Universidad de Valladolid

Escuela de Ingeniería Informática de Valladolid

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Mención Ingeniería de Software

Diseño y evaluación de contratos inteligentes para optimizar la eficiencia en la negociación en cadenas de suministros

Autor:

Diego. Capellán. Rodríguez

Tutores:

Joaquín Nicolás Adiego Rodríguez
Teresa Natalia Martín Cruz

Agradecimientos

Quisiera expresar mi más sincero agradecimiento a todas las personas que, de una manera u otra, han contribuido a la realización de este Trabajo de Fin de Grado.

En primer lugar, agradezco a **mis tutores**, **Joaquín Adiego** y **Teresa Natalia** por sus consejos a lo largo del proceso; así como a Manuel Sobrino, por cederme el caso de Renault, gracias al cual, he podido realizar este trabajo

A mi familia, por estar siempre a mi lado, por su amor incondicional, comprensión y paciencia. Sin su apoyo emocional y motivación, nada de esto habría sido posible.
A mis amigos, por compartir conmigo esta etapa, por su apoyo, ánimo y por los momentos vividos que han hecho esta experiencia mucho más llevadera.

Y, de manera **muy especial**, quiero agradecer a mi **pareja**, **Elisa Bravo**, por haber sido mi mayor apoyo durante esta etapa. No solo hemos recorrido este camino académico juntos, sino que lo hemos hecho acompañándonos en cada paso, compartiendo esfuerzos, desvelos, y también alegrías. Tu presencia, comprensión y constancia han sido un pilar fundamental para mí, y este logro también es tuyo.

Finalmente, gracias a todas las personas que, de una forma u otra, han contribuido a que este proyecto llegue a buen puerto. A todos, **gracias de corazón**.

Resumen

El objetivo es **diseñar, desarrollar y evaluar contratos inteligentes** que permitan una **negociación** transparente y **automatizada** del **precio de bienes** o servicios entre compradores y uno o varios proveedores.

En este proceso, una vez alcanzado el acuerdo, se procederá a ejecutar el pago de forma automática utilizando el token nativo de la red. Además, se llevará a cabo un análisis exhaustivo del coste de gas asociado a la ejecución de cada contrato, explorando diversas implementaciones y optimizaciones para minimizar este gasto sin comprometer la eficiencia.

Abstract

The objective is to **design, develop, and evaluate smart contract** that leads us to a transparent **negotiation** and an **automation** of the **price of the goods** and the services between one or more buyers and one or more providers

In this process, once an agreement is reached, the payment will be done automatically with network native token's help. In addition, we will be doing an exhaustive analysis of gas' price associated with the execution of the environment in each smart contract, exploring diverse implementations and optimizations to minimize the cost without compromising the efficiency.

Tabla de Contenidos

Lista de figuras

Lista de tablas

1_Introducción y objetivos

1_1_Introducción

En los últimos años, el avance de las tecnologías blockchain y, en particular, de los contratos inteligentes (*smart contracts*), ha generado un creciente interés en múltiples sectores industriales por su capacidad para automatizar procesos, reducir intermediarios y aumentar la transparencia en las transacciones. Esta transformación digital tiene un gran potencial para impactar positivamente en ámbitos tradicionalmente complejos y burocráticos, como es el caso de las **cadenas de suministro.**

Una de las fases más críticas dentro de una cadena de suministro es el proceso de **negociación entre entidades colaboradoras**, especialmente cuando se trata de suministros técnicos altamente especializados, como los componentes prototípicos que deben ser validados, negociados y acordados entre distintas unidades organizativas. En muchos entornos industriales, esta negociación continúa realizándose de manera informal o semiestructurada, con **acuerdos verbales** o **por correo electrónico**, lo que introduce una elevada incertidumbre, riesgos de malentendidos y dificultades a la hora de establecer responsabilidades contractuales claras. Esto puede desembocar en **ineficiencias**, retrasos e incluso en el **repudio posterior de los acuerdos**, con el consiguiente impacto negativo sobre la trazabilidad y la confianza entre las partes implicadas.

El presente Trabajo de Fin de Grado se centra en **diseñar y evaluar una propuesta técnica basada en contratos inteligentes que permita optimizar la eficiencia en los procesos de negociación dentro de cadenas de suministro**. El objetivo no es comercializar un producto final, sino **demostrar la viabilidad técnica** de utilizar este tipo de herramientas para formalizar acuerdos de forma automática, transparente e inalterable, reduciendo la ambigüedad y mejorando la seguridad jurídica de las relaciones interorganizativas.

A nivel tecnológico, el trabajo se ha enfocado en el desarrollo del backend del sistema, utilizando el lenguaje **Solidity** y el entorno de desarrollo **Remix IDE**, desplegando contratos sobre la red de pruebas de **Ethereum**. Aunque el proyecto se encuentra en una fase temprana y no ha sido aún desplegado en un entorno productivo ni conectado a una interfaz de usuario (*frontend*), se ha prestado especial atención al análisis funcional, al diseño de la lógica contractual y a la correcta parametrización de los procesos que deben quedar registrados de forma inmutable en la blockchain. Para ello, se han elaborado diversos diagramas de análisis y diseño utilizando la herramienta **Astah Professional**, permitiendo estructurar el comportamiento del sistema y anticipar futuras mejoras.

El caso de uso trabajado está inspirado en un **caso real** de la **empresa Renault**, en el cual se identificaron las principales limitaciones asociadas a la negociación tradicional de componentes. Si bien el trabajo no profundiza en la implementación de un sistema completo con interfaz de usuario, se ha podido observar que los contratos inteligentes **pueden aportar un valor significativo** en términos de trazabilidad y formalización de acuerdos, siempre y cuando se integren adecuadamente con herramientas que limiten y validen los datos introducidos por el usuario. De hecho, uno de los principales hallazgos ha sido precisamente que la utilidad de estos contratos se ve condicionada en gran medida por la existencia de un frontend robusto que garantice la integridad de los parámetros y evite errores humanos en la introducción de información crítica.

En resumen, este trabajo busca **sentar las bases para una solución más amplia e integrada**, que permita agilizar la **negociación entre actores** dentro de una cadena de suministro, haciendo uso de tecnologías emergentes como la blockchain y los contratos inteligentes. Aunque el **alcance actual** se limita al **backend**, las conclusiones obtenidas permiten vislumbrar un camino prometedor hacia sistemas más seguros, automáticos y eficientes en el contexto de la industria 4.0.

1_2_Objetivos

El objetivo principal de este Trabajo de Fin de Grado es **analizar, diseñar e implementar un contrato inteligente que contribuya a optimizar los procesos de negociación en cadenas de suministro**, especialmente en lo que respecta a la reducción de trámites burocráticos y la validación de acuerdos. El propósito no es únicamente técnico, sino también funcional, ya que se busca demostrar cómo los contratos inteligentes pueden ofrecer una mayor trazabilidad y transparencia, facilitando así la toma de decisiones y la resolución de conflictos entre las partes implicadas.

Para ello, se plantea como meta el desarrollo de un software capaz de **modelar un caso de uso representativo dentro de una cadena de suministro**, utilizando tecnologías propias del ecosistema blockchain. Este software, centrado en la lógica de backend, debe ser capaz de reflejar las principales fases de la negociación entre empresas, incluyendo la formalización de condiciones, validación de parámetros y almacenamiento inmutable de los compromisos adquiridos.

Aunque el proyecto se encuentra en un estado no desplegado, la implementación realizada en el entorno de pruebas **Remix** y el uso del lenguaje **Solidity** permiten ilustrar de manera práctica las capacidades de la tecnología blockchain en este contexto. Asimismo, el diseño del sistema ha sido documentado mediante diagramas estructurales y de comportamiento que recogen tanto los requisitos funcionales como las decisiones arquitectónicas adoptadas.

1_3_Contenidos de la memoria

Esta memoria se estructura en varios capítulos que recogen de forma progresiva el desarrollo del proyecto, desde los objetivos iniciales hasta las conclusiones finales. A continuación, se describen brevemente los contenidos de cada sección:

Capítulo 1: Introducción y objetivos.
Presenta el contexto del proyecto, los objetivos principales y esta guía sobre los contenidos de la memoria.

Capítulo 2: Planificación.
Describe la planificación seguida durante el desarrollo, incluyendo la metodología utilizada, el análisis de costes y una evaluación de riesgos y oportunidades.

Capítulo 3: Tecnologías utilizadas.
Se detallan las herramientas, lenguajes y plataformas empleadas, junto con una justificación de su elección y una breve explicación de su funcionamiento.

Capítulo 4: Estudio preliminar de la interacción Interempresarial.
Analiza los agentes implicados en el sistema (proveedores y fabricantes) y el flujo de una negociación empresarial, sirviendo como base conceptual para el desarrollo posterior.

Capítulo 5: Análisis.
Se recogen los requisitos funcionales y no funcionales del sistema, así como los diagramas de caso de uso que definen su comportamiento esperado.

Capítulo 6: Diseño.
Incluye los diagramas de secuencia, clases y estados, así como los patrones de diseño utilizados y un estudio de costes relacionado con el rendimiento del sistema.

Capítulo 7: Conclusiones y trabajo futuro.
Se presentan las conclusiones del trabajo realizado, se evalúan los resultados obtenidos y se proponen posibles mejoras y líneas de desarrollo futuras.

Referencias y Anexos.
Se incluyen todas las fuentes bibliográficas consultadas, así como material complementario relevante para la comprensión del proyecto.

2_Planificación

2_1_Metodologías utilizadas

Dado el carácter individual de este TFG y la limitada estructura organizativa implicada en su desarrollo, ha sido necesario optar por metodologías que no requieran una segmentación rígida de roles, y que se adapten bien a **entornos de trabajo unipersonales**. En este sentido, se ha hecho uso de una combinación de enfoques **flexibles y adaptativos**, que permiten una planificación funcional del trabajo sin imponer una sobrecarga organizativa innecesaria.

2_1_1_Task Driven Planning

La metodología **Task-Driven Planning** ha sido la base principal para la organización del trabajo durante el desarrollo del proyecto. Este enfoque se caracteriza por su simplicidad, adaptabilidad y orientación a resultados, lo que lo hace especialmente útil en contextos académicos donde el estudiante asume múltiples roles (analista, desarrollador, documentador, etc.).

Sus principales características son:

- **Listado de tareas estructurado** (checklist), que permite tener una visión clara del progreso.
- **Priorización flexible de tareas**, adaptándose a las dificultades o nuevos aprendizajes que surgen durante el proceso.
- **Ausencia de sprints formales**, eliminando la rigidez de ciclos cerrados.
- **Alta adaptabilidad** frente a cambios de enfoque o redirección de esfuerzos.
- **Adecuado para estructuras individuales o equipos reducidos.**
- **Fomenta la motivación personal**, al proporcionar una sensación de progreso constante y autonomía.

Este método ha permitido avanzar de forma iterativa pero sin comprometer la flexibilidad, lo que ha sido clave dada la naturaleza exploratoria de muchas fases del TFG.

2_1_2_Agile

Si bien Task Driven no contempla de manera explícita el componente comunicativo o de supervisión externa, se ha incorporado un enfoque **complementario inspirado en metodologías Agile**, en particular, en lo relativo a las revisiones periódicas y la retroalimentación continua.

A lo largo del desarrollo del proyecto se han llevado a cabo **reuniones periódicas con el tutor académico**, cuyo objetivo ha sido revisar el estado de avance, corregir desviaciones y orientar la adquisición de nuevos conocimientos. Estas sesiones han servido también para facilitar recursos complementarios (como tutoriales o documentación técnica sobre Remix, Solidity y su integración con blockchain) que han reforzado el proceso de aprendizaje autónomo.

Si bien no se puede hablar de una implementación formal de Agile en su totalidad (al no existir un equipo multidisciplinar ni iteraciones completas de producto funcional), la inclusión de estas dinámicas ha aportado valor añadido al proceso organizativo y ha permitido mantener un rumbo claro y guiado.

2_2_Estudio Económico

Aunque el presente Trabajo de Fin de Grado (TFG) no ha implicado transacciones económicas reales ni una financiación externa, se ha considerado pertinente realizar una **estimación teórica del coste económico** que supondría su desarrollo en un entorno profesional. Este tipo de ejercicio permite contextualizar el esfuerzo invertido desde un punto de vista económico, valorar el trabajo realizado, y proyectar su viabilidad o escalabilidad en un contexto empresarial o institucional.
A continuación, se detallan los distintos apartados que componen esta estimación, agrupados en función de su naturaleza.

2_2_1_Coste de desarrollo

El componente principal del coste estimado del proyecto corresponde al tiempo de trabajo invertido. En el marco del TFG, se ha estipulado una dedicación total de **300 horas**, distribuidas a lo largo de varias semanas durante el curso académico.
Para calcular el coste asociado, se ha tomado como referencia el **salario medio bruto** de un becario o estudiante en prácticas en el ámbito de la ingeniería informática, que se sitúa, en torno a los **1442€ mensuales** por una jornada de **40 horas semanales.**

$$\frac{1442€}{mes} * \frac{1mes}{4semanas} * \frac{1semana}{40horas} = \frac{12.01€}{hora}$$

Aplicando esta tarifa a la carga total de trabajo, se obtiene el siguiente valor:

$$300\ horas * \frac{12.01€}{hora} = 3603€$$

Este coste representa el valor económico que tendría el tiempo de dedicación si este TFG fuera realizado bajo contrato laboral en una empresa, o como servicio profesional.

