    else {
        revert("Factor de relacion de emisionesC02 invalido");
    }
}
```

Fig. 9. Cálculo del etiquetado eléctrico en Solidity.

Los márgenes proporcionales del método anterior y el cálculo del factor de relación han sido extraídos del artículo del BOE, donde encontramos la información plasmada en la Figura 10.

Las bandas de fluctuación correspondientes a cada letra serán las siguientes:

A: $E_A / E_{NAL} < 0,35$

B: $0,35 \leq E_A / E_{NAL} < 0,65$

C: $0,65 \leq E_A / E_{NAL} < 0,95$

D: $0,95 \leq E_A / E_{NAL} < 1,05$

E: $1,05 \leq E_A / E_{NAL} < 1,35$

F: $1,35 \leq E_A / E_{NAL} < 1,65$

G: $1,65 \leq E_A / E_{NAL}$

Siendo:

E_A: Emisiones de CO_2 equivalente (en gramos equivalentes de CO_2 g/kWh) asociadas a la comercializadora A.

E_{NAL}: Emisiones de CO_2 equivalente (en gramos equivalentes de CO_2 g/kWh) asociadas al conjunto de la generación eléctrica del Sistema Eléctrico Español.

Fig. 10. Cálculo del etiquetado eléctrico BOE [11] .

Dado que las emisiones de CO2 en la producción de energías renovables es nula, para calcular el factor promedio de emisiones mencionado pasamos al análisis de las emisiones indirectas asociadas con otros factores de la producción, eficiencia, transporte y el ciclo de vida de los equipos utilizados en las energías renovables.

Seguimos este proceso con el objetivo de tener un valor promedio consistente con el que poder simular diferentes etiquetados de energía en relación con la información de la comercializadora recibida.

En relación con la documentación referenciada, damos una idea aproximada de las emisiones indirectas asociadas a algunas tecnologías renovables usadas en el proyecto:

- Energía solar fotovoltaica: Las emisiones indirectas asociadas con la producción y el ciclo de vida de los paneles solares pueden oscilar entre 20 y 40 gramos de CO2 por kilovatio-hora (gCO2/kWh) de energía generada [38]. Estas emisiones están relacionadas principalmente con la fabricación de los paneles, incluyendo la extracción y procesamiento de los materiales utilizados, así como la energía necesaria durante la producción.

- Energía eólica: Las emisiones indirectas de la energía eólica también están principalmente relacionadas con la fabricación de los aerogeneradores y la infraestructura asociada. Se estima que estas emisiones pueden oscilar entre 10 y 20 gCO2/kWh de energía generada [39].

- Energía hidroeléctrica: Las emisiones indirectas de la energía hidroeléctrica están relacionadas con la construcción de presas y embalses, así como con la gestión del agua. Sin embargo, en comparación con otras tecnologías renovables, las emisiones indirectas de la energía hidroeléctrica tienden a ser muy bajas, oscilando entre 20 y 30 gCO2/kWh [40].

- Producción de biomasa: Las emisiones indirectas de la biomasa sólida están relacionadas con la producción y el procesamiento de la biomasa utilizada como combustible. Esto incluye actividades como la tala, transporte, desbroce y secado de la biomasa. Estas emisiones varían según el tipo de biomasa y las prácticas de gestión utilizadas. En general, se estima que las emisiones indirectas de la producción de biomasa sólida pueden oscilar entre 20 y 100 gCO2/kWh [41].

- Producción de biogás: Las emisiones indirectas del biogás están relacionadas con la producción de los materiales orgánicos utilizados para la generación de biogás, como residuos agrícolas, residuos de alimentos o lodos de depuradoras. Estas emisiones dependen de las prácticas de gestión de residuos utilizadas en general, la producción de biogás a partir de residuos orgánicos puede reducir las emisiones de metano, un potente gas de efecto invernadero, que de otra manera se liberaría durante la descomposición natural de los residuos. Sin embargo, las emisiones indirectas asociadas con la producción de biogás pueden oscilar entre 10 y 100 gCO2/kWh [42].

La Figura 11 muestra parte de los factores promedio de emisiones de CO2 por cada tecnología, declarados en el contrato como constantes. Estos factores son necesarios en el cálculo de la certificación en función de la tecnología seleccionada.

```
// Factor de emisión en gramos CO2 por kWh promedio, valor fijo
uint256 public constant factorEmisionCO2PromedioEolica = 15;
uint256 public constant factorEmisionCO2PromedioFotovoltaica = 30;
...
function obtenerfactorEmisionCO2Promedio(uint256 idTecnologiaEmpleada)
public pure returns (uint256) {
    if (idTecnologiaEmpleada == 1) {
        return factorEmisionCO2PromedioEolica;
    } else if (idTecnologiaEmpleada == 2) {
        return factorEmisionCO2PromedioFotovoltaica;
    }
    ...
     else {
        revert("Identificador de tecnologia empleada invalido");
    }
}
```

Fig. 11. Obtención de factores de emisiones promedio en Solidity.

Una vez los certificados se hayan generado obteniendo el etiquetado correspondiente, podrán transferirse a otro usuario siempre y cuando quien realice la transferencia sea el titular del certificado y este no haya sido transferido anteriormente. En la Figura 12 se muestran estas restricciones de uso.

La llamada a este método emite un evento con los valores del nuevo titular e indica que el contrato se ha transferido con éxito.

```
function transferirCertificado(...) public {
    require(certificados[certificadoId].titular == msg.sender,
"No eres el propietario del certificado");

    require(!certificados[certificadoId].transferido,
"El certificado ya ha sido transferido");

    certificados[certificadoId].titular = nuevoTitular;
    certificados[certificadoId].transferido = true;

    emit CertificadoTransferido(nuevoTitular, true);
}
```

Fig. 12. Funcionalidad transferir certificado en Solidity.

Conociendo la clave que identifica a cada certificado, podemos consultar la información contenida haciendo una llamada a la funcionalidad de consulta de la Figura 13, que devolverá un evento con toda la información correspondiente.

```
function consultarCertificado(bytes32 certificadoId) public {
    Certificado memory certificado = certificados[certificadoId];
    require(certificado.titular != address(0), "El certificado no existe");

    emit ConsultarCertificado(...);
}
```

Fig. 13. Funcionalidad consultar certificado en Solidity.

4.4. Implementación de la Aplicación

Para crear la aplicación en Java hacemos uso del inicializador que nos proporciona Spring que permite generar rápidamente el esqueleto de un proyecto con la configuración inicial necesaria. Spring nos proporciona una interfaz intuitiva para seleccionar las dependencias, el lenguaje de programación, la versión de Spring Boot y otras configuraciones básicas del proyecto.

Con Spring Initializr personalizamos nuestro proyecto como se ve en la Figura 14. Seleccionamos Gradle como herramienta de construcción del proyecto, la versión 3.1.0 de Spring Boot, el lenguaje de programación Java versión 17 y el paquete WAR. Por último en el listado de dependencias predefinidas añadimos Spring Web que nos proporciona características para el desarrollo de aplicaciones web como la gestión de solicitudes HTTP.

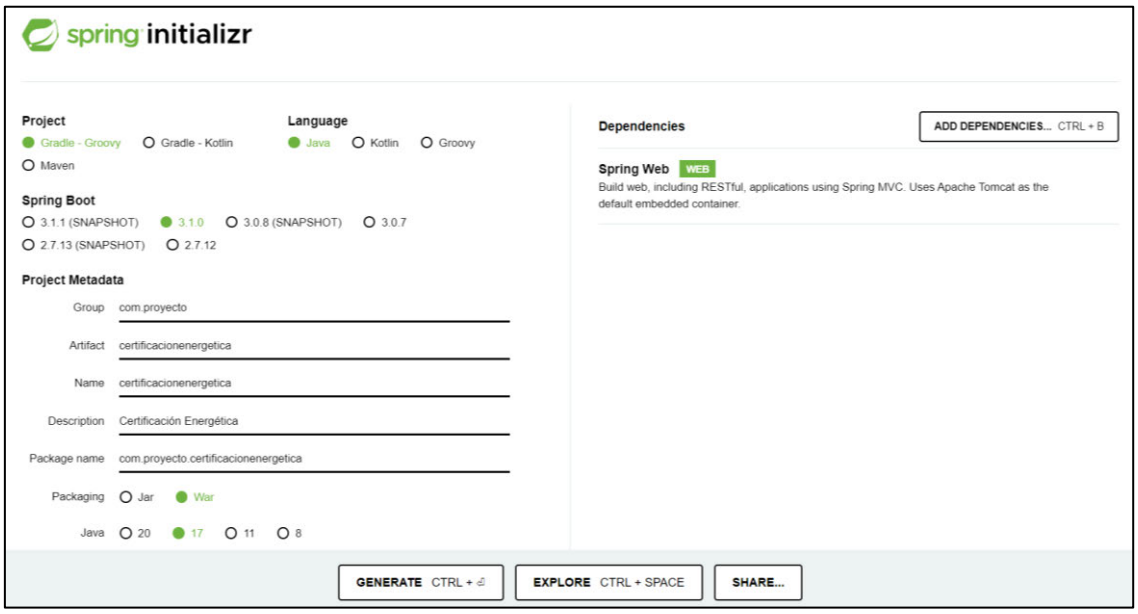

Fig. 14. Spring Initializr [43].

Esta herramienta nos permite generar el proyecto en diferentes formatos, incluyendo un archivo ZIP comprimido o un archivo de configuración en formato JSON. En nuestro caso descargamos un archivo ZIP.

Abrimos el directorio de archivos descargado en Intellij y basandonos en la arquitectura MVC del flujo de trabajo de Spring Framework, creamos el directorio mostrado en la Figura 15.