2_2_2_Costes indirectos

A pesar de que el desarrollo del proyecto se ha realizado en un entorno personal (principalmente en el domicilio del estudiante), existen una serie de costes indirectos que deben ser considerados. Estos costes incluyen, entre otros, el consumo eléctrico derivado del uso prolongado del ordenador, la conexión a internet, el desgaste del equipo informático personal (hardware), así como posibles recursos auxiliares como impresiones, almacenamiento, o copias de seguridad que, en un entorno empresarial, tendrían una repercusión económica.

Aunque es difícil establecer un valor exacto de estos costes, se ha realizado una estimación aproximada, tomando como base el consumo habitual mensual y el coste asociado en términos generales para una **duración de aproximadamente 3 a 4 meses** de trabajo activo. Se ha fijado una cifra estimada de

200€, que representa un valor prudente y proporcionado en relación con el tiempo de ejecución y los medios utilizados.

2_2_3_Coste de las herramientas de software

Durante la elaboración del TFG, se ha hecho uso de varias herramientas de software que, aunque no han supuesto un desembolso directo durante el proyecto (en caso de licencias preexistentes o licencias educativas), deben ser valoradas económicamente por su relevancia en el desarrollo del mismo.

2_2_3_1_Microsoft 365 (Word y Excel)

Para la redacción de la memoria y la gestión de datos y cronogramas, se ha empleado la **versión Microsoft Office 2007**, que incluye Word y Excel. Esta versión fue adquirida en su momento mediante una licencia perpetua, por lo que no ha supuesto un coste económico adicional durante el desarrollo del TFG.

No obstante, para efectos del estudio económico teórico, se considera relevante estimar el valor actual que tendría utilizar una solución equivalente. En el mercado actual, el software de ofimática de Microsoft se ofrece principalmente a través del modelo de suscripción denominado Microsoft 365, con un coste aproximado de **69€/año** para uso individual. Considerando que el periodo de uso efectivo se ha limitado a unos 4 meses, se calcula una estimación proporcional del coste, equivalente a:

$$\frac{4}{12} * 69€ = 23€$$

Este valor refleja un **uso parcial de la licencia**, asumiendo que su coste se distribuye uniformemente a lo largo del año.

2_2_3_2_Astah Professional

Asimismo, para la elaboración de **diagramas UML** y la **modelización de diversos componentes del sistema**, se ha utilizado Astah Professional, una herramienta especializada en el diseño de software. En su versión para estudiantes, esta herramienta tiene un **coste mensual de 8,25€**, que se ha mantenido durante tres meses aproximadamente.
Por tanto, el coste total estimado es:

$$\frac{8.25€}{mes} * 3\ meses = 24.75€$$

Este valor incluye únicamente el tiempo de uso efectivo durante el proceso de modelado y documentación técnica.

2_2_4_Coste total estimado

Sumando todos los conceptos anteriormente descritos, se obtiene el coste total estimado del proyecto, como se detalla en la siguiente tabla:

Concepto	Detalle	Coste estimado
Coste de Desarrollo	300 horas *	3603’00€
Costes Indirectos	Electricidad, conexión desgaste de tiempo, etc.	200’00€
Herramientas de Software	Suma de herramientas	27’75€
Microsoft 365	69€/año uso proporcional de 4 meses	23’00€
Astah Professional	8’25€/mes * 3 meses	24’75€
Total estimación proyecto		**3830.75€**

Tabla 1: Tabla de costes

2_3_Riesgos y oportunidades

ID	Nombre	Descripción	Categoría	Vulnerabilidad	Amenaza	Probabilidad	Impacto	Riesgo Total	Estado del Riesgo	Acciones de mitigación	Acciones correctivas
RSK001	TFG no llamativo	No hay ningún TFG disponible que sea de mi interés	Personal	No estoy interesado en realizar un TFG de un tema en el que no estoy interesado	No puefo hacer el TFG, si ninguno es de mi agrado	BAJA	BAJO	BAJO	Abierto	1. Proponer un TFG	1. Hacer el TFG igualmente
RSK002	Profesor fantasma	El profesor encargado de orientarme mientras hago el TFG, no está disponible cuando tengo dudas, o no aclara mis dudas cuando se las hago	Interpersonal	No tener dudas aclaradas, difucltades para avanzar correctamente	Me rechazan el TFG, por que algo que he implementado no ha sido a gusto de los profesores	ALTA	ALTO	ALTO	Abierto	1. Hablar con el director el problema	1. Buscar en persona al profesor para preguntarle 2. Ignorar al profesor como responsable, y depender de un tercero
RSK003	Mala estimacion de riesgos	Debido a la mala estimacion de riegos que he hecho, no he sabido adaptarme/prepararme con anterioridad	Personal	No saber consecuencias	Retardo del TFG, o incluso, falta de entrega del TFG	BAJA	MEDIO	BAJO	Cerrado	1. Ninguna	1. Buscar información sobre DAFO, y como manejar problema in situ
RSK004	Robo de TFG	Alguien termina el TFG que yo tenia preparado antes que yo	Personal	Me roban la tematica de mi TFG antes de poder defenderlo y demostrar mi autoría	Baja calificaciond e mi TFG	BAJA	ALTO	MEDIO	Abierto	1. Notificar al profesorado de mi TFG	1. Ninguna
RSK005	TFG rechazado	En la defensa del TFG, se me rechaza	Personal	No poder entregar un TFG que no ha sido aprobado	Suspender la asignatura de TFG, y perder tiempo y dinero	ALTA	MEDIO	ALTO	Abierto	1. La combinación del uso de parrafos extensos, y de profesores que no suelen leer los trabajos, favorecen la utilizacion de la redaccion extensiva con falta de información util, pero que al estar en abundancia, da la sensación de ser correcta 2. Asumir que solo quieren preguntarme cosas relacionadas con aquello que me hayan impartido	1. Poner inforamcion simple 2. Poner informacion para que parezca que se de todo lo que me ha impartido cada profesor 3. Mentir o exagerar en calculos para favorecer mi opinion, o encarecer las opiniones opuestas a las mías
O001	Profesor me ofrece un TFG	Un profesor con el que tegno confianza me ofrece un TFG	Interpersonal	Ninguna	Ninguna	BAJA	ALTO	MEDIO	Abierto	Ninguna	Ninguna
O002	TFG en grupo	Acepto un TFG, en el que trabajaremos varios alumnos	Interpersonal	El alumno me cae mal	El alumno es una persona con la que no se puede trabajar comodamente	BAJA	ALTO	MEDIO	Abierto	1. Rechazar el TFG	1. Rechazar el TFG
O003	Avanzar el TFG, en plazo extracurricular	Avanzo en el TFG, antes de solicitarlo	Cronológica	No tengo el TFG oficialmente	Me roban el TFG	ALTA	ALTO	ALTO	Abierto	Ninguna	Ninguna

Tabla 2: Riesgos y oportunidades para la realización de un TFG

3_Tecnologías utilizadas

3_1_Blockchain


Ilustración 1: Satoshi Nakamoto

Propuesto originalmente por **Satoshi Nakamoto** en 2008 para impulsar la Bitcoin(Satoshi, 2008), la tecnología blockchain ha transformado las **transacciones electrónicas** eliminando los intermediarios y habilitando operaciones **P2P descentralizadas**. Su innovación definitoria es el establecimiento de un libro de contabilidad distribuido, en el que las transacciones se registran cronológicamente y se agrupan en bloques vinculados criptográficamente. Este diseño garantiza la **inmutabilidad** y la **transparencia**:
una vez registrados los datos, ninguna entidad puede alterarlos, preservando así la confianza en la red. En implementaciones tempranas, como Bitcoin, el mecanismo de consenso de "Proof of Work" (PoW) determina qué bloque se añade a la cadena de bloques.

Si bien el despliegue inicial de la cadena de bloques (comúnmente conocida como Blockchain 1.0) se centró exclusivamente en las transacciones monetarias, los desarrollos posteriores han ampliado su alcance. Esto dio lugar a la noción de Blockchain 2.0, que introdujo contratos inteligentes y aplicaciones descentralizadas (dApps) como funcionalidades clave. Plataformas como Ethereum ampliaron aún más las capacidades de blockchain en sistemas más flexibles y programables, allanando el camino para aplicaciones en diversos sectores.

A pesar de estos avances, la cadena de bloques sigue enfrentándose a obstáculos considerables, en particular con respecto a la escalabilidad, el consumo de energía y la privacidad. Estas limitaciones han impulsado importantes esfuerzos de investigación destinados a optimizar la tecnología blockchain para su adopción masiva y garantizar su viabilidad práctica.

3_1_1_Sostenibilidad ambiental de la tecnología blockchain

A medida que las cadenas de suministro globales se esfuerzan por alinearse con los objetivos de sostenibilidad y las regulaciones ambientales, la huella ecológica de las tecnologías digitales ha sido objeto de un escrutinio cada vez mayor.Entre estas, la tecnología blockchain suele ser criticada por su percibido alto consumo energético, en particular debido a las prácticas asociadas con las primeras redes públicas. Sin embargo, esta narrativa requiere una revisión crítica a la luz de los recientes avances en la arquitectura blockchain y los mecanismos de consenso. Esta sección explora la sostenibilidad ambiental de los sistemas blockchain, aclara conceptos erróneos comunes y destaca cómo decisiones de diseño bien pensadas, como las implementadas en la dApp educativa, pueden alinear el uso de blockchain con estrategias de cadena de suministro respetuosas con el medio ambiente.

Desmintiendo el mito del consumo energético

La idea errónea generalizada de que la tecnología blockchain consume excesiva electricidad se basa principalmente en las demandas energéticas de los mecanismos de consenso de Proof of Work (PoW), cuyo ejemplo más notable es Bitcoin. Las cadenas de bloques basadas en PoW requieren una gran potencia computacional para la minería, el proceso de resolver complejos problemas criptográficos para validar transacciones y añadir nuevos bloques a la cadena. Esto se traduce en un alto consumo energético y las consiguientes emisiones de carbono. El consumo energético anual de Bitcoin se ha comparado en ocasiones con el consumo eléctrico de países enteros como Argentina o los Países Bajos. El alto consumo energético de PoW ha suscitado fuertes críticas por parte de ambientalistas y legisladores. Sin embargo, es crucial reconocer que este alto consumo energético no es una característica universal de todas las tecnologías blockchain. La comunidad blockchain ha demostrado una creciente conciencia de estas preocupaciones ambientales y ha desarrollado y transitado activamente hacia alternativas más sostenibles.

3_2_Ethereum

Propuesto por **Vitalik Buterin** en 2013(Vitalik, 2013), Ethereum representa un importante paso evolutivo al introducir una **plataforma programable** que admite **contratos inteligentes y dApps**. La máquina virtual Ethereum (EVM) permite la ejecución de código Turing completo en una red global de nodos, lo que otorga a los desarrolladores una flexibilidad sin precedentes en implementaciones de blockchain anteriores.

Ilustración 2: Vitalik Buterin

Si bien Ethereum ha experimentado una adopción generalizada, su red ha enfrentado desafíos de escalabilidad, con una alta demanda que a menudo genera congestión y tarifas de transacción elevadas. Se han propuesto múltiples iniciativas para abordar estas limitaciones. La transición de PoW a “Proof of Stake” (PoS), completada en «The Merge» durante 2022, busca reducir el consumo energético y mejorar la eficiencia. Además, se espera que las técnicas futuras, como la fragmentación y varias soluciones de capa 2, mejoren el rendimiento y reduzcan los costos de transacción.

Los protocolos de consenso en Ethereum determinan qué bloque se añade a la blockchain. En PoW, los mineros deben resolver un rompecabezas criptográfico de alto consumo computacional, lo que en realidad implica "votar" por el siguiente bloque. Este enfoque mitiga los ataques Sybil al requerir recursos computacionales sustanciales.

En cambio, con PoS, los validadores "stake" o bloquean sus tokens en la red, y la probabilidad de proponer el siguiente bloque aumenta proporcionalmente al tamaño de su stake. Este cambio reduce significativamente el consumo de energía en comparación con los sistemas basados en PoW (como Ethereum y Bitcoin originales), al tiempo que aborda uno de los principales inconvenientes de los mecanismos de consenso de alta computación. La investigación en curso sobre Ethereum se centra en mejorar aún más el rendimiento, la escalabilidad y la seguridad, esfuerzos que se consideran cruciales para la adopción generalizada de aplicaciones descentralizadas.

Ethereum es una plataforma de blockchain descentralizada que permite la creación y ejecución de contratos inteligentes y aplicaciones descentralizadas.
Su moneda principal es el “Ether” abreviado comúnmente como ETH.

3_3_Contratos inteligentes

Ilustración 3: Nick Szabo

Los contratos inteligentes, conceptualizados por primera vez por **Nick Szabo** en 1997(Szabo, 1997), son programas diseñados para facilitar, verificar y ejecutar acuerdos automáticamente, sin intermediarios. Si bien el concepto es anterior a la cadena de bloques, no fue hasta la llegada de plataformas de cadena de bloques compatibles con dichos protocolos (en particular, Ethereum en 2015) que su implementación práctica se hizo viable. La introducción de la EVM y el lenguaje de programación Solidity en Ethereum permitió a los desarrolladores crear e implementar contratos inteligentes complejos y autoejecutables.

Desde entonces, los contratos inteligentes han revolucionado diversas industrias. En las finanzas descentralizadas (DeFi), sustentan sistemas financieros abiertos que operan sin intermediarios bancarios tradicionales. En la gestión de la cadena de suministro, mejoran la trazabilidad y la transparencia de los productos. También desempeñan un papel fundamental en la gobernanza descentralizada, facilitando una participación más democrática en los procesos de toma de decisiones organizacionales.

A pesar de su potencial transformador, los contratos inteligentes también plantean desafíos significativos. Dado que las transacciones de blockchain son inmutables, errores o vulnerabilidades en el código contractual pueden tener graves consecuencias. Un ejemplo destacado es el ataque DAO de 2016, en el que una falla de codificación provocó la pérdida de millones de dólares en Ether. Otras barreras incluyen limitaciones de escalabilidad e incertidumbres legales o regulatorias en torno a la exigibilidad de los contratos inteligentes. Además, la interoperabilidad entre plataformas sigue siendo un foco de investigación continua, cuyo objetivo es maximizar la versatilidad y la robustez de los contratos inteligentes en un ecosistema multicadena.

3_4_Solidity

Solidity es un **lenguaje de programación de alto nivel**, diseñado específicamente para escribir contratos inteligentes en la Máquina Virtual de Ethereum (EVM) y otras plataformas blockchain compatibles.
Estas son sus principales características:

- **Tipado estático.** Requiere que los tipos de datos de las variables se declaren explícitamente en el código.
- **Orientado a objetos.** Permite la creación de código modular, mejorando la legibilidad mientras reduce la duplicidad del código.
- **Herencia.** Cualidad adquirida para aprovechar la orientación a objetos, que le permite declarar un objeto cuyo código pueda ser heredado para poder ser reutilizado por otro código sin la necesidad de reescribirse.
- **Herencia múltiple.** Cualidad que permite a un contrato heredar de varios contratos a la vez
- **Orientado a contratos inteligentes.** Cualidad de los programas que se pueden ejecutar en una red blockchain.
- **Inmutabilidad de los contratos.**
- Los contratos inteligentes, una vez desplegados en la red, no pueden ser mutados, ni eliminados. Esto permite a los contratos hacer simultáneamente de gestor de datos, a la vez que ejecuta la lógica de negocio.
- **No repudio.** Los contratos pueden ser firmados, cuando son firmados, dado el carácter inmutable de estos, esto permite mantener todas las interacciones de cada usuario sobre una red blockchain.

Solidity, al ser un lenguaje orientado a contratos inteligentes, ha de ser programado con especial atención en el coste de las funciones, así como el coste del almacenamiento del bytecode.
Para ello, se explotarán las ventajas de los distintos tipos de datos de este lenguaje.
Tipos de datos principales:

- Booleanos (bool). Datos con solo dos valores: true y false.
- Enteros (int uint). Enteros con o sin signo, por defecto ocupan int256 y uin256, se puede reducir en potencias de 2, especifican su tamaño. Ej. uint8 ($8 = \frac{256}{2^5}$)
- Dirección (address). Direcciones de memoria.
- Enumeraciones (enum). Listas con valores predefinidos por el programador.
- Arreglos (Arrays). Listas de tamaño “indefinido”.
- Cadenas de texto (strings).
- Estructuras (structs). Tipo de datos hecho por el programador con otros tipos de datos.
- Mapas (mapping). Par de valores clave->valor.

3_4_1_Modificadores de variables

- **Public**[1]. Variable de acceso público, crea automáticamente un getter.
- **Internal.**
- **Private.**
- **Constant**. No puede ser sobrescrito. Ha de ser inicializado en su declaración.
- **Immutable**. No puede ser sobrescrito una vez inicializado.
- **Payable**. Permite a un variable address.
- **Storage**. Ubicación permanente en la blockchain.
- **Memory**. Ubicación temporal en la blockchain.
- **Calldata**. Solo lectura.

3_4_2_Modificadores de funciones

Solidity presenta una serie de palabras clave, que permite modificar el comportamiento de una función. Además, también te permite crear tus propios modificadores de función.

3_4_2_1_Visibilidad

- **Public**. Cualquier elemento o usuario puede llamar a esta función. Por defecto serán públicas.
- **External**. La función solo podrá ser llamada desde elementos externos al contrato; ya sean, otros contratos, o un usuario.
- **Internal**. La función solo podrá ser llamada desde dentro del contrato, o desde un contrato que herede del contrato que la contiene.
- **Private**. Esta función solo puede ser llamada por el propio contrato; los contratos que hereden a un contrato con una función privada, no podrán ver la función; pero, sí podrán llamar a funciones heredadas que llamen a una función privada.

3_4_2_2_Estado

- **View**. Solo lee el estado, no permite modificar datos del contrato inteligente.
- **Pure**. No accede ni lee el estado, solo opera con parámetros aportados.

Por defecto, una función podrá leer y modificar variables que pertenezcan al contrato inteligente.

3_4_2_3_Pago

- Payable. Indica que la función puede almacenar Ether, para que más adelante sea recogido.

3_4_2_4_Herencia

- Virtual. Permite que una función sea sobrescrita por un contrato hijo.
- Override. Permite sobrescribir una función virtual.

3_4_2_5_Personalizados

Solidity presenta un modificador de función específico, para unir código, que se unirá al principio, o al final de la función.

[1] El modificador “public” para una variable en Solidity, al contrario que en otros lenguajes, no permite sobrescribir el valor de una variable; es decir, no crea un setter

```
modifier soloAdmin() {
    require(msg.sender == admin, "No autorizado");
    _;
}

function reset() public soloAdmin {
    contador = 0;
}
```

Ilustración 4: Modificador personalizado

En esencia, esto permite facilitar requisitos para el lanzamiento de una función.
Se pueden poner varios modifiers a una función. Se activarán en orden.
Primero se ejecutarán los definidos como en el ejemplo, después se ejecutará el código de la función, y finalmente se lanzarán los modifiers creados para el final del código.
La forma en la que se define si el modifier se lanzará al principio o al final, será mediante la instrucción "_;", pues reemplazará al resto de código que no pertenece al modifier. Si se pone al final del modifier, la función hará de cabecera; si se pone al principio, la función actuará como pie de función.

Se recomienda no usar modifiers como pie de función si presentan "require", pues esto aumentará el coste de revertir los cambios de la función, en caso de que no se cumpla el flujo correcto del funcionamiento de la función.

Para permitir que una función almacene Ether, se podrá añadir el modificador "payable".

3_5_Remix

Remix es un **entorno de desarrollo integrado** (***IDE***) basado en la web diseñado para el desarrollo de contratos inteligentes utilizando en lenguaje Solidity.

Su utilidad principal, es hacer de **maquina virtual** como **entorno de pruebas.**

3_5_1_File explorer

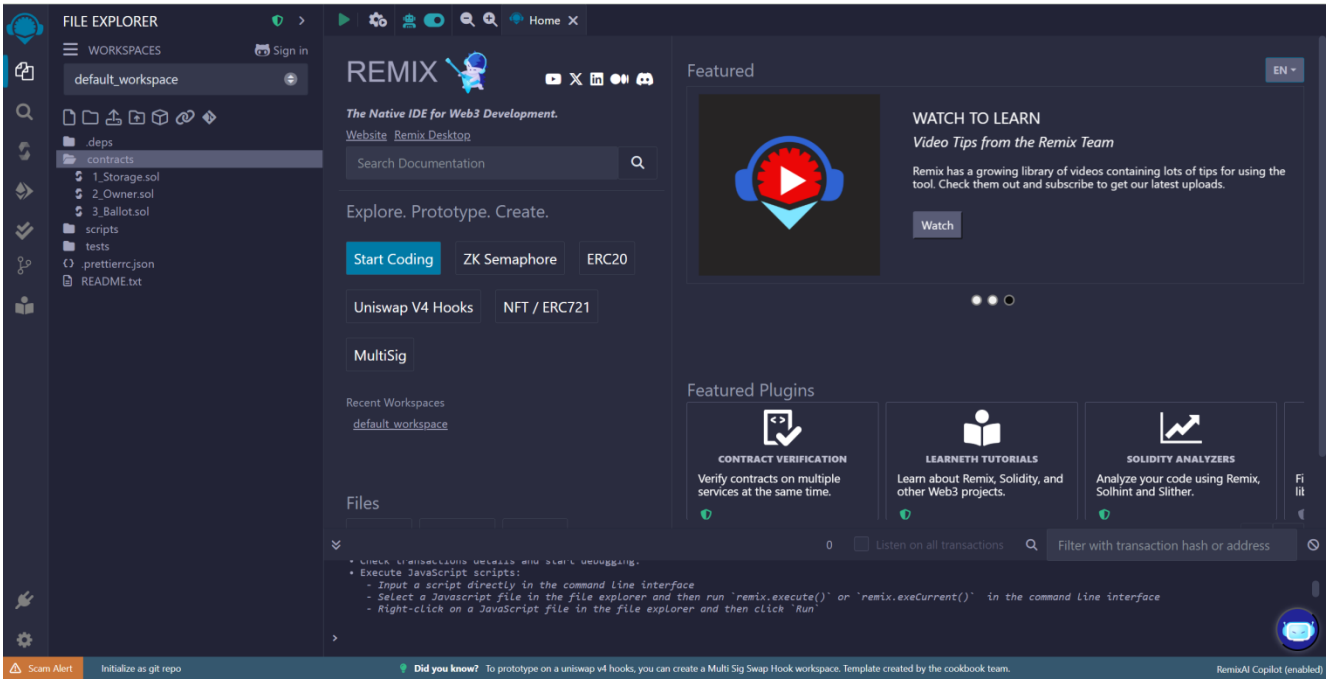


Ilustración 5: Explorador de archivos de Remix

En la pestaña de “File explorer”, podemos organizar nuestros contratos inteligentes (archivos.sol), nuestros despliegues (scenarios.json), nuestros test unitarios, y muchos más elementos que no serán necesarios para la explicación de este TFG.

Según iniciemos Remix, tendremos 3 contratos disponibles preparados para iniciarse, estos archivos se llaman “1_Storage.sol”, “2_Owner.sol” y “3_Ballot.sol”.

3_5_2_Solidity compiler

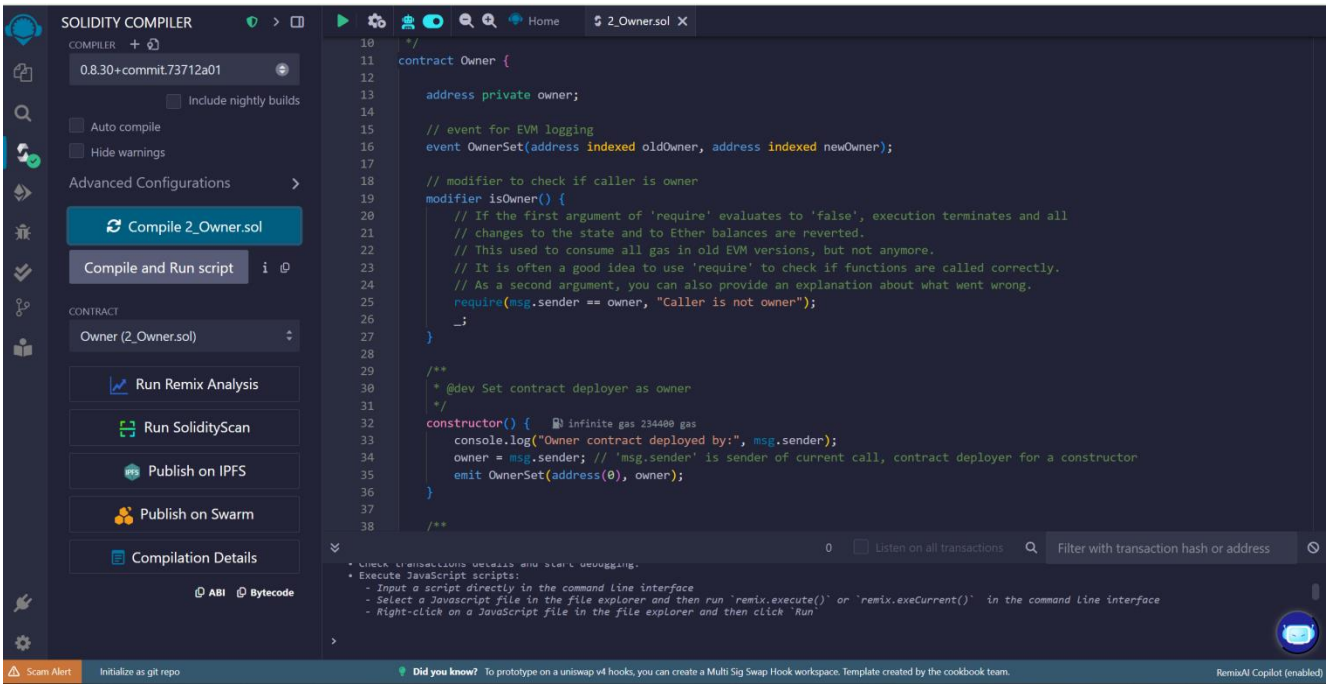


Ilustración 6: Solidity compiler de Remix

En la pestaña de “Solidity compiler” deberemos compilar nuestros códigos fuentes antes de poder desplegar nuestros contratos inteligentes.

Para esta explicación, usaremos el código que nos da por defecto Remix, cuyo nombre es “2_Owner.sol”.