```
src/
└── main/
    ├── java/
    │   └── com/
    │       └── proyecto/
    │           └── certificacionenergetica/
    │               ├── controller/
    │               │   ├── WebController.java
    │               │   └── ApiController.java
    │               ├── service/
    │               │   └── ApiService.java
    │               ├── model/
    │               │   └── CertificadoModel.java
    │               ├── repository/
    │               │   ├── CertificadosRepository.java
    │               │   └── CertificadosRepositoryImp.java
    │               └── CertificacionEnergeticaApplication.java
    └── resources/
        ├── application.properties
        └── templates/
            └── certificados.html
build.gradle
```

Fig. 15. Árbol de carpetas del proyecto.

En la Figura 16 se muestra un diagrama UML que representa toda la arquitectura de la aplicación. En él se simbolizan los componentes principales y sus correspondientes relaciones, los métodos y atributos de cada clase, interfaces e implementaciones, y se anotan usos y dependencias externas.

Con la arquitectura de la aplicación presente, pasamos a describir el desarrollo de todos los componentes principales y sus relaciones en tres apartados diferentes:

1. Configuración de la aplicación, donde explicamos las dependencias externas del proyecto, las funcionalidades predeterminadas para un correcto arranque de la plataforma, y el proceso de integración del contrato inteligente, anteriormente descrito, en la aplicación.

2. Integración del contrato inteligente, donde generamos una clase Java que hace referencia a nuestro contrato inteligente.

3. Implementación del API REST, en este apartado describimos detalladamente los componentes que toman parte en el desarrollo de la API.

4. Implementación de la web, aquí identificamos los componentes necesarios para mostrar una vista HTML en el navegador.

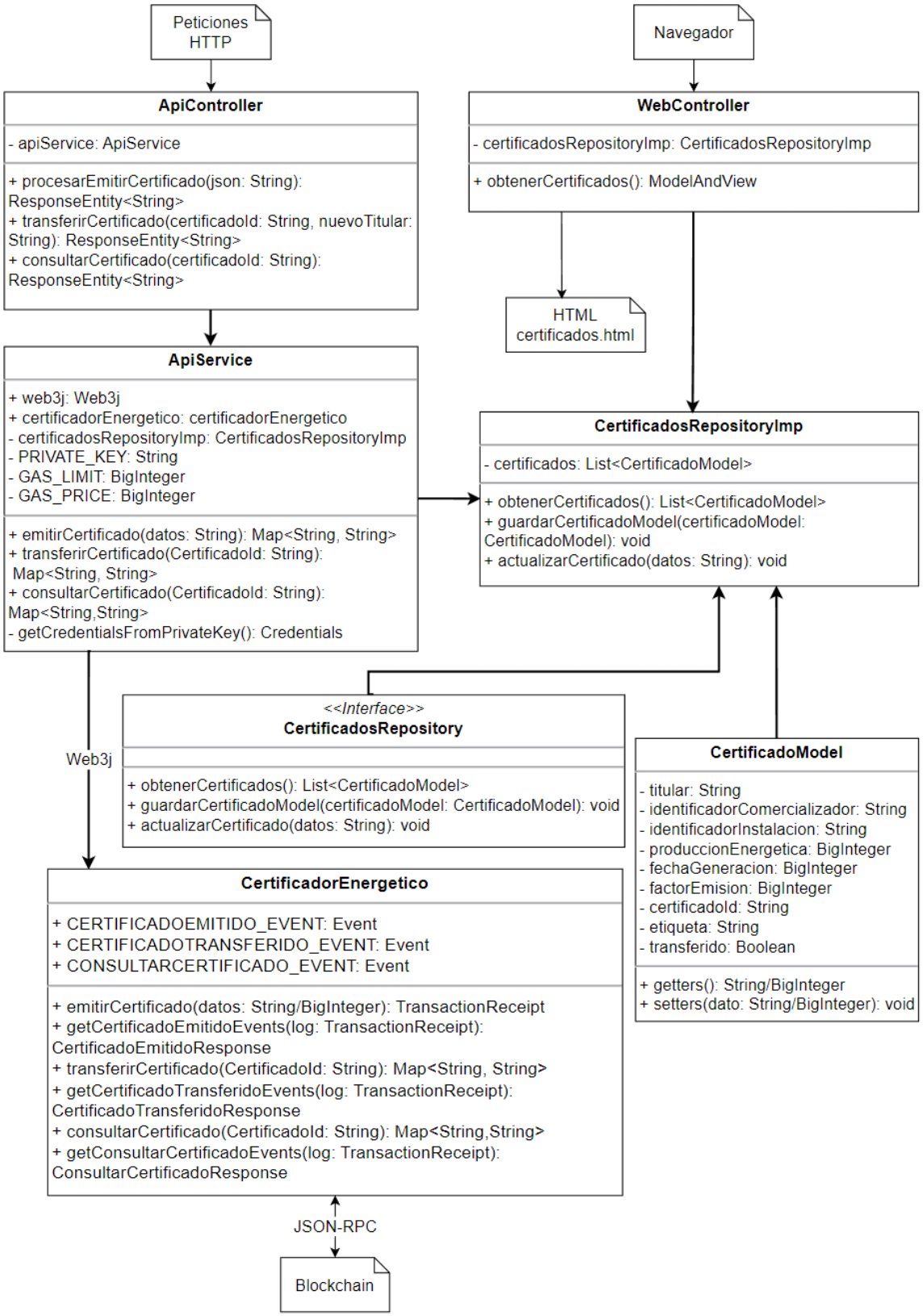

Fig. 16. Diagrama UML de la aplicación.

4.4.1. Configuración de la aplicación

4.4.1.1. Dependencias

Añadimos en el archivo build.gradle las dependencias adicionales que usamos en el proyecto.

En la Figura 17, la dependencia org.web3j:core:3.5.0 hace referencia a la biblioteca Web3j que proporciona las características y funcionalidades para interactuar con la red blockchain.

La dependencia com.fasterxml.jackson.core:jackson-databind:2.13.0 se refiere a la biblioteca Jackson Databind, que es una biblioteca de Java utilizada para convertir objetos Java en formato JSON.

Con la dependencia org.springframework.boot:spring-boot-starter-thymeleaf se integra Thymeleaf, un motor de plantillas Java basado en XML/HTML que se utiliza para generar las vistas web en la aplicación.

```
dependencies {
   implementation 'org.web3j:core:3.5.0'
   implementation 'com.fasterxml.jackson.core:jackson-
                   databind:2.13.0'
   implementation 'org.springframework.boot:spring-boot-starter-
                   thymeleaf'
}
```

Fig. 17. Dependencias del proyecto en el archivo build.gradle

En el archivo application.properties, representado en la Figura 18, añadimos la configuración de el prefijo y el sufijo de las vistas de la aplicación web.

```
spring.mvc.view.prefix=/templates/
spring.mvc.view.suffix=.html
```

Fig. 18. Configuración del prefijo y sufijo de las vistas web.

El prefijo hace referencia a la carpeta donde se encuentran las vistas HTML dentro del directorio raíz de la aplicación, y el sufijo establece que las vistas tienen una extensión .html.

4.4.1.2. Configuración de inicio

La clase principal de la aplicación se encuentra en el directorio con la anotación @SpringBootApplication. Esta clase contiene el método main que se ejecuta cuando la aplicación se inicia.

El inicio de la aplicación tiene, por funcionalidad del sistema, dependencia con la conexión y el despliegue del contrato inteligente dentro del nodo blockchain. Por esta

razón se invoca en esta clase el método de la Figura 19 que realiza una conexión con el nodo y despliega el contrato inteligente. La anotación @PostConstruct de la función, marca que este método debe ejecutarse inmediatamente después de que la aplicación se haya iniciado principal garantizando que la conexión esté lista y disponible cuando la aplicación comience a operar.