3_5_3_Deploy and run transactions

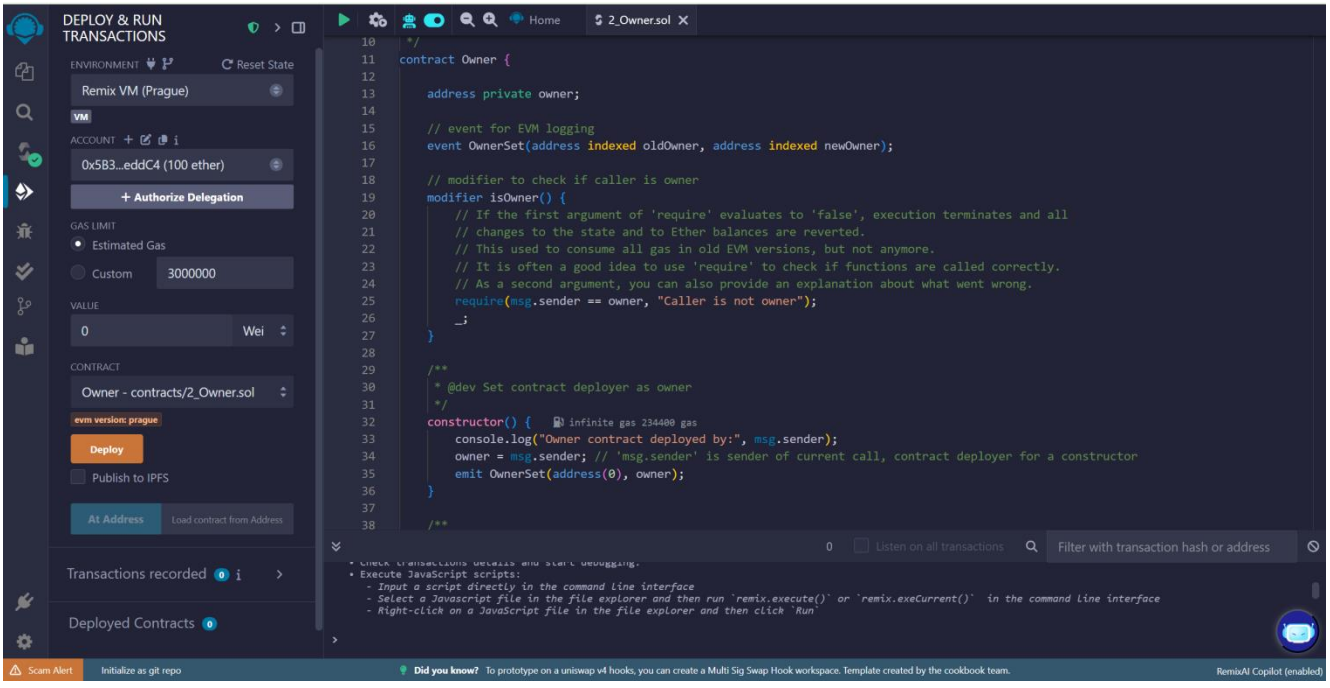


Ilustración 7: Despliegue de un contrato

En la pestaña de “Deploy and run transactions”, podemos desplegar nuestros contratos inteligentes recientemente compilados.

A la hora de desplegar un contrato, se nos permite realizar ciertas modificaciones a este, que afectarán a la manera en la que los usuarios podrán interactuar.

La primera, es el **límite de gas**, que se puede gastar en una interacción del contrato. Indicado con "GAS LIMIT".
La segunda, y la que vamos a ver con este código de ejemplo, es el **propietario del contrato inteligente** cuando este se despliegue. Indicado con "ACCOUNT".
La tercera, es **localizar un contrato**, dada su dirección de alojamiento en la blockchain.

Como se puede ver en el código, este contrato, almacenará el identificador de la cuenta del usuario que la haya desplegado.

Es muy importante, a la hora de usar remix, **seleccionar la cuenta correcta**, cada vez que se vaya a llamar a una función, pues pueden tener distintas interacciones en función de quien la llame.
Una vez se termine la fase de desarrollo, y se despliegue en una blockchain; no será necesario identificar la cuenta, pues se realizará con la extensión de navegador MetaMask.

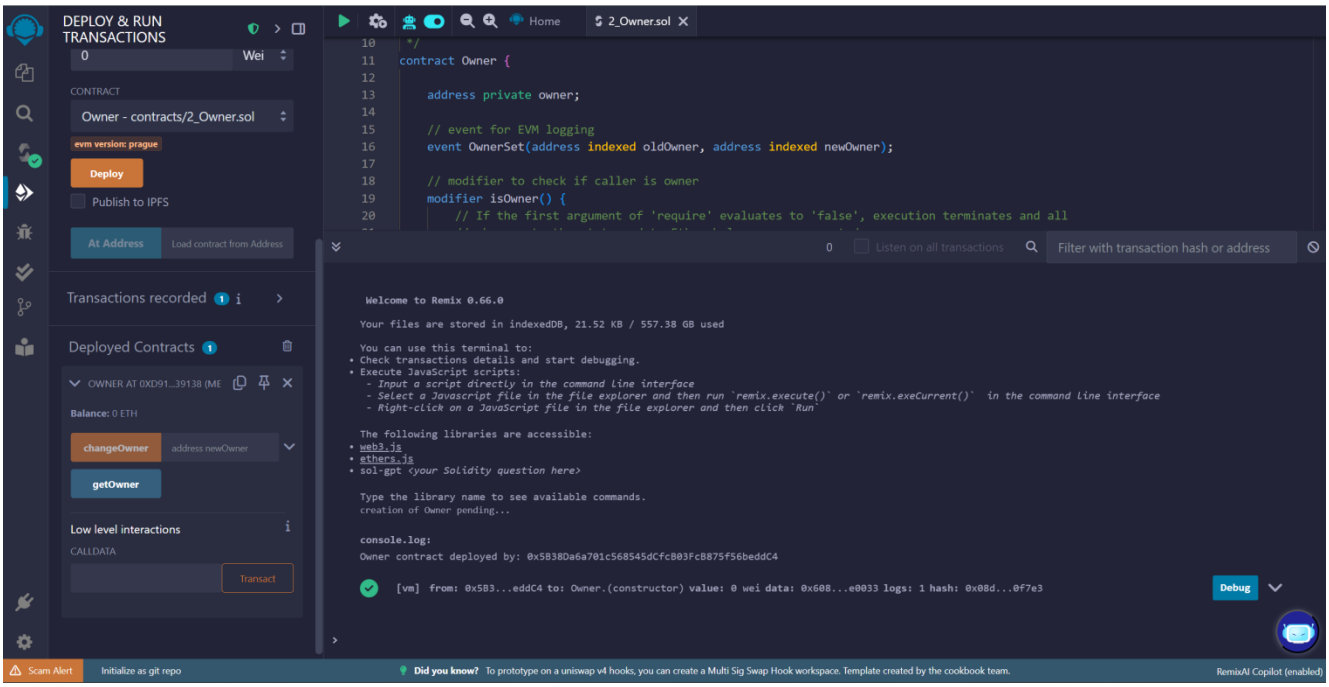


Ilustración 8: Contrato ya desplegado

Aquí podemos ver el contrato inteligente de ejemplo, ya desplegado, con la confirmación de la transacción, en la parte posterior derecha de la imagen.

En la parte posterior izquierda, podemos observar dos botones de distintos colores
Los botones azules, indican una llamada al contrato, que no tiene coste.
Los botones naranjas, indican una llamada al contrato, que altera el contrato; y en consecuencia, implica un coste.

Cada función, tendrá que ser una llamada al contrato inteligente; luego, será necesario que cada función esté en un botón azul, o en uno naranja.

Esto se lo podemos indicar al contrato inteligente con la palabra reservada “view”; al poner este modificador[2] a una función, le indicamos que la función es una llamada de consulta de datos.

```
/**
 * @dev Return owner address
 * @return address of owner
 */
function getOwner() external view returns (address) {
    return owner;
}
```

Ilustración 9: Función externa sin coste

Ilustración 10: Función con y sin modificador view

[2] El compilador comprueba que el modificador sea correcto; luego, no se puede engañar a Solidity poniendo view, para reducir los costes

3_6_MetaMask

Extensión de navegador que permite a un usuario gestionar su propia billetera de criptomonedas.
Permite a los usuarios enviar y recibir criptomonedas; además de gestionar sus claves privadas, que más adelante podrán usar para firmar contratos inteligentes.

3_7_Astah Professional

Astah es una **herramienta de modelado visual**, que permite entender la forma en la que interactúa un actor con el sistema, además de permitir ver, como interactúa el sistema consigo mismo.
Astah tiene muchas versiones con distintos planes de subscripciones.
Esta es una versión con subscripción de pago; la Uva, tiene licencia para el uso de esta versión.

3_8_Word 2007

Microsoft Word es un **procesador de textos** desarrollado por Microsoft, utilizado para crear, editar y formatear documentos. Esta memoria de TFG ha sido escrita con Word 2007.
Se eligió Word antes que LaTeX, debido a lo mala que es la herramienta de LaTeX.
El motivo por el que se eligió esta versión, es debido a la comodidad que ofrece esta versión con su barra de herramientas, la cual es bastante intuitiva y cómoda.
Requiere licencia.

3_9_Excel

Excel es un **programa de hoja de cálculo** desarrollado por Microsoft que permite a los usuarios organizar, analizar y manipular datos.
Se ha utilizado Excel, para realizar la tabla de Riesgos y oportunidades
Requiere licencia.

3_10_Outlook

Outlook **incluye aplicaciones** de correo electrónico y calendario, además de integraciones con otras aplicaciones de Microsoft como Word, Excel, PowerPoint y OneDrive, así como con servicios de terceros como Teams y Zoom.

De todas estas aplicaciones solo he usado su correo electrónico, concretamente el proporcionado por la Uva, con el fin de pasarme archivos a mí mismo cuando su tamaño era demasiado grande como para pasarlo por otras herramientas, y con el fin de contactar con el tutor.

3_11_Discord

Discord es una **plataforma gratuita de comunicación** por voz, video y texto, que permite a usuarios conectarse con amigos, comunidades de juegos y desarrolladores.

Esta herramienta se ha utilizado con el fin de organizar el TFG, transferir archivos, anotar cambios necesarios, y especificaciones a seguir.

3_12_WhatsApp

WhatsApp es una aplicación de mensajería instantánea; aunque hoy en día, cada vez se está convirtiendo más en una red social enfocada en comunidades como Instagram o Telegram.

Se ha utilizado con el fin de comunicarme con mi tutor, organizar reuniones presenciales y para anotar cosas urgentes que surgían durante el TFG.

4_Estudio Preliminar de la Interacción Interempresarial

4_1_Proveedor

En paralelo, el proveedor involucrado en la negociación presenta una estructura más simplificada, pero igualmente funcional en relación con los objetivos del proyecto. En este análisis se contemplan dos departamentos principales, aunque uno de ellos queda fuera del alcance del presente estudio:

- **Departamento de Ingeniería:** Aunque presente en la estructura del proveedor, este departamento no resulta relevante para el alcance de este trabajo, ya que el foco no se encuentra en la capacidad técnica de fabricación del proveedor, sino en los aspectos organizativos y contractuales de la negociación.
- **Departamento Comercial:** Es el encargado de gestionar la relación directa con el fabricante, actuando como interlocutor principal en las negociaciones económicas y logísticas. Su función consiste en revisar los requerimientos planteados por el fabricante, evaluar la viabilidad de los plazos y costes, y proponer una oferta ajustada a las capacidades internas del proveedor. A su vez, este departamento es responsable de garantizar que los compromisos adquiridos puedan cumplirse, lo que implica una coordinación estrecha con otras áreas internas.

Esta estructura organizativa, tanto del fabricante como del proveedor, evidencia la necesidad de establecer un canal de comunicación formalizado y eficiente entre ambas partes. Dado que en el **modelo tradicional** muchas de estas **interacciones** se realizan de forma **verbal o** a través de **correos electrónicos sin respaldo contractual** sólido, se generan situaciones de ambigüedad, descoordinación o incluso disputas posteriores. Este contexto es precisamente el que motiva la exploración del uso de contratos inteligentes como mecanismo para estructurar y automatizar los acuerdos alcanzados durante el proceso de negociación.

Estructura organizativa del proveedor

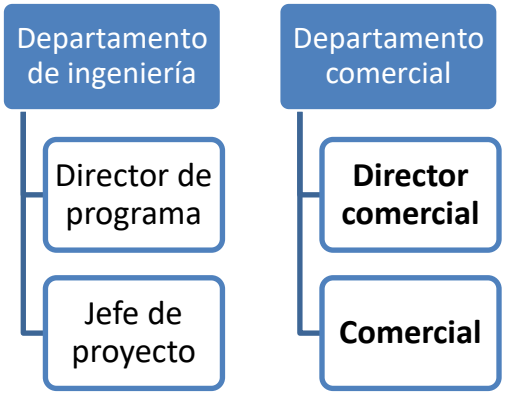


Ilustración 11: Estructura organizativa del proveedor

4_2_Fabricante

En el marco del presente estudio, se toma como referencia la estructura organizativa de un fabricante del sector industrial que persigue el desarrollo de un producto innovador, para el cual se requieren componentes específicos que, en muchos casos, no se encuentran disponibles en el mercado convencional o deben ser adaptados a requerimientos altamente técnicos. Esta situación exige una estrecha colaboración entre diversos departamentos internos del fabricante, así como una coordinación efectiva con los proveedores externos.