```
@SpringBootApplication
public class CertificacionEnergeticaApplication implements WebMvcConfigurer
{
   ...
   public static void main(String[] args) {
      SpringApplication.run(CertificacionEnergeticaApplication.class,
args);
   }
   @PostConstruct
   public void inicializarBlockchain() {
       apiService.establecerConexionBlockchain();
       apiService.desplegarSmartContract();
   }
```

Fig. 19. Declaración de la clase principal de la aplicación.

4.4.2. Integración del contrato inteligente en la aplicación

Una de las características que nos ofrece Web3j es la generación automática de clases de Java a partir de contratos inteligentes escritos en Solidity. Esto simplifica la interacción con contratos inteligentes, ya que proporciona una interfaz de programación fuertemente tipada y autocompletada para invocar funciones y leer las variables del contrato. Para generar esta clase en Java necesitamos compilar el contrato en Solidity con la herramienta solc-windows y obtener los archivos bin y abi como se muestra en la Figura 20.

```
> solc-windows .\src\main\resources\solidity\CertificadorEnergetico.sol --
bin --abi --optimize -o .\src\main\resources\solidity\out


Compiler run successful. Artifact(s) can be found in directory
".\src\main\resources\solidity\out".
```

Fig. 20. Compilar contrato inteligente por CLI usando solc-windows.

Con los archivos bin y abi ya disponibles lanzamos Web3j por la línea de consola como se ve en la Figura 21. Tras realizar esta acción obtenemos, en el directorio indicado, la clase en Java que instancia a nuestro contrato inteligente.

```
> web3j solidity generate
.\src\main\resources\solidity\out\CertificadorEnergetico.bin
.\src\main\resources\solidity\out\CertificadorEnergetico.abi -o
.\src\main\java -p com.proyecto.certificacionenergetica

Generating com.proyecto.certificacionenergetica.CertificadorEnergetico ...
File written to .\src\main\java
```

Fig. 21. Generar clase Java con Web3j del contrato en Solidity.

4.4.3. Implementación del API REST

En la implementación del API seguimos el flujo de trabajo plasmado en el diagrama UML de la figura 16, donde se realizan las siguientes acciones:

1. Se recogen las solicitudes HTTP del usuario en la clase controlador del API.

2. Estas peticiones invocan los servicios necesarios para procesar la lógica que interacciona con el contrato dentro de blockchain.

3. Con cada petición realizada el servicio se comunica con el repositorio para acceder y manipular los datos almacenados en la capa de persistencia. El repositorio se comporta como una base de datos en nuestra aplicación.

4. Las clases modelo se utilizan para representar y transferir los datos a través de las diferentes capas de la aplicación.

5. Se genera una respuesta especifica a la petición del usuario.

4.4.3.1. Controlador API

La clase que define el controlador API es ApiController.java que maneja las solicitudes y las respuestas HTTP.

En la Figura 22 la anotación @RestController indica que sus métodos deben retornar directamente objetos que serán serializados en el cuerpo de la respuesta HTTP.

```java
@@RestController
@RequestMapping("/api")
public class ApiController {
    @Autowired
    private ApiService apiService;
```

Fig. 22. Declaración de la clase Controlador API.

La anotación @RequestMapping se utiliza para mapear una URI o una parte de ella a un método específico dentro de un controlador. Con esta anotación colocada en la clase establecemos un prefijo común para todas las rutas definidas en los métodos dentro del

controlador y con la anotación establecida en los métodos definimos la ruta específica que debe coincidir para que se ejecute ese método en particular.

En esta clase aparecen declaradas las tres funcionalidades principales de nuestra plataforma; emitir certificados energéticos, transferir certificado y consultar certificado

Dado que el flujo de trabajo y la lógica que implementan estos tres métodos es similar, pasamos a describir únicamente el proceso de emitir certificados.

El método que procesa el emitir contrato de la Figura 23, procesa una operación de solicitud tipo POST que se mapea en la ruta /emitirCertificado.

```
@PostMapping("/emitirCertificado")
public ResponseEntity<String> procesarEmitirContrato(@RequestBody String
json) throws JsonProcessingException {
    try {
        CertificadoModel certificadoModel = objectMapper.readValue(json,
CertificadoModel.class);

        Map<String, String> resultadoEvento = apiService.emitirCertificado(
                certificadoModel.getIdentificadorComercializador(),
                certificadoModel.getIdentificadorInstalacion(),
                certificadoModel.getProduccionEnergetica(),
                certificadoModel.getIdTecnologiaEmpleada(),
                certificadoModel.getFechaGeneracion(),
                certificadoModel.getFactorEmision());

        // Convertir el mapa a una cadena JSON
        String jsonRespuesta =
objectMapper.writeValueAsString(resultadoEvento);

        // Devolver el JSON como ResponseEntity
        return ResponseEntity.ok(jsonRespuesta);

    } catch (Exception e) {
        String errorMessage = "Error al procesar emitirContrato(): " +
e.getMessage();
        return ResponseEntity.badRequest().body(errorMessage);
    }
}
```

Fig. 23. petición POST emitir certificado procesada por el controlador api.

Este método recibe un objeto JSON como entrada en el cuerpo de la solicitud utilizando la anotación @RequestBody. Utilizamos la biblioteca ObjectMapper para convertir la cadena JSON en un objeto de tipo CertificadoModel.

A continuación, se llama al servicio que será el componente encargado de interactuar con la blockchain de Ethereum para procesar la emisión del certificado. El resultado de esta llamada se convierte en una cadena JSON utilizando el ObjectMapper de nuevo.

Finalmente, la respuesta del API devuelve un formato JSON utilizando ResponseEntity una clase del framework de Spring que representa una respuesta HTTP.

De forma similar al método que procesa la emisión de certificados responden el resto de peticiones controladas en el sistema:

- Consultar certificado: es una operación de solicitud GET que recibe un certificadoId y devuelve toda la información relativa al certificado.

- Transferir certificado: es una operación de solicitud POST que invoca el propietario de un certificado específico para transferírselo a otro usuario obteniendo la dirección del nuevo titular y un indicativo de que el certificado se ha transferido con éxito.

4.4.3.2. Servicio API

La case que implementa los servicios del API es ApiService.java que describe la lógica que interacciona con el contrato inteligente dentro del nodo blockchain local de Ganache.

Describimos los métodos, mencionados anteriormente en el apartado de configuración de inicio, que establecen la conexión al nodo y despliegan el contrato inteligente.

Al establecer la conexión como se muestra en la Figura 24; se crea, con funcionalidades de la librería Web3j, una conexión con el nodo de Ethereum a través de HTTP. Al no especificar URL se establece la conexión predeterminada con el nodo que se está ejecutando en la máquina en el puerto 8545.

```
public void establecerConexionBlockchain(){
    web3j = Web3j.build(new HttpService());

    try {
        System.out.println("Conexión a la blockchain establecida. Versión
del cliente: " + web3j.web3ClientVersion().send().getWeb3ClientVersion());
    } catch (Exception e) {
        System.err.println("Error al establecer la conexión a la
blockchain: " + e.getMessage());
    }
}
```

Fig. 24. Conexión con el nodo blockchain.

En la Figura 25 se muestra el despliegue del contrato inteligente dentro de nuestro nodo.

```
public void desplegarSmartContract(){
    try {
        certificadorEnergetico = CertificadorEnergetico.deploy(web3j,
getCredentialsFromPrivateKey(), GAS_PRICE, GAS_LIMIT).send();

        System.out.println("Contrato desplegado correctamente. Dirección
del contrato: " + certificadorEnergetico.getContractAddress());

    } catch (Exception e) {
        System.err.println("Error al desplegar el contrato: " +
e.getMessage());
    }
}
```

Fig. 25. Despliegue del contrato inteligente.

Para realizar este despliegue necesitamos conocer u obtener los siguientes componentes:

- Una instancia de la clase que representa nuestro contrato en Solidity.

- Una clave privada: La clave privada es una pieza fundamental en la criptografía asimétrica utilizada en la blockchain. Es un número secreto que se utiliza para firmar transacciones y demostrar la propiedad y autorización de una dirección en la blockchain. La clave privada debe mantenerse en secreto y nunca debe ser compartida públicamente, ya que cualquier persona que tenga acceso a la clave privada puede acceder a los activos asociados a la dirección correspondiente.

- Unas credenciales generadas a partir de esta clave privada como se ve en la Figura 26.