El fabricante presenta una estructura basada en tres departamentos principales, cuyas funciones están claramente delimitadas pero requieren de una interacción fluida para alcanzar los objetivos comunes:

- **Departamento de Ingeniería:** este departamento es el principal interesado en la obtención de prototipos de las piezas necesarias para el diseño y desarrollo del nuevo producto. Su función abarca desde la definición de los requisitos técnicos de los componentes, hasta la validación de las propuestas recibidas por parte de los proveedores. En este sentido, su rol es esencial para garantizar la viabilidad técnica del producto final, así como para establecer los criterios que orientarán la negociación posterior.
- **Departamento de Logística:** encargado de la planificación, coordinación y supervisión de todas las actividades relacionadas con el flujo de materiales y componentes dentro de la organización. Este departamento actúa como nexo entre las distintas áreas implicadas, gestionando tanto los aspectos operativos como la programación de entregas, y asegurando que los tiempos y recursos estén alineados con las fases del proyecto. En el contexto del proceso de negociación, su intervención es clave para validar la viabilidad temporal y operativa de los acuerdos propuestos.
- **Departamento de Compras:** su misión principal es formalizar la adquisición de los componentes una vez que estos han sido aprobados por los departamentos técnicos y logísticos. No participa directamente en la definición de requisitos ni en la evaluación técnica de los prototipos, pero sí asume la responsabilidad contractual de cerrar los acuerdos económicos con los proveedores. Su actuación suele estar condicionada por los informes previos emitidos por Ingeniería y Logística, lo que lo convierte en el punto de cierre del proceso interno de decisión.

La correcta articulación entre estos tres departamentos permite que el fabricante actúe de manera coordinada, reduciendo riesgos y asegurando que las decisiones tomadas reflejen tanto las necesidades técnicas como las realidades logísticas y presupuestarias del proyecto. Sin embargo, esta estructura también introduce una cierta complejidad en la negociación con los proveedores, al requerir múltiples validaciones internas antes de emitir una aprobación final.

Estructura organizativa del fabricante

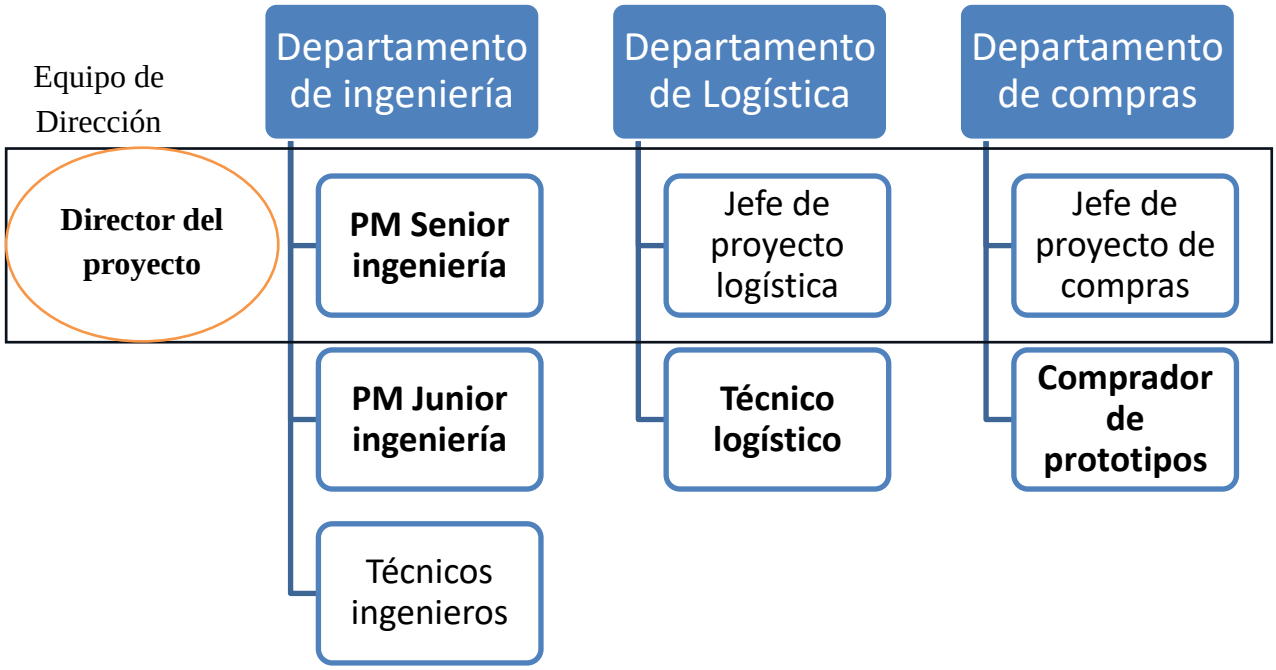


Ilustración 12: Estructura organizativa del fabricante

4_3_Flujo de una negociación real

Una negociación, desde el principio, transcurre de esta manera:

1. Un miembro del departamento de ingeniería le comenta al "Técnico Logístico" que se requiere un prototipo de algún tipo.
2. El "**Técnico Logístico**" le realiza una consulta al proveedor, sobre la factibilidad de una oferta para un prototipo.
3. El "**Comercial**" rechaza o acepta la oferta. Si la acepta, ir al paso 10.

Pasa una semana sin emitirse orden de compra.

4. El "**Project Manager Junior**" realiza una nueva oferta.
5. El "**Comercial**" rechaza o acepta la oferta. Si la acepta, ir al paso 10.

Pasa una semana sin emitirse orden de compra.

6. El "**Project Manager Senior**" realiza una nueva oferta.
7. El "**Director Comercial**" rechaza o acepta la oferta. Si la acepta, ir al paso 10.

Pasa una semana sin emitirse orden de compra.

8. El "**Director del Proyecto**" realiza una nueva oferta.
9. El "**Comercial**" acepta o rechaza la oferta. Si la rechaza, volver al paso anterior.
10. El "**Comprador de Prototipos**" emite un contrato y efectúa el pago.

5_Análisis

5_1_Requisitos

5_1_1_Funcionales

RF-01	El sistema deberá permitir al “Técnico Logístico” realizar la oferta inicial
RF-02	El sistema deberá permitir al “Project Manager Junior” realizar una oferta
RF-03	El sistema deberá permitir al “Project Manager Senior” realizar una oferta
RF-04	El sistema deberá permitir al “Director de Proyecto” realizar una oferta
RF-05	El sistema deberá permitir al “Comprados de Prototipos” emitir un contrato cuando se haya aceptado una oferta
RF-06	El sistema deberá permitir al “Comercial”, aceptar una oferta
RF-07	El sistema deberá permitir al “Comercial”, rechazar una oferta
RF-08	El sistema deberá permitir al “Director Comercial” aceptar una oferta
RF-09	El sistema deberá permitir al “Director Comercial” rechazar una oferta

Tabla 3: Requisitos funcionales

5_1_2_No funcionales

- El sistema no deberá permitir a un usuario estar asociado a más de un rol por contrato.
- Cada usuario, realizará tareas asociadas a su rol.
    - En la estructura organizativa del fabricante, tras el periodo de una semana, si no ha habido éxito en la negociación de un precio, se pasará la negociación a un rol superior de esta forma.
      “Técnico Logístico” ->“PM Junior” ->“PM Senior” ->“Director de Proyecto”
    - En la estructura organizativa del proveedor, tras el periodo de dos semanas, si no ha habido éxito en la negociación de un precio, se pasará la negociación a un rol superior de esta forma.
      “Comercial” ->“Director Comercial”

5_2_Especificación caso de uso

Nombre	Lanzar orden de factibilidad
Actores principales	“Técnico Logístico”, “Comercial” y “Comprador de Prototipos”
Actores secundarios	“PM Junior”, “PM Senior”, “Director de Proyecto” y “Director Comercial”
Descripción	Proceso de negociación entre varias figuras del área técnica y comercial, para la aceptación de una oferta de prototipo y la emisión de un contrato
Precondiciones	Ninguno
Postcondiciones	Una vez aceptada la oferta y emitido el contrato, **no se podrá volver a interactuar con el contrato inteligente.**
Flujo principal	1. El “Técnico Logístico” consulta con el proveedor sobre la factibilidad de una oferta 2. El “Comercial” rechaza la oferta. [Pasa una semana sin haber confirmación] 3. El “Project Manager Junior” emite una nueva oferta. 4. El “Comercial” rechaza la nueva oferta. [Pasa una semana sin haber confirmación] 5. El “Project Manager Senior” emite una nueva oferta. 6. El “Director Comercial” rechaza la oferta. [Pasa una semana sin haber confirmación] 7. El “Director del Proyecto” emite una nueva oferta. 8. El “Comercial” acepta. 9. El “Comprador de Prototipos” emite un contrato y realiza el pago.
Flujos alternativos	En los pasos 3, 5 y 7, si la oferta es aceptada, se va al paso 9. En el paso 9, si el Comercial rechaza la oferta del Director del Proyecto, se vuelve a intentar desde ese mismo nivel (paso 7).
Requisitos relacionados	Ninguno

Tabla 4: Lanzar orden de factibilidad

5_3_Diagrama de caso de uso

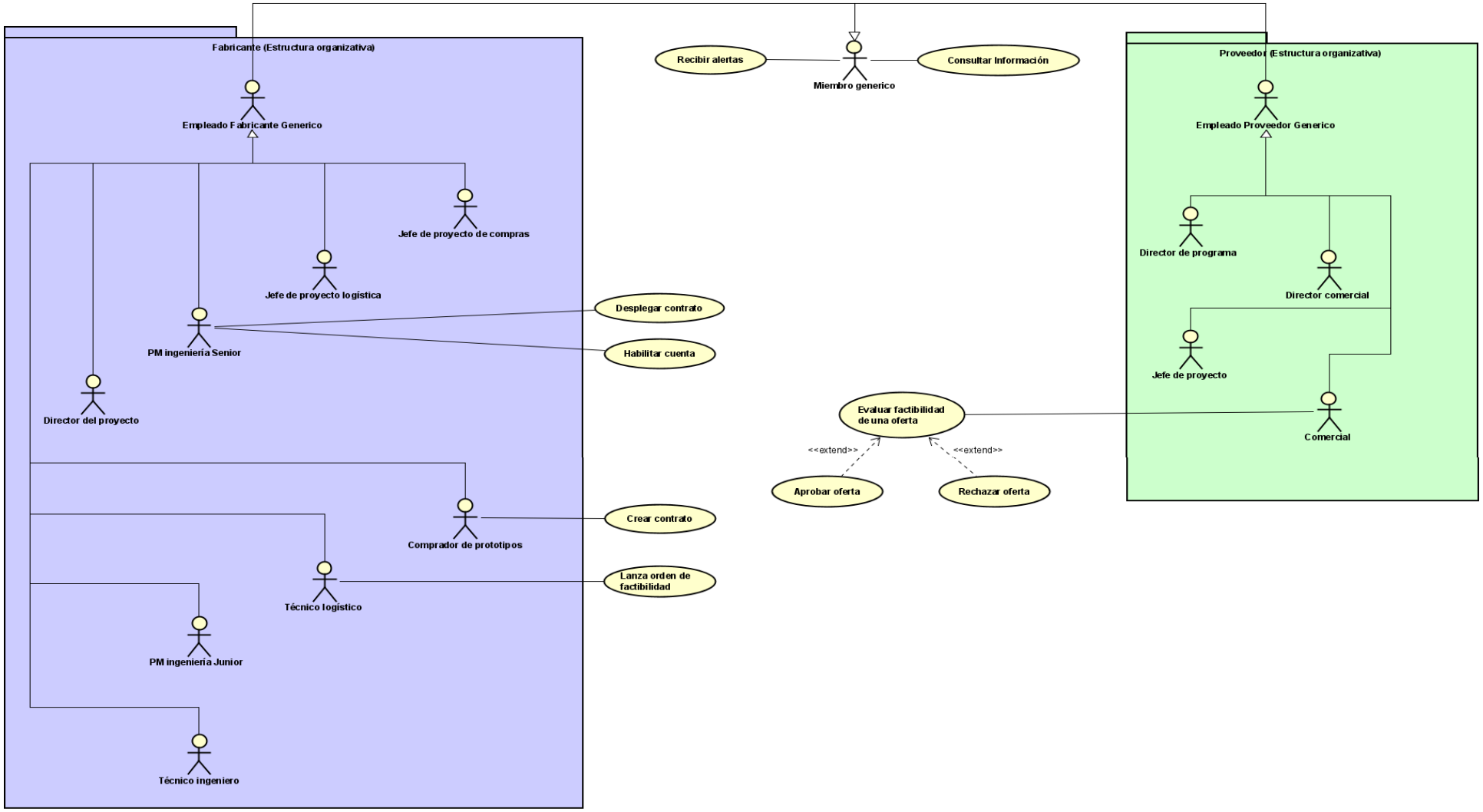
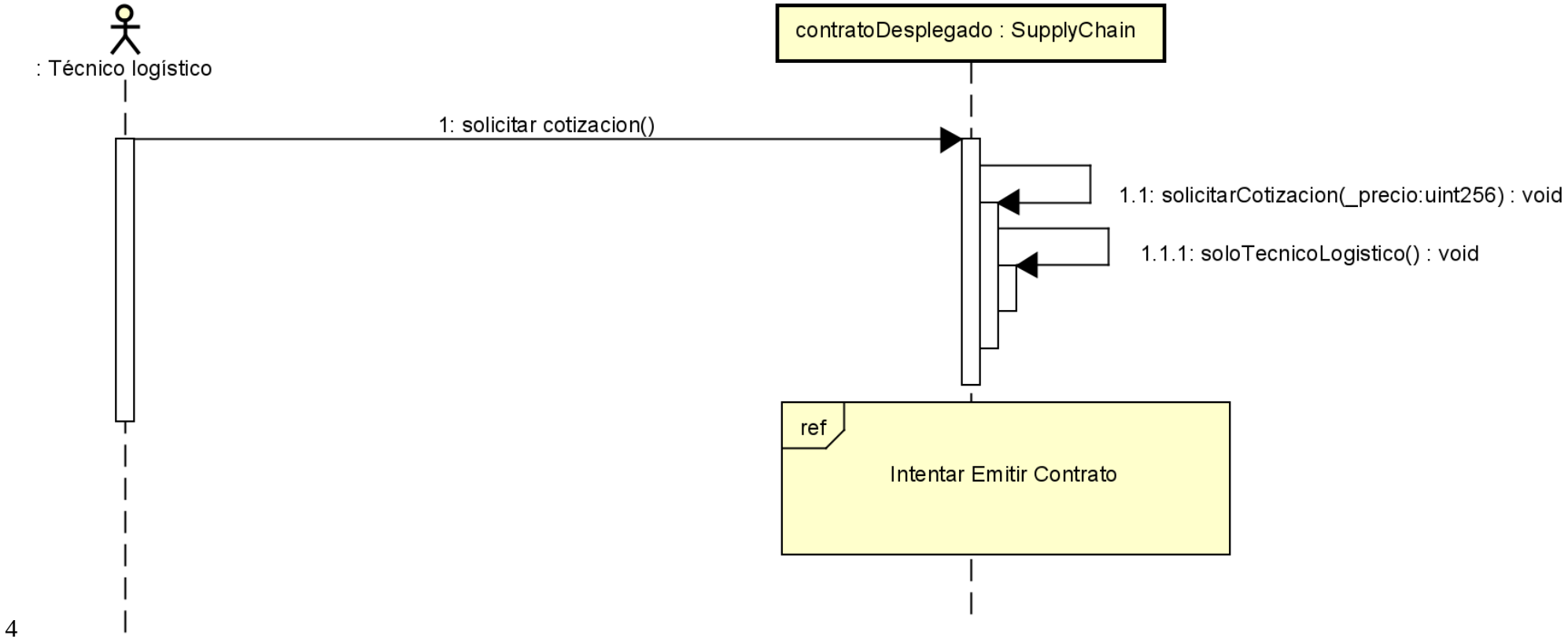