```
private Credentials getCredentialsFromPrivateKey() {
    return Credentials.create(PRIVATE_KEY);
}
```

Fig. 26. Generar credenciales desde la clave privada.

- El límite de gas: En la blockchain, el gas se utiliza para medir el costo computacional necesario para ejecutar una operación o una transacción. El límite de gas es la cantidad máxima de gas que se asigna a una transacción específica. Este límite define la cantidad máxima de operaciones o instrucciones que pueden ejecutarse dentro de una transacción. Si una transacción consume más gas del límite establecido, se cancela y se revierten todos los cambios realizados hasta ese punto.

- El precio del gas: El precio del gas es el costo en unidades de criptomoneda que se paga por cada unidad de gas utilizado en una transacción. Este precio determina la prioridad de una transacción en la red blockchain. Cuanto mayor sea el precio del gas, más incentivo hay para que los mineros incluyan la transacción en un bloque rápidamente. Los precios del gas varían según la congestión de la red y las preferencias del remitente de la transacción.

Los valores de clave privada, límite de gas y precio del gas, los podemos obtener directamente de la interfaz de Ganache.

Una vez explicados el inicio de la aplicación, la conexión al nodo y el despliegue del contrato, pasamos a describir el resto de los métodos funcionales del servicio.

Las funciones del servicio junto con la instancia en Java de nuestro contrato inteligente actúan sobre nuestro contrato en Solidity desplegado en Ganache, en la Figura 27 se muestra un ejemplo de estas llamadas.

```
certificadorEnergetico.emitirCertificado(
        identificadorComercializador,
        identificadorInstalacion,
        produccionEnergetica,
        idTecnologiaEmpleada,
        fechaGeneracion,
        factorEmision).send();
```

Fig. 27. Instancia de la llamada emitir certificado.

Después de realizar la llamada al contrato desplegado en el nodo, cada uno de los métodos del servicio crea un evento y un filtro con el objetivo de capturar la información emitida dentro de blockchain.

En primer lugar, para capturar dicha información necesitamos subscribirnos al evento correspondiente como se muestra en la Figura 28 donde se referencia a uno de los tres eventos creados en el contrato.

```
Event certificadoEvent = certificadorEnergetico.CERTIFICADOEMITIDO_EVENT;
String certificadoEventSignature = EventEncoder.encode(certificadoEvent);
```

Fig. 28. Creación de un evento.

En segundo lugar, al aplicar un filtro nos aseguramos de que la información que recogemos es correspondida y no debida a otra posible interacción con la API de un tercer usuario.

En la Figura 29 creamos un filtro en función de un rango de bloques de la cadena, del contrato desplegado y de la firma del evento codificada.

```
EthFilter filter = new EthFilter(
        DefaultBlockParameter.valueOf(startBlockNumber),
        DefaultBlockParameterName.LATEST,
        Arrays.asList(certificadorEnergetico.getContractAddress())
).addSingleTopic(certificadoTransferidoEventSignature);
```

Fig. 29. Creación de un filtro.

En tercer lugar, después de crear el evento y el filtro, pasamos a analizar e iterar los logs de la red como se muestra en la figura 30. Los logs son los registros de eventos generados por las transacciones y nuestro contrato inteligente que se ejecuta en la red.

```
 EthLog ethLog = web3j.ethGetLogs(filter).send();

List<EthLog.LogResult> logs = ethLog.getLogs();

if (logs != null) {
    for (EthLog.LogResult logResult : logs) {
        if (logResult instanceof EthLog.LogObject) {
            EthLog.LogObject log = (EthLog.LogObject) logResult;
            List<String> topics = log.getTopics();;

            if (!topics.isEmpty() &&
topics.get(0).equals(certificadoEventSignature)) {
                List<String> data = log.getTopics();
                String eventData = data.get(0);
                ...
                TransactionReceipt logReceipt = new TransactionReceipt();
                logReceipt.setLogs(Collections.singletonList(log));

                 CertificadorEnergetico.CertificadoTransferidoEventResponse
                 evento =

                 certificadorEnergetico.getCertificadoEvents(logReceipt).get
                 (0);
```

Fig. 30. Proceso de filtrado de eventos.

En cuarto y último lugar, obtenemos la referencia a todas las variables que se emiten en el contrato y desde donde ahora podemos obtener sus valores, como se representa en la Figura 31.

```
String variableContrato1 = evento.variableContrato1;
Boolean variableContrato2 = evento.variableContrato2;
```

Fig. 31. Variables de un evento filtrado.

Los eventos en la aplicación se manejan de manera **asíncrona** debido a la propia naturaleza asincrónica de las operaciones en la cadena de bloques Ethereum.

Cuando se emite un evento desde un contrato inteligente en Ethereum, el contrato genera un registro en la cadena de bloques que contiene la información del evento. Para acceder a esos registros o logs desde una aplicación cliente, se requiere la espera de la respuesta de la red.

Para permitir un manejo eficiente de las operaciones asíncronas, Web3j utiliza objetos CompletableFuture. Un CompletableFuture es una clase en Java que representa un resultado futuro de una operación asincrónica. Proporciona métodos para gestionar y manipular el resultado una vez que está disponible.

Dado que nuestra aplicación devuelve en las respuestas de las peticiones HTTP del API los resultados de estos eventos, debemos utilizar objetos CompletableFuture que esperen a capturar dichos resultados antes de procesar la respuesta de la petición como se ve en la Figura 32.

```
variableContrato1Future.complete(variableContrato1);
variableContrato2Future.complete(variableContrato2);
```

Fig. 32. Variables finales de un evento asíncrono.

4.4.3.3. Modelo

La clase CertificadoModel.java representa un objeto tipo Certificado (Figura 7) en la aplicación web, conteniendo sus correspondientes variables, getters y setters como se muestra en la Figura 33.

```
public class CertificadoModel {

    private String titular;
    private String certificadoId;
    ...
    public CertificadoModel() {}

    public String getTitular() {
        return titular;
    }
    public void setTitular(String titular) {
        this.titular = titular;
    }
    public String getCertificadoId() {
        return certificadoId;
    }
    public void setCertificadoId(String certificadoId) {
        this.certificadoId = certificadoId;
    }
    ...
```

Fig. 33. Modelo del certificado.

En la plataforma, los modelos se utilizan para el intercambio de información entre las diferentes capas; controlador, servicio y repositorio.

Aunque podría implementarse, no se realiza ningún tipo de validación de los datos antes de ser ingresado o almacenados en el repositorio.

4.4.3.4. Repositorio

El paquete repository contiene una clase CertificadosRepositoryImp.java, que hace las veces de capa de persistencia para acceder a los datos de los certificados almacenados en la plataforma.

La creación de la interfaz CertificadosRepository no es estrictamente necesaria, pero es una buena práctica en el desarrollo de software para separar la capa de acceso a datos de la capa de negocio. Utilizar una interfaz proporciona una abstracción adicional y facilita la flexibilidad y la mantenibilidad del código [44].

En esta clase almacenamos una lista de objetos tipo CertificadoModel y proporcionamos en la Figura 34 los métodos y consultas necesarios para realizar operaciones de lectura y escritura en el repositorio.

```
@Repository
public class CertificadosRepositoryImp implements CertificadosRepository{
    private List<CertificadoModel> certificados;
    ...
    @Override
    public CertificadoModel obtenerCertificado(String certificadoId) {
        for (CertificadoModel certificado : certificados) {
            if (certificado.getCertificadoId().equals(certificadoId)) {
                return certificado;
            }
        }
        return null;
    }

    @Override
    public void guardarCertificadoModel(CertificadoModel certificadoModel)
{
        System.out.println("Certificado guardado con exito: " +
certificadoModel);
        certificados.add(certificadoModel);
    }

    @Override
    public void actualizarCertificadoModel(String certificadoId, String
nuevoTitular, Boolean transferido) {
        System.out.println("Certificado actualizado con exito");
        for (CertificadoModel certificado : certificados) {
            if (certificado.getCertificadoId().equals(certificadoId)) {
                certificado.setTitular(nuevoTitular);
                ...
```

Fig. 34. Implementación y métodos del repositorio.

4.4.4. Implementación de la vista web

En la implementación de la vista web los componentes añadidos actúan de forma similar al conjunto de clases en la API, siguiendo estos el siguiente proceso:

1. El usuario a través de solicitudes en el navegador invoca a el controlador web. Esta clase devuelve la representación HTML de una vista especifica.

2. La vista HTML, a través del controlador, interactúa con el repositorio proporcionando toda la información disponible en él, estableciendo unas normas y formato de visualización determinado.

4.4.4.1. Controlador Web

El controlador web definido en WebController.java maneja las solicitudes y las respuestas relacionadas con las vistas HTML y la interfaz de usuario de la aplicación web.

Esta clase, definida en la Figura 35, realiza lectura de los certificados guardados en lista en la capa de repositorio cuando el usuario navega a la URL especificada.