Ilustración 13: Diagrama de caso de uso que modela una cadena de suministros de Renault

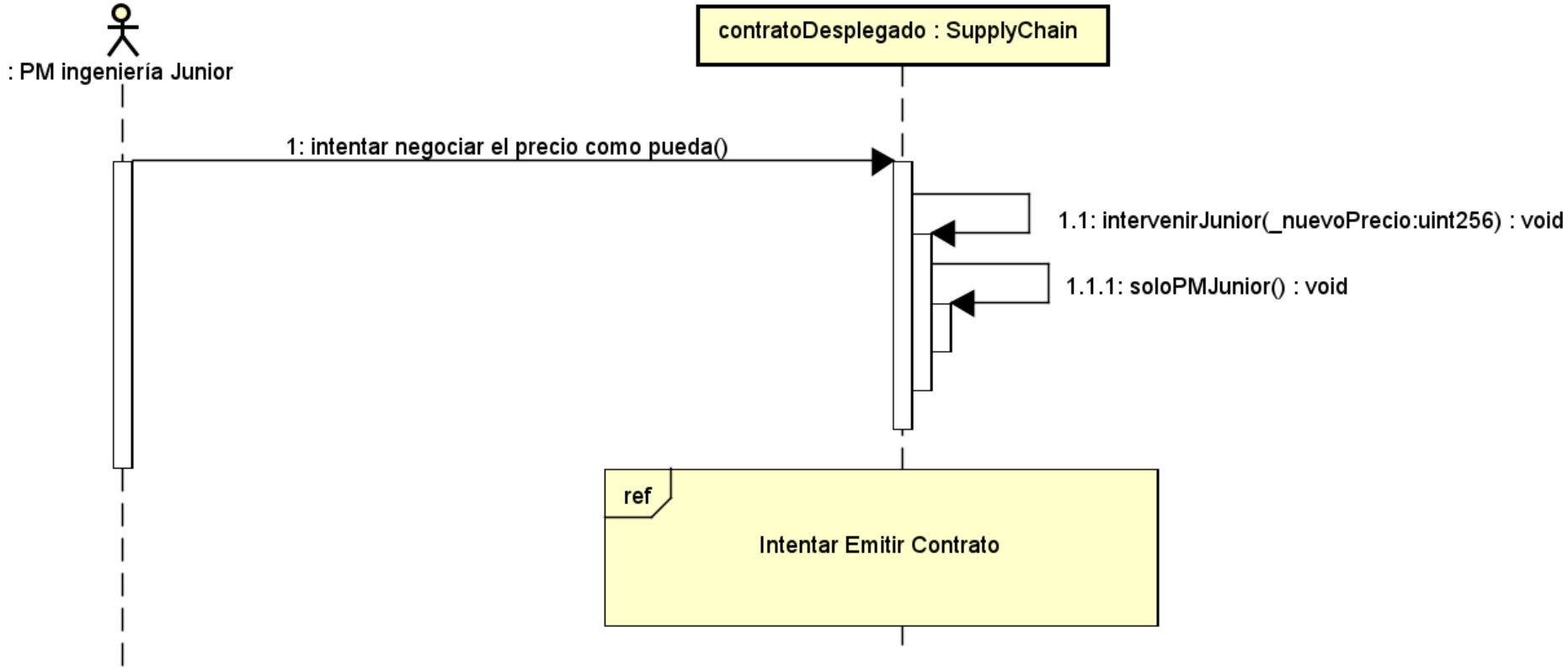
6_Diseño

6_1_Diagramas de secuencias



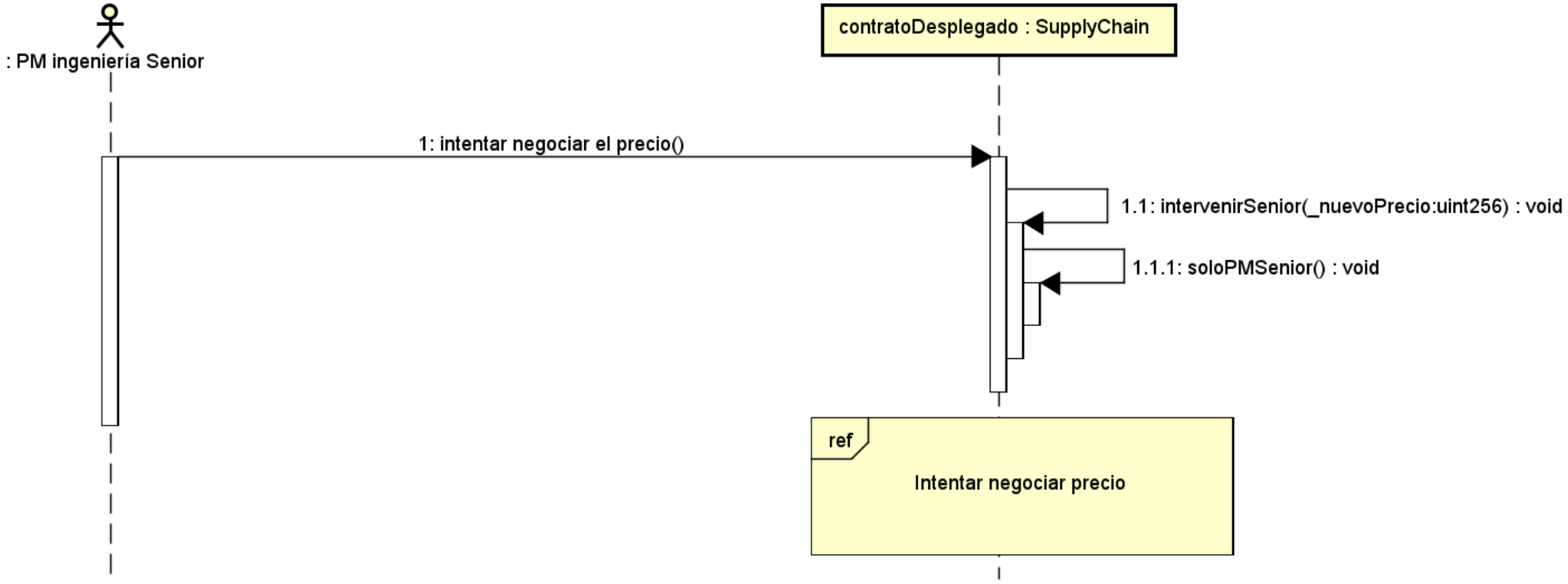
4

Ilustración 14: Interacción exclusiva del Técnico Logístico



4

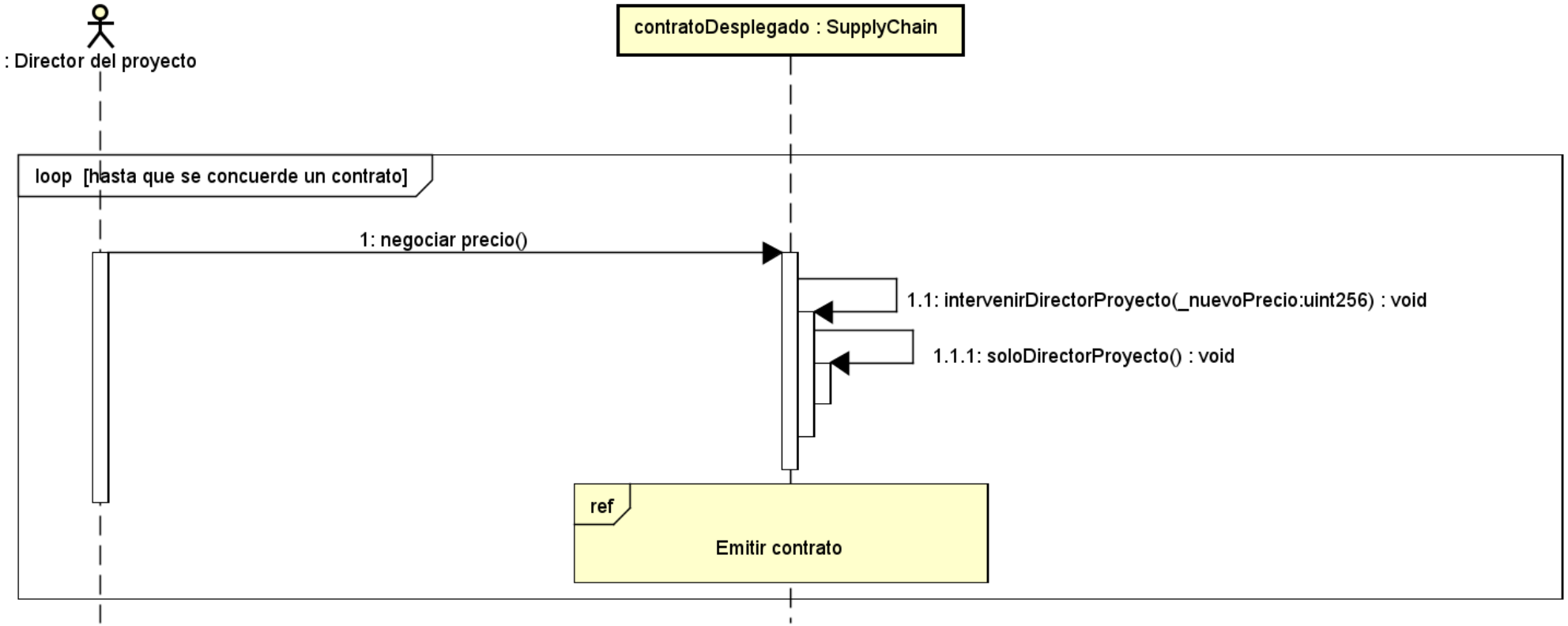
Ilustración 15: Interacción exclusiva del PM Junior



4 3

Ilustración 16: Interacción exclusiva del PM Senior

[3] Se recuerda a los lectores: al llegar a la referencia al otro caso de uso, interactuará el Director Comercial en lugar del Comercial