```
@Controller
public class WebController {
    private final CertificadosRepositoryImp certificadosRepositoryImp;
    public WebController(CertificadosRepositoryImp
certificadosRepositoryImp) {
        this.certificadosRepositoryImp = certificadosRepositoryImp;
    }
    @GetMapping("/certificados")
    public ModelAndView obtenerCertificados() {
        List<CertificadoModel> certificados =
certificadosRepositoryImp.obtenerCertificados();

        ModelAndView modelAndView = new ModelAndView("certificados");
        modelAndView.addObject("certificados", certificados);

    return modelAndView;
    }
}
```

Fig. 35. Implementación del controlador web.

El propósito principal del objeto ModelAndView es almacenar los datos que se van a mostrar en la vista y el nombre de la vista en sí. Proporcionando métodos para agregar datos al modelo y establecer el nombre de la vista.

Los certificados energéticos son mostrados al usuario en la web a través del documento HTML mostrado en la Figura 36 y donde se itera la información de la lista de certificados accesibles desde el repositorio. En esta vista, para favorecer la legibilidad de los datos, también se realiza la traducción del identificador de la tecnología empleada por el tipo de energía a la que se refiere.

```
<!DOCTYPE html>
<html lang="es" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Certificados</title>
  <style>
    ...
  </style>
</head>
<body>
<h1>Lista de Certificados</h1>
<table>
  <thead>
  <tr>
    <th>CertificadoID</th>
    <th>Titular</th>
    <th>Comercializador</th>
    <th>Instalación</th>
    <th>Produccion Energetica (kW)</th>
    <th>ID Tecnología empleada</th>
    ...
  </tr>
  </thead>
  <tbody>
  <tr th:each="certificado : ${certificados}">
    <td th:text="${certificado.certificadoId}"></td>
    <td th:text="${certificado.titular}"></td>
    <td th:text="${certificado.identificadorComercializador}"></td>
    <td th:text="${certificado.identificadorInstalacion}"></td>
    <td>
      <span th:if="${certificado.idTecnologiaEmpleada == 1}">Eolica</span>
      ...
  </tr>
  </tbody>
</table>
</body>
</html>
```

Fig. 36. Vista web HTML.

5. Resultados

5.1. Resultados de consola

Al iniciar la aplicación obtenemos los primeros resultados por consola, capturados en la Figura 37. En esta figura se muestra como obtenemos la versión del cliente web3 utilizado en la aplicación Java que utiliza la librería Web3j para interactuar con la red Ethereum, demostrando que se ha realizado la conexión con el nodo blockchain correctamente. También obtenemos la dirección que identifica a nuestro contrato inteligente dentro de la red, demostrando que se ha desplegado correctamente.

```
> Task :CertificacionEnergeticaApplication.main()

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.1.0)

2023-06-05T10:09:37.982+02:00  INFO 4720 --- [           main] c.p.c.CertificacionEnergeticaApplication
2023-06-05T10:09:37.988+02:00  INFO 4720 --- [           main] c.p.c.CertificacionEnergeticaApplication
2023-06-05T10:09:39.427+02:00  INFO 4720 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer
2023-06-05T10:09:39.441+02:00  INFO 4720 --- [           main] o.apache.catalina.core.StandardService
2023-06-05T10:09:39.441+02:00  INFO 4720 --- [           main] o.apache.catalina.core.StandardEngine
2023-06-05T10:09:39.587+02:00  INFO 4720 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]
2023-06-05T10:09:39.589+02:00  INFO 4720 --- [           main] w.s.c.ServletWebServerApplicationContext
Conexi♦n a la blockchain establecida. Versi♦n del cliente: EthereumJS TestRPC/v2.13.1/ethereum-js
Contrato desplegado correctamente. Direcci♦n del contrato: 0xe77e9e4a6b5d4e478777afd306606850004f6c45
2023-06-05T10:09:41.332+02:00  INFO 4720 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer
2023-06-05T10:09:41.344+02:00  INFO 4720 --- [           main] c.p.c.CertificacionEnergeticaApplication
```

Fig. 37. Respuesta consola al iniciar la aplicación.

En caso de error en la conexión con el nodo obtendríamos el mensaje mostrado en la Figura 38.

```
Error al establecer la conexi♦n a la blockchain: Failed to connect to localhost/[0:0:0:0:0:0:0:1]:8545
Error al desplegar el contrato: java.net.ConnectException: Failed to connect to localhost/[0:0:0:0:0:0:0:1]:8545
```

Fig. 38. Error en la conexión blockchain al iniciar la aplicación.

5.2. Resultados debidos a la interacción HTTP con la API.

Una vez la aplicación está corriendo pasamos a interactuar con el API utilizando la interfaz de usuario de Postman. Con las siguientes acciones: emisión, consulta y transferencia, comprobamos a nivel de usuario el correcto funcionamiento de las funcionalidades desarrolladas en la plataforma.

5.2.1. Emisión de certificados energéticos

Para emitir un certificado energético nuevo se introduce la URL http://localhost:8080/api/emitirCertificado con una petición POST y un JSON con la información relativa al certificado en el cuerpo, se muestra el fragmento de código de la petición curl, extraída de Postman en la Figura 39.

```
curl --location 'http://localhost:8080/api/emitirCertificado' \
--header 'Content-Type: application/json' \
--data '{
    "identificadorComercializador" : "Comercializador1",
    "identificadorInstalacion" : "Instalacion1",
    "produccionEnergetica" : 45000,
    "idTecnologiaEmpleada" : 1,
    "fechaGeneracion" : 2052023,
    "factorEmision" : 20
}
```

Fig. 39. Petición cURL en Postman: POST EmitirCertificado.

El resultado de la emision del certificado es un JSON con los parametros etiqueta y certificadoId, que simbolizan la catalogación de la energia producida y una clave que identifica al certificado respectivamente (Figura 40).

```
{"etiqueta":"E","certificadoId":"0x0ff5721a4c181a6afbe9c570091e42b85aa2283a
eadb38c6bedabd45ded30064"}
```

Fig. 40. Respuesta a petición EmitirCertificado en Postman.

Cuando se realiza la petición, se muestra por consola los logs emitidos por los eventos. En la Figura 41 podemos ver un ejemplo del log filtrado en apartados anteriores.

```
Log: Log{removed=false, logIndex='0x0', transactionIndex='0x0',
transactionHash='0xa676094a7abe17792e0c212215ca3e56e0aef467a785e41c88db2563
00666fcc',
blockHash='0x88cc3a88ba19eeb131a3fdc26453d06233fcb2427bb6981584da0b92f11aff
ff', blockNumber='0x144',
address='0x80f684ddfe8190d281ded8d1c29d98b131454a31',
data='0x0ff5721a4c181a6afbe9c570091e42b85aa2283aeadb38c6bedabd45ded30064000
0000000000000000000a80c9558848612940c32d803d68aeadce97373a400000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000', type='mined',
topics=[0x76bf287a6f02cd86f740fdb4d353815a75d9904e66982a4616419f480644b00c]
}
```

Fig. 41. Logs de registro emitido por un evento en blockchain.

También se muestran mensajes de la capa repositorio que informa que los datos del nuevo certificado se han guardado correctamente (Figura 42).

```
Certificado guardado con exito:
CertificadoModel{titular='0xa80c9558848612940c32d803d68aeadce97373a4',
identificadorComercializador='Comercializador1',
identificadorInstalacion='Instalacion1', produccionEnergetica=45000,
idTecnologiaEmpleada=1, fechaGeneracion=2052023, factorEmision=20,
certificadoId='0x0ff5721a4c181a6afbe9c570091e42b85aa2283aeadb38c6bedabd45de
d30064', etiqueta='E', transferido=false}
```

Fig. 42. Mensajes de guardado y actualizado de certificados en el repositorio.

5.2.2. Consulta de certificados energéticos

Una vez emitido el certificado y obtenida su clave podemos consultar la información de nuestro certificado enviando una petición GET como la que se ve en la Figura 43.

```
curl --
location 'http://localhost:8080/api/consultarCertificado/0x0ff5721a4c181a6a
fbe9c570091e42b85aa2283aeadb38c6bedabd45ded30064'
```

Fig. 43. Petición cURL en Postman: GET ConsultarCertificado.

El resultado de esta consulta es toda la información relativa al certificado en formato JSON, Figura 44.

```
{"etiqueta":"E","idTecnologiaEmpleada":"1","produccionEnergetica":"45000","
fechaGeneracion":"2052023","identificadorComercializador":"Comercializador1
","factorEmisionCO2":"20","transferido":"false","titular":"0xa80c9558848612
940c32d803d68aeadce97373a4","identificadorInstalacion":"Instalacion1"}
```

Fig. 44. Respuesta a petición ConsultarCertificado.

5.2.3. Transferir certificados energéticos

Teniendo el identificador clave del certificado y siendo titulares podemos transferir el certificado emitido a otro usuario. En la Figura 45 se transfiere el certificado anterior a otra de las cuentas que nos ofrece la interfaz de Ganache.

```
curl --location --
request POST 'http://localhost:8080/api/transferirCertificado/0x0ff5721a4c1
81a6afbe9c570091e42b85aa2283aeadb38c6bedabd45ded30064/nuevoTitular/0x6206CE
ca1172862653C912A39698e03e2257e502
```

Fig. 45. Petición cURL en Postman: POST TransferirCertificado.

La respuesta a la petición POST de transferencia (Figura 46) es un JSON con el nuevo titular del certificado.


```
{"nuevoTitular":"0x6206ceca1172862653c912a39698e03e2257e502","transferido":
"true"}
```


Fig. 46. Respuesta a petición TransferirCertificado.

5.2.4. Consulta de certificado actualizado tras transferencia

Una vez transferido el certificado se actualizan los datos de este en la capa del repositorio y podemos volver a realizar una consulta a la información verificando la información resaltada en la Figura 47 tras consulta.


```
{"etiqueta":"E","idTecnologiaEmpleada":"1","produccionEnergetica":"45000","
fechaGeneracion":"2052023","identificadorComercializador":"Comercializador1
","factorEmisionCO2":"20","transferido":"true","titular":"0x6206ceca1172862
653c912a39698e03e2257e502","identificadorInstalacion":"Instalacion1"}
```


Fig. 47. Respuesta a petición ConsultarCertificado tras actualización.

5.2.5. Análisis en los tiempos de respuesta

Recordamos que las peticiones de la API se hacen directamente al nodo de Ethereum y esperan hasta capturar los eventos que almacenan la información pertinente. Estas esperas mostradas en la Figura 48, tienen tiempos de acción bastante asumibles entre los 200 y 600 milisegundos, lo que no supone un problema de funcionalidad en el diseño.

Fig. 48. Tiempos de respuesta en las peticiones al API.

5.3. Resultados mostrados en la interfaz web

Añadimos algún certificado más a la plataforma vía API y desde la URL http://localhost:8080/certificados en nuestro navegador, podemos acceder a la visualización de una lista de certificados donde se nos muestra toda la información actualizada relativa a la actividad anterior, como se ve en la Figura 49.

Lista de Certificados

CertificadoID	Titular
0x0ff5721a4c181a6afbe9c570091e42b85aa2283aeadb38c6bedabd45ded30064	0x6206ceca1172862653c912a39698e03e2257e502
0xb622a9f51510cc5ff7337a005ab6c02d833b88265e3eff7b249905f234b8c9a4	0xa80c9558848612940c32d803d68aeadce97373a4
0x65746552664454c4936fc36de8d4d88f1cb8af50e906ed0d3f4840a909c0035f	0xa80c9558848612940c32d803d68aeadce97373a4
0xaaaabb3705d9c952f97d7aa6b1a926700cc6519b8523396d8ade00d5ed1cbd45	0xa80c9558848612940c32d803d68aeadce97373a4
0x2251ff7ca4365d0486dd87323fd7b6e78d2789c76184ecae83d08d9ab187cbf8	0xa80c9558848612940c32d803d68aeadce97373a4
0xfb36de22d5f8beeb3ffdd9b9450b64123da6d42fa1c242d0b5de621b838fac8f	0xa80c9558848612940c32d803d68aeadce97373a4

Fig. 49. Vista web: Lista de Certificados 1.

Más información acerca de la comercializadora de electricidad, la instalación, la tecnología empleada y otros aspectos relacionados con la electricidad y sus emisiones se muestran en la Figura 50. El etiquetado correspondiente se puede apreciar en la penúltima columna.

Comercializador	Instalación	Tecnología empleada	Produccion Energetica (kW)	Fecha de Generacion	Factor EmisionesCO2 (gramos CO2 por kWh)	Etiqueta	Transferido
Comercializador1	Instalacion1	Eolica	45000	2052023	20	E	true
Comercializador2	Instalacion2	Eolica	15000	2252023	10	C	false
Comercializador3	Instalacion3	Fotovoltaica	45000	15052000	40	E	false
Comercializador4	Instalacion4	Hidraulica	180000	19052015	10	G	false
Comercializador5	Instalacion5	Biomasa Solida	120000	11062020	38	E	false
Comercializador5	Instalacion5	Biogas	120000	14011996	30	A	false

Fig. 50. Vista web: Lista de Certificados 2.

6. Presupuesto

Para la realización de este proyecto no se ha necesitado la compra de ningún componente hardware ni software adicional. Se ha utilizado el equipo personal y todas las herramientas mencionadas en las especificaciones que son opciones gratuitas y/o de código abierto.

El salario aproximado de la mano de obra se obtiene de una relación entre el salario promedio de un programador Java anual (30000€) [45], y el salario de un programador de contratos inteligentes en Solidity (70000€) [46], ambos puestos sin experiencia y en España.

Conociendo un total de 228,57 días efectivamente trabajados al año, y una media de 8 horas trabajadas por día [47], el precio hora de nuestro salario estimado son 27,35 €h.

En la Tabla 1 desglosamos el presupuesto por horas que le corresponde a cada tarea.

Tabla 1. Presupuesto por tareas.

Tarea	**Duración en horas**	**Coste en €**
Análisis y documentación	10	275,3
Herramientas y arquitectura de diseño	15	410,25
Implementación del contrato inteligente	50	1367,5
Implementación del API	65	1777,75
Integración del contrato inteligente	70	1914,5
Implementación de la vista web	30	820,5
Pruebas de la aplicación	45	1230,75
Memoria	35	957,25
TOTAL	320	8752

Como puede apreciarse, el proyecto cuenta con una duración de 320 horas y un presupuesto total de 8752€.

7. Manual de usuario

Se lista un breve resumen de las instrucciones necesarias para poner en marcha la plataforma descrita:

1. Clonar el repositorio de la aplicación publicado en GitHub e instalar todas las herramientas y dependencias del proyecto (https://github.com/almancebo/certificacion-energetica).

2. Configurar los parámetros del nodo en Ganache con los valores mostrados en la Figura 51.

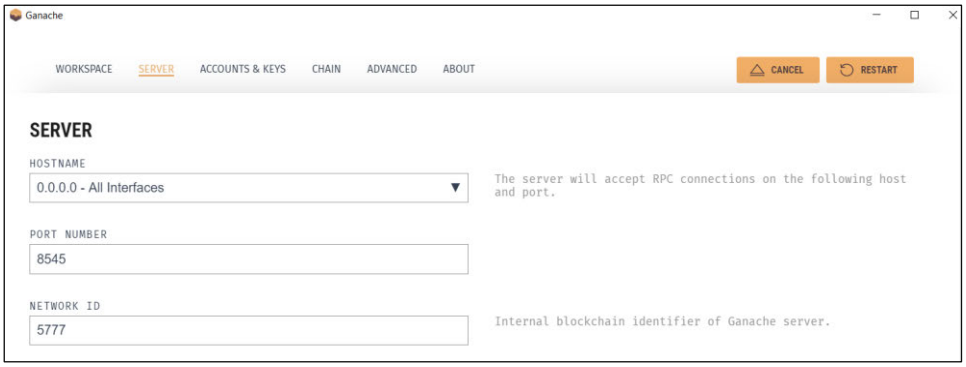

Fig. 51. Configuración de Ganache.

3. Obtener de las variables: GAS PRICE, GAS LIMIT y PRIVATE KEY en la interfaz de Ganache, figuras 52 y 53.

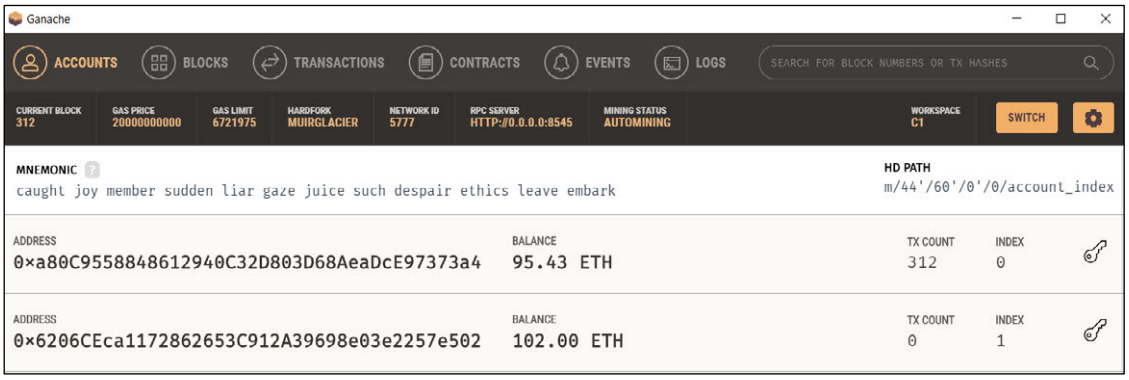

Fig. 52. Interfaz de usuario de Ganache.

Fig. 53. Información de cuenta y clave privada en Ganache.

4. Actualizar las variables de la Figura 54 en el archivo ApiService.java con los datos obtenidos en la interfaz.