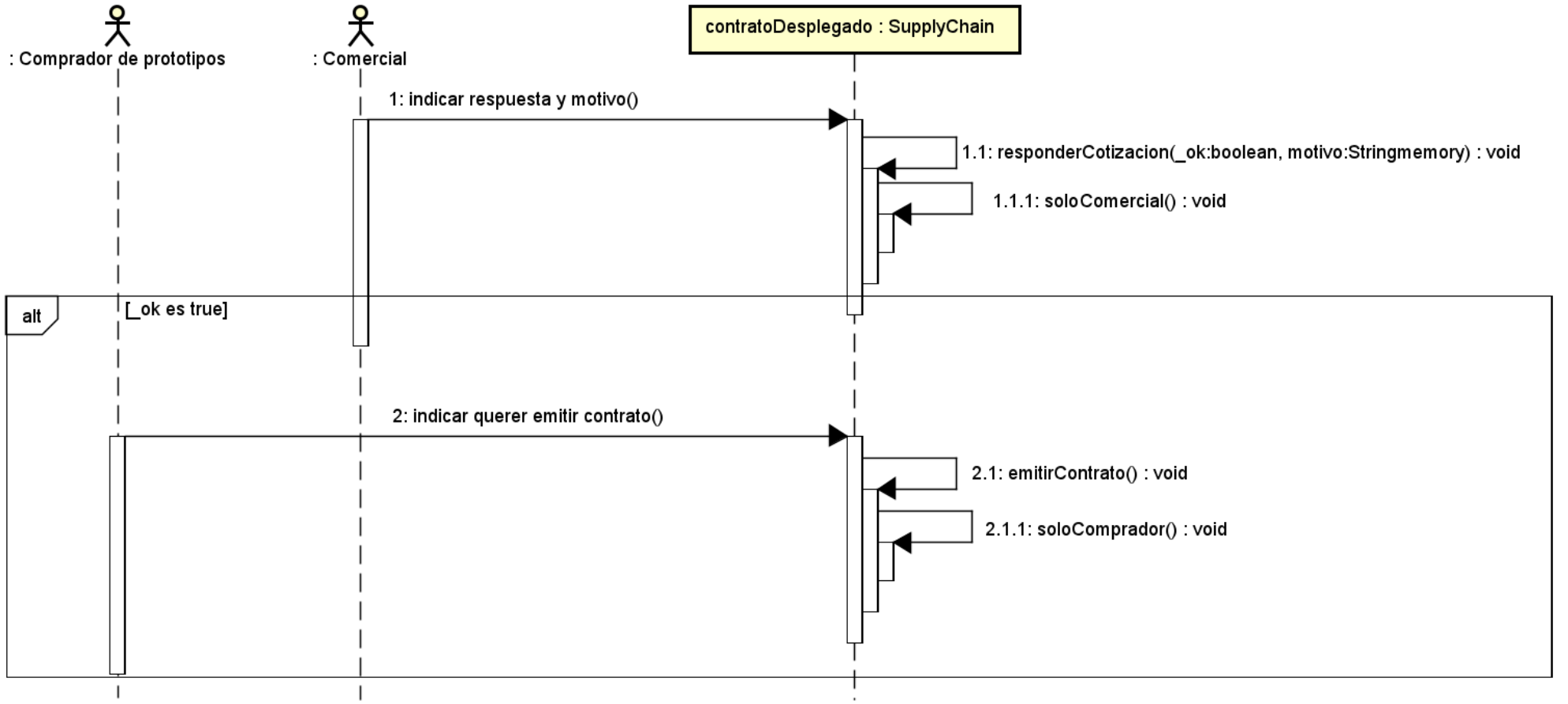


4

Ilustración 17: Interacción exclusiva del Director de Proyecto

En este diagrama se declara el fin de la cadena de suministros.

El director de proyecto (rol con mayor responsabilidad en la cadena de suministro) mandará contraofertas, hasta que el comercial acepte una de sus ofertas.



[4]

Ilustración 18: Rechazo o confirmación de una cotización

El comercial, tras haber visto la oferta aportada por alguno de los roles de los anteriores diagramas de secuencia, evaluará la oferta, y enviará una respuesta.

Si la respuesta es positiva; es decir, se ha aceptado la oferta, se procederá a emitir el contrato por parte del comprador de prototipos.

[4] Diagrama de secuencia "Emitir contrato"

6_2_Diagrama de clases

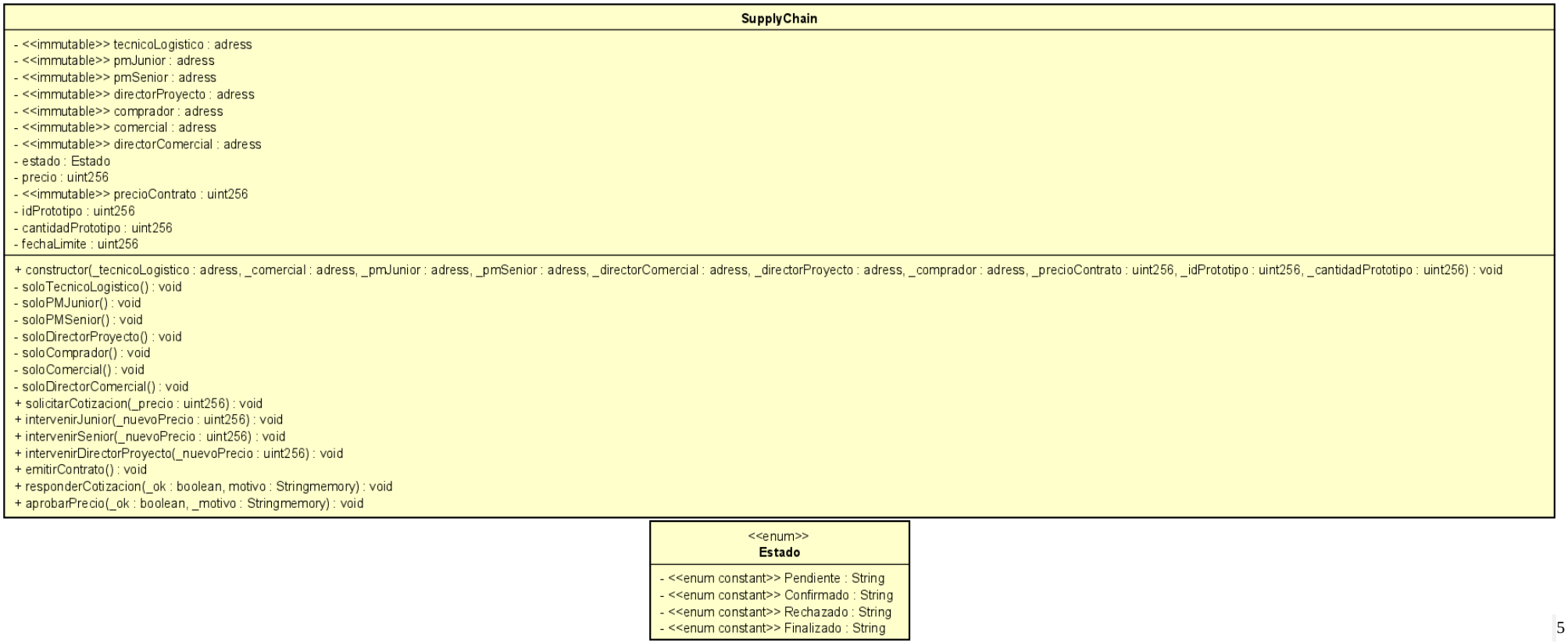


Ilustración 19: Diagrama de clases

La clase SupplyChain, se encargará de procesar los distintos pasos de la negociación de un producto. En nuestro caso, debido a que interactuarán hasta siete personas con distintos roles, la clase parece tener más complejidad de la que realmente tiene.

Se guardará el precioContrato, que será el precio establecido por los contratos previamente contractualizados; en términos simples, será un límite superior

.

[5] Pese a no estar marcado explícitamente, se recuerdo a los lectores, que todas las variables señaladas como privadas, pueden ser consultadas con un getter.

6_3_Diagrama de estados

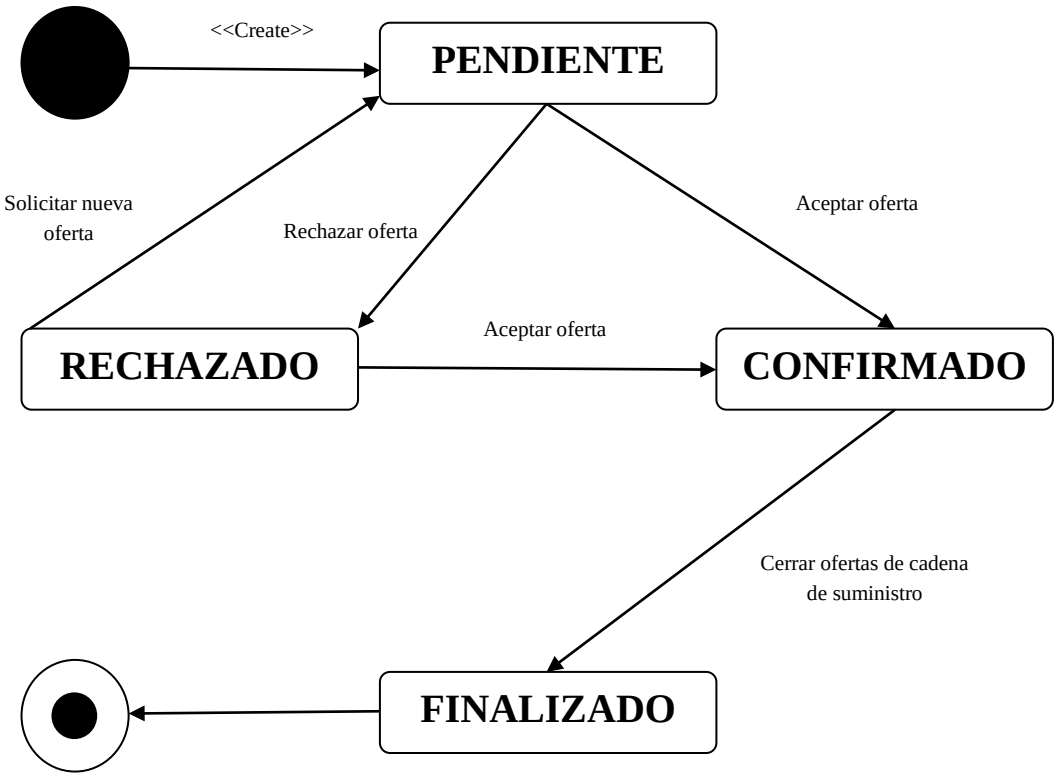
El diagrama de clases, tiene una única clase, además de un “Enum” .

El enum Estado, reflejará el estado actual de la negociación de un producto en una cadena de suministro.

Una cadena de suministro se crea como “Pendiente”.

Una cadena de suministro pasará a Rechazado, cuando el Comercial o el Director Comercial, rechacen una oferta; asimismo, una cadena de suministro pasará a Confirmado, cuando una de los dos roles mencionada anteriormente, acepte una oferta de un producto.

Una cadena de suministro, pasará a Finalizado, una vez que haya estado Confirmado, y se indique, que se ha de emitir un nuevo contrato.



6_4_Patrones conocidos usados

6_4_1_Patrón Plantilla

El patrón plantilla permite que una clase defina el flujo de un algoritmo (los pasos que lo componen), y que las subclases puedan modificar o personalizar algunos de esos pasos.

En nuestro caso, usaremos el método plantilla con las funciones de nuestro contrato de Solidity y sus modifiers.

```
modifier soloTecnicoLogistico() {
    require(msg.sender == tecnicoLogistico, "Solo el tecnico logistico puede ejecutar esta accion");
    _;
}

modifier soloComercial() {
    require(msg.sender == comercial, "Solo el comercial puede ejecutar esta accion");
    _;
}

modifier soloPMJunior() {
    require(msg.sender == pmJunior, "Solo el PM Junior puede ejecutar esta accion");
    _;
}

modifier soloPMSenior() {
    require(msg.sender == pmSenior, "Solo el PM Senior puede ejecutar esta accion");
    _;
}

modifier soloDirectorComercial() {
    require(msg.sender == directorComercial, "Solo el Director Comercial puede ejecutar esta accion");
    _;
}

modifier soloDirectorProyecto() {
    require(msg.sender == directorProyecto, "Solo el Director del Proyecto puede ejecutar esta accion");
    _;
}
```

Ilustración 20: Funciones modifiers

Esto servirá como cabecera para modificar el comportamiento inicial de las funciones.
De esta forma, estos dos códigos interactuarán de forma distinta, pese a su identidad en el código declarado en la función.

```
function negociarPrecio(uint256 _nuevoPrecio) external soloPMJunior {   infinite gas
    require(estado == Estado.NoOk, "No se puede negociar si no hay rechazo previo");
    require(_nuevoPrecio <= precioContrato, "El precio debe ser igual o menor que el precio contrato");

    precio = _nuevoPrecio;
    estado = Estado.Confirmado;

    emit CotizacionConfirmada(msg.sender, precio);
}
function intervenirPM(uint256 _nuevoPrecio) external soloPMSenior {   infinite gas
    require(estado == Estado.NoOk, "No se puede intervenir si el precio ya fue confirmado");
    require(_nuevoPrecio <= precioContrato, "El precio debe ser igual o menor que el precio contrato");

    precio = _nuevoPrecio;
    estado = Estado.Confirmado;

    emit CotizacionConfirmada(msg.sender, precio);
}
```

Ilustración 21: Ejemplo de funciones exclusivas de un rol

6_5_Estudio de costes

6_5_1_Variación de tamaño de entrada

Ilustración 22: Argumento introducido corto

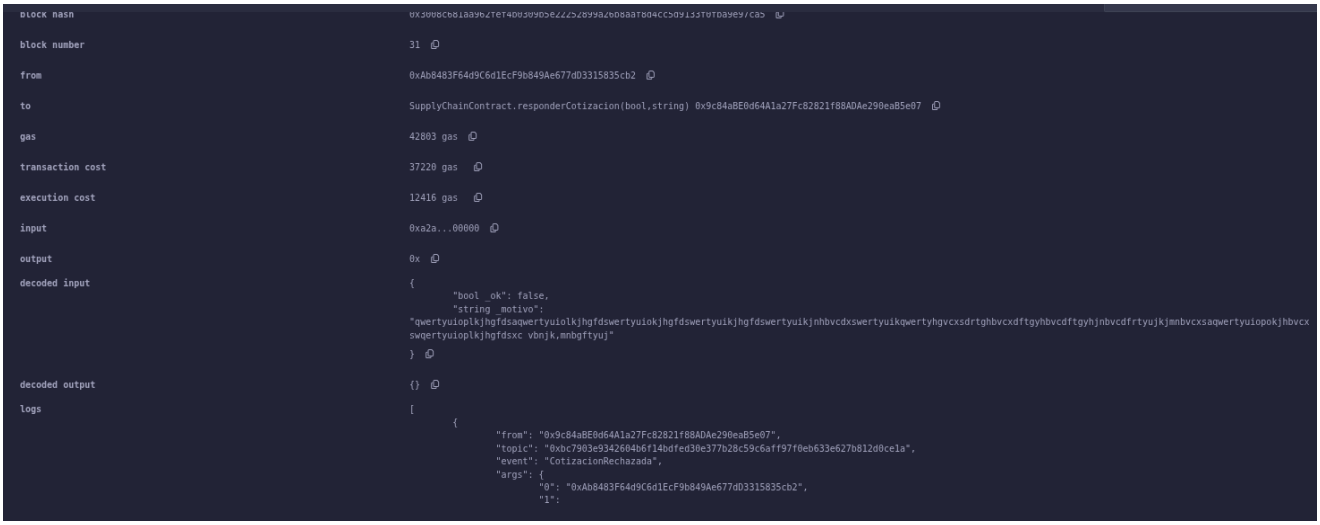


Ilustración 23: Argumento introducido largo

Como podemos observar, el añadir una motivación de rechazo muy extensa, eleva rápidamente el costo del gas (5000 de diferencia de gas en este caso muy simple).