```
private final static String PRIVATE_KEY
="80281063e78a8510bb44fc3dbb4aa894391d74175211e40c6400f9619844f295";

private final static BigInteger GAS_LIMIT = BigInteger.valueOf(6721975L);

private final static BigInteger GAS_PRICE =
BigInteger.valueOf(20000000000L);
```

Fig. 54. Variables: GAS PRICE, GAS LIMIT y PRIVATE KEY.

5. Iniciar la aplicación pulsando el botón “Run” en Intellij o ejecutando .\mvnw spring-boot:run en el directorio de la aplicación.

6. Una vez la aplicación se ha iniciado sin errores se puede hacer uso de todas las funcionalidades disponibles.

   Listado de peticiones HTTP al API:

   - Emitir certificado energético: curl -X POST -H "Content-Type: application/json" -d "{\"identificadorComercializador\":\"<comercializador>\",\"identificadorInstalacion\":\"<isntalacion>\",\"produccionEnergetica\":<producción>,\"idTecnologiaEmpleada\":<n>,\"fechaGeneracion\":<dmmaaaa>,\"factorEmision\":<n>}" http://localhost:8080/api/emitirCertificado

   - Consultar certificado: curl -X GET http://localhost:8080/api/consultarCertificado/<certificadoId>

   - Transferir certificado: curl -X POST http://localhost:8080/api/transferirCertificado/<certificadoId>/nuevoTitular/<nuevoTitularAddress>

   Listado de vistas disponibles:

   - http://localhost:8080/certificados: muestra una lista de todos los certificados emitidos y actualizados.

8. Impacto del proyecto

Dado que el proyecto está enfocado en el uso de energías renovables y plataformas basadas en blockchain, son diversos los puntos de impacto en los que se contribuye a los Objetivos de Desarrollo Sostenible [48]:

- Energía Asequible y No Contaminante, ODS 7: El proyecto busca promover el acceso a energía asequible y sostenible para todos. Utilizando fuentes de energía renovables como la solar, eólica, hidroeléctrica o biomasa, se reduce la dependencia de los combustibles fósiles y se evita la emisión de gases de efecto invernadero. Esto contribuye a mitigar el cambio climático y mejorar la calidad del aire.

- Industria, Innovación e Infraestructura, ODS 9: El uso de plataformas basadas en blockchain en el sector de las energías renovables impulsa la innovación y la mejora de la infraestructura energética. Blockchain permite la creación de registros seguros y transparentes de la generación, distribución y consumo de energía, lo que facilita la trazabilidad de los datos y promueve la eficiencia en la gestión energética. Además, blockchain puede habilitar la creación de mercados energéticos descentralizados, donde los productores y consumidores pueden interactuar directamente, fomentando la generación distribuida y la participación activa de los usuarios en la transición energética.

- Acción por el Clima, ODS 13: Al reducir las emisiones de gases de efecto invernadero y promover el uso de energías renovables, el proyecto contribuye a la mitigación del cambio climático. Al utilizar blockchain se mejora la transparencia y la confiabilidad de las transacciones energéticas, lo que facilita la implementación de mecanismos de compensación de carbono y la contabilidad precisa de las reducciones de emisiones. Asimismo, la tecnología blockchain puede impulsar la adopción de modelos de negocio basados en la economía circular, promoviendo la eficiencia en el uso de recursos y reduciendo el impacto ambiental.

9. Conclusiones

En este proyecto, nos propusimos abordar el problema de garantizar la seguridad en el etiquetado de la electricidad, a la vez que creamos una plataforma descentralizada accesible para todos y promovemos el impulso de energías renovables para enfrentar los desafíos del cambio climático.

Para resolver este problema, implementamos el uso de la tecnología blockchain, que nos permitió asegurar y autenticar el etiquetado de la electricidad o denominaciones de origen de manera confiable.

Como resultado, logramos desarrollar una plataforma descentralizada que facilita el acceso y la participación de diferentes actores en el mercado eléctrico. La plataforma presenta al usuario un API interactivo en la emisión, transmisión y consultas de certificados energéticos, a la vez que, ofrece una vista web en el navegador donde se listan todos los certificados emitidos y actualizados. Al utilizar la tecnología blockchain se proyecta eliminar intermediarios innecesarios y la posibilidad de reducir los costos asociados a los organismos tradicionales.

Como objetivos marcados y no abordados mencionamos la visualización de certificados energéticos a través de un formato QR que excedía la carga de trabajo. Siendo este un aspecto menos relevante en el desarrollo de la la solución y que incluimos como trabajos o funcionalidades futuras.

En cuanto a los resultados obtenidos, hemos demostrado que nuestra solución es operable y efectiva. Y que a través de la inyección de nuevas tecnologías que presentan diversos beneficios y posibilidades, se puede promover activamente la transición hacia un mayor uso de energías renovables.

9.1. Trabajos futuros

Si bien el proyecto ha logrado avances significativos, es importante destacar que el sistema se ha desarrollado en un entorno local y simulado, estando abierto a posibles líneas futuras de trabajo que escalablemente puedan aumentar sus funcionalidades, alcance y objetivos. Algunas de las mejoras técnicas y funcionales que se plantean son:

Funcionalidades futuras:

- Cálculo del etiquetado sobre todo tipo de energía, renovable y no renovable, que incluye la adición de métodos y fórmulas que analicen cada caso de generación en particular.
- Ofrecer funcionalidades de compra y venta de energía a compañías eléctricas y consumidores, pudiendo estos realizar la facturación a través de contratos inteligentes.

Mejoras técnicas:

- Utilizar un QR identificativo por cada certificado.
- Integrar una base de datos en la capa de repositorio.
- Investigar modelos de seguridad de claves en la emisión, transmisión y consulta de certificados.
- Ofrecer funcionalidades de operabilidad para el usuario dentro de las vistas web con el uso de HTML, CSS y Javascript.

10. Referencias

[1] Naciones Unidas, «Energías renovables: energías para un futuro más seguro.,» [En línea]. Available: https://www.un.org/es/climatechange/raising-ambition/renewable-energy. [Último acceso: 2023].

[2] H. Ritchie, «Sector by sector: where do global greenhouse gas emissions come from?,» 18 septiembre 2020. [En línea]. Available: https://ourworldindata.org/ghg-emissions-by-sector. [Último acceso: 2023].

[3] Naciones Unidas, «¿Qué es el cambio climático?,» [En línea]. Available: https://www.un.org/es/climatechange/what-is-climate-change. [Último acceso: 2023].

[4] Noticias ONU, «La ONU lanza un plan de acción mundial para impulsar la energía limpia y económica,» 4 Mayo 2022. [En línea]. [Último acceso: 2023].

[5] United Nations Climate Change, «Las emisiones mundiales de CO2 repuntaron en 2021 hasta su nivel más alto de la historia,» 14 Marzo 2022. [En línea]. Available: https://unfccc.int/es/news/las-emisiones-mundiales-de-co2-repuntaron-en-2021-hasta-su-nivel-mas-alto-de-la-historia. [Último acceso: 2023].

[6] epdata, «El cambio climático, en datos y gráficos,» 10 enero 2023. [En línea]. Available: https://www.epdata.es/datos/cambio-climatico-datos-graficos/447. [Último acceso: 2023].

[7] Ministerio para la transición ecológica y el reto demográfico, «Libro de la Energía en España 2020,» 2022. [En línea]. Available: https://energia.gob.es/balances/Balances/LibrosEnergia/Libro_Energia_Espana_2020.pdf. [Último acceso: 2023].

[8] United Nations Climate Change, «El costo de las renovables se reduce drásticamente y supera la opción más barata de combustibles fósiles,» 4 junio 2020. [En línea]. Available: https://unfccc.int/es/news/el-costo-de-las-renovables-se-reduce-drasticamente-y-supera-la-opcion-mas-barata-de-combustibles. [Último acceso: 2023].

[9] Ministerio para la Transición Energetica y el Reto Demográfico, «www.miteco.gob.es,» 21 diciembre 2021. [En línea]. Available: https://www.miteco.gob.es/es/prensa/ultimas-noticias/el-gobierno-aprueba-una-l%C3%ADnea-de-ayudas-para-la-implantaci%C3%B3n-de-renovables-t%C3%A9rmicas-en-diferentes-sectores-de-la-econom%C3%ADa/tcm:30-534372. [Último acceso: 2023].

[10] www.ree.es, «¿Cómo funciona el sistema eléctrico?,» 30 noviembre 2007. [En línea]. Available: https://www.ree.es/es/sala-de-prensa/infografias-y-mapas/como-funciona-el-sistema-electrico. [Último acceso: 2023].

[11] Agencia Estatal Boletín Oficial del Estado, «BOE-A-2021-2570,» 19 febrero 2021. [En línea]. Available: https://www.boe.es/buscar/doc.php?id=BOE-A-2021-2570. [Último acceso: 2023].

[12] Stock logistic, «LOS USOS DEL BLOCKCHAIN EN LOGÍSTICA,» 15 mayo 2018. [En línea]. Available: https://www.stocklogistic.com/blockchain-logistica/. [Último acceso: 2023].

[13] Binance Academy, «Proof of Work (PoW) vs. Proof of Stake (PoS),» 1 febrero 2023. [En línea]. Available: https://academy.binance.com/es/articles/proof-of-work-vs-proof-of-stake. [Último acceso: 2023].

[14] IBM, «Definición de contratos inteligentes,» [En línea]. Available: https://www.ibm.com/es-es/topics/smart-contracts. [Último acceso: 2023].

[15] S. Beyer, «Las 5 vulnerabilidades más habituales de los Smart Contracts,» 10 marzo 2020. [En línea]. Available: https://www.securityartwork.es/2020/03/10/las-5-vulnerabilidades-mas-habituales-de-los-smart-contracts/. [Último acceso: 2023].

[16] Albert Fattal, «Artificial Intelligence and Blockchain within Information Systems,» [En línea]. Available: https://albertfattal.com/latest-publications/what-ai-and-blockchain-can-do-for-information-systems/. [Último acceso: 2023].

[17] Devtop, «¿Qué es Proof of Stake (PoS)?,» 24 enero 2023. [En línea]. Available: https://devtop.io/que-es-proof-of-stake-pos/. [Último acceso: 2023].

[18] R. WOLFSON, «COINTELEGRAPH en Español,» 20 agosto 2021. [En línea]. Available: https://es.cointelegraph.com/news/clearing-the-air-renewably-sourced-bitcoin-may-ensure-a-clean-energy-future. [Último acceso: 2023].

[19] Iberdrola, «Cómo puede el 'blockchain' acreditar el origen de la energía verde,» [En línea]. Available: https://www.iberdrola.com/innovacion/blockchain-energia. [Último acceso: 2023].

[20] Energy Web, «Why we exist,» [En línea]. Available: https://www.energyweb.org/why-we-exist/. [Último acceso: 2023].

[21] ethereum.org, «¿QUÉ ES ETHEREUM?,» [En línea]. Available: https://ethereum.org/es/what-is-ethereum/. [Último acceso: 2023].

[22] soliditylang.org, «Solidity,» [En línea]. Available: https://soliditylang.org/. [Último acceso: 2023].

[23] Founderz, «Solidity, el lenguaje de programación de los contratos inteligentes,» [En línea]. Available: https://founderz.com/blog/solidity-que-es-caracterisiticas/. [Último acceso: 2023].

[24] remix-project.org, «REMIX PROJECT,» [En línea]. Available: https://remix-project.org/. [Último acceso: 2023].

[25] trufflesuite.com, «What is Ganache?,» [En línea]. Available: https://trufflesuite.com/docs/ganache/. [Último acceso: 2023].

[26] solidity-es.readthedocs.io, [En línea]. Available: https://solidity-es.readthedocs.io/es/latest/using-the-compiler.html. [Último acceso: 2023].

[27] java, «¿Qué es la tecnología Java y por qué la necesito?,» [En línea]. Available: https://www.java.com/es/download/help/whatis_java.html. [Último acceso: 2023].

[28] desarrolloweb.com, «Qué es HTML,» 1 enero 2001. [En línea]. Available: https://desarrolloweb.com/articulos/que-es-html.html. [Último acceso: 2023].

[29] Y. Muradas, «OpenWebinars: Qué es Gradle,» 25 febrero 2022. [En línea]. [Último acceso: 2023].

[30] IBM, «¿Qué es una API REST?,» [En línea]. Available: https://www.ibm.com/es-es/topics/rest-apis. [Último acceso: 2023].

[31] IBM, «¿Qué es Java Spring Boot?,» [En línea]. Available: https://www.ibm.com/es-es/topics/java-spring-boot. [Último acceso: 2023].

[32] R. D. Hernandez, «freeCodeCamp,» 28 junio 2021. [En línea]. Available: https://www.freecodecamp.org/espanol/news/el-modelo-de-arquitectura-view-controller-pattern/.

[33] B. J. Diaz, «Linkedin: Uso de @controller, @Service y @Repository en Spring Boot,» 23 marzo 2023. [En línea]. Available: https://es.linkedin.com/pulse/uso-de-controller-service-y-repository-en-spring-boot-bryan-j-diaz. [Último acceso: 2023].

[34] Web3j, «Web3j,» [En línea]. Available: https://docs.web3j.io/4.10.0/. [Último acceso: 2023].

[35] JET BRAINS, «¿Qué es IntelliJ IDEA?,» [En línea]. Available: https://www.jetbrains.com/es-es/idea/features/. [Último acceso: 2023].

[36] J. Saladas, «IBM Developer: What is cURL and how does it relate to APIs?,» 22 febrero 2021. [En línea]. Available: https://developer.ibm.com/articles/what-is-curl-command/. [Último acceso: 2023].

[37] postman.com, «What is Postman?,» [En línea]. Available: https://www.postman.com/product/what-is-postman/. [Último acceso: 2023].

[38] autarquiapersonal.com, «LA HUELLA DE CARBONO DE UNA INSTALACIÓN SOLAR FOTOVOLTAICA,» [En línea]. Available: https://autarquiapersonal.com/2021/02/06/la-huella-de-carbono-del-autoconsumo-fotovoltaico/. [Último acceso: 2023].

[39] P. A. C. Landazuri, «Análisis comparativo de la Huella de Carbono de Un Parque Eólico en Tenerife- España,» septiembre 2019. [En línea]. Available: https://iconline.ipleiria.pt/bitstream/10400.8/4619/1/%28VERS%C3%83O%20CORRIGIDA%29%20-%20An%C3%A1lisis%20de%20la%20Huella%20de%20Carbono%20de%20Un%20Parque%20E%C3%B3lico%20en%20Tenerife-%20Espa%C3%B1a.pdf. [Último acceso: 2023].

[40] hydropower.org, «Hydropower's carbon footprint,» [En línea]. Available: https://www.hydropower.org/factsheets/greenhouse-gas-emissions. [Último acceso: 2023].

[41] J. L. C. Cabañes, «La huella de carbono de las energías renovables. 1- La biomasa.,» 11 febrero 2016. [En línea]. Available: https://www.comunidadism.es/la-huella-de-carbono-de-las-energias-renovables-1-la-biomasa/. [Último acceso: 2023].

[42] sede.idae.gob.es, «Biogas: Guía para la justificación de la reducción de emisiones de gases de efecto invernadero,» 21 septiembre 2022. [En línea]. Available: https://sede.idae.gob.es/lang/extras/tramites-servicios/2022/BIOGAS/8._Guia_calculo_de_reduccion_de_GEI_(Actualizado_a_fecha_21.09.202).pdf. [Último acceso: 2023].

[43] Spring, «Spring Initializr,» [En línea]. Available: https://start.spring.io/;. [Último acceso: 2023].

[44] S. Culoccioni, «Programación en tres Capas con java,» 25 diciembre 2014. [En línea]. Available: https://www.solvetic.com/tutoriales/article/1378-programaci%C3%B3n-en-tres-capas-con-java/. [Último acceso: 2023].

[45] Jobted, «Sueldo del Programador Java en España,» [En línea]. Available: https://www.jobted.es/salario/programador-java. [Último acceso: 2023].

[46] Kiwi Remoto, «Sueldo de Desarrollador/a Solidity (2023),» [En línea]. Available: https://www.kiwiremoto.com/sueldo/desarrollador-solidity/. [Último acceso: 2023].

[47] Software DELSOL, «¿Cómo calcular la jornada laboral anual?,» [En línea]. Available: https://www.sdelsol.com/blog/laboral/como-calcular-la-jornada-laboral-anual/. [Último acceso: 2023].

[48] Naciones Unidas, «Objetivos de Desarrollo Sostenible,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/. [Último acceso: 2023].