Dado que las variables tipo String, no tienen un límite de tamaño, podríamos forzar el agotamiento del gas de una cuenta.
Esto en entornos de prueba, no supone un problema; pero, para redes de Ether públicas, donde no podemos saber cómo estará implementado un contrato al que vayamos a llamar, causa motivos para desconfiar en el uso de contratos inteligentes de los que no se tiene autoría o conocimiento directo.

Lo ideal, para enfocar la negociación de cadenas de suministros con el uso de contratos inteligentes, sería crear una blockchain privada, donde se gestionen las cuentas sin usar los contratos inteligentes para recolectar el dinero; además, nos veríamos obligados a restringir los tamaños de los datos que vamos a introducir a las funciones, para ello, la mejor forma sería restringir solo desde un frontend,

para dar la posibilidad a aumentarse en caso necesario, sin tener que recompilar un contrato para alterar un límite.

6_5_2_Resumen de coste

Una vez aclarado esto, se mostrará aquí una tabla con costes asintóticos y costes de gas medio de cada función, suponiendo entradas de tipo String como un tamaño razonable (1 línea ó 47 caracteres) Se denota, que se tiene en cuenta el "transaction cost" para el coste medio de gas.

Nombre función	Coste asintótico	Coste medio aproximado de gas
constructor	$O(1)$	1.550.000
solicitarCotizacion	$O(1)$	70.000
intervenirJunior	$O(1)$	28.000
intervenirSenior	$O(1)$	28.000
intervenirDirectorProyecto	$O(1)$	28.000
emitirContrato	$O(1)$	30.000
responderCotizacion	$O(1)$	48.000
aprobarPrecio	$O(1)$	48.000

Tabla 5: Costes de funciones públicas

Aclaraciones

Los cambios ligeros respecto al diagrama, como por ejemplo los modificadores, se representaron así en el Astah por su claridad; pues Astah, no tiene una forma predeterminada hasta la fecha, de indicar que una función es una "cabecera" o un "pie" de otra función.

Dado que indicarlos parámetros que se le darán a una función en un diagrama de secuencia, es incorrecto; se optó por una representación comprensible para cualquier usuario experimentado con el modelo de análisis y diseño de un software.

Esto reduce en gran medida el coste del bytecode que se almacena en un contrato, pues tener muchas funciones modifiers, sigue siendo un coste elevado, por mucho que cada una solo sea un par de líneas.

En los diagramas de secuencia no se añadieron eventos, pues mi sistema no los recoge, y no quiero dar la impresión, de que un evento se capturará más tarde dentro del propio código (ver líneas futuras).

7_Conclusiones y trabajo futuro

7_1_Conclusiones

El presente Trabajo de Fin de Grado ha tenido como objetivo principal explorar el potencial de los contratos inteligentes como herramienta para **mejorar la eficiencia en los procesos de negociación dentro de cadenas de suministro**, centrándose específicamente en la formalización técnica de los acuerdos mediante tecnologías basadas en blockchain. A lo largo del desarrollo, se ha abordado el análisis, diseño e implementación parcial de un sistema backend que represente una situación realista de negociación empresarial, utilizando como punto de partida un caso de uso industrial relacionado con la adquisición de componentes prototípicos.

Uno de los principales aportes del trabajo ha sido la **materialización de una solución funcional, desarrollada en Solidity y ejecutada en Remix**, que modela un flujo básico de negociación entre fabricante y proveedor, basado en parámetros definidos, verificables y almacenados de forma inmutable. Este enfoque contribuye directamente a mitigar dos de los problemas más frecuentes en los procesos tradicionales: **la lentitud de las negociaciones** y el riesgo de **repudio de los acuerdos alcanzados** debido a la falta de registros formales y verificables.

Durante el desarrollo del TFG se han puesto en práctica diversas metodologías de planificación individual, combinando enfoques como Task-Driven Planning y elementos inspirados en Agile, lo cual ha permitido gestionar el trabajo de manera eficiente, flexible y adaptada a las condiciones reales de un proyecto unipersonal. Asimismo, el diseño del sistema ha sido documentado mediante diagramas UML, representando los distintos módulos y flujos del sistema, lo que aporta una base sólida para futuras ampliaciones o implementaciones.

A pesar de que el proyecto no ha sido desplegado en un entorno real ni ha contado con una interfaz de usuario (*frontend*), su implementación backend permite validar la lógica esencial del contrato inteligente, demostrando su utilidad teórica y práctica como herramienta para formalizar interacciones en entornos colaborativos. En este sentido, uno de los aprendizajes más relevantes ha sido la constatación de que **la efectividad de los contratos inteligentes depende fuertemente de la calidad del frontend** que los acompaña, ya que es este el que limita y estructura los datos de entrada, previniendo errores humanos y garantizando la integridad del proceso.

Otra conclusión importante es que, aunque las tecnologías blockchain ofrecen soluciones innovadoras para la automatización y verificación de acuerdos, **su adopción práctica en entornos industriales requiere una integración cuidadosa con los sistemas ya existentes**, así como un esfuerzo adicional en términos de usabilidad, gobernanza y formación de los usuarios. La tecnología, por sí sola, no resuelve todos los problemas organizativos, pero sí puede ser un componente clave dentro de un ecosistema digital más amplio orientado a la eficiencia y la trazabilidad.

En definitiva, este Trabajo de Fin de Grado ha cumplido su objetivo de demostrar la viabilidad de utilizar contratos inteligentes para mejorar procesos críticos en cadenas de suministro. A pesar de sus

limitaciones (como la ausencia de pruebas en entorno real o de integración con una interfaz de usuario), el proyecto ofrece una base sólida sobre la cual construir futuras evoluciones más completas, y permite extraer aprendizajes relevantes tanto desde el punto de vista técnico como organizativo. Este tipo de soluciones apunta a desempeñar un papel cada vez más relevante en el contexto de la **industria 4.0**, donde la automatización, la transparencia y la descentralización serán factores clave para mantener la competitividad.

7_2_Líneas futuras

A día de hoy, el código, además de gestionar la cadena de suministro, tiene planteado un sistema de eventos en las funciones.
Como tal, no se capturan los eventos; simplemente se lanzan como si de una API para el futuro se tratara.
Estos eventos, se plantearon, debido a una especificación del escenario del caso de uso; esta es, la necesidad de enviar alertas.
Solidity no puede enviar alertas a un usuario; pero se podría simular, con el uso de eventos y un frontend que capture estos.

Además, el contrato presenta un atributo llamado fechaLimite, que es establecido para indicar el periodo máximo de espera que puede haber hasta que se envíe una alerta a un cargo organizativo superior de la cadena de suministro.

Lo que se buscaría en un futuro, sería desarrollar un Frontend que permita a los usuarios gestionar sus cuentas, con ayuda de la extensión de navegador “MetaMask”, además de realizar una interfaz más agradable para el usuario.

Se debería de crear una red blockchain Ethereum, en un entorno descentralizado y privado, aportando acceso a los distintos roles, que se enlazaría con el Frontend.

Referencias

(s.f.). Recuperado el 5 de 7 de 2025, de International Journal with DOI Publication fee under 500 IJRPR| High impact factor, Fast Publication Journal, Index in Major Database : https://ijrpr.com/uploads/V5ISSUE1/IJRPR21576.pdf

Anglen, J. (19 de 9 de 2024). *Blockchain Consensus Mechanisms: Complete Guide | PoW to Emerging Models*. Recuperado el 5 de 7 de 2025, de Rapid Innovation | Next-Gen AI for Future-Ready Enterprises: https://www.rapidinnovation.io/post/consensus-mechanisms-in-blockchain-proof-of-work-vs-proof-of-stake-and-beyond

Blockchain protocols and their energy footprint - Adan. (s.f.). Recuperado el 05 de 07 de 2025, de Adan: https://www.adan.eu/en/publication/blockchain-protocols-and-their-energy-footprint-2/

Crypto Goes Green? Ethereum's Merge Reduces Energy Use By a Stunning 99%. (s.f.). Recuperado el 05 de 07 de 2025, de CryptoVantage: https://www.cryptovantage.com/news/crypto-goes-green-ethereum-s-merge-will-reduce-energy-use-by-99-percent/

Google. (s.f.). Recuperado el 5 de 7 de 2025, de Google: https://www.google.com/?hl=es

Grafiati: Generador autómatico de citas online. (s.f.). Recuperado el 29 de 06 de 2025, de Grafiati: Оформити списки використаних джерел онлайн: https://www.grafiati.com/es/

Grincalaitis, M. (16 de 11 de 2023). *The ultimate guide to learn solidity from scratch by building a lending dapp step-by-step in 2024*. Recuperado el 29 de 06 de 2025, de Medium: https://merunasgrincalaitis.medium.com/the-ultimate-guide-to-learn-solidity-from-scratch-by-building-a-lending-dapp-step-by-step-in-2024-72d59085313

Kumar, D. (27 de 7 de 2021). *Hardhat configuration*. Recuperado el 29 de 06 de 2025, de Medium: https://medium.com/coinmonks/hardhat-configuration-c96415d4fcba

Morais, J. P. (21 de 3 de 2022). *ABI encode and decode using solidity*. Recuperado el 18 de 06 de 2025, de Medium: https://medium.com/coinmonks/abi-encode-and-decode-using-solidity-2d372a03e110

Morais, J. P. (26 de 3 de 2022). *Creating and deploying smart contracts using Truffle and Ganache*. Recuperado el 18 de 06 de 2025, de Medium: https://medium.com/coinmonks/creating-and-deploying-smart-contracts-using-truffle-and-ganache-ffe927fa70ae

Morais, J. P. (2 de 4 de 2022). *Handling events of a smart contract part 1/2*. Recuperado el 9 de 06 de 2025, de Medium: https://medium.com/coinmonks/handling-events-of-a-smart-contract-part-1-2-b086eb6696cf

Morais, J. P. (18 de 4 de 2022). *The difference between bytecode and deployed bytecode*. Recuperado el 29 de 06 de 2025, de Medium: https://medium.com/coinmonks/the-difference-between-bytecode-and-deployed-bytecode-64594db723df

Morais, J. P. (18 de 3 de 2022). *Where in storage are state variables stored?* Recuperado el 29 de 06 de 2025, de Medium: https://coinsbench.com/where-in-storage-are-state-variables-stored-a9c38eafa31a

Optimize gas on base and other L2 chains with this solidity tutorial. (s.f.). Recuperado el 29 de 06 de 2025, de Blockchain Security, Smart Contract Audits, Developer Education - Cyfrin: https://www.cyfrin.io/blog/solidity-gas-efficiency-tips-tackle-rising-fees-base-other-l2

Optimize Gas on Base and Other L2 Chains With This Solidity Tutorial. (s.f.). Recuperado el 05 de 07 de 2025, de Blockchain Security, Smart Contract Audits, Developer Education - Cyfrin: https://www.cyfrin.io/blog/solidity-gas-efficiency-tips-tackle-rising-fees-base-other-l2

Satoshi, N. (2008). *A Peer-to-Peer Electronic Cash System.* Recuperado el 5 de 7 de 2025, de A Peer-to-Peer Electronic Cash System.: https://bitcoin.org/bitcoin.pdf

Solidity by example. (s.f.). Recuperado el 29 de 06 de 2025, de Solidity by example: https://solidity-by-example.org/

Szabo, N. (1997). *The idea of smart contracts*. Recuperado el 5 de 7 de 2025, de The idea of smart contracts: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/idea.html

Template Method. (s.f.). Recuperado el 30 de 06 de 2025, de Refactoring and Design Patterns: https://refactoring.guru/design-patterns/template-method

Vitalik, B. (2013). *Ethereum White Paper*. Recuperado el 5 de 7 de 2025, de A Next-Generation Smart Contract: https://ethereum.org/en/whitepaper/

Wikipedia, the free encyclopedia. (s.f.). Recuperado el 5 de 7 de 2025, de Wikipedia: https://www.wikipedia.org/

Anexo

Enlace a Github:
https://github.com/diegocaprod/TFG-IS